Virtex-II
Platform FPGA
Handbook

XILINX®

The Xilinx logo shown above is a registered trademark of Xilinx, Inc.

The shadow X shown above is a trademark of Xilinx, Inc.

"Xilinx" and the Xilinx logo are registered trademarks of Xilinx, Inc. Any rights not expressly granted herein are reserved.

CoolRunner, RocketChips, Rocket IP, Spartan, StateBENCH, StateCAD, Virtex, XACT, XC2064, XC3090, XC4005, XC5210 are registered Trademarks of Xilinx, Inc.

ACE Controller, ACE Flash, A.K.A. Speed, Alliance Series, AllianceCORE, Bencher, ChipScope, Configurable Logic Cell, CORE Generator, CoreLINX, Dual Block, EZTag, Fast CLK, Fast CONNECT, Fast FLASH, FastMap, Fast Zero Power, Foundation, Gigabit Speeds...and Beyond!, HardWire, HDL Bencher, IRL, J Drive, JBits, LCA, LogiBLOX, Logic Cell, LogiCORE, LogicProfessor, MicroBlaze, MicroVia, MultiLINX, NanoBlaze, PicoBlaze, PLUSASM, PowerGuide, PowerMaze, QPro, Real-PCI, Rocket I/O, SelectI/O, SelectRAM, SelectRAM+, Silicon Xpresso, Smartguide, Smart-IP, SmartSearch, SMARTswitch, System ACE, Testbench In A Minute, TrueMap, UIM, VectorMaze, VersaBlock, VersaRing, Virtex-II PRO, Wave Table, WebFITTER, WebPACK, WebPOWERED, XABEL, XACT-Floorplanner, XACT-Performance, XACTstep Advanced, XACTstep Foundry, XAM, XAPP, X-BLOX +, XC designated products, XChecker, XDM, XEPLD, Xilinx Foundation Series, Xilinx XDTV, Xinfo, XSI, XtremeDSP and ZERO+ are trademarks of Xilinx, Inc.

The Programmable Logic Company is a service mark of Xilinx, Inc.

All other trademarks are the property of their respective owners.

Xilinx does not assume any liability arising out of the application or use of any product described or shown herein; nor does it convey any license under its patents, copyrights, or maskwork rights or any rights of others. Xilinx reserves the right to make changes, at any time, in order to improve reliability, function or design and to supply the best product possible. Xilinx will not assume responsibility for the use of any circuitry described herein other than circuitry entirely embodied in its products. Xilinx devices and products are protected under one or more U.S. and International Patents. Xilinx does not represent that its devices or products are free from patent infringement or from any other third party right. Xilinx assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or software support or assistance provided to a user.

Xilinx products are not intended for use in life support appliances, devices, or systems. Use of a Xilinx product in such applications without the written consent of the appropriate Xilinx officer is prohibited.

Copyright 2001 Xilinx, Inc. All Rights Reserved.

Virtex-II Platform FPGA Handbook

UG002 (v1.3) 3 December 2001

Revision History

The following table summarizes changes made to each version of this document.

Date	Version	Revision
12/06/00	1.0	Initial Release.
04/02/01	1.1	This update includes several cosmetic fixes and the following technical edits: • Changed MEM(E) to MEM(F) in Figure 1-4. • Changed PER_DRIFT to CYC_JITT on p.63. • Changed BUFGMUX and BUFGMUX_1 descriptions on p.79. • Changed PER_DRIFT to CYC_JITT on p.89. • Removed "string" from STARTUP_WAIT in VHDL template on p.104. • Changed XC2V_RAM*x*X1S... to RAM*x*X1S on pages 130 - 131. • Changed XC2V_RAM16XN_S_SUBM to XC2V_RAM16XN_D_SUBM in Table 2-19. • Removed SRLC256E_SUBM from Table 2-21. • Changed address inputs in Figure 2-57. • Changed Figure 2-68. • Changed IOBDELAY description on pages 175 and 181. • Changed DV2 values in Table 2-37. • Changed IOSTANDARD = LVDCI_25 on p.207. • Changed CLR0 to CLK0 in Figure 2-100. • Added DATA in Figure 2-101. • Changed VHDL and Verilog code examples on pages 220 - 225. • Changed bitstream lengths in Tables 3-2, 3-7, and 3-16. • Changed Figure 3-7. • Changed Step 3 on p.283. • Changed GTS_CFG to GTS_CFG_B on pages 291 - 292. • Added Packet Header & Packet Data to Table 3-24. • Added ALT_VRP & ALT_VRN definitions to Table 4-1. • Changed Table 4-2 to include ALT_VRP & ALT_VRN pin information in Banks 4 & 5. • Added ALT_VRP & ALT_VRN pins to Figures 4-1 through 4-30. • Changed Solder Land Diameter and Opening in Solder Mask Diameter values in Table 4-5. • Changed Solder Land Diameter and Inner Layer Signal Trace Width in Figures 4-56 through 4-67. • Removed XC2V3000_FF1517.BSD from the Virtex-II BSDL File Names list on p.406.
10/12/01	1.2	This update includes the following technical edits: • Replaced v1.0 of the *Virtex-II Data Sheet* with latest version of modules 1, 2, and 3. • Enhanced several descriptions and figures and updated Chapters 1 and 2. • Updated and added information on System ACE products in Chapter 3. • Clarified footnote in FG456 - FG676 Pinout Compatibility Diagram in Chapter 4. • Removed S08-V08 PROM Package Specification from Appendix C.
12/03/01	1.3	Updated Part I with the latest version of the *Virtex-II Data Sheet* (modules 1, 2, and 3).

Contents

About This Handbook

Introduction to the Virtex-II FPGA Family

Part 1: Virtex-II Data Sheet

Part 2: Virtex-II User Guide

Chapter 1: Timing Models

Chapter 2: Design Considerations

Chapter 3: Configuration

Chapter 4: PCB Design Considerations

Appendix A: Application Notes

Appendix B: BitGen and PROMGen Switches and Options

Appendix C: XC18V00 Series PROMs

Appendix D: Glossary

Figures

Part 1: Virtex-II Data Sheet

Part 2: Virtex-II User Guide

Chapter 1: Timing Models

Chapter 2: Design Considerations

Chapter 3: Configuration

Chapter 4: PCB Design Considerations

Appendix A: Application Notes

Appendix B: BitGen and PROMGen Switches and Options

Appendix C: XC18V00 Series PROMs

Appendix D: Glossary

Tables

Part 1: Virtex-II Data Sheet

Part 2: Virtex-II User Guide

Chapter 1: Timing Models

Chapter 2: Design Considerations

Chapter 3: Configuration

Chapter 4: PCB Design Considerations

Appendix A: Application Notes

Appendix B: BitGen and PROMGen Switches and Options

Appendix C: XC18V00 Series PROMs

Appendix D: Glossary

About This Handbook

This document describes the function and operation of Virtex-II devices and also includes information on FPGA configuration techniques and PCB design considerations. For Virtex-II device specifications, refer to the Virtex-II **Data Sheet** in Part I of this handbook.

For details on the following topics, see the Virtex-II *Platform FPGA User Guide* in Part II of this handbook:

- Chapter 1: Timing Models
- Chapter 2: Design Considerations
- Chapter 3: Configuration
- Chapter 4: PCB Design Considerations
- Appendix A: Application Notes
- Appendix B: BitGen and PROMGen Switches and Options
- Appendix C: XC18V00 Series PROMs
- Appendix D: Glossary

Additional Resources

The following table lists URLs for resources available on the web. For additional information, go to **http://www.xilinx.com**.

Resource	Description/URL
Handbook	This site contains the latest *Virtex-II User Guide* and *Virtex-II Data Sheet*: **http://www.xilinx.com/products/virtex/handbook/**
Application Notes	This site contains device-specific design techniques and approaches: **http://www.xilinx.com/apps/appsweb.htm**
Data Book	*The Programmable Logic Data Book* describes device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging: **http://www.xilinx.com/partinfo/databook.htm**
Xcell Journals	This site contains quarterly journals for Xilinx programmable logic users: **http://www.xilinx.com/xcell/xcell.htm**
Tech Tips	See this site for the latest news, design tips, and patch information on the Xilinx design environment: **http://www.xilinx.com/support/techsup/journals/index.htm**
Answers Database	This database provides a current listing of solution records for Xilinx software tools. Search this database using the search function at: **http://www.xilinx.com/support/searchtd.htm**

Typographical Conventions

The following typographical conventions are used in this manual:

- Red text indicates a cross-reference to information within this document. Click red text to open the specified cross-reference.
- Blue-underlined text indicates a link to a Web page. Click blue-underlined text to browse the specified Web site.
- `Courier font` indicates prompts or program outputs displayed by the system.

  `speed grade: 5`
- **`Courier bold`** indicates literal commands that you enter in a syntactical statement. However, braces "{ }" in Courier bold are not literal and square brackets "[ ]" in Courier bold are literal only in the case of bus specifications, such as bus [7:0].

  `rpt_del_net=`

  Courier bold also indicates menu commands: **`File`** → **`Open`**
- *Italic font* denotes the following items:
  - Variables that are substituted with user-defined values

    `edif2ngd` *`design_name`*
  - References to other documents.

    See the *Libraries Guide* for more information.
  - Emphasis in text

    If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
- Square brackets "[ ]" indicate an optional entry or parameter. However, in bus specifications, such as bus [7:0], they are required.

  `edif2ngd` [*option_name*] *design_name*
- Braces "{ }" enclose a list of items from which you must choose one or more, and a vertical bar " | " separates items in a list of choices:

  `lowpwr` **`=`**{**`on`** | **`off`**}
- A vertical ellipsis indicates repetitive material that has been omitted.

```
IOB #1: Name = QOUT'
IOB #2: Name = CLKIN'
.
.
.
```

- A horizontal ellipsis ". . ." indicates that an item can be repeated one or more times.

  `allow block` *block_name loc1 loc2* `. . .` *locn;*

Introduction to the Virtex-II FPGA Family

Virtex-II Platform

The Virtex-II Platform FPGA solution is the result of the largest silicon and software R&D effort in the history of programmable logic, with the goal of revolutionizing the design of complex single-chip sub-systems in terms of engineering productivity, silicon efficiency, and system flexibility.

The Virtex-II product family provides IP-Immersion™ technology which incorporates an abundance of on-chip memory options and advanced routing resources for supporting complex designs that use IP (intellectual property), such as on-chip hard-macro building blocks and a rapidly growing library of soft-IP blocks. For the first time in the programmable logic industry, innovative Virtex-II features enable system designers to:

- Eliminate external termination resistors with on-chip precision-controlled output impedance
- Manage 16 pre-engineered low-skew clock domains, with on-chip frequency and phase control
- Protect chip designs with bit-stream encryption

These unique capabilities increase engineering productivity and time-to-production by supply pre-engineered solutions for signal integrity and RF noise challenges, as well as providing a secure means to deliver designs rapidly to production.

The Virtex-II Platform FPGA family is a complete programmable solution that allows digital system designers to rapidly implement a single-chip solution with density up to 10 million system gates, in weeks rather than months or years. The inherent flexibility of Xilinx FPGA devices allows unlimited design changes throughout the development and production phases of the system, with important benefits in improved productivity, reduced design risk, and higher system flexibility. This further accelerates the industry -- from custom ASICs to FPGAs -- in fields such as optical networking systems, gigabit routers, wireless cellular base stations, modem arrays, and professional video broadcast systems.

Virtex-II Target Applications

The Virtex-II solution is developed specifically to enable rapid development of two of the most technically challenging digital system applications: data communications and digital signal processing (DSP) systems. High logic integration, fast and complex routing of wide busses, and extensive pipeline and FIFO memory requirements characterize these systems.

The Virtex-II family incorporates high logic capacity, up to 10 million system gates, a new Active Interconnect™ architecture optimized for predictable routing delays, an advanced memory array architecture with up to 4.5Mbits of on-chip memory, and built-in support for high-speed I/O standards at up to 1108 user pins.

Applications incorporating DSP functionality, such as echo cancellation, forward error-correction, and image compression/decompression, benefit from the abundance of embedded high-speed 18-bit x 18-bit multiplier blocks within the Virtex-II solution.

The unique features of the revolutionary Virtex-II architecture make it ideal for optical networking products, storage area networks (SANs), Voice-over-Internet-Protocol (VoIP), video broadcasting, medical imaging, wireless base-stations, and Internet infrastructure products, as well as many other products.

Interconnect Engine for Fast, Wide Busses in Networking Applications

The Virtex-II architecture incorporates a number of novel features specifically to support wide data widths in complex networking and transmission systems. Modern complex systems operate with multiple clock domains, with large IP-based subsystems operating independently. Large, wide FIFOs and buffer memories are needed for handling fast and wide inter-subsystem data transfer. These wide busses are required both internally for intra-chip communications and externally for switched fabric communications.

For example, wide 32-bit and larger data busses can drive multiple Ultra Low-Voltage Differential Signal (ULVDS) high-speed interface standards for data transfer across a backplane or for point-to-point communications, or be used for implementing high-speed multi-cast bus standards.

These requirements challenge and exceed the capabilities of current programmable logic devices, which lack the gate capacity, memory and routing resources, performance, and architecture flexibility to fully support these designs. The Virtex-II solution is the first platform FPGA specifically targeted to improve the “ease of speed” in the development and production of these complex systems.

Complete Solution For Rapid Time-to-Production

The Virtex-II solution combines the most flexible FPGA architecture, advanced process technology, powerful software synthesis technology, and robust IP library, to provide the most complete system integration solution today. In addition, the Virtex-II solution provides powerful features, such as Xilinx Digitally Controlled Impedance (DCI) technology, digital clock manager to help designers further reduce overall system cost and design development cycle, making Virtex-II the ideal solution for tomorrow’s high-performance system designs.

Part I: Virtex-II Data Sheet

This section contains the Virtex-II advance product specification (DS031). The latest version of this information is available online at **www.xilinx.com/apps/virtexapp.htm**.

Virtex-II 1.5V
Field-Programmable Gate Arrays

DS031-1 (v1.7) October 2, 2001

Advance Product Specification

Summary of Virtex®-II Features

- Industry First Platform FPGA Solution
- IP-Immersion™ Architecture
  - Densities from 40K to 8M system gates
  - 420 MHz internal clock speed (Advance Data)
  - 840+ Mb/s I/O (Advance Data)
- SelectRAM™ Memory Hierarchy
  - 3 Mb of True Dual-Port™ RAM in 18-Kbit block SelectRAM resources
  - Up to 1.5 Mb of distributed SelectRAM resources
  - High-performance interfaces to external memory
    - DDR-SDRAM interface
    - FCRAM interface
    - QDR™-SRAM interface
    - Sigma RAM interface
- Arithmetic Functions
  - Dedicated 18-bit x 18-bit multiplier blocks
  - Fast look-ahead carry logic chains
- Flexible Logic Resources
  - Up to 93,184 internal registers / latches with Clock Enable
  - Up to 93,184 look-up tables (LUTs) or cascadable 16-bit shift registers
  - Wide multiplexers and wide-input function support
  - Horizontal cascade chain and Sum-of-Products support
  - Internal 3-state bussing
- High-Performance Clock Management Circuitry
  - Up to 12 DCM (Digital Clock Manager) modules
    - Precise clock de-skew
    - Flexible frequency synthesis
    - High-resolution phase shifting
  - 16 global clock multiplexer buffers
- Active Interconnect™ Technology
  - Fourth generation segmented routing structure
  - Predictable, fast routing delay, independent of fanout
- SelectI/O-Ultra™ Technology
  - Up to 1,108 user I/Os
  - 19 single-ended standards and six differential standards
  - Programmable sink current (2 mA to 24 mA) per I/O
  - Digitally Controlled Impedance (DCI) I/O: on-chip termination resistors for single-ended I/O standards
  - PCI-X @ 133 MHz, PCI @ 66 MHz and 33 MHz compliance, and CardBus compliant
  - Differential Signaling
    - 840 Mb/s Low-Voltage Differential Signaling I/O (LVDS) with current mode drivers
    - Bus LVDS I/O
    - Lightning Data Transport (LDT) I/O with current driver buffers
    - Low-Voltage Positive Emitter-Coupled Logic (LVPECL) I/O
    - Built-in DDR Input and Output registers
  - Proprietary high-performance SelectLink™ Technology
    - High-bandwidth data path
    - Double Data Rate (DDR) link
    - Web-based HDL generation methodology
- Supported by Xilinx Foundation™ and Alliance™ Series Development Systems
  - Integrated VHDL and Verilog design flows
  - Compilation of 10M system gates designs
  - Internet Team Design (ITD) tool
- SRAM-Based In-System Configuration
  - Fast SelectMAP™ configuration
  - Triple Data Encryption Standard (DES) security option (Bitstream Encryption)
  - IEEE1532 support
  - Partial reconfiguration
  - Unlimited re-programmability
  - Readback capability
- 0.15 µm 8-Layer Metal process with 0.12 µm high-speed transistors
- 1.5 V (V_{CCINT}) core power supply, dedicated 3.3 V V_{CCAUX} auxiliary and V_{CCO} I/O power supplies
- IEEE 1149.1 compatible boundary-scan logic support
- Flip-Chip and Wire-Bond Ball Grid Array (BGA) packages in three standard fine pitches (0.80mm, 1.00mm, and 1.27mm)
- 100% factory tested

© 2001 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at http://www.xilinx.com/legal.htm.
All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

Table 1: **Virtex-II Field-Programmable Gate Array Family Members**

Device	System Gates	CLB (1 CLB = 4 slices = Max 128 bits)			Multiplier Blocks	SelectRAM Blocks		DCMs	Max I/O Pads(1)
		Array Row x Col.	Slices	Maximum Distributed RAM Kbits		18-Kbit Blocks	Max RAM (Kbits)		
XC2V40	40K	8 x 8	256	8	4	4	72	4	88
XC2V80	80K	16 x 8	512	16	8	8	144	4	120
XC2V250	250K	24 x 16	1,536	48	24	24	432	8	200
XC2V500	500K	32 x 24	3,072	96	32	32	576	8	264
XC2V1000	1M	40 x 32	5,120	160	40	40	720	8	432
XC2V1500	1.5M	48 x 40	7,680	240	48	48	864	8	528
XC2V2000	2M	56 x 48	10,752	336	56	56	1,008	8	624
XC2V3000	3M	64 x 56	14,336	448	96	96	1,728	12	720
XC2V4000	4M	80 x 72	23,040	720	120	120	2,160	12	912
XC2V6000	6M	96 x 88	33,792	1,056	144	144	2,592	12	1,104
XC2V8000	8M	112 x 104	46,592	1,456	168	168	3,024	12	1,108

Notes:
1. See details in Table 2, “Maximum Number of User I/O Pads”.

General Description

The Virtex-II family is a platform FPGA developed for high performance from low-density to high-density designs that are based on IP cores and customized modules. The family delivers complete solutions for telecommunication, wireless, networking, video, and DSP applications, including PCI, LVDS, and DDR interfaces.

The leading-edge 0.15µm / 0.12µm CMOS 8-layer metal process and the Virtex-II architecture are optimized for high speed with low power consumption. Combining a wide variety of flexible features and a large range of densities up to 10 million system gates, the Virtex-II family enhances programmable logic design capabilities and is a powerful alternative to mask-programmed gates arrays. As shown in Table 1, the Virtex-II family comprises 12 members, ranging from 40K to 10M system gates.

Packaging

Offerings include ball grid array (BGA) packages with 0.80mm, 1.00mm, and 1.27mm pitches. In addition to traditional wire-bond interconnects, flip-chip interconnect is used in some of the BGA offerings. The use of flip-chip interconnect offers more I/Os than is possible in wire-bond versions of the similar packages. Flip-Chip construction offers the combination of high pin count with high thermal capacity.

Table 2 shows the maximum number of user I/Os available. The Virtex-II device/package combination table (Table 6 at the end of this section) details the maximum number of I/Os for each device and package using wire-bond or flip-chip technology.

Table 2: **Maximum Number of User I/O Pads**

Device	Wire-Bond	Flip-Chip
XC2V40	88	
XC2V80	120	
XC2V250	200	
XC2V500	264	
XC2V1000	328	432
XC2V1500	392	528
XC2V2000	456	624
XC2V3000	516	720
XC2V4000		912
XC2V6000		1,104
XC2V8000		1,108

Architecture

Virtex-II Array Overview

Virtex-II devices are user-programmable gate arrays with various configurable elements. The Virtex-II architecture is optimized for high-density and high-performance logic designs. As shown in Figure 1, the programmable device is comprised of input/output blocks (IOBs) and internal configurable logic blocks (CLBs).

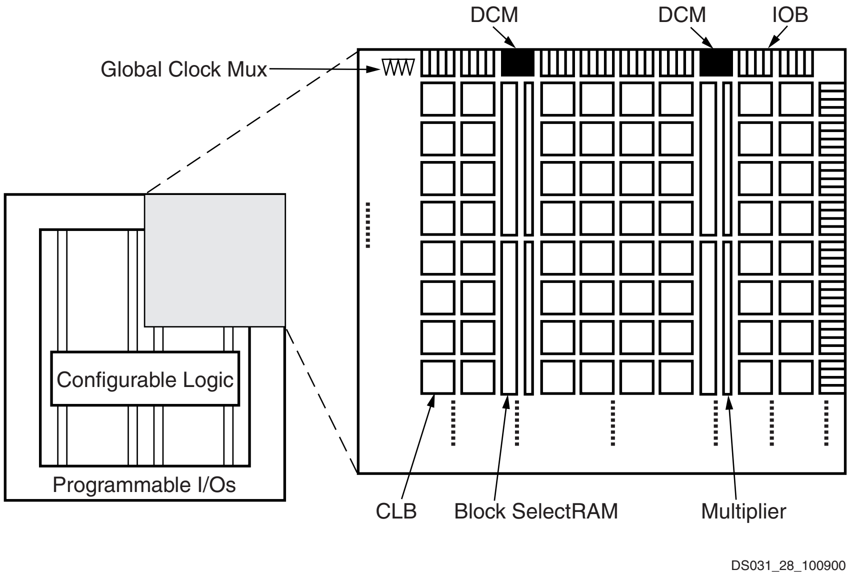


Figure 1: **Virtex-II Architecture Overview**

Programmable I/O blocks provide the interface between package pins and the internal configurable logic. Most popular and leading-edge I/O standards are supported by the programmable IOBs.

The internal configurable logic includes four major elements organized in a regular array.

- Configurable Logic Blocks (CLBs) provide functional elements for combinatorial and synchronous logic, including basic storage elements. BUFTs (3-state buffers) associated with each CLB element drive dedicated segmentable horizontal routing resources.
- Block SelectRAM memory modules provide large 18-Kbit storage elements of True Dual-Port RAM.
- Multiplier blocks are 18-bit x 18-bit dedicated multipliers.
- DCM (Digital Clock Manager) blocks provide self-calibrating, fully digital solutions for clock distribution delay compensation, clock multiplication and division, coarse and fine-grained clock phase shifting.

A new generation of programmable routing resources called Active Interconnect Technology interconnects all of these elements. The general routing matrix (GRM) is an array of routing switches. Each programmable element is tied to a switch matrix, allowing multiple connections to the general routing matrix. The overall programmable interconnection is hierarchical and designed to support high-speed designs.

All programmable elements, including the routing resources, are controlled by values stored in static memory cells. These values are loaded in the memory cells during configuration and can be reloaded to change the functions of the programmable elements.

Virtex-II Features

This section briefly describes Virtex-II features.

Input/Output Blocks (IOBs)

IOBs are programmable and can be categorized as follows:

- Input block with an optional single-data-rate or double-data-rate (DDR) register
- Output block with an optional single-data-rate or DDR register, and an optional 3-state buffer, to be driven directly or through a single or DDR register
- Bi-directional block (any combination of input and output configurations)

These registers are either edge-triggered D-type flip-flops or level-sensitive latches.

IOBs support the following single-ended I/O standards:

- LVTTL, LVCMOS (3.3 V, 2.5 V, 1.8 V, and 1.5 V)
- PCI-X at 133 MHz, PCI (3.3 V at 33 MHz and 66 MHz)
- GTL and GTLP
- HSTL (Class I, II, III, and IV)

- SSTL (3.3 V and 2.5 V, Class I and II)
- AGP-2X

The digitally controlled impedance (DCI) I/O feature automatically provides on-chip termination for each I/O element.

The IOB elements also support the following differential signaling I/O standards:

- LVDS
- BLVDS (Bus LVDS)
- ULVDS
- LDT
- LVPECL

Two adjacent pads are used for each differential pair. Two or four IOB blocks connect to one switch matrix to access the routing resources.

Configurable Logic Blocks (CLBs)

CLB resources include four slices and two 3-state buffers. Each slice is equivalent and contains:

- Two function generators (F & G)
- Two storage elements
- Arithmetic logic gates
- Large multiplexers
- Wide function capability
- Fast carry look-ahead chain
- Horizontal cascade chain (OR gate)

The function generators F & G are configurable as 4-input look-up tables (LUTs), as 16-bit shift registers, or as 16-bit distributed SelectRAM memory.

In addition, the two storage elements are either edge-triggered D-type flip-flops or level-sensitive latches.

Each CLB has internal fast interconnect and connects to a switch matrix to access general routing resources.

Block SelectRAM Memory

The block SelectRAM memory resources are 18 Kb of True Dual-Port RAM, programmable from 16K x 1 bit to 512 x 36 bits, in various depth and width configurations. Each port is totally synchronous and independent, offering three "read-during-write" modes. Block SelectRAM memory is cascadable to implement large embedded storage blocks. Supported memory configurations for dual-port and single-port modes are shown in Table 3.

Table 3: **Dual-Port And Single-Port Configurations**

16K x 1 bit	2K x 9 bits
8K x 2 bits	1K x 18 bits
4K x 4 bits	512 x 36 bits

A multiplier block is associated with each SelectRAM memory block. The multiplier block is a dedicated 18 x 18-bit multiplier and is optimized for operations based on the block SelectRAM content on one port. The 18 x 18 multiplier can be used independently of the block SelectRAM resource. Read/multiply/accumulate operations and DSP filter structures are extremely efficient.

Both the SelectRAM memory and the multiplier resource are connected to four switch matrices to access the general routing resources.

Global Clocking

The DCM and global clock multiplexer buffers provide a complete solution for designing high-speed clocking schemes.

Up to 12 DCM blocks are available. To generate de-skewed internal or external clocks, each DCM can be used to eliminate clock distribution delay. The DCM also provides 90-, 180-, and 270-degree phase-shifted versions of its output clocks. Fine-grained phase shifting offers high-resolution phase adjustments in increments of 1/256 of the clock period. Very flexible frequency synthesis provides a clock output frequency equal to any M/D ratio of the input clock frequency, where M and D are two integers. For the exact timing parameters, see **Virtex™-II Electrical Characteristics**.

Virtex-II devices have 16 global clock MUX buffers, with up to eight clock nets per quadrant. Each global clock MUX buffer can select one of the two clock inputs and switch glitch-free from one clock to the other. Each DCM block is able to drive up to four of the 16 global clock MUX buffers.

Routing Resources

The IOB, CLB, block SelectRAM, multiplier, and DCM elements all use the same interconnect scheme and the same access to the global routing matrix. Timing models are shared, greatly improving the predictability of the performance of high-speed designs.

There are a total of 16 global clock lines, with eight available per quadrant. In addition, 24 vertical and horizontal long lines per row or column as well as massive secondary and local routing resources provide fast interconnect. Virtex-II buffered interconnects are relatively unaffected by net fanout and the interconnect layout is designed to minimize crosstalk.

Horizontal and vertical routing resources for each row or column include:

- 24 long lines
- 120 hex lines
- 40 double lines
- 16 direct connect lines (total in all four directions)

Boundary Scan

Boundary scan instructions and associated data registers support a standard methodology for accessing and configuring Virtex-II devices that complies with IEEE standards 1149.1 - 1993 and 1532. A system mode and a test mode are implemented. In system mode, a Virtex-II device performs its intended mission even while executing non-test boundary-scan instructions. In test mode, boundary-scan test instructions control the I/O pins for testing purposes. The Virtex-II Test Access Port (TAP) supports BYPASS, PRELOAD, SAMPLE, IDCODE, and USERCODE non-test instructions. The EXTEST, INTEST, and HIGHZ test instructions are also supported.

Configuration

Virtex-II devices are configured by loading data into internal configuration memory, using the following five modes:

- Slave-serial mode
- Master-serial mode
- Slave SelectMAP mode
- Master SelectMAP mode
- Boundary-Scan mode (IEEE 1532)

A Data Encryption Standard (DES) decryptor is available on-chip to secure the bitstreams. One or two triple-DES key sets can be used to optionally encrypt the configuration information.

Readback and Integrated Logic Analyzer

Configuration data stored in Virtex-II configuration memory can be read back for verification. Along with the configuration data, the contents of all flip-flops/latches, distributed SelectRAM, and block SelectRAM memory resources can be read back. This capability is useful for real-time debugging.

The Integrated Logic Analyzer (ILA) core and software provides a complete solution for accessing and verifying Virtex-II devices.

Virtex-II Device/Package Combinations and Maximum I/O

Wire-bond and flip-chip packages are available. Table 4 and Table 5 show the maximum possible number of user I/Os in wire-bond and flip-chip packages, respectively. Table 6 shows the number of available user I/Os for all device/package combinations.

- CS denotes wire-bond chip-scale ball grid array (BGA) (0.80 mm pitch).
- FG denotes wire-bond fine-pitch BGA (1.00 mm pitch).
- FF denotes flip-chip fine-pitch BGA (1.00 mm pitch).
- BG denotes standard BGA (1.27 mm pitch).
- BF denotes flip-chip BGA (1.27 mm pitch).

The number of I/Os per package include all user I/Os except the 15 control pins (CCLK, DONE, M0, M1, M2, PROG_B, PWRDWN_B, TCK, TDI, TDO, TMS, HSWAP_EN, DXN, DXP, AND RSVD) and VBATT.

Table 4: **Wire-Bond Packages Information**

Package	CS144	FG256	FG456	FG676	BG575	BG728
Pitch (mm)	0.80	1.00	1.00	1.00	1.27	1.27
Size (mm)	12 x 12	17 x 17	23 x 23	27 x 27	31 x 31	35 x 35
I/Os	92	172	324	484	408	516

Table 5: **Flip-Chip Packages Information**

Package	FF896	FF1152	FF1517	BF957
Pitch (mm)	1.00	1.00	1.00	1.27
Size (mm)	31 x 31	35 x 35	40 x 40	40 x 40
I/Os	624	824	1,108	684

Table 6: **Virtex-II Device/Package Combinations and Maximum Number of Available I/Os (Advance Information)**

Package	Available I/Os										
	XC2V 40	XC2V 80	XC2V 250	XC2V 500	XC2V 1000	XC2V 1500	XC2V 2000	XC2V 3000	XC2V 4000	XC2V 6000	XC2V 8000
CS144	88	92	92								
FG256	88	120	172	172	172						
FG456			200	264	324						
FG676						392	456	484			
FF896					432	528	624				
FF1152								720	824	824	824
FF1517									912	1,104	1,108
BG575					328	392	408				
BG728							456	516			
BF957							624	684	684	684	684

Notes:

1. All devices in a particular package are pin-out (footprint) compatible. In addition, the FG456 and FG676 packages are compatible, as are the FF896 and FF1152 packages.

Virtex-II Ordering Information

Virtex-II ordering information is shown in Figure 2

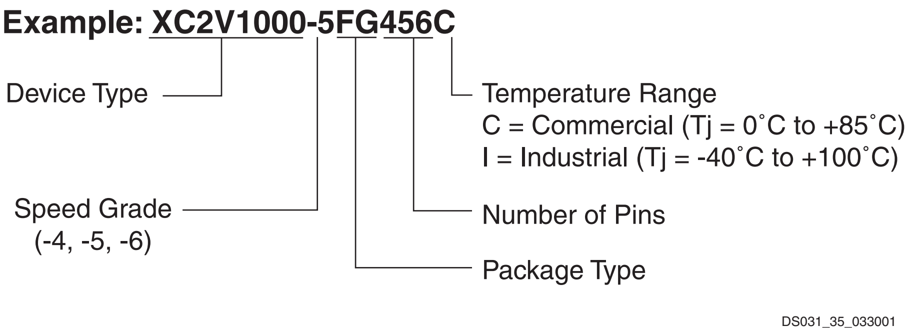


Figure 2: **Virtex-II Ordering Information**

Revision History

This section records the change history for this module of the data sheet.

Date	Version	Revision
11/07/00	1.0	Early access draft.
12/06/00	1.1	Initial release.
01/15/01	1.2	Added values to the tables in the **Virtex-II Performance Characteristics** and **Virtex-II Switching Characteristics** sections.
01/25/01	1.3	The data sheet was divided into four modules (per the current style standard).
04/02/01	1.5	Skipped v1.4 to sync up modules. Reverted to traditional double-column format.
07/30/01	1.6	Made minor changes to items listed under **Summary of Virtex®-II Features**.
10/02/01	1.7	Minor edits.

Virtex-II Data Sheet

The Virtex-II Data Sheet contains the following modules:

- DS031-1, Virtex-II 1.5V FPGAs: Introduction and Ordering Information (Module 1)
- DS031-2, Virtex-II 1.5V FPGAs: **Functional Description (Module 2)**
- DS031-3, Virtex-II 1.5V FPGAs: **DC and Switching Characteristics (Module 3)**
- DS031-4, Virtex-II 1.5V FPGAs: **Pinout Tables (Module 4)**

Virtex-II 1.5V Field-Programmable Gate Arrays

DS031-2 (v1.9) November 29, 2001

Advance Product Specification

Detailed Description

Input/Output Blocks (IOBs)

Virtex-II I/O blocks (IOBs) are provided in groups of two or four on the perimeter of each device. Each IOB can be used as input and/or output for single-ended I/Os. Two IOBs can be used as a differential pair. A differential pair is always connected to the same switch matrix, as shown in Figure 1.

IOB blocks are designed for high performances I/Os, supporting 19 single-ended standards, as well as differential signaling with LVDS, LDT, Bus LVDS, and LVPECL.

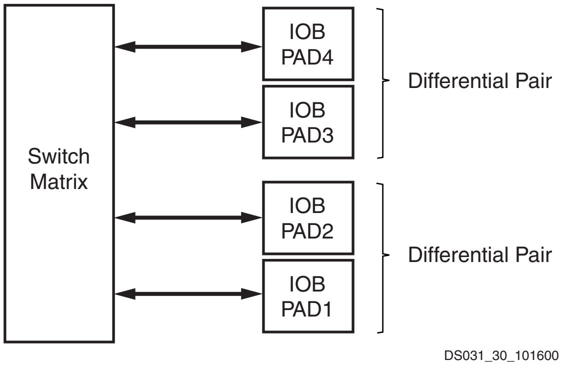


Figure 1: **Virtex-II Input/Output Tile**

Supported I/O Standards

Virtex-II IOB blocks feature SelectI/O inputs and outputs that support a wide variety of I/O signaling standards. In addition to the internal supply voltage (V_{CCINT} = 1.5V), output driver supply voltage (V_{CCO}) is dependent on the I/O standard (see Table 1). An auxiliary supply voltage (V_{CCAUX} = 3.3 V) is required, regardless of the I/O standard used. For exact supply voltage absolute maximum ratings, see **DC Input and Output Levels**.

Table 1: **Supported Single-Ended I/O Standards**

I/O Standard	Output V_{CCO}	Input V_{CCO}	Input V_{REF}	Board Termination Voltage (V_{TT})
LVTTL	3.3	3.3	N/A	N/A
LVCMOS33	3.3	3.3	N/A	N/A
LVCMOS25	2.5	2.5	N/A	N/A
LVCMOS18	1.8	1.8	N/A	N/A
LVCMOS15	1.5	1.5	N/A	N/A
PCI33_3	3.3	3.3	N/A	N/A
PCI66_3	3.3	3.3	N/A	N/A
PCI-X	3.3	3.3	N/A	N/A
GTL	Note 1	Note 1	0.8	1.2
GTLP	Note 1	Note 1	1.0	1.5
HSTL_I	1.5	N/A	0.75	0.75
HSTL_II	1.5	N/A	0.75	0.75
HSTL_III	1.5	N/A	0.9	1.5
HSTL_IV	1.5	N/A	0.9	1.5
SSTL2_I	2.5	N/A	1.25	1.25
SSTL2_II	2.5	N/A	1.25	1.25
SSTL3_I	3.3	N/A	1.5	1.5
SSTL3_II	3.3	N/A	1.5	1.5
AGP-2X/AGP	3.3	N/A	1.32	N/A

Notes:

1. V_{CCO} of GTL or GTLP should not be lower than the termination voltage or the voltage seen at the I/O pad.

© 2001 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at http://www.xilinx.com/legal.htm.
All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

Table 2: **Supported Differential Signal I/O Standards**

I/O Standard	Output V_{CCO}	Input V_{CCO}	Input V_{REF}	Output V_{OD}
LVPECL_33	3.3	N/A	N/A	490 mV to 1.22 V
LDT_25	2.5	N/A	N/A	0.430 - 0.670
LVDS_33	3.3	N/A	N/A	0.250 - 0.400
LVDS_25	2.5	N/A	N/A	0.250 - 0.400
LVDSEXT_33	3.3	N/A	N/A	0.330 - 0.700
LVDSEXT_25	2.5	N/A	N/A	0.330 - 0.700
BLVDS_25	2.5	N/A	N/A	0.250 - 0.450
ULVDS_25	2.5	N/A	N/A	0.430 - 0.670

All of the user IOBs have fixed-clamp diodes to V_{CCO} and to ground. The IOBs are not compatible or compliant with 5 V I/O standards (not 5 V tolerant).

Table 3 lists supported I/O standards with Digitally Controlled Impedance. See **Digitally Controlled Impedance (DCI)**, page 53.

Table 3: **Supported DCI I/O Standards**

I/O Standard	Output V_{CCO}	Input V_{CCO}	Input V_{REF}	Termination Type
LVDCI_33(1)	3.3	3.3	N/A	Series
LVDCI_DV2_33(1)	3.3	3.3	N/A	Series
LVDCI_25(1)	2.5	2.5	N/A	Series
LVDCI_DV2_25(1)	2.5	2.5	N/A	Series
LVDCI_18(1)	1.8	1.8	N/A	Series
LVDCI_DV2_18(1)	1.8	1.8	N/A	Series
LVDCI_15(1)	1.5	1.5	N/A	Series
LVDCI_DV2_15(1)	1.5	1.5	N/A	Series
GTL_DCI	1.2	1.2	0.8	Single
GTLP_DCI	1.5	1.5	1.0	Single
HSTL_I_DCI	1.5	1.5	0.75	Split
HSTL_II_DCI	1.5	1.5	0.75	Split
HSTL_III_DCI	1.5	1.5	0.9	Single
HSTL_IV_DCI	1.5	1.5	0.9	Single
SSTL2_I_DCI(2)	2.5	2.5	1.25	Split
SSTL2_II_DCI(2)	2.5	2.5	1.25	Split
SSTL3_I_DCI(2)	3.3	3.3	1.5	Split
SSTL3_II_DCI(2)	3.3	3.3	1.5	Split

Notes:
1. LVDCI_XX and LVDCI_DV2_XX are LVCMOS controlled impedance buffers, matching the reference resistors or half of the reference resistors.
2. These are SSTL compatible.

Logic Resources

IOB blocks include six storage elements, as shown in Figure 2.

Each storage element can be configured either as an edge-triggered D-type flip-flop or as a level-sensitive latch. On the input, output, and 3-state path, one or two DDR registers can be used.

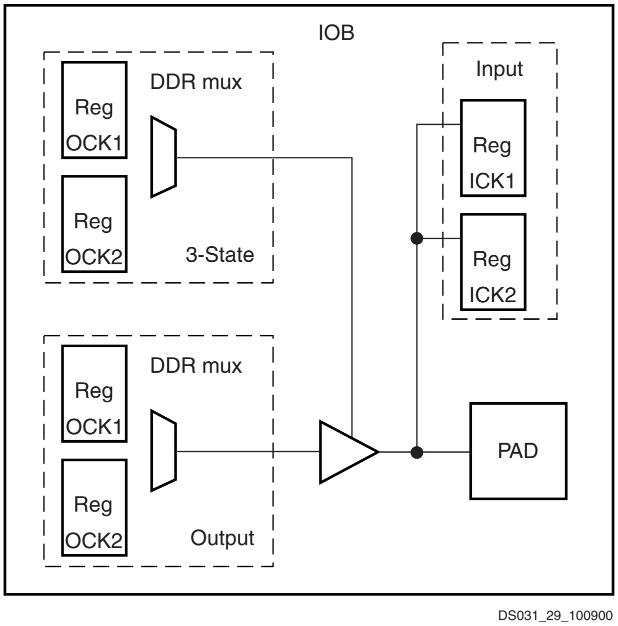


Figure 2: **Virtex-II IOB Block**

Double data rate is directly accomplished by the two registers on each path, clocked by the rising edges (or falling edges) from two different clock nets. The two clock signals are generated by the DCM and must be 180 degrees out of phase, as shown in Figure 3. There are two input, output, and 3-state data signals, each being alternately clocked out.

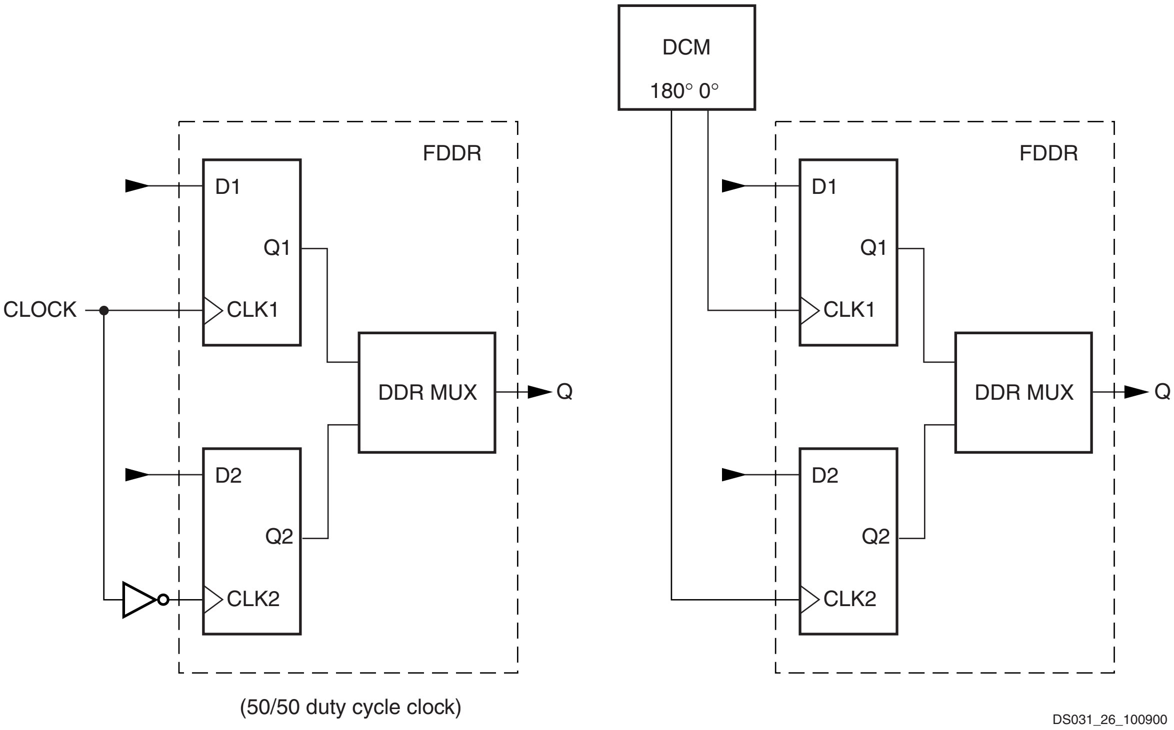


Figure 3: **Double Data Rate Registers**

This DDR mechanism can be used to mirror a copy of the clock on the output. This is useful for propagating a clock along the data that has an identical delay. It is also useful for multiple clock generation, where there is a unique clock driver for every clock load. Virtex-II devices can produce many copies of a clock with very little skew.

Each group of two registers has a clock enable signal (ICE for the input registers, OCE for the output registers, and TCE for the 3-state registers). The clock enable signals are active High by default. If left unconnected, the clock enable for that storage element defaults to the active state.

Each IOB block has common synchronous or asynchronous set and reset (SR and REV signals).

SR forces the storage element into the state specified by the SRHIGH or SRLOW attribute. SRHIGH forces a logic “1”. SRLOW forces a logic “0”. When SR is used, a second input (REV) forces the storage element into the opposite state. The reset condition predominates over the set condition. The initial state after configuration or global initialization state is defined by a separate INIT0 and INIT1 attribute. By default, the SRLOW attribute forces INIT0, and the SRHIGH attribute forces INIT1.

For each storage element, the SRHIGH, SRLOW, INIT0, and INIT1 attributes are independent. Synchronous or asynchronous set / reset is consistent in an IOB block.

All the control signals have independent polarity. Any inverter placed on a control input is automatically absorbed.

Each register or latch (independent of all other registers or latches) (see Figure 4) can be configured as follows:

- No set or reset
- Synchronous set
- Synchronous reset
- Synchronous set and reset
- Asynchronous set (preset)
- Asynchronous reset (clear)
- Asynchronous set and reset (preset and clear)

The synchronous reset overrides a set, and an asynchronous clear overrides a preset.

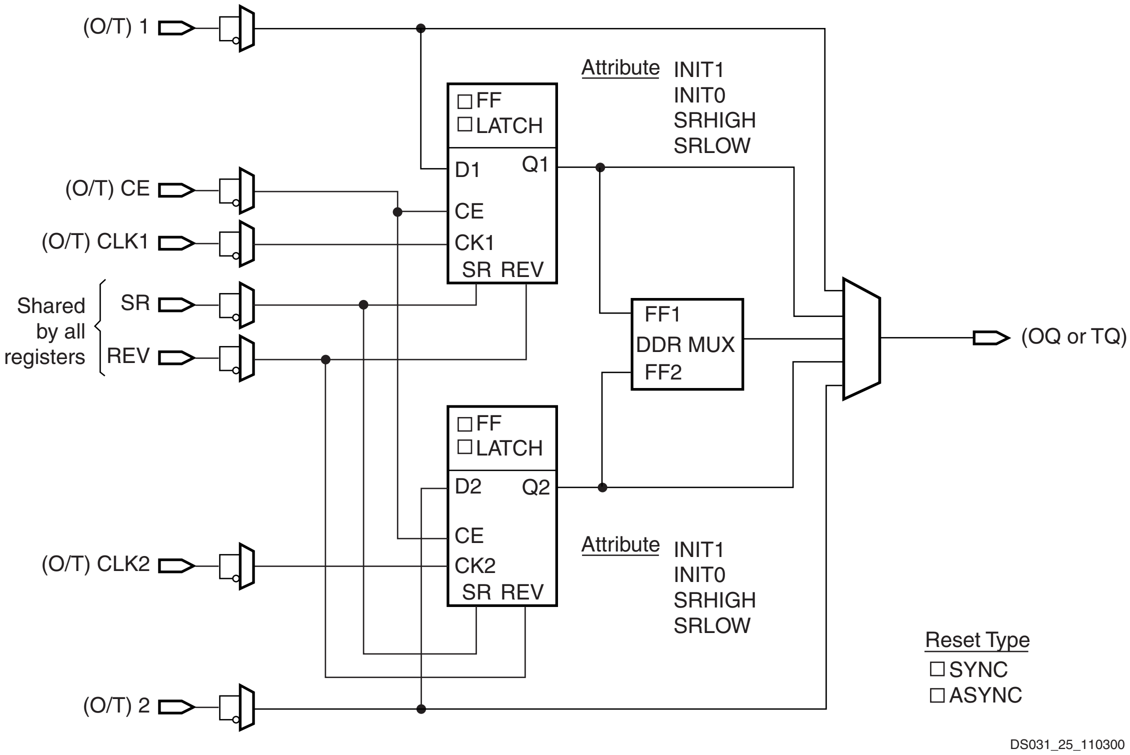


Figure 4: **Register / Latch Configuration in an IOB Block**

Input/Output Individual Options

Each device pad has optional pull-up, pull-down, and weak-keeper in LVTTL and LVCMOS SelectI/O configurations, as illustrated in Figure 5. Values of the optional pull-up and pull-down resistors are in the range 10 - 60 KΩ, which is the specification for V_{CCO} when operating at 3.3 V (from 3.0 to 3.6 V only). The clamp diode is always present, even when power is not.

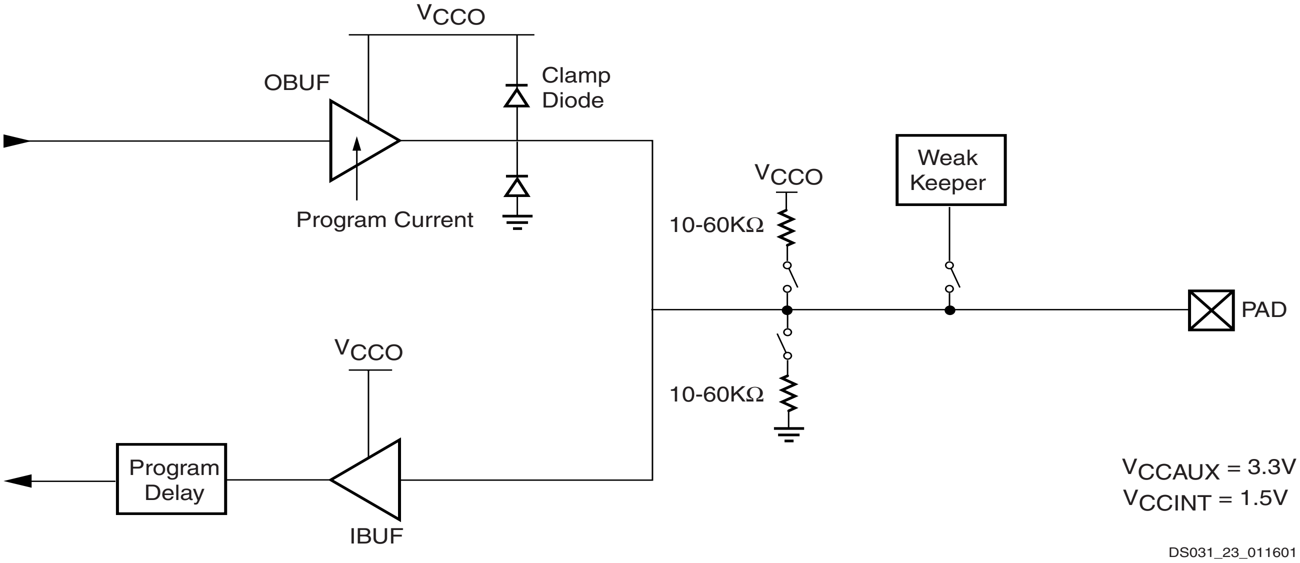


Figure 5: **LVTTL, LVCMOS or PCI SelectI/O Standards**

The optional weak-keeper circuit is connected to each output. When selected, the circuit monitors the voltage on the pad and weakly drives the pin High or Low. If the pin is connected to a multiple-source signal, the weak-keeper holds the signal in its last state if all drivers are disabled. Maintaining a valid logic level in this way eliminates bus chatter; pull-up or pull-down override the weak-keeper circuit.

LVTTL sinks and sources current up to 24 mA. The current is programmable for LVTTL and LVCMOS SelectI/O standards (see Table 4). Drive-strength and slew-rate controls for each output driver, minimize bus transients. For LVDCI and LVDCI_DV2 standards, drive strength and slew-rate controls are not available.

Table 4: **LVTTL and LVCMOS Programmable Currents (Sink and Source)**

SelectI/O	Programmable Current (Worst-Case Guaranteed Minimum)						
LVTTL	2 mA	4 mA	6 mA	8 mA	12 mA	16 mA	24 mA
LVCMOS33	2 mA	4 mA	6 mA	8 mA	12 mA	16 mA	24 mA
LVCMOS25	2 mA	4 mA	6 mA	8 mA	12 mA	16 mA	24 mA
LVCMOS18	2 mA	4 mA	6 mA	8 mA	12 mA	16 mA	n/a
LVCMOS15	2 mA	4 mA	6 mA	8 mA	12 mA	16 mA	n/a

Figure 6 shows the SSTL2, SSTL3, and HSTL configurations. HSTL can sink current up to 48 mA. (HSTL IV)

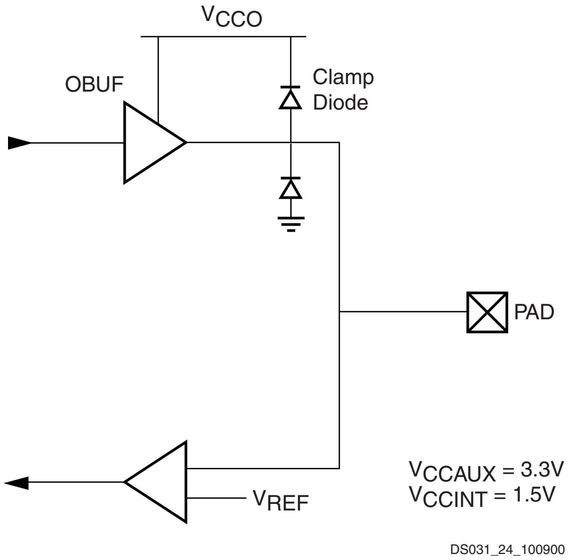


Figure 6: **SSTL or HSTL SelectI/O Standards**

All pads are protected against damage from electrostatic discharge (ESD) and from over-voltage transients. Virtex-II uses two memory cells to control the configuration of an I/O as an input. This is to reduce the probability of an I/O configured as an input from flipping to an output when subjected to a single event upset (SEU) in space applications.

Prior to configuration, all outputs not involved in configuration are forced into their high-impedance state. The pull-down resistors and the weak-keeper circuits are inactive. The dedicated pin HSWAP_EN controls the pull-up resistors prior to configuration. By default, HSWAP_EN is set high, which disables the pull-up resistors on user I/O pins. When HSWAP_EN is set low, the pull-up resistors are activated on user I/O pins.

All Virtex-II IOBs support IEEE 1149.1 compatible boundary scan testing.

Input Path

The Virtex-II IOB input path routes input signals directly to internal logic and / or through an optional input flip-flop or latch, or through the DDR input registers. An optional delay element at the D-input of the storage element eliminates pad-to-pad hold time. The delay is matched to the internal clock-distribution delay of the Virtex-II device, and when used, assures that the pad-to-pad hold time is zero.

Each input buffer can be configured to conform to any of the low-voltage signaling standards supported. In some of these standards the input buffer utilizes a user-supplied threshold voltage, V_{REF}. The need to supply V_{REF} imposes constraints on which standards can be used in the same bank. See I/O banking description.

Output Path

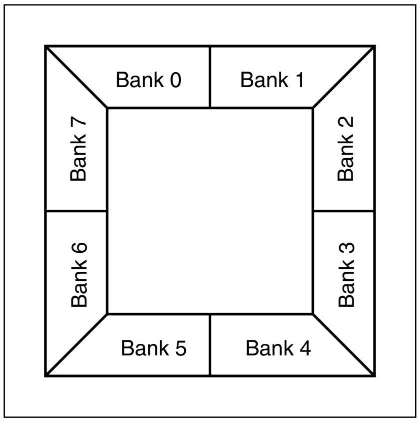
The output path includes a 3-state output buffer that drives the output signal onto the pad. The output and / or the 3-state signal can be routed to the buffer directly from the internal logic or through an output / 3-state flip-flop or latch, or through the DDR output / 3-state registers.

Each output driver can be individually programmed for a wide range of low-voltage signaling standards. In most signaling standards, the output High voltage depends on an externally supplied V_{CCO} voltage. The need to supply V_{CCO} imposes constraints on which standards can be used in the same bank. See I/O banking description.

I/O Banking

Some of the I/O standards described above require V_{CCO} and V_{REF} voltages. These voltages are externally supplied and connected to device pins that serve groups of IOB blocks, called banks. Consequently, restrictions exist about which I/O standards can be combined within a given bank.

Eight I/O banks result from dividing each edge of the FPGA into two banks, as shown in Figure 7 and Figure 8. Each bank has multiple V_{CCO} pins, all of which must be connected to the same voltage. This voltage is determined by the output standards in use.

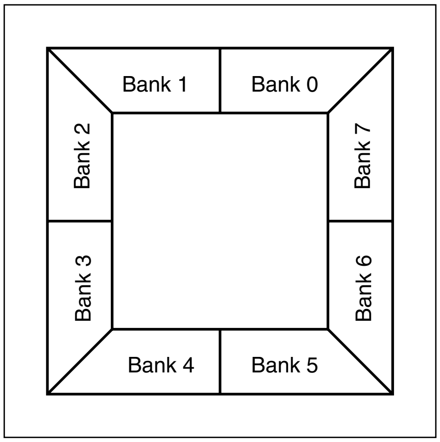


ug002_c2_014_112900

Figure 7: **Virtex-II I/O Banks: Top View for Wire-Bond Packages (CS, FG, & BG)**

Within a bank, output standards can be mixed only if they use the same V_{CCO}. Table 5 lists compatible output standards. GTL and GTLP appear under all voltages because their open-drain outputs do not depend on V_{CCO}.

Some input standards require a user-supplied threshold voltage, V_{REF}. In this case, certain user-I/O pins are automatically configured as inputs for the V_{REF} voltage. Approximately one in six of the I/O pins in the bank assume this role. Table 6 lists compatible input standards.



ds031_66_112900

Figure 8: **Virtex-II I/O Banks: Top View for Flip-Chip Packages (FF & BF)**

V_{REF} pins within a bank are interconnected internally, and consequently only one V_{REF} voltage can be used within each bank. However, for correct operation, all V_{REF} pins in the bank must be connected to the external reference voltage source.

Table 5: **Compatible Output Standards**

V_{CCO}	Compatible Standards
3.3 V	PCI, LVTTL, SSTL3 (I & II), AGP-2X, LVDS_33, LVDSEXT_33, LVCMOS33, LVDCI_33, LVDCI_DV2_33, SSTL3_DCI (I & II), LVPECL, GTL, GTLP
2.5 V	SSTL2 (I & II), LVCMOS25, GTL, GTLP, LVDS_25, LVDSEXT_25, LVDCI_25, LVDCI_DV2_25, SSTL2_DCI (I & II), LDT, ULVDS, BLVDS
1.8 V	LVCMOS18, GTL, GTLP, LVDCI_18, LVDCI_DV2_18
1.5 V	HSTL (I, II, III, & IV), LVCMOS15, GTL, GTLP, LVDCI_15, LVDCI_DV2_15, GTLP_DCI, HSTL_DCI (I,II, III & IV)
1.2V	GTL_DCI

The V_{CCO} and the V_{REF} pins for each bank appear in the device pinout tables. Within a given package, the number of V_{REF} and V_{CCO} pins can vary depending on the size of device. In larger devices, more I/O pins convert to V_{REF} pins. Since these are always a superset of the V_{REF} pins used for smaller devices, it is possible to design a PCB that permits migration to a larger device if necessary.

All V_{REF} pins for the largest device anticipated must be connected to the V_{REF} voltage and not used for I/O. In smaller devices, some V_{CCO} pins used in larger devices do not connect within the package. These unconnected pins can be left unconnected externally, or, if necessary, they can be connected to the V_{CCO} voltage to permit migration to a larger device.

Table 6: **Compatible Input Standards**

V_{CCO} V_{REF}	3.3V	2.5V	1.8V	1.5V	1.2V
No V_{REF}	LVTTL, LVDCI_33, LVDCI_DV2_33, LVCMOS33, PCI33_3, PCI66_3, PCI-X, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25[2]	LVCMOS25, LVDCI_25, LVDCI_DV2_25, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS18, LVDCI_18, LVDCI_DV2_18, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS15, LVDCI_15, LVDCI_DV2_15, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25
1.5V	LVTTL, LVDCI_33, LVDCI_DV2_33, LVCMOS33, PCI33_3, PCI66_3, PCI-X, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25, SSTL3_I_DCI, SSTL3_II_DCI	LVCMOS25, LVDCI_25, LVDCI_DV2_25, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS18, LVDCI_18, LVDCI_DV2_18, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS15, LVDCI_15, LVDCI_DV2_15, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25
	SSTL3_I, SSTL3_II	SSTL3_I, SSTL3_II	SSTL3_I, SSTL3_II	SSTL3_I, SSTL3_II	SSTL3_I, SSTL3_II
1.32V	LVTTL, LVDCI_33, LVDCI_DV2_33, LVCMOS33, PCI33_3, PCI66_3, PCI-X, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS25, LVDCI_25, LVDCI_DV2_25, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS18, LVDCI_18, LVDCI_DV2_18, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS15, LVDCI_15, LVDCI_DV2_15, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25
	AGP-2X/AGP	AGP-2X/AGP	AGP-2X/AGP	AGP-2X/AGP	AGP-2X/AGP
1.25V	LVTTL, LVDCI_33, LVDCI_DV2_33, LVCMOS33, PCI33_3, PCI66_3, PCI-X, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS25, LVDCI_25, LVDCI_DV2_25, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25, SSTL2_I_DCI, SSTL2_II_DCI	LVCMOS18, LVDCI_18, LVDCI_DV2_18, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS15, LVDCI_15, LVDCI_DV2_15, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25
	SSTL2_I, SSTL2_II	SSTL2_I, SSTL2_II	SSTL2_I, SSTL2_II	SSTL2_I, SSTL2_II	SSTL2_I, SSTL2_II

Table 6: **Compatible Input Standards**

V_{CCO} V_{REF}	3.3V	2.5V	1.8V	1.5V	1.2V
1.0V	LVTTL, LVDCI_33, LVDCI_DV2_33, LVCMOS33, PCI33_3, PCI66_3, PCI-X, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS25, LVDCI_25, LVDCI_DV2_25, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS18, LVDCI_18, LVDCI_DV2_18, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS15, LVDCI_15, LVDCI_DV2_15, GTLP_DCI, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25
	GTLP	GTLP	GTLP	GTLP	GTLP
0.9V	LVTTL, LVDCI_33, LVDCI_DV2_33, LVCMOS33, PCI33_3, PCI66_3, PCI-X, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS25, LVDCI_25, LVDCI_DV2_25, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS18, LVDCI_18, LVDCI_DV2_18, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS15, LVDCI_15, LVDCI_DV2_15, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25, HSTL_III_DCI, HSTL_IV_DCI	LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25
	HSTL_III, HSTL_IV	HSTL_III, HSTL_IV	HSTL_III, HSTL_IV	HSTL_III, HSTL_IV	HSTL_III, HSTL_IV
0.8V	LVTTL, LVDCI_33, LVDCI_DV2_33, LVCMOS33, PCI33_3, PCI66_3, PCI-X, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS25, LVDCI_25, LVDCI_DV2_25, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS18, LVDCI_18, LVDCI_DV2_18, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS15, LVDCI_15, LVDCI_DV2_15, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	GTL_DCI, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25
	GTL	GTL	GTL	GTL	GTL
0.75V	LVTTL, LVDCI_33, LVDCI_DV2_33, LVCMOS33, PCI33_3, PCI66_3, PCI-X, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS25, LVDCI_25, LVDCI_DV2_25, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS18, LVDCI_18, LVDCI_DV2_18, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25	LVCMOS15, LVDCI_15, LVDCI_DV2_15, LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25, HSTL_I_DCI, HSTL_II_DCI	LVDS_33, LVDSEXT_33, LVPECL_33, LVDS_25, LVDSEXT_25, LDT, BLVDS, ULVDS_25
	HSTL_I, HSTL_II	HSTL_I, HSTL_II	HSTL_I, HSTL_II	HSTL_I, HSTL_II	HSTL_I, HSTL_II

Notes:

1. Inputs that are V_{REF} controlled are completely independent of those that are V_{CCO} controlled. Therefore, V_{REF} controlled inputs can also be placed in banks with inputs and outputs of different voltages that are V_{CCO} controlled.
2. All non-DCI differential inputs are V_{CCAUX} controlled. This makes them (Inputs Only) very flexible in terms of banking rules.

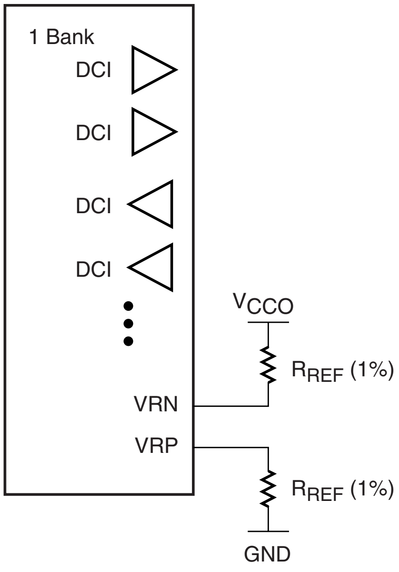
Digitally Controlled Impedance (DCI)

Today's chip output signals with fast edge rates require termination to prevent reflections and maintain signal integrity. High pin count packages (especially ball grid arrays) can not accommodate external termination resistors.

Virtex-II DCI provides controlled impedance drivers and on-chip termination for single-ended I/Os. This eliminates the need for external resistors, and improves signal integrity. The DCI feature can be used on any IOB by selecting one of the DCI I/O standards.

When applied to inputs, DCI provides input parallel termination. When applied to outputs, DCI provides controlled impedance drivers (series termination) or output parallel termination.

DCI operates independently on each I/O bank. When a DCI I/O standard is used in a particular I/O bank, external reference resistors must be connected to two dual-function pins on the bank. These resistors, voltage reference of N transistor (VRN) and the voltage reference of P transistor (VRP) are shown in Figure 9.



DS031_50_101200

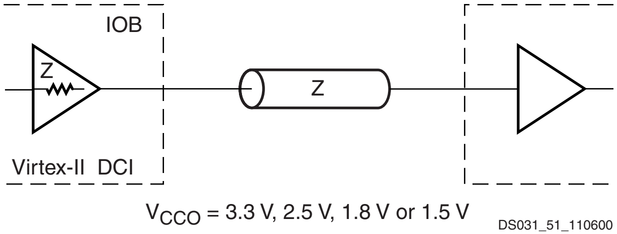
Figure 9: **DCI in a Virtex-II Bank**

When used with a terminated I/O standard, the value of resistors are specified by the standard (typically 50 Ω). When used with a controlled impedance driver, the resistors set the output impedance of the driver within the specified range (25 Ω to 100 Ω). For all series and parallel terminations listed in Table 7 and Table 8, the reference resistors must have the same value for any given bank. One percent resistors are recommended.

The DCI system adjusts the I/O impedance to match the two external reference resistors, or half of the reference resistors, and compensates for impedance changes due to voltage and/or temperature fluctuations. The adjustment is done by turning parallel transistors in the IOB on or off.

Controlled Impedance Drivers (Series Termination)

DCI can be used to provide a buffer with a controlled output impedance. It is desirable for this output impedance to match the transmission line impedance (Z). Virtex-II input buffers also support LVDCI and LVDCI_DV2 I/O standards.



DS031_51_110600

Figure 10: **Internal Series Termination**

Table 7: **SelectI/O Controlled Impedance Buffers**

V_{CCO}	DCI	DCI Half Impedance
3.3 V	LVDCI_33	LVDCI_DV2_33
2.5 V	LVDCI_25	LVDCI_DV2_25
1.8 V	LVDCI_18	LVDCI_DV2_18
1.5 V	LVDCI_15	LVDCI_DV2_15

Controlled Impedance Drivers (Parallel Termination)

DCI also provides on-chip termination for SSTL3, SSTL2, HSTL (Class I, II, III, or IV), and GTL/GTLP receivers or transmitters on bidirectional lines.

Table 8 lists the on-chip parallel terminations available in Virtex-II devices. V_{CCO} must be set according to Table 3. Note that there is a V_{CCO} requirement for GTL_DCI and GTLP_DCI, due to the on-chip termination resistor.

Table 8: **SelectI/O Buffers With On-Chip Parallel Termination**

I/O Standard	External Termination	On-Chip Termination
SSTL3 Class I	SSTL3_I	SSTL3_I_DCI(1)
SSTL3 Class II	SSTL3_II	SSTL3_II_DCI(1)
SSTL2 Class I	SSTL2_I	SSTL2_I_DCI(1)
SSTL2 Class II	SSTL2_II	SSTL2_II_DCI(1)
HSTL Class I	HSTL_I	HSTL_I_DCI
HSTL Class II	HSTL_II	HSTL_II_DCI
HSTL Class III	HSTL_III	HSTL_III_DCI
HSTL Class IV	HSTL_IV	HSTL_IV_DCI
GTL	GTL	GTL_DCI
GTLP	GTLP	GTLP_DCI

Notes:

1. SSTL Compatible

Figure 11 provides examples illustrating the use of the HSTL_I_DCI, HSTL_II_DCI, HSTL_III_DCI, and HSTL_IV_DCI I/O standards. For a complete list, see the *Virtex-II User Guide.*

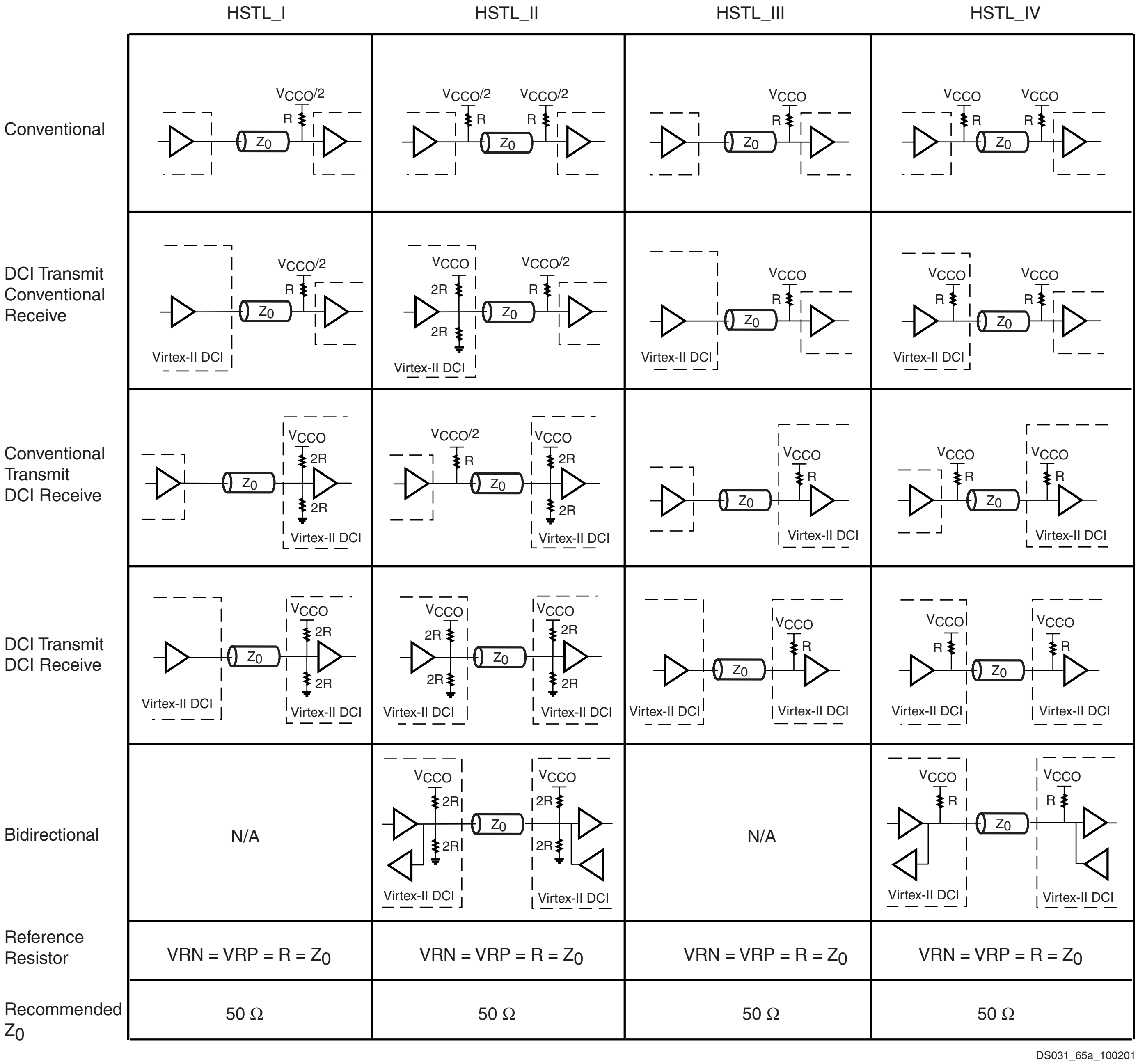


Figure 11: **HSTL DCI Usage Examples**

Figure 12 provides examples illustrating the use of the SSTL2_I_DCI, SSTL2_II_DCI, SSTL3_I_DCI, and SSTL3_II_DCI I/O standards. For a complete list, see the *Virtex-II User Guide*.

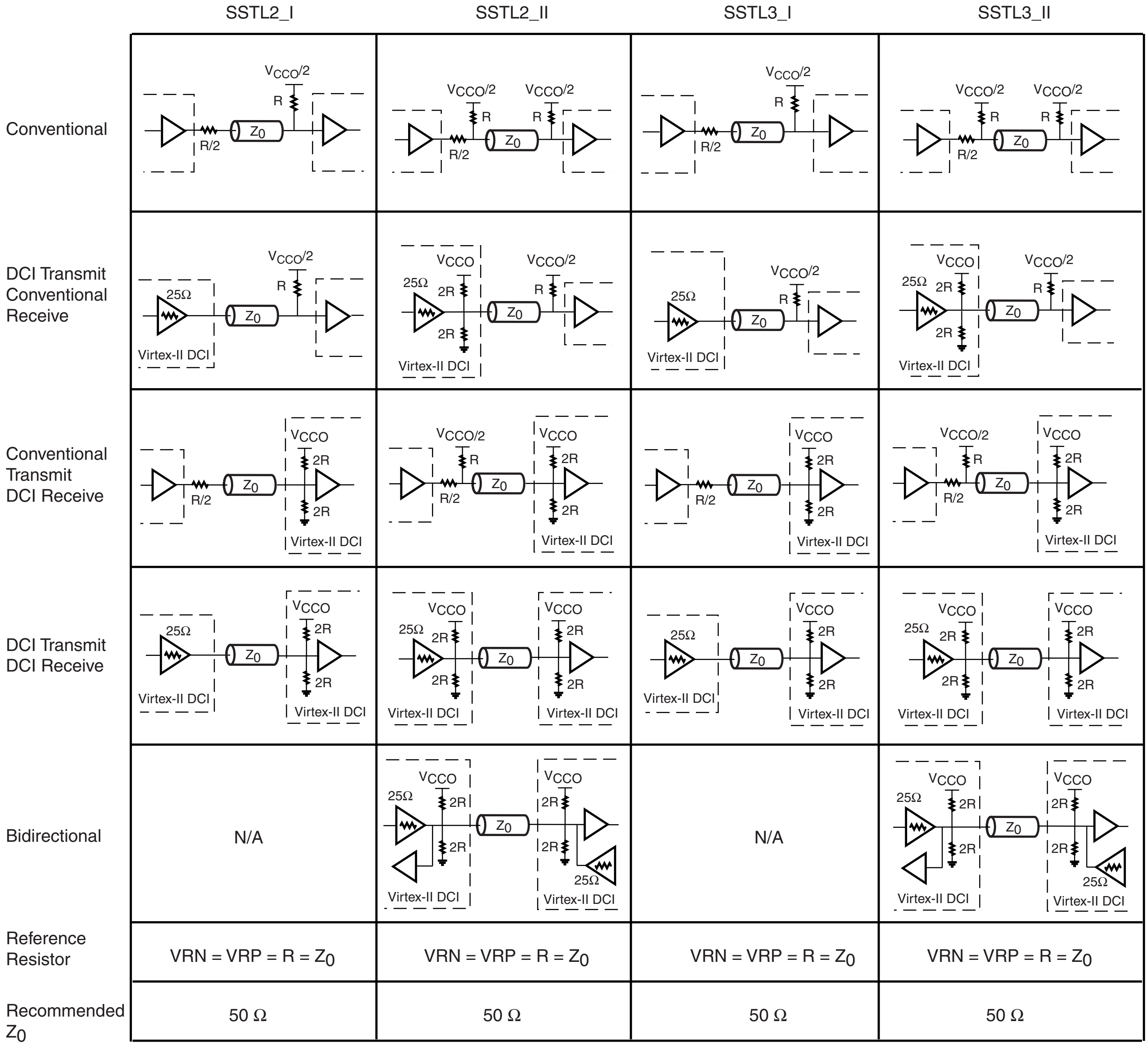


Figure 12: **SSTL DCI Usage Examples**

Configurable Logic Blocks (CLBs)

The Virtex-II configurable logic blocks (CLB) are organized in an array and are used to build combinatorial and synchronous logic designs. Each CLB element is tied to a switch matrix to access the general routing matrix, as shown in Figure 13. A CLB element comprises 4 similar slices, with fast local feedback within the CLB. The four slices are split in two columns of two slices with two independent carry logic chains and one common shift chain.

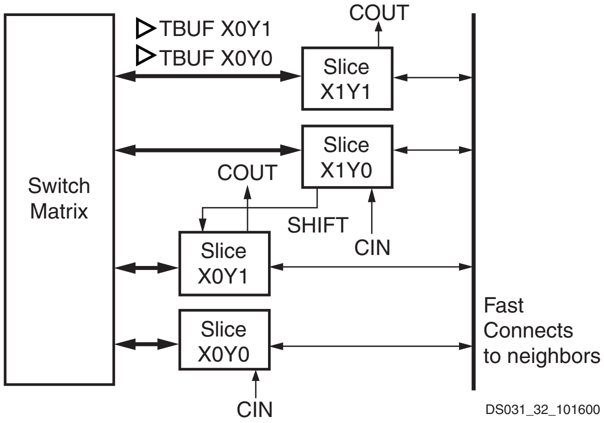


Figure 13: **Virtex-II CLB Element**

Slice Description

Each slice includes two 4-input function generators, carry logic, arithmetic logic gates, wide function multiplexers and two storage elements. As shown in Figure 14, each 4-input function generator is programmable as a 4-input LUT, 16 bits of distributed SelectRAM memory, or a 16-bit variable-tap shift register element.

The output from the function generator in each slice drives both the slice output and the D input of the storage element. Figure 15 shows a more detailed view of a single slice.

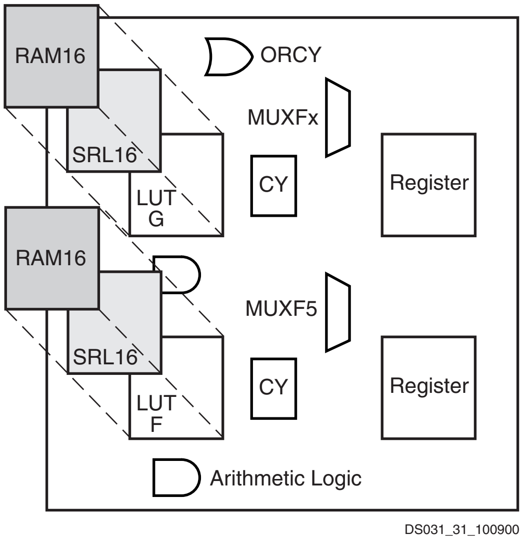


Figure 14: **Virtex-II Slice Configuration**

Configurations

Look-Up Table

Virtex-II function generators are implemented as 4-input look-up tables (LUTs). Four independent inputs are provided to each of the two function generators in a slice (F and G). These function generators are each capable of implementing any arbitrarily defined boolean function of four inputs. The propagation delay is therefore independent of the function implemented. Signals from the function generators can exit the slice (X or Y output), can input the XOR dedicated gate (see arithmetic logic), or input the carry-logic multiplexer (see fast look-ahead carry logic), or feed the D input of the storage element, or go to the MUXF5 (not shown in Figure 15).

In addition to the basic LUTs, the Virtex-II slice contains logic (MUXF5 and MUXFX multiplexers) that combines function generators to provide any function of five, six, seven, or eight inputs. The MUXFX are either MUXF6, MUXF7 or MUXF8 according to the slice considered in the CLB. Selected functions up to nine inputs (MUXF5 multiplexer) can be implemented in one slice. The MUXFX can also be a MUXF6, MUXF7, or MUXF8 multiplexers to map any functions of six, seven, or eight inputs and selected wide logic functions.

Register/Latch

The storage elements in a Virtex-II slice can be configured either as edge-triggered D-type flip-flops or as level-sensitive latches. The D input can be directly driven by the X or Y output via the DX or DY input, or by the slice inputs bypassing the function generators via the BX or BY input. The clock enable signal (CE) is active High by default. If left unconnected, the clock enable for that storage element defaults to the active state.

In addition to clock (CK) and clock enable (CE) signals, each slice has set and reset signals (SR and BY slice inputs). SR forces the storage element into the state specified by the attribute SRHIGH or SRLOW. SRHIGH forces a logic “1” when SR is asserted. SRLOW forces a logic “0”. When SR is used, a second input (BY) forces the storage element into the opposite state. The reset condition is predominant over the set condition. (See Figure 16.)

The initial state after configuration or global initial state is defined by a separate INIT0 and INIT1 attribute. By default, setting the SRLOW attribute sets INIT0, and setting the SRHIGH attribute sets INIT1.

For each slice, set and reset can be set to be synchronous or asynchronous. Virtex-II devices also have the ability to set INIT0 and INIT1 independent of SRHIGH and SRLOW.

The control signals clock (CLK), clock enable (CE) and set/reset (SR) are common to both storage elements in one slice. All of the control signals have independent polarity. Any inverter placed on a control input is automatically absorbed.

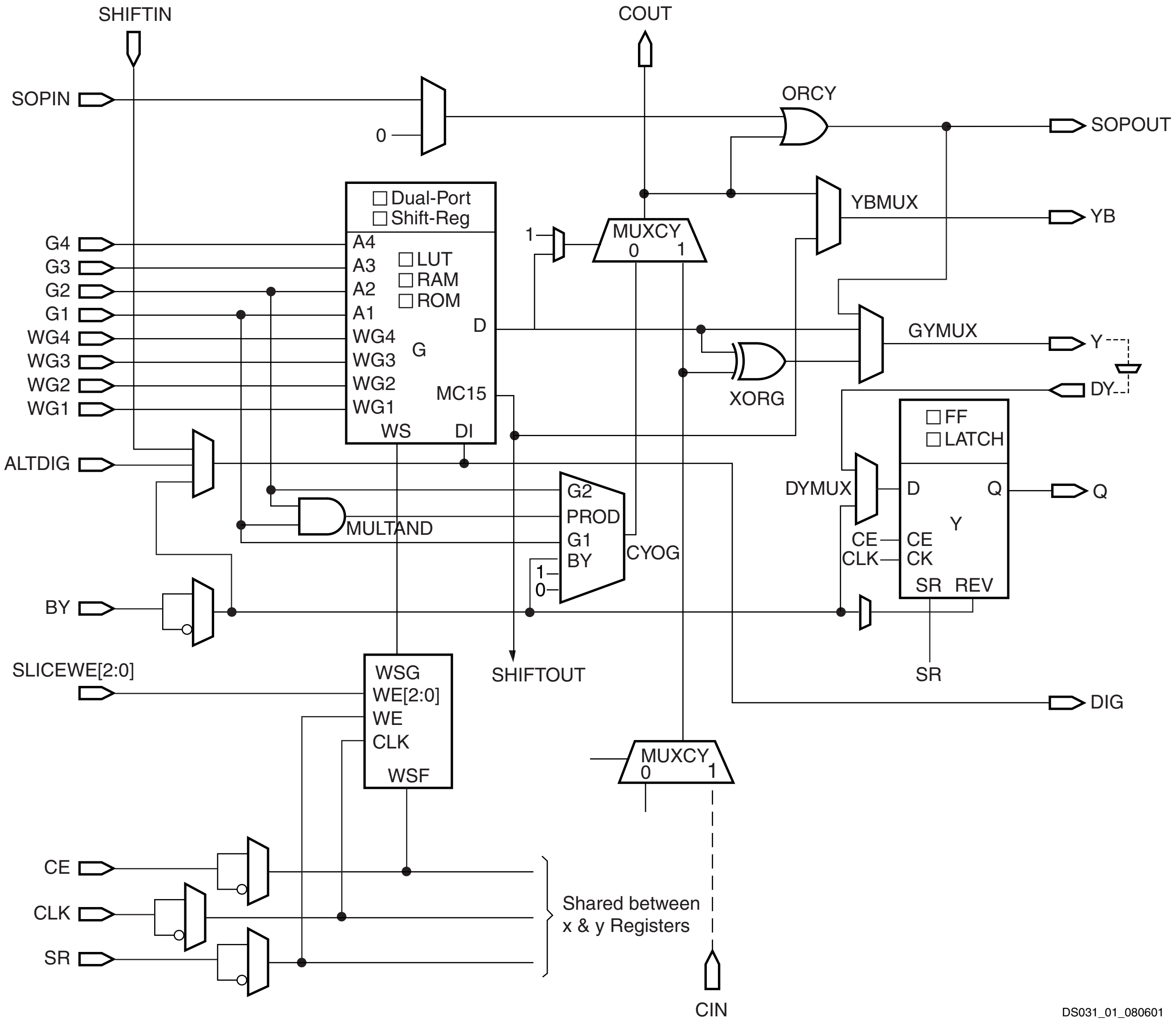


Figure 15: **Virtex-II Slice (Top Half)**

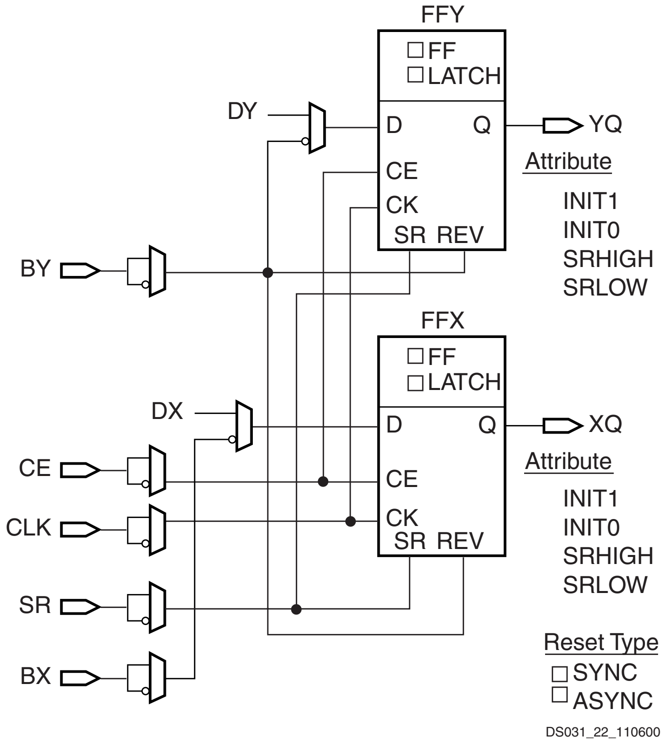


Figure 16: **Register / Latch Configuration in a Slice**

The set and reset functionality of a register or a latch can be configured as follows:

- No set or reset
- Synchronous set
- Synchronous reset
- Synchronous set and reset
- Asynchronous set (preset)
- Asynchronous reset (clear)
- Asynchronous set and reset (preset and clear)

The synchronous reset has precedence over a set, and an asynchronous clear has precedence over a preset.

Distributed SelectRAM Memory

Each function generator (LUT) can implement a 16 x 1-bit synchronous RAM resource called a distributed SelectRAM element. The SelectRAM elements are configurable within a CLB to implement the following:

- Single-Port 16 x 8 bit RAM
- Single-Port 32 x 4 bit RAM
- Single-Port 64 x 2 bit RAM
- Single-Port 128 x 1 bit RAM
- Dual-Port 16 x 4 bit RAM
- Dual-Port 32 x 2 bit RAM
- Dual-Port 64 x 1 bit RAM

Distributed SelectRAM memory modules are synchronous (write) resources. The combinatorial read access time is extremely fast, while the synchronous write simplifies high-speed designs. A synchronous read can be implemented with a storage element in the same slice. The distributed SelectRAM memory and the storage element share the same clock input. A Write Enable (WE) input is active High, and is driven by the SR input.

Table 9 shows the number of LUTs (2 per slice) occupied by each distributed SelectRAM configuration.

Table 9: **Distributed SelectRAM Configurations**

RAM	Number of LUTs
16 x 1S	1
16 x 1D	2
32 x 1S	2
32 x 1D	4
64 x 1S	4
64 x 1D	8
128 x 1S	8

Notes:

1. S = single-port configuration; D = dual-port configuration

For single-port configurations, distributed SelectRAM memory has one address port for synchronous writes and asynchronous reads.

For dual-port configurations, distributed SelectRAM memory has one port for synchronous writes and asynchronous reads and another port for asynchronous reads. The function generator (LUT) has separated read address inputs (A1, A2, A3, A4) and write address inputs (WG1/WF1, WG2/WF2, WG3/WF3, WG4/WF4).

In single-port mode, read and write addresses share the same address bus. In dual-port mode, one function generator (R/W port) is connected with shared read and write addresses. The second function generator has the A inputs (read) connected to the second read-only port address and the W inputs (write) shared with the first read/write port address.

Figure 17, Figure 18, and Figure 19 illustrate various example configurations.

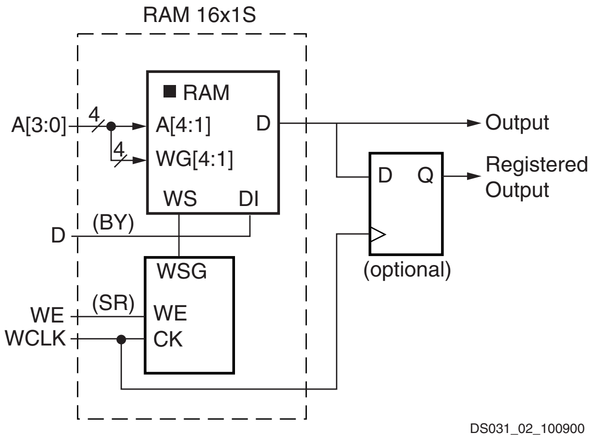


Figure 17: **Distributed SelectRAM (RAM16x1S)**

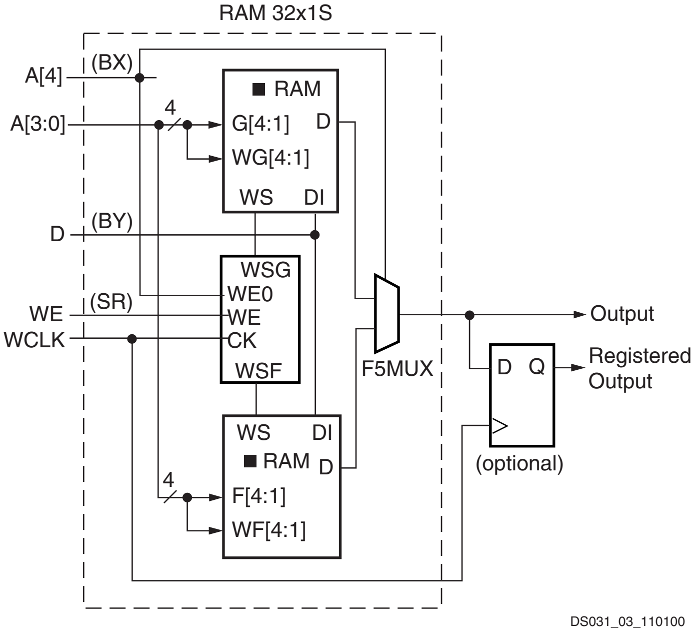


Figure 18: **Single-Port Distributed SelectRAM (RAM32x1S)**

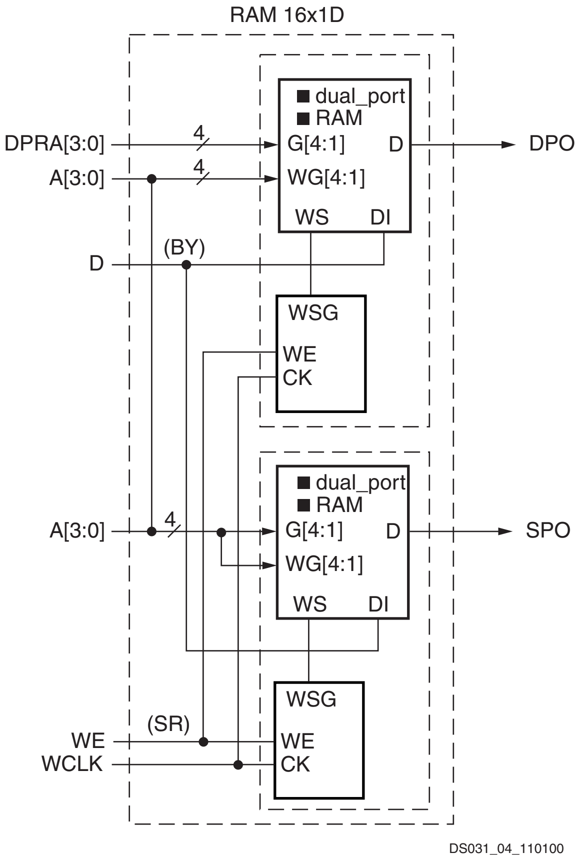


Figure 19: **Dual-Port Distributed SelectRAM (RAM16x1D)**

Similar to the RAM configuration, each function generator (LUT) can implement a 16 x 1-bit ROM. Five configurations are available: ROM16x1, ROM32x1, ROM64x1, ROM128x1, and ROM256x1. The ROM elements are cascadable to implement wider or/and deeper ROM. ROM contents are loaded at configuration. Table 10 shows the number of LUTs occupied by each configuration.

Table 10: **ROM Configuration**

ROM	Number of LUTs
16 x 1	1
32 x 1	2
64 x 1	4
128 x 1	8 (1 CLB)
256 x 1	16 (2 CLBs)

Shift Registers

Each function generator can also be configured as a 16-bit shift register. The write operation is synchronous with a clock input (CLK) and an optional clock enable, as shown in Figure 20. A dynamic read access is performed through the 4-bit address bus, A[3:0]. The configurable 16-bit shift register cannot be set or reset. The read is asynchronous, however the storage element or flip-flop is available to implement a synchronous read. The storage element should always be used with a constant address. For example, when building an 8-bit shift register and configuring the addresses to point to the 7th bit, the 8th bit can be the flip-flop. The overall system performance is improved by using the superior clock-to-out of the flip-flops.

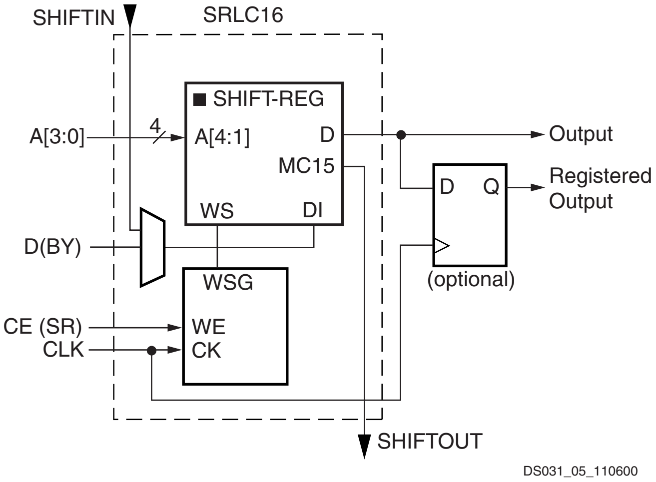


Figure 20: **Shift Register Configurations**

An additional dedicated connection between shift registers allows connecting the last bit of one shift register to the first bit of the next, without using the ordinary LUT output. (See Figure 21.) Longer shift registers can be built with dynamic access to any bit in the chain. The shift register chaining and the MUXF5, MUXF6, and MUXF7 multiplexers allow up to a 128-bit shift register with addressable access to be implemented in one CLB.

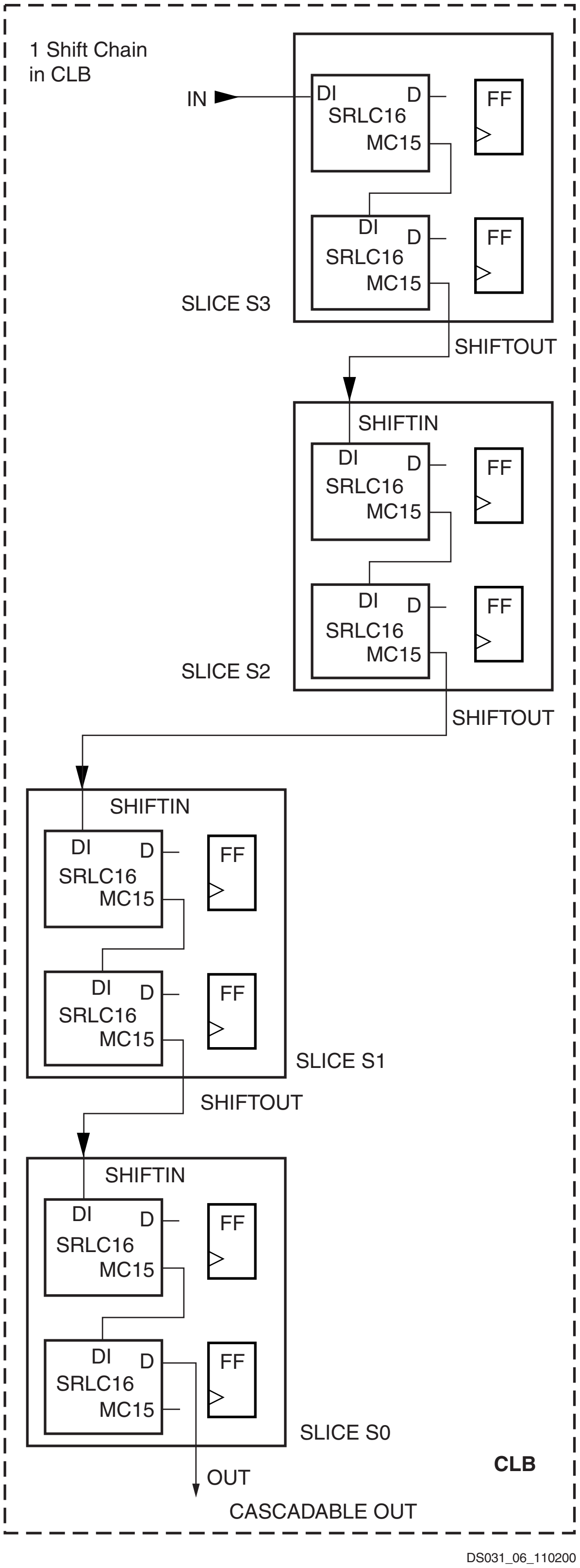


Figure 21: **Cascadable Shift Register**

Multiplexers

Virtex-II function generators and associated multiplexers can implement the following:

- 4:1 multiplexer in one slice
- 8:1 multiplexer in two slices
- 16:1 multiplexer in one CLB element (4 slices)
- 32:1 multiplexer in two CLB elements (8 slices)

Each Virtex-II slice has one MUXF5 multiplexer and one MUXFX multiplexer. The MUXFX multiplexer implements the MUXF6, MUXF7, or MUXF8, as shown in Figure 22. Each CLB element has two MUXF6 multiplexers, one MUXF7 multiplexer and one MUXF8 multiplexer. Examples of multiplexers are shown in the *Virtex-II User Guide*. Any LUT can implement a 2:1 multiplexer.

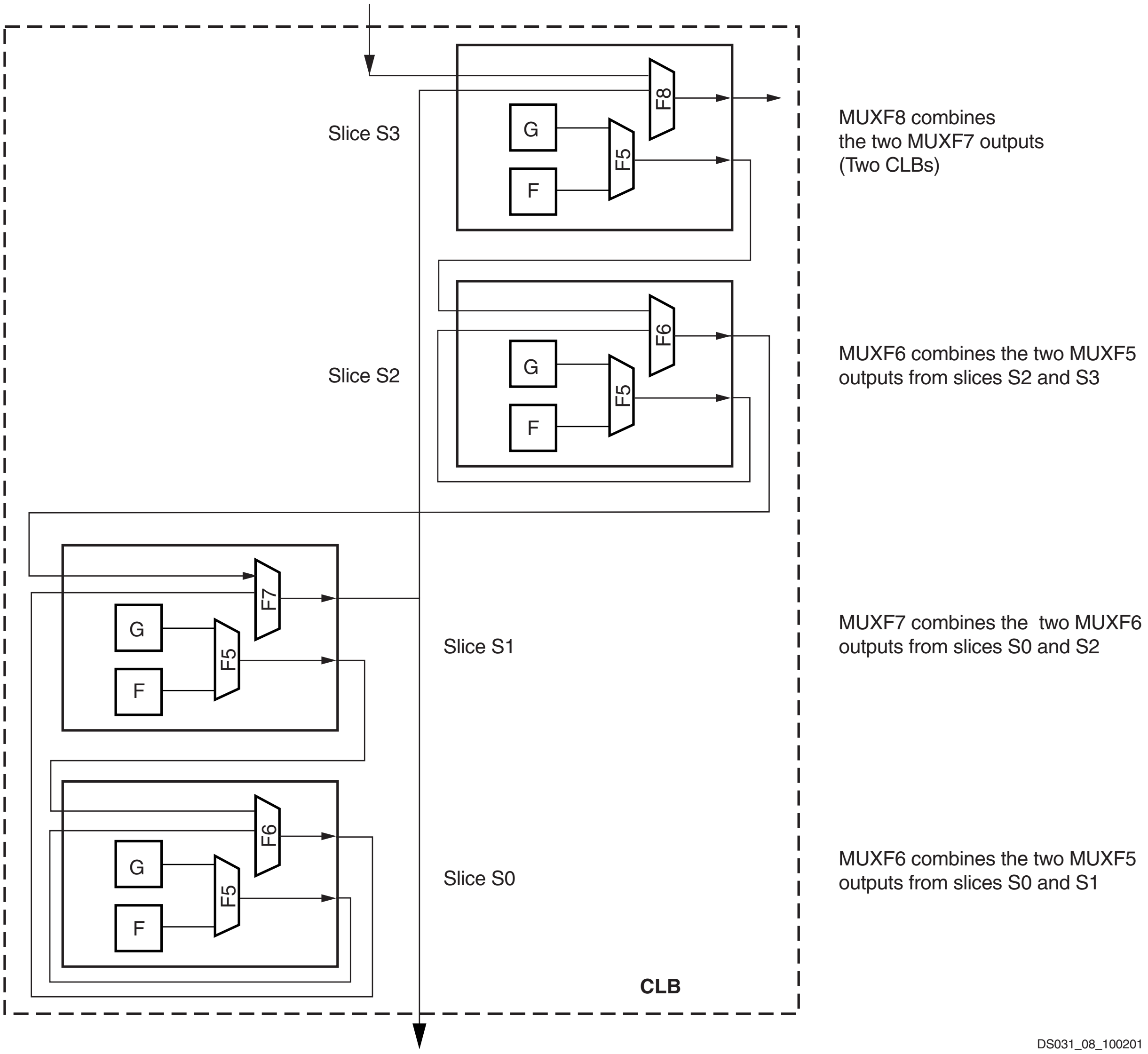


Figure 22: **MUXF5 and MUXFX multiplexers**

Fast Lookahead Carry Logic

Dedicated carry logic provides fast arithmetic addition and subtraction. The Virtex-II CLB has two separate carry chains, as shown in the Figure 23.

The height of the carry chains is two bits per slice. The carry chain in the Virtex-II device is running upward. The dedicated carry path and carry multiplexer (MUXCY) can also be used to cascade function generators for implementing wide logic functions.

Arithmetic Logic

The arithmetic logic includes an XOR gate that allows a 2-bit full adder to be implemented within a slice. In addition, a dedicated AND (MULT_AND) gate (shown in Figure 15) improves the efficiency of multiplier implementation.

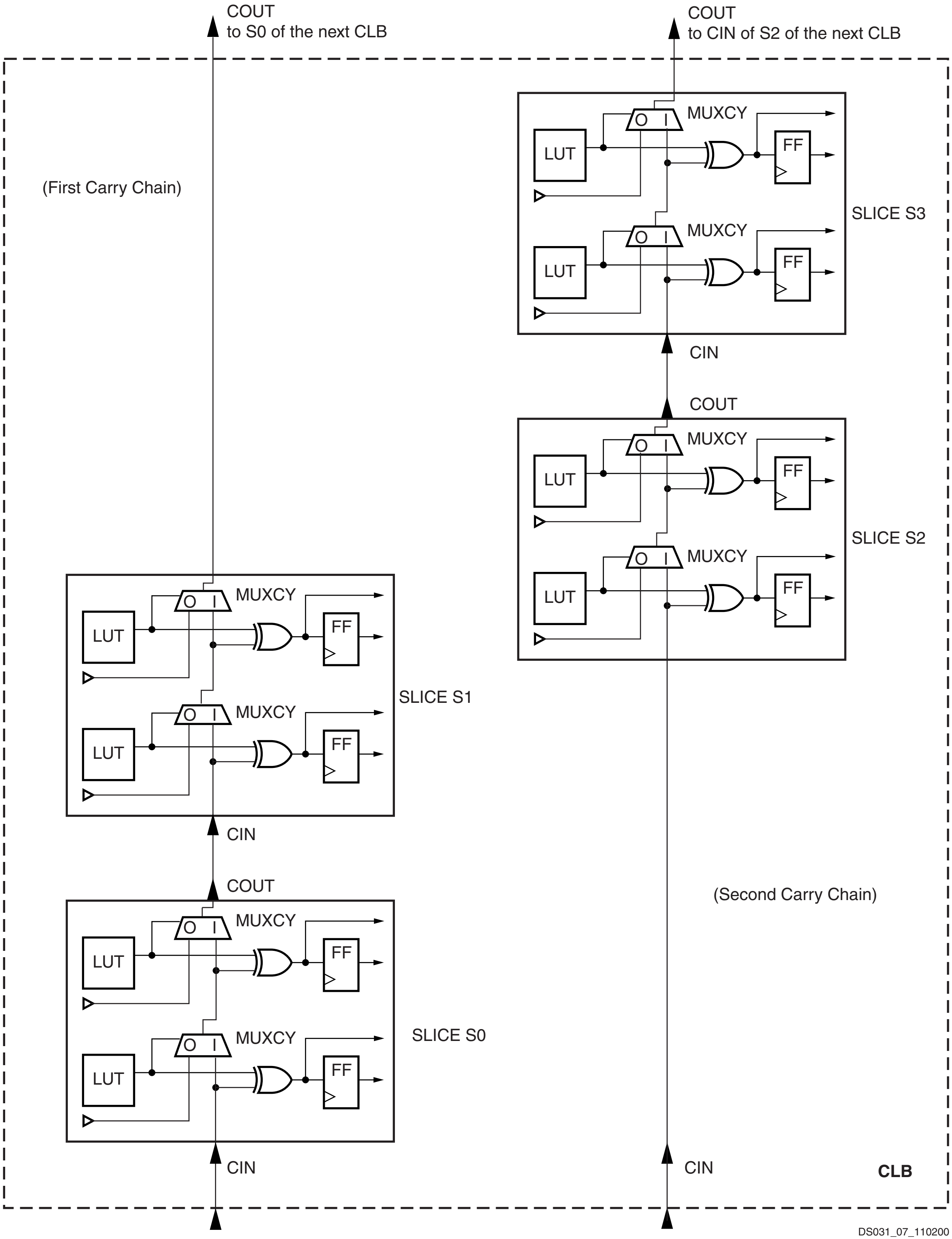


Figure 23: **Fast Carry Logic Path**

Sum of Products

Each Virtex-II slice has a dedicated OR gate named ORCY, ORing together outputs from the slices carryout and the ORCY from an adjacent slice. The ORCY gate with the dedicated Sum of Products (SOP) chain are designed for implementing large, flexible SOP chains. One input of each ORCY is connected through the fast SOP chain to the output of the previous ORCY in the same slice row. The second input is connected to the output of the top MUXCY in the same slice, as shown in Figure 24.

LUTs and MUXCYs can implement large AND gates or other combinatorial logic functions. Figure 25 illustrates LUT and MUXCY resources configured as a 16-input AND gate.

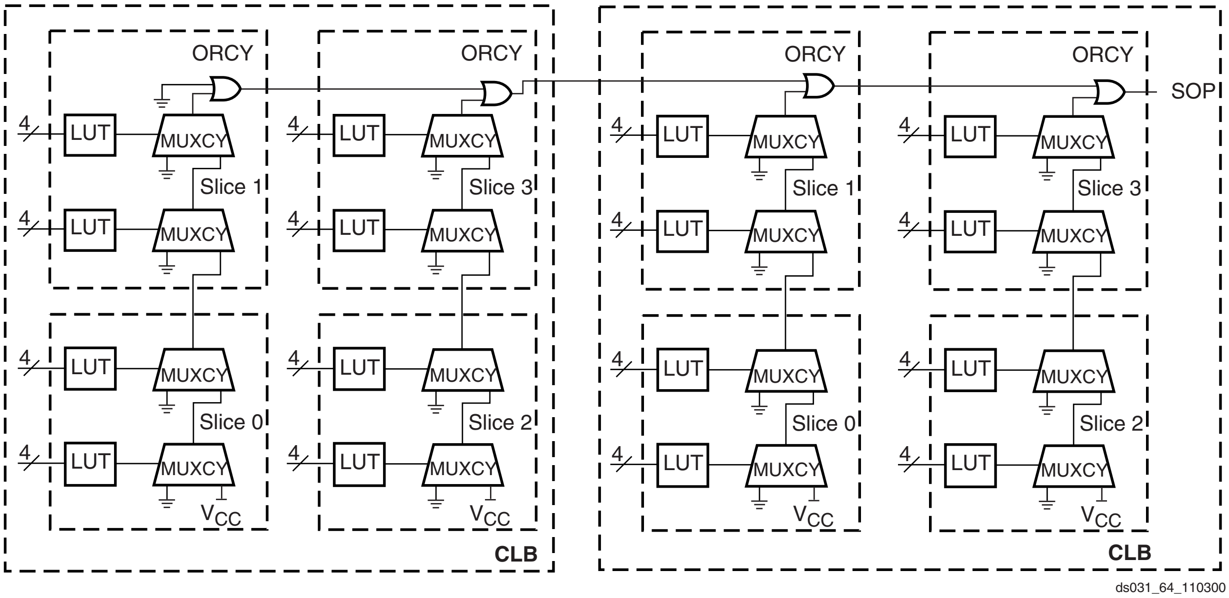


Figure 24: **Horizontal Cascade Chain**

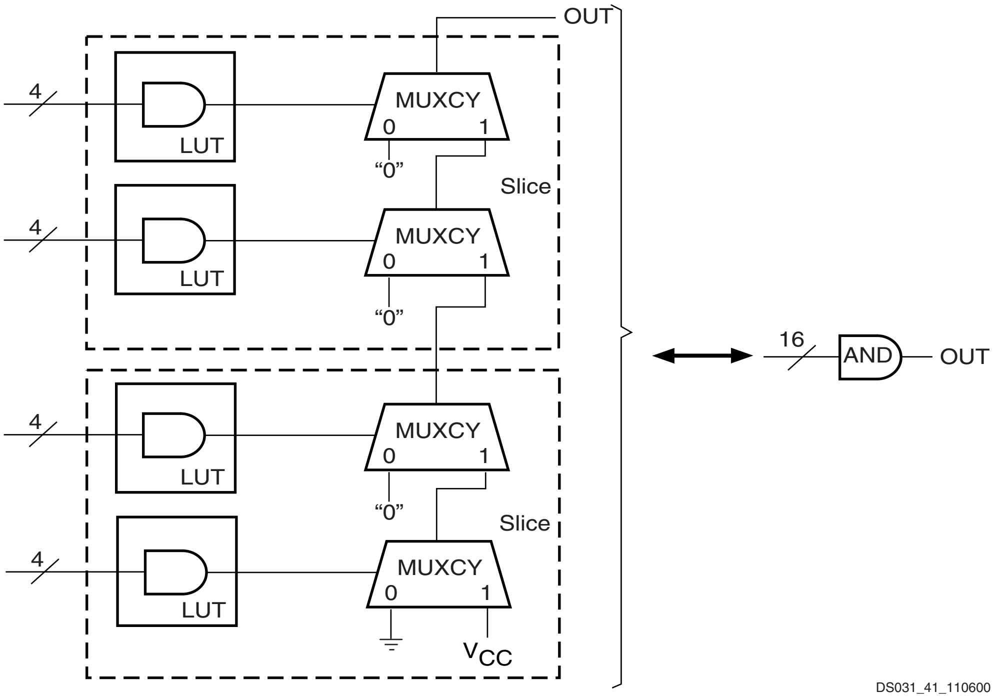


Figure 25: **Wide-Input AND Gate (16 Inputs)**

3-State Buffers

Introduction

Each Virtex-II CLB contains two 3-state drivers (TBUFs) that can drive on-chip busses. Each 3-state buffer has its own 3-state control pin and its own input pin.

Each of the four slices have access to the two 3-state buffers through the switch matrix, as shown in Figure 26. TBUFs in neighboring CLBs can access slice outputs by direct connects. The outputs of the 3-state buffers drive horizontal routing resources used to implement 3-state busses.

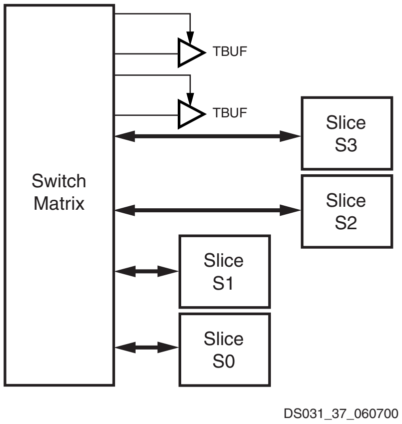


Figure 26: **Virtex-II 3-State Buffers**

The 3-state buffer logic is implemented using AND-OR logic rather than 3-state drivers, so that timing is more predictable and less load dependant especially with larger devices.

Locations / Organization

Four horizontal routing resources per CLB are provided for on-chip 3-state busses. Each 3-state buffer has access alternately to two horizontal lines, which can be partitioned as shown in Figure 27. The switch matrices corresponding to SelectRAM memory and multiplier or I/O blocks are skipped.

Number of 3-State Buffers

Table 11 shows the number of 3-state buffers available in each Virtex-II device. The number of 3-state buffers is twice the number of CLB elements.

Table 11: **Virtex-II 3-State Buffers**

Device	3-State Buffers per Row	Total Number of 3-State Buffers
XC2V40	16	128
XC2V80	16	256
XC2V250	32	768
XC2V500	48	1,536
XC2V1000	64	2,560
XC2V1500	80	3,840
XC2V2000	96	5,376
XC2V3000	112	7,168
XC2V4000	144	11,520
XC2V6000	176	16,896
XC2V8000	208	23,296

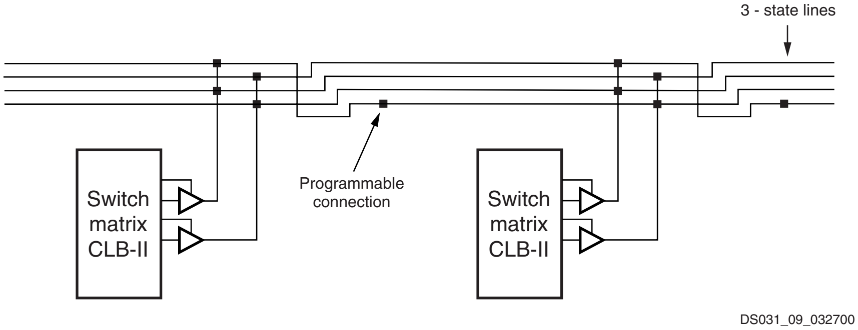


Figure 27: **3-State Buffer Connection to Horizontal Lines**

CLB/Slice Configurations

Table 12 summarizes the logic resources in one CLB. All of the CLBs are identical and each CLB or slice can be implemented in one of the configurations listed. Table 13 shows the available resources in all CLBs.

Table 12: **Logic Resources in One CLB**

Slices	LUTs	Flip-Flops	MULT_ANDs	Arithmetic & Carry-Chains	SOP Chains	Distributed SelectRAM	Shift Registers	TBUF
4	8	8	8	2	2	128 bits	128 bits	2

Table 13: **Virtex-II Logic Resources Available in All CLBs**

Device	CLB Array: Row x Column	Number of Slices	Number of LUTs	Max Distributed SelectRAM or Shift Register (bits)	Number of Flip-Flops	Number of Carry-Chains(1)	Number of SOP Chains(1)
XC2V40	8 x 8	256	516	8,192	516	16	16
XC2V80	16 x 8	512	1,024	16,384	1,024	16	32
XC2V250	24 x 16	1,536	3,072	49,152	3,072	32	48
XC2V500	32 x 24	3,072	6,144	98,304	6,144	48	64
XC2V1000	40 x 32	5,120	10,240	163,840	10,240	64	80
XC2V1500	48 x 40	7,680	15,360	245,760	15,360	80	96
XC2V2000	56 x 48	10,752	21,504	344,064	21,504	96	112
XC2V3000	64 x 56	14,336	28,672	458,752	28,672	112	128
XC2V4000	80 x 72	23,040	46,080	737,280	46,080	144	160
XC2V6000	96 x 88	33,792	67,584	1,081,344	67,584	176	192
XC2V8000	112 x 104	46,592	93,184	1,490,944	93,184	208	224

Notes:

1. The carry-chains and SOP chains can be split or cascaded.

18-Kbit Block SelectRAM Resources

Introduction

Virtex-II devices incorporate large amounts of 18-Kbit block SelectRAM. These complement the distributed SelectRAM resources that provide shallow RAM structures implemented in CLBs. Each Virtex-II block SelectRAM is an 18-Kbit true dual-port RAM with two independently clocked and independently controlled synchronous ports that access a common storage area. Both ports are functionally identical. CLK, EN, WE, and SSR polarities are defined through configuration.

Each port has the following types of inputs: Clock and Clock Enable, Write Enable, Set/Reset, and Address, as well as separate Data/parity data inputs (for write) and Data/parity data outputs (for read).

Operation is synchronous; the block SelectRAM behaves like a register. Control, address and data inputs must (and need only) be valid during the set-up time window prior to a rising (or falling, a configuration option) clock edge. Data outputs change as a result of the same clock edge.

Configuration

The Virtex-II block SelectRAM supports various configurations, including single- and dual-port RAM and various data/address aspect ratios. Supported memory configurations for single- and dual-port modes are shown in Table 14.

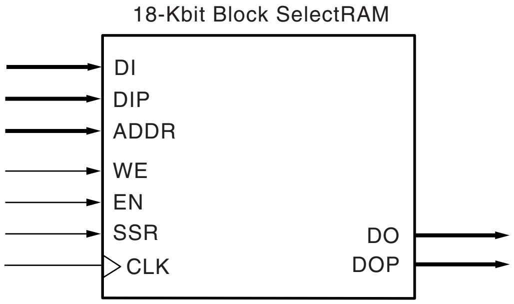
Table 14: **Dual- and Single-Port Configurations**

16K x 1 bit	2K x 9 bits
8K x 2 bits	1K x 18 bits
4K x 4 bits	512 x 36 bits

Single-Port Configuration

As a single-port RAM, the block SelectRAM has access to the 18-Kbit memory locations in any of the 2K x 9-bit, 1K x 18-bit, or 512 x 36-bit configurations and to 16-Kbit memory locations in any of the 16K x 1-bit, 8K x 2-bit, or 4K x 4-bit configurations. The advantage of the 9-bit, 18-bit and 36-bit widths is the ability to store a parity bit for each eight bits. Parity bits must be generated or checked externally in user logic. In such cases, the width is viewed as 8 + 1, 16 + 2, or 32 + 4. These extra parity bits are stored and behave exactly as the other bits, including the timing parameters. Video applications can use the 9-bit ratio of Virtex-II block SelectRAM memory to advantage.

Each block SelectRAM cell is a fully synchronous memory as illustrated in Figure 28. Input data bus and output data bus widths are identical.



DS031_10_102000

Figure 28: **18-Kbit Block SelectRAM Memory in Single-Port Mode**

Dual-Port Configuration

As a dual-port RAM, each port of block SelectRAM has access to a common 18-Kbit memory resource. These are fully synchronous ports with independent control signals for each port. The data widths of the two ports can be configured independently, providing built-in bus-width conversion.

Table 15 illustrates the different configurations available on ports A & B.

Table 15: **Dual-Port Mode Configurations**

Port A	16K x 1	16K x 1	16K x 1	16K x 1	16K x 1	16K x 1
Port B	16K x 1	8K x 2	4K x 4	2K x 9	1K x 18	512 x 36
Port A	8K x 2	8K x 2	8K x 2	8K x 2	8K x 2	
Port B	8K x 2	4K x 4	2K x 9	1K x 18	512 x 36	
Port A	4K x 4	4K x 4	4K x 4	4K x 4		
Port B	4K x 4	2K x 9	1K x 18	512 x 36		
Port A	2K x 9	2K x 9	2K x 9			
Port B	2K x 9	1K x 18	512 x 36			
Port A	1K x 18	1K x 18				
Port B	1K x 18	512 x 36				
Port A	512 x 36					
Port B	512 x 36					

If both ports are configured in either 2K x 9-bit, 1K x 18-bit, or 512 x 36-bit configurations, the 18-Kbit block is accessible from port A or B. If both ports are configured in either 16K x 1-bit, 8K x 2-bit. or 4K x 4-bit configurations, the 16 K-bit block is accessible from Port A or Port B. All other configurations result in one port having access to an 18-Kbit memory block and the other port having access to a 16 K-bit subset of the memory block equal to 16 Kbits.

Each block SelectRAM cell is a fully synchronous memory, as illustrated in Figure 29. The two ports have independent inputs and outputs and are independently clocked.

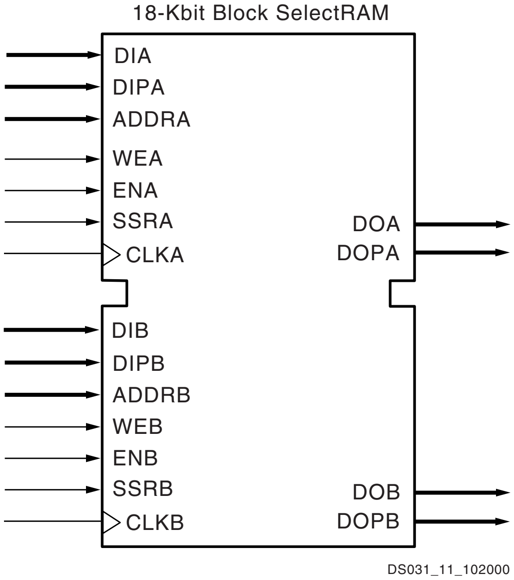


Figure 29: **18-Kbit Block SelectRAM in Dual-Port Mode**

Port Aspect Ratios

Table 16 shows the depth and the width aspect ratios for the 18-Kbit block SelectRAM. Virtex-II block SelectRAM also includes dedicated routing resources to provide an efficient interface with CLBs, block SelectRAM, and multipliers.

Table 16: **18-Kbit Block SelectRAM Port Aspect Ratio**

Width	Depth	Address Bus	Data Bus	Parity Bus
1	16,384	ADDR[13:0]	DATA[0]	N/A
2	8,192	ADDR[12:0]	DATA[1:0]	N/A
4	4,096	ADDR[11:0]	DATA[3:0]	N/A
9	2,048	ADDR[10:0]	DATA[7:0]	Parity[0]
18	1,024	ADDR[9:0]	DATA[15:0]	Parity[1:0]
36	512	ADDR[8:0]	DATA[31:0]	Parity[3:0]

Read/Write Operations

The Virtex-II block SelectRAM read operation is fully synchronous. An address is presented, and the read operation is enabled by control signals WEA and WEB in addition to ENA or ENB. Then, depending on clock polarity, a rising or falling clock edge causes the stored data to be loaded into output registers.

The write operation is also fully synchronous. Data and address are presented, and the write operation is enabled by control signals WEA or WEB in addition to ENA or ENB. Then, again depending on the clock input mode, a rising or falling clock edge causes the data to be loaded into the memory cell addressed.

A write operation performs a simultaneous read operation. Three different options are available, selected by configuration:

1. “WRITE_FIRST”

   The “WRITE_FIRST” option is a transparent mode. The same clock edge that writes the data input (DI) into the memory also transfers DI into the output registers DO as shown in Figure 30.

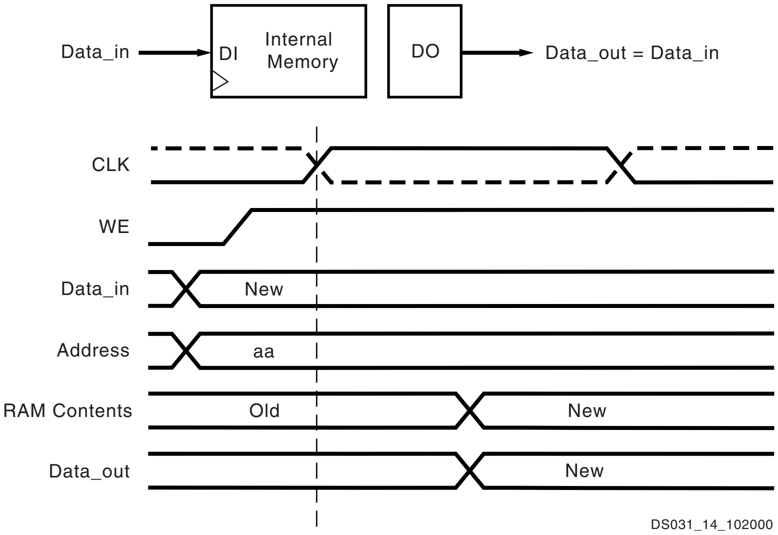


Figure 30: **WRITE_FIRST Mode**

2. “READ_FIRST”

The “READ_FIRST” option is a read-before-write mode.

The same clock edge that writes data input (DI) into the memory also transfers the prior content of the memory cell addressed into the data output registers DO, as shown in Figure 31.

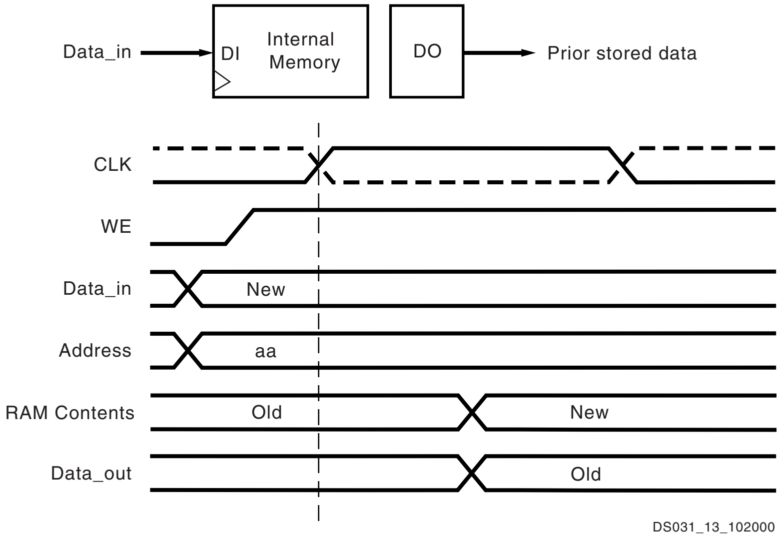


Figure 31: **READ_FIRST Mode**

3. “NO_CHANGE”

The “NO_CHANGE” option maintains the content of the output registers, regardless of the write operation. The clock edge during the write mode has no effect on the content of the data output register DO. When the port is configured as “NO_CHANGE”, only a read operation loads a new value in the output register DO, as shown in Figure 32.

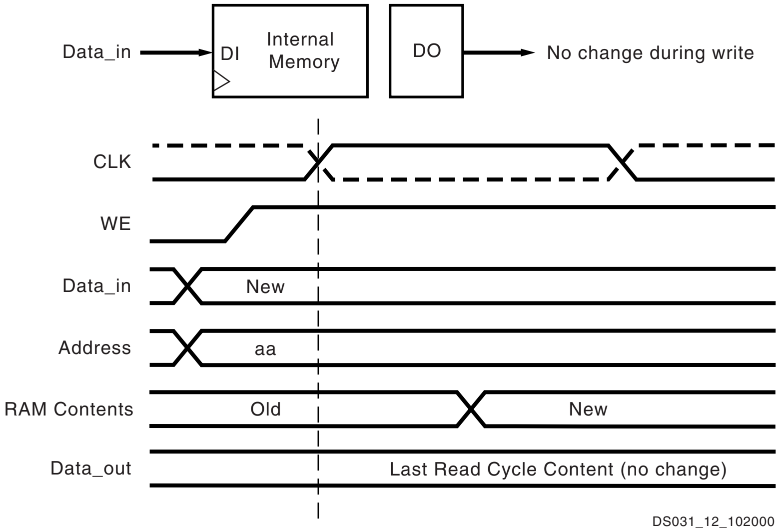


Figure 32: **NO_CHANGE Mode**

Control Pins and Attributes

Virtex-II SelectRAM memory has two independent ports with the control signals described in Table 17. All control inputs including the clock have an optional inversion.

Table 17: **Control Functions**

Control Signal	Function
CLK	Read and Write Clock
EN	Enable affects Read, Write, Set, Reset
WE	Write Enable
SSR	Set DO register to SRVAL (attribute)

Initial memory content is determined by the INIT_xx attributes. Separate attributes determine the output register value after device configuration (INIT) and SSR is asserted (SRVAL). Both attributes (INIT_B and SRVAL) are available for each port when a block SelectRAM resource is configured as dual-port RAM.

Locations

Virtex-II SelectRAM memory blocks are located in either four or six columns. The number of blocks per column depends of the device array size and is equivalent to the number of CLBs in a column divided by four. Column locations are shown in Table 18.

Table 18: **SelectRAM Memory Floor Plan**

Device	Columns	SelectRAM Blocks	
		Per Column	Total
XC2V40	2	2	4
XC2V80	2	4	8
XC2V250	4	6	24
XC2V500	4	8	32
XC2V1000	4	10	40
XC2V1500	4	12	48
XC2V2000	4	14	56
XC2V3000	6	16	96
XC2V4000	6	20	120
XC2V6000	6	24	144
XC2V8000	6	28	168

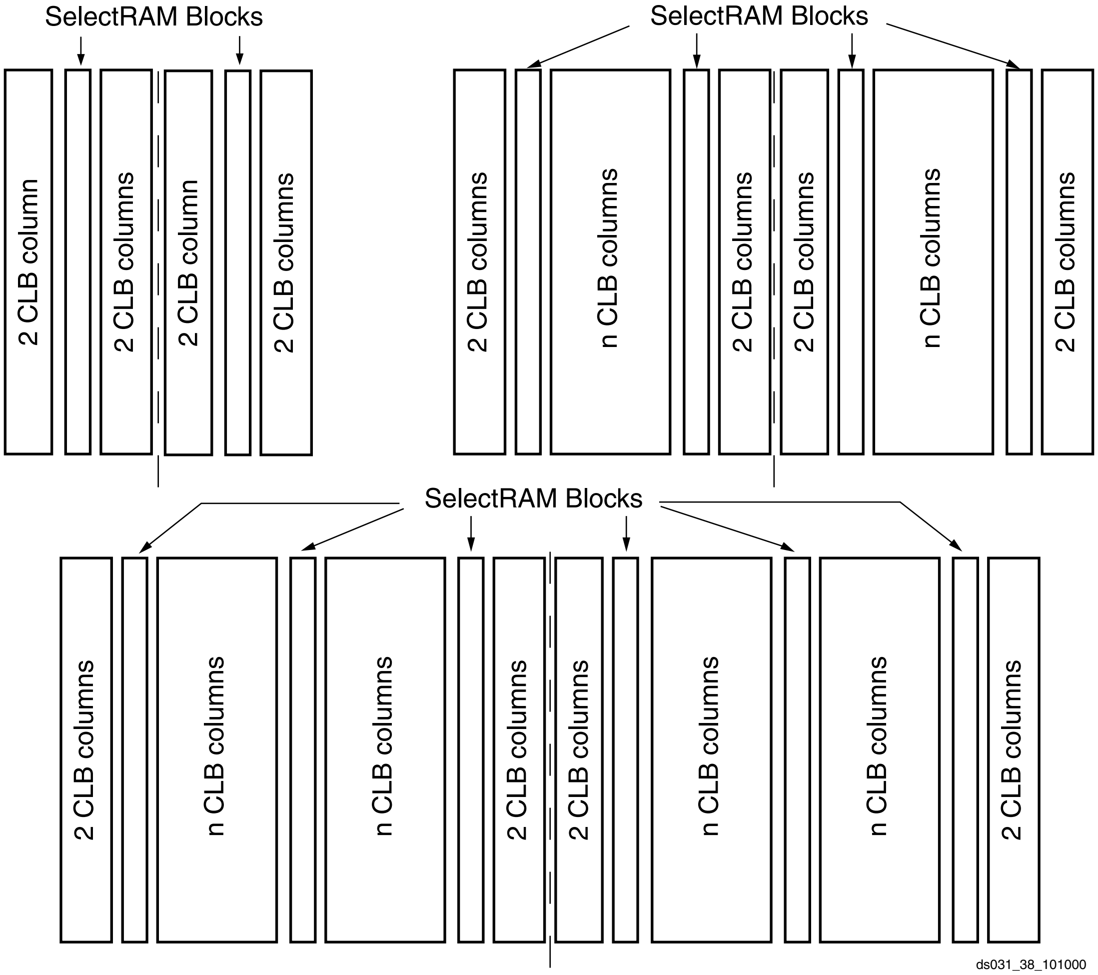


Figure 33: **Block SelectRAM (2-column, 4-column, and 6-column)**

Total Amount of SelectRAM Memory

Table 19 shows the amount of block SelectRAM memory available for each Virtex-II device. The 18-Kbit SelectRAM blocks are cascadable to implement deeper or wider single- or dual-port memory resources.

Table 19: **Virtex-II SelectRAM Memory Available**

Device	Total SelectRAM Memory		
	Blocks	**in Kbits**	**in Bits**
XC2V40	4	72	73,728
XC2V80	8	144	147,456
XC2V250	24	432	442,368
XC2V500	32	576	589,824
XC2V1000	40	720	737,280
XC2V1500	48	864	884,736
XC2V2000	56	1,008	1,032,192
XC2V3000	96	1,728	1,769,472
XC2V4000	120	2,160	2,211,840
XC2V6000	144	2,592	2,654,208
XC2V8000	168	3,024	3,096,576

18-Bit x 18-Bit Multipliers

Introduction

A Virtex-II multiplier block is an 18-bit by 18-bit 2's complement signed multiplier. Virtex-II devices incorporate many embedded multiplier blocks. These multipliers can be associated with an 18-Kbit block SelectRAM resource or can be used independently. They are optimized for high-speed operations and have a lower power consumption compared to an 18-bit x 18-bit multiplier in slices.

Each SelectRAM memory and multiplier block is tied to four switch matrices, as shown in Figure 34.

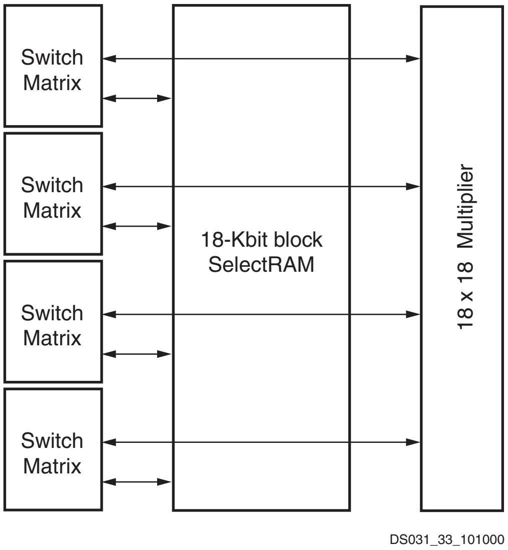


Figure 34: **SelectRAM and Multiplier Blocks**

Association With Block SelectRAM Memory

The interconnect is designed to allow SelectRAM memory and multiplier blocks to be used at the same time, but some interconnect is shared between the SelectRAM and the multiplier. Thus, SelectRAM memory can be used only up to 18 bits wide when the multiplier is used, because the multiplier shares inputs with the upper data bits of the SelectRAM memory.

This sharing of the interconnect is optimized for an 18-bit-wide block SelectRAM resource feeding the multiplier. The use of SelectRAM memory and the multiplier with an accumulator in LUTs allows for implementation of a digital signal processor (DSP) multiplier-accumulator (MAC) function, which is commonly used in finite and infinite impulse response (FIR and IIR) digital filters.

Configuration

The multiplier block is an 18-bit by 18-bit signed multiplier (2's complement). Both A and B are 18-bit-wide inputs, and the output is 36 bits. Figure 35 shows a multiplier block.

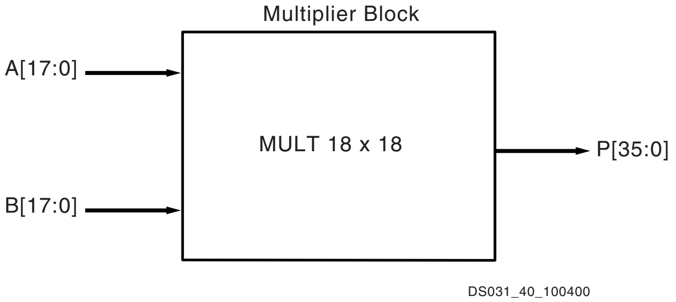


Figure 35: **Multiplier Block**

Locations / Organization

Multiplier organization is identical to the 18-Kbit SelectRAM organization, because each multiplier is associated with an 18-Kbit block SelectRAM resource.

In addition to the built-in multiplier blocks, the CLB elements have dedicated logic to implement efficient multipliers in logic. (Refer to **Configurable Logic Blocks (CLBs)**).

Table 20: **Multiplier Floor Plan**

Device	Columns	Multipliers	
		Per Column	**Total**
XC2V40	2	2	4
XC2V80	2	4	8
XC2V250	4	6	24
XC2V500	4	8	32
XC2V1000	4	10	40
XC2V1500	4	12	48
XC2V2000	4	14	56
XC2V3000	6	16	96
XC2V4000	6	20	120
XC2V6000	6	24	144
XC2V8000	6	28	168

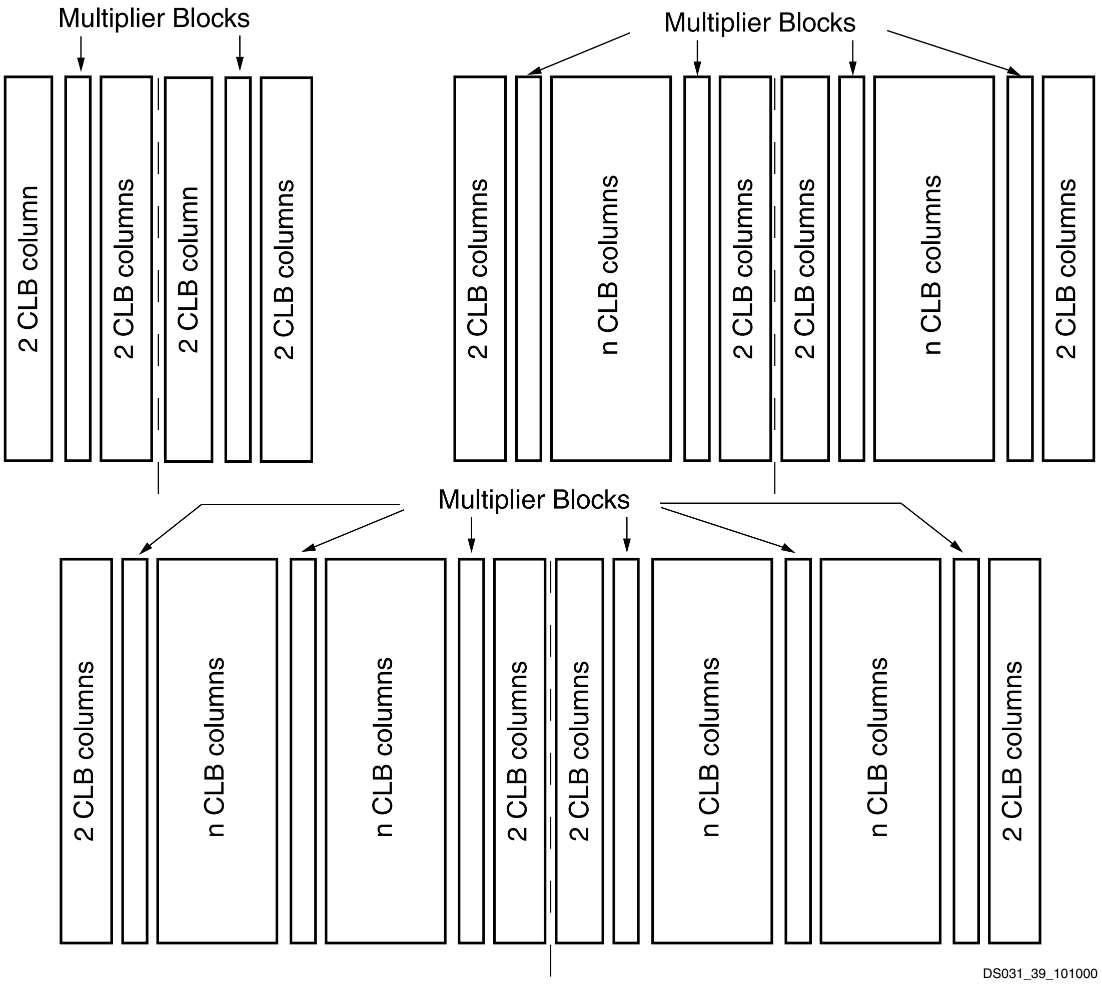


Figure 36: **Multipliers (2-column, 4-column, and 6-column)**

Global Clock Multiplexer Buffers

Virtex-II devices have 16 clock input pins that can also be used as regular user I/Os. Eight clock pads are on the top edge of the device, in the middle of the array, and eight are on the bottom edge, as illustrated in Figure 37.

The global clock multiplexer buffer represents the input to dedicated low-skew clock tree distribution in Virtex-II devices. Like the clock pads, eight global clock multiplexer buffers are on the top edge of the device and eight are on the bottom edge.

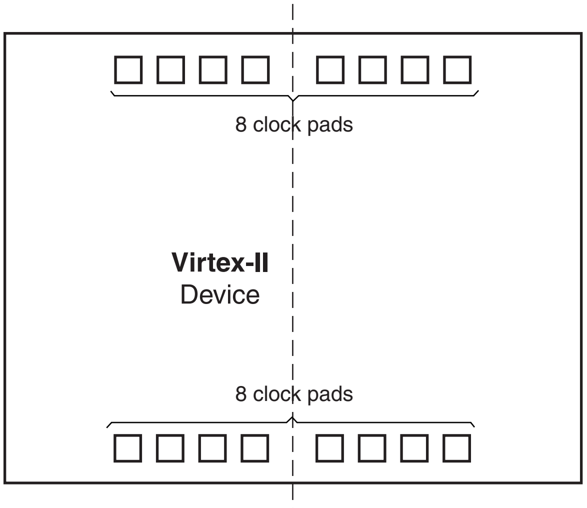


Figure 37: **Virtex-II Clock Pads**

Each global clock buffer can either be driven by the clock pad to distribute a clock directly to the device, or driven by the Digital Clock Manager (DCM), discussed in **Digital Clock Manager (DCM)**, page 74. Each global clock buffer can also be driven by local interconnects. The DCM has clock output(s) that can be connected to global clock buffer inputs, as shown in Figure 38.

Global clock buffers are used to distribute the clock to some or all synchronous logic elements (such as registers in CLBs and IOBs, and SelectRAM blocks.

Eight global clocks can be used in each quadrant of the Virtex-II device. Designers should consider the clock distribution detail of the device prior to pin-locking and floorplanning (see the *Virtex-II User Guide*).

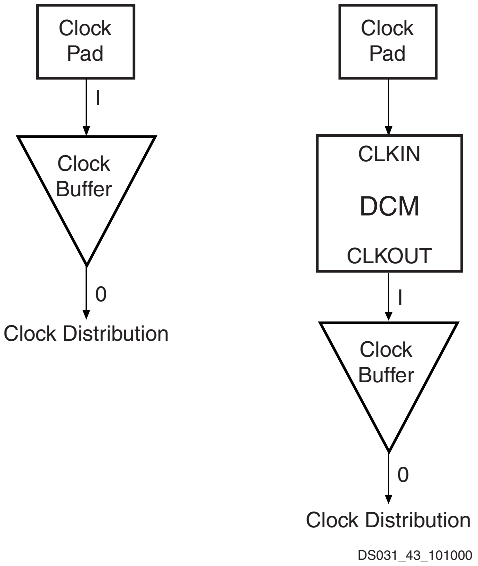


Figure 38: **Virtex-II Clock Distribution Configurations**

Figure 39 shows clock distribution in Virtex-II devices.

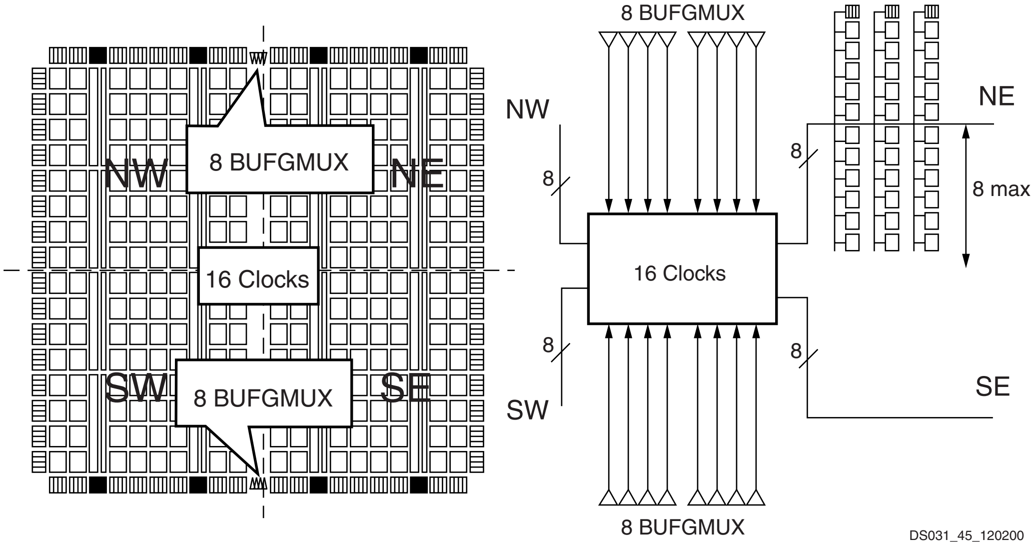


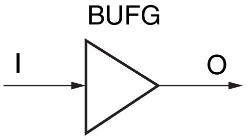
Figure 39: **Virtex-II Clock Distribution**

In each quadrant, up to eight clocks are organized in clock rows. A clock row supports up to 16 CLB rows (eight up and eight down). For the largest devices a new clock row is added, as necessary.

To reduce power consumption, any unused clock branches remain static.

Global clocks are driven by dedicated clock buffers (BUFG), which can also be used to gate the clock (BUFGCE) or to multiplex between two independent clock inputs (BUFGMUX).

The most common configuration option of this element is as a buffer. A BUFG function in this (global buffer) mode, is shown in Figure 40.



DS031_61_101200

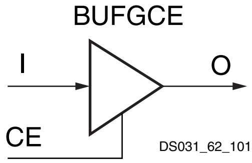
Figure 40: **Virtex-II BUFG Function**

The Virtex-II global clock buffer BUFG can also be configured as a clock enable/disable circuit (Figure 41), as well as a two-input clock multiplexer (Figure 42). A functional description of these two options is provided below. Each of them can be used in either of two modes, selected by configuration: rising clock edge or falling clock edge.

This section describes the rising clock edge option. For the opposite option, falling clock edge, just change all "rising" references to "falling" and all "High" references to "Low", except for the description of the CE or S levels. The rising clock edge option uses the BUFGCE and BUFGMUX primitives. The falling clock edge option uses the BUFGCE_1 and BUFGMUX_1 primitives.

BUFGCE

If the CE input is active (High) prior to the incoming rising clock edge, this Low-to-High-to-Low clock pulse passes through the clock buffer. Any level change of CE during the incoming clock High time has no effect.



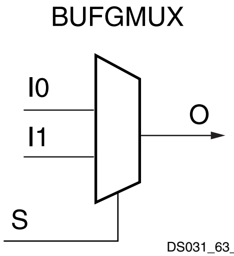
DS031_62_101200

Figure 41: **Virtex-II BUFGCE Function**

If the CE input is inactive (Low) prior to the incoming rising clock edge, the following clock pulse does not pass through the clock buffer, and the output stays Low. Any level change of CE during the incoming clock High time has no effect. CE must not change during a short setup window just prior to the rising clock edge on the BUFGCE input I. Violating this setup time requirement can result in an undefined runt pulse output.

BUFGMUX

BUFGMUX can switch between two unrelated, even asynchronous clocks. Basically, a Low on S selects the CLK0 input, a High on S selects the S1 input. Switching from one clock to the other is done in such a way that the output High and Low time is never shorter than the shortest High or Low time of either input clock. As long as the presently selected clock is High, any level change of S has no effect .



DS031_63_112900

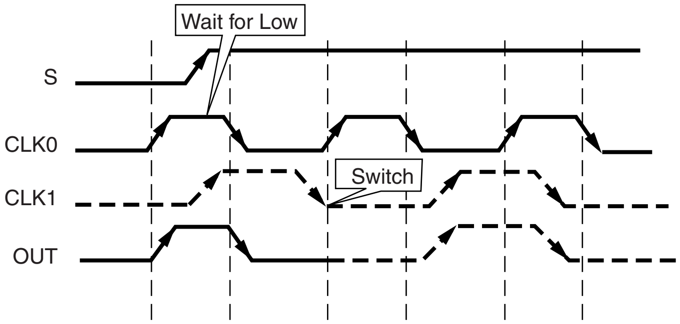
Figure 42: **Virtex-II BUFGMUX Function**

If the presently selected clock is Low while S changes, or if it goes Low after S has changed, the output is kept Low until the other ("to-be-selected") clock has made a transition from High to Low. At that instant, the new clock starts driving the output.

The two clock inputs can be asynchronous with regard to each other, and the S input can change at any time, except for a short setup time prior to the rising edge of the presently selected clock; that is, prior to the rising edge of the BUFGMUX output O. Violating this setup time requirement can result in an undefined runt pulse output.

All Virtex-II devices have 16 global clock multiplexer buffers.

Figure 43 shows a switchover from CLK0 to CLK1.



DS031_46_112900

Figure 43: **Clock Multiplexer Waveform Diagram**

- The current clock is CLK0.
- S is activated High.
- If CLK0 is currently High, the multiplexer waits for CLK0 to go Low.
- Once CLK0 is Low, the multiplexer output stays Low until CLK1 transitions High to Low.
- When CLK1 transitions from High to Low, the output switches to CLK1.
- No glitches or short pulses can appear on the output.

Digital Clock Manager (DCM)

The Virtex-II DCM offers a wide range of powerful clock management features.

- **Clock De-skew**: The DCM generates new system clocks (either internally or externally to the FPGA), which are phase-aligned to the input clock, thus eliminating clock distribution delays.
- **Frequency Synthesis**: The DCM generates a wide range of output clock frequencies, performing very flexible clock multiplication and division.
- **Phase Shifting**: The DCM provides both coarse phase shifting and fine-grained phase shifting with dynamic phase shift control.

The DCM utilizes fully digital delay lines allowing robust high-precision control of clock phase and frequency. It also utilizes fully digital feedback systems, operating dynamically to compensate for temperature and voltage variations during operation.

Up to four of the nine DCM clock outputs can drive inputs to global clock buffers or global clock multiplexer buffers simultaneously (see Figure 44). All DCM clock outputs can simultaneously drive general routing resources, including routes to output buffers.

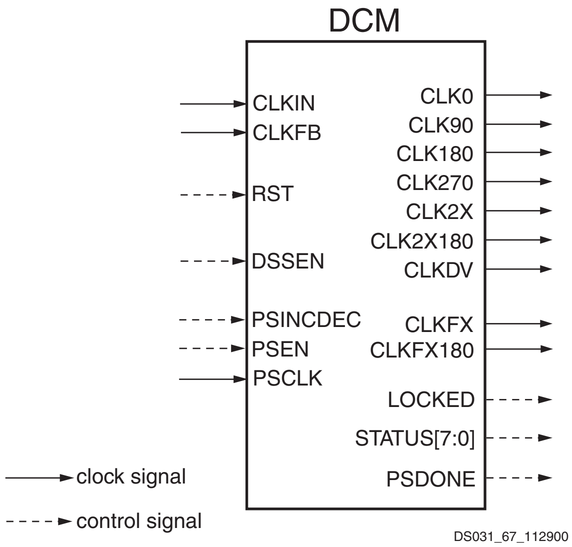


Figure 44: **Digital Clock Manager**

The DCM can be configured to delay the completion of the Virtex-II configuration process until after the DCM has achieved lock. This guarantees that the chip does not begin operating until after the system clocks generated by the DCM have stabilized.

The DCM has the following general control signals:

- RST input pin: resets the entire DCM
- LOCKED output pin: asserted High when all enabled DCM circuits have locked.
- STATUS output pins (active High): shown in Table 21.

Table 21: **DCM Status Pins**

Status Pin	Function
0	Phase Shift Overflow
1	CLKIN Stopped
2	CLKFX Stopped
3	N/A
4	N/A
5	N/A
6	N/A
7	N/A

Clock De-Skew

The DCM de-skews the output clocks relative to the input clock by automatically adjusting a digital delay line. Additional delay is introduced so that clock edges arrive at internal registers and block RAMs simultaneously with the clock edges arriving at the input clock pad. Alternatively, external clocks, which are also de-skewed relative to the input clock, can be generated for board-level routing. All DCM output clocks are phase-aligned to CLK0 and, therefore, are also phase-aligned to the input clock.

To achieve clock de-skew, the CLKFB input must be connected, and its source must be either CLK0 or CLK2X. Note that CLKFB must always be connected, unless only the CLKFX or CLKFX180 outputs are used and de-skew is not required.

Frequency Synthesis

The DCM provides flexible methods for generating new clock frequencies. Each method has a different operating frequency range and different AC characteristics. The CLK2X and CLK2X180 outputs double the clock frequency. The CLKDV output creates divided output clocks with division options of 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, and 16.

The CLKFX and CLKFX180 outputs can be used to produce clocks at the following frequency:

$$FREQ_{CLKFX} = (M/D) * FREQ_{CLKIN}$$

where M and D are two integers. Specifications for M and D are provided under **DCM Timing Parameters**. By default, M=4 and D=1, which results in a clock output frequency four times faster than the clock input frequency (CLKIN).

CLK2X180 is phase shifted 180 degrees relative to CLK2X. CLKFX180 is phase shifted 180 degrees relative to CLKFX. All frequency synthesis outputs automatically have 50/50 duty cycles (with the exception of the CLKDV output when performing a non-integer divide in high-frequency mode).

Note that CLK2X and CLK2X180 are not available in high-frequency mode.

Phase Shifting

The DCM provides additional control over clock skew through either coarse or fine-grained phase shifting. The CLK0, CLK90, CLK180, and CLK270 outputs are each phase shifted by ¼ of the input clock period relative to each other, providing coarse phase control. Note that CLK90 and CLK270 are not available in high-frequency mode.

Fine-phase adjustment affects all nine DCM output clocks. When activated, the phase shift between the rising edges of CLKIN and CLKFB is a specified fraction of the input clock period.

In variable mode, the PHASE_SHIFT value can also be dynamically incremented or decremented as determined by PSINCDEC synchronously to PSCLK, when the PSEN input is active. Figure 45 illustrates the effects of fine-phase shifting. For more information on DCM features, see the *Virtex-II User Guide*.

Table 22 lists fine-phase shifting control pins, when used in variable mode.

Table 22: **Fine-Phase Shifting Control Pins**

Control Pin	Direction	Function
PSINCDEC	in	Increment or decrement
PSEN	in	Enable ± phase shift
PSCLK	in	Clock for phase shift
PSDONE	out	Active when completed

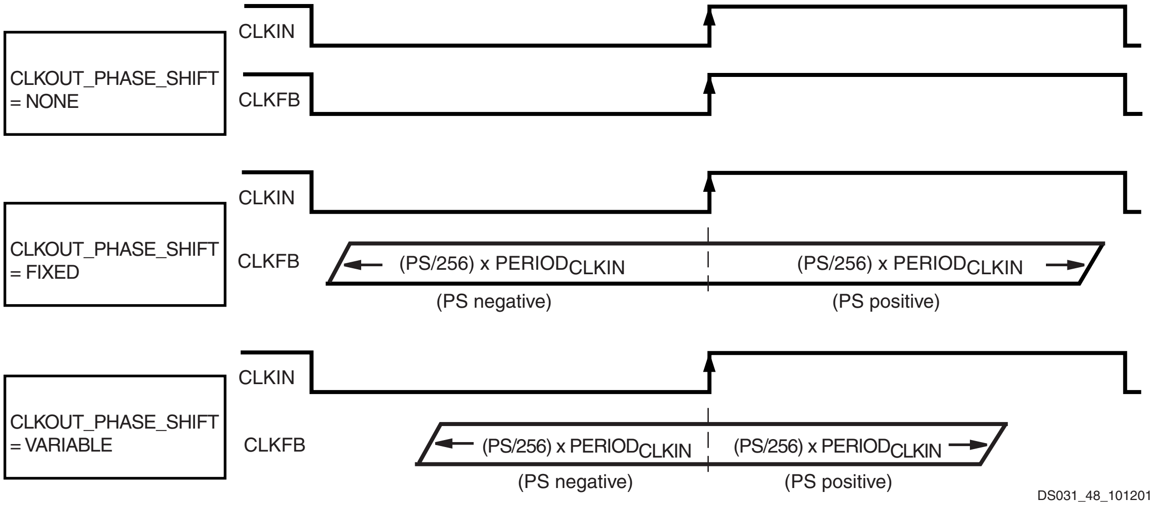


Figure 45: **Fine-Phase Shifting Effects**

Two separate components of the phase shift range must be understood:

- PHASE_SHIFT attribute range
- FINE_SHIFT_RANGE DCM timing parameter range

The PHASE_SHIFT attribute is the numerator in the following equation:

$$\text{Phase Shift (ns)} = (\text{PHASE\_SHIFT}/256) * \text{PERIOD}_{\text{CLKIN}}$$

The full range of this attribute is always -255 to +255, but its practical range varies with CLKIN frequency, as constrained by the FINE_SHIFT_RANGE component, which represents the total delay achievable by the phase shift delay line. Total delay is a function of the number of delay taps used in the circuit. Across process, voltage, and temperature, this absolute range is guaranteed to be as specified under **DCM Timing Parameters**.

Absolute range (fixed mode) = ± FINE_SHIFT_RANGE

Absolute range (variable mode) = ± FINE_SHIFT_RANGE/2

The reason for the difference between fixed and variable modes is as follows. For variable mode to allow symmetric, dynamic sweeps from -255/256 to +255/256, the DCM sets the "zero phase skew" point as the middle of the delay line, thus dividing the total delay line range in half. In fixed mode, since the PHASE_SHIFT value never changes after configuration, the entire delay line is available for insertion into either the CLKIN or CLKFB path (to create either positive or negative skew).

Taking both of these components into consideration, the following are some usage examples:

- If $\text{PERIOD}_{\text{CLKIN}}$ = 2 * FINE_SHIFT_RANGE, then PHASE_SHIFT in fixed mode is limited to ± 128, and in variable mode it is limited to ± 64.
- If $\text{PERIOD}_{\text{CLKIN}}$ = FINE_SHIFT_RANGE, then PHASE_SHIFT in fixed mode is limited to ± 255, and in variable mode it is limited to ± 128.
- If $\text{PERIOD}_{\text{CLKIN}} \leq 0.5$ * FINE_SHIFT_RANGE, then PHASE_SHIFT is limited to ± 255 in either mode.

Operating Modes

The frequency ranges of DCM input and output clocks depend on the operating mode specified, either low-frequency mode or high-frequency mode, according to Table 23. (For actual values, see **Virtex-II Switching Characteristics**). The CLK2X, CLK2X180, CLK90, and CLK270 outputs are not available in high-frequency mode.

High or low-frequency mode is selected by an attribute.

Table 23: **DCM Frequency Ranges**

Output Clock	Low-Frequency Mode		High-Frequency Mode	
	CLKIN Input	**CLK Output**	**CLKIN Input**	**CLK Output**
CLK0, CLK180	CLKIN_FREQ_DLL_LF	CLKOUT_FREQ_1X_LF	CLKIN_FREQ_DLL_HF	CLKOUT_FREQ_1X_HF
CLK90, CLK270	CLKIN_FREQ_DLL_LF	CLKOUT_FREQ_1X_LF	NA	NA
CLK2X, CLK2X180	CLKIN_FREQ_DLL_LF	CLKOUT_FREQ_2X_LF	NA	NA
CLKDV	CLKIN_FREQ_DLL_LF	CLKOUT_FREQ_DV_LF	CLKIN_FREQ_DLL_HF	CLKOUT_FREQ_DV_HF
CLKFX, CLKFX180	CLKIN_FREQ_FX_LF	CLKOUT_FREQ_FX_LF	CLKIN_FREQ_FX_HF	CLKOUT_FREQ_FX_HF

Locations/Organization

Virtex-II DCMs are placed on the top and bottom of each block RAM and multiplier column. The number of DCMs depends on the device size, as shown in Table 24.

Table 24: **DCM Organization**

Device	Columns	DCMs
XC2V40	2	4
XC2V80	2	4
XC2V250	4	8
XC2V500	4	8
XC2V1000	4	8
XC2V1500	4	8
XC2V2000	4	8
XC2V3000	6	12
XC2V4000	6	12
XC2V6000	6	12
XC2V8000	6	12

Active Interconnect Technology

Local and global Virtex-II routing resources are optimized for speed and timing predictability, as well as to facilitate IP cores implementation. Virtex-II Active Interconnect Technology is a fully buffered programmable routing matrix. All routing resources are segmented to offer the advantages of a hierarchical solution. Virtex-II logic features like CLBs, IOBs, block RAM, multipliers, and DCMs are all connected to an identical switch matrix for access to global routing resources, as shown in Figure 46.

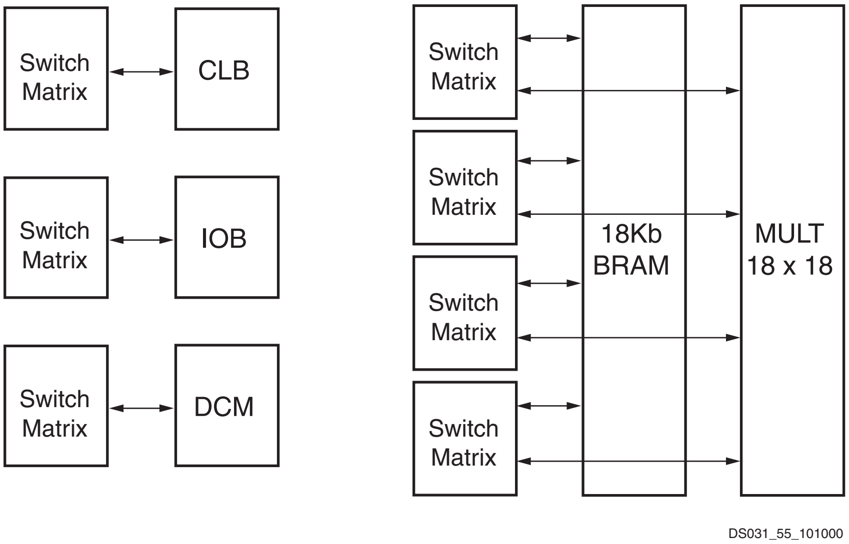


Figure 46: **Active Interconnect Technology**

Each Virtex-II device can be represented as an array of switch matrixes with logic blocks attached, as illustrated in Figure 47.

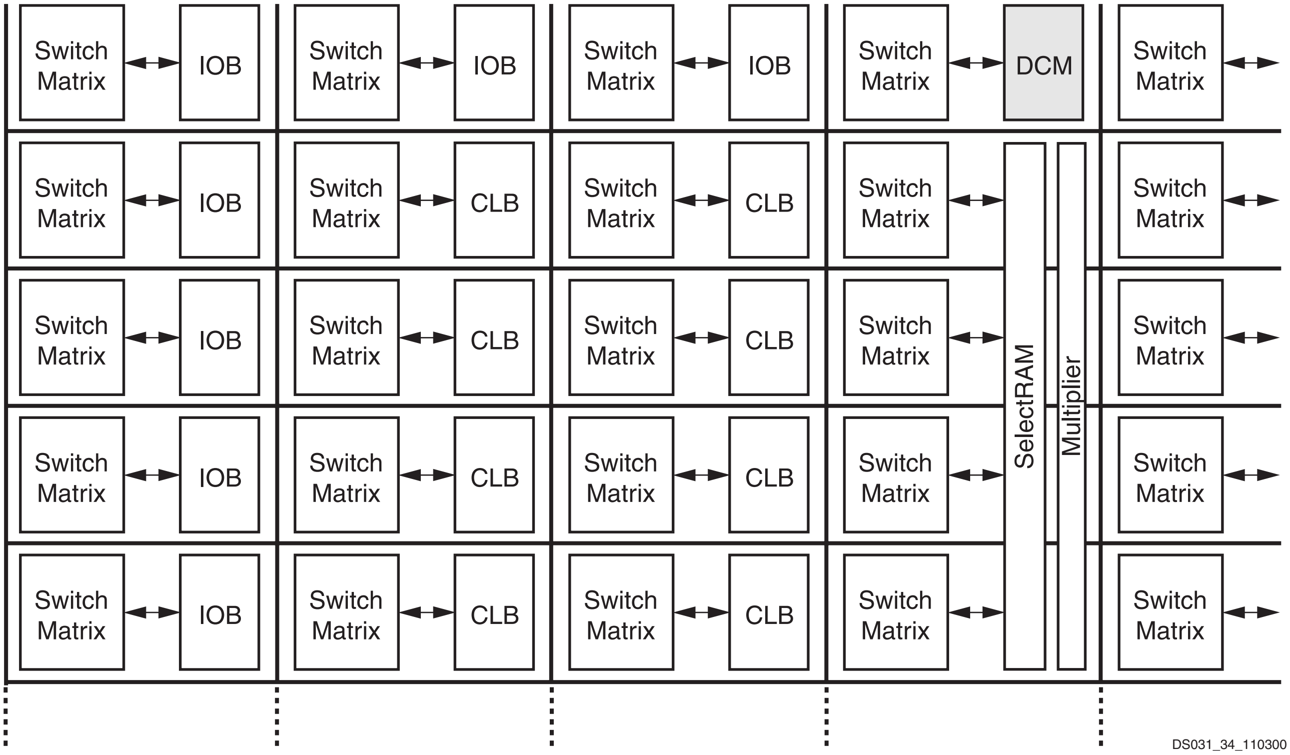


Figure 47: **Routing Resources**

Place-and-route software takes advantage of this regular array to deliver optimum system performance and fast compile times. The segmented routing resources are essential to guarantee IP cores portability and to efficiently handle an incremental design flow that is based on modular implementations. Total design time is reduced due to fewer and shorter design iterations.

Hierarchical Routing Resources

Most Virtex-II signals are routed using the global routing resources, which are located in horizontal and vertical routing channels between each switch matrix.

As shown in Figure 48, Virtex-II has fully buffered programmable interconnections, with a number of resources counted between any two adjacent switch matrix rows or columns. Fanout has minimal impact on the performance of each net.

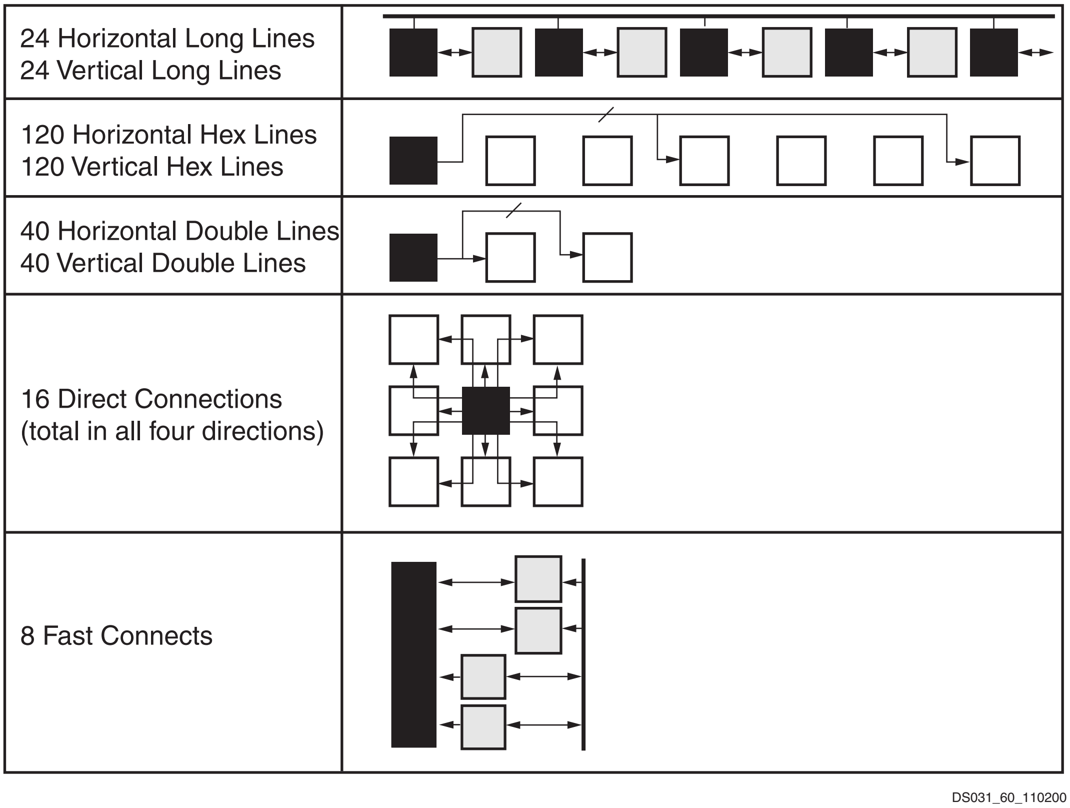


Figure 48: **Hierarchical Routing Resources**

- The long lines are bidirectional wires that distribute signals across the device. Vertical and horizontal long lines span the full height and width of the device.
- The hex lines route signals to every third or sixth block away in all four directions. Organized in a staggered pattern, hex lines can only be driven from one end. Hex-line signals can be accessed either at the endpoints or at the midpoint (three blocks from the source).
- The double lines route signals to every first or second block away in all four directions. Organized in a staggered pattern, double lines can be driven only at their endpoints. Double-line signals can be accessed either at the endpoints or at the midpoint (one block from the source).
- The direct connect lines route signals to neighboring blocks: vertically, horizontally, and diagonally.
- The fast connect lines are the internal CLB local interconnections from LUT outputs to LUT inputs.

Dedicated Routing

In addition to the global and local routing resources, dedicated signals are available.

- There are eight global clock nets per quadrant (see **Global Clock Multiplexer Buffers**).
- Horizontal routing resources are provided for on-chip 3-state busses. Four partitionable bus lines are provided per CLB row, permitting multiple busses within a row. (See **3-State Buffers**.)
- Two dedicated carry-chain resources per slice column (two per CLB column) propagate carry-chain MUXCY output signals vertically to the adjacent slice. (See **CLB/Slice Configurations**.)

- One dedicated SOP chain per slice row (two per CLB row) propagate ORCY output logic signals horizontally to the adjacent slice. (See **Sum of Products**.)
- One dedicated shift-chain per CLB connects the output of LUTs in shift-register mode to the input of the next LUT in shift-register mode (vertically) inside the CLB. (See **Shift Registers**, page 60.)

Creating a Design

Creating Virtex-II designs is easy with Xilinx Integrated Synthesis Environment (ISE) development systems, which support advanced design capabilities, including ProActive Timing Closure, integrated logic analysis, and the fastest place and route runtimes in the industry. ISE solutions enable designers to get the performance they need, quickly and easily.

As a result of the ongoing cooperative development efforts between Xilinx and EDA Alliance partners, designers can take advantage of the benefits provided by EDA technologies in the programmable logic design process. Xilinx development systems are available in a number of easy to use configurations, collectively known as the ISE Series.

ISE Alliance

The ISE Alliance solution is designed to plug and play within an existing design environment. Built using industry standard data formats and netlists, these stable, flexible products enable Alliance EDA partners to deliver their best design automation capabilities to Xilinx customers, along with the time to market benefits of ProActive Timing Closure.

ISE Foundation

The ISE Foundation solution delivers the benefits of true HDL-based design in a seamlessly integrated design environment. An intuitive project navigator, as well as powerful HDL design and two HDL synthesis tools, ensure that high-quality results are achieved quickly and easily. The ISE Foundation product includes:

- State Diagram entry using Xilinx StateCAD
- Automatic HDL Testbench generation using Xilinx HDLBencher
- HDL Simulation using ModelSim XE

Design Flow

Virtex-II design flow proceeds as follows:

- Design Entry
- Synthesis
- Implementation
- Verification

Most programmable logic designers iterate through these steps several times in the process of completing a design.

Design Entry

All Xilinx ISE development systems support the mainstream EDA design entry capabilities, ranging from schematic design to advanced HDL design methodologies. Given the high densities of the Virtex-II family, designs are created most efficiently using HDLs. To further improve their time to market, many Xilinx customers employ incremental, modular, and Intellectual Property (IP) design techniques. When properly used, these techniques further accelerate the logic design process.

To enable designers to leverage existing investments in EDA tools, and to ensure high performance design flows, Xilinx jointly develops tools with leading EDA vendors, including:

- Aldec®
- Cadence®
- Exemplar®
- Mentor Graphics®
- Model Technology®
- Synopsys®
- Synplicity®

Complete information on Alliance Series partners and their associated design flows is available at www.xilinx.com on the Xilinx Alliance Series web page.

The ISE Foundation product offers schematic entry and HDL design capabilities as part of an integrated design solution - enabling one-stop shopping. These capabilities are powerful, easy to use, and they support the full portfolio of Xilinx programmable logic devices. HDL design capabilities include a color-coded HDL editor with integrated language templates, state diagram entry, and Core generation capabilities.

Synthesis

The ISE Alliance product is engineered to support advanced design flows with the industry's best synthesis tools. Advanced design methodologies include:

- Physical Synthesis
- Incremental synthesis
- RTL floorplanning
- Direct physical mapping

The ISE Foundation product seamlessly integrates synthesis capabilities purchased directly from Exemplar, Synopsys, and Synplicity. In addition, it includes the capabilities of Xilinx Synthesis Technology.

A benefit of having two seamlessly integrated synthesis engines within an ISE design flow is the ability to apply alternative sets of optimization techniques on designs, helping to ensure that designers can meet even the toughest timing requirements.

Design Implementation

The ISE Series development systems include Xilinx timing-driven implementation tools, frequently called “place and route” or “fitting” software. This robust suite of tools enables the creation of an intuitive, flexible, tightly integrated design flow that efficiently bridges “logical” and “physical” design domains. This simplifies the task of defining a design, including its behavior, timing requirements, and optional layout (or floorplanning), as well as simplifying the task of analyzing reports generated during the implementation process.

The Virtex-II implementation process is comprised of Synthesis, translation, mapping, place and route, and configuration file generation. While the tools can be run individually, many designers choose to run the entire implementation process with the click of a button. To assist those who prefer to script their design flows, Xilinx provides Xflow, an automated single command line process.

Design Verification

In addition to conventional design verification using static timing analysis or simulation techniques, Xilinx offers powerful in-circuit debugging techniques using ChipScope ILA (Integrated Logic Analysis). The reconfigurable nature of Xilinx FPGAs means that designs can be verified in real time without the need for extensive sets of software simulation vectors.

For simulation, the system extracts post-layout timing information from the design database, and back-annotates this information into the netlist for use by the simulator. The back annotation features a variety of patented Xilinx techniques, resulting in the industry’s most powerful simulation flows. Alternatively, timing-critical portions of a design can be verified using the Xilinx static timing analyzer or a third party static timing analysis tool like Synopsys Prime Time™, by exporting timing data in the STAMP data format.

For in-circuit debugging, ChipScope ILA enables designers to analyze the real-time behavior of a device while operating at full system speeds. Logic analysis commands and captured data are transferred between the ChipScope software and ILA cores within the Virtex-II FPGA, using industry standard JTAG protocols. These JTAG transactions are driven over an optional download cable (MultiLINX or JTAG), connecting the Virtex device in the target system to a PC or workstation.

ChipScope ILA was designed to look and feel like a logic analyzer, making it easy to begin debugging a design immediately. Modifications to the desired logic analysis can be downloaded directly into the system in a matter of minutes.

Other Unique Features of Virtex-II Design Flow

Xilinx design flows feature a number of unique capabilities. Among these are efficient incremental HDL design flows; a robust capability that is enabled by Xilinx exclusive hierarchical floorplanning capabilities. Another powerful design capability only available in the Xilinx design flow is “Modular Design”, part of the Xilinx suite of team design tools, which enables autonomous design, implementation, and verification of design modules.

Incremental Synthesis

Xilinx unique hierarchical floorplanning capabilities enable designers to create a programmable logic design by isolating design changes within one hierarchical “logic block”, and perform synthesis, verification and implementation processes on that specific logic block. By preserving the logic in unchanged portions of a design, Xilinx incremental design makes the high-density design process more efficient.

Xilinx hierarchical floorplanning capabilities can be specified using the high-level floorplanner or a preferred RTL floorplanner (see the Xilinx web site for a list of supported EDA partners). When used in conjunction with one of the EDA partners’ floorplanners, higher performance results can be achieved, as many synthesis tools use this more predictable detailed physical implementation information to establish more aggressive and accurate timing estimates when performing their logic optimizations.

Modular Design

Xilinx innovative modular design capabilities take the incremental design process one step further by enabling the designer to delegate responsibility for completing the design, synthesis, verification, and implementation of a hierarchical “logic block” to an arbitrary number of designers - assigning a specific region within the target FPGA for exclusive use by each of the team members.

This team design capability enables an autonomous approach to design modules, changing the hand-off point to the lead designer or integrator from “my module works in simulation” to “my module works in the FPGA”. This unique design methodology also leverages the Xilinx hierarchical floorplanning capabilities and enables the Xilinx (or EDA partner) floorplanner to manage the efficient implementation of very high-density FPGAs.

Configuration

Virtex-II devices are configured by loading application specific configuration data into the internal configuration memory. Configuration is carried out using a subset of the device pins, some of which are dedicated, while others can be re-used as general purpose inputs and outputs once configuration is complete.

Depending on the system design, several configuration modes are supported, selectable via mode pins. The mode pins M2, M1 and M0 are dedicated pins. An additional pin, HSWAP_EN is used in conjunction with the mode pins to select whether user I/O pins have pull-ups during configuration. By default, HSWAP_EN is tied High (internal pull-up) which shuts off the pull-ups on the user I/O pins during configuration. When HSWAP_EN is tied Low, user I/Os have pull-ups during configuration. Other dedicated pins are CCLK (the configuration clock pin), DONE, PROG_B, and the boundary-scan pins: TDI, TDO, TMS, and TCK. Depending on the configuration mode chosen, CCLK can be an output generated by the FPGA, or an input accepting an externally generated clock. The configuration pins and boundary scan pins are independent of the V_{CCO}. The auxiliary power supply (V_{CCAUX}) of 3.3 V is used for these pins. All configuration pins are LVTTL 12 mA. (See **Virtex-II DC Characteristics**.)

A persist option is available which can be used to force the configuration pins to retain their configuration function even after device configuration is complete. If the persist option is not selected then the configuration pins with the exception of CCLK, PROG_B, and DONE can be used as user I/O in normal operation. The persist option does not apply to the boundary-scan related pins. The persist feature is valuable in applications which employ partial reconfiguration or reconfiguration on the fly.

Configuration Modes

Virtex-II supports the following five configuration modes:

- Slave-serial mode
- Master-serial mode
- Slave SelectMAP mode
- Master SelectMAP mode
- Boundary-Scan mode (IEEE 1532/IEEE 1149)

A detailed description of configuration modes is provided in the *Virtex-II User Guide*.

Slave-Serial Mode

In slave-serial mode, the FPGA receives configuration data in bit-serial form from a serial PROM or other serial source of configuration data. The CCLK pin on the FPGA is an input in this mode. The serial bitstream must be setup at the DIN input pin a short time before each rising edge of the externally generated CCLK.

