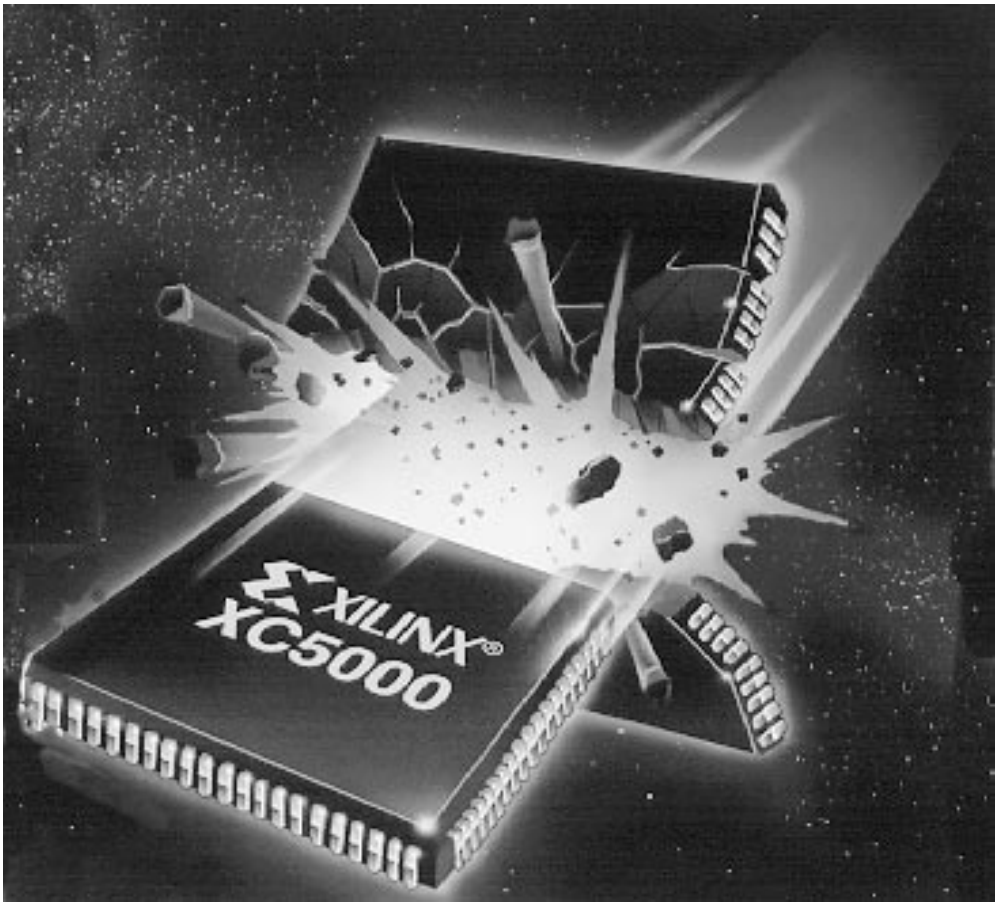
XILINX®

Technical
Data

XC5200
Libraries Guide
Supplement



Preliminary (v2.0) • May 1995

Conventions

The following conventions are used in this manual's syntactical statements:

`Courier font regular`	System messages or program files appear in regular Courier font.
`Courier font bold`	Literal commands that you must enter in syntax statements are in bold Courier font.
italic font	Variables that you replace in syntax statements are in italic font.
`[ ]`	Square brackets denote optional items or parameters. However, in bus specifications, such as bus [7:0], they are required.
`{ }`	Braces enclose a list of items from which you must choose one or more.
. . .	A vertical ellipsis indicates material that has been omitted.
. . .	A horizontal ellipsis indicates that the preceding can be repeated one or more times.
|	A vertical bar separates items in a list of choices.
↵	This symbol denotes a carriage return.

Contents

Chapter 1

Xilinx Unified Libraries

Xilinx maintains software libraries with thousands of functional design elements (primitives and macros) for different device architectures. New functional elements are assembled with each release of development system software. The latest catalog of design elements are known as “Unified Libraries.” Elements in these libraries are common to all Xilinx device architectures. This “unified” approach means that you can use your circuit design created with “unified” library elements across all current Xilinx device architectures that recognize the element you are using.