Multiple FPGAs can be daisy-chained for configuration from a single source. After a particular FPGA has been configured, the data for the next device is routed internally to the DOUT pin. The data on the DOUT pin changes on the rising edge of CCLK.

Slave-serial mode is selected by applying <111> to the mode pins (M2, M1, M0). A weak pull-up on the mode pins makes slave serial the default mode if the pins are left unconnected.

Master-Serial Mode

In master-serial mode, the CCLK pin is an output pin. It is the Virtex-II FPGA device that drives the configuration clock on the CCLK pin to a Xilinx Serial PROM which in turn feeds bit-serial data to the DIN input. The FPGA accepts this data on each rising CCLK edge. After the FPGA has been loaded, the data for the next device in a daisy-chain is presented on the DOUT pin after the rising CCLK edge.

The interface is identical to slave serial except that an internal oscillator is used to generate the configuration clock (CCLK). A wide range of frequencies can be selected for CCLK which always starts at a slow default frequency. Configuration bits then switch CCLK to a higher frequency for the remainder of the configuration.

Slave SelectMAP Mode

The SelectMAP mode is the fastest configuration option. Byte-wide data is written into the Virtex-II FPGA device with a BUSY flag controlling the flow of data. An external data source provides a byte stream, CCLK, an active Low Chip Select (CS_B) signal and a Write signal (RDWR_B). If BUSY is asserted (High) by the FPGA, the data must be held until BUSY goes Low. Data can also be read using the SelectMAP mode. If RDWR_B is asserted, configuration data is read out of the FPGA as part of a readback operation.

After configuration, the pins of the SelectMAP port can be used as additional user I/O. Alternatively, the port can be retained to permit high-speed 8-bit readback using the persist option.

Multiple Virtex-II FPGAs can be configured using the SelectMAP mode, and be made to start-up simultaneously. To configure multiple devices in this way, wire the individual CCLK, Data, RDWR_B, and BUSY pins of all the devices in parallel. The individual devices are loaded separately by deasserting the CS_B pin of each device in turn and writing the appropriate data.

Master SelectMAP Mode

This mode is a master version of the SelectMAP mode. The device is configured byte-wide on a CCLK supplied by the Virtex-II FPGA device. Timing is similar to the Slave SerialMAP mode except that CCLK is supplied by the Virtex-II FPGA.

Boundary-Scan (JTAG, IEEE 1532) Mode

In boundary-scan mode, dedicated pins are used for configuring the Virtex-II device. The configuration is done entirely through the IEEE 1149.1 Test Access Port (TAP). Virtex-II device configuration using Boundary scan is compliant with IEEE 1149.1-1993 standard and the new IEEE 1532 standard for In-System Configurable (ISC) devices. The IEEE 1532 standard is backward compliant with the IEEE 1149.1-1993 TAP and state machine. The IEEE Standard 1532 for In-System Configurable (ISC) devices is intended to be programmed, reprogrammed, or tested on the board via a physical and logical protocol.

Configuration through the boundary-scan port is always available, independent of the mode selection. Selecting the boundary-scan mode simply turns off the other modes.

Table 25: **Virtex-II Configuration Mode Pin Settings**

Configuration Mode[1]	M2	M1	M0	CCLK Direction	Data Width	Serial D_{OUT}[2]
Master Serial	0	0	0	Out	1	Yes
Slave Serial	1	1	1	In	1	Yes
Master SelectMAP	0	1	1	Out	8	No
Slave SelectMAP	1	1	0	In	8	No
Boundary Scan	1	0	1	N/A	1	No

Notes:

1. The HSWAP_EN pin controls the pullups. Setting M2, M1, and M0 selects the configuration mode, while the HSWAP_EN pin controls whether or not the pullups are used.
2. Daisy chaining is possible only in modes where Serial D_{OUT} is used. For example, in SelectMAP modes, the first device does NOT support daisy chaining of downstream devices.

Table 26 lists the total number of bits required to configure each device.

Table 26: **Virtex-II Bitstream Lengths**

Device	# of Configuration Bits
XC2V40	360,096
XC2V80	635,296
XC2V250	1,697,184
XC2V500	2,761,888
XC2V1000	4,082,592
XC2V1500	5,659,296
XC2V2000	7,492,000
XC2V3000	10,494,368
XC2V4000	15,659,936
XC2V6000	21,849,504
XC2V8000	29,063,072

Configuration Sequence

The configuration of Virtex-II devices is a three-phase process. First, the configuration memory is cleared. Next, configuration data is loaded into the memory, and finally, the logic is activated by a start-up process.

Configuration is automatically initiated on power-up unless it is delayed by the user. The INIT_B pin can be held Low using an open-drain driver. An open-drain is required since INIT_B is a bidirectional open-drain pin that is held Low by a Virtex-II FPGA device while the configuration memory is being cleared. Extending the time that the pin is Low causes the configuration sequencer to wait. Thus, configuration is delayed by preventing entry into the phase where data is loaded.

The configuration process can also be initiated by asserting the PROG_B pin. The end of the memory-clearing phase is signaled by the INIT_B pin going High, and the completion of the entire process is signaled by the DONE pin going High. The Global Set/Reset (GSR) signal is pulsed after the last frame of configuration data is written but before the start-up sequence. The GSR signal resets all flip-flops on the device.

The default start-up sequence is that one CCLK cycle after DONE goes High, the global 3-state signal (GTS) is released. This permits device outputs to turn on as necessary. One CCLK cycle later, the Global Write Enable (GWE) signal is released. This permits the internal storage elements to begin changing state in response to the logic and the user clock.

The relative timing of these events can be changed via configuration options in software. In addition, the GTS and GWE events can be made dependent on the DONE pins of multiple devices all going High, forcing the devices to start synchronously. The sequence can also be paused at any stage, until lock has been achieved on any or all DCMs, as well as the DCI.

Readback

In this mode, configuration data from the Virtex-II FPGA device can be read back. Readback is supported only in the SelectMAP (master and slave) and Boundary Scan mode.

Along with the configuration data, it is possible to read back the contents of all registers, distributed SelectRAM, and block RAM resources. This capability is used for real-time debugging. For more detailed configuration information, see the *Virtex-II User Guide.*

Bitstream Encryption

Virtex-II devices have an on-chip decryptor using one or two sets of three keys for triple-key Data Encryption Standard (DES) operation. Xilinx software tools offer an optional encryption of the configuration data (bitstream) with a triple-key DES determined by the designer.

The keys are stored in the FPGA by JTAG instruction and retained by a battery connected to the V_{BATT} pin, when the device is not powered. Virtex-II devices can be configured with the corresponding encrypted bitstream, using any of the configuration modes described previously.

A detailed description of how to use bitstream encryption is provided in the *Virtex-II User Guide*. Your local FAE can also provide specific information on this feature.

Partial Reconfiguration

Partial reconfiguration of Virtex-II devices can be accomplished in either Slave SelectMAP mode or Boundary-Scan mode. Instead of resetting the chip and doing a full configuration, new data is loaded into a specified area of the chip, while the rest of the chip remains in operation. Data is loaded on a column basis, with the smallest load unit being a configuration “frame” of the bitstream (device size dependent).

Partial reconfiguration is useful for applications that require different designs to be loaded into the same area of a chip, or that require the ability to change portions of a design without having to reset or reconfigure the entire chip.

Revision History

This section records the change history for this module of the data sheet.

Date	Version	Revision
11/07/00	1.0	Early access draft.
12/06/00	1.1	Initial release.
01/15/01	1.2	Added values to the tables in the **Virtex-II Performance Characteristics** and **Virtex-II Switching Characteristics** sections.
01/25/01	1.3	The data sheet was divided into four modules (per the current style standard). A note was added to Table 1.
04/02/01	1.5	• Under **Input/Output Individual Options**, the range of values for optional pull-up and pull-down resistors was changed to 10 - 60 KΩ from 50 - 100 KΩ. • Skipped v1.4 to sync up modules. Reverted to traditional double-column format.
07/30/01	1.6	• Added Table 6. • Changed definition of multiply and divide integer ranges under **Digital Clock Manager (DCM)**. • Made numerous minor edits throughout this module.
10/02/01	1.7	• Updated descriptions under **Digitally Controlled Impedance (DCI)**, **Global Clock Multiplexer Buffers**, **Digital Clock Manager (DCM)**, and **Creating a Design**.
10/12/01	1.8	• Made clarifying edits under **Digital Clock Manager (DCM)**.
11/29/01	1.9	• Changed bitstream lengths for each device in Table 26.

Virtex-II Data Sheet

The Virtex-II Data Sheet contains the following modules:

- DS031-1, Virtex-II 1.5V FPGAs: **Introduction and Ordering Information (Module 1)**
- DS031-2, Virtex-II 1.5V FPGAs: Functional Description (Module 2)
- DS031-3, Virtex-II 1.5V FPGAs: **DC and Switching Characteristics (Module 3)**
- DS031-4, Virtex-II 1.5V FPGAs: **Pinout Tables (Module 4)**

Virtex-II 1.5V
Field-Programmable Gate Arrays

DS031-3 (v2.0) November 28, 2001

Advance Product Specification

Virtex™-II Electrical Characteristics

Virtex-II devices are provided in –4, –5, and –6 speed grades, with –6 having the highest performance.

Virtex-II DC and AC characteristics are specified for both commercial and industrial grades. Except the operating temperature range or unless otherwise noted, all the DC and AC electrical parameters are the same for a particular speed grade (that is, the timing characteristics of a –4 speed grade industrial device are the same as for a –4 speed grade commercial device). However, only selected speed grades and/or devices might be available in the industrial range.

All supply voltage and junction temperature specifications are representative of worst-case conditions. The parameters included are common to popular designs and typical applications. Contact Xilinx for design considerations requiring more detailed information.

All specifications are subject to change without notice.

Virtex-II DC Characteristics

Table 1: **Absolute Maximum Ratings**

Symbol	Description		Units
V_{CCINT}	Internal Supply voltage relative to GND	–0.5 to 1.65	V
V_{CCAUX}	Auxiliary supply voltage relative to GND	–0.5 to 4.0	V
V_{CCO}	Output drivers supply voltage relative to GND	–0.5 to 4.0	V
V_{BATT}	Key memory battery backup supply	–0.5 to 4.0	V
V_{REF}	Input Reference Voltage	–0.5 to 4.0	V
V_{IN}	Input voltage relative to GND (user and dedicated I/Os)	–0.5 to 4.0	V
V_{TS}	Voltage applied to 3-state output (user and dedicated I/Os)	–0.5 to 4.0	V
T_{STG}	Storage temperature (ambient)	–65 to +150	°C
T_{SOL}	Maximum soldering temp.	+220	°C
T_J	Operating junction temperature	+125	°C

Notes:

1. Stresses beyond those listed under Absolute Maximum Ratings might cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those listed under Operating Conditions is not implied. Exposure to Absolute Maximum Ratings conditions for extended periods of time might affect device reliability.

Table 2: **Recommended Operating Conditions**

Symbol	Description		Min	Max	Units
V_{CCINT}	Internal Supply voltage relative to GND, T_J = 0 °C to +85°C	Commercial	1.425	1.575	V
	Internal Supply voltage relative to GND, T_J = –40°C to +100°C	Industrial	1.425	1.575	V
V_{CCAUX}	Auxiliary supply voltage relative to GND, T_J = 0 °C to +85°C	Commercial	3.0	3.6	V
	Auxiliary supply voltage relative to GND, T_J = –40°C to +100°C	Industrial	3.0	3.6	V
V_{CCO}	Supply voltage relative to GND, T_J = 0 °C to +85°C	Commercial	1.2	3.6	V
	Supply voltage relative to GND, T_J = –40°C to +100°C	Industrial	1.2	3.6	V
V_{BATT}	Battery voltage relative to GND, T_J = 0 °C to +85°C	Commercial	1.0	3.6	V
	Battery voltage relative to GND, T_J = –40°C to +100°C	Industrial	1.0	3.6	V

Notes:

1. If V_{CCAUX} and V_{CCO} are both at 3.3 V, they must use a common supply voltage.
2. If battery is not used, do not connect V_{BATT}.
3. For LVDS operation, V_{CCAUX} min is 3.13 V and max is 3.47 V.

© 2001 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at http://www.xilinx.com/legal.htm.
All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

Table 3: **DC Characteristics Over Recommended Operating Conditions**

Symbol	Description	Device	Min	Max	Units
V_{DRINT}	Data Retention V_{CCINT} Voltage	All	1.2		V
V_{DRI}	Data Retention V_{CCAUX} Voltage	All	2.5		V
I_{REF}	V_{REF} current per bank	All	–10	+10	µA
I_L	Input leakage current	All	–10	+10	µA
C_{IN}	Input capacitance	All		10	pF
I_{RPU}	Pad pull-up (when selected) @ V_{IN} = 0 V, V_{CCO} = 3.3 V (sample tested)	All	Note 1	250	µA
I_{RPD}	Pad pull-down (when selected) @ V_{IN} = 3.6 V (sample tested)	All	Note 1	250	µA
I_{BATT}	Battery supply current	All		100	nA

Notes:

1. Internal pull-up and pull-down resistors guarantee valid logic levels at unconnected input pins. These pull-up and pull-down resistors do not guarantee valid logic levels when input pins are connected to other circuits.

Table 4: **Quiescent Supply Current**

Symbol	Description	Device	Min	Typical	Max	Units
I_{CCINTQ}	Quiescent V_{CCINT} supply current	XC2v40		75	TBD	mA
		XC2v80		75	TBD	
		XC2v250		75	TBD	
		XC2v500		100	TBD	
		XC2v1000		100	250	
		XC2v1500		150	TBD	
		XC2v2000		200	TBD	
		XC2v3000		200	TBD	
		XC2v4000		250	TBD	
		XC2v6000		250	1000	
		XC2v8000		TBD	TBD	
I_{CCOQ}	Quiescent V_{CCO} supply current$^{(1,2)}$	XC2v40		1	TBD	mA
		XC2v80		1	TBD	
		XC2v250		1	TBD	
		XC2v500		1	TBD	
		XC2v1000		1	2	
		XC2v1500		2	TBD	
		XC2v2000		2	TBD	
		XC2v3000		2	TBD	
		XC2v4000		2	TBD	
		XC2v6000		2	4	
		XC2v8000		TBD	TBD	
I_{CCAUXQ}	Quiescent V_{CCAUX} supply current$^{(1,2)}$	XC2v40		10	TBD	mA
		XC2v80		10	TBD	
		XC2v250		10	TBD	
		XC2v500		10	TBD	
		XC2v1000		10	25	
		XC2v1500		20	TBD	
		XC2v2000		20	TBD	
		XC2v3000		20	TBD	
		XC2v4000		25	TBD	
		XC2v6000		25	100	
		XC2v8000		TBD	TBD	

Notes:

1. With no output current loads, no active input pull-up resistors, all I/O pins are 3-state and floating.
2. If DCI or differential signaling is used, more accurate quiescent current estimates can be obtained by using the Power Estimator or XPOWER™.
3. Data are retained even if V_{CCO} drops to 0 V.
4. Values specified for quiescent supply current parameters are Commercial Grade only.

Power-On Power Supply Requirements

Xilinx FPGAs require a certain amount of supply current during power-on to insure proper device operation. The actual current consumed depends on the power-on ramp rate of the power supply.

The V_{CCINT}, V_{CCAUX}, and V_{CCO} power supplies shall ramp on no faster than 1 ms and no slower than 50 ms. Ramp on is defined as: 0 V_{DC} to minimum supply voltages.

V_{CCAUX} and V_{CCO} for bank 4 must be connected together (3.3 V_{DC}) to meet the following specification.

Table 5 shows the minimum current required by Virtex-II devices for proper power on and configuration.

Power supplies can be turned on in any sequence, as long as V_{CCAUX} and V_{CCO} are connected together for bank 4.

If any V_{CCO} bank powers up before V_{CCAUX}, then each bank draws up to 600 mA, worst case, until the V_{CCAUX} powers on(1). This does not harm the device. If the current is limited to the minimum value above, or larger, the device powers on properly after all three supplies have passed through their power on reset threshold voltages.

Once initialized and configured, use the power calculator to estimate current drain on these supplies.

Notes:

1. The 600 mA is transient current (peak); it eventually dissipates even if V_{CCAUX} does not power up.

Table 5: **Power On Current for Virtex-II Devices**

	Device (mA)										
	2v40	2v80	2v250	2v500	2v1000	2v1500	2v2000	2v3000	2v4000	2v6000	2v8000
$I_{CCINTMIN}$	250	250	250	250	500	500	500	500	750	1000	TBD
$I_{CCAUXMIN}$	100	100	100	100	100	100	100	100	100	100	TBD
I_{CCOMIN}	50	50	50	50	50	100	100	100	100	100	TBD

Notes:

1. Values specified for power on current parameters are Commercial Grade only.

DC Input and Output Levels

Values for V_{IL} and V_{IH} are recommended input voltages. Values for I_{OL} and I_{OH} are guaranteed over the recommended operating conditions at the V_{OL} and V_{OH} test points. Only selected standards are tested. These are chosen to ensure that all standards meet their specifications. The selected standards are tested at minimum V_{CCO} with the respective V_{OL} and V_{OH} voltage levels shown. Other standards are sample tested.

Table 6: **DC Input and Output Levels**

Input/Output Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, min	V, max	V, min	V, max	V, Max	V, Min	mA	mA
LVTTL(1)	– 0.5	0.8	2.0	V_{CCO} + 0.5	0.4	2.4	24	– 24
LVCMOS33	– 0.5	0.8	2.0	V_{CCO} + 0.5	0.4	V_{CCO} – 0.4	24	– 24
LVCMOS25	– 0.5	0.7	1.7	V_{CCO} + 0.5	0.4	V_{CCO} – 0.4	24	– 24
LVCMOS18	– 0.5	20% V_{CCO}	70% V_{CCO}	V_{CCO} + 0.5	0.4	V_{CCO} – 0.45	16	– 16
LVCMOS15	– 0.5	20% V_{CCO}	70% V_{CCO}	V_{CCO} + 0.5	0.4	V_{CCO} – 0.45	16	– 16
PCI33_3	– 0.5	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.5	10% V_{CCO}	90% V_{CCO}	Note 2	Note 2
PCI66_3	– 0.5	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.5	10% V_{CCO}	90% V_{CCO}	Note 2	Note 2
PCI–X	– 0.5	Note 2	Note 2	Note 2	Note 2	Note 2	Note 2	Note 2
GTLP	– 0.5	V_{REF} – 0.1	V_{REF} + 0.1	V_{CCO} + 0.5	0.6	n/a	36	n/a
GTL	– 0.5	V_{REF} – 0.05	V_{REF} + 0.05	V_{CCO} + 0.5	0.4	n/a	40	n/a
HSTL I	– 0.5	V_{REF} – 0.1	V_{REF} + 0.1	V_{CCO} + 0.5	0.4	V_{CCO} – 0.4	8	– 8
HSTL II	– 0.5	V_{REF} – 0.1	V_{REF} + 0.1	V_{CCO} + 0.5	0.4	V_{CCO} – 0.4	16	– 16

Table 6: **DC Input and Output Levels** ***(Continued)***

Input/Output Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, min	V, max	V, min	V, max	V, Max	V, Min	mA	mA
HSTL III	– 0.5	V_{REF} – 0.1	V_{REF} + 0.1	V_{CCO} + 0.5	0.4	V_{CCO} – 0.4	24	– 8
HSTL IV	– 0.5	V_{REF} – 0.1	V_{REF} + 0.1	V_{CCO} + 0.5	0.4	V_{CCO} – 0.4	48	– 8
SSTL3 I	– 0.5	V_{REF} – 0.2	V_{REF} + 0.2	V_{CCO} + 0.5	V_{REF} – 0.6	V_{REF} + 0.6	8	– 8
SSTL3 II	– 0.5	V_{REF} – 0.2	V_{REF} + 0.2	V_{CCO} + 0.5	V_{REF} – 0.8	V_{REF} + 0.8	16	– 16
SSTL2 I	– 0.5	V_{REF} – 0.2	V_{REF} + 0.2	V_{CCO} + 0.5	V_{REF} – 0.65	V_{REF} + 0.65	7.6	– 7.6
SSTL2 II	– 0.5	V_{REF} – 0.2	V_{REF} + 0.2	V_{CCO} + 0.5	V_{REF} – 0.80	V_{REF} + 0.80	15.2	– 15.2
AGP	– 0.5	V_{REF} – 0.2	V_{REF} + 0.2	V_{CCO} + 0.5	10% V_{CCO}	90% V_{CCO}	Note 2	Note 2

Notes:
1. V_{OL} and V_{OH} for lower drive currents are sample tested. The DONE pin is always LVTTL 12 mA.
2. Tested according to the relevant specifications.

LDT Differential Signal DC Specifications (LDT_25)

Table 7: **LDT DC Specifications**

DC Parameter	Symbol	Conditions	Min	Typ	Max	Units
Differential Output Voltage	V_{OD}	R_T = 100 ohm across Q and $\overline{Q}$ signals	500	600	700	mV
Change in V_{OD} Magnitude	Δ V_{OD}		–15		15	mV
Output Common Mode Voltage	V_{OCM}	R_T = 100 ohm across Q and $\overline{Q}$ signals	560	600	640	mV
Change in V_{OS} Magnitude	Δ V_{OCM}		–15		15	mV
Input Differential Voltage	V_{ID}		200	600	1000	mV
Change in V_{ID} Magnitude	Δ V_{ID}		–15		15	mV
Input Common Mode Voltage	V_{ICM}		500	600	700	mV
Change in V_{ICM} Magnitude	Δ V_{ICM}		–15		15	mV

LVDS DC Specifications (LVDS_33 & LVDS_25)

Table 8: **LVDS DC Specifications**

DC Parameter	Symbol	Conditions	Min	Typ	Max	Units
Supply Voltage	V_{CCO}			3.3 or 2.5		V
Output High Voltage for Q and $\overline{Q}$	V_{OH}	R_T = 100 Ω across Q and $\overline{Q}$ signals			1.475	V
Output Low Voltage for Q and $\overline{Q}$	V_{OL}	R_T = 100 Ω across Q and $\overline{Q}$ signals	0.925			V
Differential Output Voltage (Q – $\overline{Q}$), Q = High ($\overline{Q}$ – Q), $\overline{Q}$ = High	V_{ODIFF}	R_T = 100 Ω across Q and $\overline{Q}$ signals	250	350	400	mV
Output Common-Mode Voltage	V_{OCM}	R_T = 100 Ω across Q and $\overline{Q}$ signals	1.125	1.2	1.275	V
Differential Input Voltage (Q – $\overline{Q}$), Q = High ($\overline{Q}$ – Q), $\overline{Q}$ = High	V_{IDIFF}	Common-mode input voltage = 1.25 V	100	350	N/A	mV
Input Common-Mode Voltage	V_{ICM}	Differential input voltage = ±350 mV	0.2	1.25	2.2	V

Extended LVDS DC Specifications (LVDSEXT_33 & LVDSEXT_25)

Table 9: **Extended LVDS DC Specifications**

DC Parameter	Symbol	Conditions	Min	Typ	Max	Units
Supply Voltage	V_{CCO}			3.3 or 2.5		V
Output High Voltage for Q and $\overline{Q}$	V_{OH}	R_T = 100 Ω across Q and $\overline{Q}$ signals			1.70	V
Output Low Voltage for Q and $\overline{Q}$	V_{OL}	R_T = 100 Ω across Q and $\overline{Q}$ signals	0.705			V
Differential Output Voltage (Q – $\overline{Q}$), Q = High ($\overline{Q}$ – Q), $\overline{Q}$ = High	V_{ODIFF}	R_T = 100 Ω across Q and $\overline{Q}$ signals	440		820	mV
Output Common-Mode Voltage	V_{OCM}	R_T = 100 Ω across Q and $\overline{Q}$ signals	1.125	1.200	1.275	V
Differential Input Voltage (Q – $\overline{Q}$), Q = High ($\overline{Q}$ – Q), $\overline{Q}$ = High	V_{IDIFF}	Common-mode input voltage = 1.25 V	100	350	N/A	mV
Input Common-Mode Voltage	V_{ICM}	Differential input voltage = ±350 mV	0.2	1.25	2.2	V

LVPECL DC Specifications

These values are valid when driving a 100 Ω differential load only, i.e., a 100 Ω resistor between the two receiver pins. The V_{OH} levels are 200 mV below standard LVPECL levels and are compatible with devices tolerant of lower common-mode ranges. Table 10 summarizes the DC output specifications of LVPECL.

Table 10: **LVPECL DC Specifications**

DC Parameter	Min	Max	Min	Max	Min	Max	Units
V_{CCO}	3.0		3.3		3.6		V
V_{OH}	1.8	2.11	1.92	2.28	2.13	2.41	V
V_{OL}	0.96	1.27	1.06	1.43	1.30	1.57	V
V_{IH}	1.49	2.72	1.49	2.72	1.49	2.72	V
V_{IL}	0.86	2.125	0.86	2.125	0.86	2.125	V
Differential Input Voltage	0.3	–	0.3	–	0.3	–	V

Virtex-II Performance Characteristics

This section provides the performance characteristics of some common functions and designs implemented in Virtex-II devices. The numbers reported here are worst-case values; they have all been fully characterized. Note that these values are subject to the same guidelines as **Virtex-II Switching Characteristics**, page 92 (speed files).

Table 11 provides pin-to-pin values (in nanoseconds) including IOB delays; that is, delay through the device from input pin to output pin. In the case of multiple inputs and outputs, the worst delay is reported.

Table 11: **Pin-to-Pin Performance**

Description	Pin-to-Pin (w/ I/O delays)	Device Used & Speed Grade
Basic Functions		
16-bit Address Decoder	6.3	XC2V1000 –5
32-bit Address Decoder	7.7	XC2V1000 –5
64-bit Address Decoder	9.3	XC2V1000 –5
4:1 MUX	5.7	XC2V1000 –5
8:1 MUX	6.5	XC2V1000 –5
16:1 MUX	6.7	XC2V1000 –5
32:1 MUX	8.7	XC2V1000 –5
Combinatorial (pad to LUT to pad)	5.0	XC2V1000 –5
Memory		
Block RAM		
Pad to setup	1.6	
Clock to Pad	9.5	
Distributed RAM		
Pad to setup	2.7	XC2V1000 –5
Clock to Pad	5.1 (no clk skew)	XC2V1000 –5

Table 12 shows internal (register-to-register) performance. Values are reported in MHz.

Table 12: **Register-to-Register Performance**

Description	Register-to-Register Performance	Device Used & Speed Grade
Basic Functions		
16-bit Address Decoder	398	XC2V1000 –5
32-bit Address Decoder	291	XC2V1000 –5
64-bit Address Decoder	275	XC2V1000 –5
4:1 MUX	563	XC2V1000 –5
8:1 MUX	454	XC2V1000 –5
16:1 MUX	414	XC2V1000 –5
32:1 MUX	323	XC2V1000 –5
Register to LUT to Register	613	XC2V1000 –5
8-bit Adder	292	XC2V1000 –5
16-bit Adder	239	XC2V1000 –5
64-bit Adder	114	XC2V1000 –5
64-bit Counter	114	XC2V1000 –5
64-bit Accumulator	110	XC2V1000 –5

Table 12: **Register-to-Register Performance** ***(Continued)***

Description	Register-to-Register Performance	Device Used & Speed Grade
Multiplier 18x18 (with Block RAM inputs)	88	XC2V1000 –5
Multiplier 18x18 (with Register inputs)	105	XC2V1000 –5
Memory		
Block RAM		
Single-Port 4096 x 4 bits	265	
Single-Port 2048 x 9 bits	N/A	
Single-Port 1024 x 18 bits	N/A	
Single-Port 512 x 36 bits	N/A	
Dual-Port A:4096 x 4 bits & B:1024 x 18 bits	N/A	
Dual-Port A:1024 x 18 bits & B:1024 x 18 bits	N/A	
Dual-Port A:2048 x 9 bits & B: 512 x 36 bits	N/A	
Distributed RAM		
Single-Port 32 x 8-bit	385	XC2V1000 –5
Single-Port 64 x 8-bit	335	XC2V1000 –5
Single-Port 128 x 8-bit	266	XC2V1000 –5
Dual-Port 16 x 8	400	XC2V1000 –5
Dual-Port 32 x 8	300	XC2V1000 –5
Dual-Port 64 x 8	294	XC2V1000 –5
Shift Registers		
128-bit SRL	N/A	
256-bit SRL	N/A	
FIFOs (Async. in Block RAM)		
1024 x 18-bit	N/A	
1024 x 18-bit	N/A	
FIFOs (Sync. in SRL)		
128 x 8-bit	N/A	
128 x 16-bit	N/A	
CAMs in Block RAM		
32 x 9-bit	N/A	
64 x 9-bit	N/A	
128 x 9-bit	N/A	
256 x 9-bit	N/A	
CAMs in SRL		
32 x 16-bit	N/A	
64 x 32-bit	N/A	
128 x 40-bit	N/A	
256 x 48-bit	N/A	
1024 x 16-bit	N/A	
1024 x 72-bit	N/A	

Virtex-II Switching Characteristics

Switching characteristics are specified on a per-speed-grade basis and can be designated as Advance, Preliminary, or Production. Note that **Virtex-II Performance Characteristics**, page 90 are subject to these guidelines, as well. Each designation is defined as follows:

Advance: These speed files are based on simulations only and are typically available soon after device design specifications are frozen. Although speed grades with this designation are considered relatively stable and conservative, some under-reporting might still occur.

Preliminary: These speed files are based on complete ES (engineering sample) silicon characterization. Devices and speed grades with this designation are intended to give a better indication of the expected performance of production silicon. The probability of under-reporting delays is greatly reduced as compared to Advance data.

Production: These speed files are released once enough production silicon of a particular device family member has been characterized to provide full correlation between speed files and devices over numerous production lots. There is no under-reporting of delays, and customers receive formal notification of any subsequent changes. Typically, the slowest speed grades transition to Production before faster speed grades.

Since individual family members are produced at different times, the migration from one category to another depends completely on the status of the fabrication process for each device. Table 13 correlates the current status of each Virtex-II device with a corresponding speed file designation.

Table 13: **Virtex-II Device Speed Grade Designations**

	Speed Grade Designations		
Device	**Advance**	**Preliminary**	**Production**
XC2V40	–6, –5, –4		
XC2V80	–6, –5, –4		
XC2V250	–6, –5, –4		
XC2V500	–6, –5, –4		
XC2V1000	–6		–5, –4
XC2V1500	–6, –5, –4		
XC2V2000	–6, –5, –4		
XC2V3000	–6, –5, –4		
XC2V4000	–6, –5, –4		
XC2V6000	–6, –5, –4		
XC2V8000	–5, –4		

All specifications are always representative of worst-case supply voltage and junction temperature conditions.

Testing of Switching Characteristics

All devices are 100% functionally tested. Internal timing parameters are derived from measuring internal test patterns. Listed below are representative values. For more specific, more precise, and worst-case guaranteed data, use the values reported by the Xilinx static timing analyzer and back-annotate to the simulation net list. Unless otherwise noted, values apply to all Virtex-II devices.

IOB Input Switching Characteristics

Input delays associated with the pad are specified for LVTTL levels. For other standards, adjust the delays with the values shown in **IOB Input Switching Characteristics Standard Adjustments**, page 94.

Table 14: **IOB Input Switching Characteristics**

			Speed Grade			
Description	**Symbol**	**Device**	**–6**	**–5**	**–4**	**Units**
Propagation Delays						
Pad to I output, no delay	T_{IOPI}	All	0.69	0.76	0.88	ns, max
Pad to I output, with delay	T_{IOPID}	2v40	3.15	3.46	3.98	ns, max
		2v80	3.15	3.46	3.98	ns, max
		2v250	3.15	3.46	3.98	ns, max
		2v500	3.15	3.46	3.98	ns, max
		2v1000	3.15	3.46	3.98	ns, max
		2v1500	3.15	3.46	3.98	ns, max
		2v2000	3.15	3.46	3.98	ns, max
		2v3000	3.24	3.56	4.10	ns, max
		2v4000	3.24	3.56	4.10	ns, max
		2v6000	3.51	3.86	4.44	ns, max
		2v8000	TBD	TBD	TBD	ns, max

Table 14: **IOB Input Switching Characteristics** ***(Continued)***

			Speed Grade			
Description	**Symbol**	**Device**	**–6**	**–5**	**–4**	**Units**
Propagation Delays						
Pad to output IQ via transparent latch, no delay	T_{IOPLI}	All	0.99	1.08	1.24	ns, max
Pad to output IQ via transparent latch, with delay	T_{IOPLID}	2v40	3.44	3.78	4.35	ns, max
		2v80	3.44	3.78	4.35	ns, max
		2v250	3.44	3.78	4.35	ns, max
		2v500	3.44	3.78	4.35	ns, max
		2v1000	3.44	3.78	4.35	ns, max
		2v1500	3.44	3.78	4.35	ns, max
		2v2000	3.44	3.78	4.35	ns, max
		2v3000	3.53	3.88	4.46	ns, max
		2v4000	3.53	3.88	4.46	ns, max
		2v6000	3.80	4.18	4.81	ns, max
		2v8000	TBD	TBD	TBD	ns, max
Clock CLK to output IQ	T_{IOCKIQ}	All	0.63	0.69	0.80	ns, max
Setup and Hold Times With Respect to Clock at IOB Input Register						
Pad, no delay	T_{IOPICK}/T_{IOICKP}	All	0.88/–0.36	0.96/–0.39	1.11/–0.45	ns, min
Pad, with delay	$T_{IOPICKD}/T_{IOICKPD}$	2v40	3.33/–2.07	3.66/–2.28	4.21/–2.63	ns, min
		2v80	3.33/–2.07	3.66/–2.28	4.21/–2.63	ns, min
		2v250	3.33/–2.07	3.66/–2.28	4.21/–2.63	ns, min
		2v500	3.33/–2.07	3.66/–2.28	4.21/–2.63	ns, min
		2v1000	3.33/–2.07	3.66/–2.28	4.21/–2.63	ns, min
		2v1500	3.33/–2.07	3.66/–2.28	4.21/–2.63	ns, min
		2v2000	3.33/–2.07	3.66/–2.28	4.21/–2.63	ns, min
		2v3000	3.42/–2.14	3.76/–2.35	4.33/–2.71	ns, min
		2v4000	3.42/–2.14	3.76/–2.35	4.33/–2.71	ns, min
		2v6000	3.69/–2.33	4.06/–2.56	4.67/–2.95	ns, min
		2v8000	TBD	TBD	TBD	ns, min
ICE input	$T_{IOICECK}/T_{IOCKICE}$	All	0.19/ 0.03	0.21/ 0.04	0.24/ 0.04	ns, min
SR input (IFF, synchronous)	$T_{IOSRCKI}$	All	0.27	0.30	0.34	ns, min
Set/Reset Delays						
SR input to IQ (asynchronous)	T_{IOSRIQ}	All	1.11	1.22	1.40	ns, max
GSR to output IQ	T_{GSRQ}	All	7.27	7.99	9.19	ns, max

Notes:

1. Input timing for LVTTL is measured at 1.4 V. For other I/O standards, see Table 18.

IOB Input Switching Characteristics Standard Adjustments

Table 15: **IOB Input Switching Characteristics Standard Adjustments**

			Speed Grade			
Description	**Symbol**	**Standard**	**–6**	**–5**	**–4**	**Units**
Data Input Delay Adjustments						
Standard-specific data input delay adjustments	T_{ILVTTL}	LVTTL	0.00	0.00	0.00	ns
	$T_{ILVCMOS33}$	LVCMOS33	0.00	0.00	0.00	ns
	$T_{ILVCMOS25}$	LVCMOS25	0.10	0.11	0.12	ns
	$T_{ILVCMOS18}$	LVCMOS18	0.39	0.43	0.49	ns
	$T_{ILVCMOS15}$	LVCMOS15	0.91	1.00	1.15	ns
	T_{ILVDS_25}	LVDS_25	0.55	0.60	0.69	ns
	T_{ILVDS_33}	LVDS_33	0.55	0.60	0.69	ns
	$T_{ILVPECL\_33}$	LVPECL	0.55	0.60	0.69	ns
	T_{IPCI33_3}	PCI, 33 MHz, 3.3 V	0.00	0.00	0.00	ns
	T_{IPCI66_3}	PCI, 66 MHz, 3.3 V	0.00	0.00	0.00	ns
	T_{IPCIX}	PCI–X, 133 MHz, 3.3 V	0.00	0.00	0.00	ns
	T_{IGTL}	GTL	0.38	0.42	0.48	ns
	T_{IGTLP}	GTLP	0.38	0.42	0.48	ns
	T_{IHSTL_I}	HSTL I	0.38	0.42	0.48	ns
	T_{IHSTL_II}	HSTL II	0.38	0.42	0.48	ns
	T_{IHSTL_III}	HSTL III	0.38	0.42	0.48	ns
	T_{IHSTL_IV}	HSTL IV	0.38	0.42	0.48	ns
	$T_{IHSTL\_I\_18}$	HSTL I_18	0.38	0.42	0.48	ns
	$T_{IHSTL\_II\_18}$	HSTL II_18	0.38	0.42	0.48	ns
	$T_{IHSTL\_III\_18}$	HSTL III_18	0.38	0.42	0.48	ns
	$T_{IHSTL\_IV\_18}$	HSTL IV_18	0.38	0.42	0.48	ns
	T_{ISSTL2_I}	SSTL2 I	0.38	0.42	0.48	ns
	T_{ISSTL2_II}	SSTL2 II	0.38	0.42	0.48	ns
	T_{ISSTL3_I}	SSTL3 I	0.32	0.35	0.40	ns
	T_{ISSTL3_II}	SSTL3 II	0.32	0.35	0.40	ns
	T_{IAGP}	AGP	0.32	0.35	0.40	ns
	T_{ILVDCI_33}	LVDCI_33	0.00	0.00	0.00	ns
	T_{ILVDCI_25}	LVDCI_25	0.10	0.11	0.12	ns
	T_{ILVDCI_18}	LVDCI_18	0.39	0.43	0.49	ns

Table 15: **IOB Input Switching Characteristics Standard Adjustments** ***(Continued)***

			Speed Grade			
Description	**Symbol**	**Standard**	**–6**	**–5**	**–4**	**Units**
	T_{ILVDCI_15}	LVDCI_15	0.91	1.00	1.14	ns
	$T_{ILVDCI\_DV2\_33}$	LVDCI_DV2_33	0.00	0.00	0.00	ns
	$T_{ILVDCI\_DV2\_25}$	LVDCI_DV2_25	0.10	0.11	0.12	ns
	$T_{ILVDCI\_DV2\_18}$	LVDCI_DV2_18	0.39	0.43	0.49	ns
	$T_{ILVDCI\_DV2\_15}$	LVDCI_DV2_15	0.91	1.00	1.14	ns
	T_{IGTL_DCI}	GTL_DCI	0.38	0.42	0.48	ns
	T_{IGTLP_DCI}	GTLP_DCI	0.38	0.42	0.48	ns
	$T_{IHSTL\_I\_DCI}$	HSTL_I_DCI	0.38	0.42	0.48	ns
	$T_{IHSTL\_II\_DCI}$	HSTL_II_DCI	0.38	0.42	0.48	ns
	$T_{IHSTL\_III\_DCI}$	HSTL_III_DCI	0.38	0.42	0.48	ns
	$T_{IHSTL\_IV\_DCI}$	HSTL_IV_DCI	0.38	0.42	0.48	ns
	$T_{IHSTL\_I\_DCI\_18}$	HSTL_I_DCI_18	0.38	0.42	0.48	ns
	$T_{IHSTL\_II\_DCI\_18}$	HSTL_II_DCI_18	0.38	0.42	0.48	ns
	$T_{IHSTL\_III\_DCI\_18}$	HSTL_III_DCI_18	0.38	0.42	0.48	ns
	$T_{IHSTL\_IV\_DCI\_18}$	HSTL_IV_DCI_18	0.38	0.42	0.48	ns
	$T_{ISSTL2\_I\_DCI}$	SSTL2_I_DCI	0.38	0.42	0.48	ns
	$T_{ISSTL2\_II\_DCI}$	SSTL2_II_DCI	0.38	0.42	0.48	ns
	$T_{ISSTL3\_I\_DCI}$	SSTL3_I_DCI	0.32	0.35	0.40	ns
	$T_{ISSTL3\_II\_DCI}$	SSTL3_II_DCI	0.32	0.35	0.40	ns
	T_{ILDT_25}	LDT_25	0.45	0.49	0.56	ns
	T_{IULVDS_25}	ULVDS_25	0.45	0.49	0.56	ns

Notes:

1. Input timing for LVTTL is measured at 1.4 V. For other I/O standards, see Table 18.

IOB Output Switching Characteristics

Output delays terminating at a pad are specified for LVTTL with 12 mA drive and fast slew rate. For other standards, adjust the delays with the values shown in **IOB Output Switching Characteristics Standard Adjustments**, page 97.

Table 16: **IOB Output Switching Characteristics**

		Speed Grade			
Description	**Symbol**	**–6**	**–5**	**–4**	**Units**
Propagation Delays					
O input to Pad	T_{IOOP}	2.39	2.63	3.03	ns, max
O input to Pad via transparent latch	T_{IOOLP}	2.69	2.95	3.40	ns, max
3-State Delays					
T input to Pad high-impedance(1)	T_{IOTHZ}	0.51	0.56	0.64	ns, max
T input to valid data on Pad	T_{IOTON}	2.34	2.57	2.96	ns, max
T input to Pad high-impedance via transparent latch(1)	$T_{IOTLPHZ}$	0.80	0.88	1.01	ns, max
T input to valid data on Pad via transparent latch	$T_{IOTLPON}$	2.63	2.89	3.33	ns, max
GTS to Pad high impedance(1)	T_{GTS}	6.56	7.22	8.30	ns, max
Sequential Delays					
Clock CLK to Pad	T_{IOCKP}	2.72	2.99	3.44	ns, max
Clock CLK to Pad high-impedance (synchronous)(1)	T_{IOCKHZ}	0.95	1.04	1.20	ns, max
Clock CLK to valid data on Pad (synchronous)	T_{IOCKON}	2.78	3.06	3.51	ns, max
Setup and Hold Times Before/After Clock CLK					
O input	T_{IOOCK}/T_{IOCKO}	0.31/–0.08	0.34/–0.09	0.39/–0.11	ns, min
OCE input	$T_{IOOCECK}/T_{IOCKOCE}$	0.19/–0.06	0.21/–0.07	0.24/–0.08	ns, min
SR input (OFF)	$T_{IOSRCKO}/T_{IOCKOSR}$	0.27/–0.05	0.30/–0.06	0.34/–0.07	ns, min
3–State Setup Times, T input	T_{IOTCK}/T_{IOCKT}	0.28/–0.06	0.31/–0.07	0.35/–0.08	ns, min
3–State Setup Times, TCE input	$T_{IOTCECK}/T_{IOCKTCE}$	0.19/–0.06	0.21/–0.07	0.24/–0.08	ns, min
3–State Setup Times, SR input (TFF)	$T_{IOSRCKT}/T_{IOCKTSR}$	0.27/–0.05	0.30/–0.06	0.34/–0.07	ns, min
Set/Reset Delays					
SR input to Pad (asynchronous)	T_{IOSRP}	3.37	3.71	4.26	ns, max
SR input to Pad high-impedance (asynchronous)(1)	T_{IOSRHZ}	1.52	1.67	1.92	ns, max
SR input to valid data on Pad (asynchronous)	T_{IOSRON}	3.35	3.68	4.23	ns, max
GSR to Pad	T_{IOGSRQ}	5.44	5.98	6.88	ns, max

Notes:

1. The 3-state turn-off delays should not be adjusted.

IOB Output Switching Characteristics Standard Adjustments

Output delays terminating at a pad are specified for LVTTL with 12 mA drive and fast slew rate. For other standards, adjust the delays by the values shown.

Table 17: **IOB Output Switching Characteristics Standard Adjustments**

			Speed Grade			
Description	**Symbol**	**Standard**	**–6**	**–5**	**–4**	**Units**
Output Delay Adjustments						
Standard-specific adjustments for output delays terminating at pads (based on standard capacitive load, Csl)	T_{OLVTTL_S2}	LVTTL, Slow, 2 mA	8.53	9.38	10.79	ns
	T_{OLVTTL_S4}	4 mA	5.15	5.67	6.52	ns
	T_{OLVTTL_S6}	6 mA	3.75	4.12	4.74	ns
	T_{OLVTTL_S8}	8 mA	2.57	2.83	3.25	ns
	T_{OLVTTL_S12}	12 mA	2.00	2.19	2.52	ns
	T_{OLVTTL_S16}	16 mA	1.17	1.28	1.48	ns
	T_{OLVTTL_S24}	24 mA	0.85	0.94	1.08	ns
	T_{OLVTTL_F2}	LVTTL, Fast, 2 mA	5.37	5.90	6.79	ns
	T_{OLVTTL_F4}	4 mA	2.19	2.41	2.77	ns
	T_{OLVTTL_F6}	6 mA	0.94	1.03	1.18	ns
	T_{OLVTTL_F8}	8 mA	0.06	0.07	0.08	ns
	T_{OLVTTL_F12}	12 mA	0.00	0.00	0.00	ns
	T_{OLVTTL_F16}	16 mA	–0.30	–0.33	–0.38	ns
	T_{OLVTTL_F24}	24 mA	–0.44	–0.48	–0.55	ns
	T_{OLVDS_25}	LVDS	–1.02	–1.12	–1.29	ns
	T_{OLVDS_33}	LVDS	–1.07	–1.18	–1.36	ns
	$T_{OLVDSEXT\_25}$	LVDS	–0.94	–1.03	–1.19	ns
	$T_{OLVDSEXT\_33}$	LVDS	–0.95	–1.05	–1.21	ns
	T_{OLDT_25}	LDT	–1.01	–1.11	–1.28	ns
	T_{OBLVDS_25}	BLVDS	0.63	0.69	0.79	ns
	T_{OULVDS_25}	ULVDS	–1.01	–1.11	–1.28	ns
	$T_{OLVPECL\_33}$	LVPECL	0.74	0.81	0.93	ns
	T_{OPCI33_3}	PCI, 33 MHz, 3.3 V	1.06	1.17	1.34	ns
	T_{OPCI66_3}	PCI, 66 MHz, 3.3 V	–0.14	–0.15	–0.18	ns
	T_{OPCIX}	PCI–X, 133 MHz, 3.3 V	–0.14	–0.16	–0.18	ns
	T_{OGTL}	GTL	1.13	1.24	1.43	ns
	T_{OGTLP}	GTLP	0.43	0.47	0.54	ns
	T_{OHSTL_I}	HSTL I	0.19	0.21	0.24	ns
	T_{OHSTL_II}	HSTL II	0.01	0.01	0.01	ns
	T_{OHSTL_III}	HSTL III	–0.17	–0.18	–0.21	ns
	T_{OHSTL_IV}	HSTL IV	–0.22	–0.24	–0.28	ns
	$T_{OHSTL\_I\_18}$	HSTL I_18	0.22	0.25	0.28	ns
	$T_{OHSTL\_II\_18}$	HSTL II_18	0.13	0.14	0.16	ns
	$T_{OHSTL\_III\_18}$	HSTL III_18	0.11	0.12	0.14	ns
	$T_{OHSTL\_IV\_18}$	HSTL IV_18	0.15	0.17	0.19	ns

Table 17: **IOB Output Switching Characteristics Standard Adjustments** ***(Continued)***

			Speed Grade			
Description	**Symbol**	**Standard**	**–6**	**–5**	**–4**	**Units**
	T_{OSSTL2_I}	SSTL2 I	0.20	0.22	0.25	ns
	T_{OSSTL2_II}	SSTL2 II	–0.36	–0.39	–0.45	ns
	T_{OSSTL3_I}	SSTL3 I	0.29	0.32	0.36	ns
	T_{OSSTL3_II}	SSTL3 II	–0.14	–0.16	–0.18	ns
	T_{OAGP}	AGP	–0.44	–0.48	–0.56	ns
	$T_{OLVCMOS33\_S2}$	LVCMOS33, Slow, 2 mA	7.03	7.74	8.90	ns
	$T_{OLVCMOS33\_S4}$	4 mA	3.83	4.22	4.85	ns
	$T_{OLVCMOS33\_S6}$	6 mA	2.73	3.00	3.45	ns
	$T_{OLVCMOS33\_S8}$	8 mA	1.97	2.17	2.50	ns
	$T_{OLVCMOS33\_S12}$	12 mA	1.46	1.60	1.84	ns
	$T_{OLVCMOS33\_S16}$	16 mA	0.87	0.96	1.10	ns
	$T_{OLVCMOS33\_S24}$	24 mA	0.82	0.91	1.04	ns
	$T_{OLVCMOS33\_F2}$	LVCMOS33, Fast, 2 mA	5.46	6.01	6.91	ns
	$T_{OLVCMOS33\_F4}$	4 mA	2.12	2.33	2.68	ns
	$T_{OLVCMOS33\_F6}$	6 mA	0.62	0.68	0.79	ns
	$T_{OLVCMOS33\_F8}$	8 mA	–0.08	–0.09	–0.11	ns
	$T_{OLVCMOS33\_F12}$	12 mA	–0.22	–0.24	–0.28	ns
	$T_{OLVCMOS33\_F16}$	16 mA	–0.42	–0.46	–0.53	ns
	$T_{OLVCMOS33\_F24}$	24 mA	–0.51	–0.56	–0.65	ns
	$T_{OLVCMOS25\_S2}$	LVCMOS25, Slow, 2 mA	8.34	9.17	10.55	ns
	$T_{OLVCMOS25\_S4}$	4 mA	4.69	5.16	5.93	ns
	$T_{OLVCMOS25\_S6}$	6 mA	4.14	4.56	5.24	ns
	$T_{OLVCMOS25\_S8}$	8 mA	3.61	3.97	4.57	ns
	$T_{OLVCMOS25\_S12}$	12 mA	2.51	2.76	3.18	ns
	$T_{OLVCMOS25\_S16}$	16 mA	1.98	2.18	2.51	ns
	$T_{OLVCMOS25\_S24}$	24 mA	1.62	1.78	2.05	ns
	$T_{OLVCMOS25\_F2}$	LVCMOS25, Fast, 2 mA	3.90	4.29	4.94	ns
	$T_{OLVCMOS25\_F4}$	4 mA	0.92	1.01	1.17	ns
	$T_{OLVCMOS25\_F6}$	6 mA	0.41	0.45	0.51	ns
	$T_{OLVCMOS25\_F8}$	8 mA	0.23	0.25	0.29	ns
	$T_{OLVCMOS25\_F12}$	12 mA	–0.13	–0.14	–0.17	ns
	$T_{OLVCMOS25\_F16}$	16 mA	–0.22	–0.24	–0.28	ns
	$T_{OLVCMOS25\_F24}$	24 mA	–0.38	–0.42	–0.48	ns
	$T_{OLVCMOS18\_S2}$	LVCMOS18, Slow, 2 mA	15.71	17.28	19.87	ns
	$T_{OLVCMOS18\_S4}$	4 mA	10.38	11.42	13.13	ns
	$T_{OLVCMOS18\_S6}$	6 mA	7.46	8.21	9.44	ns
	$T_{OLVCMOS18\_S8}$	8 mA	6.92	7.61	8.75	ns
	$T_{OLVCMOS18\_S12}$	12 mA	5.31	5.84	6.71	ns
	$T_{OLVCMOS18\_S16}$	16 mA	5.05	5.56	6.39	ns
	$T_{OLVCMOS18\_F2}$	LVCMOS18, Fast, 2 mA	4.64	5.10	5.87	ns

Table 17: **IOB Output Switching Characteristics Standard Adjustments** ***(Continued)***

			Speed Grade			
Description	**Symbol**	**Standard**	**–6**	**–5**	**–4**	**Units**
	$T_{OLVCMOS18\_F4}$	4 mA	1.48	1.63	1.87	ns
	$T_{OLVCMOS18\_F6}$	6 mA	0.66	0.73	0.83	ns
	$T_{OLVCMOS18\_F8}$	8 mA	0.59	0.65	0.75	ns
	$T_{OLVCMOS18\_F12}$	12 mA	0.12	0.14	0.16	ns
	$T_{OLVCMOS18\_F16}$	16 mA	0.13	0.14	0.16	ns
	$T_{OLVCMOS15\_S2}$	LVCMOS15, Slow, 2 mA	19.67	21.63	24.88	ns
	$T_{OLVCMOS15\_S4}$	4 mA	13.13	14.44	16.61	ns
	$T_{OLVCMOS15\_S6}$	6 mA	12.55	13.80	15.87	ns
	$T_{OLVCMOS15\_S8}$	8 mA	9.54	10.49	12.06	ns
	$T_{OLVCMOS15\_S12}$	12 mA	9.46	10.41	11.97	ns
	$T_{OLVCMOS15\_S16}$	16 mA	8.56	9.41	10.83	ns
	$T_{OLVCMOS15\_F2}$	LVCMOS15, Fast, 2 mA	4.32	4.75	5.46	ns
	$T_{OLVCMOS15\_F4}$	4 mA	1.59	1.75	2.02	ns
	$T_{OLVCMOS15\_F6}$	6 mA	1.21	1.33	1.53	ns
	$T_{OLVCMOS15\_F8}$	8 mA	0.79	0.87	1.00	ns
	$T_{OLVCMOS15\_F12}$	12 mA	0.63	0.69	0.79	ns
	$T_{OLVCMOS15\_F16}$	16 mA	0.59	0.65	0.75	ns
	T_{OLVDCI_33}	LVDCI_33	0.66	0.73	0.83	ns
	T_{OLVDCI_25}	LVDCI_25	0.57	0.62	0.71	ns
	T_{OLVDCI_18}	LVDCI_18	1.40	1.54	1.77	ns
	T_{OLVDCI_15}	LVDCI_15	2.96	3.26	3.75	ns
	$T_{OLVDCI\_DV2\_33}$	LVDCI_DV2_33	0.15	0.17	0.19	ns
	$T_{OLVDCI\_DV2\_25}$	LVDCI_DV2_25	0.31	0.34	0.39	ns
	$T_{OLVDCI\_DV2\_18}$	LVDCI_DV2_18	1.07	1.18	1.35	ns
	$T_{OLVDCI\_DV2\_15}$	LVDCI_DV2_15	2.05	2.25	2.59	ns
	T_{OGTL_DCI}	GTL_DCI	2.82	3.10	3.56	ns
	T_{OGTLP_DCI}	GTLP_DCI	2.03	2.23	2.56	ns
	$T_{OHSTL\_I\_DCI}$	HSTL_I_DCI	0.50	0.55	0.63	ns
	$T_{OHSTL\_II\_DCI}$	HSTL_II_DCI	0.39	0.43	0.50	ns
	$T_{OHSTL\_III\_DCI}$	HSTL_III_DCI	0.15	0.17	0.19	ns
	$T_{OHSTL\_IV\_DCI}$	HSTL_IV_DCI	–0.01	–0.01	–0.02	ns
	$T_{OHSTL\_I\_DCI\_18}$	HSTL_I_DCI_18	0.21	0.23	0.26	ns
	$T_{OHSTL\_II\_DCI\_18}$	HSTL_II_DCI_18	0.94	1.03	1.19	ns
	$T_{OHSTL\_III\_DCI\_18}$	HSTL_III_DCI_18	0.18	0.20	0.23	ns
	$T_{OHSTL\_IV\_DCI\_18}$	HSTL_IV_DCI_18	–0.14	–0.16	–0.18	ns
	$T_{OSSTL2\_I\_DCI}$	SSTL2_I_DCI	0.25	0.28	0.32	ns
	$T_{OSSTL2\_II\_DCI}$	SSTL2_II_DCI	0.05	0.06	0.07	ns
	$T_{OSSTL3\_I\_DCI}$	SSTL3_I_DCI	0.30	0.33	0.38	ns
	$T_{OSSTL3\_II\_DCI}$	SSTL3_II_DCI	0.16	0.18	0.20	ns

Table 18: **Delay Measurement Methodology**

Standard	V_L(1)	V_H(1)	Meas. Point	V_{REF} (Typ)(2)
LVTTL	0	3	1.4	–
LVCMOS33	0	3.3	1.65	–
LVCMOS25	0	2.5	1.25	–
LVCMOS18	0	1.8	0.9	–
LVCMOS15	0	1.5	0.75	–
PCI33_3	Per PCI Specification			–
PCI66_3	Per PCI Specification			–
PCIX33_3	Per PCI–X Specification			–
GTL	V_{REF} – 0.2	V_{REF} + 0.2	V_{REF}	0.80
GTLP	V_{REF} – 0.2	V_{REF} + 0.2	V_{REF}	1.0
HSTL Class I	V_{REF} – 0.5	V_{REF} + 0.5	V_{REF}	0.75
HSTL Class II	V_{REF} – 0.5	V_{REF} + 0.5	V_{REF}	0.75
HSTL Class III	V_{REF} – 0.5	V_{REF} + 0.5	V_{REF}	0.90
HSTL Class IV	V_{REF} – 0.5	V_{REF} + 0.5	V_{REF}	0.90
SSTL3 I & II	V_{REF} – 1.0	V_{REF} + 1.0	V_{REF}	1.5
SSTL2 I & II	V_{REF} – 0.75	V_{REF} + 0.75	V_{REF}	1.25
AGP	V_{REF} – (0.2xV_{CCO})	V_{REF} + (0.2xV_{CCO})	V_{REF}	Per AGP Spec
LVDS_25	1.2 – 0.125	1.2 + 0.125	1.2	
LVDS_33	1.2 – 0.125	1.2 + 0.125	1.2	
LVDSEXT_25	1.2 – 0.125	1.2 + 0.125	1.2	
LVDSEXT_33	1.2 – 0.125	1.2 + 0.125	1.2	
ULVDS_25	0.6 – 0.125	0.6 + 0.125	0.6	
LDT_25	0.6 – 0.125	0.6 + 0.125	0.6	
LVPECL	1.6 –0.3	1.6 + 0.3	1.6	

Notes:

1. Input waveform switches between V_L and V_H.
2. Measurements are made at V_{REF} (Typ), Maximum, and Minimum. Worst-case values are reported.

Table 19: **Standard Capacitive Loads**

Standard	Csl (pF)
LVTTL Fast Slew Rate, 2mA drive	35
LVTTL Fast Slew Rate, 4mA drive	35
LVTTL Fast Slew Rate, 6mA drive	35
LVTTL Fast Slew Rate, 8mA drive	35
LVTTL Fast Slew Rate, 12mA drive	35
LVTTL Fast Slew Rate, 16mA drive	35
LVTTL Fast Slew Rate, 24mA drive	35
LVTTL Slow Slew Rate, 2mA drive	35
LVTTL Slow Slew Rate, 4mA drive	35
LVTTL Slow Slew Rate, 6mA drive	35
LVTTL Slow Slew Rate, 8mA drive	35
LVTTL Slow Slew Rate, 12mA drive	35
LVTTL Slow Slew Rate, 16mA drive	35
LVTTL Slow Slew Rate, 24mA drive	35
LVCMOS33	35
LVCMOS25	35
LVCMOS18	35
LVCMOS15	35
PCI 33MHZ 3.3 V	10
PCI 66 MHz 3.3 V	10
PCI–X 133 MHz 3.3 V	10
GTL	0
GTLP	0
HSTL Class I	20
HSTL Class II	20
HSTL Class III	20
HSTL Class IV	20
SSTL2 Class I	30
SSTL2 Class II	30
SSTL3 Class I	30
SSTL3 Class II	30
AGP	10

Notes:

1. I/O parameter measurements are made with the capacitance values shown above.
2. I/O standard measurements are reflected in the IBIS model information except where the IBIS format precludes it.
3. Use of IBIS models results in a more accurate prediction of the propagation delay:
    a. Model the output in an IBIS simulation into the standard capacitive load.
    b. Record the relative time to the V_{OH} or V_{OL} transition of interest.
    c. Remove the capacitance, and model the actual PCB traces (transmission lines) and actual loads from the appropriate IBIS models for driven devices.
    d. Record the results from the new simulation.
    e. Compare with the capacitance simulation. The increase or decrease in delay from the capacitive load delay simulation should be added or subtracted from the value above to predict the actual delay.

Clock Distribution Switching Characteristics

Table 20: **Clock Distribution Switching Characteristics**

Description	Symbol	Speed Grade			Units
		–6	–5	–4	
Global Clock Buffer I input to O output	T_{GIO}	0.17	0.19	0.21	ns, max

CLB Switching Characteristics

Delays originating at F/G inputs vary slightly according to the input used (see Figure 15). The values listed below are worst-case. Precise values are provided by the timing analyzer.

Table 21: **CLB Switching Characteristics**

Description	Symbol	Speed Grade			Units
		–6	–5	–4	
Combinatorial Delays					
4-input function: F/G inputs to X/Y outputs	T_{ILO}	0.35	0.39	0.44	ns, max
5-input function: F/G inputs to F5 output	T_{IF5}	0.57	0.63	0.72	ns, max
5-input function: F/G inputs to X output	T_{IF5X}	0.76	0.83	0.95	ns, max
FXINA or FXINB inputs to Y output via MUXFX	T_{IFXY}	0.36	0.39	0.45	ns, max
FXINA input to FX output via MUXFX	T_{INAFX}	0.26	0.28	0.32	ns, max
FXINB input to FX output via MUXFX	T_{INBFX}	0.26	0.28	0.32	ns, max
SOPIN input to SOPOUT output via ORCY	T_{SOPSOP}	0.35	0.38	0.44	ns, max
Incremental delay routing through transparent latch to XQ/YQ outputs	T_{IFNCTL}	0.41	0.45	0.51	ns, max
Sequential Delays					
FF Clock CLK to XQ/YQ outputs	T_{CKO}	0.45	0.50	0.57	ns, max
Latch Clock CLK to XQ/YQ outputs	T_{CKLO}	0.54	0.59	0.68	ns, max
Setup and Hold Times Before/After Clock CLK					
BX/BY inputs	T_{DICK}/T_{CKDI}	0.30/–0.07	0.33/–0.08	0.37/–0.09	ns, min
DY inputs	T_{DYCK}/T_{CKDY}	0.30/–0.07	0.33/–0.08	0.37/–0.09	ns, min
DX inputs	T_{DXCK}/T_{CKDX}	0.30/–0.07	0.33/–0.08	0.37/–0.09	ns, min
CE input	T_{CECK}/T_{CKCE}	0.19/–0.06	0.21/–0.07	0.24/–0.08	ns, min
SR/BY inputs (synchronous)	T_{RCK}/T_{CKR}	0.21/–0.02	0.23/–0.03	0.26/–0.03	ns, min
Clock CLK					
Minimum Pulse Width, High	T_{CH}	0.61	0.67	0.77	ns, min
Minimum Pulse Width, Low	T_{CL}	0.61	0.67	0.77	ns, min
Set/Reset					
Minimum Pulse Width, SR/BY inputs	T_{RPW}	0.61	0.67	0.77	ns, min
Delay from SR/BY inputs to XQ/YQ outputs (asynchronous)	T_{RQ}	1.06	1.17	1.34	ns, max
Toggle Frequency (MHz) (for export control)	F_{TOG}	826.45	751.31	653.59	MHz

CLB Distributed RAM Switching Characteristics

Table 22: **CLB Distributed RAM Switching Characteristics**

Description	Symbol	Speed Grade			Units
		–6	–5	–4	
Sequential Delays					
Clock CLK to X/Y outputs (WE active) in 16 x 1 mode	$T_{SHCKO16}$	1.63	1.79	2.05	ns, max
Clock CLK to X/Y outputs (WE active) in 32 x 1 mode	$T_{SHCKO32}$	1.97	2.17	2.49	ns, max
Clock CLK to F5 output	$T_{SHCKOF5}$	1.77	1.94	2.23	ns, max
Setup and Hold Times Before/After Clock CLK					
BX/BY data inputs (DIN)	T_{DS}/T_{DH}	0.53/–0.09	0.58/–0.10	0.67/–0.11	ns, min
F/G address inputs	T_{AS}/T_{AH}	0.40/ 0.00	0.44/ 0.00	0.50/ 0.00	ns, min
CE input (WE)	T_{WES}/T_{WEH}	0.42/–0.01	0.46/–0.01	0.53/–0.01	ns, min
Clock CLK					
Minimum Pulse Width, High	T_{WPH}	0.57	0.63	0.72	ns, min
Minimum Pulse Width, Low	T_{WPL}	0.57	0.63	0.72	ns, min
Minimum clock period to meet address write cycle time	T_{WC}	1.14	1.25	1.44	ns, min

CLB Shift Register Switching Characteristics

Table 23: **CLB Shift Register Switching Characteristics**

Description	Symbol	Speed Grade			Units
		–6	–5	–4	
Sequential Delays					
Clock CLK to X/Y outputs	T_{REG}	2.31	2.54	2.92	ns, max
Clock CLK to X/Y outputs	T_{REG32}	2.65	2.92	3.35	ns, max
Clock CLK to XB output via MC15 LUT output	T_{REGXB}	2.23	2.46	2.82	ns, max
Clock CLK to YB output via MC15 LUT output	T_{REGYB}	2.18	2.40	2.75	ns, max
Clock CLK to Shiftout	T_{CKSH}	1.92	2.11	2.43	ns, max
Clock CLK to F5 output	T_{REGF5}	2.45	2.69	3.09	ns, max
Setup and Hold Times Before/After Clock CLK					
BX/BY data inputs (DIN)	T_{SRLDS}/T_{SRLDH}	0.53/–0.07	0.58/–0.08	0.67/–0.09	ns, min
CE input (WS)	T_{WSS}/T_{WSH}	0.19/–0.06	0.21/–0.07	0.24/–0.08	ns, min
Clock CLK					
Minimum Pulse Width, High	T_{SRPH}	0.57	0.63	0.72	ns, min
Minimum Pulse Width, Low	T_{SRPL}	0.57	0.63	0.72	ns, min

Multiplier Switching Characteristics

Table 24: **Multiplier Switching Characteristics**

Description	Symbol	Speed Grade			Units
		–6	**–5**	**–4**	
Propagation Delay to Output Pin					
Input to Pin35	T_{MULT_P35}	6.49	8.50	10.36	ns, max
Input to Pin34	T_{MULT_P34}	6.36	8.33	10.14	ns, max
Input to Pin33	T_{MULT_P33}	6.23	8.16	9.92	ns, max
Input to Pin32	T_{MULT_P32}	6.10	7.99	9.70	ns, max
Input to Pin31	T_{MULT_P31}	5.97	7.82	9.48	ns, max
Input to Pin30	T_{MULT_P30}	5.84	7.65	9.26	ns, max
Input to Pin29	T_{MULT_P29}	5.71	7.48	9.04	ns, max
Input to Pin28	T_{MULT_P28}	5.58	7.31	8.82	ns, max
Input to Pin27	T_{MULT_P27}	5.45	7.14	8.60	ns, max
Input to Pin26	T_{MULT_P26}	5.32	6.97	8.38	ns, max
Input to Pin25	T_{MULT_P25}	5.19	6.80	8.16	ns, max
Input to Pin24	T_{MULT_P24}	5.06	6.63	7.94	ns, max
Input to Pin23	T_{MULT_P23}	4.93	6.46	7.72	ns, max
Input to Pin22	T_{MULT_P22}	4.80	6.29	7.50	ns, max
Input to Pin21	T_{MULT_P21}	4.67	6.12	7.28	ns, max
Input to Pin20	T_{MULT_P20}	4.54	5.95	7.06	ns, max
Input to Pin19	T_{MULT_P19}	4.41	5.78	6.84	ns, max
Input to Pin18	T_{MULT_P18}	4.28	5.61	6.62	ns, max
Input to Pin17	T_{MULT_P17}	4.15	5.44	6.40	ns, max
Input to Pin16	T_{MULT_P16}	4.02	5.27	6.18	ns, max
Input to Pin15	T_{MULT_P15}	3.89	5.10	5.96	ns, max
Input to Pin14	T_{MULT_P14}	3.76	4.93	5.74	ns, max
Input to Pin13	T_{MULT_P13}	3.63	4.76	5.52	ns, max
Input to Pin12	T_{MULT_P12}	3.50	4.59	5.30	ns, max
Input to Pin11	T_{MULT_P11}	3.37	4.42	5.08	ns, max
Input to Pin10	T_{MULT_P10}	3.24	4.25	4.86	ns, max
Input to Pin9	T_{MULT_P9}	3.11	4.08	4.64	ns, max
Input to Pin8	T_{MULT_P8}	2.98	3.91	4.42	ns, max
Input to Pin7	T_{MULT_P7}	2.85	3.74	4.20	ns, max
Input to Pin6	T_{MULT_P6}	2.72	3.57	3.98	ns, max
Input to Pin5	T_{MULT_P5}	2.59	3.40	3.76	ns, max
Input to Pin4	T_{MULT_P4}	2.46	3.23	3.54	ns, max
Input to Pin3	T_{MULT_P3}	2.33	3.06	3.32	ns, max
Input to Pin2	T_{MULT_P2}	2.20	2.89	3.10	ns, max
Input to Pin1	T_{MULT_P1}	2.07	2.72	2.88	ns, max
Input to Pin0	T_{MULT_P0}	1.94	2.55	2.66	ns, max

Table 25: **Pipelined Multiplier Switching Characteristics**

Description	Symbol	Speed Grade			Units
		–6	**–5**	**–4**	
Setup and Hold Times Before/After Clock					
Data Inputs	$T_{MULIDCK}/T_{MULCKID}$	3.00/0.000	3.45/0.000	3.89/0.000	ns, max
Clock Enable	$T_{MULIDCK\_CE}/T_{MULCKID\_CE}$	0.72/0.000	0.80/0.000	0.86/0.000	ns, max
Reset	$T_{MULIDCK\_RST}/T_{MULCKID\_RST}$	0.72/0.000	0.80/0.000	0.86/0.000	ns, max
Clock to Output Pin					
Clock to Pin35	T_{MULTCK_P35}	4.11	6.91	8.11	ns, max
Clock to Pin34	T_{MULTCK_P34}	3.98	6.75	7.92	ns, max
Clock to Pin33	T_{MULTCK_P33}	3.86	6.59	7.74	ns, max
Clock to Pin32	T_{MULTCK_P32}	3.73	6.43	7.55	ns, max
Clock to Pin31	T_{MULTCK_P31}	3.60	6.27	7.37	ns, max
Clock to Pin30	T_{MULTCK_P30}	3.47	6.11	7.18	ns, max
Clock to Pin29	T_{MULTCK_P29}	3.34	5.95	6.99	ns, max
Clock to Pin28	T_{MULTCK_P28}	3.22	5.79	6.81	ns, max
Clock to Pin27	T_{MULTCK_P27}	3.09	5.63	6.62	ns, max
Clock to Pin26	T_{MULTCK_P26}	2.96	5.47	6.44	ns, max
Clock to Pin25	T_{MULTCK_P25}	2.83	5.31	6.25	ns, max
Clock to Pin24	T_{MULTCK_P24}	2.70	5.15	6.06	ns, max
Clock to Pin23	T_{MULTCK_P23}	2.58	4.99	5.88	ns, max
Clock to Pin22	T_{MULTCK_P22}	2.45	4.83	5.69	ns, max
Clock to Pin21	T_{MULTCK_P21}	2.32	4.67	5.51	ns, max
Clock to Pin20	T_{MULTCK_P20}	2.19	4.51	5.32	ns, max
Clock to Pin19	T_{MULTCK_P19}	2.06	4.35	5.13	ns, max
Clock to Pin18	T_{MULTCK_P18}	1.94	4.19	4.95	ns, max
Clock to Pin17	T_{MULTCK_P17}	1.81	4.03	4.76	ns, max
Clock to Pin16	T_{MULTCK_P16}	1.68	3.87	4.58	ns, max
Clock to Pin15	T_{MULTCK_P15}	1.68	3.71	4.39	ns, max
Clock to Pin14	T_{MULTCK_P14}	1.68	3.55	4.20	ns, max
Clock to Pin13	T_{MULTCK_P13}	1.68	3.39	4.02	ns, max
Clock to Pin12	T_{MULTCK_P12}	1.68	3.23	3.83	ns, max
Clock to Pin11	T_{MULTCK_P11}	1.68	3.07	3.65	ns, max
Clock to Pin10	T_{MULTCK_P10}	1.68	2.91	3.46	ns, max
Clock to Pin9	T_{MULTCK_P9}	1.68	2.75	3.27	ns, max
Clock to Pin8	T_{MULTCK_P8}	1.68	2.59	3.09	ns, max
Clock to Pin7	T_{MULTCK_P7}	1.68	2.43	2.90	ns, max
Clock to Pin6	T_{MULTCK_P6}	1.68	2.27	2.72	ns, max
Clock to Pin5	T_{MULTCK_P5}	1.68	2.11	2.53	ns, max
Clock to Pin4	T_{MULTCK_P4}	1.68	1.95	2.34	ns, max
Clock to Pin3	T_{MULTCK_P3}	1.68	1.79	2.16	ns, max
Clock to Pin2	T_{MULTCK_P2}	1.68	1.63	1.97	ns, max
Clock to Pin1	T_{MULTCK_P1}	1.68	1.47	1.79	ns, max
Clock to Pin0	T_{MULTCK_P0}	1.68	1.31	1.60	ns, max

Block SelectRAM Switching Characteristics

Table 26: **Block SelectRAM Switching Characteristics**

Description	Symbol	Speed Grade			Units
		–6	–5	–4	
Sequential Delays					
Clock CLK to DOUT output	T_{BCKO}	2.10	2.31	2.65	ns, max
Setup and Hold Times Before Clock CLK					
ADDR inputs	T_{BACK}/T_{BCKA}	0.29/ 0.00	0.32/ 0.00	0.36/ 0.00	ns, min
DIN inputs	T_{BDCK}/T_{BCKD}	0.29/ 0.00	0.32/ 0.00	0.36/ 0.00	ns, min
EN input	T_{BECK}/T_{BCKE}	0.95/–0.46	1.04/–0.50	1.20/–0.58	ns, min
RST input	T_{BRCK}/T_{BCKR}	1.31/–0.71	1.44/–0.78	1.65/–0.90	ns, min
WEN input	T_{BWCK}/T_{BCKW}	0.57/–0.19	0.63/–0.21	0.72/–0.25	ns, min
Clock CLK					
Minimum Pulse Width, High	T_{BPWH}	1.17	1.29	1.48	ns, min
Minimum Pulse Width, Low	T_{BPWL}	1.17	1.29	1.48	ns, min

TBUF Switching Characteristics

Table 27: **TBUF Switching Characteristics**

Description	Symbol	Speed Grade			Units
		–6	–5	–4	
Combinatorial Delays					
IN input to OUT output	T_{IO}	0.23	0.25	0.29	ns, max
TRI input to OUT output high-impedance	T_{OFF}	0.44	0.48	0.55	ns, max
TRI input to valid data on OUT output	T_{ON}	0.44	0.48	0.55	ns, max

JTAG Test Access Port Switching Characteristics

Table 28: **JTAG Test Access Port Switching Characteristics**

Description	Symbol		Units
TMS and TDI Setup times before TCK	T_{TAPTK}	5.5	ns, min
TMS and TDI Hold times after TCK	T_{TCKTAP}	0.0	ns, min
Output delay from clock TCK to output TDO	T_{TCKTDO}	10.0	ns, max
Maximum TCK clock frequency	F_{TCK}	33	MHz, max

Virtex-II Pin-to-Pin Output Parameter Guidelines

All devices are 100% functionally tested. Listed below are representative values for typical pin locations and normal clock loading. Values are expressed in nanoseconds unless otherwise noted.

Global Clock Input to Output Delay for LVTTL, 12 mA, Fast Slew Rate, *With* DCM

Table 29: **Global Clock Input to Output Delay for LVTTL, 12 mA, Fast Slew Rate, *With* DCM**

Description	Symbol	Device	Speed Grade			Units
			–6	–5	–4	
LVTTL Global Clock Input to Output Delay using Output Flip-flop, 12 mA, Fast Slew Rate, *with* DCM. For data *output* with different standards, adjust the delays with the values shown in **IOB Output Switching Characteristics Standard Adjustments**, page 97.						
Global Clock and OFF with DCM	$T_{ICKOFDCM}$	2v40	2.19	2.40	2.76	ns
		2v80	2.19	2.40	2.76	ns
		2v250	2.19	2.40	2.76	ns
		2v500	2.19	2.40	2.76	ns
		2v1000	2.19	2.40	2.76	ns
		2v1500	2.19	2.40	2.76	ns
		2v2000	2.19	2.40	2.76	ns
		2v3000	2.28	2.50	2.88	ns
		2v4000	2.28	2.50	2.88	ns
		2v6000	2.73	3.00	3.45	ns
		2v8000	TBD	TBD	TBD	ns

Notes:

1. Listed above are representative values where one global clock input drives one vertical clock line in each accessible column, and where all accessible IOB and CLB flip-flops are clocked by the global clock net.
2. Output timing is measured at 50% V_{CC} threshold with 35 pF external capacitive load. For other I/O standards and different loads, see Table 18.
3. DCM output jitter is already included in the timing calculation.

Global Clock Input to Output Delay for LVTTL, 12 mA, Fast Slew Rate, *Without* DCM

Table 30: **Global Clock Input to Output Delay for LVTTL, 12 mA, Fast Slew Rate, *Without* DCM**

Description	Symbol	Device	Speed Grade			Units
			–6	–5	–4	
LVTTL Global Clock Input to Output Delay using Output Flip-flop, 12 mA, Fast Slew Rate, *without* DCM. For data *output* with different standards, adjust the delays with the values shown in **IOB Output Switching Characteristics Standard Adjustments**, page 97.						
Global Clock and OFF without DCM	T_{ICKOF}	2v40	4.28	4.70	4.98	ns
		2v80	4.28	4.70	4.98	ns
		2v250	4.50	5.00	5.75	ns
		2v500	4.50	5.00	5.75	ns
		2v1000	5.10	5.40	5.90	ns
		2v1500	5.10	5.40	5.90	ns
		2v2000	5.20	5.55	6.10	ns
		2v3000	5.20	5.70	6.55	ns
		2v4000	5.50	6.00	6.90	ns
		2v6000	5.73	6.30	7.22	ns
		2v8000	TBD	TBD	TBD	ns

Notes:

1. Listed above are representative values where one global clock input drives one vertical clock line in each accessible column, and where all accessible IOB and CLB flip-flops are clocked by the global clock net.
2. Output timing is measured at 50% V_{CC} threshold with 35 pF external capacitive load. For other I/O standards and different loads, see Table 18.

Virtex-II Pin-to-Pin Input Parameter Guidelines

All devices are 100% functionally tested. Listed below are representative values for typical pin locations and normal clock loading. Values are expressed in nanoseconds unless otherwise noted.

Global Clock Setup and Hold for LVTTL Standard, *With* DCM

Table 31: **Global Clock Setup and Hold for LVTTL Standard, *With* DCM**

Description	Symbol	Device	Speed Grade			Units
			–6	**–5**	**–4**	
Input Setup and Hold Time Relative to Global Clock Input Signal for LVTTL Standard. For data input with different standards, adjust the setup time delay by the values shown in **IOB Input Switching Characteristics Standard Adjustments**, page 94.						
No Delay Global Clock and IFF with DCM	T_{PSDCM}/T_{PHDCM}	2v40	1.60/–0.90	1.60/–0.90	1.84/–0.76	ns
		2v80	1.60/–0.90	1.60/–0.90	1.84/–0.76	ns
		2v250	1.60/–0.90	1.60/–0.90	1.84/–0.76	ns
		2v500	1.60/–0.90	1.60/–0.90	1.84/–0.76	ns
		2v1000	1.60/–0.90	1.60/–0.90	1.84/–0.76	ns
		2v1500	1.60/–0.90	1.60/–0.90	1.84/–0.76	ns
		2v2000	1.70/–0.90	1.70/–0.90	1.96/–0.76	ns
		2v3000	1.70/–0.90	1.70/–0.90	1.96/–0.76	ns
		2v4000	1.70/–0.90	1.70/–0.90	1.96/–0.76	ns
		2v6000	1.70/–0.90	1.70/–0.90	1.96/–0.76	ns
		2v8000	TBD	TBD	TBD	ns

Notes:

1. IFF = Input Flip-Flop or Latch
2. Setup time is measured relative to the Global Clock input signal with the fastest route and the lightest load. Hold time is measured relative to the Global Clock input signal with the slowest route and heaviest load.
3. DCM output jitter is already included in the timing calculation.

Global Clock Setup and Hold for LVTTL Standard, *Without* DCM

Table 32: **Global Clock Setup and Hold for LVTTL Standard, *Without* DCM**

Description	Symbol	Device	Speed Grade			Units
			–6	–5	–4	
Input Setup and Hold Time Relative to Global Clock Input Signal for LVTTL Standard. For data input with different standards, adjust the setup time delay by the values shown in **IOB Input Switching Characteristics Standard Adjustments**, page 94.						
Full Delay Global Clock and IFF without DCM	T_{PSFD}/T_{PHFD}	2v40	1.92/ 0.00	1.92/ 0.00	2.21/ 0.00	ns
		2v80	1.92/ 0.00	1.92/ 0.00	2.21/ 0.00	ns
		2v250	1.92/ 0.00	1.92/ 0.00	2.21/ 0.00	ns
		2v500	1.92/ 0.00	1.92/ 0.00	2.21/ 0.00	ns
		2v1000	1.92/ 0.00	1.92/ 0.00	2.21/ 0.00	ns
		2v1500	1.92/ 0.00	1.92/ 0.00	2.21/ 0.00	ns
		2v2000	1.92/ 0.00	1.92/ 0.00	2.21/ 0.00	ns
		2v3000	1.92/ 0.00	1.92/ 0.00	2.21/ 0.00	ns
		2v4000	1.92/ 0.00	1.92/ 0.00	2.21/ 0.00	ns
		2v6000	1.92/ 0.46	1.92/ 0.50	2.21/ 0.50	ns
		2v8000	TBD	TBD	TBD	ns

Notes:

1. IFF = Input Flip-Flop or Latch
2. Setup time is measured relative to the Global Clock input signal with the fastest route and the lightest load. Hold time is measured relative to the Global Clock input signal with the slowest route and heaviest load.

DCM Timing Parameters

Testing of switching parameters is modeled after testing methods specified by MIL-M-38510/605; all devices are 100% functionally tested. Because of the difficulty in directly measuring many internal timing parameters, those parameters are derived from benchmark timing patterns. The following guidelines reflect worst-case values across the recommended operating conditions. All output jitter and phase specifications are determined through statistical measurement at the package pins.

Operating Frequency Ranges

Table 33: **Operating Frequency Ranges**

Description	Symbol	Constraints	Speed Grade			Units
			–6	–5	–4	
Output Clocks (Low Frequency Mode)						
CLK0, CLK90, CLK180, CLK270	CLKOUT_FREQ_1X_LF_Min		24.00	24.00	24.00	MHz
	CLKOUT_FREQ_1X_LF_Max		230.00	210.00	180.00	MHz
CLK2X, CLK2X180	CLKOUT_FREQ_2X_LF_Min		48.00	48.00	48.00	MHz
	CLKOUT_FREQ_2X_LF_Max		450.00	420.00	360.00	MHz
CLKDV	CLKOUT_FREQ_DV_LF_Min		1.50	1.50	1.50	MHz
	CLKOUT_FREQ_DV_LF_Max		150.00	140.00	120.00	MHz
CLKFX, CLKFX180	CLKOUT_FREQ_FX_LF_Min		24.00	24.00	24.00	MHz
	CLKOUT_FREQ_FX_LF_Max		260.00	240.00	210.00	MHz
Input Clocks (Low Frequency Mode)						
CLKIN (using DLL outputs[1])	CLKIN_FREQ_DLL_LF_Min		24.00	24.00	24.00	MHz
	CLKIN_FREQ_DLL_LF_Max		230.00	210.00	180.00	MHz
CLKIN (using CLKFX outputs)	CLKIN_FREQ_FX_LF_Min		1.00	1.00	1.00	MHz
	CLKIN_FREQ_FX_LF_Max		260.00	240.00	210.00	MHz
PSCLK	PSCLK_FREQ_LF_Min		0.01	0.01	0.01	MHz
	PSCLK_FREQ_LF_Max		450.00	420.00	360.00	MHz
Output Clocks (High Frequency Mode)						
CLK0, CLK180	CLKOUT_FREQ_1X_HF_Min		48.00	48.00	48.00	MHz
	CLKOUT_FREQ_1X_HF_Max		450.00	420.00	360.00	MHz
CLKDV	CLKOUT_FREQ_DV_HF_Min		3.00	3.00	3.00	MHz
	CLKOUT_FREQ_DV_HF_Max		300.00	280.00	240.00	MHz
CLKFX, CLKFX180	CLKOUT_FREQ_FX_HF_Min		210.00	210.00	210.00	MHz
	CLKOUT_FREQ_FX_HF_Max		350.00	320.00	270.00	MHz
Input Clocks (High Frequency Mode)						
CLKIN (using DLL outputs[1])	CLKIN_FREQ_DLL_HF_Min		48.00	48.00	48.00	MHz
	CLKIN_FREQ_DLL_HF_Max		450.00	420.00	360.00	MHz
CLKIN (using CLKFX outputs)	CLKIN_FRQ_FX_HF_Min		50.00	50.00	50.00	MHz
	CLKIN_FRQ_FX_HF_Max		350.00	320.00	270.00	MHz
PSCLK	PSCLK_FREQ_HF_Min		0.01	0.01	0.01	MHz
	PSCLK_FREQ_HF_Max		450.00	420.00	360.00	MHz

Notes:

1. “”DLL outputs” is used here to describe the outputs: CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, and CLKDV.

Input Clock Tolerances

Table 34: **Input Clock Tolerances**

Description	Symbol	Constraints F_{CLKIN}	Speed Grade						Units
			–6		–5		–4		
			Min	Max	Min	Max	Min	Max	
Input Clock Low/high Pulse Width									
PSCLK	PSCLK_PULSE	< 1MHz	25.00		25.00		25.00		ns
CLKIN(2)	CLKIN_PULSE	1 – 10 MHz	25.00		25.00		25.00		ns
		10 – 25 MHz	10.00		10.00		10.00		ns
		25 – 50 MHz	5.00		5.00		5.00		ns
		50 – 100 MHz	3.00		3.00		3.00		ns
		100 – 150 MHz	2.40		2.40		2.40		ns
		150 – 200 MHz	2.00		2.00		2.00		ns
		200 – 250 MHz	1.80		1.80		1.80		ns
		250 – 300 MHz	1.50		1.50		1.50		ns
		300 – 350 MHz	1.30		1.30		1.30		ns
		350 – 400 MHz	1.15		1.15		1.15		ns
		> 400 MHz	1.05		1.05		1.05		ns
Input Clock Cycle-Cycle Jitter (Low Frequency Mode)									
CLKIN (using DLL outputs[1])	CLKIN_CYC_JITT_DLL_LF			±300		±300		±300	ps
CLKIN (using CLKFX outputs)	CLKIN_CYC_JITT_FX_LF			±300		±300		±300	ps
Input Clock Cycle-Cycle Jitter (High Frequency Mode)									
CLKIN (using DLL outputs[1])	CLKIN_CYC_JITT_DLL_HF			±150		±150		±150	ps
CLKIN (using CLKFX outputs)	CLKIN_CYC_JITT_FX_HF			±150		±150		±150	ps
Input Clock Period Jitter (Low Frequency Mode)									
CLKIN (using DLL outputs[1])	CLKIN_PER_JITT_DLL_LF			±1		±1		±1	ns
CLKIN (using CLKFX outputs)	CLKIN_PER_JITT_FX_LF			±1		±1		±1	ns
Input Clock Period Jitter (High Frequency Mode)									
CLKIN (using DLL outputs[1])	CLKIN_PER_JITT_DLL_HF			±1		±1		±1	ns
CLKIN (using CLKFX outputs)	CLKIN_PER_JITT_FX_HF			±1		±1		±1	ns
Feedback Clock Path Delay Variation									
CLKFB off-chip feedback	CLKFB_DELAY_VAR_EXT			±1		±1		±1	ns

Notes:

1. “”DLL outputs” is used here to describe the outputs: CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, and CLKDV.
2. Specification also applies to PSCLK.

Output Clock Jitter

Table 35: **Output Clock Jitter**

Description	Symbol	Constraints	Speed Grade			Units
			–6	–5	–4	
Clock Synthesis Period Jitter						
CLK0	CLKOUT_PER_JITT_0		±100	±100	±100	ps
CLK90	CLKOUT_PER_JITT_90		±150	±150	±150	ps
CLK180	CLKOUT_PER_JITT_180		±150	±150	±150	ps
CLK270	CLKOUT_PER_JITT_270		±150	±150	±150	ps
CLK2X, CLK2X180	CLKOUT_PER_JITT_2X		±200	±200	±200	ps
CLKDV (integer division)	CLKOUT_PER_JITT_DV1		±150	±150	±150	ps
CLKDV (non-integer division)	CLKOUT_PER_JITT_DV2		±300	±300	±300	ps
CLKFX, CLKFX180	CLKOUT_PER_JITT_FX		Note 1	Note 1	Note 1	ps

Notes:

1. Values for this parameter are available on www.xilinx.com.

Output Clock Phase Alignment

Table 36: **Output Clock Phase Alignment**

Description	Symbol	Constraints	Speed Grade			Units
			–6	–5	–4	
Phase Offset Between CLKIN and CLKFB						
CLKIN/CLKFB	CLKIN_CLKFB_PHASE		±50	±50	±50	ps
Phase Offset Between Any DCM Outputs						
All CLK* outputs	CLKOUT_PHASE		±140	±140	±140	ps
Duty Cycle Precision						
DLL outputs(1)	CLKOUT_DUTY_CYCLE_DLL		±150	±150	±150	ps
CLKFX outputs	CLKOUT_DUTY_CYCLE_FX		±100	±100	±100	ps

Notes:

1. “”DLL outputs” is used here to describe the outputs: CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, and CLKDV.
2. Specification also applies to PSCLK.

Miscellaneous Timing Parameters

Table 37: **Miscellaneous Timing Parameters**

Description	Symbol	Constraints F_{CLKIN}	Speed Grade			Units
			–6	–5	–4	
Time Required to Achieve LOCK						
Using DLL outputs(1)	LOCK_DLL					
	LOCK_DLL_60	> 60MHz	20.0	20.0	20.0	µs
	LOCK_DLL_50_60	50 - 60 MHz	25.0	25.0	25.0	µs
	LOCK_DLL_40_50	40 - 50 MHz	50.0	50.0	50.0	µs
	LOCK_DLL_30_40	30 - 40 MHz	90.0	90.0	90.0	µs
	LOCK_DLL_24_30	24 - 30 MHz	120.0	120.0	120.0	µs
Using CLKFX outputs	LOCK_FX_MIN		10.0	10.0	10.0	ms
	LOCK_FX_MAX		10.0	10.0	10.0	ms
Additional lock time with fine-phase shifting	LOCK_DLL_FINE_SHIFT		50.0	50.0	50.0	µs
Fine-Phase Shifting						
Absolute shifting range	FINE_SHIFT_RANGE		10.0	10.0	10.0	ns
Delay Lines						
Tap delay resolution	DCM_TAP_MIN		30.0	30.0	30.0	ps
	DCM_TAP_MAX		60.0	60.0	60.0	ps

Notes:

1. “”DLL outputs” is used here to describe the outputs: CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, and CLKDV.
2. Specification also applies to PSCLK.

Frequency Synthesis

Table 38: **Frequency Synthesis**

Attribute	Min	Max
CLKFX_MULTIPLY	2	32
CLKFX_DIVIDE	1	32

Parameter Cross Reference

Table 39: **Parameter Cross Reference**

Libraries Guide	Data Sheet
DLL_CLKOUT_{MIN|MAX}_LF	CLKOUT_FREQ_{1X|2X|DV}_LF
DFS_CLKOUT_{MIN|MAX}_LF	CLKOUT_FREQ_FX_LF
DLL_CLKIN_{MIN|MAX}_LF	CLKIN_FREQ_DLL_LF
DFS_CLKIN_{MIN|MAX}_LF	CLKIN_FREQ_FX_LF
DLL_CLKOUT_{MIN|MAX}_HF	CLKOUT_FREQ_{1X|DV}_HF
DFS_CLKOUT_{MIN|MAX}_HF	CLKOUT_FREQ_FX_HF
DLL_CLKIN_{MIN|MAX}_HF	CLKIN_FREQ_DLL_HF
DFS_CLKIN_{MIN|MAX}_HF	CLKIN_FREQ_FX_HF

Revision History

This section records the change history for this module of the data sheet.

Date	Version	Revision
11/07/00	1.0	Early access draft.
12/06/00	1.1	Initial release.
01/15/01	1.2	Added values to the tables in the **Virtex-II Performance Characteristics** and **Virtex-II Switching Characteristics** sections.
01/25/01	1.3	• The data sheet was divided into four modules (per the current style standard). • Updated values in the **Virtex-II Performance Characteristics** and **Virtex-II Switching Characteristics** tables. • Table 18, "Delay Measurement Methodology," on page 100
04/23/01	1.5	• Updated values in the **Virtex-II Performance Characteristics** and **Virtex-II Switching Characteristics** tables. • Added T_{REG32} symbol to Table 23. • Skipped v1.4 to sync with other modules. Reverted to traditional double-column format.
07/30/01	1.6	• Updated values in the **Virtex-II Performance Characteristics** and **Virtex-II Switching Characteristics** tables. • Added values to the **Virtex-II Pin-to-Pin Output Parameter Guidelines** and **Virtex-II Pin-to-Pin Input Parameter Guidelines** tables. • Added **Frequency Synthesis** table.
10/02/01	1.7	• Updated values in the **Virtex-II Performance Characteristics** and **Virtex-II Switching Characteristics** tables. • Updated the speed grade designations used in data sheets, and added Table 13, which shows the current speed grade designation for each device.
10/05/01	1.8	• Corrected the speed grade designation for the XC2V1000 device in Table 13.
10/12/01	1.9	• Updated values in the **Virtex-II Performance Characteristics** and **Virtex-II Switching Characteristics** tables.
11/28/01	2.0	• Updated values in Table 3, Table 4, Table 5, **Virtex-II Performance Characteristics**, and **Virtex-II Switching Characteristics** tables.

Virtex-II Data Sheet

The Virtex-II Data Sheet contains the following modules:

- DS031-1, Virtex-II 1.5V FPGAs: **Introduction and Ordering Information (Module 1)**
- DS031-2, Virtex-II 1.5V FPGAs: **Functional Description (Module 2)**
- DS031-3, Virtex-II 1.5V FPGAs: DC and Switching Characteristics (Module 3)
- DS031-4, Virtex-II 1.5V FPGAs: **Pinout Tables (Module 4)**

Part II: Virtex-II User Guide

This section contains information on how to configure and use Virtex-II devices. The following topics are covered:

Chapter 1

Timing Models

Summary

The following topics are covered in this chapter:

- CLB / Slice Timing Model
- Block SelectRAM Timing Model
- Embedded Multiplier Timing Model
- IOB Timing Model
- Pin-to-Pin Timing Model
- Digital Clock Manager Timing Model

Introduction

Due to the large size and complexity of Virtex-II FPGAs, understanding the timing associated with the various paths and functional elements has become a difficult and important problem. Although it is not necessary to understand the various timing parameters in order to implement most designs using Xilinx, Inc. software, a thorough timing model can assist advanced users in analyzing critical paths, or planning speed-sensitive designs.

The Timing Model chapter is broken up into five sections consisting of three basic components:

- Functional Element Diagram - basic architectural schematic illustrating pins and connections.
- Timing Parameters - **Virtex-II Data Sheet** timing parameter definitions.
- Timing Diagram - illustrates functional element timing parameters relative to each other.

This chapter was written with the Xilinx Timing Analyzer software (TRCE) in mind. All pin names, parameter names, and paths are consistent with Post Route Timing and Pre-Route Static Timing reports. Use the models in this chapter in conjunction with both the Timing Analyzer software and the section on switching characteristics in the **Virtex-II Data Sheet**. Most of the timing parameters found in the section on switching characteristics are described in this chapter.

CLB / Slice Timing Model

Introduction

This section describes all timing parameters reported in the **Virtex-II Data Sheet** that are associated with slices and Configurable Logic Blocks (CLBs). It consists of three parts corresponding to their respective (switching characteristics) sections in the data sheet:

- General Slice Timing Model and Parameters (CLB Switching Characteristics)
- Slice Distributed RAM Timing Model and Parameters (CLB Distributed RAM Switching Characteristics)
- Slice SRL Timing Model and Parameters (CLB SRL Switching Characteristics)

General Slice Timing Model and Parameters

Figure 1-1 illustrates the details of a Virtex-II slice.

Note: Some elements of the Virtex-II slice have been omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

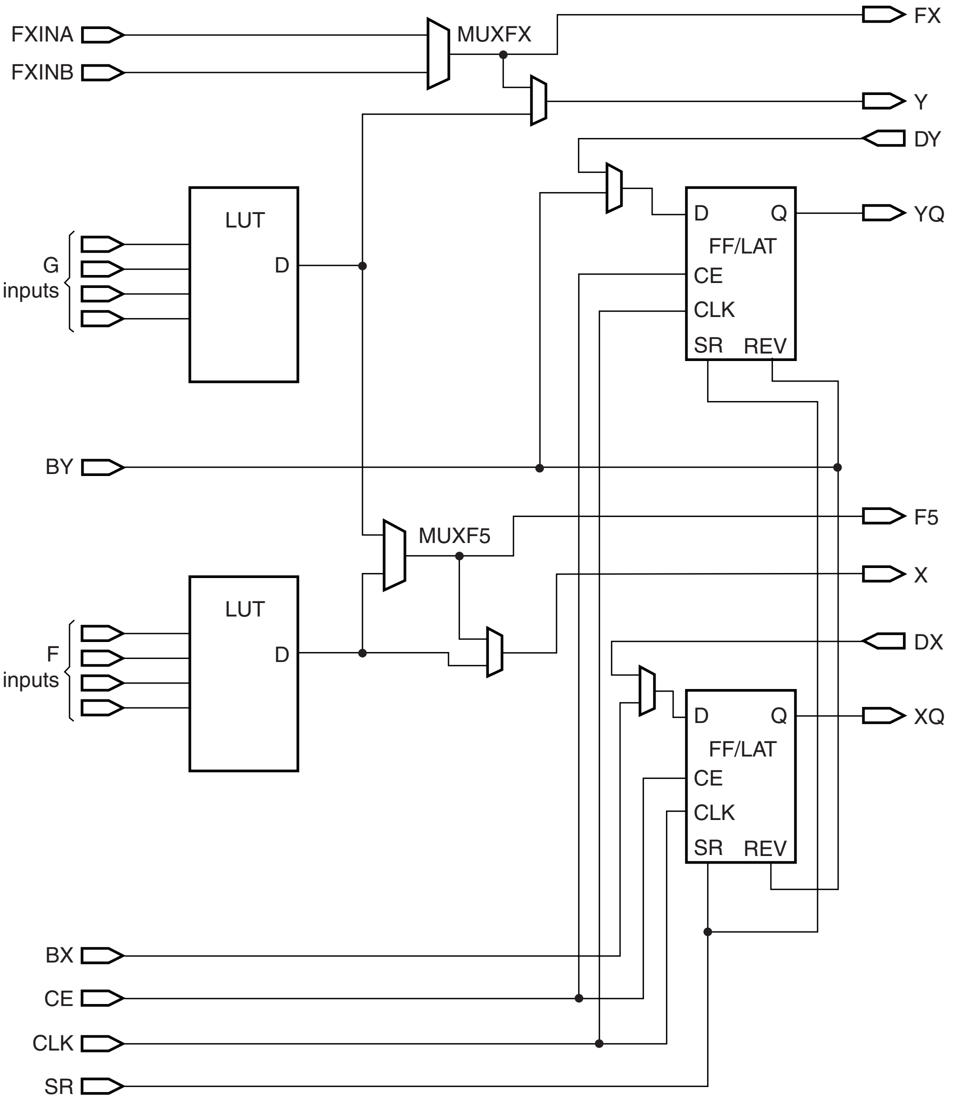


Figure 1-1: **General Slice Diagram**

Timing Parameters

Parameter	Function	Control Signal	Description
Combinatorial Delays			
T_{ILO}	F/G inputs to X/Y outputs		Propagation delay from the F/G inputs of the slice, through the look-up tables (LUTs), to the X/Y outputs of the slice.
T_{IF5}	F/G inputs to F5 output		Propagation delay from the F/G inputs of the slice, through the LUTs and MUXF5 to the F5 output of the slice.
T_{IF5X}	F/G inputs to X output		Propagation delay from the F/G inputs of the slice, through the LUTs and MUXF5 to the X output of the slice.
T_{IFXY}	FXINA/FXINB inputs to Y output		Propagation delay from the FXINA/FXINB inputs, through MUXFX to the Y output of the slice.
T_{IFNCTL}	Transparent Latch input to XQ/YQ outputs		Incremental delay through a transparent latch to XQ/YQ outputs.
Sequential Delays			
T_{CKO}	FF Clock (CLK) to XQ/YQ outputs		Time after the clock that data is stable at the XQ/YQ outputs of the slice sequential elements (configured as a flip-flop).
T_{CKLO}	Latch Clock (CLK) to XQ/YQ outputs		Time after the clock that data is stable at the XQ/YQ outputs of the slice sequential elements (configured as a latch).
Setup and Hold for Slice Sequential Elements			
T_{xxCK} = Setup time (before clock edge) T_{CKxx} = Hold time (after clock edge)			The following descriptions are for setup times only.
T_{DICK}/T_{CKDI}	BX/BY inputs		Time before Clock (CLK) that data from the BX or BY inputs of the slice must be stable at the D-input of the slice sequential elements (configured as a flip-flop).
T_{DYCK}/T_{CKDY}	DY input		Time before Clock (CLK) that data from the DY input of the slice must be stable at the D-input of the slice sequential elements (configured as a flip-flop).
T_{DXCK}/T_{CKDX}	DX input		Time before Clock (CLK) that data from the DX input of the slice must be stable at the D-input of the slice sequential elements (configured as a flip-flop).
T_{CECK}/T_{CKCE}	CE input		Time before Clock (CLK) that the CE (Clock Enable) input of the slice must be stable at the CE-input of the slice sequential elements (configured as a flip-flop).

Parameter	Function	Control Signal	Description
T_{RCK}/T_{CKR}	SR/BY inputs		Time before CLK that the SR (Set/Reset) and the BY (Rev) inputs of the slice must be stable at the SR/Rev-inputs of the slice sequential elements (configured as a flip-flop). Synchronous set/reset only.
Clock CLK			
T_{CH}			Minimum Pulse Width, High.
T_{CL}			Minimum Pulse Width, Low.
Set/Reset			
T_{RPW}			Minimum Pulse Width for the SR (Set/Reset) and BY (Rev) pins.
T_{RQ}			Propagation delay for an asynchronous Set/Reset of the slice sequential elements. From SR/BY inputs to XQ/YQ outputs.
F_{TOG}			Toggle Frequency - Maximum Frequency that a CLB flip-flop can be clocked: $1/(T_{CH}+T_{CL})$

Figure 1-2 illustrates general timing characteristics of a Virtex-II slice.

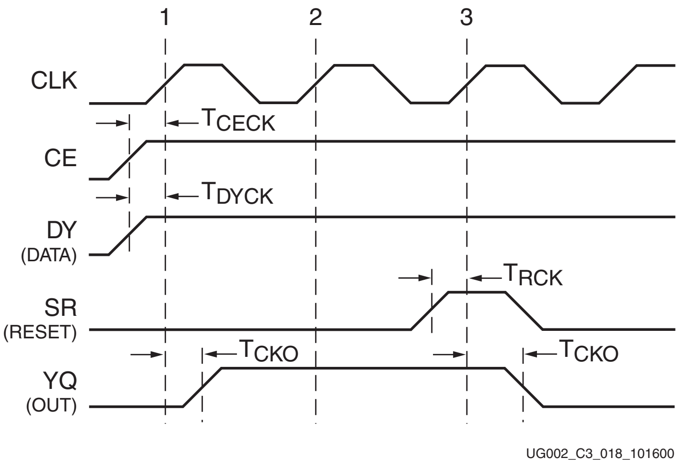


Figure 1-2: **General Slice Timing Diagram**

- At time T_{CECK} before Clock Event 1, the Clock-Enable signal becomes valid-high at the CE input of the slice register.
- At time T_{DYCK} before Clock Event 1, data from the DY input becomes valid-high at the D input of the slice register and is reflected on the YQ pin at time T_{CKO} after Clock Event 1*.
- At time T_{RCK} before Clock Event 3, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting the slice register, and this is reflected on the YQ pin at time T_{CKO} after Clock Event 3.

  * NOTE: In most cases software uses the DX/DY inputs to route data to the slice registers when at all possible. This is the fastest path to the slice registers and saves other slice routing resources.

Slice Distributed RAM Timing Model and Parameters

Figure 1-3 illustrates the details of distributed RAM implemented in a Virtex-II slice.

Note: Some elements of the Virtex-II slice have been omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

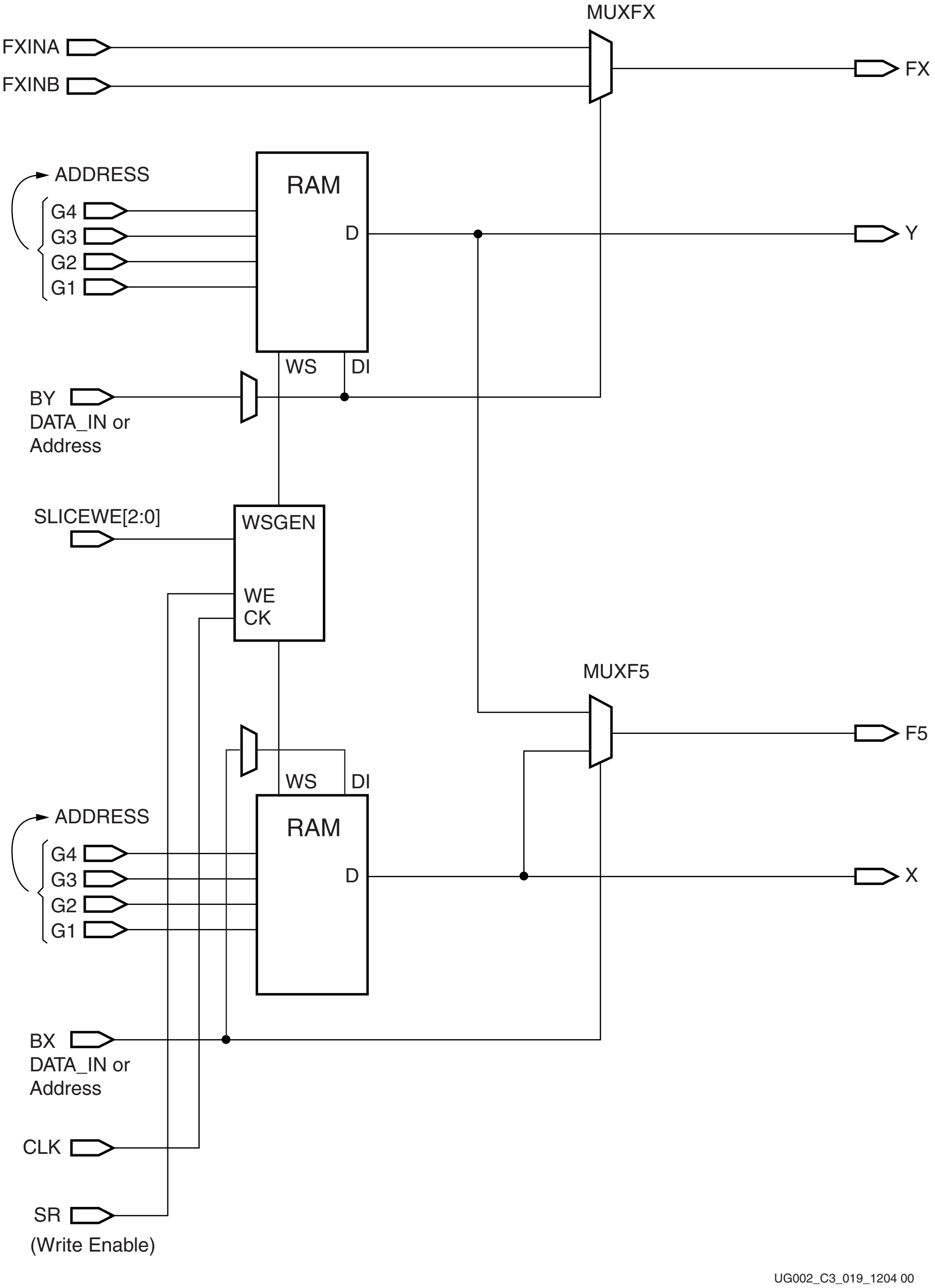


Figure 1-3: **Slice Distributed RAM Diagram**

Timing Parameters

Parameter	Function	Control Signal	Description
Sequential Delays for Slice LUT Configured as RAM (Distributed RAM)			
$T_{SHCKO16}$	CLK to X/Y outputs (WE active) in 16x1 mode		Time after the Clock (CLK) of a WRITE operation that the data written to the distributed RAM (in 16x1 mode) is stable on the X/Y outputs of the slice.
$T_{SHCKO32}$	CLK to X/Y outputs (WE active) in 32x1 mode		Time after the Clock (CLK) of a WRITE operation that the data written to the distributed RAM (in 32x1 mode) is stable on the X/Y outputs of the slice.
$T_{SHCKOF5}$	CLK to F5 output (WE active)		Time after the Clock (CLK) of a WRITE operation that the data written to the distributed RAM is stable on the F5 output of the slice.
Setup and Hold for Slice LUT Configured as RAM (Distributed RAM)			
T_{xS} = Setup time (before clock edge) T_{xH} = Hold time (after clock edge)			The following descriptions are for setup times only.
T_{DS}/T_{DH}	BX/BY Data inputs (DI)		Time before the clock that data must be stable at the DI input of the slice LUT (configured as RAM), via the slice BX/BY inputs.
T_{AS}/T_{AH}	F/G Address inputs		Time before the clock that address signals must be stable at the F/G inputs of the slice LUT (configured as RAM).
T_{WES}/T_{WEH}	WE input (SR)		Time before the clock that the Write Enable signal must be stable at the WE input of the slice LUT (configured as RAM).
Clock CLK			
T_{WPH}			Minimum Pulse Width, High (for a Distributed RAM clock).
T_{WPL}			Minimum Pulse Width, Low (for a Distributed RAM clock).
T_{WC}			Minimum clock period to meet address write cycle time.

Figure 1-4 illustrates the timing characteristics of a 16-bit distributed RAM implemented in a Virtex-II slice (LUT configured as RAM).

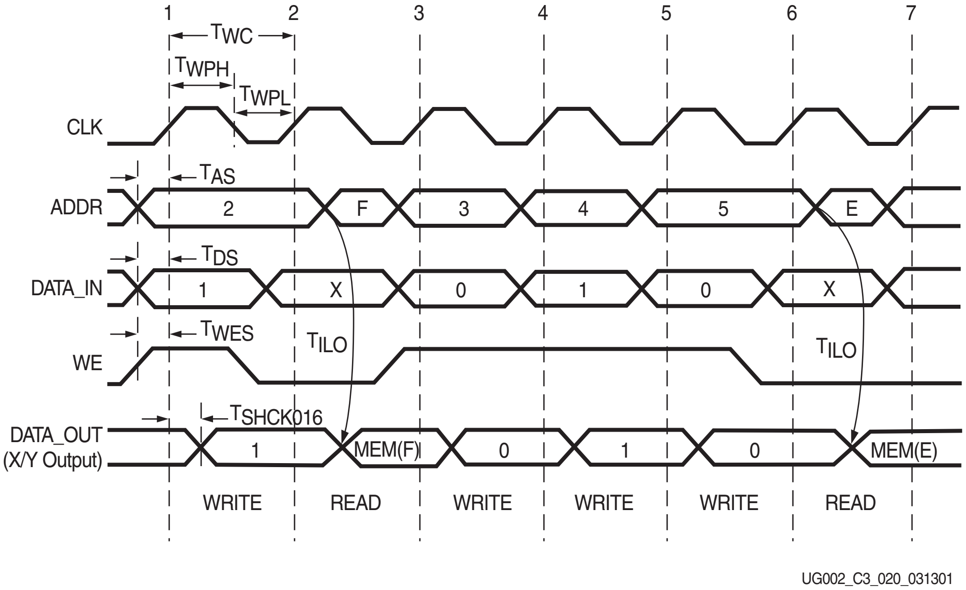


Figure 1-4: **Slice Distributed RAM Timing Diagram**

Clock Event 1: WRITE Operation

During a WRITE operation, the contents of the memory at the address on the ADDR inputs is changed. The data written to this memory location is reflected on the X/Y outputs synchronously.

- At time T_{WES} before Clock Event 1, the Write Enable signal (WE) becomes valid-high, enabling the RAM for the following WRITE operation.
- At time T_{AS} before Clock Event 1, the address (2) becomes valid at the F/G inputs of the RAM.
- At time T_{DS} before Clock Event 1, the DATA becomes valid (1) at the DI input of the RAM and is reflected on the X/Y output at time $T_{SHCKO16}$ after Clock Event 1.

Clock Event 2: READ Operation

All READ operations are asynchronous in distributed RAM. As long as write-enable (WE) is Low, the address bus can be asserted at any time, and the contents of the RAM at that address are reflected on the X/Y outputs after a delay of length T_{ILO} (propagation delay through a LUT). Note that the Address (F) is asserted *after* Clock Event 2, and that the contents of the RAM at that location are reflected on the output after a delay of length T_{ILO}.

Slice SRL Timing Model and Parameters

Figure 1-5 illustrates shift register implementation in a Virtex-II slice.

Note: Some elements of the Virtex-II slice have been omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

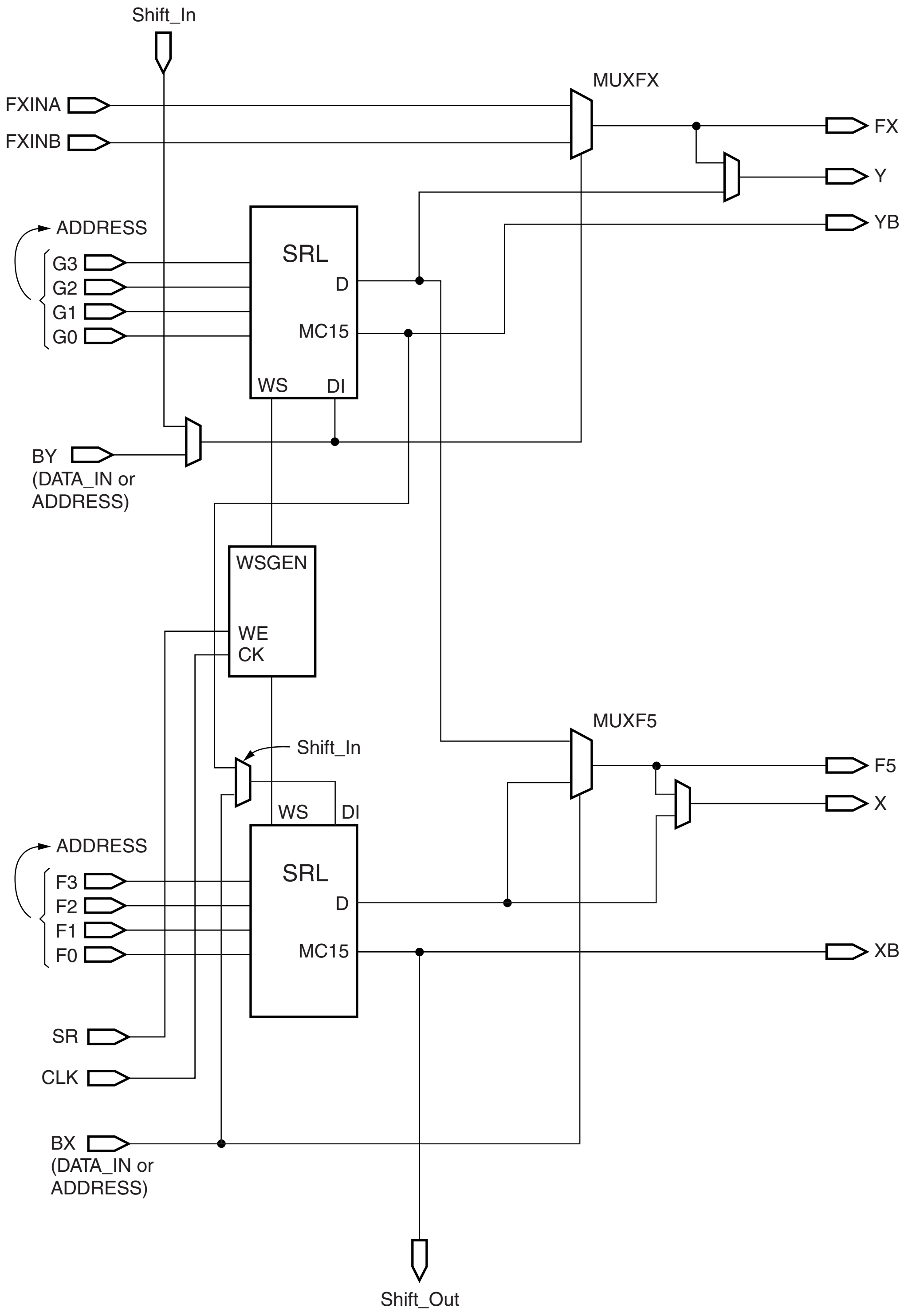


Figure 1-5: **Slice SLR Diagram**

Timing Parameters

Parameter	Function	Control Signal	Description
Sequential Delays for Slice LUT Configured as SRL (Select Shift Register)			
T_{REG}	CLK to X/Y outputs		Time after the Clock (CLK) of a WRITE operation that the data written to the SRL is stable on the X/Y outputs of the slice.
T_{CKSH}	CLK to Shiftout		Time after the Clock (CLK) of a WRITE operation that the data written to the SRL is stable on the Shiftout or XB/YB outputs of the slice.
T_{REGF5}	CLK to F5 output		Time after the Clock (CLK) of a WRITE operation that the data written to the SRL is stable on the F5 output of the slice.
Setup/Hold for Slice LUT Configured as SRL (Select Shift Register)			
T_{xxS} = Setup time (before clock edge) T_{xxH}= Hold time (after clock edge)			The following descriptions are for setup times only.
T_{SRLDS}/ T_{SRLDH}	BX/BY Data inputs (DI)		Time before the clock that data must be stable at the DI input of the slice LUT (configured as SRL), via the slice BX/BY inputs.
T_{WSS}/T_{WSH}	CE input (WE)		Time before the clock that the Write Enable signal must be stable at the WE input of the slice LUT (configured as SRL).
Clock CLK			
T_{SRPH}			Minimum Pulse Width, High (for an SRL clock).
T_{SRPL}			Minimum Pulse Width, Low (for an SRL clock).

Figure 1-6 illustrates the timing characteristics of a 16-bit shift register implemented in a Virtex-II slice (LUT configured as SRL).

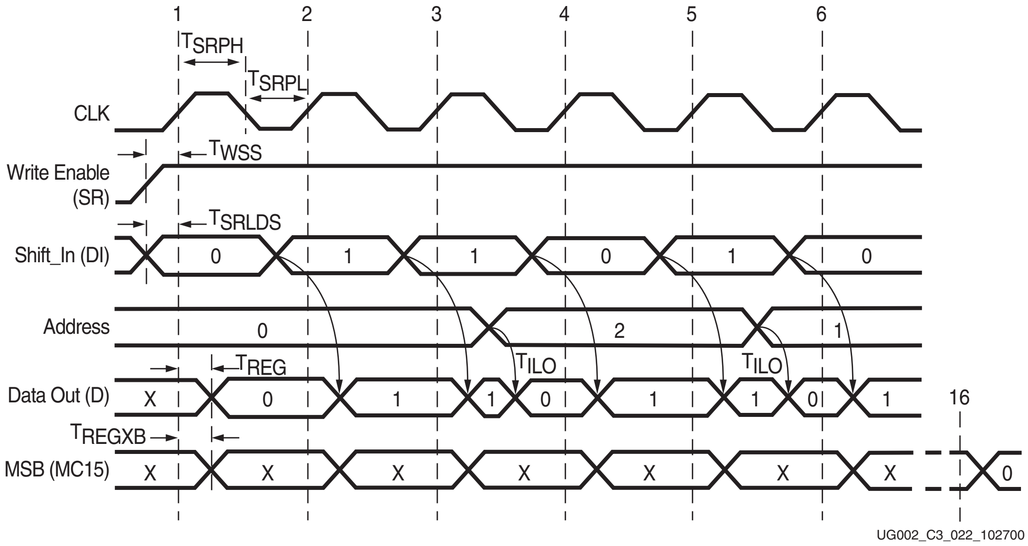


Figure 1-6: **Slice SLR Timing Diagram**

Clock Event 1: Shift_In

During a WRITE (Shift_In) operation, the single-bit content of the register at the address on the ADDR inputs is changed, as data is shifted through the SRL. The data written to this register is reflected on the X/Y outputs synchronously, if the address is unchanged during the clock event. If the ADDR inputs are changed during a clock event, the value of the data at the addressable output (D) is invalid.

- At time T_{WSS} before Clock Event 1, the Write Enable signal (SR) becomes valid-high, enabling the SRL for the WRITE operation that follows.
- At time T_{SRLDS} before Clock Event 1 the data becomes valid (0) at the DI input of the SRL and is reflected on the X/Y output after a delay of length T_{REG} after Clock Event 1*.

  * Note: Since the address 0 is specified at Clock Event 1, the data on the DI input is reflected at the D output, because it is written to Register 0.

Clock Event 2: Shift_In

- At time T_{SRLDS} before Clock Event 2, the data becomes valid (1) at the DI input of the SRL and is reflected on the X/Y output after a delay of length T_{REG} after Clock Event 2*.

  * Note: Since the address 0 is still specified at Clock Event 2, the data on the DI input is reflected at the D output, because it is written to Register 0.

Clock Event 3: Shift_In / Addressable (Asynchronous) READ

All READ operations are asynchronous. If the address is changed (between clock events), the contents of the register at that address are reflected at the addressable output (X/Y outputs) after a delay of length T_{ILO} (propagation delay through a LUT).

- At time T_{SRLDS} before Clock Event 3 the Data becomes valid (1) at the DI input of the SRL, and is reflected on the X/Y output T_{REG} time after Clock Event 3.
- Notice that the address is changed (from 0 to 2) some time after Clock Event 3. The value stored in Register 2 at this time is a 0 (in this example, this was the first data shifted in), and it is reflected on the X/Y output after a delay of length T_{ILO}.

Clock Event 16: MSB (Most Significant Bit) Changes

- At time T_{REGXB} after Clock Event 16, the first bit shifted into the SRL becomes valid (logical 0 in this case) on the XB output of the slice via the MC15 output of the LUT (SRL).

Block SelectRAM Timing Model

Introduction

This section describes the timing parameters associated with the block SelectRAM (illustrated in Figure 1-7) in Virtex-II FPGA devices. This section is intended to be used with the section on switching characteristics in the **Virtex-II Data Sheet** and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the switching characteristics section in the **Virtex-II Data Sheet**.

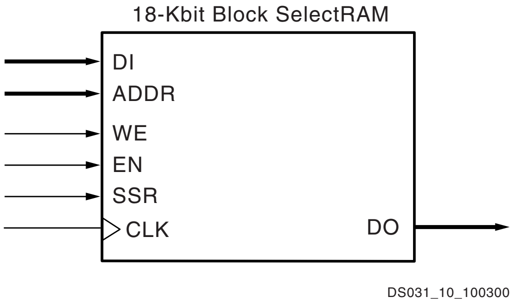
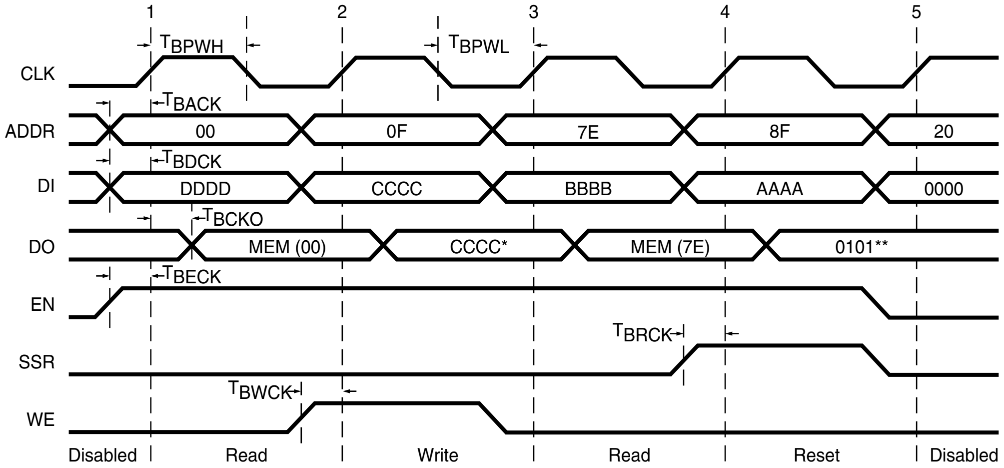


Figure 1-7: **Block SelectRAM Block Diagram**

Timing Parameters

Parameter	Function	Control Signal	Description
Setup and Hold Relative to Clock (CLK)			
T_{BxCK} = Setup time (before clock edge) T_{BCKx} = Hold time (after clock edge)			The following descriptions are for setup times only.
T_{BACK}/T_{BCKA}	Address inputs	ADDR	Time before the clock that address signals must be stable at the ADDR inputs of the block RAM.
T_{BDCK}/T_{BCKD}	Data inputs	DI	Time before the clock that data must be stable at the DI inputs of the block RAM.
T_{BECK}/T_{BCKE}	Enable	EN	Time before the clock that the enable signal must be stable at the EN input of the block RAM.
T_{BRCK}/T_{BCKR}	Synchronous Set/Reset	SSR	Time before the clock that the synchronous set/reset signal must be stable at the SSR input of the block RAM.
T_{BWCK}/T_{BCKW}	Write Enable	WE	Time before the clock that the write enable signal must be stable at the WE input of the block RAM.
Clock to Out			
T_{BCKO}	Clock to Output	CLK to DO	Time after the clock that the output data is stable at the DO outputs of the block RAM.
Clock			
T_{BPWH}	Clock	CLK	Minimum pulse width, high.
T_{BPWL}	Clock	CLK	Minimum pulse width, low.

The timing diagram in Figure 1-8 describes a single-port block RAM in Write-First mode. The timing for Read-First and No-Change modes are similar (see chapter 2, block RAM section.)



* Write Mode = "WRITE_FIRST"

** SRVAL = 0101

ug002_c3_002_100300

Figure 1-8: **Block SelectRAM Timing Diagram**

At time 0, the block RAM is disabled; EN (enable) is low.

Clock Event 1

READ Operation:

During a read operation, the contents of the memory at the address on the ADDR inputs are unchanged.

- T_{BACK} before Clock Event 1, address 00 becomes valid at the ADDR inputs of the block RAM.
- At time T_{BECK} before Clock Event 1, Enable goes High at the EN input of the block RAM, enabling the memory for the READ operation that follows.
- At time T_{BCKO} after Clock Event 1, the contents of the memory at address 00 become stable at the DO pins of the block RAM.

Clock Event 2

WRITE Operation:

During a write operation, the content of the memory at the location specified by the address on the ADDR inputs is replaced by the value on the DI pins and is immediately reflected on the output latches (in WRITE-FIRST mode); EN (enable) is high.

- At time T_{BACK} before Clock Event 2, address 0F becomes valid at the ADDR inputs of the block RAM.
- At time T_{BDCK} before Clock Event 2, data CCCC becomes valid at the DI inputs of the block RAM.
- At time T_{BWCK} before Clock Event 2, Write Enable becomes valid at the WE following the block RAM.
- At time T_{BCKO} after Clock Event 2, data CCCC becomes valid at the DO outputs of the block RAM.

Clock Event 4

SSR (Synchronous Set/Reset) Operation

During an SSR operation, initialization parameter value SRVAL is loaded into the output latches of the block SelectRAM. The SSR operation does NOT change the contents of the memory and is independent of the ADDR and DI inputs.

- At time T_{BRCK} before Clock Event 4, the synchronous set/reset signal becomes valid (High) at the SSR input of the block RAM.
- At time T_{BCKO} after Clock Event 4, the SRVAL 0101 becomes valid at the DO outputs of the block RAM.

Clock Event 5

Disable Operation:

De-asserting the enable signal EN disables any write, read or SSR operation. The disable operation does NOT change the contents of the memory or the values of the output latches.

- At time T_{BECK} before Clock Event 5, the enable signal becomes valid (Low) at the EN input of the block RAM.
- After Clock Event 5, the data on the DO outputs of the block RAM is unchanged.

Timing Model

Figure 1-9 illustrates the delay paths associated with the implementation of block SelectRAM. This example takes the simplest paths on and off chip (these paths can vary greatly depending on the design). This timing model demonstrates how and where the block SelectRAM timing parameters are used.

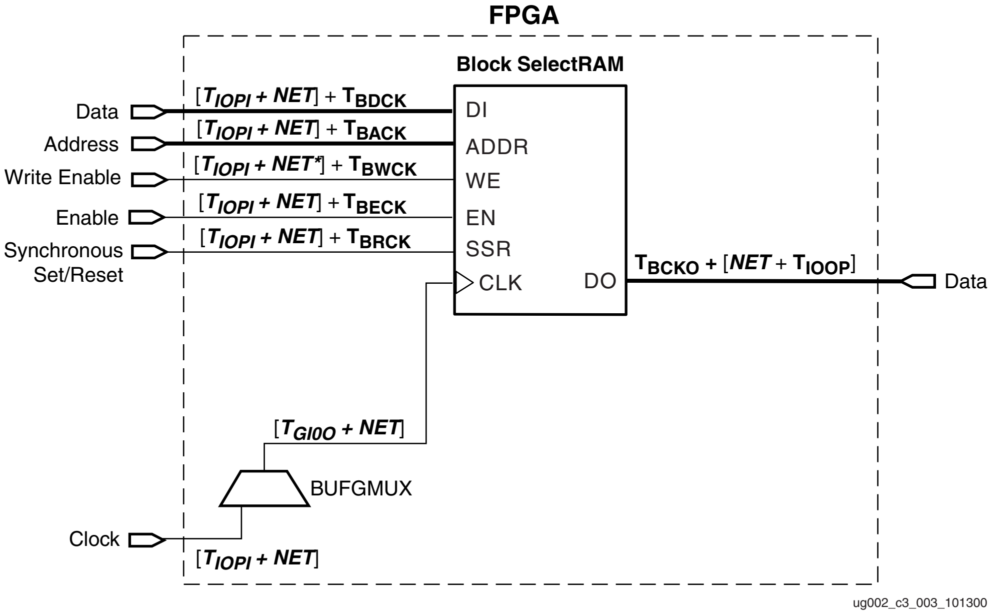


Figure 1-9: **Block SelectRAM Timing Model**

NET = Varying interconnect delays

$\mathbf{T_{IOPI}}$ = Pad to I-output of IOB delay

$\mathbf{T_{IOOP}}$ = O-input of IOB to pad delay

$\mathbf{T_{GI0O}}$ = BUFGMUX delay

Embedded Multiplier Timing Model

Introduction

This section explains all timing parameters associated with the use of embedded 18-bit x 18-bit multipliers in Virtex-II FPGAs (see Figure 1-10). The propagation delays through the embedded multiplier differ based on the size of the multiplier function implemented. The longest delay through the multiplier is to the highest order bit output (P35). Therefore, if an 18-bit x 18-bit signed multiplier is implemented, the worst-case delay for this function is the longest delay associated with the embedded multiplier block. If smaller (LSB) multipliers are used, shorter delays can be realized.

This section is intended to be used in conjunction with the section on switching characteristics in the **Virtex-II Data Sheet** and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the **Virtex-II Data Sheet**.

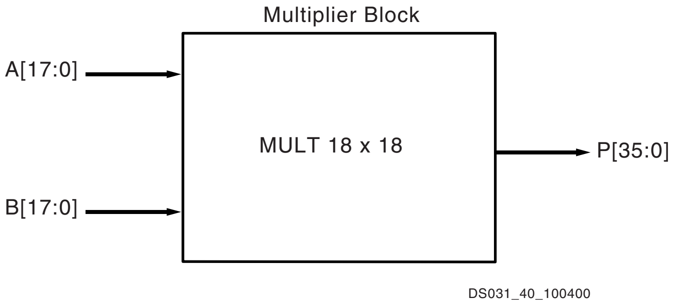


Figure 1-10: **Embedded 18-bit x 18-bit Multiplier Block**

Timing Parameters

Propagation Delays (All Worst-Case)

Table 1-1 lists the different values for the T_{MULT} timing parameter reported by the Timing Analyzer software. These values correspond to the propagation delay through the multiplier to a specific output pin of the multiplier block.

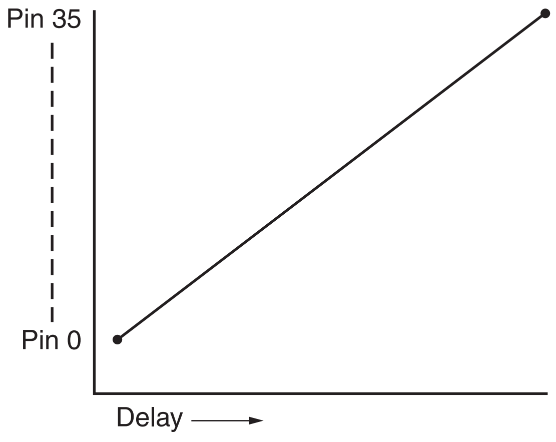
Table 1-1: **Multiplier Switching Characteristics**

Description	Symbol
Propagation Delay to Output Pin	
Input to Pin35	T_{MULT}
Input to Pin34	T_{MULT}
Input to Pin33	T_{MULT}
Input to Pin32	T_{MULT}
Input to Pin31	T_{MULT}
Input to Pin30	T_{MULT}
Input to Pin29	T_{MULT}
Input to Pin28	T_{MULT}
Input to Pin27	T_{MULT}
Input to Pin26	T_{MULT}
Input to Pin25	T_{MULT}
Input to Pin24	T_{MULT}
Input to Pin23	T_{MULT}

Table 1-1: **Multiplier Switching Characteristics** ***(Continued)***

Description	Symbol
Input to Pin22	T_{MULT}
Input to Pin21	T_{MULT}
Input to Pin20	T_{MULT}
Input to Pin19	T_{MULT}
Input to Pin18	T_{MULT}
Input to Pin17	T_{MULT}
Input to Pin16	T_{MULT}
Input to Pin15	T_{MULT}
Input to Pin14	T_{MULT}
Input to Pin13	T_{MULT}
Input to Pin12	T_{MULT}
Input to Pin11	T_{MULT}
Input to Pin10	T_{MULT}
Input to Pin9	T_{MULT}
Input to Pin8	T_{MULT}
Input to Pin7	T_{MULT}
Input to Pin6	T_{MULT}
Input to Pin5	T_{MULT}
Input to Pin4	T_{MULT}
Input to Pin3	T_{MULT}
Input to Pin2	T_{MULT}
Input to Pin1	T_{MULT}
Input to Pin0	T_{MULT}

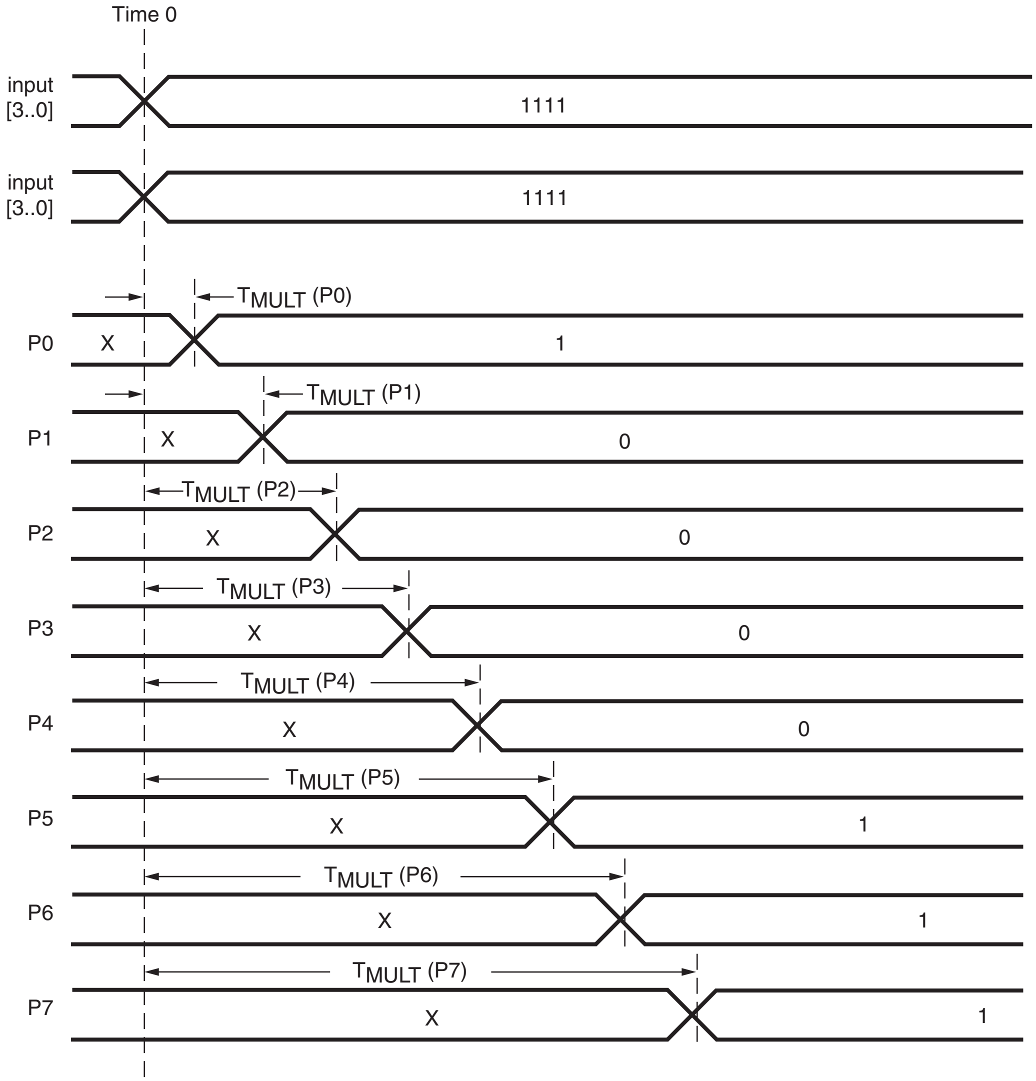
The shortest delay is to pin 0 and the longest delay to pin 35. Notice that the delay-to-pin ratio is essentially linear (see Figure 1-11). This implies that smaller multiply functions are faster than larger ones. This is true as long as the LSB inputs are used.



UG002_C3_023_092500

Figure 1-11: **Pin-to-Delay Ratio Curve**

Figure 1-12 illustrates the result (outputs) of a 4-bit x 4-bit unsigned multiply implemented in an embedded multiplier block.



UG002_C3_024_101300

Figure 1-12: **Embedded Multiplier Block Timing Diagram**

At time 0 the two 4-bit numbers to be multiplied become valid at the A[0..3], B[0..3] inputs to the embedded multiplier. The result appears on the output pins P[0..7] in a staggered fashion. First, P0 becomes valid at time T_{MULT}(P0), followed by each subsequent output pin, until P7 becomes valid at time T_{MULT} (P7). In this case, the delay for this multiply function should correspond to that of Pin 7. In other words, the result is not valid until all output pins become valid.

IOB Timing Model

Introduction

This section describes all timing parameters associated with the Virtex-II IOB. The section consists of three parts:

- IOB Input Timing Model and Parameters
- IOB Output Timing Model and Parameters
- IOB 3-State Timing Model and Parameters

This section is intended to be used in conjunction with the section on switching characteristics in the **Virtex-II Data Sheet** and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the **Virtex-II Data Sheet**.

A Note on I/O Standard Adjustments:

The "IOB Input and Output Switching Characteristics Standard Adjustments" tables in the switching characteristics section of the **Virtex-II Data Sheet** are delay adders (+/-) to be added to all timing parameter values associated with the IOB and the Global Clock (see "Pin-to-Pin Timing Model" on page 145), if an I/O standard other than LVTTL is used.

All values specified in the **Virtex-II Data Sheet** for the parameters covered in this section are specified for LVTTL. If another I/O standard is used, these delays change. However, there are several exceptions. The following parameters associated with the pad going to high-impedance (3-State buffer OFF) should NOT be adjusted:

- T_{IOTHZ}
- $T_{IOTLPHZ}$
- T_{GTS}
- T_{IOCKHZ}
- T_{IOSRHZ}

IOB Input Timing Model and Parameters

Figure 1-13 illustrates IOB inputs.

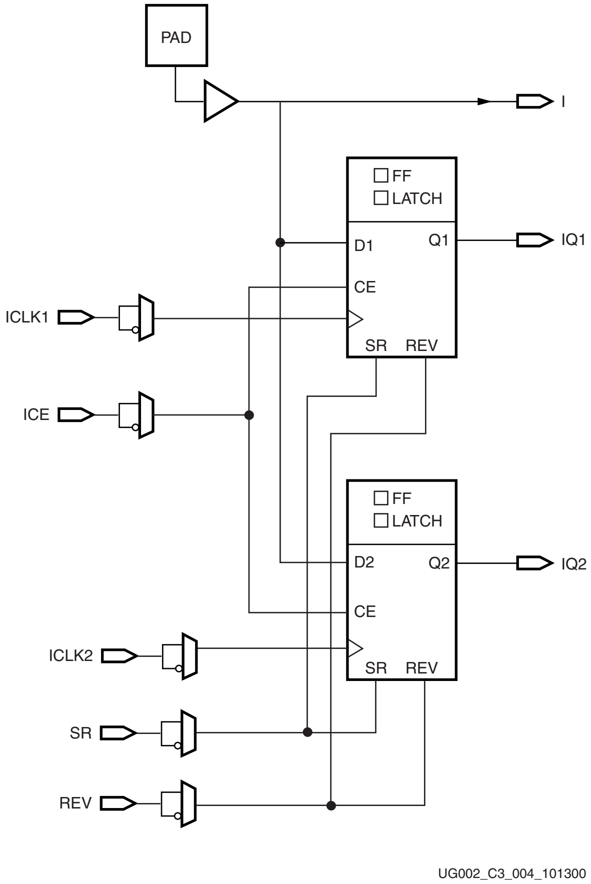


Figure 1-13: **Virtex-II IOB Input Diagram**

Timing Parameters

Parameter	Function	Control Signal	Description
Propagation Delays			
T_{IOPI}			Propagation delay from the pad to I output of the IOB with no delay adder.
T_{IOPID}			Propagation delay from the pad to I output of the IOB with the delay adder.
T_{IOPLI}			Propagation delay from the pad to IQ output of the IOB via transparent latch with no delay adder.
T_{IOPLID}			Propagation delay from the pad to IQ output of the IOB via transparent latch with the delay adder.
Setup and Hold With Respect to Clock at IOB Input Register			
T_{xxCK} = Setup time (before clock edge) T_{xxCKxx} = Hold time (after clock edge)			The following descriptions are for setup times only.
T_{IOPICK}/T_{IOICKP}	ID input with NO delay		Time before the clock that the input signal from the pad must be stable at the ID input of the IOB Input Register, with no delay.

Parameter	Function	Control Signal	Description
$T_{IOPICKD}/T_{IOICKPD}$	ID input with delay		Time before the clock that the input signal from the pad must be stable at the ID input of the IOB Input Register, with delay.
$T_{IOICECK}/T_{IOCKICE}$	ICE input		Time before the clock that the Clock Enable signal must be stable at the ICE input of the IOB Input Register.
$T_{IOSRCKI}$	SR input (IFF, synchronous)		Time before the clock that the Set/Reset signal must be stable at the SR input of the IOB Input Register.
Clock to Out			
T_{IOCKIQ}	Clock (CLK) to (IQ) output		Time after the clock that the output data is stable at the IQ output of the IOB Input Register.
Set/Reset Delays			
T_{IOSRIQ}	SR Input to IQ (asynchronous)		Time after the Set/Reset signal of the IOB is toggled that the output of the IOB input register (IQ) reflects the signal.
T_{GSRQ}	GSR to output IQ		Time after the Global Set/Reset is toggled that the output of the IOB input register (IQ) reflects the set or reset.

Figure 1-14 illustrates IOB input register timing.

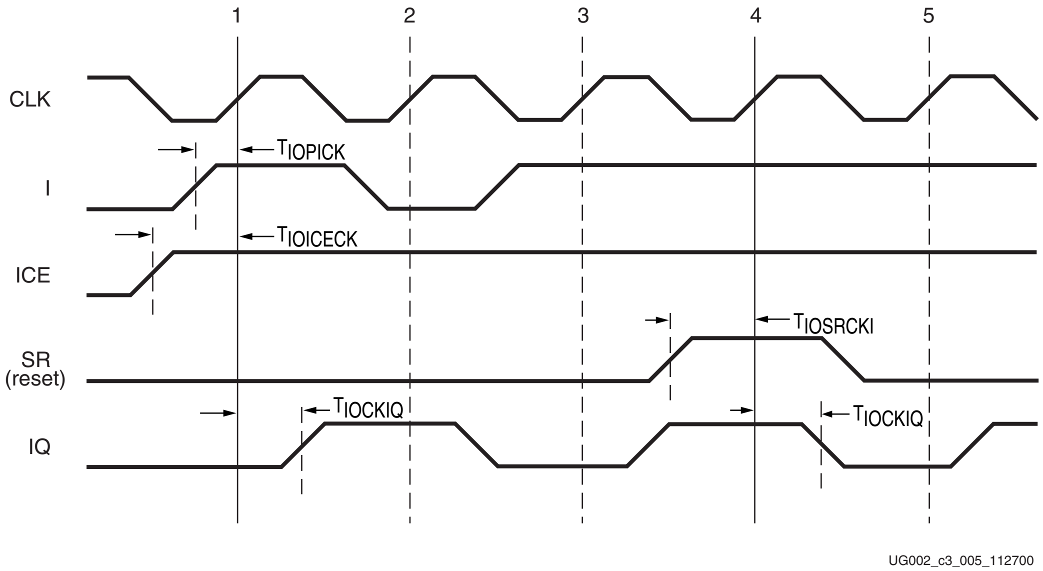


Figure 1-14: **IOB Input Register Timing Diagram**

Clock Events

- At time $T_{IOICECK}$ before Clock Event 1, the input clock enable signal becomes valid-high at the ICE input of the input register, enabling the input register for incoming data.
- At time T_{IOPICK} before Clock Event 1, the input signal becomes valid-high at the I input of the input register and is reflected on the IQ output of the input register at time T_{IOCKIQ} after Clock Event 1.
- At time $T_{IOSRCKI}$ before Clock Event 4 the SR signal (configured as synchronous reset in this case) becomes valid-high resetting the input register and reflected at the IQ output of the IOB at time T_{IOCKIQ} after Clock Event 4.

Figure 1-15 illustrates IOB DDR input register timing.

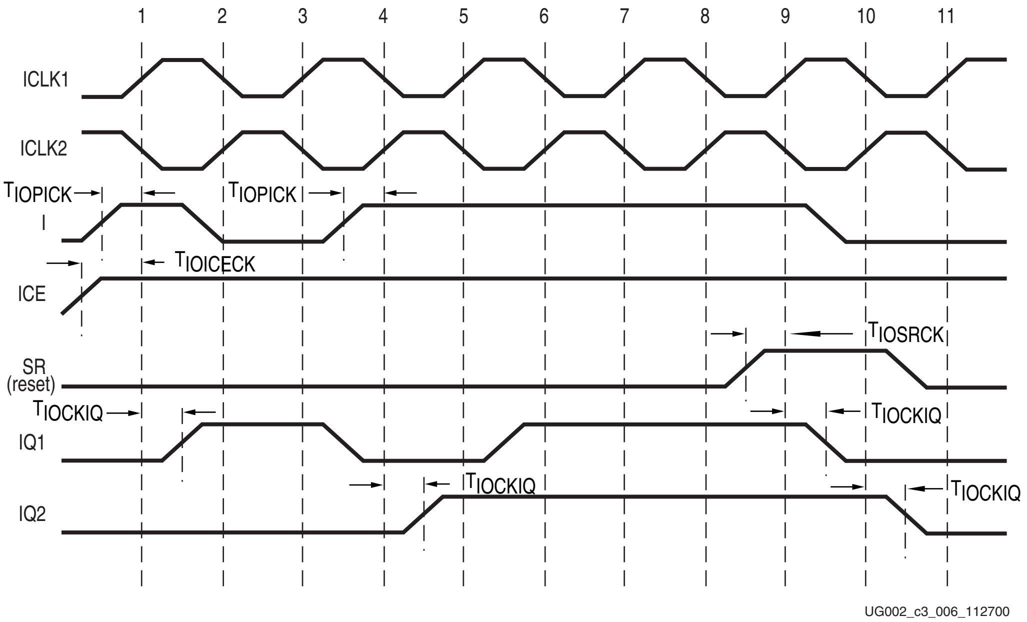


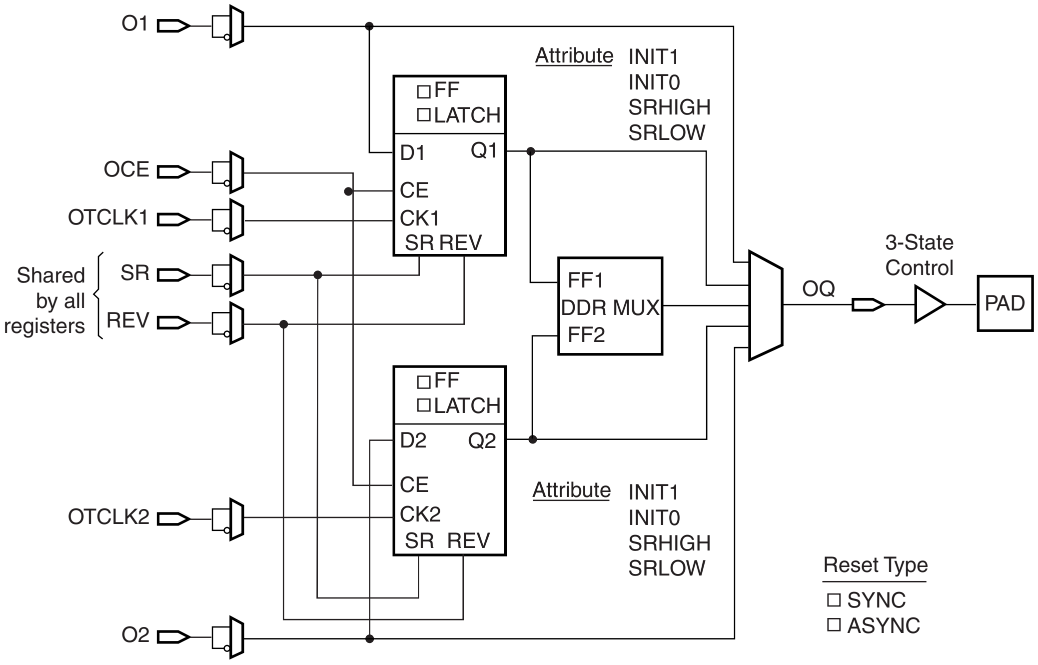
Figure 1-15: **IOB DDR Input Register Timing Diagram**

Clock Events

- At time $T_{IOICECK}$ before Clock Event 1 the input clock enable signal becomes valid-high at the ICE input of both of the DDR input registers, enabling them for incoming data. Since the ICE and I signals are common to both DDR registers, care must be taken to toggle these signals between the rising edges of ICLK1 and ICLK2 as well as meeting the register setup-time relative to both clocks.
- At time T_{IOPICK} before Clock Event 1 (rising edge of ICLK1) the input signal becomes valid-high at the I input of both registers and is reflected on the IQ1 output of input-register 1 at time T_{IOCKIQ} after Clock Event 1.
- At time T_{IOPICK} before Clock Event 2 (rising edge of ICLK2) the input signal becomes valid-low at the I input of both registers and is reflected on the IQ2 output of input-register 2 at time T_{IOCKIQ} after Clock Event 2 (no change in this case).
- At time $T_{IOSRCKI}$ before Clock Event 9 the SR signal (configured as synchronous reset in this case) becomes valid-high resetting input-register 1 (IQ1) at time T_{IOCKIQ} after Clock Event 9, and input-register 2 (IQ2) at time T_{IOCKIQ} after Clock Event 10.

IOB Output Timing Model and Parameters

Figure 1-16 illustrates IOB outputs.



UG002_C3_007_101300

Figure 1-16: **Virtex-II IOB Output Diagram**

Timing Parameters

Parameter	Function	Control Signal	Description
Propagation Delays			
T_{IOOP}			Propagation delay from the O input of the IOB to the pad.
T_{IOOLP}			Propagation delay from the O input of the IOB to the pad via transparent latch.
Setup and Hold With Respect to Clock at IOB Output Register			
T_{xxCK} = Setup time (before clock edge) T_{xxCKxx} = Hold time (after clock edge)			The following descriptions are for setup times only.
T_{IOOCK}/T_{IOCKO}	O input		Time before the clock that data must be stable at the O input of the IOB Output Register.
$T_{IOOCECK}/T_{IOCKOCE}$	OCE input		Time before the clock that the Clock Enable signal must be stable at the OCE input of the IOB Output Register.
$T_{IOSRCKO}/T_{IOCKOSR}$	SR input (OFF)		Time before the clock that the Set/Reset signal must be stable at the SR input of the IOB Output Register.
Clock to Out			
T_{IOCKP}	Clock (CLK) to pad		Time after the clock that the output data is stable at the pad.
Set/Reset Delays			
T_{IOSRP}	SR Input to pad (asynchronous)		Time after the Set/Reset input of the IOB is toggled that the pad reflects the set or reset.
T_{IOGSRQ}	GSR to pad		Time after the Global Set/Reset is toggled that the pad reflects the set or reset.

Figure 1-17 illustrates IOB output register timing.

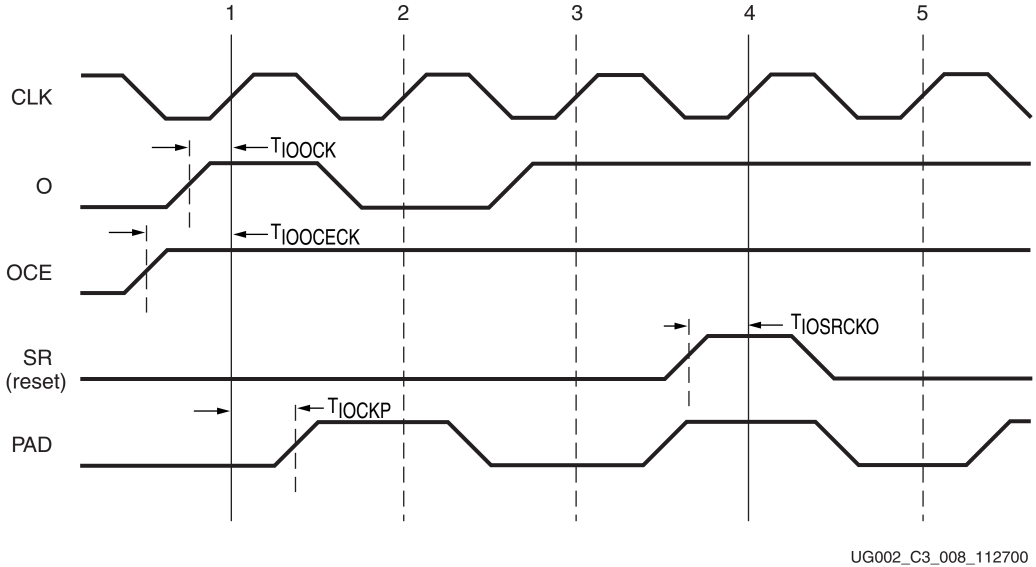


Figure 1-17: **IOB Output Register Timing Diagram**

Clock Events

- At time $T_{IOOCECK}$ before Clock Event 1, the output clock enable signal becomes valid-high at the OCE input of the output register, enabling the output register for incoming data.
- At time T_{IOOCK} before Clock Event 1, the output signal becomes valid-high at the O input of the output register and is reflected on the pad at time T_{IOCKP} after Clock Event 1.
- At time $T_{IOSRCKO}$ before Clock Event 4, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting the output register and reflected on the pad at time T_{IOCKP} after Clock Event 4.

Figure 1-18 illustrates IOB DDR output register timing.

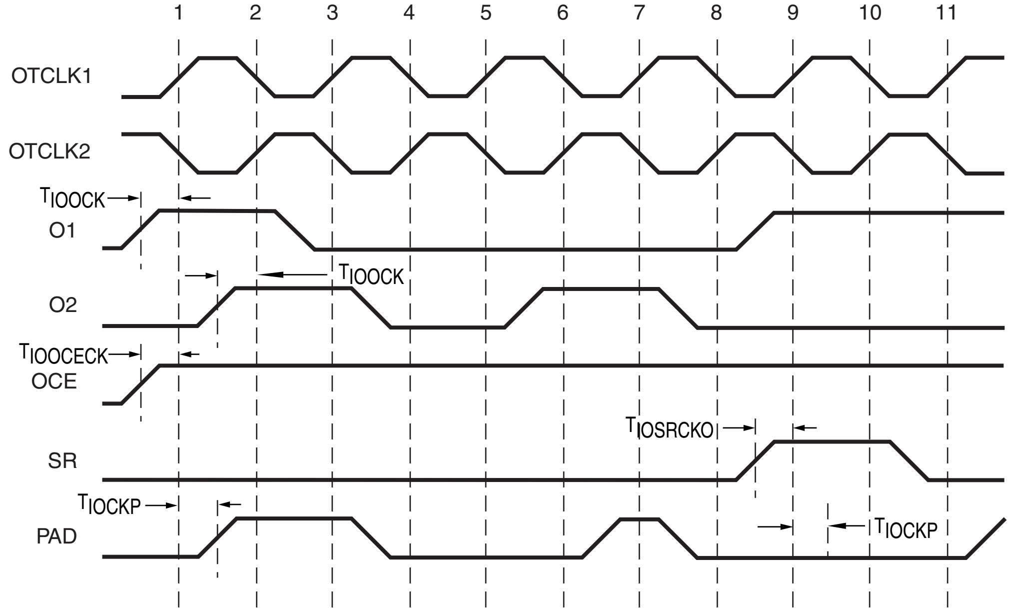


Figure 1-18: **IOB DDR Output Register Timing Diagram**

Clock Events

- At time $T_{IOOCECK}$ before Clock Event 1, the output clock enable signal becomes valid-high at the OCE input of both of the DDR output registers, enabling them for incoming data. Since the OCE signal is common to both DDR registers, care must be taken to toggle this signal between the rising edges of OTCLK1 and OTCLK2 as well as meeting the register setup-time relative to both clocks.
- At time T_{IOOCK} before Clock Event 1 (rising edge of OTCLK1), the output signal O1 becomes valid-high at the O1 input of output register 1 and is reflected on the pad at time T_{IOCKP} after Clock Event 1.
- At time T_{IOOCK} before Clock Event 2 (rising edge of OTCLK2), the output signal O2 becomes valid-high at the O2 input of output register 2 and is reflected on the pad at time T_{IOCKP} after Clock Event 2 (no change on the pad in this case).
- At time $T_{IOSRCKO}$ before Clock Event 9, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting output-register 1 (reflected on the pad at time T_{IOCKP} after Clock Event 9) (no change in this case) and output-register 2 (reflected on the pad at time T_{IOCKP} after Clock Event 10) (no change in this case).

IOB 3-State Timing Model and Parameters

Figure 1-19 illustrates IOB 3-state timing

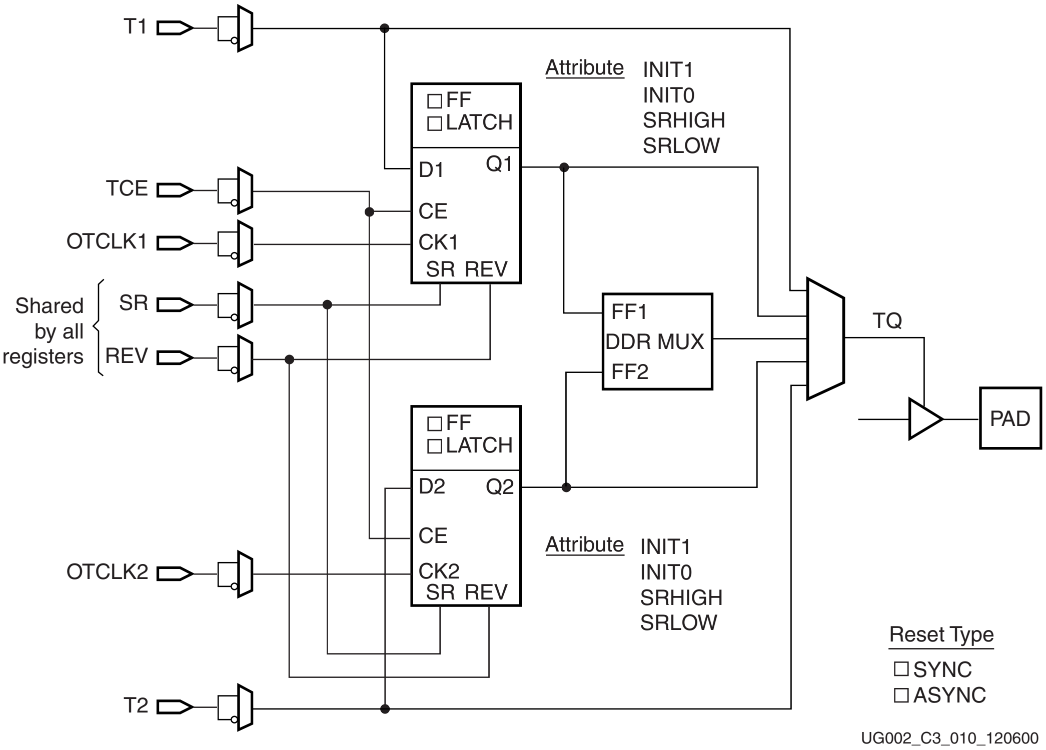


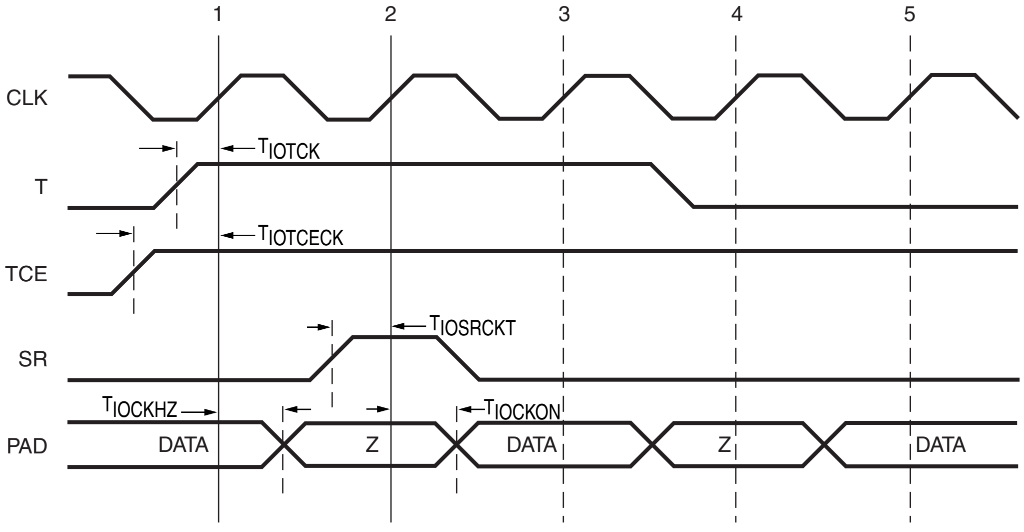
Figure 1-19: **Virtex-II IOB 3-State Diagram**

Timing Parameters

Parameter	Function	Control Signal	Description
Propagation Delays			
T_{IOTHZ}			Time after T input of the IOB is toggled that the pad goes to high-impedance.
T_{IOTON}			Time after the T input of the IOB is toggled that the pad goes from high-impedance to valid data.
$T_{IOTLPHZ}$			Time after the T input of the IOB via transparent latch is toggled that the pad goes to high-impedance.
$T_{IOTLPON}$			Time after the T input of the IOB via transparent latch is toggled that the pad goes from high-impedance to valid data.
T_{GTS}			Time after the Global 3-state signal is asserted that the pad goes to high-impedance.
Setup and Hold With Respect to Clock at IOB 3-State Register			
T_{xxCK} = Setup time (before clock edge) T_{xxCKxx} = Hold time (after clock edge)			The following descriptions are for setup times only.
T_{IOTCK}/T_{IOCKT}	T input		Time before the clock that the signal must be stable at the T input of the IOB 3-state Register.

Parameter	Function	Control Signal	Description
$T_{IOTCECK}$/$T_{IOCKTCE}$	TCE input		Time before the clock that the clock enable signal must be stable at the TCE input of the IOB 3-state Register.
$T_{IOSRCKT}$/$T_{IOCKTSR}$	SR input (TFF)		Time before the clock that the set/reset signal.
Clock to Out			
T_{IOCKHZ}	Clock (CLK) to pad High-Z		Time after clock that the pad goes to high-impedance.
T_{IOCKON}	Clock (CLK) to valid data on pad		Time after clock that the pad goes from high-impedance to valid data.
Set/Reset Delays			
T_{IOSRHZ}	SR Input to pad High-Z (asynchronous)		Time after the SR signal is toggled that the pad goes to high-impedance.
T_{IOSRON}	SR Input to valid data on pad (asynchronous)		Time after the SR signal is toggled that the pad goes from high-impedance to valid data.

Figure 1-20 illustrates IOB 3-state register timing.



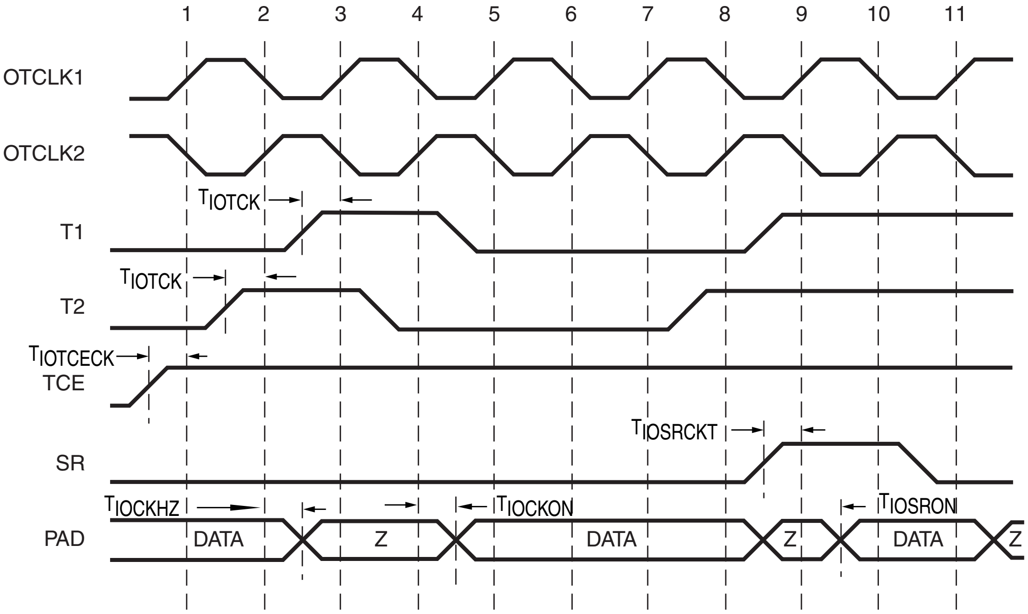
UG002_c3_011_101300

Figure 1-20: **IOB 3-State Register Timing Diagram**

Clock Events

- At time $T_{IOTCECK}$ before Clock Event 1, the 3-state clock enable signal becomes valid-high at the TCE input of the 3-state register, enabling the 3-state register for incoming data.
- At time T_{IOTCK} before Clock Event 1 the 3-state signal becomes valid-high at the T input of the 3-state register, returning the pad to high-impedance at time T_{IOCKHZ} after Clock Event 1.
- At time $T_{IOSRCKT}$ before Clock Event 2, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting the 3-state register and returning the pad to valid data at time T_{IOSRON} after Clock Event 2.

Figure 1-21 illustrates IOB DDR 3-state register timing.



UG002_c3_012_101300

Figure 1-21: **IOB DDR 3-State Register Timing Diagram**

Clock Events

- At time $T_{IOTCECK}$ before Clock Event 1, the 3-state clock enable signal becomes valid-high at the TCE input of both of the DDR 3-state registers, enabling them for incoming data. Since the TCE signal is common to both DDR registers, care must be taken to toggle this signal between the rising edges of OTCLK1 and OTCLK2 as well as meeting the register setup-time relative to both clocks.
- At time T_{IOTCK} before Clock Event 2 (rising edge of OTCLK2), the 3-state signal T2 becomes valid-high at the T2 input of 3-state register 2, switching the pad to high-impedance at time T_{IOCKHZ} after Clock Event 2.
- At time T_{IOTCK} before Clock Event 3 (rising edge of OTCLK1), the 3-state signal T1 becomes valid-high at the T1 input of 3-state register 1, keeping the pad at high-impedance for another half clock cycle (half the period of OTCLK1 or 2).
- At time T_{IOTCK} before Clock Event 4 (rising edge of OTCLK2), the 3-state signal T2 becomes valid-low at the T2 input of 3-state register 2, switching the pad to valid data at time T_{IOCKON} after Clock Event 4. This is repeated for 3-state signal T1 at the following clock event (5) maintaining valid data on the pad until Clock Event 8.
- At time T_{IOTCK} before Clock Event 8 (rising edge of OTCLK2), the 3-state signal T2 becomes valid-high at the T2 input of 3-state register 2, switching the pad to high-impedance at time T_{IOCKHZ} after Clock Event 8.
- At time $T_{IOSRCKT}$ before Clock Event 9 (rising edge of OTCLK1), the SR signal (configured as synchronous reset in this case) becomes valid-high at the SR input of 3-state Register 1, returning the pad to valid data at time T_{IOSRON} after Clock Event 9.

Pin-to-Pin Timing Model

Introduction

This section explains the delays and timing parameters associated with the use of the Global Clock network and the DCM. These delays are true pin-to-pin delays relative to the Global Clock pin and an output or input pin with or without the DCM.

This section consists of two parts:

- Global Clock Input to Output
- Global Clock Setup and Hold

The former describes the delay from the Global Clock pin (with and without the DCM) to an output pin via an Output flip-flop. The latter describes the set-up time for an Input flip-flop from an input pin relative to the Global Clock pin (with and without the DCM).

The values reported in the switching characteristics section of the **Virtex-II Data Sheet** are for LVTTL I/O standards. For different I/O standards, adjust these values with those shown in the "IOB Switching Characteristics Standard Adjustments" tables.

This section is intended to be used in conjunction with the section on switching characteristics in the **Virtex-II Data Sheet** and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the **Virtex-II Data Sheet**.

Global Clock Input to Output

Figure 1-22 illustrates the paths associated with the timing parameters defined in this section. Note that they differ only in their use of the DCM.

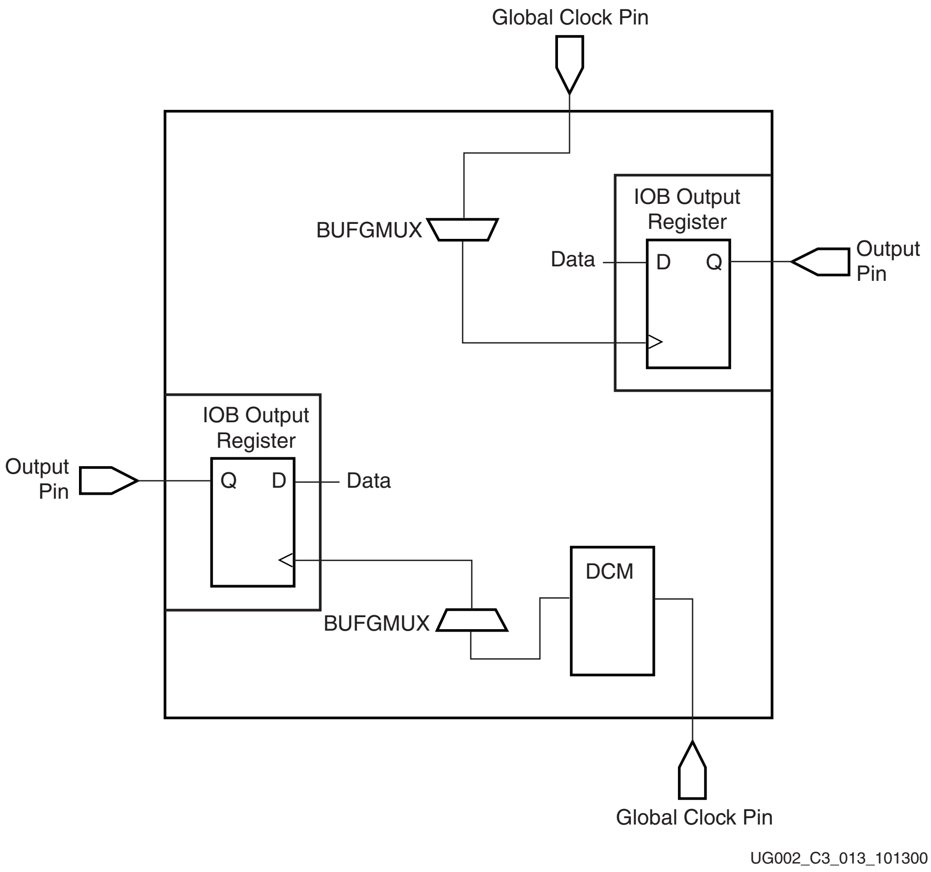


Figure 1-22: **Global Clock Input to Output Model**

Timing Parameters

Parameter	Description
$T_{ICKOFDLL}$	Time after the Global Clock (pin), using the DCM, that the output data from an IOB Output flip-flop is stable at the output pin.
T_{ICKOF}	Time after the Global Clock (pin), without the DCM, that the output data from an IOB Output flip-flop is stable at the output pin.

The waveforms depicted in Figure 1-23 demonstrate the relation of the Global Clock pin, the output data, and the use of the timing parameters.

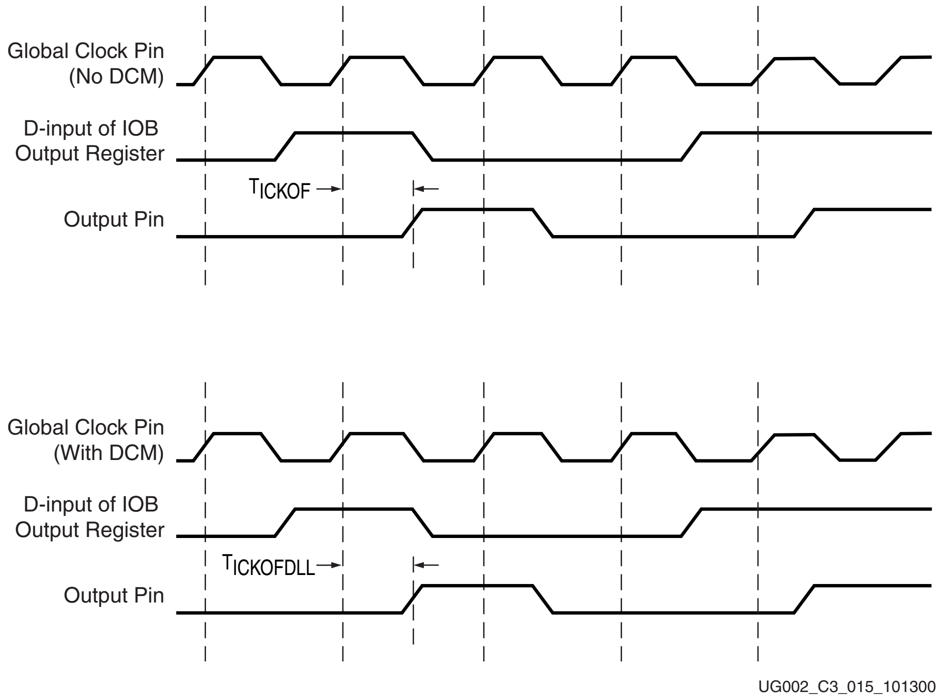
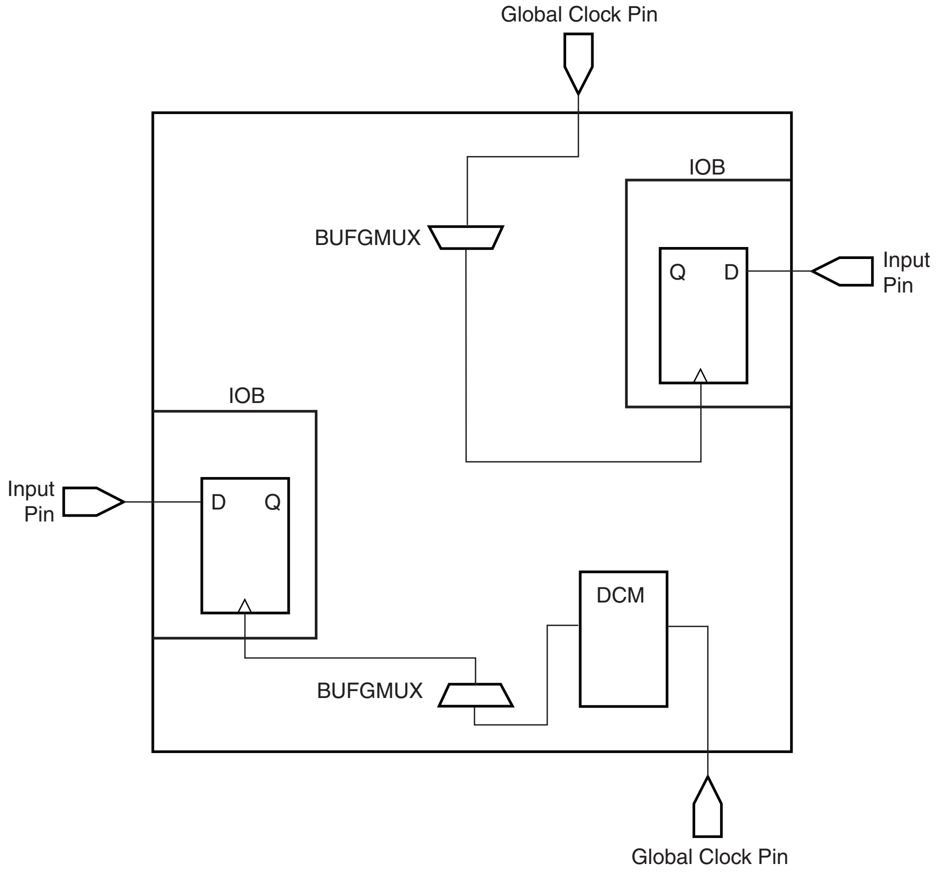


Figure 1-23: **Global Clock Input to Output Timing Diagram**

Global Clock Setup and Hold

Figure 1-24 illustrates the paths associated with the timing parameters defined in this section. Note, they differ only in their use of the DCM.



UG002_C3_014_101300

Figure 1-24: **Global Clock Setup and Hold Model**

Timing Parameters

Setup and Hold for Input Registers Relative to the Global Clock (pin):

- T_{PSDLL} / T_{PHDLL} - Time before the Global Clock (pin), with DCM, that the input signal must be stable at the D-input of the IOB input register.
- T_{PSFD} / T_{PHFD} - Time before the Global Clock (pin), without DCM, that the input signal must be stable at the D-input of the IOB input register.

  Note: T_{PSFD} = Setup time (before clock edge) and T_{PHFD} = Hold time (after clock edge). The previous descriptions are for setup times only.

The waveforms depicted in Figure 1-25 demonstrate the relation of the Global Clock pin, the input data, and the use of the timing parameters.

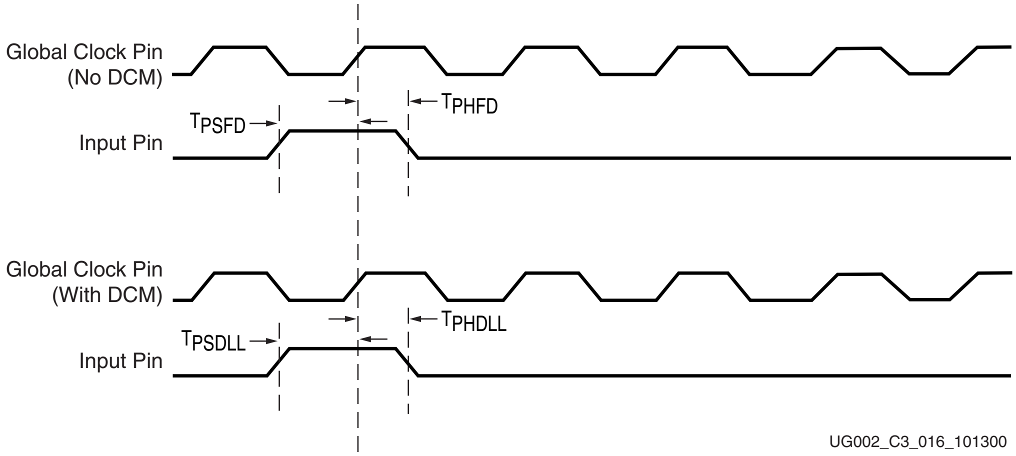


Figure 1-25: **Global Clock Setup and Hold Timing Diagram**

Digital Clock Manager Timing Model

This section describes the timing parameters associated with the Digital Clock Manager (DCM), which are reported in the Virtex-II Data Sheet. Note that these parameters are not used by the Timing Analyzer software in the production of timing reports; they are all measured values and are fully characterized in silicon. For specific timing parameter values, refer to the Virtex-II Data Sheet. This section discusses the following:

- **Operating Frequency Ranges:** The minimum and maximum frequencies supported by the DCM for all clock inputs and outputs.
- **Input Clock Tolerances:** Input clock period (pulse widths), jitter, and drift requirements for proper function of the DCM for all clock inputs.
- **Output Clock Precision:** Output clock period jitter, phase offsets, and duty cycle for all clock outputs of the DCM (worst case).
- **Miscellaneous Timing Parameters:** DCM lock times, Tap delay and shifting range.

For a detailed description of input clock tolerance, jitter, and phase offset see the waveforms at the end of this section.

Operating Frequency Ranges

Figure 1-26 illustrates the DCM functional block and corresponding timing parameters for all clock inputs and outputs.

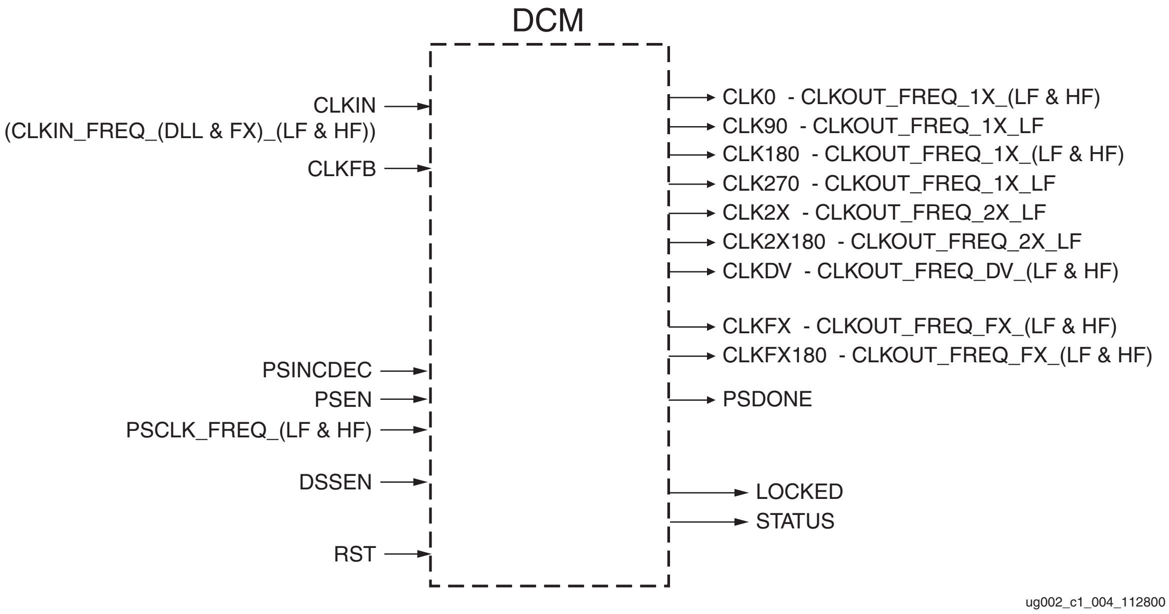


Figure 1-26: **DCM Functional Block: Operating Frequency Ranges**

Timing Parameters

Parameter	Description
Low Frequency Mode	
CLKOUT_FREQ_1X_LF	The minimum and maximum frequency for the CLK0, CLK90, CLK180, CLK270 outputs of the DCM in low-frequency mode.
CLKOUT_FREQ_2X_LF	The minimum and maximum frequency for the CLK2X and CLK2X180 outputs of the DCM in low-frequency mode.
CLKOUT_FREQ_DV_LF	The minimum and maximum frequency for the CLKDV output of the DCM in low-frequency mode.
CLKOUT_FREQ_FX_LF	The minimum and maximum frequency for the CLKFX and CLKFX180 outputs of the DCM in low-frequency mode.
CLKIN_FREQ_DLL_LF[1]	The minimum and maximum frequency for the CLKIN input to the DCM in low-frequency mode when using the delay-locked loop (DLL) outputs.
CLKIN_FREQ_FX_LF[2]	The minimum and maximum frequency for the CLKIN input to the DCM in low-frequency mode when using the FX outputs.
PSCLK_FREQ_LF	The minimum and maximum frequency for the PSCLK input to the DCM in low-frequency mode.
High Frequency Mode	
CLKOUT_FREQ_1X_HF	The minimum and maximum frequency for the CLK0, CLK180 outputs of the DCM in high-frequency mode.
CLKOUT_FREQ_DV_HF	The minimum and maximum frequency for the CLKDV output of the DCM in high-frequency mode.
CLKOUT_FREQ_FX_HF	The minimum and maximum frequency for the CLKFX and CLKFX180 outputs of the DCM in high-frequency mode.
CLKIN_FREQ_DLL_HF	The minimum and maximum frequency for the CLKIN input to the DCM in high-frequency mode when using the DLL outputs.
CLKIN_FREQ_FX_HF	The minimum and maximum frequency for the CLKIN input to the DCM in high-frequency mode when using the FX outputs.
PSCLK_FREQ_HF	The minimum and maximum frequency for the PSCLK input to the DCM in high-frequency mode.

Notes:

1. Delay-locked loop (DLL) outputs include: CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, and CLKDV.
2. FX outputs include: CLKFX and CLKFX180

Input Clock Tolerances

Timing Parameters

Parameter	Description
PSCLK_PULSE[1]	The minimum pulse width (HIGH and LOW) that the PSCLK input to the DCM can have over a range of frequencies.
CLKIN_PULSE	The minimum pulse width (HIGH and LOW) that the CLKIN input to the DCM can have over a range of frequencies. Also applies to PSCLK.
CLKFB_DELAY_VAR_EXT	The maximum allowed variation in delay (across environmental changes) of the feedback clock path when routed externally for board-level de-skew.
Low Frequency Mode	
CLKIN_CYC_JITT_DLL_LF	The maximum cycle-to-cycle jitter the CLKIN input to the DCM can have when using the DLL outputs in low-frequency mode.
CLKIN_CYC_JITT_FX_LF	The maximum cycle-to-cycle jitter the CLKIN input to the DCM can have when using the FX outputs in low-frequency mode.
CLKIN_PER_JITT_DLL_LF	The maximum period jitter the CLKIN input to the DCM can have when using the DLL outputs in low-frequency mode.
CLKIN_PER_JITT_FX_LF	The maximum period jitter the CLKIN input to the DCM can have when using the FX outputs in low-frequency mode.
High Frequency Mode	
CLKIN_CYC_JITT_DLL_HF	The maximum cycle-to-cycle jitter the CLKIN input to the DCM can have when using the DLL outputs in high-frequency mode.
CLKIN_CYC_JITT_FX_HF	The maximum cycle-to-cycle jitter the CLKIN input to the DCM can have when using the FX outputs in high-frequency mode.
CLKIN_PER_JITT_DLL_HF	The maximum period jitter the CLKIN input to the DCM can have when using the DLL outputs in high-frequency mode.
CLKIN_PER_JITT_FX_HF	The maximum period jitter the CLKIN input to the DCM can have when using the FX outputs in high-frequency mode.

Notes:

1. The frequencies applicable to CLKIN_PULSE range from 1 to >400 MHz. These frequencies also apply to PSCLK_PULSE. Since PSCLK can be less than 1 MHz, the pulse width under this condition is specified for PSCLK only.

Output Clock Precision

Timing Parameters

Parameter	Description
CLKOUT_PER_JITT_0	The maximum period jitter of the CLK0 output clock from the DCM (worst case).
CLKOUT_PER_JITT_90	The maximum period jitter of the CLK90 output clock from the DCM (worst case).
CLKOUT_PER_JITT_180	The maximum period jitter of the CLK180 output clock from the DCM (worst case).
CLKOUT_PER_JITT_270	The maximum period jitter of the CLK270 output clock from the DCM (worst case).
CLKOUT_PER_JITT_2X	The maximum period jitter of the CLK2X and CLK2X180 output clocks from the DCM (worst case).
CLKOUT_PER_JITT_DV1	The maximum period jitter of the CLKDV (integer division) output clock from the DCM (worst case).
CLKOUT_PER_JITT_DV2	The maximum period jitter of the CLKDV (non-integer division) output clock from the DCM (worst case).
CLKOUT_PER_JITT_FX	The maximum period jitter of the FX output clocks from the DCM (worst case).
CLKIN_CLKFB_PHASE	Maximum phase offset between the CLKIN and CLKFB inputs to the DCM.
CLKOUT_PHASE	Maximum phase offset between any DCM clock outputs.
CLKOUT_DUTY_CYCLE_DLL	The duty-cycle precision for all DLL outputs.
CLKOUT_DUTY_CYCLE_FX	The duty-cycle precision for the FX outputs.

Miscellaneous DCM Timing Parameters

Table 1-2: **Miscellaneous DCM Timing Parameters**

Parameter	Description
LOCK_DLL	Time required for DCM to lock over a range of clock frequencies when using the DLL outputs.
LOCK_FX	Time required for DCM to lock when using the FX outputs.
LOCK_DLL_FINE_SHIFT	Additional lock time when performing fine phase shifting.
FINE_SHIFT_RANGE	Absolute range for fine phase shifting.
DCM_TAP	Resolution of delay line.

The waveforms in Figure 1-27 demonstrate the relationship between clock tolerance, jitter, and phase.

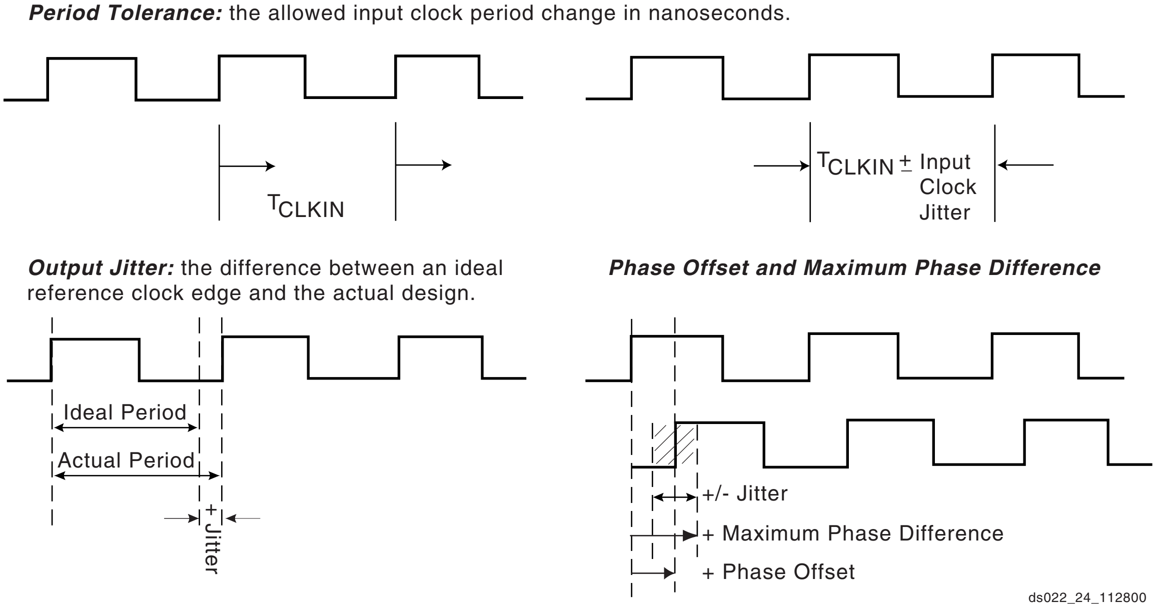


Figure 1-27: **DCM Jitter, Phase, and Tolerance Timing Waveforms**

Output jitter is period jitter measured on the DLL output clocks, excluding input clock jitter.

Phase offset between CLKIN and CLKFB is the worst-case fixed time difference between rising edges of CLKIN and CLKFB, excluding output jitter and input clock jitter.

Phase offset between clock outputs on the DLL is the worst-case fixed time difference between rising edges of any two DLL outputs, excluding output jitter and input clock jitter.

Maximum phase difference between CLKIN an CLKFB is the sum of output jitter and phase offset between CLKIN and CLKFB, or the greatest difference between CLKIN and CLKFB rising edges due to DLL alone (excluding input clock jitter).

Maximum phase difference between clock outputs on the DLL is the sum of output jitter and phase offset between any DLL clock outputs, or the greatest difference between any two DLL output rising edges due to DLL alone (excluding input clock jitter).

Chapter 2

Design Considerations

Summary

This chapter covers the following topics:

- Using Global Clock Networks
- Using Digital Clock Managers (DCMs)
- Using Block SelectRAM™ Memory
- Using Distributed SelectRAM Memory
- Using Look-Up Tables as Shift Registers (SRLUTs)
- Designing Large Multiplexers
- Implementing Sum of Products (SOP) Logic
- Using Embedded Multipliers
- Using Single-Ended SelectI/O Resources
- Using Digitally Controlled Impedance (DCI)
- Using Double-Data-Rate (DDR) I/O
- Using LVDS I/O
- Using Bitstream Encryption
- Using the CORE Generator System

Introduction

This chapter describes how to take advantage of the many special features of the Virtex-II architecture to achieve maximum density and performance. In many cases, the functions described can be automatically generated using the Xilinx CORE Generator™ tool. This is noted throughout the chapter, in the following sections specifically:

- Using Block SelectRAM™ Memory
- Using Distributed SelectRAM Memory
- Using Look-Up Tables as Shift Registers (SRLUTs)
- Designing Large Multiplexers
- Using Embedded Multipliers

Using Global Clock Networks

Introduction

Virtex-II devices support very high frequency designs and thus require low-skew advanced clock distribution. With device density up to 10 million system gates, numerous global clocks are necessary in most designs. Therefore, to provide a uniform and portable solution (soft-IP), all Virtex-II devices from XC2V40 to XC2V8000 have 16 global clock buffers and support 16 global clock domains. Up to eight of these clocks can be used in any quadrant of the device by the synchronous logic elements (that is, registers, 18Kb block RAM, pipeline multipliers) and the IOBs. The software tools place and route these global clocks automatically.

If the design uses between 8 and 16 clocks, it must be partitioned into quadrants, with up to 8 clocks per quadrant. If more than 16 clocks are required, the backbone (24 horizontal and vertical long lines routing resources) can be used as additional clock network.

In addition to clock distribution, the 16 clock buffers are also “glitch-free” synchronous 2:1 multiplexers. These multiplexers are capable of switching between two asynchronous (or synchronous) clocks at any time. No particular phase relations between the two clocks are needed. The clock multiplexers can also be configured as a global clock buffer with a clock enable. The clock can be stopped High or Low at the clock buffer output.

Clock Distribution Resources

The various resources available to manage and distribute the clocks include:

- 16 clock pads that can be used as regular user I/Os if not used as clock inputs. The 16 clock pads can be configured for any I/O standard, including differential standards (for example, LVDS, LVPECL, and so forth).
- 16 “IBUFG” elements that represent the clock inputs in a VHDL or Verilog design.
- 8 “IBUFGDS” elements (that is, attributes LVPECL_33, LVDS_25, LVDS_33, LDT_25, or ULVDS_25) that represent the differential clock input pairs in a VHDL or Verilog design. Each IBUFGDS replaces two IBUFG elements.
- 4 to 12 Digital Clock Managers (DCMs), depending on the device size, to de-skew and generate the clocks. For more information on DCMs, see "Using Digital Clock Managers (DCMs)" on page 175.
- 16 “BUFGMUX” elements that can consist of up to 16 global clock buffers (BUFG), global clock buffers with a clock enable (BUFGCE), or global clock multiplexers (BUFGMUX).

Figure 2-1 illustrates the placement of these clock resources in Virtex-II devices (the XC2V250 through the XC2V2000) that have eight DCMs.

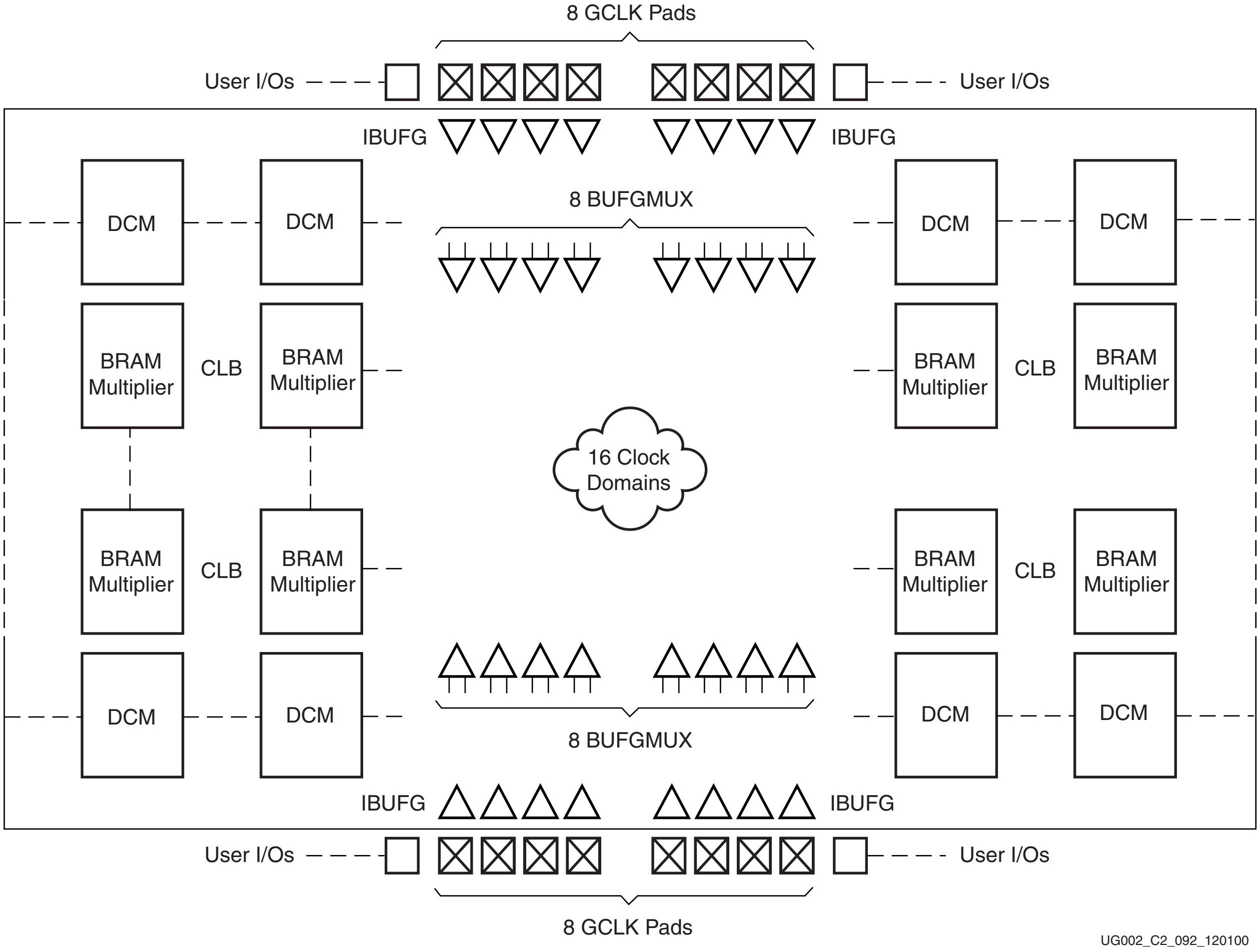


Figure 2-1: **Clock Resources in Virtex-II Devices**

The simple scheme to distribute an external clock in the device is to implement a clock pad with an IBUFG input buffer connected to a BUFG global buffer, as shown in Figure 2-2 and Figure 2-3. The primary (GCLKP) and secondary (GCLKS) clock pads have no relationship with the P-side and N-side of differential clock inputs. In banks 0 and 1, the GCLKP corresponds to the N-side, and the GCLKS corresponds to the P-side of a differential clock input. In banks 4 and 5, this correspondence is reversed.

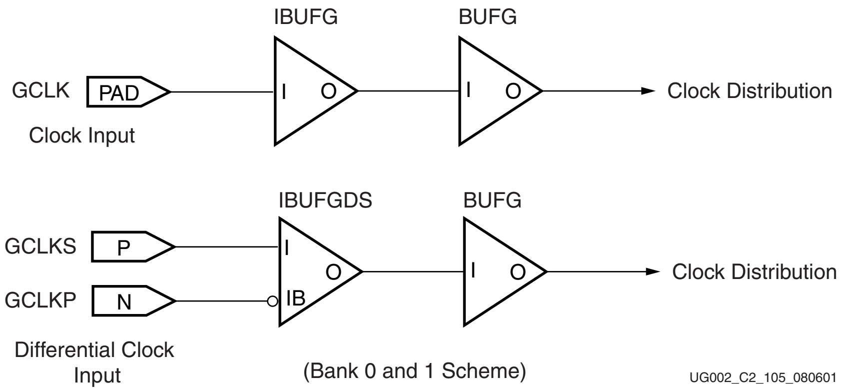


Figure 2-2: **Simple Clock Distribution (Bank 0 and 1 Scheme)**

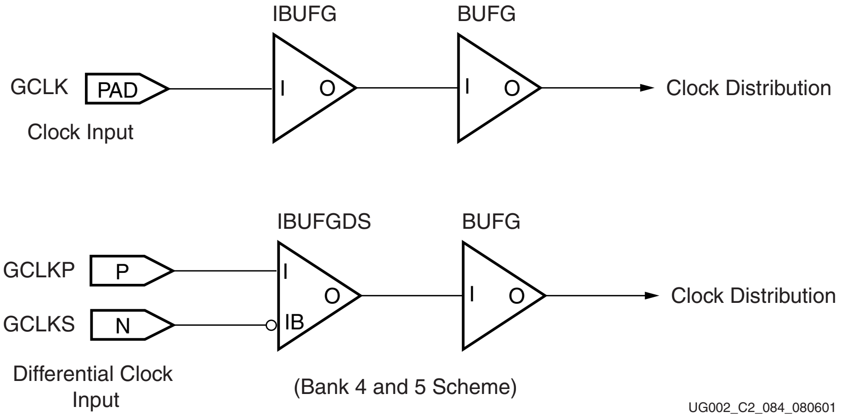


Figure 2-3: **Simple Clock Distribution (Bank 4 and 5 Scheme)**

Major synthesis tools automatically infer the IBUFG and BUFG when the corresponding input signal is used as a clock in the VHDL or Verilog code.

A high frequency or adapted (frequency, phase, and so forth) clock distribution with low skew is implemented by using a DCM between the output of the IBUFG and the input of the BUFG, as shown in Figure 2-4. "Using Digital Clock Managers (DCMs)" on page 175 provides details about DCMs and their use.

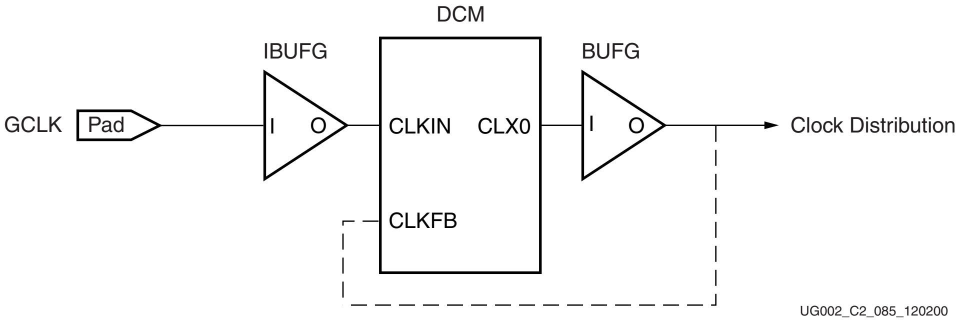


Figure 2-4: **Clock Distribution with DCM**

Clock distribution from internal sources is also possible with a BUFG only or with a DCM, as shown in Figure 2-5.

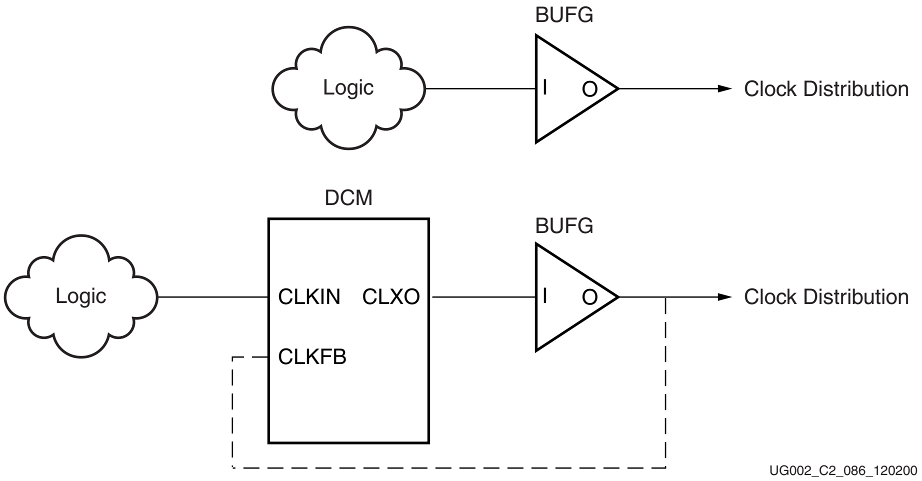


Figure 2-5: **Internal Logic Driving Clock Distribution**

Global Clock Inputs

The clock buffer inputs are fed either by one of the 16 clock pads (refer to the **Virtex-II Data Sheet**), by the outputs of the DCM, or by local interconnect. Each clock buffer can be a synchronous "glitch-free" 2:1 multiplexer with two clock inputs and one select input. Internal logic (or alternatively a regular IOB) can feed the clock inputs. Any internal or external signal can drive the select input or clock enable input.

The possible inputs driving a global clock buffer or multiplexer are summarized in Table 2-1.

Table 2-1: **Inputs Driving Global Clock Buffers or DCMs**

Source	**Destination**				
	BUFG(I) or BUFGCE(I)	**BUFGCE (CE)**	**BUFGMUX (I0 or I1)**	**BUFGMUX (S)**	**DCM (CLKIN)**
External Clock via IBUFG(O)	Dedicated in same quadrant[1]	NA	Dedicated in same quadrant[1]	NA	Same edge
DCM Clock Outputs	Same edge (top or bottom)[2]	NA	Same edge (top or bottom)[2]	NA	General interconnect[3]
Internal Logic	General interconnect	General interconnect	General interconnect	General interconnect	General interconnect[3]
User I/O Pad via IBUF(O) (not IBUFG)	General interconnect	General interconnect	General interconnect	General interconnect	General interconnect[3]
BUFG(O)	NA	NA	NA	NA	Global clock net
BUFGMUX(O)	NA	NA	General interconnect	NA	Global clock net

Notes:

1. Not all IBUFGs in the quadrant have a dedicated connection to a specific BUFG. Others would require general interconnect to be hooked up.
2. Same edge (top or bottom) enables use of dedicated routing resources.
3. Pad to DCM input skew is not compensated.

All BUFG (BUFGCE, BUFGMUX) outputs are available at the quadrant boundaries.

The output of the global clock buffer can be routed to non-clock pins.

Primary and Secondary Global Multiplexers

Each global clock buffer is a self-synchronizing circuit called a clock multiplexer.

The 16 global clock buffers or multiplexers are divided as follows:

- Eight primary clock multiplexers
- Eight secondary clock multiplexers

No hardware difference exists between a primary and a secondary clock multiplexer. However, some restrictions apply to primary/secondary multiplexers, because they share input connections, as well as access to a quadrant.

Each Virtex-II device is divided into four quadrants: North-West, South-West, North-East, and South-East. Each quadrant has two primary and two secondary clock multiplexers. The clock multiplexers are indexed 0 to 7, with one primary and one secondary for each index, alternating on the top and on the bottom (i.e., clock multiplexer "0P" at the bottom is facing clock multiplexer "0S" at the top).

In each device, the eight top/bottom clock multiplexers are divided into four primary and four secondary, indexed 0 to 7, as shown in Figure 2-6.

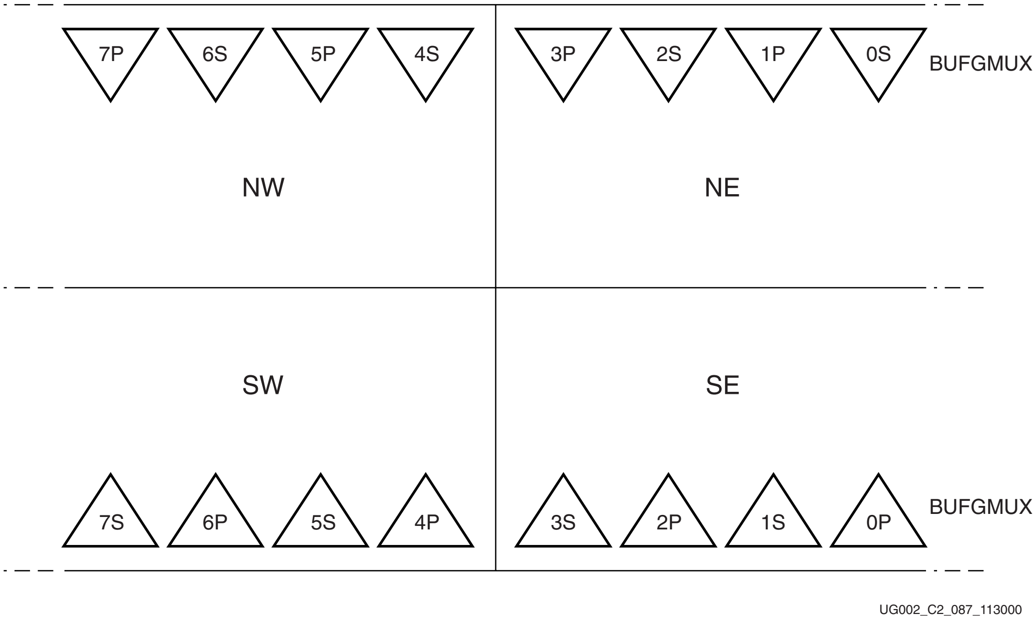


Figure 2-6: **Primary and Secondary Clock Multiplexer Locations**

Primary/Secondary: Rule 1

Considering two "facing" clock multiplexers (BUFG#P and BUFG#S), one or the other of these clock outputs can enter any quadrant of the chip to drive a clock within that quadrant, as shown in Figure 2-7. Note that the clock multiplexers "xP" and "xS" compete for quadrant access. For example, BUFG0P output cannot be used in the same quadrant as BUFG0S.

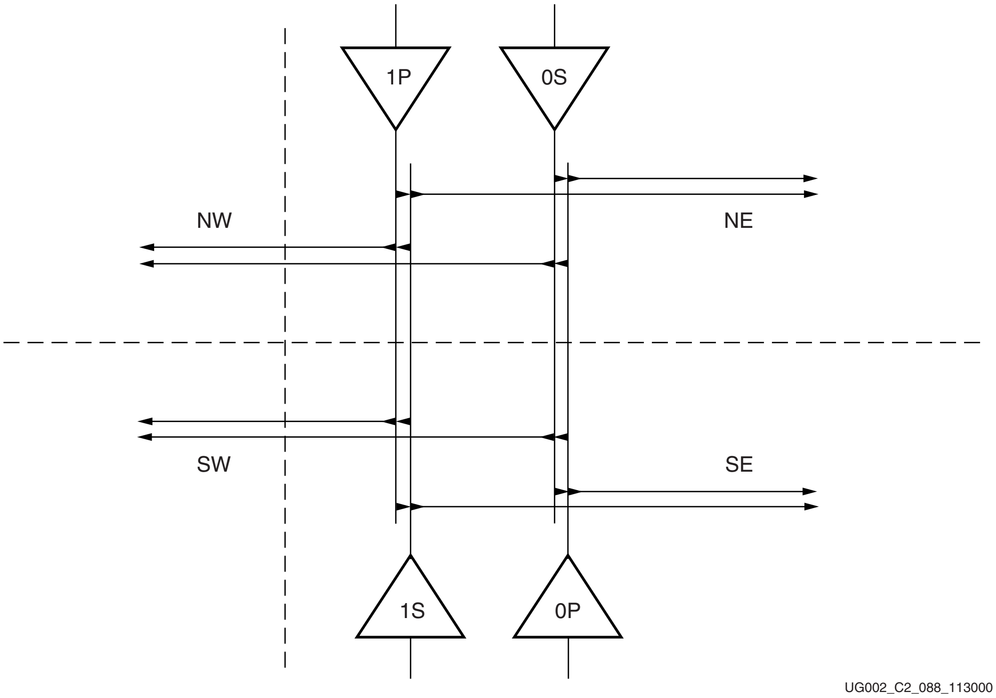


Figure 2-7: **Facing BUFG#P and BUFG#S Connections**

Primary/Secondary: Rule 2

In a BUFGCE or BUFGMUX configuration, shared inputs have to be considered. Any two adjacent clock multiplexers share two inputs, as shown in Figure 2-8. The clock multiplexer "1P" and "0S" have common I0/I1 and I1/I0 inputs.

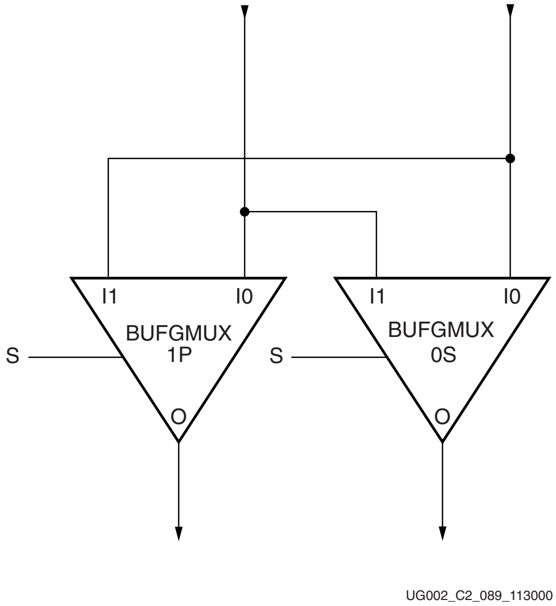


Figure 2-8: **Clock Multiplexer Pair Sharing Clock Multiplexer Inputs**

Table 2-2 lists the clock multiplexer pairs in any Virtex-II device. The primary multiplexer inputs I1/I0 are common with the corresponding secondary multiplexer inputs I0/I1 (i.e., Primary I1 input is common with secondary I0 input, and primary I0 input is common with secondary I1 input).

Table 2-2: **Top Clock Multiplexer Pairs**

Primary I1/I0	1P	3P	5P	7P
Secondary I0/I1	0S	2S	4S	6S

Table 2-3: **Bottom Clock Multiplexer Pairs**

Primary I1/I0	0P	2P	4P	6P
Secondary I0/I1	1S	3S	5S	7S

Primary/Secondary Usage

For up to eight global clocks, it is safe to use the eight primary global multiplexers (1P, 3P, 5P, 7P on the top and 0P, 2P, 4P, 6P on the bottom). Because of the shared inputs, a maximum of eight independent global clock multiplexers can be used in a design, as shown in Figure 2-9.

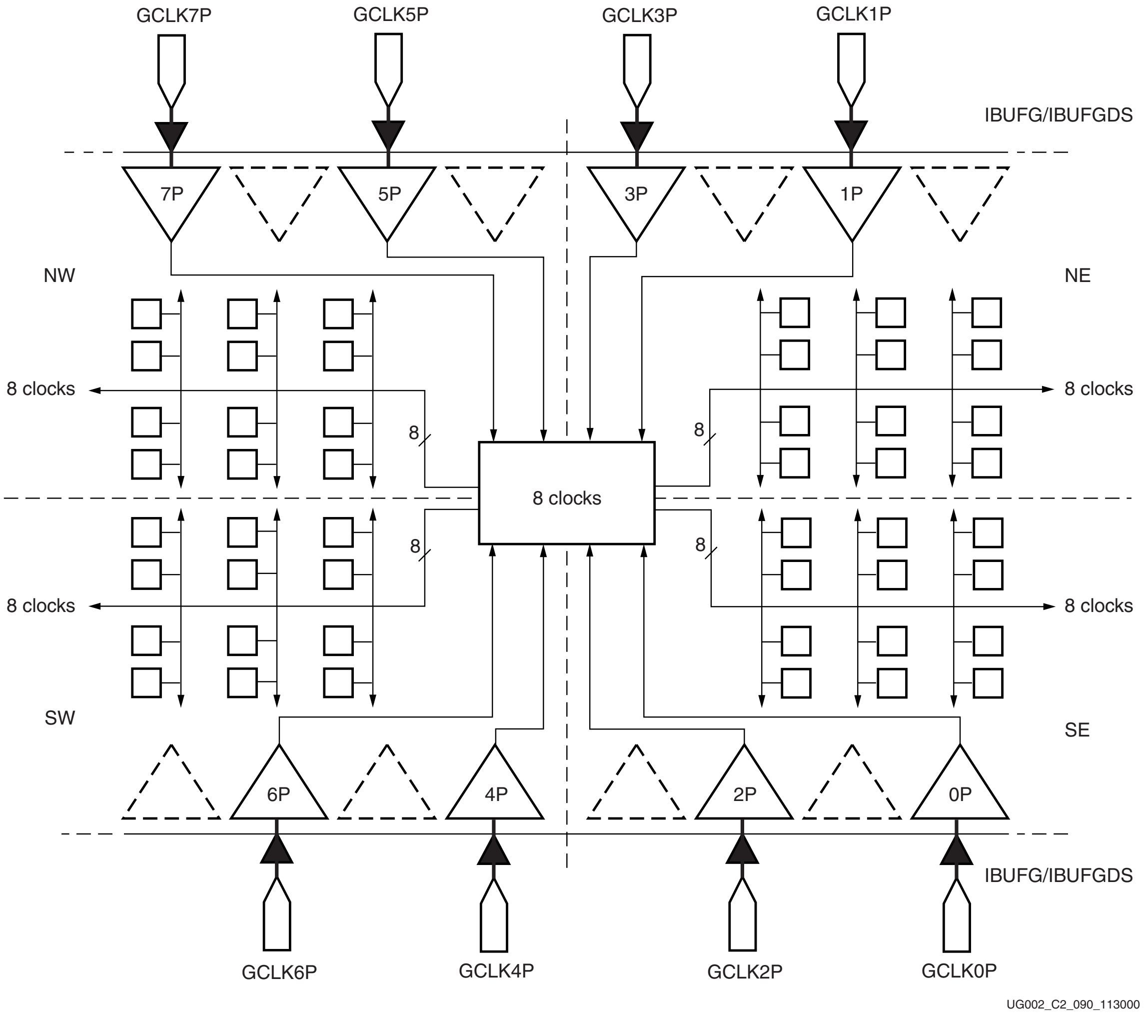


Figure 2-9: **Eight Global Clocks Design**

DCM Clocks

The four clock pins (IBUFG) in a quadrant can feed all DCMs in the same edge of the device. The clock-to-out and setup times are identical for all DCMs. Up to four clock outputs per DCM can be used to drive any clock multiplexer on the same edge (top or bottom), as shown in Figure 2-10.

BUFG Exclusivity

Each DCM has a restriction on the number of BUFGs it can drive on its (top or bottom) edge. Pairs of buffers with shared dedicated routing resources exist such that only one buffer from each dedicated pair can be driven by a single DCM. The exclusive pairs for each edge are: 1:5, 2:6, 3:7, and 4:8.

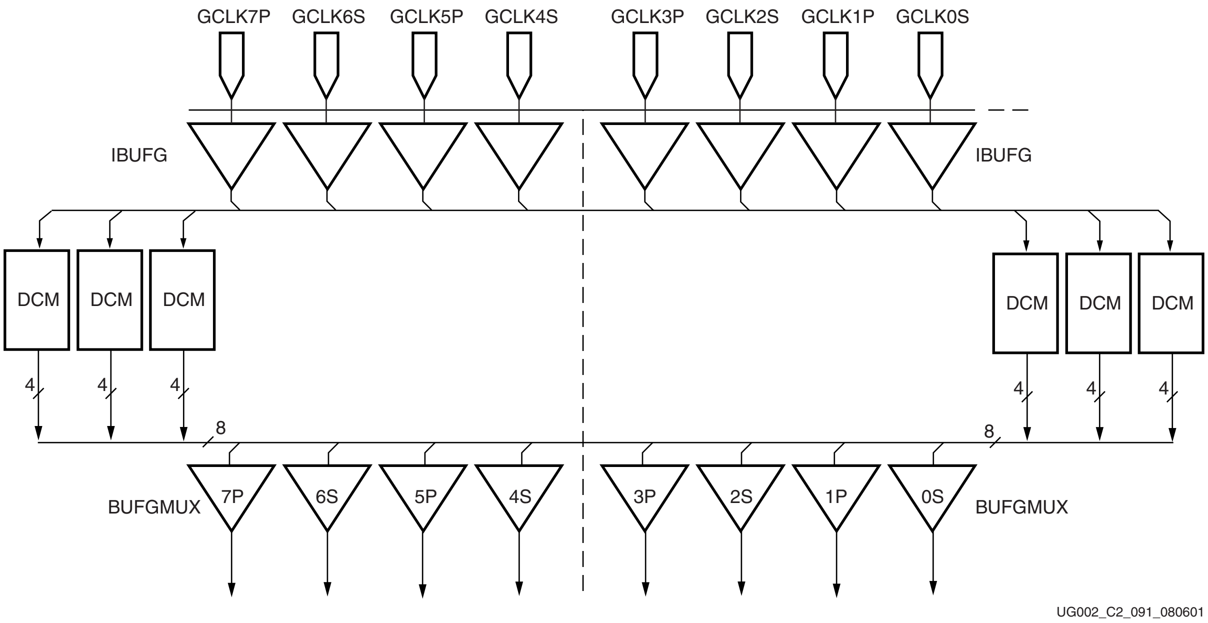


Figure 2-10: **DCM Clocks**

Clock Output

The clock distribution is based on eight clock trees per quadrant. Each clock multiplexer output is driving one global clock net. The Virtex-II device has eight dedicated low-skew clock nets. The device is divided into four quadrants (NW, NE, SW and SE) with eight global clocks available per quadrant.

Eight clock buffers are in the middle of the top edge and eight are in the middle of the bottom edge. Any of these 16 clock buffer outputs can be used in any quadrant, up to a maximum of eight clocks per quadrant, as illustrated in Figure 2-11, provided there is not a primary vs. secondary conflict.

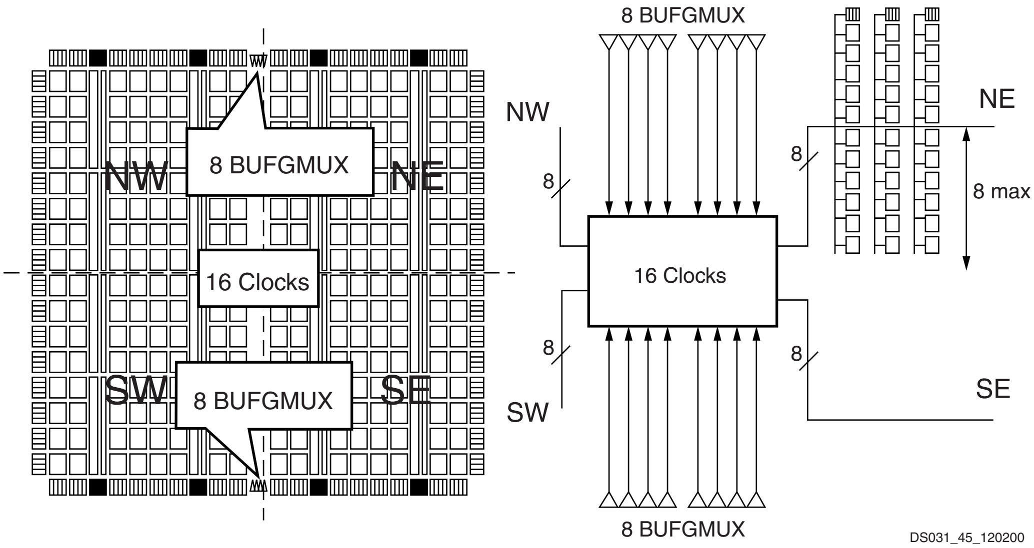


Figure 2-11: **Clock Buffer Outputs per Quadrant**

Designs with more than eight clocks must be floorplanned manually or automatically, distributing the clocks in each quadrant. As an example, a design with 16 clocks can be floorplanned as shown in Figure 2-12.

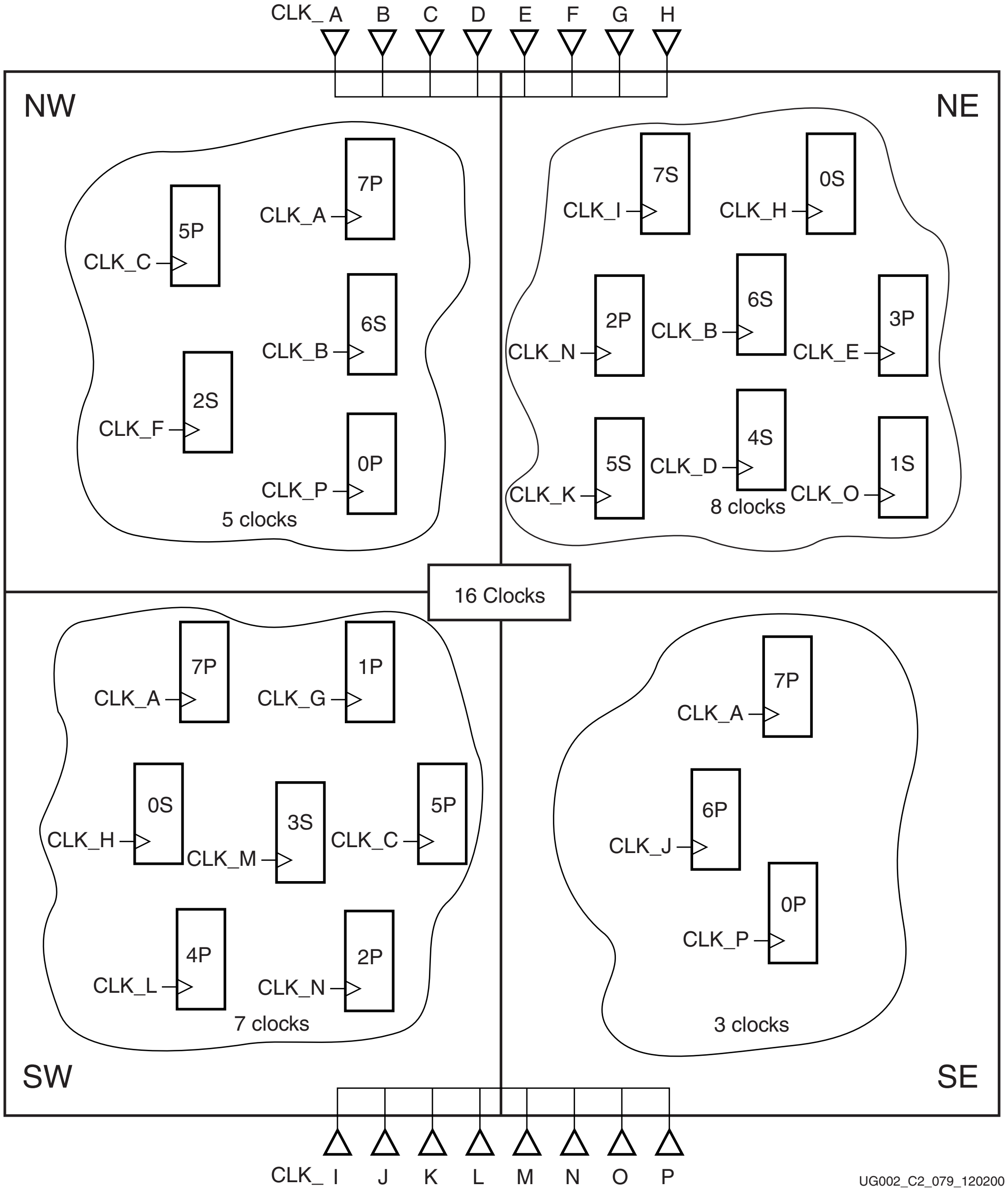


Figure 2-12: **16-Clock Floorplan**

The clock nets and clock buffers in this example are associated as shown in Table 2-4.

Table 2-4: **Clock Net Association With Clock Buffers**

Clock Net (top edge)	CLK_A	CLK_B	CLK_C	CLK_D	CLK_E	CLK_F	CLK_G	CLK_H
BUFG	7P	6S	5P	4S	3P	2S	1P	0S
Clock Net (bottom edge)	CLK_I	CLK_J	CLK_K	CLK_L	CLK_M	CLK_N	CLK_O	CLK_P
BUFG	7S	6P	5S	4P	3S	2P	1S	0P
Quadrant NW	CLK_A	CLK_B	CLK_C	–	–	CLK_F	–	CLK_P
Quadrant SW	CLK_A	–	CLK_C	CLK_L	CLK_M	CLK_N	CLK_G	CLK_H
Quadrant NE	CLK_I	CLK_B	CLK_K	CLK_D	CLK_E	CLK_N	CLK_O	CLK_H
Quadrant SE	CLK_A	CLK_J	–	–	–	–	–	CLK_P

CLK_A is used in three quadrants, and the other clocks are used in one or two quadrants, regardless of the position of the clock buffers (multiplexers), as long as they are not competing to access the same quadrant. (That is, CLK_A (BUFG7P) cannot be used in the same quadrant with CLK_I (BUFG7S). Refer to "Primary/Secondary: Rule 1" on page 160.) In other words, two buffers with the same index (0 to 7) cannot be used in the same quadrant.

Each register, block RAM, registered multiplier, or DDR register (IOB) can be connected to any of the eight clock nets available in a particular quadrant.

Note that if a global clock (primary buffer) is used in four quadrants, the corresponding secondary buffer is not available.

Power Consumption

Clock trees have been designed for low skew and low-power operation. Any unused branch is disconnected, as shown in Figure 2-13.

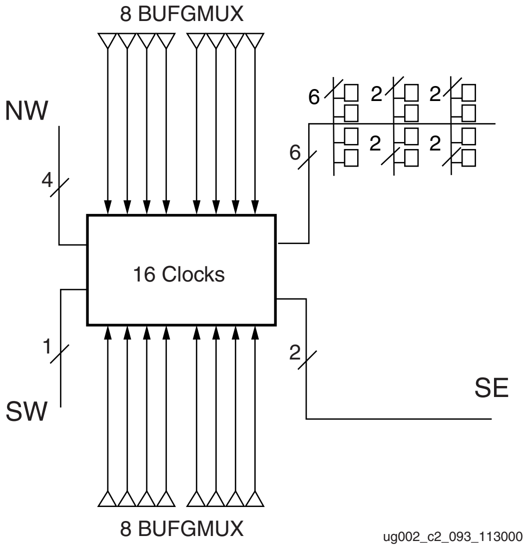


Figure 2-13: **Low-Power Clock Network**

Also available to reduce overall power consumption are the BUFGCE feature, for dynamically driving a clock tree only when the corresponding module is used, and the BUFGMUX feature, for switching from a high-frequency clock to a low-frequency clock.

The frequency synthesizer capability of the DCM can generate the low (or high) frequency clock from a single source clock, as illustrated in Figure 2-14. (See "Using Digital Clock Managers (DCMs)" on page 175).

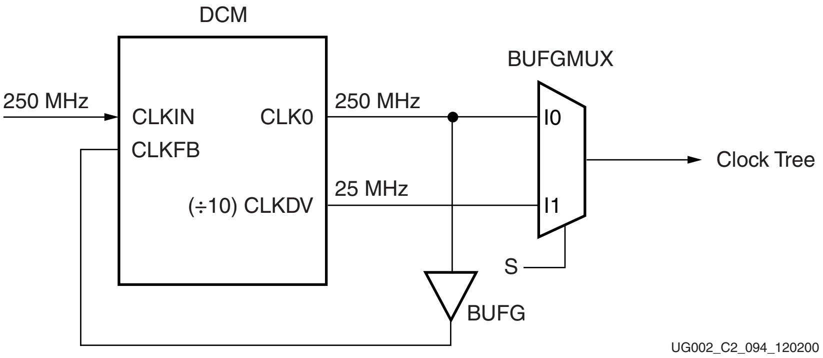


Figure 2-14: **Dynamic Power Reduction Scheme**

Library Primitives and Submodules

The primitives in Table 2-5 are available with the input, output, and control pins listed.

Table 2-5: **Clock Primitives**

Primitive	Input	Output	Control
IBUFG	I	O	–
IBUFGDS	I, IB	O	–
BUFG	I	O	–
BUFGMUX	I0, I1	O	S
BUFGMUX_1	I0, I1	O	S

Refer to "Using Single-Ended SelectI/O Resources" on page 258 for a list of the attributes available for IBUFG and Refer to "Using LVDS I/O" on page 317 for a list of the attributes available for IBUFGDS.

The submodules in Table 2-6 are available with the input, output, and control pins listed.

Table 2-6: **Clock Submodules**

Submodule	Input	Output	Control
BUFGCE	I	O	CE
BUFGCE_1	I	O	CE

Primitive Functions

IBUFG

IBUFG is an input clock buffer with one clock input and one clock output.

IBUFGDS

IBUFGDS is a differential input clock buffer with two clock inputs (positive and negative polarity) and one clock output.

BUFG

All Virtex-II devices have 16 global clock buffers (each of which can be used as BUFG, BUFGMUX, or BUFGCE).

BUFG is a global clock buffer with one clock input and one clock output, driving a low-skew clock distribution network. The output follows the input, as shown in Figure 2-15.

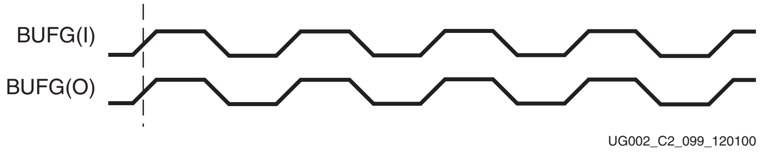


Figure 2-15: **BUFG Waveforms**

BUGMUX and BUFGMUX_1

BUFGMUX (see Figure 2-16) can switch between two unrelated, even asynchronous clocks. Basically, a Low on S selects the CLK0 input, a High on S selects the S1 input. Switching from one clock to the other is done in such a way that the output High and Low time is never shorter than the shortest High or Low time of either input clock. As long as the presently selected clock is High, any level change of S has no effect .

BUFGMUX is the preferred circuit for rising edge clocks, while BUFGMUX_1 is preferred for falling edge clocks.

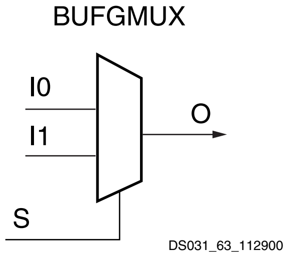


Figure 2-16: **Virtex-II BUFGMUX or BUFGMUX_1 Function**

Operation of the BUFGMUX Circuit

If the presently selected clock is Low while S changes, or if it goes Low after S has changed, the output is kept Low until the other ("to-be-selected") clock has made a transition from High to Low. At that instant, the new clock starts driving the output.

The two clock inputs can be asynchronous with regard to each other, and the S input can change at any time, except for a short setup time prior to the rising edge of the presently selected clock; that is, prior to the rising edge of the BUFGMUX output O. Violating this setup time requirement can result in an undefined runt pulse output.

Figure 2-17 shows a switchover from CLK0 to CLK1.

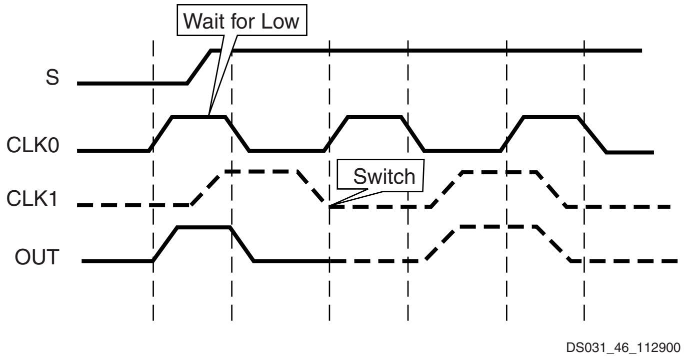


Figure 2-17: **BUFGMUX Waveform Diagram**

- The current clock is CLK0.
- S is activated High.
- If CLK0 is currently High, the multiplexer waits for CLK0 to go Low.
- Once CLK0 is Low, the multiplexer output stays Low until CLK1 transitions High to Low.
- When CLK1 transitions from High to Low, the output switches to CLK1.
- No glitches or short pulses can appear on the output.

Operation of the BUFGMUX_1 Circuit

If the presently selected clock is High while S changes, or if it goes High after S has changed, the output is kept High until the other ("to-be-selected") clock has made a transition from Low to High. At that instant, the new clock starts driving the output.

The two clock inputs can be asynchronous with regard to each other, and the S input can change at any time, except for a short setup time prior to the falling edge of the presently selected clock; that is, prior to the falling edge of the BUFGMUX output O. Violating this setup time requirement can result in an undefined runt pulse output.

Figure 2-18 shows a switchover from CLK0 to CLK1.

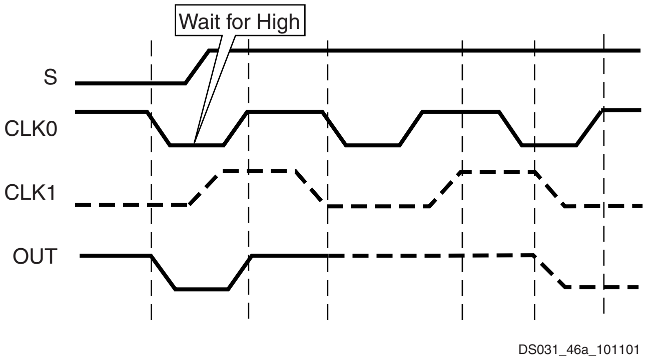


Figure 2-18: **BUFGMUX_1 Waveform Diagram**

- The current clock is CLK0.
- S is activated High.
- If CLK0 is currently Low, the multiplexer waits for CLK0 to go High.
- Once CLK0 is High, the multiplexer output stays High until CLK1 transitions Low to High.
- When CLK1 transitions from Low to High, the output switches to CLK1.
- No glitches or short pulses can appear on the output.

Submodules

BUFGCE and BUFGCE_1

BUFGCE and BUFGCE_1 are submodules based on BUFGMUX and BUFGMUX_1, respectively. BUFGCE and BUFGCE_1 are global clock buffers incorporating a smart enable function that avoids output glitches or runt pulses. The select signal must meet the setup time for the clock.

BUFGCE is the preferred circuit for clocking on the rising edge, while BUFGCE_1 is preferred when clocking on the falling edge.

Operation of the BUFGCE Circuit

If the CE input (see Figure 2-19) is active (High) prior to the incoming rising clock edge, this Low-to-High-to-Low clock pulse passes through the clock buffer. Any level change of CE during the incoming clock High time has no effect.

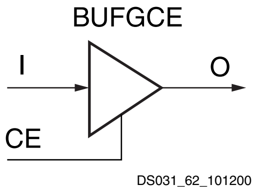


Figure 2-19: **Virtex-II BUFGCE or BUFGCE_1 Function**

If the CE input is inactive (Low) prior to the incoming rising clock edge, the following clock pulse does not pass through the clock buffer, and the output stays Low. Any level change of CE during the incoming clock High time has no effect. CE must not change during a short setup window just prior to the rising clock edge on the BUFGCE input I. Violating this setup time requirement can result in an undefined runt pulse output.

This means the output stays Low when the clock is disabled, but it completes the clock-High pulse when the clock is being disabled, as shown in Figure 2-20.

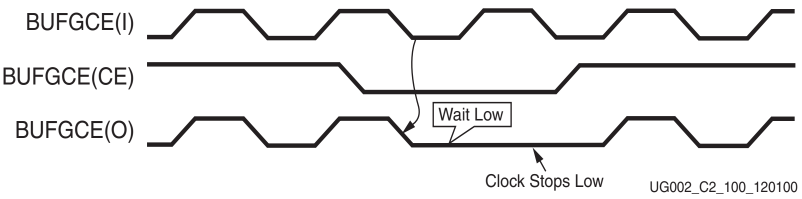


Figure 2-20: **BUFGCE Waveforms**

Operation of the BUFGCE_1 circuit

If the CE input is active (High) prior to the incoming falling clock edge, this High-to-Low-to-High clock pulse passes through the clock buffer. Any level change of CE during the incoming clock Low time has no effect.

If the CE input is inactive (Low) prior to the incoming falling clock edge, the following clock pulse does not pass through the clock buffer, and the output stays High. Any level change of CE during the incoming clock Low time has no effect. CE must not change during a short setup window just prior to the falling clock edge on the BUFGCE input I. Violating this setup time requirement can result in an undefined runt pulse output.

This means the output stays High when the clock is disabled, but it completes the clock-Low pulse when the clock is being disabled, as shown in Figure 2-21.

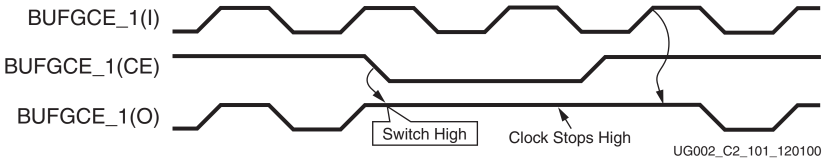


Figure 2-21: **BUFGCE_1 Waveforms**

When BUFGCE (or BUFGCE_1) is used with DCM outputs, a second BUFG can be used for clock feedback. Buffer sharing the inputs with BUFGCE is the preferred solution.

Summary

Table 2-7 shows the maximum resources available per Virtex-II device.

Table 2-7: **Resources per Virtex-II Device (from XC2V40 to XC2V8000)**

Resource	Maximum Number
Single-ended IBUFG (pads)	16
Differential IBUFGDS (pairs)	8
BUFG (Global Clock Buffer)	16
BUFGCE (or BUFGCE_1)	8
BUFGMUX (or BUFGMUX_1)	8

Characteristics

The following are characteristics of global clocks in Virtex-II devices:

- Low-skew clock distribution.
- Synchronous "glitch-free" multiplexer that avoids runt pulses. Switching between two asynchronous clock sources is usually considered unsafe, but it is safe with the Virtex-II global clock multiplexer.

- Any level change on S must meet a setup time requirement with respect to the signal on the output O (rising edge for BUFGMUX, falling edge for BUFGMUX_1). Any level change on CE must meet a setup time requirement with respect to the signal on the Input I (rising edge for BUFGCE, falling edge for BUFGCE_1).
- Two BUFGMUX (or BUFGMUX_1) resources can be cascaded to create a 3 to 1 clock multiplexer.

Location Constraints

BUFGMUX and BUFGMUX_1 (primitives) and IBUFG (IBUFGDS) instances can have LOC properties attached to them to constrain placement. The LOC properties use the following form to constrain a clock net:

```
NET “clock_name” LOC=”BUFGMUX#P/S”;
```

Each clock pad (or IBUFG) has a direct connection with a specific global clock multiplexer (input I0). A placement that does not conform to this rule causes the software to send a warning.

If the clock pad (or IBUFG) has LOC properties attached, the DCM allows place and route software maximum flexibility, as compared to a direct connection to the global clock buffer (BUFG).

Secondary Clock Network

If more clocks are required, the 24 horizontal and vertical long lines in Virtex-II devices can be used to route additional clock nets. Skew is minimized by the place and route software, if the USELOWSKEWLINES constraint is attached to the net.

VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available as examples (see “VHDL and Verilog Templates” on page 170) for all primitives and submodules.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

VHDL and Verilog Templates

The following are templates for primitives:

- BUFGMUX_INST
- BUFGMUX_1_INST

The following are templates for submodules:

- BUFGCE_SUBM
- BUFGCE_1_SUBM

As examples, the BUFGMUX_INST.vhd, BUFGMUX_1_INST.vhd, BUFGCE_SUBM.vhd, and BUFGCE_1_SUBM.vhd VHDL templates are shown. In addition, the BUFGMUX_INST.v, BUFGMUX_1_INST.v, BUFGCE_1_SUBM.v, and BUFGCE_SUBM.v Verilog templates are shown.

VHDL Template

```
-- Module: BUFGMUX_INST
-- Description: VHDL instantiation template
-- Global Clock Multiplexer (Switch Low)
-- Device: Virtex-II Family
---------------------------------------------------------------------
-- Component Declarations:
--
```

```
component BUFGMUX
  port (
  I0   : in std_logic;
        I1   : in std_logic;
        S    : in std_logic;
        O    : out std_logic
  );
end component;
--
-- Architecture  section:
--
-- Global Clock Buffer Instantiation
U_BUFGMUX: BUFGMUX
  port map (
  I0      => , -- insert clock input used when select (S) is Low
  I1      => , -- insert clock input used when select (S) is High
  S       => , -- insert Mux-Select input
  O       =>   -- insert clock output
  );
--
---------------------------------------------------------------------
-- Module: BUFGMUX_1_INST
-- Description: VHDL instantiation template
--  Global Clock Multiplexer (Switch High)
--
-- Device: Virtex-II Family
---------------------------------------------------------------------
-- Component Declarations:
component BUFGMUX_1
  port (
  I0   : in std_logic;
        I1   : in std_logic;
        S    : in std_logic;
        O    : out std_logic
  );
end component;
--
-- Architecture  section:
--
-- Global Clock Buffer Instantiation
U_BUFGMUX_1: BUFGMUX_1
  port map (
  I0      => , -- insert clock input used when select (S) is Low
  I1      => , -- insert clock input used when select (S) is High
  S       => , -- insert Mux-Select input
  O       =>   -- insert clock output
  );
--
---------------------------------------------------------------------
-- Module: BUFGCE_SUBM
-- Description: VHDL instantiation template
-- Global Clock Buffer with Clock Enable:
-- Input Clock Buffer to BUFGMUX - Clock disabled = Low
-- Device: Virtex-II Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--
-- pragma translate_off
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;
-- pragma translate_on
```

```
--
entity BUFGCE_SUBM is
  port (
        I:  in std_logic;
        CE: in std_logic;
        O:  out std_logic
       );
end BUFGCE_SUBM;
--
architecture BUFGCE_SUBM_arch of BUFGCE_SUBM is
--
-- Component Declarations:
component BUFGMUX
  port (
  I0   : in std_logic;
        I1    : in std_logic;
        S     : in std_logic;
        O     : out std_logic
  );
end component;
--
-- signal declarations
signal GND : std_logic;
signal CE_B : std_logic;
--
begin
GND <= '0';
--
CE_B <= not CE;
--
-- Global Clock Buffer Instantiation
U_BUFGMUX: BUFGMUX
  port map (
  I0      => I,
  I1      => GND,
  S       => CE_B,
  O       => O
  );
--
end BUFGCE_SUBM_arch;
---------------------------------------------------------------------
-- Module: BUFGCE_1_SUBM
-- Description: VHDL instantiation template
-- Global Clock Buffer with Clock Enable:
-- Input Clock Buffer to BUFGMUX_1 - Clock disabled = High
-- Device: Virtex-II Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--
-- pragma translate_off
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;
-- pragma translate_on
--
entity BUFGCE_1_SUBM is
  port (
        I:  in std_logic;
        CE: in std_logic;
        O:  out std_logic
       );
end BUFGCE_1_SUBM;
```

```
--
architecture BUFGCE_1_SUBM_arch of BUFGCE_1_SUBM is
--
-- Component Declarations:
component BUFGMUX_1
  port (
  I0    : in std_logic;
        I1    : in std_logic;
        S     : in std_logic;
        O     : out std_logic
  );
end component;
--
-- signal declarations
signal VCC : std_logic;
--
signal CE_B : std_logic;
--
begin
VCC <= '1';
--
CE_B <= not CE;
--
-- Global Clock Buffer Instantiation
U_BUFGMUX_1: BUFGMUX_1
  port map (
  I0      => I,
  I1      => VCC,
  S       => CE_B,
  O       => O
  );
--
end BUFGCE_1_SUBM_arch;
```

Verilog Template

```
//-------------------------------------------------------------------
// Module:      BUFGMUX_INST
// Description: Verilog Instantiation Template
//  Global Clock Multiplexer (Switch Low)
//
//
// Device: Virtex-II Family
//-------------------------------------------------------------------
//
//BUFGMUX Instantiation
BUFGMUX  U_BUFGMUX
              (.I0(),  // insert clock input used when select(S) is Low
              .I1(),  // insert clock input used when select(S) is High
               .S(),   // insert Mux-Select input
               .O()    // insert clock output
              );
//-------------------------------------------------------------------
// Module:      BUFGMUX_1_INST
// Description: Verilog Instantiation Template
//  Global Clock Multiplexer (Switch High)
//
//
// Device: Virtex-II Family
//-------------------------------------------------------------------
//
//BUFGMUX_1 Instantiation
BUFGMUX_1  U_BUFGMUX_1
```

```
               (.I0(),  // insert clock input used when select(S) is Low
               .I1(),  // insert clock input used when select(S) is High
                .S(),   // insert Mux-Select input
                .O()    // insert clock output
               );
//-----------------------------------------------------------------
// Module:      BUFGCE_SUBM
// Description: Verilog Submodule
//  Global Clock Buffer with Clock Enable:
//  Input Clock Buffer to BUFGMUX - Clock disabled = Low
//
// Device: Virtex-II Family
//-----------------------------------------------------------------
module BUFGCE_SUBM (I,
                    CE,
                    O);
input   I,
        CE;

output  O;

wire GND;

assign GND = 1'b0;

BUFGMUX U_BUFGMUX
             (.I0(I),
              .I1(GND),
              .S(~CE),
              .O(O)
             );
//
endmodule
//-----------------------------------------------------------------
// Module: BUFGCE_1_SUBM
// Description: Verilog Submodule
// Global Clock Buffer with Clock Enable:
// Input Clock Buffer to BUFGMUX_1 - Clock disabled = High
//
// Device: Virtex-II Family
//-----------------------------------------------------------------

module BUFGCE_1_SUBM (I,
                      CE,
                      O);
input   I,
        CE;

output  O;

wire VCC;

assign VCC = 1'b1;

BUFGMUX_1 U_BUFGMUX_1
             (.I0(I),
              .I1(VCC),
              .S(~CE),
              .O(O)
             );
//
endmodule
```

Using Digital Clock Managers (DCMs)

Overview

Virtex-II devices have 4 to 12 DCMs, and each DCM provides a wide range of powerful clock management features:

- **Clock De-skew**: The DCM contains a digitally-controlled feedback circuit (delay-locked loop) that can completely eliminate clock distribution delays. Clock de-skew works as follows:

  The incoming clock drives a long chain of delay elements (individual small buffers). A wide multiplexer selects any one of these buffers as an output. A controller drives the select inputs of this multiplexer. The phase detector in this controller compares the incoming clock signal (CLKIN) against a feedback input (CLKFB), which must be another version of the same clock signal, usually from the far end of the internal clock distribution network (but it can also be from an output pin).

  The phase detector steers the controller to adjust the tap selection, and thus the through-delay in the DCM, in such a way that the two inputs to the phase comparator coincide.(This is a typical servo loop.) The tap controller adds exactly the right amount of delay to the clock distribution network to give it a total delay of one full clock period. For a repetitive clock signal, this effectively eliminates the clock distribution delay completely.

- **Frequency Synthesis**: Separate outputs provide a doubled frequency (CLK2X and CLK2X180). Another output (CLKDV) provides a frequency that is a specified fraction of the input frequency (÷1.5, ÷2, ÷2.5, and so forth, up to ÷15 and ÷16.)

  Two other outputs (CLKFX and CLKFX180) provide an output frequency that is derived from the input clock by simultaneous frequency division and multiplication. The user can specify any integer multiplier (M) and divisor (D) within the range specified in the DCM Timing Parameters section of the Virtex-II Data Sheet. An internal calculator figures out the appropriate tap selection, so that the output edge coincides with the input clock whenever that is mathematically possible. For example, M=9 and D=5, multiply the frequency by 1.8, and the output rising edge is coincident with the input rising edge every 5 input periods = every 9 output periods.

- **Phase Shifting**: Three outputs drive the same frequency as CLCK0 but are delayed by 1/4, 1/2, and 3/4 of a clock period. An additional control optionally shifts all nine clock outputs by a fixed fraction of the clock period (defined during configuration, and described in multiples of the clock period divided by 256).

  The user can also dynamically and repetitively move the phase forwards or backwards by one unit of the clock period divided by 256. Note that any such phase shift is always invoked as a specific fraction of the clock period, but is always implemented by moving delay taps with a resolution of DCM_TAP (see DCM Timing Parameters in the Virtex-II Data Sheet).

- **General Control Signals**: The RST input, when High, resets the entire DCM. The LOCKED output is High when all enabled DCM circuits have locked. The active High STATUS outputs indicate the following:
  - Phase Shift Overflow (STATUS[0])
  - CLKIN Stopped (STATUS[1])
  - CLKFX Stopped (STATUS[2])

Clock De-Skew

The Virtex-II Digital Clock Manager (DCM) offers a fully digital, dedicated on-chip de-skew circuit providing zero propagation delay, low clock skew between output clock signals distributed throughout the device, and advanced clock domain control. These features can be used to implement several circuits that improve and simplify system level design.

Any four of the nine outputs of the DCM can be used to drive a global clock network. All DCM outputs can drive general interconnect at the same time; for example, DCM output can be used to generate board-level clocks. The well-buffered global clock distribution network minimizes clock skew caused by loading differences. By monitoring a sample of the output clock (CLK0 or CLK2X), the de-skew circuit compensates for the delay on the routing network, effectively eliminating the delay from the external input port to the individual clock loads within the device.

Figure 2-22 shows all of the inputs and outputs relevant to the DCM de-skew feature.

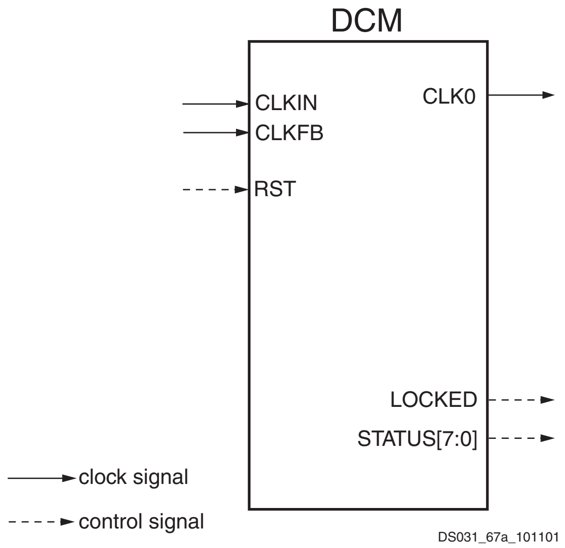


Figure 2-22: **Clock De-Skew Outputs**

The de-skew feature can also act as a clock mirror. By driving the CLK0 or CLK2X output off-chip and then back in again, the de-skew feature can be used to de-skew a board-level clock serving multiple devices.

By taking advantage of the de-skew circuit to remove on-chip clock delay, the designer can greatly simplify and improve system level design involving high-fanout, high-performance clocks.

Operation

A de-skew circuit in its simplest form consists of variable delay line and control logic. The delay line produces a delayed version of the input clock (CLKIN). The clock distribution network routes the clock to all internal registers and to the clock feedback CLKFB pin. The control logic samples the input clock, as well as the feedback clock, and adjusts the delay line.

For optimum performance, the Virtex-II DCM uses a discrete digital delay line, which is a series of buffer elements each with an intrinsic delay of less than DCM_TAP (see AC characteristics in the Virtex-II Data Sheet).

A de-skew circuit works by inserting delay between the input clock and the feedback clock until the two rising edges align, putting the two clocks 360 degrees out of phase, which means they are in phase. When the edges from the input clock line up with the edges from the feedback clock, the DCM achieves "lock." The two clocks have no discernible difference. Thus, the DCM output clock compensates for the delay in the clock distribution network, effectively removing the delay between the source clock and its loads.

Input Clock Requirements

The clock input of the DCM can be driven either by an IBUFG, an IBUF, or a BUFGMUX. An LVDS clock can also be used as input.

The output clock signal of a DCM, essentially a delayed version of the input clock signal, reflects any instability on the input clock in the output waveform. A DCM cannot improve the input jitter. The DCM input clock requirements are specified in the **Virtex-II Data Sheet**.

Once locked, the DCM can tolerate input clock period variations of up to the value specified by CLKIN_CYC_JITT_DLL_HF (at high frequencies) or CLKIN_CYC_JITT_DLL_LF (at low frequencies). Larger frequency changes can cause the DCM to lose lock, which is indicated by the LOCKED output going low. The user must then reset the DCM. The cycle-to-cycle input jitter must be kept to less than CLKIN_PER_JITT_DLL_LF in the low frequencies and CLKIN_PER_JITT_DLL_HF for the high frequencies.

Input Clock Changes

Changing the period of the input clock beyond the maximum drift amount requires a manual reset of the DCM. Failure to reset the DCM produces an unreliable lock signal and output clock.

It is possible to stop the input clock with little impact to the de-skew circuit. The clock should be stopped for no more than 100 ms to minimize the effect of device cooling, which would change the tap delays. The clock should be stopped during a Low phase, and when restored, must generate a full High half-period. During this time, LOCKED stays High and remains High when the clock is restored. So a High on LOCKED does not necessarily mean that a valid clock is available.

When the clock is being stopped, one to four more clock cycles are still generated as the delay line is flushed. When the clock is restarted, the output clock cycles are not generated for one to four clocks as the delay line is filled. The most common case is two or three clocks. In a similar manner, a phase shift of the input clock is also possible. The phase shift propagates to the output one to four clocks after the original shift, with no disruption to the DCM control.

Output Clocks

Some restrictions apply regarding the connectivity of the output pins. The DCM clock outputs can each drive an OBUF, a global clock buffer BUFGMUX, or they can route directly to the clock input of a synchronous element. The DCM clock outputs can drive BUFGMUXs that are on the same edge of the device (top or bottom).

Do not use the DCM output clock signals until after activation of the LOCKED signal. Prior to the activation of the LOCKED signal, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement.

Characteristics of the De-Skew Circuit

- Can eliminate clock distribution delay by effectively adding one clock period delay. Clocks are de-skewed to within CLKOUT_PHASE, specified in the **Virtex-II Data Sheet**.
- Can be used to eliminate on-chip as well as off-chip clock delay.
- Has no restrictions on the delay in the feedback clock path.
- Requires a continuously running input clock.
- Adapts to a wide range of frequencies. However, once locked to a frequency, cannot tolerate large variations of the input frequency.
- De-skew circuit is part of the DCM, which also includes phase adjustment, frequency synthesis, and spread spectrum techniques that are described in this document.
- Does not eliminate jitter. The de-skew circuit output jitter is the sum of input jitter and some jitter value that the de-skew circuit might add.
- The completion of configuration can be delayed until after DCM locks to guarantee the system clock is established prior to initiating the device.

Port Signals

Source Clock Input — CLKIN

The CLKIN pin provides the user source clock (the clock signal on which the de-skew circuit operates) to the DCM. The CLKIN frequency must fall in the ranges specified in the **Virtex-II Data Sheet**. The clock input signal can be provided by one of the following:

IBUF — Input buffer
IBUFG — Global clock input buffer on the same edge of the device (top or bottom)
BUFGMUX — Internal global clock buffer

Feedback Clock Input — CLKFB

A reference or feedback signal is required to delay-compensate the output. Connect only the CLK0 or CLK2X DCM outputs to the feedback clock input (CLKFB) pin to provide the necessary feedback to the DCM. The feedback clock input signal can be driven by an internal global clock buffer (BUFGMUX), one of the global clock input buffers (IBUFG) on the same edge of the device (top or bottom), or IBUF (the input buffer.)

If an IBUFG sources the CLKFB pin, the following special rules apply:

1. An external input port must source the signal that drives the IBUFG input pin.
2. That signal must directly drive only OBUFs and nothing else.

Reset Input — RST

When the reset pin is activated, the LOCKED signal deactivates within four source clock cycles. The RST pin, active High, must either connect to a dynamic signal or be tied to ground. As the DCM delay taps reset to zero, glitches can occur on the DCM clock output pins. Activation of the RST pin can also severely affect the duty cycle of the clock output pins. Furthermore, the DCM output clocks no longer de-skew with respect to one another. For these reasons, use the reset pin only when reconfiguring the device or changing the input frequency. The reset input signal is asynchronous and should be held HIGH for at least 2 ns. It takes approximately 120 µs for the DCM to achieve lock after a reset in the slowest frequency range. The DCM locks faster at higher frequencies. See the LOCK_DLL teiming parameter in the **Virtex-II Data Sheet**.

Locked Output — LOCKED

In order to achieve lock, the DCM may need to sample several thousand clock cycles. After the DCM achieves lock, the LOCKED signal goes High. The DCM timing parameter section of the **Virtex-II Data Sheet** provides estimates for locking times.

To guarantee that the system clock is established prior to the device “waking up,” the DCM can delay the completion of the device configuration process until after the DCM locks. The STARTUP_WAIT attribute activates this feature.

Until the LOCKED signal activates, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. In particular, the CLK2X output appears as a 1x clock with a 25/75 duty cycle.

Status - STATUS

The STATUS output is an 8-bit output, of which STATUS[1] reveals the loss of the input clock, CLKIN to the DCM.

Attributes

The following attributes provide access to some of the Virtex-II series de-skew features, (for example, clock division and duty cycle correction).

Frequency Mode

The de-skew feature of the DCM is achieved with a delay-locked loop (DLL). This attribute specifies either the high or low-frequency mode of the DLL. The default is low-frequency mode. In high-frequency mode, the only outputs available from the DLL are the CLK0, CLK180, CLKDV, and LOCKED. (CLK90, CLK270, CLK2X, and CLK2X180 are not available in high-frequency mode.) The frequency ranges for both frequency modes are specified in the **Virtex-II Data Sheet**. To set the DLL to high-frequency mode, attach the DLL_FREQUENCY_MODE=HIGH attribute in the source code or schematic.

Feedback Input

This attribute specifies the feedback input to the DCM (CLK0, or CLK2x). CLK0 is the default feedback. When both the CLK0 and the CLK2x outputs are used internally or externally to the device, the feedback input can be either the CLK0 or CLK2x. In order to set the feedback to CLK2X, attach the CLOCK_FEEDBACK=2X attribute in the source code or schematic.

Duty Cycle Correction

The 1x clock outputs, CLK0, CLK90, CLK180, and CLK270, use the duty cycle corrected default such that they exhibit a 50/50 duty cycle. The DUTY_CYCLE_CORRECTION attribute (by default TRUE) controls this feature.

To deactivate the DCM duty cycle correction for the 1x clock outputs, attach the DUTY_CYCLE_CORRECTION=FALSE attribute in the source code or schematic. This makes the output clocks have the same duty cycle as the source clock.

Startup Delay

The default value of the STARTUP_WAIT attribute is FALSE. When STARTUP_WAIT is set to TRUE, and the LCK_cycle BitGen option is used, then the configuration startup sequence waits in the specified cycle until the DCM locks. For details, see Chapter 3: Configuration and Appendix B: BitGen and PROMGen Switches and Options.

Legacy Support

The Virtex/Virtex-E library primitives/sub modules are supported in Virtex-II for legacy purposes. The following are supported primitives/submodules:

- CLKDLL
- CLKDLLE
- CLKDLLHF
- BUFGDLL

Library Primitive

Only a single library primitive is available for the DLL, a part of the DCM. It is labeled the 'DCM' primitive.

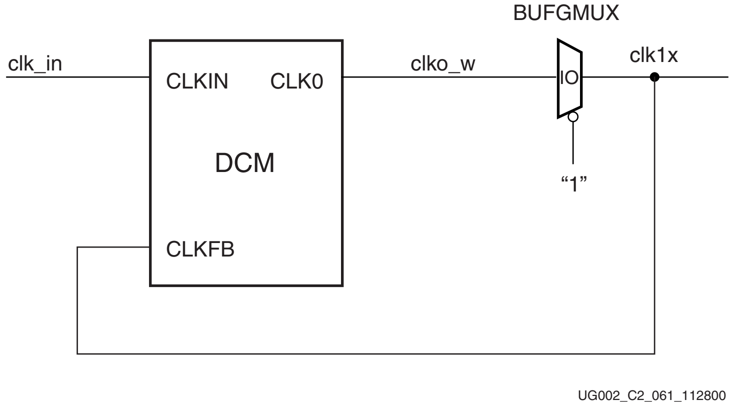


Figure 2-23: **BUFG_CLK0_SUBM**

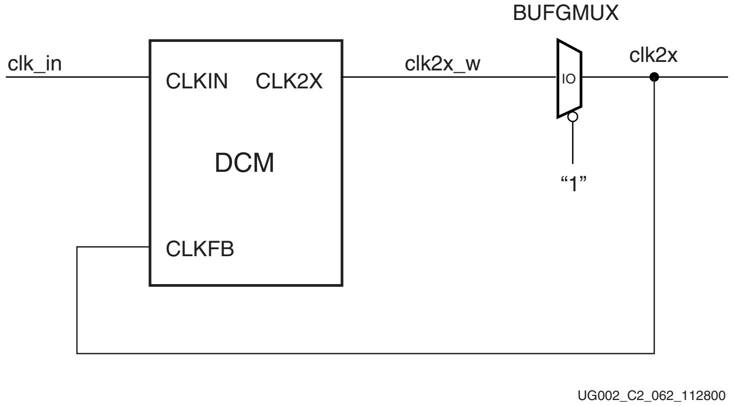


Figure 2-24: **BUFG_CLK2X_SUBM**

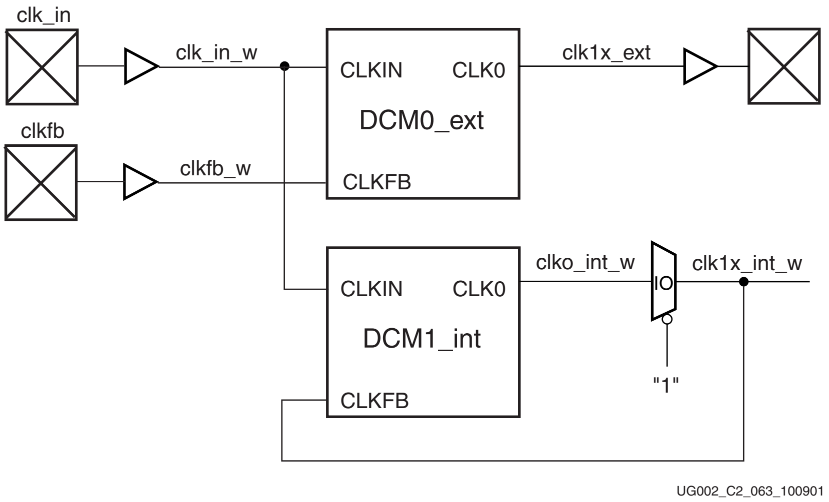


Figure 2-25: **BUFG_CLK0_FB_SUBM**

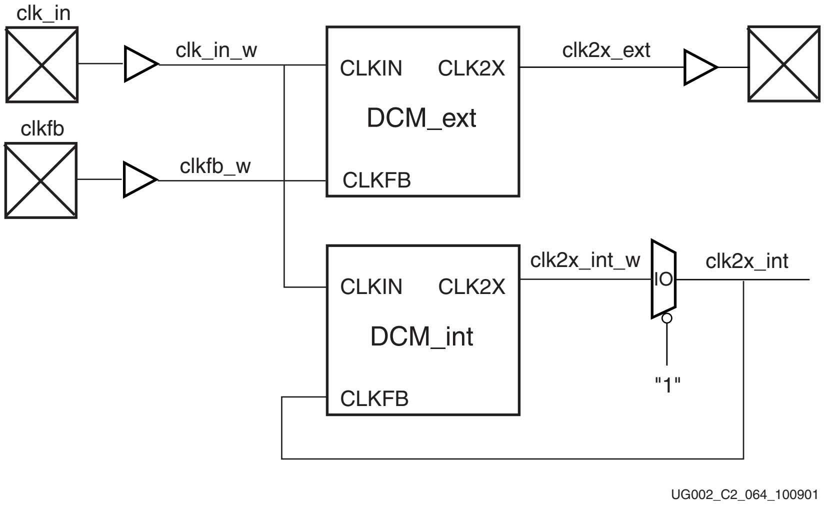


Figure 2-26: **BUFG_CLK2X_FB_SUBM**

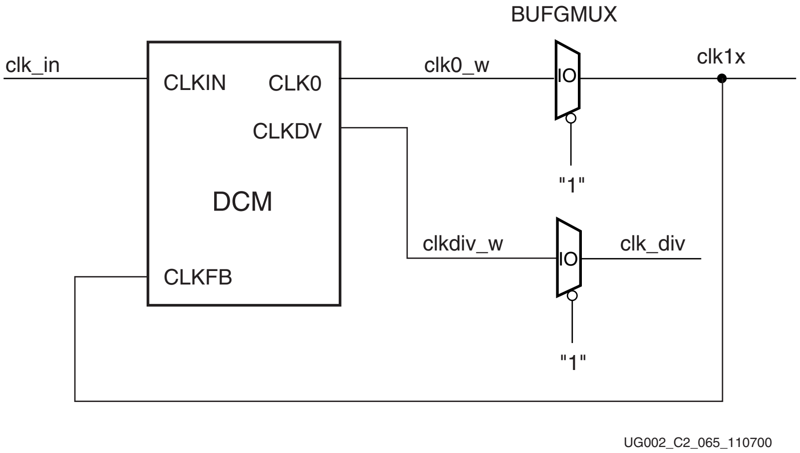


Figure 2-27: **BUFG_CLKDV_SUBM**

Frequency Synthesis

The DCM provides several flexible methods for generating new clock frequencies. Each method has a different operating frequency range and different AC characteristics. The CLK2X and CLK2X180 outputs double the clock frequency. The CLKDV output provides divided output clocks with division options of 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, and 16.

The DCM also offers a fully digital, dedicated Frequency Synthesizer output (CLKFX) and its opposite phase (CLKFX180). The output frequency can be any function of the input clock frequency described by M ÷ D, where M is the multipler (numerator) and D is the divisor (denominator).

The two counter-phase frequency synthesized outputs can drive global clock routing networks within the device. The well-buffered global clock distribution network minimizes clock skew due to differences in distance or loading. See Figure 2-28.

Operation

The DCM clock output CLKFX is any M/D product of the clock input to the DCM. Specifications for M and D, as well as input and output frequency ranges for the frequency synthesizer, are provided in the **Virtex-II Data Sheet**. The frequency synthesizer output is phase aligned to the clock output, CLK0, only if feedback is provided to the CLKFB input of the DCM.

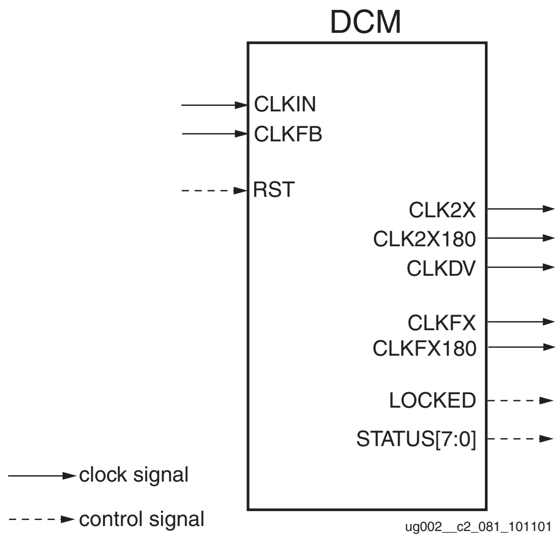


Figure 2-28: **Frequency Synthesis Outputs**

The internal operation of the frequency synthesizer is complex and beyond the scope of this document. The frequency synthesizer multiplies the incoming frequencies by the pre-calculated quotient M/D and generates the correct output frequencies as long as it is within the range specified in the **Virtex-II Data Sheet**.

For example, assume input frequency = 50 MHz, M = 25, and D = 8 (note that M and D values have no common factors and hence cannot be reduced). The output frequency is correctly 156.25 MHz, although 25 x 50 MHz = 1.25 GHz and 50 MHz / 8 = 6.25 MHz, and both of these values are far outside the range of the input frequency.

Frequency Synthesizer Characteristics

- The frequency synthesizer provides an output frequency equal to the input frequency multiplied by M and divided by D.
- The outputs CLKFX and CLKFX180 always have a 50/50 duty-cycle.
- Smaller M and D values achieve faster lock times. The user should divide M and D by the largest common factor.
- The outputs are phase aligned with CLK0 when CLKFB is connected.

Port Signals

Source Clock Input — CLKIN

The CLKIN pin provides the user source clock to the DCM. The CLKIN frequency must fall in the ranges specified in the **Virtex-II Data Sheet**. The clock input signal can be provided by one of the following:

- IBUF — Input buffer
- IBUFG — Global clock input buffer
- BUFGMUX — Internal global clock buffer

2x Clock Output — CLK2X

The CLK2X output provides a frequency-doubled clock with an automatic 50/50 duty-cycle correction. This output is not available in high-frequency mode.

Until the DCM has achieved lock, the CLK2X output appears as a 1x version of the input clock with a 25/75 duty cycle. This behavior allows the DCM to lock on the correct edge with respect to source clock.

Clock Divide Output — CLKDV

The clock divide output pin CLKDV provides a lower frequency version of the source clock. The CLKDV_DIVIDE property controls CLKDV such that the source clock is divided by N where N is either 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16.

This feature provides automatic duty cycle correction such that the CLKDV output pin has a 50/50 duty cycle always in low-frequency mode, as well as for all all integer values of the division factor N in high-frequency mode.

Frequency Synthesized Clock Output - CLKFX

The CLKFX output provides a frequency-synthesized clock (M/D * CLKIN) with a 50/50 duty cycle. For the CLKFX output to be phase-aligned with CLKIN, the clock feedback (CLK0) must be provided at the CLKFB input. With M and D adjusted such that they have no common factor, the alignment occurs only once every D input clock cycles.

Frequency Synthesized Clock Output 180º Phase Shifted - CLKFX180

The CLKFX180 output is a 180º phase shifted version of the CLKFX clock output, also with a 50/50 duty cycle.

Locked Output — LOCKED

The LOCKED signal is activated after the DCM has achieved the parameter values set by the user parameters. To guarantee that the system clock is established prior to the device "waking up," the DCM can delay the completion of the device configuration process until after the DCM locks. The STARTUP_WAIT attribute activates this feature. Until the LOCKED signal activates, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious signals.

Reset Input — RST

When the reset pin activates, the LOCKED signal deactivates within four source clock cycles. The M and D values at configuration are maintained after the reset. The RST pin, active High, must either connect to a dynamic signal or be tied to ground. Activation of the RST pin can also severely affect the duty cycle of the clock output pins. For this reason, activate the reset pin only when reconfiguring the device or changing the input frequency. The reset input signal is asynchronous and should be held High for at least 2 ns.

Status - STATUS

The STATUS output is an 8-bit output:

- STATUS[1] indicates the loss of the input clock, CLKIN, only when CLKFB is connected.
- STATUS[2] indicates loss of CLKFX and CLKFX180 even though LOCKED might still be High. Note that this "CLKFX stopped" status functions only when CLKIN is present.

Attributes

The following attributes provide access to some of the Virtex-II series frequency synthesis features, (for example, clock multiplication, clock division).

Clock Divide

The CLKDV_DIVIDE attribute specifies how the signal on the CLKDV pin is frequency divided with respect to the CLK0 pin. The values allowed for this attribute are 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16; the default value is 2.

Frequency Mode for Frequency Synthesis

This attribute specifies either the high or low-frequency mode of the frequency synthesizer. The default is low-frequency mode. The frequency ranges for both frequency modes are specified in the **Virtex-II Data Sheet**.

To set the frequency synthesizer to high-frequency mode, attach the DFS_FREQUENCY_MODE=HIGH attribute in the source code or schematic.

Multiply/Divide Attribute

The M and D values can be set using the CLKFX_MULTIPLY and the CLKFX_DIVIDE attributes. The default settings are M = 4 and D = 1.

Startup Delay

The default value of the STARTUP_WAIT attribute is FALSE. When STARTUP_WAIT is set to TRUE, and the LCK_cycle BitGen option is used, then the configuration startup sequence waits in the specified cycle until the DCM locks. For details, see Chapter 3: Configuration and Appendix B: BitGen and PROMGen Switches and Options.

Submodules

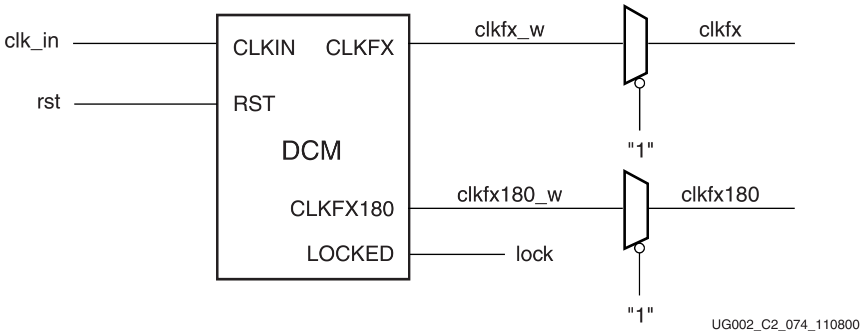


Figure 2-29: **BUFG_DFS_SUBM**

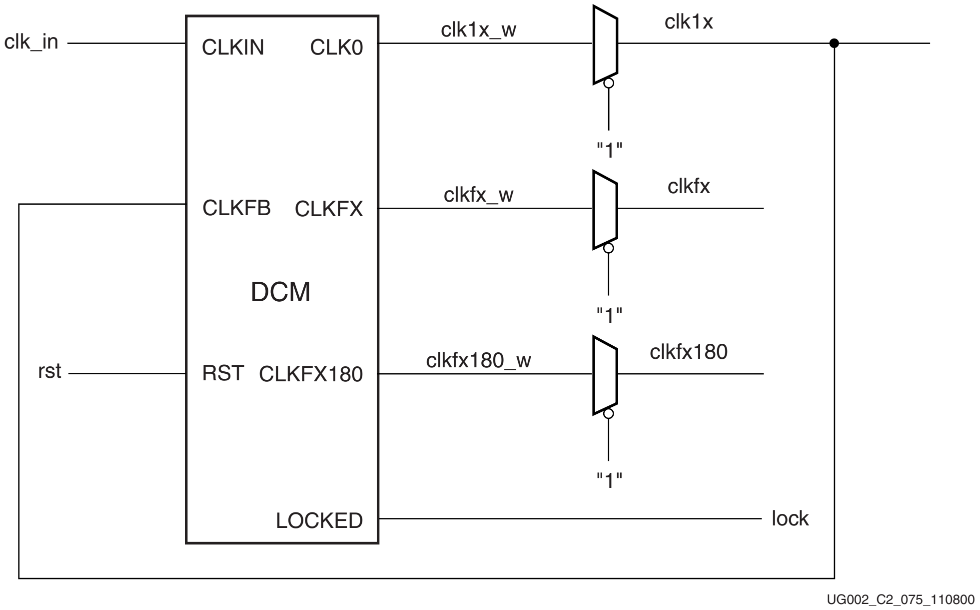


Figure 2-30: **BUFG_DFS_FB_SUBM**

Phase Shifting

The DCM can also provide coarse and fine-grained phase shifting. The CLK0, CLK90, CLK180, and CLK270 outputs are each phase shifted by ¼ of the input clock period relative to each other, providing coarse phase control. Note that CLK90 and CLK270 are not available in high-frequency mode.

Operation

Figure 2-31 shows a block diagram of the DCM and all of the outputs affected by the circuitry of the phase shift feature.

Figure 2-32

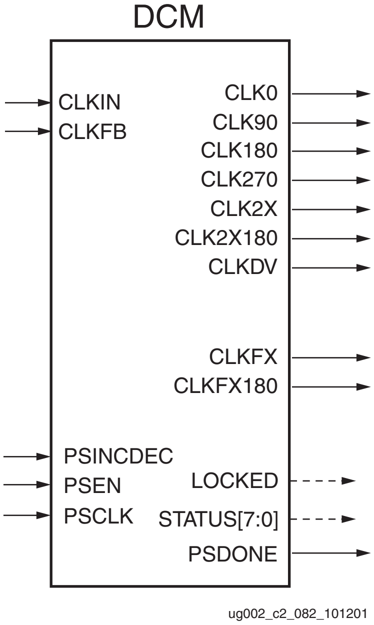


Figure 2-31: **Phase Shift Outputs**

Fine-phase adjustment affects all nine DCM output clocks. When activated, the phase shift between the rising edges of CLKIN and CLKFB is a specified fraction of the input clock period.

In variable mode, the PHASE_SHIFT value can also be dynamically incremented or decremented as determined by PSINCDEC synchronously to PSCLK, when the PSEN input is active. Figure 2-32 illustrates the effects of fine-phase shifting.

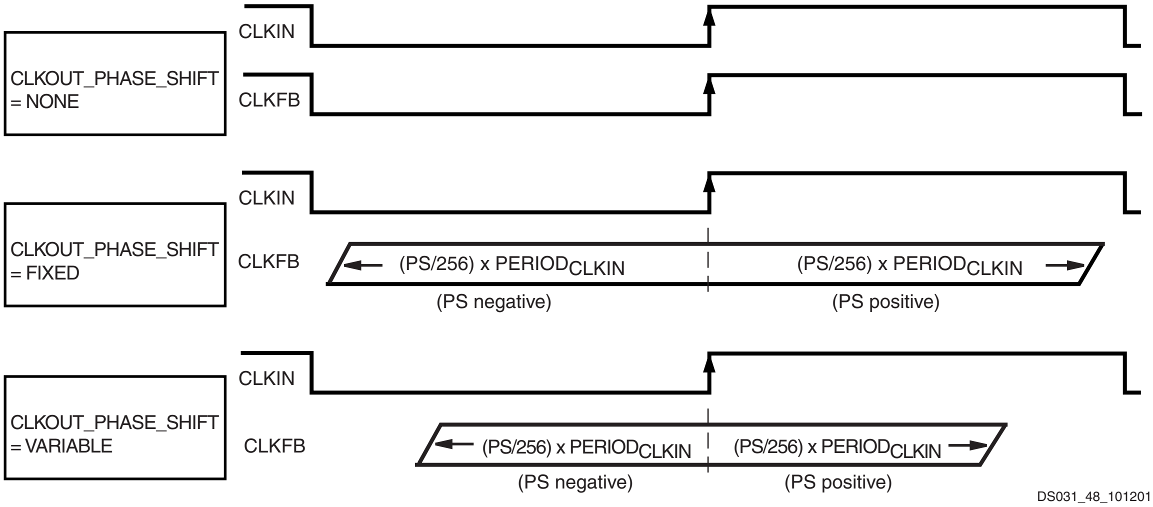


Figure 2-32: **Phase Shift Effects**

Two separate components of the phase shift range must be understood:

- PHASE_SHIFT attribute range
- FINE_SHIFT_RANGE DCM timing parameter range

The PHASE_SHIFT attribute is the numerator in the following equation:

$$\text{Phase Shift (ns)} = (\text{PHASE\_SHIFT}/256) * \text{PERIOD}_{\text{CLKIN}}$$

The full range of this attribute is always -255 to +255, but its practical range varies with CLKIN frequency, as constrained by the FINE_SHIFT_RANGE component, which represents the total delay achievable by the phase shift delay line. Total delay is a function of the number of delay taps used in the circuit. Across process, voltage, and temperature, this absolute range is guaranteed to be as specified in the DCM Timing Parameters section of the **Virtex-II Data Sheet**.

Absolute range (fixed mode) = ± FINE_SHIFT_RANGE

Absolute range (variable mode) = ± FINE_SHIFT_RANGE/2

The reason for the difference between fixed and variable modes is as follows. For variable mode to allow symmetric, dynamic sweeps from -255/256 to +255/256, the DCM sets the "zero phase skew" point as the middle of the delay line, thus dividing the total delay line range in half. In fixed mode, since the PHASE_SHIFT value never changes after configuration, the entire delay line is available for insertion into either the CLKIN or CLKFB path (to create either positive or negative skew).

Taking both of these components into consideration, the following are some usage examples:

- If $\text{PERIOD}_{\text{CLKIN}}$ = two times FINE_SHIFT_RANGE, then PHASE_SHIFT in fixed mode is limited to ± 128, and in variable mode it is limited to ± 64.
- If $\text{PERIOD}_{\text{CLKIN}}$ = FINE_SHIFT_RANGE, then PHASE_SHIFT in fixed mode is limited to ± 255, and in variable mode it is limited to ± 128.
- If $\text{PERIOD}_{\text{CLKIN}} \leq$ half of the FINE_SHIFT_RANGE, then PHASE_SHIFT is limited to ± 255 in either mode.

In variable mode, the phase factor can be changed by activating PSEN for one period of PSCLK. Increments or decrements to the phase factor can be made by setting the PSINCDEC pin to a High or Low, respectively. When the de-skew circuit has completed an increment or decrement operation, the signal PSDONE goes High for a single PSCLK cycle. This indicates to the user that the next change may be made.

The user interface and the physical implementation are different. The user interface describes the phase shift as a fraction of the clock period (N/256). The physical implementation adds the appropriate number of buffer stages (each DCM_TAP) to the clock delay. The DCM_TAP granularity limits the phase resolution at higher clock frequencies.

Phase Shift Characteristics

- Offers fine-phase adjustment with a resolution of ±1/256 of the clock period (or ± one DCM_TAP, whichever is greater) by configuration and also dynamically under user control.
- The phase shift settings affect all nine DCM outputs.
- V_{CC} and temperature do not affect the phase shift.

Port Signals

1x Clock Outputs — CLK[0|90|180|270]

The 1x clock output pin CLK0 represents a delay-compensated version of the source clock (CLKIN) signal. In low-frequency mode, the DCM provides three phase-shifted versions of the CLK0 signal (CLK90, CLK180, and CLK270), whereas in high-frequency mode, only the 180 phase-shifted version is provided. All four (including CLK0) of the phase shifted outputs can be used simultaneously in low-frequency mode. The relationship between phase shift and the corresponding period shift appears in Table 2-8. The timing diagrams in Figure 2-33 illustrate the DLL clock output characteristics.

Table 2-8: **Relationship of Phase-Shifted Output Clock to Period Shift**

Phase (degrees)	% Period Shift
0	0%
90	25%
180	50%
270	75%

By default, the DCM provides a 50/50 duty cycle correction on all 1x clock outputs. The DUTY_CYCLE_CORRECTION attribute (TRUE by default), controls this feature. Attach the DUTY_CYCLE_CORRECTION=FALSE property to the DCM symbol in order to deactivate the DCM duty cycle correction. With duty cycle correction deactivated, the output clocks have the same duty cycle as the source clock.

The DCM clock outputs can drive an OBUF, a BUFGMUX, or they can route directly to the clock input of a synchronous element.

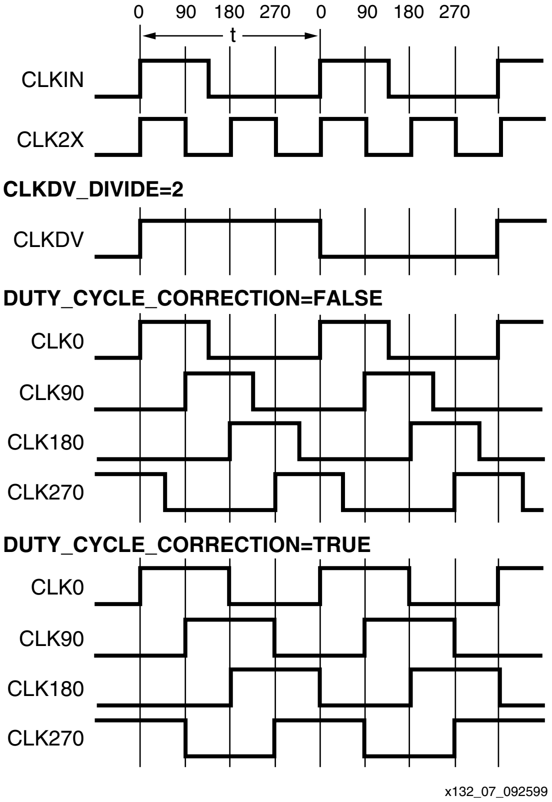


Figure 2-33: **DLL Output Characteristics**

Source Clock Input — CLKIN

The CLKIN pin provides the user source clock to the DCM. The CLKIN frequency must fall in the ranges specified in the **Virtex-II Data Sheet**. The clock input signal can be provided by one of the following:

- IBUF — Input buffer
- IBUFG — Global clock input buffer
- BUFGMUX — Internal global clock buffer

Feedback Clock Input — CLKFB

A DCM requires a reference or feedback signal to provide delay-compensated output. Connect only the CLK0 or CLK2X DCM outputs to the feedback clock input (CLKFB) pin to provide the necessary feedback to the DCM. The feedback clock input signal can be driven by an internal global clock buffer (BUFGMUX), one of the global clock input buffers (IBUFG) on the same edge of the device (top or bottom), or IBUF (the input buffer.)

If an IBUFG sources the CLKFB pin, the following special rules apply:

1. An external input port must source the signal that drives the IBUFG input pin.
2. That signal must directly drive only OBUFs and nothing else.

Phase Shift Clock - PSCLK

The PSCLK input can be sourced by the CLKIN signal to the DCM, or it can be a lower or higher frequency signal provided from any clock source (external or internal). The frequency range of PSCLK is defined by PSCLK_FREQ_LF/HF (see the **Virtex-II Data Sheet**). This input has to be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Phase Shift Increment/Decrement - PSINCDEC

The PSINCDEC signal is synchronous to PSCLK and is used to increment or decrement the phase shift factor. In order to increment or decrement the phase shift by 1/256 of clock period, the PSINCDEC signal must be High for increment or Low for decrement. This input has to be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Phase Shift Enable - PSEN

To initiate a variable phase-shift operation, the PSEN input must be activated for one period of PSCLK. The phase change becomes effective after up to 100 CLKIN pulse cycles plus three PSCLK cycles, and is indicated by a High pulse on PSDONE. During the phase transition there are no sporadic changes or glitches on any output. PSEN must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Reset Input — RST

When the reset pin is activated, the LOCKED signal deactivates within four source clock cycles. After reset, the phase shift value is set to its value at configuration in both the fixed and variable modes. The RST pin, active High, must either connect to a dynamic signal or be tied to ground. Activation of the RST pin can also severely affect the duty cycle of the clock output pins. For this reason, activate the reset pin only when reconfiguring the device or changing the input frequency. The reset input signal is asynchronous and should be held High for at least 2 ns.

Locked Output — LOCKED

The LOCKED signal activates after the DCM has achieved lock. To guarantee that the system clock is established prior to the device "waking up," the DCM can delay the completion of the device configuration process until after the DCM locks. The STARTUP_WAIT attribute activates this feature. Until the LOCKED signal activates, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. For details, refer to Chapter 3: Configuration.

Phase Shift DONE - PSDONE

The PSDONE signal is synchronous to PSCLK and it indicates, by pulsing High for one period of PSCLK, that the requested phase shift was achieved. This signal also indicates to the user that a new change to the phase shift numerator can be made. This output signal is not valid if the phase shift feature is not being used or is in FIXED mode.

Status - STATUS

STATUS[0] indicates the overflow of the phase shift numerator and that the absolute delay range of the phase shift delay line is exceeded.

Attributes

The following attributes provide access to the Virtex-II fine-phase adjustment capability.

Clock Out Phase Shift

The CLKOUT_PHASE_SHIFT attribute controls the use of the PHASE_SHIFT value. It can be set to NONE, FIXED, or VARIABLE. By default, this attribute is set to NONE, indicating that the phase shift feature is not being used. When this attribute is set to NONE, the PHASE_SHIFT value has no effect on the DCM outputs. If the CLKOUT_PHASE_SHIFT attribute is set to FIXED or NONE, then the PSEN, PSINCDEC, and the PSCLK inputs must be tied to ground. The effects of the CLKOUT_PHASE_SHIFT attribute are shown in Figure 2-32.

PHASE_SHIFT

This attribute specifies the phase shift numerator as any value from -255 to 255.

Submodules

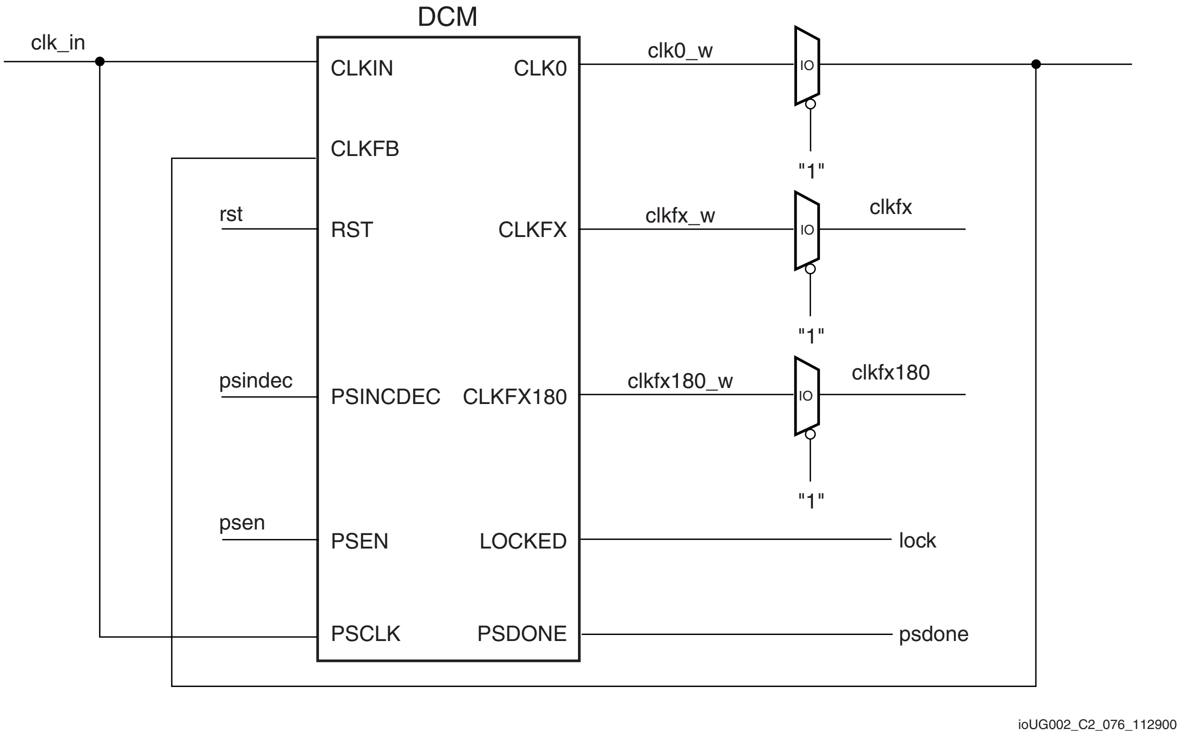


Figure 2-34: **BUFG_PHASE_CLKFX_FB_SUBM**

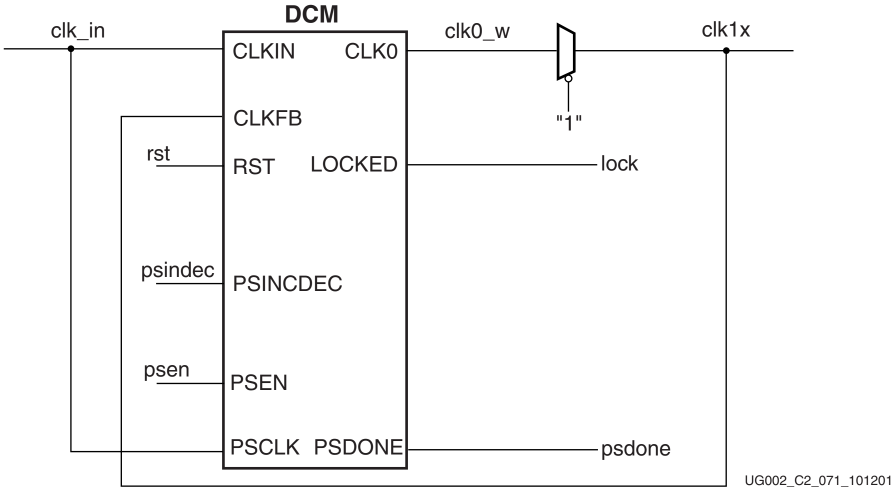


Figure 2-35: **BUFG_PHASE_CLK0_SUBM**

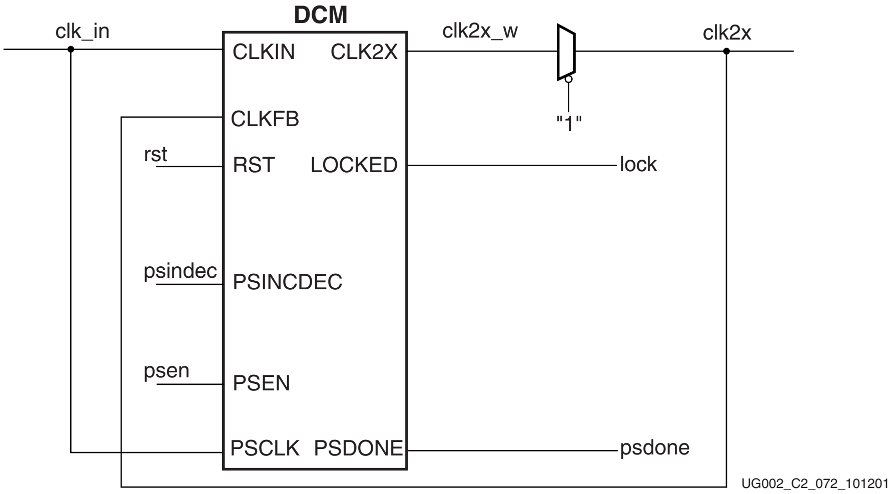


Figure 2-36: **BUFG_PHASE_CLK2X_SUBM**

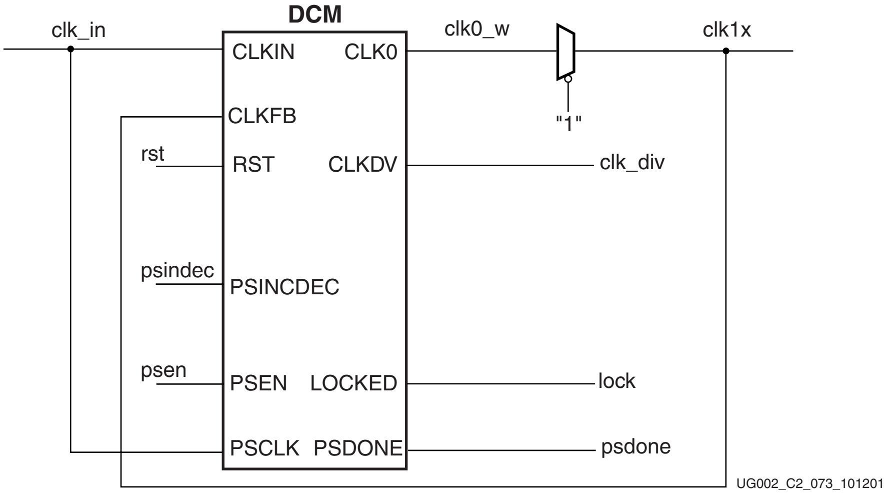


Figure 2-37: **BUFG_PHASE_CLKDV_SUBM**

VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available as examples (see "VHDL and Verilog Templates" on page 191) for all submodules.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

VHDL and Verilog Templates

The following submodules described in this section are available:

- BUFG_CLK0_SUBM
- BUFG_CLK2X_SUBM
- BUFG_CLK0_FB_SUBM
- BUFG_CLK2X_FB_SUBM
- BUFG_CLKDV_SUBM
- BUFG_DFS_SUBM
- BUFG_DFS_FB_SUBM
- BUFG_PHASE_CLKFX_FB_SUBM
- BUFG_PHASE_CLK0_SUBM
- BUFG_PHASE_CLK2X_SUBM
- BUFG_PHASE_CLKDV_SUBM

The corresponding submodules must be synthesized with the design. The BUFG_CLK0_SUBM submodule is provided in VHDL and Verilog as an example.

VHDL Template

```
-- Module: BUFG_CLK0_SUBM
-- Description: VHDL submodule
-- DCM with CLK0 deskew
-- Device: Virtex-II Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--
-- pragma translate_off
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;
-- pragma translate_on
--
entity BUFG_CLK0_SUBM is
  port (
        CLK_IN : in std_logic;
        RST    : in std_logic;
        CLK1X  : out std_logic;
        LOCK   : out std_logic
       );
end BUFG_CLK0_SUBM;
--
architecture BUFG_CLK0_SUBM_arch of BUFG_CLK0_SUBM is
-- Components Declarations:
component BUFG
  port (
         I : in std_logic;
         O : out std_logic
       );
end component;
component DCM
```

```
-- pragma translate_off
    generic (
             DLL_FREQUENCY_MODE : string := "LOW";
             DUTY_CYCLE_CORRECTION : boolean := TRUE;
             STARTUP_WAIT : boolean := FALSE
            );
-- pragma translate_on
    port ( CLKIN     : in  std_logic;
           CLKFB     : in  std_logic;
           DSSEN     : in  std_logic;
           PSINCDEC  : in  std_logic;
           PSEN      : in  std_logic;
           PSCLK     : in  std_logic;
           RST       : in  std_logic;
           CLK0      : out std_logic;
           CLK90     : out std_logic;
           CLK180    : out std_logic;
           CLK270    : out std_logic;
           CLK2X     : out std_logic;
           CLK2X180  : out std_logic;
           CLKDV     : out std_logic;
           CLKFX     : out std_logic;
           CLKFX180  : out std_logic;
           LOCKED    : out std_logic;
           PSDONE    : out std_logic;
           STATUS    : out std_logic_vector(7 downto 0)
          );
end component;
-- Attributes
attribute DLL_FREQUENCY_MODE : string;
attribute DUTY_CYCLE_CORRECTION : string;
attribute STARTUP_WAIT : string;
attribute DLL_FREQUENCY_MODE of U_DCM: label is "LOW";
attribute DUTY_CYCLE_CORRECTION of U_DCM: label is "TRUE";
attribute STARTUP_WAIT of U_DCM: label is "FALSE";
-- Signal Declarations:
signal GND : std_logic;
signal CLK0_W: std_logic;
signal CLK1X_W: std_logic;
begin
GND <= '0';
CLK1X <= CLK1X_W;
-- DCM Instantiation
U_DCM: DCM
  port map (
            CLKIN =>    CLK_IN,
            CLKFB =>    CLK1X_W,
            DSSEN =>    GND,
            PSINCDEC => GND,
            PSEN =>     GND,
            PSCLK =>    GND,
            RST =>      RST,
            CLK0 =>     CLK0_W,
            LOCKED =>   LOCK
     );
-- BUFG Instantiation
U_BUFG: BUFG
  port map  (
       I => CLK0_W,
       O => CLK1X_W
            );
end BUFG_CLK0_SUBM_arch;
```

Verilog Template

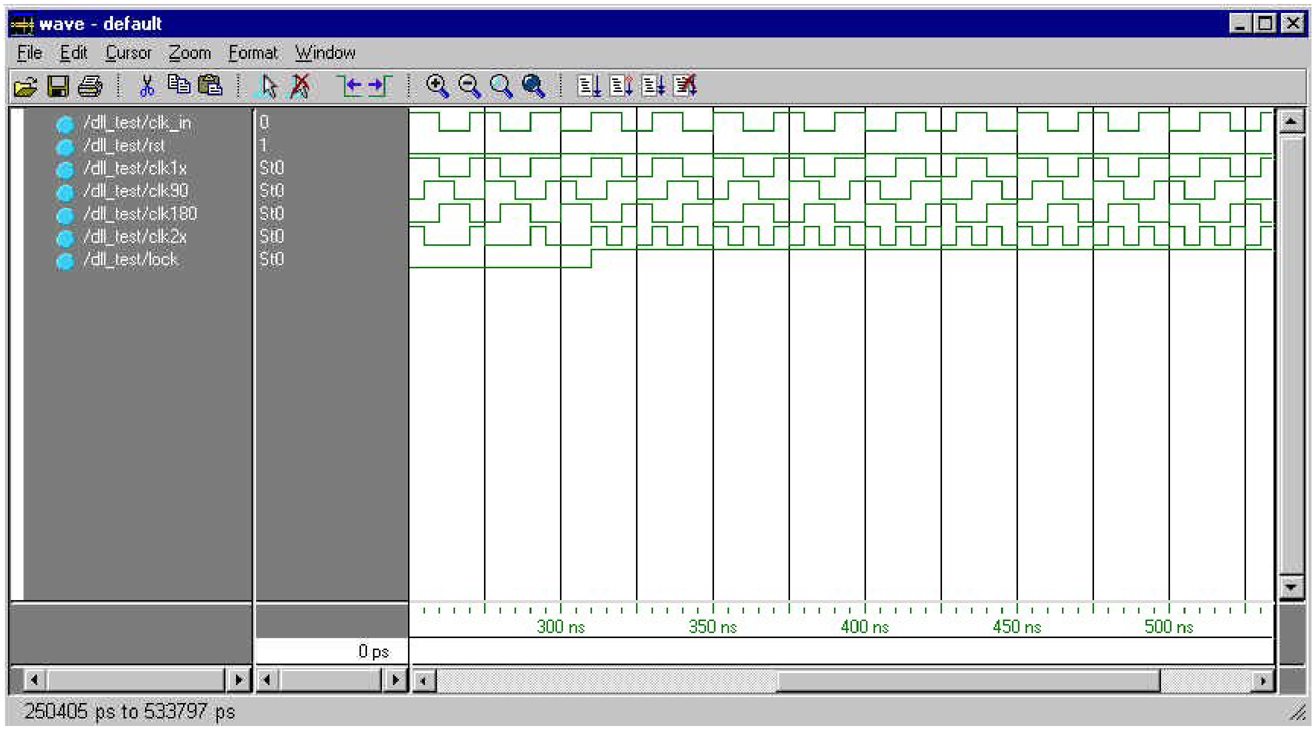
```
// Module:      BUFG_CLK0_SUBM
// Description: Verilog Submodule
//  DCM with CLK0 deskew
//
// Device: Virtex-II Family
//---------------------------------------------------------------------

module BUFG_CKL0_SUBM (
                       CLK_IN,
                       RST,
                       CLK1X,
                       LOCK
                      );

    input CLK_IN;
    input RST;

    output CLK1X;
    output LOCK;

    wire CLK0_W;
    wire GND;

    assign GND = 1'b0;

//BUFG Instantiation
//
BUFG  U_BUFG
            (.I(CLK0_W),
             .O(CLK1X)
            );

// Attributes for functional simulation//
// synopsys translate_off
       defparam U_DCM.DLL_FREQUENCY_MODE = "LOW";
       defparam U_DCM.DUTY_CYCLE_CORRECTION = "TRUE";
       defparam U_DCM.STARTUP_WAIT = "FALSE";
// synopsys translate_on

// Instantiate the DCM primitive//
  DCM U_DCM (
                  .CLKFB(CLK1X),
                  .CLKIN(CLK_IN),
                  .DSSEN(GND),
                  .PSCLK(GND),
      .PSEN(GND),
                  .PSINCDEC(GND),
                  .RST(RST),
                  .CLK0(CLK0_W),
      .LOCKED(LOCK)
                 );

// synthesis attribute declarations
  /* synopsys attribute

  DLL_FREQUENCY_MODE "LOW"
  DUTY_CYCLE_CORRECTION "TRUE"
  STARTUP_WAIT "FALSE"
  */
endmodule
```

DCM Waveforms

The DCM waveforms shown below are the results of functional simulation using Model Technology's ModelSim EE/Plus 5.3a_p1 simulator. Note that the time scale for these simulations were set to 1ns/1ps. It is important to set the unused inputs of the DCM to logic 0 and to set the attribute values to the correct data types. For example, the PHASE_SHIFT, CLKFX_DIVIDE, and CLKFX_MULTIPLY attributes are integers and should be set to values as shown.

```
defparam U_DCM.DFS_FREQUENCY_MODE = “LOW”;
defparam U_DCM.CLKFX_DIVIDE = 1; (this value’s range is specified under
Frequency Synthesis in the Virtex-II Data Sheet)
defparam U_DCM.CLKFX_MULTIPLY = 4; (this value’s range is specified
under Frequency Synthesis in the Virtex-II Data Sheet)
defparam U_DCM.CLKOUT_PHASE_SHIFT = “FIXED”;
defparam U_DCM.PHASE_SHIFT = 150;        (Any value from 1 to 255)
defparam U_DCM.STARTUP_WAIT = “FALSE”;
```

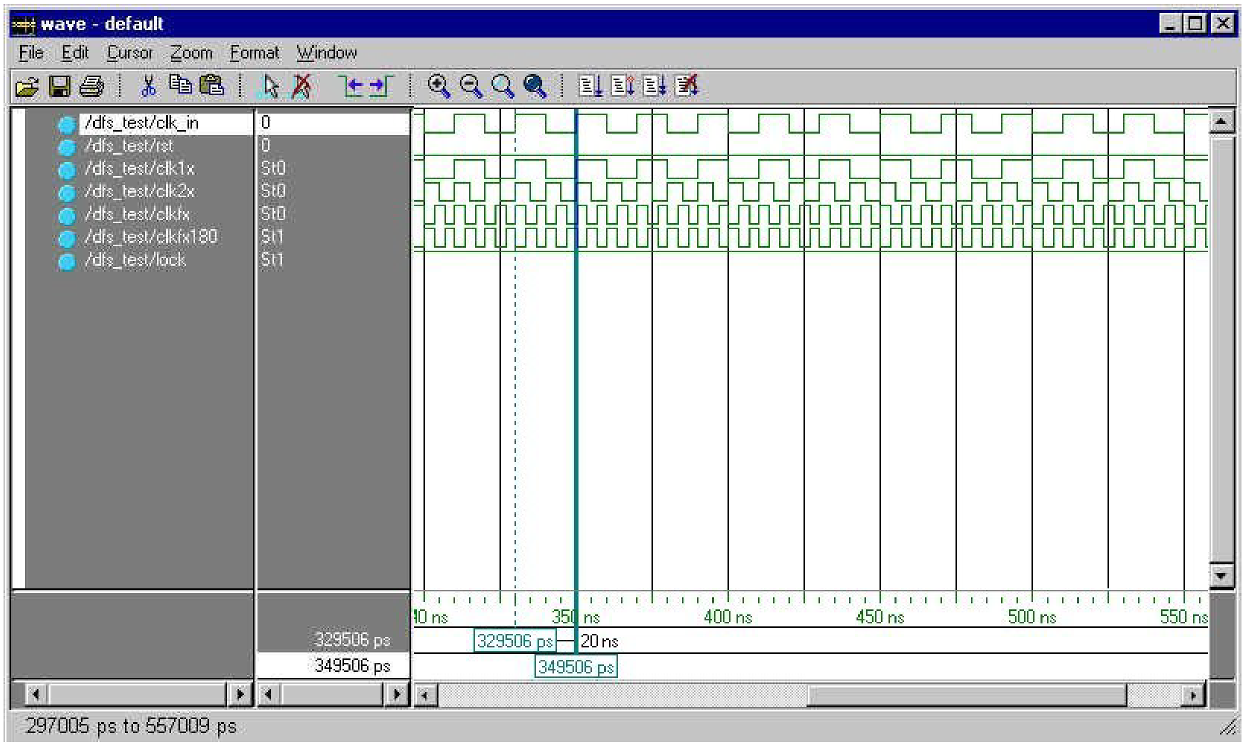
The input clock, 'clk_in' (CLKIN input of DCM) in all these waveforms is 50 MHz. The DCM_DLL waveforms in Figure 2-38 shows four DCM outputs, namely, clk1x (CLK0 output of DCM), clk2x (CLK2X output of DCM), clk90 (CLK90 output of DCM), and clk180 (CLK180 output of DCM).



ug002_c2_095_113000

Figure 2-38: **DCM_DLL Waveforms**

The DCM_DFS Waveforms in Figure 2-39 shows four DCM outputs namely, clk1x (CLK0 output of DCM), clk2x (CLK2X output of DCM), clkfx (CLKFX output of DCM), and clkfx180 (CLKFX180 output of DCM). In this case the attributes, CLKFX_DIVIDE = 1, and the CLKFX_MULTIPLY = 3.



/dfs_test/clk_in 0
/dfs_test/rst 0
/dfs_test/clk1x St0
/dfs_test/clk2x St0
/dfs_test/clkfx St0
/dfs_test/clkfx180 St1
/dfs_test/lock St1

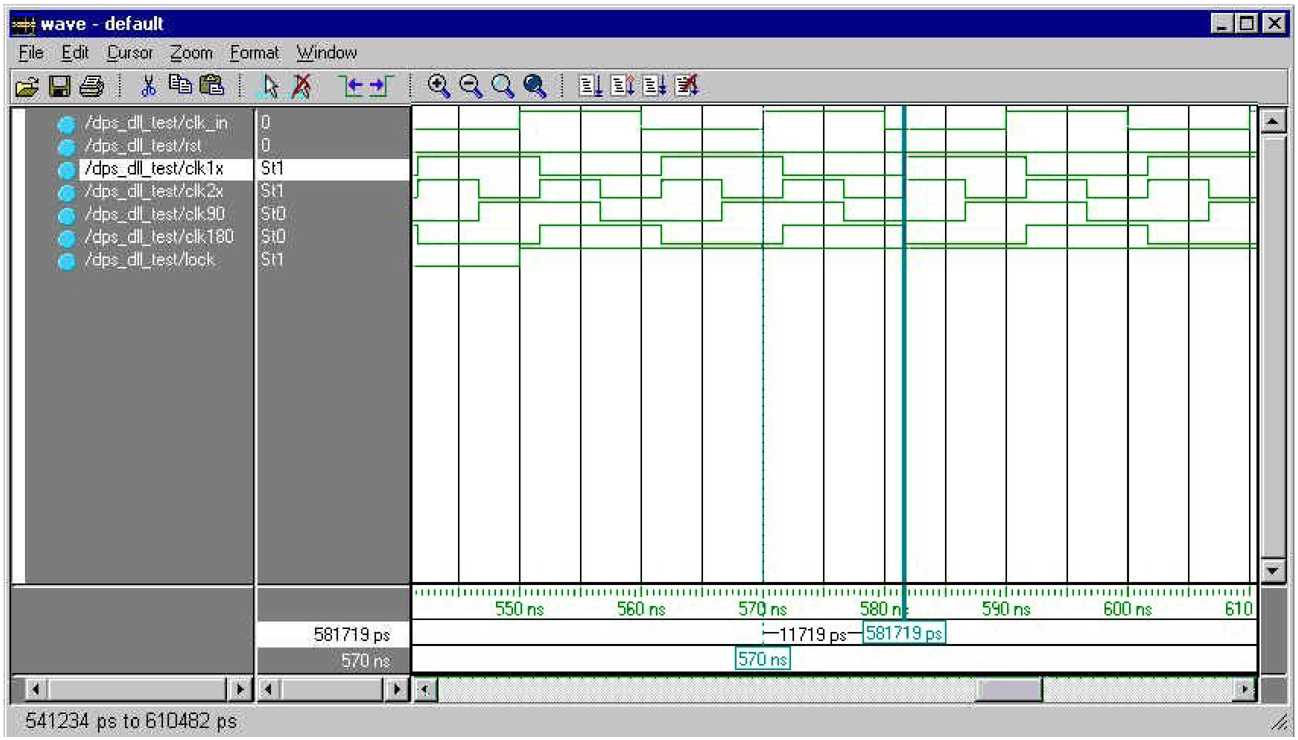
329506 ps
349506 ps

297005 ps to 557009 ps

ug002_c2_096_113000

Figure 2-39: **DCM_DFS Waveforms**

The DCM_DPS waveforms in Figure 2-40 shows four DCM outputs, namely, clk1x (CLK0 output of DCM), clk2x (CLK2X output of DCM), clk90 (CLK90 output of DCM), and clk180 (CLK180 output of DCM). In this case, the attribute PHASE_SHIFT = 150 which translates to a phase shift of (150 x 20 ns)/256 = 11.719 ns, where 20 ns is the clock period.



/dps_dll_test/clk_in 0
/dps_dll_test/rst 0
/dps_dll_test/clk1x St1
/dps_dll_test/clk2x St1
/dps_dll_test/clk90 St0
/dps_dll_test/clk180 St0
/dps_dll_test/lock St1

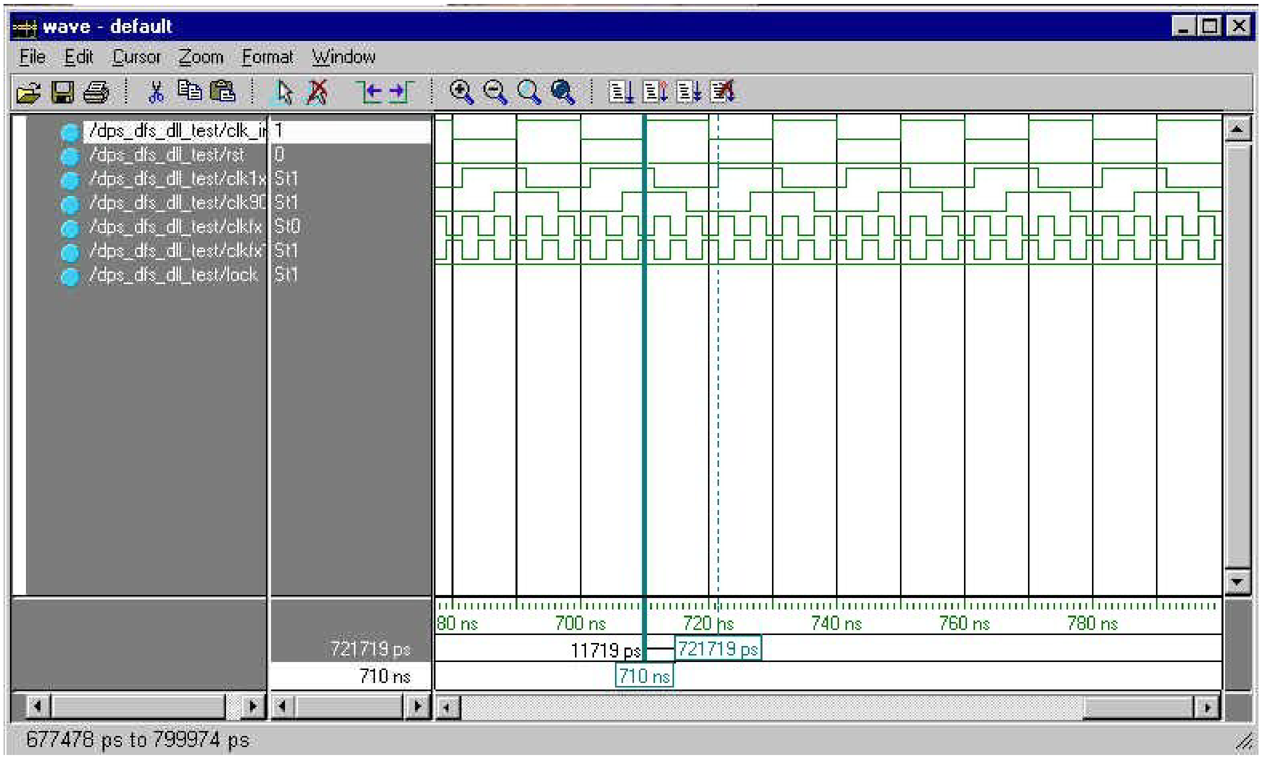
581719 ps
570 ns

541234 ps to 610482 ps

ug002_c2_097_113000

Figure 2-40: **DCM_DPS Waveforms**

The DCM_DPS_DFS waveforms in Figure 2-41 shows four DCM outputs namely, clk1x (CLK0 output of DCM), clk90 (CLK90 output of DCM), clkfx (CLKFX output of DCM), and clkfx180 (CLKFX180 output of DCM). In this case, the attributes, CLKFX_DIVIDE = 1, and the CLKFX_MULTIPLY = 4. The attribute, PHASE_SHIFT = 150 which translates to a phase shift of (150 x 20 ns)/256 = 11.719 ns, where 20 ns is the clock period.



```
/dps_dfs_dll_test/clk_ir 1
/dps_dfs_dll_test/rst     0
/dps_dfs_dll_test/clk1x   St1
/dps_dfs_dll_test/clk90   St1
/dps_dfs_dll_test/clkfx   St0
/dps_dfs_dll_test/clkfx   St1
/dps_dfs_dll_test/lock    St1
                721719 ps
                   710 ns
677478 ps to 799974 ps
```

ug002_c2_098_113000

Figure 2-41: **DCM_DPS_DFS Waveforms**

Using Block SelectRAM™ Memory

Introduction

In addition to distributed SelectRAM memory, Virtex-II devices feature a large number of 18 Kb block SelectRAM memories. The block SelectRAM memory is a True Dual-Port™ RAM, offering fast, discrete, and large blocks of memory in the device. The memory is organized in columns, and the total amount of block SelectRAM memory depends on the size of the Virtex-II device. The 18 Kb blocks are cascadable to enable a deeper and wider memory implementation, with a minimal timing penalty incurred through specialized routing resources.

Embedded dual- or single-port RAM modules, ROM modules, synchronous and asynchronous FIFOs, and data width converters are easily implemented using the Xilinx CORE Generator "Block Memory" modules. Asynchronous FIFOs can also be generated using the CORE Generator Asynchronous FIFO module. Starting with IP Update #3, the designer can also generate synchronous FIFOs using Block Memory.

Synchronous Dual-Port and Single-Port RAM

Data Flow

The 18Kb block SelectRAM dual-port memory consists of an 18-Kb storage area and two completely independent access ports, A and B. The structure is fully symmetrical, and both ports are interchangeable.

Data can be written to either port and can be read from the same or the other port. Each port is synchronous, with its own clock, clock enable, and write enable. Note that the read operation is also synchronous and requires a clock edge.

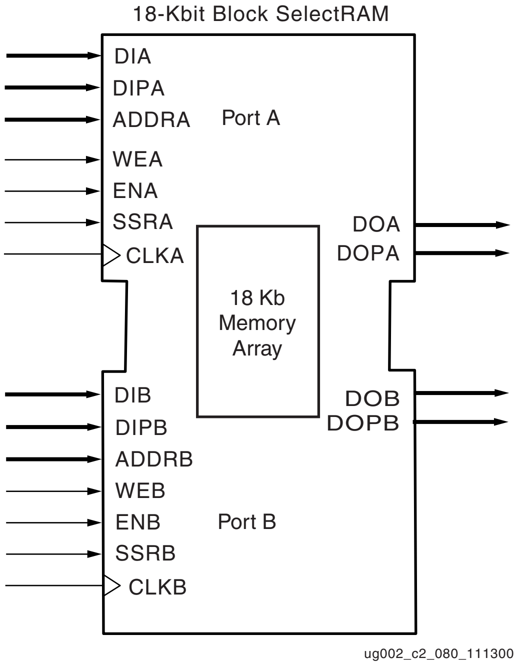


Figure 2-42: **Dual-Port Data Flows**

As described below, there are three options for the behavior of the data output during a write operation on its port. There is no dedicated monitor to arbitrate the result of identical addresses on both ports. It is up to the user to time the two clocks appropriately. However, conflicting simultaneous writes to the same location never cause any physical damage.

Operating Modes

To maximize utilization of the True Dual-Port memory at each clock edge, the block SelectRAM memory supports three different write modes for each port. The "read during write" mode offers the flexibility of using the data output bus during a write operation on the same port. Output behavior is determined by the configuration. This choice increases the efficiency of block SelectRAM memory at each clock cycle and allows designs that use maximum bandwidth.

Read Operation

The read operation uses one clock edge. The read address is registered on the read port, and the stored data is loaded into the output latches after the RAM access interval passes.

Write Operations

A write operation is a single clock-edge operation. The write address is registered on the write port, and the data input is stored in memory.

Three different modes are used to determine data available on the output latches after a write clock edge.

WRITE_FIRST or Transparent Mode (Default)

In WRITE_FIRST mode, the input data is simultaneously written into memory and stored in the data output (transparent write), as shown in Figure 2-43.

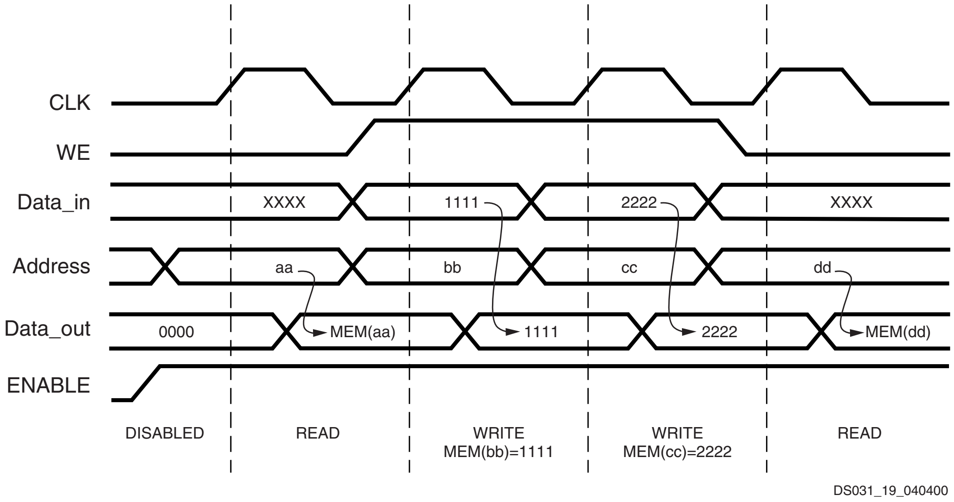


Figure 2-43: **WRITE_FIRST Mode Waveforms**

READ_FIRST or Read-Before-Write Mode

In READ_FIRST mode, data previously stored at the write address appears on the output latches, while the input data is being stored in memory (read before write). See Figure 2-44.

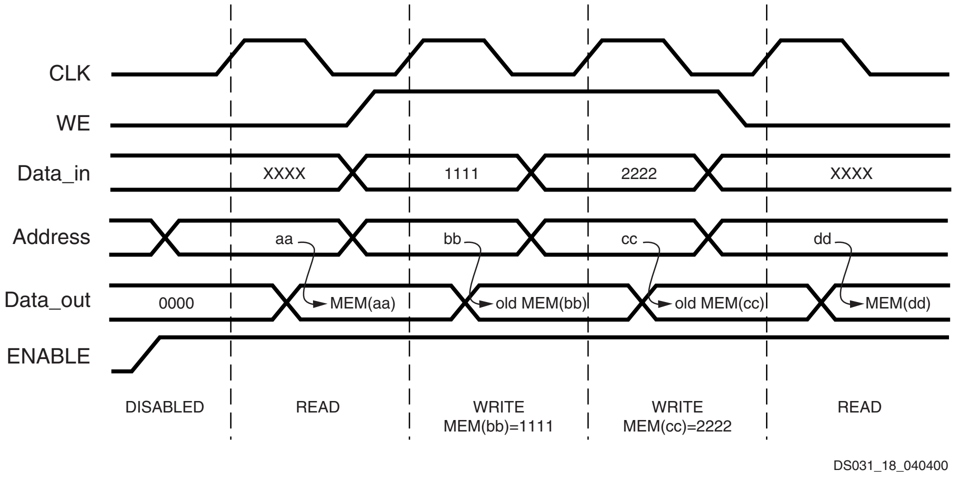


Figure 2-44: **READ_FIRST Mode Waveforms**

NO_CHANGE Mode

In NO_CHANGE mode, the output latches remain unchanged during a write operation. As shown in Figure 2-45, data output is still the last read data and is unaffected by a write operation on the same port.

Mode selection is set by configuration. One of these three modes is set individually for each port by an attribute. The default mode is WRITE_FIRST.

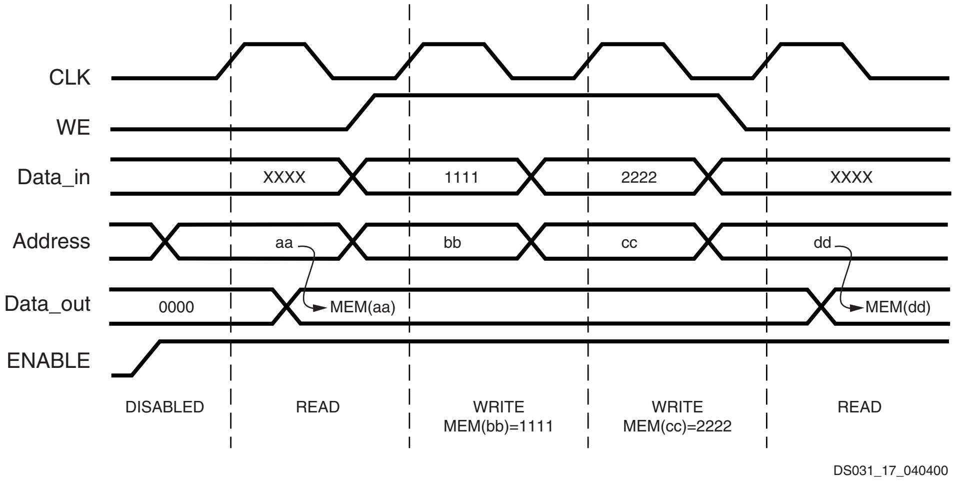


Figure 2-45: **NO_CHANGE Mode Waveforms**

Conflict Resolution

Virtex-II block SelectRAM memory is a True Dual-Port RAM that allows both ports to simultaneously access the same memory cell. When one port writes to a given memory cell, the other port must not address that memory cell (for a write or a read) within the clock-to-clock setup window. Figure 2-46 describes this asynchronous operation.

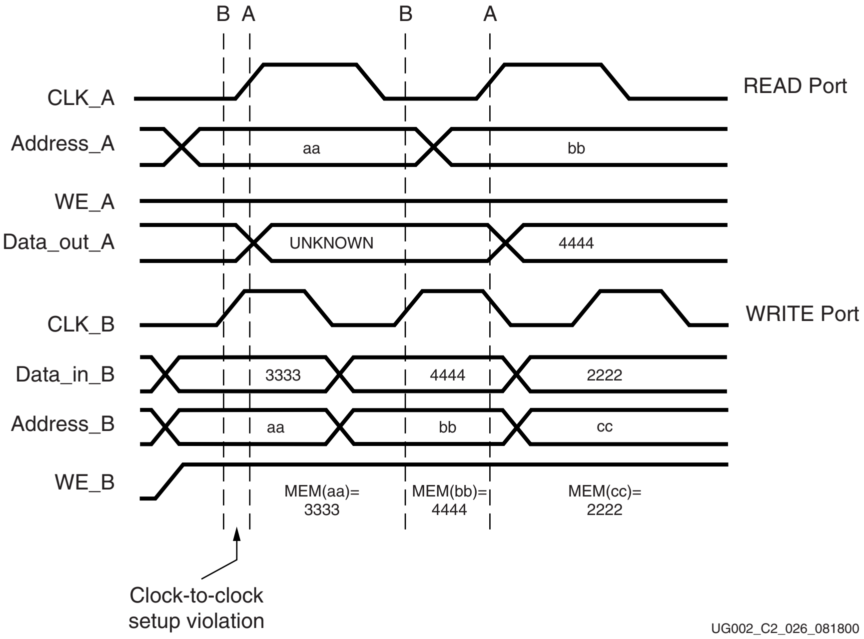


Figure 2-46: **READ-WRITE Conditions**

If port A and port B are configured with different widths, only the overlapping bits are invalid when conflicts occur.

Asynchronous Clocks

The first CLK_A clock edge violates the clock-to-clock setup parameter, because it occurs too soon after the last CLK_B clock edge. The write operation on port B is valid, and the read operation on port A is invalid.

At the second rising edge of the CLK_B pin, the write operation is valid. The memory location (bb) contains 4444. The second rising edge of CLK_A reads the new data at the same location (bb), which now contains 4444.

The clock-to-clock setup timing parameter is specified together with other block SelectRAM switching characteristics in the **Virtex-II Data Sheet**.

Synchronous Clocks

When both clocks are synchronous or identical, the result of simultaneous accesses from both ports to the same memory cell is best described in words:

- If both ports read simultaneously from the same memory cell:
  Both Data_out ports will have the same data.
- If both ports write simultaneously into the same memory cell:
  The data stored in that cell becomes invalid (unless both ports write identical data).
- If one port writes and the other one reads from the same memory cell:
  The write operation succeeds, and the write port’s Data_out behaves as determined by the read output mode (write_first, read_first, or no_change).

  If the write port is in read_first mode, the read port's Data_out represents the previous content of the memory cell. If the write port is in write_first mode or in no_change mode, the read port's Data_out becomes invalid. Obviously, the read port's mode setting does not affect this operation.

Characteristics

- A write operation requires only one clock edge.
- A read operation requires only one clock edge.
- All inputs are registered with the port clock and have a setup-to-clock timing specification.
- All outputs have a read-through function or one of three read-during-write functions, depending on the state of the WE pin. The outputs relative to the port clock are available after the clock-to-out timing interval.
- Block SelectRAM cells are true synchronous RAM memories and do not have a combinatorial path from the address to the output.
- The ports are completely independent of each other (that is, clocking, control, address, read/write functions, initialization, and data width) without arbitration.
- Output ports are latched with a self-timed circuit, guaranteeing glitch-free reads. The state of the output port does not change until the port executes another read or write operation.
- Data input and output signals are always busses; that is, in a 1-bit width configuration, the data input signal is DI[0] and the data output signal is DO[0].

Library Primitives

The input and output data busses are represented by two busses for 9-bit width (8+1), 18-bit width (16+2), and 36-bit width (32+4) configurations. The ninth bit associated with each byte can store parity or error correction bits. No specific function is performed on this bit.

The separate bus for parity bits facilitates some designs. However, other designs safely use a 9-bit, 18-bit, or 36-bit bus by merging the regular data bus with the parity bus. Read/write and storage operations are identical for all bits, including the parity bits.

Figure 2-47 shows the generic dual-port block RAM primitive. DIA, DIPA, ADDRA, DOA, DOPA, and the corresponding signals on port B are busses.

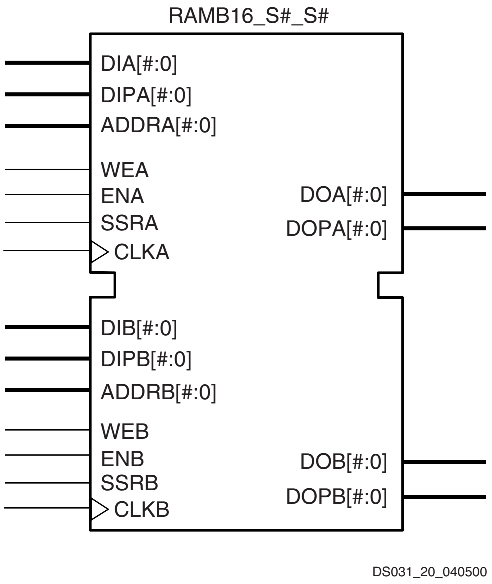


Figure 2-47: **Dual-Port Block RAM Primitive**

Table 2-9 lists the available dual-port primitives for synthesis and simulation.

Table 2-9: **Dual-Port Block RAM Primitives**

Primitive	Port A Width	Port B Width
RAMB16_S1_S1	1	1
RAMB16_S1_S2		2
RAMB16_S1_S4		4
RAMB16_S1_S9		(8+1)
RAMB16_S1_S18		(16+2)
RAMB16_S1_S36		(32+4)
RAMB16_S2_S2	2	2
RAMB16_S2_S4		4
RAMB16_S2_S9		(8+1)
RAMB16_S2_S18		(16+2)
RAMB16_S2_S36		(32+4)
RAMB16_S4_S4	4	4
RAMB16_S4_S9		(8+1)
RAMB16_S4_S18		(16+2)
RAMB16_S4_S36		(32+4)
RAMB16_S9_S9	(8+1)	(8+1)
RAMB16_S9_S18		(16+2)
RAMB16_S9_S36		(32+4)
RAMB16_S18_S18	(16+2)	(16+2)
RAMB16_S18_S36		(32+4)
RAMB16_S36_S36	(32+4)	(32+4)

Figure 2-48 shows the generic single-port block RAM primitive. DI, DIP, ADDR, DO, and DOP are busses.

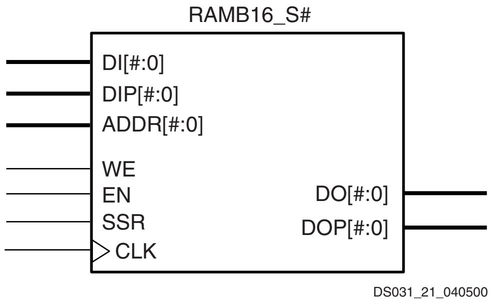


Figure 2-48: **Single-Port Block RAM Primitive**

Table 2-10 lists all of the available single-port primitives for synthesis and simulation.

Table 2-10: **Single-Port Block RAM Primitives**

Primitive	Port Width
RAMB16_S1	1
RAMB16_S2	2
RAMB16_S4	4
RAMB16_S9	(8+1)
RAMB16_S18	(16+2)
RAMB16_S36	(32+4)

VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available as examples (see "VHDL and Verilog Templates" on page 207).

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The SelectRAM_Ax templates (with x = 1, 2, 4, 9, 18, or 36) are single-port modules and instantiate the corresponding RAMB16_Sx module.

SelectRAM_Ax_By templates (with x = 1, 2, 4, 9, 18, or 36 and y = = 1, 2, 4, 9, 18, or 36) are dual-port modules and instantiate the corresponding RAMB16_Sx_Sy module.

Port Signals

Each block SelectRAM port operates independently of the other while accessing the same set of 18K-bit memory cells.

Clock - CLK[A|B]

Each port is fully synchronous with independent clock pins. All port input pins have setup time referenced to the port CLK pin. The data bus has a clock-to-out time referenced to the CLK pin. Clock polarity is configurable (rising edge by default).

Enable - EN[A|B]

The enable pin affects the read, write, and set/reset functionality of the port. Ports with an inactive enable pin keep the output pins in the previous state and do not write data to the memory cells. Enable polarity is configurable (active High by default).

Write Enable - WE[A|B]

Both EN and WE are active when the contents of the data input bus is written to memory at the address pointed to by the address bus. The output latches are loaded or not loaded according to the write configuration (WRITE_FIRST, READ_FIRST, NO_CHANGE). When inactive, a read operation occurs, and the contents of the memory cells referenced by the address bus reflect on the data-out bus, regardless of the write mode attribute. Write enable polarity is configurable (active High by default).

Set/Reset - SSR[A|B]

The SSR pin forces the data output latches to contain the value "SRVAL" (see "Attributes" on page 205). The data output latches are synchronously asserted to 0 or 1, including the parity bit. In a 36-bit width configuration, each port has an independent SRVAL[A|B] attribute of 36 bits. This operation does not affect RAM memory cells and does not disturb write operations on the other port. Like the read and write operation, the set/reset function is active only when the enable pin of the port is active. Set/reset polarity is configurable (active High by default).

Address Bus - ADDR[A|B]<#:0>

The address bus selects the memory cells for read or write. The width of the port determines the required address bus width, as shown in Table 2-11.

Table 2-11: **Port Aspect Ratio**

Port Data Width	Depth	ADDR Bus	DI Bus / DO Bus	DIP Bus / DOP Bus
1	16,384	<13:0>	<0>	NA
2	8,192	<12:0>	<1:0>	NA
4	4,096	<11:0>	<3:0>	NA
9	2,048	<10:0>	<7:0>	<0>
18	1,024	<9:0>	<15:0>	<1:0>
36	512	<8:0>	<31:0>	<3:0>

Data-In Busses - DI[A|B]<#:0> & DIP[A|B]<#:0>

Data-in busses provide the new data value to be written into RAM. The regular data-in bus (DI) and the parity data-in bus (when available) have a total width equal to the port width. For example the 36-bit port data width is represented by DI<31:0> and DIP<3:0>, as shown in Table 2-11.

Data-Out Busses - DO[A|B]<#:0> & DOP[A|B]<#:0>

Data-out busses reflect the contents of memory cells referenced by the address bus at the last active clock edge during a read operation. During a write operation (WRITE_FIRST or READ_FIRST configuration), the data-out busses reflect either the data-in busses or the stored value before write. During a write operation in NO_CHANGE mode, data-out busses are not affected. The regular data-out bus (DO) and the parity data-out bus (DOP) (when available) have a total width equal to the port width, as shown in Table 2-11.

Inverting Control Pins

For each port, the four control pins (CLK, EN, WE, and SSR) each have an individual inversion option. Any control signal can be configured as active High or Low, and the clock can be active on a rising or falling edge (active High on rising edge by default) without requiring other logic resources.

Unused Inputs

Non-connected Data and/or address inputs should be connected to logic “1”.

GSR

The global set/reset (GSR) signal of a Virtex-II device is an asynchronous global signal that is active at the end of device configuration. The GSR can also restore the initial Virtex-II state at any time. The GSR signal initializes the output latches to the INIT, or to the INIT_A and INIT_B value (see “Attributes” on page 205). A GSR signal has no impact on internal memory contents. Because it is a global signal, the GSR has no input pin at the functional level (block SelectRAM primitive).

Address Mapping

Each port accesses the same set of 18,432 memory cells using an addressing scheme dependent on the width of the port. The physical RAM locations addressed for a particular width are determined using the following formula (of interest only when the two ports use different aspect ratios):

```
END = ((ADDR + 1) * Width) -1 START= ADDR * Width
```

Table 2-12 shows low-order address mapping for each port width.

Table 2-12: **Port Address Mapping**

<table>
<tr><th>Port Width</th><th colspan="4">Parity Locations</th><th colspan="32">Data Locations</th></tr>
<tr><td>1</td><td colspan="4" rowspan="3">N.A.</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>2</td><td colspan="2">15</td><td colspan="2">14</td><td colspan="2">13</td><td colspan="2">12</td><td colspan="2">11</td><td colspan="2">10</td><td colspan="2">9</td><td colspan="2">8</td><td colspan="2">7</td><td colspan="2">6</td><td colspan="2">5</td><td colspan="2">4</td><td colspan="2">3</td><td colspan="2">2</td><td colspan="2">1</td><td colspan="2">0</td></tr>
<tr><td>4</td><td colspan="4">7</td><td colspan="4">6</td><td colspan="4">5</td><td colspan="4">4</td><td colspan="4">3</td><td colspan="4">2</td><td colspan="4">1</td><td colspan="4">0</td></tr>
<tr><td>8 + 1</td><td>3</td><td>2</td><td>1</td><td>0</td><td colspan="8">3</td><td colspan="8">2</td><td colspan="8">1</td><td colspan="8">0</td></tr>
<tr><td>16 + 2</td><td colspan="2">1</td><td colspan="2">0</td><td colspan="16">1</td><td colspan="16">0</td></tr>
<tr><td>32 + 4</td><td colspan="4">0</td><td colspan="32">0</td></tr>
</table>

Attributes

Content Initialization - INIT_xx

INIT_xx attributes define the initial memory contents. By default block SelectRAM memory is initialized with all zeros during the device configuration sequence. The 64 initialization attributes from INIT_00 through INIT_3F represent the regular memory contents. Each INIT_xx is a 64-digit hex-encoded bit vector. The memory contents can be partially initialized and are automatically completed with zeros.

The following formula is used for determining the bit positions for each INIT_xx attribute.

Given yy = conversion hex-encoded to decimal (xx), INIT_xx corresponds to the memory cells as follows:

- from [(yy + 1) * 256] -1
- to (yy) * 256

For example, for the attribute INIT_1F, the conversion is as follows:

- yy = conversion hex-encoded to decimal X"1F" = 31
- from [(31+1) * 256] -1 = 8191
- to 31 * 256 = 7936

More examples are given in Table 2-13.

Table 2-13: **Block SelectRAM Initialization Attributes**

Attribute	Memory Cell	
	from	to
INIT_00	255	0
INIT_01	511	256
INIT_02	767	512
…	…	…
INIT_0E	3839	3584
INIT_0F	4095	3840
INIT_10	4351	4096
…	…	…
INIT_1F	8191	7936
INIT_20	8447	8192
…	…	…
INIT_2F	12287	12032
INIT_30	12543	12288
..	…	…
INIT_3F	16383	16128

Content Initialization - INITP_xx

INITP_xx attributes define the initial contents of the memory cells corresponding to DIP/DOP busses (parity bits). By default these memory cells are also initialized to all zeros. The eight initialization attributes from INITP_00 through INITP_07 represent the memory contents of parity bits. Each INITP_xx is a 64-digit hex-encoded bit vector and behaves like a regular INIT_xx attribute. The same formula can be used to calculate the bit positions initialized by a particular INITP_xx attribute.

Output Latches Initialization - INIT (INIT_A & INIT_B)

The INIT (single-port) or INIT_A and INIT_B (dual-port) attributes define the output latches values after configuration. The width of the INIT (INIT_A & INIT_B) attribute is the port width, as shown in Table 2-14. These attributes are hex-encoded bit vectors and the default value is 0.

Output Latches Synchronous Set/Reset - SRVAL (SRVAL_A & SRVAL_B)

The SRVAL (single-port) or SRVAL_A and SRVAL_B (dual-port) attributes define output latch values when the SSR input is asserted. The width of the SRVAL (SRVAL_A and SRVAL_B) attribute is the port width, as shown in Table 2-14. These attributes are hex-encoded bit vectors and the default value is 0.

Table 2-14: **Port Width Values**

Port Data Width	DOP Bus	DO Bus	INIT / SRVAL
1	NA	<0>	1
2	NA	<1:0>	2
4	NA	<3:0>	4
9	<0>	<7:0>	(1+8) = 9
18	<1:0>	<15:0>	(2+16) = 18
36	<3:0>	<31:0>	(4 + 32) = 36

Initialization in VHDL or Verilog Codes

Block SelectRAM memory structures can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the attributes are attached to the block SelectRAM instantiation and are copied in the EDIF output file to be compiled by Xilinx Alliance Series™ tools. The VHDL code simulation uses a `generic` parameter to pass the attributes. The Verilog code simulation uses a `defparam` parameter to pass the attributes.

The XC2V_RAMB_1_PORT block SelectRAM instantiation code examples (in VHDL and Verilog) illustrate these techniques (see "VHDL and Verilog Templates" on page 207).

Location Constraints

Block SelectRAM instances can have LOC properties attached to them to constrain placement. Block SelectRAM placement locations differ from the convention used for naming CLB locations, allowing LOC properties to transfer easily from array to array.

The LOC properties use the following form:

```
LOC = RAMB16_X#Y#
```

The RAMB16_X0Y0 is the bottom-left block SelectRAM location on the device.

Applications

Creating Larger RAM Structures

Block SelectRAM columns have specialized routing to allow cascading blocks with minimal routing delays. Wider or deeper RAM structures are achieved with a smaller timing penalty than is encountered when using normal routing resources.

The CORE Generator program offers the designer a painless way to generate wider and deeper memory structures using multiple block SelectRAM instances. This program outputs VHDL or Verilog instantiation templates and simulation models, along with an EDIF file for inclusion in a design.

Multiple RAM Organizations

The flexibility of block SelectRAM memories allows designs with various types of RAM in addition to regular configurations. Application notes at www.xilinx.com describe some of these designs, with VHDL and Verilog reference designs included.

Virtex-II block SelectRAM can be used as follows:

- Two independent single-port RAM resources
- One 72-bit single-port RAM resource
- One triple-port (1 Read/Write and 2 Read ports) RAM resource

Application notes with VHDL and Verilog reference designs at www.xilinx.com also describe other implementations using block SelectRAM memory, such as:

- xapp258 "FIFOs Using Virtex-II Block RAM"
- xapp260 "Fast Read/Write CAM Solution"

VHDL and Verilog Templates

VHDL and Verilog templates are available for all single-port and dual-port primitives. The A and B numbers indicate the width of the ports.

The following are single-port templates:

- SelectRAM_A1
- SelectRAM_A2
- SelectRAM_A4
- SelectRAM_A9
- SelectRAM_A18
- SelectRAM_A36

The following are dual-port templates:

- SelectRAM_A1_B1
- SelectRAM_A1_B2
- SelectRAM_A1_B4
- SelectRAM_A1_B9
- SelectRAM_A1_B18
- SelectRAM_A1_B36
- SelectRAM_A2_B2
- SelectRAM_A2_B4
- SelectRAM_A2_B9
- SelectRAM_A2_B18
- SelectRAM_A2_B36
- SelectRAM_A4_B4

- SelectRAM_A4_B9
- SelectRAM_A4_B18
- SelectRAM_A4_B36
- SelectRAM_A9_B9
- SelectRAM_A9_B18
- SelectRAM_A9_B36
- SelectRAM_A18_B18
- SelectRAM_A18_B36
- SelectRAM_A36_B36

VHDL Template

As an example, the **XC2V_RAMB_1_PORT.vhd** file uses the SelectRAM_A36 template:

```
-- Module: XC2V_RAMB_1_PORT
-- Description: 18Kb Block SelectRAM example
-- Single Port 512 x 36 bits
-- Use template “SelectRAM_A36.vhd"
--
-- Device: Virtex-II Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--
-- Syntax for Synopsys FPGA Express
-- pragma translate_off
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;
-- pragma translate_on
--
entity XC2V_RAMB_1_PORT is
   port (
    DATA_IN : in std_logic_vector (35 downto 0);
        ADDRESS    : in std_logic_vector (8 downto 0);
        ENABLE: in std_logic;
        WRITE_EN     : in std_logic;
        SET_RESET    : in std_logic;
        CLK    : in std_logic;
        DATA_OUT     : out std_logic_vector (35 downto 0)
       );
end XC2V_RAMB_1_PORT;
--
architecture XC2V_RAMB_1_PORT_arch of XC2V_RAMB_1_PORT is
--
-- Components Declarations:
--
component BUFG
  port (
  I : in std_logic;
  O : out std_logic
  );
end component;
--
-- Syntax for Synopsys FPGA Express
component RAMB16_S36
-- pragma translate_off
  generic (
-- "Read during Write" attribute for functional simulation
  WRITE_MODE : string := "READ_FIRST" ; -- WRITE_FIRST(default)/
READ_FIRST/ NO_CHANGE
```

```
-- Output value after configuration
  INIT : bit_vector(35 downto 0)   := X"000000000";
-- Output value if SSR active
  SRVAL : bit_vector(35 downto 0)   := X"012345678";
-- Plus bits initial content
       INITP_00 : bit_vector(255 downto 0) :=
X"000000000000000000000000000000000000000000000000FEDCBA9876543210";
       INITP_01 : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
       INITP_02 : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
       INITP_03 : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
       INITP_04 : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
       INITP_05 : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
       INITP_06 : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
       INITP_07 : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
-- Regular bits initial content
       INIT_00 : bit_vector(255 downto 0) :=
X"000000000000000000000000000000000000000000000000FEDCBA9876543210";
       INIT_01 : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_02 : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
       ... (cut)
       INIT_3E : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_3F : bit_vector(255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000"
  );
-- pragma translate_on
  port (
        DI      : in std_logic_vector (31 downto 0);
        DIP     : in std_logic_vector (3 downto 0);
        ADDR    : in std_logic_vector (8 downto 0);
        EN      : in STD_LOGIC;
        WE      : in STD_LOGIC;
        SSR     : in STD_LOGIC;
        CLK     : in STD_LOGIC;
        DO      : out std_logic_vector (31 downto 0);
        DOP     : out std_logic_vector (3 downto 0)
  );
end component;
--
-- Attribute Declarations:
attribute WRITE_MODE : string;
attribute INIT: string;
attribute SRVAL: string;
--
attribute INITP_00: string;
attribute INITP_01: string;
attribute INITP_02: string;
attribute INITP_03: string;
attribute INITP_04: string;
attribute INITP_05: string;
attribute INITP_06: string;
attribute INITP_07: string;
--
```

```
attribute INIT_00: string;
attribute INIT_01: string;
attribute INIT_02: string;
… (cut)
attribute INIT_3E: string;
attribute INIT_3F: string;
--
-- Attribute "Read during Write mode" = WRITE_FIRST(default)/
READ_FIRST/ NO_CHANGE
attribute WRITE_MODE of U_RAMB16_S36: label is "READ_FIRST";
attribute INIT of U_RAMB16_S36: label is "000000000";
attribute SRVAL of U_RAMB16_S36: label is "012345678";
--
-- RAMB16 memory initialization for Alliance
-- Default value is "0" / Partial initialization strings are padded
-- with zeros to the left
attribute INITP_00 of U_RAMB16_S36: label is
"000000000000000000000000000000000000000000000000FEDCBA9876543210";
attribute INITP_01 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_02 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_03 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_04 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_05 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_06 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_07 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
--
attribute INIT_00 of U_RAMB16_S36: label is
"000000000000000000000000000000000000000000000000FEDCBA9876543210";
attribute INIT_01 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INIT_02 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
... (cut)
attribute INIT_3E of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INIT_3F of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
--
--
-- Signal Declarations:
--
-- signal VCC : std_logic;
-- signal GND : std_logic;
signal CLK_BUFG: std_logic;
signal INV_SET_RESET : std_logic;
--
begin
-- VCC <= '1';
-- GND <= '0';
--
-- Instantiate the clock Buffer
U_BUFG: BUFG
  port map (
  I => CLK,
  O => CLK_BUFG
```

```
  );
--
-- Use of the free inverter on SSR pin
INV_SET_RESET <= NOT SET_RESET;

-- Block SelectRAM Instantiation
U_RAMB16_S36: RAMB16_S36
  port map (
        DI      => DATA_IN (31 downto 0), -- insert 32 bits data-in bus
(<31 downto 0>)
        DIP     => DATA_IN (35 downto 32), -- insert 4 bits parity data-
in bus (or <35 downto 32>)
      ADDR   => ADDRESS (8 downto 0), -- insert 9 bits address bus
        EN      => ENABLE, -- insert enable signal
        WE      => WRITE_EN, -- insert write enable signal
        SSR     => INV_SET_RESET, -- insert set/reset signal
        CLK     => CLK_BUFG, -- insert clock signal
       DO     => DATA_OUT (31 downto 0), -- insert 32 bits data-out bus
(<31 downto 0>)
       DOP     => DATA_OUT (35 downto 32)  -- insert 4 bits parity data-
out bus (or <35 downto 32>)
  );
--
end XC2V_RAMB_1_PORT_arch;
---------------------------------------------------------------------
```

Verilog Template

```
// Module: XC2V_RAMB_1_PORT
// Description: 18Kb Block SelectRAM-II example
//  Single Port 512 x 36 bits
//  Use template "SelectRAM_A36.v"
//
// Device: Virtex-II Family
//-------------------------------------------------------------------

module XC2V_RAMB_1_PORT (CLK, SET_RESET, ENABLE, WRITE_EN, ADDRESS,
DATA_IN, DATA_OUT);


input CLK, SET_RESET, ENABLE, WRITE_EN;
input [35:0] DATA_IN;
input [8:0] ADDRESS;
output [35:0] DATA_OUT;

wire CLK_BUFG, INV_SET_RESET;

//Use of the free inverter on SSR pin
assign INV_SET_RESET = ~SET_RESET;


// initialize block ram for simulation
// synopsys translate_off
defparam
  //”Read during Write” attribute for functional simulation
  U_RAMB16_S36.WRITE_MODE = “READ_FIRST”,   //WRITE_FIRST(default)/
READ_FIRST/ NO_CHANGE
  //Output value after configuration
  U_RAMB16_S36.INIT = 36'h000000000,
  //Output value if SSR active
  U_RAMB16_S36.SRVAL = 36'h012345678,
```

```
  //Plus bits initial content
  U_RAMB16_S36.INITP_00 =
256'h0123456789ABCDEF000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_01 =
256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_02 =
256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_03 =
256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_04 =
256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_05 =
256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_06 =
256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_07 =
256'h0000000000000000000000000000000000000000000000000000000000000000,

  //Regular bits initial content
  U_RAMB16_S36.INIT_00 =
256'h0123456789ABCDEF000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INIT_01 =
256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INIT_02 =
256'h0000000000000000000000000000000000000000000000000000000000000000,
...<cut>
  U_RAMB16_S36.INIT_3E =
256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INIT_3F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
 // synopsys translate_on

//Instantiate the clock Buffer
BUFG U_BUFG (  .I(CLK), .O(CLK_BUFG));

//Block SelectRAM Instantiation
RAMB16_S36 U_RAMB16_S36 (   .DI(DATA_IN[31:0]),
         .DIP(DATA_IN-PARITY[35:32]),
         .ADDR(ADDRESS),
         .EN(ENABLE),
         .WE(WRITE_EN),
         .SSR(INV_SET_RESET),
         .CLK(CLK_BUFG),
         .DO(DATA_OUT[31:0]),
         .DOP(DATA_OUT-PARITY[35:32]));

// synthesis attribute declarations
  /* synopsys attribute

  WRITE_MODE "READ_FIRST"
  INIT "000000000"
  SRVAL "012345678"

  INITP_00
"0123456789ABCDEF000000000000000000000000000000000000000000000000"
  INITP_01
"0000000000000000000000000000000000000000000000000000000000000000"
  INITP_02
"0000000000000000000000000000000000000000000000000000000000000000"
  INITP_03
"0000000000000000000000000000000000000000000000000000000000000000"
```

```
  INITP_04
"0000000000000000000000000000000000000000000000000000000000000000"
  INITP_05
"0000000000000000000000000000000000000000000000000000000000000000"
  INITP_06
"0000000000000000000000000000000000000000000000000000000000000000"
  INITP_07
"0000000000000000000000000000000000000000000000000000000000000000"

  INIT_00
"0123456789ABCDEF000000000000000000000000000000000000000000000000"
  INIT_01
"0000000000000000000000000000000000000000000000000000000000000000"
  INIT_02
"0000000000000000000000000000000000000000000000000000000000000000"
...<cut>
  INIT_3E
"0000000000000000000000000000000000000000000000000000000000000000"
  INIT_3F
"0000000000000000000000000000000000000000000000000000000000000000"
  */

endmodule
```

Using Distributed SelectRAM Memory

Introduction

In addition to 18Kb SelectRAM blocks, Virtex-II devices feature distributed SelectRAM modules. Each function generator or LUT of a CLB resource can implement a 16 x 1-bit synchronous RAM resource. Distributed SelectRAM memory writes synchronously and reads asynchronously. However, a synchronous read can be implemented using the register that is available in the same slice. This 16 x 1-bit RAM is cascadable for a deeper and/or wider memory implementation, with a minimal timing penalty incurred through specialized logic resources.

Distributed SelectRAM modules up to a size of 128 x 1 are available as primitives. Two 16 x 1 RAM resources can be combined to form a dual-port 16 x 1 RAM with one dedicated read/write port and a second read-only port. One port writes into both 16 x1 RAMs simultaneously, but the second port reads independently.

This section provides generic VHDL and Verilog reference code examples implementing *n*-bit-wide single-port and dual-port distributed SelectRAM memory.

Distributed SelectRAM memory enables many high-speed applications that require relatively small embedded RAM blocks, such as FIFOs, which are close to the logic that uses them.

Virtex-II Distributed SelectRAM memories can be generated using the CORE Generator Distributed Memory module (V2.0 or later). The user can also generate Distributed RAM-based Asynchronous and Synchronous FIFOs using the CORE Generator.

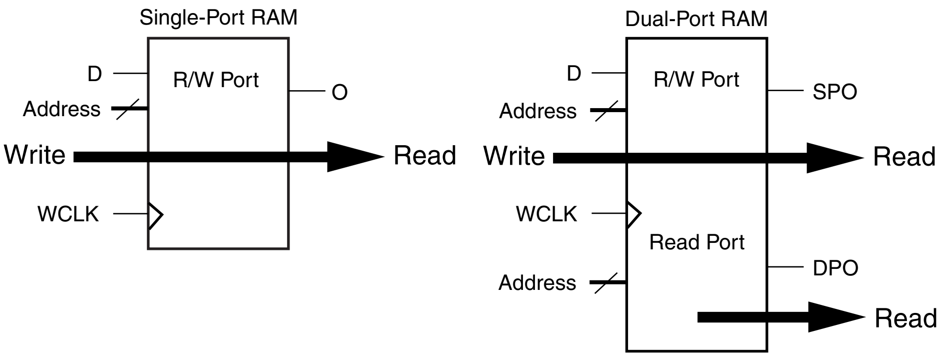
Single-Port and Dual-Port RAM

Data Flow

Distributed SelectRAM memory supports the following:

- Single-port RAM with synchronous write and asynchronous read
- Dual-port RAM with one synchronous write and two asynchronous read ports

As illustrated in the Figure 2-49, the dual port has one read/write port and an independent read port.



ug002_c2_001_061400

Figure 2-49: **Single-Port and Dual-Port Distributed SelectRAM**

Any read/write operation can occur simultaneously with and independently of a read operation on the other port.

Write Operations

The write operation is a single clock-edge operation, with a write enable that is active High by default. When the write enable is Low, no data is written into the RAM. When the write enable is High, the clock edge latches the write address and writes the data on D into the RAM.

Read Operation

The read operation is a combinatorial operation. The address port (single or dual port) is asynchronous with an access time equivalent to the logic delay.

Read During Write

When new data is synchronously written, the output reflects the data in the memory cell addressed (transparent mode). The timing diagram in Figure 2-50 illustrates a write operation, with the previous data read on the output port, before the clock edge and then the new data.

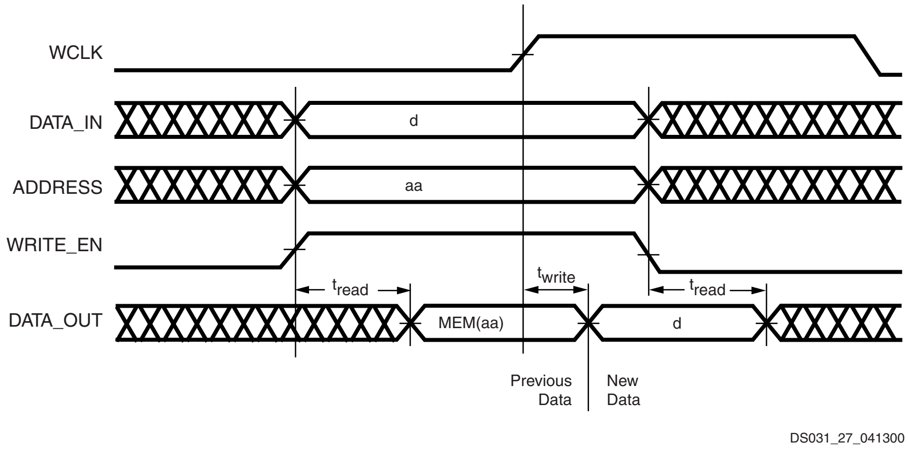


Figure 2-50: **Write Timing Diagram**

Characteristics

- A write operation requires only one clock edge.
- A read operation requires only the logic access time.
- Outputs are asynchronous and dependent only on the logic delay.
- Data and address inputs are latched with the write clock and have a setup-to-clock timing specification. There is no hold time requirement.
- For dual-port RAM, one address is the write and read address, the other address is an independent read address.

Library Primitives

Seven library primitives from 16 x 1-bit to 128 x 1-bit are available. Four primitives are single-port RAM and three primitives are True Dual-Port RAM, as shown in Table 2-15.

Table 2-15: **Single-Port and Dual-Port Distributed SelectRAM**

Primitive	RAM Size	Type	Address Inputs
RAM16X1S	16 bits	single-port	A3, A2, A1, A0
RAM32X1S	32 bits	single-port	A4, A3, A2, A1, A0
RAM64X1S	64 bits	single-port	A5, A3, A2, A1, A0
RAM128X1S	128 bits	single-port	A6, A4, A3, A2, A1, A0
RAM16X1D	16 bits	dual-port	A3, A2, A1, A0
RAM32X1D	32 bits	dual-port	A4, A3, A2, A1, A0
RAM64X1D	64 bits	dual-port	A5, A4, A3, A2, A1, A0

The input and output data are 1-bit wide. However, several distributed SelectRAM memories can be used to implement wide memory blocks.

Figure 2-51 shows generic single-port and dual-port distributed SelectRAM primitives. The A and DPRA signals are address busses.

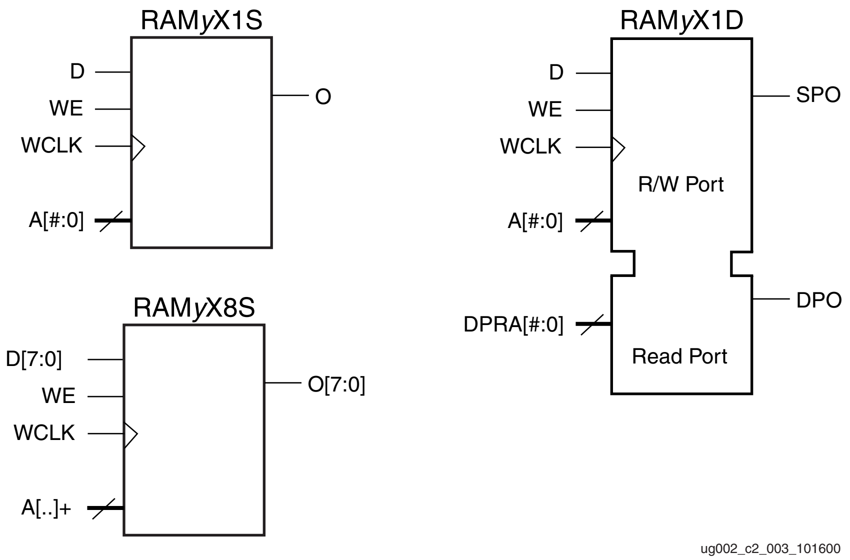


Figure 2-51: **Single-Port and Dual-Port Distributed SelectRAM Primitive**

As shown in Table 2-16, wider library primitives are available for 2-bit, 4-bit, and 8-bit RAM.

Table 2-16: **Wider Library Primitives**

Primitive	RAM Size	Data Inputs	Address Inputs	Data Outputs
RAM16x2S	16 x 2-bit	D1, D0	A3, A2, A1, A0	O1, O0
RAM32X2S	32 x 2-bit	D1, D0	A4, A3, A2, A1, A0	O1, O0
RAM64X2S	64 x 2-bit	D1, D0	A5, A4, A3, A2, A1, A0	O1, O0
RAM16X4S	16 x 4-bit	D3, D2, D1, D0	A3, A2, A1, A0	O3, O2, O1, O0
RAM32X4S	32 x 4-bit	D3, D2, D1, D0	A4,A3, A2, A1, A0	O3, O2, O1, O0
RAM16X8S	16 x 8-bit	D <7:0>	A3, A2, A1, A0	O <7:0>
RAM32X8S	32 x 8-bit	D <7:0>	A4,A3, A2, A1, A0	O <7:0>

VHDL and Verilog Instantiation

VHDL and Verilog instantiations templates are available as examples (see "VHDL and Verilog Templates" on page 221).

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The SelectRAM_*x*S templates (with *x* = 16, 32, 64, or 128) are single-port modules and instantiate the corresponding RAM*x*X1S primitive.

SelectRAM_*x*D templates (with *x* = 16, 32, or 64) are dual-port modules and instantiate the corresponding RAM*x*X1D primitive.

Ports Signals

Each distributed SelectRAM port operates independently of the other while reading the same set of memory cells.

Clock - WCLK

The clock is used for the synchronous write. The data and the address input pins have setup time referenced to the WCLK pin.

Enable - WE

The enable pin affects the write functionality of the port. An inactive Write Enable prevents any writing to memory cells. An active Write Enable causes the clock edge to write the data input signal to the memory location pointed to by the address inputs.

Address - A0, A1, A2, A3 (A4, A5, A6)

The address inputs select the memory cells for read or write. The width of the port determines the required address inputs. Note that the address inputs are not a bus in VHDL or Verilog instantiations.

Data In - D

The data input provides the new data value to be written into the RAM.

Data Out - O, SPO, and DPO

The data out O (Single-Port or SPO) and DPO (Dual-Port) reflects the contents of the memory cells referenced by the address inputs. Following an active write clock edge, the data out (O or SPO) reflects the newly written data.

Inverting Control Pins

The two control pins (WCLK and WE) each have an individual inversion option. Any control signal, including the clock, can be active at 0 (negative edge for the clock) or at 1 (positive edge for the clock) without requiring other logic resources.

GSR

The global set/reset (GSR) signal does not affect distributed SelectRAM modules.

Attributes

Content Initialization - INIT

With the INIT attributes, users can define the initial memory contents after configuration. By default distributed SelectRAM memory is initialized with all zeros during the device configuration sequence. The initialization attribute INIT represents the specified memory contents. Each INIT is a hex-encoded bit vector. Table 2-17 shows the length of the INIT attribute for each primitive.

Table 2-17: **INIT Attributes Length**

Primitive	Template	INIT Attribute Length
RAM16X1S	SelectRAM_16S	4 digits
RAM32X1S	SelectRAM_32S	8 digits
RAM64X1S	SelectRAM_64S	16 digits
RAM128X1S	SelectRAM_128S	32 digits
RAM16X1D	SelectRAM_16S	4 digits
RAM32X1D	SelectRAM_32S	8 digits
RAM64X1D	SelectRAM_64S	16 digits

Initialization in VHDL or Verilog Codes

Distributed SelectRAM memory structures can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the attributes are attached to the distributed SelectRAM instantiation and are copied in the EDIF output file to be compiled by Xilinx Alliance Series™ tools. The VHDL code simulation uses a `generic` parameter to pass the attributes. The Verilog code simulation uses a `defparam` parameter to pass the attributes.

The distributed SelectRAM instantiation templates (in VHDL and Verilog) illustrate these techniques (see "VHDL and Verilog Templates" on page 221).

Location Constraints

The CLB has four slices S0, S1, S2 and S3. As an example, in the bottom left CLB, the slices have the coordinates shown below: S

Slice S3	Slice S2	Slice S1	Slice S0
X1Y1	X1Y0	X0Y1	X0Y0

Distributed SelectRAM instances can have LOC properties attached to them to constrain placement. The RAM16X1S primitive fits in any LUT of slices S0 or S1.

For example, the instance `U_RAM16` is placed in slice X0Y0 with the following LOC properties:

```
INST "U_RAM16" LOC = "SLICE_X0Y0";
```

The RAM16X1D primitive occupies half of two slices, as shown in Figure 2-52. The first slice (output SPO) implements the read/write port with the same address A[3:0] for read and write. The second slice implements the second read port with the address DPRA[3:0] and is written simultaneously with the first slice to the address A[3:0].

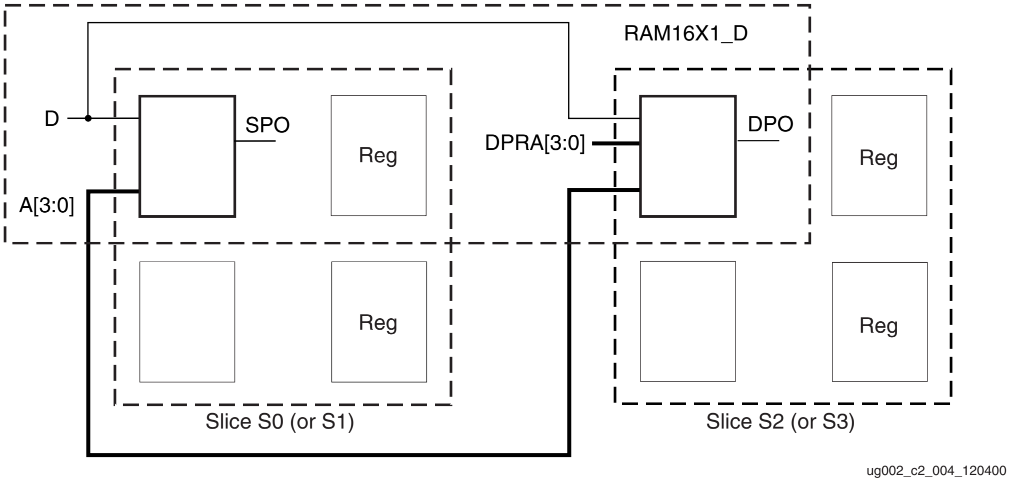


Figure 2-52: **RAM16X1_D Placement**

In the same CLB module, the dual-port RAM16X1_D either occupies half of slices S0 (X0Y0) and S2 (X1Y0), or half of slices S1 (X0Y1) and S3 (X1Y1).

If a dual-port 16 x 2-bit module is built, the two RAM16X1_D primitives occupy two slices, as long as they share the same clock and write enable, as illustrated in Figure 2-53.

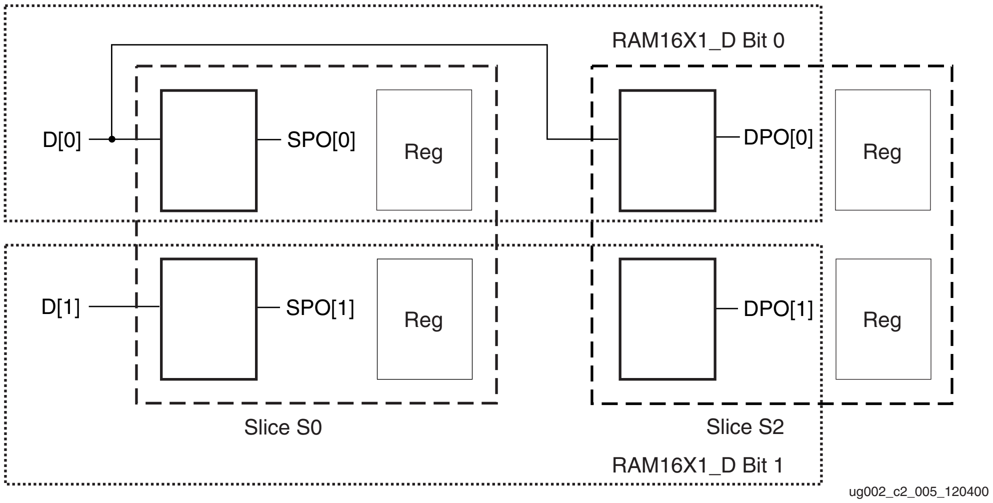


Figure 2-53: **Two RAM16X1_D Placement**

A RAM32X1S primitive fits in one slice, as shown in Figure 2-54.

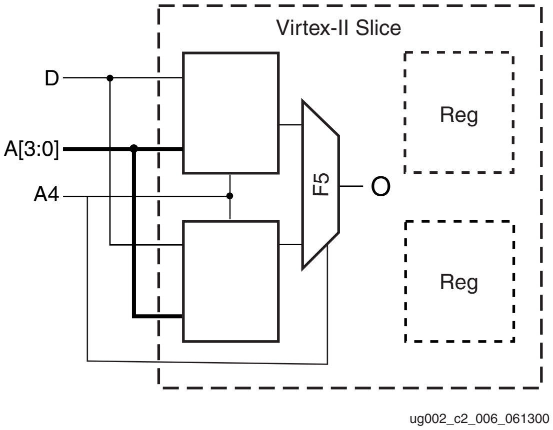


Figure 2-54: **RAM32X1_S Placement**

Following the same rules, a RAM32X1_D primitive fits in two slices, with one slice implementing the read/write port and the second slice implementing the second read port.

The RAM64X1_S primitive occupies two slices and the RAM64X1_D primitive occupies four slices (one CLB element), with two slices implementing the read/write port and two other slices implementing the second read port. The RAM64X1_S read path is built on the MUXF5 and MUXF6 multiplexers.

The RAM128X1_S primitive occupies four slices, equivalent to one CLB element.

Distributed SelectRAM placement locations use the slice location naming convention, allowing LOC properties to transfer easily from array to array.

Applications

Creating Larger RAM Structures

The memory compiler program generates wider and/or deeper memory structures using distributed SelectRAM instances. Along with an EDIF file for inclusion in a design, this program produces VHDL and Verilog instantiation templates and simulation models.

Table 2-18 shows the generic VHDL and Verilog distributed SelectRAM examples provided to implement *n*-bit-wide memories.

Table 2-18: **VHDL and Verilog Submodules**

Submodules	Primitive	Size	Type
XC2V_RAM16XN_S_SUBM	RAM16X1S	16 words x *n*-bit	single-port
XC2V_RAM32XN_S_SUBM	RAM32X1S	32 words x *n*-bit	single-port
XC2V_RAM64XN_S_SUBM	RAM64X1S	64 words x *n*-bit	single-port
XC2V_RAM128XN_S_SUBM	RAM128X1S	128 words x *n*-bit	single-port
XC2V_RAM16XN_D_SUBM	RAM16X1D	16 words x *n*-bit	dual-port
XC2V_RAM32XN_D_SUBM	RAM32X1D	32 words x *n*-bit	dual-port
XC2V_RAM64XN_D_SUBM	RAM64X1D	64 words x *n*-bit	dual-port

By using the read/write port for the write address and the second read port for the read address, a FIFO that can read and write simultaneously is easily generated. Simultaneous access doubles the effective throughput of the memory.

VHDL and Verilog Templates

VHDL and Verilog templates are available for all single-port and dual-port primitives. The number in each template indicates the number of bits (for example, SelectRAM_16S is the template for the 16 x 1-bit RAM); S indicates single-port, and D indicates dual-port.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The following are single-port templates:

- SelectRAM_16S
- SelectRAM_32S
- SelectRAM_64S
- SelectRAM_128S

The following are dual-port templates:

- SelectRAM_16D
- SelectRAM_32D
- SelectRAM_64D

Templates for the SelectRAM_16S module are provided in VHDL and Verilog code as examples.

VHDL Template

```
--
-- Module: SelectRAM_16S
--
-- Description: VHDL instantiation template
--              Distributed SelectRAM
--              Single Port 16 x 1
--              can be used also for RAM16X1S_1
--
-- Device: Virtex-II Family
--
---------------------------------------------------------------------
--
-- Components Declarations:
--
component RAM16X1S
-- pragma translate_off
   generic (
-- RAM initialization (“0” by default) for functional simulation:
        INIT : bit_vector := X"0000"
        );
-- pragma translate_on
  port (
        D    : in std_logic;
        WE   : in std_logic;
        WCLK : in std_logic;
        A0   : in std_logic;
        A1   : in std_logic;
        A2   : in std_logic;
        A3   : in std_logic;
        O    : out std_logic
        );
end component;
--
---------------------------------------------------------------------
--
-- Architecture  section:
--
-- Attributes for RAM initialization ("0" by default):
attribute INIT: string;
--
attribute INIT of U_RAM16X1S: label is "0000";
--
-- Distributed SelectRAM Instantiation
U_RAM16X1S: RAM16X1S
  port map (
    D      => , -- insert input signal
    WE     => , -- insert Write Enable signal
    WCLK   => , -- insert Write Clock signal
    A0     => , -- insert Address 0 signal
    A1     => , -- insert Address 1 signal
    A2     => , -- insert Address 2 signal
    A3     => , -- insert Address 3 signal
    O      =>   -- insert output signal
  );
--
---------------------------------------------------------------------
```

Verilog Template

```
//
// Module: SelectRAM_16S
//
// Description: Verilog instantiation template
//              Distributed SelectRAM
//              Single Port 16 x 1
//              can be used also for RAM16X1S_1
//
// Device: Virtex-II Family
//
//-------------------------------------------------------------------
//
//
// Syntax for Synopsys FPGA Express
// synopsys translate_off

  defparam

       //RAM initialization (“0” by default) for functional simulation:
  U_RAM16X1S.INIT = 16'h0000;
// synopsys translate_on


//Distributed SelectRAM Instantiation
RAM16X1S U_RAM16X1S (  .D(),      // insert input signal
                 .WE(),     // insert Write Enable signal
                .WCLK(),   // insert Write Clock signal
                 .A0(),     // insert Address 0 signal
                 .A1(),     // insert Address 1 signal
                 .A2(),     // insert Address 2 signal
                 .A3(),     // insert Address 3 signal
                 .O()       // insert output signal
                  );

// synthesis attribute declarations
  /* synopsys attribute

  INIT "0000"
  */
```

Using Look-Up Tables as Shift Registers (SRLUTs)

Introduction

Virtex-II can configure any look-up table (LUT) as a 16-bit shift register without using the flip-flops available in each slice. Shift-in operations are synchronous with the clock, and output length is dynamically selectable. A separate dedicated output allows the cascading of any number of 16-bit shift registers to create whatever size shift register is needed. Each CLB resource can be configured using the 8 LUTs as a 128-bit shift register.

This section provides generic VHDL and Verilog submodules and reference code examples for implementing from 16-bit up to 128-bit shift registers. These submodules are built from 16-bit shift-register primitives and from dedicated MUXF5, MUXF6, MUXF7, and MUXF8 multiplexers.

These shift registers enable the development of efficient designs for applications that require delay or latency compensation. Shift registers are also useful in synchronous FIFO and content-addressable memory (CAM) designs. To quickly generate a Virtex-II shift register without using flip-flops (i.e., using the SRL16 element(s)), use the CORE Generator RAM-based Shift Register module.

Shift Register Operations

Data Flow

Each shift register (SRL16 primitive) supports:

- Synchronous shift-in
- Asynchronous 1-bit output when the address is changed dynamically
- Synchronous shift-out when the address is fixed

In addition, cascadable shift registers (SRLC16) support synchronous shift-out output of the last (16th) bit. This output has a dedicated connection to the input of the next SRLC16 inside the CLB resource. Two primitives are illustrated in Figure 2-55.

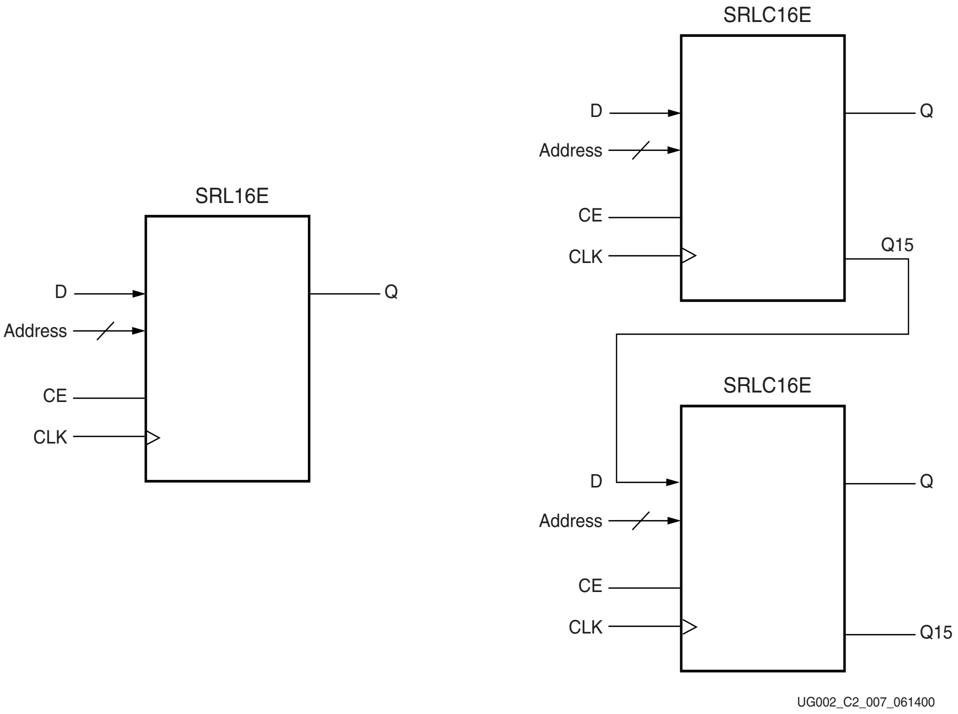


Figure 2-55: **Shift Register and Cascadable Shift Register**

Shift Operation

The shift operation is a single clock-edge operation, with an active High clock enable feature. When enable is High, the input (D) is loaded into the first bit of the shift register, and each bit is shifted to the next highest bit position. In a cascadable shift register configuration (such as SRLC16), the last bit is shifted out on the Q15 output.

The bit selected by the 4-bit address appears on the Q output.

Dynamic Read Operation

The Q output is determined by the 4-bit address. Each time a new address is applied to the 4-input address pins, the new bit position value is available on the Q output after the time delay to access the LUT. This operation is asynchronous and independent of the clock and clock enable signals.

Figure 2-56 illustrates the shift and dynamic read operations.

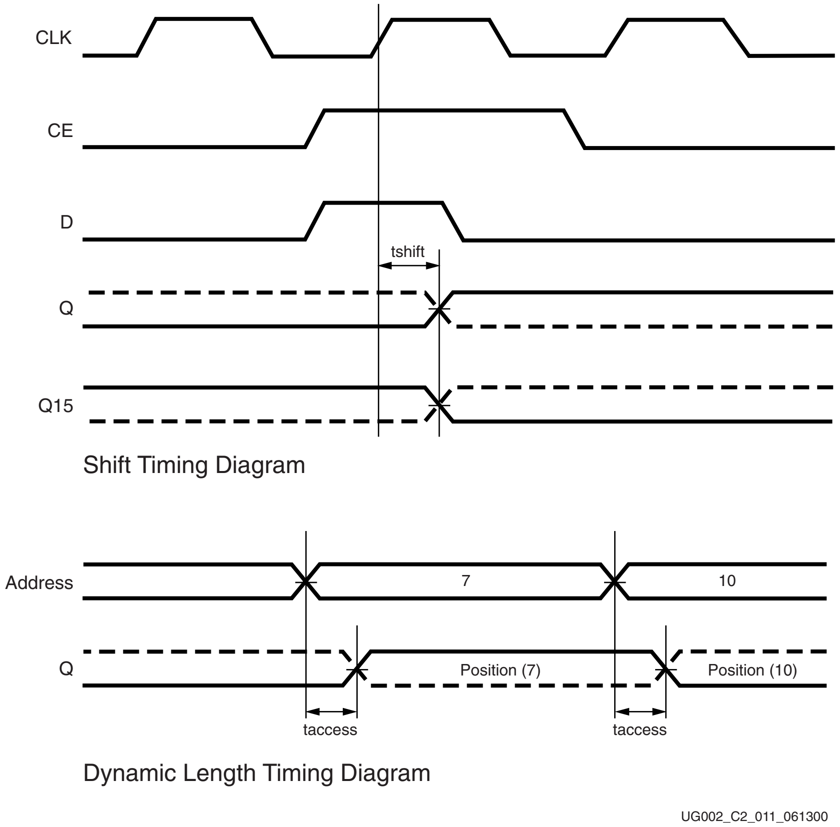


Figure 2-56: **Shift- and Dynamic-Length Timing Diagrams**

Static Read Operation

If the 4-bit address is fixed, the Q output always uses the same bit position. This mode implements any shift register length up 1 to 16 bits in one LUT. Shift register length is (N+1) where N is the input address.

The Q output changes synchronously with each shift operation. The previous bit is shifted to the next position and appears on the Q output.

Characteristics

- A shift operation requires one clock edge.
- Dynamic-length read operations are asynchronous (Q output).
- Static-length read operations are synchronous (Q output).
- The data input has a setup-to-clock timing specification.
- In a cascadable configuration, the Q15 output always contains the last bit value.
- The Q15 output changes synchronously after each shift operation.

Library Primitives and Submodules

Eight library primitives are available that offer optional clock enable (CE), inverted clock ($\overline{\text{CLK}}$) and cascadable output (Q15) combinations.

Table 2-19 lists all of the available primitives for synthesis and simulation.

Table 2-19: **Shift Register Primitives**

Primitive	Length	Control	Address Inputs	Output
SRL16	16 bits	CLK	A3,A2,A1,A0	Q
SRL16E	16 bits	CLK, CE	A3,A2,A1,A0	Q
SRL16_1	16 bits	$\overline{\text{CLK}}$	A3,A2,A1,A0	Q
SRL16E_1	16 bits	$\overline{\text{CLK}}$, CE	A3,A2,A1,A0	Q
SRLC16	16 bits	CLK	A3,A2,A1,A0	Q, Q15
SRLC16E	16 bits	CLK, CE	A3,A2,A1,A0	Q, Q15
SRLC16_1	16 bits	$\overline{\text{CLK}}$	A3,A2,A1,A0	Q, Q15
SRLC16E_1	16 bits	$\overline{\text{CLK}}$, CE	A3,A2,A1,A0	Q, Q15

In addition to the 16-bit primitives, three submodules that implement 32-bit, 64-bit, and 128-bit cascadable shift registers are provided in VHDL and Verilog code. Table 2-20 lists available submodules.

Table 2-20: **Shift Register Submodules**

Submodule	Length	Control	Address Inputs	Output
SRLC32E_SUBM	32 bits	CLK, CE	A4,A3,A2,A1,A0	Q, Q31
SRLC64E_SUBM	64 bits	CLK, CE	A5, A4, A3,A2,A1,A0	Q, Q63
SRLC128E_SUBM	128 bits	CLK, CE	A6, A5, A4, A3,A2,A1,A0	Q, Q127

The submodules are based on SRLC16E primitives, which are associated with dedicated multiplexers (MUXF5, MUXF6, and so forth). This implementation allows a fast static- and dynamic-length mode, even for very large shift registers.

Figure 2-57 represents the cascadable shift registers (32-bit and 64-bit) implemented by the submodules in Table 2-20.

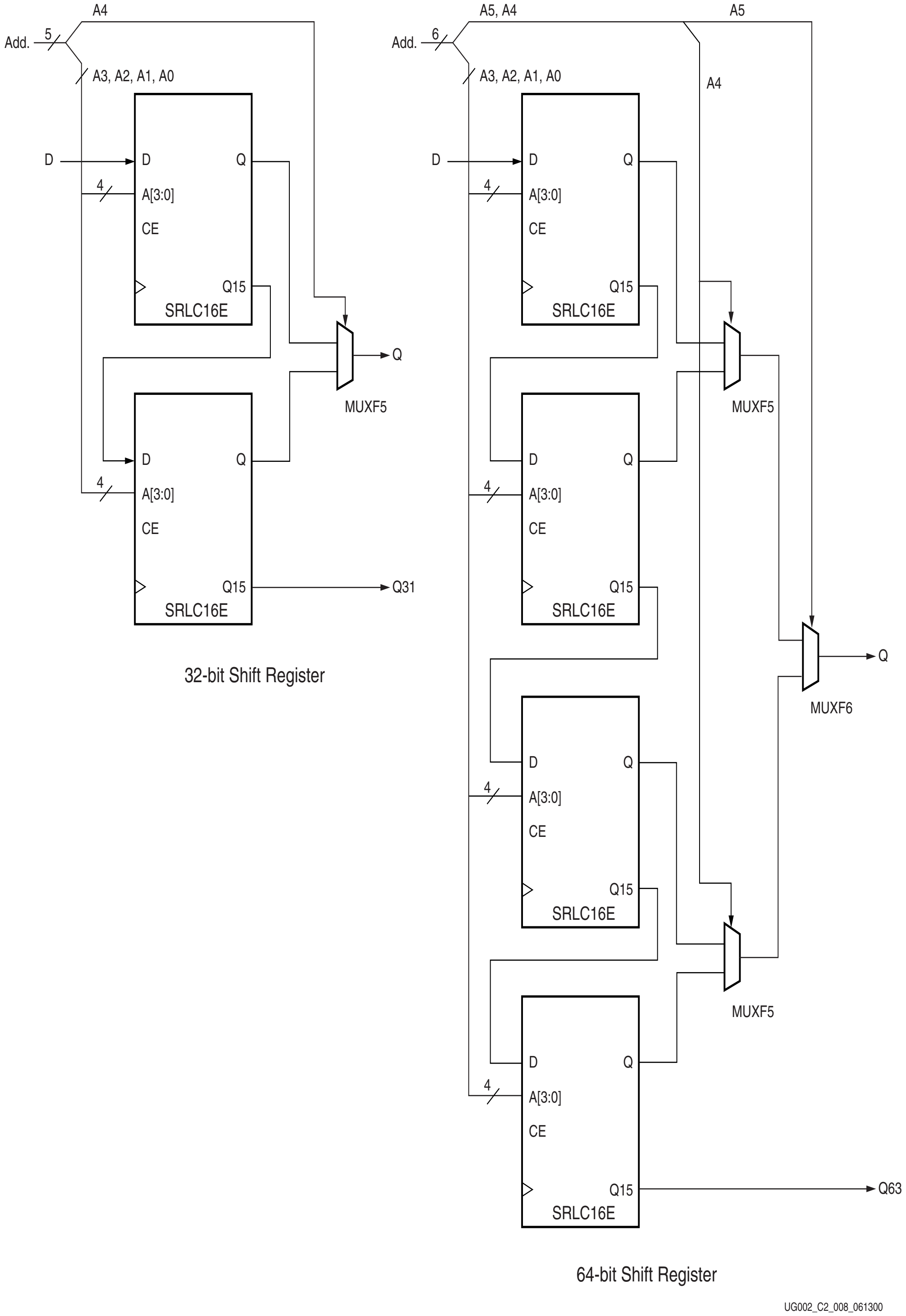


Figure 2-57: **Shift-Register Submodules (32-bit, 64-bit)**

A 128-bit shift register is built on the same scheme and uses MUXF7 (address input A6).

All clock enable (CE) and clock (CLK) inputs are connected to one global clock enable and one clock signal per submodule. If a global static- or dynamic-length mode is not required, the SRLC16E primitive can be cascaded without multiplexers.

Initialization in VHDL and Verilog Code

A shift register can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the attribute is attached to the 16-bit shift register instantiation and is copied in the EDIF output file to be compiled by Xilinx Alliance Series tools. The VHDL code simulation uses a `generic` parameter to pass the attributes. The Verilog code simulation uses a `defparam` parameter to pass the attributes.

The V2_SRL16E shift register instantiation code examples (in VHDL and Verilog) illustrate these techniques (see "VHDL and Verilog Templates" on page 232). V2_SRL16E.vhd and .v files are not a part of the documentation.

Port Signals

Clock - CLK

Either the rising edge or the falling edge of the clock is used for the synchronous shift-in. The data and clock enable input pins have set-up times referenced to the chosen edge of CLK.

Data In - D

The data input provides new data (one bit) to be shifted into the shift register.

Clock Enable - CE (optional)

The clock enable pin affects shift functionality. An inactive clock enable pin does not shift data into the shift register and does not write new data. Activating the clock enable allows the data in (D) to be written to the first location and all data to be shifted by one location. When available, new data appears on output pins (Q) and the cascadable output pin (Q15).

Address - A0, A1, A2, A3

Address inputs select the bit (range 0 to 15) to be read. The n^{th} bit is available on the output pin (Q). Address inputs have no effect on the cascadable output pin (Q15), which is always the last bit of the shift register (bit 15).

Data Out - Q

The data output Q provides the data value (1 bit) selected by the address inputs.

Data Out - Q15 (optional)

The data output Q15 provides the last bit value of the 16-bit shift register. New data becomes available after each shift-in operation.

Inverting Control Pins

The two control pins (CLK, CE) have an individual inversion option. The default is the rising clock edge and active High clock enable.

GSR

The global set/reset (GSR) signal has no impact on shift registers.

Attributes

Content Initialization - INIT

The INIT attribute defines the initial shift register contents. The INIT attribute is a hex-encoded bit vector with four digits (0000).The left-most hexadecimal digit is the most significant bit. By default the shift register is initialized with all zeros during the device configuration sequence, but any other configuration value can be specified.

Location Constraints

Each CLB resource has four slices: S0, S1, S2, and S3. As an example, in the bottom left CLB resource, each slice has the coordinates shown in Table 2-21.

Table 2-21: **Slice Coordinates in the Bottom-Left CLB Resource**

Slice S3	Slice S2	Slice S1	Slice S0
X1Y1	X1Y0	X0Y1	X0Y0

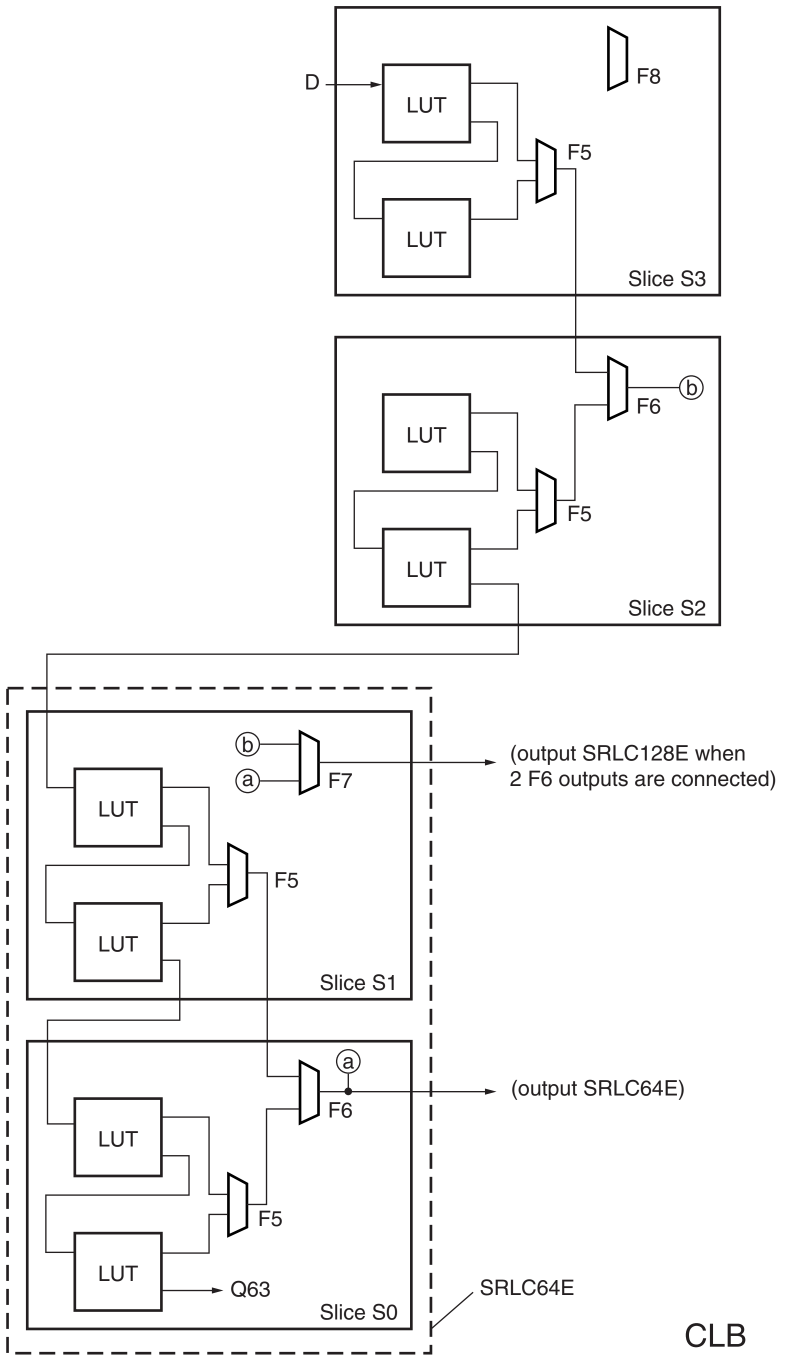
To constrain placement, shift register instances can have LOC properties attached to them. Each 16-bit shift register fits in one LUT.

A 32-bit shift register in static or dynamic address mode fits in one slice (two LUTs and one MUXF5). This shift register can be placed in any slice.

A 64-bit shift register in static or dynamic address mode fits in two slices. These slices are either S0 and S1, or S2 and S3. Figure 2-58 illustrates the position of the four slices in a CLB resource.

The dedicated CLB shift chain runs from the top slice to the bottom slice. The data input pin must either be in slice S1 or in S3. The address selected as the output pin (Q) is the MUXF6 output.

A 128-bit shift register in static or dynamic address mode fits in a four-slice CLB resource. The data input pin has to be in slice S3. The address selected as the output pin (Q) is the MUXF7 output.



UG002_C2_009_120400

Figure 2-58: **Shift Register Placement**

Fully Synchronous Shift Registers

All shift-register primitives and submodules do not use the register(s) available in the same slice(s). To implement a fully synchronous read and write shift register, output pin Q must be connected to a flip-flop. Both the shift register and the flip-flop share the same clock, as shown in Figure 2-59.

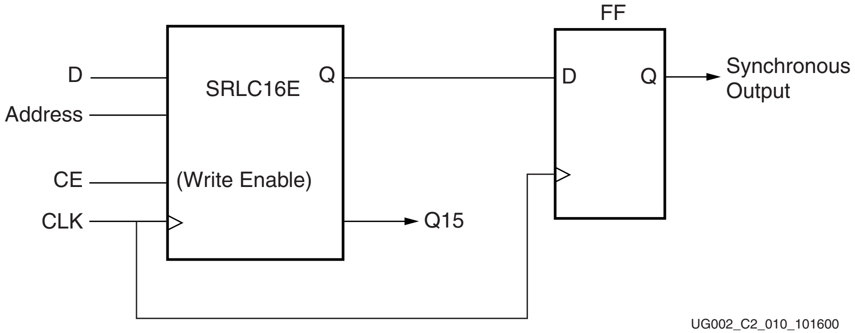


Figure 2-59: **Fully Synchronous Shift Register**

This configuration provides a better timing solution and simplifies the design. Because the flip-flop must be considered to be the last register in the shift-register chain, the static or dynamic address should point to the desired length minus one. If needed, the cascadable output can also be registered in a flip-flop.

Static-Length Shift Registers

The cascadable16-bit shift register implements any static length mode shift register without the dedicated multiplexers (MUXF5, MUXF6,…). Figure 2-60 illustrates a 40-bit shift register. Only the last SRLC16E primitive needs to have its address inputs tied to “0111”. Alternatively, shift register length can be limited to 39 bits (address tied to “0110”) and a flip-flop can be used as the last register. (In an SRLC16E primitive, the shift register length is the address input + 1.)

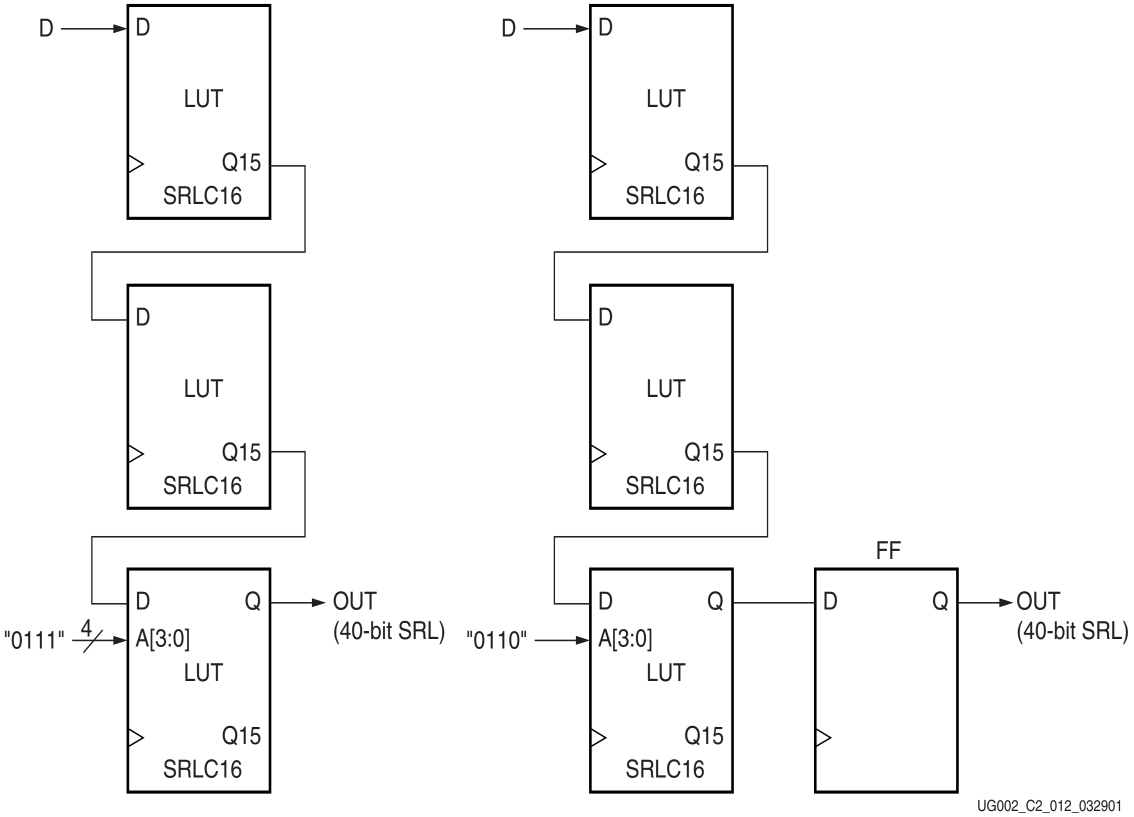


Figure 2-60: **40-bit Static-Length Shift Register**

VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available for all primitives and submodules.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The ShiftRegister_C_x (with x = 16, 32, 64, 128, or 256) templates are cascadable modules and instantiate the corresponding SRLCxE primitive (16) or submodule (32, 64, 128, or 256).

The ShiftRegister_16 template can be used to instantiate an SRL16 primitive.

VHDL and Verilog Templates

In template names, the number indicates the number of bits (for example, SHIFT_SELECT_16 is the template for the 16-bit shift register) and the "C" extension means the template is cascadable.

The following are templates for primitives:

- SHIFT_REGISTER_16
- SHIFT_REGISTER_16_C

The following are templates for submodules:

- SHIFT_REGISTER_32_C (submodule: SRLC32E_SUBM)
- SHIFT_REGISTER_64_C (submodule: SRLC64E_SUBM)
- SHIFT_REGISTER_128_C (submodule: SRLC128E_SUBM)

The corresponding submodules have to be synthesized with the design.

Templates for the SHIFT_REGISTER_16_C module are provided in VHDL and Verilog code as an example.

VHDL Template:

```
-- Module: SHIFT_REGISTER_C_16
-- Description: VHDL instantiation template
--  CASCADABLE 16-bit shift register with enable (SRLC16E)
-- Device: Virtex-II Family
---------------------------------------------------------------------
-- Components Declarations:
--
component SRLC16E
-- pragma translate_off
  generic (
-- Shift Register initialization ("0" by default) for functional
simulation:
        INIT : bit_vector := X"0000"
  );
-- pragma translate_on
  port (
        D : in std_logic;
        CE    : in std_logic;
        CLK   : in std_logic;
        A0    : in std_logic;
        A1    : in std_logic;
        A2    : in std_logic;
        A3    : in std_logic;
        Q     : out std_logic;
        Q15   : out std_logic
  );
end component;
```

```
-- Architecture  Section:
--
-- Attributes for Shift Register initialization (“0” by default):
attribute INIT: string;
--
attribute INIT of U_SRLC16E: label is “0000”;
--
-- ShiftRegister Instantiation
U_SRLC16E: SRLC16E
  port map (
  D       => , -- insert input signal
  CE      => , -- insert Clock Enable signal (optional)
  CLK     => , -- insert Clock signal
  A0      => , -- insert Address 0 signal
  A1      => , -- insert Address 1 signal
  A2      => , -- insert Address 2 signal
  A3      => , -- insert Address 3 signal
  Q       => , -- insert output signal
  Q15     =>   -- insert cascadable output signal
  );
```

Verilog Template:

```
// Module: SHIFT_REGISTER_16
// Description: Verilog instantiation template
// Cascadable 16-bit Shift Register with Clock Enable (SRLC16E)
// Device: Virtex-II Family
//--------------------------------------------------------------------
// Syntax for Synopsys FPGA Express
// synopsys translate_off

  defparam

//Shift Register initialization ("0" by default) for functional
simulation:
  U_SRLC16E.INIT = 16'h0000;
// synopsys translate_on

//SelectShiftRegister-II Instantiation
   SRLC16E U_SRLC16E   ( .D(),
                         .A0(),
                         .A1(),
                         .A2(),
                        .A3(),
                         .CLK(),
                         .CE(),
                         .Q(),
                        .Q15()
          );

// synthesis attribute declarations
  /* synopsys attribute
  INIT "0000"
  */
```

Designing Large Multiplexers

Introduction

Virtex-II slices contain dedicated two-input multiplexers (one MUXF5 and one MUXFX per slice). These multiplexers combine the 4-input LUT outputs or the outputs of other multiplexers. Using the multiplexers MUXF5, MUXF6, MUXF7 and MUXF8 allows to combine 2, 4, 8 and 16 LUTs. Specific routing resources are associated with these 2-input multiplexers to guarantee a fast implementation of any combinatorial function built upon LUTs and MUXFX.

The combination of the LUTs and the MUXFX offers an unique solution to the design of wide-input functions. This section illustrates the implementation of large multiplexers up to 32:1. Any Virtex-II slice can implement a 4:1 multiplexer, any CLB can implement a 16:1 multiplexer, and 2 CLBs can implement a 32:1 multiplexer. Such multiplexers are just one example of wide-input combinatorial function taking advantage of the MUXFX feature. Many other logic functions can be mapped in the LUT and MUXFX features.

This section provides generic VHDL and Verilog reference code implementing multiplexers. These submodules are built from LUTs and the dedicated MUXF5, MUXF6, MUXF7, and MUXF8 multiplexers. To automatically generate large multiplexers using these dedicated elements, use the CORE Generator Bit Multiplexer and Bus Multiplexer modules.

For applications like comparators, encoder-decoders or “case” statement in VHDL or Verilog, these resources offer an optimal solution.

Virtex-II CLB Resources

Slice Multiplexers

Each Virtex-II slice has a MUXF5 to combine the outputs of the 2 LUTs and an extra MUXFX. Figure 2-61 illustrates a combinatorial function with up to 9 inputs in one slice.

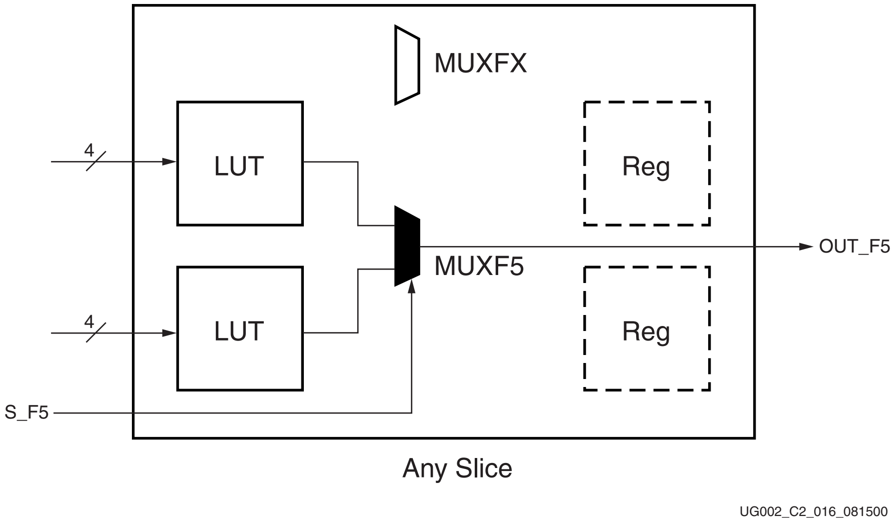


Figure 2-61: **LUTs and MUXF5 in a Slice**

Each Virtex-II CLB contains 4 slices. The second MUXFX implements a MUXF6, MUXF7 or MUXF8 according to the position of the slice in the CLB. These MUXFX are designed to allow LUTs combination up to 16 LUTs in two adjacent CLBs.

Figure 2-62 shows the relative position of the slices in the CLB.

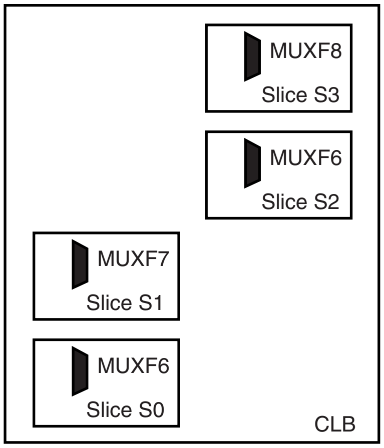


Figure 2-62: **Slice Positions in a CLB**

Slices S0 and S2 have a MUXF6, designed to combine the outputs of two MUXF5 resources. Figure 2-63 illustrates a combinatorial function up to 18 inputs in the slices S0 and S1, or in the slices S2 and S3.

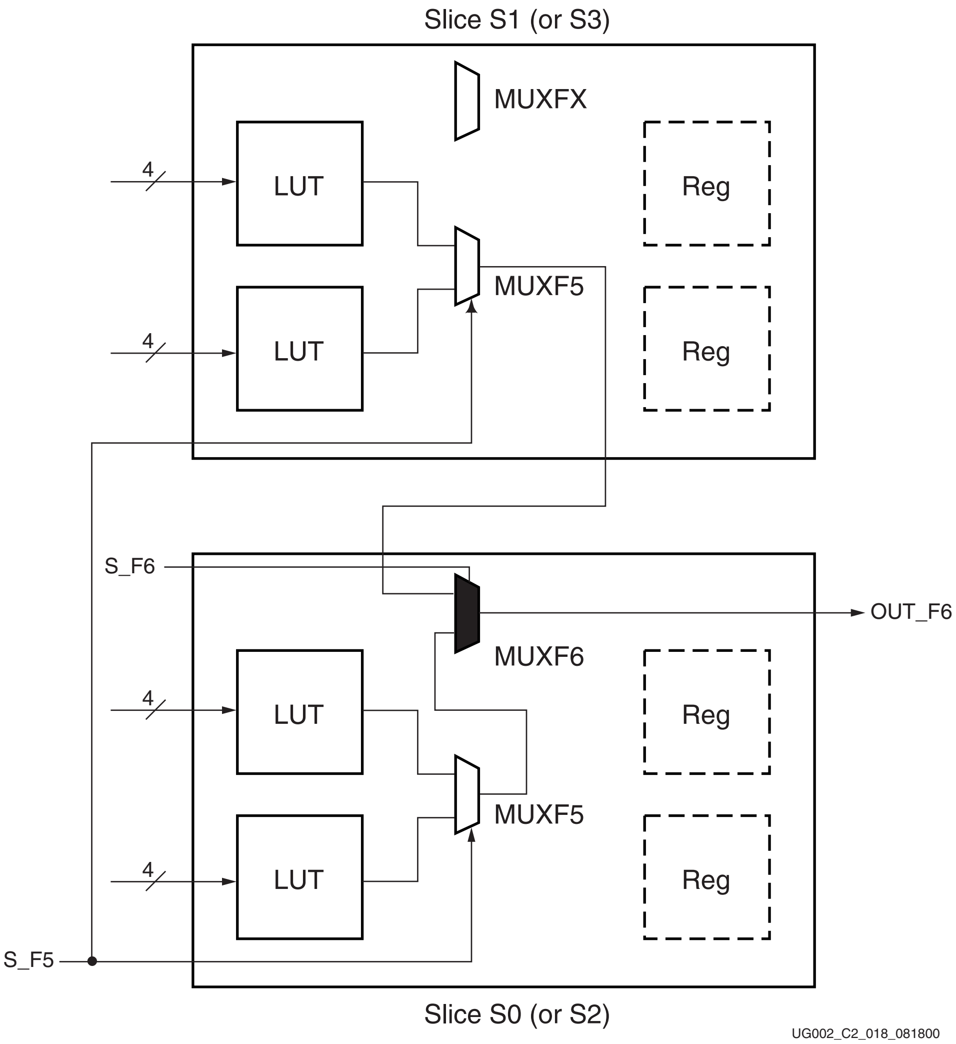
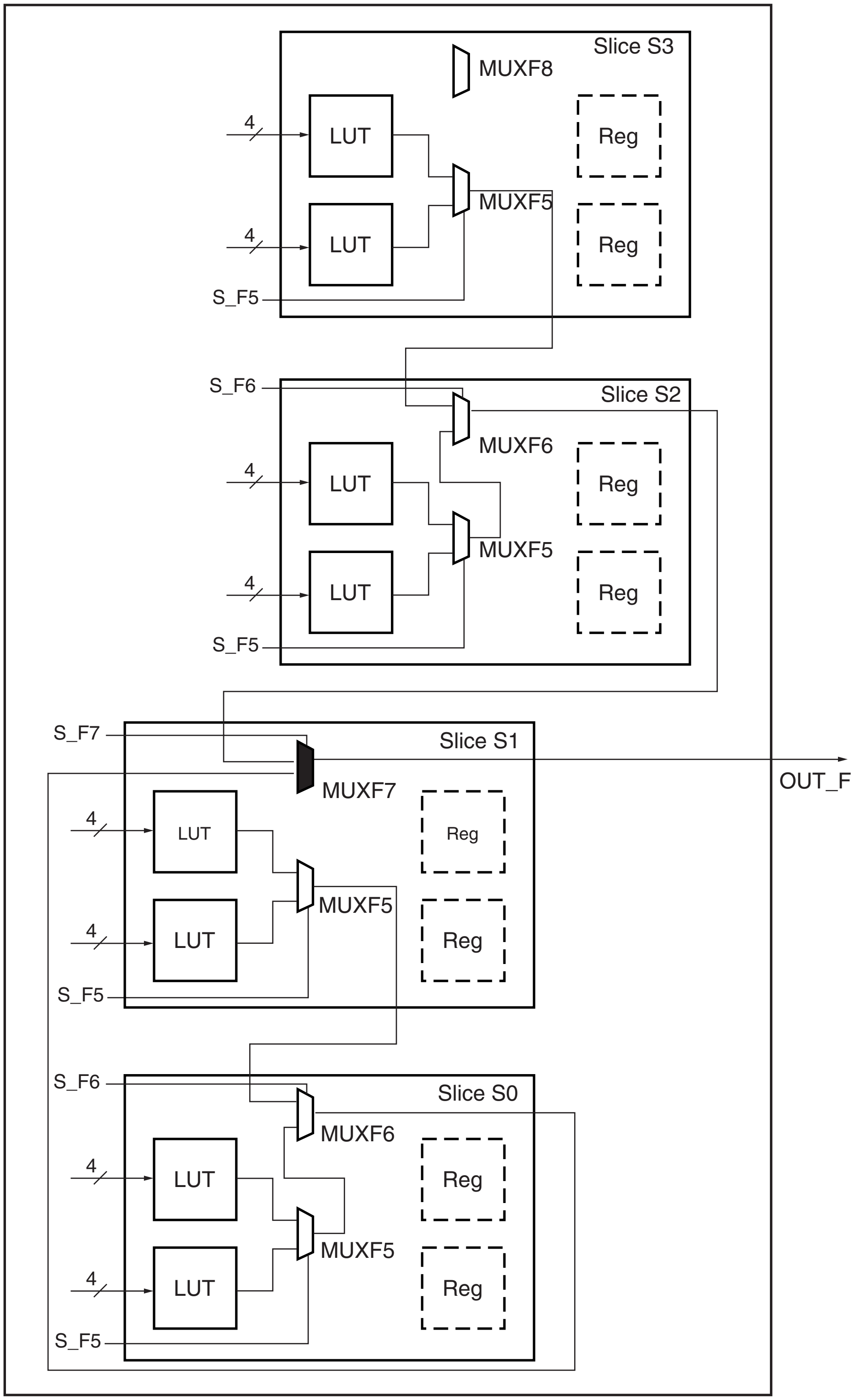


Figure 2-63: **LUTs and (MUXF5 and MUXF6) in Two Slices**

The slice S1 has a MUXF7, designed to combine the outputs of two MUXF6. Figure 2-64 illustrates a combinatorial function up to 35 inputs in a Virtex-II CLB.



UG002_C2_019_081600

Figure 2-64: **LUTs and (MUXF5, MUXF6, and MUXF7) in One CLB**

The slice S3 of each CLB has a MUXF8. combinatorial functions of up to 68 inputs fit in two CLBs as shown in Figure 2-65. The outputs of two MUXF7 are combined through dedicated routing resources between two adjacent CLBs in a column.

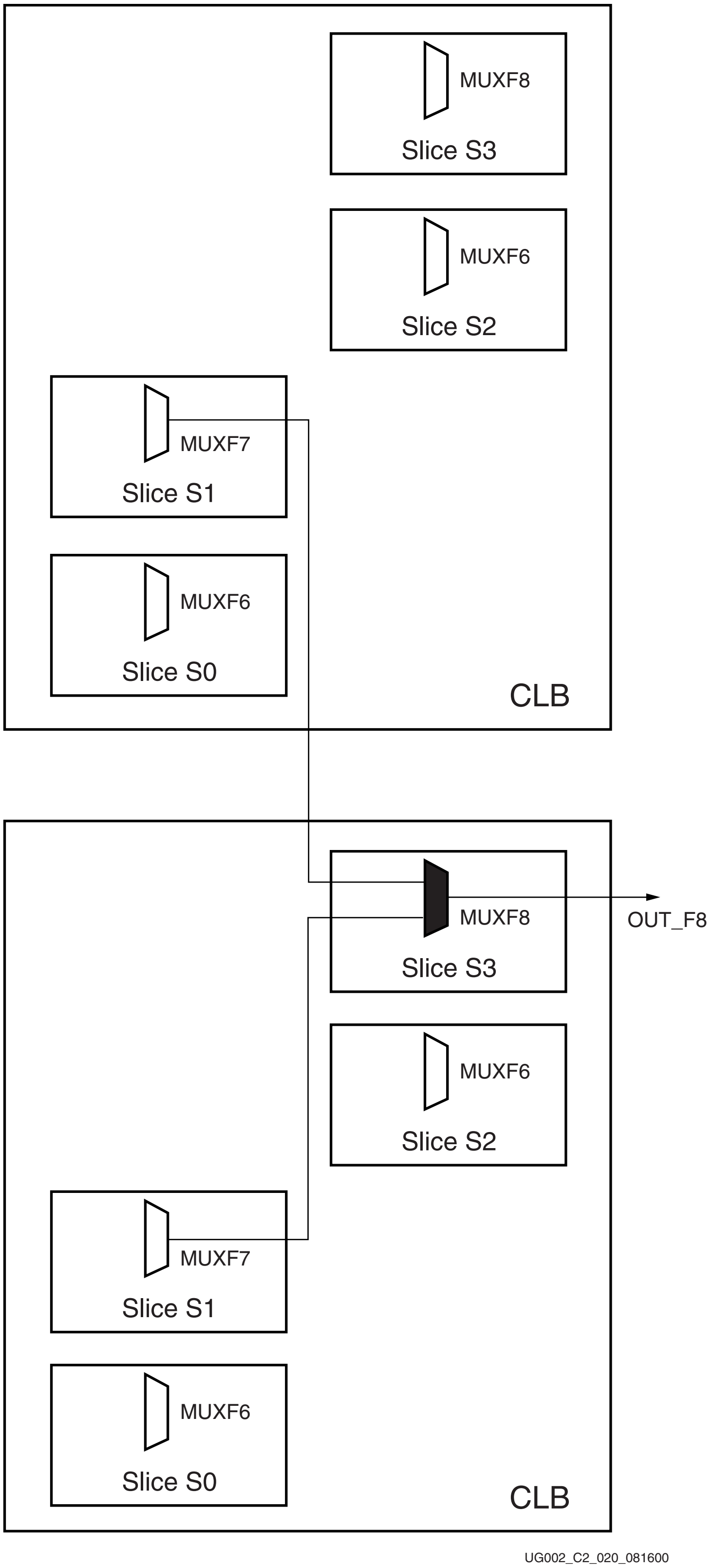


Figure 2-65: **MUXF8 Combining Two Adjacent CLBs**

Wide-Input Multiplexers

Each LUT can implement a 2:1 multiplexer. In each slice, the MUXF5 and two LUTs can implement a 4:1 multiplexer. As shown in Figure 2-66, the MUXF6 and two slices can implement a 8:1 multiplexer. The MUFXF7 and the four slices of any CLB can implement a 16:1 and the MUXF8 and two CLBs can implement a 32:1 multiplexer.

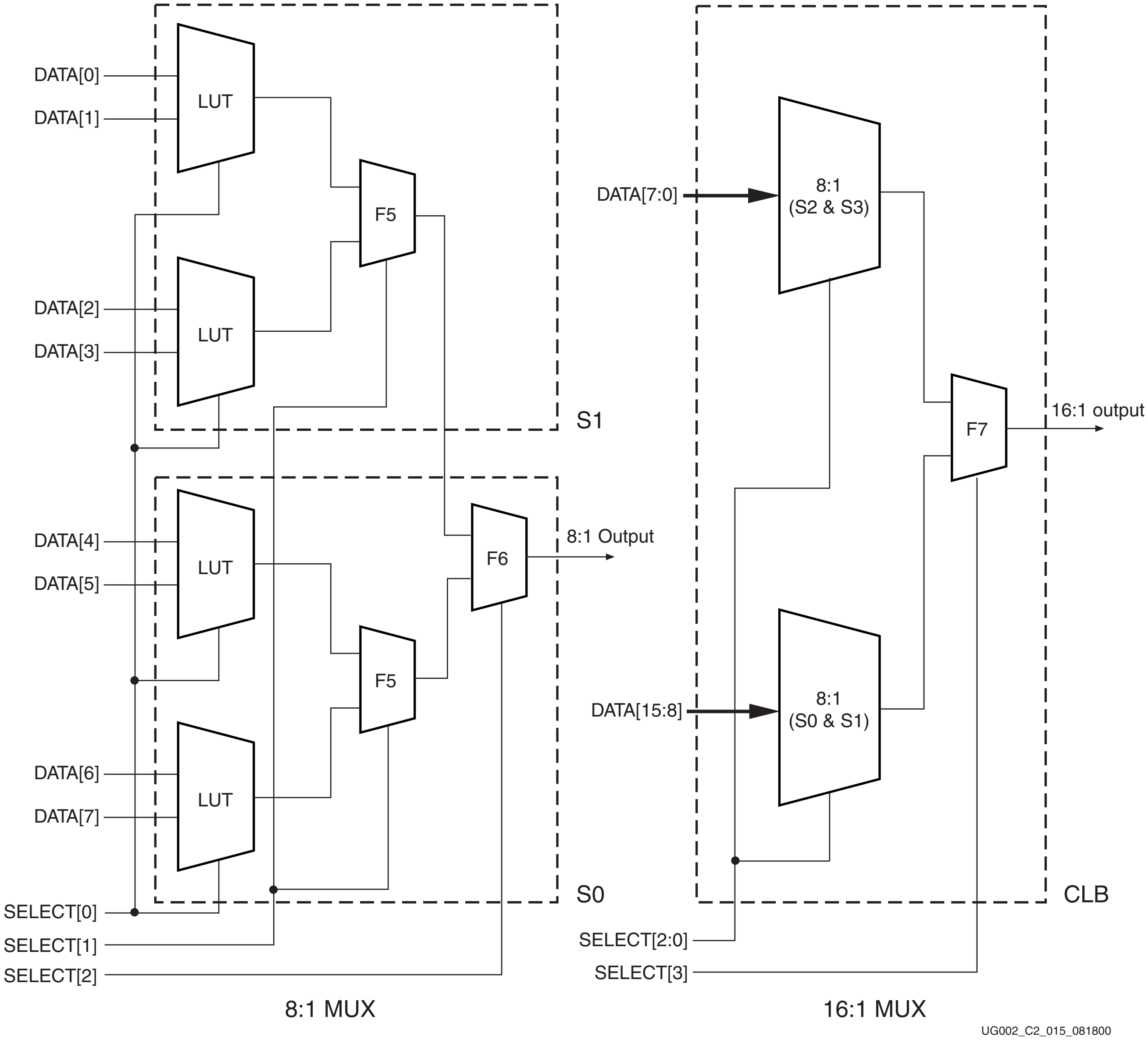


Figure 2-66: **8:1 and 16:1 Multiplexers**

Characteristics

- Implementation in one level of logic (LUT) and dedicated MUXFX
- Full combinatorial path

Library Primitives and Submodules

Four library primitives are available that offer access to the dedicated MUXFX in each slice.

In the example shown in Table 2-22, MUXF7 is available only in slice S1.

Table 2-22: **MUXFX Resources**

Primitive	Slice	Control	Input	Output
MUXF5	S0, S1, S2, S3	S	I0, I1	O
MUXF6	S0, S2	S	I0, I1	O
MUXF7	S1	S	I0, I1	O
MUXF8	S3	S	I0, I1	O

In addition to the primitives, five submodules that implement multiplexers from 2:1 to 32:1 are provided in VHDL and Verilog code. Synthesis tools can automatically infer the above primitives (MUXF5, MUXF6, MUXF7, and MUXF8); however, the submodules described in this section used instantiation of the new MUXFX to guarantee an optimized result. Table 2-23 lists available submodules:

Table 2-23: **Available Submodules**

Submodule	Multiplexer	Control	Input	Output
MUX_2_1_SUBM	2:1	SELECT_I	DATA_I[1:0]	DATA_O
MUX_4_1_SUBM	4:1	SELECT_I[1:0]	DATA_I[3:0]	DATA_O
MUX_8_1_SUBM	8:1	SELECT_I[2:0]	DATA_I[8:0]	DATA_O
MUX_16_1_SUBM	16:1	SELECT_I[3:0]	DATA_I[15:0]	DATA_O
MUX_32_1_SUBM	32:1	SELECT_I[4:0]	DATA_I[31:0]	DATA_O

Port Signals

Data In - DATA_I

The data input provides the data to be selected by the SELECT_I signal(s).

Control In - SELECT_I

The select input signal or bus determines the DATA_I signal to be connected to the output DATA_O. For example, the MUX_4_1_SUBM multiplexer has a 2-bit SELECT_I bus and a 4-bit DATA_I bus. Table 2-24 shows the DATA_I selected for each SELECT_I value.

Table 2-24: **Selected Inputs**

SELECT_I[1:0]	DATA_O
0 0	DATA_I[0]
0 1	DATA_I[1]
1 0	DATA_I[2]
1 1	DATA_I[3]

Data Out - DATA_O

The data output O provides the data value (1 bit) selected by the control inputs.

Applications

Multiplexers are used in various applications. These are often inferred by synthesis tools when a "case" statement is used (see the example below). Comparators, encoder-decoders and wide-input combinatorial functions are optimized when they are based on one level of LUTs and dedicated MUXFX resources of the Virtex-II CLBs.

VHDL and Verilog Instantiation

The primitives (MUXF5, MUXF6, and so forth) can be instantiated in VHDL or Verilog code, to design wide-input functions.

The submodules (MUX_2_1_SUBM, MUX_4_1_SUBM, and so forth) can be instantiated in VHDL or Verilog code to implement multiplexers. However the corresponding submodule must be added to the design directory as hierarchical submodule. For example, if a module is using the MUX_16_1_SUBM, the MUX_16_1_SUBM.vhd file (VHDL code) or MUX_16_1_SUBM.v file (Verilog code) must be compiled with the design source code. The submodule code can also be "cut and pasted" into the designer source code.

VHDL and Verilog Submodules

VHDL and Verilog submodules are available to implement multiplexers up to 32:1. They illustrate how to design with the MUXFX resources. When synthesis infers the corresponding MUXFX resource(s), the VHDL or Verilog code is behavioral code ("case" statement). Otherwise, the equivalent "case" statement is provided in comments and the correct MUXFX are instantiated. However, most synthesis tools support the inference of all of the MUXFX. The following examples can be used as guidelines for designing other wide-input functions.

The following submodules are available:

- MUX_2_1_SUBM (behavioral code)
- MUX_4_1_SUBM
- MUX_8_1_SUBM
- MUX_16_1_SUBM
- MUX_32_1_SUBM

The corresponding submodules have to be synthesized with the design

The submodule MUX_16_1_SUBM in VHDL and Verilog are provided as example.

VHDL Template

```
-- Module: MUX_16_1_SUBM
-- Description: Multiplexer 16:1
--
-- Device: Virtex-II Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

-- Syntax for Synopsys FPGA Express
-- pragma translate_off
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;
-- pragma translate_on

entity MUX_16_1_SUBM is
    port (
        DATA_I: in std_logic_vector (15 downto 0);
        SELECT_I: in std_logic_vector (3 downto 0);
        DATA_O: out std_logic
  );
```

```
end MUX_16_1_SUBM;

architecture MUX_16_1_SUBM_arch of MUX_16_1_SUBM is
-- Component Declarations:
component MUXF7
    port (
    I0: in std_logic;
    I1: in std_logic;
    S: in std_logic;
    O: out std_logic
    );
end component;
--
-- Signal Declarations:
signal DATA_MSB : std_logic;
signal DATA_LSB : std_logic;
--
begin
--
-- If synthesis tools support MUXF7 :
--SELECT_PROCESS: process (SELECT_I, DATA_I)
--begin
--case SELECT_I is
--  when "0000" => DATA_O <= DATA_I (0);
--  when "0001" => DATA_O <= DATA_I (1);
--  when "0010" => DATA_O <= DATA_I (2);
--  when "0011" => DATA_O <= DATA_I (3);
--  when "0100" => DATA_O <= DATA_I (4);
--  when "0101" => DATA_O <= DATA_I (5);
--  when "0110" => DATA_O <= DATA_I (6);
--  when "0111" => DATA_O <= DATA_I (7);
--  when "1000" => DATA_O <= DATA_I (8);
--  when "1001" => DATA_O <= DATA_I (9);
--  when "1010" => DATA_O <= DATA_I (10);
--  when "1011" => DATA_O <= DATA_I (11);
--  when "1100" => DATA_O <= DATA_I (12);
--  when "1101" => DATA_O <= DATA_I (13);
--  when "1110" => DATA_O <= DATA_I (14);
--  when "1111" => DATA_O <= DATA_I (15);
--  when others => DATA_O <= 'X';
--end case;
--end process SELECT_PROCESS;
--
-- If synthesis tools DO NOT support MUXF7 :
SELECT_PROCESS_LSB: process (SELECT_I, DATA_I)
begin
  case SELECT_I (2 downto 0) is
    when "000" => DATA_LSB <= DATA_I (0);
    when "001" => DATA_LSB <= DATA_I (1);
    when "010" => DATA_LSB <= DATA_I (2);
    when "011" => DATA_LSB <= DATA_I (3);
    when "100" => DATA_LSB <= DATA_I (4);
    when "101" => DATA_LSB <= DATA_I (5);
    when "110" => DATA_LSB <= DATA_I (6);
    when "111" => DATA_LSB <= DATA_I (7);
    when others => DATA_LSB <= 'X';
  end case;
end process SELECT_PROCESS_LSB;
--
SELECT_PROCESS_MSB: process (SELECT_I, DATA_I)
begin
  case SELECT_I (2 downto 0) is
```

```
   when "000" => DATA_MSB <= DATA_I (8);
   when "001" => DATA_MSB <= DATA_I (9);
   when "010" => DATA_MSB <= DATA_I (10);
   when "011" => DATA_MSB <= DATA_I (11);
   when "100" => DATA_MSB <= DATA_I (12);
   when "101" => DATA_MSB <= DATA_I (13);
   when "110" => DATA_MSB <= DATA_I (14);
   when "111" => DATA_MSB <= DATA_I (15);
   when others => DATA_MSB <= 'X';
  end case;
end process SELECT_PROCESS_MSB;
--
-- MUXF7 instantiation
U_MUXF7: MUXF7
    port map (
    I0 => DATA_LSB,
    I1 => DATA_MSB,
    S  => SELECT_I (3),
    O  => DATA_O
    );
--
end MUX_16_1_SUBM_arch;
--
```

Verilog Template

```
// Module: MUX_16_1_SUBM
//
// Description: Multiplexer 16:1
// Device: Virtex-II Family
//-------------------------------------------------------------------
//
module MUX_16_1_SUBM (DATA_I, SELECT_I, DATA_O);

input [15:0]DATA_I;
input [3:0]SELECT_I;

output DATA_O;

wire [2:0]SELECT;

reg DATA_LSB;
reg DATA_MSB;

assign SELECT[2:0] = SELECT_I[2:0];

/*
//If synthesis tools supports MUXF7 :
always @ (DATA_I or SELECT_I)

    case (SELECT_I)
       4'b0000 : DATA_O <= DATA_I[0];
  4'b0001 : DATA_O <= DATA_I[1];
  4'b0010 : DATA_O <= DATA_I[2];
  4'b0011 : DATA_O <= DATA_I[3];
        4'b0100 : DATA_O <= DATA_I[4];
  4'b0101 : DATA_O <= DATA_I[5];
  4'b0110 : DATA_O <= DATA_I[6];
  4'b0111 : DATA_O <= DATA_I[7];
        4'b1000 : DATA_O <= DATA_I[8];
  4'b1001 : DATA_O <= DATA_I[9];
  4'b1010 : DATA_O <= DATA_I[10];
  4'b1011 : DATA_O <= DATA_I[11];
```

```
        4'b1100 : DATA_O <= DATA_I[12];
  4'b1101 : DATA_O <= DATA_I[13];
  4'b1110 : DATA_O <= DATA_I[14];
  4'b1111 : DATA_O <= DATA_I[15];
  default : DATA_O <= 1'bx;
    endcase
*/

always @ (SELECT or DATA_I)

    case (SELECT)
        3'b000 : DATA_LSB <= DATA_I[0];
  3'b001 : DATA_LSB <= DATA_I[1];
  3'b010 : DATA_LSB <= DATA_I[2];
  3'b011 : DATA_LSB <= DATA_I[3];
        3'b100 : DATA_LSB <= DATA_I[4];
  3'b101 : DATA_LSB <= DATA_I[5];
  3'b110 : DATA_LSB <= DATA_I[6];
  3'b111 : DATA_LSB <= DATA_I[7];
  default : DATA_LSB <= 1'bx;
    endcase

always @ (SELECT or DATA_I)

    case (SELECT)
        3'b000 : DATA_MSB <= DATA_I[8];
  3'b001 : DATA_MSB <= DATA_I[9];
  3'b010 : DATA_MSB <= DATA_I[10];
  3'b011 : DATA_MSB <= DATA_I[11];
        3'b100 : DATA_MSB <= DATA_I[12];
  3'b101 : DATA_MSB <= DATA_I[13];
  3'b110 : DATA_MSB <= DATA_I[14];
  3'b111 : DATA_MSB <= DATA_I[15];
  default : DATA_MSB <= 1'bx;
    endcase

// MUXF7 instantiation

MUXF7 U_MUXF7   (.I0(DATA_LSB),
     .I1(DATA_MSB),
     .S(SELECT_I[3]),
     .O(DATA_O)
     );
endmodule

//
*/
```

Implementing Sum of Products (SOP) Logic

Introduction

Virtex-II slices contain a dedicated two-input multiplexer (MUXCY) and a two-input OR gate (ORCY) to perform operations involving wide AND and OR gates. These combine the four-input LUT outputs. These gates can be cascaded in a chain to provide the wide AND functionality across slices. The output from the cascaded AND gates can then be combined with the dedicated ORCY to produce the Sum of Products (SOP).

Virtex-II CLB Resources

Each Virtex-II slice has a MUXCY, which uses the output from the LUTs as a SELECT signal. Depending on the width of data desired, several slices can be used to provide the SOP output. Figure 2-67 illustrates the logic involved in designing a 16-input AND gate. It utilizes the 4-input LUT to provide the necessary SELECT signal for the MUXCY. Only when all of the input signals are High, can the V_{CC} at the bottom reach the output. This use of carry logic helps to perform AND functions at high speed and saves logic resources.

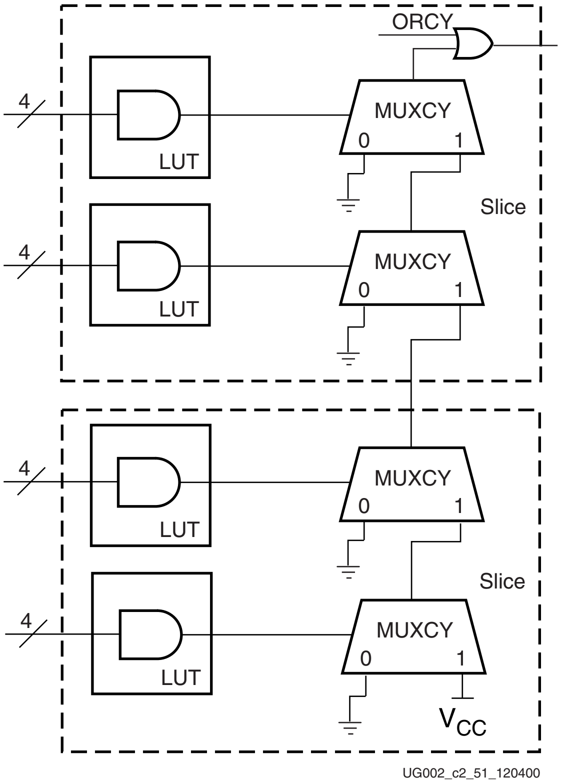
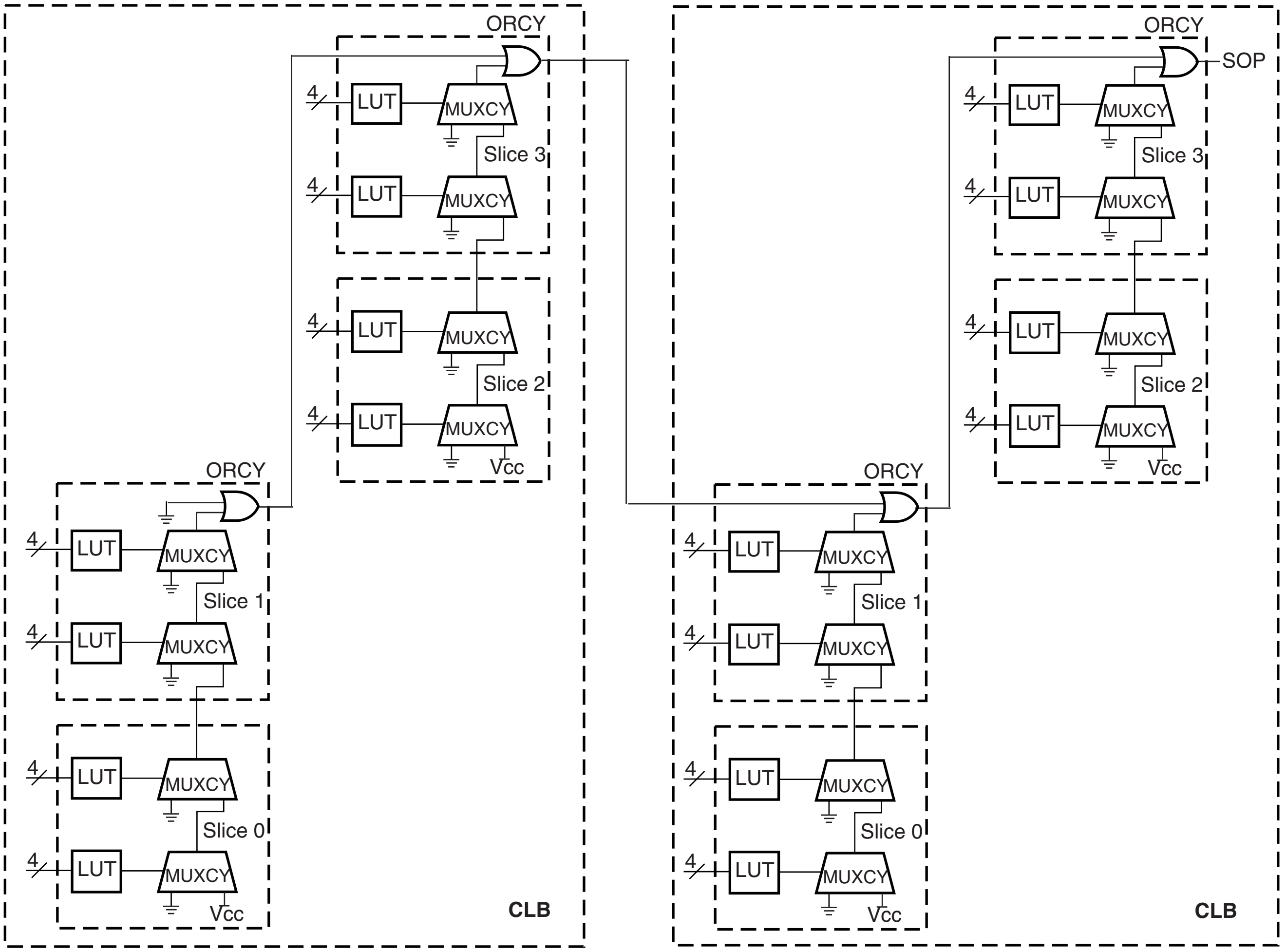


Figure 2-67: **Implementing a 16-bit Wide AND Gate Using MUXCY & ORCY**

The output from the chain of AND gates is passed as one of the inputs of the dedicated OR gate, ORCY. To calculate the SOP, these AND chains can be cascaded vertically across several CLBs, depending on the width of the input data. Figure 2-68 illustrates how the AND outputs are then passed in through the ORCY gates in a horizontal cascade, the sum of which is the Sum of Products.



UG002_c2_42_120400

Figure 2-68: **64-bit Input SOP Design**

Port Signals

AND_WIDTH Parameter

The width of each AND gate used in the cascade.

PROD_TERM Parameter

The number of AND gates used along each vertical cascade.

AND_IN Parameter

Data input to the AND gates. The total width of data is calculated from the product of AND_WIDTH and PROD_TERM

SOP_OUT Parameter

The Sum of Products (SOP) output data from the cascade chain.

Applications

These logic gates can be used in various applications involving very wide AND gates and Sum of Products (SOP) functions.

VHDL and Verilog Instantiation

To implement wide-input AND functions, MUXCY and ORCY primitives can be instantiated in VHDL or Verilog code. The submodule code provided can be used to implement wide-input AND gates for any width of input data.

VHDL and Verilog Submodules

VHDL and Verilog submodules are available to implement the cascade chain of wide-input AND gates and OR gates to calculate the Sum of Products (SOP). The VHDL module provided uses a generic case, where the width of data and the product terms can be specified in the case. The Verilog module provides a 64-bit input example, using four wide AND chains, each of which handle 16 bits of data.

VHDL Templates

```
-- Module : AND_CHAIN
-- Description : 16 input AND gate
--
-- Device : Virtex-II Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--library UNISIM;
--use UNISIM.VCOMPONENTS.ALL;

entity AND_CHAIN is
 generic (
    input_width : integer); --must be a 4x value
 port (
    data_in : in std_logic_vector( input_width-1 downto 0);
    carry_in : in std_logic;
    out_andor_chain : out std_logic);
end AND_CHAIN;

architecture AND_CHAIN_arch of AND_CHAIN is

component ORCY
  port( i : std_logic;
        ci : in std_logic;
        o : out std_logic);
end component;

component AND_LOGIC
  port( sel_data : in std_logic_vector(3 downto 0);
        data_cin : in std_logic;
        data_out : out std_logic);
end component;

signal VCC, GND : std_logic;
signal cout : std_logic_vector(input_width/4 downto 0);
signal out_and_chain : std_logic;

begin

VCC <= '1';
GND <= '0';

--initialisation of first input for MUXCY
cout(0) <= VCC;

and_chain_x : for i in (input_width/4) - 1 downto 0 generate
```

```
  AND_LOGIC_inst : AND_LOGIC
        port map (
           sel_data => data_in((4 * i + 3) downto (4 * i)),
           data_cin => cout(i),
           data_out => cout(i + 1));
end generate;

out_and_chain <= cout(input_width/4);

orcy_inst : ORCY
  port map( i => out_and_chain,
        ci => carry_in,
        o => out_andor_chain);

end AND_CHAIN_arch;

---------------------------------------------------------------------
-- Module AND_LOGIC
-- Description : 4-input AND gate
--
-- Device : Virtex-II Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--library UNISIM;
--use UNISIM.VCOMPONENTS.ALL;

entity AND_LOGIC is
  port(
      sel_data : in std_logic_vector(3 downto 0); -- data for select
signal for MUXCY from LUT
      data_cin : in std_logic;  -- result from previous stage
      data_out : out std_logic);
end AND_LOGIC;

architecture AND_LOGIC_arch of AND_LOGIC is

component MUXCY
 port(
     DI : in std_logic;
     CI : in std_logic;
     s : in std_logic;
     o : out std_logic);
end component;

signal GND : std_logic;
signal sel:std_logic;

begin

GND <= '0';
sel <= sel_data(0) and sel_data(1) and sel_data(2) and sel_data(3);

--Wide AND gate using MUXCY
MUX : MUXCY
     port map (
        DI => GND,
        CI => data_cin,
        s => sel,
        o => data_out);

end AND_LOGIC_arch;
```

```
---------------------------------------------------------------------
-- Module : SOP_SUBM
-- Description : Implementing SOP using MUXCY and ORCY
--
-- Device : Virtex-II Family
---------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
--library UNISIM;
--use UNISIM.VCOMPONENTS.ALL;

entity SOP_SUBM is
  generic(
     and_width : integer :=16 ;
     prod_term : integer := 4 );
  port(
     and_in : in std_logic_vector(and_width * prod_term - 1 downto 0);
     sop_out : out std_logic);
end SOP_SUBM;

architecture SOP_SUBM_arch of SOP_SUBM is

component AND_CHAIN
 generic (
    input_width : integer); --must be a 4x value
 port (
    data_in : in std_logic_vector( input_width-1 downto 0);
    carry_in : in std_logic;
    out_andor_chain : out std_logic);
end component;

signal VCC, GND : std_logic;
signal carry : std_logic_vector(prod_term downto 0);

begin

VCC <= '1';
GND <= '0';

carry(0) <= GND;
andor_inst : for i in 0 to (prod_term - 1) generate
      and_chainx : AND_CHAIN
             generic map(
                input_width => and_width)
             port map(
              data_in => and_in((and_width * i + (and_width -1)) downto
(and_width * i)),
                carry_in => carry(i),
                out_andor_chain => carry(i + 1));
end generate;
sop_out <= carry(prod_term);

end SOP_SUBM_arch;
```

Verilog Templates

```
// Module : AND_CHAIN
// Description : 16 input AND gate
//
// Device : Virtex-II Family
//-------------------------------------------------------------------
module AND_CHAIN(data_in, carry_in, out_andor_chain);
input [15:0] data_in;
input carry_in;
output out_andor_chain;
wire VCC = 1'b1;
wire out_and_chain;
wire dat_out1, data_out2, data_out3;
AND_LOGIC_OR u4(.sel_data(data_in[15:12]), .data_cin(data_out3),
.carry_in(carry_in), .data_out(out_andor_chain));

AND_LOGIC u3(.sel_data(data_in[11:8]), .data_cin(data_out2),
.data_out(data_out3));
AND_LOGIC u2(.sel_data(data_in[7:4]), .data_cin(data_out1),
.data_out(data_out2));
AND_LOGIC u1(.sel_data(data_in[3:0]), .data_cin(VCC),
.data_out(data_out1));
endmodule

//-------------------------------------------------------------------
// Module AND_LOGIC
// Description : 4-input AND gate
//
// Device : Virtex-II Family
//-------------------------------------------------------------------
// Module : init_and
//
module AND_LOGIC(sel_data, data_cin, data_out);
input[3:0] sel_data;
input data_cin;
output data_out;
wire GND = 1'b0;
wire VCC = 1'b1;
wire and_out;
assign and_out = sel_data[3] & sel_data[2] & sel_data[1] & sel_data[0];
MUXCY muxcy_inst (.DI(GND), .CI(data_cin), .S(and_out), .O(data_out));
endmodule

// Module AND_LOGIC + ORCY
module AND_LOGIC_OR(sel_data, data_cin, carry_in, data_out);
input[3:0] sel_data;
input data_cin;
input carry_in;
output data_out;
wire data_mux_out;
wire GND = 1'b0;
wire VCC = 1'b1;
wire and_out;
assign and_out = sel_data[3] & sel_data[2] & sel_data[1] & sel_data[0];
MUXCY muxcy_inst (.DI(GND), .CI(data_cin), .S(and_out),
.O(data_mux_out)) /* synthesis  RLOC="x0y0" */;
ORCY u5(.I(carry_in), .CI(data_mux_out), .O(data_out)) /* synthesis
RLOC="x0y0" */;
endmodule
```

```
//---------------------------------------------------------------------
// Module : SOP_SUBM
// Description : Implementing SOP using MUXCY and ORCY
//
// Device : Virtex-II Family
//---------------------------------------------------------------------
module SOP_SUBM(and_in, sop_out);
input [63:0] and_in;
output sop_out;
wire out_andor_chain1, out_andor_chain2, out_andor_chain3;
wire GND = 1'b0;
AND_CHAIN u4(.data_in(and_in[63:48]), .carry_in(out_andor_chain3),
.out_andor_chain(sop_out));
AND_CHAIN u3(.data_in(and_in[47:32]), .carry_in(out_andor_chain2),
.out_andor_chain(out_andor_chain3));
AND_CHAIN u2(.data_in(and_in[31:16]), .carry_in(out_andor_chain1),
.out_andor_chain(out_andor_chain2));
AND_CHAIN u1(.data_in(and_in[15:0]), .carry_in(GND),
.out_andor_chain(out_andor_chain1));
endmodule
```

Using Embedded Multipliers

Introduction

Virtex-II devices feature a large number of embedded 18-bit X 18-bit two's-complement embedded multipliers. The embedded multipliers offer fast, efficient means to create 18-bit signed by 18-bit signed multiplication products. The multiplier blocks share routing resources with the Block SelectRAM memory, allowing for increased efficiency for many applications. Cascading of multipliers can be implemented with additional logic resources in local Virtex-II slices.

Applications such as signed-signed, signed-unsigned, and unsigned-unsigned multiplication, logical, arithmetic, and barrel shifters, two's-complement and magnitude return are easily implemented.

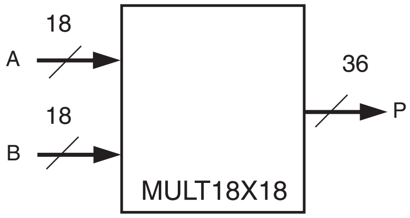
Using the CORE Generator, the designer can quickly generate multipliers that make use of the embedded 18-bit x 18-bit two's-complement multipliers (V2.0 or later) of the Multiplier core for Virtex-II devices.

Two's-Complement Signed Multiplier

Data Flow

Each embedded multiplier block (MULT18X18 primitive) supports two independent dynamic data input ports: 18-bit signed or 17-bit unsigned. The MULT18X18 primitive is illustrated in Figure 2-69.

In addition, efficient cascading of multipliers up to 35-bit X 35-bit signed can be accomplished by using 4 embedded multipliers, one 36-bit adder, and one 53-bit adder. See Figure 2-70.



UG002_C2_025_082100

Figure 2-69: **Embedded Multiplier**

Library Primitives and Submodules

One library primitive (MULT18X18) is available. Table 2-25 lists the attributes of this primitive.

Table 2-25: **Embedded Multiplier Primitive**

Primitive	A width	B width	P width	Signed/Unsigned
MULT18X18	18	18	36	Signed (2's complement)

In addition to the primitive, 15 submodules that implement various widths of signed and unsigned multipliers and two's-complement return functions are provided in VHDL and Verilog code. Multipliers using cascaded MULT18X18 primitives are included with registers between stages causing three cycles of latency. Multipliers that make use of the embedded Virtex-II 18-bit by 18-bit two's complement multipliers can be easily generated using V2.0 of the CORE Generator Multiplier module. Table 2-26 lists cascaded multiplier submodules.

Table 2-26: **Embedded Multiplier Submodules - Cascaded MULT18X18**

Submodule	A Width	B Width	P Width	Signed/Unsigned
MULT35X35_S	35	35	70	Signed
MULT34X34_U	34	34	68	Unsigned

Figure 2-70 represents the cascaded scheme used to implement a 35-bit by 35-bit signed multiplier utilizing four embedded multipliers and two adders.

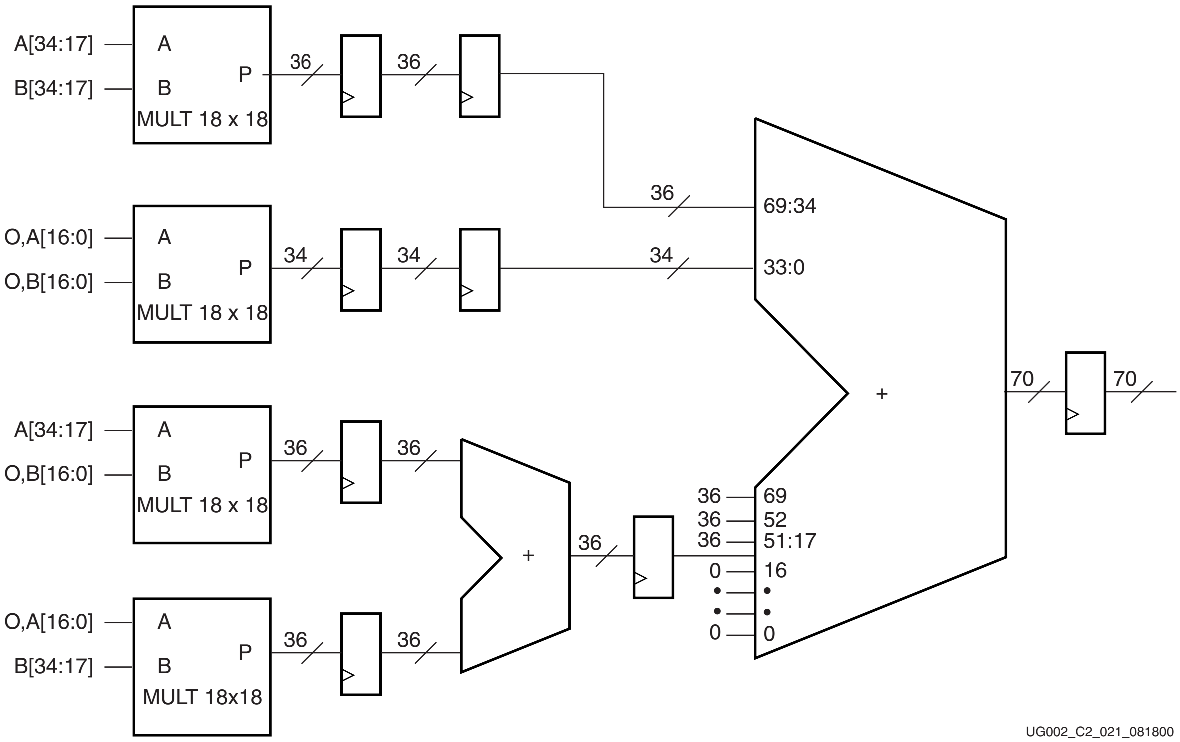


Figure 2-70: **MULT35X35_S Submodule**

The fixed adder is 53 bits wide (17 LSBs are always 0 on one input).

The 34-bit by 34-bit unsigned submodule is constructed in a similar manner with the most significant bit on each operand being tied to logic low.

Table 2-26 lists multipliers and two's-complement return functions that utilize one MULT18X18 primitive and are not registered.

Table 2-27: **Embedded Multiplier Submodules - Single MULT18X18**

Submodule	A width	B width	P width	Signed/Unsigned
MULT17X17_U	17	17	34	Unsigned
MULT8X8_S	8	8	16	Signed
MULT8X8_U	8	8	16	Unsigned
MULT4X4_S	4	4	8	Signed
MULT4X4_U	4	4	8	Unsigned
MULT_6X6S_5X5U	6 5	6 5	12 10	Signed Unsigned
MULT_5X5S_6X6U	5 6	5 6	10 12	Signed Unsigned
MULT_5X5U_5X5U	5 5	5 5	10 10	Unsigned Unsigned
MULT_4X4S_7X7U	4 7	4 7	8 14	Signed Unsigned
MULT_4X4S_3X3S	4 3	4 3	8 6	Signed Signed
TWOS_CMP18	18	-	18	-
TWOS_CMP9	9	-	9	-
MAGNTD_18	18	-	17	-

Multipliers of form MULT_aXaS_bXbU use one embedded multiplier to implement two multipliers with separate outputs. The submodules listed above use optimized pin assignments to achieve shortest possible through-delay.

Figure 2-71 and Figure 2-72 represent 4-bit by 4-bit signed multiplier and 4-bit by 4-bit unsigned multiplier implementations, respectively.

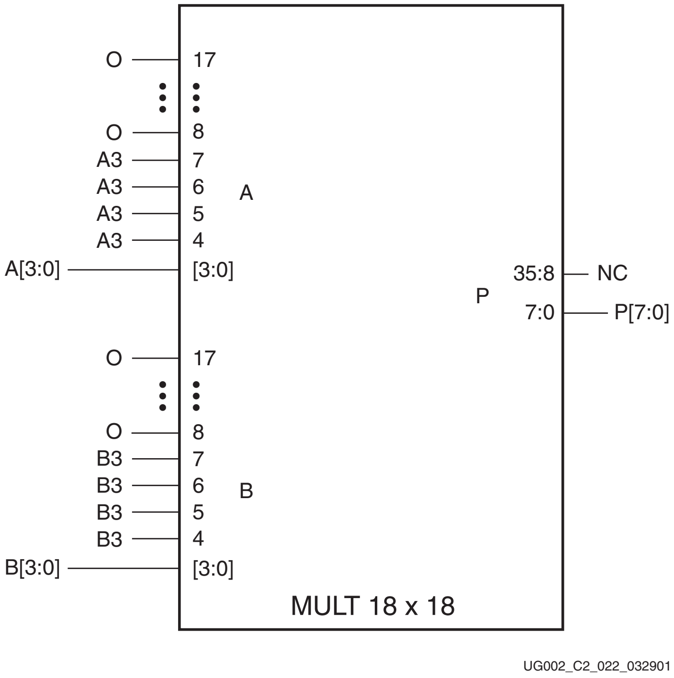


Figure 2-71: **MULT4X4_S Submodule**

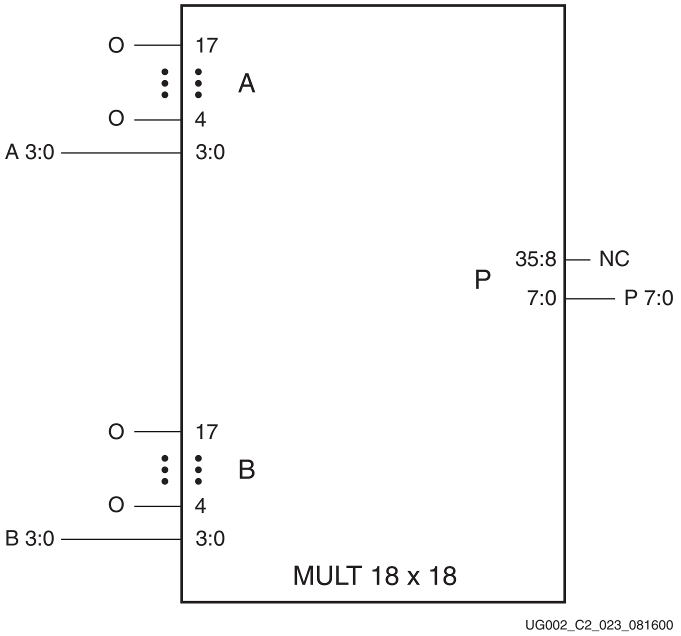


Figure 2-72: **MULT4X4_U Submodule**

Submodule MAGNTD_18 performs a magnitude return (i.e., absolute value) of a two's-complement number. An incoming negative number returns with a positive number, while an incoming positive number remains unchanged. Submodules TWOS_CMP18 and TWOS_CMP9 perform a two's-complement return function. The incoming number in two's-complement form (either signed or unsigned) is complemented when the DO_COMP pin is asserted High. Additional slice logic can be used with these submodules to efficiently convert sign-magnitude to two's-complement or vice-versa. Figure 2-73 shows the connections to a MULT18X18 to create the submodule TWOS_CMP9.

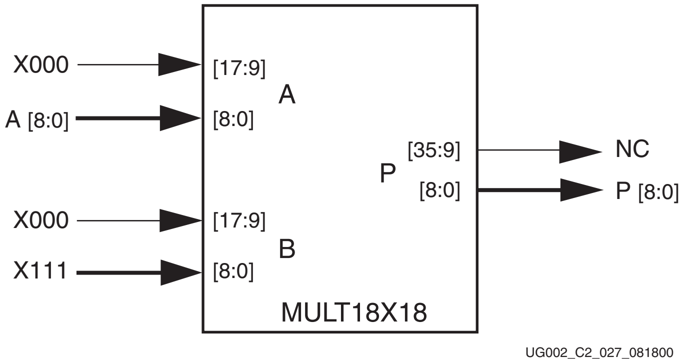


Figure 2-73: **TWOS_CMP9 Submodule**

Two Multipliers in a Single Primitive

Two multipliers can be implemented in a single primitive. For simplified illustration purposes, an assumption of two squares being implemented in the same MULT18X18 primitive is used. The following equation shows the form of the multiplication.

Two Multipliers per Primitive:

$$(X * 2^n + Y)(X * 2^n + Y) = (X^2 * 2^{2n}) + (Y^2) + (XY * 2^{n+1})$$

$(X * 2^n)$ is the input X appearing on the MSBs while Y appears on the LSBs to form the value $(X * 2^n + Y)$. Two multipliers can coexist in one MULT18X18 primitive, if the conditions in the following inequalities are met when neither X nor Y are 0.

Inequality Conditions for Two Multipliers per Primitive:

$$(X^2 * 2^{2n})_{min} > (XY * 2^{n+1})_{max},\ (XY * 2^{n+1})_{min} > (Y^2)_{max}$$

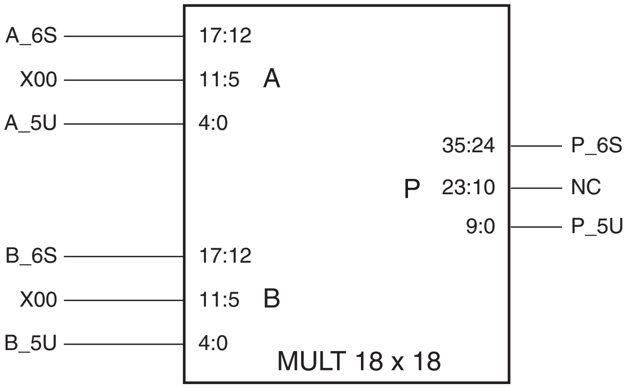
For values 0 on X or Y, the equation becomes:

$$X^2 * 2^{2n} \quad \{Y=0\}$$

$$Y^2 \quad \{X=0\}$$

$$0 \quad \{X=0, Y=0\}$$

Figure 2-74 represents the MULT_6X6S_5X5U submodule.



UG002_C2_024_081800

Figure 2-74: **MULT_6X6S_5X5U -- Connections to a MULT18X18 Primitive**

Table 2-28 shows values for X and Y where these conditions are met.

Table 2-28: **Two Multipliers per MULT18X18 Allowable Sizes**

X * X		Y * Y	
Signed Size	**Unsigned Size**	**Signed Size**	**Unsigned Size**
7 X 7	6 X 6	-	4 X 4
6 X 6	5 X 5	-	5 X 5
5 X 5	4 X 4	3 X 3	6 X 6
4 X 4	3 X 3	3 X 3	7 X 7
3 X 3	2 X 2	4 X 4	8 X 8

VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available as examples of primitives and submodules (see "VHDL and Verilog Templates" on page 256).

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signals names.

Port Signals

Data In - A

The data input provides new data (up to 18 bits) to be used as one of the multiplication operands.

Data In - B

The data input provides new data (up to 18 bits) to be used as one of the multiplication operands.

Data Out - P

The data output bus P provides the data value (up to 36 bits) of two's-complement multiplication for operands A and B.

Location Constraints

Each embedded multiplier has location coordinates of the form XrowYcolumn. To constrain placement, multiplier instances can have LOC properties attached to

MULT18X18 embedded multiplier instances can have LOC properties attached to them to constrain placement. MULT18X18 placement locations differ from the convention used for naming CLB locations, allowing LOC properties to transfer easily from array to array.

The LOC properties use the following form:

```
LOC = MULT18X18_X#Y#
```

For example, MULT18X18_X0Y0 is the bottom-left MULT18X18 location on the device.

VHDL and Verilog Templates

VHDL and Verilog templates are available for the primitive and submodules.

The following is a template for the primitive:

- SIGNED_MULT_18X18 (primitive: MULT18X18)

The following are templates for submodules:

- SIGNED_MULT_35X35 (submodule: MULT35X35_S)
- UNSIGNED_MULT_34X34 (submodule: MULT34X34_U)
- UNSIGNED_MULT_17X17 (submodule: MULT17X17_U)
- SIGNED_MULT_8X8 (submodule: MULT8X8_S)
- UNSIGNED_MULT_8X8 (submodule: MULT8X8_U)
- SIGNED_MULT_4X4 (submodule: MULT4X4_S)
- UNSIGNED_MULT_4X4 (submodule: MULT4X4_U)
- DUAL_MULT_6X6S_5X5U (submodule: MULT_6X6S_5X5U)
- DUAL_MULT_5X5S_6X6U (submodule: MULT_5X5S_6X6U)
- DUAL_MULT_5X5U_5X5U (submodule: MULT_5X5U_5X5U)
- DUAL_MULT_4X4S_7X7U (submodule: MULT_4X4S_7X7U)
- DUAL_MULT_4X4S_3X3S (submodule: MULT_4X4S_3X3S)
- TWOS_COMPLEMENTER_18BIT (submodule: TWOS_CMP18)
- TWOS_COMPLEMENTER_9BIT (submodule: TWOS_CMP9)
- MAGNITUDE_18BIT (submodule: MAGNTD_18)

The corresponding submodules have to be synthesized with the design.

Templates for the SIGNED_MULT_18X18 module are provided in VHDL and Verilog code as an example.

VHDL Template:

```
-- Module: SIGNED_MULT_18X18
-- Description: VHDL instantiation template
-- 18-bit X 18-bit embedded signed multiplier (asynchronous)
--
-- Device: Virtex-II Family
---------------------------------------------------------------------
-- Components Declarations
component MULT18X18
  port(
       A :  in std_logic_vector (17 downto 0);
       B :  in std_logic_vector (17 downto 0);
       P : out std_logic_vector (35 downto 0)
  );
end component;
--
-- Architecture Section
--
U_MULT18X18  : MULT18X18
  port map (
    A => , -- insert input signal #1
    B => , -- insert input signal #2
    P =>   -- insert output signal
  );
```

Verilog Template:

```
// Module: SIGNED_MULT_18X18
// Description: Verilog instantiation template
// 18-bit X 18-bit embedded signed multiplier (asynchronous)
//
// Device: Virtex-II Family
//-------------------------------------------------------------------
// Instantiation Section
//
MULT18X18 U_MULT18X18
   (
    .A () , // insert input signal #1
    .B () , // insert input signal #2
    .P ()   // insert output signal
  );
```

Using Single-Ended SelectI/O Resources

Summary

The Virtex-II FPGA series includes a highly configurable, high-performance single-ended SelectI/O resource that supports a wide variety of I/O standards. The SelectI/O resource includes a robust set of features, including programmable control of output drive strength, slew rate, and input delay and hold time. Taking advantage of the flexibility of SelectI/O features and the design considerations described in this document can improve and simplify system-level design.

Introduction

As FPGAs continue to grow in size and capacity, the larger and more complex systems designed for them demand an increased variety of I/O standards. Furthermore, as system clock speeds continue to increase, the need for high-performance I/O becomes more important. Chip-to-chip delays have an increasingly substantial impact on overall system speed. The task of achieving the desired system performance is becoming more difficult with the proliferation of low-voltage I/O standards. SelectI/O resolves this potential problem by providing a highly configurable, high-performance alternative to I/O resources used in more conventional programmable devices.

Virtex-II SelectI/O blocks can support up to 19 single-ended I/O standards. Supporting such a variety of I/O standards allows support for a wide variety of applications.

Each Input/Output Block (IOB) includes six registers, two each from the input, output, and 3-state signals within the IOB. These registers are optionally configured as either a D-type flip-flop or as a level-sensitive latch. The purpose of having six registers is to allow designers to design double-data-rate (DDR) logic in the I/O blocks. Each pair of the flip-flop (FF) has different clocks so that the flip-flops can be driven by two clocks with a 180-degree phase shift to achieve DDR. All I/O flip-flops still share the same reset/preset line.

The input buffer has an optional delay element used to guarantee a zero hold time requirement for input signals registered within the IOB.

Virtex-II SelectI/O features also provide dedicated resources for input reference voltage (V_{REF}) and input output source voltage (V_{CCO}), along with a convenient banking system that simplifies board design. Virtex-II inputs and outputs are powered from V_{CCO}. Differential amplifier inputs, such as GTL and SSTL, are powered from V_{REF}.

Fundamentals

Modern bus applications, pioneered by the largest and most influential components in the digital electronics industry, are commonly introduced with a new I/O standard tailored specifically to the needs of that application. The bus I/O standards provide specifications to other vendors who create products designed to interface with these applications. Each standard often has its own specifications for current, voltage, I/O buffering, and termination techniques.

The ability to provide the flexibility and time-to-market advantages of programmable logic is increasingly dependent on the capability of the programmable logic device to support an ever increasing variety of I/O standards.

SelectI/O resources feature highly configurable input and output buffers that provide support for a wide variety of I/O standards. An input buffer can be configured as either a simple buffer or as a differential amplifier input. An output buffer can be configured as either a Push-Pull output or as an Open Drain output. Table 2-29 illustrates all of the supported single-ended I/O standards in Virtex-II devices. Each buffer type can support a variety of current and voltage requirements.

Table 2-29: **Supported Single-Ended I/O Standards**

I/O Standard	Input Reference Voltage (V_{REF})	Input Source Voltage (V_{CCO})	Output Source Voltage (V_{CCO})	Board Termination Voltage (V_{TT})
LVTTL	N/A	3.3	3.3	N/A
LVCMOS15	N/A	1.5	1.5	N/A
LVCMOS18	N/A	1.8	1.8	N/A
LVCMOS25	N/A	2.5	2.5	N/A
LVCMOS33	N/A	3.3	3.3	N/A
PCI33_3	N/A	3.3	3.3	N/A
PCI66_3	N/A	3.3	3.3	N/A
PCIX	N/A	3.3	3.3	N/A
GTL	0.80	N/A	N/A	1.2
GTL+	1.0	N/A	N/A	1.5
HSTL_I	0.75	N/A	1.5	0.75
HSTL_II	0.75	N/A	1.5	0.75
HSTL_III	0.9	N/A	1.5	1.5
HSTL_IV	0.9	N/A	1.5	1.5
SSTL3_I	1.5	N/A	3.3	1.5
SSTL3_II	1.5	N/A	3.3	1.5
SSTL2_I	1.25	N/A	2.5	1.25
SSTL2_II	1.25	N/A	2.5	1.25
AGP-2X	1.32	N/A	3.3	N/A

Overview of Supported I/O Standards

This section provides a brief overview of I/O standards supported by all Virtex-II devices.

While most I/O standards specify a range of allowed voltages, this document records typical voltage values only. Detailed information on each specification can be found on the Electronic Industry Alliance JEDEC website at:

http://www.jedec.org

LVTTL - Low-Voltage TTL

The low-voltage TTL, or LVTTL, standard is a general purpose EIA/JESDSA standard for 3.3 V applications that use an LVTTL input buffer and a Push-Pull output buffer. This standard requires a 3.3 V input and output source voltage (V_{CCO}), but does not require the use of a reference voltage (V_{REF}) or a termination voltage (V_{TT}).

LVCMOS33 - 3.3 Volt Low-Voltage CMOS

This standard is an extension of the LVCMOS standard (JESD 8.-5). It is used in general purpose 3.3 V applications.The standard requires a 3.3 V input/output source voltage (V_{CCO}), but does not require the use of a reference voltage (V_{REF}) or a termination voltage (V_{TT}).

LVCMOS25 - 2.5 Volt Low-Voltage CMOS

This standard is an extension of the LVCMOS standard (JESD 8.-5). It is used in general purpose 2.5 volts or lower applications. This standard requires a 2.5 V input /output source voltage (V_{CCO}), but does not require the use of a reference voltage (V_{REF}) or a board termination voltage (V_{TT}).

LVCMOS18 - 1.8 Volt Low-Voltage CMOS

This standard is an extension of the LVCMOS standard. It is used in general purpose 1.8 V applications. The use of a reference voltage (V_{REF}) or board termination voltage (V_{TT}) is not required.

LVCMOS15 - 1.5 Volt Low-Voltage CMOS

This standard is an extension of the LVCMOS standard. It is used in general purpose 1.5 V applications. The use of a reference voltage (V_{REF}) or a board termination voltage (V_{TT}) is not required.

PCI - Peripheral Component Interface

The PCI standard specifies support for 33 MHz, 66 MHz and 133 MHz PCI bus applications. It uses a LVTTL input buffer and a Push-Pull output buffer. This standard does not require the use of a reference voltage (V_{REF}) or a board termination voltage (V_{TT}), however, it does require 3.3 V input output source voltage (V_{CCO}).

GTL -Gunning Transceiver Logic Terminated

The GTL standard is a high-speed bus standard (JESD8.3) invented by Xerox. Xilinx has implemented the terminated variation for this standard. This standard requires a differential amplifier input buffer and a open Drain output buffer.

GTL+ - Gunning Transceiver Logic Plus

The Gunning Transceiver Logic Plus, or GTL+ standard is a high-speed bus standard (JESD8.3) first used by the Pentium Pro Processor.

HSTL - High-speed Transceiver Logic

The high-speed Transceiver Logic, or HSTL standard is a general purpose high-speed, 1.5V bus standard sponsored by IBM (EIA/JESD8-6). This standard has four variations or classes. Virtex-II SelectI/O supports all four Classes. This standard requires a Differential Amplifier input buffer and a Push-pull output buffer.

SSTL3 - Stub Series Terminated Logic for 3.3V

The Stub Series Terminated Logic for 3.3V, or SSTL3 standard is a general purpose 3.3V memory bus standard also sponsored by Hitachi and IBM (JESD8-8). This standard has two classes, I and II. Virtex-II SelectI/O supports both classes for the SSTL3 standard. This standard requires a Differential Amplifier input buffer and a Push-Pull output buffer.

SSTL2 - Stub Series Terminated Logic for 2.5V

The Stub Series Terminated Logic for 2.5V, or SSTL2 standard is a general purpose 2.5V memory bus standard also sponsored by Hitachi and IBM (JESD8-8). This standard has two classes, I and II. Virtex-II SelectI/O supports both classes for the SSTL2 standard. This standard requires a Differential Amplifier input buffer and a Push-Pull output buffer

AGP-2X - Advanced Graphics Port

The Intel AGP standard is a 3.3V Advanced Graphics Port-2X bus standard used with the Pentium II processor for graphic applications. This standard requires a Push-Pull output buffer and a Differential Amplifier input buffer.

Library Symbols

The Xilinx library includes an extensive list of symbols designed to provide support for the variety of SelectI/O features. Most of these symbols represent variations of the five generic SelectI/O symbols.

- IBUF (input buffer)
- IBUFG (clock input buffer)
- OBUF (output buffer)
- OBUFT (3-state output buffer)
- IOBUF (input/output buffer)

IBUF

Signals used as inputs to a Virtex-II device must source an input buffer (IBUF) via an external input port. The generic Virtex-II IBUF symbol is shown in Figure 2-75. The extension to the base name defines which I/O standard the IBUF uses. The assumed standard is LVTTL when the generic IBUF has no specified extension.

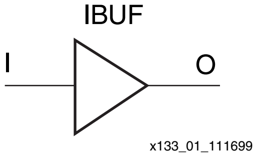


Figure 2-75: **Input Buffer (IBUF) Symbols**

Table 2-30 details variations of the IBUF symbol for single-ended Virtex-II I/O standards:

Table 2-30: **Variations of the IBUF Symbol**

IBUF	IBUF_HSTL_III
IBUF_LVCMOS15	IBUF_HSTL_IV
IBUF_LVCMOS18	IBUF_SSTL2_I
IBUF_LVCMOS25	IBUF_SSTL2_II
IBUF_LVCMOS33	IBUF_SSTL3_I
IBUF_APG	IBUF_SSTL3_II
IBUF_GTL	IBUF_PCI33_3
IBUF_GTLP	IBUF_PCI66_3
IBUF_HSTL_I	IBUF_PCIX
IBUF_HSTL_II	IBUF_AGP

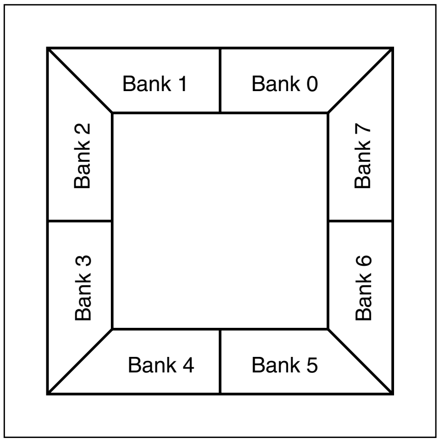
When the IBUF symbol supports an I/O standard that requires a differential amplifier input, the IBUF is automatically configured as a differential amplifier input buffer. The low-voltage I/O standards with a differential amplifier input require an external reference voltage input V_{REF}.

The voltage reference signal is "banked" within the Virtex-II device on a half-edge basis, such that for all packages there are eight independent V_{REF} banks internally. For a representation of the Virtex-II I/O banks, see Figure 2-77. Within each bank approximately one of every six I/O pins is automatically configured as a V_{REF} input. After placing a differential amplifier input signal within a given V_{REF} bank, the same external source must drive all I/O pins configured as a V_{REF} input.

IBUF placement restrictions require that any differential amplifier input signals within a bank be of the same standard. How to specify a specific location for the IBUF via the LOC property is described below. Table 2-31 summarizes compatibility requirements of Virtex-II input standards.

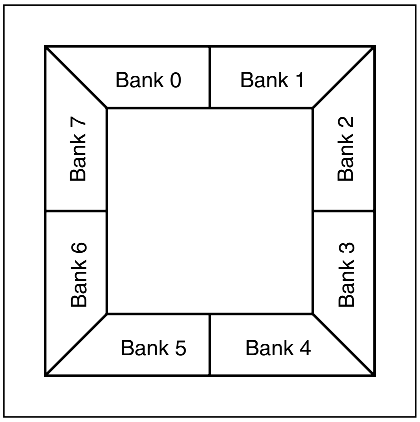
An optional delay element in the input data path is associated with each IBUF. When the IBUF drives a flip-flop within the IOB, the delay element is activated by default to ensure a zero hold-time requirement at the device input pin. The IOBDELAY = NONE property overrides this default, thus reducing the input set-up time, but risking a hold-time requirement.

When the IBUF does not drive a flip-flop within the IOB, the delay element is deactivated by default to provide a shorter input set-up time. To delay the input signal, activate the delay element with the IOBDELAY = BOTH property.



ds031_66_112900

Figure 2-76: **Virtex-II I/O Banks: Top View for Flip-Chip Packages (FF & BF)**



ug002_c2_014_112900

Figure 2-77: **Virtex-II I/O Banks: Top View for Wire-Bond Packages (CS, FG, & BG)**

Table 2-31: **Xilinx Input Standard Compatibility Requirements**

Rule 1	Standards with the same V_{CCO}, and V_{REF} can be placed within the same bank.
Rule 2	Standards that don't require a V_{REF} can be placed within the same bank with the standards that have the same V_{CCO} values

Each bank has its own V_{CCO} and V_{REF} voltage. Details on compatible input standards for each V_{CCO} / V_{REF} voltage combination are available in the **Virtex-II Data Sheet** .

OBUF

An OBUF must drive outputs through an external output port. Figure 2-78 shows the generic output buffer (OBUF) symbol.

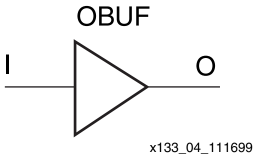


Figure 2-78: **Virtex-II Output Buffer (OBUF) Symbol**

The extension to the base name defines which I/O standard the OBUF uses. With no extension specified for the generic OBUF symbol, the assumed standard is slew rate limited LVTTL with 12mA drive strength.

The LVTTL and LVCMOS OBUFs can additionally support one of two slew rate modes to minimize bus transients. By default, the slew rate for each output buffer is reduced to minimize power bus transients, when switching non-critical signals.

LVTTL and LVCMOS output buffers have selectable drive strengths. The format for these OBUF symbol names is as follows:

OBUF_<slew_rate>_<drive_strength>

<slew_rate> is either F (fast) or S (slow) and <drive_strength> is specified in milliamperes. For LVTTL, LVCMOS25, and LVCMOS33, the supported drive strengths are 2, 4, 6, 8, 12, 16, and 24. For LVCMOS15, and LVCMOS18, the supported drive strengths are 2, 4, 6, 8, 12, and 16.

Table 2-32 details variations of the OBUF symbol.

Table 2-32: **Variations of the OBUF Symbol**

OBUF	OBUF_LVCMOS18_S_2	OBUF_LVCMOS33_S_4
OBUF_S_2	OBUF_LVCMOS18_S_4	OBUF_LVCMOS33_S_6
OBUF_S_4	OBUF_LVCMOS18_S_6	OBUF_LVCMOS33_S_8
OBUF_S_6	OBUF_LVCMOS18_S_8	OBUF_LVCMOS33_S_12
OBUF_S_8	OBUF_LVCMOS18_S_12	OBUF_LVCMOS33_S_16
OBUF_S_12	OBUF_LVCMOS18_S_16	OBUF_LVCMOS33_S_24
OBUF_S_16	OBUF_LVCMOS18_F_2	OBUF_LVCMOS33_F_2
OBUF_S_24	OBUF_LVCMOS18_F_4	OBUF_LVCMOS33_F_4
OBUF_F_2	OBUF_LVCMOS18_F_6	OBUF_LVCMOS33_F_6
OBUF_F_4	OBUF_LVCMOS18_F_8	OBUF_LVCMOS33_F_8
OBUF_F_6	OBUF_LVCMOS18_F_12	OBUF_LVCMOS33_F_12
OBUF_F_8	OBUF_LVCMOS18_F_16	OBUF_LVCMOS33_F_16
OBUF_F_12	OBUF_LVCMOS25	OBUF_LVCMOS33_F_24
OBUF_F_16	OBUF_LVCMOS25_S_2	OBUF_PCI33_3
OBUF_F_24	OBUF_LVCMOS25_S_4	OBUF_PCI66-3
OBUF_LVCMOS15	OBUF_LVCMOS25_S_6	OBUF_PCIX
OBUF_LVCMOS15_S_2	OBUF_LVCMOS25_S_8	OBUF_GTL
OBUF_LVCMOS15_S_4	OBUF_LVCMOS25_S_12	OBUF_GTLP
OBUF_LVCMOS15_S_6	OBUF_LVCMOS25_S_16	OBUF_HSTL_I

Table 2-32: **Variations of the OBUF Symbol** *(Continued)*

OBUF_LVCMOS15_S_8	OBUF_LVCMOS25_S_24	OBUF_HSTL_II
OBUF_LVCMOS15_S_12	OBUF_LVCMOS25_F_2	OBUF_HSTL_III
OBUF_LVCMOS15_S_16	OBUF_LVCMOS25_F_4	OBUF_HSTL_IV
OBUF_LVCMOS15_F_2	OBUF_LVCMOS25_F_6	OBUF_SSTL3_I
OBUF_LVCMOS15_F_4	OBUF_LVCMOS25_F_8	OBUF_SSTL3_II
OBUF_LVCMOS15_F_6	OBUF_LVCMOS25_F_12	OBUF_SSTL2_I
OBUF_LVCMOS15_F_8	OBUF_LVCMOS25_F_16	OBUF_SSTL2_II
OBUF_LVCMOS15_F_12	OBUF_LVCMOS25_F_24	OBUF_AGP
OBUF_LVCMOS15_F_16	OBUF_LVCMOS33	
OBUF_LVCMOS18	OBUF_LVCMOS33_S_2	

OBUF placement restrictions require that within a given V_{CCO} bank each OBUF share the same output source drive voltage. Input buffers with the same V_{CCO} and output buffers that do not require V_{CCO} can be placed within any V_{CCO} bank. Table 2-33 summarizes Virtex-II output compatibility requirements. The LOC property can specify a location for the OBUF.

Table 2-33: **Output Standards Compatibility Requirements**

Rule 1	Only outputs with standards which share compatible V_{CCO} can be used within the same bank.
Rule 2	There are no placement restrictions for outputs with standards that do not require a V_{CCO}

Each bank has its own V_{CCO} voltage. Details on compatible output standards for each V_{CCO} voltage combination are available in the **Virtex-II Data Sheet** .

OBUFT

The generic 3-state output buffer OBUFT, shown in Figure 2-79, typically implements 3-state outputs or bidirectional I/O.

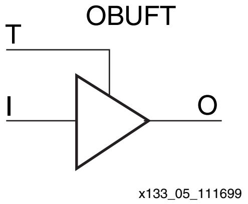


Figure 2-79: **3-State Output Buffer Symbol (OBUFT)**

The extension to the base name defines which I/O standard OBUFT uses. With no extension specified for the generic OBUFT symbol, the assumed standard is slew rate limited LVTTL with 12mA drive strength.

The LVTTL and LVCMOS OBUFTs additionally can support one of two slew rate modes to minimize bus transients. By default, the slew rate for each output buffer is reduced to minimize power bus transients, when switching non-critical signals.

LVTTL and LVCMOS 3-state buffers have selectable drive strengths. The format for these OBUFT symbol names is as follows:

OBUFT_<slew_rate>_<drive_strength>

<slew_rate> is either F(fast) or S(slow) and <drive_strength> is specified in milliamperes. For LVTTL, LVCMOS25, and LVCMOS33, the supported drive strengths are 2, 4, 6, 8, 12, 16, and 24. For LVCMOS15 and LVCMOS18, the supported drive strengths are 2, 4, 6, 8, 12, and 16.

Table 2-34 details variations of the OBUFT symbol.

Table 2-34: **Variations of the OBUFT Symbol**

OBUFT	OBUFT_LVCMOS18_S_2	OBUFT_LVCMOS33_S_4
OBUFT_S_2	OBUFT_LVCMOS18_S_4	OBUFT_LVCMOS33_S_6
OBUFT_S_4	OBUFT_LVCMOS18_S_6	OBUFT_LVCMOS33_S_8
OBUFT_S_6	OBUFT_LVCMOS18_S_8	OBUFT_LVCMOS33_S_12
OBUFT_S_8	OBUFT_LVCMOS18_S_12	OBUFT_LVCMOS33_S_16
OBUFT_S_12	OBUFT_LVCMOS18_S_16	OBUFT_LVCMOS33_S_24
OBUFT_S_16	OBUFT_LVCMOS18_F_2	OBUFT_LVCMOS33_F_2
OBUFT_S_24	OBUFT_LVCMOS18_F_4	OBUFT_LVCMOS33_F_4
OBUFT_F_2	OBUFT_LVCMOS18_F_6	OBUFT_LVCMOS33_F_6
OBUFT_F_4	OBUFT_LVCMOS18_F_8	OBUFT_LVCMOS33_F_8
OBUFT_F_6	OBUFT_LVCMOS18F_12	OBUFT_LVCMOS33_F_12
OBUFT_F_8	OBUFT_LVCMOS18_F_16	OBUFT_LVCMOS33_F_16
OBUFT_F_12	OBUFT_LVCMOS25	OBUFT_LVCMOS33_F_24
OBUFT_F_16	OBUFT_LVCMOS25_S_2	OBUFT_PCI33_3
OBUFT_F_24	OBUFT_LVCMOS25_S_4	OBUFT_PCI66-3
OBUFT_LVCMOS15	OBUFT_LVCMOS25_S_6	OBUFT_PCIX
OBUFT_LVCMOS15_S_2	OBUFT_LVCMOS25_S_8	OBUFT_GTL
OBUFT_LVCMOS15_S_4	OBUFT_LVCMOS25_S_12	OBUFT_GTLP
OBUFT_LVCMOS15_S_6	OBUFT_LVCMOS25_S_16	OBUFT_HSTL_I
OBUFT_LVCMOS15_S_8	OBUFT_LVCMOS25_S_24	OBUFT_HSTL_II
OBUFT_LVCMOS15_S_12	OBUFT_LVCMOS25_F_2	OBUFT_HSTL_III
OBUFT_LVCMOS15_S_16	OBUFT_LVCMOS25_F_4	OBUFT_HSTL_IV
OBUFT_LVCMOS15_F_2	OBUFT_LVCMOS25_F_6	OBUFT_SSTL3_I
OBUFT_LVCMOS15_F_4	OBUFT_LVCMOS25_F_8	OBUFT_SSTL3_II
OBUFT_LVCMOS15_F_6	OBUFT_LVCMOS25_F_12	OBUFT_SSTL2_I
OBUFT_LVCMOS15_F_8	OBUFT_LVCMOS25_F_16	OBUFT_SSTL2_II
OBUFT_LVCMOS15_F_12	OBUFT_LVCMOS25_F_24	OBUFT_AGP
OBUFT_LVCMOS15_F_16	OBUFT_LVCMOS33	
OBUFT_LVCMOS18	OBUFT_LVCMOS33_S_2	

OBUFT placement restrictions require that within a given V_{CCO} bank each OBUFT share the same output source drive voltage. Input buffers with the same V_{CCO} and output buffers that do not require V_{CCO} can be placed within any V_{CCO} bank. The LOC property can specify a location for the OBUFT.

3-state output buffers and bidirectional buffers can have either a weak pull-up resistor, a weak pull-down resistor, or a weak "keeper" circuit. Control this feature by adding the appropriate symbol to the output net of the OBUFT (PULLUP, PULLDOWN, or KEEPER).

The weak "keeper" circuit requires the input buffer within the IOB to sample the I/O signal. Thus, OBUFTs programmed for an I/O standard that requires a V_{REF} have automatic placement of a V_{REF} in the bank with an OBUFT configured with a weak "keeper" typically implement a bidirectional I/O. In this case, the IBUF (and the corresponding V_{REF}) are placed explicitly.

IOBUF

Use the IOBUF symbol for bidirectional signals that require both an input buffer and a 3-state output buffer with an active High 3-state pin. Figure 2-80 shows the generic input/output IOBUF buffer.

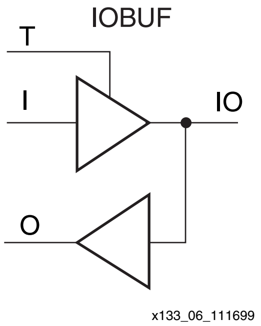


Figure 2-80: **Input/Output Buffer Symbol (IOBUF)**

The extension to the base name defines which I/O standard the IOBUF uses. With no extension specified for the generic IOBUF symbol, the assumed standard is LVTTL input buffer and slew rate limited LVTTL with 12mA drive strength for the output buffer.

The LVTTL and LVCMOS IOBUFs can additionally support one of two slew rate modes to minimize bus transients. By default, the slew rate for each output buffer is reduced to minimize power bus transients, when switching non-critical signals.

LVTTL and LVCMOS output buffers have selectable drive strengths. The format for these OBUF symbol names is as follows:

OBUF_<slew_rate>_<drive_strength>

<slew_rate> is either F (fast) or S (slow) and <drive_strength> is specified in milliamperes. For LVTTL, LVCMOS25 and LVCMOS33, the supported drive strengths are 2, 4, 6, 8, 12, 16, and 24. For LVCMOS15, and LVCMOS18, the supported drive strengths are 2, 4, 6, 8, 12, and 16. Table 2-35 details variations of the IOBUF symbol.

Table 2-35: **Variations of the IOBUF Symbol**

IOBUF	IOBUF_LVCMOS18_S_2	IOBUF_LVCMOS33_S_4
IOBUF_S_2	IOBUF_LVCMOS18_S_4	IOBUF_LVCMOS33_S_6
IOBUF_S_4	IOBUF_LVCMOS18_S_6	IOBUF_LVCMOS33_S_8
IOBUF_S_6	IOBUF_LVCMOS18_S_8	IOBUF_LVCMOS33_S_12
IOBUF_S_8	IOBUF_LVCMOS18_S_12	IOBUF_LVCMOS33_S_16
IOBUF_S_12	IOBUF_LVCMOS18_S_16	IOBUF_LVCMOS33_S_24
IOBUF_S_16	IOBUF_LVCMOS18_F_2	IOBUF_LVCMOS33_F_2
IOBUF_S_24	IOBUF_LVCMOS18_F_4	IOBUF_LVCMOS33_F_4
IOBUF_F_2	IOBUF_LVCMOS18_F_6	IOBUF_LVCMOS33_F_6
IOBUF_F_4	IOBUF_LVCMOS18_F_8	IOBUF_LVCMOS33_F_8
IOBUF_F_6	IOBUF_LVCMOS18F_12	IOBUF_LVCMOS33_F_12
IOBUF_F_8	IOBUF_LVCMOS18_F_16	IOBUF_LVCMOS33_F_16

Table 2-35: **Variations of the IOBUF Symbol** ***(Continued)***

IOBUF_F_12	IOBUF_LVCMOS25	IOBUF_LVCMOS33_F_24
IOBUF_F_16	IOBUF_LVCMOS25_S_2	IOBUF_PCI33_3
IOBUF_F_24	IOBUF_LVCMOS25_S_4	IOBUF_PCI66-3
IOBUF_LVCMOS15	IOBUF_LVCMOS25_S_6	IOBUF_PCIX
IOBUF_LVCMOS15_S_2	IOBUF_LVCMOS25_S_8	IOBUF_GTL
IOBUF_LVCMOS15_S_4	IOBUF_LVCMOS25_S_12	IOBUF_GTLP
IOBUF_LVCMOS15_S_6	IOBUF_LVCMOS25_S_16	IOBUF_HSTL_I
IOBUF_LVCMOS15_S_8	IOBUF_LVCMOS25_S_24	IOBUF_HSTL_II
IOBUF_LVCMOS15_S_12	IOBUF_LVCMOS25_F_2	IOBUF_HSTL_III
IOBUF_LVCMOS15_S_16	IOBUF_LVCMOS25_F_4	IOBUF_HSTL_IV
IOBUF_LVCMOS15_F_2	IOBUF_LVCMOS25_F_6	IOBUF_SSTL3_I
IOBUF_LVCMOS15_F_4	IOBUF_LVCMOS25_F_8	IOBUF_SSTL3_II
IOBUF_LVCMOS15_F_6	IOBUF_LVCMOS25_F_12	IOBUF_SSTL2_I
IOBUF_LVCMOS15_F_8	IOBUF_LVCMOS25_F_16	IOBUF_SSTL2_II
IOBUF_LVCMOS15_F_12	IOBUF_LVCMOS25_F_24	IOBUF_AGP
IOBUF_LVCMOS15_F_16	IOBUF_LVCMOS33	
IOBUF_LVCMOS18	IOBUF_LVCMOS33_S_2	

When the IOBUF symbol supports an I/O standard that requires a differential amplifier input, IOBUF is automatically configured as a differential amplifier input buffer. Low-voltage I/O standards with a differential amplifier input require an external reference voltage input V_{REF}.

The voltage reference signal is "banked" within the Virtex-II device on a half-edge basis, such that for all packages there are eight independent V_{REF} banks internally. For a representation of the Virtex-II I/O banks, see Figure 2-77. Within each bank approximately one of every twelve I/O pins is automatically configured as a V_{REF} input. After placing a differential amplifier input signal within a given V_{REF} bank, the same external source must drive all I/O pins configured as a V_{REF} input.

IOBUF placement restrictions require any differential amplifier input signals within a bank be of the same standard.

Additional restrictions on Virtex-II SelectI/O IOBUF placement require that within a given V_{CCO} bank each IOBUF share the same output source drive voltage. Input buffers with the same V_{CCO} and output buffers that do not require V_{CCO} can be placed within any V_{CCO} bank. The LOC property can specify a location for the OBUF.

An optional delay element is associated with the input path in each IOBUF. When the IOBUF drives an input flip-flop within the IOB, the delay element is activated by default to ensure the zero hold-time requirement. Override this default with the IOBDELAY = NONE property.

In the case when the IOBUF does not drive an input flip-flop within the IOB, the delay element is deactivated by default to provide higher performance. To delay the input signal, deactivate the delay element with the IOBDELAY = BOTH property.

3-state output buffers and bidirectional buffers can have a weak pull-up resistor, a weak pull-down resistor, or a weak "keeper" circuit. Control this feature by adding the appropriate symbol to the output net of the IOBUF (PULLUP, PULLDOWN, or KEEPER).

SelectI/O Properties

Access to some SelectI/O features (for example, location constraints, input delay, output drive strength, and slew rate) is available through properties associated with these features.

Input Delay Properties

An optional delay element is associated with the input path in each IBUF. When the IBUF drives an input flip-flop within the IOB, the delay element activates by default to ensure the zero hold-time requirement. Override this default with the IOBDELAY = NONE property.

In the case when the IBUF does not drive an input flip-flop within the IOB, the delay element is deactivated by default to provide higher performance. To delay the input signal, activate the delay element with the IOBDELAY = BOTH property.

IOB Flip-Flop/Latch Properties

The Virtex-II series I/O block (IOB) includes two optional registers on the input path, two optional registers on the output path, and two optional registers on the 3-state control pin. The design implementation software automatically takes advantage of these registers when the following option for the MAP program is specified.

```
Map -pr b <filename>
```

Alternatively, the IOB = TRUE property can be placed on a register to force the mapper to place the register in an IOB.

The two registers for each path makes designing double-data-rate (DDR) logic much simpler. Each pair of the registers has separate clock inputs, which can be driven by either the positive edge or the negative edge of the clock. Users can use both edges of the clocks to clock data in and out from the IOB. For details on DDR, see "Using Double-Data-Rate (DDR) I/O" on page 303.

Location Constraints

Specify the location of each SelectI/O symbol with the location constraint LOC attached to the SelectI/O symbol. The external port identifier indicates the value of the location constrain. The format of the port identifier depends on the package chosen for the specified design.

The LOC properties use the following form:

- LOC=A42;
- LOC=P37;

Output Slew Rate Property

As mentioned above, a variety of symbol names provide the option of choosing the desired slew rate for the output buffers. In the case of the LVTTL or LVCMOS output buffers (OBUF, OBUFT, and IOBUF), slew rate control can be alternatively programmed with the SLEW = property. By the default, the slew rate for each output buffer is reduced to minimize power bus transients when switching non-critical signals. The SLEW = property has one of the two following values:

- SLEW = SLOW
- SLEW = FAST

Output Drive Strength Property

The desired output drive strength can be additionally specified by choosing the appropriate library symbol. The Xilinx library also provides an alternative method for specifying this feature. For the LVTTL, and LVCMOS output buffers (OBUF, OBUFT, and

IOBUF), the desired drive strength can be specified with the DRIVE = property. This property could have one of the following values:

- DRIVE = 2
- DRIVE = 4
- DRIVE = 6
- DRIVE = 8
- DRIVE = 12
- DRIVE = 16
- DRIVE = 24

Design Considerations

Reference Voltage (V_{REF}) Pins

Low-voltage I/O standards with a differential amplifier input buffer require an input reference voltage (V_{REF}). Provide the V_{REF} as an external signal to the device.

The voltage reference signal is "banked" within the Virtex-II device on a half-edge basis such that for all packages there are eight independent V_{REF} banks internally. See Figure 2-77 for a representation of the Virtex-II I/O banks. Within each bank approximately one of every twelve I/O pins is automatically configured as a V_{REF} input. After placing a differential amplifier input signal within a given V_{REF} bank, the same external source must drive all I/O pins configured as a V_{REF} input.

Within each V_{REF} bank, any input buffers that require a V_{REF} signal must be of the same type. Output buffers that have the same V_{CCO} values as the input buffers can be placed within the same V_{REF} bank.

Output Drive Source Voltage (V_{CCO}) Pins

Many of the low-voltage I/O standards supported by SelectI/O devices require a different output drive source voltage (V_{CCO}). As a result each device can often have to support multiple output drive source voltages.

Output buffers within a given V_{CCO} bank must share the same output drive source voltage. Input buffers for LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, LVCMOS33, PCI33_3, PCI66_3, PCIX use the V_{CCO} voltage for input V_{CCO} voltage.

Transmission Line Effects

The delay of an electrical signal along a wire is dominated by the rise and fall times when the signal travels a short distance. Transmission line delays vary with inductance and capacitance. But a well-designed board can experience delays of approximately 180ps per inch. Transmission line effects, or reflections, typically start at 1.5" for fast (1.5ns) rise and fall times. Poor (or non-existent) termination or changes in the transmission line impedance cause these reflections and can cause additional delay in longer traces. As a system speeds continue to increase, the effect of I/O delays can become a limiting factor and therefore transmission line termination becomes increasingly more important.

Termination Techniques

A variety of termination techniques reduce the impact of transmission line effects.

The following are output termination techniques:

- None
- Series
- Parallel (Shunt)
- Series and Parallel (Series-Shunt)

The following are input termination techniques:

- None
- Parallel (Shunt)

These termination techniques can be applied in any combination. A generic example of each combination of termination methods appears in Figure 2-81.

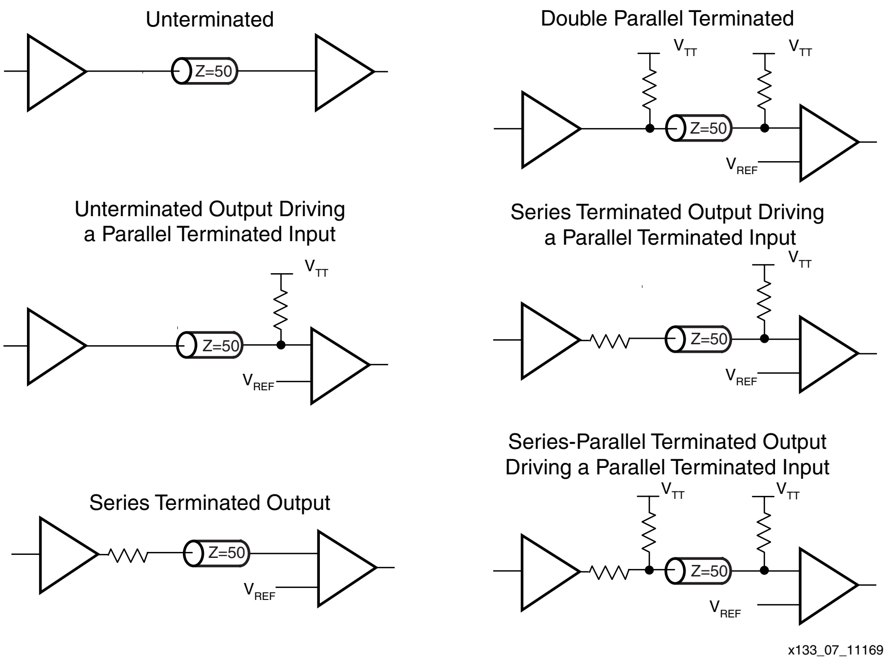


Figure 2-81: **Overview of Standard Input and Output Termination Methods**

Simultaneous Switching Guidelines

Ground bounce can occur with high-speed digital I_{CS} when multiple outputs change states simultaneously, causing undesired transient behavior on an output or in the internal logic. This problem is also referred to as the Simultaneous Switching Output (SSO) problem.

Ground bounce is primarily due to current changes in the combined inductance of ground pins, bond wires, and group metallization. The IC internal ground level deviates from the external system ground level for a short duration (a few nanoseconds) after multiple outputs change state simultaneously.

Ground bounce affects stable low outputs and all inputs because they interpret the incoming signal by comparing it to the internal ground. If the ground bounce amplitude exceeds the actual instantaneous noise margin, then a non-changing input can be interpreted as a short pulse with a polarity opposite to the ground bounce. Table 2-36 provides the guidelines for the maximum number of simultaneously switching outputs allowed per output power/ground pair to avoid the effects of ground bounce. Refer to Table 2-37 for the number of effective output power/ground pairs for each Virtex-II device and package combination.

Table 2-36: **Guidelines for Max Number of Simultaneously Switching Outputs per Power/Ground Pair**

Standard	Package			
	FG,BG,FF,BF	CS	XC2V40-FG	XC2V40-CS
LVTTL2_slow	68	51	51	34
LVTTL4_slow	41	31	31	21
LVTTL6_slow	29	22	22	15
LVTTL8_slow	22	17	17	11
LVTTL12_slow	15	11	11	8
LVTTL16_slow	11	8	8	6
LVTTL24_slow	7	5	5	4
LVTTL2_fast	40	30	30	20
LVTTL4_fast	24	18	18	12
LVTTL6_fast	17	13	13	9
LVTTL8_fast	13	10	10	7
LVTTL12_fast	10	8	8	5
LVTTL16_fast	8	6	6	4
LVTTL24_fast	5	4	4	3
LVDCI_15 50 Ω impedance	10	8	8	5
LVDCI_DV2_15 25 Ω impedance	5	4	4	3
LVCMOS15_2_slow	51	38	38	26
LVCMOS15_4_slow	31	23	23	16
LVCMOS15_6_slow	22	17	17	11
LVCMOS15_8_slow	17	13	13	9
LVCMOS15_12_slow	11	8	8	6
LVCMOS15_16_slow	8	6	6	4
LVCMOS15_2_fast	30	23	23	15
LVCMOS15_4_fast	18	14	14	9
LVCMOS15_6_fast	13	10	10	7
LVCMOS15_8_fast	10	8	8	5
LVCMOS15_12_fast	8	6	6	4
LVCMOS15_16_fast	6	5	5	3
LVDCI_18 50 Ω impedance	11	8	8	6
LVDCI_DV2_18 25 Ω impedance	6	4	4	3
LVCMOS18_2_slow	58	44	44	29
LVCMOS18_4_slow	35	26	26	18

Table 2-36: **Guidelines for Max Number of Simultaneously Switching Outputs per Power/Ground Pair** ***(Continued)***

Standard	Package			
	FG,BG,FF,BF	CS	XC2V40-FG	XC2V40-CS
LVCMOS18_6_slow	25	19	19	13
LVCMOS18_8_slow	19	14	14	10
LVCMOS18_12_slow	13	10	10	7
LVCMOS18_16_slow	10	8	8	5
LVCMOS18_2_fast	34	26	26	17
LVCMOS18_4_fast	20	15	15	10
LVCMOS18_6_fast	15	11	11	8
LVCMOS18_8_fast	11	8	8	6
LVCMOS18_12_fast	9	7	7	5
LVCMOS18_16_fast	7	5	5	4
LVDCI_25 50 Ω impedance	13	10	10	7
LVDCI_DV2_25 25 Ω impedance	7	5	5	3
LVCMOS25_2_slow	68	51	51	34
LVCMOS25_4_slow	41	31	31	21
LVCMOS25_6_slow	29	22	22	15
LVCMOS25_8_slow	22	17	17	11
LVCMOS25_12_slow	15	11	11	8
LVCMOS25_16_slow	11	8	8	6
LVCMOS25_24_slow	7	5	5	4
LVCMOS25_2_fast	40	30	30	20
LVCMOS25_4_fast	24	18	18	12
LVCMOS25_6_fast	17	13	13	9
LVCMOS25_8_fast	13	10	10	7
LVCMOS25_12_fast	10	8	8	5
LVCMOS25_16_fast	8	6	6	4
LVCMOS25_24_fast	5	4	4	2
LVDCI_33 50 Ω impedance	13	10	10	7
LVDCI_DV2_33 25 Ω impedance	7	5	5	3
LVCMOS33_2_slow	68	51	51	34
LVCMOS33_4_slow	41	31	31	21
LVCMOS33_6_slow	29	22	22	15
LVCMOS33_8_slow	22	17	17	11

Table 2-36: **Guidelines for Max Number of Simultaneously Switching Outputs per Power/Ground Pair *(Continued)***

Standard	**Package**			
	FG,BG,FF,BF	**CS**	**XC2V40-FG**	**XC2V40-CS**
LVCMOS33_12_slow	15	11	11	8
LVCMOS33_16_slow	11	8	8	6
LVCMOS33_24_slow	7	5	5	4
LVCMOS33_2_fast	40	30	30	20
LVCMOS33_4_fast	24	18	18	12
LVCMOS33_6_fast	17	13	13	9
LVCMOS33_8_fast	13	10	10	7
LVCMOS33_12_fast	10	8	8	5
LVCMOS33_16_fast	8	6	6	4
LVCMOS33_24_fast	5	4	4	2
PCI33/66/X	8	6	6	4
GTL	4	3	3	2
GTL_DCI	3	2	2	1
GTL+	4	3	3	2
GTL+_DCI	3	2	2	1
HSTLI	20	15	15	10
HSTLI_DCI	20	15	15	10
HSTLII	10	8	8	5
HSTLII_DCI	7	5	5	4
HSTLIII	8	6	6	4
HSTLIII_DCI	8	6	6	4
HSTLIV	4	3	3	2
HSTLIV_DCI	4	3	3	2
SSTL2I	15	11	11	8
SSTL2I_DCI	15	11	11	8
SSTL2II	10	8	8	5
SSTL2II_DCI	5	4	4	3
SSTL3I	12	9	9	6
SSTL3I_DCI	12	9	9	6
SSTL3II	8	6	6	4
SSTL3II_DCI	4	3	3	2
AGP	9	7	7	5

Table 2-37: **Virtex-II Equivalent Power/Ground Pairs per Bank**

Package	XC2V Device											
	40	**80**	**250**	**500**	**1000**	**1500**	**2000**	**3000**	**4000**	**6000**	**8000**	**10000**
CS144[1]	1	1	1									
FG256[1]	1	2	3	3	3							
FG456[1]			3	4	5							
FG676[1]						6	7	7				
BG575[1]					5	6	6					
BG728[1]							7	8				
FF896[2]					7	8	10					
FF1152[2]								11	13	13	13	13
FF1517[2]									14	17	17	17
BF957[2]							10	10	10	11	11	11

Notes:

1. Wire-bond only.
2. Flip-chip only.

Application Example

Creating a design with the SelectI/O feature requires either assignment of the `IOSTANDARD` attribute in the constraint file or instantiation of the desired library symbol within the design code.

To enter the `IOSTANDARD` attribute in the constraint file (UCF file), the following syntax can be used:

```
NET <pad net name> IOSTANDARD=<the name of the standard>
```

For example, to enter PCIX standard, use

```
NET <pad net name> IOSTANDARD=PCIX;
```

To instantiate a library symbol in the HDL code, use the proper input or output buffer name, and follow the standard syntax of instantiation.

For example, to instantiate a GTL input buffer in VHDL, the following syntax can be used:

```
GTL_buffer : IBUF_GTL port map (I=>data_in, O=>data_gtl_in);
```

At the board level, designers need to know the termination techniques required for each I/O standard.

This section describes some common application examples illustrating the termination techniques recommended by each of the single-ended standard supported by the SelectI/O features.

Termination Example

Circuit examples involving typical termination techniques for each of the SelectI/O standards follow. For a full range of accepted values for the DC voltage specifications for each standard, refer to the table associated with each figure.

The resistors used in each termination technique example and the transmission lines depicted represent board level components and are not meant to represent components on the device.

GTL

A sample circuit illustrating a valid termination technique for GTL is shown in Figure 2-82.

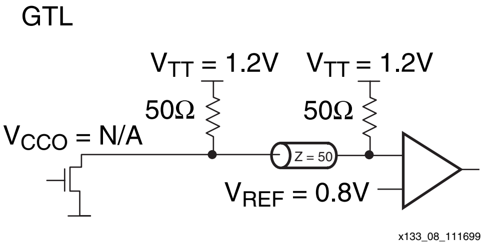


Figure 2-82: **GTL Terminated**

Table 2-38 lists DC voltage specifications.

Table 2-38: **GTL Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	-	N/A	-
$V_{REF} = N \times V_{TT}$[1]	0.74	0.8	0.86
V_{TT}	1.14	1.2	1.26
$V_{IH} \geq V_{REF} + 0.05$	0.79	0.85	-
$V_{IL} \leq V_{REF} - 0.05$	-	0.75	0.81
V_{OH}	-	-	-
V_{OL}	-	0.2	0.4
I_{OH} at V_{OH} (mA)	-	-	-
I_{OL}at V_{OL} (mA) at 0.4 V	32	-	-
I_{OL}at V_{OL} (mA) at 0.2 V	-	-	40

Notes:

1. N must be greater than or equal to 0.653 and less than or equal to 0.68.

GTL +

Figure 2-83 shows a sample circuit illustrating a valid termination technique for GTL+.

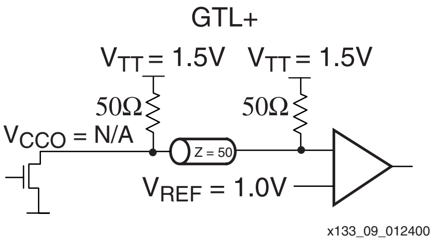


Figure 2-83: **GTL+ Terminated**

Table 2-39 lists DC voltage specifications.

Table 2-39: **GTL+ Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	-	-	-
$V_{REF} = N \times V_{TT}$[1]	0.88	1.0	1.12
V_{TT}	1.35	1.5	1.65
$V_{IH} \geq V_{REF} + 0.1$	0.98	1.1	-
$V_{IL} \leq V_{REF} - 0.1$	-	0.9	1.02
V_{OH}	-	-	-
V_{OL}	0.3	0.45	0.6
I_{OH} at V_{OH} (mA)	-	-	-
I_{OL}at V_{OL} (mA) at 0.6V	36	-	-
I_{OL}at V_{OL} (mA) at 0.3V	-	-	48

Notes:

1. N must be greater than or equal to 0.653 and less than or equal to 0.68.

HSTL Class I

Figure 2-88 shows a sample circuit illustrating a valid termination technique for HSTL_I.

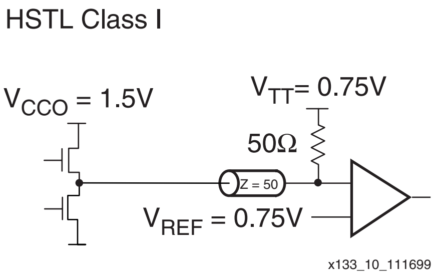


Figure 2-84: **Terminated HSTL Class I**

Table 2-44 lists DC voltage specifications.

Table 2-40: **HSTL Class I Voltage Specification**

Parameter	MIN	TYP	MAX
V_{CCO}	1.40	1.50	1.60
V_{REF}	0.68	0.75	0.90
V_{TT}	-	$V_{CCO} \times 0.5$	-
V_{IH}	$V_{REF} + 0.1$	-	-
V_{IL}	-	-	$V_{REF} - 0.1$
V_{OH}	$V_{CCO} - 0.4$	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–8	-	-
I_{OL}at V_{OL} (mA)	8	-	-

HSTL Class II

Figure 2-89 shows a sample circuit illustrating a valid termination technique for HSTL_II.

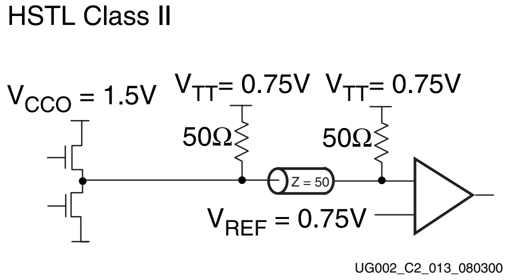


Figure 2-85: **Terminated HSTL Class II**

Table 2-45 lists DC voltage specifications.

Table 2-41: **HSTL Class II Voltage Specification**

Parameter	MIN	TYP	MAX
V_{CCO}	1.40	1.50	1.60
V_{REF} (1)	-	0.75	-
V_{TT}	-	$V_{CCO} \times 0.5$	-
V_{IH}	$V_{REF} + 0.1$	-	-
V_{IL}	-	-	$V_{REF} - 0.1$
V_{OH}	$V_{CCO} - 0.4$	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–16	-	-
I_{OL}at V_{OL} (mA)	16	-	-

Notes:

1. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

HSTL Class III

Figure 2-90 shows a sample circuit illustrating a valid termination technique for HSTL_III.

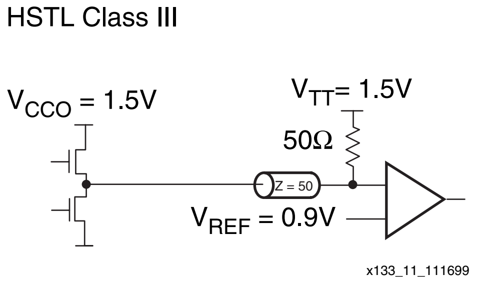


Figure 2-86: **Terminated HSTL Class III**

Table 2-46 lists DC voltage specifications.

Table 2-42: **HSTL Class III Voltage Specification**

Parameter	MIN	TYP	MAX
V_{CCO}	1.40	1.50	1.60
V_{REF} (1)	-	0.90	-
V_{TT}	-	V_{CCO}	-
V_{IH}	$V_{REF} + 0.1$	-	-
V_{IL}	-	-	$V_{REF} - 0.1$
V_{OH}	$V_{CCO} - 0.4$	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–8	-	-
I_{OL}at V_{OL} (mA)	24	-	-

Notes:

1. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

HSTL Class IV

Figure 2-91 shows a sample circuit illustrating a valid termination technique for HSTL_IV.

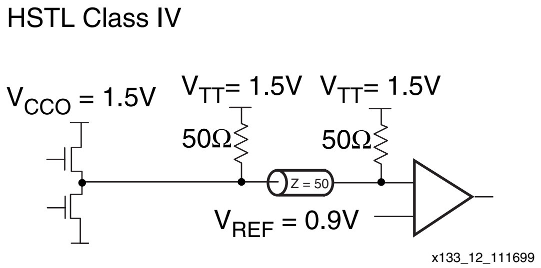


Figure 2-87: **Terminated HSTL Class IV**

Table 2-47 lists DC voltage specifications.

Table 2-43: **HSTL Class IV Voltage Specification**

Parameter	MIN	TYP	MAX
V_{CCO}	1.40	1.50	1.60
V_{REF}	-	0.90	-
V_{TT}	-	V_{CCO}	-
V_{IH}	$V_{REF} + 0.1$	-	-
V_{IL}	-	-	$V_{REF} - 0.1$
V_{OH}	$V_{CCO} - 0.4$	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–8	-	-
I_{OL}at V_{OL} (mA)	48	-	-

Notes:

1. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user.

HSTL Class I (1.8V)

Figure 2-88 shows a sample circuit illustrating a valid termination technique for HSTL_I.

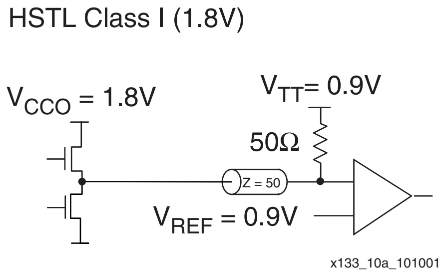


Figure 2-88: **Terminated HSTL Class I (1.8V)**

Table 2-44 lists DC voltage specifications.

Table 2-44: **HSTL Class I (1.8V) Voltage Specification**

Parameter	MIN	TYP	MAX
V_{CCO}	1.7	1.8	1.9
V_{REF}	0.8	0.9	1.1
V_{TT}	-	$V_{CCO} \times 0.5$	-
V_{IH}	$V_{REF} + 0.1$	-	-
V_{IL}	-	-	$V_{REF} - 0.1$
V_{OH}	$V_{CCO} - 0.4$	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	−8	-	-
I_{OL}at V_{OL} (mA)	8	-	-

HSTL Class II (1.8V)

Figure 2-89 shows a sample circuit illustrating a valid termination technique for HSTL_II.

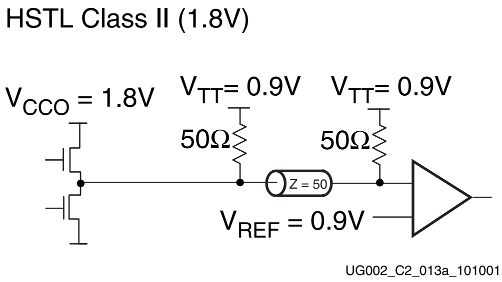


Figure 2-89: **Terminated HSTL Class II (1.8V)**

Table 2-45 lists DC voltage specifications.

Table 2-45: **HSTL Class II (1.8V) Voltage Specification**

Parameter	MIN	TYP	MAX
V_{CCO}	1.7	1.8	1.9
V_{REF} (1)	-	0.9	-
V_{TT}	-	$V_{CCO} \times 0.5$	-
V_{IH}	$V_{REF} + 0.1$	-	-
V_{IL}	-	-	$V_{REF} - 0.1$
V_{OH}	$V_{CCO} - 0.4$	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–16	-	-
I_{OL}at V_{OL} (mA)	16	-	-

Notes:

1. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

HSTL Class III (1.8V)

Figure 2-90 shows a sample circuit illustrating a valid termination technique for HSTL_III.

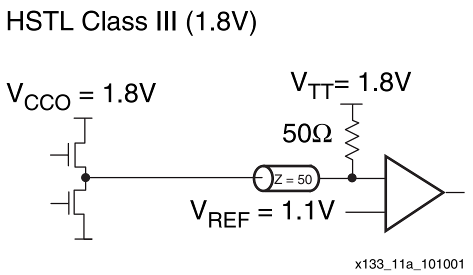


Figure 2-90: **Terminated HSTL Class III (1.8V)**

Table 2-46 lists DC voltage specifications.

Table 2-46: **HSTL Class III (1.8V) Voltage Specification**

Parameter	MIN	TYP	MAX
V_{CCO}	1.7	1.8	1.9
V_{REF} (1)	-	1.1	-
V_{TT}	-	V_{CCO}	-
V_{IH}	$V_{REF} + 0.1$	-	-
V_{IL}	-	-	$V_{REF} - 0.1$
V_{OH}	$V_{CCO} - 0.4$	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–8	-	-
I_{OL}at V_{OL} (mA)	24	-	-

Notes:

1. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

HSTL Class IV (1.8V)

Figure 2-91 shows a sample circuit illustrating a valid termination technique for HSTL_IV.

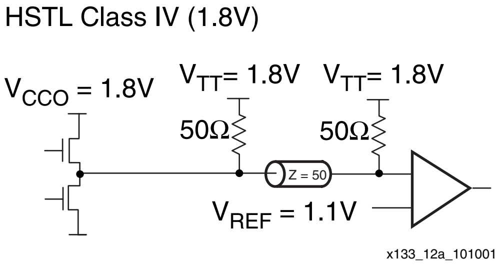


Figure 2-91: **Terminated HSTL Class IV (1.8V)**

Table 2-47 lists DC voltage specifications.

Table 2-47: **HSTL Class IV (1.8V) Voltage Specification**

Parameter	MIN	TYP	MAX
V_{CCO}	1.7	1.8	1.9
V_{REF}	-	1.1	-
V_{TT}	-	V_{CCO}	-
V_{IH}	V_{REF} + 0.1	-	-
V_{IL}	-	-	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–8	-	-
I_{OL}at V_{OL} (mA)	48	-	-

Notes:

1. Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user.

SSTL3_I

Figure 2-92 shows a sample circuit illustrating a valid termination technique for SSTL3_I.

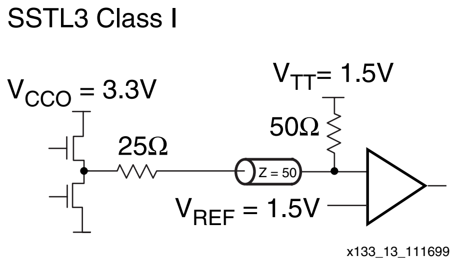


Figure 2-92: **Terminated SSTL3_I**

Table 2-48 lists DC voltage specifications.

Table 2-48: **SSTL3_I Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	3.0	3.3	3.6
$V_{REF} = 0.45 \times V_{CCO}$	1.3	1.5	1.7
$V_{TT} = V_{REF}$	1.3	1.5	1.7
$V_{IH} \geq V_{REF} + 0.2$	1.5	1.7	3.9[1]
$V_{IL} \leq V_{REF} - 0.2$	–0.3[2]	1.3	1.5
$V_{OH} \geq V_{REF} + 0.6$	1.9	2.1	-
$V_{OL} \leq V_{REF} - 0.6$	-	0.9	1.1
I_{OH} at V_{OH} (mA)	–8	-	-
I_{OL}at V_{OL} (mA)	8	-	-

Notes:

1. V_{IH} maximum is $V_{CCO} + 0.3$
2. V_{IL} minimum does not conform to the formula

SSTL3_II

Figure 2-93 shows a sample circuit illustrating a valid termination technique for SSTL3_II.

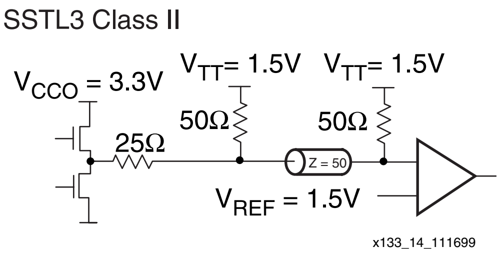


Figure 2-93: **Terminated SSTL3_II**

Table 2-49 lists DC voltage specifications.

Table 2-49: **SSTL3_II Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	3.0	3.3	3.6
$V_{REF} = 0.45 \times V_{CCO}$	1.3	1.5	1.7
$V_{TT} = V_{REF}$	1.3	1.5	1.7
$V_{IH} \geq V_{REF} + 0.2$	1.5	1.7	3.9(1)
$V_{IL} \leq V_{REF} - 0.2$	–0.3(2)	1.3	1.5
$V_{OH} \geq V_{REF} + 0.8$	2.1	2.3	-
$V_{OL} \leq V_{REF} - 0.8$	-	0.7	0.9
I_{OH} at V_{OH} (mA)	–16	-	-
I_{OL}at V_{OL} (mA)	16	-	-

Notes:

1. V_{IH} maximum is $V_{CCO} + 0.3$
2. V_{IL} minimum does not conform to the formula

SSTL2_I

Figure 2-94 shows a sample circuit illustrating a valid termination technique for SSTL2_I.

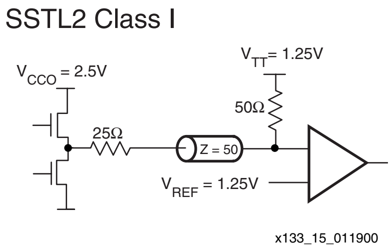


Figure 2-94: **Terminated SSTL2_I**

Table 2-50 lists DC voltage specifications.

Table 2-50: **SSTL2_I Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	2.3	2.5	2.7
$V_{REF} = 0.5 \times V_{CCO}$	1.15	1.25	1.35
$V_{TT} = V_{REF} + N^{(1)}$	1.11	1.25	1.39
$V_{IH} \geq V_{REF} + 0.18$	1.33	1.43	3.0(2)
$V_{IL} \leq V_{REF} - 0.18$	–0.3(3)	1.07	1.17
$V_{OH} \geq V_{REF} + 0.61$	1.76	1.82	1.96
$V_{OL} \leq V_{REF} - 0.61$	0.54	0.64	0.74
I_{OH} at V_{OH} (mA)	–7.6	-	-
I_{OL}at V_{OL} (mA)	7.6	-	-

Notes:

1. N must be greater than or equal to -0.04 and less than or equal to 0.04.
2. V_{IH} maximum is $V_{CCO} + 0.3$.
3. V_{IL} minimum does not conform to the formula.

SSTL2_II

Figure 2-95 shows a sample circuit illustrating a valid termination technique for SSTL2_II.

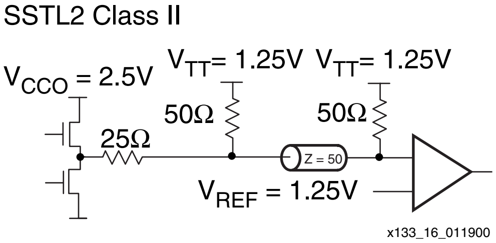


Figure 2-95: **Terminated SSTL2_II**

Table 2-51 lists DC voltage specifications.

Table 2-51: **SSTL2_II Voltage Specifications**

Parameter	**Min**	**Typ**	**Max**
V_{CCO}	2.3	2.5	2.7
$V_{REF} = 0.5 \times V_{CCO}$	1.15	1.25	1.35
$V_{TT} = V_{REF} + N^{(1)}$	1.11	1.25	1.39
$V_{IH} \geq V_{REF} + 0.18$	1.33	1.43	$3.0^{(2)}$
$V_{IL} \leq V_{REF} - 0.18$	$-0.3^{(3)}$	1.07	1.17
$V_{OH} \geq V_{REF} + 0.8$	1.95	2.05	-
$V_{OL} \leq V_{REF} - 0.8$	-	0.45	0.55
I_{OH} at V_{OH} (mA)	–15.2	-	-
I_{OL}at V_{OL} (mA)	15.2	-	-

Notes:

1. N must be greater than or equal to –0.04 and less than or equal to 0.04.
2. V_{IH} maximum is V_{CCO} + 0.3.
3. V_{IL} minimum does not conform to the formula.

PCI33_3, PCI66_3, and PCIX

Table 2-52 lists DC voltage specifications.

Table 2-52: **PCI33_3, PCI66_3, and PCIX Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	3.0	3.3	3.5
V_{REF}	-	-	-
V_{TT}	-	-	-
$V_{IH} = 0.5 \times V_{CCO}$	1.5	1.65	$V_{CCO} + 0.5$
$V_{IL} = 0.3 \times V_{CCO}$	– 0.5	0.99	1.08
$V_{OH} = 0.9 \times V_{CCO}$	2.7	-	-
$V_{OL} = 0.1 \times V_{CCO}$	-	-	0.36
I_{OH} at V_{OH} (mA)	Note 1	-	-
I_{OL}at V_{OL} (mA)	Note 1	-	-

Notes:

1. Tested according to the relevant specification.

LVTTL

Table 2-53 lists DC voltage specifications.

Table 2-53: **LVTTL Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	3.0	3.3	3.6
V_{REF}	-	-	-
V_{TT}	-	-	-
V_{IH}	2.0	-	3.6
V_{IL}	–0.5	-	0.8
V_{OH}	2.4	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–24	-	-
I_{OL}at V_{OL} (mA)	24	-	-

Notes:

1. $V_{OL}$and V_{OH} for lower drive currents are sample tested.

LVCMOS15

Table 2-54 lists DC voltage specifications.

Table 2-54: **LVCMOS15 Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	-	1.5	-
V_{REF}	-	-	-
V_{TT}	-	-	-
$V_{IH} = 0.7 \times V_{CCO}$	1.05	-	1.65
$V_{IL} = 0.2 \times V_{CCO}$	–0.5	-	0.3
$V_{OH} = V_{CCO} - 0.45$	-	1.05	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–16	-	-
I_{OL}at V_{OL} (mA)	16	-	-

LVCMOS18

Table 2-55 lists DC voltage specifications.

Table 2-55: **LVCMOS18 Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	1.7	1.8	1.9
V_{REF}	-	-	-
V_{TT}	-	-	-
$V_{IH} = 0.7 \times V_{CCO}$	1.19	-	1.95
$V_{IL} = 0.2 \times V_{CCO}$	–0.5	-	0.4
$V_{OH} = V_{CCO} - 0.4$	1.3	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–16	-	-
I_{OL}at V_{OL} (mA)	16	-	-

LVCMOS25

Table 2-56 lists DC voltage specifications.

Table 2-56: **LVCMOS25 Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	2.3	2.5	2.7
V_{REF}	-	-	-
V_{TT}	-	-	-
V_{IH}	1.7	-	2.7
V_{IL}	–0.5	-	0.7
V_{OH}	1.9	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–24	-	-
I_{OL}at V_{OL} (mA)	24	-	-

LVCMOS33

Table 2-57 lists DC voltage specifications.

Table 2-57: **LVCMOS33 Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	3.0	3.3	3.6
V_{REF}	-	-	-
V_{TT}	-	-	-
V_{IH}	2.0	-	3.6
V_{IL}	–0.5	-	0.8
V_{OH}	2.6	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–24	-	-
I_{OL}at V_{OL} (mA)	24	-	-

AGP-2X

Table 2-58 lists DC voltage specifications.

Table 2-58: **AGP-2X Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	3.0	3.3	3.6
$V_{REF} = N \times V_{CCO}^{(1)}$	1.17	1.32	1.48
V_{TT}	-	-	-
$V_{IH} \geq V_{REF} + 0.2$	1.37	1.52	-
$V_{IL} \leq V_{REF} - 0.2$	-	1.12	1.28
$V_{OH} = 0.9 \times V_{CCO}$	2.7	3.0	-
$V_{OL} = 0.1 \times V_{CCO}$	-	0.33	0.36
I_{OH} at V_{OH} (mA)	Note 2	-	-
I_{OL}at V_{OL} (mA)	Note 2	-	-

Notes:

1. N must be greater than or equal to 0.39 and less than or equal to 0.41.
2. Tested according to the relevant specification.

Using Digitally Controlled Impedance (DCI)

Introduction

As FPGAs get bigger and system clock speeds get faster, PCB board design and manufacturing has become more difficult. With ever faster edge rates, maintaining signal integrity becomes a critical issue. Designers must make sure that most PC board traces are terminated properly to avoid reflections or ringing.

To terminate a trace, resistors are traditionally added to make the output and/or input match the impedance of the receiver or driver to the impedance of the trace. However, due to the increase in the device I/O counts, adding resistors close to the device pins increases the board area and component count and might even be physically impossible. To address these issues and to achieve better signal integrity, Xilinx developed a new I/O technology for the Virtex-II device family, Digitally Controlled Impedance (DCI).

DCI adjusts the output impedance or input termination to accurately match the characteristic impedance of the transmission line. DCI actively adjusts the impedance of the I/O to equal an external reference resistance. This compensates for changes in I/O impedance due to process variation. It also continuously adjusts the impedance of the I/O to compensate for variations of temperature and supply voltage fluctuations.

In the case of controlled impedance drivers, DCI controls the driver impedance to match two reference resistors, or optionally, to match half the value of these reference resistors. DCI eliminates the need for external termination resistors.

DCI provides the termination for transmitters or receivers. This eliminates the need for termination resistors on the board, reduces board routing difficulties and component count, and improves signal integrity by eliminating stub reflection. Stub reflection occurs when termination resistors are located too far from the end of the transmission line. With DCI, the termination resistors are as close as possible to the output driver or the input buffer, thus, eliminating stub reflections completely.

Xilinx DCI

DCI uses two multi-purpose reference pins in each bank to control the impedance of the driver or the parallel termination value for all of the I/Os of that bank. The N reference pin (VRN) must be pulled up to V_{CCO} by a reference resistor, and the P reference pin (VRP) must be pulled down to ground by another reference resistor. The value of each reference resistor should be equal to the characteristic impedance of the PC board traces, or should be twice that value (configuration option).

When a DCI I/O standard is used on a particular bank, the two multi-purpose reference pins cannot be used as regular I/Os. however, if DCI I/O standards are not used in the bank, these pins are available as regular I/O pins. Check the Virtex-II pinout for detailed pin descriptions.

DCI adjusts the impedance of the I/O by selectively turning transistors in the I/Os on or off. The impedance is adjusted to match the external reference resistors. The impedance adjustment process has two phases. The first phase, which compensates for process variations, is done during the device startup sequence. The second phase, which maintains the impedance in response to temperature and supply voltage changes, begins immediately after the first phase and continues indefinitely, even while the part is operating. By default, the DONE pin does not go High until the impedance adjustment process has completed.

For controlled impedance output drivers, the impedance can be adjusted either to match the reference resistors or half the resistance of the reference resistors. For on-chip termination, the termination is always adjusted to match the reference resistors.

DCI can configure output drivers to be the following types:

1. Controlled Impedance Driver (Source Termination)
2. Controlled Impedance Driver with Half Impedance (Source Termination)

It can also configure inputs to have he following types of on-chip terminations:

1. Termination to V_{CCO} (Single Termination)
2. Termination to $V_{CCO}/2$ (Split Termination, Thevenin equivalent)

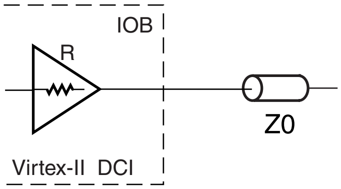
For bidirectional operation, both ends of the line can be DCI-terminated permanently:

1. Termination to V_{CCO} (Single Termination)
2. Termination to $V_{CCO}/2$ (Split Termination, Thevenin equivalent)

Alternatively, bidirectional point-to-point lines can use controlled-impedance drivers (with 3-state buffers) on both ends.

Controlled Impedance Driver (Source Termination)

Some I/O standards, such as LVTTL, LVCMOS, etc., must have a drive impedance that matches the characteristic impedance of the driven line. DCI can provide a controlled impedance output drivers that eliminate reflections without an external source termination. The impedance is set by the external reference resistors, whose resistance should be equal to the trace impedance. Figure 2-96 illustrates a controlled impedance driver inside Virtex-II device. The DCI I/O standards that support Controlled Impedance Driver are: LVDCI_15, LVDCI_18, LVDCI_25, and LVDCI_33.



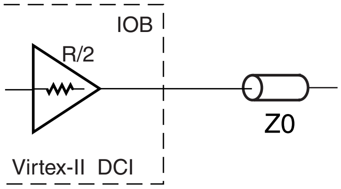
UG002_c2_047_120400

Figure 2-96: **Controlled Impedance Driver**

Controlled Impedance Driver With Half Impedance (Source Termination)

DCI can also provide drivers with one half of the impedance of the reference resistors. The DCI I/O standards that support controlled impedance driver with half impedance are: LVDCI_DV2_15, LVDCI_DV2_18, LVDCI_DV2_25, and LVDCI_DV2_33

Figure 2-97 illustrates a controlled driver with half impedance inside a Virtex-II device.

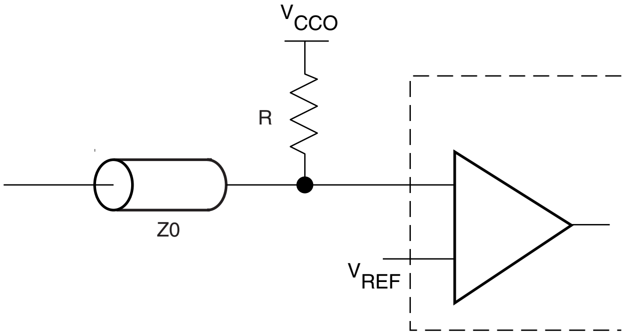


UG002_c2_052_120400

Figure 2-97: **Controlled Impedance Driver With Half Impedance**

Termination to V_{CCO} (Single Termination)

Some I/O standards, such as HSTL Class III, IV, etc., require an input termination to V_{CCO}. See Figure 2-98.

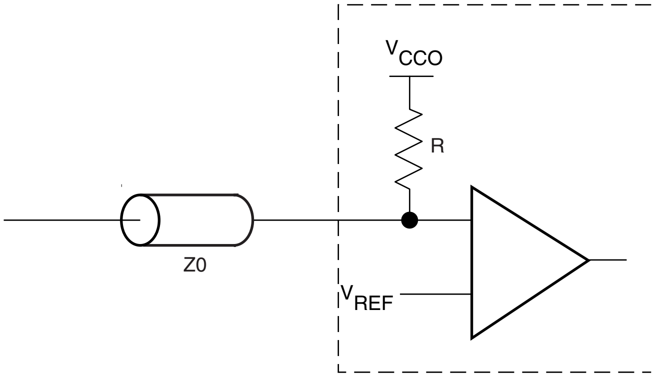


ug002_c2_053_110600

Figure 2-98: **Single Termination Without DCI**

DCI can provide this termination to V_{CCO} using single termination. The termination resistance is set by the reference resistors. For GTL and HSTL standards, they should be controlled by 50-ohm reference resistors. The DCI I/O standards that support single termination are: GTL_DCI, GTLP_DCI, HSTL_III_DCI, and HSTL_IV_DCI.

Figure 2-99 illustrates single termination inside a Virtex-II device.



ug002_c2_054_110600

Figure 2-99: **Single Termination Using DCI**

Termination to V_{CCO}/2 (Split Termination)

Some I/O standards, such as HSTL Class I, II, SSTL3_I, etc., require an input termination voltage of V_{CCO}/2. See Figure 2-100.

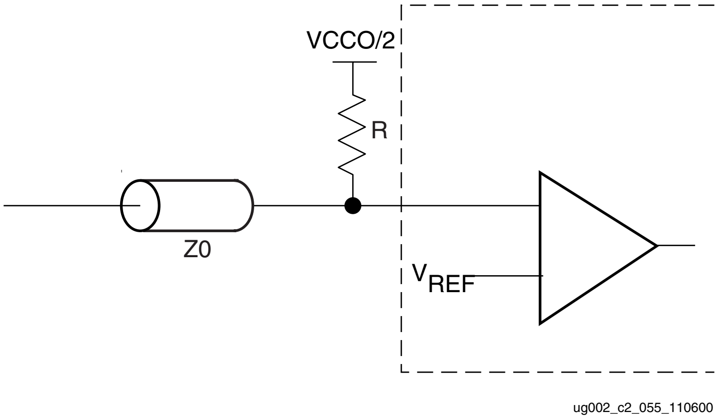


Figure 2-100: **Split Termination Without DCI**

This is equivalent to having a split termination composed of two resistors. One terminates to V_{CCO}, the other to ground. The resistor values are 2R. DCI provides termination to V_{CCO}/2 using split termination. The termination resistance is set by the external reference resistors, i.e., the resistors to V_{CC} and ground are each twice the reference resistor value. If users are planning to use HSTL or SSTL standards, the reference resistors should be 50-ohms. The DCI I/O standards that support split termination are: HSTL_I_DCI, HSTL_II_DCI, SSTL2_I_DCI, SSTL2_II_DCI, SSTL3_I_DCI, and SSTL3_II_DCI.

Figure 2-101 illustrates split termination inside a Virtex-II device.

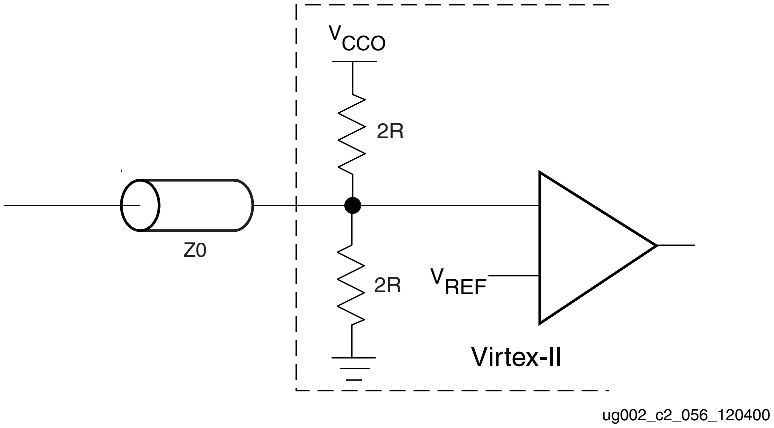


Figure 2-101: **Split Termination Using DCI**

Driver With Single Termination

Some I/O standards, such as HSTL Class IV, require an output termination to V_{CCO}. Figure 2-102 illustrates the output termination to V_{CCO}.

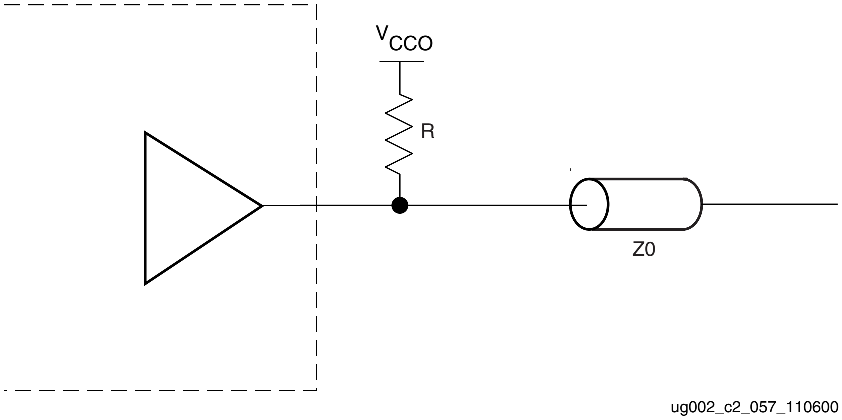


Figure 2-102: **Driver With Single Termination Without DCI**

DCI can provide this termination to V_{CCO} using single termination. In this case, DCI only controls the impedance of the termination, but not the driver. If users are planning to use GTL or HSTL standards, the external reference resistors should be 50-ohms. The DCI I/O standards that support a driver with single termination are: GTL_DCI, GTLP_DCI, and HSTL_IV_DCI.

Figure 2-103 illustrates a driver with single termination inside a Virtex-II device

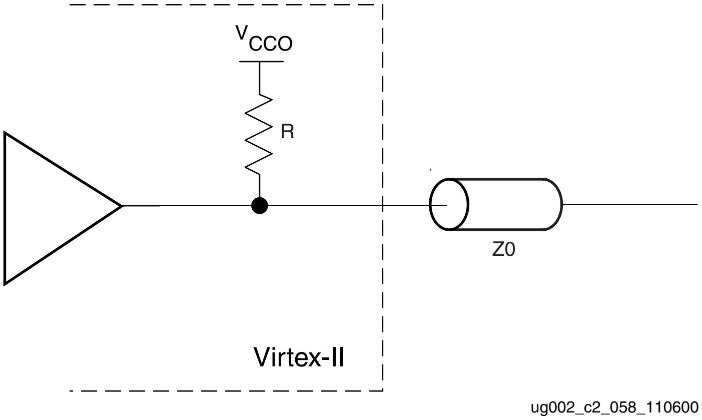


Figure 2-103: **Driver With Single Termination Using DCI**

Driver With Split Termination

Some I/O standards, such as HSTL Class II, require an output termination to $V_{CCO}/2$. See Figure 2-104.

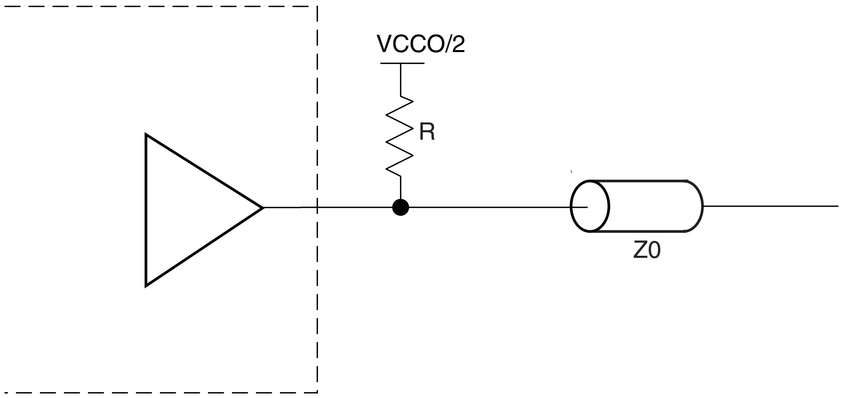


Figure 2-104: **Driver With Split Terminating**

DCI can provide this termination to $V_{CCO}/2$ using split termination. It only controls the impedance of the termination, but not the driver. For HSTL or SSTL standards, the external reference resistors should be 50-ohms. The DCI I/O standards that support a Driver with split termination are: HSTL_II_DCI, SSTL2_II_DCI, and SSTL3_II_DCI.

Figure 2-105 illustrates a driver with split termination inside a Virtex-II device.

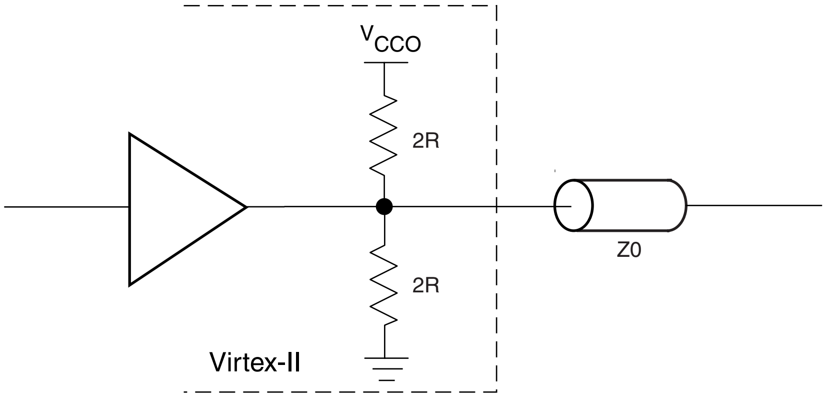


Figure 2-105: **Driver With Split Termination Using DCI**

Software Support

This section lists the valid DCI I/O buffer library components and describes how to use DCI in the Xilinx software.

DCI I/O Buffer Library Components

The DCI input buffer library components, including global clock buffer, are the following:

- IBUFG_GTLP_DCI
- IBUFG_GTL_DCI
- IBUFG_HSTL_I_DCI
- IBUFG_HSTL_II_DCI
- IBUFG_HSTL_III_DCI
- IBUFG_HSTL_IV_DCI
- IBUFG_LVDCI_15
- IBUFG_LVDCI_18
- IBUFG_LVDCI_25
- IBUFG_LVDCI_33
- IBUFG_LVDCI_DV2_15
- IBUFG_LVDCI_DV2_18
- IBUFG_LVDCI_DV2_25
- IBUFG_LVDCI_DV2_33
- IBUFG_SSTL2_I_DCI
- IBUFG_SSTL2_II_DCI
- IBUFG_SSTL3_I_DCI
- IBUFG_SSTL3_II_DCI
- IBUF_GTLP_DCI
- IBUF_GTL_DCI
- IBUF_HSTL_I_DCI
- IBUF_HSTL_II_DCI
- IBUF_HSTL_III_DCI
- IBUF_HSTL_IV_DCI
- IBUF_LVDCI_15
- IBUF_LVDCI_18
- IBUF_LVDCI_25
- IBUF_LVDCI_33
- IBUF_LVDCI_DV2_15
- IBUF_LVDCI_DV2_18
- IBUF_LVDCI_DV2_25
- IBUF_LVDCI_DV2_33
- IBUF_SSTL2_I_DCI
- IBUF_SSTL2_II_DCI
- IBUF_SSTL3_I_DCI
- IBUF_SSTL3_II_DCI

The following are DCI output buffer library components:

- OBUF_GTLP_DCI
- OBUF_GTL_DCI
- OBUF_HSTL_I_DCI
- OBUF_HSTL_II_DCI
- OBUF_HSTL_III_DCI
- OBUF_HSTL_IV_DCI
- OBUF_LVDCI_15
- OBUF_LVDCI_18
- OBUF_LVDCI_25
- OBUF_LVDCI_33
- OBUF_LVDCI_DV2_15
- OBUF_LVDCI_DV2_18
- OBUF_LVDCI_DV2_25
- OBUF_LVDCI_DV2_33
- OBUF_SSTL2_I_DCI
- OBUF_SSTL2_II_DCI
- OBUF_SSTL3_I_DCI
- OBUF_SSTL3_II_DCI

The following are DCI 3 state output buffer library components:

- OBUFT_GTLP_DCI
- OBUFT_GTL_DCI
- OBUFT_HSTL_I_DCI
- OBUFT_HSTL_II_DCI
- OBUFT_HSTL_III_DCI
- OBUFT_HSTL_IV_DCI
- OBUFT_LVDCI_15
- OBUFT_LVDCI_18
- OBUFT_LVDCI_25
- OBUFT_LVDCI_33
- OBUFT_LVDCI_DV2_15
- OBUFT_LVDCI_DV2_18
- OBUFT_LVDCI_DV2_25
- OBUFT_LVDCI_DV2_33
- OBUFT_SSTL2_I_DCI
- OBUFT_SSTL2_II_DCI
- OBUFT_SSTL3_I_DCI
- OBUFT_SSTL3_II_DCI

The following are DCI I/O buffer library components:

- IOBUF_GTLP_DCI
- IOBUF_GTL_DCI
- IOBUF_HSTL_II_DCI
- IOBUF_HSTL_IV_DCI
- IOBUF_SSTL2_II_DCI
- IOBUF_SSTL3_II_DCI
- IOBUF_LVDCI_15
- IOBUF_LVDCI_18
- IOBUF_LVDCI_25
- IOBUF_LVDCI_33
- IOBUF_LVDCI_DV2_15
- IOBUF_LVDCI_DV2_18
- IOBUF_LVDCI_DV2_25
- IOBUF_LVDCI_DV2_33

How to Use DCI in the Software

There are two ways for users to use DCI for Virtex-II devices:

1. Use the IOSTANDARD attribute in the constraint file.
2. Instantiate DCI input or output buffers in the HDL code.

IOSTANDARD Attribute

The IOSTANDARD attribute can be entered through the NCF or UCF file. The syntax is as follows:

```
NET <net name> IOSTANDARD = LVDCI_25;
```

Where <net name> is the name between the IPAD and IBUF or OPAD or OBUF. For HDL designs, this name is the same as the port name.

The following are valid DCI attributes for output drivers:

- LVDCI_15
- LVDCI_18
- LVDCI_25
- LVDCI_33
- LVDCI_DV2_15
- LVDCI_DV2_15
- LVDCI_DV2_25
- LVDCI_DV2_33

The following are valid DCI attributes for terminations:

- GTL_DCI
- GTLP_DCI
- HSTL_I_DCI
- HSTL_II_DCI

- HSTL_III_DCI
- HSTL_IV_DCI
- SSTL2_I_DCI
- SSTL2_II_DCI
- SSTL3_I_DCI
- SSTL3_II_DCI

VHDL Example

Instantiating DCI input and output buffers is the same as instantiating any other I/O buffers. Users must make sure that the correct I/O buffer names are used and follow the standard syntax of instantiation.

For example, to instantiate a HSTL Class I output DCI buffer, the following syntax can be used:

```
HSTL_DCI_buffer: OBUF_HSTL_I_DCI port map (I=>data_out, O=>data_out_DCI);
```

Below is an example VHDL code that instantiates four 2.5 V LVDCI drivers and four HSTL Class I outputs.

```
-- Module: DCI_TEST
--
-- Description: VHDL example for DCI SelectI/O
-- Device: Virtex-II Family
---------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity dci_test is
port (clk, reset, ce, control : in std_logic;
  A, B : in std_logic_vector (3 downto 0);
  Dout : out std_logic_vector (3 downto 0);
  muxout : out std_logic_vector (3 downto 0));
end dci_test;

architecture dci_arch of dci_test is

--DCI output buffer component declaration
component OBUF_LVDCI_25 port (I : in std_logic; O : out std_logic);
end component;
attribute syn_black_box of OBUF_LVDCI_25 : component is true;
attribute black_box_pad_pin of OBUF_LVDCI_25 : component is "O";

--HSTL Class I DCI output buffer component declaration
component OBUF_HSTL_I_DCI port (I : in std_logic; O: out std_logic);
end component;
attribute syn_black_box of OBUF_HSTL_I_DCI : component is true;
attribute black_box_pad_pin of OBUF_HSTL_I_DCI : component is "O";

signal muxout_int : std_logic_vector (3 downto 0);
signal dout_int : std_logic_vector (3 downto 0);

begin

process (clk, reset)
begin
  if (reset = '1') then
         dout_int<="0000";
    elsif (clk'event and clk='1') then
```

```
      dout_int<=dout_int+1;
end if;
end process;

process (controls, A, B, DOUT_INT)

begin
  if (control='1') then
    muxout_int<=A and B;
else
    muxout_int<=Dout_int;
  end if;
end process;

U0 : OBUF_LVDCI_25 port map(
  I=>dout_int(0),
  O=>dout(0));

U1 : OBUF_LVDCI_25 port map(
       I=>dout_int(1),
       O=>dout(1));
U2 : OBUF_LVDCI_25 port map(
       I=>dout_int(2),
       O=>dout(2));
U3 : OBUF_LVDCI_25 port map(
       I=>dout_int(3),
       O=>dout(3));

K0 : OBUF_HSTL_I_DCI port map(
  I=>muxout_int(0),
  O=>muxout(0));

K1 : OBUF_HSTL_I_DCI port map(
       I=>muxout_int(1),
       O=>muxout(1));
K2 : OBUF_HSTL_I_DCI port map(
       I=>muxout_int(2),
       O=>muxout(2));
K3 : OBUF_HSTL_I_DCI port map(
       I=>muxout_int(3),
       O=>muxout(3));

end dci_arch;
```

DCI in Virtex-II Hardware

DCI only works with certain single-ended I/O standards and does not work with any differential I/O standard. DCI supports the following Virtex-II standards:

LVDCI, LVDCI_DV2, GTL_DCI, GTLP_DCI, HSTL_I_DCI, HSTL_II_DCI, HSTL_III_DCI, HSTL_IV_DCI, SSTL2_I_DCI, SSTL2_II_DCI, SSTL3_I_DCI, and SSTL3_II_DCI.

To correctly use DCI in a Virtex-II device, users must follow the following rules:

1. V_{CCO} pins must be connected to the appropriate V_{CCO} voltage based on the IOSTANDARDs in that bank.
2. Correct DCI I/O buffers must be used in the software either by using IOSTANDARD attributes or instantiations in the HDL code.
3. External reference resistors must be connected to multi-purpose pins (VRN and VRP) in the bank cannot be used as regular I/Os. Refer to the Virtex-II pinouts for the

specific pin locations. Pin VRN must be pulled up to V_{CCO} by its reference resistor. Pin VRP must be pulled down to ground by its reference resistor.

4. The value of the external reference resistors should be selected to give the desired output impedance. If using GTL_DCI, HSTL_DCI, or SSTL_DCI I/O standards, then they should be 50 ohms.
5. The values of the reference resistors must be within the supported range. Availability of this range is planned for the next release of the **Virtex-II Data Sheet**. (~30 to 100 Ω)
6. Follow the DCI I/O banking rules.

The DCI I/O banking rules are the following:

1. V_{REF} must be compatible for all of the inputs in the same bank.
2. V_{CCO} must be compatible for all of the inputs and outputs in the same bank.
3. No more than one DCI I/O standard using Single Termination type is allowed per bank.
4. No more than one DCI I/O standard using Split Termination type is allowed per bank.
5. Single Termination and Split Termination, Controlled Impedance Driver, and Controlled Impedance Driver with Half Impedance can co-exist in the same bank.

The behavior of DCI 3-state outputs is as follows:

If a LVDCI or LVDCI_DV2 driver is in 3-state, the driver is 3-stated. If a Driver with Single or Split Termination is in 3-state, the driver is 3-stated but the termination resistor remains.

The following section lists any special care actions that must be taken for each DCI I/O standard.

LVDCI_15, LVDCI_18, LVDCI_25, LVDCI_33

Using these buffers configures the outputs as controlled impedance drivers. The number extension at the end indicates the V_{CCO} voltage that should be used. For example, 15 means V_{CCO}=1.5 V, etc. There is no slew rate control or drive strength settings for LVDCI drivers.

LVDCI_DV2_15, LVDCI_DV2_18, LVDCI_DV2_25, LVDCI_DV_33

Using these buffers configures the outputs as controlled drivers with half impedance. The number extension at the end indicates the V_{CCO} voltage that should be used. For example, 15 means V_{CCO}=1.5 V, etc. There is no slew rate control or drive strength settings for LVDCI_DV2 drivers.

GTL_DCI

GTLP does not require a V_{CCO} voltage. However, for GTL_DCI, V_{CCO} must be connected to 1.2 V. GTL_DCI provides single termination to V_{CCO} for inputs or outputs.

GTLP_DCI

GTL+ does not require a V_{CCO} voltage. However, for GTLP_DCI, V_{CCO} must be connected to 1.5 V. GTLP_DCI provides single termination to V_{CCO} for inputs or outputs.

HSTL_ I_DCI, HSTL_ III_DCI

HSTL_I_DCI provides split termination to $V_{CCO}/2$ for inputs. HSTL_III_DCI provides single termination to V_{CCO} for inputs.

HSTL_ II_DCI, HSTL_ IV_DCI

HSTL_II_DCI provides split termination to $V_{CCO}/2$ for inputs or outputs. HSTL_IV_ DCI provides single termination to V_{CCO} for inputs or outputs.

SSTL2_ I_DCI, SSTL3_I_DCI

SSTL2_I_DCI and SSTL3_I_DCI provide split termination to $V_{CCO}/2$ for inputs. Then I/O standards are SSTL compatible.

SSTL2_II_DCI, SSTL3_II_DCI

SSTL2_II_DCI and SSTL3_II_DCI provide split termination to $V_{CCO}/2$ for inputs. Then I/O standards are SSTL compatible.

Figure 2-106 provides examples illustrating the use of the HSTL_I_DCI, HSTL_II_DCI, HSTL_III_DCI, and HSTL_IV_DCI I/O standards.

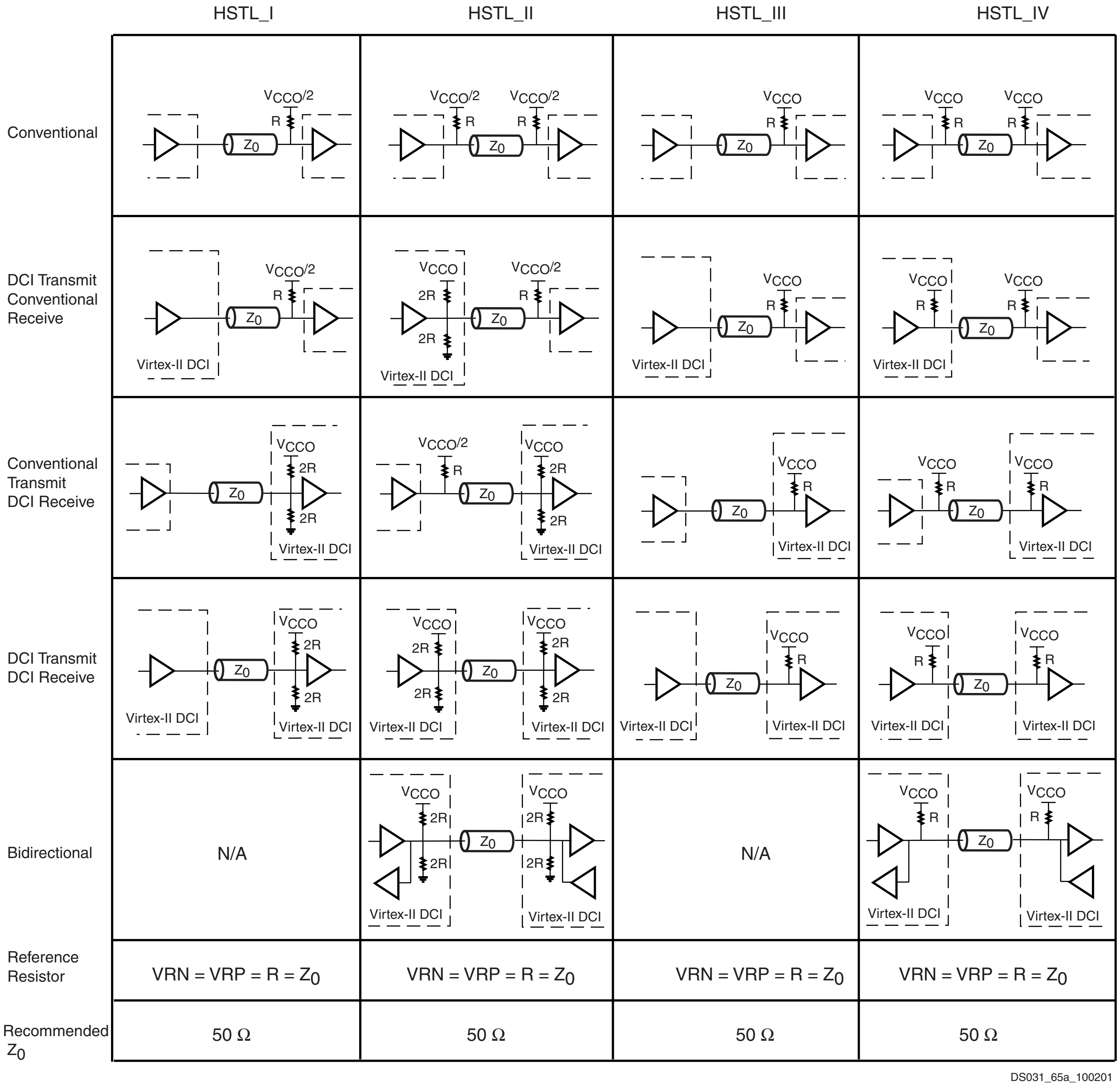


Figure 2-106: **HSTL DCI Usage Examples**

Figure 2-107 provides examples illustrating the use of the SSTL2_I_DCI, SSTL2_II_DCI, SSTL3_I_DCI, and SSTL3_II_DCI I/O standards.

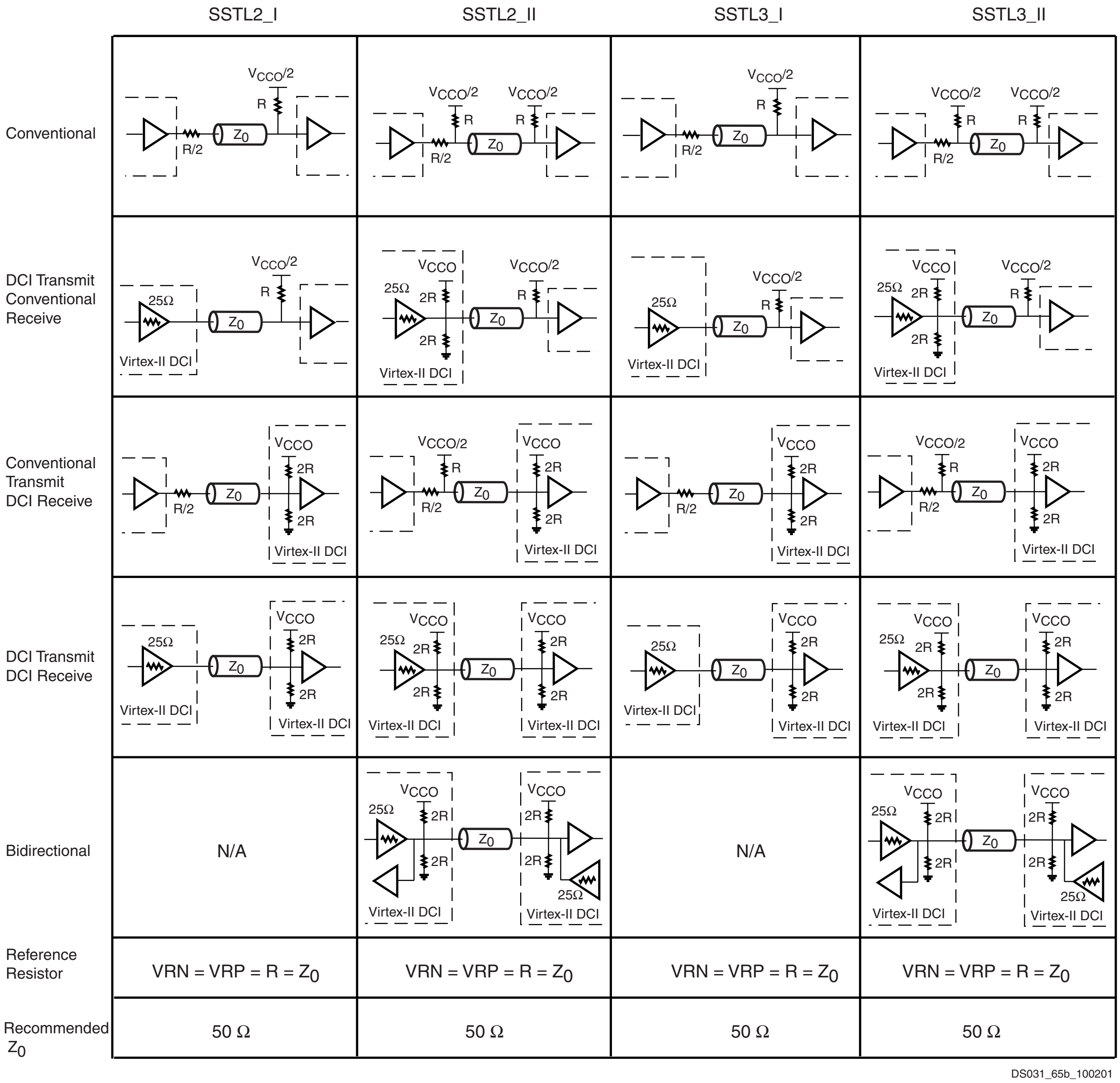


Figure 2-107: **SSTL DCI Usage Examples**

Using Double-Data-Rate (DDR) I/O

Introduction

Virtex-II devices have dedicated registers in a single IOB to implement input, output, and output with 3-state control Double-Data-Rate (DDR) registers. Input and output DDR is accomplished with the use of two registers in the IOB. A single clock triggers one register on a Low to High transition and a second register on a High to Low transition. Output DDR with 3-state requires the use of four registers in the IOB clocked in a similar fashion. Since the introduction of DLLs, Xilinx devices can generate low-skew clock signals that are 180 degrees out of phase, with a 50/50 duty cycle. These clocks reach the DDR registers in the IOB via dedicated routing resources.

Data Flow

Input DDR

Input DDR is accomplished via a single input signal driving two registers in the IOB. Both registers are clocked on the rising edge of their respective clocks. With proper clock forwarding, alternating bits from the input signal are clocked in on the rising edge of the two clocks, which are 180 degrees out of phase. Figure 2-108 depicts the input DDR registers and the signals involved.

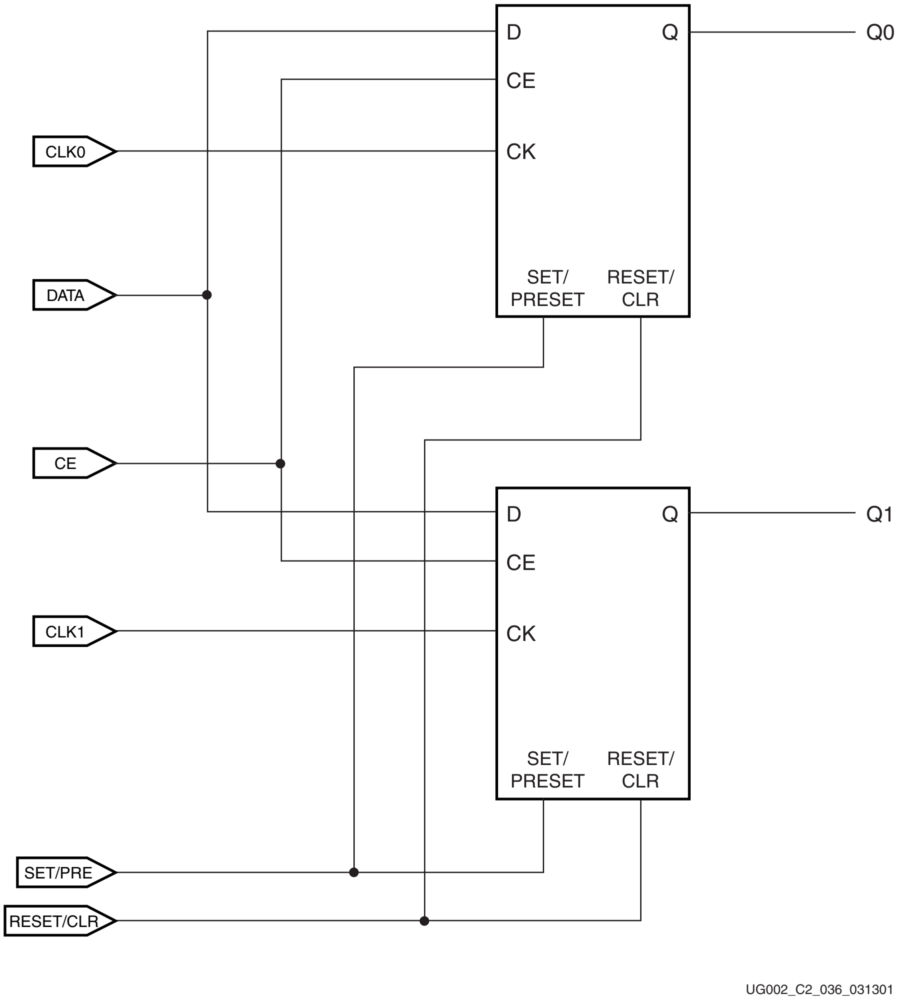
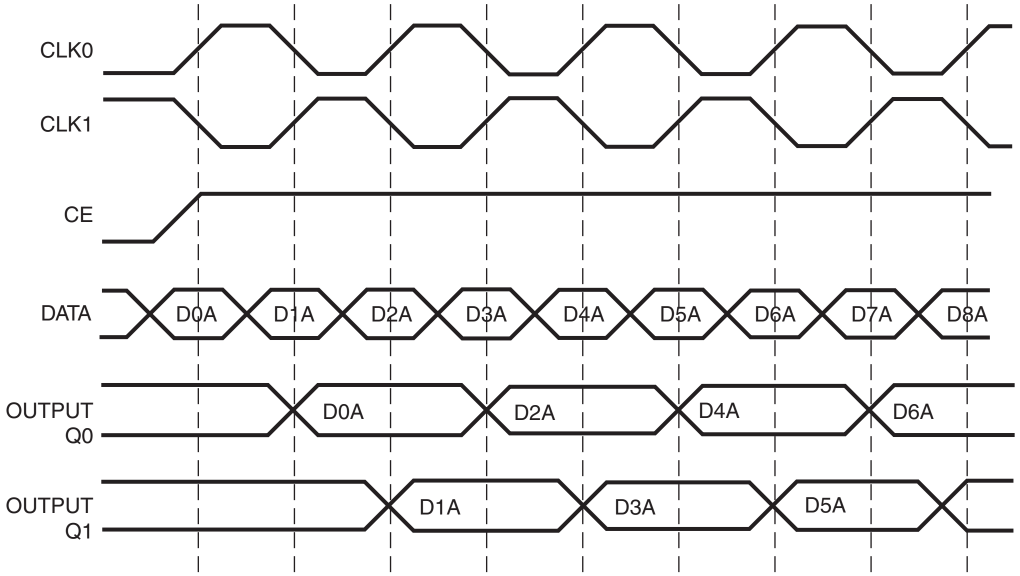


Figure 2-108: **Input DDR**

CLK0 and CLK1 are 180 degrees out of phase. Both registers share the SET/PRE and RESET/CLR lines. As shown in Figure 2-109, alternating bits on the DATA line are clocked in via Q0 and Q1 while CE is High. The clocks are shifted out of phase by the DCM (CLK0 and CLK180 outputs) or by the inverter available on the CLK1 clock input..

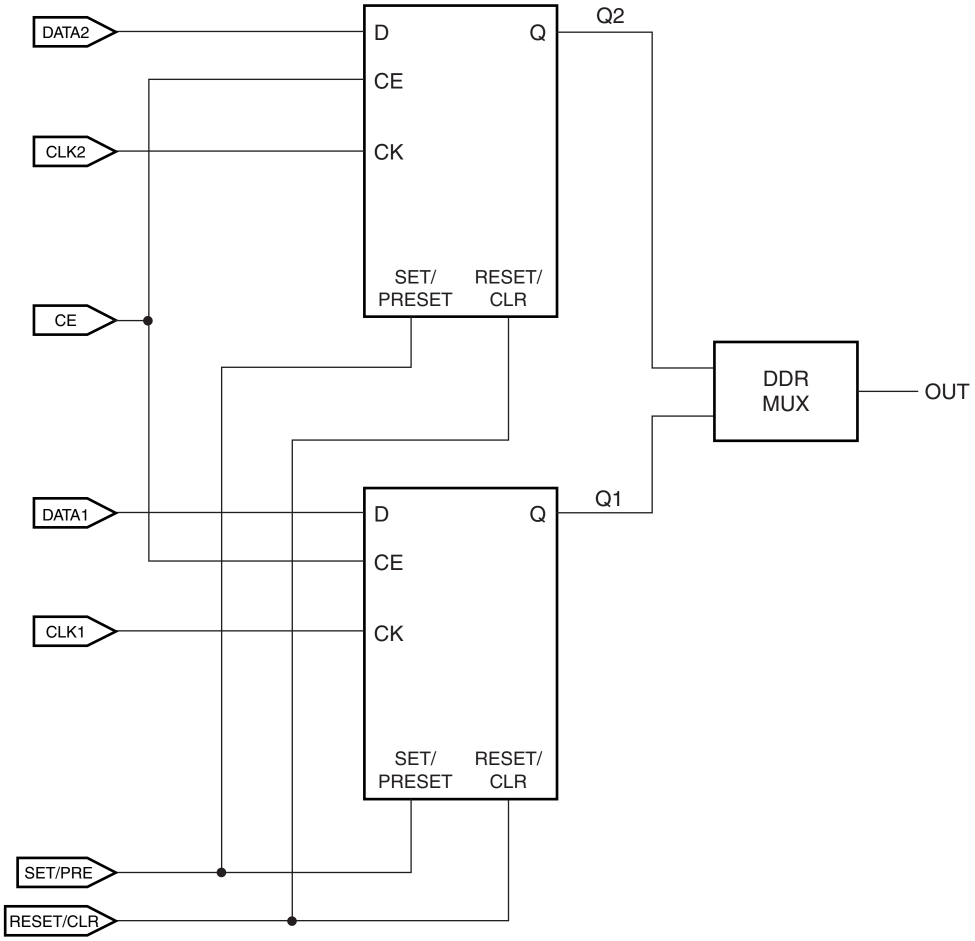


UG002_C2_037_032201

Figure 2-109: **Input DDR Timing Diagram**

Output DDR

Output DDR registers are used to clock output from the chip at twice the throughput of a single rising-edge clocking scheme. Clocking for output DDR is the same as input DDR. The clocks driving both registers are 180 degrees out of phase. The DDR MUX selects the register outputs. The output consists of alternating bits from DATA_1 and DATA_2. Figure 2-110 depicts the output DDR registers and the signals involved.



UG002_C2_038_101300

Figure 2-110: **Output DDR**

Both registers share the SET/PRE and RESET/CLR line. Both registers share the CE line which must be High for outputs to be seen on Q1 and Q2. Figure 2-111 shows the data flow for the output DDR registers.

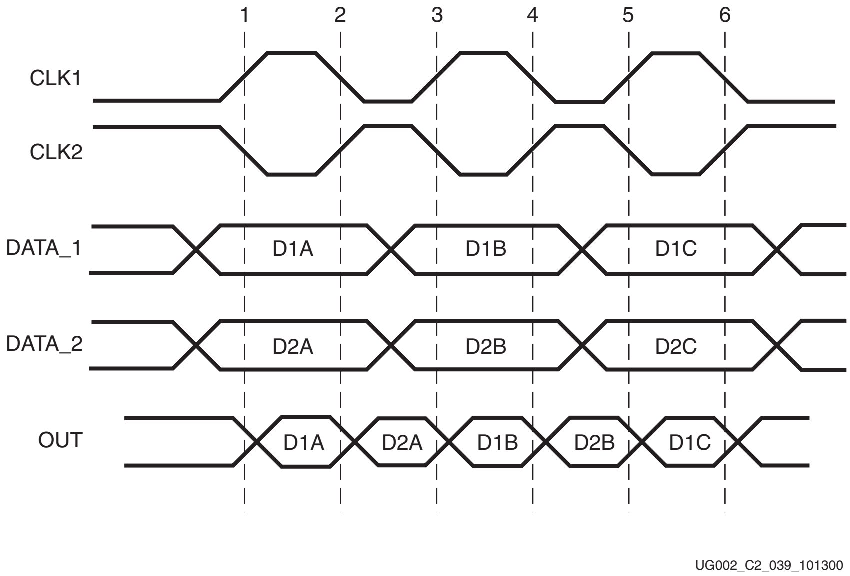


Figure 2-111: **Output DDR Timing Diagram**

Output DDR With 3-State Control

The 3-state control allows the output to have one of two values, either the output from the DDR MUX or high impedance.

The Enable signal is driven by a second DDR MUX (Figure 2-112). This application requires the instantiation of two output DDR primitives.

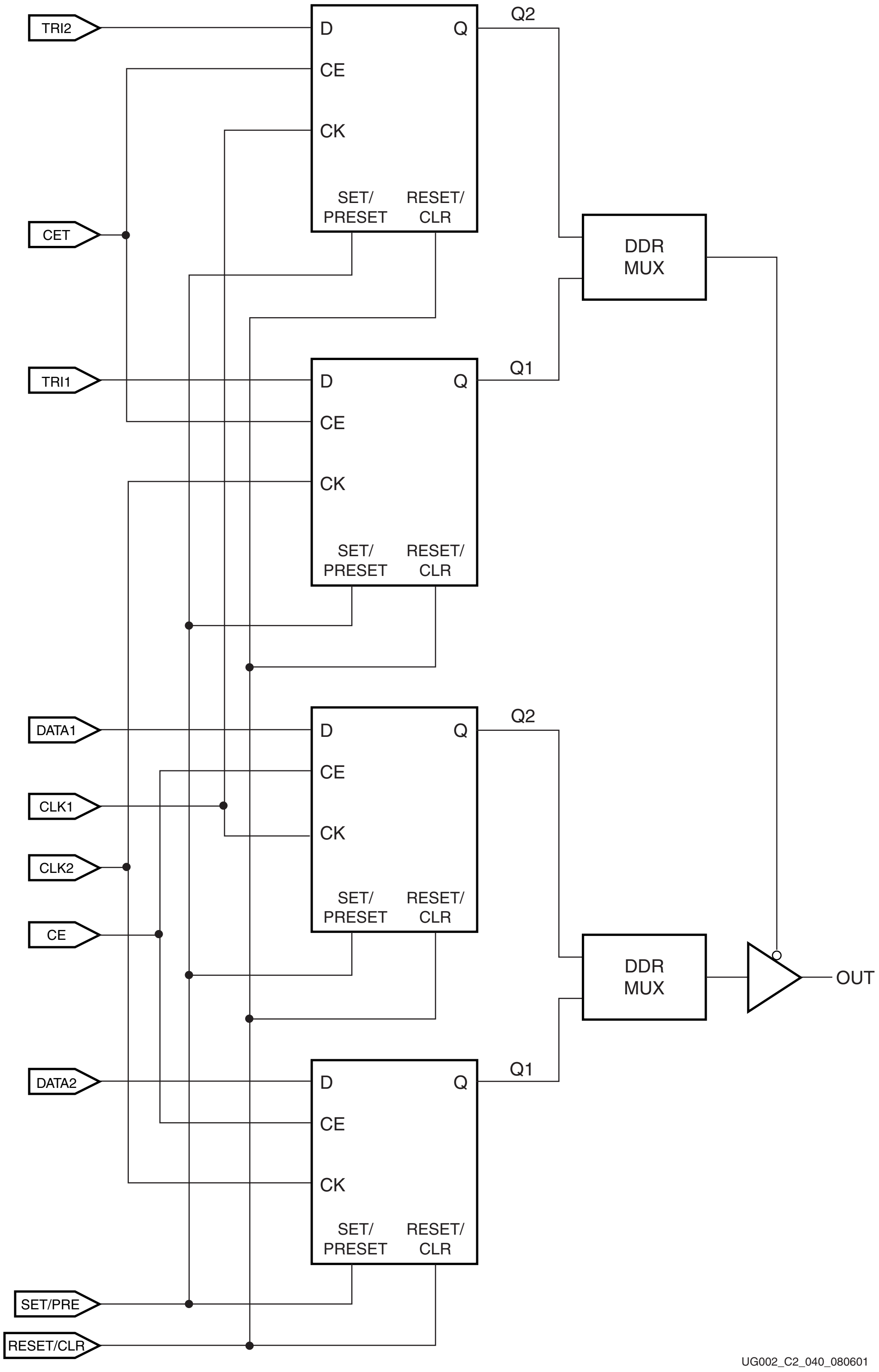


Figure 2-112: **Output DDR With 3-State Control**

All four registers share the SET/PRESET and RESET/CLEAR lines. Two registers are required to accomplish the DDR task and two registers are required for the 3-state control. There are two Clock Enable signals, one for output DDRs performing the DDR function and another for the output DDRs performing the 3-state control function. Two 180 degree out of phase clocks are used. CLK1 clocks one of the DDR registers and a 3-state register. CLK2 clocks the other DDR register and the other 3-state register.

The DDR registers and 3-state registers are associated by the clock that is driving them. Therefore, the DDR register that is clocked by CLK1 is associated to the 3-state register being clocked by CLK1. The remaining two registers are associated by CLK2. If both 3-state registers are driving a logic High, the output sees a high impedance. If both 3-state registers are driving a logic Low, the output sees the values from the DDR MUX see Figure 2-113).

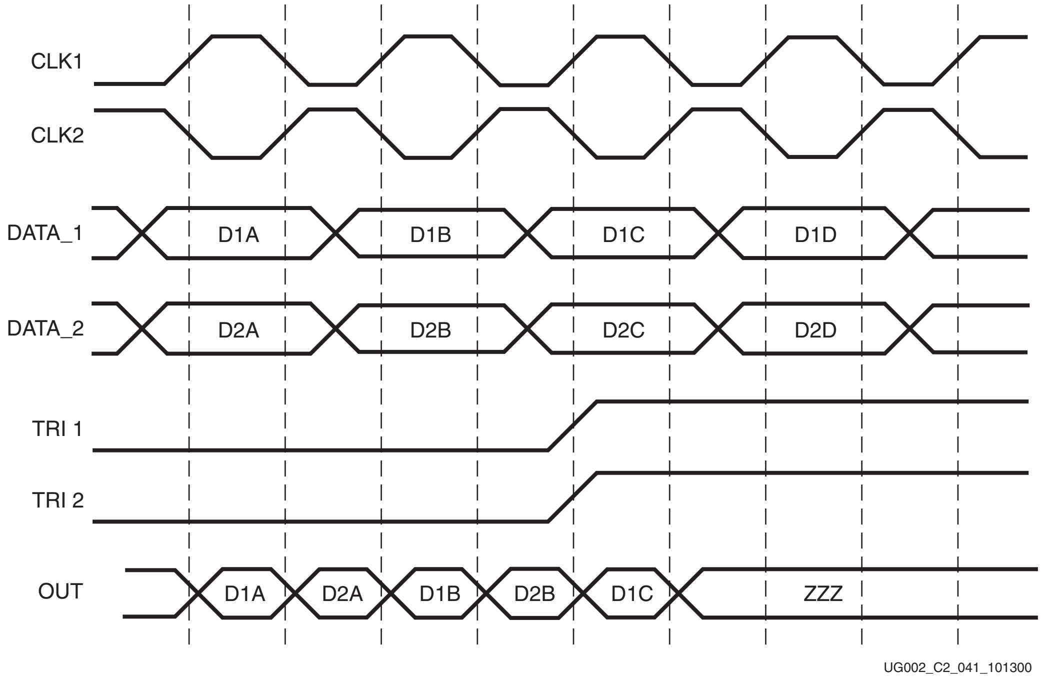


Figure 2-113: **Timing Diagram for Output DDR With 3-State Control**

When the 3-state registers are not driving the same logic value, the 3-state register being clocked by CLK1 is called TREG1. The other 3-state register TREG2 is clocked by CLK2. Similarly, the DDR register being clocked by CLK1 is called DREG1, and the other DDR register DREG2 is clocked by CLK2. If TREG1 is driving a logic High and TREG2 is driving a logic Low, the output sees a high impedance when CLK1 is High and the value out of DREG2 when CLK2 is High. If TREG2 is driving a logic High and TREG1 is driving a logic Low, the output sees a high impedance when CLK2 is High and the value out of DREG1 when CLK1 is High.

Characteristics

- All registers in an IOB share the same SET/PRE and RESET/CLR lines.
- The 3-State and Output DDR registers have common clocks (OTCLK1 & OTCLK2).
- All signals can be inverted (with no added delay) inside the IOB.
- DDR MUXing is handled automatically within the IOB. There is no manual control of the MUX-select. This control is generated from the clock.
- When several clocks are used, and when using DDR registers, the floorplan of a design should take into account that the input clock to an IOB is shared with a pair of IOBs.

Library Primitives

Input DDR registers are inferred, and dedicated output DDR registers have been provided as primitives for Virtex-II designs. Input DDR registers consist of two inferred registers that clock in a single data line on each edge. Generating 3-state output with DDR registers is as simple as instantiating a primitive.

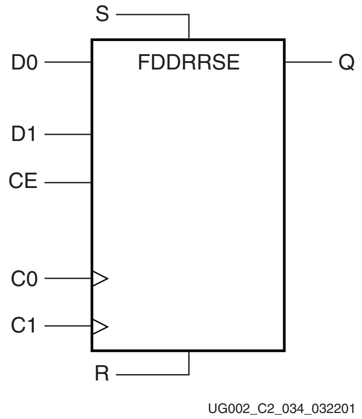


Figure 2-114: **FDDRRSE Symbol: DDR Flip-Flop With Clock Enable and Synchronous Reset and Set**

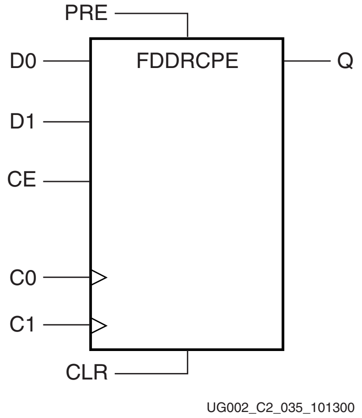


Figure 2-115: **FDDRCPE Symbol: DDR Flip-Flop With Clock Enable and Asynchronous PRESET and CLR**

VHDL and Verilog Instantiation

Examples are available in "VHDL and Verilog Templates" on page 311.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

Constraints file syntax is provided where input registers need to be used. These settings force the input DDR registers into the IOB. The output registers should be instantiated and do not require any constraints file syntax to be pushed into the IOB.

Port Signals

FDDRRSE

Data inputs - D0 and D1

D0 and D1 are the data inputs into the DDR flip-flop. Data on the D0 input is loaded into the flip-flop when R and S are Low and CE is High during a Low-to-High C0 clock transition. Data on the D1 input is loaded into the flip-flop when R and S are Low and CE is High during a Low-to-High C1 clock transition.

Clock Enable - CE

The enable pin affects the loading of data into the DDR flip-flop. When Low, new data is not loaded into the flip-flop. CE must be High to load new data into the flip-flop.

Clocks - C0 and C1

These two clocks are phase shifted 180 degrees (via the DLL) and allow selection of two separate data inputs (D0 and D1).

Synchronous Set - S and Synchronous Reset - R

The Reset (R) input, when High, overrides all other inputs and resets the output Low during any Low-to-High clock transition (C0 or C1). Reset has precedence over Set. When the Set (S) input is High and R is Low, the flip-flop is set, output High, during a Low-to-High clock transition (C0 or C1).

Data Output - Q

When power is applied, the flip-flop is asynchronously cleared and the output is Low.

During normal operation, The value of Q is either D0 or D1. The Data Inputs description above states how the value of Q is chosen.

FDDRCPE

Data inputs - D0 and D1

D0 and D1 are the data inputs into the DDR flip-flop. Data on the D0 input is loaded into the flip-flop when PRE and CLR are Low and CE is High during a Low-to-High C0 clock transition. Data on the D1 input is loaded into the flip-flop when PRE and CLR are Low and CE is High during a Low-to-High C1 clock transition.

Clock Enable - CE

The enable pin affects the loading of data into the DDR flip-flop. When Low, clock transitions are ignored and new data is not loaded into the flip-flop. CE must be High to load new data into the flip-flop.

Clocks - C0 and C1

These two clocks are phase shifted 180 degrees (via the DLL) and allow selection of two separate data inputs (D0 and D1).

Asynchronous Preset - PRE and Asynchronous Clear - CLR

The Preset (PRE) input, when High, sets the Q output High. When the Clear (CLR) input is High, the output is reset to Low.

Data Output - Q

When power is applied, the flip-flop is asynchronously cleared and the output is Low. During normal operation, The value of Q is either D0 or D1. The Data Inputs description above states how the value of Q is chosen.

Initialization in VHDL or Verilog

Output DDR primitives can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the attributes are attached to the output DDR instantiation and are copied in the EDIF output file to be compiled by Xilinx tools. The VHDL code simulation uses a `generic` parameter to pass the attributes. The Verilog code simulation uses the `defparam` parameter to pass the attributes.

The DDR code examples (in VHDL ad Verilog) illustrate the following techniques.

Location Constraints

DDR instances can have LOC properties attached to them to constrain pin placement.

The LOC constraint uses the following form.

```
NET <net_name> LOC=A8;
```

Where “A8” is a valid I/O pin location.

Applications

DDR SDRAM

The DDR SDRAM is an enhancement to the Synchronous DRAM by effectively doubling the data throughput of the memory device. Commands are registered at every positive clock edge. Input data is registered on both edges of the data strobe, and output data is referenced to both edges of the data strobe, as well as both edges of the clock.

Clock Forwarding

DDR can be used to forward a copy of the clock on the output. This can be useful for propagating a clock along with double-data-rate data that has an identical delay. It is also useful for multiple clock generation, where there is a unique clock driver for every clock load.

VHDL and Verilog Templates

VHDL and Verilog templates are available for output, output with 3-state enable, and input DDR registers.

Input DDR

To implement an Input DDR application, paste the following template in your code.

DDR_input.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DDR_Input is

  Port (
    clk : in std_logic;
    d : in std_logic;
    rst : in std_logic;
    q1 : out std_logic;
    q2 : out std_logic
        );

end DDR_Input;

--Describe input DDR registers (behaviorally) to be inferred

architecture behavioral of DDR_Input is
```

```
begin

q1reg : process (clk, d, rst)

begin
   if rst='1' then --asynchronous reset, active high
     q1 <= '0';
   elsif clk'event and clk='1' then  --Clock event - posedge
     q1 <= d;

   end if;
end process;

q2reg : process (clk, d, rst)

begin
   if rst='1' then --asynchronous reset, active high
      q2 <= '0';
   elsif clk'event and clk='0' then  --Clock event - negedge
      q2 <= d;
   end if;
end process;

end behavioral;


-- NOTE: You must include the following constraints in the .ucf
-- file when running back-end tools,
-- in order to ensure that IOB DDR registers are used:
--
-- INST "q2_reg" IOB=TRUE;
-- INST "q1_reg" IOB=TRUE;
--
-- Depending on the synthesis tools you use, it may be required to
-- check the edif file for modifications to
-- original net names...in this case, Synopsis changed the
-- names: q1 and q2 to q1_reg and q2_reg
```

DDR_input.v

```
module DDR_Input (data_in , q1, q2, clk, rst);

input data_in, clk, rst;
output q1, q2;
reg q1, q2;


//Describe input DDR registers (behaviorally) to be inferred

always @ (posedge clk or posedge rst) //rising-edge DDR reg. and
asynchronous reset

  begin
  if (rst)
     q1 = 1'b0;
  else
     q1 = data_in;
  end
```

```
always @ (negedge clk or posedge rst) //falling-edge DDR reg. and
asynchronous reset

begin
  if (rst)
     q2 = 1'b0;
  else
     q2 = data_in;
  end

assign data_out = q1 & q2;



endmodule

/* NOTE: You must include the following constraints in the .ucf file
when running back-end tools, \
  in order to ensure that IOB DDR registers are used:

INST "q2_reg" IOB=TRUE;
INST "q1_reg" IOB=TRUE;

Depending on the synthesis tools you use, it may be required to check
the edif file for modifications to
original net names...in this case, Synopsis changed the names: q1 and q2
to q1_reg and q2_reg

*/
```

Output DDR

To implement an Output DDR application, paste the following template in your code.

DDR_out.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- pragma translate_off
LIBRARY UNISIM;
use UNISIM.VCOMPONENTS.ALL;
--pragma translate_on

entity DDR_Output is
  Port(
   clk : in std_logic; --clk and clk180 can be outputs from the DCM or
clk180 can be the
   clk180 : in std_logic;  --logical inverse of clk (the inverter is
located in the IOB and will be inferred.
   d0 : in std_logic;  --data in to fddr
   d1 : in std_logic;  --data in to fddr
   ce : in std_logic;  --clock enable
   rst : in std_logic;  --reset
   set : in std_logic;  --set
   q : out std_logic  --DDR output
   );

end DDR_Output;

architecture behavioral of DDR_Output is

component FDDRRSE
  port(
```

```
        Q   : out std_logic;
        D0  : in std_logic;
        D1  : in std_logic;
        C0  : in std_logic;
        C1  : in std_logic;
        CE  : in std_logic;
        R   : in std_logic;
        S : in std_logic
        );
    end component;

    begin

    U0: FDDRRSE
      port map (
        Q => q,
        D0  => d0,
        D1  => d1,
        C0  => clk,
        C1  => clk180,
        CE  => ce,
        R => rst,
        S => set
        );


    end behavioral;
```

DDR_out.v

```
module DDR_Output (d0 , d1, q, clk, clk180, rst, set, ce);

input d0, d1, clk, clk180, rst, set, ce;
output q;


//Synchronous Output DDR primitive instantiation

FDDRRSE U1  ( .D0(d0),
              .D1(d1),
              .C0(clk),
              .C1(clk180),
              .CE(ce),
              .R(rst),
              .S(set),
              .Q(q)
             );
endmodule
```

Output DDR With 3-State Enable

To implement an Output DDR with 3-state Enable, paste the following template in your code:

DDR_3state.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- pragma translate_off
LIBRARY UNISIM;
use UNISIM.VCOMPONENTS.ALL;
--pragma translate_on
```

```
entity DDR_3state is
  Port(
    clk : in std_logic; --clk and clk180 can be outputs from the DCM or
clk180 can be the
    clk180 : in std_logic; --logical inverse of clk (the inverter is
located in the IOB and will be inferred.
    d0 : in std_logic; --data in to fddr
    d1 : in std_logic; --data in to fddr
    ce : in std_logic; --clock enable
    set : in std_logic; --set
    rst : in std_logic; --reset
    en0 : in std_logic; --enable signal
    en1 : in std_logic; --enable signal
    data_out : out std_logic --data seen at pad
  );

end DDR_3state;

architecture behavioral of DDR_3state is

signal ddr_out, tri : std_logic;

component FDDRRSE
   port  (
    Q   : out std_logic;
    D0  : in std_logic;
    D1  : in std_logic;
    C0  : in std_logic;
    C1  : in std_logic;
    CE  : in std_logic;
    R   : in std_logic;
    S   : in std_logic
         );
end component;


begin

--Instantiate Ouput DDR registers
U0: FDDRRSE port map(Q  => tri,
        D0 => en0,
        D1 => en1,
        C0 => clk,
        C1 => clk180,
        CE => ce,
        R  => rst,
        S  => set
       );

--Instantiate three-state DDR registers
U1: FDDRRSE port map(   Q   => ddr_out,
        D0 => d0,
        D1 => d1,
        C0 => clk,
        C1 => clk180,
        CE => ce,
        R  => rst,
        S  => set
       );

--inferr the 3-State buffer
process(tri, ddr_out)
```

```
begin
  if tri = '1' then
     data_out <= 'Z';
  elsif tri = '0' then
     data_out <= ddr_out;
  end if;
end process;

end behavioral;
```

DDR_3state.v

```
module DDR_3state (d0 , d1, data_out, en_0, en_1, clk, clk180, rst, set,
ce);

input d0, d1, clk, clk180, rst, set, ce, en_0, en_1;

output data_out;
reg data_out;

wire q, q_tri;

//Synchronous Output DDR primitive instantiation

FDDRRSE U1  ( .D0(d0),
              .D1(d1),
              .C0(clk),
              .C1(clk180),
              .CE(ce),
              .R(rst),
              .S(set),
              .Q(q)
            );

//Synchronous 3-State DDR primitive instantiation

FDDRRSE U2  ( .D0(en_0),
              .D1(en_1),
              .C0(clk),
              .C1(clk180),
              .CE(ce),
              .R(rst),
              .S(set),
              .Q(q_tri)
            );

//3-State buffer description

always @ (q_tri or q)
  begin
  if (q_tri)
    data_out = 1'bz;
  else
    data_out = q;
  end

endmodule
```

Using LVDS I/O

Introduction

Low Voltage Differential Signaling (LVDS) is a very popular and powerful high-speed interface in many system applications. Virtex-II I/Os are designed to comply with the IEEE electrical specifications for LVDS to make system and board design easier. With the addition of an LVDS current-mode driver in the IOBs, which eliminates the need for external source termination in point-to-point applications, and with the choice of two different voltage modes and an extended mode, Virtex-II devices provide the most flexible solution for doing an LVDS design in an FPGA.

Table 2-59 lists all LVDS primitives that are available for Virtex-II devices.

Table 2-59: **Available Virtex-II LVDS Primitives**

Input	Output	3-State	Clock	Bi-Directional
IBUF_LVDS	**OBUF_LVDS**	**OBUFT_LVDS**	**IBUFG_LVDS**	**IOBUF_LVDS**
IBUFDS_LVDS_25	OBUFDS_LVDS_25	OBUFTDS_LVDS_25	IBUFGDS_LVDS_25	
IBUFDS_LVDS_33	OBUFDS_LVDS_33	OBUFTDS_LVDS_33	IBUFGDS_LVDS_33	
IBUFDS_LVDSEXT_25	OBUFDS_LVDSEXT_25	OBUFTDS_LVDSEXT_25	IBUFGDS_LVDSEXT_25	
IBUFDS_LVDSEXT_33	OBUFDS_LVDSEXT_33	OBUFTDS_LVDSEXT_33	IBUFGDS_LVDSEXT_33	

The primitives in **bold** type are pre-existing LVDS primitives used in Virtex-E and earlier designs. They are not current-mode drivers and are still required for BLVDS (Bus LVDS) applications.

*DS_LVDS_25 = 2.5V V_{CCO} LVDS Buffer

*DS_LVDS_33 = 3.3V V_{CCO} LVDS Buffer

There is no difference in the AC characteristics of either voltage-mode LVDS I/O. These choices now provide more flexibility for mixed-I/O banking rules; that is, an LVTTL I/O can coexist with the 3.3V LVDS buffer in the same bank.

DS_LVDSEXT = Extended mode LVDS buffer

This buffer provides a higher drive capability and voltage swing (350 - 750 mV), which makes it ideal for long-distance or cable LVDS links.

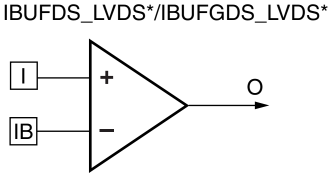
The output AC characteristics of this LVDS driver are not within the EIA/TIA specifications. This LVDS driver is intended for situations that require higher drive capabilities in order to produce an LVDS signal that is within EIA/TIA specification at the receiver.

Creating an LVDS Input/Clock Buffer

Figure 2-116 illustrates the LVDS input and clock buffer primitives shown in Table 2-60. The pin names used are the same as those used in the HDL library primitives.

Table 2-60: **LVDS Input and Clock Buffer Primitives**

LVDS Inputs	LVDS Clocks
IBUFDS_LVDS_25	IBUFGDS_LVDS_25
IBUFDS_LVDS_33	IBUFGDS_LVDS_33
IBUFDS_LVDSEXT_25	IBUFGDS_LVDSEXT_25
IBUFDS_LVDSEXT_33	IBUFGDS_LVDSEXT_33



UG002_C2_031_100200

Figure 2-116: **LVDS Input and Clock Primitives**

To create an LVDS input, instantiate the desired mode (2.5 V, 3.3 V, or Extended) LVDS input buffer. Notice that the P and N channels are included in the primitive (I = P, IB = N). Software automatically uses the appropriate pin from an adjacent IOB for the N channel. The same applies to LVDS clocks: Use IBUFGDS_LVDS*

LVDS Input HDL Examples

VHDL Instantiation

```
U1: IBUFDS_LVDS_25
  port map  (
                I => data_in_P,
                IB => data_in_N
                O => data_in
              );
```

Verilog Instantiation

```
IBUFDS_LVDS_25   U1   ( .I(data_in_P),
                         .IB(data_in_N),
                         .O(data_in)
                       );
```

Port Signals

I = P-channel data input to the LVDS input buffer

IB = N-channel data input to the LVDS input buffer

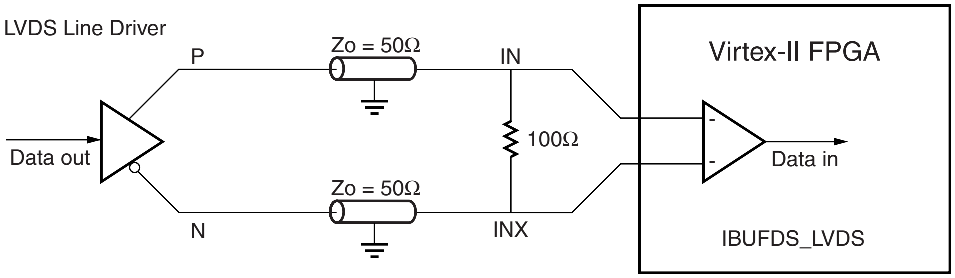
O = Non-differential input data from LVDS input buffer

Location Constraints

```
NET "data_in_P" LOC= "NS";
```

LVDS Receiver Termination

All LVDS receivers require standard termination. Figure 2-117 is an example of a typical termination for an LVDS receiver on a board with 50Ω transmission lines.



UG002_C2_028_100300

Figure 2-117: **LVDS Receiver Termination**

Creating an LVDS Output Buffer

Figure 2-118 illustrates the LVDS output buffer primitives:

- OBUFDS_LVDS_25
- OBUFDS_LVDS_33
- OBUFDS_LVDSEXT_25
- OBUFDS_LVDSEXT_33

The pin names used are the same as those used in the HDL library primitives.

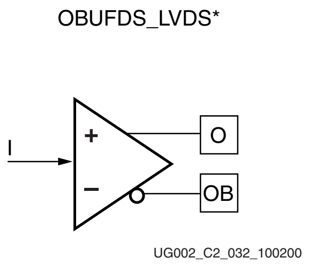


Figure 2-118: **LVDS Output Buffer Primitives**

To create an LVDS output, instantiate the desired mode (2.5, 3.3V, or Extended) LVDS output buffer. Notice that the P and N channels are included in the primitive (O = P, OB = N). Software automatically uses the appropriate pin from an adjacent IOB for the N channel.

LVDS Output HDL Examples

VHDL Instantiation

```
U1: OBUFDS_LVDS_25
  port map  (
               I => data_out,
               O => data_out_P,
               OB => data_out_N
             );
```

Verilog Instantiation

```
OBUFDS_LVDS_25 U1  ( .I(data_out),
                     .O(data_out_P),
                     .OB(data_out_N)
                   );
```

Port Signals

I = data input to the LVDS input buffer

O = P-channel data output

OB = N-channel data output

Location Constraints

```
NET "data_out_P" LOC= "NS";
```

LVDS Transmitter Termination

The Virtex-II LVDS transmitter does not require any termination. Table 2-59 lists primitives that correspond to the Virtex-II LVDS current-mode drivers. Virtex-II LVDS current-mode drivers are a true current source and produce the proper (IEEE/EIA/TIA compliant) LVDS signal. Figure 2-119 illustrates a Virtex-II LVDS transmitter on a board with 50Ω transmission lines.

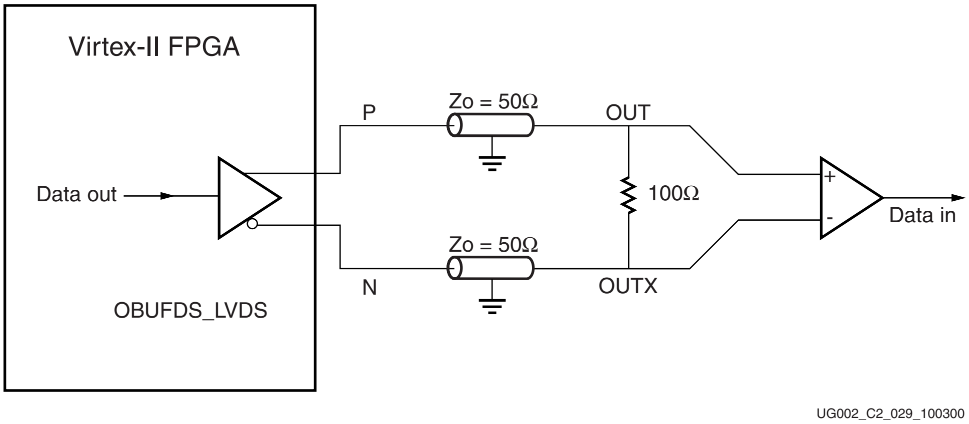


Figure 2-119: **LVDS Transmitter Termination**

Creating an LVDS Output 3-State Buffer

Figure 2-120 illustrates the LVDS 3-State buffer primitives:

- OBUFTDS_LVDS_25
- OBUFTDS_LVDS_33
- OBUFTDS_LVDSEXT_25
- OBUFTDS_LVDSEXT_33

The pin names used are the same as those used in the HDL library primitives.

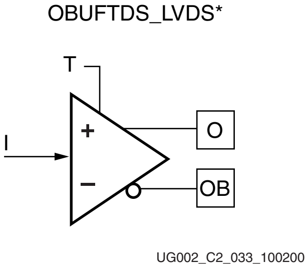


Figure 2-120: **LVDS 3-State Primitives**

To create an LVDS 3-State output, instantiate the desired mode (2.5V, 3.3V, or Extended) LVDS 3-State buffer. Notice that the P and N channels are included in the primitive (O = P, OB = N). Software automatically uses the appropriate pin from an adjacent IOB for the N channel.

LVDS 3-State HDL Example

VHDL Instantiation

```
U1: OBUFTDS_LVDS_25
  port map  (
                I => data_out,
                T => tri,
                O => data_out_P,
                OB => data_out_N
              );
```

Verilog Instantiation

```
OBUFTDS_LVDS_25  U1  (  .I(data_out),
                        .T(tri),
                        .O(data_out_P),
                        .OB(data_out_N)
                     );
```

Port Signals

`I` = data input to the 3-state output buffer

T = 3-State control signal

O = P-channel data output

OB = N-channel data output

Location Constraints

```
NET “data_out_P” LOC = “NS”;
```

LVDS 3-State Termination

The Virtex-II LVDS 3-state buffer does not require any termination. Table 2-59 lists primitives that correspond to Virtex-II LVDS current-mode drivers. These drivers are a true current source, and they produce the proper (IEEE/EIA/TIA compliant) LVDS signal. Figure 2-121 illustrates a simple redundant point-to-point LVDS solution with two LVDS 3-state transmitters sharing a bus with one LVDS receiver and the required termination for the circuit.

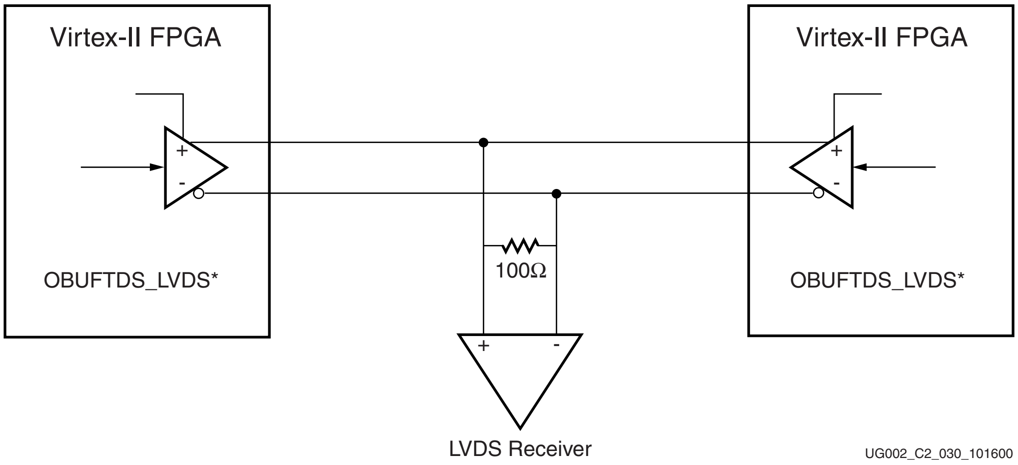


Figure 2-121: **LVDS 3-State Termination**

Creating a Bidirectional LVDS Buffer

Since LVDS is intended for point-to-point applications, BLVDS (Bus-LVDS) is not an IEEE/EIA/TIA standard implementation and requires careful adaptation of I/O and PCB layout design rules. The primitive supplied in the software library for bi-directional LVDS does not use the Virtex-II LVDS current-mode driver. Therefore, source termination is required. Refer to xapp243 for examples of BLVDS termination.

The following are VHDL and Verilog instantiation examples of Virtex-II BLVDS primitves.

VHDL Instantiation

```
blvds_io: IOBUFDS_BLVDS_25
port map  (
           I => data_out,
           O => data_in,
           T => tri,
           IO => data_IO_P,
           IOB => data_IO_N
          );
```

Verilog Instantiation

```
IOBUFDS_BLVDS_25  blvds_io  (  .I(data_out),
                               .O(data_in),
                               .T(tri),
                               .IO(data_IO_P),
                               .IOB(data_IO_N)
                            );
```

Port Signals

I = data output: internal logic to LVDS I/O buffer

T = 3-State control to LVDS I/O buffer

IO = P-channel data I/O to or from BLVDS pins

IOB = N-channel data I/O to or from BLVDS pins

O = Data input: off-chip data to LVDS I/O buffer

Location Constraints

Only the P or N channel must be constrained. Software automatically places the corresponding channel of the pair on the appropriate pin.

LDT

Lightning Data Transport (LDT) is a new high speed interface and protocol introduced by Advanced Micro Devices. LDT is a differential signaling based interface that is very similar to LVDS. Virtex-II IOBs are equipped with LDT buffers. These buffers also have corresponding software primitives as follows:

IBUFDS_LDT_25

IBUFGDS_LDT_25

OBUFDS_LDT_25

OBUFTDS_LDT_25

LDT Implementation

LDT implementation is the same as LVDS with DDR, so follow all of the rules and guidelines set forth earlier in this chapter for LVDS-DDR, and replace the LVDS buffer with the corresponding LDT buffer. For more information on Virtex-II LDT electrical specification, refer to the **Virtex-II Data Sheet**.

Using Bitstream Encryption

Virtex-II devices have an on-chip decryptor that can be enabled to make the configuration bitstream (and thus the whole logic design) secure. The user can encrypt the bitstream in the Xilinx software, and the Virtex-II chip then performs the reverse operation, decrypting the incoming bitstream, and internally recreating the intended configuration.

This method provides a very high degree of design security. Without knowledge of the encryption/decryption key or keys, potential pirates cannot use the externally intercepted bitstream to analyze, or even to clone the design. System manufacturers can be sure that their Virtex-II implemented designs cannot be copied and reverse engineered. Also, IP Virtex-II chips that contain the correct decryption key.

The Virtex-II devices store the internal decryption keys in a few hundred bits of dedicated RAM, backed up by a small externally connected battery. At <100 nA load, the endurance of the battery is only limited by its shelf life.

The method used to encrypt the data is Data Encryption Standard (DES). This is an official standard supported by the National Institute of Standards and Technology (NIST) and the U. S. Department of Commerce. DES is a symmetric encryption standard that utilizes a 56-bit key. Because of the increased sophistication and speed of today's computing hardware, single DES is no longer considered to be secure. However, the Triple Data Encryption Algorithm (TDEA), otherwise known as triple DES, is authorized for use by U. S. federal organizations to protect sensitive data and is used by many financial institutions to protect their transactions. Triple DES has yet to be cracked. Both DES and triple DES are available in Virtex-II devices.

What DES Is

DES and triple DES are symmetric encryption algorithms. This means that the key to encrypt and the key to decrypt are the same. The security of the data is kept by keeping the key secret. This contrasts to a public key system, like RSA or PGP. One thing to note is that Virtex-II devices use DES in Cipher Block Chaining mode. This means that each block is combined with the previous encrypted block for added security. DES uses a single 56-bit key to encrypt 64-bit blocks one at a time.

How Triple DES is Different

Triple DES uses three keys (known as a key bundle or key set), and the encryption algorithm is repeated for each of those keys. If $E_K(I)$ and $D_K(I)$ denote the encryption and decryption of a data block I using key K, the Triple DES encryption algorithm is as follows (known as E-D-E):

$$\text{Output}_{\text{encrypted}} = E_{K3}(D_{K2}(E_{K1}(I)))$$

And the decryption algorithm is as follows (known as D-E-D):

$$\text{Output}_{\text{decrypted}} = D_{K1}(E_{K2}(D_{K3}(I)))$$

$K_1 = K_2 = K_3$ gives the same result as single DES.

For a detailed description of the DES standard, refer to:

http://www.itl.nist.gov/fipspubs/fip46-2.htm

For a popular description of the origin and the basic concept of DES and many other older and newer encryption schemes, see the recent best-seller:

The Code Book by Simon Singh, Doubleday 1999, ISBN 0-385-49531-5

Classification and Export Considerations

Virtex-II FPGAs have been classified by the U. S. Department of Commerce as an FPLD (3A001.a.7), which is the same classification as current FPGAs. Only the decryptor is on-chip and can only be used to decrypt an incoming bitstream, so the classification has not changed and no new paperwork is required. The software has been classified under ECCN#:5D002 and can be exported globally under license exception ENC. No changes to current export practices are necessary.

Creating Keys

For Virtex-II, DES or triple DES (TDEA) can be used. DES uses a single 56-bit key, where triple-DES always uses three such keys. All of the keys can be chosen by the BitGen program at random, or can be explicitly specified by the user.

Virtex-II devices can have six separate keys programmed into the device. A particular Virtex-II device can store two sets of triple-DES keys and can thus accept alternate bitstreams from two competing IP vendors, without providing access to each other's design. However, all of the keys must be programmed at once.

An encrypted bitstream is created by the BitGen program. Keys and key options can be chosen in two ways: by command-line arguments to BitGen, or by specifying a KeyFile (with the –g KeyFile command-line option). The BitGen options relevant to encryption are listed in Table 2-61.:

Table 2-61: **BitGen Encryption Options**

Option	Description	Values (default first where appropriate)
Encrypt	Whether to encrypt the bitstream	No, Yes
Key0	DES Key 0	pick, <hex string>
Key1	DES Key 1	pick, <hex string>
Key2	DES Key 2	pick, <hex string>
Key3	DES Key 3	pick, <hex string>
Key4	DES Key 4	pick, <hex string>
Key5	DES Key 5	pick, <hex string>
KeyFile	Location of separate key definition file	<string>
Keyseq0	Set the key sequence for key 0 (S = single, F = first, M = middle, L = last)	S,F,M,L
Keyseq1	Set the key sequence for key 1	S,F,M,L
Keyseq2	Set the key sequence for key 2	S,F,M,L
Keyseq3	Set the key sequence for key 3	S,F,M,L
Keyseq4	Set the key sequence for key 4	S,F,M,L
Keyseq5	Set the key sequence for key 5	S,F,M,L
StartKey	Key number to start decryption	0,3
StartCBC	Constant Block Chaining start value	pick, <string>

The key sequence (Keyseq) is set to S for single key encryption, F for first key in multi-key encryption, M for middle key in multi-key encryption, and L for last key in multi-key encryption. When the KeyFile option is specified, BitGen looks in that file for all other DES key options listed above. An example for the input KeyFile using triple DES is:

```
# Comment for key file
Key 0 0x9ac28ebeb2d83b;
Key 1 pick;
Key 2 string for my key;
Key 3 0x00000000000000;
Key 4 8774eb3ebb4f84;
Keyseq 0 F;
Keyseq 1 M;
Keyseq 2 L;
Keyseq 3 F;
Keyseq 4 M;
Keyseq 5 L;
Key StartCBC 503f2f655b1b2f82;
StartKey 0;
```

Every key is given in the output key file, with unused key locations set to "0x00000000000000." The proper key sequence prefix is added for all used keys. The prefix is preserved for unused keys, if the user specified a value. The output key file has the same base file name as the **.bit** file, but with a **.nky** file extension.

The command line equivalent of the input key file above is as follows:

```
bitgen –g Encrypt:Yes –g Key0: 0x9ac28ebeb2d83b –g Key1:pick –g Key2:”
string for my key” –g Key30x00000000000000 –g Key4:8774eb3ebb4f84 –g
Keyseq0:F, -g Keyseq1:M, -gKeyseq2:L –g Keyseq3:F –g Keyseq4:M –g
Keyseq5:L –g StartCBC:503f2f655b1b2f82 –g StartKey:0 myinput.ncd
```

If the key file is used, the command line is as follows:

```
Bitgen –g Encrypt:Yes –g KeyFile: mykeyfile myinput.ncd
```

The output key file from either of the above inputs looks something like this:

```
Device 2v40CS144;
Key 0 0x9ac28ebeb2d83b;
Key 1 0xdb1adb5f08b972;
Key 2 0x5452032773c286;
Key 3 0x00000000000000;
Key 4 0x8774eb3ebb4f84;
Key 5 0x00000000000000;
Keyseq 0 F;
Keyseq 1 M;
Keyseq 2 L;
Keyseq 3 F;
Keyseq 4 M;
Keyseq 5 L;
Key StartCBC 0x503f2f655b1b2f82;
StartKey 0;
```

In the case of the string for Key2, if the keyvalue is a character string, BitGen encodes the string into a 56-bit hex string. The same character string gives the same 56-bit hex string every time. This enables passwords or phrases to be used instead of hex strings.

The above keys are all specified as 64 bits each. The first 8 bits are used by Xilinx as header information and the following 56 bits as the key. BitGen accepts 64 bit keys, but automatically overrides the header, if necessary.

Because of security issues, the **–g** Compress option cannot be used with bitstream encryption. Also, partial reconfiguration is not allowed.

Loading Keys

DES keys can only be loaded through JTAG. The JTAG Programmer and iMPACT™ tools have the capability to take a .nky file and program the device with the keys. In order to program the keys, a “key-access mode” is entered. When this mode is entered, all of the FPGA memory, including the keys and configuration data, is cleared. Once the keys are programmed, they cannot be reprogrammed without clearing the entire device. This “key access mode” is completely transparent to most users.

Keys are programmed using the ISC_PROGRAM instruction, as detailed in the JTAG 1532 specification. SVF generation is also supported, if keys are to be programmed using a different method, such as a microprocessor or JTAG test software.

Loading Encrypted Bitstreams

Once the device has been programmed with the correct keys, the device can be configured with an encrypted bitstream. Non-encrypted bitstreams may also be used to configure the device, and the stored keys are ignored. The method of configuration is not at all affected by encryption. Any of the modes may be used, and the signaling does not change (refer to Chapter 3: Configuration). However, *all* bitstreams must configure the entire device, since partial reconfiguration is not permitted.

Once the device has been configured with an encrypted bitstream, it cannot be reconfigured without toggling the PROG pin, cycling power, or performing the JTAG JSTART instruction. All of these events fully clear the configuration memory, but none of these events reset the keys as long as V_{BATT} or V_{CCAUX} are maintained.

V_{BATT}

V_{BATT} is a separate battery voltage to allow the keys to remain programmed in the Virtex-II device. V_{BATT} draws very little current (on the order of nA) to keep the keys programmed. A small watch battery is suitable (refer to V_{BATT} DC Characteristics in the **Virtex-II Data Sheet** and the battery’s specifications to estimate its lifetime).

While the auxiliary voltage (V_{CCAUX}) is applied, V_{BATT} does not draw any current, and the battery can be removed or exchanged.

Using the CORE Generator System

Introduction

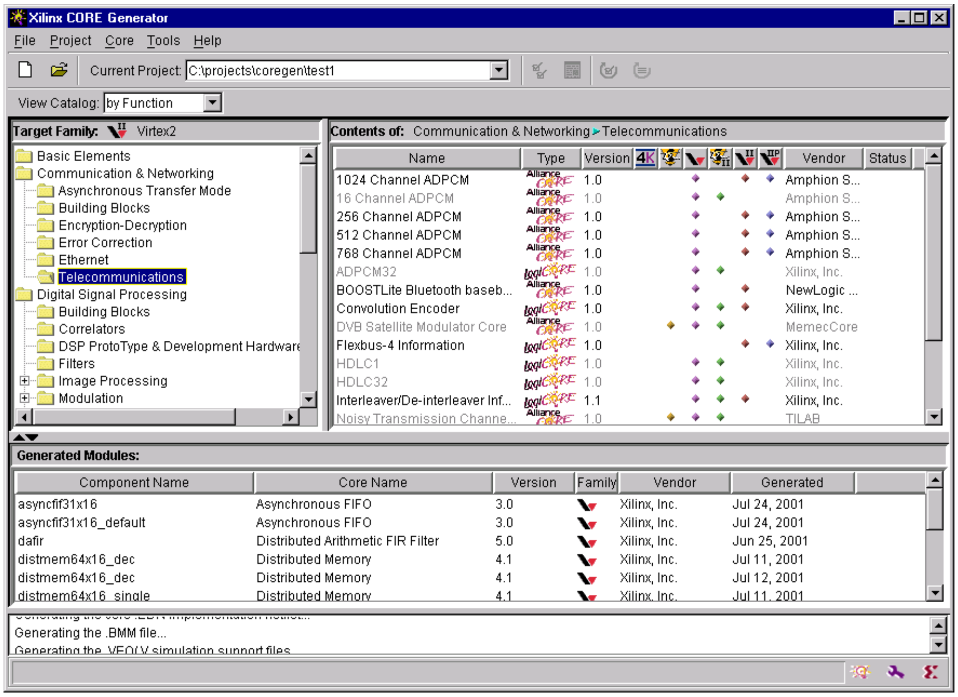
This section on the Xilinx CORE Generator System™ and the Xilinx Intellectual Property (IP) Core offerings is provided as an overview of products that facilitate the Virtex-II design process. For more detailed and complete information, consult the *CORE Generator Guide*, which can be accessed online in the Xilinx software installation, as well as at the **http://toolbox.xilinx.com/docsan/xilinx4/manuals.htm** site, under the "Design Entry Tools" heading.

The CORE Generator System

The Xilinx CORE Generator System is the cataloging, customization, and delivery vehicle for IP cores targeted to Xilinx FPGAs. This tool is included with all Xilinx ISE BaseX, ISE Foundation, and ISE Alliance Series software packages. The CORE Generator provides centralized access to a catalog of ready-made IP functions ranging in complexity from simple arithmetic operators, such as adders, accumulators, and multipliers, to system-level building blocks, such as filters, transforms, and memories. Cores can be displayed alphabetically, by function, by vendor, or by type. Each core comes with its own data sheet, which documents the core's functionality in detail.

The CORE Generator User Interface (see Figure 2-122) has direct links to key Xilinx web support pages, such as the Xilinx IP Center website (**www.xilinx.com/ipcenter**) and Xilinx Technical Support, making it very easy to access the latest Virtex-II IP releases and get helpful, up-to-date specifications and information on technical issues. Links to partner IP providers are also built into the informational GUIs for the various partner-supplied AllianceCORE products described under "AllianceCORE Program" on page 331.

The use of CORE Generator IP cores in Virtex-II designs enables designers to shorten design time, and it also helps them realize high levels of performance and area efficiency without any special knowledge of the Virtex-II architecture. The IP cores achieve these high levels of performance and logic density by using Xilinx Smart-IP™ technology.



Current Project: C:\projects\coregen\test1

View Catalog: by Function

Target Family: Virtex2

- Basic Elements
- Communication & Networking
  - Asynchronous Transfer Mode
  - Building Blocks
  - Encryption-Decryption
  - Error Correction
  - Ethernet
  - Telecommunications
- Digital Signal Processing
  - Building Blocks
  - Correlators
  - DSP ProtoType & Development Hardware
  - Filters
  - Image Processing
  - Modulation

Contents of: Communication & Networking ▸ Telecommunications

Name	Type	Version	4K						Vendor	Status
1024 Channel ADPCM	AllianceCORE	1.0			◆		◆	◆	Amphion S...	
16 Channel ADPCM	AllianceCORE	1.0			◆	◆			Amphion S...	
256 Channel ADPCM	AllianceCORE	1.0			◆		◆	◆	Amphion S...	
512 Channel ADPCM	AllianceCORE	1.0			◆		◆	◆	Amphion S...	
768 Channel ADPCM	AllianceCORE	1.0			◆		◆	◆	Amphion S...	
ADPCM32	LogiCORE	1.0			◆	◆			Xilinx, Inc.	
BOOSTLite Bluetooth baseb...	AllianceCORE	1.0			◆		◆		NewLogic ...	
Convolution Encoder	LogiCORE	1.0			◆	◆	◆		Xilinx, Inc.	
DVB Satellite Modulator Core	AllianceCORE	1.0		◆	◆	◆			MemecCore	
Flexbus-4 Information	LogiCORE	1.0					◆	◆	Xilinx, Inc.	
HDLC1	LogiCORE	1.0			◆	◆			Xilinx, Inc.	
HDLC32	LogiCORE	1.0			◆	◆			Xilinx, Inc.	
Interleaver/De-interleaver Inf...	LogiCORE	1.1			◆	◆	◆		Xilinx, Inc.	
Noisy Transmission Channe...	AllianceCORE	1.0		◆	◆	◆			TILAB	

Generated Modules:

Component Name	Core Name	Version	Family	Vendor	Generated
asyncfif31x16	Asynchronous FIFO	3.0		Xilinx, Inc.	Jul 24, 2001
asyncfif31x16_default	Asynchronous FIFO	3.0		Xilinx, Inc.	Jul 24, 2001
dafir	Distributed Arithmetic FIR Filter	5.0		Xilinx, Inc.	Jun 25, 2001
distmem64x16_dec	Distributed Memory	4.1		Xilinx, Inc.	Jul 11, 2001
distmem64x16_dec	Distributed Memory	4.1		Xilinx, Inc.	Jul 12, 2001
distmem64x16_single	Distributed Memory	4.1		Xilinx, Inc.	Jul 11, 2001

```
Generating the .BMM file...
Generating the .VEO/.V simulation support files
```

ug002_c2_068b_100901

Figure 2-122: **Core Generator User Interface**

Smart-IP Technology

Smart-IP technology leverages Xilinx FPGA architectural features, such as look-up tables (LUTs), distributed RAM, segmented routing and floorplanning information, as well as relative location constraints and expert logic mapping to optimize the performance of every core instance in a given Xilinx FPGA design. In the context of Virtex-II cores, Smart-IP technology includes the use of the special high-performance Virtex-II architectural features, such as embedded 18x18 multipliers, block memory, shift register look-up tables (SRL16's), and special wide mux elements.

Smart-IP technology delivers:

- Physical layouts optimized for high performance
- Predictable high performance and efficient resource utilization
- Reduced power requirements through compact design and interconnect minimization
- Performance independent of device size
- Ability to use multiple cores without deterioration of performance
- Reduced compile time over competing architectures

CORE Generator Design Flow

A block diagram of the CORE Generator design flow is shown in Figure 2-123.

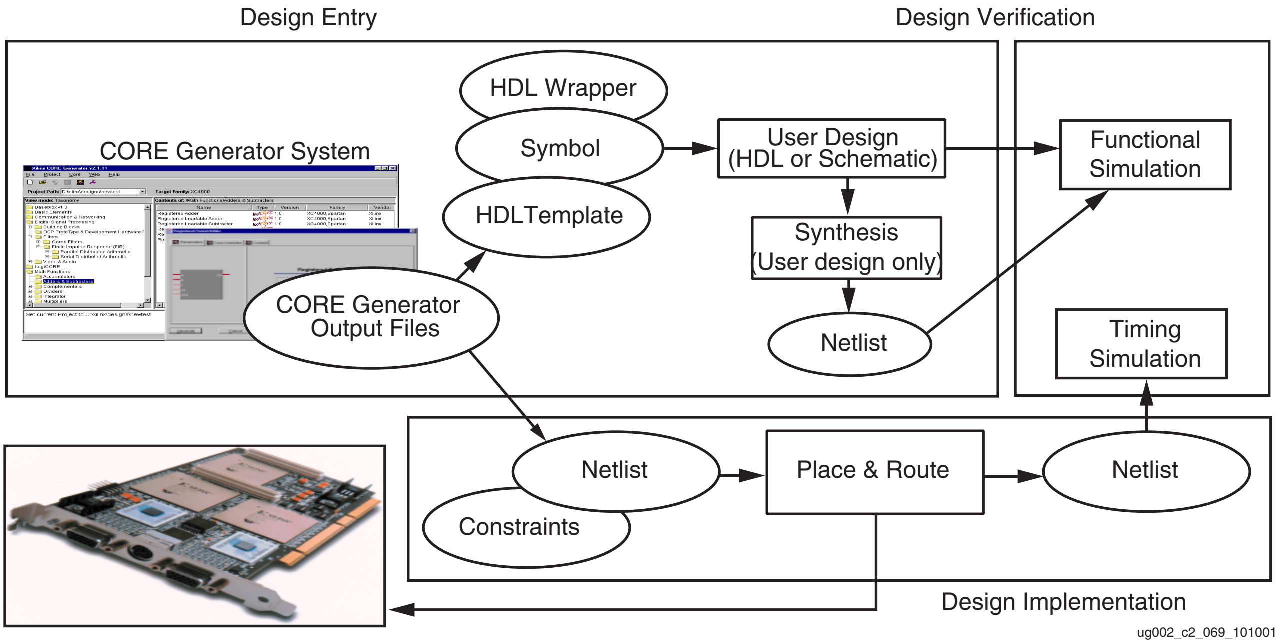


Figure 2-123: **CORE Generator Design Flow**

Note:

1. The outputs produced by the CORE Generator consist of an implementation Netlist and optional schematic symbol, HDL template files, and HDL simulation model wrapper files.

Core Types

Parameterized Cores

The CORE Generator System supplies a wide assortment of parameterized IP cores that can be customized to meet specific Virtex-II design needs and size constraints. See Figure 2-124. For each parameterized core, the CORE Generator System supplies:

- A customized EDIF implementation netlist (.EDN)

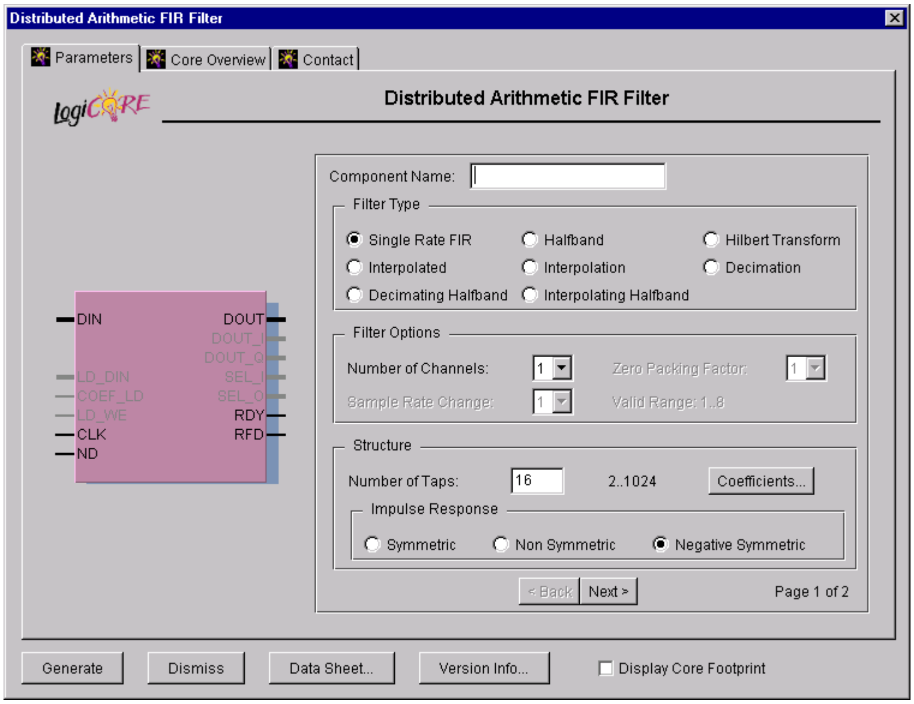
- A parameterized Verilog or VHDL behavioral simulation model (.V, .VHD) and corresponding wrapper file (also .V, .VHD)
- Verilog or VHDL templates (.VEO, .VHO)
- An ISE Foundation or Viewlogic® schematic symbol

The EDIF implementation netlist is used by the Xilinx tools to implement the core. The other design files generated depend on the Design Entry settings specified (target CAE vendor, and design flow type -- schematic or HDL). Schematic symbol files are generated when a schematic design flow is specified for the project.

Parameterized HDL simulation models are provided in two separate HDL simulation libraries, one for Verilog functional simulation support, and the other for VHDL functional simulation support. The libraries, which are included as part of the Xilinx installation, are in the following locations:

$XILINX/verilog/src/XilinxCoreLib

$XILINX/vhdl/src/XilinxCoreLib



Distributed Arithmetic FIR Filter

Component Name:

Filter Type
● Single Rate FIR ○ Halfband ○ Hilbert Transform
○ Interpolated ○ Interpolation ○ Decimation
○ Decimating Halfband ○ Interpolating Halfband

Filter Options
Number of Channels: 1
Zero Packing Factor: 1
Sample Rate Change: 1
Valid Range: 1..8

Structure
Number of Taps: 16 2..1024

Impulse Response
○ Symmetric ○ Non Symmetric ● Negative Symmetric

Page 1 of 2

☐ Display Core Footprint

ug002_c2_070a_100501

Figure 2-124: **Core Customization Window for a Parameterized Core**

If using a compiled simulator, these libraries must be precompiled before performing a functional simulation of the cores. An analyze_order file describing the required compile order of these models is included with each XilinxCoreLib library, one for Verilog (verilog_analyze_order) and one for VHDL (vhdl_analyze_order).

For an HDL design flow, Verilog and VHDL templates (.VEO and .VHO files) are also provided to facilitate the integration of the core into the design for the purposes of functional simulation, synthesis, and implementation. The Verilog (.V) and VHDL (.VHD) wrapper files are also generated. The wrapper files for a particular core are compiled like normal simulation models. They convey custom parameter values to the corresponding generic, parameterized behavioral model for that core in the XilinxCoreLib library. The custom parameter values are used to tailor the behavior of the customized core.

The following is a sample VHO template:

```
component adder8
    port (
    a: IN std_logic_VECTOR(7 downto 0);
    b: IN std_logic_VECTOR(7 downto 0);
    c: IN std_logic;
    ce: IN std_logic;
    ci: IN std_logic;
    clr: IN std_logic;
    s: OUT std_logic_VECTOR(8 downto 0));
end component;

-- Synplicity black box declaration
attribute black_box : boolean;
attribute black_box of test: component is true;

-- COMP_TAG_END ------ End COMPONENT Declaration ------------

-- The following code must appear in the VHDL architecture
-- body. Substitute your own instance name and net names.

------------- Begin Cut here for INSTANTIATION Template ----- INST_TAG
your_instance_name : adder8
    port map (
    a => a,
    b => b,
    c => c,
    ce => ce,
    ci => ci,
    clr => clr,
    s => s);

-- INST_TAG_END ------ End INSTANTIATION Template ------------
-- You must compile the wrapper file test.vhd when simulating
-- the core, test. When compiling the wrapper file, be sure to
-- reference the XilinxCoreLib VHDL simulation library. For detailed
-- instructions, please refer to the "Core Generator Guide".
```

Fixed Netlist Cores

The other type of Virtex-II core provided by the CORE Generator is the fixed netlist core. These are preset, non-parameterized designs that are shipped with the following:

- A fixed EDIF implementation netlist (as opposed to one that is customized on the fly)
- .VEO and .VHO templates
- Non-parameterized .V and .VHD behavioral simulation models
- Schematic symbol support

Examples include the fixed netlist Xilinx FFTs and most AllianceCORE products.

Since the HDL behavioral models for fixed netlist cores are not parameterized, the corresponding .VEO and .VHO template files are correspondingly simple. They do not need to pass customizing parameter values to a library behavioral model.

Xilinx IP Solutions and the IP Center

The CORE Generator works in conjunction with the Xilinx IP Center on the world wide web to provide the latest IP and software upgrades. To make the most of this resource, Xilinx highly recommends that whenever starting a design, first do a quick search of the Xilinx IP Center (www.xilinx.com/ipcenter) to see whether a ready-made core solution is already available.

A complete catalog of Xilinx cores and IP tools resides on the IP Center, including:

- LogiCORE Products
- AllianceCORE Products
- Reference Designs
- XPERTS Partner Consultants
- Design Reuse Tools

When installing the CORE Generator software, the designer gains immediate access to dozens of cores supplied by the LogiCORE Program. In addition, data sheets are available for all AllianceCORE products, and additional, separately licensed, advanced function LogiCORE products are also available. New and updated Virtex-II IP for the CORE Generator can be downloaded from the IP Center and added to the CORE Generator catalog.

LogiCORE Program

LogiCORE products are designed, sold, licensed, and supported by Xilinx. LogiCORE products include a wide selection of generic, parameterized functions, such as muxes, adders, multipliers, and memory cores which are bundled with the Xilinx CORE Generator software at no additional cost to licensed software customers. System-level cores, such as PCI, Reed-Solomon, ADPCM, HDLC, POS-PHY, and Color Space Converters are also available as optional, separately licensed products. Probably, the most common application of the CORE Generator is to use it to quickly generate Virtex-II block and distributed memories. A more detailed listing of available Virtex-II LogiCORE products is available in Table 2-62 and on the Xilinx IP Center website (www.xilinx.com/ipcenter).

Types of IP currently offered by the Xilinx LogiCORE program include:

- Basic Elements: logic gates, registers, multiplexers, adders, multipliers
- Communications and Networking: ADPCM modules, HDLC controllers, ATM building blocks, forward error correction modules, and POS-PHY Interfaces
- DSP and Video Image Processing: cores ranging from small building blocks (e.g., Time Skew Buffers) to larger system-level functions (e.g., FIR Filters and FFTs)
- System Logic: accumulators, adders, subtracters, complementers, multipliers, integrators, pipelined delay elements, single and dual-port distributed and block RAM, ROM, and synchronous and asynchronous FIFOs
- Standard Bus Interfaces: PCI 64/66 (64-bit, 66 MHz), 64/33 (64-bit, 33 MHz), and 32/33 (32-bit, 3 3MHz) Interfaces

AllianceCORE Program

The AllianceCORE program is a cooperative effort between Xilinx and third-party IP developers to provide additional system-level IP cores optimized for Xilinx FPGAs. To ensure a high level of quality, AllianceCORE products are implemented and verified in a Xilinx device as part of the certification process.

Xilinx develops relationships with AllianceCORE partners who can complement the Xilinx LogiCORE product offering. Where Xilinx does not offer a LogiCORE for a particular function, Xilinx partners with an AllianceCORE partner to offer that function. A large percentage of Xilinx AllianceCORE partners focus on data and telecommunication applications, as well as processor and processor peripheral designs.

Together, Xilinx and the AllianceCORE partners are able to provide an extensive library of cores to accelerate the design process. AllianceCORE products include customizable cores which can be configured to exact needs, as well as fixed netlist cores targeted toward specific applications. In many cases, partners can provide cores customized to meet the specific design needs if the primary offerings do not fit the requirements. Additionally, source code versions of the cores are often available from the partners at additional cost for those who need maximum flexibility.

The library of Xilinx and AllianceCORE IP cores allows designers to leverage the expertise of experienced designers who are well-versed in optimizing designs for Virtex-II and other Xilinx architectures. This enables designers to obtain high performance and density in the target Virtex-II device with a faster time to market.

Reference Designs

Xilinx offers two types of reference designs; application notes (XAPPs) developed by Xilinx, and reference designs developed through the Xilinx Reference Design Alliance Program. Both types are extremely valuable to customers looking for guidance when designing systems. Reference designs can often be used as starting points for implementing a broad spectrum of functions in Xilinx programmable logic.

Application notes developed by Xilinx usually include supporting design files. They are supplied free of charge, without technical support or warranty. To see currently available reference designs, visit the **www.xilinx.com/products/logicore/refdes.htm** website.

Reference designs developed through the Xilinx Reference Design Alliance Program are developed, owned, and controlled by the partners in the program. The goal of the program is to form strategic engineering and marketing partnerships with other semiconductor manufacturers and design houses so as to assist in the development of high quality, multicomponent reference designs that incorporate Xilinx devices and demonstrate how they can operate at the system level with other specialized and general purpose semiconductors.

The reference designs in the Xilinx Reference Design Alliance Program are fully functional and applicable to a wide variety of digital electronic systems, including those used for networking, communications, video imaging, and DSP applications. Visit the **www.xilinx.com/company/reference_design/referencepartners.htm** website to see a list of designs currently available through this program.

XPERTS Program

Xilinx established the XPERTS Program to provide customers with access to a worldwide network of certified design consultants proficient with Xilinx Platform FPGAs, software, and IP core integration. All XPERT members are certified and have extensive expertise and experience with Xilinx technology in various vertical applications, such as communications and networking, DSP, video and image processing, system I/O interfaces, and home networking.

XPERTS partners are an integral part of Xilinx strategy to provide customers with cost-efficient design solutions, while accelerating time to market. For more information on Xilinx XPERTS Program, visit the **www.xilinx.com/company/consultants/index.htm** website.

Design Reuse Tools

To facilitate the archiving and sharing of IP created by different individuals and workgroups within a company, Xilinx offers the IP Capture Tool. The IP Capture Tool helps to package design modules created by individual engineers in a standardized format so that they can be cataloged and distributed using the Xilinx CORE Generator. A core can take the form of synthesizable VHDL or Verilog code, or a fixed function netlist. Once it is packaged by the IP Capture Tool and installed into the CORE Generator, the *"captured"* core can be shared with other designers within a company through an internal network. The IP Capture Tool is supplied as a separate utility through the Xilinx IP Center. For more information, see the **www.xilinx.com/ipcenter/designreuse/ipic.htm** website.

CORE Generator Summary

The CORE Generator delivers a complete catalog of IP including behavioral models, synthesis templates, and netlists with performance guaranteed by Xilinx Smart-IP technology. It is a repository for LogiCORE products from Xilinx, AllianceCORE products from Xilinx partners, and it supports Design Reuse for internally developed IP. In addition, LogiCORE products are continuously updated to add support for new Xilinx architectures, such as Virtex-II. The most current IP updates are available from the Xilinx IP Center.

Utilizing the CORE Generator library of parameterizable cores, designed by Xilinx for Xilinx FPGAs, the designer can enjoy the advantages of design reuse, including faster time to market and lower cost solutions. For more information, visit the Xilinx IP Center www.xilinx.com/ipcenter website.

Virtex-II IP Cores Support

Table 2-62 provides a partial listing of cores available for Virtex-II designs. For a complete catalog of Virtex-II IP, visit the Xilinx IP Center www.xilinx.com/ipcenter website.

Table 2-62: **Virtex-II IP Cores Support**

Function	Vendor Name	IP Type	Implementation Example			Key Features	Application Examples
			Occ	MHz	Device		
Basic Elements							
BUFE-based Multiplexer Slice	Xilinx	LogiCORE				1-256 bits wide	
BUFT-based Multiplexer Slice	Xilinx	LogiCORE				1-256 bits wide	
Binary Counter	Xilinx	LogiCORE				2-256 bits output width	
Binary Decoder	Xilinx	LogiCORE				2-256 bits output width	
Bit Bus Gate	Xilinx	LogiCORE				1-256 bits wide	
Bit Gate	Xilinx	LogiCORE				1-256 bits wide	
Bit Multiplexer	Xilinx	LogiCORE				1-256 bits wide	
Bus Gate	Xilinx	LogiCORE				1-256 bits wide	
Bus Multiplexer	Xilinx	LogiCORE				IO widths up to 256 bits	
Comparator	Xilinx	LogiCORE				1-256 bits wide	
FD-based Parallel Register	Xilinx	LogiCORE				1-256 bits wide	
FD-based Shift Register	Xilinx	LogiCORE				1-64 bits wide	
LD-based Parallel Latch	Xilinx	LogiCORE				1-256 bits wide	
RAM-based Shift Register	Xilinx	LogiCORE				1-256 bits wide, 1024 words deep	
Communication & Networking							
3G FEC Package	Xilinx	LogiCORE				Viterbi Decoder, Turbo Codec, Convolutional Enc	3G Wireless Infrastructure
3GPP Compliant Turbo Convolutional Decoder	Xilinx	LogiCORE	80%	40	XC2V500	3GPP specs, 2 Mbps, BER=10-6 for 1.5dB SNR	3G Wireless Infrastructure
3GPP Compliant Turbo Convolutional Encoder	Xilinx	LogiCORE	65%	60	XC2V250	Compliant w/ 3GPP, puncturing	3G Wireless Infrastructure
3GPP Turbo Decoder	SysOnChip	AllianceCORE	87%	66	XC2V500-5	3GPP/UMTS compliant, IMT-2000, 2Mbps data	Error correction, wireless

Table 2-62: **Virtex-II IP Cores Support** ***(Continued)***

Function	Vendor Name	IP Type	Implementation Example			Key Features	Application Examples
			Occ	MHz	Device		
8b/10b Decoder	Xilinx	LogiCORE	1 BRAM	100	XC2V1000	Industry std 8b/10b en/decode for serial data transmission	Physical layer of Fiber Channel
8b/10b Encoder	Xilinx	LogiCORE	1 BRAM	100	XC2V1000	Industry std 8b/10b en/decode for serial data transmission	Physical layer of Fiber Channel
ADPCM 1024 Channel	Amphion	AllianceCORE				G.721, 723, 726, 726a, 727, 727a, u-law, a-law	DECT, VOIP, cordless telephony
ADPCM 256 Channel	Amphion	AllianceCORE				G.721, 723, 726, 726a, 727, 727a, u-law, a-law	DECT, VOIP, cordless telephony
ADPCM 512 Channel	Amphion	AllianceCORE					
ADPCM 768 Channel	Amphion	AllianceCORE	89%	50	XC2V500-5	G.721, 723, 726, 726a, 727, 727a, u-law, a-law	DECT, VOIP, cordless telephony
ADPCM Speech Codec, 32 Channel (DO-DI-ADPCM32)	Xilinx	LogiCORE	62%	25	XC2V500	G.726, G.727, 32 duplex channels	DECT, VOIP, Wireless local loop, DSLAM, PBX
ADPCM Speech Codec, 64 Channel (DO-DI-ADPCM64)	Xilinx	LogiCORE	61%	27	XC2V500	G.726, G.727, 64 duplex channels	DECT, VOIP, wireless local loop, DSLAM, PBX
BOOST LITE Bluetooth Baseband Processor	NewLogic	AllianceCORE	73%	33%	XC2V1000-4	Compliant to Bluetooth v1.1, BQB qualified software for L2CAP, LHP, HC1, voice support	Bluetooth applications
BOOST Lite Bluetooth Baseband Processor	NewLogic	AllianceCORE	73%	33%	XC2V1000-4	Compliant to Bluetooth v1.1, BQB qualified software for L2CAP, LHP, HC1, voice support	Bluetooth applications
Convolutional Encoder	Xilinx	LogiCORE	10%	26	XC2V40	k from 3 to 9, puncturing from 2/3 to 12/13	3G base stations, broadcast, wireless LAN, cable modem, xDSL, satellite com, uwave
DVB-RCS Turbo Decoder	iCODING	AllianceCORE	54%	69	XC2V2000-5	DVB-RCS compliant, 9Mbps, data rate, switchable code rates and frame sizes	Error correction, wireless, DVB, Satellite data link
Flexbus 4 Interface Core, 16-Channel (DO-DI-FLX4C16)	Xilinx	LogiCORE	31%	200	XC2V3000 FG676-5		Line card: terabit routers & optical switches
Flexbus 4 Interface Core, 4-Channel (DO-DI-FLX4C4)	Xilinx	LogiCORE	27%	200	XC2V1000 FG456-5		Line card: terabit routers & optical switches
Flexbus 4 Interface Core, 1-Channel (DO-DI-FLX4C1)	Xilinx	LogiCORE	12%	200	XC2V1000 FG456-5		Line card: terabit routers & optical switches

Table 2-62: **Virtex-II IP Cores Support** ***(Continued)***

Function	Vendor Name	IP Type	Implementation Example			Key Features	Application Examples
			Occ	MHz	Device		
HDLC Controller Core, 32 Channels	Xilinx	LogiCORE	34%	81	XC2V250	32 full duplex, CRC-16/32, 8/16-bit address insertion/deletion	X.25, POS, cable modems, frame relay switches, video confer. over ISDN
HDLC Controller Core, Single Channel	Xilinx	LogiCORE	15%	115	XC2V250	16/32-bit frame seq, 8/16-bit addr insert/delete, flag/zerop insert/detect	X.25, POS, cable modems, frame relay switches, video conf. over ISDN
Interleaver/De-interleaver	Xilinx	LogiCORE	30%	187	XC2V40	Convolutional, width up to 256 bits, 256 branches	Broadcast, wireless LAN, cable modem, xDSL, satellite com,uwave nets, digital TV
PE-MACMII Dual Speed 10/100 Mbps Ethernet MAC	Alcatel	AllianceCORE	33%	60	XC2V500-4	802.3 compliant, Supports single & multimode fiber optic devices, M11 interfaces, RMON and Etherstate statistics	Networking, Broadband, NIC, SOHO, Home networking, storage, routers, switches, printers,
POS-PHY Level 3 Link Layer Interface Core, 48 Channel (DO-DI-POSL3LINK48A)	Xilinx	LogiCORE	33%	104	XC2V6000 FF1152-4		
POS-PHY L3 Link Layer Interface, 16-Ch (DO-DI-POSL3LINK16)	Xilinx	LogiCORE	40%	104	XC2V1000 FG456-4		Line card: terabit routers & optical switches
POS-PHY L3 Link Layer Interface, 4-Ch (DO-DI-POSL3LINK4)	Xilinx	LogiCORE	15%	104	XC2V1000 FG456-4		Line card: terabit routers & optical switches
POS-PHY L3 Link Layer Interface, 2-Ch (DO-DI-POSL3LINK2)	Xilinx	LogiCORE	55%	104	XCV50E-8		Line card: terabit routers & optical switches
POS-PHY L3 Link Layer Interface, Single Channel	Xilinx	LogiCORE	6%	104	XC2V1000 FG456-4		
POS-PHY L4 Multi-Channel Interface (DO-DI-POSL4MC)	Xilinx	LogiCORE	29%	104	XC2V3000 FG676-5		
Reed-Solomon Decoder	Xilinx	LogiCORE	40%	98	XC2V250	Std or custom coding, 3-12 bit symbol width, up to 4095 symbols	Broadcast, wireless LAN, digital TV, cable modem, xDSL, satellite com,uwave nets
Reed-Solomon Decoder	TILAB	AllianceCORE	56%	61	XC2V1000-5	parameterizable, RTL available	Error correction, wireless, DSL

Table 2-62: **Virtex-II IP Cores Support** ***(Continued)***

Function	Vendor Name	IP Type	Implementation Example			Key Features	Application Examples
			Occ	MHz	Device		
Reed-Solomon Encoder	Xilinx	LogiCORE	42%	180	XC2V40	Std or cust coding, 3-12 bit width, up to 4095 symbols with 256 check symb.	Broadcast, wireless LAN, digital TV, cable modem, xDSL, satellite com,uwave nets
SDLC Controller	CAST	AllianceCORE	38%	158	XC2V100-5	Like Intel 8XC152 Global Serial Channel, Serial Comm., HDLC apps, telecom	Embedded systems, professional audio, video
SPEEDROUTER Network Processor	IP	AllianceCORE	64%	80 MHz, 2.5 Gbps	XC2V1500-5	Solution requires SPEEDAnalyzer ASIC, 2.5 Gbps fdx wire speed; net processor (NPV)	Networking, edge and access, Switches and routers
Turbo Decoder - 3GPP	SysOnChip	AllianceCORE	88%	65	XC2V2000-5	3GPP/UMTS compliant, 2Mbps data rate	Error correction, wireless
Turbo Encoder	TILAB	AllianceCORE	48%	120	XC2V80-5	3GPP/UMTS compliant, upto 4 interleaver laws	Error correction, wireless
TURBO_DEC Turbo Decoder	TILAB	AllianceCORE	99%	65	XC2V2000-5	3GPP/UMTS compliant, >2Mbps data rate	Error correction, wireless
Viterbi Decoder	Xilinx	LogiCORE	80%	100	XC2V250	Puncturing, serial & parallel architecture,	3G base stations, broadcast, wireless LAN, cable modem, xDSL, satellite com, uwave
Viterbi Decoder, IEEE 802-compatible	Xilinx	LogiCORE	70%	147	XC2V250	Constraint length(k)=7, G0=171, G1=133	L/MMDS, cable modem, broadcast equip, wireless LAN, xDSL, sat com, uwave nets
Digital Signal Processing							
1024-Point Complex FFT IFFT for Virtex-II	Xilinx	LogiCORE	62%	41us, 100 MHz	XC2V500	16 bit complex data, 2's comp, forward and inverse transform	
16-Point Complex FFT IFFT for Virtex-II	Xilinx	LogiCORE	37%	123ns, 130 MHz	XC2V500	16 bit complex data, 2's comp, forward and inverse transform	
256-Point Complex FFT IFFT for Virtex-II	Xilinx	LogiCORE	54%	7.7us, 100 MHz	XC2V500	16 bit complex data, 2's comp, forward and inverse transform	
32 Point Complex FFT/IFFT	Xilinx	LogiCORE					
64-Point Complex FFT IFFT for Virtex-II	Xilinx	LogiCORE	38%	1.9us, 100 MHz	XC2V500	16 bit complex data, 2's comp, forward and inverse transform	
Bit Correlator	Xilinx	LogiCORE				4096 taps, serial/parallel input, 4096 bits width	
Cascaded Integrator Comb (CIC)	Xilinx	LogiCORE				32 bits data width, rate change from 8 to 16384	

Table 2-62: **Virtex-II IP Cores Support** ***(Continued)***

Function	**Vendor Name**	**IP Type**	**Implementation Example**			**Key Features**	**Application Examples**
			Occ	MHz	Device		
Direct Digital Synthesizer	Xilinx	LogiCORE				8-65K samples, 32-bits output precision, phase dithering/offset	
Distributed Arithmetic FIR Filter	Xilinx	LogiCORE				32-bit input/coeff width, 1024 taps, 1-8 chan, polyphase, online coeff reload	
GVA-300 Virtex-II DSP Hardware Accelerator	GV	AllianceCORE	NA	NA		2 Virtex-II, Spartan-II FPGAs, 1 CPLD, Matlab I/F	DSP prototyping
LFSR, Linear Feedback Shift Register	Xilinx	LogiCORE				168 input widths, SRL16/register implementation	
Math Functions							
Accumulator	Xilinx	LogiCORE				1-256s bit wide	
Adder Subtracter	Xilinx	LogiCORE				1-256s bit wide	
DFP2INT Floating Point to Integer Converter	Digital	AllianceCORE	39%	66	XC2V250-5	Full IEEE-754 compliance, 4 pipelines, Single precision real format support	DSP, Math, Arithmetic apps
DFPADD Floating Point Adder	Digital	AllianceCORE	39%	66	XC2V250-5	Full IEEE-754 compliance, 4 pipelines, Single precision real format support	DSP, Math, Arithmetic apps
DFPCOMP Floating Point Comparator	Digital	AllianceCORE	16%	91	XC2V80-5	Full IEEE-754 compliance, 4 pipelines, Single precision real format support	DSP, Math, Arithmetic apps.
DFPDIV Floating Point Divider	Digital	AllianceCORE	99%	53	XC2V250-5	Full IEEE-754 compliance, 15 pipelines, Single precision real format support	DSP, Math, Arithmetic apps
DFPMUL Floating Point Multiplier	Digital	AllianceCORE	44%	74	XC2V250-5	Full IEEE-754 compliance, 7 pipelines,32x32 mult, Single precision real format support	DSP, Math, Arithmetic apps.
DFPSQRT Floating Point Square Root	Digital	AllianceCORE	39%	66	XC2V250-5	Full IEEE-754 compliance, 4 pipelines, Single precision real format support	DSP, Math, Arithmetic apps
DINT2FP Integer to Floating Point Converter	Digital	AllianceCORE	37%	73	XC2V250-5	Full IEEE-754 compliance, double word input, 2 pipelines, Single precision real output	DSP, Math, Arithmetic apps
Multiply Accumulator (MAC)	Xilinx	LogiCORE				Input width up to 32 bits, 65-bit accumulator, truncation rounding	

Table 2-62: **Virtex-II IP Cores Support** ***(Continued)***

Function	Vendor Name	IP Type	Implementation Example			Key Features	Application Examples
			Occ	MHz	Device		
Multiply Generator	Xilinx	LogiCORE				64-bit input data width, constant, reloadable or variable inputs, parallel/sequential implementation	
Pipelined Divider	Xilinx	LogiCORE				32-bit input data width, multiple clock per output	
Sine Cosine Look Up Table	Xilinx	LogiCORE				3-10 bit in, 4-32 bit out, distributed/block ROM	
Twos Complementer	Xilinx	LogiCORE				Input width up to 256 bits	
Memories & Storage Elements							
Asynchronous FIFO	Xilinx	LogiCORE				1-256 bits, 15-65535 words, DRAM or BRAM, independent I/O clock domains	
Content Addressable Memory (CAM)	Xilinx	LogiCORE				1-512 bits, 2-10K words, SRL16	
Distributed Memory	Xilinx	LogiCORE				1-1024 bit, 16-65536 word, RAM/ROM/SRL16, opt output regs and pipelining	
Dual-Port Block Memory	Xilinx	LogiCORE				1-256 bits, 2-13K words	
Single-Port Block Memory	Xilinx	LogiCORE				1-256 bits, 2-128K words	
Synchronous FIFO	Xilinx	LogiCORE				1-256 bits, 16-256 words, distributed/block RAM	
Microprocessors, Controllers & Peripherals							
10/100 Ethernet MAC	Xilinx	LogiCORE				Interfaces through OPB to MicroBlaze	Networking, comm., processor applications
AX1610 16-bit RISC Processor	Loarant	AllianceCORE	12%	91	XC2V500-5	44 opcode, 64-K word data, program, Harvard arch.	Control functions, State mach, Coprocessor
C165X MicroController	CAST	AllianceCORE	60%	134	XC2V80-5	Microchip 16C5X PIC like	Embedded systems, telecom
C68000 Microprocessor	CAST	AllianceCORE	90%	32	XC2V500-5	MC68000 Compatible	Embedded systems, pro audio, video
CPU FPGA (Virtex-II) MicroEngine Cards	NMI	AllianceCORE	NA	NA	NA	Hitachi SH-3 CPU	Embedded systems
CZ80CPU Microprocessor	CAST	AllianceCORE	55%	72	XC2V500-5	Zilog Z80 compatible, 8-bit processor	Embedded systems, Communications

Table 2-62: **Virtex-II IP Cores Support** ***(Continued)***

Function	Vendor Name	IP Type	Implementation Example			Key Features	Application Examples
			Occ	MHz	Device		
DDR SDRAM Controller Core	Memec-Core	AllianceCORE	7%	133	XC2V1000-4	DDR SDRAM burst length support for 2,4,8 per access, supports data 16,32, 64, 72.	Digital video, embedded computing , networking
DFPIC125X Fast RISC MicroController	Digital	AllianceCORE	49%	126	XC2V80-5	PIC 12c4x like, 2X faster, 12-bit wide instruction set, 33 instructions	Embedded systems, telecom, audio and video
DFPIC1655X Fast RISC MicroController	Digital	AllianceCORE	79%	140	XC2V80-5	S/W compatible with PIC16C55X, 14-bit instruction set, 35 instructions	Embedded systems, telecom, audio and video
DFPIC165X Fast RISC MicroController	Digital	AllianceCORE	49%	126	XC2V80-5	PIC 12c4x like, 2X faster, 12-bit wide instruction set, 33 instructions	Embedded systems, telecom, audio and video
DI2CM I2C Bus Controller Master	Digital	AllianceCORE	58%	143	XC2V50-5	I2C-like, multi master, fast/std. modes	Embedded systems
DI2CM I2C Bus Controller Slave	Digital	AllianceCORE	28%	157	XC2V50-5	I2C-like, Slave	Embedded
DI2CSB I2C Bus Controller Slave Base	Digital	AllianceCORE	15%	187	XC2V50-5	I2C-like, Slave	Embedded Systems
DR8051 RISC MicroController	Digital	AllianceCORE	68%	73	XC2V250-5	80C31 instruction set, RISC architecture 6.7X faster than standard 8051	Embedded systems, telecom, video
DR8051BASE RISC MicroController	Digital	AllianceCORE	46%	80-90	XC2V250-5	80C31 instruction set, high speed multiplier, RISC architecture 6.7X faster than standard 8051	Embedded systems, telecom, video
DR8052EX RISC MicroController	Digital	AllianceCORE	99%	71	XC2V250-5	80C31 instruction set, high speed mult/div ,RISC 6.7X faster than standard 8051	Embedded systems, telecom, video
e8254 Programmable Interval Timer/Counter	einfochips	AllianceCORE	1%	175	XC2V1000-5	Three 8-bit parallel ports, 24 programmable IO lines, 8-bit bidi data bus	Processor, I/O interface
e8255 Peripheral Interface	einfochips	AllianceCORE	1%	175	XC2V1000-5	Three 8-bit parallel ports, 24 programmable IO lines, 8-bit bidi data bus	Processor, I/O interface
Flip805x-PS Microprocessor	Dolphin	AllianceCORE	39%	38	XC2V1000-5	Avg 8X faster & code compatible v. legacy 8051, verification bus monitor, SFR IF, DSP focused	DSP, Telecom, industrial, high speed control
IIC	Xilinx	LogiCORE				Interfaces through OPB to MicroBlaze	Networking, com, processor applic
LavaCORE Configurable Java Processor Core	Derivation	AllianceCORE	38%	20	XC2V1000-5	32b data/address optional DES	Internet appliance, industrial control
LavaCORE Configurable Java Processor Core	Derivation	AllianceCORE	38%	20	XC2V1000-5	32b data/address optional DES	Internet appliance, industrial control

Table 2-62: **Virtex-II IP Cores Support** ***(Continued)***

Function	Vendor Name	IP Type	Implementation Example			Key Features	Application Examples
			Occ	MHz	Device		
Lightfoot 32-bit Java Processor Core	Digital	AllianceCORE	33%	40	XC2V1000-5	32bit data, 24 bit address, 3 Stage pipeline, Java/C dev. tools	Internet appliance, industrial control, HAVi multimedia, set top boxes
MicroBlaze Soft RISC Processor	Xilinx	LogiCORE		125		Soft RISC Processor, small footprint	Networking, communications
OPB Arbiter	Xilinx	LogiCORE		125		Bundled in the MicroBlaze Development Kit	Processor applications
OPB GPIO	Xilinx	LogiCORE		125		Bundled in the MicroBlaze Development Kit	Processor applications
OPB Interrupt Controller	Xilinx	LogiCORE		125		Bundled in the MicroBlaze Development Kit	Processor applications
OPB Memory Interface (Flash, SRAM)	Xilinx	LogiCORE		125		Bundled in the MicroBlaze Development Kit	Processor applications
OPB Timer/Counter	Xilinx	LogiCORE		125		Bundled in the MicroBlaze Development Kit	Processor applications
OPB UART (16450, 16550)	Xilinx	LogiCORE		125		Interfaces through OPB to MicroBlaze	Processor applications
OPB UART Lite	Xilinx	LogiCORE		125		Bundled in the MicroBlaze Development Kit	Processor applications
OPB WDT	Xilinx	LogiCORE		125		Bundled in the MicroBlaze Development Kit	Processor applications
PF3100 PC/104-Plus Reconfigurable Module	Derivation	AllianceCORE	N/A	N/A	XC2V1000 FG256	PC/104 & PC/104+ devlopment board	Internet appliance, industrial control
SPI	Xilinx	LogiCORE				Interfaces through OPB to MicroBlaze	Networking, communications, processor applications
XF-UART Asynchronous Communications Core	Memec-Core	AllianceCORE	15%	50	XCS20-4	UART and baud rate generator	Serial data communication
Standard Bus Interfaces							
PCI-X 64/100 Interface for Virtex-II (DO-DI-PCIX64-VE)	Xilinx	LogiCORE	30%	100	XC2V1000 FG456-5	PCI-X 1.0 comp, 64/32-bit, 66 MHz PCI-X initiator and target IF, PCI 2.2 comp, 64/32-bit, 33 MHz PCI initiator and target IF, 3.3 V PCI-X at 33-66 MHz, 3.3 V PCI at 0-33 MHz	Server,Embedded, gb ethernet,U320 SCSI,Fibre Ch,RAID cntl,graphics

Table 2-62: **Virtex-II IP Cores Support** ***(Continued)***

Function	**Vendor Name**	**IP Type**	**Implementation Example**			**Key Features**	**Application Examples**
			Occ	MHz	Device		
PCI32 Virtex Interface Design Kit (DO-DI-PCI32-DKT)	Xilinx	LogiCORE	6%	66	XC2V1000 FG456-5	Includes PCI32 board, drive development kit, and customer education 3-day training class	
PCI32 Virtex Interface, IP Only (DO-DI-PCI32-IP)	Xilinx	LogiCORE	6%	66	XC2V1000 FG456-5	v2.2 comp, assured PCI timing, 3.3/5-V, 0-waitstate, CPCI hot swap friendly	PC add-in boards, CPCI, Embedded
PCI64 & PCI32, IP Only (DO-DI-PCI-AL)	Xilinx	LogiCORE	6 - 7%	66	XC2V1000 FG456-5	v2.2 comp, assured PCI timing, 3.3/5-V, 0-waitstate, CPCI hot swap friendly	PC boards,CPCI,Embedded,hiperf video,gb ethernet
PCI64 Virtex Interface Design Kit (DO-DI-PCI64-DKT)	Xilinx	LogiCORE	7%	66	XC2V1000 FG456-5	v2.2 comp, assured PCI timing, 3.3/5-V, 0-waitstate, CPCI hot swap friendly	PC boards, CPCI, Embedded, hiperf video, gb ethernet
PCI64 Virtex Interface, IP Only (DO-DI-PCI64-IP)	Xilinx	LogiCORE	7%	66	XC2V1000 FG456-5	v2.2 comp, assured PCI timing, 3.3/5-V, 0-waitstate, CPCI hot swap friendly	PC boards,CPCI,Embedded,hiperf video,gb ethernet
RapidIO 8-bit port LP-LVDS Phy Layer (DO-DI-RIO8-PHY)	Xilinx	LogiCORE	24%	250	XC2V1000 FG456-5	RapidIO Interconnect v1.1 compliant, verified with Motorola's RapidIO bus functional model v1.4	Routers, switches, backplane, control plane, data path, embedded sys, high speed interface to memory and encryption engines, high end video
USB 1.1 Device Controller	Memec-Core	AllianceCORE	21%	12	XC2V1000-5	Compliant with USB1.1 spec., Supports VCI bus, Performs CRC, Supports 1.5 Mbps & 12 Mbps	Scanners, Printers, Handhelds, Mass Storage
Video & Image Processing							
1-D Discrete Cosine Transform	Xilinx	LogiCORE				8-24 bits for coeff & input, 8-64 pts	
2-D DCT/IDCT Forward/Inverse Discrete Cosine Transform	Xilinx	LogiCORE					image, video phone, color laser printers
FASTJPEG_BW Decoder	BARCO-SILEX	AllianceCORE	67%	73	XC2V1000-4	Conforms to ISO/IEC Baseline 10918-1, Gray-Scale	Video editing, digital camera, scanners
FASTJPEG_C Decoder	BARCO-SILEX	AllianceCORE	78%	56	XC2V1000-4	Conforms to ISO/IEC Baseline 10918-1, color, multi-scan, Gray-Scale	Video editing, digital camera, scanners

Chapter 3

Configuration

Summary

This chapter covers the following topics:

- Introduction
- Configuration Solutions
- Master Serial Programming Mode
- Slave Serial Programming Mode
- Master SelectMAP Programming Mode
- Slave SelectMAP Programming Mode
- JTAG/ Boundary Scan Programming Mode
  - Boundary-Scan for Virtex-II Devices Using IEEE Standard 1149.1
  - Boundary-Scan for Virtex-II Devices Using IEEE Standard 1532
- Configuration With MultiLINX
- Configuration Details
- Readback

Introduction

Virtex-II devices are configured by loading application-specific configuration data into internal memory. Configuration is carried out using a subset of the device pins, some of which are dedicated, while others can be reused as general-purpose inputs and outputs after configuration is complete.

Depending on the system design, several configuration modes are selectable via mode pins. The mode pins M2, M1, and M0 are dedicated pins. An additional pin, HSWAP_EN, is used in conjunction with the mode pins to select whether user I/O pins have pull-up resistors during configuration. By default, HSWAP_EN is tied High (internal pull-up resistor), which shuts off pull-up resistors on the user I/O pins during configuration. When HSWAP_EN is tied Low, the pull-up resistors are on and therefore, the user I/Os have pull-up resistors during configuration.

Other dedicated pins are:

- CCLK - the configuration clock pin
- DONE - configuration status pin
- TDI, TDO, TMS, TCK - boundary-scan pins
- PROG_B - configuration reset pin

Depending on the configuration mode selected, CCLK can be an output generated by the Virtex-II FPGA or an input accepting externally generated clock data. For correct operation, these pins require a V_{CCAUX} of 3.3 V to permit low-voltage transistor-to-transistor logic (LVTTL) operations.

All dual-function configuration pins are contained in banks 4 and 5. Bank 4 contains pins used in serial configuration modes, and banks 4 and 5 contain pins used for SelectMAP modes.

A persist option is available, which can be used to force pins to retain their configuration function even after device configuration is complete. If the persist option is not selected, then the configuration pins with the exception of CCLK, PROG_B, and DONE can be used for user I/O in normal operation. The persist option does not apply to boundary-scan related pins. The persist feature is valuable in applications that employ partial reconfiguration, dynamic reconfiguration, or readback.

Configuration Modes

Virtex-II supports the following configuration modes:

- Master-Serial
- Slave-Serial (default)
- Master SelectMAP
- Slave SelectMAP
- Boundary-Scan (IEEE 1532 and IEEE 1149)

Table 3-1 shows Virtex-II configuration mode pin settings.

Table 3-1: **Virtex-II Configuration Mode Pin Settings**

Configuration Mode[1]	M2	M1	M0	CCLK Direction	Data Width	Serial Dout[2]
Master Serial	0	0	0	Out	1	Yes
Slave Serial	1	1	1	In	1	Yes
Master SelectMAP	0	1	1	Out	8	No
Slave SelectMAP	1	1	0	In	8	No
Boundary Scan	1	0	1	N/A	1	No

Notes:

1. The HSWAP_EN pin controls the pullups. Setting M2, M1, and M0 selects the configuration mode, while the HSWAP_EN pin controls whether or not the pullups are used.
2. Daisy chaining is possible only in modes where Serial Dout is used. For example, in SelectMAP modes, the first device does NOT support daisy chaining of downstream devices.

Table 3-2 lists the total number of bits required to configure each device:

Table 3-2: **Virtex-II Bitstream Lengths**

Device	Total Number of Configuration Bits (including header)
XC2V40	360,096
XC2V80	635,296
XC2V250	1,697,184
XC2V500	2,761,888
XC2V1000	4,082,592
XC2V1500	5,659,296
XC2V2000	7,492,000
XC2V3000	10,494,368
XC2V4000	15,659,936
XC2V6000	21,849,504
XC2V8000	29,063,072

Configuration Process and Flow

The configuration process involves loading the configuration bitstream into the FPGA using the selected mode. There are four major phases in the configuration process:

- Clearing Configuration Memory
- Initialization
- Loading Configuration Data
- Device Startup

Figure 3-1 illustrates the configuration process flow.

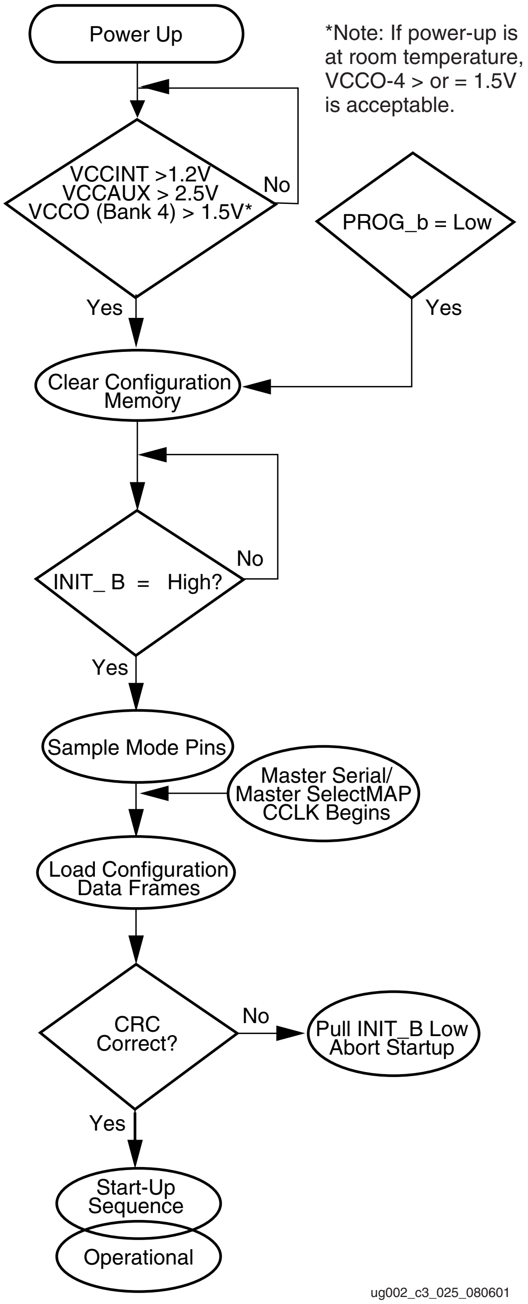


Figure 3-1: **Configuration Process**

Power Up

The V_{CCINT} power pins must be supplied with a 1.5 V source. (Refer to the **Virtex-II Data Sheet** for DC characteristics.) The IOB voltage input for Bank 4 (V_{CCO_4}) and the auxiliary voltage input (V_{CCAUX}) are also used as a logic input to the Power-On-Reset (POR) circuitry. Even if this bank is not being used, V_{CCO_4} must be connected to a 1.5 V or greater source.

Clearing Configuration Memory

In the memory clear phase, non-configuration I/O pins are 3-stated with optional pull-up resistors. The INIT_B and DONE pins are driven Low by the FPGA, and the memory is cleared. After PROG_B transitions High, memory is cleared twice and initialization can begin.

The INIT_B pin transitions High when the clearing of configuration memory is complete. A logic Low on the PROG_B input resets the configuration logic and holds the FPGA in the clear configuration memory state. When PROG_B is released, the FPGA continues to hold INIT_B Low until it has completed clearing all of the configuration memory. The minimum Low pulse time for PROG_B is defined by the $T_{PROGRAM}$ timing parameter. There is no maximum value. The power-up timing of configuration signals is shown in Figure 3-2 and the corresponding timing characteristics are listed in Table 3-3.

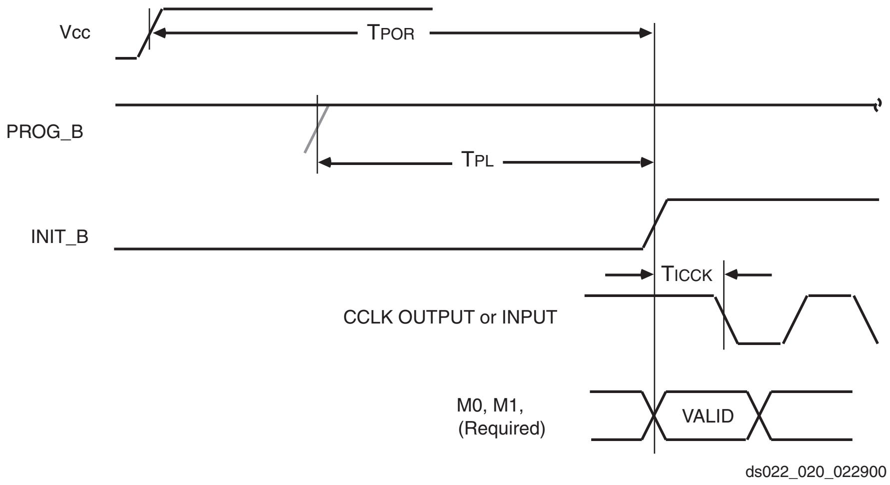


Figure 3-2: **Power-Up Timing Configuration Signals**

Table 3-3: **Power-Up Timing Characteristics**

Description	Symbol	Value	Units
Program Latency	T_{PL}	T_{PL} (2V8000)	4 µs per frame max
Power-on-Reset	T_{POR}	T_{PL} +2	ms, max
CCLK (output) Delay	T_{ICCK}	0.5	µs, min
		4.0	µs, max
Program Pulse Width	$T_{PROGRAM}$	300	ns, min

Initialization

For the initialization phase, the INIT_B pin is released, the mode pins are sampled, the appropriate pins become active, and the configuration process begins. It is possible to delay configuration by externally holding INIT_B Low.

Delaying Configuration

The INIT_B pin can also be held Low externally to delay configuration of the FPGA. The FPGA samples its mode pins on the rising edge of INIT_B. After INIT_B transitions to High, configuration can begin. No additional time-out or waiting periods are required, but configuration does not need to commence immediately after the transition of INIT_B. The configuration logic does not begin processing data until the synchronization word from the bitstream is loaded.

Loading Configuration Data

Once configuration begins, the target FPGA starts to receive data frames. Cyclic Redundancy Checking (CRC) is performed before and after the last data frame. CRC is also automatically checked after each block write to an internal data register (FDRI). If the CRC checks prove valid, the device start-up phase can begin.

If the CRC values do not match, INIT_B is asserted Low to indicate that a CRC error has occurred, startup is aborted, and the FPGA does not become active.

To reconfigure the device, the PROG_B pin should be asserted to reset the configuration logic. Recycling power also resets the FPGA for configuration. For more information on CRC calculation, see "Cyclic Redundancy Checking Algorithm" on page 395.

The details of loading configuration data in each of the five modes are discussed in the following sections:

- "Master Serial Programming Mode" on page 359
- "Master SelectMAP Programming Mode" on page 362
- "Slave Serial Programming Mode" on page 360
- "Slave SelectMAP Programming Mode" on page 364
- "JTAG/ Boundary Scan Programming Mode" on page 368

Device Startup

Device startup is a transition phase from the configuration mode to normal programmed device operation. Although the order of the start-up events are user programmable via software, the default sequence of events is as follows:

Upon completion of the start-up sequence, the target FPGA is operational.

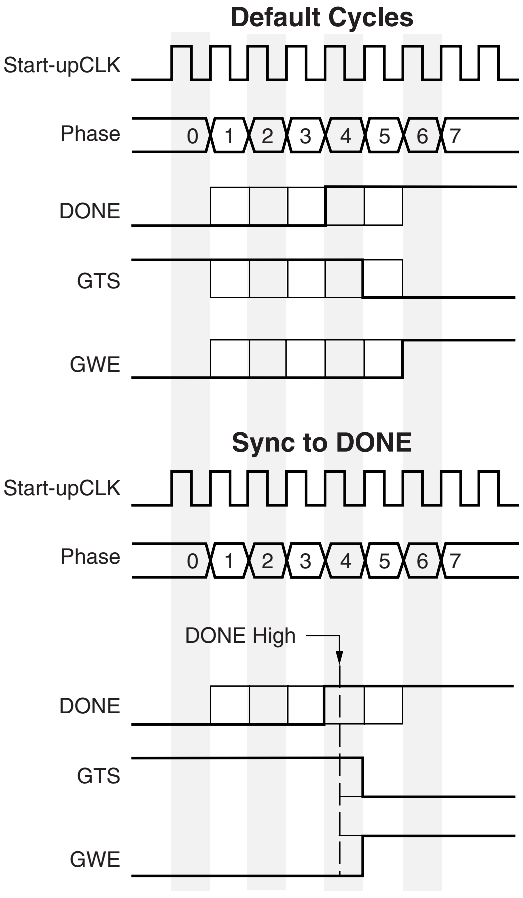
The Start-Up Sequencer is an 8-phase sequential state machine that counts from phase 0 to phase 7. (See Figure 3-3.)

The Start-Up Sequencer performs the following tasks:

- Release the DONE pin.
- Negate GTS, activating all of the I/Os.
- Assert GWE, allowing all RAMs and flip-flops to change state.
- Assert EOS. The End-Of-Start-Up flag is always set in phase 7. This is an internal flag that is not user accessible.

BitGen options control the order of the Start-Up Sequence. The default Start-Up Sequence is the bold line in Figure 3-3. The Start-Up Sequence can also be stalled at any phase until either DONE has been externally forced High, or a specified DCM or DCI has established LOCK. For details, see Appendix B, "BitGen and PROMGen Switches and Options."

At the cycle selected for the DONE to be released, the sequencer always waits in that state until the DONE is externally released. However, this does not delay the GTS or GWE if they are selected to be released prior to DONE. Therefore, DONE is selected first in the sequence for default settings.



x138_01_071100

Figure 3-3: **Default Start-Up Sequence**

Configuration Pins

Certain pins in the FPGA are designated for configuration and are listed in Table 3-4. Some pins are dedicated to the configuration function and others are dual-function pins that can be user I/O after configuration.

Table 3-4: **Configuration Pins**

Name	Direction	Driver Type	Description
Dedicated Pins			
CCLK	Input/Output	Active	Configuration clock. Output in Master mode.
PROG_B	Input		Asynchronous reset to configuration logic.
DONE	Input/Output	Active/ Open-Drain	Configuration status and start-up control.
M2, M1, M0	Input		Configuration mode selection.
HSWAP_EN	Input		I/O pullups during configuration.
TMS	Input		Boundary Scan Mode Select.
TCK	Input		Boundary Scan Clock.
TDI	Input		Boundary Scan Data Input.
TDO	Output	Active	Boundary Scan Data Output.
Dual Function Pins			
DIN (D0)	Input/Output	Active Bidirectional	Serial configuration data input/SelectMAP readback data output.
D1:D7	Input/Output	Active Bidirectional	SelectMAP configuration data input, readback data output.
CS_B	Input		Chip Select (SelectMAP mode only).
RDWR_B	Input		Active Low write select, read select (SelectMAP mode only).
BUSY/DOUT	Output	Active	Serial configuration data output for serial daisy-chains (active).
INIT_B	Input/Output	Open-Drain	Delay configuration, indicate configuration error.

Mixed Voltage Environments

Virtex-II devices have separate voltage sources. V_{CCINT} = 1.5 V powers the internal circuitry, V_{CCAUX} = 3.3 V powers the input buffers and auxiliary circuitry, and V_{CCO} (1.5, 1.8, 2.5, or 3.3 V) powers the IOB circuitry. SelectI/O is separated into eight banks of I/O groups. Each bank can be configured with one of several I/O standards. Refer to the Design Considerations section for I/O banking rules and available I/O standards. Before and during configuration, all I/O banks are set for the LVTTL standard, which requires an output voltage (V_{CCO}) of 3.3 V for normal operation.

If V_{CCO} is less than 3.3 V on banks 4 and 5, serial and SelectMAP configuration modes might have a lower frequency. (See Table 3-5).

Table 3-5: **Configuration Modes and V_{CCO} Voltages**

Configuration Mode	Pins Used	V_{CCO_4}	V_{CCO_5}
JTAG	Dedicated Pins	not a concern	not a concern
Serial	Dedicated Pins plus DOUT, DIN, and INIT	3.3 V	not a concern
SelectMAP	Dedicated Pins plus dual-function pins	3.3 V	3.3 V

Notes:

1. If less than 3.3 V ($V_{CCO\_4/5}$ = 2.5 V), the configuration frequency might be as low as half of the typical frequency.

All dedicated configuration pins are powered by V_{CCAUX}. All dual-function configuration pins are located within banks 4 and 5. As described under Configuration Process and Flow, the V_{CCO_4} input voltage is used as a logic input to the power-on-reset (POR) circuitry.

For JTAG configuration mode, JTAG inputs are independent of V_{CCO} and work between 2.5 V and 3.3 V TTL levels. The JTAG output (TDO) is sourced from V_{CCAUX}.

For serial configuration mode, V_{CCO_4} pins require a 3.3 V supply for output configuration pins to operate normally. In serial mode, all of the configuration pins are in bank 4.

For SelectMAP configuration mode, V_{CCO_4} and V_{CCO_5} pins require a 3.3 V supply for output configuration pins to operate normally. In SelectMAP mode, all of the configuration pins are in banks 4 and 5.

If the Virtex-II device is being configured in serial or SelectMAP mode, and banks 4 and 5 are being configured for an I/O standard that requires a V_{CCO} other than 3.3 V, then V_{CCO_4} and V_{CCO_5} (SelectMAP only) must be switched from 3.3 V and used during configuration at the same voltage required after configuration. If readback is performed using SelectMAP mode after configuration, then V_{CCO_4} and V_{CCO_5} require a 3.3 V supply after configuration, as well.

Configuration Solutions

Several configuration solutions are available to support Virtex-II, each targeted to specific application requirements. Guidance and support (application notes, reference designs, and so forth) is also available for designers looking to develop and implement their own configuration solution for Virtex FPGAs.

System Advanced Configuration Environment (System ACE™) Series

The System ACE series of configuration solutions offers a system-level configuration manager for designers using multiple FPGAs or FPGAs requiring multiple bitstreams. This solution combines standard industry Flash storage with Xilinx-designed configuration control. Features common to the entire System ACE family include:

- Support for multiple bitstreams
- Built-in support for embedded processors in FPGAs
- Support for reconfiguring, updating, or debugging systems over a network
- Built-in system interface
- Scalability (density) and re-useability (across many designs)
- Centralization of configuration control for reduced board space and simpler debugging
- Use of excess storage capacity for non-configuration, system storage

System ACE CF

System ACE CF (CompactFlash™) solution combines a standard CompactFlash Association (CFA) Type-I or Type-II memory module (CompactFlash or 1" disk drive) with a Xilinx-designed ACE Controller™ configuration control chip. See Figure 3-4.

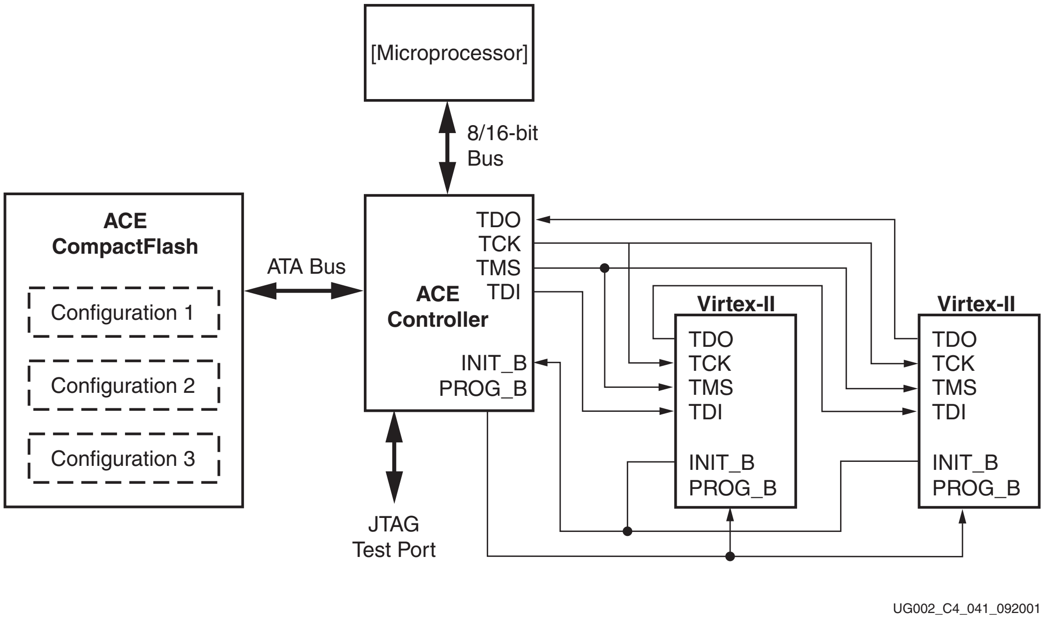


Figure 3-4: **System ACE CompactFlash and Controller**

The CompactFlash card stores an unlimited number of bitstreams and ranges in density from 128 Mb to 3 Gb. This card is capable of storing one large bitstream or several smaller bitstreams. If several bitstreams are used, the system can be set up so that individual bitstreams are callable as needed, allowing for dynamic reconfiguration of the Virtex-II device and other Xilinx FPGAs in the JTAG chain.

The ACE Controller drives bits through the FPGA JTAG chain and has three other ports:

- A port for interfacing with a microprocessor, a network, or a MultiLINX cable
- A port for interfacing with the CompactFlash card
- A port that provides access to the FPGA JTAG chain for FPGA testing or configuration via automatic test equipment or via desktop or third-party programmers

For further information on any System ACE product, visit the **www.xilinx.com/systemace** website.

System ACE Multi-Package Module (MPM)

System ACE MPM is a multi-package module consisting of a packaged standard Flash from AMD, a packaged FPGA, and a packaged configuration PROM, all in a 388-pin BGA package. The Flash stores configuration and other data, while the FPGA acts as an advanced configuration controller and is configured by the PROM. This solution provides high density and high-speed configuration capability in a single package, helping to simplify the design and manufacturing process. It is available in 16-Mbit, 32-Mbit, and 64-Mbit densities.

System ACE Soft Controller (SC)

System ACE SC is a downloadable version of the configuration controller found in System ACE MPM; versions are provided that support various standard Flash interfaces. System ACE SC provides all of the features of System MPM without the Single Package. It allows designers to use the Flash memory already in their system to store configuration data. The System ACE SC controller is available free of charge in the form of a PROM file that can be downloaded from the System ACE website. This pre-engineered solution is implemented by connecting up to four Flash chips on a board to an FPGA that will be used as a configuration controller and then downloading the controller file into a PROM. Figure 3-5 describes the controller for both System ACE MPM and System ACE SC.

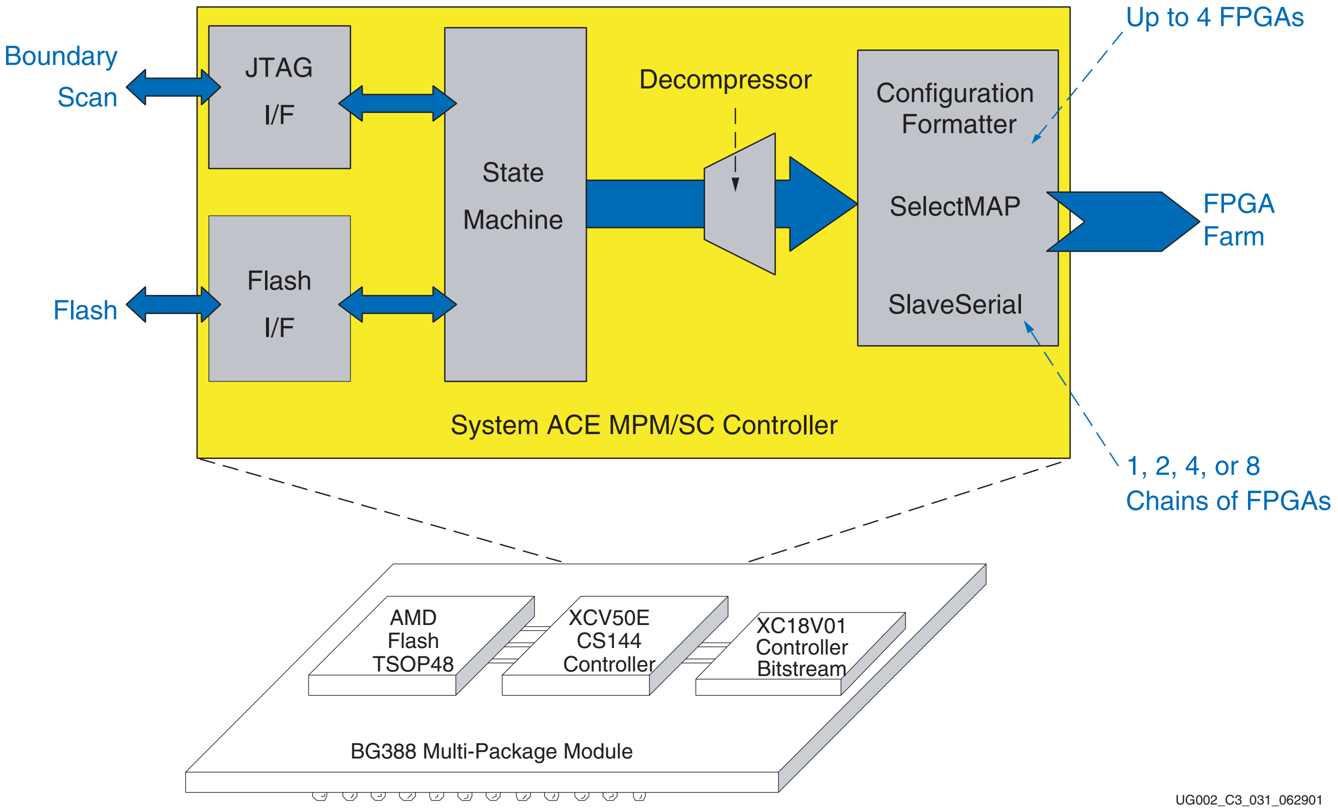


Figure 3-5: **System ACE MPM/SC Controller**

System ACE MPM and System ACE SC have these unique features:

- High speed configuration up to 154 Mb/sec
- Support for both SelectMAP (8-bit) (see Figure 3-6) and Slave Serial (1-bit) (see Figure 3-7) configuration
- Configuration of multiple FPGAs in parallel
- Bitstream compression for increased storage capability
- Storage of up to 8 different bitstreams

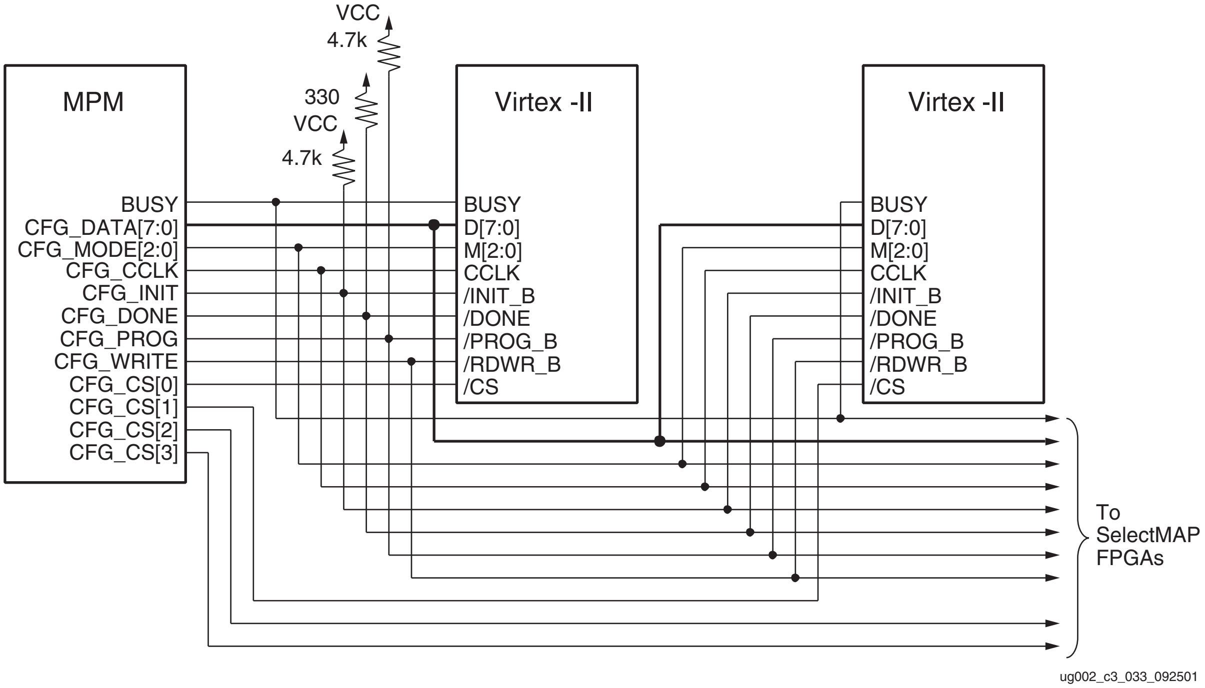


Figure 3-6: **SelectMAP (8-bit) Configuration**

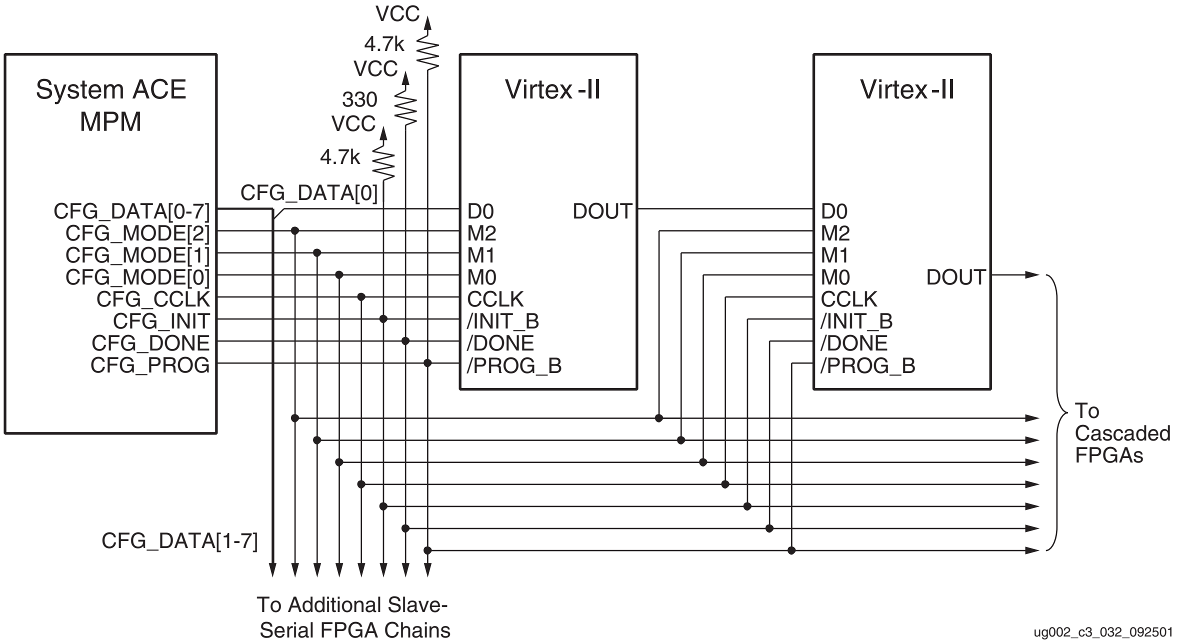


Figure 3-7: **Slave Serial (1-bit) Configuration**

Configuration PROMs

Using XC18V00 PROMs

The XC18V00 family of Flash in-system programmable (ISP) configuration PROMs offers the flexibility of re-programmability and multiple package offerings, combined with both serial and SelectMAP FPGA configurability. This family is JTAG programmable and ranges in density from 256 Kb to 4 Mb; these PROMs can also be cascaded to support larger bitstreams.

The 18V00 family offers data throughput rates of up to 264 Mb/s. It is also capable of triggering FPGA reconfiguration via a JTAG command. The parts can be JTAG programmed via cable, HW-130, or standard third party programmers. The XC18V00 PROMs are available in SO20, PC20, VQ44, and PC44 packages. Refer to Appendix C, "XC18V00 Series PROMs" for the latest version of the XC18V00 PROMs data sheet and package diagrams for the entire PROM family. See Table 3-8 to determine which PROMs go with which Virtex-II FPGAs.

Using XC17V00 PROMs

The XC17V00 family of one-time programmable (OTP) PROMs provides a proven, low-cost, compact, and pre-engineered configuration solution. Ranging from 1 Mb to 16 Mb, this family is also the PROM density leader; it can also be daisy-chained to support larger bitstreams. This family supports serial configuration of Virtex-II FPGAs; in addition, the XC17V08 and XC17V16 support SelectMAP configuration modes.

The XC17V00 family can be used for stabilized designs that are in a high-volume production flow and/or for designs requiring a low-cost solution. XC17V00 PROMs can be programmed either by using the HW-130 or by using a variety of third-party programmers. The XC17V00 PROMs are available in VO8, SO20, PC20, VQ44, and PC44 packages. Data sheets for PROMs are available at www.xilinx.com. See Table 3-8 to determine which PROMs go with which Virtex-II FPGAs and see Appendix C, "XC18V00 Series PROMs" for package diagrams.

Flash PROMs With a CPLD Configuration Controller

Some designers prefer to leverage existing Flash memory in their system to store the configuration bitstreams. A small CPLD-based configuration controller can provide the mechanism to access the bitstreams in the FLASH and deliver them quickly to Virtex-II devices. The following application notes describe the details for a serial or SelectMAP configuration architecture using FLASH memories and CPLDs:

- XAPP079: *Configuring* Xilinx *FPGAs Using an XC9500 CPLD and Parallel PROM* (**www.xilinx.com/apps/xappsumm.htm#xapp079**) describes an architecture that configures a chain of Virtex-II devices using Master-Serial mode. See Figure 3-8 for an example of FPGA configuration using a CPLD and a parallel PROM.

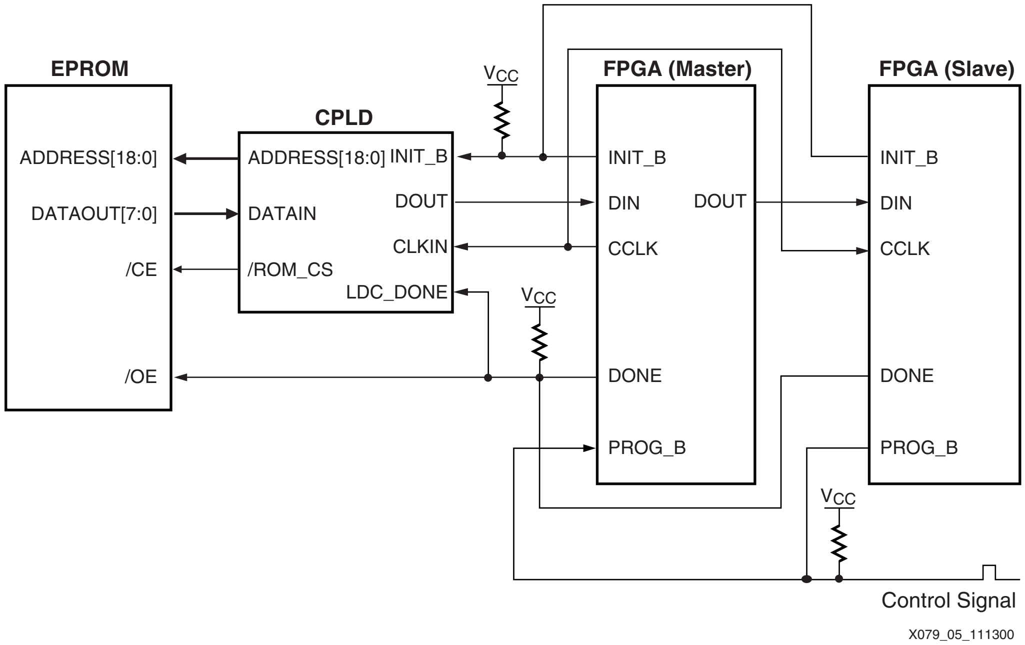


Figure 3-8: **Configuring Virtex-II Using a CPLD and Parallel PROM**

- XAPP137: *Configuring Virtex FPGAs From Parallel EPROMs With a CPLD* (**www.xilinx.com/apps/xappsumm.htm#xapp137**) describes an architecture that configures one or more Virtex-II devices using the Slave SelectMAP mode. See Figure 3-9 for an example of FPGA configuration using a CPLD and a parallel EPROM.

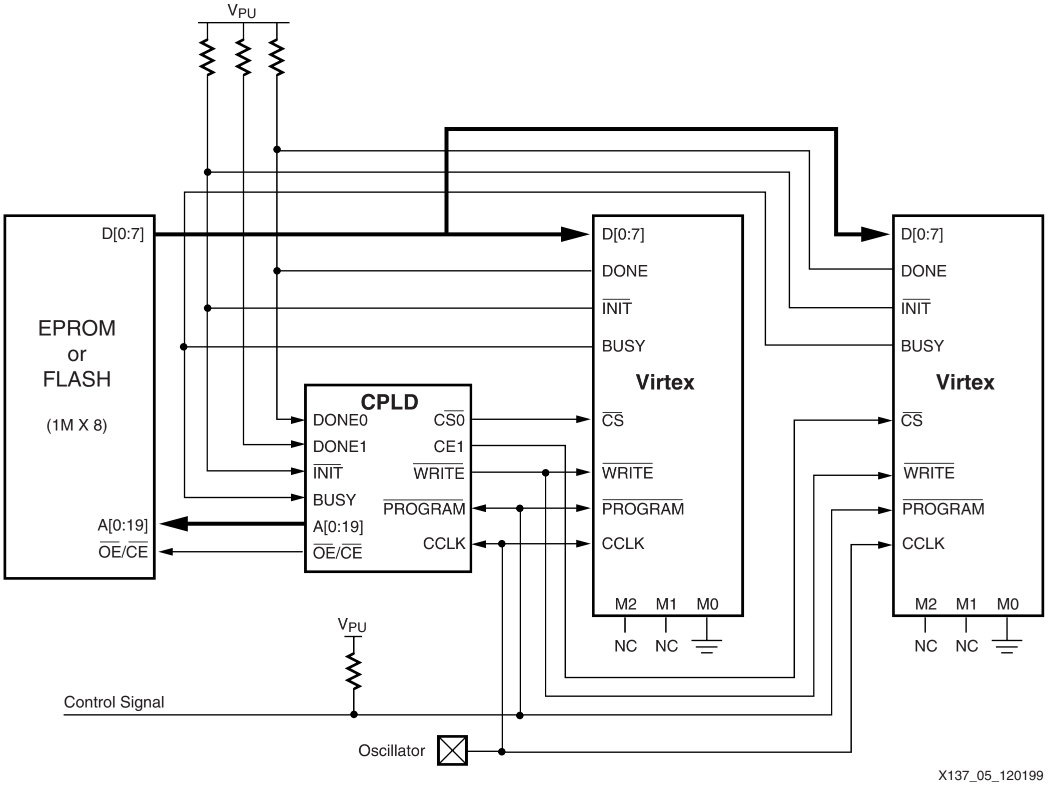


Figure 3-9: **Configuring Virtex-II from Parallel EPROMs**

Embedded Solutions

Using an Embedded Microcontroller

XAPP058: Xilinx *In-System Programming Using an Embedded Microcontroller* (**www.xilinx.com/apps/xappsumm.htm#xapp058**) describes a compact and robust process that (re)configures Virtex-II devices directly from a microprocessor through the JTAG test port of the Virtex-II device. The process additionally supports (re)configuration of XC18V00 ISP PROMs and CPLDs that reside on the JTAG scan chain. Portable, reference C-code is provided with the application note for rapid implementation.

Using IEEE Standard 1532

Systems that implement an IEEE Standard 1532 player are able to configure Virtex-II devices and any other 1532-compliant devices using the device BSDL file and 1532 data file.

Using MultiLINX or Other Cables

During the development or prototype design stage, designers can program their Virtex-II devices directly in system via the Xilinx Parallel Cable III or MultiLINX programming cables using the Xilinx JTAG Programmer software or Integrated Logic Analyzer (ILA) ChipScope software. The operating system (see Table 3-6) and configuration mode (see Table 3-7) determine the appropriate cable selection.

3

Table 3-6: **Xilinx Cable Operating System Support**

Cable	Connection	Windows 98	Windows NT	Windows 2000	Solaris	HP-UX
Parallel Cable III	Parallel Port	Supported	Supported	Supported	N/A	N/A
MultiLINX	USB	Supported	N/A	Supported	N/A	N/A
MultiLINX	RS-232	Supported	Supported	Supported	Supported	Supported

Table 3-7: **Xilinx Cable Configuration Mode Support**

Cable	JTAG	Slave Serial	Slave SelectMAP
Parallel Cable III	Supported	Supported	N/A
MultiLINX	Supported	Supported	Supported

SelectMAP is the fastest cable configuration mode. JTAG and serial modes provide roughly equivalent configuration speeds but are slower than SelectMAP.

PROM Selection Guide

Use Table 3-8 to determine which PROMs go with which Virtex-II FPGAs.

Table 3-8: **Using Virtex-II Devices With PROMs**

Virtex-II Device	Bitstream Length (bits)	PROM Family		PROM Package				
		18Vxx	17Vxx	V08	SO20	PC20	PC44	VQ44
XCV2V40	360,096	18V01	17V01	x(1)	x	x		x(2)
XCV2V80	635,296	18V01	17V01	x(1)	x	x		x(2)
XCV2V250	1,697,184	18V02	17V02			x(1)	x	x
XCV2V500	2,761,888	18V04	17V04			x(1)	x	x
XCV2V1000	4,082,592	18V04	17V04			x(1)	x	x
XCV2V1500	5,659,296	18V04 18V02	17V08				x	x
XCV2V2000	7,492,000	2, 18V04	17V08				x	x
XCV2V3000	10,494,368	3, 18V04	17V16				x	x
XCV2V4000	15,659,936	4, 18V04	17V16				x	x
XCV2V6000	21,849,504	5, 18V04 18V02	17V16 17V08				x	x
XCV2V8000	29,063,072	7, 18V04	2, 17V16				x	x

Notes:

1. 17Vxx only
2. 18Vxx only

Master Serial Programming Mode

In serial configuration mode, the FPGA is configured by loading one bit per CCLK cycle. In Master Serial mode, the FPGA drives the CCLK pin. In Slave Serial mode, the FPGAs CCLK pin is driven by an external source. In both serial configuration modes, the MSB of each data byte is always written to the DIN pin first.

The Master Serial mode is designed so the FPGA can be configured from a Serial PROM, Figure 3-10. The speed of the CCLK is selectable by BitGen options, see Appendix B, "BitGen and PROMGen Switches and Options." Be sure to select a CCLK speed supported by the PROM.

Figure 3-10 shows a Master Serial FPGA configuring from a PROM.

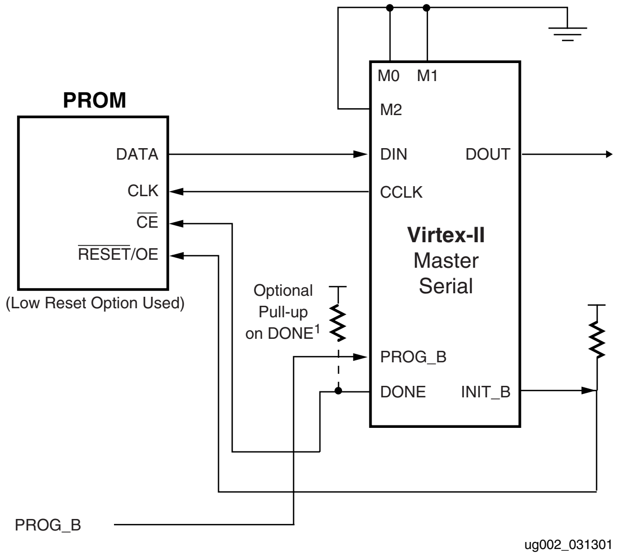


Figure 3-10: **Master Serial Mode Circuit Diagram**

Notes:

1. If the Virtex-II device has not selected the DriveDONE option, then an external pull-up resistor of 330Ω should be added to the DONE pin. This pull-up resistor is not needed if DriveDONE = Yes.

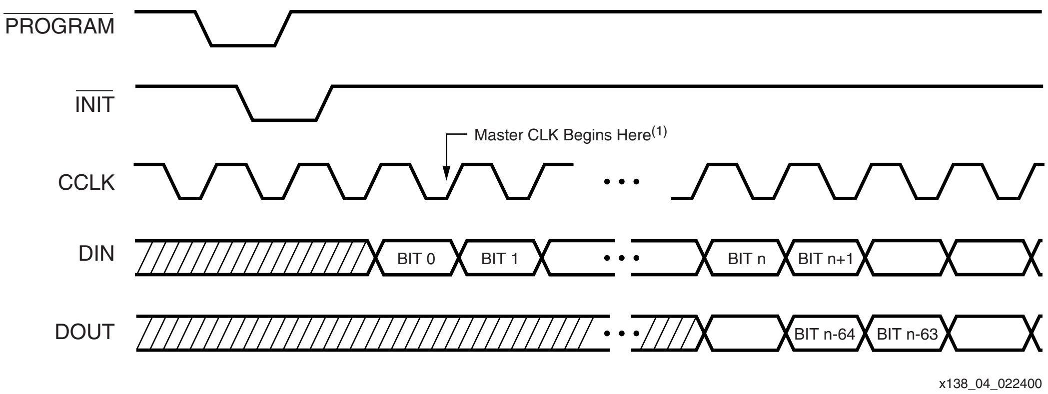


Figure 3-11: **Master Serial Configuration Cloaking Sequence**

Notes:

1. For Master configurations, the CCLK does not transition until after initialization as indicated by the arrow.

Slave Serial Programming Mode

In serial configuration mode, the FPGA is configured by loading one bit per CCLK cycle. In Slave Serial mode, the FPGAs CCLK pin is driven by an external source. In both serial configuration modes, the MSB of each data byte is always written to the DIN pin first.

The Slave Serial configuration mode allows for FPGAs to be configured from other logic devices, such as microprocessors, or in a daisy-chain fashion. Figure 3-12 shows a Master Serial FPGA configuring from a PROM with a Slave Serial FPGA in a daisy-chain with the Master.

Daisy-Chain Configuration

Virtex-II FPGAs can be used in a daisy-chain configuration only with XC4000X, SpartanXL, Spartan-II or other Virtex FPGAs. There are no restrictions on the order of the chain. However, if a Virtex-II FPGA is placed as the Master and a non-Virtex-II FPGA is placed as a slave, select a configuration CCLK speed supported by *all* devices in the chain.

The separate bitstreams for the FPGAs in a daisy-chain are required to be combined into a single PROM file, by using either the PROM File Formatter or the PROMGen utility (see Appendix B, "BitGen and PROMGen Switches and Options"). Separate PROM files can *not* be simply concatenated together to form a daisy-chain bitstream.

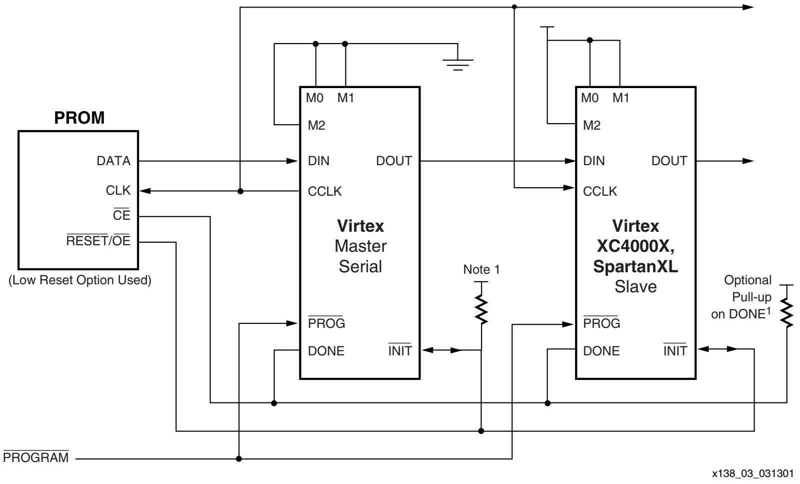


Figure 3-12: **Master/Slave Serial Mode Circuit Diagram**

Notes:

1. If none of the devices have been selected to DriveDONE, then an external pull-up resistor of 330 Ω should be added to the common DONE line. This pull-up resistor is not needed if DriveDONE = Yes. If used, DriveDONE should be selected only for the last device in the configuration chain.

The first device in the chain is the first to be configured. No data is passed onto the DOUT pin until all the data frames, start-up command, and CRC check have been loaded. CRC checks only include the data for the current device, not for any others in the chain. After finishing the first stream, data for the next device is loaded. The data for the downstream device appears on DOUT typically about 80 CCLK cycles after being loaded into DIN. This is due to internal packet processing. Each daisy-chained bitstream carries its own synchronization word. Nothing of the first bitstream is passed to the next device in the chain other than the daisy-chained configuration data.

The DONE_cycle must be set before GTS, or during the same cycle to guarantee each Virtex-II device to move to the operation state when all the DONE pins have been released. When daisy-chaining multiple devices, either set the last device in the chain to DriveDONE, or add external pull-up resistors to counteract the combined capacitive loading on DONE. If non-Virtex devices are included in the daisy-chain, it is important to set their bitstreams to SyncToDONE with BitGen options. For more information on Virtex BitGen options, see Appendix B, "BitGen and PROMGen Switches and Options.".

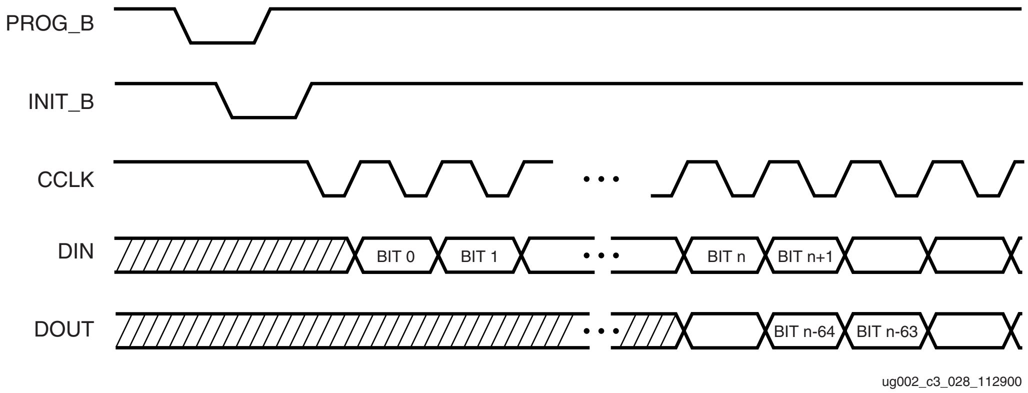


Figure 3-13: **Serial Configuration Clocking Sequence**

Notes:

1. For Slave configurations, a free running CCLK can be used, as shown in Figure 3-13.

Table 3-9: **Master/Slave Serial Mode Programming Switching**

	Description	Symbol	Values	Units
CCLK	DIN setup/hold, slave mode	T_{DCC}/T_{CCD}	5.0/0.0	ns, min
	DIN setup/hold, master mode	T_{DSCK}/T_{SCKD}	5.0/0.0	ns, min
	DOUT	T_{CCO}	12.0	ns, max
	High time	T_{CCH}	5.0	ns, min
	Low time	T_{CCL}	5.0	ns, min
	Maximum Frequency	F_{CC_SERIAL}	66	MHz, max
	Frequency Tolerance, master mode with respect to nominal		+45% -30%	

Master SelectMAP Programming Mode

The SelectMAP mode provides an 8-bit bidirectional data bus interface to the Virtex-II configuration logic that can be used for both configuration and readback. Virtex-II devices can not be serially daisy-chained when the SelectMAP interface is used. However, they can be connected in a parallel-chain as shown in Figure 3-16. The DATA pins (D0:D7), CCLK, RDWR_B, BUSY, PROG_B, DONE, and INIT_B can be connected in common between all of the devices. CS_B inputs should be kept separate so each device can be accessed individually. If all devices are to be configured with the same bitstream, readback is not being used, and CCLK is less than $F_{CC_}$SelectMAP, the CS_B pins can be connected to a common line so the devices are configured simultaneously.

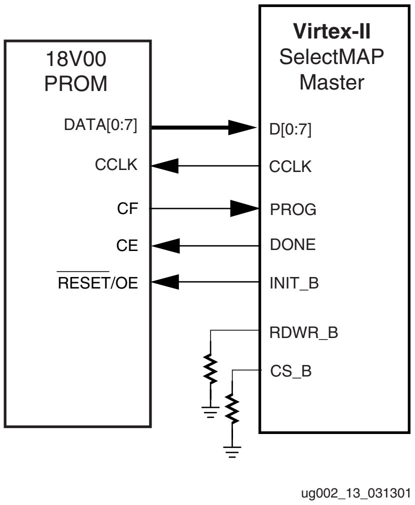


Figure 3-14: **Virtex-II Interfaced With an 18V00 PROM**

Notes:

1. If none of the Virtex-II devices have been selected to DriveDONE, add an external 330 Ω pull-up resistor to the common DONE line. This pull-up resistor is not needed if DriveDONE is selected. If used, DriveDONE should be selected only for the last device in the configuration chain.

The following pins are involved in Master SelectMAP configuration mode:

DATA Pins (D[0:7])

The D0 through D7 pins function as a bidirectional data bus in the SelectMAP mode. Configuration data is written to the bus, and readback data is read from the bus. The bus direction is controlled by the RDWR_B signal. see "Configuration Details" on page 387. The D0 pin is considered the MSB of each byte.

RDWR_B

When asserted Low, the RDWR_B signal indicates that data is being written to the data bus. When High, the RDWR_B signal indicates that data is being read from the data bus.

CS_B

The Chip Select input (CS_B) enables the SelectMAP data bus. To write or read data onto or from the bus, the CS_B signal must be asserted Low. When CS_B is High, Virtex-II devices do not drive onto or read from the bus.

CCLK

The CCLK pin is a clock output in the Master SelectMAP interface. It synchronizes all loading and reading of the data bus for configuration and readback. The CCLK pin is driven by the FPGA.

Data Loading

To load data in the Master SelectMAP mode, a data byte is loaded on every rising CCLK edge as shown in Figure 3-15. If the CCLK frequency is less than F_{CC}_SelectMAP, this can be done without handshaking. For frequencies above F_{CC}_SelectMAP, the BUSY signal must be monitored. If BUSY is High, the current byte must be reloaded when BUSY is Low.

The first byte can be loaded on the first rising CCLK edge that INIT_B is High, and when both CS_B and RDWR_B are asserted Low. CS_B and RDWR_B can be asserted anytime before or after INIT_B has gone High. However, the SelectMAP interface is not active until after INIT_B has gone High. The order of CS_B and RDWR_B does not matter, but RDWR_B must be asserted throughout configuration. If RDWR_B is de-asserted before all data has been loaded, the FPGA aborts the operation. To complete configuration, the FPGA must be reset by PROG_B and reconfigured with the entire stream. For applications that need to de-assert RDWR_B between bytes, see "Controlled CCLK" on page 367.

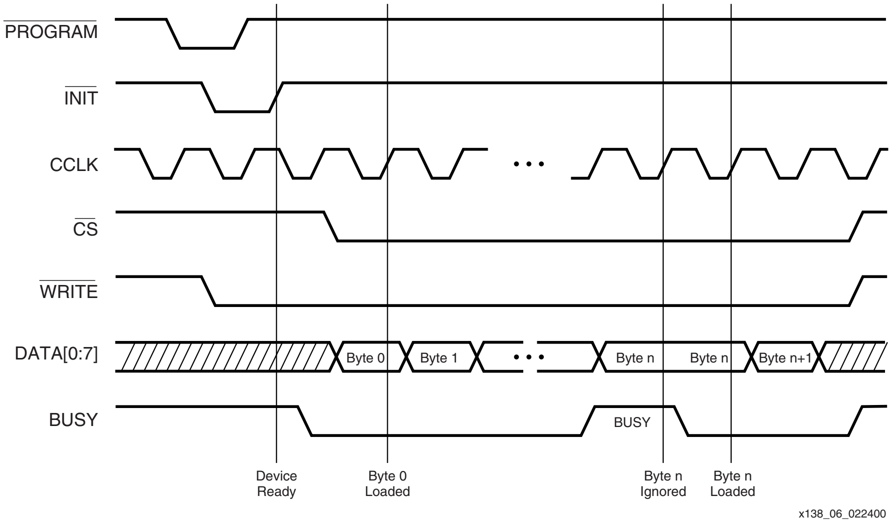


Figure 3-15: **Data Loading in SelectMAP**

Slave SelectMAP Programming Mode

The SelectMAP mode provides an 8-bit bidirectional data bus interface to the Virtex-II configuration logic that can be used for both configuration and readback. Virtex-II devices can not be serially daisy-chained when the SelectMAP interface is used. However, they can be connected in a parallel-chain as shown in Figure 3-16. The DATA pins (D0:D7), CCLK, RDWR_B, BUSY, PROG_B, DONE, and INIT_B can be connected in common between all of the devices. CS_B inputs should be kept separate so each device can be accessed individually. If all devices are to be configured with the same bitstream, readback is not being used, and CCLK is less than F_{CC}_SelectMAP, the CS_B pins can be connected to a common line so the devices are configured simultaneously.

Although Figure 3-16 does not show a control module for the SelectMAP interface, the SelectMAP interface is typically driven by a processor, micro controller, or some other logic device such as an FPGA or a CPLD.

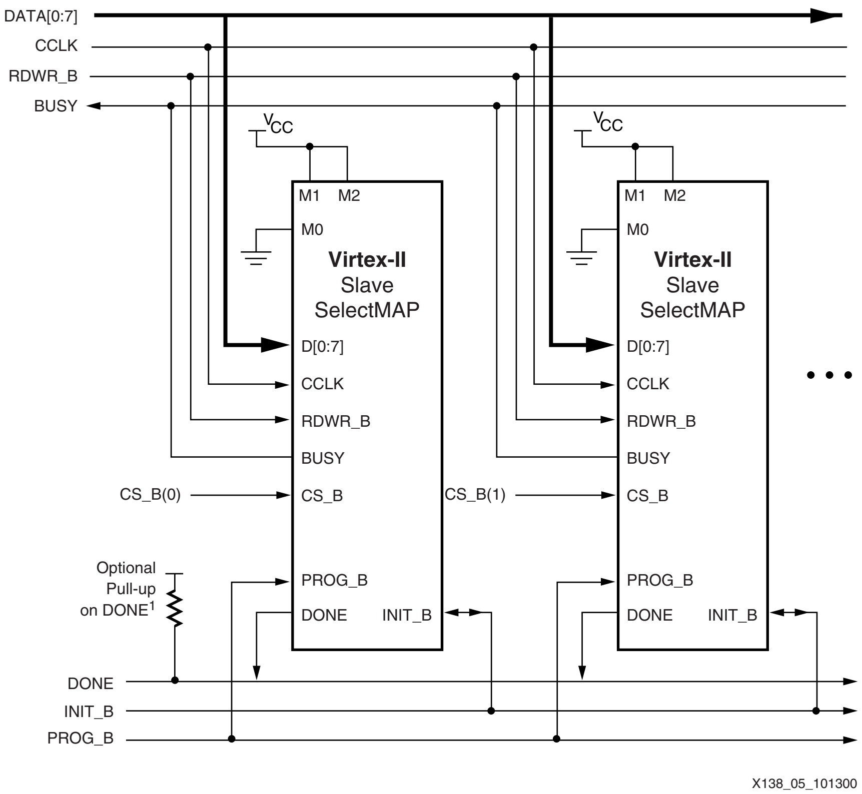


Figure 3-16: **Slave SelectMAP Mode Circuit Diagram**

Notes:

1. If none of the Virtex-II devices have been selected to DriveDONE, add an external 330 Ω pull-up resistor to the common DONE line. This pull-up resistor is not needed if DriveDONE = Yes. If used, DriveDONE should be selected only for the last device in the configuration chain.

The following pins are involved in Slave SelectMAP configuration mode:

DATA Pins (D[0:7])

The D0 through D7 pins function as a bidirectional data bus in the SelectMAP mode. Configuration data is written to the bus, and readback data is read from the bus. The bus direction is controlled by the RDWR_B signal. see "Configuration Details" on page 387.. The D0 pin is considered the MSB of each byte.

RDWR_B

When asserted Low, the RDWR_B signal indicates that data is being written to the data bus. When asserted High, the RDWR_B signal indicates that data is being read from the data bus.

CS_B

The Chip Select input (CS_B) enables the SelectMAP data bus. To write or read data onto or from the bus, the CS_B signal must be asserted Low. When CS_B is High, Virtex-II devices do not drive onto or read from the bus.

BUSY

When CS_B is asserted, the BUSY output indicates when the FPGA can accept another byte. If BUSY is Low, the FPGA reads the data bus on the next rising CCLK edge where both CS_B and RDWR_B are asserted Low. If BUSY is High, the current byte is ignored and must be reloaded on the next rising CCLK edge when BUSY is Low. When CS_B is *not* asserted, BUSY is 3-stated.

BUSY is only necessary for CCLK frequencies above $F_{CC_}$SelectMAP. For frequencies at or below $F_{CC_}$SelectMAP, BUSY is ignored, see “Data Loading” on page 363. For parallel chains, as shown in Figure 3-16, where the same bitstream is to be loaded into multiple devices simultaneously, BUSY should not be used. Thus, the maximum CCLK frequency for such an application must be less than $F_{CC_}$SelectMAP.

CCLK

Unlike the Master SelectMAP mode of configuration, the CCLK pin is an input in the Slave SelectMAP mode interface. The CCLK signal synchronizes all loading and reading of the data bus for configuration and readback. Additionally, the CCLK drives internal configuration circuitry. The CCLK can be driven either by a free running oscillator or an externally-generated signal.

Several scenarios exist when configuring the FPGA in SelectMAP mode, depending on the source of CCLK.

Free-Running CCLK

A free-running oscillator can be used to drive Virtex-II CCLK pins. For applications that can provide a continuous stream of configuration data, refer to the timing diagram discussed in "Data Loading" on page 363. For applications that cannot provide a continuous data stream, missing the clock edges, refer to the timing diagram discussed in "Non-Contiguous Data Strobe" on page 366. An alternative to a free-running CCLK is discussed in "Controlled CCLK" on page 367.

Express-Style Loading

In express-style loading, a data byte is loaded on every rising CCLK edge as shown in Figure 3-17. If the CCLK frequency is less than $F_{CC_}$SelectMAP, this can be done without handshaking. For frequencies above $F_{CC_}$SelectMAP, the BUSY signal must be monitored. If BUSY is High, the current byte must be reloaded when BUSY is Low.

The first byte can be loaded on the first rising CCLK edge that INIT_B is High, and when both CS_B and RDWR_B are asserted Low. CS_B and RDWR_B can be asserted anytime before or after INIT_B has gone High. However, the SelectMAP interface is not active until after INIT_B has gone High. The order of CS_B and RDWR_B does not matter, but RDWR_B must be asserted throughout configuration. If RDWR_B is de-asserted before all data has been loaded, the FPGA aborts the operation. To complete configuration, the FPGA must be reset by PROG_B and reconfigured with the entire stream.

For applications that need to de-assert RDWR_B between bytes, see "Controlled CCLK" on page 367.

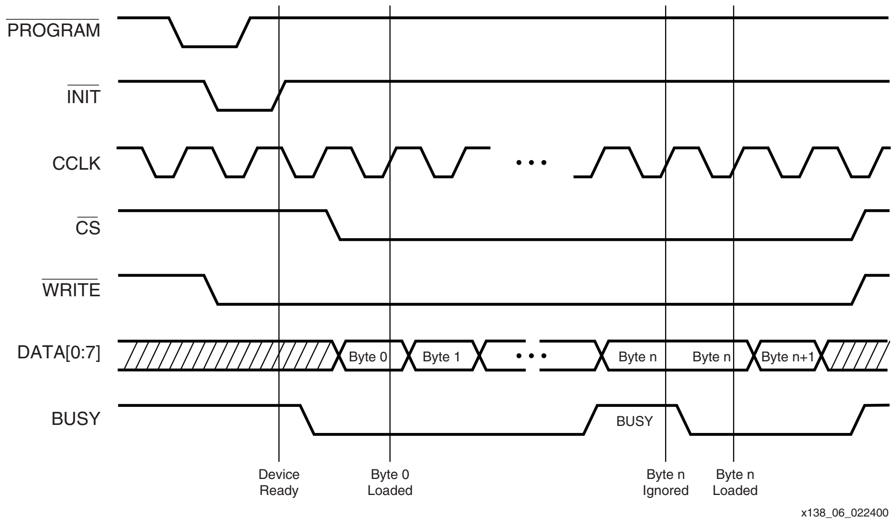


Figure 3-17: **"Express Style" Continuous Data Loading in SelectMAP**

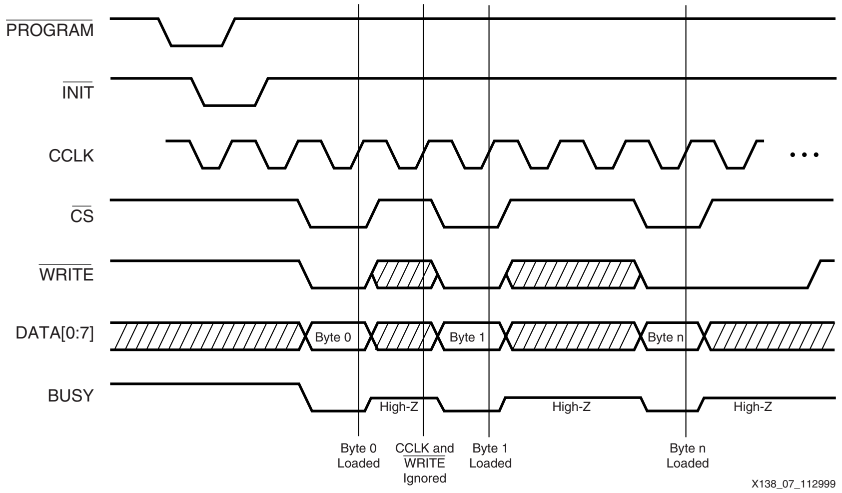


Figure 3-18: **Separating Data Loads by Multiple CCLK Cycles Using CS_B**

Non-Contiguous Data Strobe

In applications where multiple clock cycles might be required to access the configuration data before each byte can be loaded into the SelectMAP interface, data might not be ready for each consecutive CCLK edge. In such a case, the CS_B signal can be de-asserted until the next data byte is valid on the DATA[0:7] pins. This is demonstrated in Figure 3-18. While CS_B is High, the SelectMAP interface does not expect any data and ignores all CCLK transitions. However, RDWR_B must continue to be asserted while CS_B is asserted. If RDWR_B is High during a positive CCLK transition while CS_B is asserted, the FPGA aborts the operation. For applications that need to de-assert the RDWR_B signal without de-asserting CS_B, see "Controlled CCLK".

Controlled CCLK

Some applications require that RDWR_B be de-asserted between the loading of configuration data bytes asynchronously from the CS_B. Typically, this would be due to the RDWR_B signal being a common connection to other devices on the board, such as memory storage elements. In such a case, driving CCLK as a controlled signal instead of a free-running oscillator makes this type of operation possible. In Figure 3-19, the CCLK, CS_B, and RDWR_B are asserted Low while a data byte becomes active. Once the CCLK has gone High, the data is loaded. RDWR_B can be de-asserted and re-asserted as many times as necessary, just as long as it is Low before the next rising CCLK edge.

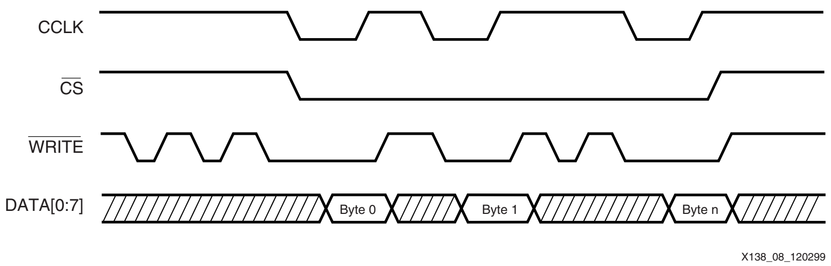


Figure 3-19: **Controlling CCLK for RDWR_B De-Assertion**

Table 3-10: **SelectMAP Write Timing Characteristics**

	Description	Symbol	Value	Units
CCLK	D_{0-7} Setup/Hold	T_{SMDCC}/T_{SMCCD}	5.0/0.0	ns, min
	CS_B Setup/Hold	T_{SMCSCC}/T_{SMCCCS}	7.0/0.0	ns, min
	RDWR_B Setup/Hold	T_{SMCCW}/T_{SMWCC}	7.0/0.0	ns, min
	BUSY Propagation Delay	T_{SMCKBY}	12.0	ns, max
	Maximum Frequency	$F_{CC_}$SelectMAP	66	MHz, max
	Maximum Frequency with no handshake	F_{CCNH}	66	MHz, max

JTAG/ Boundary Scan Programming Mode

Introduction

Virtex-II devices support the new IEEE 1532 standard for In-System Configuration (ISC), based on the IEEE 1149.1 standard. The IEEE 1149.1 Test Access Port and Boundary-Scan Architecture is commonly referred to as JTAG. JTAG is an acronym for the Joint Test Action Group, the technical subcommittee initially responsible for developing the standard. This standard provides a means to assure the integrity of individual components and the interconnections between them at the board level. With increasingly dense multi-layer PC boards, and more sophisticated surface mounting techniques, boundary-scan testing is becoming widely used as an important debugging standard.

Devices containing boundary-scan logic can send data out on I/O pins in order to test connections between devices at the board level. The circuitry can also be used to send signals internally to test the device specific behavior. These tests are commonly used to detect opens and shorts at both the board and device level.

In addition to testing, boundary-scan offers the flexibility for a device to have its own set of user-defined instructions. The added common vendor specific instructions, such as configure and verify, have increased the popularity of boundary-scan testing and functionality.

Boundary-Scan for Virtex-II Devices Using IEEE Standard 1149.1

The Virtex-II family is fully compliant with the IEEE Standard 1149.1 Test Access Port and Boundary-Scan Architecture. The architecture includes all mandatory elements defined in the IEEE 1149.1 Standard. These elements include the Test Access Port (TAP), the TAP controller, the instruction register, the instruction decoder, the boundary-scan register, and the bypass register. The Virtex-II family also supports some optional instructions; the 32-bit identification register, and a configuration register in full compliance with the standard. Outlined in the following sections are the details of the JTAG architecture for Virtex-II devices.

Test Access Port

The Virtex-II TAP contains four mandatory dedicated pins as specified by the protocol (Table 3-11).

Table 3-11: **Virtex-II TAP Controller Pins**

Pin	Description
TDI	Test Data In
TDO	Test Data Out
TMS	Test Mode Select
TCK	Test Clock

There are three input pins and one output pin to control the 1149.1 boundary-scan TAP controller. There are optional control pins, such as $\overline{\text{TRST}}$ (Test Reset) and enable pins, which might be found on devices from other manufacturers. It is important to be aware of these optional signals when interfacing Xilinx devices with parts from different vendors, because they might need to be driven.

The TAP controller is a 16-state state machine shown in Figure 3-20. The four mandatory TAP pins are outlined below.

- TMS - This pin determines the sequence of states through the TAP controller on the rising edge of TCK. TMS has an internal resistive pull-up to provide a logic High if the pin is not driven.

- TCK - This pin is the JTAG test clock. It sequences the TAP controller and the JTAG registers in the Virtex-II devices.
- TDI - This pin is the serial input to all JTAG instruction and data registers. The state of the TAP controller and the current instruction held in the instruction register determine which register is fed by the TDI pin for a specific operation. TDI has an internal resistive pull-up to provide a logic High to the system if the pin is not driven. TDI is applied into the JTAG registers on the rising edge of TCK.
- TDO - This pin is the serial output for all JTAG instruction and data registers. The state of the TAP controller and the current instruction held in the instruction register determine which register (instruction or data) feeds TDO for a specific operation. TDO changes state on the falling edge of TCK and is only active during the shifting of instructions or data through the device. This pin is 3-stated at all other times.

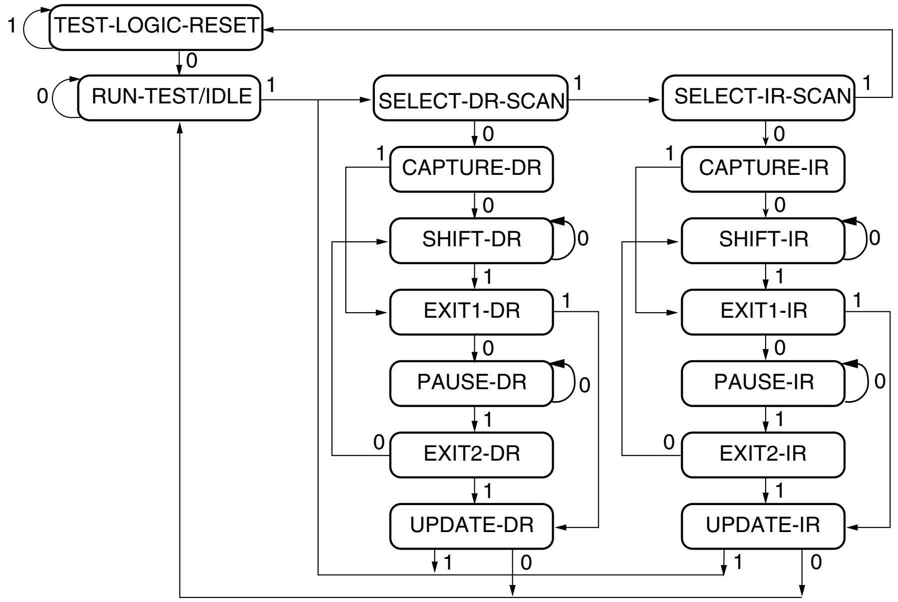
Notes:

As specified by the IEEE Standard, the TMS and TDI pins all have internal pull-up resistors. These internal pull-up resistors of 50-150 kΩ are active, regardless of the mode selected.

JTAG inputs are independent of V_{CCO} and work between 2.5 V and 3.3 V TTL levels. TDO is sourced from V_{CCAUX}.

TAP Controller

Figure 3-20 diagrams a 16-state finite state machine. The four TAP pins control how data is scanned into the various registers. The state of the TMS pin at the rising edge of TCK determines the sequence of state transitions. There are two main sequences, one for shifting data into the data register and the other for shifting an instruction into the instruction register.



NOTE: The value shown adjacent to each state transition in this figure represents the signal present at TMS at the time of a rising edge at TCK.

x139_01_112399

Figure 3-20: **State Diagram for the TAP Controller**

Boundary-Scan Instruction Set

To determine the operation to be invoked, an instruction is loaded into the Instruction Register (IR). The Instruction Register is 6 bits long in Virtex-II devices to support the new IEEE Standard 1532 for In-System Configurable (ISC) devices. Table 3-12 lists the available instructions for Virtex-II devices.

Table 3-12: **Virtex-II Boundary Scan Instructions**

Boundary Scan Command	Binary Code (5:0)	Description
EXTEST	000000	Enables boundary-scan EXTEST operation
SAMPLE	000001	Enables boundary-scan SAMPLE operation
USER1	000010	Access user-defined register 1
USER2	000011	Access user-defined register 2
CFG_OUT	000100	Access the configuration bus for readback
CFG_IN	000101	Access the configuration bus for configuration
INTEST	000111	Enables boundary-scan INTEST operation
USERCODE	001000	Enables shifting out user code
IDCODE	001001	Enables shifting out of ID code
HIGHZ	001010	3-states output pins while enabling the bypass register
JSTART	001100	Clocks the start-up sequence when StartClk is TCK
JSHUTDOWN	001101	Clocks the shutdown sequence
BYPASS	111111	Enables BYPASS
JPROG_B	001011	Equivalent to and has the same affect as PROG_B
RESERVED	All other codes	Xilinx reserved instructions

The mandatory IEEE 1149.1 commands are supported in Virtex-II devices, as well as several Xilinx vendor-specific commands. Virtex-II devices have a powerful command set. The EXTEST, INTEST, SAMPLE/PRELOAD, BYPASS, IDCODE, USERCODE, and HIGHZ instructions are all included. The TAP also supports two internal user-defined registers (USER1 and USER2) and configuration/readback of the device. The Virtex-II boundary-scan operations are independent of mode selection. The boundary-scan mode in Virtex-II devices overrides other mode selections. For this reason, boundary-scan instructions using the boundary-scan register (SAMPLE/PRELOAD, INTEST, EXTEST) must not be performed during configuration. All instructions except USER1 and USER2 are available before a Virtex-II device is configured. After configuration, all instructions are available.

JSTART and JSHUTDOWN are instructions specific to the Virtex-II architecture and configuration flow. As described in Table 3-12, the JSTART and JSHUTDOWN instructions clock the startup sequence when the appropriate bitgen option is selected. The instruction does not work correctly without the correct bitgen option selected.

```
bitgen -g startupclk:jtagclk designName.ncd
```

For details on the standard boundary-scan instructions EXTEST, INTEST, and BYPASS, refer to the IEEE Standard. The user-defined registers (USER1/USER2) are described in "USER1, USER2 Registers" on page 374.

Boundary-Scan Architecture

Virtex-II device registers include all registers required by the IEEE 1149.1 Standard. In addition to the standard registers, the family contains optional registers for simplified testing and verification (Table 3-13).

Table 3-13: **Virtex-II JTAG Registers**

Register Name	Register Length	Description
Instruction register	6 bits	Holds current instruction OPCODE and captures internal device status.
Boundary scan register	3 bits per I/O	Controls and observes input, output, and output enable.
Bypass register	1 bit	Device bypass.
Identification register	32 bits	Captures device ID.
JTAG configuration register	64 bits	Allows access to the configuration bus when using the CFG_IN or CFG_OUT instructions.
USERCODE register	32 bits	Captures user-programmable code

Boundary-Scan Register

The test primary data register is the boundary-scan register. Boundary-scan operation is independent of individual IOB configurations. Each IOB, bonded or un-bonded, starts as bidirectional with 3-state control. Later, it can be configured to be an input, output, or 3-state only. Therefore, three data register bits are provided per IOB (Figure 3-21).

When conducting a data register (DR) operation, the DR captures data in a parallel fashion during the CAPTURE-DR state. The data is then shifted out and replaced by new data during the SHIFT-DR state. For each bit of the DR, an update latch is used to hold the input data stable during the next SHIFT-DR state. The data is then latched during the UPDATE-DR state when TCK is Low.

The update latch is opened each time the TAP Controller enters the UPDATE-DR state. Care is necessary when exercising an INTEST or EXTEST to ensure that the proper data has been latched before exercising the command. This is typically accomplished by using the SAMPLE/PRELOAD instruction.

Consider internal pull-up and pull-down resistors when developing test vectors for testing opens and shorts. The boundary-scan mode determines if the IOB has a pull-up resistor. Figure 3-21 is a representation of Virtex-II Boundary-Scan Architecture.

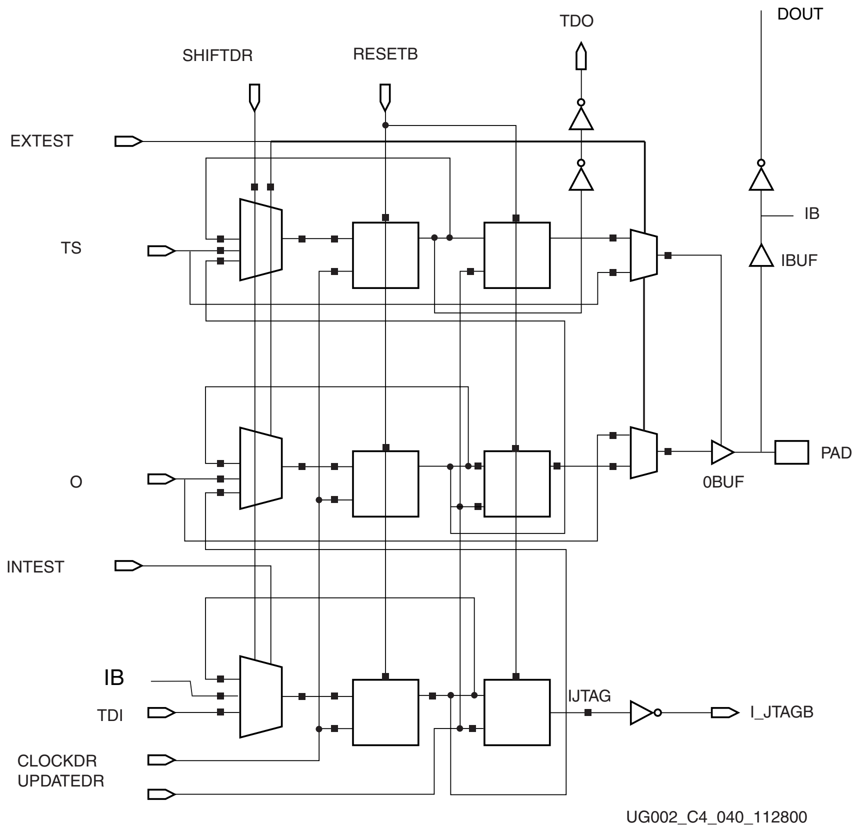


Figure 3-21: **Virtex Series Boundary Scan Logic**

Bit Sequence

The order in each non-TAP IOB is described in this section. The input is first, then the output, and finally the 3-state IOB control. The 3-state IOB control is closest to the TDO. The input-only pins contribute only the input bit to the boundary-scan I/O data register. The bit sequence of the device is obtainable from the "Boundary-Scan Description Language Files" (BSDL files) for the Virtex family. These files can be obtained from the Xilinx software download area. The bit sequence is independent of the design. It always has the same bit order and the same number of bits.

Bypass Register

The other standard data register is the single flip-flop BYPASS register. It passes data serially from the TDI pin to the TDO pin during a bypass instruction. This register is initialized to zero when the TAP controller is in the CAPTURE-DR state.

Instruction Register

The instruction register is a 6-bit register that loads the OPCODE necessary for the Virtex-II boundary-scan instruction set. This register loads the current OPCODE and captures internal device status.

Configuration Register (Boundary-Scan)

The configuration register is a 64-bit register. This register allows access to the configuration bus and readback operations.

Identification Register

Virtex devices have a 32-bit identification register, commonly referred to as the IDCODE register. This register is based upon IEEE Standard 1149.1 and allows easy identification of the part being tested or programmed via boundary scan.

Virtex-II Identification Register

The Virtex-II JTAG ID Code register has the following format.

```
3322 2222222 211111111 110000000000  ←
1098 7654321 098765432 109876543210  ←  bit positions(00 to 31)
vvvv:fffffff:aaaaaaaaa:ccccccccccc1
```

where

```
v is the revision code and

f is the 7-bit family code = 0001000 0x08

a is the number of array rows in the part expressed in 9 bits.
```

```
   XC2V40 =   8 =   0x08
   XC2V80 =  10 =   0x010
  XC2V250 =  24 =   0x018
  XC2V500 =  32 =   0x020
 XC2V1000 =  40 =   0x028
 XC2V1500 =  48 =   0x030
 XC2V2000 =  56 =   0x038
 XC2V3000 =  64 =   0x040
 XC2V4000 =  80 =   0x050
 XC2V6000 =  96 =   0x060
 XC2V8000 = 112 =   0x070
```

`c is the company code = 00001001001 = 0x049*`

*Since the last bit of the JTAG IDCODE is always one, the last three hex digits appear as `0x093`.

	vvvv	ffff	fff	a	aaaa	aaaa	cccc	cccc	cccc
XC2V250		0001	000	0	0001	1000	0000	1001	0011
	v	1	0		1	8	0	9	3
XC2V500	v	1	0		2	0	0	9	3

ID Codes assigned to Virtex-II FPGAs are shown in Table 3-14.

Table 3-14: **Virtex-II Device ID Codes**

FPGA	IDCODE
XC2V40	v01008093
XC2V80	v01010093
XC2V250	v01018093
XC2V500	v01020093
XC2V000	v01028093
XC2V1500	v01030093
XC2V2000	v01038093
XC2V3000	v01040093
XC2V4000	v01050093
XC2V6000	v01060093
XC2V8000	v01070093

Notes:

1. The “v” in the IDCODE is the revision code field.

USERCODE Register

USERCODE is supported in the Virtex family as well. This register allows a user to specify a design-specific identification code. The USERCODE can be programmed into the device and read back for verification at a later time. The USERCODE is embedded into the bitstream during bitstream generation (bitgen -g UserID option) and is valid only after configuration.

USER1, USER2 Registers

The USER1 and USER2 registers are only valid after configuration. These two registers must be defined by the user within the design. These registers can be accessed after they are defined by the TAP pins.

The BSCAN_VIRTEX2 library macro is required when creating these registers. This symbol is only required for driving internal scan chains (USER1 and USER2). The BSCAN_VIRTEX2 macro provides two user pins (SEL1 and SEL2) for determining usage of USER1 or USER2 instructions respectively. For these instructions, two corresponding pins (TDO1 and TDO2) allow user scan data to be shifted out of TDO. In addition, there are individual clock pins (DRCK1 and DRCK2) for each user register. There is a common input pin (TDI) and shared output pins that represent the state of the TAP controller (RESET, SHIFT, and UPDATE). Unlike earlier FPGA families that required the BSCAN macro to dedicate TAP pins for boundary scan, Virtex-II TAP pins are dedicated and do not require the BSCAN_VIRTEX2 macro for normal boundary-scan instructions or operations.

Note that these are user-defined registers. The example (Figure 3-22) is one of many implementations. For HDL, the BSCAN_VIRTEX2 macro needs to be instantiated in the design.

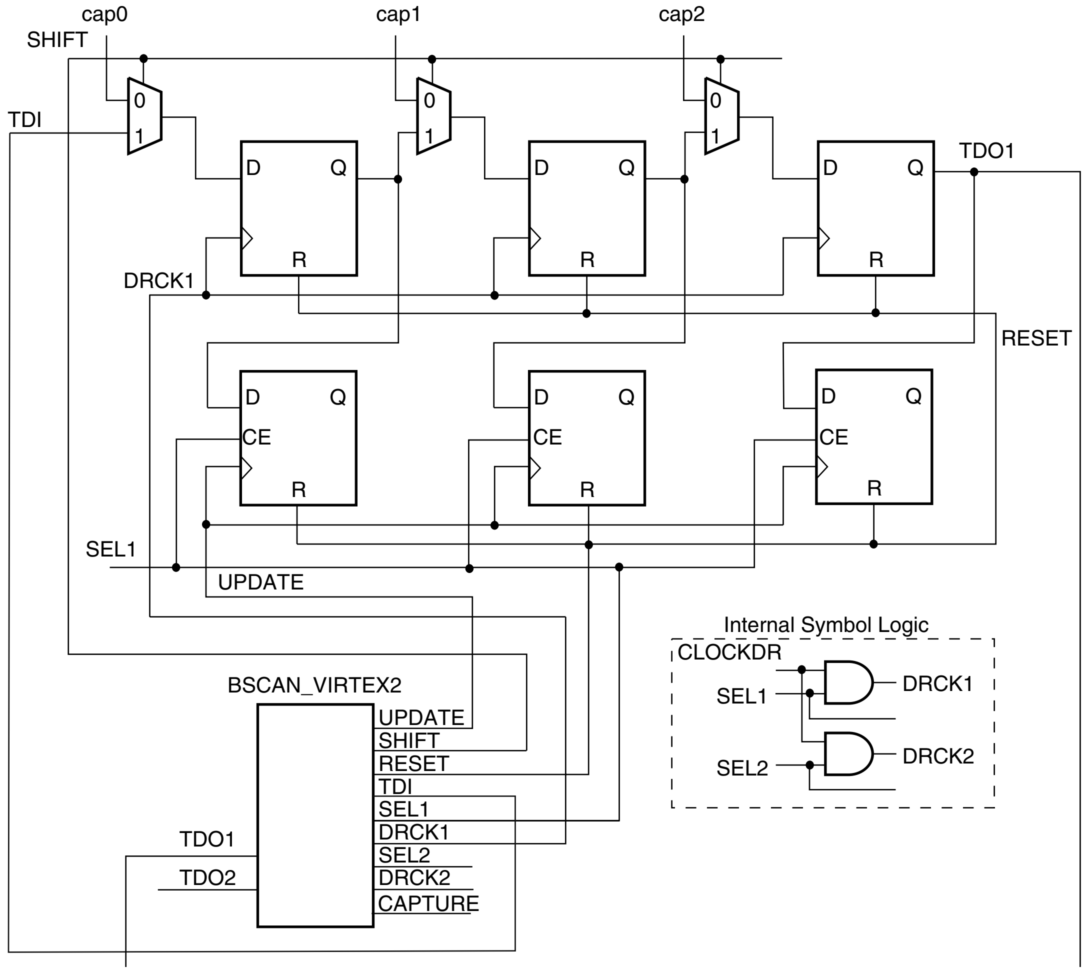


Figure 3-22: **BSCAN_VIRTEX2 (Example Usage)**

Using Boundary Scan in Virtex-II Devices

Characterization data for some of the most commonly requested timing parameters shown in Figure 3-23 is listed in Table 3-15.

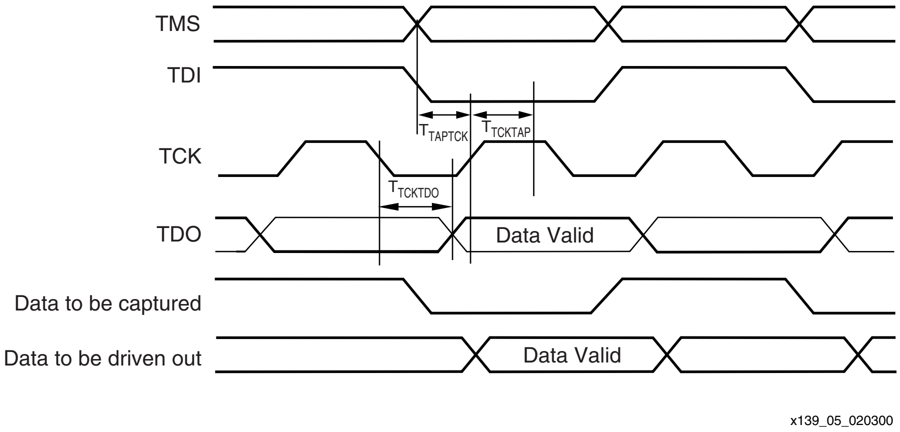


Figure 3-23: **Virtex-II Boundary Scan Port Timing Waveforms**

Table 3-15: **Boundary-Scan Port Timing Specifications**

Symbol	Parameter	Value	Units
T_{TAPTCK}	TMS and TDI setup time before TCK	4.0	ns, min
T_{TCKTAP}	TMS and TDI hold times after TCK	2.0	ns, min
T_{TCKTDO}	TCK falling edge to TDO output valid	11.0	ns, min
F_{TCK}	Maximum TCK clock frequency	33.0	MHz, max

For further information on the Startup sequence, bitstream, and internal configuration registers referenced here, refer to "Readback" on page 396.

Configuring Through Boundary-Scan

One of the most common boundary-scan vendor-specific instructions is the configure instruction. An individual Virtex-II device is configured via JTAG on power-up using TAP. If the Virtex-II device is configured on power-up, it is advisable to tie the mode pins to the boundary-scan configuration mode settings; 101 (M2 = 1, M1 = 0, M0 = 1).

Configuration flow for Virtex-II device configuration with JTAG is shown in Figure 3-24. The sections that follow describe how the Virtex-II device can be configured as a single device via boundary-scan or as part of a multiple-device scan chain.

A configured device can be reconfigured by toggling the TAP and entering a CFG_IN instruction after pulsing the PROG_B pin or issuing the shut-down sequence. (Refer to "Power Up" on page 347). For additional details on power-up or the start-up sequence in Virtex-II devices, see "Device Startup" on page 349.

For additional detailed information on using Virtex devices in an embedded solution, see Xilinx application note XAPP058.

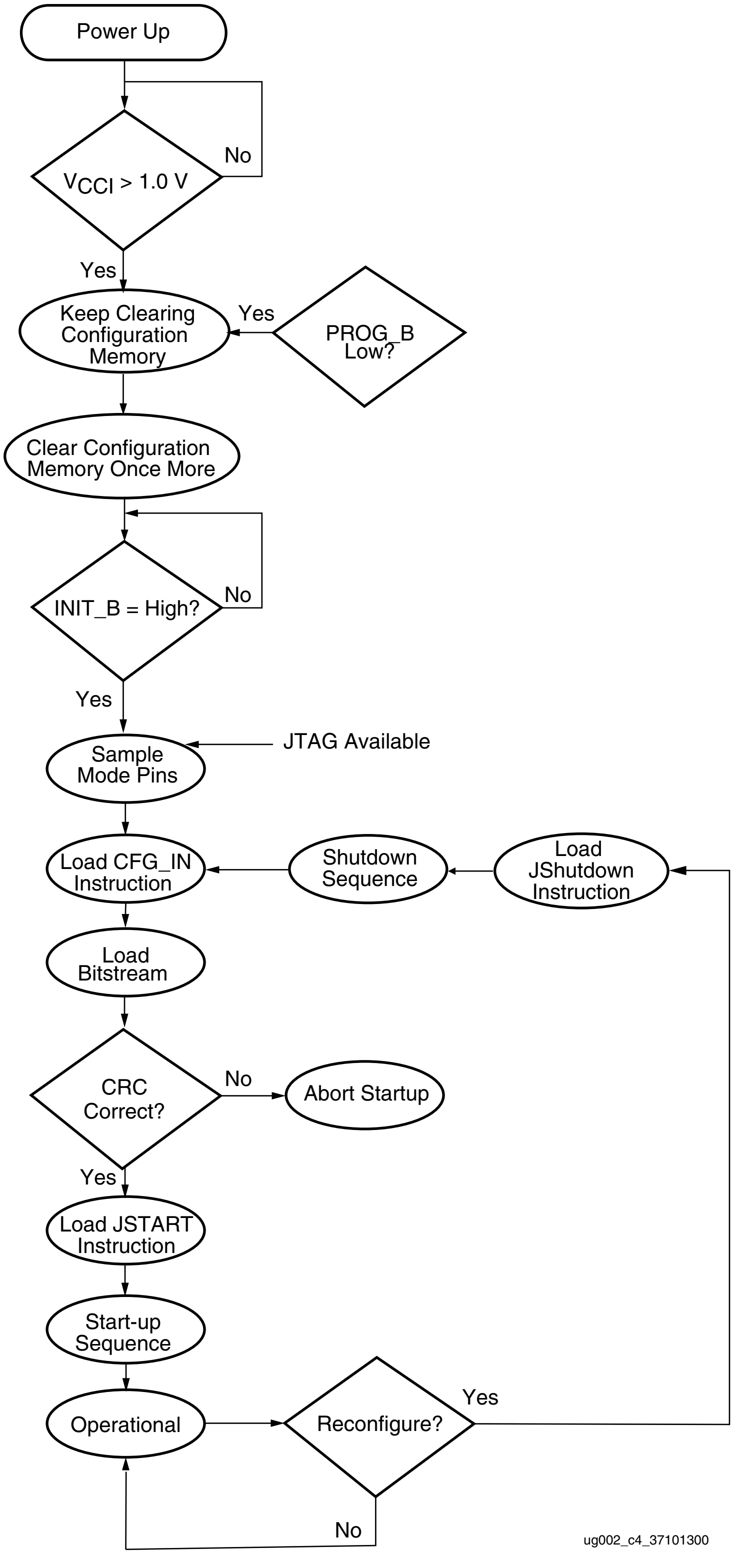


Figure 3-24: **Device Configuration Flow Diagram**

Single Device Configuration

Configure a Virtex-II part as a single device via boundary-scan operations as follows. Ensure that the bitstream is generated with the JTAG clock option.

```
bitgen -g startupclk:jtagclk designName.ncd
```

Also, when using JTAG Programmer software, verify that the most current version is being used.

Table 3-16 describes the TAP controller commands required to configure a Virtex-II device. Refer to Figure 3-20 for TAP controller states. These TAP controller commands are issued automatically if configuring the part with the JTAG Programmer software.

Table 3-16: **Single Device Configuration Sequence**

TAP Controller Step Description		Set & Hold		# of Clocks
		TDI	**TMS**	**TCK**
1	On power-up, place a logic "one" on the TMS and clock the TCK five times. This ensures starting in the TLR (Test-Logic-Reset) state.	X	1	5
2	Move into the RTI state.	X	0	1
3	Move into the SELECT-IR state.	X	1	2
4	Enter the SHIFT-IR state.	X	0	2
5	Start loading the CFG_IN instruction.	00101	0	5
6	Load the last bit of CFG_IN instruction when exiting SHIFT-IR. as defined in the IEEE standard.	0	1	1
7	Enter the SELECT-DR state.	X	1	2
8	Enter the SHIFT-DR state.	X	0	2
9	Shift in the Virtex-II bitstream. Bit$_n$ (MSB) is the first bit in the bitstream[1].	bit$_1$ ...bit$_n$	0	(bits in bitstream) – 1
10	Shift in the last bit of the bitstream. Bit$_0$ (LSB) shifts on the transition to EXIT1-DR.	bit$_0$	1	1
11	Enter UPDATE-DR state.	X	1	1
12	Enter the SELECT-IR state.	X	1	2
13	Move to the SHIFT-IR state.	X	0	2
14	Start loading the JSTART instruction. The JSTART instruction initializes the startup sequence.	01100	0	5
15	Load the last bit of the JSTART instruction.	0	1	1
16	Move to RTI and clock the STARTUP sequence by applying a minimum of 12 clock cycles to the TCK.	X	0	≥12
17	Move to the TLR state. The device is now functional.	X	1	3

Notes:

1. In the Configuration Register, data is shifted in from the right (TDI) to the left (TDO).

Multiple Device Configuration

It is possible to configure multiple Virtex-II devices in a chain. The devices in the JTAG chain are configured one at a time. The multiple device configuration steps are described generally to be applied to any size chain. Ensure the bitstream is generated with the JTAG clock option.

```
bitgen -g startupclk:jtagclk designName.ncd
```

Refer to the State Diagram in Figure 3-20 for the following TAP controller steps.

1. On power-up, place a logic "one" on the TMS and clock the TCK five times. This ensures starting in the TLR (Test-Logic-Reset) state.
2. Load the CFG_IN instruction into the target device (and BYPASS in all other devices).Go through RTI (RUN-TEST/IDLE).

   Repeat steps 2 through 4 for each successive device.
3. Load the JSTART command into all devices.
4. Go to RTI and clock TCK 12 times.

   All devices are active at this point.

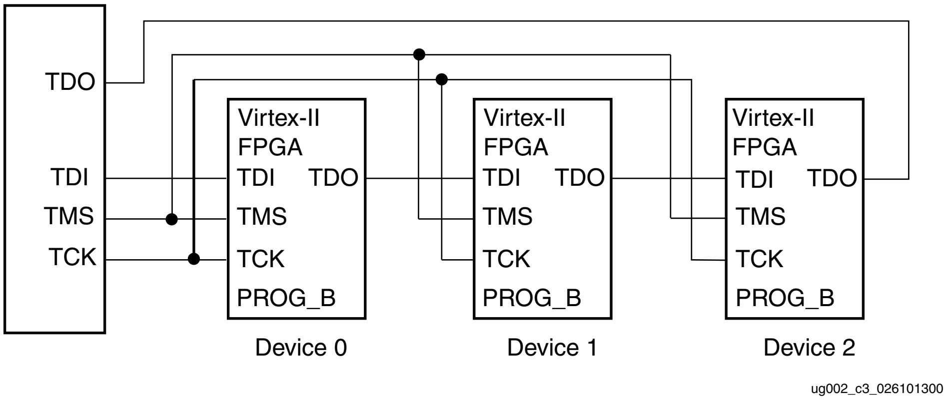


Figure 3-25: **Boundary Scan Chain of Devices**

Notes:

1. PROG_B pin should be deasserted during JTAG operation.

Reconfiguring Through Boundary Scan

The ability of Virtex-II devices to perform partial reconfiguration is the reason that the configuration memory is not cleared when reconfiguring the device. When reconfiguring a chain of devices, refer to step 3 in Table 3-16. There are two methods to reconfigure Virtex-II devices without possible internal contention. The first method is to pulse the PROG_B pin which resets the internal configuration memory. The alternate method is to perform a shutdown sequence, placing the device in a safe state. The following shutdown sequence includes using internal registers. (For details on internal registers, refer to "Readback" on page 396.)

1. Load the CFG_IN instruction.
2. In SHIFT-DR state, load the synchronization word followed by the Reset CRC Register (RCRC) command.

```
1111 1111 1111 1111 1111 1111 1111 1111    -> Dummy word
1010 1010 1001 1001 0101 0101 0110 0110    -> Synchronization word
0011 0000 0000 0000 1000 0000 0000 0001    -> Header: Write to CMD
                                              register
0000 0000 0000 0000 0000 0000 0000 0111    -> RCRC command
0000 0000 0000 0000 0000 0000 0000 0000    -> flush pipe
0000 0000 0000 0000 0000 0000 0000 0000    -> flush pipe
```

3. Load JSHUTDOWN.
4. Go to RTI and clock TCK at least 12 times to clock the shutdown sequence.
5. Proceed to SHIFT-IR state and load the CFG_IN instruction again.
6. Go to SHIFT-DR state and load the configuration bits. Make sure the configuration bits contain AGHIGH command, which asserts the global signal GHIGH_B. This prevents contention while writing configuration data.

```
0011 0000 0000 0000 1000 0000 0000 0001 -> Header: Write to CMD
0000 0000 0000 0000 0000 0000 0000 1000 -> AGHIGH command asserts
                                           GHIGH_B
```

7. When all configuration bits have been loaded, go to SHIFT-IR state and load the JSTART instruction.
8. Go to RTI and clock TCK at least 12 times to clock the startup sequence.
9. Go to TLR state to complete the reconfiguration process.

Debugging Configuration

To verify successful configuration, there are several options. Some of the most helpful verification steps include using TAP pins and the readback command. Using the Virtex-II TAP controller and status pins is discussed first.

When using TAP controller pins, TDO is driven only in the SHIFT-DR and SHIFT-IR state. If the output of the TDO can be changed via an external pull-up resistor, the TAP is not in SHIFT-IR or SHIFT-DR. If the TAP can be controlled precisely, use this to test the application.

In JTAG configuration, the status pin (DONE) functions the same as in the other configuration modes. The DONE pin can be monitored to determine if a bitstream has been completely loaded into the device. If DONE is Low, the entire bitstream has not been sent or the start-up sequence is not finished. If DONE is High, the entire bitstream has been received correctly. The INIT_B pin functions similar to a normal INIT_B but does not indicate a configuration error in boundary-scan configuration.

In addition to external pin monitoring, an internal test can be conducted. The second method includes the following steps to capture the internal device status register contents:

1. Move the TAP to TLR state.
2. Go to SHIFT-IR state and load in the CFG_IN instruction.
3. Go to SHIFT-DR state and shift in the following 64-bit pattern with the MSB (left-most bit), shifted in first.

   `1111 1111 1111 1111 1111 1111 1111 1111` -> Dummy word

   `1010 1010 1001 1001 0101 0101 0110 0110` -> Synchronization word

   `0010 1000 0000 0000 1110 0000 0000 0010` -> Read STATus Register 1)

   `0000 0000 0000 0000 0000 0000 0000 0000` -> flush pipe

   `0000 0000 0000 0000 0000 0000 0000 0000` -> flush pipe

   `0000 0000 0000 0000 0000 0000 0000 0000` -> flush pipe

   Notes:

   1. Since the JTAG readback shift register is 64-bit long, two 32-bit words are needed to fill the shift register.

4. After shifting in the pattern, load the CFG_OUT instruction in the SHIFT-IR state.
5. Move to SHIFT-DR state and clock TCK 32 times while reading TDO. The data seen on TDO is the content of the status register. The last bit out is a one if a CRC error occurred. If successful, it should read as follows.

   `0000 0000 0000 0000 0001 1MMM 1110 11101` 1,2)

   Notes:

   1. MMM is the mode pins value.
   2. Assuming that the device is in normal operation mode.

   Since the read status activity causes the crc_error status to be asserted, it is important to clear the crc_error status to ensure normal device operation. This can be done by writing the precalculated CRC value to the CRC register or writing an RCRC command.

6. Go to SHIFT-IR state and load the CFG_IN instruction again.
7. Move to SHIFT-DR state and shift in the following bit pattern:

   `0011 0000 0000 0000 1000 0000 0000 0001` -> Header: Write to CMD register

   `0000 0000 0000 0000 0000 0000 0000 0111` -> RCRC command

   `0000 0000 0000 0000 0000 0000 0000 0000` -> flush pipe

   `0000 0000 0000 0000 0000 0000 0000 0000` -> flush pipe

8. Put the TAP in TLR state when finished.

The device status register also gives the status of the DONE and INIT_B signals. For information on the status register, refer to Figure 3-30.

ISC Modal States

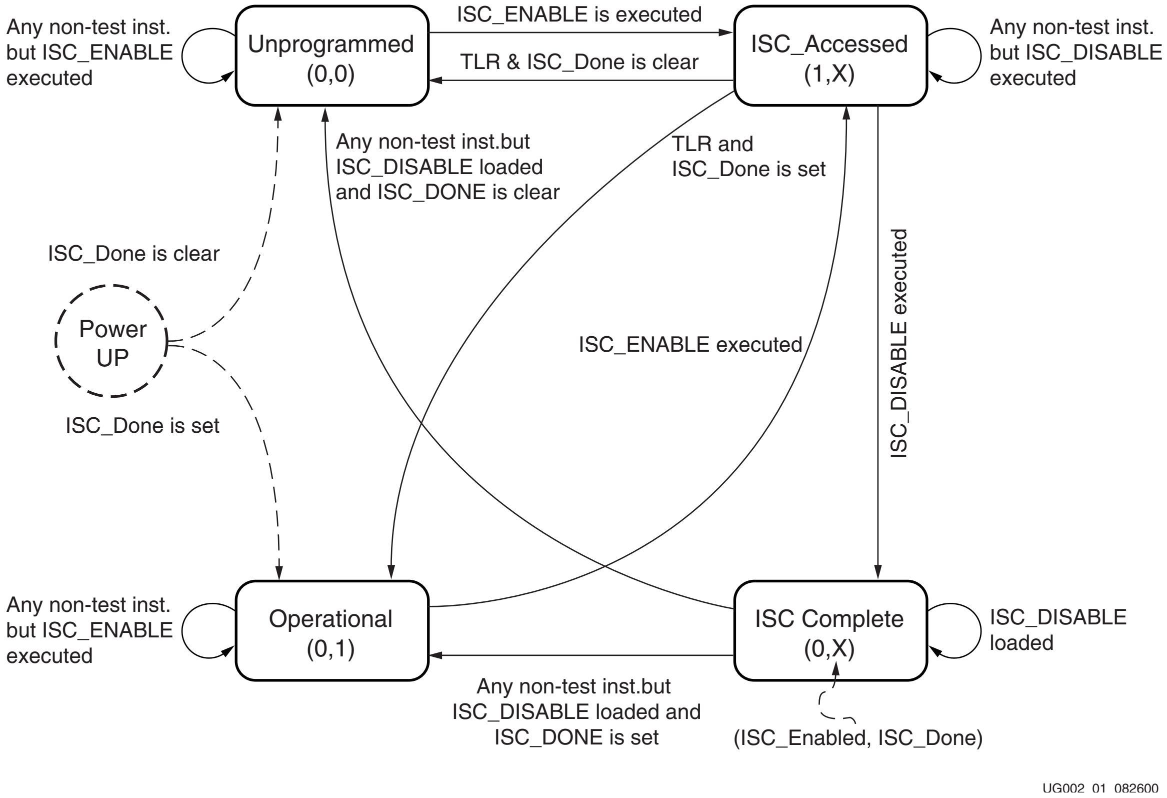


Figure 3-26: **ISC Modal States**

Once the device is powered up, it goes to an Unprogrammed state. The I/Os are all either 3-stated or pulled up. When ISC_ENABLE is successfully executed, the ISC_Enabled signal is asserted, and the device moves to ISC_Accessed state. When the device moves to ISC_Accessed state from Operational state, the shutdown sequence is executed. The I/Os are all either 3-stated or pulled up.

The StartUp sequence is executed when in the ISC_Accessed state. At the end of the StartUp Sequence, ISC_Enabled is cleared and the device moves to ISC_Complete. The minimum clock cycle requirement is the number of clock cycles required to complete the StartUp sequence. At the completion of the minimum required clock cycles, ISC_Enabled is deasserted.

Whether the StartUp sequence is successful or not is determined by CRC or configuration error status from the configuration processor. If the startup is completed, ISC_Done is asserted; otherwise, ISC_Done stays Low. The I/Os are either 3-stated or pulled up.

When ISC_Done is set in ISC_Complete state, the device moves to the Operational state. Otherwise, if ISC_Done is clear, the device moves to an Unprogrammed state. However, if the TAP controller goes to TLR state while the device is in ISC_Accessed state and if ISC_Done is set, then the device moves to the Operational state. However, the I/O is not active yet because the Startup sequence has not been performed. The Startup sequence has to be performed in the Operational state to bring the I/O active.

Clocking Startup and Shutdown Sequence (JTAG Version)

There are three clock sources for Startup and Shutdown sequence, CCLK, UserCLK, and JTAGCLK. Clock selection is set by bitgen. The Startup sequence is executed in ISC_Accessed state. When it is clocked by JTAGCLK, the Startup sequence receives the JTAGCLK in TAP Run/Test Idle state while ISC_DISABLE is the current JTAG instruction. The number of clock cycles in Run/Test Idle state for successful completion of ISC_DISABLE is determined by the number of clock cycles needed to complete the Startup sequence.

When UserCLK or CCLK is used to clock the Startup sequence, the user should know how many JTAGCLK cycles should be spent in Run/Test Idle to successfully complete the Startup sequence.

The Shutdown sequence is executed when the device transitions from an Operational to ISC_Accessed state. Shutdown is done while executing the ISC_ENABLE instruction. When the Shutdown sequence is clocked using JTAGCLK, the clock is supplied in the Run/Test Idle state of the ISC_ENABLE instruction. The number of clock cycles in Run/Test Idle is determined by the number of clock cycles needed to complete the Shutdown sequence.

When the Shutdown sequence is clocked by CCLK or UserCLK, the user is responsible for knowing how many JTAGCLK cycles in Run/Test Idle are needed to complete the Shutdown sequence.

Notes:

1. It has been decided that when configuring the device through JTAG, the startup and shutdown clock should come from TCK, regardless of the selection in bitgen.
2. In IEEE 1532 configuration mode, Startup and Shutdown clock source is always TCK.

Configuration Flows Using JTAG

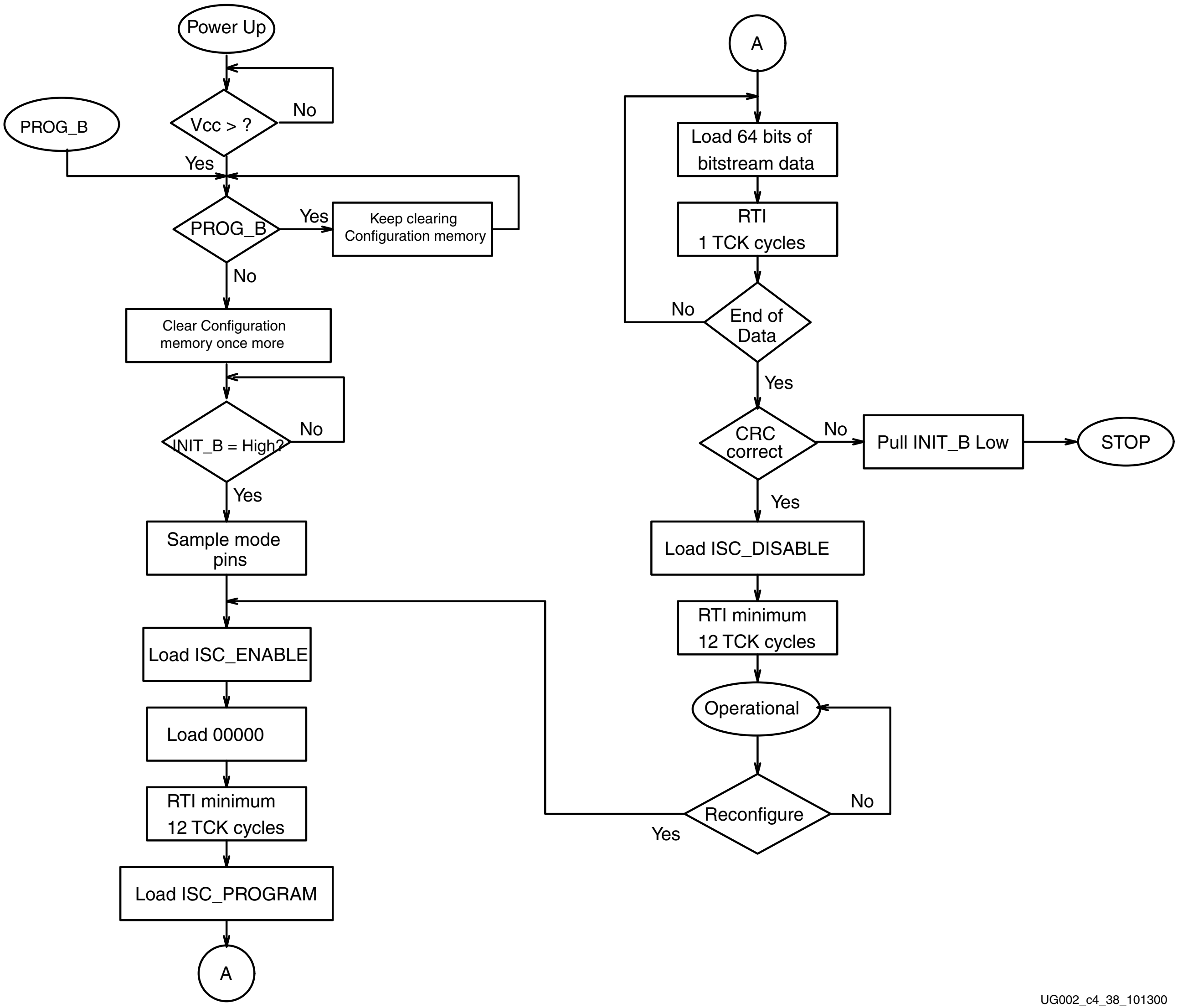


Figure 3-27: **IEEE 1532 Configuration Flow**

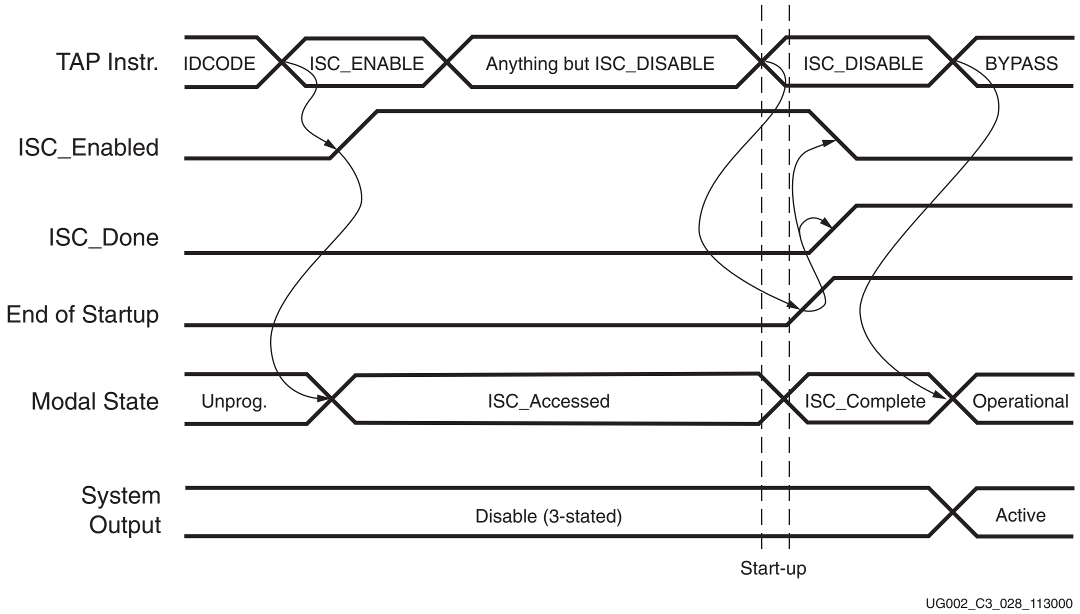


Figure 3-28: **Signal Diagram for Successful First Time ISC Configuration**

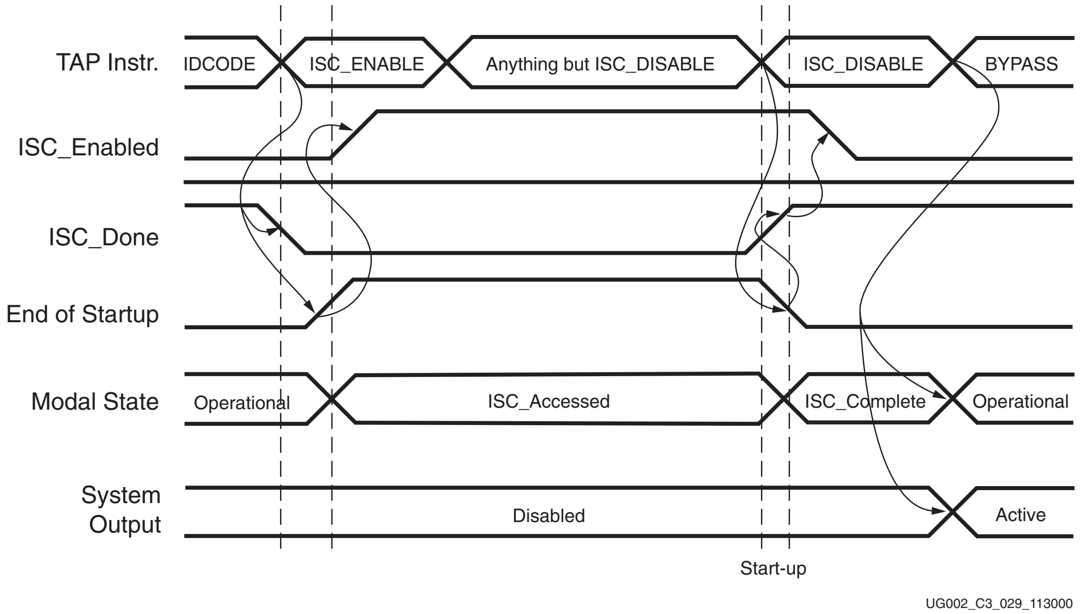


Figure 3-29: **Signal Diagram for Successful ISC Partial and Full Reconfiguration**

Software Support and Data Files

For Virtex-II devices, the Xilinx tool set includes the JTAG Programmer software to program and get Virtex-II IDCODEs. For test vectors EXTEST or INTEST, or to utilize other JTAG features present in the devices, see www.xilinx.com under Configuration Solutions for third party boundary-scan software tools.

IMPORTANT NOTE:

To perform any configuration operations through JTAG, the bitgen option should be set for the JTAG clock option.

```
bitgen -g startupclk:jtagclk designName.ncd
```

For Readback operations, this option can be used.

```
bitgen -w -l -m -g readback
```

Readback is not supported in the current version of JTAG Programmer Software.

JTAG Programmer

JTAG Programmer software is a standard feature of the Alliance Series™ and Foundation Series™ software packages. JTAG Programmer is a part of Web Pack, which can be downloaded from the following site:

http://support.xilinx.com/support/software.htm

Configuration With MultiLINX

The MultiLINX cable set is a peripheral hardware product. It is used primarily for downloading configuration and programming data from a host computer to Xilinx FPGAs and CPLDs in a target system.

The MultiLINX system supports a USB (Universal Serial Bus) interface with communication speeds up to 12 Mb/s, reducing download times by a factor of 120X relative to previous cables. The MultiLINX cable set includes all appropriate flying leads for multiple configuration mode support. In addition, MultiLINX cable sets support readback modes, such as verification and the Virtex-II SelectMAP interface.

MultiLINX cable internal hardware is upgraded via software, facilitating the addition of new cable features and simplifying support. Upgrades are completely seamless and invisible to users.

For additional information on the MultiLINX cable set and other Xilinx hardware products, refer to the *Hardware User Guide* on the web, or go to the following site: **http://www.xilinx.com/support/programr/cables.htm**

Configuration Details

This section provides a bit-level understanding of the configuration stream. For the purpose of debugging, designing embedded readback operations, or otherwise complex styles of configuring multiple FPGAs, the Virtex-II bitstream, internal configuration logic, and internal processing of configuration data are described here.

Data Frames

The internal configuration memory is partitioned into segments called “Frames.” The portions of the bitstream that actually get written to the configuration memory are “Data Frames.” The number and size of frames varies with device size as shown in Table 3-17. The total number of configuration bits for a particular device is calculated by multiplying the number of frames by the number of bits per frame, and then adding the total number of bits needed to perform the *Configuration Register Writes* shown in Table 3-17.

Table 3-17: **Virtex-II Configuration Data Frames and Programming Times**

Device	No. of Frames	Frame Length in Bits	Configuration Bits	Total No. of Bits (including header)	Approx. SelectMAP Download Time (50 MHz) ms	Approx. Serial Download Time (50 MHz) ms	Approx. JTAG Download Time (33 MHz) ms
XC2V40	404	832	360,096	339,040	0.84	6.72	10.19
XC2V80	404	1472	635,296	598,880	1.49	11.89	18.02
XC2V250	752	2112	1,697,184	1,593,696	3.97	31.76	48.13
XC2V500	928	2752	2,761,888	2,560,608	6.38	51.08	77.39
XC2V1000	1104	3392	4,082,592	3,752,800	9.36	74.90	113.48
XC2V1500	1280	4032	5,659,296	5,170,272	12.90	103.22	156.39
XC2V2000	1456	4672	7,492,000	6,813,024	17.01	136.05	206.13
XC2V3000	1804	5312	10,494,368	9,594,720	23.96	191.66	290.39
XC2V4000	2156	6592	15,659,936	14,226,784	35.53	284.25	430.68
XC2V6000	2508	7872	21,849,504	19,759,968	49.36	394.86	598.27
XC2V8000	2860	9152	29,063,072	26,194,272	65.44	523.49	793.17

Configuration Registers

The Virtex-II configuration logic was designed so that an external source can have complete control over all configuration functions by accessing and loading addressed internal configuration registers over a common configuration bus. The internal configuration registers that are used for configuration and readback are listed in Table 3-18. All configuration data, except the synchronization word and dummy words, is written to internal configuration registers.

Table 3-18: **Internal Configuration Registers**

Symbol	Register Name	Address
CRC	CRC Register	00000
FAR	Frame Address Register	00001
FDRI	Frame Data Input Register (Write Configuration Data)	00010
FDRO	Frame Data Output Register (Readback Configuration Data)	00011
CMD	Command Register	00100
CTL	Control Register	00101
MASK	Masking Register for CTL	00110
STAT	Status Register	00111
LOUT	Legacy Output Register (DOUT for daisy chain)	01000
COR	Configuration Option Register	01001
MFWR	Multiple Frame Write	01010
FLR	Frame Length Register	01011
IDCODE	Product ID Code Register	01110

Command Register (CMD)

Commands shown in Table 3-19 are executed by loading the binary code into the CMD register.

Table 3-19: **CMD Register Commands**

Symbol	Command	Binary Code
WCFG	Write Configuration Data	0001
MFWR	Multi-Frame Write	0010
DGHIGH	De-asserts GHIGH	0011
RCFG	Read Configuration Data	0100
START	Begin STARTUP Sequence	0101
RCAP	Reset CAPTURE (after Single-Shot Capture)	0110
RCRC	Reset CRC Register	0111
AGHIGH	Assert GHIGH	1000
SWITCH	Switch CCLK Frequency	1001
GRESTORE	Pulse GRESTORE Signal	1010
SHUTDOWN	Begin SHUTDOWN Sequence	1011
GCAPTURE	Pulse GCAPTURE Signal (one shot)	1100
DESYNCH	Forces realignment to 32 bits	1101

Frame Length Register (FLR)

The FLR is used to indicate the frame size to the internal configuration logic. This allows the internal configuration logic to be identical for all Virtex-II devices. The value loaded into this register is the number of actual configuration words that get loaded into the configuration memory frames.

Configuration Option Register (COR)

The COR is loaded with the user selected options from bitstream generation. See Appendix B, "BitGen and PROMGen Switches and Options.".

Table 3-20: **Configuration Option Register**

Name	Description	Bits
CRC_BYPASS	Does not check against updated CRC value.	29
SHUT_RST_DCI	DCI resets if SHUTDOWN and AGHIGH are performed.	27
SHUT_RST_DCM	DCM resets if SHUTDOWN and AGHIGH are performed.	26
DONE_PIPE	Add pipeline stage to DONEIN.	25
DRIVE_DONE	DONE pin is an active driver, not open drain.	24
SINGLE	Readback capture is one shot.	23
OSCFSEL	Select CCLK frequency in Master Serial Mode.	22:17
SSCLKSRC	Select STARTUP block clock source.	16:15
DONE_CYCLE	Startup cycle when DONE is asserted/de-asserted.	14:12
MATCH_CYCLE	Stall in this Startup cycle until DCI match signals are asserted.	11:9
LOCK_CYCLE	Stall in this Startup cycle until DCM signals are asserted.	8:6
GTS_CYCLE	Startup cycle when GTS_CFG_B is de-asserted.	5:3
GWE_CYCLE	Startup cycle when GWE is asserted.	2:0

Control Register (CTL)

The CTL controls internal functions such as *Security* and *Port Persistence*.

Table 3-21: **Control Register**

Name	Description	Bits
SBITS	Security level.	4:5
PERSIST	Configuration ports remain after configuration.	3
Reserved	For internal use.	2:1
GTS_USR_B	Active Low global 3-state I/Os. Turns off pullups if GTS_CFG_B is also asserted.	0

Mask Register (MASK)

The MASK is a safety mechanism that controls which bits of the CTL register can be reloaded. Prior to loading new data into the CTL register, each bit must be independently enabled by its corresponding bit in the MASK register. Any CTL bit not selected by the MASK register is ignored when reloading the CTL register.

Frame Address Register (FAR)

The FAR sets the starting frame address for the next configuration data input write cycle.

Frame Data Register Input (FDRI)

The FDRI is the input stage for configuration data frames to be stored in the configuration memory. Starting with the frame address specified in the FAR, the FDRI writes its contents to the configuration memory frames. The FDRI automatically increments the frame address after writing each frame for the number of frames specified in the FDRI write command. This is detailed in the next section.

CRC Register (CRC)

The CRC is loaded with a CRC value that is embedded in the bitstream and compared against an internally calculated CRC value. Resetting the CRC register and circuitry is controlled by the CMD register.

Frame Data Register Output (FDRO)

FDRO is an output stage for reading frame data from the configuration memory during readback. This works the same as the FDRI but with data flowing in the other direction.

Legacy Data Output Register (LOUT)

LOUT is pipeline data to be sent out the DOUT pin for serially daisy-chained configuration data output.

Status Register (STAT)

The STAT register contains bits that indicate the state of the device. Such bits include the status of error pins, global signals, the DCM, and DCI. This register is read-only and can be read using the JTAG or SelectMAP port for debugging purposes.

				RESERVED	RESERVED	ID_ERROR	DONE	INIT_B	MODE			GHIGH_B	GWE	GTS_CFG_B	IN_ERROR	DCI_MATCH	DCM_LOCK	RESERVED	CRC_ERROR
31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 3-30: **Status Register Fields**

Table 3-22: **Status Register**

Name	Description	Bit Location
ID_ERROR	IDCODE not validated while trying to write FDRI	13
DONE	DONEIN input form DONE pin	12
INIT_B	Value of CFG_RDY (INIT_B)	11
MODE	Value or MODE pins (M2, M1, M0)	10:8
GHIGH_B	Status of GHIGH	7
GWE	Status of GWE	6
GTS_CFG_B	Status of GTS_CFG_B	5
IN_ERROR	Legacy input error	4
DCI_MATCH	DCI matched	3
DCM_LOCK	DCM matched	2
Reserved	For internal use	1
CRC_ERROR	CRC error	0

Configuration Data Processing Flow

The complete (standard) reconfiguration of a Virtex-II device follows the internal flow shown in Figure 3-31. All associated configuration commands are listed in Table 3-23.

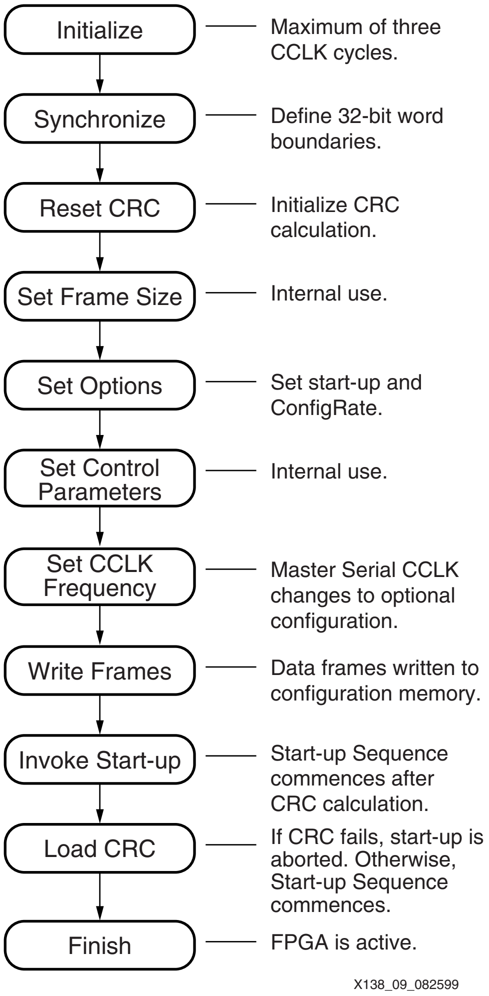


Figure 3-31: **Internal Configuration Processing Flow**

Table 3-23: **Configuration Register Writes**

Type	Number of 32-bit Words
Command Set 1	
Dummy words	1
Synchronization word	1
Write CMD (RCRC)	2
Write FLR	2
Write COR	2
Write ID	2
Write MASK	2
Write CMD (SWITCH)	2
Command Set 2	
Write FAR	2
Write CMD (WCFG)	2
Write FDRI	part size dependent
Write CMD (DGHIGH)	2
Command Set 3	
Write COR	2
Write CMD (START)	2
Write CTL	2
Write CRC	2
Write CMD (DESYNCH)	
Dummy words	4
TOTAL	40

The first command set prepares the internal configuration logic for the loading of the data frames. The internal configuration logic is first initialized with several CCLK cycles represented by dummy words, then it is synchronized to recognize the 32-bit word boundaries by the synchronization word. The CRC register and circuitry must then be reset by writing the RCRC command to the CMD register. The frame length size for the device being configured is then loaded into the FLR register. The configuration options are loaded into the COR. The CCLK frequency selected is specified in the COR; however, to switch to that frequency the SWITCH command must be loaded into the CMD register. The ID register is written to ensure that the correct bitstream is being used. Now the data frames can be loaded.

The second command set loads the configuration data frames. First, a WCFG (Write Configuration) command is loaded into the CMD register activating the circuitry that writes the data loaded into the FDRI into the configuration memory cells. To load a set of data frames, the starting address for the first frame is first loaded to the FAR, followed by a write command, and then by the data frames to the FDRI. The FDRI write command also specifies the amount of data that is to follow in terms of the number of 32-bit words that comprise the data frames being written. When all but the last frame has been loaded, an initial CRC checksum is loaded into the CRC register. The De-assert GHIGH (DGHIGH) is loaded into the CMD register.

The third command set initializes the Start-Up Sequence and finishes CRC checking. After all the data frames have been loaded, the START command is loaded into the CMD register, followed by any internal control data to CTL, the final CRC value into the CRC register, and the DESYNCH command to the CMD register. The four dummy words at the end are flushed through the system to provide the finishing CCLK cycles to activate the FPGA.

Standard Bitstream

Virtex-II devices have the ability to be only partially re-configured or read back. The standard bitstream, currently generated by BitGen, follows the format shown in Table 3-24, Table 3-25, and Table 3-26. *This format assumes D0 is considered the MSB.* It is divided into three tables to follow the three command sets described in the previous subsection.

Table 3-24 shows the first set of commands in the bitstream that prepare the configuration logic for rewriting the memory frames. All commands are described as 32-bit words, since configuration data is internally processed from a common 32-bit bus.

Table 3-24: **Bitstream Header and Configuration Options**

Data Type
Dummy word
Synchronization word
Packet Header: Write to CMD register
Packet Data: RCRC
Packet Header: Write to FLR register
Packet Data: Frame Length
Packet Header: Write to COR
Packet Data: Configuration options (user defined)
Packet Header: Write to ID register
Packet Data: IDCODE
Packet Header: Write to CMD register
Packet Data: SWITCH
Packet Header: Write to CMD register
Packet Data: WCFG

From Table 3-24, the first dummy word pads the front of the bitstream to provide the clock cycles necessary for initialization of the configuration logic. No actual processing takes place until the synchronization word is loaded. Since the Virtex-II configuration logic processes data as 32-bit words, but can be configured from a serial or 8-bit source, the synchronization word is used to define the 32-bit word boundaries. That is, the first bit after the synchronization word is the first bit of the next 32-bit word, and so on.

After synchronization, all data (register writes and frame data) are encapsulated in *packets.* There are two kinds of packets, Header and Data. A header packet has two types: Type 1 and Type 2. Type 1 Packet Headers are used for register writes. A combination of Type 1 and Type Packet Headers are used for frame data writes. A Type 1 Packet Header, shown in Figure 3-32, is always a single 32-bit word that describes the header type, whether it is a read/write function to a specific configuration register address (see Table 3-18) as the destination, and how many 32-bit words are in the following Packet Data portion. A Type 1 Packet Data portion can contain anywhere from 0 to 2,047 32-bit data words.

Packet Header	Type	Operation (Write/Read)	Register Address (Destination)	Byte Address	Word Count (32-bit Words)
Bits[31:0]	31:29	28:27	26:13	12:11	10:0
Type 1	001	10 / 01	XXXXXXXXXXXXXX	XX	XXXXXXXXXXX

X138_10_082599

Figure 3-32: **Type 1 Packet Header**

Packet Header	Type	Operation (Write/Read)	Word Count (32-bit Words)
Bits[31:0]	31:29	28:27	26:0
Type 2	010	10 / 01	XXXXXXXXXXXXXXXXXXXXXXXXXXX

X138_11_082599

Figure 3-33: **Type 2 Packet Header**

The first packet header in Table 3-24 is a Type 1 packet header that specifies writing one data word to the CMD register. The following packet data is a data word specifying a reset of the CRC register (compare the data field of Table 3-24 to the binary codes of Table 3-19).

The second packet header in Table 3-24 loads the frame size into the FLR.

The third packet header loads the configuration options into the COR register. The binary description of this register is not documented. Following this is a similar write of the SWITCH command to the CMD register which selects the CCLK frequency specified in the COR. Finally, the WCFG command is loaded into the CMD register so that the loading of frame data can commence.

The fourth packet header writes to the ID register. This ensures the correct bitstream for the correct Virtex-II family member.

Table 3-25 shows the packets that load all of the data frames, starting with a Type 1 packet header to load the starting frame address, which is always 0h.

Table 3-25: **Bitstream Data Frames and CRC Sequence**

Data Type
Packet Header: Write to FAR register
Packet Data: Starting frame address
Packet Header: Write to FDRI
Packet Header Type 2: Data words
Packet Data: Configuration data frames in 32-bit words. Total number of words specified in Type 2 Packet Header
Packet Data: CRC value
Packet Header: Write to CMD register
Packet Data: GRESTORE
Packet Header: Write to CMD register
Packet Data: DGHIGH
Packet Header: NO OP
Packet Data: one frame of NO OP

The loading of data frames requires a combination of Type 1 and Type 2 packet headers. Type 2 packet headers must always be preceded by a Type 1 packet header. The Type 2 packet data can be up to 67,108,863 data words in size.

The Type 2 packet header, shown in Figure 3-33, differs slightly from a Type 1 packet header in that there is no Register Address or Byte Address fields.

To write a set of data frames to the configuration memory, after the starting frame address has been loaded into the FAR, a Type 1 packet header issues a write command to the FDRI, followed by a Type 2 packet *header* specifying the number of data words to be loaded, and then followed by the actual frame data as Type 2 packet *data*. Writing data frames might require a Type 1/Type 2 packet header combination, or a Type 1 only. This depends on the amount of data being written.

Table 3-26 shows the packets needed to issue the start-up operations and load the final CRC check. The FPGA does not go active until after the final CRC is loaded. The number of clock cycles required to complete the start-up sequence depends on the BitGen options selected. Completion of the configuration process requires 8 to 16 clock cycles after the DESYNCH command. The DESYNCH command forces realignment to 32-bit boundaries and, therefore, a synchronization word is needed.

Table 3-26: **Bitstream Final CRC and Start-Up Sequence**

Data Type
Packet Header: Write to CMD register
Packet Data: START
Packet Header: Write to MASK
Packet Data: CTL mask
Packet Header: Write to CTL
Packet Data: Control commands
Packet Header: Write to CRC
Packet Data: CRC value
Packet Header: Write to CMD
Packet Data: DESYNCH command
Dummy word
Dummy word
Dummy word
Dummy word

Typically, DONE is released within the first seven CCLK cycles after the final CRC value is loaded, but the rest of the dummy data at the end of the stream should continue to be loaded. The FPGA needs the additional clock cycles to finish internal processing, but this is not a concern when a free-running oscillator is used for CCLK. In serial mode, this requires only 16 bits (two bytes), but in SelectMAP mode, this requires 16 bytes of dummy words at the end of the bitstream. Since the intended configuration mode to be used is unknown by Bitgen, four 32-bit dummy words (16 bytes) are always placed at the end of the bitstream.

Cyclic Redundancy Checking Algorithm

Virtex-II configuration uses a standard 16-bit CRC checksum algorithm to verify bitstream integrity during configuration. The 16-bit CRC polynomial is shown below.

$$\mathrm{CRC\text{-}16} = X^{16} + X^{15} + X^{2} + 1$$

The algorithm is implemented by shifting the data stream into a 16-bit shift register, shown in Figure 3-34. Register Bit(0) receives an XOR of the incoming data and the output of Bit(15). Bit(2) receives an XOR of the input to Bit(0) and the output of Bit(1). Bit(15) receives an XOR of the input to Bit(0) and the output of Bit(14).

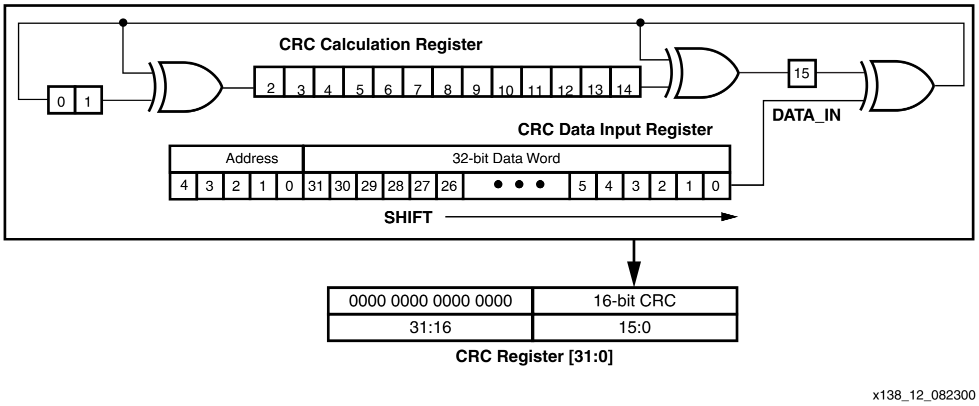


Figure 3-34: **Serial 16-bit CRC Circuitry**

A CRC Reset resets all the CRC registers to zero. As data is shifted into the CRC circuitry, a CRC calculation accumulates in the registers. When the CRC value is loaded into the CRC calculation register, the ending CRC checksum is loaded into the CRC Register. The value loaded into the CRC Register should be zero; otherwise, the configuration failed CRC check.

Not all of the configuration stream is loaded into the CRC circuitry. Only data that is written to one of the registers shown in Table 3-23 is included. For each 32-bit word that is written to one of the registers (Table 3-23), the address code for the register and the 32-bit data word is shifted LSB first into the CRC calculation circuitry, see Figure 3-34. When multiple 32-bit words are written to the same register, the same address is loaded after each word. All other data in the configuration stream is ignored and does not affect the CRC checksum.

This description is a model that can be used to generate an identical CRC value. The actual circuitry in the device is a slightly more complex Parallel CRC circuit that produces the same result.

Readback

Readback is the process of reading all the data in the internal configuration memory. This can be used to verify that the current configuration data is correct and to read the current state of all internal CLB and IOB registers as well as the current LUT RAM and block RAM values.

Readback is only available through the SelectMAP and Boundary Scan interfaces. This discussion covers the use of the SelectMAP interface for performing readback. For information on using the Boundary Scan interface for readback see “Readback When Using Boundary Scan” on page 397.

Readback Verification and Capture

Readback verification is used to verify the validity of the stored configuration data. This is most commonly used in space-based applications where exposure to radiation might alter the data stored in the configuration memory cells.

Readback capture is used to list the states of all the internal flip-flops. This can be used for hardware debugging and functional verification. When Capture is initiated, the internal register states are loaded into unused spaces in the configuration memory which can be extracted after a readback of the configuration memory.

While both *Verify* and *Capture* can be performed in one readback, each require slightly different preparation and post processing.

Preparing for Readback in Design Entry

If only a readback verification is to be performed, there are no additional steps at the time of design entry. However, if readback capture is to be used, the Virtex-II library primitive CAPTURE_VIRTEX2 must be instantiated in the user design as shown in Figure 3-35.

The CAPTURE_VIRTEX2 component is used in the FPGA design to control when the logic states of all the registers are captured into configuration memory. The CLK pin can be driven by any clock source that would synchronize Capture to the changing logic states of the registers. The CAP pin is an enable control. When CAP is asserted, the register states are captured in memory on the next CLK rising edge.

Capture can be performed in two ways: single-shot or continuous. In continuous capture, the CAP line is held High until the desired capture event occurs causing CAP to go Low. See Figure 3-35. Continuous capture does not require a readback operation to reset the CAPTURE block. In single-shot capture, the CAP line is pulsed once, and subsequent pulses are ignored until a readback operation has been performed. Captured data is read using the same process as a normal readback.

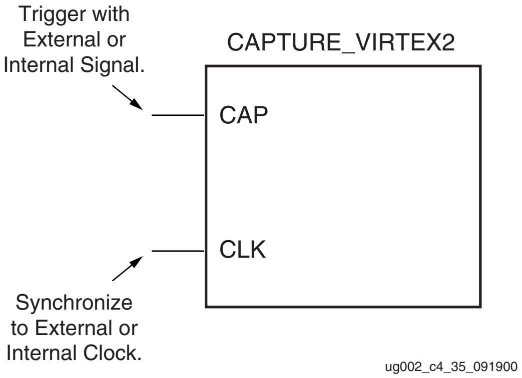


Figure 3-35: **Readback CAPTURE_VIRTEX2 Library Primitive**

Enabling Readback in the Software

Since readback is performed through the SelectMAP interface after configuration, the configuration ports must continue to be active by setting the persistence switch in BitGen. Additionally, a readback bit file, which contains the commands to execute a readback and a bitmap for data verification, can optionally be generated by setting the readback option in BitGen. An example of the BitGen command line is shown below.

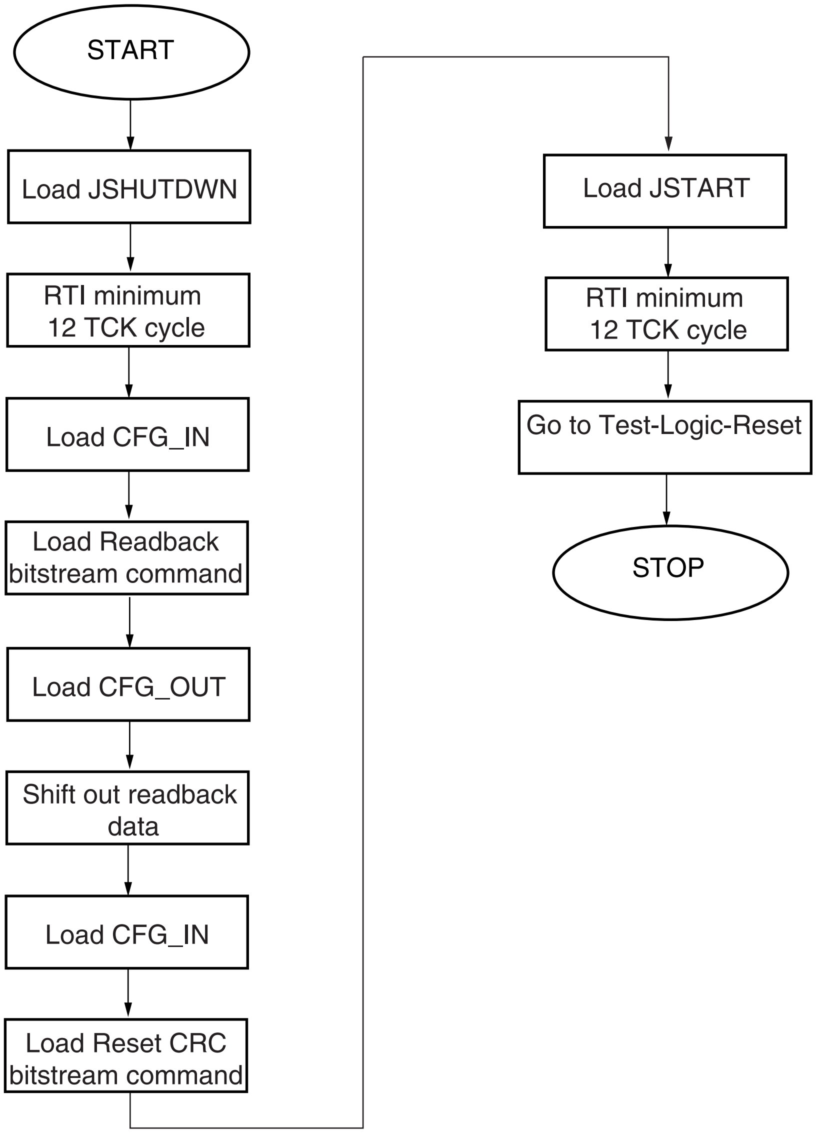
```
bitgen -w -l -m -g readback -g persist:yes...
```

The **-w** option overwrites existing output. The **-l** option generates a *Logic Allocation* file. The **-m** option generates a *Mask* file. The **-g readback** option generates a *readback bit* file, and the -**g persist:yes** option keeps the SelectMAP interface active after configuration. For more information on BitGen options, see Appendix B, "BitGen and PROMGen Switches and Options."

Readback When Using Boundary Scan

Regular Readback Flow

It is highly recommended to perform shutdown before reading back bitstream to ensure normal operation. The Shutdown Sequence can be executed by loading the JSHUTDOWN instruction and spending at least 12 TCK cycles in RTI TAP controller state. CRC_ERROR status and configuration error (CFGERR) must be cleared after readback by issuing Reset CRC bitstream command or writing the correct CRC value to CRC register.

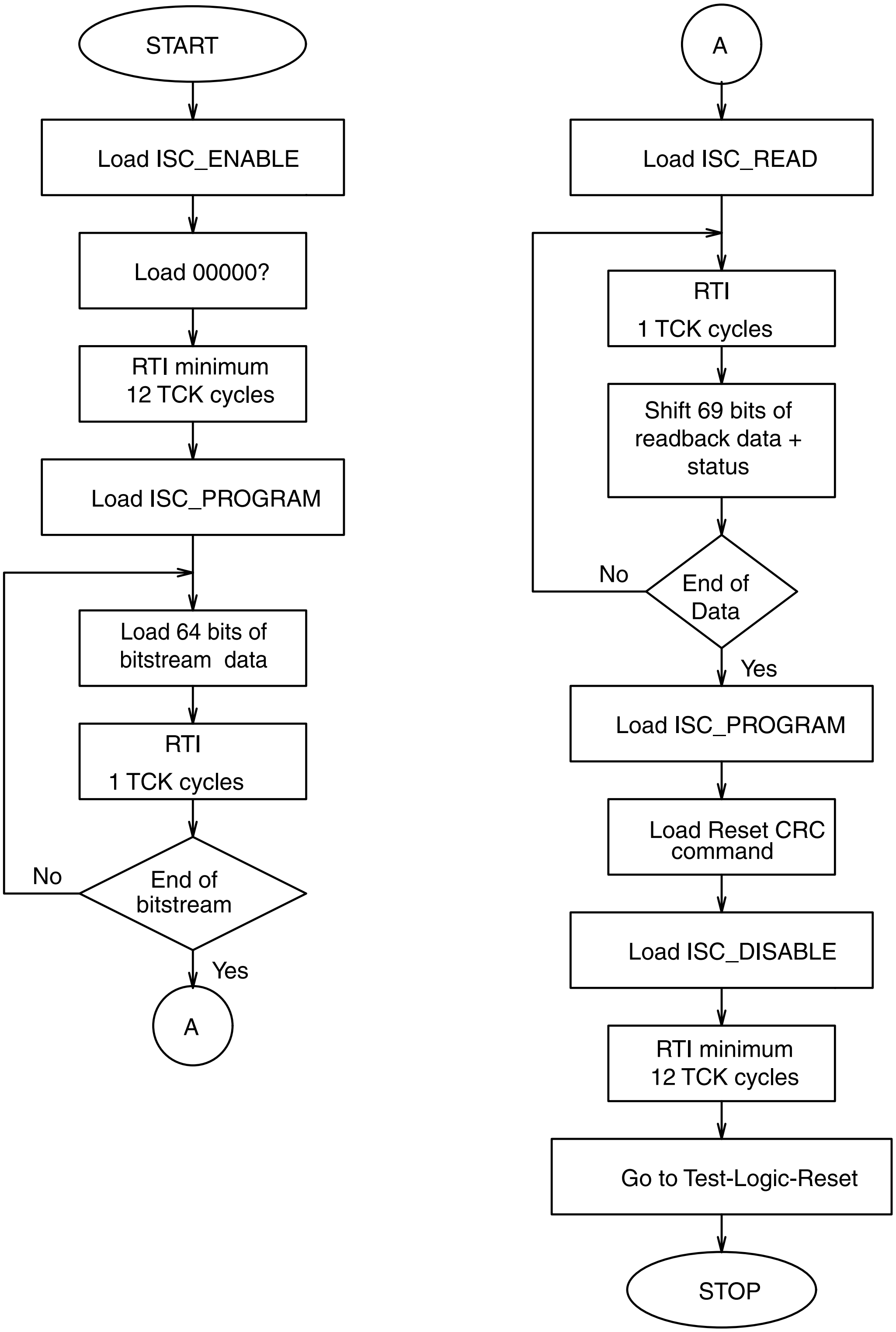


UG002_c4_36_091900

Figure 3-36: **Regular Readback Flow**

IEEE 1532 Readback Flow

In IEEE 1532 readback mode, full chip shutdown is performed when ISC_ENABLE is executed. At the end of readback, CRC Error status must be cleared by issuing Reset CRC command or writing the correct CRC value to CRC register. ISC_DISABLE cannot be executed correctly unless the CRC error status is cleared.



UG002_c4_39_092100

Figure 3-37: **IEEE 1532 Readback Flow**

Using ChipScope ILA

The ChipScope ILA functional verification tool is currently sold separately through the Xilinx web site. This program uses a combination of PC software and instantiated soft cores to capture states of internal signals. This data is read out of the JTAG USER1 scan chain using the MultiLINX cable or a parallel cable. ChipScope ILA supports only the Virtex architecture and allows for the functional verification and debugging of an FPGA configured design.

ChipScope ILA supports the high speed USB interface to the MultiLINX cable set on Windows 98/2000 platforms and the RS232 connection on Windows 95/98/2000/NT platforms. UNIX support is not available. More details are available under ChipScope ILA on the web at: **www.xilinx.com**

Chapter 4

PCB Design Considerations

Summary

This chapter covers the following topics:

- Pinout Information
- Pinout Diagrams
- Package Specifications
- Flip-Chip Packages
- Thermal Data
- Printed Circuit Board Considerations
- Board Routability Guidelines
- Power Consumption
- IBIS Models
- BSDL and Boundary Scan Models

Pinout Information

Introduction

This section describes the pinouts for Virtex-II devices in the following packages:

- CS144: wire-bond chip-scale ball grid array (BGA) of 0.80 mm pitch
- FG256, FG456, and FG676: wire-bond fine-pitch BGA of 1.00 mm pitch
- FF896, FF1152, FF1517: flip-chip fine-pitch BGA of 1.00 mm pitch
- BG575 and BG728: wire-bond BGA of 1.27 mm pitch
- BF957: flip-chip BGA of 1.27 mm pitch

All of the devices supported in a particular package are pinout compatible and are listed in the same table (one table per package). In addition, the FG456 and FG676 packages are compatible, as are the FF896 and FF1152 packages. Pins that are not available for the smallest devices are listed in right-hand columns.

Each device is split into eight I/O banks to allow for flexibility in the choice of I/O standards (see the **Virtex-II Data Sheet**). Global pins, including JTAG, configuration, and power/ground pins, are listed at the end of each table. Table 4-1 provides definitions for all pin types.

The FG256 pinouts (Table 4-2) is included as an example. All Virtex-II pinout tables are available on the distribution CD-ROM, or on the **www.xilinx.com** website.

Pin Definitions

Table 4-1 provides a description of each pin type listed in Virtex-II pinout tables.

Table 4-1: **Virtex-II Pin Definitions**

Pin Name	Direction	Description
User I/O Pins		
IO_LXXY_#	Input/Output	All user I/O pins are capable of differential signalling and can implement LVDS, ULVDS, BLVDS, or LDT pairs. Each user I/O is labeled "**IO_LXXY_#**", where: **IO** indicates a user I/O pin. **LXXY** indicates a differential pair, with **XX** a unique pair in the bank and **Y = P/N** for the positive and negative sides of the differential pair. # indicates the bank number (0 through 7)
Dual-Function Pins		
IO_LXXY_#/ZZZ		The dual-function pins are labelled "**IO_LXXY_#/ZZZ**", where **ZZZ** can be one of the following pins: Per Bank - **VRP, VRN,** or **VREF** Globally - **GCLKX(S/P), BUSY/DOUT, INIT_B, DIN/D0 – D7, RDWR_B,** or **CS_B**
With /ZZZ:		
DIN / D0, D1, D2, D3, D4, D5, D6, D7	Input/Output	In SelectMAP mode, D0 through D7 are configuration data pins. These pins become user I/Os after configuration, unless the SelectMAP port is retained. In bit-serial modes, DIN (D0) is the single-data input. This pin becomes a user I/O after configuration.
CS_B	Input	In SelectMAP mode, this is the active-low Chip Select signal. The pin becomes a user I/O after configuration, unless the SelectMAP port is retained.

Table 4-1: **Virtex-II Pin Definitions** ***(Continued)***

Pin Name	Direction	Description
RDWR_B	Input	In SelectMAP mode, this is the active-low Write Enable signal. The pin becomes a user I/O after configuration, unless the SelectMAP port is retained.
BUSY/DOUT	Output	In SelectMAP mode, BUSY controls the rate at which configuration data is loaded. The pin becomes a user I/O after configuration, unless the SelectMAP port is retained. In bit-serial modes, DOUT provides preamble and configuration data to downstream devices in a daisy-chain. The pin becomes a user I/O after configuration.
INIT_B	Bidirectional (open-drain)	When Low, this pin indicates that the configuration memory is being cleared. When held Low, the start of configuration is delayed. During configuration, a Low on this output indicates that a configuration data error has occurred. The pin becomes a user I/O after configuration.
GCLKx (S/P)	Input	These are clock input pins that connect to Global Clock Buffers. These pins become regular user I/Os when not needed for clocks.
VRP	Input	This pin is for the DCI voltage reference resistor of P transistor (per bank).
VRN	Input	This pin is for the DCI voltage reference resistor of N transistor (per bank).
ALT_VRP	Input	This is the alternative pin for the DCI voltage reference resistor of P transistor.
ALT_VRN	Input	This is the alternative pin for the DCI voltage reference resistor of N transistor.
V_{REF}	Input	These are input threshold voltage pins. They become user I/Os when an external threshold voltage is not needed (per bank).
Dedicated Pins[1]		
CCLK	Input/Output	Configuration clock. Output in Master mode or Input in Slave mode.
PROG_B	Input	Active Low asynchronous reset to configuration logic. This pin has a permanent weak pull-up resistor.
DONE	Input/Output	DONE is a bidirectional signal with an optional internal pull-up resistor. As an output, this pin indicates completion of the configuration process. As an input, a Low level on DONE can be configured to delay the start-up sequence.
M2, M1, M0	Input	Configuration mode selection.
HSWAP_EN	Input	Enable I/O pullups during configuration.
TCK	Input	Boundary Scan Clock.
TDI	Input	Boundary Scan Data Input.
TDO	Output	Boundary Scan Data Output.
TMS	Input	Boundary Scan Mode Select.
PWRDWN_B	Input	Power down pin.
Other Pins		
DXN, DXP	N/A	Temperature-sensing diode pins (Anode: DXP, Cathode: DXN).
V_{BATT}	Input	Decryptor key memory backup supply. (Do not connect if battery is not used.)
RSVD	N/A	Reserved pin - do not connect.
V_{CCO}	Input	Power-supply pins for the output drivers (per bank).
V_{CCAUX}	Input	Power-supply pins for auxiliary circuits.
V_{CCINT}	Input	Power-supply pins for the internal core logic.
GND	Input	Ground.

Notes:

1. All dedicated pins (JTAG and configuration) are powered by V_{CCAUX} (independent of the bank V_{CCO} voltage).

FG256 Fine-Pitch BGA Package

As shown in Table 4-2, XC2V40, XC2V80, XC2V250, XC2V500, and XC2V1000 Virtex-II devices are available in the FG256 fine-pitch BGA package. Pins in the XC2V250, XC2V500, and XC2V1000 devices are the same. The No Connect column shows pin differences for the XC2V40 and XC2V80 devices.

The FG256 pinout information (Table 4-2) is included as an example. All Virtex-II pinout tables are available on the distribution CD-ROM, or on the web (at http://www.xilinx.com).

Table 4-2: **FG256 BGA — XC2V40, XC2V80, XC2V250, XC2V500, and XC2V1000**

Bank	Pin Description	Pin Number	No Connect in XC2V40	No Connect in XC2V80
0	IO_L01N_0	C4		
0	IO_L01P_0	B4		
0	IO_L02N_0	D5		
0	IO_L02P_0	C5		
0	IO_L03N_0/VRP_0	B5		
0	IO_L03P_0/VRN_0	A5		
0	IO_L04N_0/VREF_0	D6	NC	NC
0	IO_L04P_0	C6	NC	NC
0	IO_L05N_0	B6	NC	NC
0	IO_L05P_0	A6	NC	NC
0	IO_L92N_0	E6	NC	NC
0	IO_L92P_0	E7	NC	NC
0	IO_L93N_0	D7	NC	NC
0	IO_L93P_0	C7	NC	NC
0	IO_L94N_0/VREF_0	B7		
0	IO_L94P_0	A7		
0	IO_L95N_0/GCLK7P	D8		
0	IO_L95P_0/GCLK6S	C8		
0	IO_L96N_0/GCLK5P	B8		
0	IO_L96P_0/GCLK4S	A8		
1	IO_L96N_1/GCLK3P	A9		
1	IO_L96P_1/GCLK2S	B9		
1	IO_L95N_1/GCLK1P	C9		
1	IO_L95P_1/GCLK0S	D9		
1	IO_L94N_1	A10		
1	IO_L94P_1/VREF_1	B10		
1	IO_L93N_1	C10	NC	NC
1	IO_L93P_1	D10	NC	NC
1	IO_L92N_1	E10	NC	NC
1	IO_L92P_1	E11	NC	NC
1	IO_L05N_1	A11	NC	NC
1	IO_L05P_1	B11	NC	NC

Table 4-2: **FG256 BGA — XC2V40, XC2V80, XC2V250, XC2V500, and XC2V1000**

Bank	Pin Description	Pin Number	No Connect in XC2V40	No Connect in XC2V80
1	IO_L04N_1	C11	NC	NC
1	IO_L04P_1/VREF_1	D11	NC	NC
1	IO_L03N_1/VRP_1	A12		
1	IO_L03P_1/VRN_1	B12		
1	IO_L02N_1	C12		
1	IO_L02P_1	D12		
1	IO_L01N_1	B13		
1	IO_L01P_1	C13		
2	IO_L01N_2	C16		
2	IO_L01P_2	D16		
2	IO_L02N_2/VRP_2	D14		
2	IO_L02P_2/VRN_2	D15		
2	IO_L03N_2	E13		
2	IO_L03P_2/VREF_2	E14		
2	IO_L04N_2	E15	NC	
2	IO_L04P_2	E16	NC	
2	IO_L06N_2	F13	NC	
2	IO_L06P_2	F14	NC	
2	IO_L43N_2	F15	NC	NC
2	IO_L43P_2	F16	NC	NC
2	IO_L45N_2	F12	NC	NC
2	IO_L45P_2/VREF_2	G12	NC	NC
2	IO_L91N_2	G13	NC	
2	IO_L91P_2	G14	NC	
2	IO_L93N_2	G15	NC	
2	IO_L93P_2/VREF_2	G16	NC	
2	IO_L94N_2	H13		
2	IO_L94P_2	H14		
2	IO_L96N_2	H15		
2	IO_L96P_2	H16		
3	IO_L96N_3	J16		
3	IO_L96P_3	J15		
3	IO_L94N_3	J14		
3	IO_L94P_3	J13		
3	IO_L93N_3/VREF_3	K16	NC	
3	IO_L93P_3	K15	NC	
3	IO_L91N_3	K14	NC	

Table 4-2: **FG256 BGA — XC2V40, XC2V80, XC2V250, XC2V500, and XC2V1000**

Bank	Pin Description	Pin Number	No Connect in XC2V40	No Connect in XC2V80
3	IO_L91P_3	K13	NC	
3	IO_L45N_3/VREF_3	K12	NC	NC
3	IO_L45P_3	L12	NC	NC
3	IO_L43N_3	L16	NC	NC
3	IO_L43P_3	L15	NC	NC
3	IO_L06N_3	L14	NC	
3	IO_L06P_3	L13	NC	
3	IO_L04N_3	M16	NC	
3	IO_L04P_3	M15	NC	
3	IO_L03N_3/VREF_3	M14		
3	IO_L03P_3	M13		
3	IO_L02N_3/VRP_3	N15		
3	IO_L02P_3/VRN_3	N14		
3	IO_L01N_3	N16		
3	IO_L01P_3	P16		
4	IO_L01N_4/DOUT	T14		
4	IO_L01P_4/INIT_B	T13		
4	IO_L02N_4/D0	P13		
4	IO_L02P_4/D1	R13		
4	IO_L03N_4/D2/ALT_VRP_4	N12		
4	IO_L03P_4/D3/ALT_VRN_4	P12		
4	IO_L04N_4/VREF_4	R12	NC	NC
4	IO_L04P_4	T12	NC	NC
4	IO_L05N_4/VRP_4	N11	NC	NC
4	IO_L05P_4/VRN_4	P11	NC	NC
4	IO_L91N_4/VREF_4	R11	NC	NC
4	IO_L91P_4	T11	NC	NC
4	IO_L92N_4	M11	NC	NC
4	IO_L92P_4	M10	NC	NC
4	IO_L93N_4	N10	NC	NC
4	IO_L93P_4	P10	NC	NC
4	IO_L94N_4/VREF_4	R10		
4	IO_L94P_4	T10		
4	IO_L95N_4/GCLK3S	N9		
4	IO_L95P_4/GCLK2P	P9		
4	IO_L96N_4/GCLK1S	R9		
4	IO_L96P_4/GCLK0P	T9		

Table 4-2: **FG256 BGA — XC2V40, XC2V80, XC2V250, XC2V500, and XC2V1000**

Bank	Pin Description	Pin Number	No Connect in XC2V40	No Connect in XC2V80
5	IO_L96N_5/GCLK7S	T8		
5	IO_L96P_5/GCLK6P	R8		
5	IO_L95N_5/GCLK5S	P8		
5	IO_L95P_5/GCLK4P	N8		
5	IO_L94N_5	T7		
5	IO_L94P_5/VREF_5	R7		
5	IO_L93N_5	P7	NC	NC
5	IO_L93P_5	N7	NC	NC
5	IO_L92N_5	M7	NC	NC
5	IO_L92P_5	M6	NC	NC
5	IO_L91N_5	T6	NC	NC
5	IO_L91P_5/VREF_5	R6	NC	NC
5	IO_L05N_5/VRP_5	P6	NC	NC
5	IO_L05P_5/VRN_5	N6	NC	NC
5	IO_L04N_5	T5	NC	NC
5	IO_L04P_5/VREF_5	R5	NC	NC
5	IO_L03N_5/D4/ALT_VRP_5	P5		
5	IO_L03P_5/D5/ALT_VRN_5	N5		
5	IO_L02N_5/D6	R4		
5	IO_L02P_5/D7	P4		
5	IO_L01N_5/RDWR_B	T4		
5	IO_L01P_5/CS_B	T3		
6	IO_L01P_6	P1		
6	IO_L01N_6	N1		
6	IO_L02P_6/VRN_6	N3		
6	IO_L02N_6/VRP_6	N2		
6	IO_L03P_6	M4		
6	IO_L03N_6/VREF_6	M3		
6	IO_L04P_6	M2	NC	
6	IO_L04N_6	M1	NC	
6	IO_L06P_6	L4	NC	
6	IO_L06N_6	L3	NC	
6	IO_L43P_6	L2	NC	NC
6	IO_L43N_6	L1	NC	NC
6	IO_L45P_6	L5	NC	NC
6	IO_L45N_6/VREF_6	K5	NC	NC
6	IO_L91P_6	K4	NC	
6	IO_L91N_6	K3	NC	

Table 4-2: **FG256 BGA — XC2V40, XC2V80, XC2V250, XC2V500, and XC2V1000**

Bank	Pin Description	Pin Number	No Connect in XC2V40	No Connect in XC2V80
6	IO_L93P_6	K2	NC	
6	IO_L93N_6/VREF_6	K1	NC	
6	IO_L94P_6	J4		
6	IO_L94N_6	J3		
6	IO_L96P_6	J2		
6	IO_L96N_6	J1		
7	IO_L96P_7	H1		
7	IO_L96N_7	H2		
7	IO_L94P_7	H3		
7	IO_L94N_7	H4		
7	IO_L93P_7/VREF_7	G1	NC	
7	IO_L93N_7	G2	NC	
7	IO_L91P_7	G3	NC	
7	IO_L91N_7	G4	NC	
7	IO_L45P_7/VREF_7	G5	NC	NC
7	IO_L45N_7	F5	NC	NC
7	IO_L43P_7	F1	NC	NC
7	IO_L43N_7	F2	NC	NC
7	IO_L06P_7	F3	NC	
7	IO_L06N_7	F4	NC	
7	IO_L04P_7	E1	NC	
7	IO_L04N_7	E2	NC	
7	IO_L03P_7/VREF_7	E3		
7	IO_L03N_7	E4		
7	IO_L02P_7/VRN_7	D2		
7	IO_L02N_7/VRP_7	D3		
7	IO_L01P_7	D1		
7	IO_L01N_7	C1		
0	VCCO_0	F8		
0	VCCO_0	F7		
0	VCCO_0	E8		
1	VCCO_1	F10		
1	VCCO_1	F9		
1	VCCO_1	E9		
2	VCCO_2	H12		
2	VCCO_2	H11		
2	VCCO_2	G11		

Table 4-2: **FG256 BGA — XC2V40, XC2V80, XC2V250, XC2V500, and XC2V1000**

Bank	Pin Description	Pin Number	No Connect in XC2V40	No Connect in XC2V80
3	VCCO_3	K11		
3	VCCO_3	J12		
3	VCCO_3	J11		
4	VCCO_4	M9		
4	VCCO_4	L10		
4	VCCO_4	L9		
5	VCCO_5	M8		
5	VCCO_5	L8		
5	VCCO_5	L7		
6	VCCO_6	K6		
6	VCCO_6	J6		
6	VCCO_6	J5		
7	VCCO_7	H6		
7	VCCO_7	H5		
7	VCCO_7	G6		
NA	CCLK	P15		
NA	PROG_B	A2		
NA	DONE	R14		
NA	M0	T2		
NA	M1	P2		
NA	M2	R3		
NA	HSWAP_EN	B3		
NA	TCK	A15		
NA	TDI	C2		
NA	TDO	C15		
NA	TMS	B14		
NA	PWRDWN_B	T15		
NA	RSVD	A4		
NA	RSVD	A3		
NA	VBATT	A14		
NA	RSVD	A13		
NA	VCCAUX	R16		
NA	VCCAUX	R1		
NA	VCCAUX	B16		
NA	VCCAUX	B1		
NA	VCCINT	N13		
NA	VCCINT	N4		

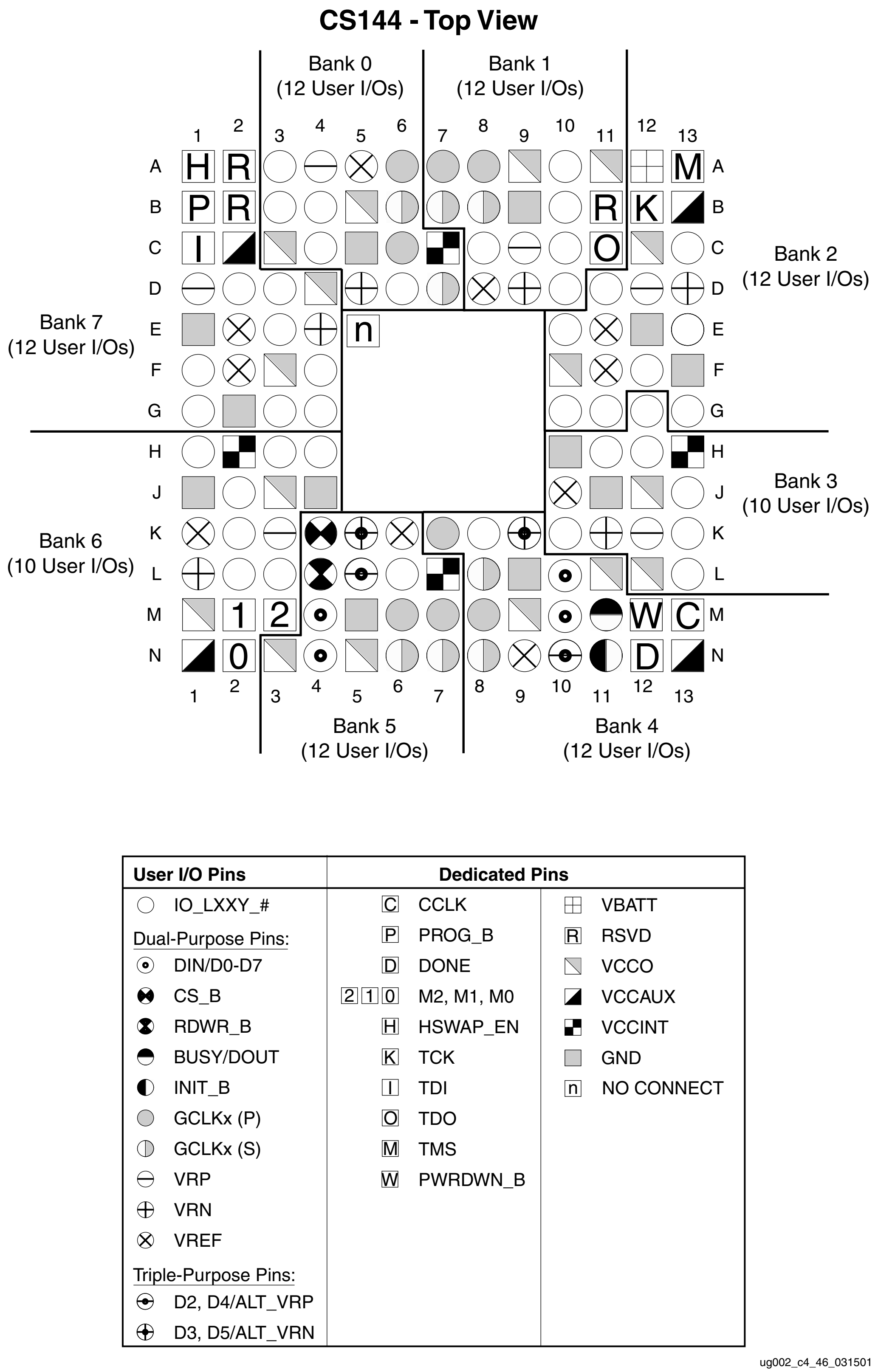
Table 4-2: **FG256 BGA — XC2V40, XC2V80, XC2V250, XC2V500, and XC2V1000**

Bank	Pin Description	Pin Number	No Connect in XC2V40	No Connect in XC2V80
NA	VCCINT	M12		
NA	VCCINT	M5		
NA	VCCINT	E12		
NA	VCCINT	E5		
NA	VCCINT	D13		
NA	VCCINT	D4		
NA	GND	T16		
NA	GND	T1		
NA	GND	R15		
NA	GND	R2		
NA	GND	P14		
NA	GND	P3		
NA	GND	L11		
NA	GND	L6		
NA	GND	K10		
NA	GND	K9		
NA	GND	K8		
NA	GND	K7		
NA	GND	J10		
NA	GND	J9		
NA	GND	J8		
NA	GND	J7		
NA	GND	H10		
NA	GND	H9		
NA	GND	H8		
NA	GND	H7		
NA	GND	G10		
NA	GND	G9		
NA	GND	G8		
NA	GND	G7		
NA	GND	F11		
NA	GND	F6		
NA	GND	C14		
NA	GND	C3		
NA	GND	B15		
NA	GND	B2		
NA	GND	A16		
NA	GND	A1		

Pinout Diagrams

This section contains pinout diagrams for the following Virtex-II packages:

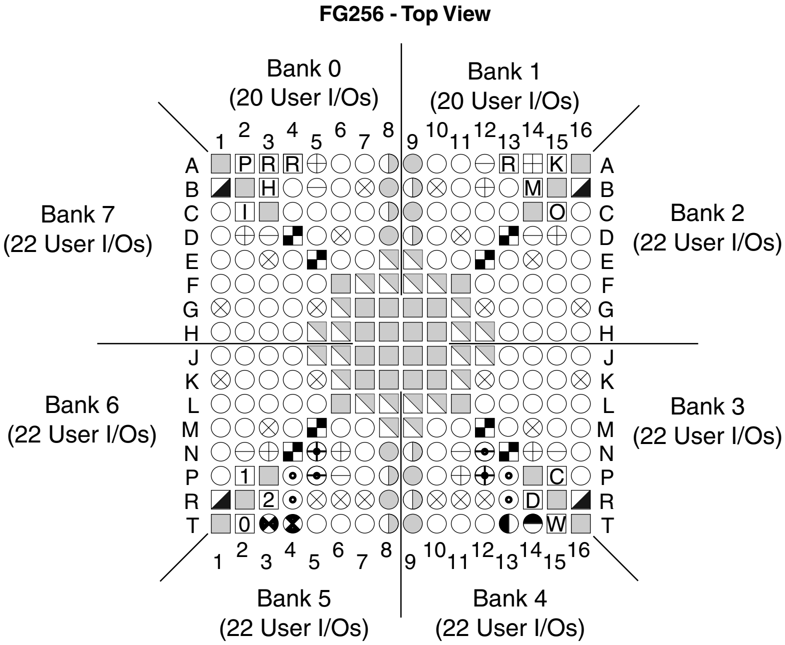
CS144 Chip-Scale BGA Composite Pinout Diagram



User I/O Pins	**Dedicated Pins**	
○ IO_LXXY_#	C CCLK	⊞ VBATT
Dual-Purpose Pins:	P PROG_B	R RSVD
⊙ DIN/D0-D7	D DONE	◸ VCCO
⧓ CS_B	2 1 0 M2, M1, M0	◢ VCCAUX
⧗ RDWR_B	H HSWAP_EN	▚ VCCINT
◓ BUSY/DOUT	K TCK	▨ GND
◐ INIT_B	I TDI	n NO CONNECT
● GCLKx (P)	O TDO	
◑ GCLKx (S)	M TMS	
⊖ VRP	W PWRDWN_B	
⊕ VRN		
⊗ VREF		
Triple-Purpose Pins:		
⊖ D2, D4/ALT_VRP		
⊕ D3, D5/ALT_VRN		

Figure 4-1: **CS144 Chip-Scale BGA Composite Pinout Diagram**

FG256 Fine-Pitch BGA Composite Pinout Diagram



User I/O Pins	Dedicated Pins	
○ IO_LXXY_#	C CCLK	
Dual-Purpose Pins:	P PROG_B	
⊙ DIN/D0-D7	D DONE	⊞ VBATT
⧓ CS_B	2 1 0 M2, M1, M0	R RSVD
⧗ RDWR_B	H HSWAP_EN	⧅ VCCO
◓ BUSY/DOUT	K TCK	◪ VCCAUX
◐ INIT_B	I TDI	▚ VCCINT
● GCLKx (P)	O TDO	▩ GND
◍ GCLKx (S)	M TMS	n NO CONNECT
⊖ VRP	W PWRDWN_B	
⊕ VRN		
⊗ VREF		
Triple-Purpose Pins:		
⊝ D2, D4/ALT_VRP		
⨁ D3, D5/ALT_VRN		

ug002_c4_47_031501

Figure 4-2: **FG256 Fine-Pitch BGA Composite Pinout Diagram**

FG256 Bank Information

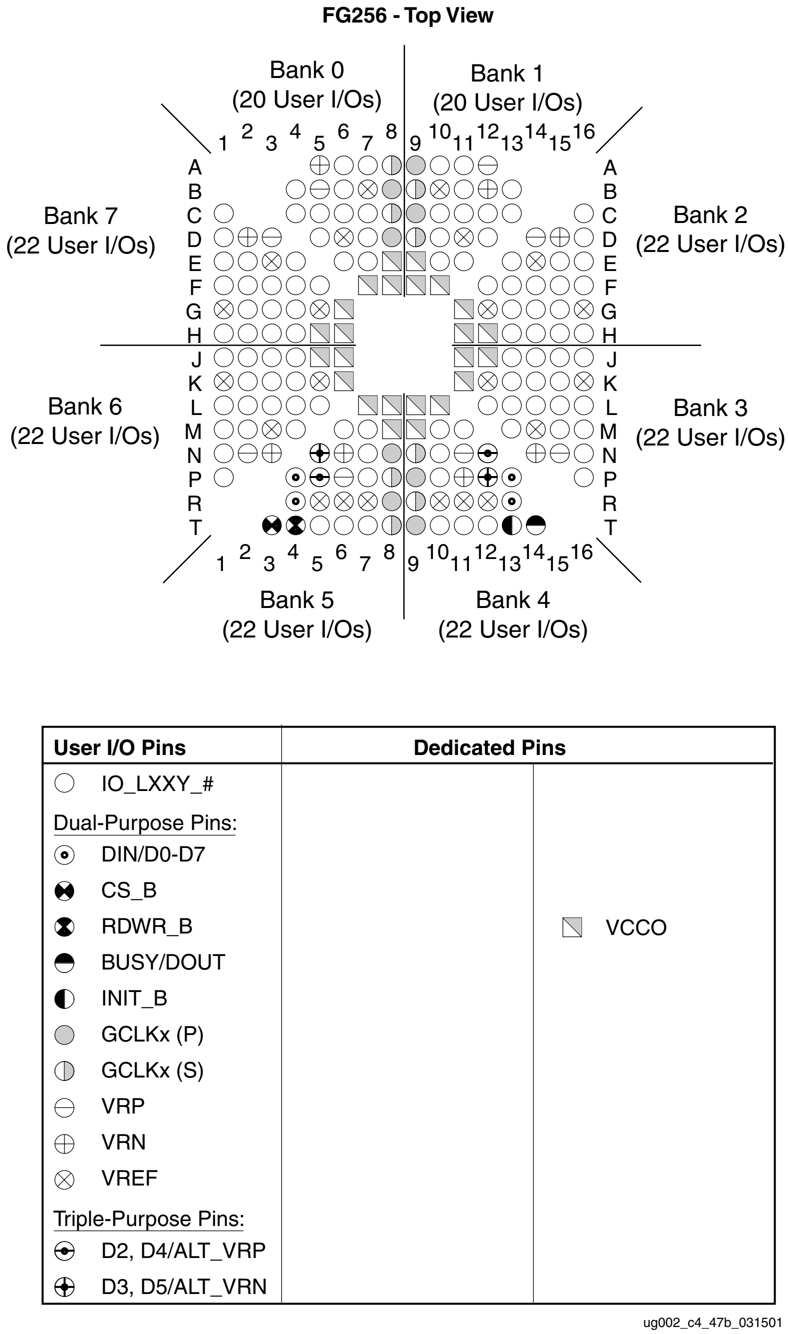
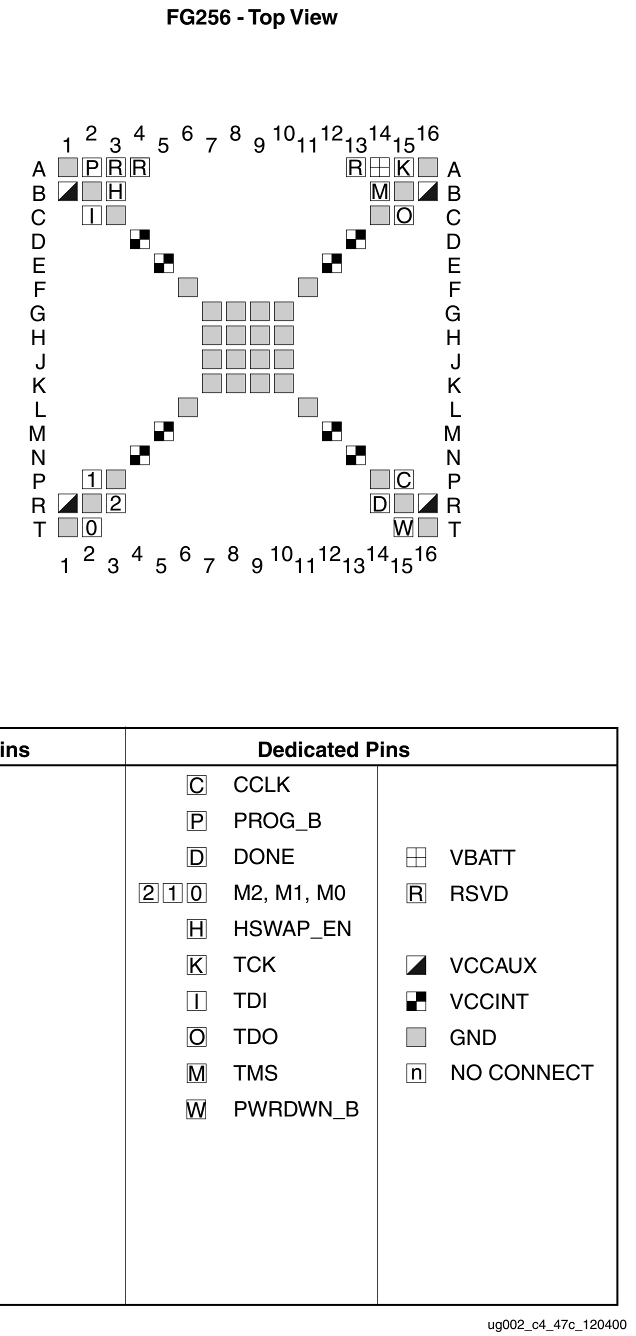


Figure 4-3: **FG256 Bank Information**

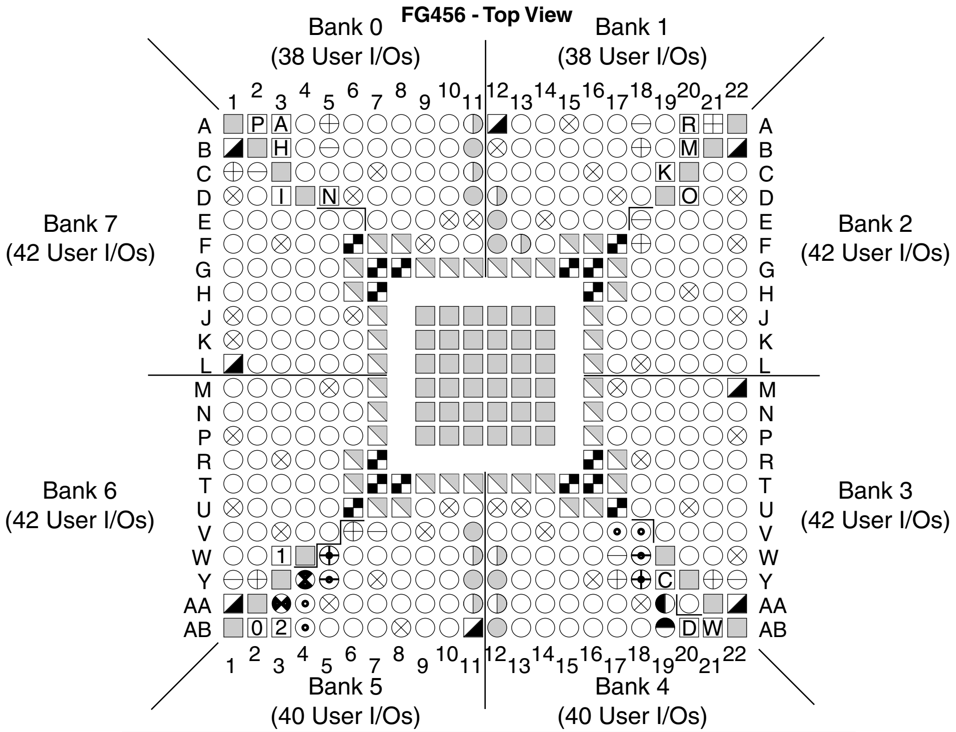
FG256 Dedicated Pins



User I/O Pins	Dedicated Pins			
	C	CCLK		
	P	PROG_B		
	D	DONE	⊞	VBATT
	2 1 0	M2, M1, M0	R	RSVD
	H	HSWAP_EN		
	K	TCK	◪	VCCAUX
	I	TDI	▚	VCCINT
	O	TDO	▨	GND
	M	TMS	n	NO CONNECT
	W	PWRDWN_B		

Figure 4-4: **FG256 Dedicated Pins**

FG456 Fine-Pitch BGA Composite Pinout Diagram

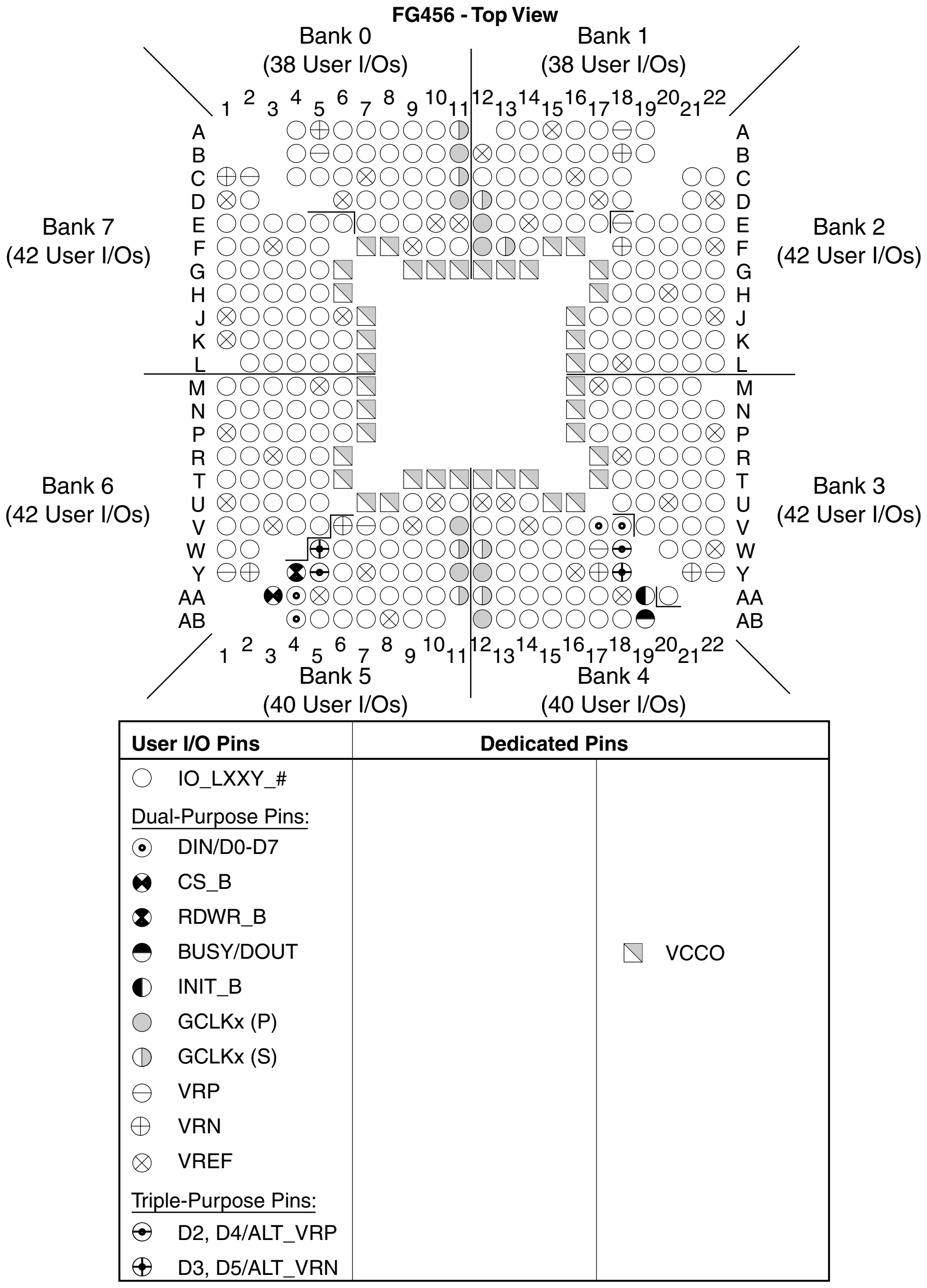


User I/O Pins	Dedicated Pins	
○ IO_LXXY_#	C CCLK	N DXN
Dual-Purpose Pins:	P PROG_B	A DXP
⊙ DIN/D0-D7	D DONE	⊞ VBATT
⋈ CS_B	2 1 0 M2, M1, M0	R RSVD
⧗ RDWR_B	H HSWAP_EN	▧ VCCO
◓ BUSY/DOUT	K TCK	◢ VCCAUX
◐ INIT_B	I TDI	▚ VCCINT
● GCLKx (P)	O TDO	■ GND
◑ GCLKx (S)	M TMS	n NO CONNECT
⊖ VRP	W PWRDWN_B	
⊕ VRN		
⊗ VREF		
Triple-Purpose Pins:		
⊖ D2, D4/ALT_VRP		
⊕ D3, D5/ALT_VRN		

ug002_c4_48_031501

Figure 4-5: **FG456 Fine-Pitch BGA Composite Pinout Diagram**

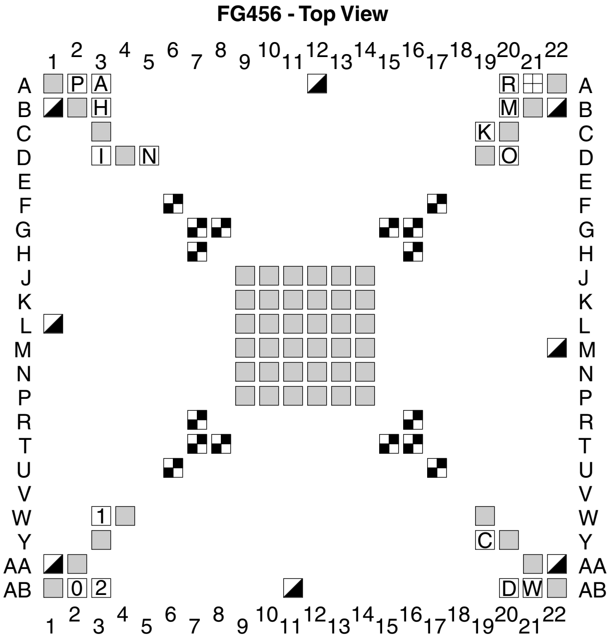
FG456 Bank Information



ug002_c4_48b_031501

Figure 4-6: **FG456 Bank Information**

FG456 Dedicated Pins



User I/O Pins	Dedicated Pins			
	C	CCLK	N	DXN
	P	PROG_B	A	DXP
	D	DONE	⊞	VBATT
	2 1 0	M2, M1, M0	R	RSVD
	H	HSWAP_EN		
	K	TCK	◪	VCCAUX
	I	TDI	▚	VCCINT
	O	TDO	▨	GND
	M	TMS	n	NO CONNECT
	W	PWRDWN_B		

ug002_c4_48c_120400

Figure 4-7: **FG456 Dedicated Pins**

FG676 Fine-Pitch BGA Composite Pinout Diagram

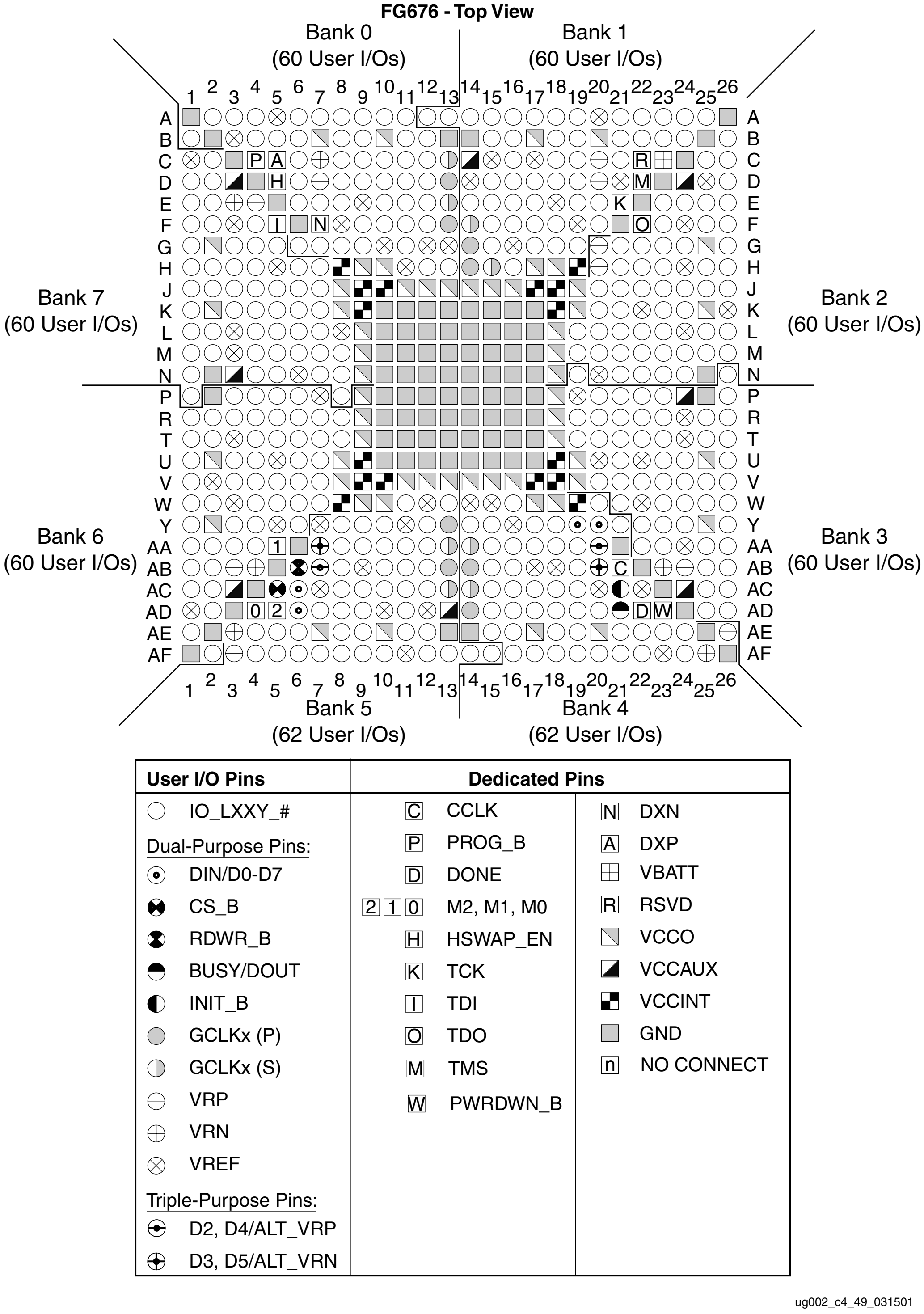
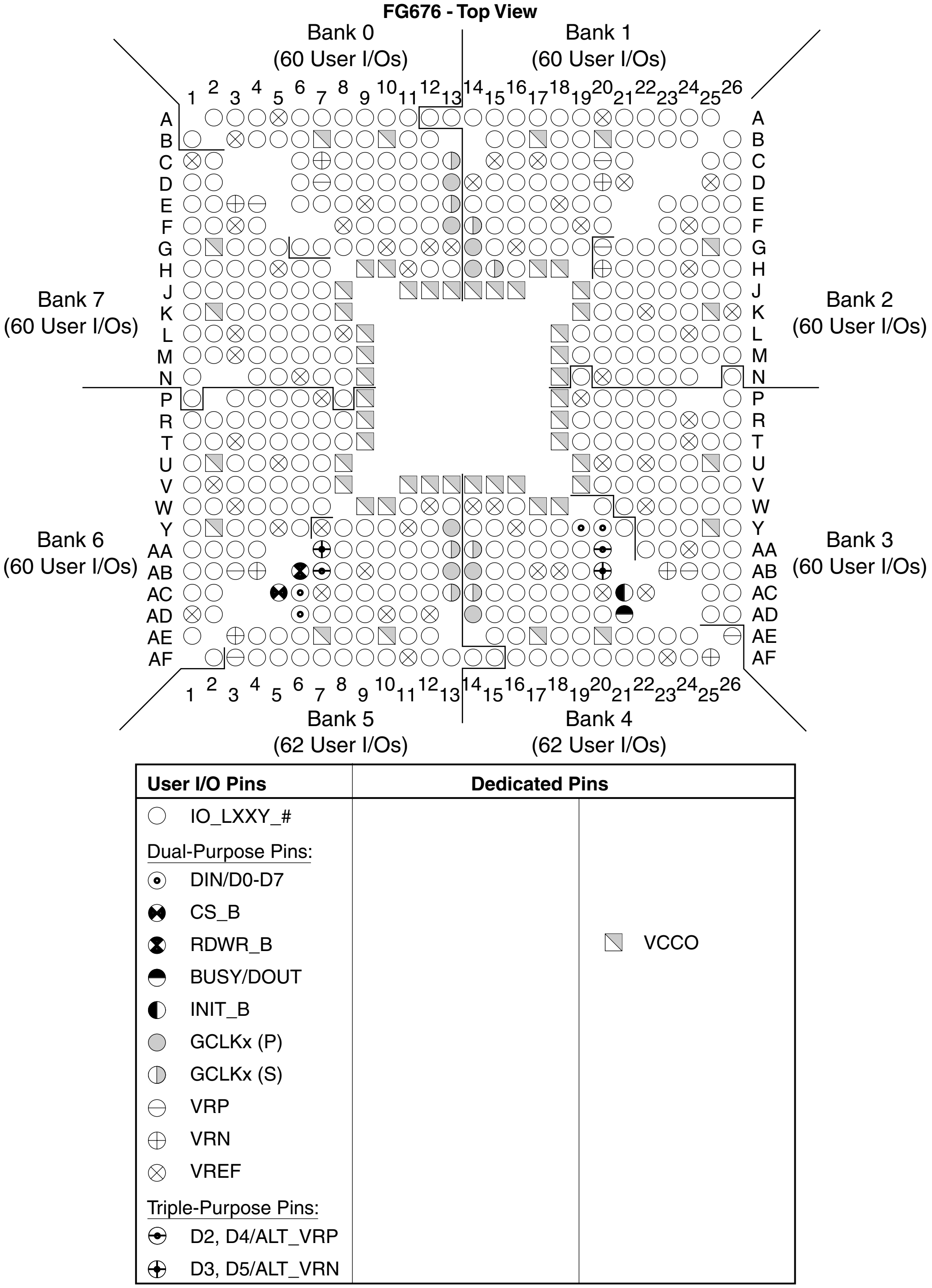


Figure 4-8: **FG676 Fine-Pitch BGA Composite Pinout Diagram**

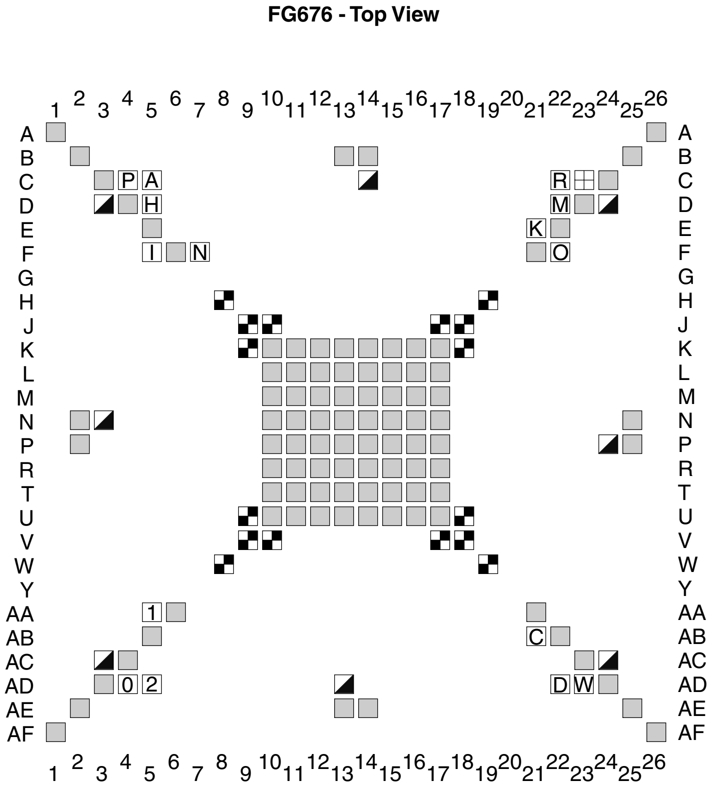
FG676 Bank Information



ug002_c4_49b_031501

Figure 4-9: **FG676 Bank Information**

FG676 Dedicated Pins



User I/O Pins	Dedicated Pins			
	C	CCLK	N	DXN
	P	PROG_B	A	DXP
	D	DONE	⊞	VBATT
	2 1 0	M2, M1, M0	R	RSVD
	H	HSWAP_EN		
	K	TCK	◩	VCCAUX
	I	TDI	▚	VCCINT
	O	TDO	▨	GND
	M	TMS	n	NO CONNECT
	W	PWRDWN_B		

ug002_c4_49c_120400

Figure 4-10: **FG676 Dedicated Pins**

BG575 Standard BGA Composite Pinout Diagram

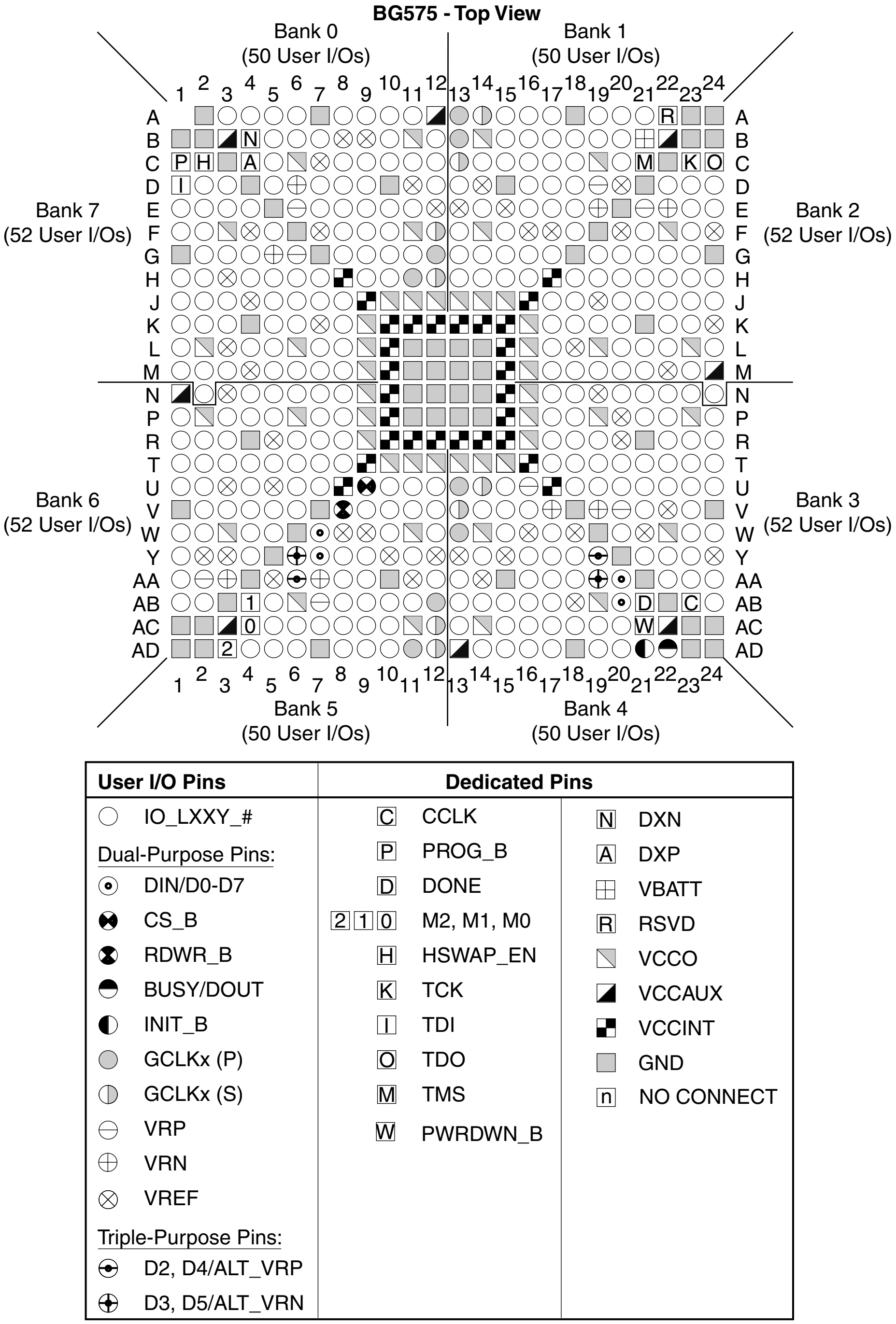


Figure 4-11: **BG575 Standard BGA Composite Pinout Diagram**

BG575 Bank Information

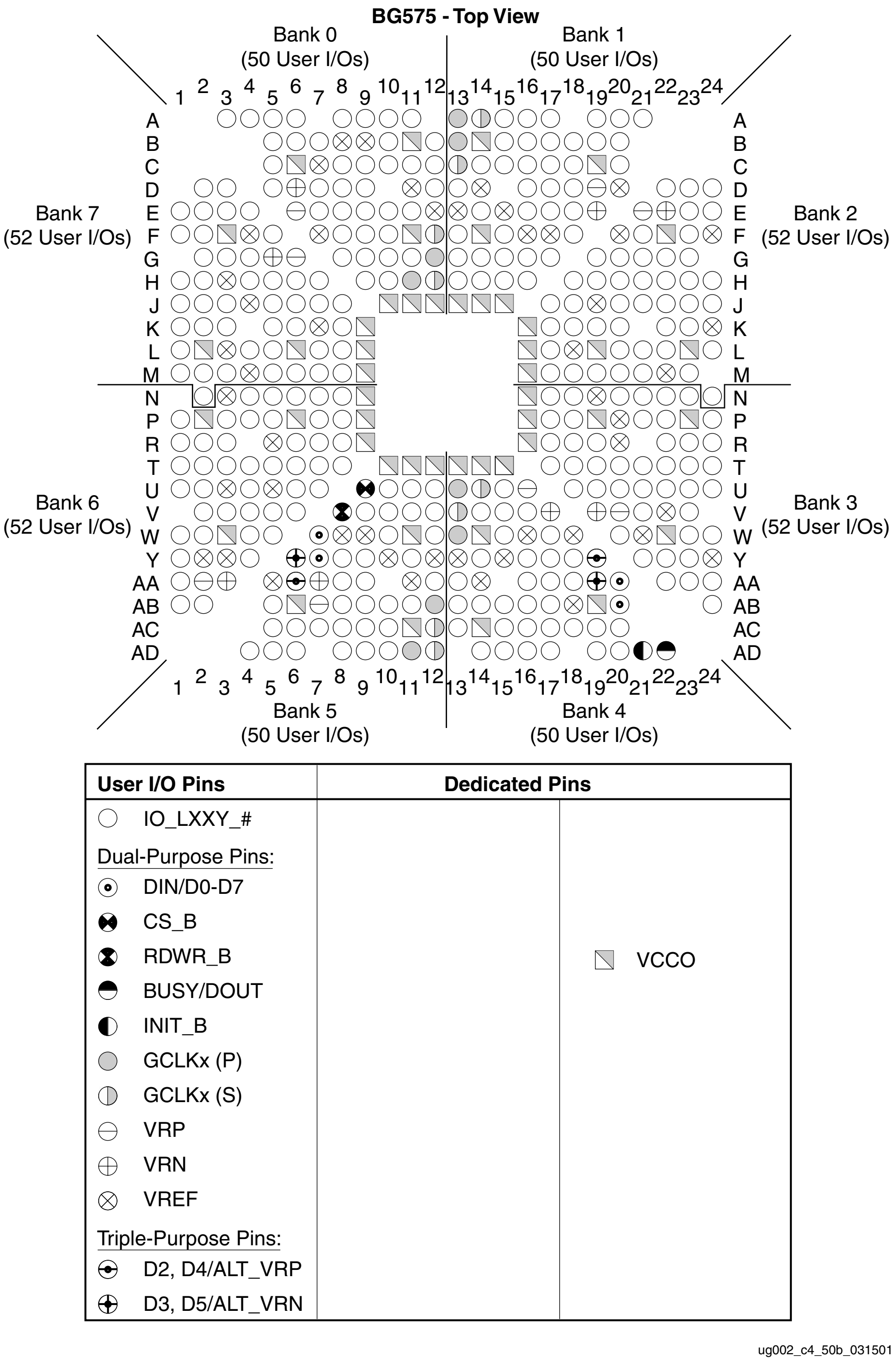


Figure 4-12: **BG575 Bank Information**

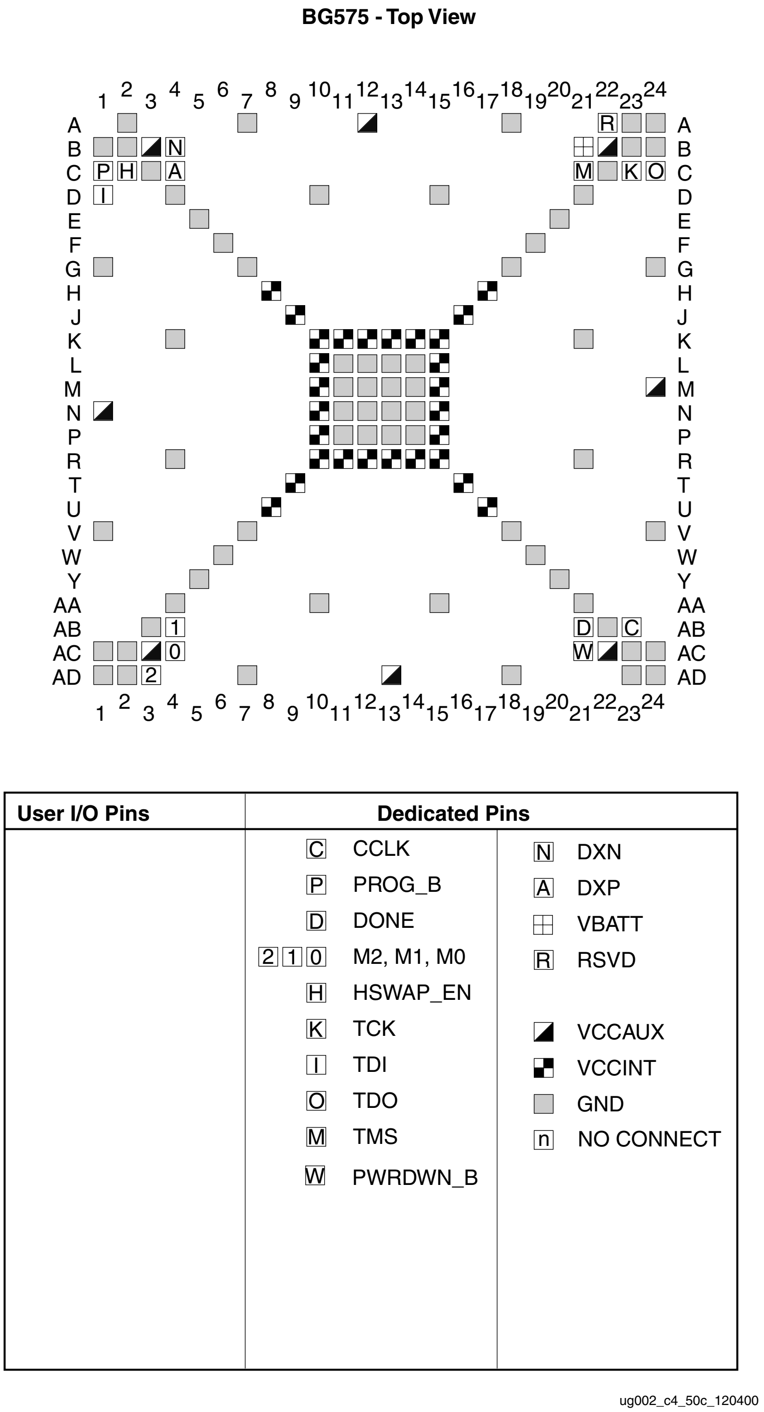


Figure 4-13: **BG575 Dedicated Pins**

BG728 Standard BGA Composite Pinout Diagram

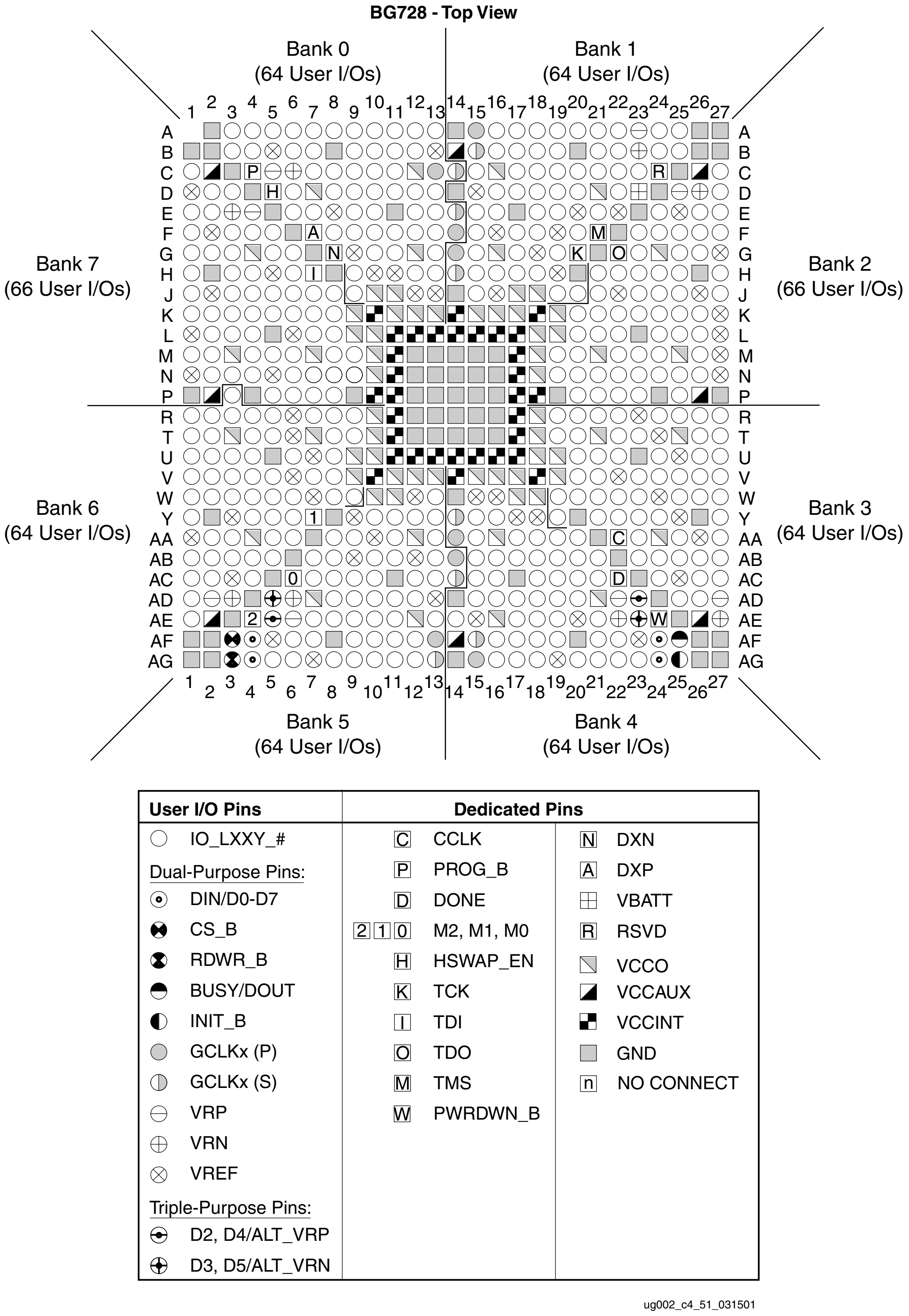


Figure 4-14: **BG728 Standard BGA Composite Pinout Diagram**

BG728 Bank Information

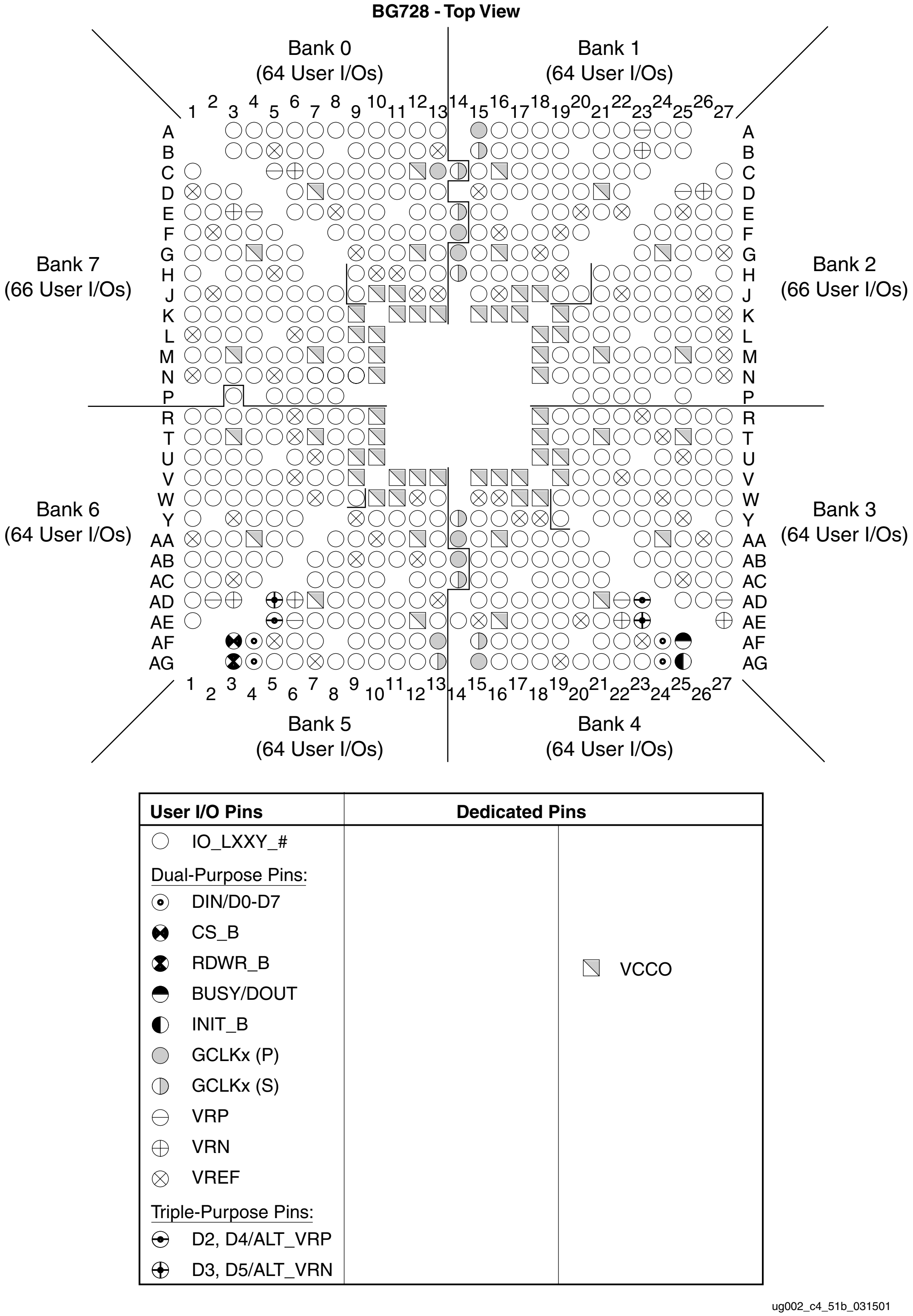
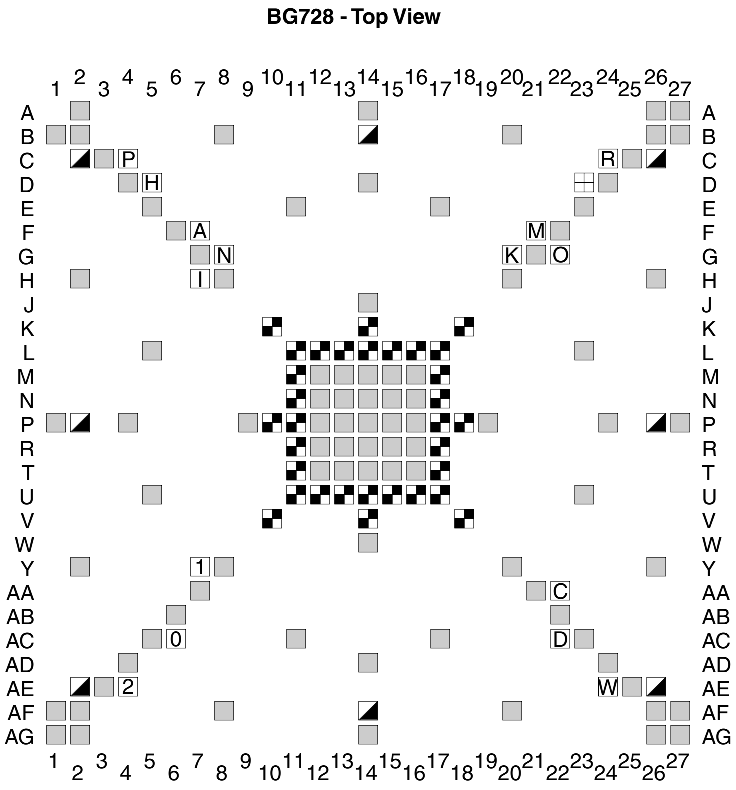


Figure 4-15: **BG728 Bank Information**

BG728 Dedicated Pins



User I/O Pins	Dedicated Pins			
	C	CCLK	N	DXN
	P	PROG_B	A	DXP
	D	DONE	⊞	VBATT
	2 1 0	M2, M1, M0	R	RSVD
	H	HSWAP_EN		
	K	TCK	◪	VCCAUX
	I	TDI	▚	VCCINT
	O	TDO	▨	GND
	M	TMS	n	NO CONNECT
	W	PWRDWN_B		

ug002_c4_51c_120400

Figure 4-16: **BG728 Dedicated Pins**

FF896 Flip-Chip Fine-Pitch BGA Composite Pinout Diagram

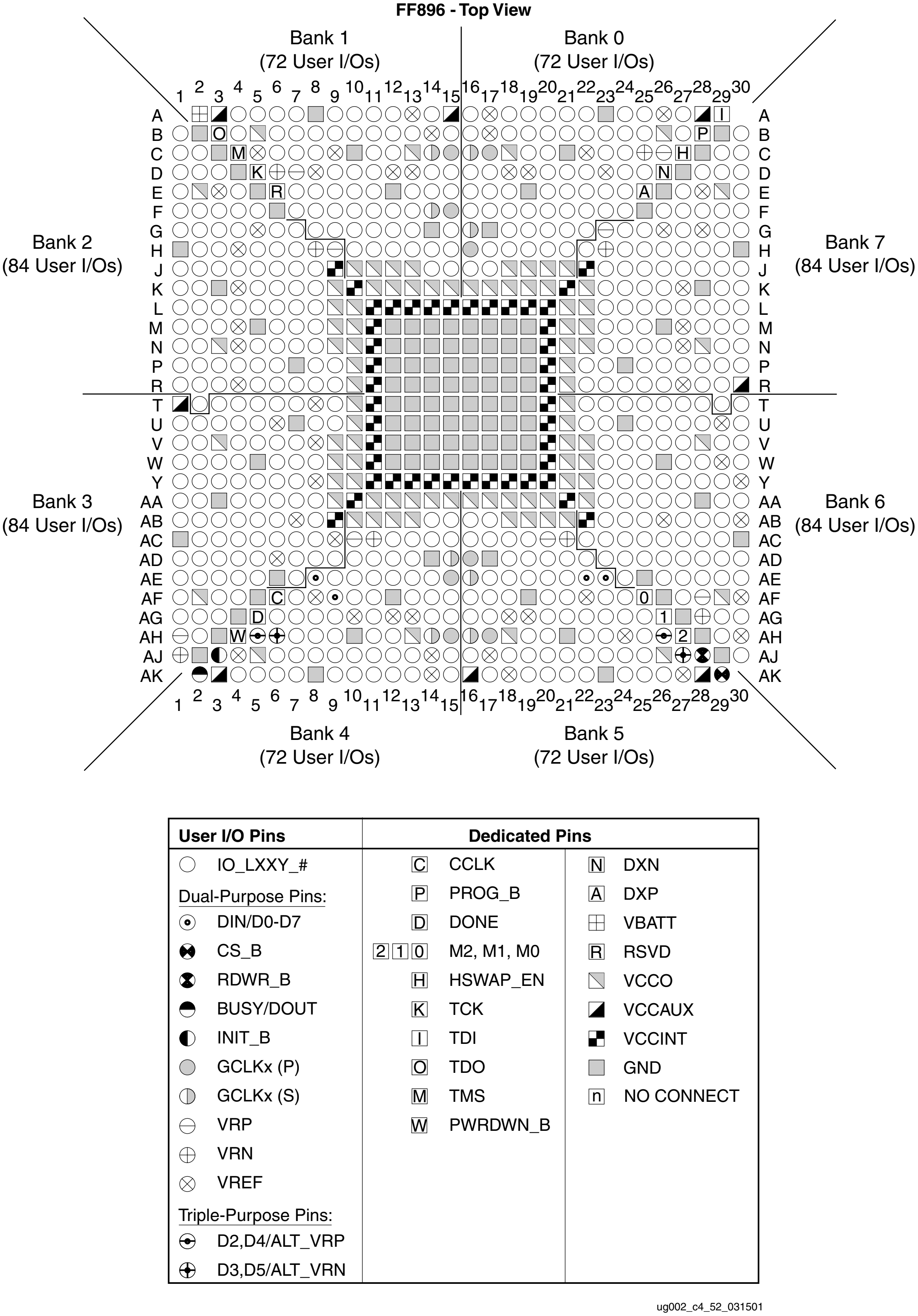


Figure 4-17: **FF896 Flip-Chip Fine-Pitch BGA Composite Pinout Diagram**

FF896 Bank Information

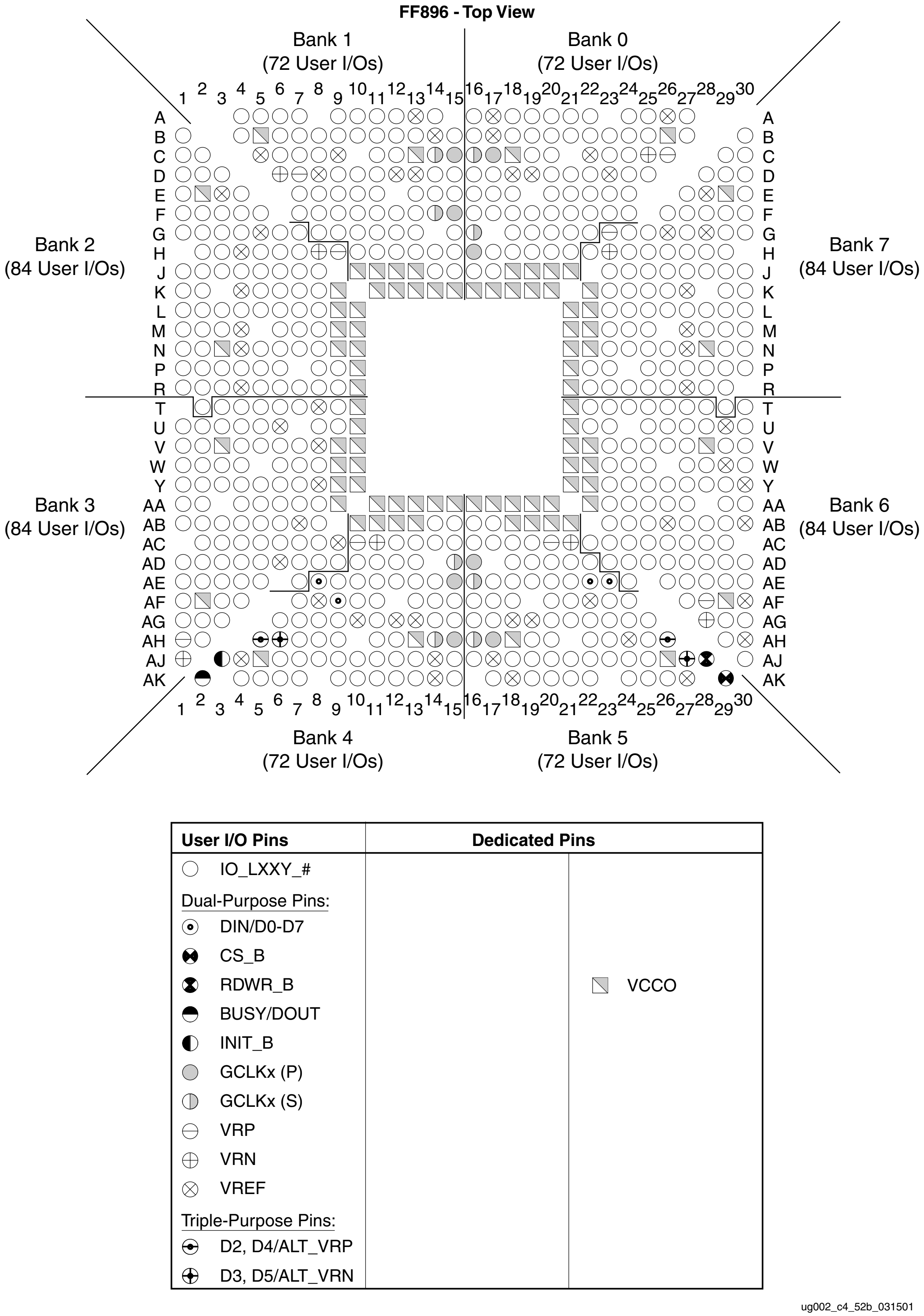
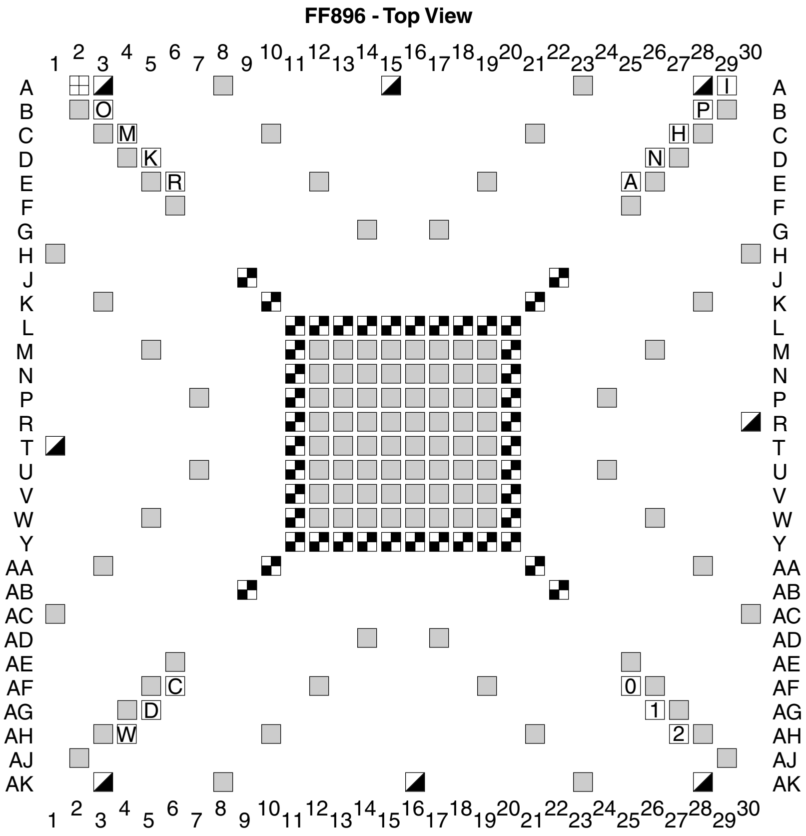


Figure 4-18: **FF896 Bank Information**

FF896 Dedicated Pins

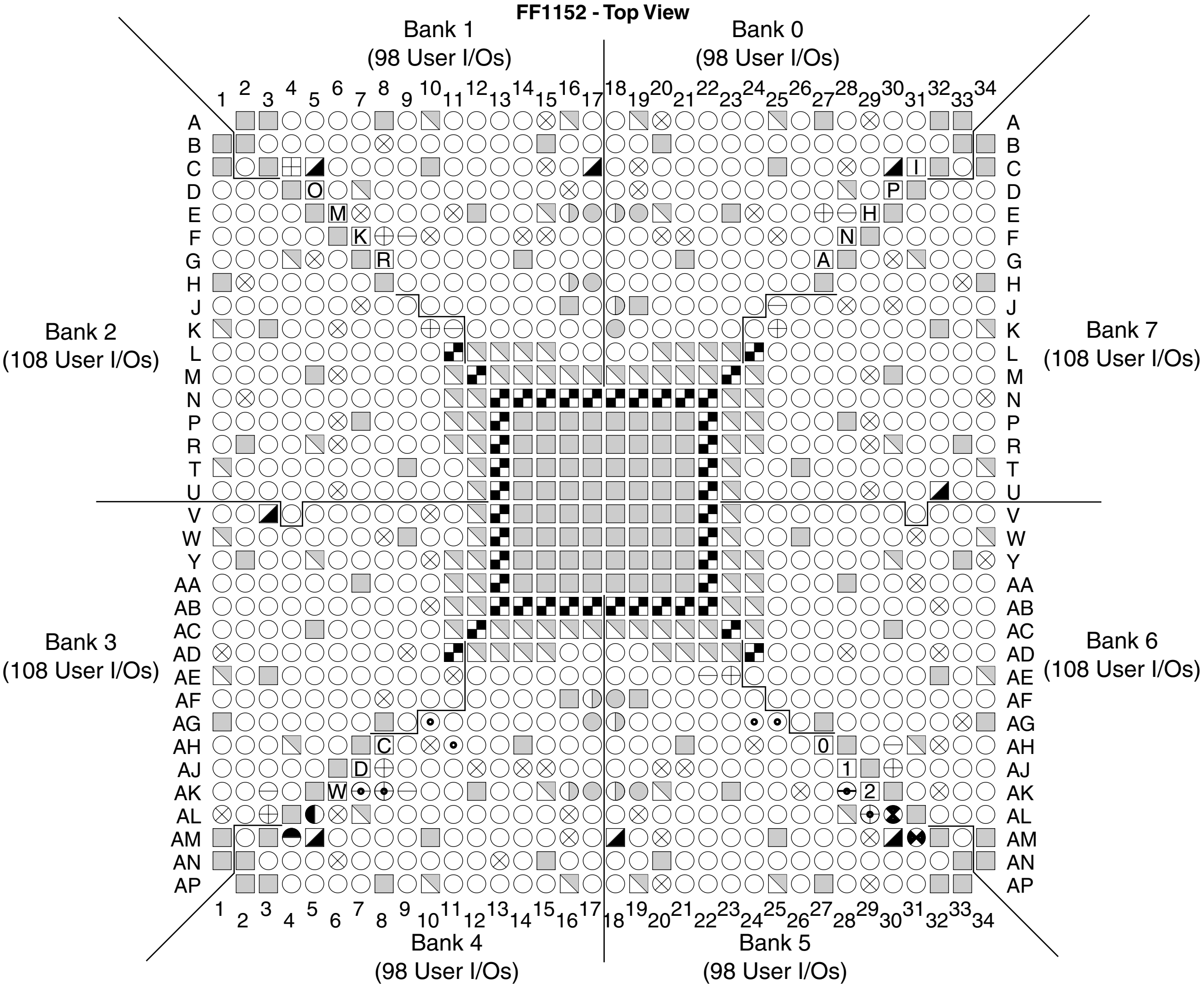

FF896 - Top View

User I/O Pins	Dedicated Pins			
	C	CCLK	N	DXN
	P	PROG_B	A	DXP
	D	DONE	⊞	VBATT
	2 1 0	M2, M1, M0	R	RSVD
	H	HSWAP_EN		
	K	TCK	◪	VCCAUX
	I	TDI	▚	VCCINT
	O	TDO	▨	GND
	M	TMS	n	NO CONNECT
	W	PWRDWN_B		

ug002_c4_52c_120400

Figure 4-19: **FF896 Dedicated Pins**

FF1152 Flip-Chip Fine-Pitch BGA Composite Pinout Diagram



User I/O Pins		Dedicated Pins			
○	IO_LXXY_#	C	CCLK	N	DXN
Dual-Purpose Pins:		P	PROG_B	A	DXP
⊙	DIN/D0-D7	D	DONE	⊞	VBATT
⧓	CS_B	2 1 0	M2, M1, M0	R	RSVD
⧗	RDWR_B	H	HSWAP_EN	⧄	VCCO
◓	BUSY/DOUT	K	TCK	◢	VCCAUX
◐	INIT_B	I	TDI	▚	VCCINT
●	GCLKx (P)	O	TDO	▨	GND
◑	GCLKx (S)	M	TMS	n	NO CONNECT
⊖	VRP	W	PWRDWN_B		
⊕	VRN				
⊗	VREF				
Triple-Purpose Pins:					
⊝	D2, D4/ALT_VRP				
⊛	D3, D5/ALT_VRN				

ug002_c4_53_031501

Figure 4-20: **FF1152 Flip-Chip Fine-Pitch BGA Composite Pinout Diagram**

FF1152 Bank Information

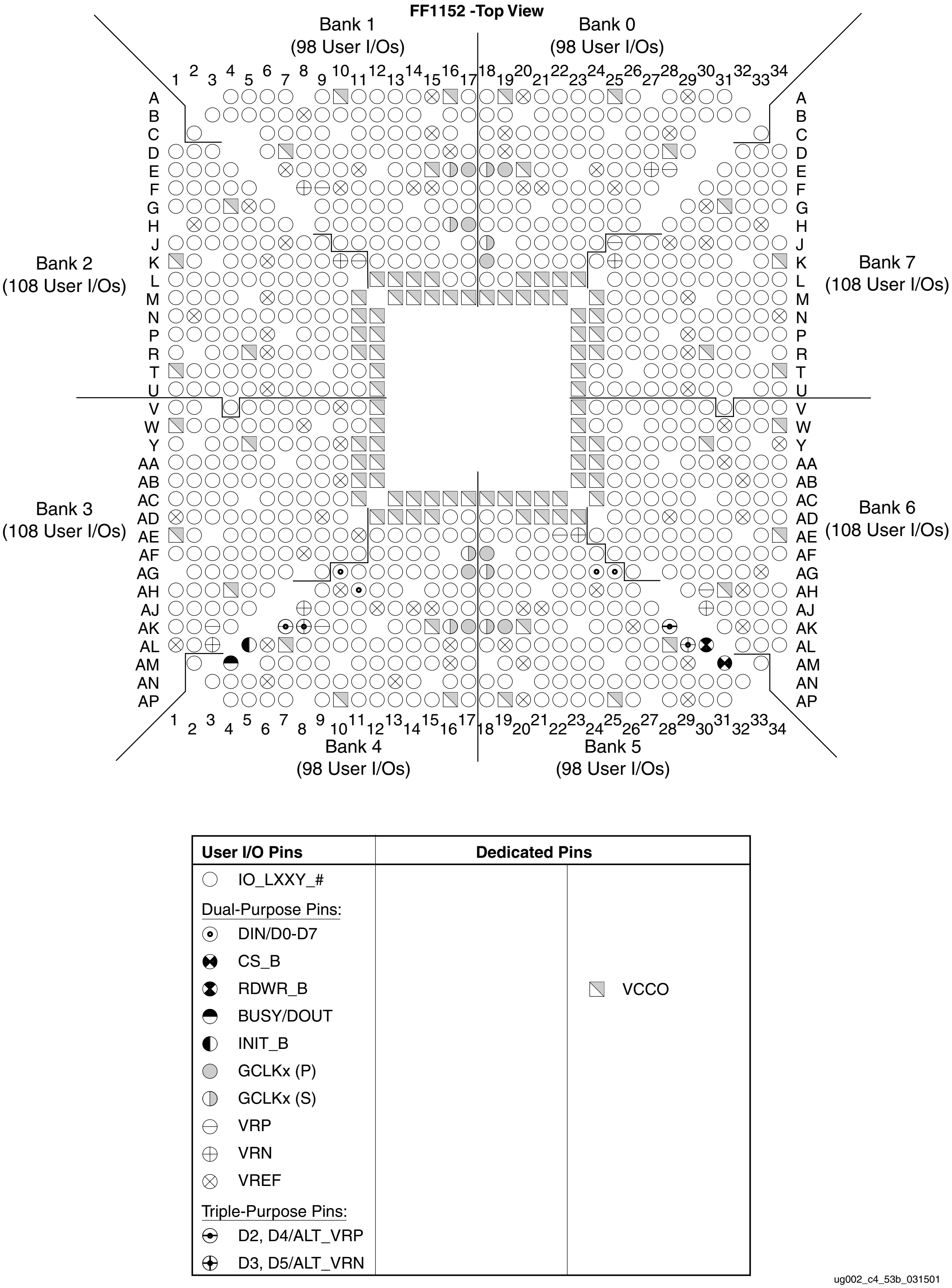
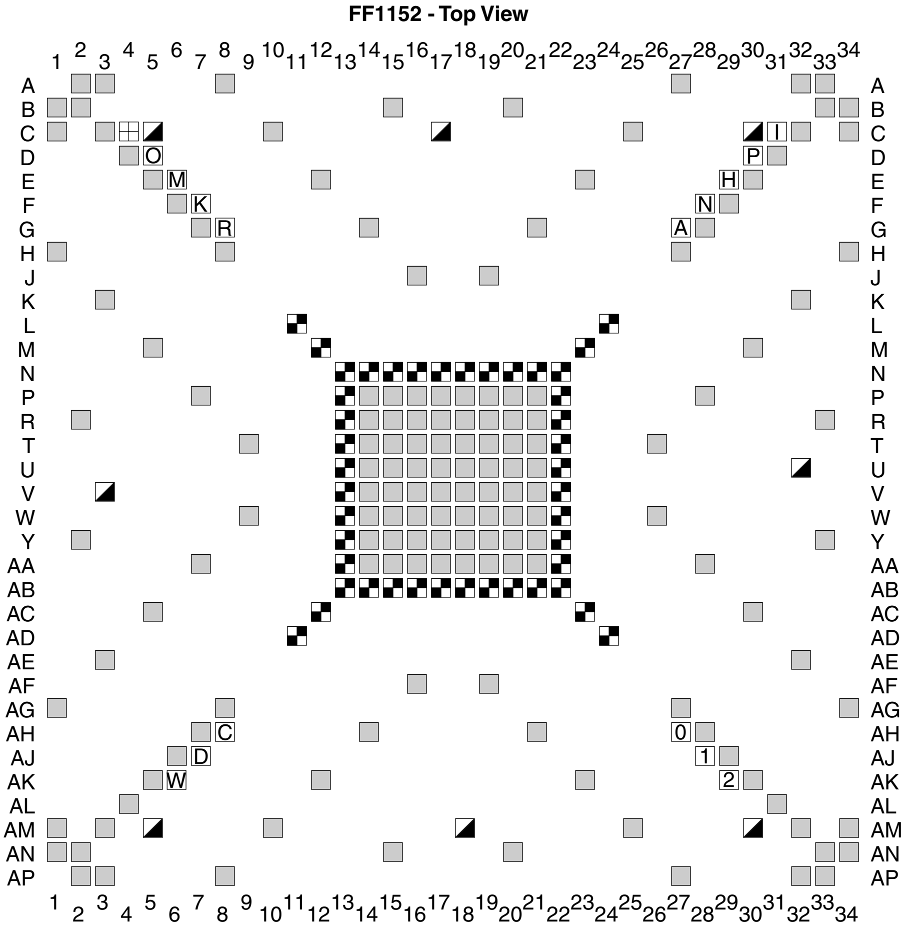


Figure 4-21: **FF1152 Bank Information**

FF1152 Dedicated Pins



User I/O Pins	Dedicated Pins			
	C	CCLK	N	DXN
	P	PROG_B	A	DXP
	D	DONE	⊞	VBATT
	2 1 0	M2, M1, M0	R	RSVD
	H	HSWAP_EN		
	K	TCK	◪	VCCAUX
	I	TDI	▚	VCCINT
	O	TDO	▨	GND
	M	TMS	n	NO CONNECT
	W	PWRDWN_B		

ug002_c4_53c_120400

Figure 4-22: **FF1152 Dedicated Pins**

FF1517 Flip-Chip Fine-Pitch BGA Composite Pinout Diagram

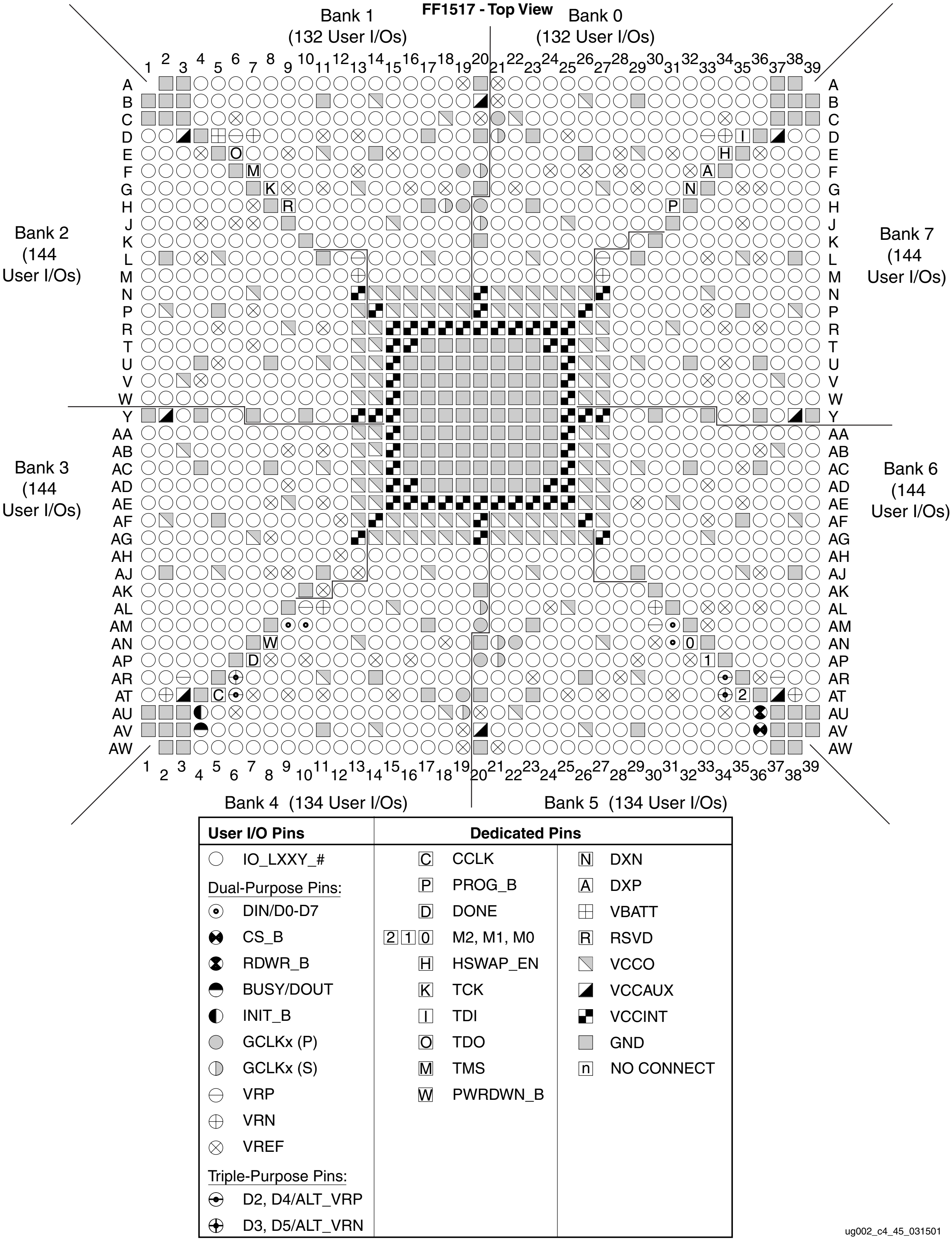


Figure 4-23: **FF1517 Flip-Chip Fine-Pitch BGA Composite Pinout Diagram**

FF1517 Bank Information

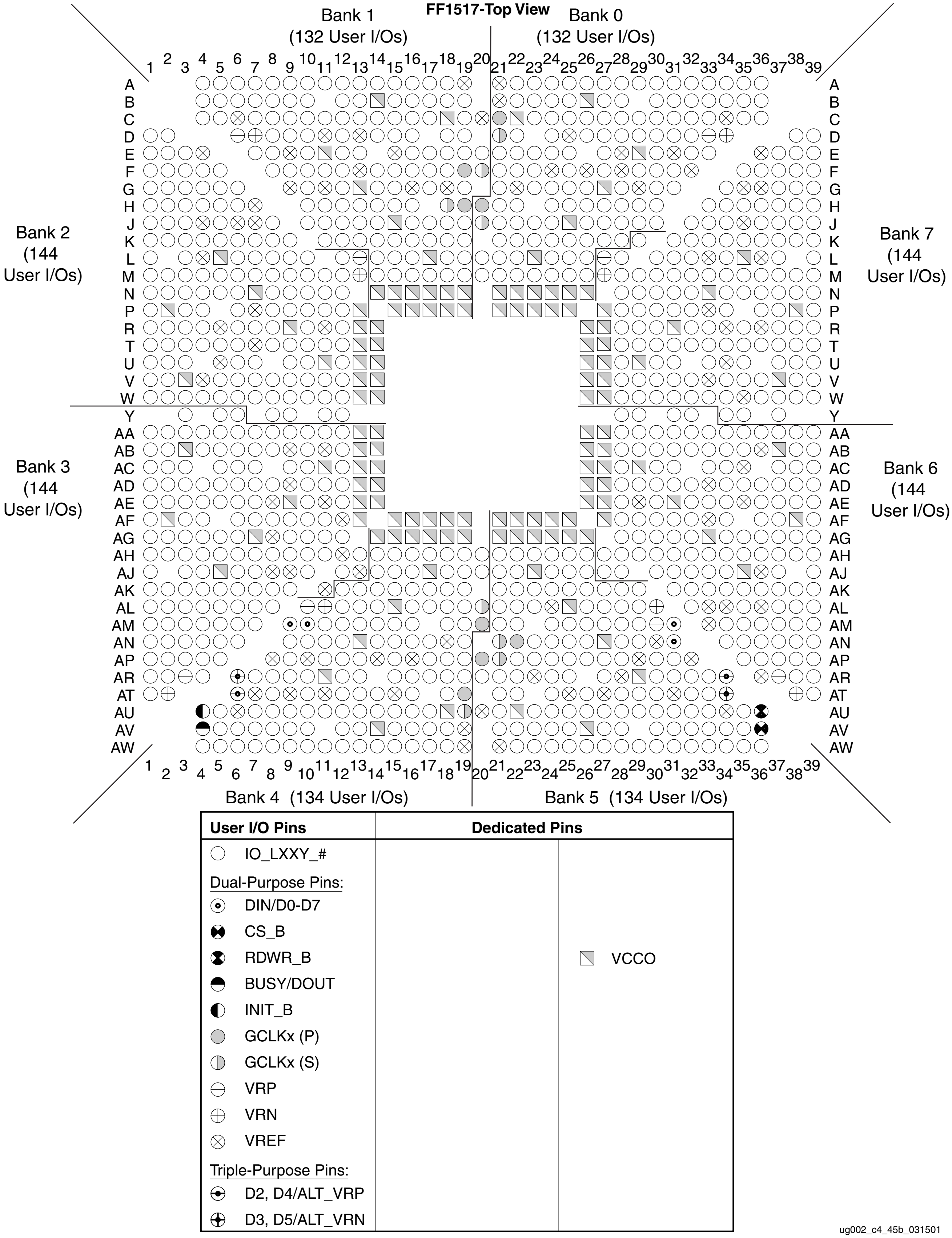


Figure 4-24: **FF1517 Bank Information**

FF1517 Dedicated Pins

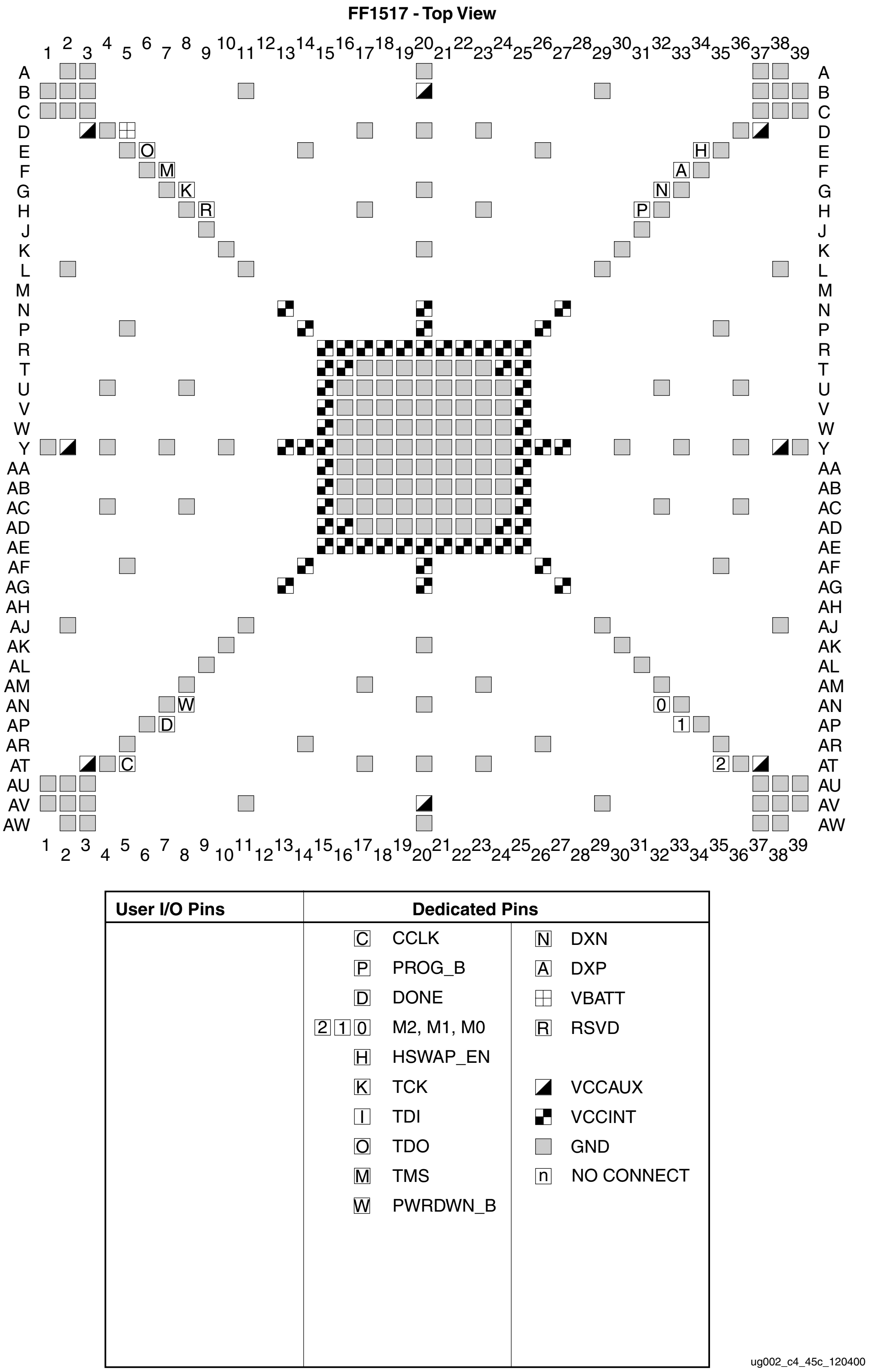
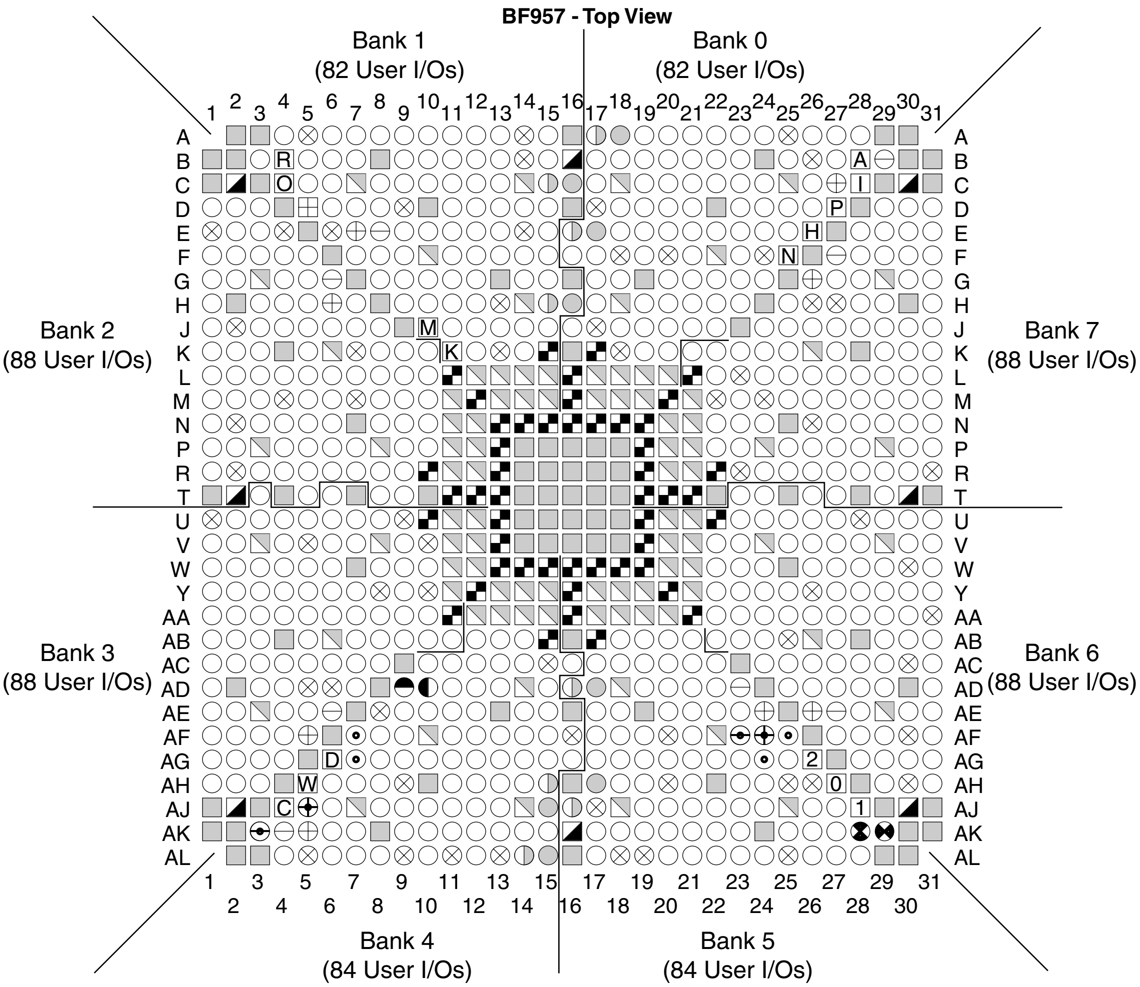


Figure 4-25: **FF1517 Dedicated Pins**

BF957 Flip-Chip BGA Composite Pinout Diagram



User I/O Pins	**Dedicated Pins**	
○ IO_LXXY_#	C CCLK	N DXN
Dual-Purpose Pins:	P PROG_B	A DXP
⊙ DIN/D0-D7	D DONE	⊞ VBATT
⋈ CS_B	2 1 0 M2, M1, M0	R RSVD
⧗ RDWR_B	H HSWAP_EN	▧ VCCO
◓ BUSY/DOUT	K TCK	◢ VCCAUX
◐ INIT_B	I TDI	▚ VCCINT
● GCLKx (P)	O TDO	■ GND
◑ GCLKx (S)	M TMS	n NO CONNECT
⊖ VRP	W PWRDWN_B	
⊕ VRN		
⊗ VREF		
Triple-Purpose Pins:		
⊖ D2, D4/ALT_VRP		
⊕ D3, D5/ALT_VRN		

ug002_c4_54_032901

Figure 4-26: **BF957 Flip-Chip BGA Composite Pinout Diagram**

BF957 Bank Information

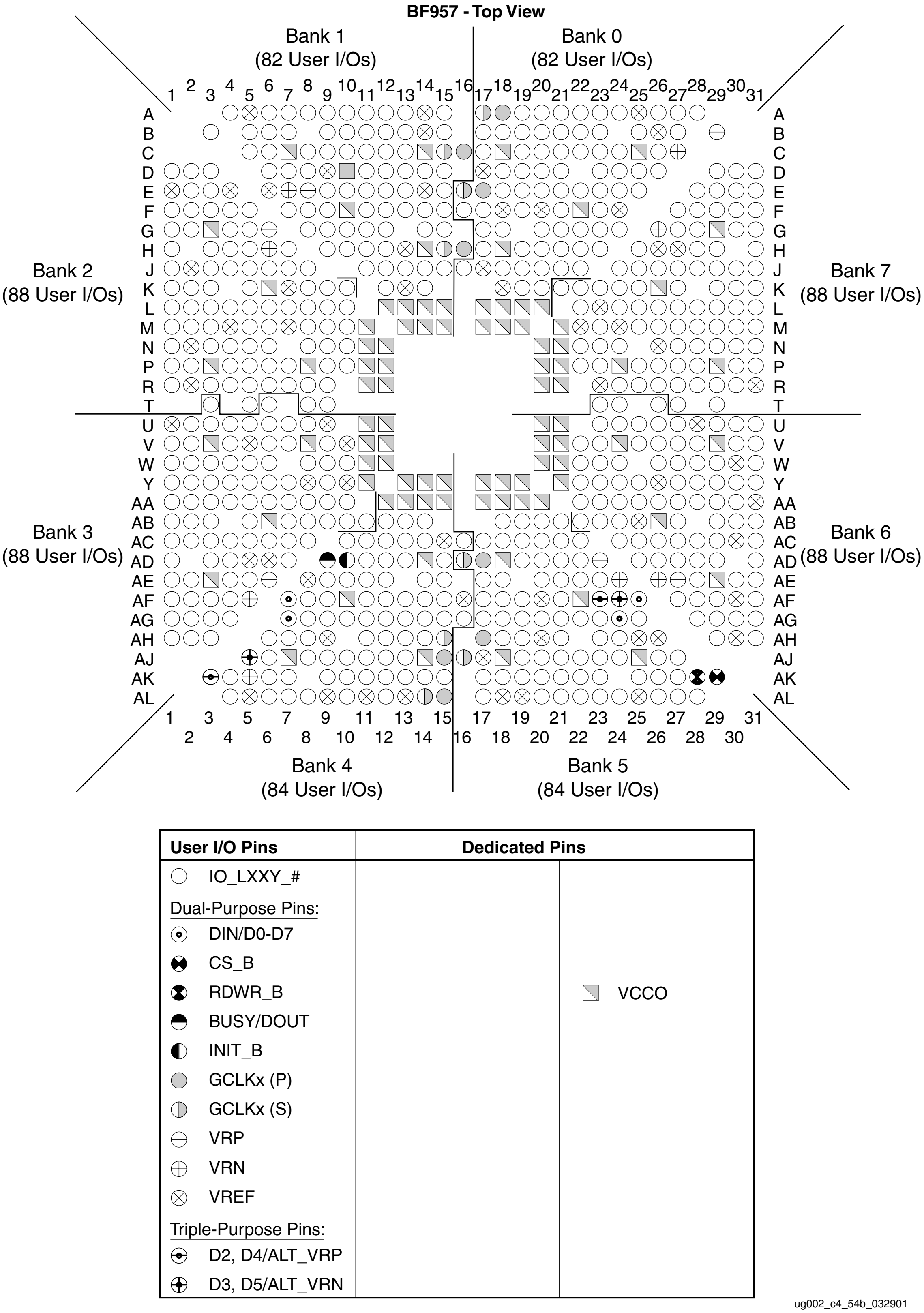
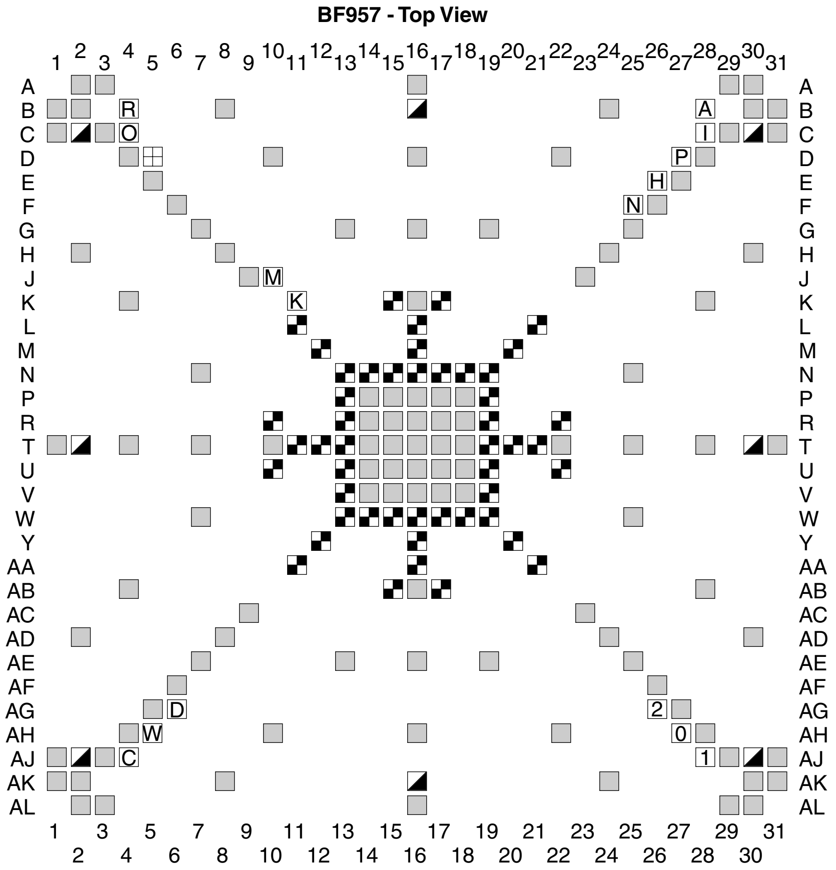


Figure 4-27: **BF957 Bank Information**

BF957 Dedicated Pins

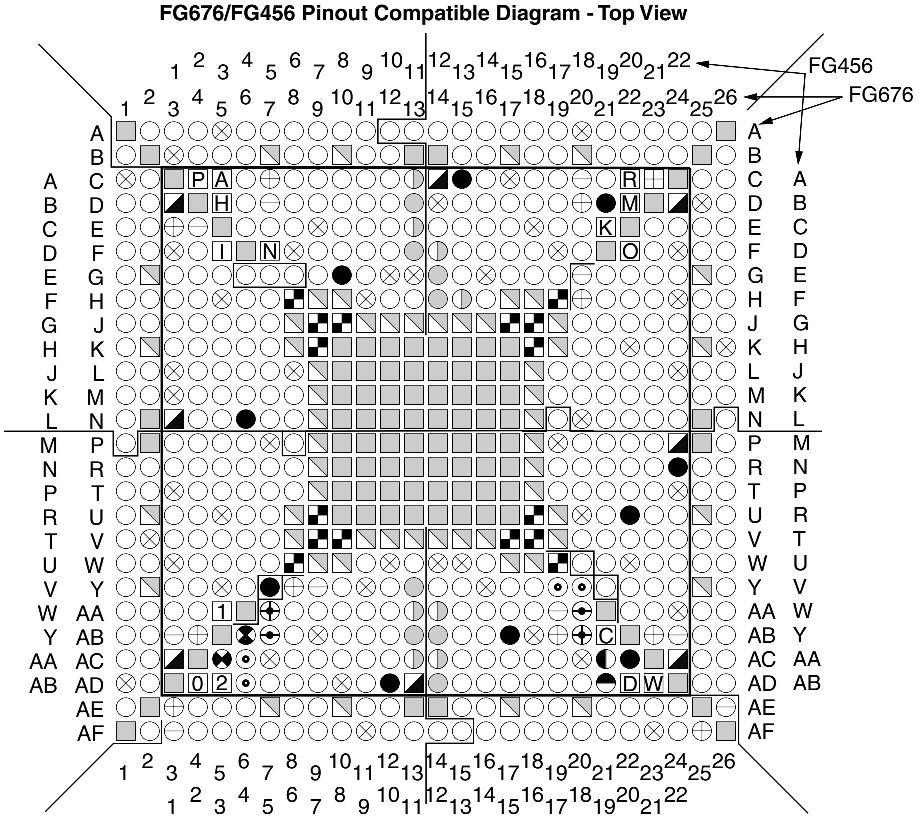


User I/O Pins	Dedicated Pins			
	C	CCLK	N	DXN
	P	PROG_B	A	DXP
	D	DONE	⊞	VBATT
	2 1 0	M2, M1, M0	R	RSVD
	H	HSWAP_EN		
	K	TCK	◪	VCCAUX
	I	TDI	▚	VCCINT
	O	TDO	▢	GND
	M	TMS	n	NO CONNECT
	W	PWRDWN_B		

ug002_c4_54c_120400

Figure 4-28: **BF957 Dedicated Pins**

FG456 - FG676 Pinout Compatibility Diagram



Note: FF456 and FG676 are pinout compatible with with the exception of the LVDS pairs. I/O V_{REF} pins in FG676 are user I/O pins in FG456. In addition, some user I/O pins are not in the same bank (see ○○ lines). VRP (V7) and VRN (V6) in Bank 5 and VRP (W17) and VRN (Y17) in Bank 4 are only user I/Os in FG676.

User I/O Pins	Dedicated Pins	
○ IO_LXXY_#	C CCLK	N DXN
Dual-Purpose Pins:	P PROG_B	A DXP
⊙ DIN/D0-D7	D DONE	⊞ VBATT
⧓ CS_B	2 1 0 M2, M1, M0	R RSVD
⧗ RDWR_B	H HSWAP_EN	◺ VCCO
◓ BUSY/DOUT	K TCK	◢ VCCAUX
◐ INIT_B	I TDI	▚ VCCINT
◍ GCLKx (P)	O TDO	▨ GND
⦶ GCLKx (S)	M TMS	n NO CONNECT
⊖ VRP	W PWRDWN_B	
⊕ VRN		
⊗ VREF		
● VREF on FG676 User I/O on FG456		
Triple-Purpose Pins:		
⊝ D2, D4/ALT_VRP		
⨁ D3, D5/ALT_VRN		

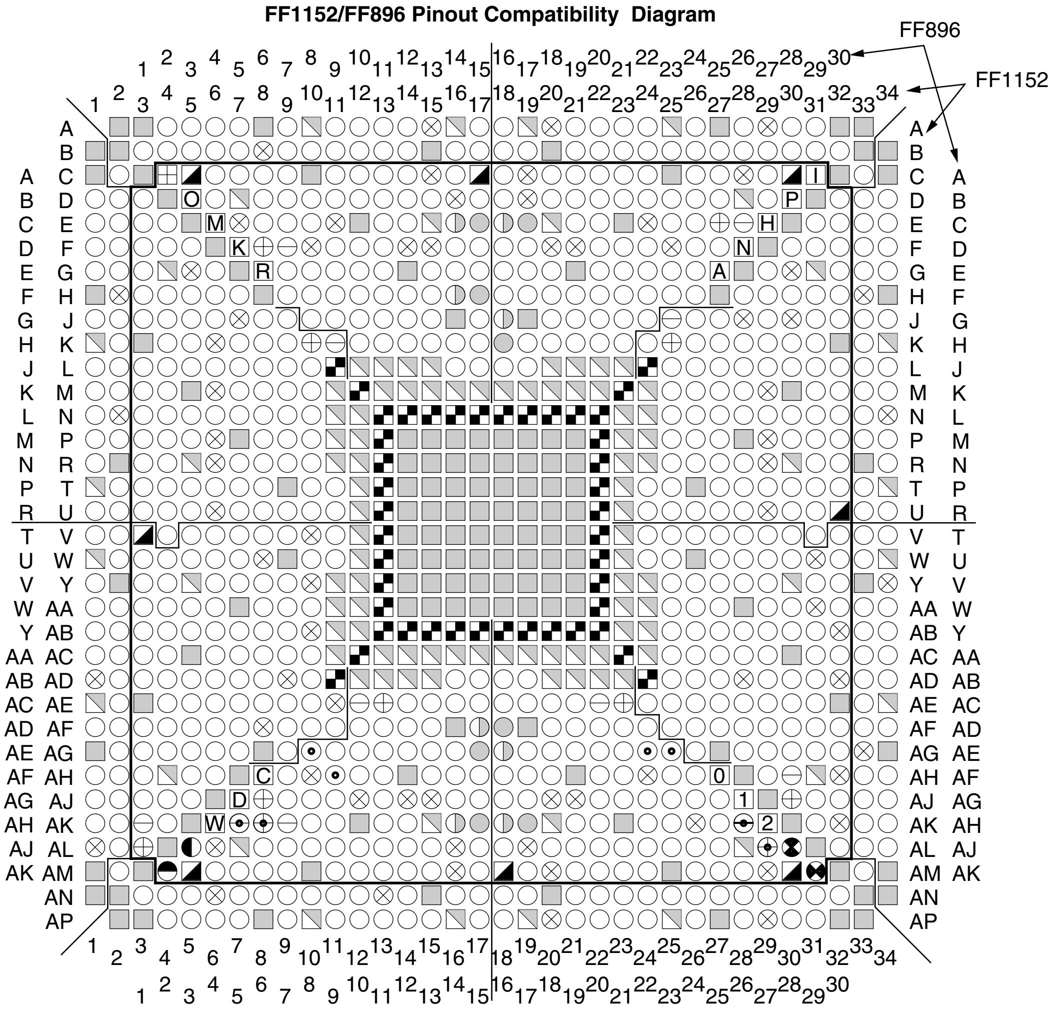
Corresponding Pinouts

FG456	FG676
A1	C3
.	.
.	.
.	.
.	.
.	.
AB22	AD24

ug002_c4_56_080601

Figure 4-29: **FG456 - FG676 Pinout Compatibility Diagram**

FF896 - FF1152 Pinout Compatibility Diagram



Note: FF896 is pinout compatible with the FF1152 except for LVDS pairs. Also, in Bank 4, VRP/VRN pins are not compatible: for FF896, VRP is in AC10 and VRN is in AC11, and for FF1152, VRP is in AK9 and VRN is in AJ8. If DCI is not used in Bank 4, or is used with ALT_VRP or ALT_VRN, then the user I/Os are compatible.

User I/O Pins	Dedicated Pins	
○ IO_LXXY_#	C CCLK	N DXN
Dual-Purpose Pins:	P PROG_B	A DXP
⊙ DIN/D0-D7	D DONE	⊞ VBATT
⧓ CS_B	2 1 0 M2, M1, M0	R RSVD
⧗ RDWR_B	H HSWAP_EN	▧ VCCO
◒ BUSY/DOUT	K TCK	◢ VCCAUX
◐ INIT_B	I TDI	▚ VCCINT
● GCLKx (P)	O TDO	■ GND
◑ GCLKx (S)	M TMS	n NO CONNECT
⊖ VRP	W PWRDWN_B	
⊕ VRN		
⊗ VREF		
Triple-Purpose Pins:		
⊖ D2, D4/ALT_VRP		
⊕ D3, D5/ALT_VRN		

Corresponding Pinouts

FF896	FF1152
A2	C4
.	.
.	.
.	.
.	.
.	.
AK29	AM31

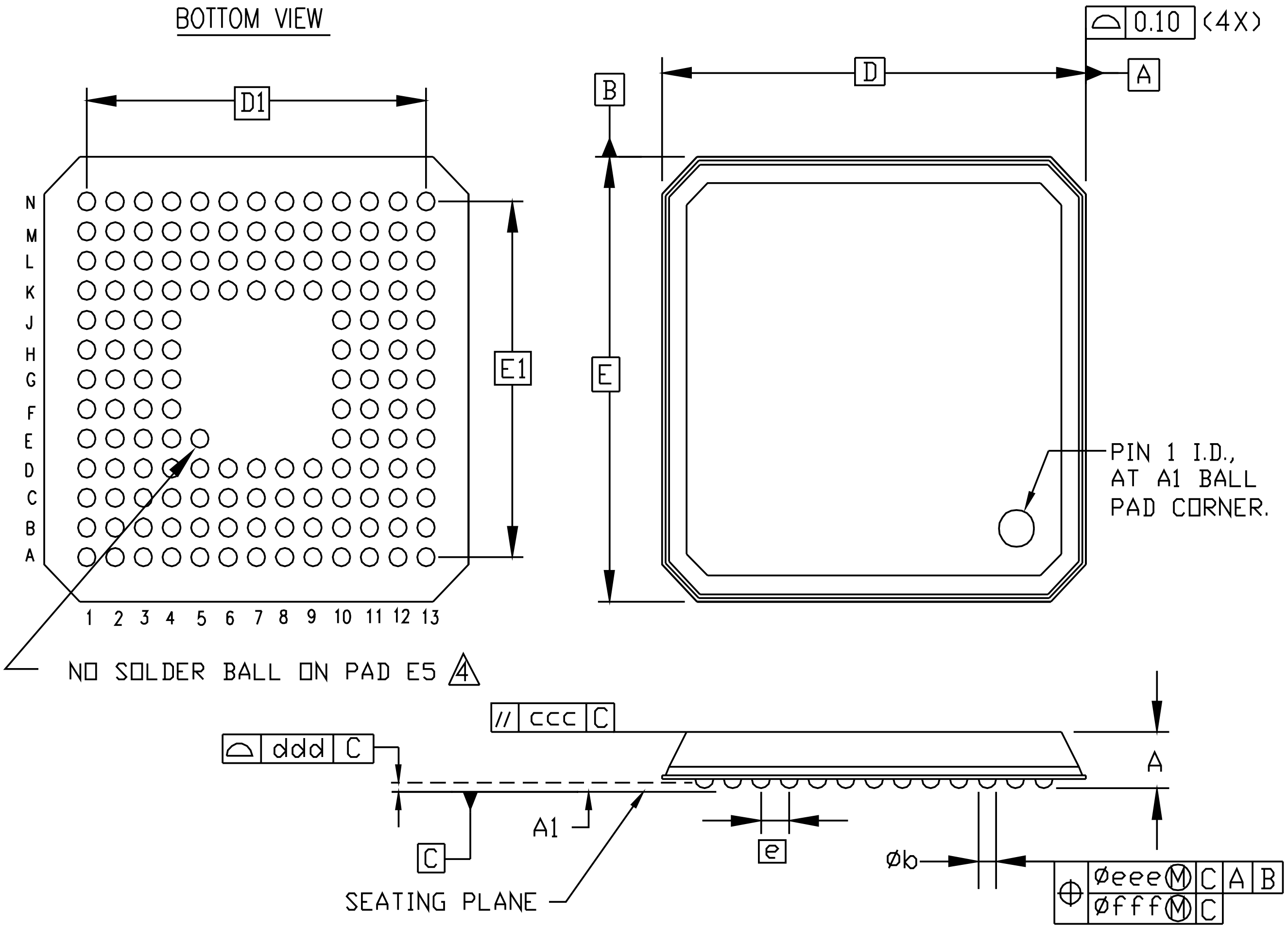
ug002_c4_55_032901

Figure 4-30: **FF896 - FF1152 Pinout Compatibility Diagram**

Package Specifications

This section contains specifications for the following Virtex-II packages:

CS144 Chip-Scale BGA Package (0.80 mm Pitch)



SYMBOL	MILLIMETERS		
	MIN.	NOM.	MAX.
A	—	—	1.20
A_1	0.35	0.40	0.45
D/E	12.00 BSC		
D_1/E_1	9.60 BSC		
e	0.80 BSC		
Øb	0.45	0.50	0.55
ccc	—	—	0.10
ddd	—	—	0.12
eee	—	—	0.15
fff	—	—	0.08
M	13		

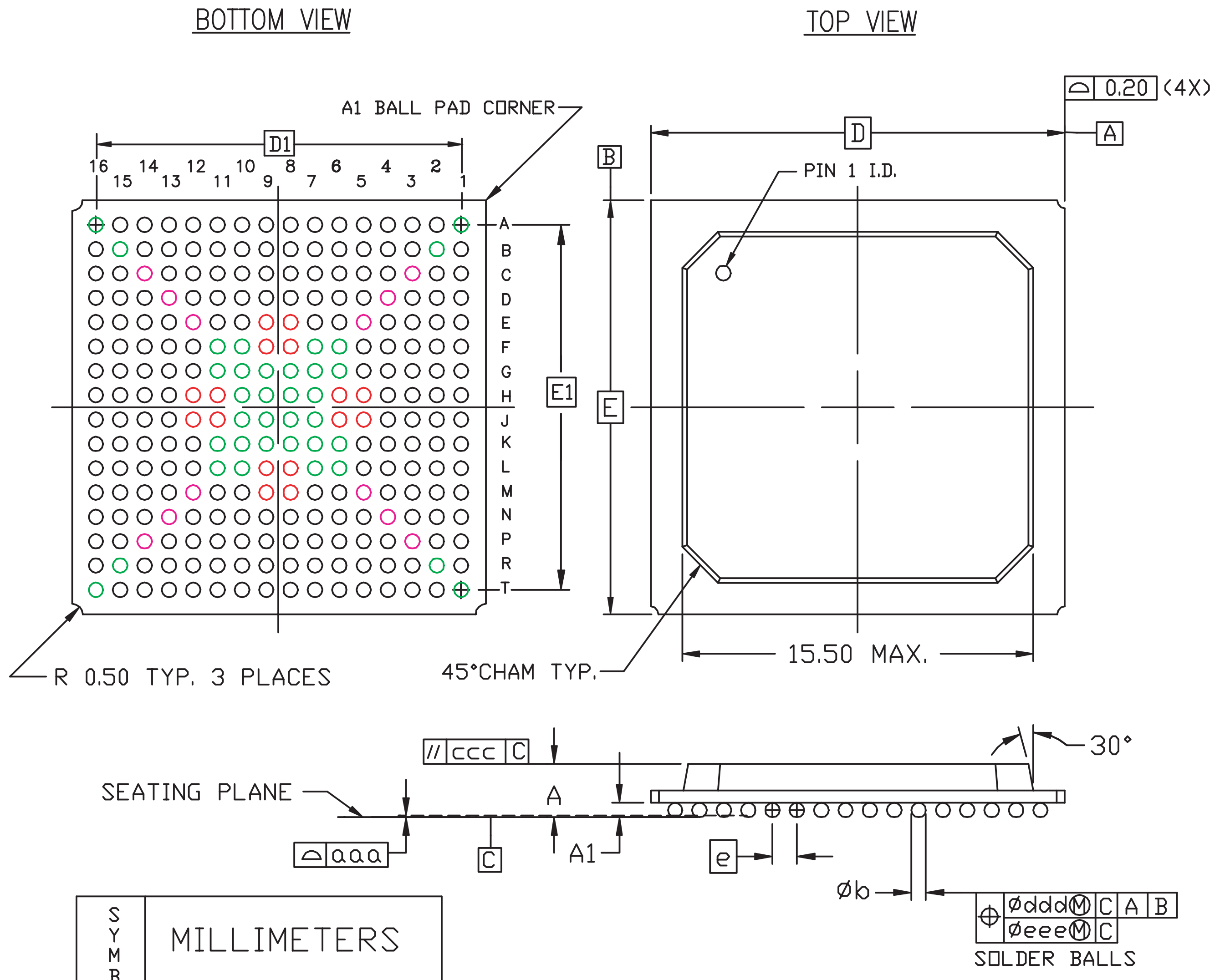
NOTES:

1. ALL DIMENSIONING AND TOLERANCING CONFORM TO ASME Y14.5M-1994
2. SYMBOL "M" IS THE PIN MATRIX SIZE.
3. CONFORMS TO JEDEC MO-205-BE (DEPOPULATED).

4. PAD 'E5' IS FOR PAD 'A1' CORNER INDICATION.

Figure 4-31: **CS144 Chip-Scale BGA Package**

FG256 Fine-Pitch BGA Package (1.00 mm Pitch)



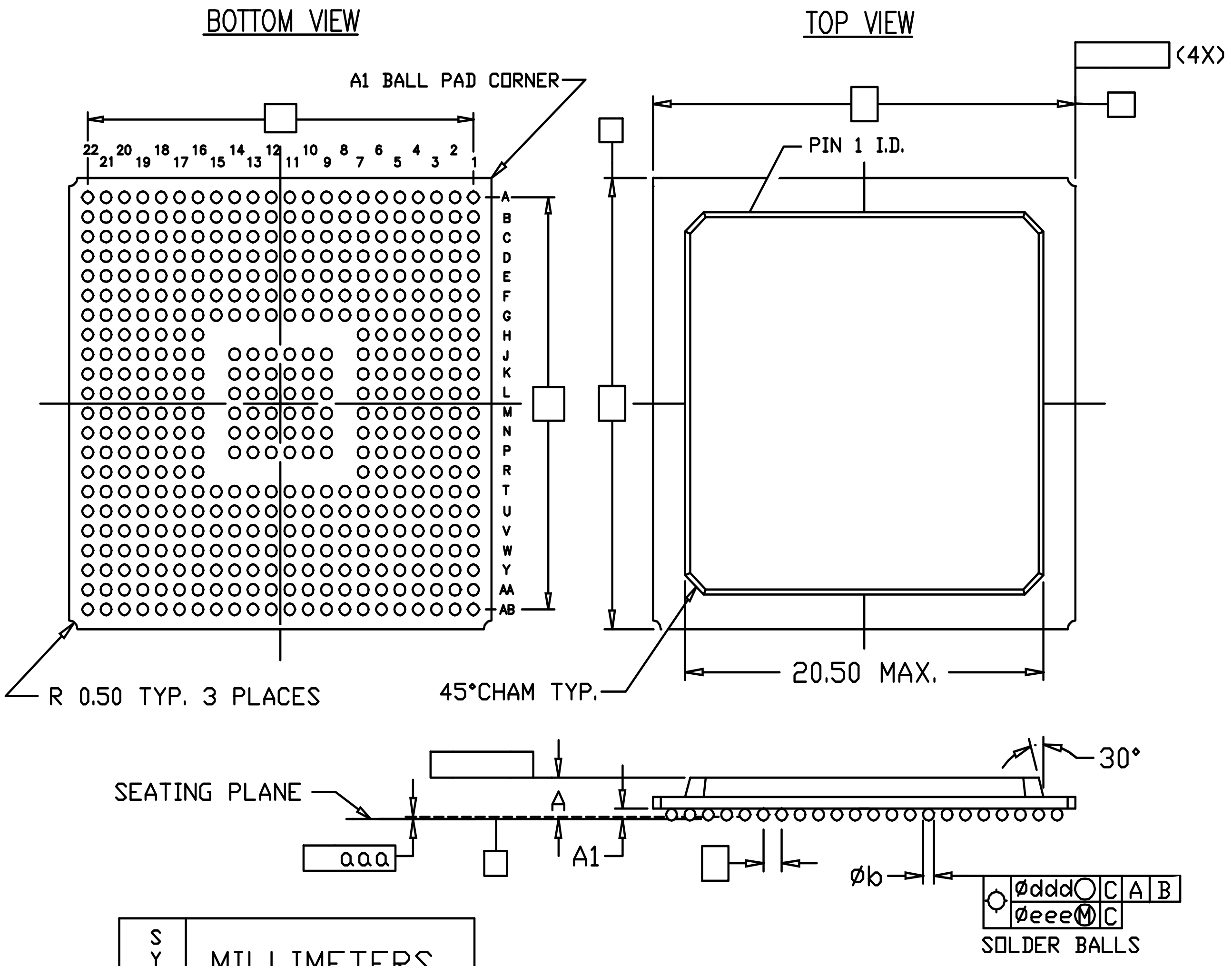
SYMBOL	MILLIMETERS		
	MIN.	NOM.	MAX.
A	~	1.73	2.00
A_1	0.40	0.50	0.60
D/E	17.00 BSC		
D_1/E_1	15.00 REF		
e	1.00 BSC		
Øb	0.50	0.60	0.70
aaa	~	~	0.20
ccc	~	~	0.35
ddd	~	~	0.30
eee	~	~	0.10
M	16		

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M–1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MO–151 AAF–1.

Figure 4-32: **FG256 Fine-Pitch BGA Package**

FG456 Fine-Pitch BGA Package (1.00 mm Pitch)



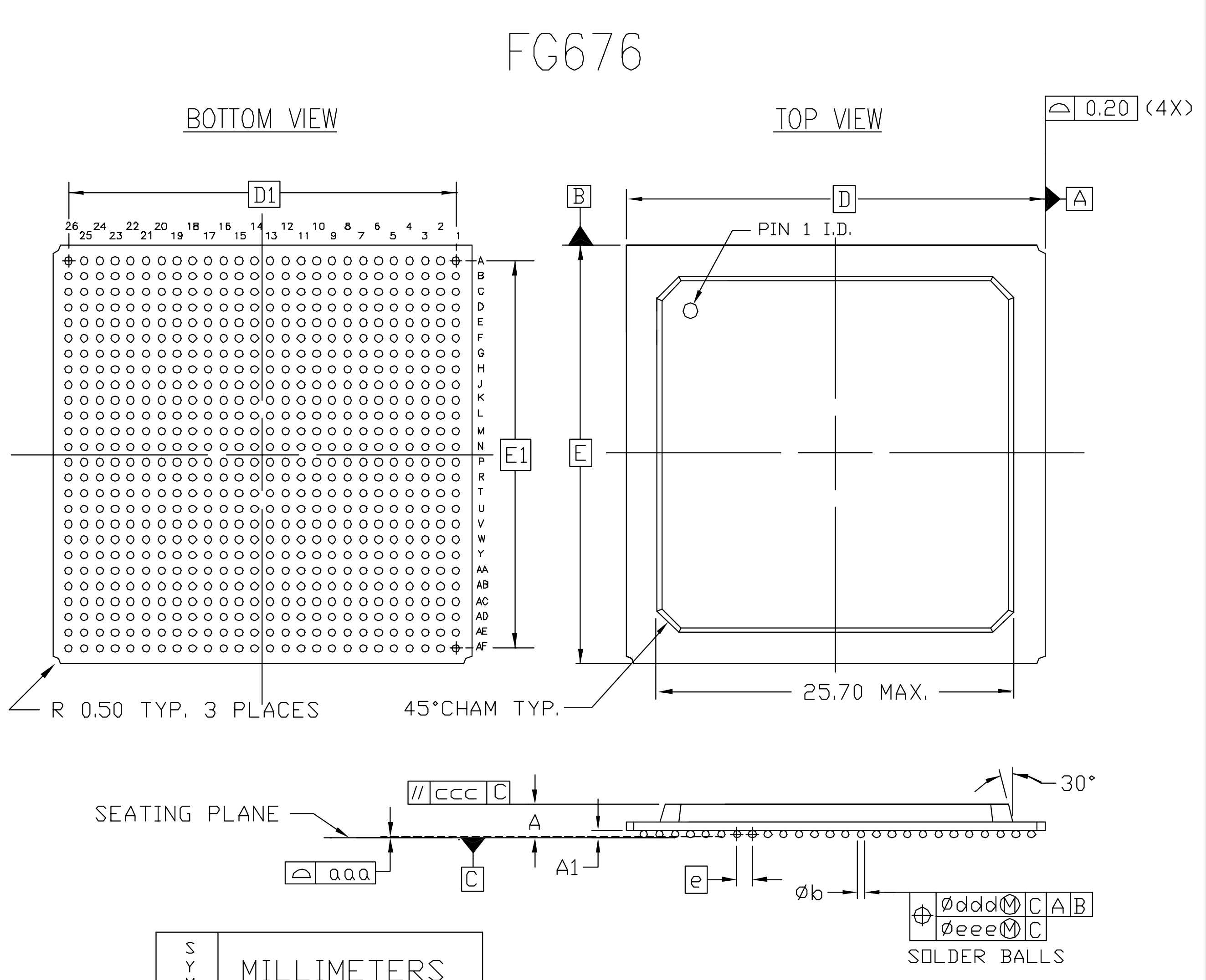
SYMBOL	MILLIMETERS		
	MIN.	NOM.	MAX.
A	~	2.20	2.60
A_1	0.40	0.50	0.60
D/E	23.00 BSC		
D_1/E_1	21.00 REF		
e	1.00 BSC		
Øb	0.50	0.60	0.70
aaa	~	~	0.20
ccc	~	~	0.35
ddd	~	~	0.30
eee	~	~	0.10
M	22		

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M–1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS–034–AAJ–1 (DEPOPULATED)

Figure 4-33: **FG456 Fine-Pitch BGA Package**

FG676 Fine-Pitch BGA Package (1.00 mm Pitch)



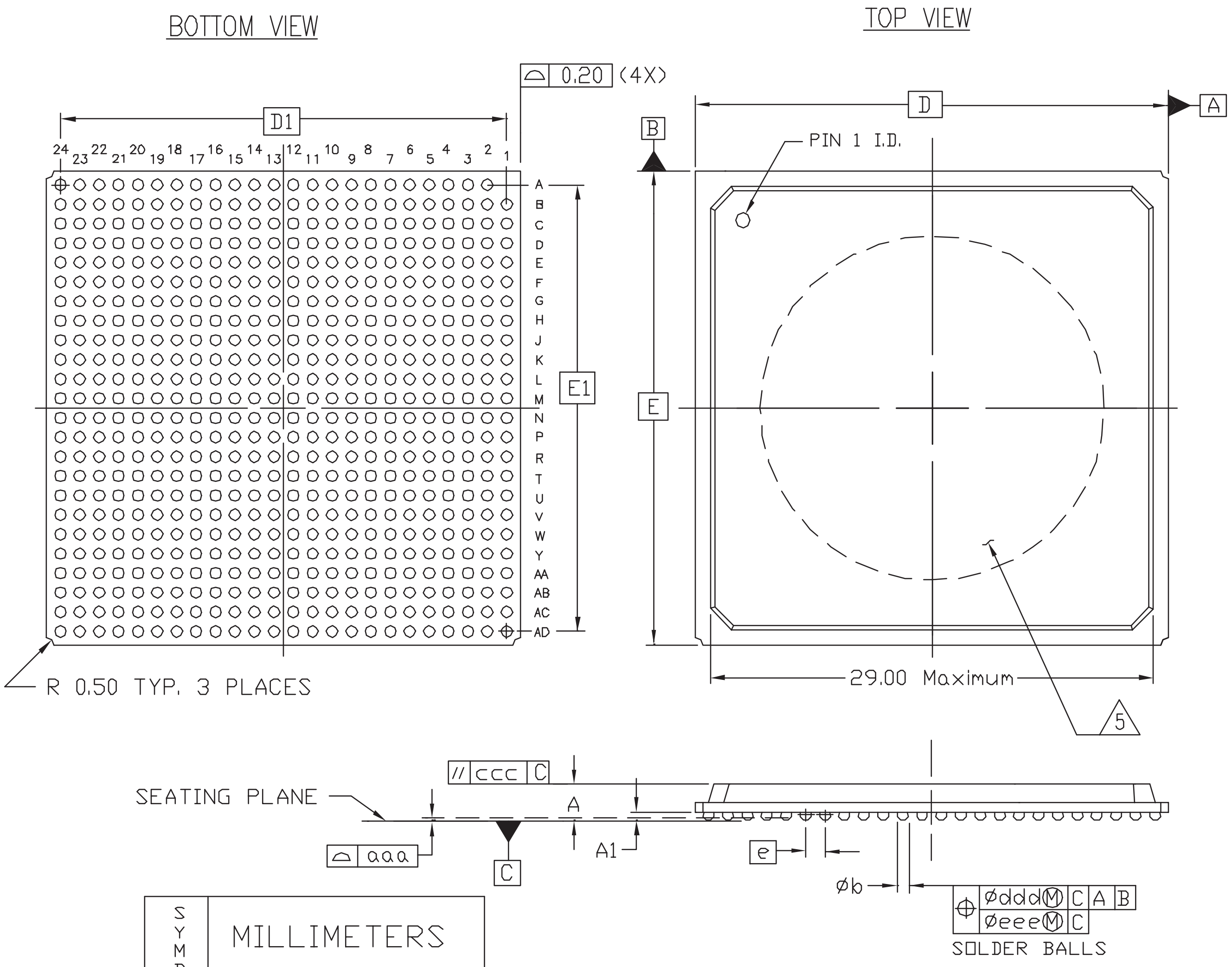
SYMBOL	MILLIMETERS		
	MIN.	NOM.	MAX.
A	—	2.25	2.60
A_1	0.40	0.50	0.60
D/E	27.00 BSC		
D_1/E_1	25.00 REF		
e	1.00 BSC		
Øb	0.50	0.60	0.70
aaa	—	—	0.20
ccc	—	—	0.35
ddd	—	—	0.30
eee	—	—	0.10
M	26		

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS-034-AAL-1

Figure 4-34: **FG676 Fine-Pitch BGA Package**

BG575 Standard BGA Package (1.27 mm Pitch)



SYMBOL	MILLIMETERS		
	MIN.	NOM.	MAX.
A	—	2.33	2.60
A_1	0.50	0.60	0.70
D/E	31.00 BSC		
D_1/E_1	29.21 REF		
e	1.27BSC		
Øb	0.60	0.75	0.90
aaa	—	—	0.20
ccc	—	—	0.35
ddd	—	—	0.30
eee	—	—	0.15
M	24		

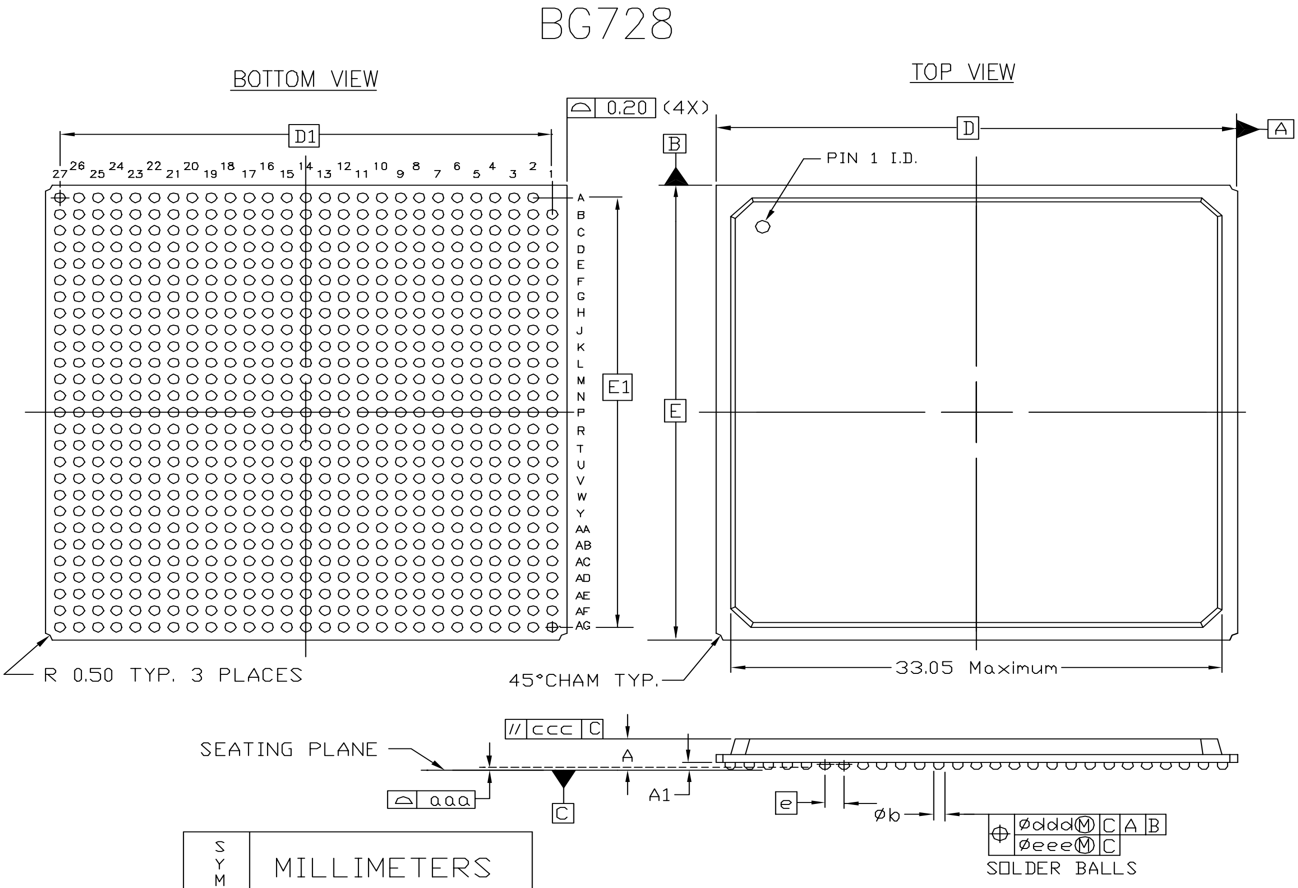
NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS-034 BAN-1 (DEPOPULATED)
4. NO SOLDER BALL AND LAND ON A1.

⚠5 HEAT SINK OPTION

Figure 4-35: **BG575 Standard BGA Package**

BG728 Standard BGA Package (1.27 mm Pitch)



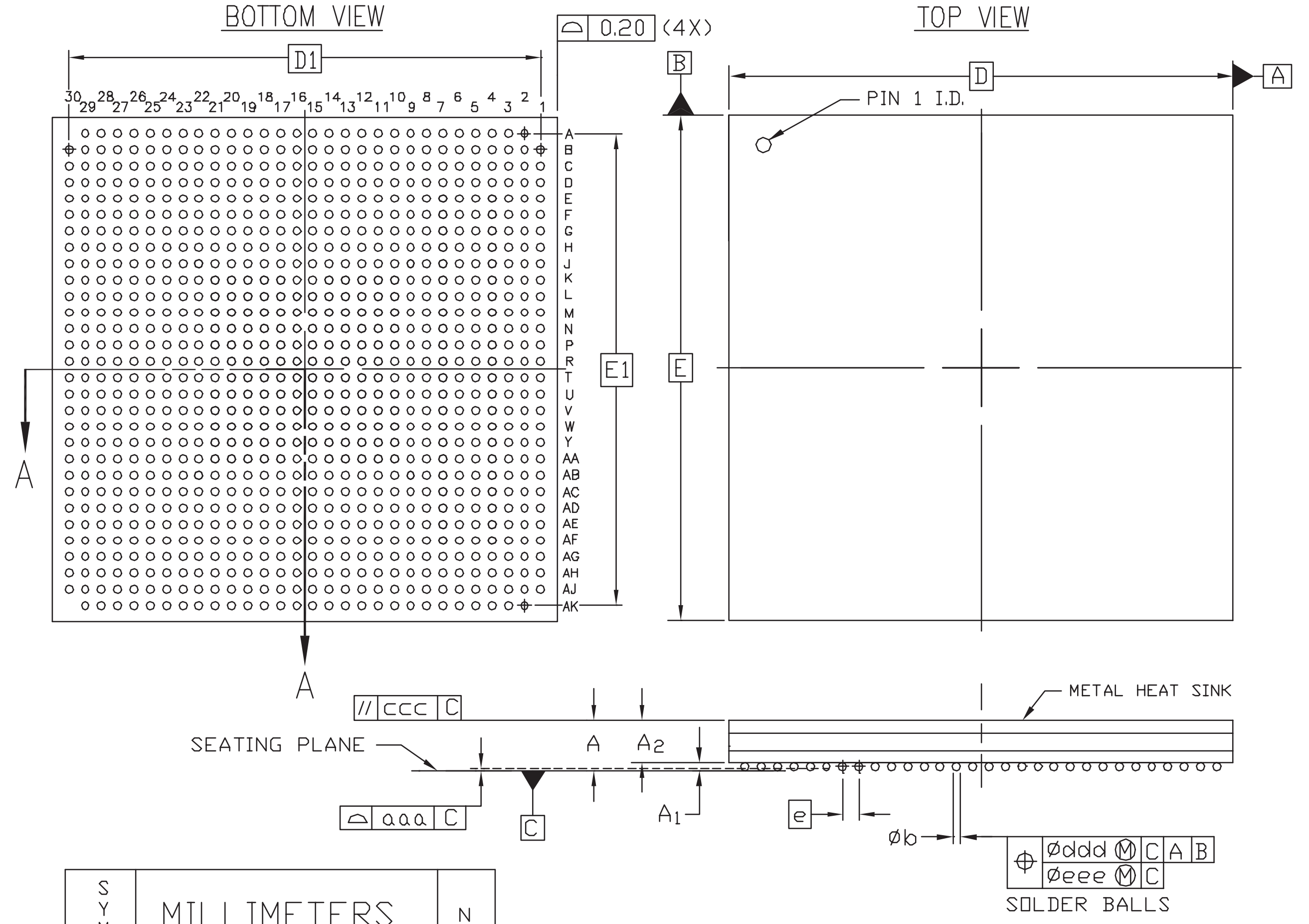
SYMBOL	MILLIMETERS		
	MIN.	NOM.	MAX.
A	—	2.33	2.60
A1	0.50	0.60	0.70
D/E	35.00 BSC		
D1/E1	33.02 REF		
e	1.27BSC		
ϕb	0.60	0.75	0.90
aaa	—	—	0.20
ccc	—	—	0.35
ddd	—	—	0.30
eee	—	—	0.15
M	27		

NOTES:

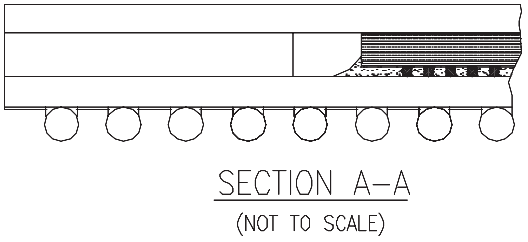
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS-034-BAR-1
4. NO SOLDER BALL AND LAND ON A1.

Figure 4-36: **BG728 Standard BGA Package**

FF896 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)



S Y M B O L	MILLIMETERS			N O T E
	MIN.	NOM.	MAX.	
A	-	3.20	3.40	
A_1	0.40	0.50	0.60	
A_2	-	-	2.80	
D/E	31.00 BASIC			
D_1/E_1	29.00 REF			
e	1.00 BASIC			
Øb	0.50	0.60	0.70	
aaa	-	-	0.20	
ccc	-	-	0.35	
ddd	-	-	0.30	
eee	-	-	0.10	
M	30			2

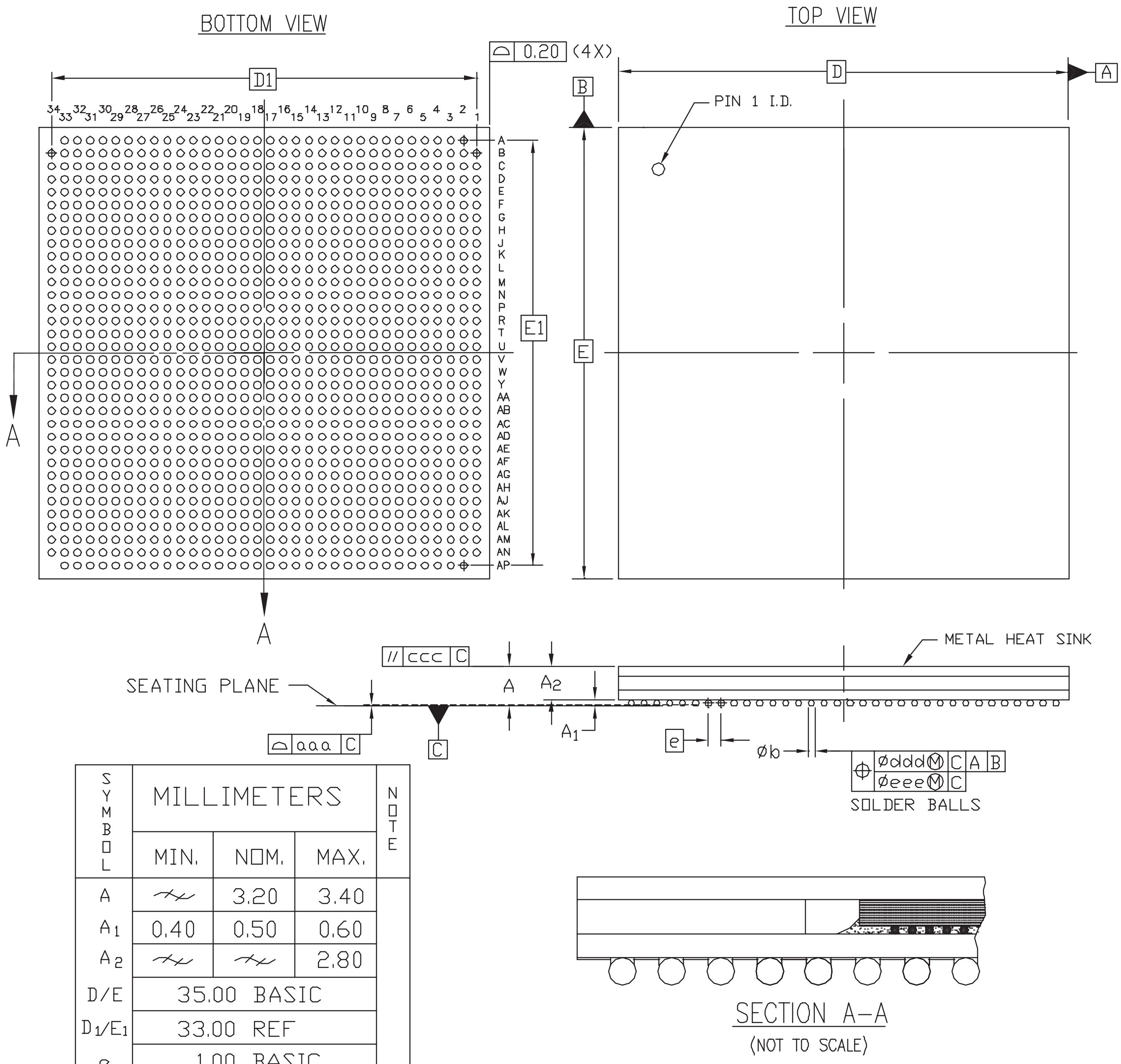


NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M–1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS–034–AAN–1 (DEPOPULATED)

Figure 4-37: **FF896 Flip-Chip Fine-Pitch BGA Package**

FF1152 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)



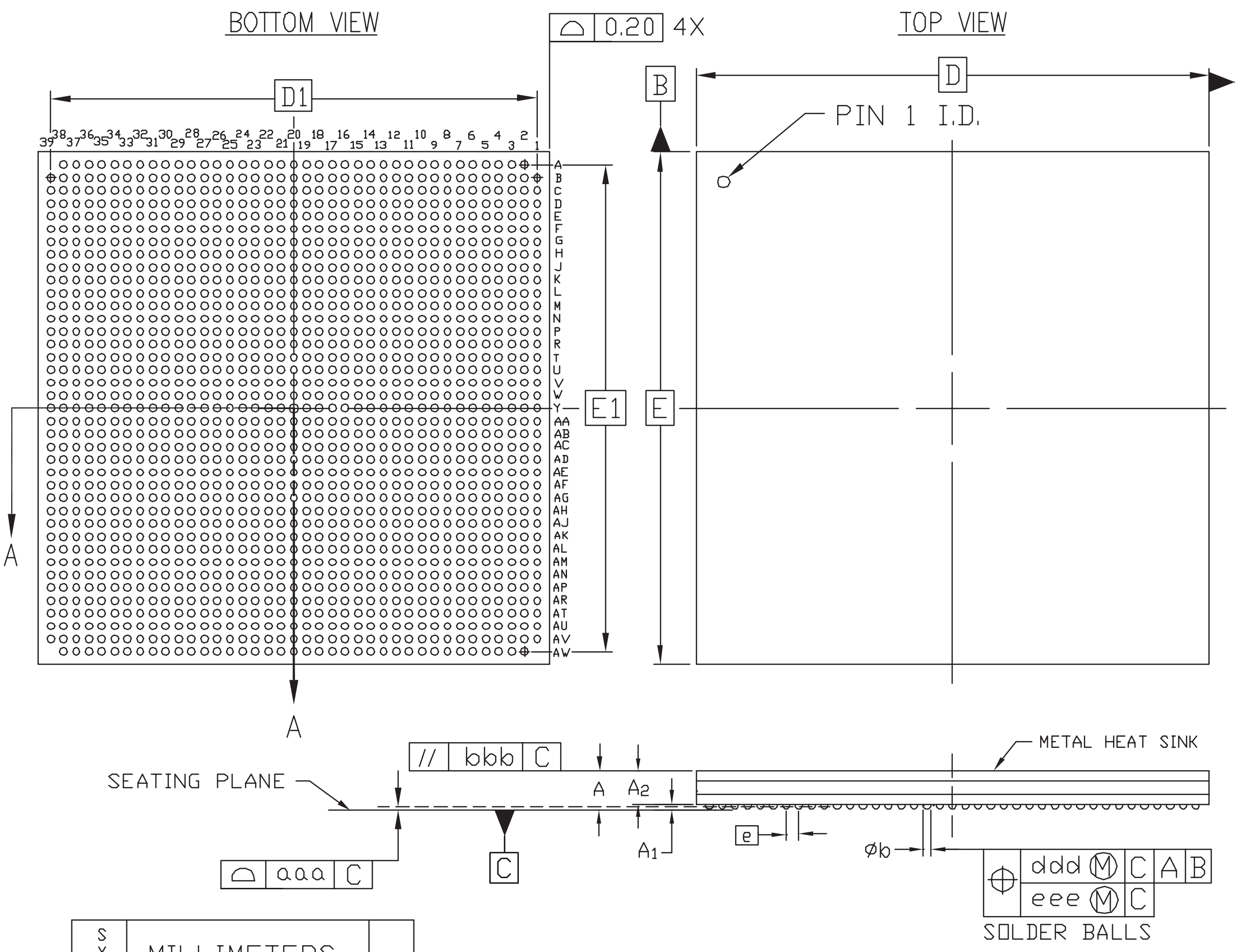
SYMBOL	MILLIMETERS			NOTE
	MIN.	NOM.	MAX.	
A	~	3.20	3.40	
A_1	0.40	0.50	0.60	
A_2	~	~	2.80	
D/E	35.00 BASIC			
D_1/E_1	33.00 REF			
e	1.00 BASIC			
Øb	0.50	0.60	0.70	
aaa	~	~	0.20	
ccc	~	~	0.35	
ddd	~	~	0.30	
eee	~	~	0.10	
M	34			2

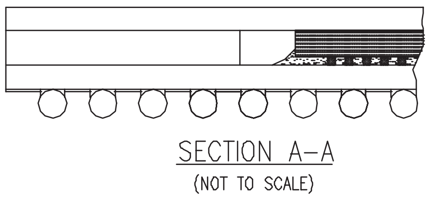
NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M–1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS–034–AAR–1 (DEPOPULATED)

Figure 4-38: **FF1152 Flip-Chip Fine-Pitch BGA Package**

FF1517 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)





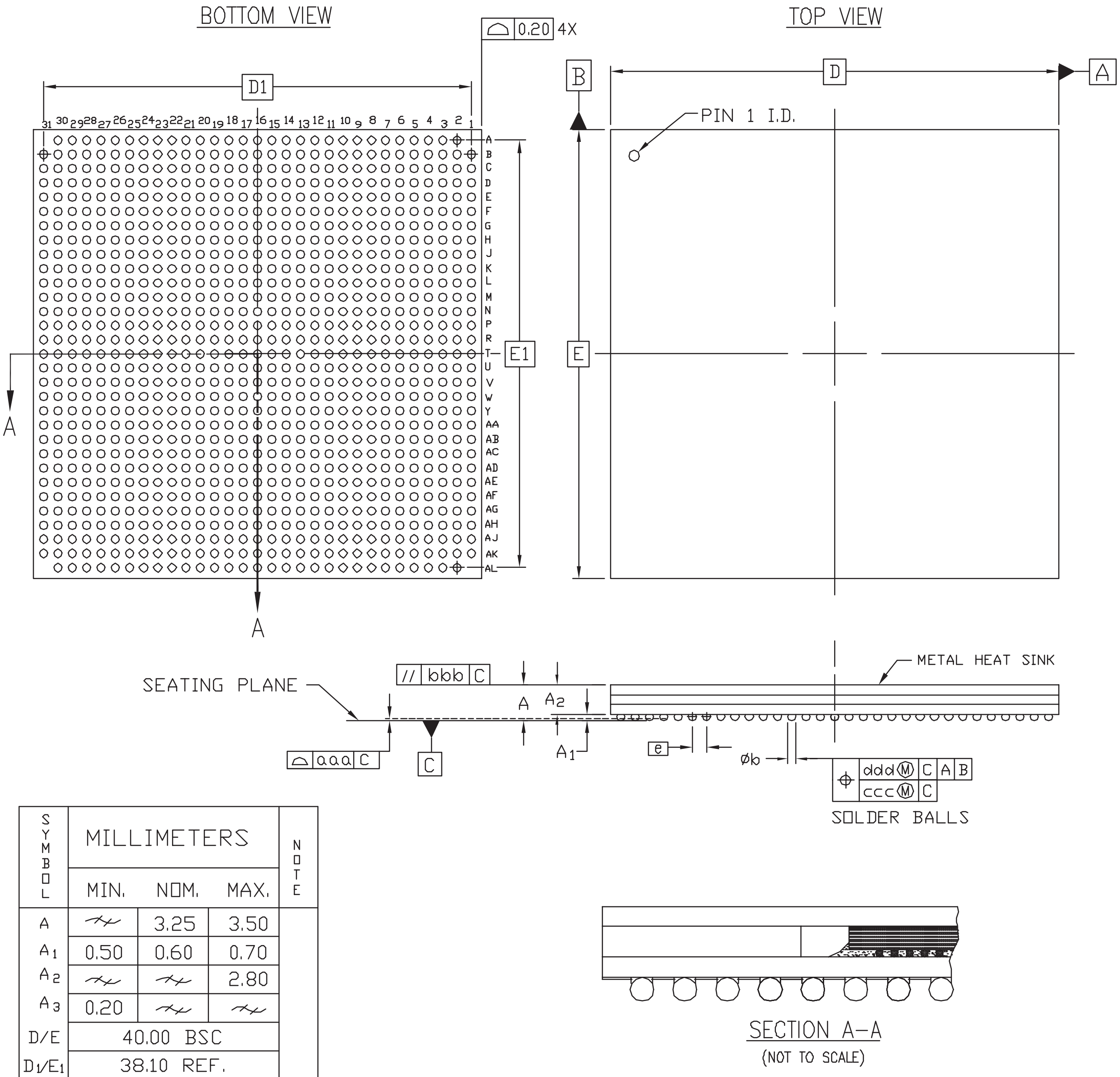
SYMBOL	MILLIMETERS			NOTE
	MIN.	NOM.	MAX.	
A	~	3.20	3.40	
A_1	0.40	0.50	0.60	
A_2	~	~	2.80	
D/E	40.00 BASIC			
D_1/E_1	38.00 REF			
e	1.00 BASIC			
∅b	0.50	0.60	0.70	
aaa	~	~	0.20	
bbb	~	~	0.25	
ddd	~	~	0.25	
eee	~	~	0.10	
M		39		2

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ASME Y14.5M-1994
2. SYMBOL "M" IS THE PIN MATRIX SIZE.
3. CONFORMS TO JEDEC MS-034-AAU-1 (DEPOPULATED)

Figure 4-39: **FF1517 Flip-Chip Fine-Pitch BGA Package**

BF957 Flip-Chip BGA Package (1.27 mm Pitch)



SYMBOL	MILLIMETERS			NOTE
	MIN.	NOM.	MAX.	
A	—	3.25	3.50	
A_1	0.50	0.60	0.70	
A_2	—	—	2.80	
A_3	0.20	—	—	
D/E	40.00 BSC			
D_1/E_1	38.10 REF.			
e	1.27 BSC			
Øb	0.60	0.75	0.90	
aaa	—	—	0.20	
bbb	—	—	0.25	
ccc	—	—	0.15	
ddd	—	—	0.30	
M	31			

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M–1994
2. SYMBOL "M" IS THE PIN MATRIX SIZE.
3. CONFORMS TO JEDEC MS–034–BAU–1 (DEPOPULATED)

Figure 4-40: **BF957 Flip-Chip BGA Package**

Flip-Chip Packages

As silicon devices become more integrated with smaller feature sizes as well as increased functionality and performance, packaging technology is also evolving to take advantage of these silicon advancements. Flip-chip packaging is the latest packaging option introduced by Xilinx to meet the demand for high I/O count and high performance required by today's advanced applications.

Flip-chip packaging interconnect technology replaces peripheral bond pads of traditional wire-bond interconnect technology with area array interconnect at the die/substrate interface.

The area array pads contain wettable metallization for solders (either eutectic or high-lead), where a controlled amount of solder is deposited either by plating or screen-printing. These parts are then reflowed to yield bumped dies with relatively uniform solder bumps spread over the surface of the device. Unlike traditional packaging in which the die is attached to the substrate face up and the connection is made by using wire, the bumped die in a flip-chip package is flipped over and placed face down, with the conductive bumps connecting directly to the matching metal pads on the ceramic or organic laminate substrate. The solder material at molten stage is self-aligning and produces good joints even if the chip is placed offset on the substrate.

Flip-chip packages are assembled on high-density, multi-layer ceramic or organic laminate substrates. Since flip-chip bump pads are in area array configuration, very fine lines and geometry on the substrates are required to be able to successfully route the signals from the die to the periphery of the substrates. Multi-layer build-up structures offer this layout flexibility on flip-chip packages, and they provide improvements in power distribution and signal transmission characteristics.

Advantages of Flip-Chip Technology

Flip-chip interconnections in combination with the advanced multi-layer laminated substrates provide superior performance over traditional wire-bond packaging. Benefits include:

- Easy access to core power/ground and shorter interconnects, resulting in better electrical performance
- Better noise control since the inductance of flip-chip interconnect is lower
- Excellent thermal performance due to direct heatsinking to backside of the die
- Higher I/O density since bond pads are in area array format
- Smaller size

Thermal Data

Thermal Considerations

Due to the variety of applications in which Virtex-II FPGA devices are likely to be used, it is traditionally a challenge to predict the power requirements, and thus the thermal management needs, of a particular application. Virtex-II devices in general are characterized by high I/O counts and very high user gate counts. The attributes that make the devices popular with users also give the devices the potential of being clocked fast, which results in high power consumption. Because of this high heat-generating potential, the Virtex-II package offering (see Table 4-3) includes medium and high power capable packaging options.

Table 4-3 shows thermal resistance parameters for Virtex-II packages. These include: Junction-to-ambient, Junction-to-case and Junction-to-board. Estimated power consumption capability is given, as well. These values were derived with some typical thermal management assumptions, stated in the table.

Table 4-3: **Thermal Data for Virtex-II Packages**

Package	Lead Pitch (mm)	Junction to Ambient Theta-J_A Range °C/Watt in Air	Junction to Case Theta-J_C Typical °C/Watt	Junction to Board Psi-J_B ("Theta-J_B") Typical °C/Watt	Max Power Bare Pkg (Watts) T_A = 50 °C T_{JMAX} = 100 °C	Power With Heatsink (Watts) Theta-SA = 1.5 °C/Watt Theta-cs = 0.1 °C/Watt T_A = 50° C T_J = 100° C
CS144 Flex Based 12x12	0.8	32 - 36	1	20	1.5	
FG256 2- 4L PCB 17x17	1.0	30 -35	3.5	19	1.5	
FG456 4L PCB 23x23	1.0	15 - 28	2.0	11	2.4	
FG676 4L PCB 27x27	1.0	14 -22	1.8	9	2.8	15
BG575 4L PCB 31x31	1.27	13 - 20	1.6	7	3.1	16
BG728 4L PCB 35x35	1.27	12 -20	1.5	6	3.3	16
BF957 40x40 Flip-Chip	1.27	8 - 13	0.7	3	5.0	22
FF896 31x31 Flip-Chip	1.0	9 - 14	0.8	4	4.5	21
FF1152 35x35 Flip-Chip	1.0	8 - 13	0.8	4	4.5	21
FF1517 40x40 Flip-Chip	1.0	8 - 12	0.7	3	5.0	22

Virtex-II packages can be grouped into three broad performance categories: low, medium, and high, based on their power handling capabilities. All of the packages can use external thermal enhancements, which can range from simple airflow to schemes that can include passive as well as active heatsinks. This is particularly true for high-performance flip-chip packages where system designers have the option to further enhance the packages to handle in excess of 25 watts, with arrangements that take system physical constraints into consideration. Table 4-4 shows simple but incremental power management schemes that can be brought to bear on flip-chip packages.

Table 4-4: **Virtex-II Flip-Chip Thermal Management**

Power	Technique	Description
Low End (1 - 6 watts)	Bare package with moderate air 8 - 12 **°C/Watt**	Bare package. Package can be used with moderate airflow within a system.
Mid Range (4 - 10 watts)	Passive heatsink with air 5 - 10 **°C/Watt**	Package is used with various forms of passive heatsinks and heat spreader techniques.
High End (8 - 25 watts)	Active heatsink 2 - 3 **°C/Watt or better**	Package is used with active heatsinks, TEC, and board-level heat spreader techniques

Thermal Management Options

The following are thermal management options to consider:

- For moderate power dissipation (2 to 6 watts), the use of passive heatsinks and heatspreaders attached with thermally conductive double-sided tapes or retainers can offer quick thermal solutions.
- The use of lightweight finned external passive heatsinks can be effective for dissipating up to 10 watts. If implemented with forced air as well, the benefit can be a 40% to 50% increase in heat handling efficiency over bare packages. The more efficient external heatsinks tend to be tall and heavy. To help protect component joints from bulky heatsink induced stresses, the use of spring loaded pins or clips that transfer the mounting stress to a circuit board is advisable. The diagonals of some of these heatsinks can be designed with extensions to allow direct connections to the board.
- Flip-chip packages: All flip-chip packages are thermally enhanced BGAs with die facing down. They are offered with exposed metal heatsink at the top. These high-end thermal packages lend themselves to the application of external heatsinks (passive or active) for further heat removal efficiency. Again, precaution should be taken to prevent component damage when a bulky heatsink is attached.
- Active heatsinks can include a simple heatsink incorporating a mini fan or even a Peltier Thermoelectric Cooler (TECs) with a fan to blow away any heat generated. Any considerations to apply TEC in heat management should require consultation with experts in using the device, since these devices can be reversed and cause damage to the components. Also, condensation can be an issue.
- Molded packages (FG456, FG676, BG575, BG728, and so forth) with or without exposed metal at the top can also use heatsinks at the top for further heat removal. These BGA packages are similar in construction to those used in Graphics cards in PC applications, and heatsinks used for those applications can easily be used for these packages, as well. In this case, the Junction-to-Case resistance is the limiting consideration.
- Outside the package itself, the board on which the package sits can have a significant impact on thermal performance. Board designs can be implemented to take advantage of a board's ability to spread heat. The effect of the board is dependent on its size and how it conducts heat. Board size, the level of copper traces on it, and the number of buried copper planes all lower the junction-to-ambient thermal resistance for packages mounted on the board.

  The junction-to-board thermal resistance for Virtex-II packages are given in Table 4-3. A standard JEDEC type board was used for obtaining the data. Users need to be aware that a direct heat path to the board from a component also exposes the component to the effect of other heat sources - particularly if the board is not cooled effectively. An otherwise cooler component might be heated by other heat contributing components on the board.

Printed Circuit Board Considerations

Layout Considerations

The PC board is no longer just a means to hold ICs in place. At today's high clock rates and fast signal transitions, the PC board performs a vital function in feeding stable supply voltages to the IC and in maintaining signal integrity between devices.

VCC and Ground Planes

Since CMOS power consumption is dynamic, it is a non-trivial task to assure stable supply voltages at the device pins and to minimize ground differentials. A multi-layer PC board is a must, with four layers for the simplest circuits, 6 to 12 layers for typical boards. Ground and V_{CC} must each be distributed in complete layers with few holes. Slots in these layers would cause an unacceptable inductive voltage drop, when the supply current changes at a rate of 1 A/ns, or even faster. Besides an uninterrupted ground plane, Virtex-II devices require one plane for V_{CCINT} (1.5 V) plus one plane for V_{CCAUX} (3.3 V). V_{CCO} can be distributed on wide signal traces with sufficient bypass capacitors.

Beyond low resistance and inductance, ground and V_{CC} planes combined can also provide a small degree of V_{CC} decoupling. The capacitance between two planes is ~100 pF/inch^2 or ~15 pF/cm^2, assuming 10 mil (0.25 mm) spacing with FR4 epoxy.

V_{CC} Decoupling

Fast changing Icc transitions must be supplied by local decoupling capacitors, placed very closely to the V_{CC} device pins or balls. These capacitors must have sufficient capacitance to supply Icc for a few ns and must have low intrinsic resistance and inductance. X7R or NPO ceramic surface-mounted capacitors of 0.01 to 0.1 µF, one per V_{CC} device pin, are appropriate. 0.1 µF can supply 1A for 2ns with a 20 mV voltage droop.

$$1\text{A} \bullet 2\text{ns} = 2 \text{ nanocoulomb} = 100 \text{ nF} \bullet 0.02 \text{ V}$$

Low impedance at >100 MHz is important, but capacitance variation with temperature is acceptable. These small capacitors are the first-line source for Icc, and they must be placed very close to the V_{CC} pins. A half-inch or 10 mm trace represents an inductance of several nanohenries, defeating the purpose of the decoupling capacitor. Backing up this local decoupling is one tantalum capacitor of 10 to 100 µF, able to supply multiple amperes for about 100 ns.

Finally, each board needs a power-supply decoupling electrolytic capacitor of 1000 to 10,000 µF able to supply even more current for a portion of the supply switching period. As described below, larger capacitors inevitably have higher series resistance and inductance, which is the reason for the above-mentioned hierarchy of supply decoupling. As a general rule, multiple capacitors in parallel always offer lower resistance and inductance than any single capacitor.

Decoupling Capacitors

The ideal decoupling capacitor would present a short circuit to ground for all ac signals. A real capacitor combines a given amount of capacitance with unavoidable parasitics, a small series resistance and inductance. At low frequencies, the composite impedance is capacitive, i.e., it decreases with increasing frequency. At high frequencies, it is inductive and increases with frequency, making the decoupling ineffective. In-between, there is the LC resonant frequency, where the capacitor looks like a small resistor.

Different technologies provide different trade-offs between desirable features like small size and high capacitance, and undesirable features like series resistance and inductance. Electrolytic and tantalum capacitors offer the largest capacitance in a given physical size, but also have the highest inductance. This makes them useful for decoupling low frequencies and storing large amounts of charge, but useless for high frequency decoupling. Surface-mount ceramic capacitors, on the other hand, offer the lowest inductance and the best high-frequency performance, but offer only a small amount of capacitance, less than a microfarad.

Figure 4-41 shows the frequency-dependent impedance and resistance of a typical electrolytic capacitor of 1500 µF, while Figure 4-42 and Figure 4-43 show the equivalent data for ceramic bypass capacitors of 33,000 and 3,300 pF, respectively. Note that the resonant frequency for the small ceramic bypass capacitor at 100 MHz is 10,000 times higher than the resonance frequency of the large electrolytic capacitor at 10 KHz. For more technical information on decoupling capacitors, see the manufacturers' websites.

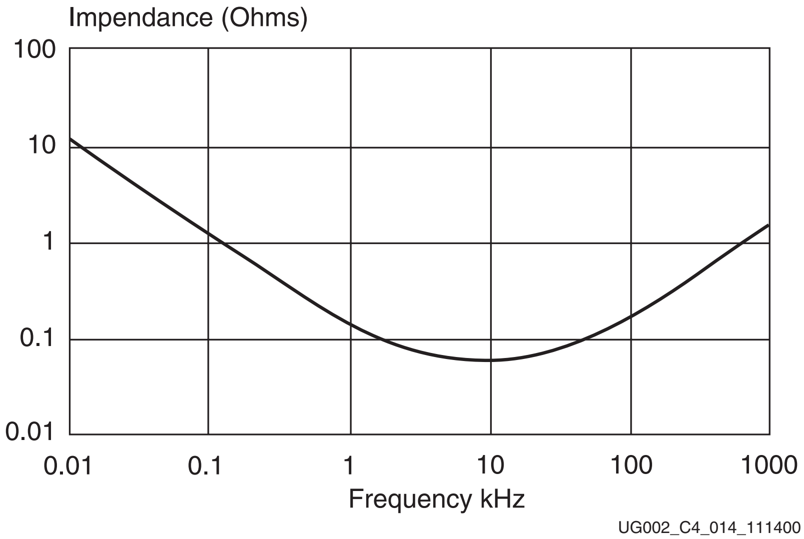


Figure 4-41: **1500 µF Electrolytic Capacitor Frequency Response Curve**

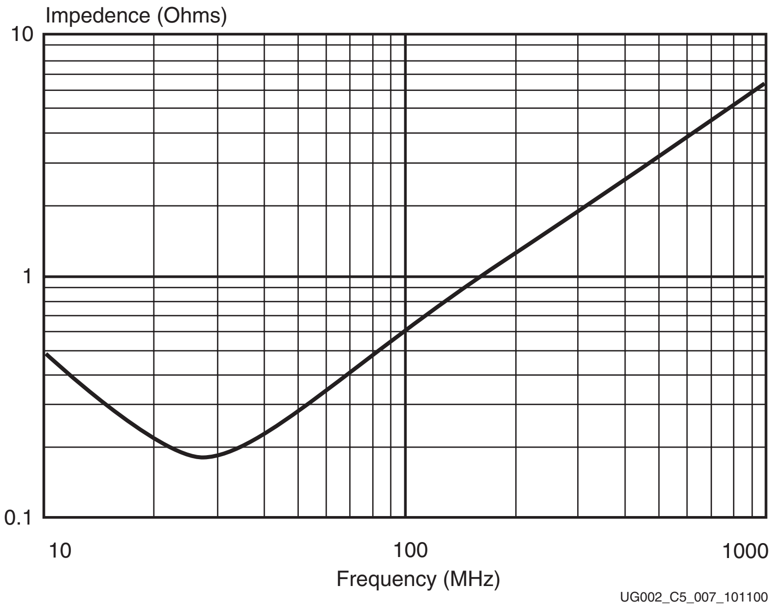


Figure 4-42: **33000 pF X7R Component Frequency Response Curve**

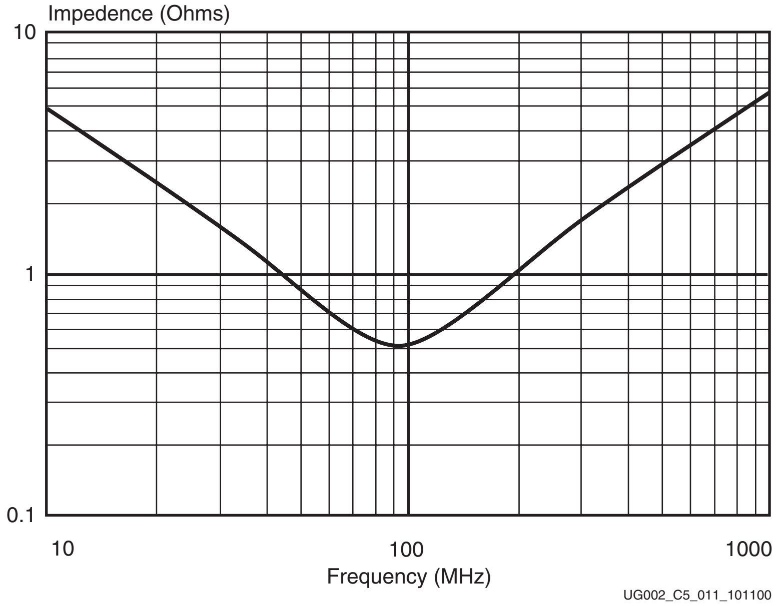


Figure 4-43: **3300 pF X7R Component Frequency Response Curve**

Transmission Line Reflections and Terminations

A PC board trace must be analyzed as a transmission line. Its series resistance and parallel conductance can generally be ignored, but series inductance and parallel capacitance per unit length are important parameters. Any signal transition (rising or falling edge) travels along the trace at a speed determined by the incremental inductance and capacitance.

For an outer-layer trace (air on one side) the propagation delay is 140 ps/inch, or 55 ps/cm. For an inner-layer trace (FR4 with ε=4.5 on both sides), the propagation delay is 180 ps/inch, or 70 ps/cm.

The voltage-to-current ratio at any point along the transmission line is called the characteristic impedance Z_0. It is determined by w/d, the ratio of trace width w to the distance d above the ground or V_{CC} plane.

For an outer layer trace (microstrip),

Z_0=50 Ω when w = 2d (e.g., w = 12 mil, d = 6 mil),

Z_0=75 Ω when w = d (e.g., both 6 mil = 0.15 mm).

For an inner layer trace between two ground or V_{CC} planes (stripline),

Z_0=50 Ω when w = 0.6•d (e.g., w = 5 mil, d = 8 mil),

Z_0=75 Ω when w = 0.25•d (impractical).

Most signal traces fall into the range of 40 to 80 Ω.

A slow transition treats a short narrow trace as a lumped capacitance of about 2 pF per inch (0.8 pF per cm). However, if the trace is so long, or the signal transition is so fast that the potential echo from the far end arrives after the end of the transition, then the trace must be analyzed as a transmission line.

In this case, the driver sees the trace not as a lumped capacitance, but rather as a pure resistance of Z_0. The signal transition then travels along the trace at the speed mentioned above. At any trace-impedance discontinuity all or part of the signal is reflected back to the origin. If the far end is resistively terminated with R=Z_0, then there is no reflection. If, however, the end is open, or loaded with only a CMOS input, then the transition doubles in amplitude, and this new wave travels back to the driver, where it may be reflected again, resulting in the familiar ringing. Such ringing has a serious impact on signal integrity, reduces noise margins, and can lead to malfunction, especially if an asynchronous signal or a clock signal crosses the input threshold voltage unpredictably. Two alternate ways to avoid reflections and ensure signal integrity are parallel termination and series termination.

Parallel Termination

Reflections from the far end of the transmission line are avoided if the far end is loaded with a resistor equal to Z_0. A popular variation uses two resistors, one to V_{CC}, one to ground, as the Thevenin equivalent of Z_0. This reduces the load current for one signal level, while increasing it for the other. Parallel termination causes dc power consumption which can be eliminated by inserting a capacitor between the terminating resistor and ground. The value of this capacitor is determined as follows:

Signal transition time << RC << signal level duration

For example, 50 Ω • 120 pF for a 2 ns transition every 20 ns. See Figure 4-44.

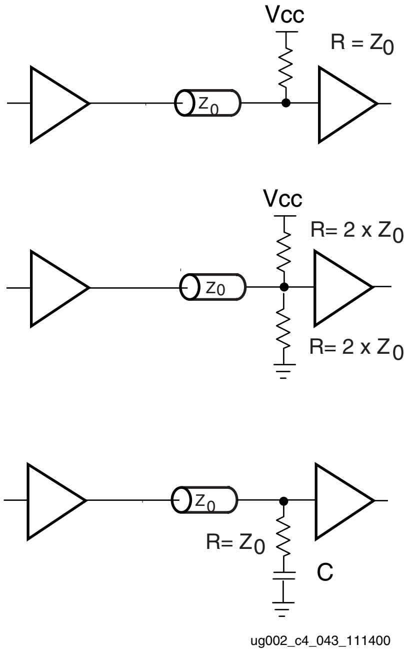


Figure 4-44: **Parallel Termination**

Series Termination

While parallel termination eliminates reflections, series termination relies on the reflection from the far end to achieve a full-amplitude signal. For series termination, the driver impedance is adjusted to equal Z_0, thus driving a half-amplitude signal onto the transmission line. At the unterminated far end, the reflection creates a full-amplitude signal, which then travels back to the driver where it gets absorbed, since the output impedance equals Z_0. See Figure 4-45.

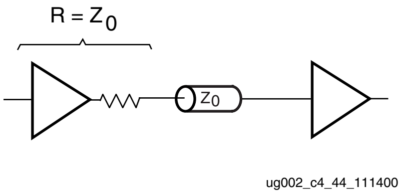


Figure 4-45: **Series Termination**

Series termination dissipates no dc power, but the half-amplitude round-trip delay signal means that there must be no additional loads along the line. Series termination is ideal (and only meaningful) for single-source-single-destination interconnects.

Virtex-II devices offer digitally controlled output impedance drivers and digitally-controlled input termination, thus eliminating the need for any external termination resistors. This feature is extremely valuable with high pin-count, high density packages.

These PC board considerations apply to all modern systems with fast current and voltage transitions, irrespective of the actual clock frequency. The designer of relatively slow systems is more likely caught off-guard by the inherent speed of modern CMOS ICs, where di/dt is measured in A/ns, dV/dt is measured in V/ns, and input flip-flops can react to 1 ns pulses, that are invisible on mid-range oscilloscopes. Powerful tools like HyperLynx can analyze signal integrity on the PC board and can often be amortized by one eliminated board-respin.

JTAG Configuration and Test Signals

Poor signal integrity and limitations of devices in a JTAG scan chain can reduce the maximum JTAG test clock (TCK) rate and reliability of JTAG-based configuration and test procedures. The JTAG TCK and test mode (TMS) signals must be buffered, distributed, and routed with the same care as any clock signal especially for long JTAG scan chains. The devices in a JTAG scan chain should be ordered such that the connections from the TDO of one device to the TDI of the next device are minimized. When high-speed JTAG-based configuration for the Virtex-II devices is required, devices with lower-specified maximum TCK rates can be placed in a separate JTAG scan chain.

Crosstalk

Crosstalk can happen when two signals are routed closely together. Current through one of the traces creates a magnetic field that induces current on the neighboring trace, or the voltage on the trace couples capacitively to its neighbor. Crosstalk can be accurately modeled with signal integrity software, but two easy to remember rules of thumb are:

- Crosstalk falls off with the square of increasing distance between the traces.
- Crosstalk also falls off with the square of decreasing distance to a ground plane.

$$\text{Peak Crosstalk Voltage} = \frac{DV}{1 + (D/H)^2}$$

where

DV is the voltage swing

D is the distance between traces (center to center)

H is the spacing above the ground plane

Example:

3.3V swing, and two stripline traces 50 mils apart and 50 mils above the ground plane.

Peak Crosstalk Voltage = (3.3 V)/(1 + $(0.05/0.05)^2$) = 1.65 V

This can cause a false transition on the neighboring trace. Separating the trace by an additional 50 mils is significantly better:

Peak Crosstalk Voltage = (3.3 V)/(1 + $(0.1/0.05)^2$) = 0.66 V

Signal Routing to and from Package Pins

Signal escaping (traces leaving the pin/ball area) can be quite difficult for the large FG and flip-chip packages. The number of signal layers required to escape all the pins depends on the PCB design rules. The thinner the traces, the more signals per layer can be routed, and the fewer layers are needed. The thinner traces have higher characteristic impedance, so choose an impedance plan that makes sense, and then be consistent. Traces from 40 to 80 ohms are common.

If only one signal can be escaped between two pads, only two rows of pins can be escaped per layer. For FG packages (1.0mm pitch) one signal of width 5 mils (0.13mm) can be escaped between two pads, assuming a space constraint equal to the trace width. For a discussion of signal routing specific to Virtex-II devices, see www.xilinx.com for currently available application notes.

As packages are able to handle more I/Os with a minimum increase in size, the signal integrity of those signals must be considered, regardless of clock frequency. Especially with the largest packages, precise PCB layer stackup is required. Parameters such as board material, trace width, pad type, and stackup must be defined based on simulation, and the fabrication drawings must be marked with "precise layer stackup" and the stackup specified. A number of board-level signal integrity simulators exist, and careful attention to PCB design rules creates a robust design with low EMI and high signal reliability.

Board Routability Guidelines

Board-Level BGA Routing Challenges

Xilinx ball grid array (BGA) wire-bond and flip-chip packages contain a matrix of solder balls (see Figure 4-46). These packages are made of multilayer BT substrates. Signal balls are in a perimeter format. Power and ground pins are grouped together appropriately.

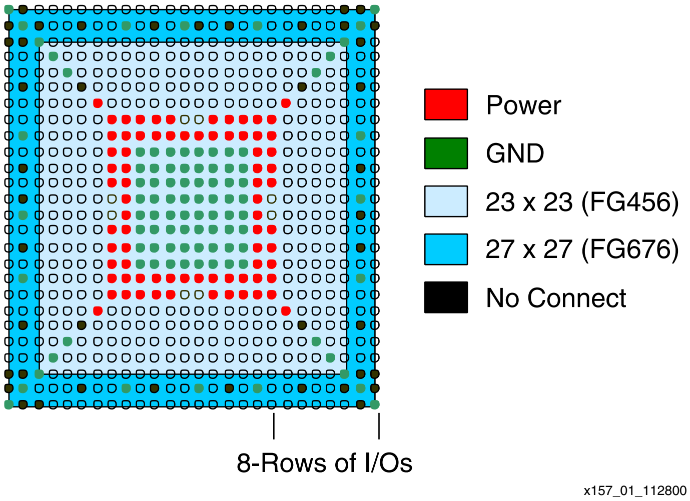


Figure 4-46: **Fine-Pitch BGA Pin Assignments**

The number of layers required for effective routing of these packages is dictated by the layout of pins in each package. If several other technologies and components are already present on the board, the system cost is factored with every added board layer. The intent of a board designer is to optimize the number of layers required to route these packages, considering both cost and performance. This section provides guidelines for minimizing required board layers for routing BGA products using standard PCB technologies (5 mils-wide lines and spaces or 6 mils-wide lines and spaces).

For high performance and other system needs, designers can use premium technologies with finer lines/spaces on the board. The pin assignment and pin grouping scheme in BGA packages enables efficient routing of the board with an optimum number of required board layers.

Board Routing Strategy

The diameter of a land pad on the component side is provided by Xilinx. This information is required prior to the start of board layout when designing the board pads to match component-side land geometry. Typical values for these land pads are described in Figure 4-47 and summarized in Table 4-5.

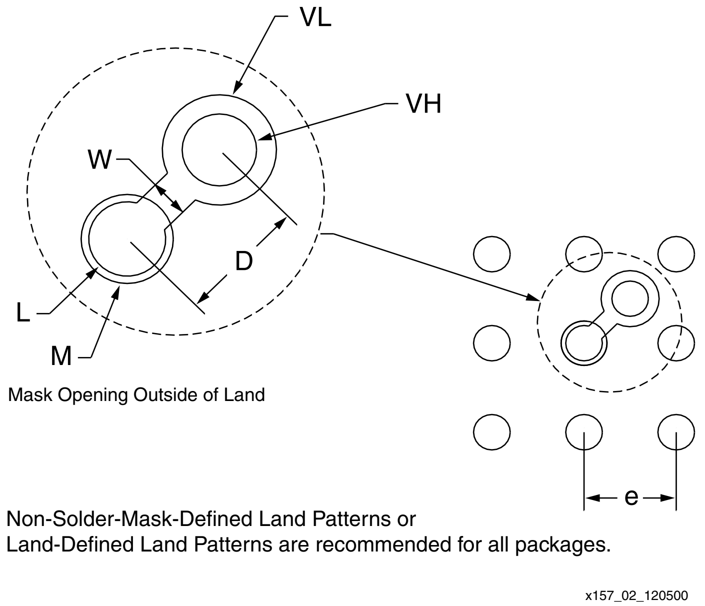


Figure 4-47: **Suggested Board Layout of Soldered Pads for BGA Packages**

Table 4-5: **Summary of Typical Land Pad Values (mm)**

Land Pad Characteristics	CS144	FG256	FG456	FG676	BG575	BG728	FF896	FF1152	FF1517	BF957
Component Land Pad Diameter (SMD)[4]	0.35	0.45	0.45	0.45	0.61	0.61	0.58	0.58	0.58	0.61
Solder Land (L) Diameter	0.33	0.40	0.40	0.40	0.56	0.56	0.50	0.50	0.50	0.56
Opening in Solder Mask (M) Diameter	0.44	0.50	0.50	0.50	0.66	0.66	0.60	0.60	0.60	0.66
Solder (Ball) Land Pitch (e)	0.80	1.00	1.00	1.00	1.27	1.27	1.00	1.00	1.00	1.27
Line Width Between Via and Land (w)	0.130	0.130	0.130	0.130	0.203	0.203	0.130	0.130	0.130	0.203
Distance Between Via and Land (D)	0.56	0.70	0.70	0.70	0.90	0.90	0.70	0.70	0.70	0.90
Via Land (VL) Diameter	0.51	0.61	0.61	0.61	0.65	0.65	0.61	0.61	0.61	0.65
Through Hole (VH), Diameter	0.250	0.300	0.300	0.300	0.356	0.356	0.300	0.300	0.300	0.356
Pad Array	-	Full	Full	Full	Full	Full	Full	Full	Full	Full
Matrix or External Row	13 x 13	16 x 16	22 x 22	26 x 26	24 x 24	27 x 27	30 x 30	34 x 34	39 x 39	31 x 31
Periphery Rows	4	-	7[3]	-	-	-	-	-	-	-

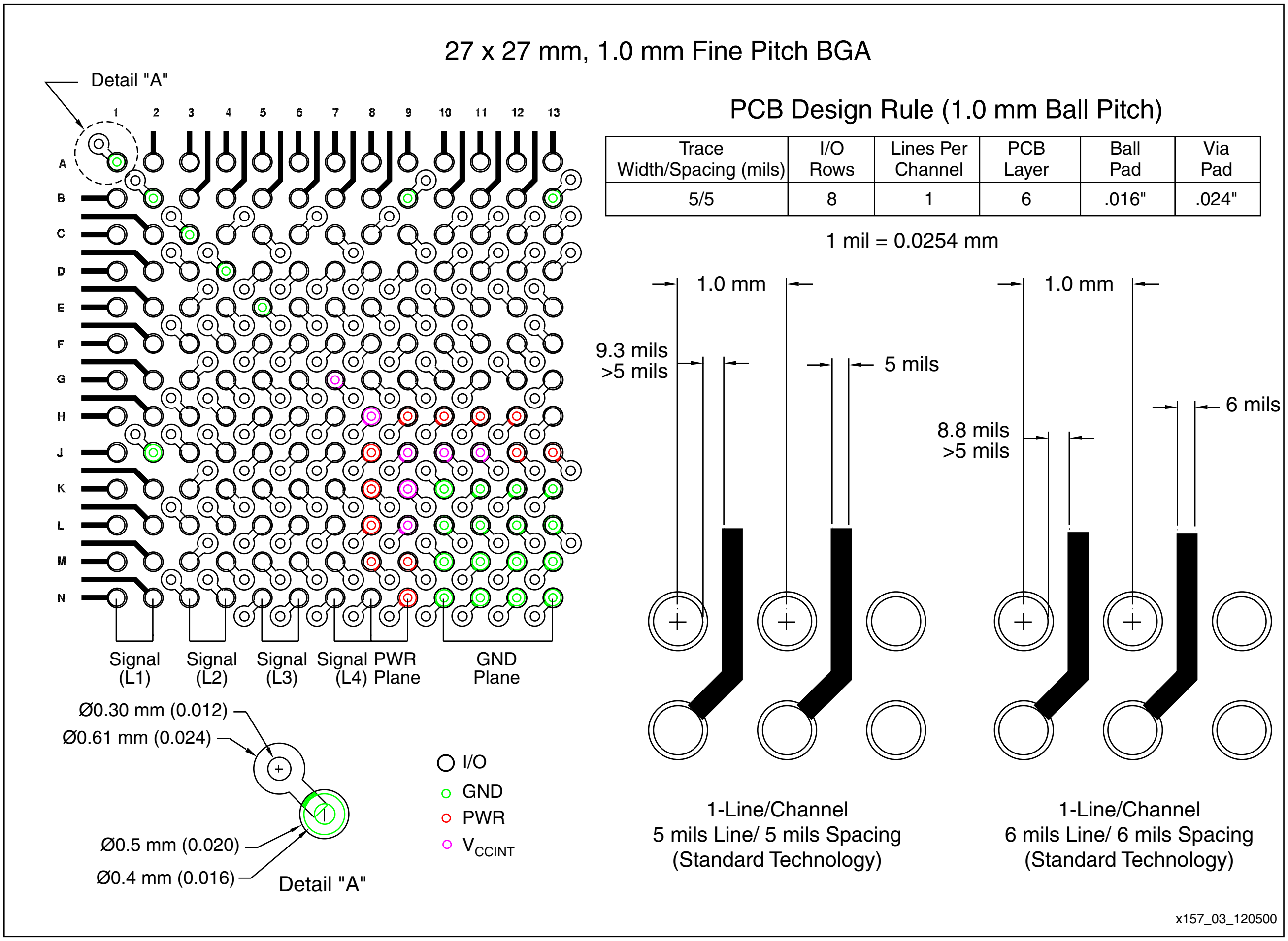
Notes:

1. Dimension in millimeters.
2. 3 x 3 matrix for illustration only, one land pad shown with via connection.
3. FG456 package has solder balls in the center in addition to the periphery rows of balls.
4. Component land pad diameter refers to the pad opening on the component side (solder-mask defined).

For Xilinx BGA packages, non-solder-mask defined (NSMD) pads on the board are suggested. This allows a clearance between the land metal (diameter L) and the solder mask opening (diameter M) as shown in Figure 4-47. The space between the NSMD pad and the solder mask, as well as the actual signal trace widths, depend on the capability of the PCB vendor. The cost of the PCB is higher when the line width and spaces are smaller.

Selection of pad types and sizes determines the available space between adjacent balls for signal escape. Based on PCB capability, the number of lines that can share the available space is described in Figure 4-48. Based on geometrical considerations, if one signal escapes between adjacent balls, then two signal rows can be routed on a single metal layer. This is illustrated in Figure 4-48 as routing with one line/channel, either at 6 mils-wide lines and spaces or 5 mils-wide lines and spaces. Using this suggested routing scheme, a minimum of eight PCB layers are required to route 10 signal rows in a package.

A slightly lower trace width can be used by the inner signal rows routed in internal layers than the width used in top and bottom external or exposed traces. Depending on the signal being handled, the practice of "necking down" a trace in the critical space between the BGA balls is allowable. Changes in width over very short distances can cause small impedance changes. Validate these issues with the board vendor and signal integrity engineers responsible for the design.



Trace Width/Spacing (mils)	I/O Rows	Lines Per Channel	PCB Layer	Ball Pad	Via Pad
5/5	8	1	6	.016"	.024"

Figure 4-48: **FG676 PC Board Layout/Land Pattern**

Figure 4-48 describes a board-level layout strategy for a Xilinx 1.0 mm pitch FG676 package. Detail A in Figure 4-48 describes the opening geometry for the Land Pad and the Solder Mask. Routing with 5 mils-wide lines or spaces allows one signal per channel (between the balls). For successful routing, eight-row deep signal traces require six PCB layers. Figure 4-49 shows the suggested schematic of layers for the six-layer routing scheme.

Using premium board technology, such as Microvia Technology (allowing up to 4 mils-wide lines and spaces), efficient routing is possible with a reduced number of board layers. A grouping scheme for power, ground, control, and I/O pins, might also enable efficient routing.

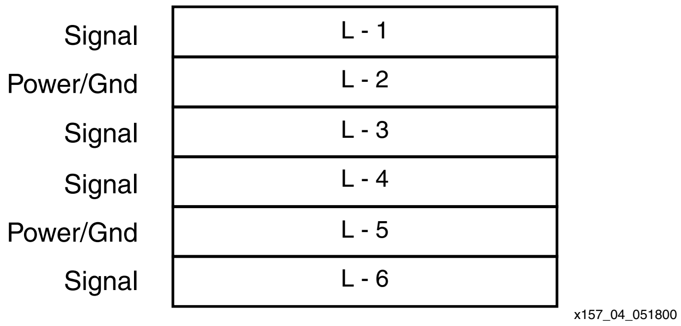


Figure 4-49: **Six-Layer Routing Scheme**

Figure 4-50 through Figure 4-67 show suggested layer-by-layer board routing for each Virtex-II package, including flip-chip packages. These drawings assume a standard PCB technology of 5 mils-wide lines and spaces. Table 4-6 lists the layer-by-layer routing examples provided. More details are contained in XAPP157, which is available on the web at **www.xilinx.com/xapp/xapp157.pdf**, as is a full-color (PDF) version of this document.

Table 4-6: **Layer-By-Layer Board Routing Examples**

Package	Standard Routing	Routing With LVDS Pairs
FG256	Top and bottom layers	Top and bottom layers
FG456	Top, 2nd, and bottom layers	Top, 2nd, and bottom layers
FG676	Top, 2nd, 3rd, and bottom layers	Top, 2nd, 3rd, and bottom layers
BG575	Top, 2nd, and bottom layers	Top, 2nd, and bottom layers
BG728	Top, 2nd, 3rd, and bottom layers	Top, 2nd, 3rd, and bottom layers
FF896	Top, 2nd, 3rd, and bottom layers	Top, 2nd, 3rd, and bottom layers
FF1152	Top, 2nd, 3rd, 4th, and bottom layers	Top, 2nd, 3rd, 4th, and bottom layers
FF1517	Top, 2nd, 3rd, 4th, 5th, and bottom layers	Top, 2nd, 3rd, 4th, 5th, and bottom layers
BF957	Top, 2nd, 3rd, and bottom layers	Top, 2nd, 3rd, and bottom layers

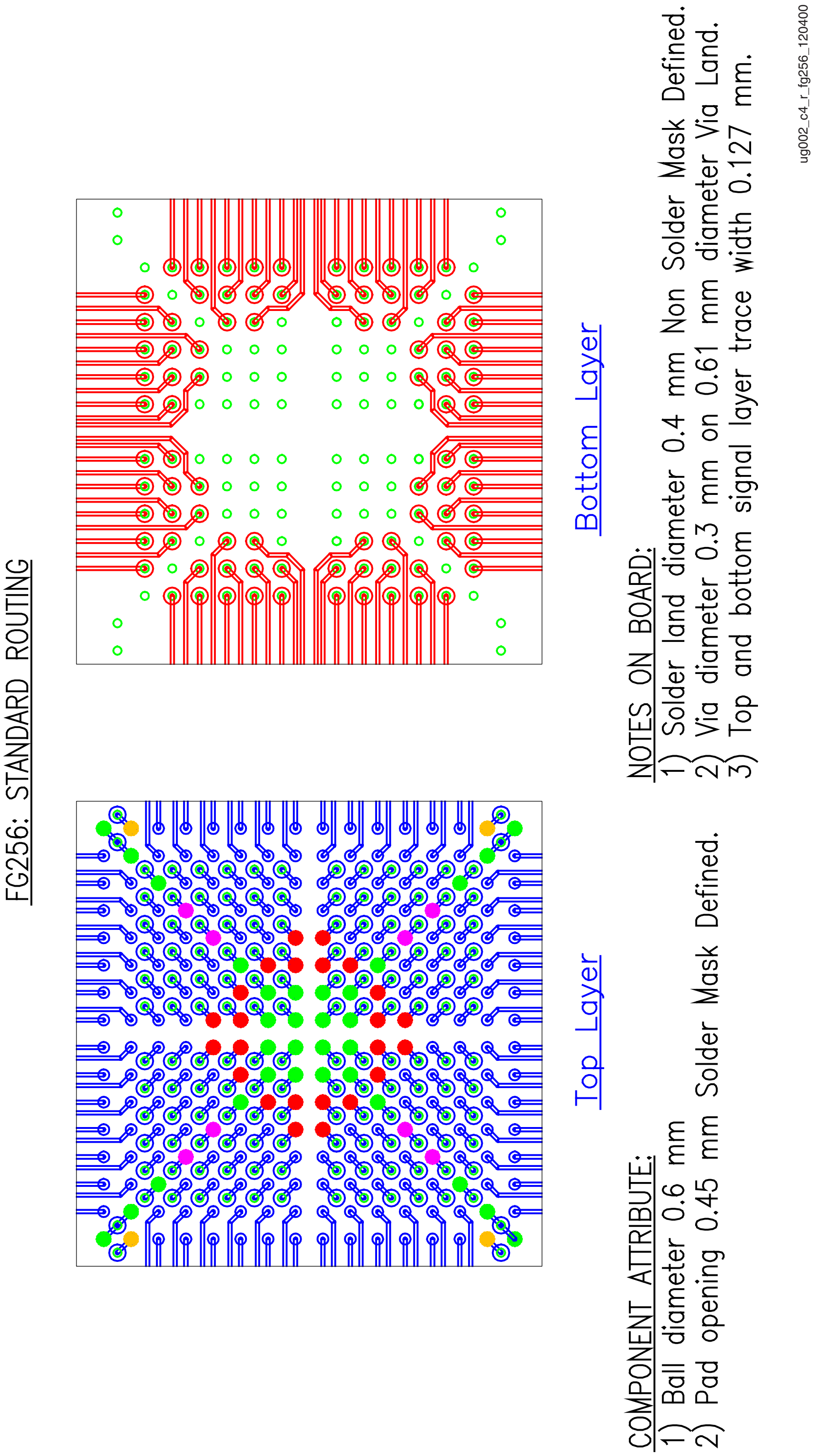
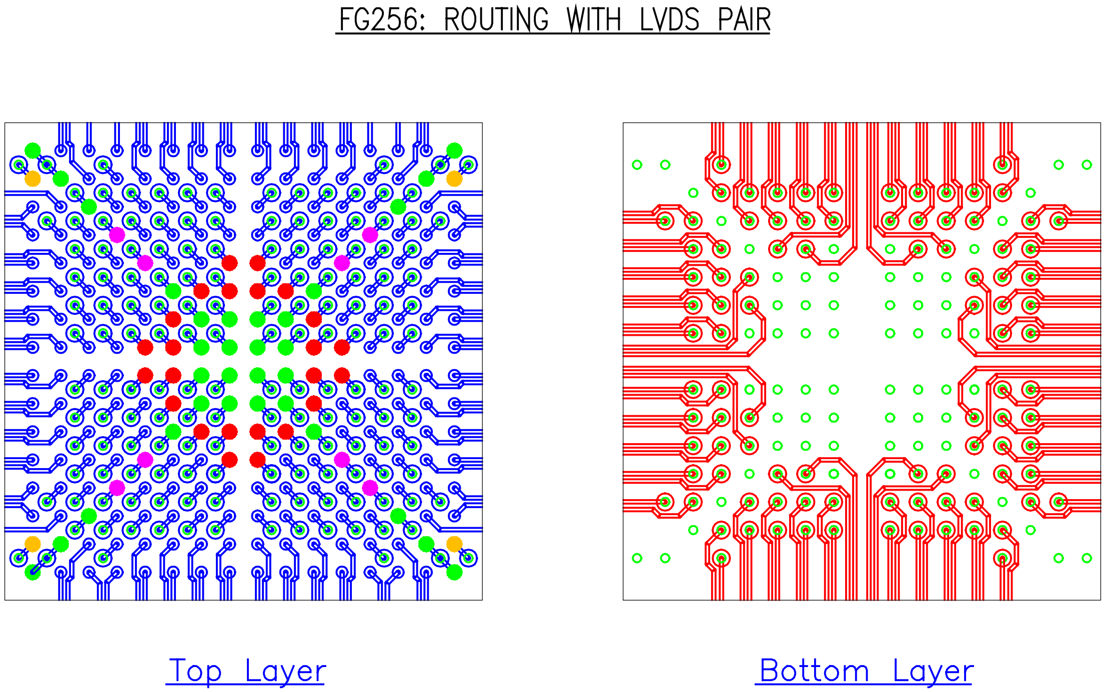


Figure 4-50: **FG256 Standard Routing**

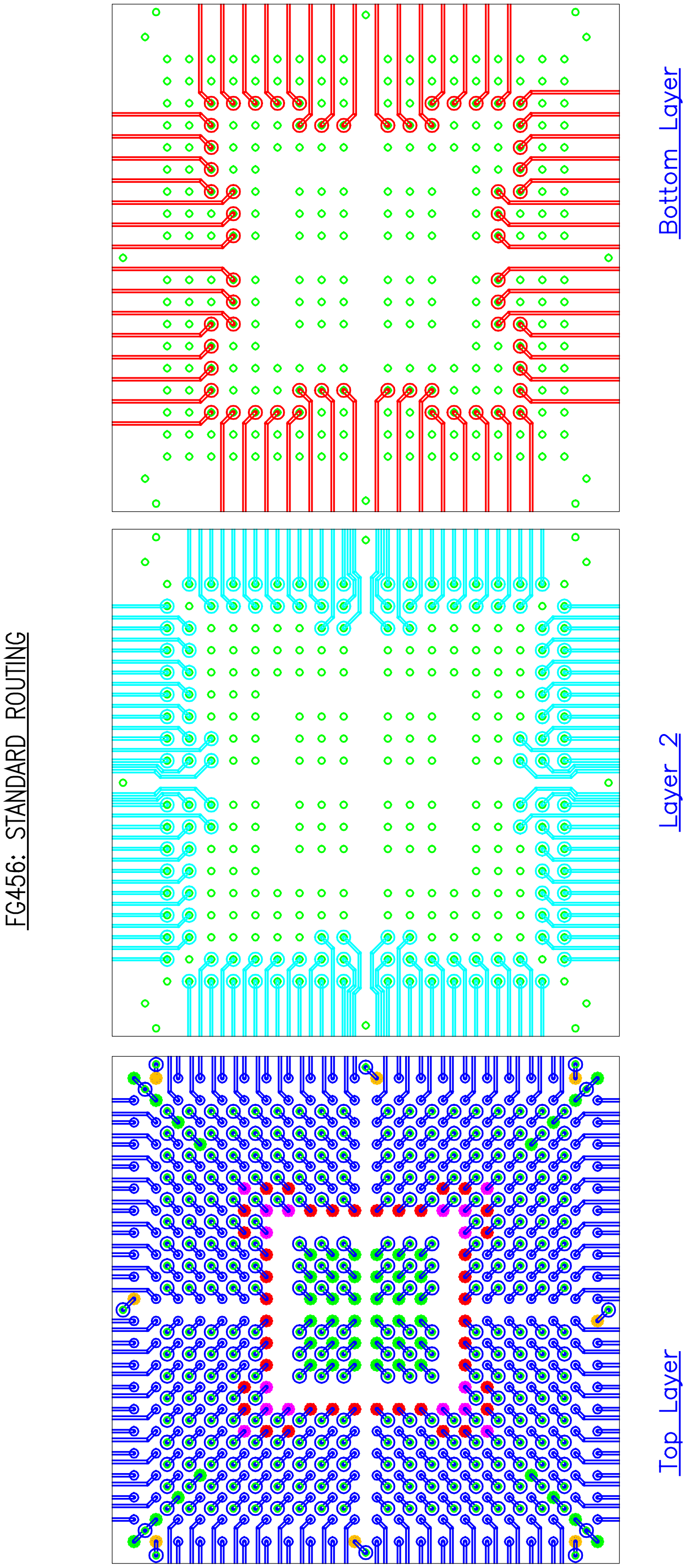


COMPONENT ATTRIBUTE:
1) Ball diameter 0.6 mm
2) Pad opening 0.45 mm Solder Mask Defined.

NOTES ON BOARD:
1) Solder land diameter 0.4 mm Non Solder Mask Defined.
2) Via diameter 0.3 mm on 0.61 mm diameter Via Land.
3) Top and bottom signal layer trace width 0.127 mm.

ug002_c4_r_fg256lvdspair_120400

Figure 4-51: **FG256 Routing With LVDS Pairs**



COMPONENT ATTRIBUTE:
1) Ball diameter 0.6 mm
2) Pad opening 0.45 mm Solder Mask Defined.

NOTES ON BOARD:
1) Solder land diameter 0.4 mm Non Solder Mask Defined.
2) Via diameter 0.3 mm on 0.61 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

Figure 4-52: **FG456 Standard Routing**

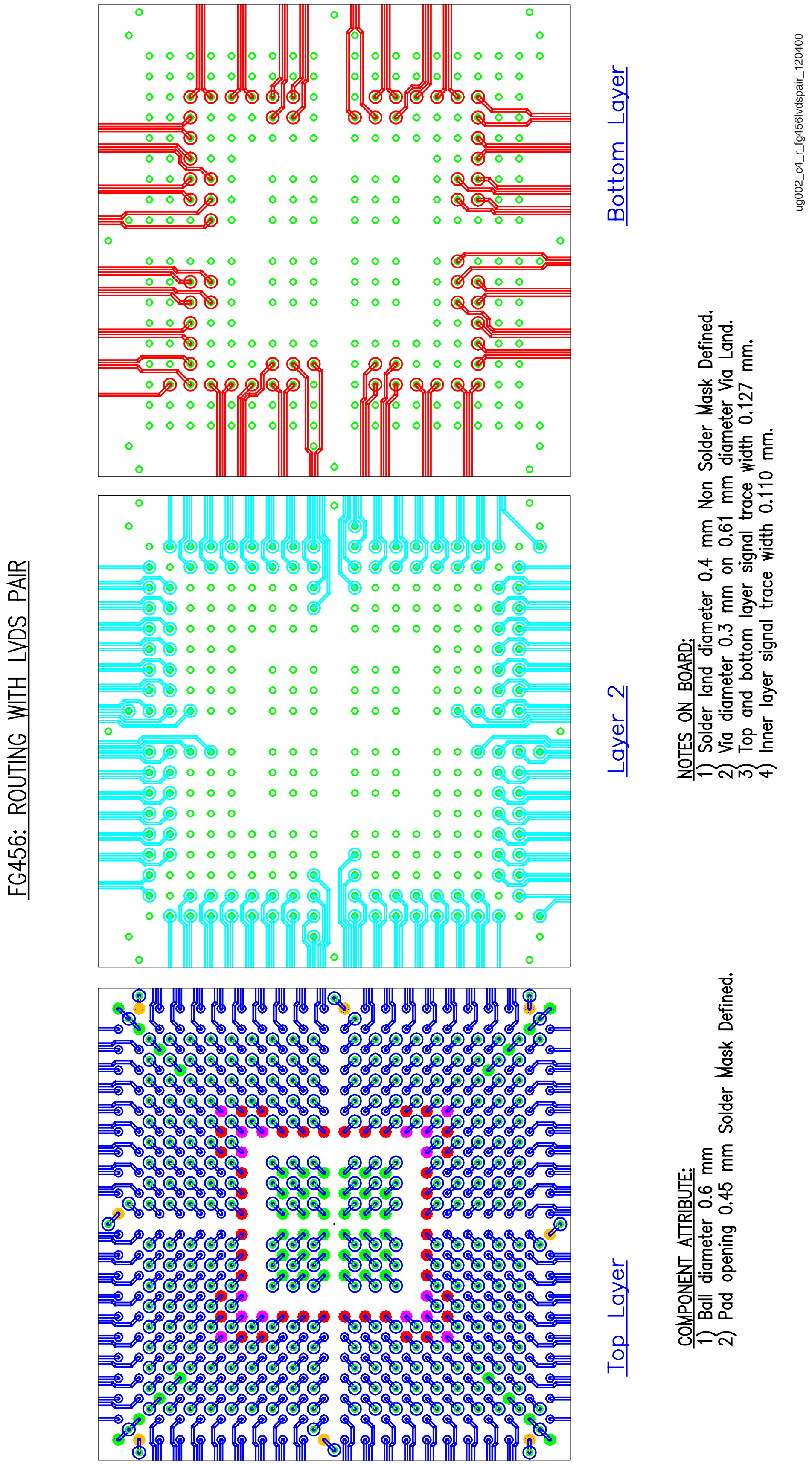


Figure 4-53: **FG456 Routing With LVDS Pairs**

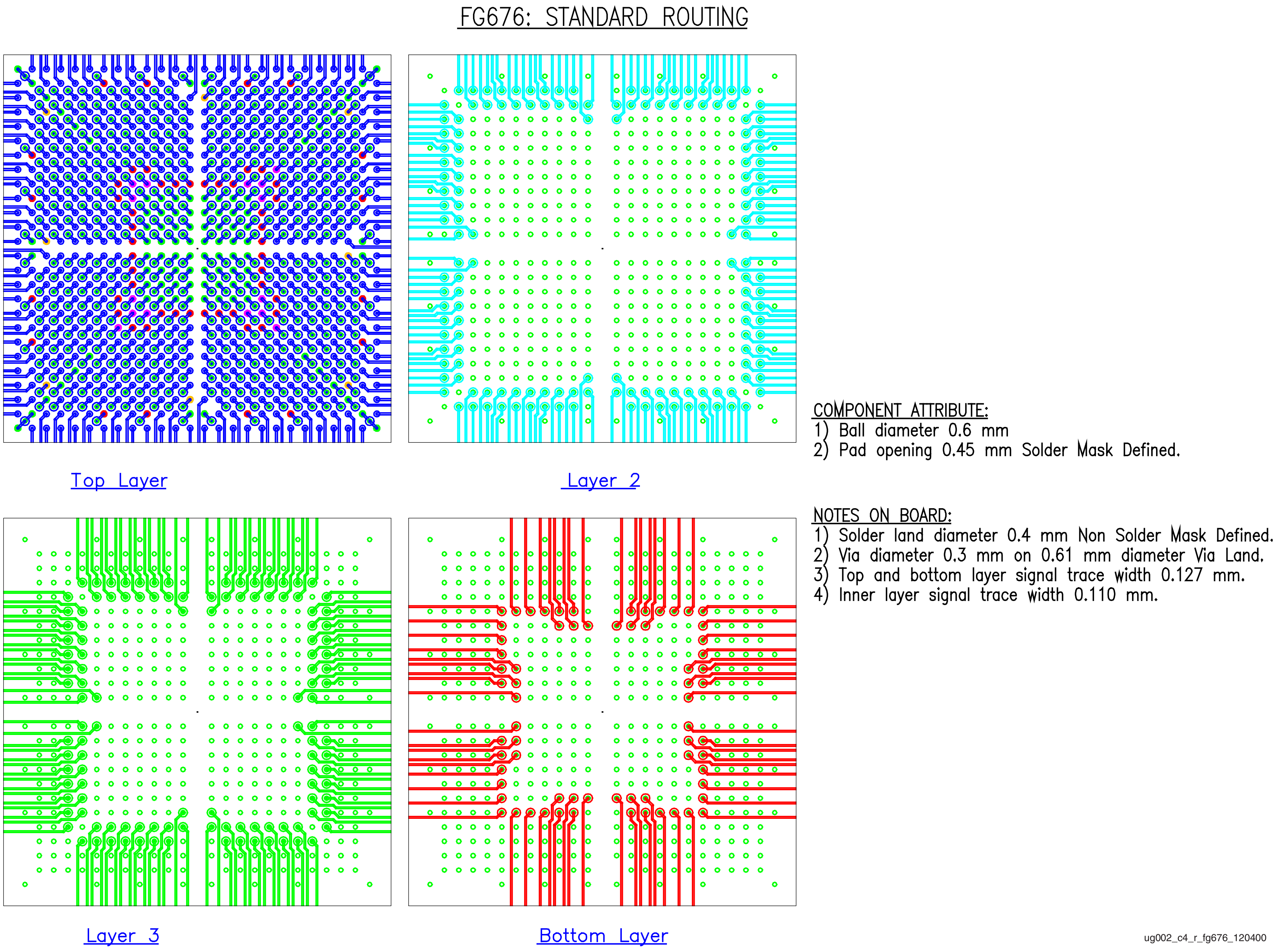
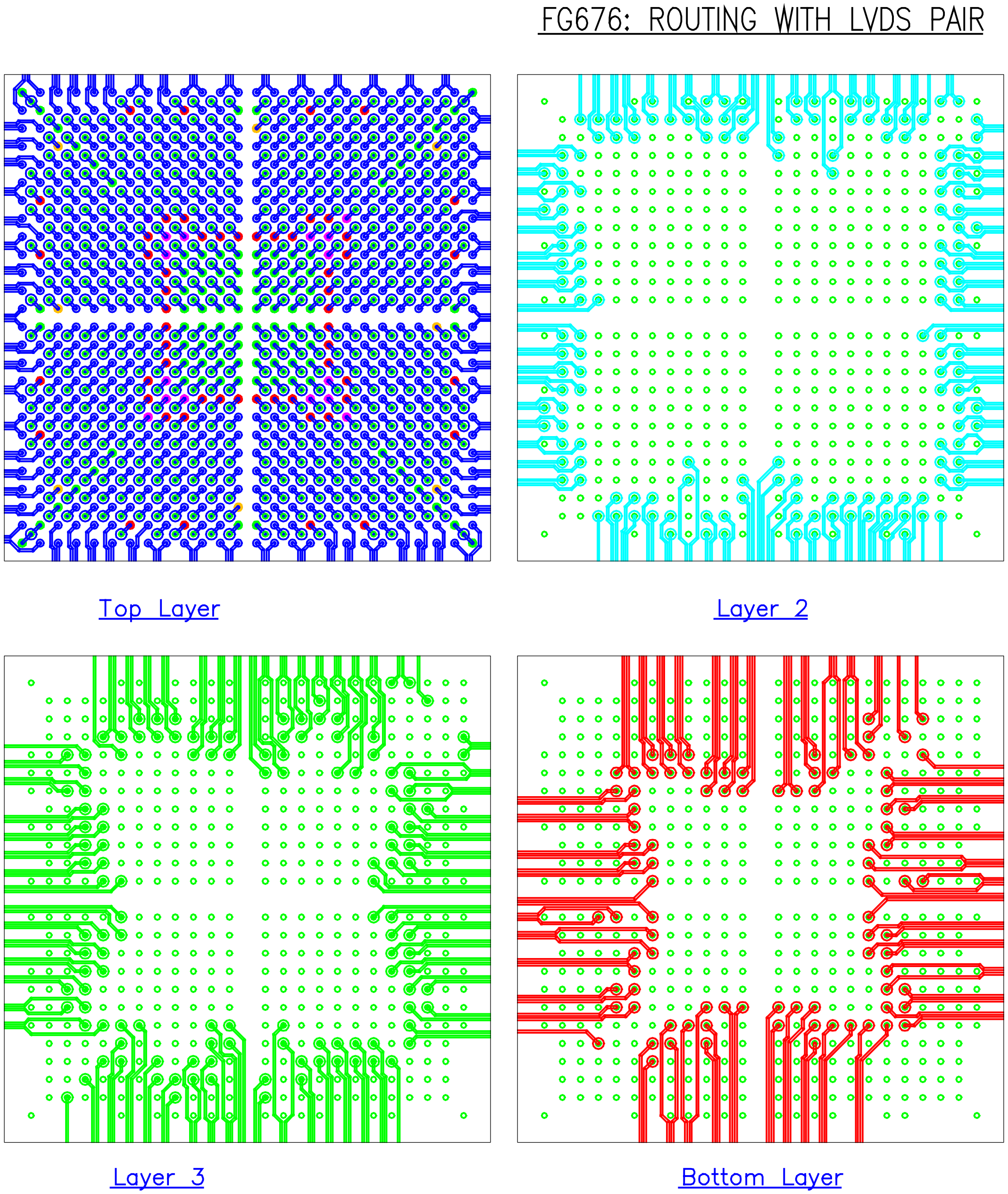


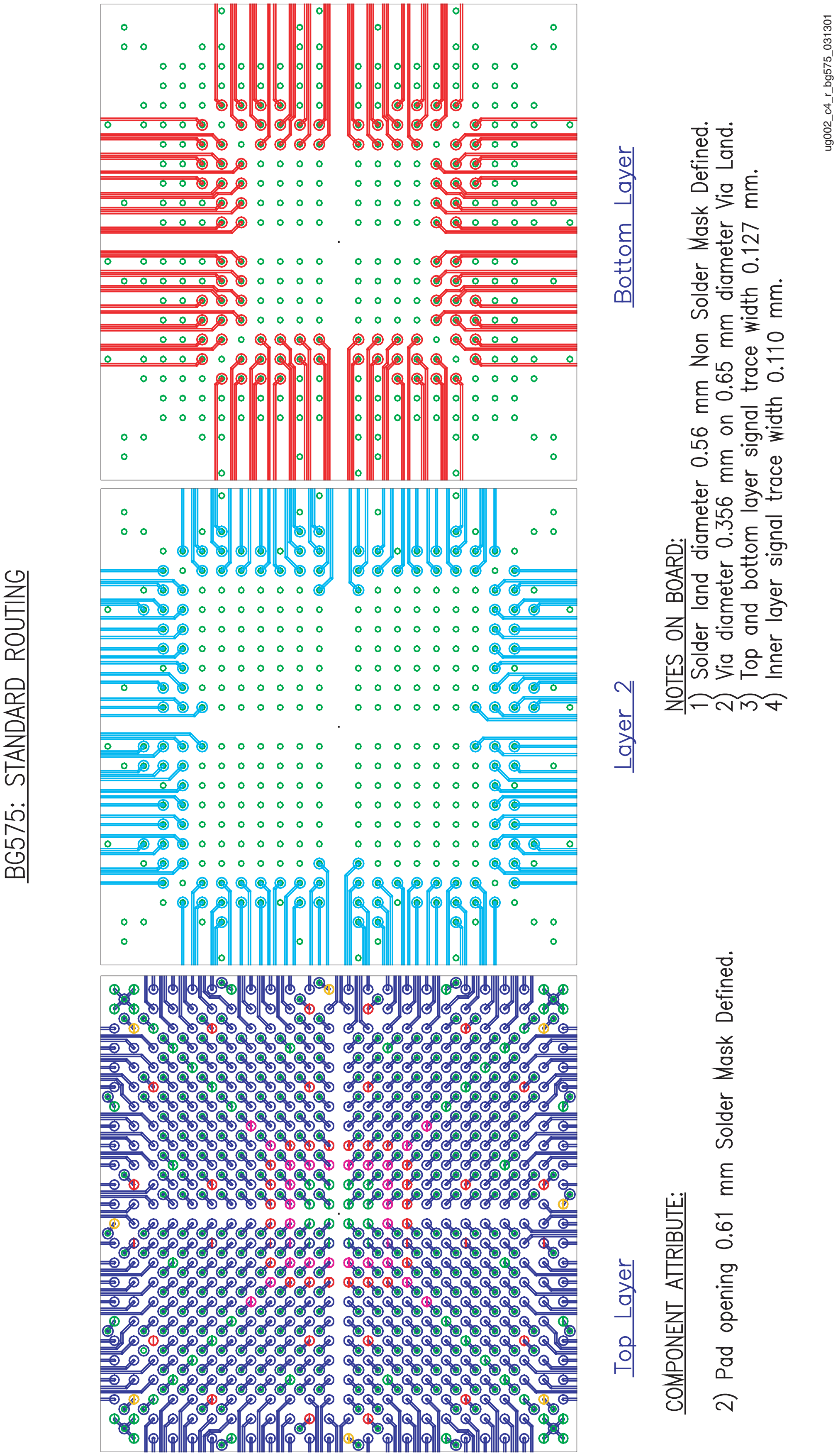
Figure 4-54: **FG676 Standard Routing**



COMPONENT ATTRIBUTE:
1) Ball diameter 0.6 mm
2) Pad opening 0.45 mm Solder Mask Defined.

NOTES ON BOARD:
1) Solder land diameter 0.4 mm Non Solder Mask Defined.
2) Via diameter 0.3 mm on 0.61 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

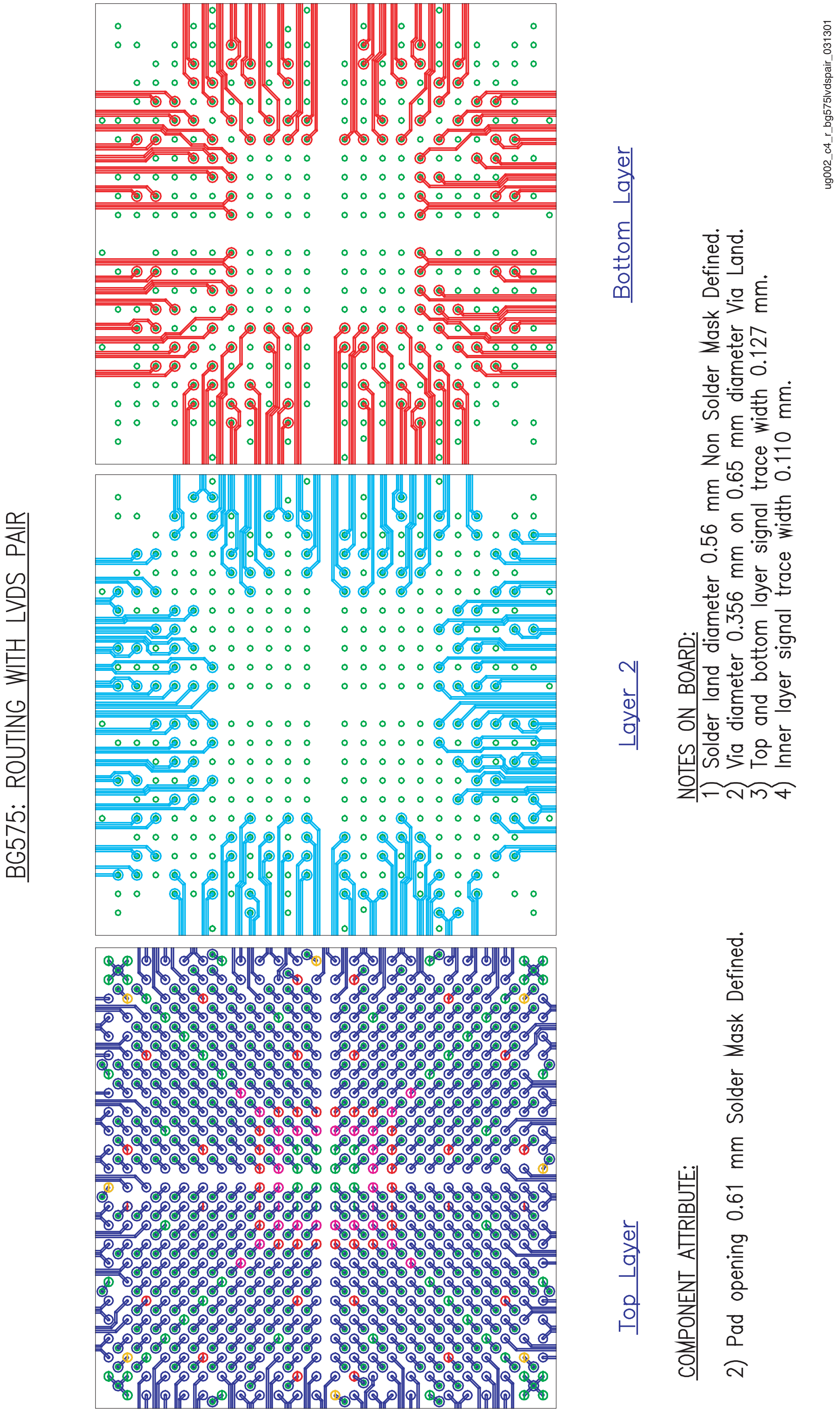
Figure 4-55: **FG676 Routing With LVDS Pairs**



NOTES ON BOARD:
1) Solder land diameter 0.56 mm Non Solder Mask Defined.
2) Via diameter 0.356 mm on 0.65 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

COMPONENT ATTRIBUTE:
2) Pad opening 0.61 mm Solder Mask Defined.

Figure 4-56: **BG575 Standard Routing**



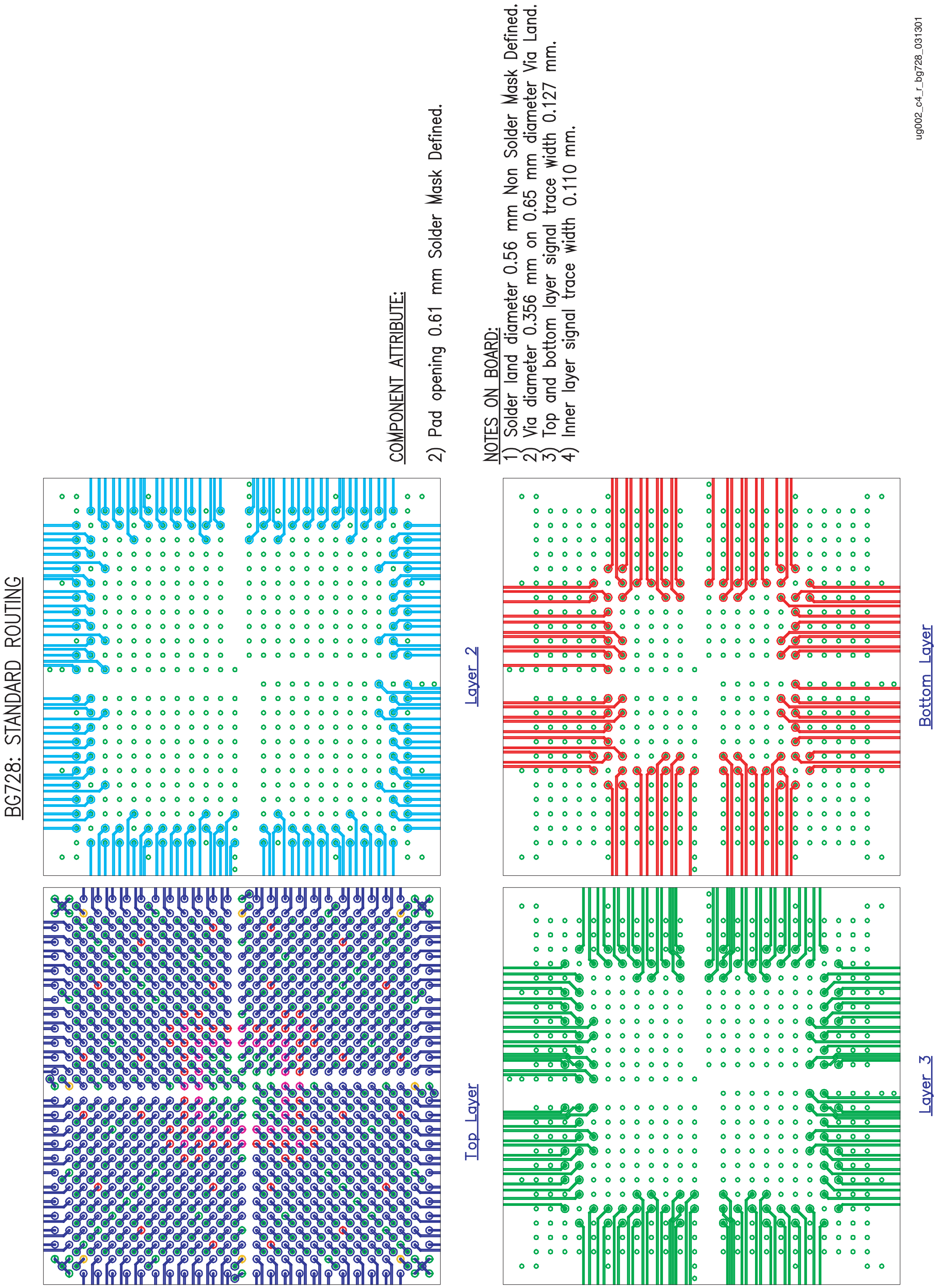
COMPONENT ATTRIBUTE:

2) Pad opening 0.61 mm Solder Mask Defined.

NOTES ON BOARD:

1) Solder land diameter 0.56 mm Non Solder Mask Defined.
2) Via diameter 0.356 mm on 0.65 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

Figure 4-57: **BG575 Routing With LVDS Pairs**



COMPONENT ATTRIBUTE:

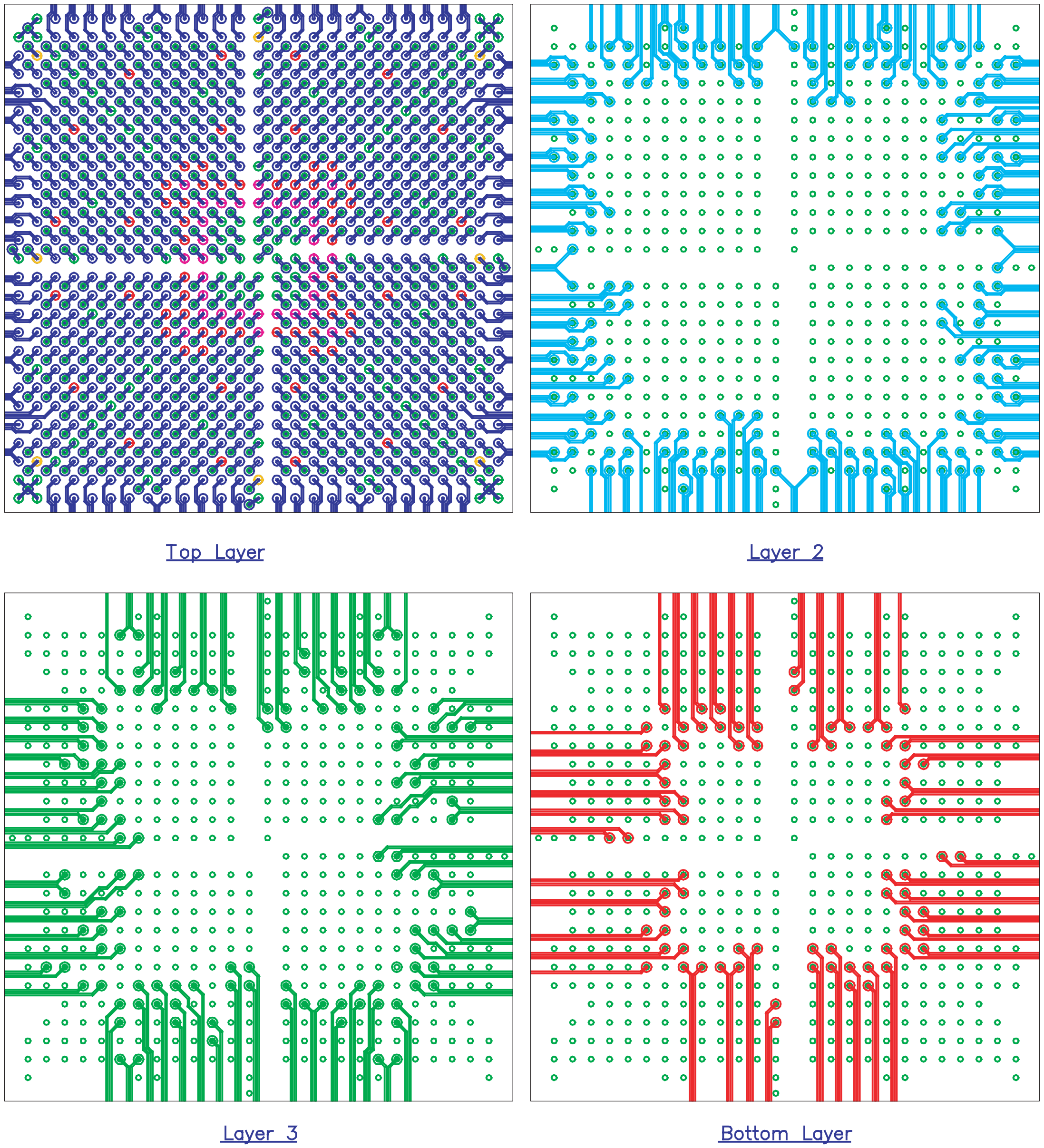
2) Pad opening 0.61 mm Solder Mask Defined.

NOTES ON BOARD:

1) Solder land diameter 0.56 mm Non Solder Mask Defined.
2) Via diameter 0.356 mm on 0.65 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

Figure 4-58: **BG728 Standard Routing**

BG728: ROUTING WITH LVDS PAIR



COMPONENT ATTRIBUTE:

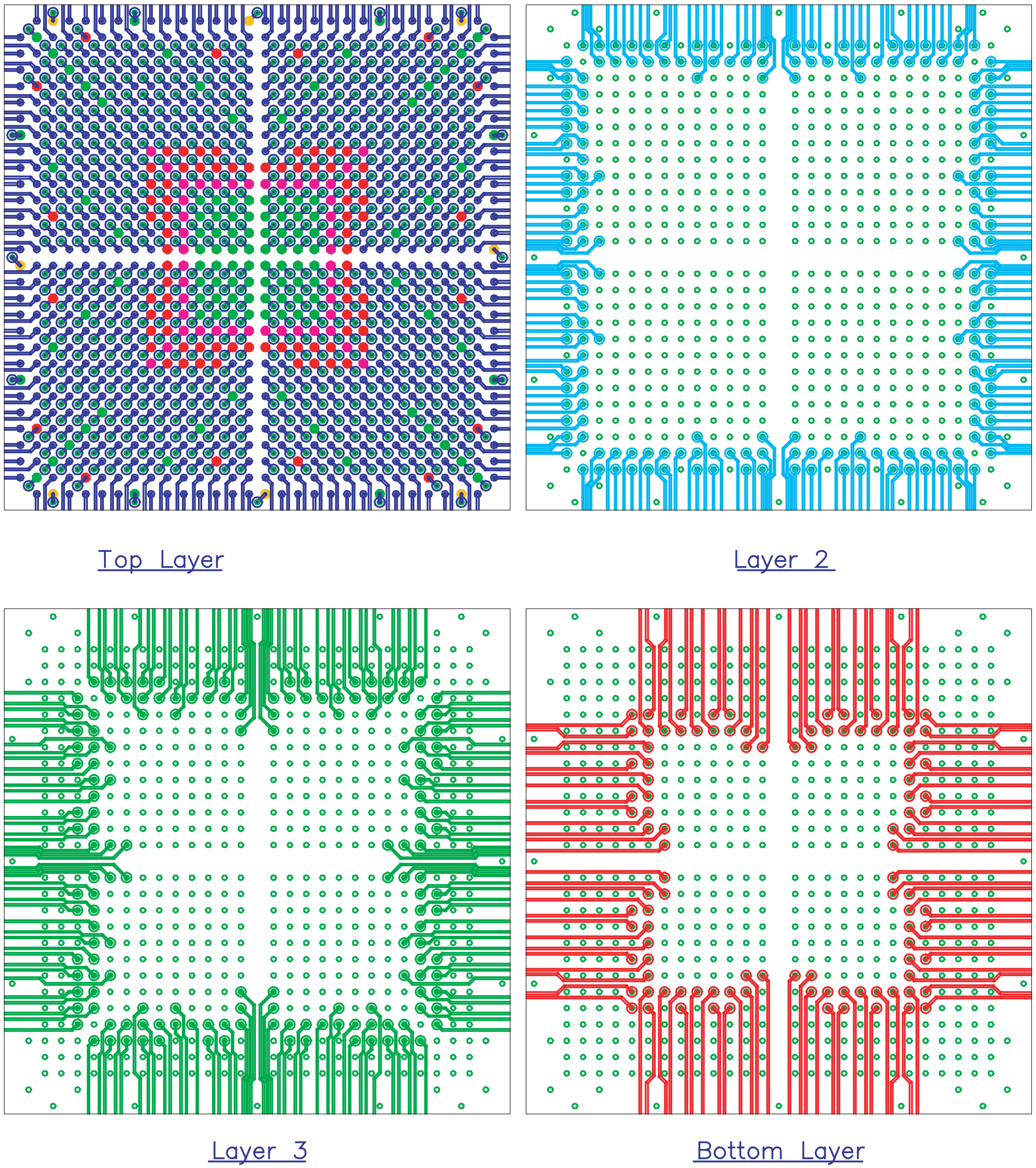
2) Pad opening 0.61 mm Solder Mask Defined.

NOTES ON BOARD:
1) Solder land diameter 0.56 mm Non Solder Mask Defined.
2) Via diameter 0.356 mm on 0.65 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

ug002_c4_r_bg728lvdspair_031301

Figure 4-59: **BG728 Routing With LVDS Pairs**

FF896: STANDARD ROUTING



COMPONENT ATTRIBUTE:

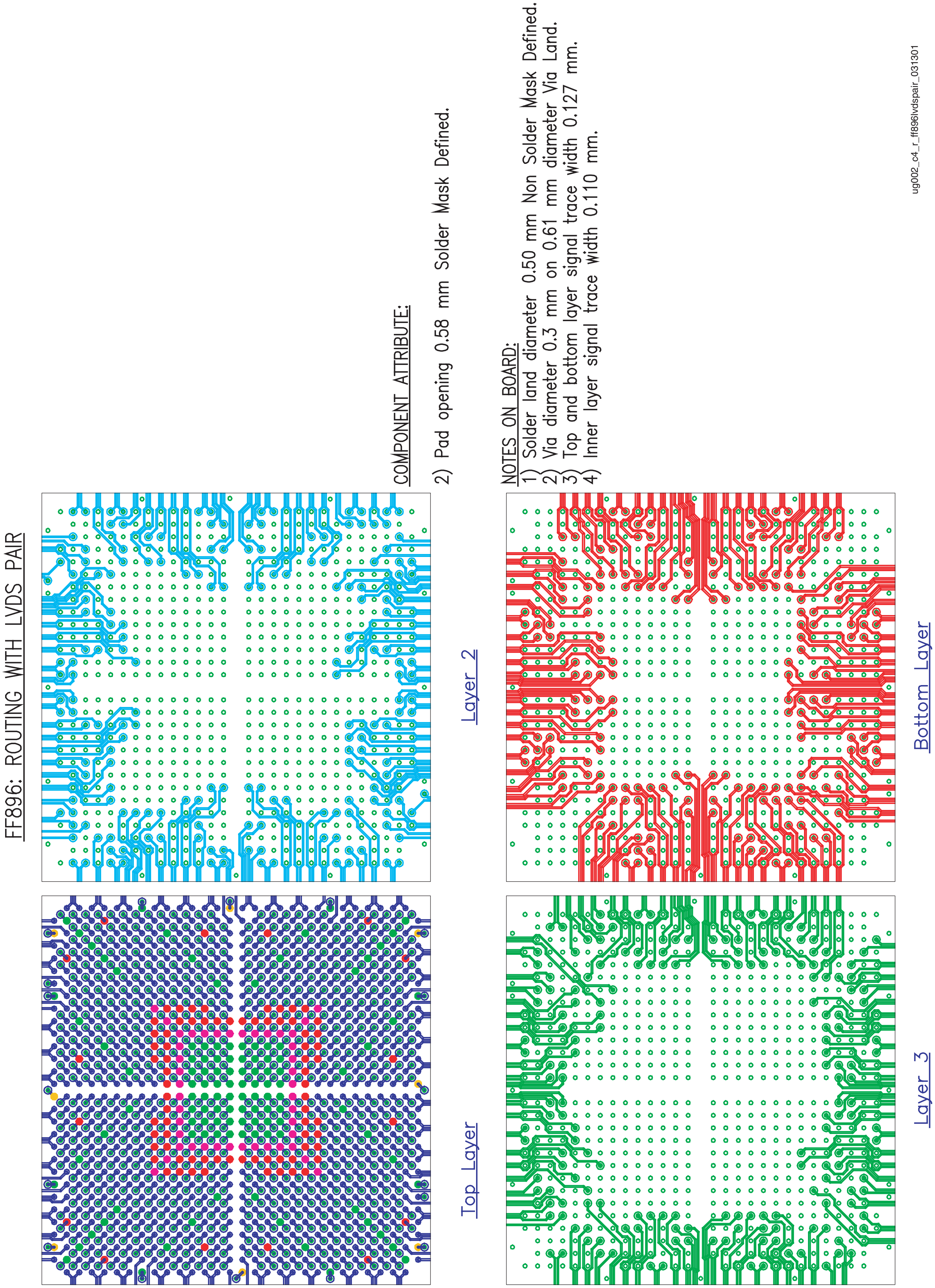
2) Pad opening 0.58 mm Solder Mask Defined.

NOTES ON BOARD:

1) Solder land diameter 0.50 mm Non Solder Mask Defined.
2) Via diameter 0.3 mm on 0.61 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

ug002_c4_r_ff896_031301

Figure 4-60: **FF896 Standard Routing**



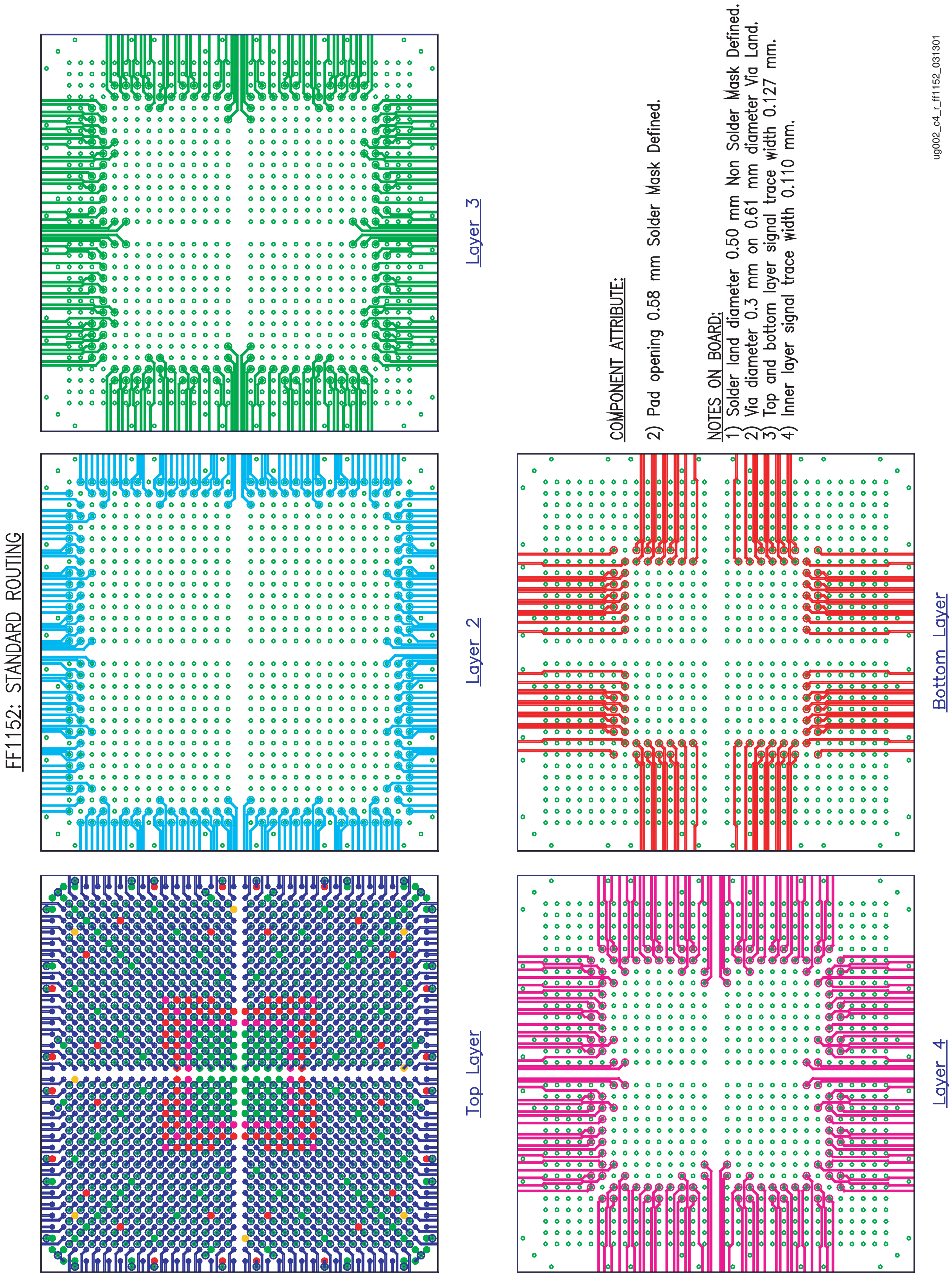
COMPONENT ATTRIBUTE:

2) Pad opening 0.58 mm Solder Mask Defined.

NOTES ON BOARD:

1) Solder land diameter 0.50 mm Non Solder Mask Defined.
2) Via diameter 0.3 mm on 0.61 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

Figure 4-61: **FF896 Routing With LVDS Pairs**



FF1152: STANDARD ROUTING

COMPONENT ATTRIBUTE:

2) Pad opening 0.58 mm Solder Mask Defined.

NOTES ON BOARD:
1) Solder land diameter 0.50 mm Non Solder Mask Defined.
2) Via diameter 0.3 mm on 0.61 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

Figure 4-62: **FF1152 Standard Routing**

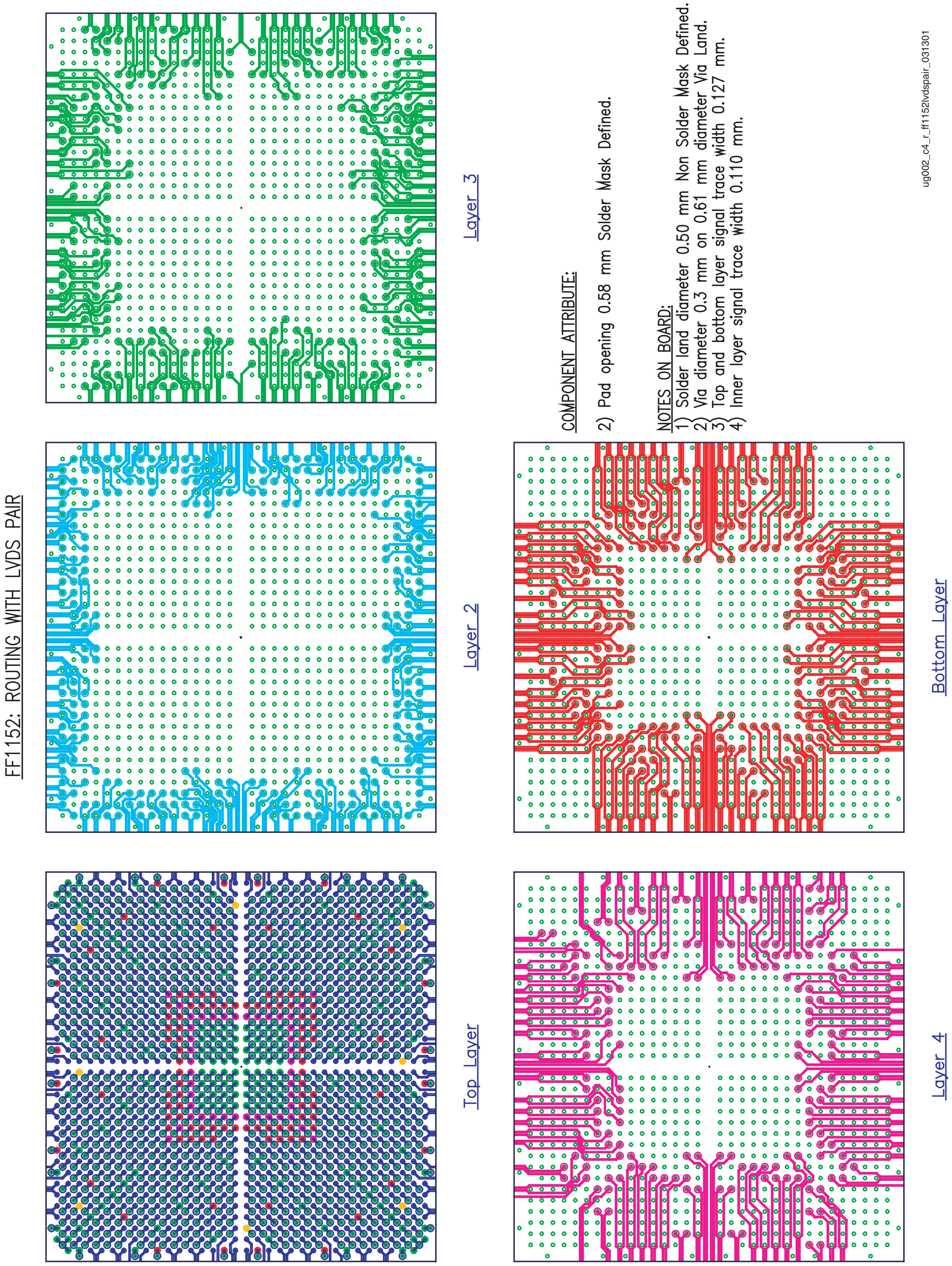


Figure 4-63: **FF1152 Routing With LVDS Pairs**

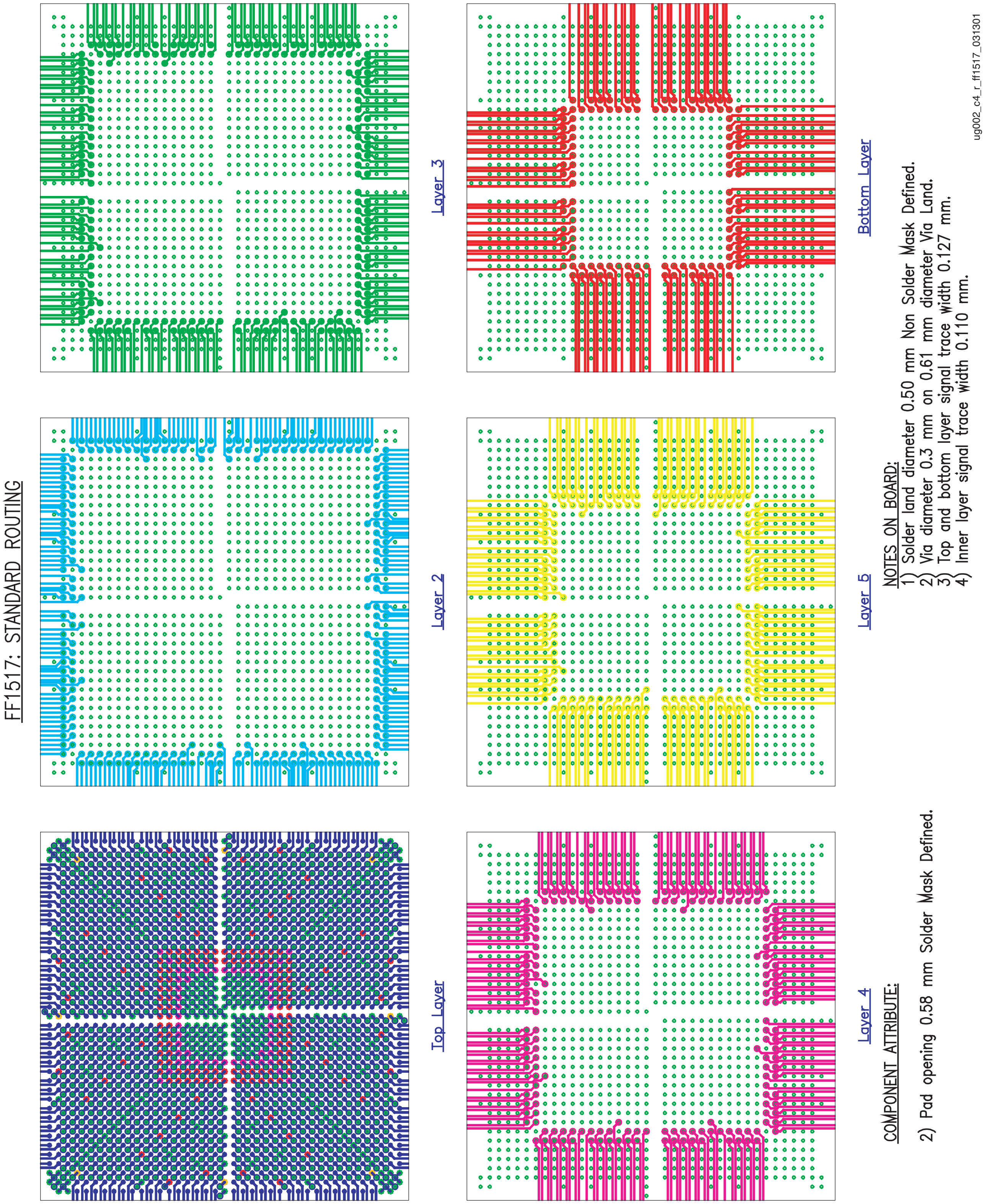
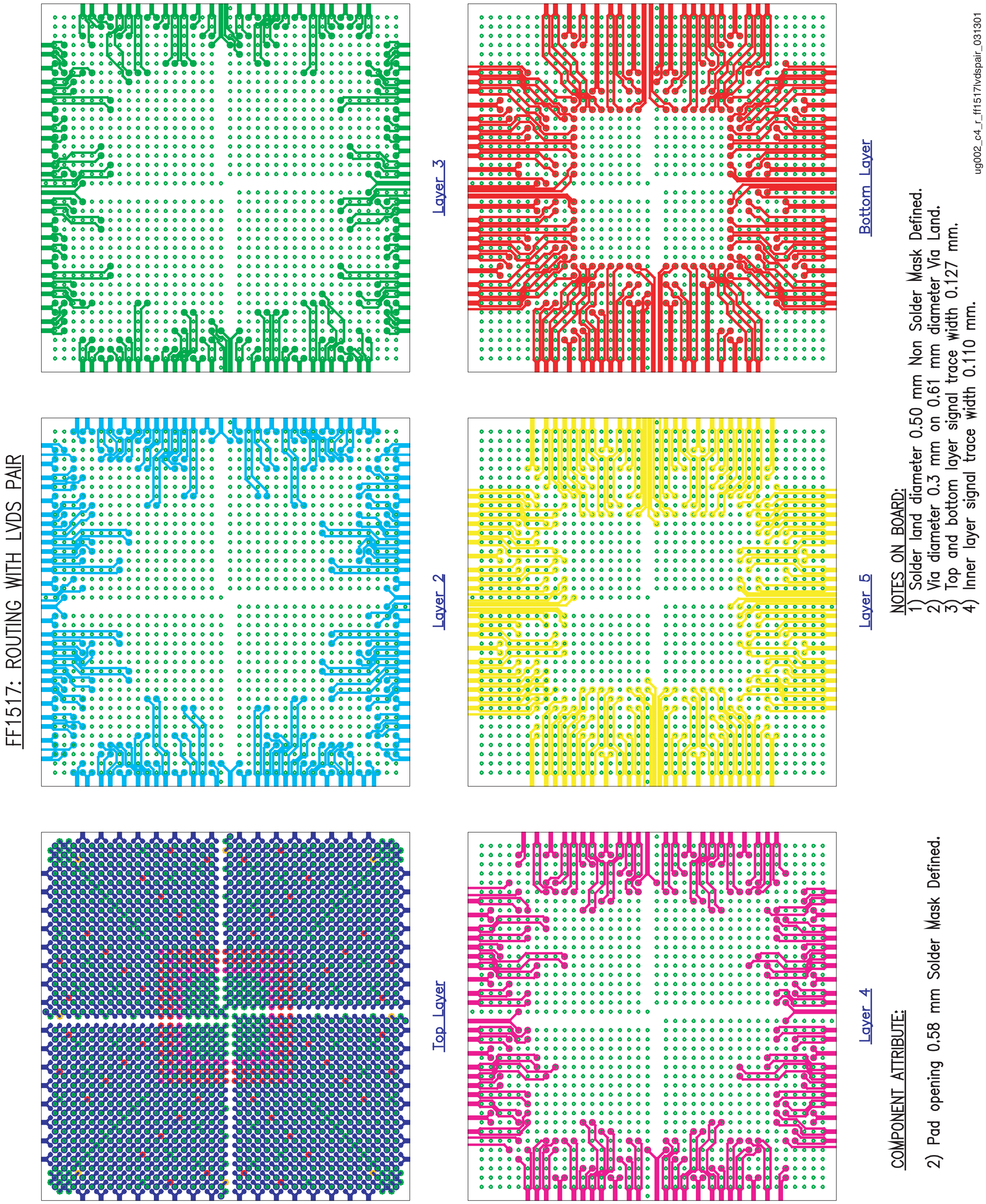


Figure 4-64: **FF1517 Standard Routing**



NOTES ON BOARD:
1) Solder land diameter 0.50 mm Non Solder Mask Defined.
2) Via diameter 0.3 mm on 0.61 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

COMPONENT ATTRIBUTE:

2) Pad opening 0.58 mm Solder Mask Defined.

Figure 4-65: **FF1517 Routing With LVDS Pairs**

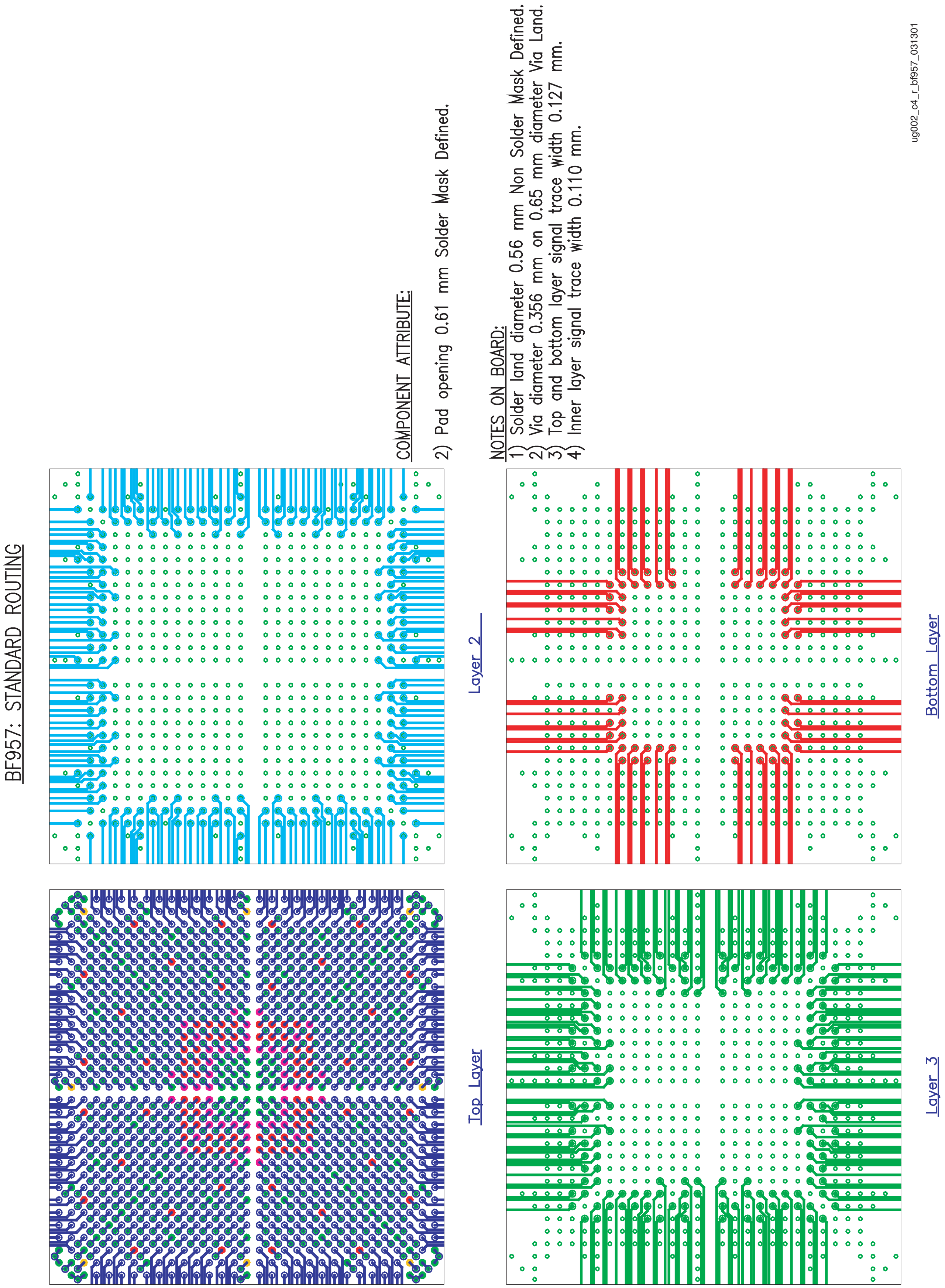
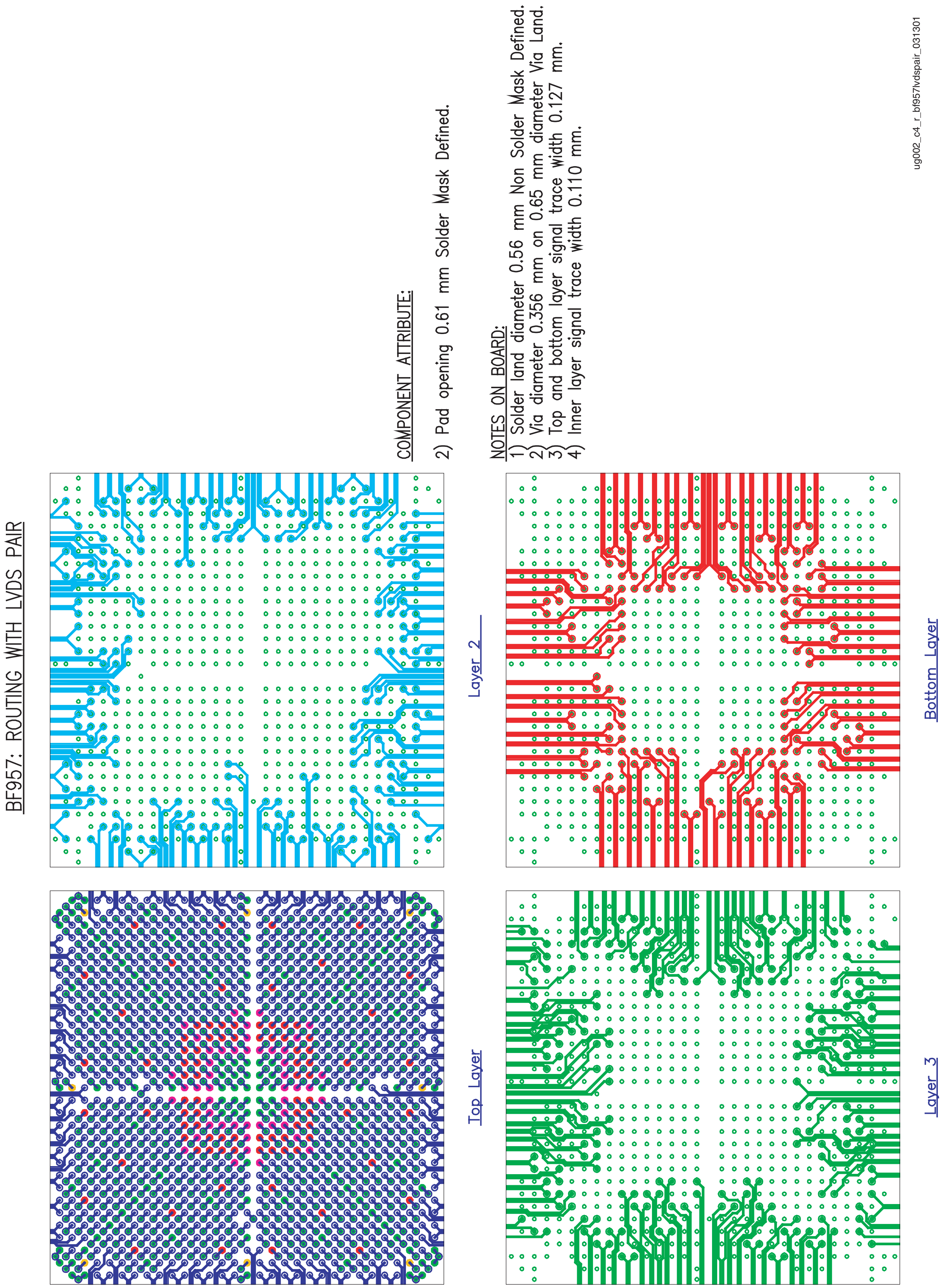


Figure 4-66: **BF957 Standard Routing**



BF957: ROUTING WITH LVDS PAIR

COMPONENT ATTRIBUTE:

2) Pad opening 0.61 mm Solder Mask Defined.

NOTES ON BOARD:

1) Solder land diameter 0.56 mm Non Solder Mask Defined.
2) Via diameter 0.356 mm on 0.65 mm diameter Via Land.
3) Top and bottom layer signal trace width 0.127 mm.
4) Inner layer signal trace width 0.110 mm.

Figure 4-67: **BF957 Routing With LVDS Pairs**

Power Consumption

The Virtex-II power estimator worksheet estimates power consumption for a Virtex-II design before it is downloaded. It considers the design resource usage, toggle rates, I/O power, and many other factors in the estimation. The formulas used for calculations in the program are based on test design measurements.

Xilinx provides two versions of the power estimator, an Excel 97 version that works with Microsoft Office 97 software, and a CGI version for use with web browsers. They are identical in terms of estimations and data entries.

This section explains how to use the Power Estimator Worksheet to calculate estimated power consumption for Virtex-II designs. Since this is an estimation tool, results may not match precisely with what is measured on the board.

The power estimator consists of six categories: CLB (configurable logic block) logic power, dedicated non-multiplier power, dedicated registered multiplier power, block SelectRAM power, DCM (digital clock management), input/output power, and the results. To estimate power with the worksheet, a designer must determine how to group portions of the design into modules, what resources each module contains, the respective clock frequencies, and average toggle rates.

Note:

1. The Virtex-II power estimation is still under development. The table entries in this section may be different from the entries in the released version of the power estimation tool,

CLB Logic Power

Table 4-7 shows the data entries required for the CLB Logic Power section in the Power Estimator. This section estimates the power consumption of the CLBs for a Virtex-II design. In this section, users need to partition designs into modules, specify area utilization, and toggle rates.

Table 4-7: **CLB Logic Power**

Module	Frequency (MHz)	CLB Slices	Flip-Flops/ Latches	LUT		Average Toggle Rate (%)	Routing Amount
				Shift Register	SelectRAM		
User Module 1	0	0	0	0	0	0%	Medium
User Module 2	0	0	0	0	0	0%	Medium
User Module 3	0	0	0	0	0	0%	Medium
User Module 4	0	0	0	0	0	0%	Medium
User Module 5	0	0	0	0	0	0%	Medium
User Module 6	0	0	0	0	0	0%	Medium
User Module 7	0	0	0	0	0	0%	Medium
User Module 8	0	0	0	0	0	0%	Medium

Modules

Modules are portions of a design. A designer could treat the entire design as one module and calculate its toggle rate. However, estimating power this way is not as accurate as when the design is divided into multiple modules. Generally, with more modules the estimate is better.

The Virtex-II power estimator allows designs to be partitioned into a maximum of eight modules. Determining how to partition the design into modules depends on user preference. Three partitioning approaches are presented below as guidelines.

Grouping by Hierarchy

If a design contains hierarchical components at the top level, these components may be separated or grouped together to represent modules.

Grouping by Clocks

If a design has several different clocks, the logic associated with each clock should be treated as a module. For accuracy, it is recommended that each module contains only one clock.

Grouping by Functionality

For a design with sub-components that perform different functions, each sub-component can be considered as a module. For example, a microprocessor can be thought of as three main modules: an ALU, a Register File, and a Control System.

Frequency (MHz)

Frequency is the clock speed for the module. Again, it is strongly recommended that each module contains only one clock.

CLB Slices

This involves the total CLB usage of a module. This number is available from the synthesis report in a specific synthesis tool. For a more accurate result, MAP only this module in Xilinx Foundation software, and take the numbers from the map.mrp file. The map.mrp file is the output resource usage file produced by running the MAP program in the Xilinx Foundation software.

For schematic-based designs, obtaining this number is slightly more difficult. Designers can either estimate CLB usage based on the design structure or MAP the module and read the numbers from the map.mrp file.

Flip Flops or Latches

The total number of flip-flop and latch elements used for each module can be obtained from the synthesis report, the map.mrp file, or by adding up the registers from the schematics.

Shift Register LUTs

This is the total number of SRL16 elements used in each module.

SelectRAM LUTs

This is the total number of LUTs used as Distributed Select RAM components. For Virtex-II devices, one 16 x 1 synchronous RAM is equivalent to one LUT, and one 16 x 1 dual-port RAM is equivalent to two LUTs (split between two slices).

Average Toggle Rate (%)

The toggle rate describes how often the output changes with respect to the input clock, usually between 6% and 12% for a typical module. Functional simulation is required to accurately calculate the toggle rate. Designers need to simulate all the flip-flop outputs in each module with regard to the clock, and calculate how often the flip-flop outputs change in relation to the clock.

Measuring the toggle rate becomes a more complex and a time-consuming process as module size increases. A toggle flip-flop has a 100% toggle rate, an 8-bit counter has 28%, and 16-bit counter has 14%.

Figure 4-68 is an example of how to calculate the toggle rate for a 4-bit counter.

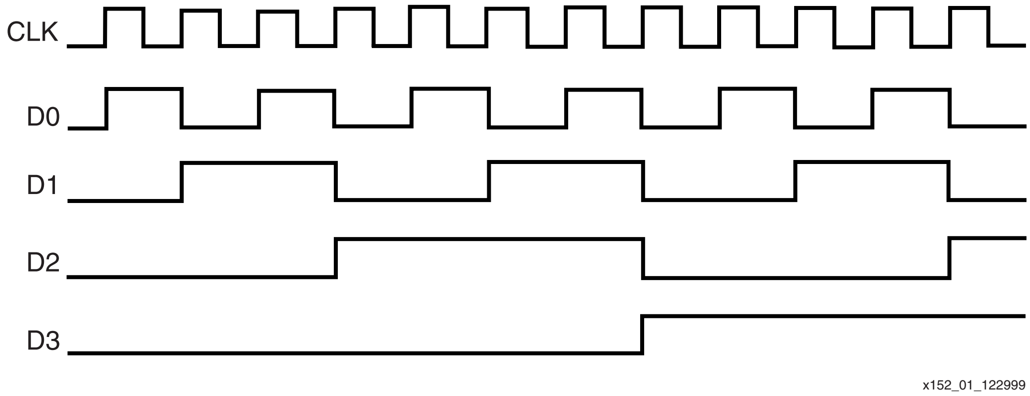


Figure 4-68: **Output Waveform of a 4-bit Counter**

Figure 4-68 shows the simulation wave form of a 4-bit counter. D0 stands for the LSB of the count, and D3 stands for the MSB. The toggle rate of D0 is 100% because D0 changes after every clock cycle. The toggle rate of D1 is 50% because D1 changes after every two clock cycles. The toggle rate of D2 is 25% because D2 changes after every four clock cycles. The toggle rate of D3 is 12.5% because D3 changes after every eight clock cycles. In this example, the average toggle rate of a 4-bit counter derived in the following equation is 46.875%.

$$\frac{(100 + 50 + 25 + 12.5)}{4} = 46.875$$

Routing Amount

There are three levels concerning the amount of routing to be used: low, medium, and high. The routing level is determined by the primary logic type of the module. Typical data path logic typically requires a low routing usage, random logic calls for a medium level, and control logic needs a high level.

Each designer needs to determine the routing that is most appropriate for each module.

Routing, which is determined by the type of logic in the module, is divided into three levels: low, medium, and high. Each designer needs to determine the routing that is most appropriate for each module.

1. Typical data path logic, which uses combinatorial logic such as multiplexers, adders, AND gates, and OR gates, usually requires a low routing usage. This also applies to any other signals that have one or two fanouts between structures.
2. Random logic, such as decoders, encoders, or any logic that has three to five fanouts, calls for a medium level of routing usage.
3. Control logic is typically logic with high fanout signals (excluding clocks) such as clock enables or reset signals. Control logic used in state machines also belongs to this category.

Block SelectRAM Power

Table 4-8 shows the data entries required for the Block SelectRAM Power section. This section is used to specify how many block RAMs are used and to determine their estimated power consumption. Before doing the calculation, designers can either treat all the RAMB16 cells as one module or break them down into smaller modules. RAMB16 is the base name for the Virtex-II Block SelectRAM component.

RAMB16 Cells

This is total number of Block Select RAMs (RAMB16 cells) used in each module.

Port A Frequency (MHz)

This is the frequency on the CLKA pin.

Port A Width

This is data width of DIA and DOA busses.

Port A Enable Rate (%)

This specifies how often ENA is enabled with respect to the clock. For a typical design, the rate may be 100% because the enable could be enabled all the time. For a FIFO design, the rate could be approximately 50% due to bursting of data into and out of the RAM.

Port B Frequency (MHz)

This is the frequency on the CLKB pin.

Port B Width

This is the data width of DIB and DOB busses.

Port B Enable Rate (%)

This specifies how often ENB is enabled with respect to the clock.

Table 4-8: **Block SelectRAM Power**

Module	RAMB16 Cells	Port A			Port B		
		Frequency (Mhz)	Width	Enable Rate (%)	Frequency (MHz)	Width	Enable Rate (%)
User Module 1	0	0	0	0	0%	0	0%
User Module 2	0	0	0	0	0%	0	0%
User Module 3	0	0	0	0	0%	0	0%
User Module 4	0	0	0	0	0%	0	0%
User Module 5	0	0	0	0	0%	0	0%
User Module 6	0	0	0	0	0%	0	0%
User Module 7	0	0	0	0	0%	0	0%
User Module 8	0	0	0	0	0%	0	0%

Digital Clock Management Power

Table 4-9 shows the data entries required for the DCM Power section and is used to estimate how much power DCMs consume. Only the clock input frequencies to the CLKIN pin needs to be entered.

Table 4-9: **Clock Delay Locked Loop Power**

Module	Clock Input Frequency (MHz)
User DCM 1	0
User DCM 2	0
User DCM 3	0
User DCM 4	0
User DCM 5	0
User DCM 6	0
User DCM 7	0
User DCM 8	0
User DCM 9	0
User DCM 10	0
User DCM 11	0
User DCM 12	0

Non-Registered Multiplier Power

The data entries for the Non-Registered Multiplier Power section are shown in Table 4-10. These entries are used to estimate Non-Registered Multiplier power consumption.

Table 4-10: **Data Entries for Non-Registered Multiplier Power**

Module	Mult18x18 Cell	Port A Width	Port B Width
User Module 1	0	0	0
User Module 2	0	0	0
User Module 3	0	0	0
User Module 4	0	0	0
User Module 5	0	0	0
User Module 6	0	0	0
User Module 7	0	0	0
User Module 8	0	0	0

Multi18x18 Cell

Multi18x18 cell is the total number of Multipliers used in each module.

Port A Width

Port A width is the data width of A busses.

Port B Width

Port B width is the data width of B busses.

Registered Multiplier Power

Data entries for the Registered Multiplier Power section are shown in Table 4-11. They are used to estimate Registered Multiplier power consumption.

Frequency

This is the frequency that the Multipliers operate at.

Multi18x18 Cell

Multi18x18 cell is the total number of Multipliers used in each module.

Port A Width

Port A width is the data width of A busses.

Port B Width

Port B width is the data width of B busses.

Average Toggle Rate

This is the toggle rate for the multiplier modules. This number can be obtained in the same way as obtaining the Average Toggle Rate in the CLB logic power section.

Table 4-11: **Data Entries for Registered Multiplier Power**

Module	Frequency (MHz)	Mult18x18 Cell	Port A Width	Port B Width	Average Toggle Rate
User Module 1	0	0	0	0	0
User Module 2	0	0	0	0	0
User Module 3	0	0	0	0	0
User Module 4	0	0	0	0	0
User Module 5	0	0	0	0	0
User Module 6	0	0	0	0	0
User Module 7	0	0	0	0	0
User Module 8	0	0	0	0	0

Input/Output Power

Table 4-12 shows the data entries for the Input/Output Power section used to estimate the power dissipation of the Inputs and Outputs. I/Os should be grouped into modules based on their I/O standard type. If the entire design has only one I/O standard type, all of the I/Os can be treated as one module. However, separating the I/Os into smaller modules makes it easier to obtain more accurate results.

Frequency (MHz)

This is the frequency of the module.

I/O Standard Type

This is the type of I/Os used in the module. Each module can have only one I/O standard type. I/O power is strongly influenced by the I/O standard used.

Inputs

This is the total number of the input buffers in each module.

Outputs

This is the total number of the output buffers in each module.

Average Output Toggle Rate (%)

This number can be obtained in the same way as obtaining the Average Toggle rate in the CLB Logic Power section.

Average Output Load (pF)

This specifies the average capacitive load on the outputs.

Table 4-12: **Data Entries for Input/Output Power**

Module	Frequency (MHz)	I/O Standard Type	Inputs	Outputs	Average Output Toggle Rate (%)	Average Output Load (pF)
User Module 1	0	LVTTL_12	0	0	0%	0
User Module 2	0	LVTTL_12	0	0	0%	0
User Module 3	0	LVTTL_12	0	0	0%	0
User Module 4	0	LVTTL_12	0	0	0%	0
User Module 5	0	LVTTL_12	0	0	0%	0
User Module 6	0	LVTTL_12	0	0	0%	0
User Module 7	0	LVTTL_12	0	0	0%	0
User Module 8	0	LVTTL_12	0	0	0%	0

Results

The results section of the power estimator are shown in Table 4-13. The four sections of the power estimator program independently estimate power consumption, and the results are displayed at the end of each section.

The total design power consumption is the summation of those, and is displayed at the very top of the program.

Table 4-13: **Power Estimator Results**

Target		Estimated Design Power Values (mW)					
Device	Package	Total Power	V_{CCINT} 1.5 V	V_{CCO} 3.3 V	V_{CCO} 2.5 V	V_{CCO} 1.5 V	Output Sink Power
XC2V500	FG256	0	0	0	0	0	0

Target Device

This refers to the target Virtex-II device size.

Note: No checking is done to verify that the module entries fit into the amount of resources available in the selected devices.

Target Package

This refers to the package of the device.

Note: No checking is done to verify that the selected device-package combination is valid.

Estimated Total Power

This section displays the total power consumption of the design. It is the summation of CLB Logic power, Block Select RAM power, Multiplier power, DCM power, and Input/Output power.

Estimated V_{CCINT} 1.5V Power

This section displays the total power consumption from the core supply voltage (V_{CCINT}). It does not include the power consumption from the input and output source voltage (V_{CCO}).

Estimated V_{CCAUX} 3.3V Power

This section displays the power consumption from auxiliary circuits.

Estimated V_{CCO} 3.3V Power

This section displays the V_{CCO} power consumption of 3.3 V applications. The I/O standards that use 3.3V V_{CCO} are LVTTL, LVCMOS33 PCI, SSTL3 Class I and II, and AGP2X.

Estimated V_{CCO} 2.5V Power

This section displays the V_{CCO} power consumption of 2.5 V applications. The supported I/O standards are LVCMOS25 and SSTL2 Class I and II.

Estimated V_{CCO} 1.5V Power

This section displays the V_{CCO} power consumption of 1.5 V applications. The supported I/O standards are LVCMOS15, and HSTL Class I, II, III, and IV.

Estimated Output Sink Power

This section displays the power consumption when sinking current to ground. The supported I/O standards are GTL and GTL+.

IBIS Models

The need for higher system performance leads to faster output transitions. Signals with fast transitions cannot be considered purely digital; it is important to understand their analog behavior for signal integrity analysis.

To simulate the signal integrity on printed circuit boards (PCB) accurately and solve design problems before the PCB is fabricated, models of the I/O characteristics are required. SPICE models are most frequently used for this purpose. A manufacturer's SPICE models, however, contain proprietary circuit-level information. Therefore, simpler models are devised to extract SPICE parameters for the proprietary information to remain protected. One such standard is the I/O Buffer Information Specification (IBIS) format originally suggested by Intel.

In the early 1990's, the IBIS Open Forum was formed and the first IBIS specification was written to promote tool independent I/O models for system signal integrity analysis.

IBIS is now the ANSI/EIA-656 and IEC 62014-1 standard. IBIS accurately describes the signal behavior of the interconnections without disclosing the actual technology and circuitry used to implement the I/O. The standard is basically a black-box approach to protecting proprietary information.

Using IBIS Models

IBIS models are used by designers for system-level analysis of signal integrity issues, such as the evaluation and matching of loads to drivers for ringing and ground bounce, examining effects of cross talk, and predicting RFI/EMI. It is useful in that complete designs can be simulated and evaluated before additional costs are incurred for PCB fabrication and assembly time.

IBIS models consist of look-up tables that predict the I/V characteristics and dV/dt of integrated circuit inputs and outputs when combined with the PCB wiring. The predictions are performed for the typical case, minimum case (weak transistors, low V_{CC}, hot temperatures), and maximum case (strong transistors, high V_{CC}, cold temperatures).

IBIS models have limitations in that they do not contain internal delay modeling and are limited in package modeling. IBIS models contain package parasitic information for simulation of ground bounce. Although the data is available within the model file, not all simulators are able to use the data to simulate ground bounce. Simulation results may not agree with the actual results due to package, die, and PCB ground plane modeling problems. Similarly, because simultaneous switching outputs (SSOs) are also difficult to model, only a first approximation is provided to the designer.

IBIS Generation

IBIS is generated either from SPICE simulations, or actual measurements of final devices. IBIS models that are derived from measurements do not have process corner information, unlike IBIS models that are derived from SPICE simulations. The measurements are of only a few parts, and the extremes of production are not represented by such a method.

SPICE is a transistor model based on detailed equations using device geometry, and properties of materials. A SPICE netlist of the CMOS buffer is required for V/I and dV/dt curve simulations. These SPICE simulations are then converted to IBIS format/syntax.

Advantages of IBIS

SPICE requires a greater knowledge of the internal workings of the circuits being modeled, and as such, errors may be made in simulation indicating a problem when there is none. IBIS models are easy to use, and because many of the decisions required for simulation parameters have been organized. IBIS simulations are faster compared to SPICE simulations, because IBIS does not contain circuit details. The voltage/current/time information provided in the IBIS model is only for the external nodes of the building block, making IBIS ideal for system-level interconnects design. Although IBIS models are not as accurate as SPICE models, they are entirely adequate for system-level analysis.

IBIS File Structure

An IBIS file contains two sections, the header and the model data for each component. One IBIS file can describe several devices. The following is the contents list in a typical IBIS file:

- IBIS Version
- File Name
- File Revision
- Component
- Package R/L/C
- Pin - name, model, R/L/C
- Model (i.e., 3-state)
- Temperature Range (typical, minimum, and maximum)
- Voltage Range (typical, minimum, and maximum)
- Pull-Up Reference
- Pull-Down Reference
- Power Clamp Reference
- Ground Clamp Reference
- V/I Tables for:
    - Pullup
    - Pulldown
    - Power Clamp
    - Ground Clamp
- Rise and Fall dV/dt for minimum, typical, and maximum conditions (driving 50 ohms)
- Package Model (optional) XXXX.pkg with RLC sections.

IBIS I/V and dV/dt Curves

A digital buffer can be measured in receive (3-state mode) and drive mode. IBIS I/V curves are based on the data of both these modes. The transition between modes is achieved by phasing in/out the difference between the driver and the receiver models, while keeping the receiver model constantly in the circuit.

The I/V curve range required by the IBIS specification is $-V_{CC}$ to (2 x V_{CC}). This wide voltage range exists because the theoretical maximum overshoot due to a full reflection is twice the signal swing. The ground clamp I/V curve must be specified over the range $-V_{CC}$ to V_{CC}, and the power clamp I/V curve must be specified from V_{CC} to (2 x V_{CC}).

The three supported conditions for the IBIS buffer models are typical values (required), minimum values (optional), and maximum values (optional). For CMOS buffers, the minimum condition is defined as high temperature and low supply voltage, and the maximum condition is defined as low temperature and high supply voltage.

An IBIS model of a digital buffer has four I/V curves:

- The pull-down I/V curve contains the mode data for the driver driving low. The origin of the curve is at 0 V for CMOS buffers.
- The pull-up I/V curve contains the mode data for the driver driving high. The origin of the curve is at the supply voltage (V_{CC} or V_{DD}).
- The ground clamp I/V curve contains receive (3-state) mode data, with the origin of the curve at 0 V for CMOS buffers.

- The power clamp I/V curve contains receive (3-state) mode data, with the origin of the curve at the supply voltage (V_{CC} or V_{DD}). For 3.3 V buffers that are 5 V tolerant, the power clamp is referenced to 5 V while the pullup is referenced to 3.3 V.

Ramp and dV/dt Curves

The Ramp keyword contains information on how fast the pull-up and pull-down transistors turn on/off. The dV/dt curves give the same information, while including the effects of die capacitance (C_comp). C_comp is the total die capacitance as seen at the die pad, excluding the package capacitance.

dV/dt curves describe the transient characteristics of a buffer more accurately than ramps. A minimum of four dV/dt curves is required to describe a CMOS buffer: pull-down ON, pull-up OFF, pull-down OFF, and pull-up ON. dV/dt curves incorporate the clock-to-out delay, and the length of the dV/dt curve corresponds to the clock speed at which the buffer is used. Each dV/dt curve has t = 0, where the pulse crosses the input threshold.

IBIS Simulations

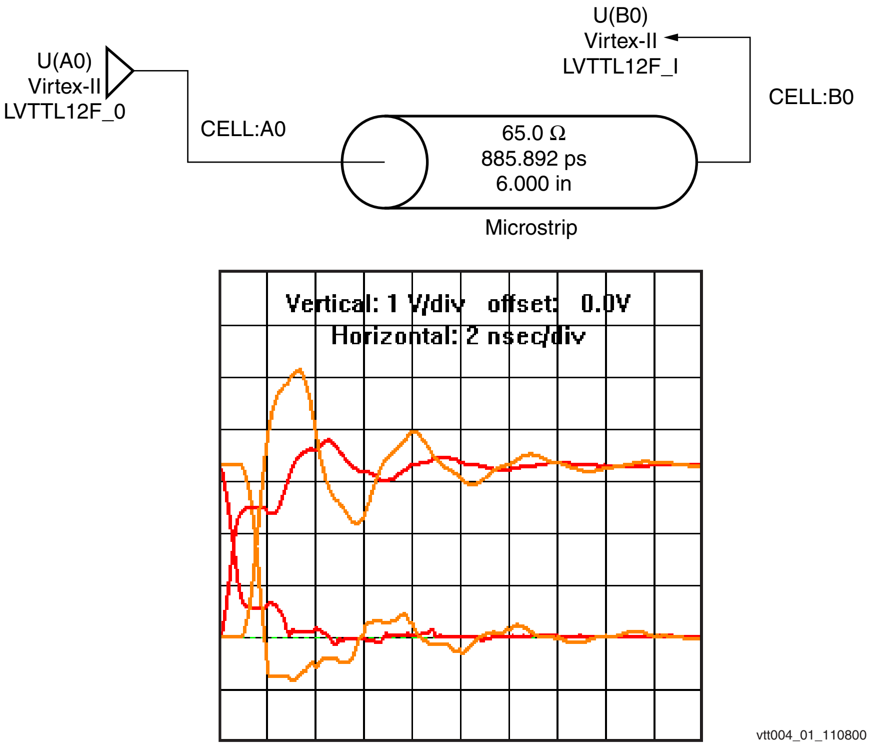


Figure 4-69: **Unterminated Example**

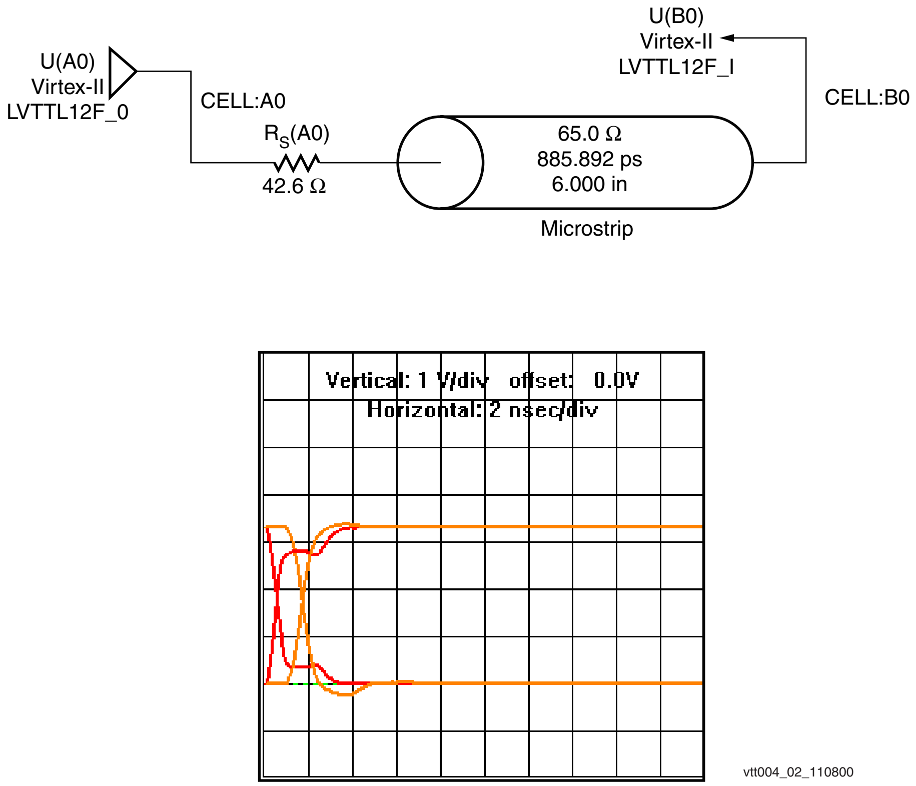


Figure 4-70: **Series Termination Example**

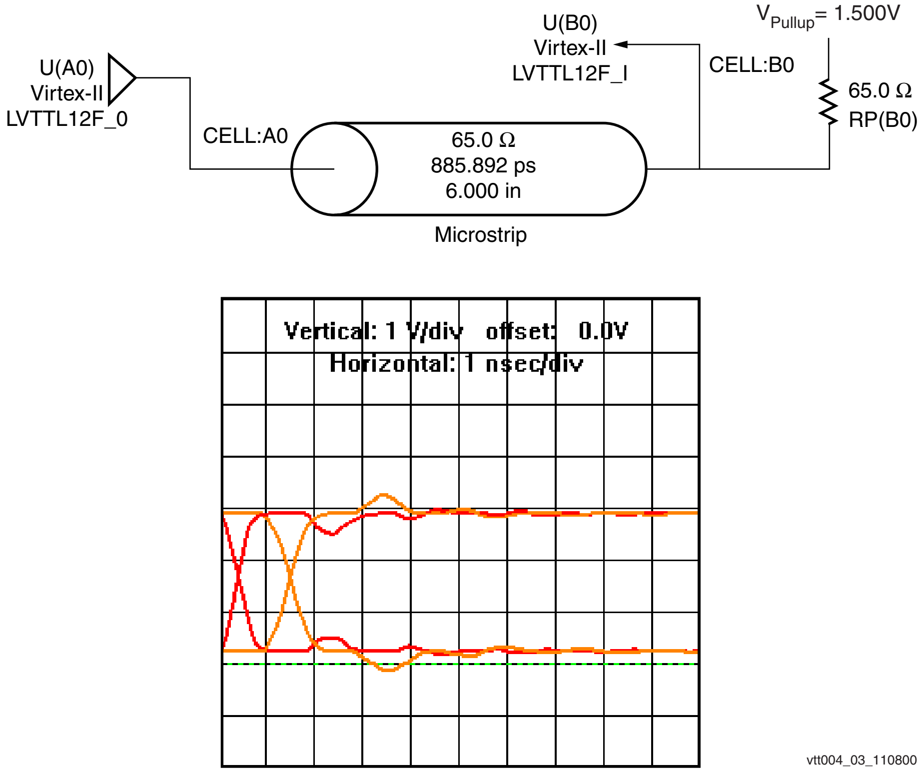


Figure 4-71: **Parallel Termination Example**

IBIS Simulators

Several different IBIS simulators are available today, and each simulator provides different results. An overshoot or undershoot of ±10% of the measured result is tolerable. Differences between the model and measurements occur, because not all parameters are modeled. Simulators for IBIS models are provided by the following vendors:

- Cadence
- Avanti Corporation
- Hyperlynx
- Mentor
- Microsim
- Intusoft
- Veribest
- Viewlogic

Xilinx IBIS Advantages

Xilinx provides preliminary IBIS files before working silicon has been verified (before tape out), as well as updated versions of IBIS files after the ICs are verified. Preliminary IBIS files are generated from SPICE models before working silicon has been verified. After the IC (device) is verified, appropriate changes are made to the existing IBIS files. These IBIS files are available at the following web site:

http://www.xilinx.com/support/sw_ibis.htm

IBIS Reference Web Site

http://www.eia.org/eig/ibis/ibis.htm

BSDL and Boundary Scan Models

Boundary scan is a technique that is used to improve the testability of ICs. With Virtex-II devices, registers are placed on I/Os that are connected together as a long shift register. Each register can be used to either save or force the state of the I/O. There are additional registers for accessing test modes.

The most common application for boundary scan is testing for continuity of the IC to the board. Some packages make visual inspection of solder joints impossible, e.g. BGA. The large number of I/Os available requires the use of such packages, and also increases the importance of testing. A large number of I/Os also means a long scan chain.

Test software is available to support testing with boundary scan. The software requires a description of the boundary scan implementation of the IC. The IEEE 1149.1 specification provides a language description for Boundary Scan Description Language (BSDL). Boundary scan test software accepts BSDL descriptions.

The IEEE 1149.1 spec also defines a 4 to 5 pin interface known as the JTAG interface. IEEE 1532 is a capability extension of IEEE 1149.1.

BSDL Files

Preliminary BSDL files are provided from the IC Design Process. Final BSDL files have been verified by an external third party test and verification vendor. The following are Virtex-II BDSL file names.

Virtex-II BSDL File Names	
XC2V40_CS144.BSD	XC2V2000_FF896.BSD
XC2V40_FG256.BSD	XC2V2000_BG575.BSD
XC2V80_CS144.BSD	XC2V2000_BG728.BSD
XC2V80_FG256.BSD	XC2V2000_BF957.BSD
XC2V250_CS144.BSD	XC2V3000_FG676.BSD
XC2V250_FG256.BSD	XC2V3000_FF1152.BSD
XC2V250_FG456.BSD	XC2V3000_BG728.BSD
XC2V500_FG256.BSD	XC2V3000_BF957.BSD
XC2V500_FG456.BSD	XC2V4000_FF1152.BSD
XC2V1000_FG256.BSD	XC2V4000_FF1517.BSD
XC2V1000_FG456.BSD	XC2V4000_BF957.BSD
XC2V1000_FF896.BSD	XC2V6000_FF1152.BSD
XC2V1000_BG575.BSD	XC2V6000_FF1517.BSD
XC2V1500_FG676.BSD	XC2V6000 _BF957.BSD
XC2V1500_FF896.BSD	XC2V8000_FF1152.BSD
XC2V1500_BG575.BSD	XC2V8000_FF1517.BSD
XC2V2000_FG676.BSD	XC2V8000_BF957.BSD

Appendix A

Application Notes

This section briefly describes relevant application notes. The latest versions of these documents are available online (at www.xilinx.com).

Memory Application Notes for Virtex-II Devices:

XAPP252: SigmaRAM DDR SRAM Interface for Virtex-II Devices

The SigmaRAM consortium, comprised of seven SRAM makers, defined SDR and DDR versions of separate I/O SRAM specifications. This application note describes the implementation of a Double Data Rate (DDR) SigmaRAM interface with a GS8170DxxB-333 device proposed by GSI Technology. This reference interface, implemented for Virtex-II FPGAs, is targeted to support user configurable mode, late write, and early write features of SigmaRAM modules. The current GSI interface can be synthesized to produce a 333 MHz data rate using the DDR mode with a 72-bit wide bus.

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP253: DDR SDRAM Controller for Virtex-II Devices

DDR (Double Data Rate) SDRAM is an enhancement to standard SDRAM. It activates output on both the rising and falling edge of the system clock rather than on just the rising edge, potentially doubling the output.

The DDR, DCM, and SelectI/O™ features in the Virtex™-II architecture make it the perfect choice for implementing a controller of a Double Data Rate (DDR) SDRAM. The Digital Clock Manager (DCM) provides the required Delay Locked Loop (DLL), Digital Phase Shift (DPS), and Digital Frequency Synthesis (DFS) functions. This application note describes a controller design for a 16-bit DDR SDRAM. The application note and reference design are enhanced versions of XAPP200 targeted to the Virtex-II series of FPGAs. At a clock rate of 133 MHz, 16-bit data changes at both clock edges. The reference design is fully synthesizable and achieves 133 MHz performance with automatic place and route tools.

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP254: SiberCAM Interface for Virtex-II Devices

SiberCAM is a Content-Addressable Memory (CAM) for use with ternary data of variable widths. SiberCAM Ultra-2M is a full family of large capacity CAMs in a single device: array configurations of 64k x 36, 32k x 72, 16k x 144, and 8k x 288 are all supported in one 2M - ternary density chip. In addition, the chip can be configured for variable-width operation, where any combination of 36-, 72-, 144-, and 288- bit entries are stored efficiently on the same chip, without wasting any storage resources. All device configurations are register selectable.

Ternary (3-state: 0, 1, or X) data is stored at addresses inside the SiberCAM module using maintenance operations. Search data is presented to the SiberCAM module through the search data port. After several clock cycles, an address containing data that best matches the data inside the SiberCAM module is returned on the search address (results) port.

An RTL reference design is provided that demonstrates the interface between a 32-bit host and a SiberCAM module, or cascade of SiberCAM modules. This Verilog/VHDL code is fully synthesizable, and the design is implemented using a Virtex-II FPGA. The DDR registers in Virtex-II devices are used for bursting data streams into the CAM.

This example interface demonstrates a way to initiate searches, obtain search results, and perform maintenance operations on SiberCAM modules, via a single interface from a host system with 32-bit access. Whether using the separate maintenance port or the search data port in two-port mode, the SiberCAM module expects maintenance operations to be performed in 36-bit/72-bit multiplexed quantities. This interface provides a mechanism to perform these operations using a 32-bit interface that resembles an SRAM.

XAPP256: FIFOs Using Virtex-II Shift Registers

The RAM-based shift registers available in Virtex-II devices ideally serve to build synchronous FIFOs. FIFOs using the SRL16 shift registers are very flexible for cascading together FIFOs of any width (1-bit) and depth (in multiples of 16). This application note describes a synchronous FIFO built using the SRL16 shift registers. It includes synthesizable code for configuring FIFOs of any desired width and depth.

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP257: Asynchronous FIFO in Virtex-II Devices

Asynchronous FIFOs with independent clocks for writing and reading are a popular method of transferring data across asynchronous clock-domain boundaries. The true dual-port operation of the Virtex-II block RAMs is ideal for implementing asynchronous FIFOs. In this application, each block RAM is configured as a 18-bit wide, 1024-deep RAM, with independent addressing, clocking, and clock enable for both ports, one write port and one read port. Clock cycle time for either clock can be as short as 5 ns. This application note concentrates on the remaining tasks, generating the Grey-coded write and read addresses, and generating the FULL and EMPTY control flags.

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP258: FIFOs Using Virtex-II Block RAM

This application note uses the fully synchronous dual-ported RAMs with 18K of memory cells available in Virtex-II devices. These blocks are ideal for FIFO applications, and each port can be configured independently as 16K x 1, 8K x 2, 4K x 4, 2K x 9, 1K x 18, or 512 x 36.

This application note describes a common-clock (synchronous) version and an independent-clock (asynchronous) version of a 511 x 36 FIFO, with the depth and width being adjustable within the Verilog or VHDL code. The size of the FIFO is 511 x 36 instead of 512 x 36 since one address is dropped out of the FIFO in order to provide distinct Empty/Full conditions. First the design for a 511 x 36 FIFO with common Read and Write clocks is described, and then the design changes required for the more difficult case of independent Read and Write clocks are presented.

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP260: Using Block RAM for High Performance Read/Write CAMs

CAM (Content Addressable Memory) offers increased data search speed. In various applications based on CAM, there are differing requirements for data organization and read/write performance. The design described in this application note is suited for small embedded CAMs with high-speed match and write requirements.

The Virtex-II block RAM can be used as a 32-word deep by 8-bit wide (32 x 8) CAM using the innovative design techniques described in this application note. A reference design provides parameterizable Verilog and VHDL code to cascade several block RAMs configured as 32 x 8 CAM. CAM speed is equivalent to the access time of a Virtex-II block RAM for a single clock cycle match (read), and a one or two clock cycles write. Medium size CAMs can be implemented in Virtex slices with different design techniques.

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP261: Data-Width Conversion FIFOs Using Virtex-II Block RAM Memory

This application note is an enhancement to XAPP258 (FIFOs using Virtex-II block RAM). In general, only the changes from XAPP258 will be covered. Four different data-width conversion FIFOs are described in this document. The first has a common clock with a 511 x 36 Write port and a 2044 x 9 Read port. The second has a common clock, a 2044 x 9 Write port, and a 511 x 36 Read port. The last two are similar FIFOs with independent Read and Write clocks.

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP262: QDR SRAM Interface for Virtex-II Devices

Virtex-II series FPGAs provide access to a variety of on-chip and off-chip RAM resources. In addition to on-chip distributed RAM and block SelectRAM features, Virtex-II FPGAs can interface with a variety of external high-speed memory devices. The combination of high-speed SelectI/O resources and on-chip Digital Clock Manager (DCM) circuits enable a high bandwidth interface to Quad Data Rate (QDR) architecture SRAM modules. This application note describes an interface implemented between a Cypress xx QDR SRAM module and a Virtex-II device.

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP266: FCRAM Controller for Virtex-II Devices

Delay-Locked Loop (DLL), SelectI/O, and enhanced DDR features make Virtex-II FPGAs the perfect choice for implementing a DDR FCRAM controller. This application note describes a controller implementation for Toshiba and Fujitsu FCRAMs. A reference design is provided with the following features:

- Utilization of the enhanced Virtex-II DDR interface
- Support for various FCRAM size offerings and scalable for next-generation FCRAM devices
- Maximum frequency of 154 MHz, with data throughput of 308 Mb/s per pin
- Programmable burst lengths
- Programmable column address strobe (CAS) latency
- Automatic refresh timer

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP267: Parity Generation and Validation in Virtex-II Devices

Ensuring correct parity helps to determine the validity of data transmitted and received. This application note shows how to generate and validate parity in a design using a Virtex-II device. The parity generation block generates parity from input data and stores it in the block RAM on the DIP bus which is available for storing parity. The parity validation block also generates parity and compares it against the value available from the block RAM to ensure data validity. The application note details 8-bit, 16-bit, and 32-bit parity checks.

Fully synthesizable Verilog/VHDL code is available for the reference design.

XAPP269: Fast CAM in Virtex-II Devices

Content Addressable Memories (CAM) allow fast searches for specific data in a memory. A wide variety of CAMs can be implemented in Virtex-II devices by using the basic LUT as a Shift Register (SRL16). Each CAM application will have different requirements. Designing CAMs with Virtex-II devices offers a flexible approach to specifying CAM depth and width.

This application note describes a fast CAM design finding a match in a single clock cycle. A methodology for designing flexible, small to medium size CAMs in Virtex-II slices. By using shift register primitives built into a Virtex-II slice, a reconfigurable LUT (two LUTs per slice) is used to implement a single-clock-cycle read CAM. A 4-bit CAM word fits into each LUT. A 32-word by 16-bit CAM would require 128 LUTs. The write operation uses the shift register mode and requires 16 clock cycles.

Fully synthesizable Verilog/VHDL code is available for the reference design.

Virtex-II Application Notes

XAPP251: Hot-Swapping Virtex-II Devices

Hot-swapping or hot insertion describes a potentially dangerous method of inserting an un-powered board into a power-on (hot) running system. There are several concerns: the insertion must not cause physical harm or permanent damage to the system or the inserted board, and the insertion must not cause data corruption or any transient system upsets. This application note describes the physical aspects of hot-inserting a Virtex™-II based card into a system or system backplane, using sequenced connectors, where VCC and GND mate well before any signal pins can mate.

XAPP268: Dynamic Clock Data Alignment

The phase alignment of clock and data inputs is an important feature available in Virtex-II devices. This application note explains the use of this feature by comparing the clock and data inputs and using the Digital Phase Shifter to align them.

Fully synthesizable Verilog/VHDL code is available for the reference design.

Appendix B

BitGen and PROMGen Switches and Options

Using BitGen

BitGen produces a bitstream for Xilinx device configuration. After the design has been completely routed, it is necessary to configure the device so that it can execute the desired function. The Xilinx bitstream necessary to configure the device is generated with BitGen. BitGen takes a fully routed NCD (Circuit Description) file as its input and produces a configuration bitstream—a binary file with a .bit extension.

The BIT file contains all of the configuration information from the NCD file defining the internal logic and interconnections of the FPGA, plus device-specific information from other files associated with the target device. The binary data in the BIT file can then be downloaded into the FPGA memory cells, or it can be used to create a PROM file (see Figure B-1).

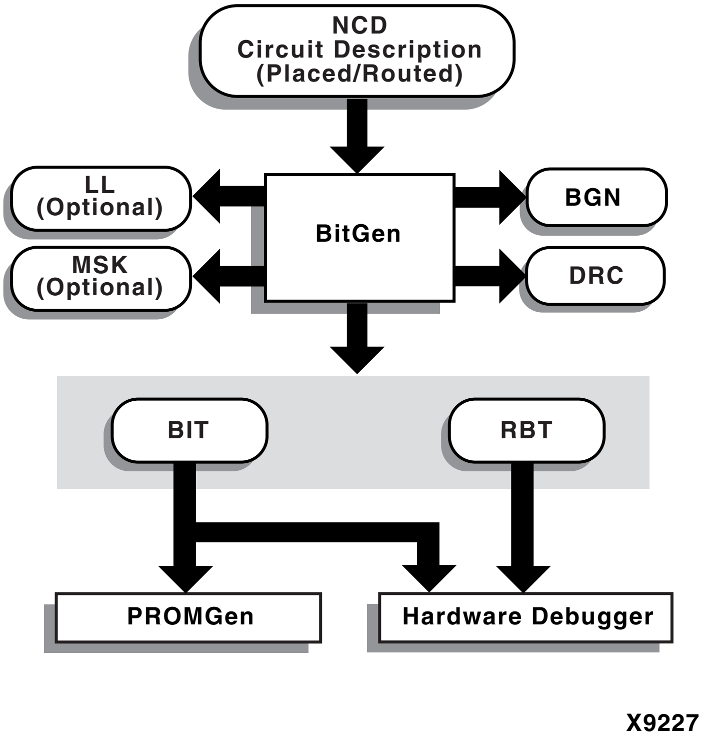


Figure B-1: **BitGen**

BitGen Syntax

The following syntax creates a bitstream from your NCD file.

```
bitgen [options] infile[.ncd] [outfile] [pcf_file]
```

options is one or more of the options listed in the "BitGen Options" on page 511.

Infile is the name of the NCD design for which you want to create the bitstream. You can specify only one design file, and it must be the first file specified on the command line.

You do not have to use an extension. If you do not, `.ncd` is assumed. If you do use an extension, it must be `.ncd`.

Outfile is the name of the output file. If you do not specify an output file name, BitGen creates one in the same directory as the input file. If you specify -l on the command line, the extension is .ll (see -l command line option). If you specify -m (see -m command line option), the extension is .msk. If you specify -b, the extension is .rbt. Otherwise the extension is .bit. If you do not specify an extension, BitGen appends one according to the aforementioned rules. If you do include an extension, it must also conform to the rules.

Pcf_file is the name of a physical constraints (PCF) file. BitGen uses this file to determine which nets in the design are critical for tiedown, which is not available for Virtex families. BitGen automatically reads the .pcf file by default. If the physical constraints file is the second file specified on the command line, it must have a .pcf extension. If it is the third file specified, the extension is optional; .pcf is assumed. If a .pcf file name is specified, it must exist, otherwise the input design name with a .pcf extension is read if that file exists.

A report file containing all BitGen's output is automatically created under the same directory as the output file. The report file has the same root name as the output file with a .bgn extension.

BitGen Files

This section describes input files that BitGen requires and output files that BitGen generates.

Input Files

Input to BitGen consists of the following files.

- NCD file—a physical description of the design mapped, placed and routed in the target device. The NCD file must be fully routed.
- PCF—an optional user-modifiable ASCII Physical Constraints File. If you specify a PCF file on the BitGen command line, BitGen uses this file to determine which nets in the design are critical for tiedown (not used for Virtex families).

Output Files

Output from BitGen consists of the following files.

- BIT file—a binary file with a .bit extension. The BIT file contains all of the configuration information from the NCD file defining the internal logic and interconnections of the FPGA, plus device-specific information from other files associated with the target device. The binary data in the BIT file can then be downloaded into the FPGA memory cells, or it can be used to create a PROM file (see "Using PROMGen" on page 515).
- RBT file—an optional "rawbits" file with an .rbt extension. The rawbits file is ASCII ones and zeros representing the data in the bitstream file. If you enter a -b option on the BitGen command line, an RBT file is produced in addition to the binary BIT file (see "-b (Create Rawbits File)" on page 511).
- LL file—an optional ASCII logic allocation file with a .ll extension. The logic allocation file indicates the bitstream position of latches, flip-flops, and IOB inputs and outputs. A .ll file is produced if you enter a -l option on the BitGen command line (see "-l (Create a Logic Allocation File)" on page 515).

- MSK file—an optional mask file with an .msk extension. This file is used to compare relevant bit locations for executing a readback of configuration data contained in an operating FPGA. A MSK file is produced if you enter a -m option on the BitGen command line (see "-m (Generate a Mask File)" on page 515).
- BGN file—a report file containing information about the BitGen run.
- DRC file—a Design Rule Check (DRC) file for the design. A DRC runs and the DRC file is produced unless you enter a -d option on the BitGen command line (see "-d (Do Not Run DRC)" on page 511).

BitGen Options

Following is a description of command line options and how they affect BitGen behavior.

-b (Create Rawbits File)

Create a "rawbits" (*file_name*.rbt) file. The rawbits file consists of ASCII ones and zeros representing the data in the bitstream file.

If you are using a microprocessor to configure a single FPGA, you can include the rawbits file in the source code as a text file to represent the configuration data. The sequence of characters in the rawbits file is the same as the sequence of bits written into the FPGA.

-d (Do Not Run DRC)

Do not run DRC (Design Rule Check). Without the -d option, BitGen runs a DRC and saves the DRC results in two output files: the BitGen report file (*file_name*.bgn) and the DRC file (*file_name*.drc). If you enter the -d option, no DRC information appears in the report file and no DRC file is produced.

Running DRC before a bitstream is produced detects any errors that could cause the FPGA to malfunction. If DRC does not detect any errors, BitGen produces a bitstream file (unless you use the -j option described in the "-j (No BIT File)" on page 515).

-f (Execute Commands File)

-f *command_file*

The -f option executes the command line arguments in the specified *command_file*.

-g (Set Configuration)

-g *option***:***setting*

The -g option specifies the startup timing and other bitstream options for Xilinx FPGAs. The settings for the -g option depend on the design's architecture. These options have the following syntax.

Compress

Enable bitstream compression using multiple frame writes (MFW).

Readback

This allows the user to perform Readback by the creating the necessary bitstream (**.rbb** file).

CRC

Virtex-II allows the user to enable or disable the CRC checking. If CRC checking is disabled, a CBC (Constant Bit Check) is used instead.

Settings: Enable, Disable
Default: Enable

DebugBitstream

This option creates a modified bitstream which loads each frame individually, and places an LOUT write after each, for debugging purposes. This option should be used only in Master or Slave Serial downloads.

Settings: Yes, No

Default: No

ConfigRate

Virtex-II devices use an internal oscillator to generate CCLK when configuring in Master SelectMAP or Master Serial modes. This option sets the CCLK rate in MHz.

Settings: 4,5,6,7,8,10,13,15,20,26,30,34,41,45,51,55,60,130

Default: 4

StartupClk

The last few cycles of configuration is called the startup sequence. The startup sequence can be clocked by CCLK signal, a User clock (connected to the STARTUP block), or TCK (the JTAG clock).

Settings: CCLK, UserClk, JTAGClk

Default: CCLK

PowerdownStatus

This options allows the user to choose whether the DONE pin is used as the PowerDown pin after configuration.

Settings: Enable, Disable

Default: Enable

DCMShutdown

If the DCMShutdown option is enabled, the DCM resets if the SHUTDOWN and AGHIGH commands are performed.

Settings: Enable, Disable

Default: Enable

CclkPin

This option selects an internal pullup on the CCLK pin.

Settings: Pullnone, Pullup

Default: Pullup

DonePin

This option selects an internal pullup on the DONE pin.

Settings: Pullnone, Pullup

Default: Pullup

M0Pin

This option selects an internal pullup or pulldown on the M0 (Mode 0) pin.

Settings: Pullnone, Pullup, Pulldown

Default: Pullup

M1Pin

This option selects an internal pullup or pulldown on the M1 (Mode 1) pin.

Settings: Pullnone, Pullup, Pulldown

Default: Pullup

M2Pin

This option selects an internal pullup or pulldown on the M2 (Mode 2) pin.

Settings: Pullnone, Pullup, Pulldown

Default: Pullup

ProgPin

This options selects an internal pullup on the PROGRAM pin.

Settings: Pullnone, Pullup

Default: Pullup

TckPin

This option selects an internal pullup or pulldown on the TCK (JTAG Clock) pin.

Settings: Pullnone, Pullup, Pulldown

Default: Pullup

TdiPin

This option selects an internal pullup or pulldown on the TDI (JTAG Input) pin.

Settings: Pullnone, Pullup, Pulldown

Default: Pullup

TdoPin

This option selects an internal pullup or pulldown on the TDO (JTAG Output) pin.

Settings: Pullnone, Pullup, Pulldown

Default: Pullnone

TmsPin

This option selects an internal pullup or pulldown on the TMS (JTAG Mode Select) pin.

Settings: Pullnone, Pullup, Pulldown

Default: Pullup

UnusedPin

This option selects an internal pullup or pulldown on all unused I/Os.

Settings: Pullnone, Pullup, Pulldown

Default: Pulldown

GWE_cycle

Selects the startup phase that asserts the internal write enable to flip-flops, LUT RAMs, shift registers, and BRAMs. Before the startup phase both BRAM writing and reading are disabled. The Done setting asserts GWE when the DoneIn signal is high. DoneIn is either the value of the DONE pin or a delayed version if DonePipe=Yes. The Keep setting is used to keep the current value of the GWE signal.

Settings: 1, 2, 3, 4, 5, 6, Done, Keep

Default: 6

GTS_cycle

Selects the startup phase that releases the internal 3-state control to the I/O buffers. The Done setting releases GTSA when the DoneIn signal is high. DoneIn is either the value of the DONE pin or a delayed version if DonePipe=Yes. The Keep setting is used to keep the current value of the GTS signal.

Settings: 1, 2, 3, 4, 5, 6, Done, Keep

Default: 5

LCK_cycle

Selects the startup phase to wait until DCM locks are asserted.

Settings: 0, 1, 2, 3, 4, 5, 6, NoWait

Default: NoWait

MATCH_cycle

Selects the startup phase to wait until DCI locks are asserted.

Settings: 0, 1, 2, 3, 4, 5, 6, NoWait

Default: NoWait

DONE_cycle

Selects the startup phase that activates the FPGA DONE signal. DONE is delayed when DonePipe=Yes.

Settings: 1, 2, 3, 4, 5, 6

Default: 4

Persist

This option is needed for Readback and Partial Reconfiguration using the configuration pins. If Persist=Yes, all the configuration pins used retain their function. Which configuration pins are persisted is determined by the mode pin settings. If a serial mode is chosen, the persisted pins would be $\overline{\text{INIT}}$, DOUT, and DIN. If a SelectMAP mode is chosen, the persisted pins would be $\overline{\text{INIT}}$, BUSY, D0-D7, $\overline{\text{CS}}$, and $\overline{\text{WRITE}}$.

Settings: Yes, No

Default: No

DriveDone

This option actively drives the DONE pin high as opposed to an open-drain driver. Take care when setting DriveDone=Yes in daisy chain applications.

Settings: Yes, No

Default: No

DonePipe

This option is intended for use with FPGAs being set up in a high-speed daisy chain configuration. When set to Yes, the FPGA waits on the DONE pin, and waits for the first StartupClk edge before moving to the Done state.

Settings: Yes, No

Default: No

Security

This options selects the level of bitstream security. Selecting Level 1 disables Readback, and selecting Level 2 disables Readback and reconfiguration.

Settings: Level1, Level2, None

Default: None

UserID

The user can enter up to an 8-digit hexadecimal code (32-bit value) in the UserID register. You can use the register to identify implementation or design revisions.

Settings: <any 8-digit hex string>

Default: 0xFFFFFFFF

-h or -help (Command Usage)

-h *architecture*

Displays a usage message for BitGen. The usage message displays all available options for BitGen operating on the specified *architecture*.

-j (No BIT File)

Do not create a bitstream file (.bit file). This option is generally used when you want to generate a report without producing a bitstream. For example, if you wanted to run DRC without producing a bitstream file, you would use the -j option.

Note: The .msk or .rbt files might still be created.

-l (Create a Logic Allocation File)

This option creates an ASCII logic allocation file (*design*.**ll**) for the selected design. The logic allocation file indicates the bitstream position of latches, flip-flops, and IOB inputs and outputs.

In some applications, you may want to observe the contents of the FPGA internal registers at different times. The file created by the -l option helps you identify which bits in the current bitstream represent outputs of flip-flops and latches. Bits are referenced by frame and bit number within the frame.

The Hardware Debugger uses the **design.ll** file to locate signal values inside a readback bitstream.

-m (Generate a Mask File)

Creates a mask file. This file is used to compare relevant bit locations for executing a readback of configuration data contained in an operating FPGA.

-w (Overwrite Existing Output File)

Enables you to overwrite an existing BIT, LL, MSK, or RBT output file.

Using PROMGen

The PROMGen program is compatible with the following families.

- Virtex/Virtex-E/Virtex-II

PROMGen formats a BitGen-generated configuration bitstream (BIT) file into a PROM format file (Figure B-2).

The PROM file contains configuration data for the FPGA device. PROMGen converts a BIT file into one of three PROM formats: MCS-86 (Intel), EXORMAX (Motorola), or TEKHEX (Tektronix). It can also generate a Hex file format.

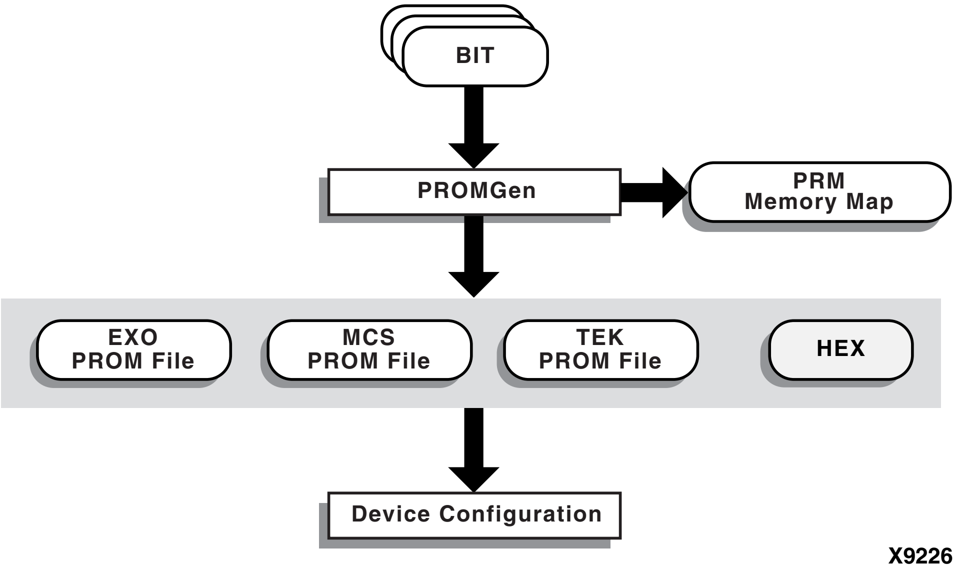


Figure B-2: **PROMGen**

There are two functionally equivalent versions of PROMGen. There is a stand-alone version you can access from an operating system prompt. You can also access an interactive version, called the PROM File Formatter, from inside the Design Manager for Alliance or the Project Manager in Foundation. This chapter describes the stand-alone version; the interactive version is described in the *PROM File Formatter Guide*.

You can also use PROMGen to concatenate bitstream files to daisy-chain FPGAs.

Note: If the destination PROM is one of the Xilinx Serial PROMs, you are using a Xilinx PROM Programmer, and the FPGAs are not being daisy-chained, it is not necessary to make a PROM file. See the *Hardware User Guide* for more information about daisy-chained designs

PROMGen Syntax

Use the following syntax to start PROMGen from the operating system prompt:

```
promgen [options]
```

Options can be any number of the options listed in"PROMGen Options" on page 517. Separate multiple options with spaces.

PROMGen Files

This section describes the PROMGen input and output files.

Input Files

The input to PROMGEN consists of BIT files— one or more bitstream files. BIT files contain configuration data for an FPGA design.

Output Files

Output from PROMGEN consists of the following files.

- PROM files—The file or files containing the PROM configuration information. Depending on the PROM file format used by the PROM programmer, you can output a TEK, MCS, or EXO file. If you are using a microprocessor to configure your devices, you can output a HEX file, containing a hexadecimal representation of the bitstream.
- PRM file—The PRM file is a PROM image file. It contains a memory map of the output PROM file. The file has a **.prm** extension.

Bit Swapping in PROM Files

PROMGen produces a PROM file in which the bits within a byte are swapped compared to the bits in the input BIT file. Bit swapping (also called "bit mirroring") reverses the bits within each byte, as shown in Figure B-3.

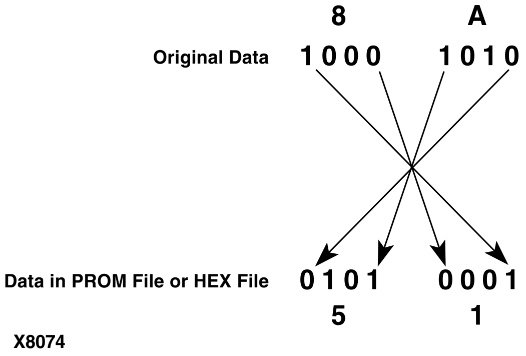


Figure B-3: **Bit Swapping**

In a bitstream contained in a BIT file, the Least Significant Bit (LSB) is always on the left side of a byte. But when a PROM programmer or a microprocessor reads a data byte, it identifies the LSB on the right side of the byte. In order for the PROM programmer or microprocessor to read the bitstream correctly, the bits in each byte must first be swapped so they are read in the correct order.

In this release of the Xilinx Development System, the bits are swapped for all of the PROM formats: MCS, EXO, and TEK. For a HEX file output, bit swapping is on by default, but it can be turned off by entering a -b PROMGen option that is available only for HEX file format.

PROMGen Options

This section describes the options that are available for the PROMGen command.

-b (Disable Bit Swapping—HEX Format Only)

This option only applies if the -p option specifies a HEX file for the output of PROMGen. By default (no -b option), bits in the HEX file are swapped compared to bits in the input BIT files. If you enter a -b option, the bits are not swapped. Bit swapping is described in "Bit Swapping in PROM Files" on page 517.

-c (Checksum)

```
promgen -c
```

The -c option generates a checksum value appearing in the .prm file. This value should match the checksum in the prom programmer. Use this option to verify that correct data was programmed into the prom.

-d (Load Downward)

```
promgen -d hexaddress0 filename filename...
```

This option loads one or more BIT files from the starting address in a downward direction. Specifying several files after this option causes the files to be concatenated in a daisy chain. You can specify multiple -d options to load files at different addresses. You must specify this option immediately before the input bitstream file.

The multiple file syntax is as follows:

```
promgen -d hexaddress0 filename filename...
```

The multiple **-d** options syntax is as follows:

```
promgen -d hexaddress1 filename -d hexaddress2 filename...
```

-f (Execute Commands File)

```
-f command_file
```

The -f option executes the command line arguments in the specified *command_file*.

-help (Command Help)

This option displays help that describes the PROMGen options.

-l option (Disable Length Count)

```
promgen -l
```

The -l option disables the length counter in the FPGA bitstream. It is valid only for 4000EX, 4000XL, 4000XLA, 4000XV, and SpartanXL Devices. Use this option when chaining together bitstreams exceeding the 24 bit limit imposed by the length counter.

-n (Add BIT FIles)

```
-n file1[.bit] file2[.bit]...
```

This option loads one or more BIT files up or down from the next available address following the previous load. The first -n option *must* follow a -u or -d option because -n does not establish a direction. Files specified with this option are not daisy-chained to previous files. Files are loaded in the direction established by the nearest prior -u, -d, or -n option.

The following syntax shows how to specify multiple files. When you specify multiple files, PROMGen daisy-chains the files.

```
promgen -d hexaddress file0 -n file1 file2...
```

The following syntax when using multiple -n options prevents the files from being daisy-chained:

```
promgen -d hexaddress file0 -n file1 -n file2...
```

-o (Output File Name)

```
-o file1[.ext] file2[.ext]...
```

This option specifies the output file name of a PROM if it is different from the default. If you do not specify an output file name, the PROM file has the same name as the first BIT file loaded.

ext is the extension for the applicable PROM format.

Multiple file names may be specified to split the information into multiple files. If only one name is supplied for split PROM files (by you or by default), the output PROM files are named *file*_#.ext, where *file* is the base name, # is 0, 1, etc., and *ext* is the extension for the applicable PROM format.

```
promgen -d hexaddress file0 -o filename
```

-p (PROM Format)

```
-p {mcs | exo | tek | hex}
```

This option sets the PROM format to one of the following: MCS (Intel MCS86), EXO (Motorola EXORMAX), TEK (Tektronix TEKHEX). The option may also produce a HEX file, which is a hexadecimal representation of the configuration bitstream used for microprocessor downloads. If specified, the -p option must precede any -u, -d, or -n options. The default format is MCS.

-r (Load PROM File)

```
-r promfile
```

This option reads an existing PROM file as input instead of a BIT file. All of the PROMGen output options may be used, so the -r option can be used for splitting an existing PROM file into multiple PROM files or for converting an existing PROM file to another format.

-s (PROM Size)

```
-s promsize1 promsize2...
```

This option sets the PROM size in kilobytes. The PROM size must be a power of 2. The default value is 64 kilobytes. The -s option must precede any -u, -d, or -n options.

Multiple *promsize* entries for the -s option indicates the PROM will be split into multiple PROM files.

Note: PROMGen PROM sizes are specified in bytes. *The Programmable Logic Data Book* specifies PROM sizes in bits for Xilinx serial PROMs (see -x option).

-u (Load Upward)

```
-u hexaddress0 filename1 filename2...
```

This option loads one or more BIT files from the starting address in an upward direction. When you specify several files after this option, PROMGen concatenates the files in a daisy chain. You can load files at different addresses by specifying multiple -u options.

This option must be specified immediately before the input bitstream file.

-x (Specify Xilinx PROM)

```
-x xilinx_prom1 xilinx_prom2...
```

The -x option specifies one or more Xilinx serial PROMs for which the PROM files are targeted. Use this option instead of the -s option if you know the Xilinx PROMs to use.

Multiple xilinx*_prom* entries for the -x option indicates the PROM will be split into multiple PROM files.

Examples

To load the file test.bit up from address 0x0000 in MCS format, enter the following information at the command line.

```
promgen -u 0 test
```

To daisy-chain the files test1.bit and test2.bit up from address 0x0000 and the files test3.bit and test4.bit from address 0x4000 while using a 32K PROM and the Motorola EXORmax format, enter the following information at the command line.

```
promgen -s 32 -p exo -u 00 test1 test2 -u 4000 test3 test4
```

To load the file test.bit into the PROM programmer in a downward direction starting at address 0x400, using a Xilinx XC1718D PROM, enter the following information at the command line.

```
promgen -x xc1718d -d 0x400 test
```

To specify a PROM file name that is different from the default file name enter the following information at the command line.

```
promgen options filename -o newfilename
```

Appendix C

XC18V00 Series PROMs

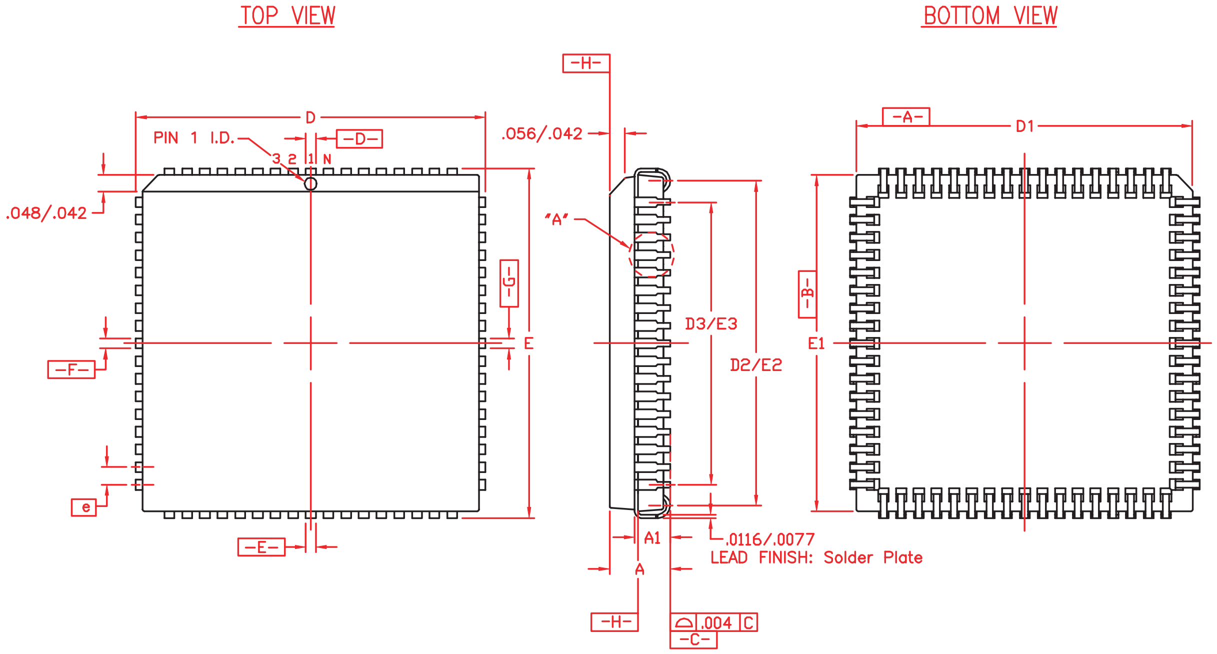
This appendix contains package specifications for the XC18V00 Series of In-System Programmable Configuration PROMs, as well as the XC18V00 Series product specification (DS026). The latest version of this information is available online (at www.xilinx.com).

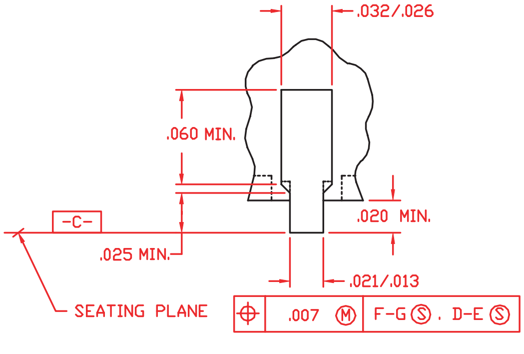
PROM Package Specifications

This section contains specifications for the following Virtex-II packages:

- PC20-84 Specification
- SO20 Specification
- VQ44 Specification

PC20-84 Specification





DETAIL "A"

SYMBOL	INCHES									
	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.
A	.165	.180	.165	.180	.165	.180	.165	.200	.165	.200
A_1	.090	.120	.099	.110	.090	.120	.090	.130	.090	.130
D/E	.385	.395	.485	.495	.685	.695	.985	.995	1.185	1.195
D_1/E_1	.350	.356	.450	.456	.650	.656	.950	.958	1.150	1.158
D_2/E_2	.290	.330	.390	.430	.590	.630	.890	.930	1.090	1.130
D_3/E_3	.200 REF.		.300 REF.		.500 REF.		.800 REF.		1.000 REF.	
e	.050 BSC		.050 BSC		.050 BSC		.050 BSC		.050 BSC	
N	20		28		44		68		84	

NOTES:

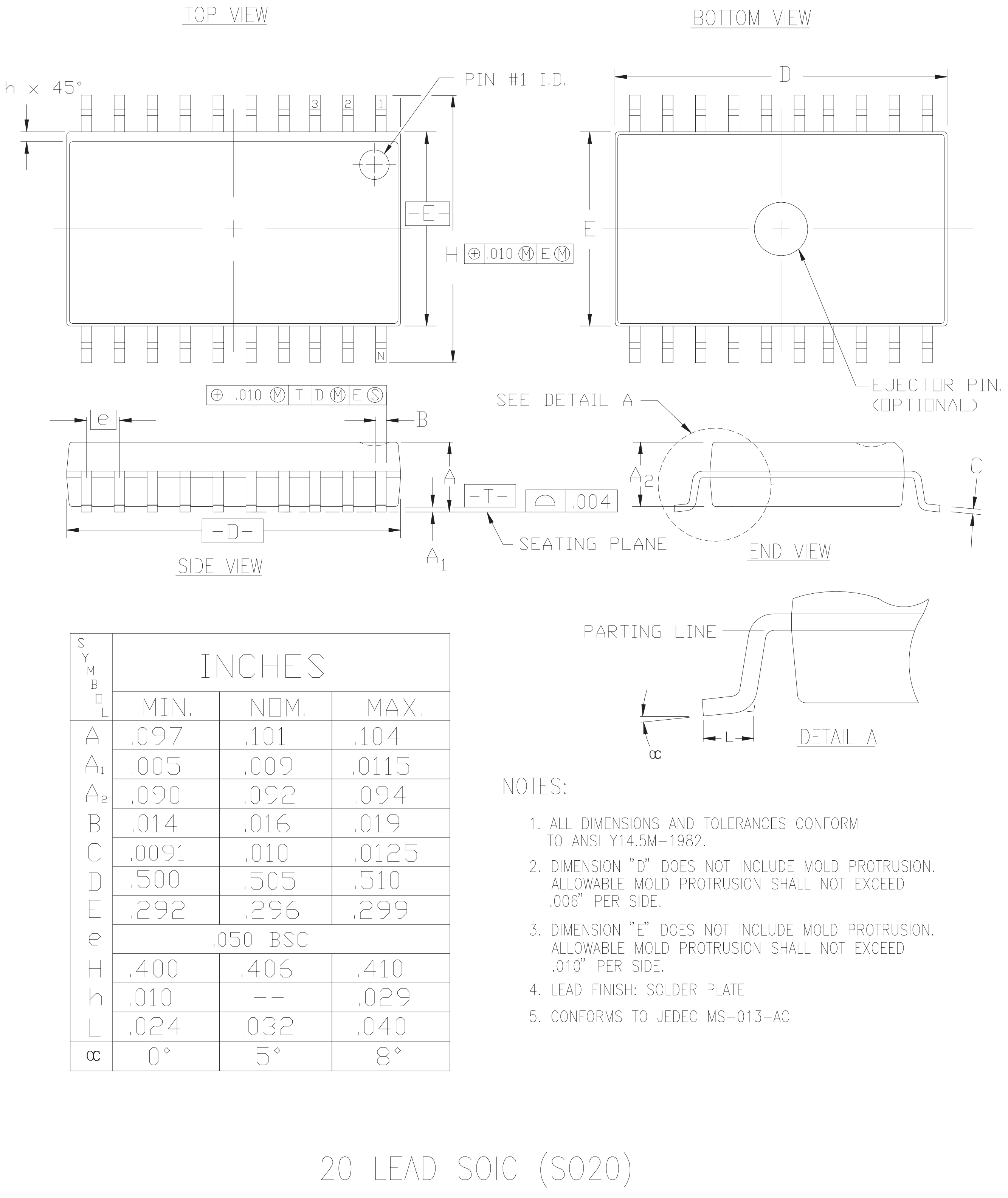
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1982.
2. DIMENSIONS 'D1' AND 'E1' DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS. MOLD FLASH OR PROTRUSIONS SHALL NOT EXCEED .010 PER SIDE.
3. 'N' IS NUMBER OF TERMINALS.
4. CONFORM TO JEDEC MO-047
5. TOP OF PACKAGE MAY BE SMALLER THAN BOTTOM BY .010".

20, 28, 44, 68 and 84-PIN PLCC (PC20 THRU PC84)

UG002_app_01_111600

Figure C-1: **PC20-84 Specification**

SO20 Specification



SYMBOL	INCHES		
	MIN.	NOM.	MAX.
A	.097	.101	.104
A_1	.005	.009	.0115
A_2	.090	.092	.094
B	.014	.016	.019
C	.0091	.010	.0125
D	.500	.505	.510
E	.292	.296	.299
e	.050 BSC		
H	.400	.406	.410
h	.010	--	.029
L	.024	.032	.040
α	0°	5°	8°

NOTES:

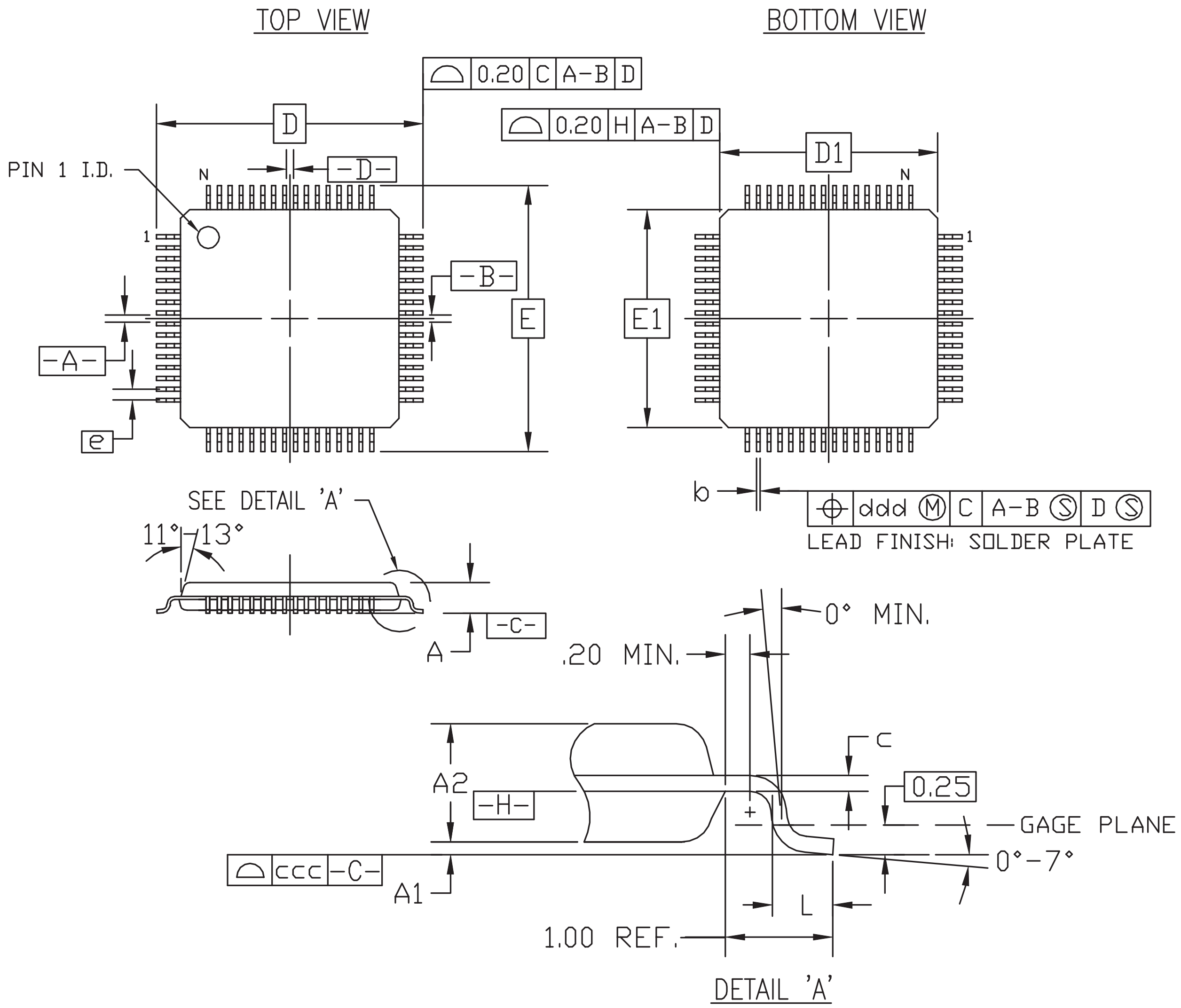
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1982.
2. DIMENSION "D" DOES NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION SHALL NOT EXCEED .006" PER SIDE.
3. DIMENSION "E" DOES NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION SHALL NOT EXCEED .010" PER SIDE.
4. LEAD FINISH: SOLDER PLATE
5. CONFORMS TO JEDEC MS-013-AC

20 LEAD SOIC (SO20)

UG002_app_02_111600

Figure C-2: **SO20 Specification**

VQ44 Specification



SYMBOL	VQ44			VQ64			VQ100		
	MILLIMETERS			MILLIMETERS			MILLIMETERS		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	~	~	1.20	~	~	1.20	~	~	1.20
A_1	0.05	~	0.15	0.05	0.10	0.15	0.05	0.10	0.15
A_2	0.95	1.00	1.05	0.95	1.00	1.05	0.95	1.00	1.05
D/E	12.00 BSC			12.00 BSC.			16.00 BSC.		
D_1/E_1	10.00 BSC			10.00 BSC.			14.00 BSC.		
b	0.30	0.37	0.45	0.17	0.22	0.27	0.17	0.22	0.27
c	0.09	~	0.20	0.09	~	0.20	0.09	~	0.20
e	0.80 BSC.			0.50 BSC.			0.50 BSC.		
L	0.45	0.60	0.75	0.45	0.60	0.75	0.45	0.60	0.75
ccc	~	~	0.10	~	~	0.08	~	~	0.08
ddd	~	~	0.20	~	~	0.08	~	~	0.08
N	44			64			100		
REF.	JEDEC MS-026-ACB			JEDEC MS-026-ACD			JEDEC MS-026-AED		

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1982.
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION SHALL NOT EXCEED 0.25mm PER SIDE.
3. THE TOP OF PACKAGE MAY BE SMALLER THAN THE BOTTOM OF PACKAGE BY 0.15mm.

44, 64, 100-PIN PLASTIC VERY THIN QFP (VQ44, VQ64, VQ100)

UG002_app_04_111600

Figure C-3: **VQ44 Specification**

XC18V00 Series of In-System Programmable Configuration PROMs

DS026 (v3.0) November 12, 2001

Product Specification

Features

- In-system programmable 3.3V PROMs for configuration of Xilinx FPGAs
    - Endurance of 20,000 program/erase cycles
    - Program/erase over full commercial/industrial voltage and temperature range
- IEEE Std 1149.1 boundary-scan (JTAG) support
- Simple interface to the FPGA
- Cascadable for storing longer or multiple bitstreams
- Low-power advanced CMOS FLASH process
- Dual configuration modes
    - Serial Slow/Fast configuration (up to 33 MHz)
    - Parallel (up to 264 Mb/s at 33 MHz)
- 5V tolerant I/O pins accept 5V, 3.3V and 2.5V signals
- 3.3V or 2.5V output capability
- Available in PC20, SO20, PC44 and VQ44 packages
- Design support using the Xilinx Alliance and Foundation series software packages.
- JTAG command initiation of standard FPGA configuration

Description

Xilinx introduces the XC18V00 series of in-system programmable configuration PROMs (Figure 1). Initial devices in this 3.3V family are a 4-megabit, a 2-megabit, a 1-megabit, a 512-Kbit, and a 256-Kbit PROM that provide an easy-to-use, cost-effective method for re-programming and storing large Xilinx FPGA or CPLD configuration bitstreams.

When the FPGA is in Master Serial mode, it generates a configuration clock that drives the PROM. A short access time after the rising CCLK, data is available on the PROM DATA (D0) pin that is connected to the FPGA D_{IN} pin. The FPGA generates the appropriate number of clock pulses to complete the configuration. When the FPGA is in Slave Serial mode, the PROM and the FPGA are clocked by an external clock.

When the FPGA is in Slave-Parallel or SelectMAP Mode, an external oscillator generates the configuration clock that drives the PROM and the FPGA. After the rising CCLK edge, data are available on the PROMs DATA (D0-D7) pins. The data is clocked into the FPGA on the following rising edge of the CCLK. Neither Slave-Parallel nor SelectMAP utilize a Length Count, so a free-running oscillator can be used.

Multiple devices can be concatenated by using the $\overline{\text{CEO}}$ output to drive the $\overline{\text{CE}}$ input of the following device. The clock inputs and the DATA outputs of all PROMs in this chain are interconnected. All devices are compatible and can be cascaded with other members of the family or with the XC17V00 one-time programmable Serial PROM family.

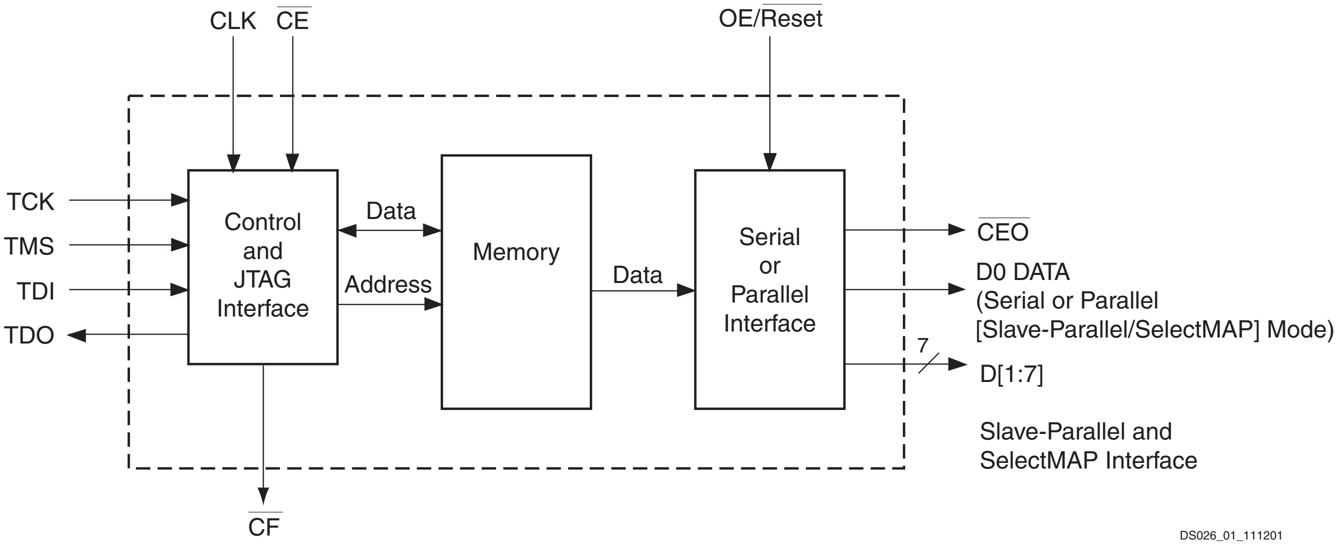


Figure 1: **XC18V00 Series Block Diagram**

© 2001 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at http://www.xilinx.com/legal.htm.
All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

Pinout and Pin Description

Table 1: **Pin Names and Descriptions (pins not listed are “no connect”)**

Pin Name	Boundary Scan Order	Function	Pin Description	44-pin VQFP	44-pin PLCC	20-pin SOIC and PLCC
D0	4	DATA OUT	D0 is the DATA output pin to provide data for configuring an FPGA in serial mode.	40	2	1
	3	OUTPUT ENABLE				
D1	6	DATA OUT	D0-D7 are the output pins to provide parallel data for configuring a Xilinx FPGA in Slave-Parallel/SelectMap mode.	29	35	16
	5	OUTPUT ENABLE				
D2	2	DATA OUT		42	4	2
	1	OUTPUT ENABLE				
D3	8	DATA OUT		27	33	15
	7	OUTPUT ENABLE				
D4	24	DATA OUT		9	15	7(1)
	23	OUTPUT ENABLE				
D5	10	DATA OUT		25	31	14
	9	OUTPUT ENABLE				
D6	17	DATA OUT		14	20	9
	16	OUTPUT ENABLE				
D7	14	DATA OUT		19	25	12
	13	OUTPUT ENABLE				
CLK	0	DATA IN	Each rising edge on the CLK input increments the internal address counter if both $\overline{\text{CE}}$ is Low and OE/$\overline{\text{RESET}}$ is High.	43	5	3
OE/$\overline{\text{RESET}}$	20	DATA IN	When Low, this input holds the address counter reset and the DATA output is in a high-impedance state. This is a bidirectional open-drain pin that is held Low while the PROM is reset. Polarity is NOT programmable.	13	19	8
	19	DATA OUT				
	18	OUTPUT ENABLE				
$\overline{\text{CE}}$	15	DATA IN	When $\overline{\text{CE}}$ is High, this pin puts the device into standby mode and resets the address counter. The DATA output pin is in a high-impedance state, and the device is in low power standby mode.	15	21	10

Table 1: **Pin Names and Descriptions (pins not listed are "no connect")** ***(Continued)***

Pin Name	Boundary Scan Order	Function	Pin Description	44-pin VQFP	44-pin PLCC	20-pin SOIC and PLCC
$\overline{\text{CF}}$	22 21	DATA OUT OUTPUT ENABLE	Allows JTAG CONFIG instruction to initiate FPGA configuration without powering down FPGA. This is an open-drain output that is pulsed Low by the JTAG CONFIG command.	10	16	7(1)
$\overline{\text{CEO}}$	11 12	DATA OUT OUTPUT ENABLE	Chip Enable Output ($\overline{\text{CEO}}$) is connected to the $\overline{\text{CE}}$ input of the next PROM in the chain. This output is Low when $\overline{\text{CE}}$ is Low and OE/$\overline{\text{RESET}}$ input is High, AND the internal address counter has been incremented beyond its Terminal Count (TC) value. When OE/$\overline{\text{RESET}}$ goes Low, $\overline{\text{CEO}}$ stays High until the PROM is brought out of reset by bringing OE/$\overline{\text{RESET}}$ High.	21	27	13
GND			GND is the ground connection.	6, 18, 28 & 41	3, 12, 24 & 34	11
TMS		MODE SELECT	The state of TMS on the rising edge of TCK determines the state transitions at the Test Access Port (TAP) controller. TMS has an internal 50K ohm resistive pull-up on it to provide a logic "1" to the device if the pin is not driven.	5	11	5
TCK		CLOCK	This pin is the JTAG test clock. It sequences the TAP controller and all the JTAG test and programming electronics.	7	13	6
TDI		DATA IN	This pin is the serial input to all JTAG instruction and data registers. TDI has an internal 50K ohm resistive pull-up on it to provide a logic "1" to the system if the pin is not driven.	3	9	4
TDO		DATA OUT	This pin is the serial output for all JTAG instruction and data registers. TDO has an internal 50K ohm resistive pull-up on it to provide a logic "1" to the system if the pin is not driven.	31	37	17
V_{CC}			Positive 3.3V supply voltage for internal logic and input buffers.	17, 35 & 38	23, 41 & 44	18 & 20
V_{CCO}			Positive 3.3V or 2.5V supply voltage connected to the output voltage drivers.	8, 16, 26 & 36	14, 22, 32 & 42	19

Notes:
1. Pin 7 is $\overline{\text{CF}}$ in Serial Mode, D4 in Slave-Parallel Mode for 20-pin packages.

Xilinx FPGAs and Compatible PROMs

Table 2 provides a list of Xilinx FPGAs and compatible PROMs.

Table 2: **Xilinx FPGAs and Compatible PROMs**

Device	Configuration Bits	XC18V00 Solution
XC2V40	360,160	XC18V512
XC2V80	635,360	XC18V01
XC2V250	1,697,248	XC18V02
XC2V500	2,761,952	XC18V04
XC2V1000	4,082,656	XC18V04
XC2V1500	5,659,360	XC18V04 + XC18V02
XC2V2000	7,492,064	2 of XC18V04
XC2V3000	10,494,432	3 of XC18V04
XC2V4000	15,660,000	4 of XC18V04
XC2V6000	21,849,568	5 of XC18V04 + XC18V02
XC2V8000	29,063,136	7 of XC18V04
XCV50	559,200	XC18V01
XCV100	781,216	XC18V01
XCV150	1,040,096	XC18V01
XCV200	1,335,840	XC18V02
XCV300	1,751,808	XC18V02
XCV400	2,546,048	XC18V04
XCV600	3,607,968	XC18V04
XCV800	4,715,616	XC18V04 + XC18V512
XCV1000	6,127,744	XC18V04 + XC18V02
XCV50E	630,048	XC18V01
XCV100E	863,840	XC18V01
XCV200E	1,442,106	XC18V02
XCV300E	1,875,648	XC18V02
XCV400E	2,693,440	XC18V04
XCV405E	3,430,400	XC18V04
XCV600E	3,961,632	XC18V04
XCV812E	6,519,648	2 of XC18V04
XCV1000E	6,587,520	2 of XC18V04
XCV1600E	8,308,992	2 of XC18V04
XCV2000E	10,159,648	3 of XC18V04
XCV2600E	12,922,336	4 of XC18V04
XCV3200E	16,283,712	4 of XC18V04
XC2S15	197,696	XC18V256
XC2S30	336,768	XC18V512
XC2S50	559,200	XC18V01
XC2S100	781,216	XC18V01
XC2S150	1,040,096	XC18V01
XC2S200	1,335,840	XC18V02
XC2S50E	630,048	XC18V01
XC2S100E	863,840	XC18V01
XC2S150E	1,134,528	XC18V02
XC2S200E	1,442,016	XC18V02
XC2S300E	1,875,648	XC18V02

Capacity

Devices	Configuration Bits
XC18V04	4,194,304
XC18V02	2,097,152
XC18V01	1,048,576
XC18V512	524,288
XC18V256	262,144

In-System Programming

In-System Programmable PROMs can be programmed individually, or two or more can be daisy-chained together and programmed in-system via the standard 4-pin JTAG protocol as shown in Figure 2. In-system programming offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices. The Xilinx development system provides the programming data sequence using either Xilinx JTAG Programmer software and a download cable, a third-party JTAG development system, a JTAG-compatible board tester, or a simple microprocessor interface that emulates the JTAG instruction sequence. The JTAG Programmer software also outputs serial vector format (SVF) files for use with any tools that accept SVF format and with automatic test equipment.

All outputs are held in a high-impedance state or held at clamp levels during in-system programming.

OE/$\overline{\text{RESET}}$

The ISP programming algorithm requires issuance of a reset that causes OE to go Low.

External Programming

Xilinx reprogrammable PROMs can also be programmed by the Xilinx HW-130 device programmer. This provides the added flexibility of using pre-programmed devices in board design and boundary-scan manufacturing tools, with an in-system programmable option for future enhancements and design changes.

Reliability and Endurance

Xilinx in-system programmable products provide a guaranteed endurance level of 20,000 in-system program/erase cycles and a minimum data retention of 20 years. Each device meets all functional, performance, and data retention specifications within this endurance limit.

Design Security

The Xilinx in-system programmable PROM devices incorporate advanced data security features to fully protect the programming data against unauthorized reading. Table 3 shows the security setting available.

The read security bit can be set by the user to prevent the internal programming pattern from being read or copied via JTAG. When set, it allows device erase. Erasing the entire device is the only way to reset the read security bit.

Table 3: **Data Security Options**

Default = Reset	Set
Read Allowed Program/Erase Allowed	Read Inhibited via JTAG Erase Allowed

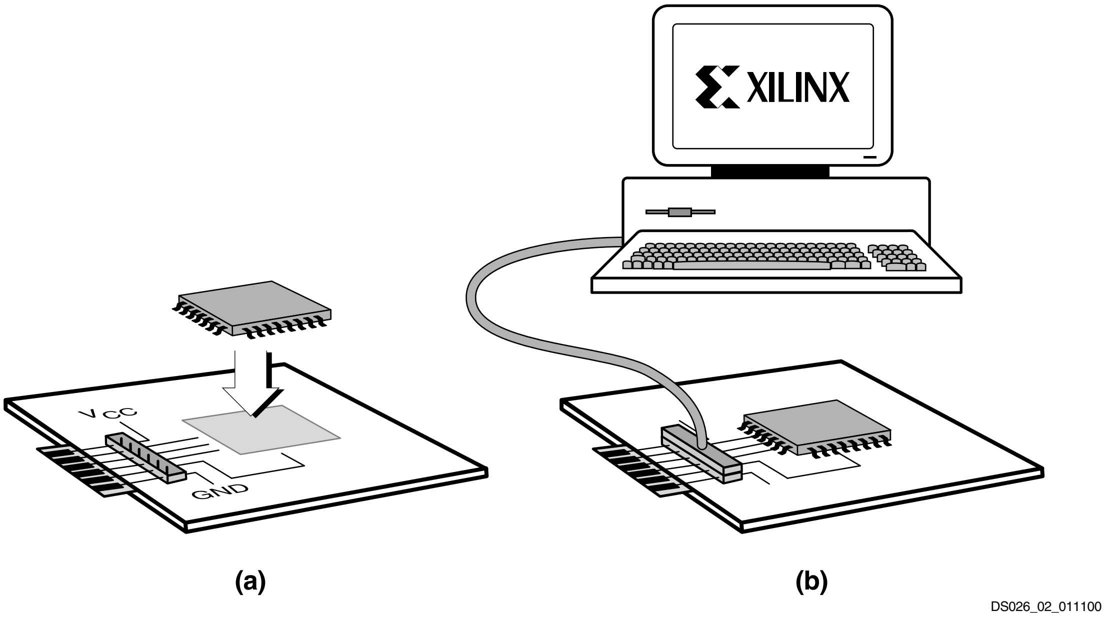


Figure 2: **In-System Programming Operation (a) Solder Device to PCB and (b) Program Using Download Cable**

IEEE 1149.1 Boundary-Scan (JTAG)

The XC18V00 family is fully compliant with the IEEE Std. 1149.1 Boundary-Scan, also known as JTAG. A Test Access Port (TAP) and registers are provided to support all required boundary scan instructions, as well as many of the optional instructions specified by IEEE Std. 1149.1. In addition, the JTAG interface is used to implement in-system programming (ISP) to facilitate configuration, erasure, and verification operations on the XC18V00 device.

Table 4 lists the required and optional boundary-scan instructions supported in the XC18V00. Refer to the IEEE Std. 1149.1 specification for a complete description of boundary-scan architecture and the required and optional instructions.

Table 4: **Boundary Scan Instructions**

Boundary-Scan Command	Binary Code [7:0]	Description
Required Instructions		
BYPASS	11111111	Enables BYPASS
SAMPLE/ PRELOAD	00000001	Enables boundary-scan SAMPLE/PRELOAD operation
EXTEST	00000000	Enables boundary-scan EXTEST operation
Optional Instructions		
CLAMP	11111010	Enables boundary-scan CLAMP operation
HIGHZ	11111100	all outputs in high-impedance state simultaneously
IDCODE	11111110	Enables shifting out 32-bit IDCODE
USERCODE	11111101	Enables shifting out 32-bit USERCODE
XC18V00 Specific Instructions		
CONFIG	11101110	Initiates FPGA configuration by pulsing $\overline{\text{CF}}$ pin Low

Instruction Register

The Instruction Register (IR) for the XC18V00 is eight bits wide and is connected between TDI and TDO during an instruction scan sequence. In preparation for an instruction scan sequence, the instruction register is parallel loaded with a fixed instruction capture pattern. This pattern is shifted out onto TDO (LSB first), while an instruction is shifted into the instruction register from TDI. The detailed composition of the instruction capture pattern is illustrated in Figure 3.

The ISP Status field, IR(4), contains logic "1" if the device is currently in ISP mode; otherwise, it contains logic "0". The Security field, IR(3), contains logic "1" if the device has been programmed with the security option turned on; otherwise, it contains logic "0".

	IR[7:5]	IR[4]	IR[3]	IR[2]	IR[1:0]	
TDI->	0 0 0	ISP Status	Security	0	0 1	->TDO

Notes:
1. IR(1:0) = 01 is specified by IEEE Std. 1149.1

Figure 3: **Instruction Register Values Loaded into IR as Part of an Instruction Scan Sequence**

Boundary Scan Register

The boundary-scan register is used to control and observe the state of the device pins during the EXTEST, SAMPLE/PRELOAD, and CLAMP instructions. Each output pin on the XC18V00 has two register stages that contribute to the boundary-scan register, while each input pin only has one register stage.

For each output pin, the register stage nearest to TDI controls and observes the output state, and the second stage closest to TDO controls and observes the High-Z enable state of the pin.

For each input pin, the register stage controls and observes the input state of the pin.

Identification Registers

The IDCODE is a fixed, vendor-assigned value that is used to electrically identify the manufacturer and type of the device being addressed. The IDCODE register is 32 bits wide. The IDCODE register can be shifted out for examination by using the IDCODE instruction. The IDCODE is available to any other system component via JTAG.

The IDCODE register has the following binary format:

```
vvvv:ffff:ffff:aaaa:aaaa:cccc:cccc:ccc1
```

where

v = the die version number

f = the family code (50h for XC18V00 family)

a = the ISP PROM product ID (26h for the XC18V04)

c = the company code (49h for Xilinx)

Note: The LSB of the IDCODE register is always read as logic "1" as defined by IEEE Std. 1149.1

Table 5 lists the IDCODE register values for the XC18V00 devices.

Table 5: **IDCODES Assigned to XC18V00 Devices**

ISP-PROM	IDCODE
XC18V01	05024093h
XC18V02	05025093h
XC18V04	05026093h
XC18V256	05022093h
XC18V512	05023093h

The USERCODE instruction gives access to a 32-bit user programmable scratch pad typically used to supply information about the device's programmed contents. By using the USERCODE instruction, a user-programmable identification code can be shifted out for examination. This code is loaded into the USERCODE register during programming of the XC18V00 device. If the device is blank or was not loaded during programming, the USERCODE register contains FFFFFFFFh.

XC18V00 TAP Characteristics

The XC18V00 family performs both in-system programming and IEEE 1149.1 boundary-scan (JTAG) testing via a single 4-wire Test Access Port (TAP). This simplifies system designs and allows standard Automatic Test Equipment to perform both functions. The AC characteristics of the XC18V00 TAP are described as follows.

TAP Timing

Figure 4 shows the timing relationships of the TAP signals. These TAP timing characteristics are identical for both boundary-scan and ISP operations.

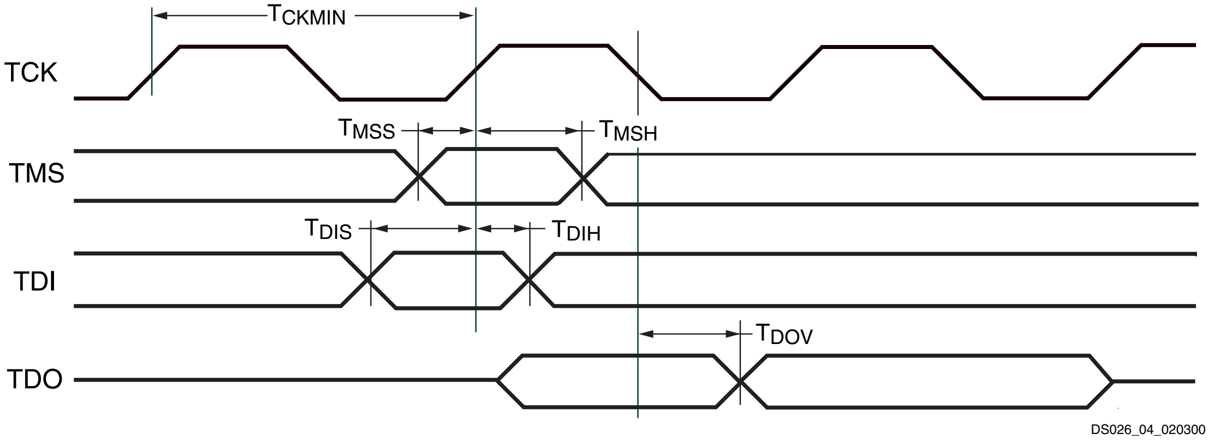


Figure 4: **Test Access Port Timing**

TAP AC Parameters

Table 6 shows the timing parameters for the TAP waveforms shown in Figure 4

Table 6: **Test Access Port Timing Parameters**

Symbol	Parameter	Min	Max	Units
T_{CKMIN1}	TCK minimum clock period	100	-	ns
T_{CKMIN2}	TCK minimum clock period, Bypass Mode	50	-	ns
T_{MSS}	TMS setup time	10	-	ns
T_{MSH}	TMS hold time	25	-	ns
T_{DIS}	TDI setup time	10	-	ns
T_{DIH}	TDI hold time	25	-	ns
T_{DOV}	TDO valid delay	-	25	ns

Connecting Configuration PROMs

Connecting the FPGA device with the configuration PROM (see Figure 6).

- The DATA output(s) of the PROM(s) drives the D_{IN} input of the lead FPGA device.
- The Master FPGA CCLK output drives the CLK input(s) of the PROM(s) (in Master Serial mode only).
- The $\overline{CEO}$ output of a PROM drives the $\overline{CE}$ input of the next PROM in a daisy chain (if any).
- The OE/$\overline{RESET}$ input of all PROMs is best driven by the $\overline{INIT}$ output of the lead FPGA device. This connection assures that the PROM address counter is reset before the start of any (re)configuration, even when a reconfiguration is initiated by a V_{CC} glitch.
- The PROM $\overline{CE}$ input can be driven from the DONE pin. The $\overline{CE}$ input of the first (or only) PROM can be driven by the DONE output of the first FPGA device, provided that DONE is not permanently grounded. $\overline{CE}$ can also be permanently tied Low, but this keeps the DATA output active and causes an unnecessary supply current of 10 mA maximum.
- Slave-Parallel/SelectMap mode is similar to slave serial mode. The DATA is clocked out of the PROM one byte per CCLK instead of one bit per CCLK cycle. See FPGA data sheets for special configuration requirements.

Initiating FPGA Configuration

The XC18V00 devices incorporate a pin named $\overline{\text{CF}}$ that is controllable through the JTAG CONFIG instruction. Executing the CONFIG instruction through JTAG pulses the $\overline{\text{CF}}$ low for 300-500 ns, which resets the FPGA and initiates configuration.

The $\overline{\text{CF}}$ pin must be connected to the $\overline{\text{PROGRAM}}$ pin on the FPGA(s) to use this feature.

The JTAG Programmer software can also issue a JTAG CONFIG command to initiate FPGA configuration through the “Load FPGA” setting.

Selecting Configuration Modes

The XC18V00 accommodates serial and parallel methods of configuration. The configuration modes are selectable through a user control register in the XC18V00 device. This control register is accessible through JTAG, and is set using the “Parallel mode” setting on the Xilinx JTAG Programmer software. Serial output is the default programming mode.

Master Serial Mode Summary

The I/O and logic functions of the Configurable Logic Block (CLB) and their associated interconnections are established by a configuration program. The program is loaded either automatically upon power up, or on command, depending on the state of the three FPGA mode pins. In Master Serial mode, the FPGA automatically loads the configuration program from an external memory. Xilinx PROMs are designed to accommodate the Master Serial mode.

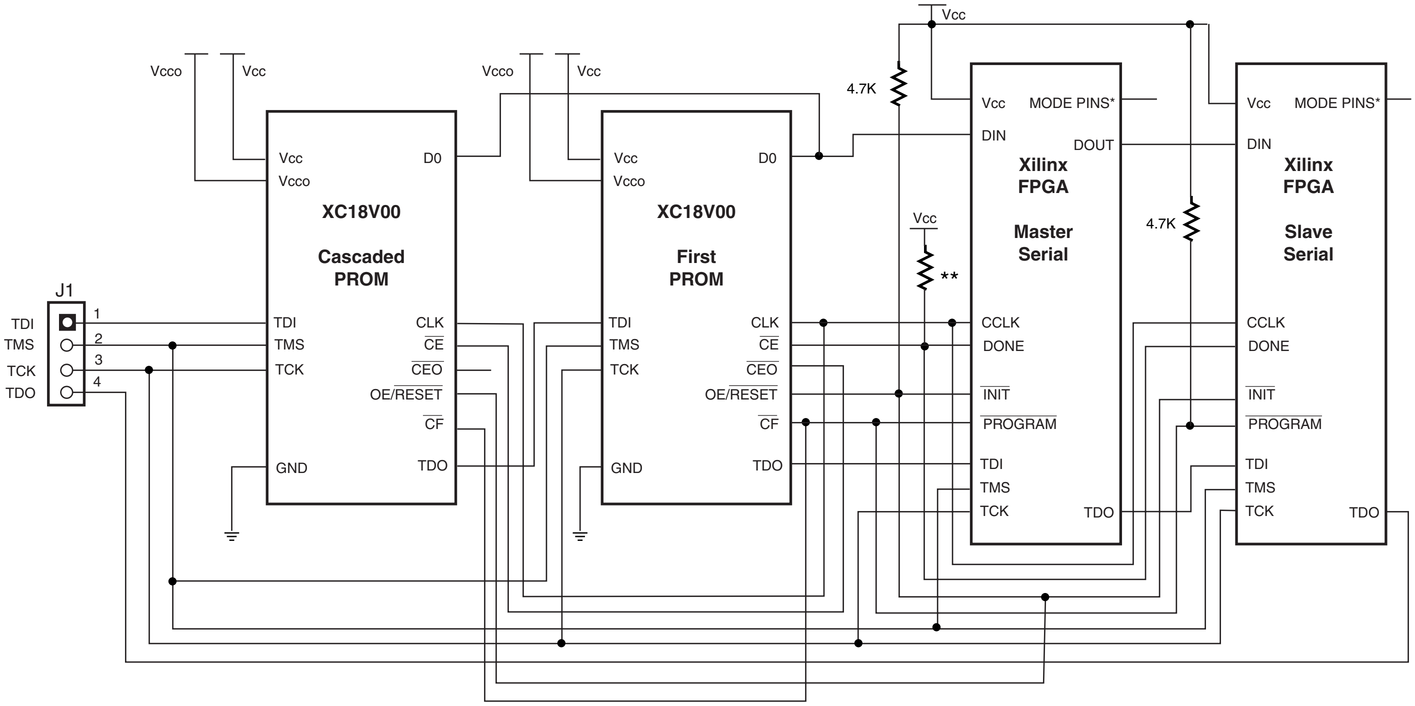
Upon power-up or reconfiguration, an FPGA enters the Master Serial mode whenever all three of the FPGA mode-select pins are Low (M0=0, M1=0, M2=0). Data is read from the PROM sequentially on a single data line. Synchronization is provided by the rising edge of the temporary signal CCLK, which is generated by the FPGA during configuration.

Master Serial Mode provides a simple configuration interface. Only a serial data line, a clock line, and two control lines are required to configure an FPGA. Data from the PROM is read sequentially, accessed via the internal address and bit counters which are incremented on every valid rising edge of CCLK. If the user-programmable, dual-function D_{IN} pin on the FPGA is used only for configuration, it must still be held at a defined level during normal operation. The Xilinx FPGA families take care of this automatically with an on-chip pull-up resistor.

Cascading Configuration PROMs

For multiple FPGAs configured as a serial daisy-chain, or a single FPGA requiring larger configuration memories in a serial or SelectMAP configuration mode, cascaded PROMs provide additional memory (Figure 5). Multiple XC18V00 devices can be concatenated by using the $\overline{\text{CEO}}$ output to drive the $\overline{\text{CE}}$ input of the downstream device. The clock inputs and the data outputs of all XC18V00 devices in the chain are interconnected. After the last bit from the first PROM is read, the next clock signal to the PROM asserts its $\overline{\text{CEO}}$ output Low and drives its DATA line to a high-impedance state. The second PROM recognizes the Low level on its $\overline{\text{CE}}$ input and enables its DATA output. See Figure 6.

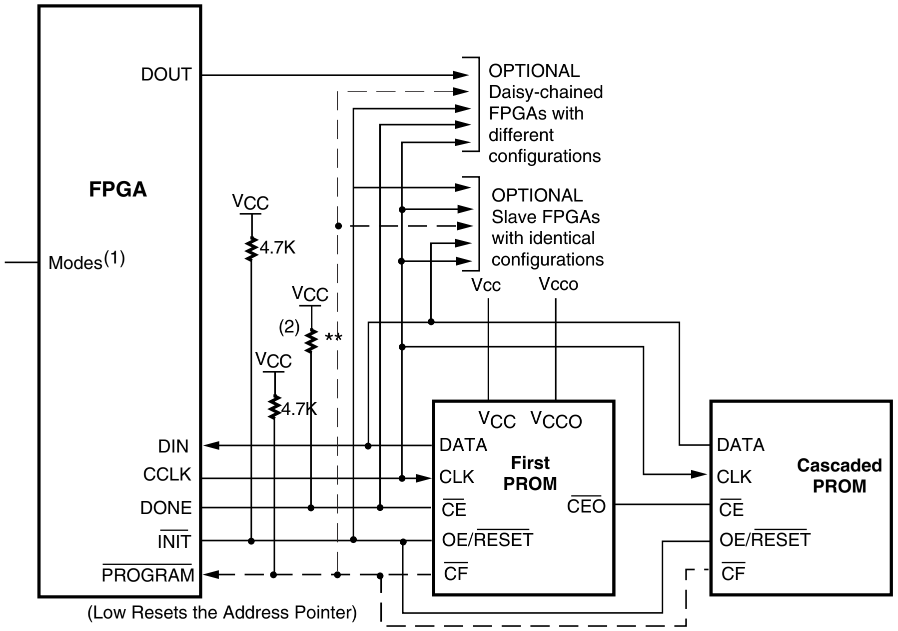
After configuration is complete, address counters of all cascaded PROMs are reset if the PROM OE/$\overline{\text{RESET}}$ pin goes Low.



* For Mode pin connections, refer to appropriate FPGA data sheet.
** Virtex, Virtex-E is 300 ohms, all others are 4.7K.

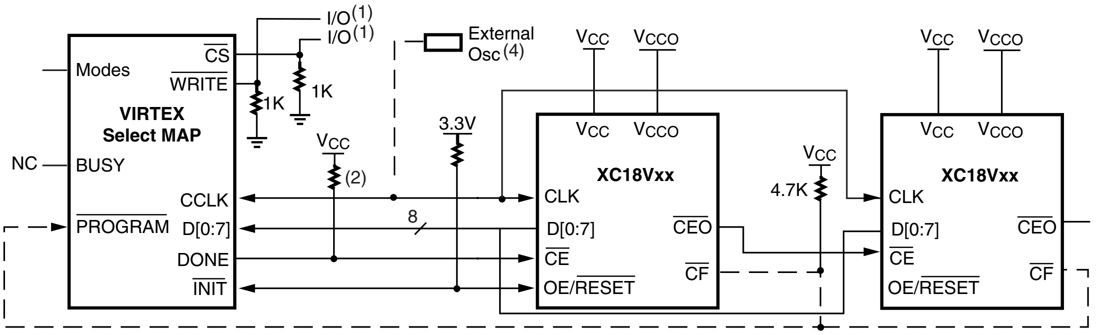
DS026_08_011501

Figure 5: **JTAG Chain for Configuring Devices in Master Serial Mode**



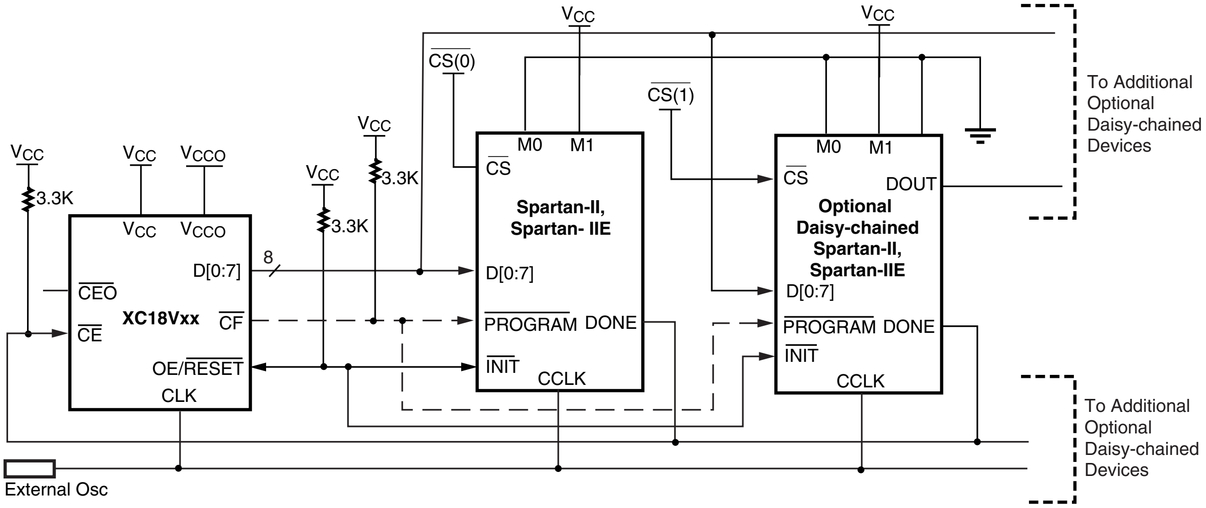
(1) For Mode pin connections, refer to the appropriate FPGA data sheet.
(2) Virtex is 300 ohms.

Master Serial Mode



(1) CS and WRITE must be pulled down to be used as I/O. One option is shown.
(2) Virtex, Virtex-E is 300 ohms, all others are 4.7K.
(3) For Mode pin connections, refer to the appropriate FPGA data sheet.
(4) External oscillator required for Virtex/E SelectMAP or Virtex-II Slave SelectMAP modes.

Virtex SelectMAP Mode



Spartan-II/IIE Slave-Parallel Mode

DS026_05_111201

Figure 6: **(a) Master Serial Mode (b) Virtex SelectMAP Mode (c) Spartan-II/IIE Slave-Parallel Mode**
(dotted lines indicate optional connection)

5V Tolerant I/Os

The I/Os on each re-programmable PROM are fully 5V tolerant even through the core power supply is 3.3V. This allows 5V CMOS signals to connect directly to the PROM inputs without damage. In addition, the 3.3V V_{CC} power supply can be applied before or after 5V signals are applied to the I/Os. In mixed 5V/3.3V/2.5V systems, the user pins, the core power supply (V_{CC}), and the output power supply (V_{CCO}) can have power applied in any order. This makes the PROM devices immune to power supply sequencing issues.

Reset Activation

On power up, OE/$\overline{\text{RESET}}$ is held low until the XC18V00 is active (1 ms) and able to supply data after receiving a CCLK pulse from the FPGA. OE/$\overline{\text{RESET}}$ is connected to an external resistor to pull OE/$\overline{\text{RESET}}$ HIGH releasing the FPGA $\overline{\text{INIT}}$ and allowing configuration to begin. OE/$\overline{\text{RESET}}$ is held low until the XC18V00 voltage reaches the operating voltage range. If the power drops below 2.0V, the PROM resets. OE/$\overline{\text{RESET}}$ polarity is NOT programmable.

Standby Mode

The PROM enters a low-power standby mode whenever $\overline{\text{CE}}$ is asserted High. The output remains in a high-impedance state regardless of the state of the OE input. JTAG pins TMS, TDI and TDO can be in a high-impedance state or High.

Customer Control Pins

The XC18V00 PROMs have various control bits accessible by the customer. These can be set after the array has been programmed using “Skip User Array” in Xilinx JTAG Programmer Software.

Table 7: **Truth Table for PROM Control Inputs**

Control Inputs		Internal Address	Outputs		
OE/RESET	**CE**		**DATA**	**CEO**	**I_{CC}**
High	Low	If address $\leq$ TC$^{(1)}$: increment If address > TC$^{(1)}$: don’t change	Active High-Z	High Low	Active Reduced
Low	Low	Held reset	High-Z	High	Active
High	High	Held reset	High-Z	High	Standby
Low	High	Held reset	High-Z	High	Standby

Notes:
1. TC = Terminal Count = highest address value. TC + 1 = address 0.

Absolute Maximum Ratings[1,2]

Symbol	Description	Value	Units
V_{CC}	Supply voltage relative to GND	–0.5 to +4.0	V
V_{IN}	Input voltage with respect to GND	–0.5 to +5.5	V
V_{TS}	Voltage applied to High-Z output	–0.5 to +5.5	V
T_{STG}	Storage temperature (ambient)	–65 to +150	°C
T_{SOL}	Maximum soldering temperature (10s @ 1/16 in.)	+260	°C
T_J	Junction temperature	+150	°C

Notes:

1. Maximum DC undershoot below GND must be limited to either 0.5V or 10 mA, whichever is easier to achieve. During transitions, the device pins can undershoot to –2.0V or overshoot to +7.0V, provided this over- or undershoot lasts less then 10 ns and with the forcing current being limited to 200 mA.
2. Stresses beyond those listed under Absolute Maximum Ratings might cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those listed under Operating Conditions is not implied. Exposure to Absolute Maximum Ratings conditions for extended periods of time might affect device reliability.

Recommended Operating Conditions

Symbol	Parameter		Min	Max	Units
V_{CCINT}	Internal voltage supply (T_A = 0°C to +70°C)	Commercial	3.0	3.6	V
	Internal voltage supply (T_A = –40°C to +85°C)	Industrial	3.0	3.6	V
V_{CCO}	Supply voltage for output drivers for 3.3V operation		3.0	3.6	V
	Supply voltage for output drivers for 2.5V operation		2.3	2.7	V
V_{IL}	Low-level input voltage		0	0.8	V
V_{IH}	High-level input voltage		2.0	5.5	V
V_O	Output voltage		0	V_{CCO}	V
T_{VCC}	V_{CC} rise time from 0V to nominal voltage(1)		1	50	ms

Notes:

1. At power up, the device requires the V_{CC} power supply to monotonically rise from 0V to nominal voltage within the specified V_{CC} rise time. If the power supply cannot meet this requirement, then the device might not perform power-on-reset properly.

Quality and Reliability Characteristics

Symbol	Description	Min	Max	Units
T_{DR}	Data retention	20	-	Years
N_{PE}	Program/erase cycles (Endurance)	20,000	-	Cycles
V_{ESD}	Electrostatic discharge (ESD)	2,000	-	Volts

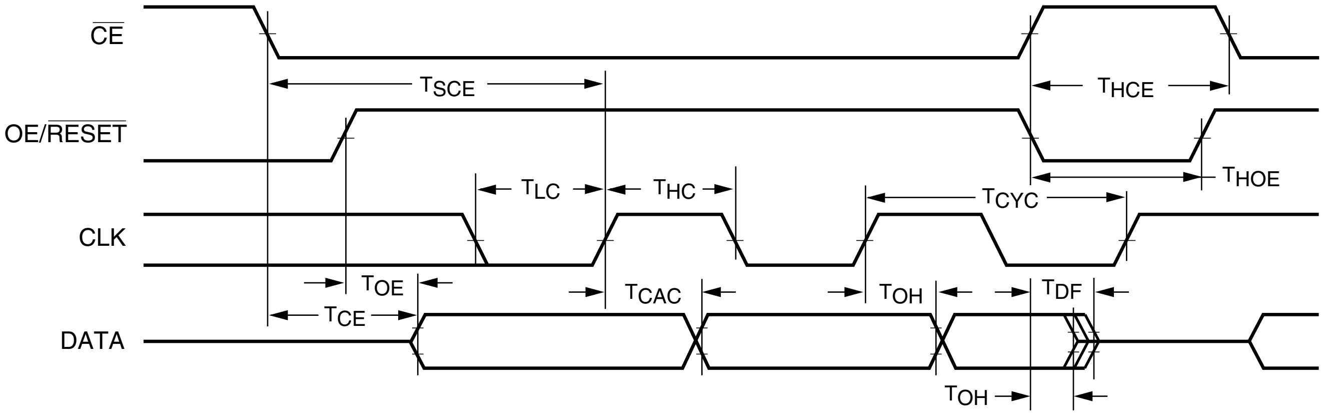
DC Characteristics Over Operating Conditions

Symbol	Parameter	Test Conditions	Min	Max	Units
V_{OH}	High-level output voltage for 3.3V outputs	I_{OH} = –4 mA	2.4	-	V
	High-level output voltage for 2.5V outputs	I_{OH} = –500 µA	90% V_{CCO}	-	V
V_{OL}	Low-level output voltage for 3.3V outputs	I_{OL} = 8 mA	-	0.4	V
	Low-level output voltage for 2.5V outputs	I_{OL} = 500 µA	-	0.4	V
I_{CC}	Supply current, active mode	25 MHz	-	25	mA
I_{CCS}	Supply current, standby mode		-	10	mA
I_{ILJ}	JTAG pins TMS, TDI, and TDO	$V_{CC\,=}$ MAX V_{IN} = GND	–100	-	µA
I_{IL}	Input leakage current	V_{CC} = Max V_{IN} = GND or V_{CC}	–10	10	µA
I_{IH}	Input and output High-Z leakage current	V_{CC} = Max V_{IN} = GND or V_{CC}	–10	10	µA
C_{IN} and C_{OUT}	Input and output capacitance	V_{IN} = GND f = 1.0 MHz	-	10	pF

Notes:

1. 18V01/18V512/18V256 only, cascadable.
2. 18V01/18V512/18V256 only, non-cascadable, no brown-out protection.

AC Characteristics Over Operating Conditions for XC18V04 and XC18V02

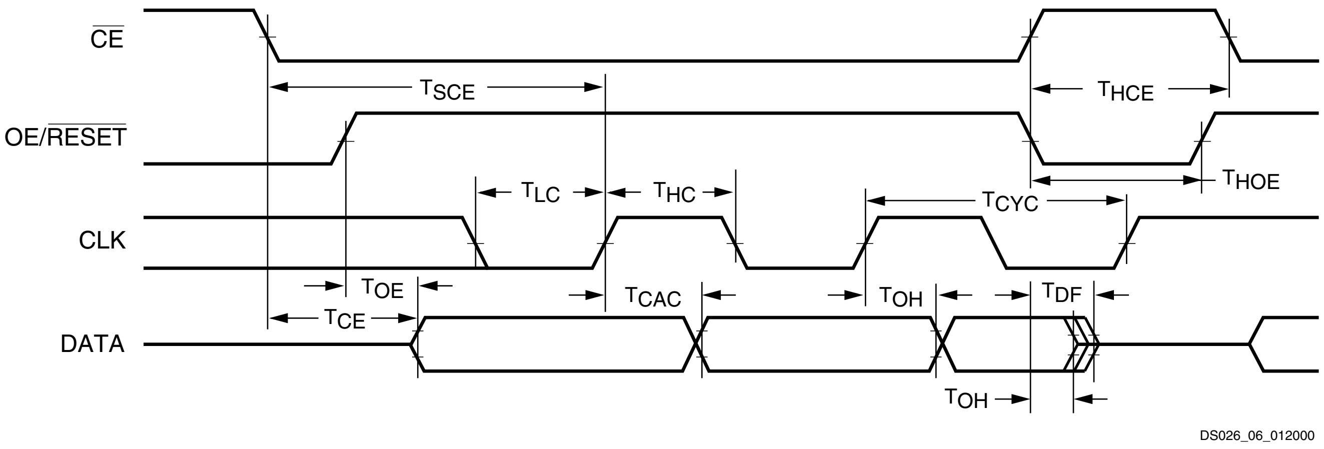


DS026_06_012000

Symbol	Description	Min	Max	Units
T_{OE}	OE/$\overline{RESET}$ to data delay	-	10	ns
T_{CE}	$\overline{CE}$ to data delay	-	20	ns
T_{CAC}	CLK to data delay	-	20	ns
T_{OH}	Data hold from $\overline{CE}$, OE/$\overline{RESET}$, or CLK	0	-	ns
T_{DF}	$\overline{CE}$ or OE/$\overline{RESET}$ to data float delay$^{(2)}$	-	25	ns
T_{CYC}	Clock periods	50	-	ns
T_{LC}	CLK Low time$^{(3)}$	10	-	ns
T_{HC}	CLK High time$^{(3)}$	10	-	ns
T_{SCE}	$\overline{CE}$ setup time to CLK (to guarantee proper counting)$^{(3)}$	25	-	ns
T_{HCE}	$\overline{CE}$ High time (to guarantee proper counting)	2	-	µs
T_{HOE}	OE/$\overline{RESET}$ hold time (guarantees counters are reset)	25	-	ns

Notes:
1. AC test load = 50 pF.
2. Float delays are measured with 5 pF AC loads. Transition is measured at ±200 mV from steady state active levels.
3. Guaranteed by design, not tested.
4. All AC parameters are measured with V_{IL} = 0.0V and V_{IH} = 3.0V.
5. If T_{HCE} High < 2 µs, T_{CE} = 2 µs.

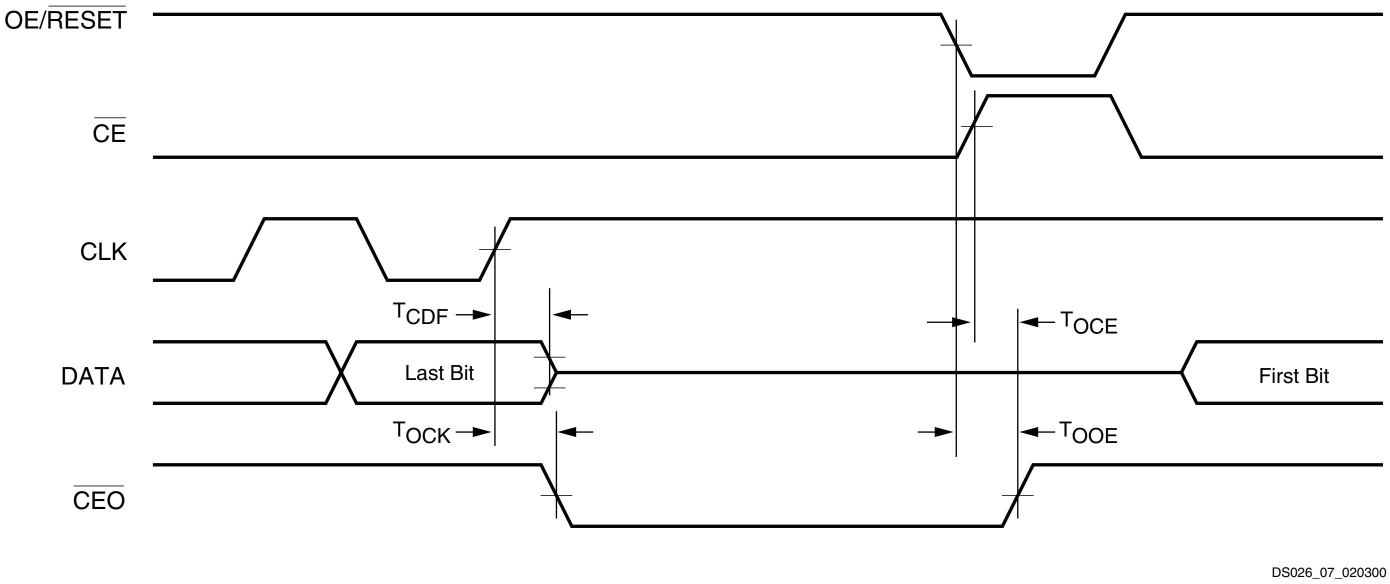
AC Characteristics Over Operating Conditions for XC18V01, XC18V512, and XC18V256



Symbol	Description	Min	Max	Units
T_{OE}	OE/$\overline{\text{RESET}}$ to data delay	-	10	ns
T_{CE}	$\overline{\text{CE}}$ to data delay	-	15	ns
T_{CAC}	CLK to data delay	-	15	ns
T_{OH}	Data hold from $\overline{\text{CE}}$, OE/$\overline{\text{RESET}}$, or CLK	0	-	ns
T_{DF}	$\overline{\text{CE}}$ or OE/$\overline{\text{RESET}}$ to data float delay$^{(2)}$	-	25	ns
T_{CYC}	Clock periods	30	-	ns
T_{LC}	CLK Low time$^{(3)}$	10	-	ns
T_{HC}	CLK High time$^{(3)}$	10	-	ns
T_{SCE}	$\overline{\text{CE}}$ setup time to CLK (to guarantee proper counting)$^{(3)}$	20	-	ns
T_{HCE}	$\overline{\text{CE}}$ hold time to CLK (to guarantee proper counting)	2	-	µs
T_{HOE}	OE/$\overline{\text{RESET}}$ hold time (guarantees counters are reset)	20	-	ns

Notes:
1. AC test load = 50 pF.
2. Float delays are measured with 5 pF AC loads. Transition is measured at ±200 mV from steady state active levels.
3. Guaranteed by design, not tested.
4. All AC parameters are measured with $V_{IL} = 0.0V$ and $V_{IH} = 3.0V$.
5. If T_{HCE} High < 2 µs, T_{CE} = 2 µs.

AC Characteristics Over Operating Conditions When Cascading for XC18V04 and XC18V02

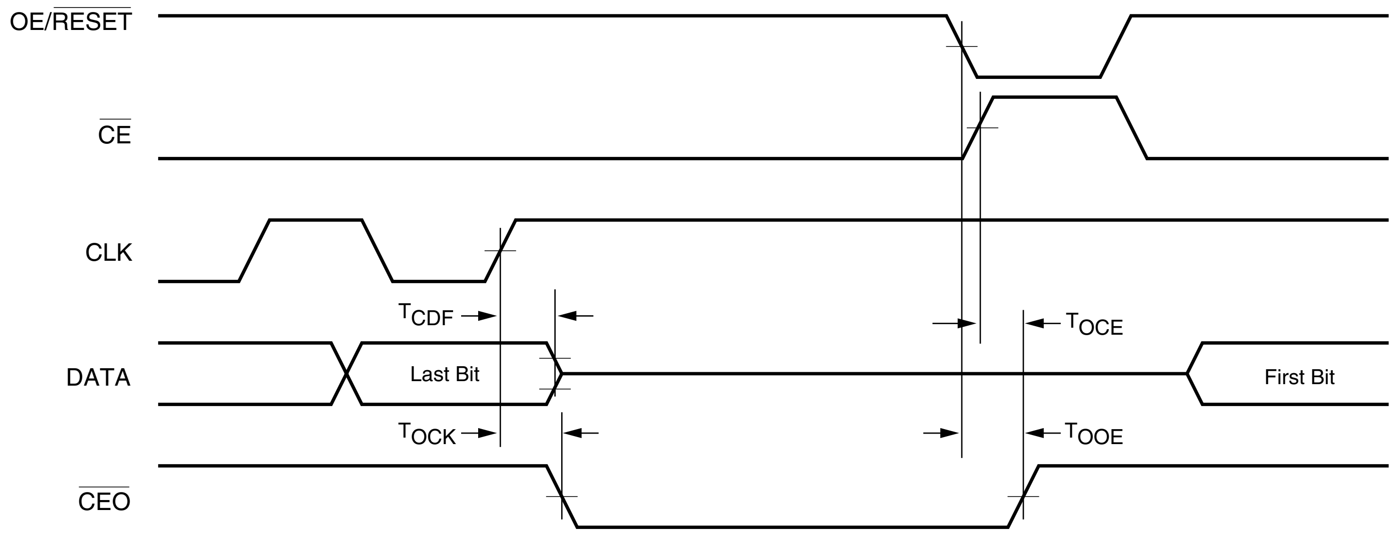


Symbol	Description	Min	Max	Units
T_{CDF}	CLK to data float delay(2,3)	-	25	ns
T_{OCK}	CLK to $\overline{CEO}$ delay(3)	-	20	ns
T_{OCE}	CE to $\overline{CEO}$ delay(3)	-	20	ns
T_{OOE}	OE/$\overline{RESET}$ to $\overline{CEO}$ delay(3)	-	20	ns

Notes:

1. AC test load = 50 pF.
2. Float delays are measured with 5 pF AC loads. Transition is measured at ±200 mV from steady state active levels.
3. Guaranteed by design, not tested.
4. All AC parameters are measured with V_{IL} = 0.0V and V_{IH} = 3.0V.

AC Characteristics Over Operating Conditions When Cascading for XC18V01, XC18V512, and XC18V256



DS026_07_020300

Symbol	Description	Min	Max	Units
T_{CDF}	CLK to data float delay(2,3)	-	25	ns
T_{OCK}	CLK to $\overline{\text{CEO}}$ delay(3)	-	20	ns
T_{OCE}	CE to $\overline{\text{CEO}}$ delay(3)	-	20	ns
T_{OOE}	OE/$\overline{\text{RESET}}$ to $\overline{\text{CEO}}$ delay(3)	-	20	ns

Notes:
1. AC test load = 50 pF.
2. Float delays are measured with 5 pF AC loads. Transition is measured at ±200 mV from steady state active levels.
3. Guaranteed by design, not tested.
4. All AC parameters are measured with V_{IL} = 0.0V and V_{IH} = 3.0V.

Ordering Information

XC18V04 VQ44 C

Device Number

XC18V04
XC18V02
XC18V01
XC18V512
XC18V256

Package Type

VQ44 = 44-pin Plastic Quad Flat Package
PC44 = 44-pin Plastic Chip Carrier(1)
SO20 = 20-pin Small-Outline Package(2)
PC20 = 20-pin Plastic Leaded Chip Carrier(2)

Operating Range/Processing

C = Commercial (T_A = 0° to +70°C)
I = Industrial (T_A = –40° to +85°C)

Notes:
1. XC18V04 and XC18V02 only.
2. XC18V01, XC18V512, and XC18V256 only.

Valid Ordering Combinations

XC18V04VQ44C	XC18V02VQ44C	XC18V01VQ44C	XC18V512VQ44C	XC18V256VQ44C
XC18V04PC44C	XC18V02PC44C	XC18V01PC20C	XC18V512PC20C	XC18V256PC20C
		XC18V01SO20C	XC18V512SO20C	XC18V256SO20C
XC18V04VQ44I	XC18V02VQ44I	XC18V01VQ44I	XC18V512VQ44I	XC18V256VQ44I
XC18V04PC44I	XC18V02PC44I	XC18V01PC20I	XC18V512PC20I	XC18V256PC20I
		XC18V01SO20I	XC18V512SO20I	XC18V256SO20I

Marking Information

44-pin Package

XC18V04 VQ44 C

Device Number

XC18V04
XC18V02
XC18V01
XC18V512
XC18V256

Package Type

VQ44 = 44-pin Plastic Quad Flat Package
PC44 = 44-pin Plastic Leaded Chip Carrier(1)

Notes:
1. XC18V02 and XC18V04 Only.

Operating Range/Processing

C = Commercial (T_A = 0° to +70°C)
I = Industrial (T_A = –40° to +85°C)

20-pin Package(1)

Due to the small size of the commercial serial PROM packages, the complete ordering part number cannot be marked on the package. The XC prefix is deleted and the package code is simplified. Device marking is as follows:

18V01 S C

Device Number

18V01
18V512
18V256

Package Type

S20 = 20-pin Small-Outline Package
J20 = 20-pin Plastic Leaded Chip Carrier

Notes:
1. XC18V01, XC18V512, and XC18V256 only.

Operating Range/Processing

C = Commercial (T_A = 0° to +70°C)
I = Industrial (T_A = –40° to +85°C)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
2/9/99	1.0	First publication of this early access specification
8/23/99	1.1	Edited text, changed marking, added $\overline{\text{CF}}$ and parallel load
9/1/99	1.2	Corrected JTAG order, Security and Endurance data.
9/16/99	1.3	Corrected SelectMAP diagram, control inputs, reset polarity. Added JTAG and $\overline{\text{CF}}$ description, 256 Kbit and 128 Kbit devices.
01/20/00	2.0	Added Q44 Package, changed XC18xx to XC18Vxx
02/18/00	2.1	Updated JTAG configuration, AC and DC characteristics
04/04/00	2.2	Removed stand alone resistor on INIT pin in Figure 5. Added Virtex-E and EM parts to FPGA table.
06/29/00	2.3	Removed XC18V128 and updated format. Added AC characteristics for XC18V01, XC18V512, and XC18V256 densities.
11/13/00	2.4	Features: changed 264 MHz to 264 Mb/s at 33 MHz; AC Spec.: T_{SCE} units to ns, T_{HCE} CE High time units to µs. Removed Standby Mode statement: “The lower power standby modes available on some XC18V00 devices are set by the user in the programming software”. Changed 10,000 cycles endurance to 20,000 cycles.
01/15/01	2.5	Updated Figures 5 and 6, added 4.7 resistors. Identification registers: changes ISP PROM product ID from 06h to 26h.
04/04/01	2.6	Updated Figure 6, Virtex SelectMAP mode; added XC2V products to Compatible PROM table; changed Endurance from 10,000 cycles, 10 years to 20,000, 20 years;
04/30/01	2.7	Updated Figure 6: removed Virtex-E in Note 2, fixed SelectMAP mode connections. Under AC Characteristics Over Operating Conditions for XC18V04 and XC18V02, changed T_{SCE} from 25 ms to 25 ns.
06/11/01	2.8	AC Characteristics Over Operating Conditions for XC18V01, XC18V512, and XC18V256 Changed Min values for T_{SCE} from 20 ms to 20 ns and for T_{HCE} from 2 ms to 2 µs.
09/28/01	2.9	Changed the boundary scan order for the CEO pin in Table 1, updated the configuration bits values in the table under Xilinx FPGAs and Compatible PROMs, and added information to the Recommended Operating Conditions table.
11/12/01	3.0	Updated for Spartan-IIE FPGA family.

Appendix D

Glossary

AQL

Acceptable quality level. The relative number of devices, expressed in parts-per-million (ppm), that might not meet specification or might be defective. Typical values are around 10 ppm,

ASIC

Application-specific integrated circuit, also called a *gate array*.

asynchronous

Logic that is not synchronized by a clock. Asynchronous designs can be faster than synchronous ones, but are more sensitive to parametric changes, and are thus less robust.

ATM

Asynchronous transfer mode. A very-high-speed (megahertz to gigahertz) connection-oriented bit-serial protocol for transmitting data and real-time voice and video in fixed-length packets (480byte payload, 5-byte header).

back annotation

Automatically attaching timing values to the entered design format after the design has been placed and routed in a field-programmable gate array (FPGA).

behavioral language

Top-down description from an even higher level than VDHL.

bitstream

The bitstream is a binary representation of an implemented FPGA design. The bitstream is generated by Xilinx bit generation tools (BitGen and Makebits) and is denoted with the **.bit** extension. For information on creating BIT files, refer to the *Hardware Debugger Reference/User Guide*.

block RAM

An 18-Kbit block of random access memory (RAM) inside the Virtex-II device. Dual-port and synchronous operation are desirable.

block SelectRAM

Fully-synchronous, dual-port memories in the Virtex-II FPGAs. Each of these memories contain 18 x 1024 (18,432) bits. The organization of each memory is configurable. Block SelectRAM resources complement smaller, distributed, LUT-based SelectRAM resources.

Boundary Scan interface

One of the configuration interfaces on the Virtex device. This is a bit-serial interface. The Boundary Scan interface is also known as the JTAG port. Also see *SelectMAP interface*.

capture data

The flip-flop and pad data saved from the logic cells and I/O blocks into the bitstream for readback. Use the CAPTURE_VIRTEX primitive in your HDL code to specify the trigger and clock for the capture operation.

compiler

Software that converts a higher-language description into a lower-level representation. For FPGAs, the complete partition, place, and process.

configurable logic block (CLB)

Xilinx-specific name for a block of logic surrounded by routing resources. The functional elements for constructing logic circuits. The Virtex-II CLB is made up of four slices, and each slice contains two Logic Cells.

configuration file

The internally stored file that controls the FPGA sot hat it performs the desired logic function. Also, the act of loading an FPGA with that file. That is, the process of programming Xilinx SRAM-based FPGAs with a bitstream.

configuration bitstream

Configuration commands with configuration data.

configuration clock (CCLK)

During configuration, the configuration clock (CCLK) is an output in Master modes or in the Asynchronous Peripheral mode but is an input in Slave, Synchronous Peripheral, Express, and SelectMAP/Slave Serial modes. After configuration, CCLK has a weak pull-up and can be selected as the readback clock.

configuration commands

Instructions for the Virtex-II device. There are two classes of Configuration Command — Major and Minor. The Major Commands read and write data to configuration registers in the Virtex-II device. The Minor commands instruct the Virtex-II configuration logic to perform specific functions.

configuration data

Bits that directly define the state of programmable logic. These are written to a Virtex-II device in a configuration bitstream, and read as readback data from a Virtex-II device.

configuration frame

The configuration bits in a Virtex-II device are organized in columns. A column of CLBs with the I/O blocks above and below the CLBs contain 48 frames of configuration bits. The smallest number of bits that can be read or written through the configuration interfaces is one frame.

configuration interface

A logical interface on the Virtex-II device through which configuration commands and data can be read and written. A interface consists of one or more physical device pins.

configuration readback

The operation of reading configuration data (also known as readback data) from a Virtex-II device.

constraints

Performance requirements imposed on the design, usually in the form of maximum allowable delay, or the required operating frequency.

$\overline{\text{CS}}$ pin

The $\overline{\text{CS}}$ pin is the Chip Enable pin for Virtex-II devices. It is used only in SelectMAP mode. When $\overline{\text{CS}}$ is asserted (Low) the device examines data on the Data bus. When $\overline{\text{CS}}$ is de-asserted (High), all CCLK transitions are ignored.

DataFrame

A DataFrame is a block of configuration data. A configuration bit-stream contains many such frames, each with a start bit and stop bits. Also see *configuration frame*.

device pin

One of the electrical connections on the package containing the Virtex-II device.

digital signal processing (DSP)

The manipulation of analog data that has been sampled and converted into a digital representation. Examples are filtering, convolution, Fast-Fourier-Transform, and so on.

DIN pin

During serial configuration, the DIN pin is the serial configuration data input receiving data on the rising edge of CCLK. During parallel configuration, DIN is the D0 input. After configuration, DIN is a user-programmable I/O pin.

DONE pin

The DONE pin on a Xilinx FPGA is a bidirectional signal with an optional internal pull-up resistor. As an output, it indicates the completion of the configuration process. As an input, a low level on DONE can be configured to delay the global logic initialization and the enabling of outputs.

DOUT pin

During configuration in any mode except Express and SelectMAP, the DOUT pin is the serial configuration data output that can drive the DIN pin of daisy-chained slave FPGAs. DOUT data changes on the falling edge of CCLK, one-and-a-half CCLK periods after it is received at the DIN pin (in Master Serial Mode only).

DOUT/BUSY pin

For Virtex-II devices, the DOUT/BUSY pin has a dual purpose, depending on device mode. When the device is in Serial mode, this pin functions as DOUT. When the device is in SelectMAP/Slave Parallel mode, this pin functions as a handshaking signal. If BUSY is asserted (High) on a rising edge of CCLK, the data is not seen on the data bus, and should be held until the data is accepted.

dynamic random access memory (DRAM)

A low-cost read-write memory where data is stored on capacitors and must be refreshed periodically. DRAMs are usually addressed by a sequence of two addresses, row address, and column address, which makes them slower and more difficult to use than SRAMs. Also see *SRAM*.

electronic data interchange format (EDIF)

Industry standard for specifying a logic design in text (ASCII) form.

electrostatic discharge (ESD)

High-voltage discharge can rupture the input transistor gate oxide. ESD-protection diodes divert the current to the supply leads.

failure in time (FIT)

Describes the number of device failures statistically expected for a certain number of device-hours. Expressed as failures per one billion (10^9) device hours. Device temperature must be specified. Mean time between failure (MTBF) can be calculated from FIT. 10 FITs are good; 100 FITs are bad.

first-in first-out (FIFO)

FIFO memory where data is stored in the incoming sequence and is read out in the same sequence. Input and output can be asynchronous to each other. A FIFO needs no external addresses, although all modern FIFOs are implemented internally with RAMs driven by circular read and write counters.

flash

Non-volatile programmable technology, and alternative to electrically-erasable programmable read-only memory (EEPROM) technology. The memory content can be erased by an electrical signal. This allows in-system programmability and eliminates the need for ultraviolet light and quartz windows in the package.

flip-flop

Single-bit storage cell that samples its data input at the active (rising or falling) clock edge, and then presents the new state on its Q output after that clock edge, holding it there until after the next active clock edge.

frame

Also see *configuration frame*.

field programmable gate array (FPGA)

An integrated circuit that contains configurable (programmable) logic blocks and configurable interconnect between these blocks. Xilinx FPGAs are SRAM- based programmable logic devices (PLDs).

function generator

Also called a look-up table (LUT), with N inputs and one output. Can implement any logic function of its N inputs. N can be between 3 and 6; 4-input function generators are most popular.

gate

Smallest logic element with several inputs and one output. The AND gate output is High when all inputs are High. The OR gate output is High when at least one input is High. The NAND gate output is Low when all inputs are High. A 2-input NAND gate is used as the measurement unit for gate array complexity.

gate array

ASIC where transistors are predefined, and only the interconnect pattern is customized for the individual application.

graphical user interface (GUI)

The way of representing the computer output on the screen as graphics, pictures, icons, and windows. Pioneered by Xerox and the Apple Macintosh, now universally adopted, e.g., by Windows95 and others.

HDL

Hardware Description Language.

HardWire

Xilinx name for a low-cost derivative of an FPGA, where the configuration is fixed, but functionality and footprint are identical with the original FPGA-based design.

HDC pin

The High during configuration (HDC) pin is driven High until the I/Os become active in the Startup sequence. It is available as a control output indicating that configuration is not yet complete. After configuration, HDC is a user-programmable I/O pin.

hierarchical design

Design description in multiple layers, from the highest (overview) tot he lowest (circuit details). An alternative is flat design, where everything is described at the same level of detail.

$\overline{\text{INIT}}$ pin

The $\overline{\text{INIT}}$ pin is a quadruple function signal. Before and during configuration, $\overline{\text{INIT}}$ is a bidirectional signal. A 1 - 10 kΩ external pull-up resistor is recommended. As an active-Low open-drain output, $\overline{\text{INIT}}$ is held Low during power stabilization and internal clearing of the configuration memory. As an active-Low input, it can be used to hold the FPGA in the internal WAIT state before the start of configuration. During configuration, a Low on this output indicates that a configuration data error has occurred. After the I/O become active in the Startup sequence, $\overline{\text{INIT}}$ becomes a user-programmable I/O.

intellectual property (IP)

In the legal sense, patents, copyrights, and trade secrets. In integrated circuits (ICs), predefined large functions, called "cores," that help the user complete a large design faster.

JTAG

Joint Test Action Group. Previous name for IEEE 1149.1 boundary scan, a method for testing boards and integrated circuits. Also see *Parallel Cable III*.

LogiBLOX

Library of logic modules, often with user-definable parameters, like data width. Similar to LPM.

logic cell (LC)

Metric for FPGA density. The basic building block of the Virtex-II CLB. An LC includes a 4-input function generator, carry logic, and a storage element.

LDC pin

Low during configuration (LDC) is driven Low until the I/Os become active in the Startup sequence. It is available as a control output indicating that configuration isn't complete. After configuration, LDC is a user-programmable I/O pin.

LPM

Library of Parametrized Modules. Library of logic modules, often with user-definable parameters, like data width. Similar to LogiBLOX.

LUT

Look-up table, also called a function generator with N inputs and one output. Can implement any logic function of its N inputs. N is between 3 and 6; most popular are 4-input LUTs.

LUT SelectRAM

Shallow RAM structure implemented in CLB look-up tables (LUTs). Also see *block SelectRAM*.

mapping

Process of assigning portions of the logic design to the physical chip resources (CLBs). With FPGAs, mapping is more demanding and more important a process than with gate arrays. Also see *synthesis*.

MTBF

Mean Time Between Failure. The statistically relevant up-time between equipment failure. Also see *failure in time (FIT)*.

MultiLINX cable

The MultiLINX cable provides many complex functions and can be loaded with new firmware as it becomes available. It can be connected to the host computer in two ways: via a Serial port or a USB port. The MultiLINX cable is supported by the Hardware Debugger software for Slave Serial and SelectMAP/Slave Parallel programming (as appropriate), as well as readback/verify. It is also supported by the JTAG programmer software for JTAG programming of both CPLDs and FPGAs.

netlist

Textual description of logic and interconnects. Also see *XNF file* and *electronic data interchange format (EDIF)*.

NRE

Non-Recurring Engineering charges. Start-up cost for the creation of an ASIC, gate array, or HardWire. Pays for layout, masks, and test development. FPGAs and CPLD do not require NRE.

optimization

Design change to improve performance. Also see *synthesis*.

pad

Pad bits are extra bits used to make the total number of bits in a frame an integral multiple of 32, the number of bits in a configuration word. A pad word is an extra word used at the end of a configuration frame for pipelining. A pad frame is an extra configuration frame used at the beginning of a configuration readback and at the end of a configuration write for pipelining.

Parallel Cable III

The Xilinx Parallel Cable III (model DLC5) is a serial download cable. The Parallel cable uses a serial 25-pin interface to the parallel port of a host computer and two 6-pin headers for flying-wire connectors to a target board. The Parallel cable is supported by the Hardware Debugger software for performing Slave Serial configuration of FPGAs only. The Parallel cable is also supported by the JTAG Programmer software for performing Slave Serial and Boundary Scan configuration of FPGAs, and Boundary Scan programming of CPLDs. For more information on using the Parallel cable, refer to Chapter 8 or this guide, the Hardware Debugger Reference/Users Guide, and the JTAG Programmer Guide.

partitioning

In FPGAs, the process of dividing the logic into subfunctions that can later be placed into individual CLBs. Partitioning precedes placement.

PCI

Peripheral Component Interface. Synchronous bus standard characterized by short range, light loading, low cost, and high performance. __?-MHz PCI can support data byte transfers up to ___? megabytes per second (Mb/s) on __? parallel data lines (including parity) and a common clock.

PCMCIA

Personal Computer Memory Card Interface Association. Physical and electrical standard for small plug-in boards for portable computers.

pin-locking

Rigidly defining and maintaining the functionality and timing requirements of device pins while the internal logic is still being designed or modified. Pin-locking has become important, since circuit board fabrication times are longer than PLD design implementation times.

PIP

Programmable Interconnect Point. In Xilinx FPGAs, a point where two signal lines can be connected, as determined by the device configuration.

placement

In FPGAs, the process of assigning specific parts of the design to specific locations (CLBs) on the chip. Usually done automatically. Also see *partitioning*.

PLD

Programmable Logic Device. Generic name for all programmable logic: PALs, CPLDs, and FPGAs.

preamble

The Preamble is a 4-bit binary sentinel (“0010”b) used to indicate the beginning of the LengthCount in the Header portion of the bitstream. At the beginning of configuration, FPGAs ignore all data prior to the preamble but counts the number of data bits preceding the preamble, and the LengthCount counter increments for every rising CCLK edge, even the ones proceeding the preamble.

programmable interconnect point

See *PIP*.

$\overline{\text{PROGRAM}}$ pin

The $\overline{\text{PROGRAM}}$ pin is an active-Low input that forces clearing of the FPGA configuration memory and is used to initiate a configuration cycle. While $\overline{\text{PROGRAM}}$ is held Low, the FPGA drives $\overline{\text{INIT}}$ Low and continues to clear the configuration memory. When $\overline{\text{PROGRAM}}$ goes High, the FPGA finishes the current clear cycle, executes another complete clear cycle, goes into a WAIT state, and releases $\overline{\text{INIT}}$.

readback

Initiating a readback causes the configuration memory to become accessible to be serially clocked out and read from the device, or (byte-wide in SelectMAP/Slave Parallel modes). The configuration memory contains the configuration data, facilitating a Read-Verification of the data. The configuration memory can also contain the CLB output logic states facilitating a Read-Capture of the internal logic states.Read-Verification and Read-Capture are used by the Hardware Debugger for hardware verification. For information on the readback specification and timing, refer to *The Programmable Logic Data Book*. For information on using the readback component in a design, refer to the *Libraries Guide*. For information on enabling the readback function in the Implementation Software, refer to the *Development System Reference Guide*. For information on using the Hardware Debugger refer to the *Hardware Debugger Reference/User Guide*. For information on connecting the XChecker cable for readback, refer to the *Hardware Users Guide*.

readback data

Configuration data read from a Virtex-II device. The data is organized as configuration frames.

routing

The interconnection or the process of creating the desired interconnection of logic cells to make them perform the desired function. Routing follows after partitioning and placement.

schematic

Graphic representation of a logic design in the form of interconnected gates, flip-flops, and larger blocks. Older and more visually intuitive alternative to the increasingly more popular equation-based or high-level language textual description of a logic design.

SelectMAP interface

One of the configuration interfaces on the Virtex-II device. This is a byte-serial interface. The pins in the SelectMAP interface can be used as user I/O after configuration has been completed or remain configured as a configuration interface.

SelectRAM

Xilinx-specific name for RAM implemented in CLBs.

simulation

Computer modeling of logic and (sometimes) timing behavior of logic driven by simulation inputs (stimuli or vectors).

slice

A subdivision of the Virtex-II CLB. There are four vertical slices in each Virtex-II CLB. Each slice contains two Logic Cells.

SRAM

Static random access memory. Read-Write memory with data stored in latches. Faster than DRAM and with simpler timing requirements, but smaller in size and about four times more expensive than DRAM of the same capacity.

static timing

Detailed description of on-chip logic and interconnect delays.

submicron

The smallest feature size is usually expressed in micron (μ = millionth of a meter, or a thousandth of a millimeter). The state of the art is moving from 0.35μ to 0.25μ and soon may reach 0.18μ. The wavelength of visible light is 0.4μ to 0.8μ. 25.4μ = 1 mil, a thousandth of an inch.

synchronous

Circuitry that changes state only in response to a common clock, as opposed to asynchronous circuitry that responds to a multitude of derived signals. Synchronous circuits are easier to design, debug, modify, and better tolerate parameter changes and speed upgrades than asynchronous circuits.

sync word

A 32-bit word with a value that is used to synchronize the configuration logic.

synthesis

Optimization process of adapting a logic design to the logic resources available on the chip, like look-up tables, Longline, and dedicated carry. Synthesis precedes mapping.

TBUFs

Buffers with a 3-state option, where the output can be made inactive. Used for multiplexing different data sources onto a common bus. The pulldown-only option can use the bus as a "wired AND" function.

timing

Relating to delays, performance, or speed.

timing driven

A design or layout method that takes performance requirements into consideration.

UART

Universal asynchronous receiver/transmitter. An 8-bit parallel-to-serial and serial-to-parallel converter, combined with parity and start-detect circuitry, and sometimes even FIFO buffers. Used widely in asynchronous serial communications interface, e.g., modems.

USB

Universal Serial Bus, A low-cost, low-speed, self-clocking bit-serial bus (1.5 MHz and 12 MHz) using four wires (V_{CC}, ground, differential data) to daisy-chain up to 128 devices.

VME

Older bus standard, popular with MC68000-based industrial computers.

$\overline{\text{WRITE}}$ pin

The $\overline{\text{WRITE}}$ pin is an input to Virtex-II devices in the SelectMAP/Slave Parallel mode, indicating to the device which direction data is flowing on the Data bus. When $\overline{\text{WRITE}}$ is asserted (Low), data is entering the device (configuration). When $\overline{\text{WRITE}}$ is de-asserted (High), data is leaving the device (readback). If $\overline{\text{WRITE}}$ changes state when the device isn't expecting it, an abort occurs. For more information on the $\overline{\text{WRITE}}$ pin, refer to *The Programmable Logic Data Book* and "Design Considerations" on page 155.

XChecker cable

The Xilinx XChecker Cable (model DLC4) is a serial download cable. The XChecker uses a serial 9-pin interface to the communication port of a host computer and two 8-pin headers for flying-wire connectors to a target board. The XChecker cable is supported by the Hardware Debugger software for performing Slave Serial configuration and readback of FPGAs. The XChecker cable is also supported by the JTAG Programmer software for performing Slave Serial and Boundary Scan configuration of FPGAs, and Boundary Scan programming of CPLDs. For more information on using the XChecker cable refer to the *Hardware Users Guide* and the *Hardware Debugger Reference/Users Guide*.

XNF file

Xilinx-proprietary description format for a logic design. Alternative is EDIF.

Index

XILINX®

Xilinx Sales Offices

Headquarters

2100 Logic Drive
San Jose, CA 95124
Tel: (408) 559-7778
Fax: (408) 559-7114
TWX: (510) 600-8750

North America

Madison, AL
Tel: (256) 722-4050
Fax: (256) 722-9912

Phoenix, AZ
Tel: (480) 753-4503
Fax: (480) 753-4504

Irvine, CA
Tel: (949) 727-0780
Fax: (949) 727-3128

San Diego, CA
Tel: (858) 558-5974
Fax: (858) 558-6418

Sunnyvale, CA
Tel: (408) 245-9850
Fax: (408) 245-9865

Greenwood Village, CO
Tel: (303) 220-7541
Fax: (303) 220-8641

Winter Park, FL
Tel: (407) 673-8661
Fax: (407) 673-8663

Schaumburg, IL
Tel: (847) 605-1972
Fax: (847) 605-1976

Deephaven, MN
Tel: (612) 473-4816
Fax: (612) 473-5060

Marriottsville, MD
Tel: (410) 442-9748
Fax: (410) 442-9749

Nashua, NH
Tel: (603) 891-1098
Fax: (603) 891-0890

Ledgewood, NJ
Tel: (973) 584-7199
Fax: (973) 584-1390

Raleigh, NC
Tel: (919) 846-3922
Fax: (919) 846-8316

Brecksfield, OH
Tel: (330) 659-3131
Fax: (330) 659-9254

Portland, OR
Tel: (503) 293-9016
Fax: (503) 293-3858

West Chester, PA
Tel: (610) 430-3300
Fax: (610) 430-0470

Dallas, TX
Tel: (972) 960-1043
Fax: (972) 960-0927

Salt Lake City, UT
Tel: (801) 268-3434
Fax: (801) 266-9021

Bellevue, WA
Tel: (425) 451-7000
Fax: (425) 990-8989

Canada

Oakville, Ontario Canada
Tel: (905) 337-0894
Fax: (905) 337-3554

Kanata, Ontario Canada
Tel: (613) 271-5264
Fax: (613) 592-4256

European Headquarters

Benchmark House, 203 Brooklands Rd.
Weybridge Surrey KT13 0RH
United Kingdom
Tel: (44)1-870-7350-600
Fax: (44)1-870-7350-601

Benelux

Tel : (32)-53-848310
Fax: (32)-53-848311

France and Spain

Tel: (33) 1-34-63-01-01
Fax: (33) 1-34-63-01-09

Germany, Switzerland, and Austria

Tel: (49) 89-93088-0
Fax: (49) 89-93088-188

Italy

Tel: (39) 02-487-12-101
Fax: (39) 02-400-94-700

Sweden, Norway, Denmark, and Finland

Tel: (46) 8-594-61-660
Fax: (46) 8-594-61-661

United Kingdom and Ireland

Tel: (44) 870-7350-603
Fax: (44) 870-7350-604

Japan

Tel: (81) 3-5321-7711
Fax: (81) 3-5321-7765

Asia Pacific Headquarters

Tel: 852-2-424-5200
Fax: 852-2-494-7159

Korea

Tel : 822-761-4277
Fax : 822-761-4278

Shanghai

Tel: +86-21-6886-2323, 2322
Fax: +86-21-6886-2333

Taiwan

Tel: 886-2-2174-1388
Fax: 886- 2-2758-8367

For information on Xilinx North American Sales Representative offices, see http://www.xilinx.com/company/sales/na_reps.htm

For information on Xilinx International Sales Representative offices, see http://www.xilinx.com/company/sales/int_reps.htm