Elements that exist in multiple architectures look and function the same, but their implementations might differ to make them more efficient for a particular architecture. A separate library still exists for each architecture and common symbols are duplicated in each one, which is necessary for simulation (especially board level) where timing depends on a particular architecture.

Overview

The *XACT XC5200 Libraries Guide* lists the XC4000 library elements that are not supported, and describes the new and / or modified primitives and macro logic elements for the XC5200 architecture. Also addressed are attributes and constraints that apply uniquely to the XC5200 architecture. To use this libraries guide effectively, you will need the *XACT Libraries Guide* (0401098 01) issued April, 1994 as a reference.

This book is organized into three parts.

- Unsupported XC4000 design elements
- New XC5200 design elements
- Constraints and attributes

Unsupported XC4000 Design Elements

The design elements that are not supported by the XC5200 architecture are listed for each functional category. The remaining XC4000 library elements described in the 1994 *XACT Libraries Guide* are available in the XC5200 library. The common elements will look and function the same, although the implementation may be different to maximize efficiency within the XC5200 architecture.

New XC5200 Design Elements

A list of new XC5200 library elements is provided, organized by function.

The following information is provided for each library element.

- Graphical symbol
- Functional description
- Primitive or macro designation
- Truth table (when applicable)
- Schematic for macros

Note: Design elements with bussed or multiple I/O pins typically include just one schematic.

Attributes and Constraints

The "Attributes and Constraints" chapter provides information on the unique XC5200 attributes and constraints. Attributes are instructions placed on symbols or nets in a schematic to indicate their placement, implementation, naming, directionality, and so forth. Constraints are a type of attribute used only to indicate where an element should be placed.

Chapter 2

Unsupported XC4000 Design Elements

XC4000 library elements listed in this chapter are not supported in XC5200. Where an entire functional category has no XC4000 elements supported in XC5200, the list is condensed to the notation "**None** supported."

For completeness, functional categories whose XC4000 elements are supported without exception in XC5200 appear with the notation "All supported."

Arithmetic Functions

All supported

Buffers

- BUFGP
- BUFGS
- BUFOD

Comparators

All supported

Counters

All supported

Data Registers

All supported

Decoders

All supported; see page 3-5 for specific XC5200 implementation of DECODE4, DECODE8, and DECODE16.

Encoders

All supported

Flip-Flops

All supported

General

- BSCAN (modified for XC5200)
- STARTUP (modified for XC5200)
- OSC4

I/O Flip-Flops

- IFDI
- OFDEI
- OFDTI

I/O Functions

All supported

Input Latches

- ILDI
- ILDI_1

Logic Primitives

- WAND1
- WAND4
- WAND8
- WAND16
- WOR2AND

Map Elements

All supported

Memory Elements

None supported

Multiplexers

All supported

Shift Registers

All supported

Shifters

All supported

Chapter 3

New XC5200 Design Elements

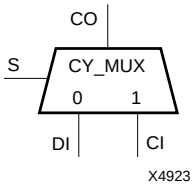
This chapter contains new design elements for the XC5200 architecture, listed in functional categories.

Categories that contain no new XC5200 elements appear with the notation “None.”

Arithmetic Functions

CY_MUX

2-to-1 Multiplexer for Carry Logic (Primitive)



CY_MUX is used to implement a 1-bit high-speed carry propagate function. One such function can be implemented per logic cell (LC), for a total of four bits per configurable logic block (CLB). The Direct Input (DI) of an LC is connected to the DI input of the CY_MUX. The Carry In (CI) input of an LC is connected to the CI input of the CY_MUX. The select input (S) of the CY_MUX is driven from the output of the Lookup Table (LUT), configured as an XOR function. The output (CO) of the CY_MUX reflects the state of the selected input and implements the Carry Out function of each LC. When Low, S selects DI; when High, S selects CI.

Inputs			Outputs
S	**DI**	**CI**	**CO**
0	1	X	1
0	0	X	0
1	X	1	1
1	X	0	0

Figure 3-1 illustrates the application of the CY_MUX for a 4-bit adder. Also shown are the associated FMAP symbols and the CY_INIT function.

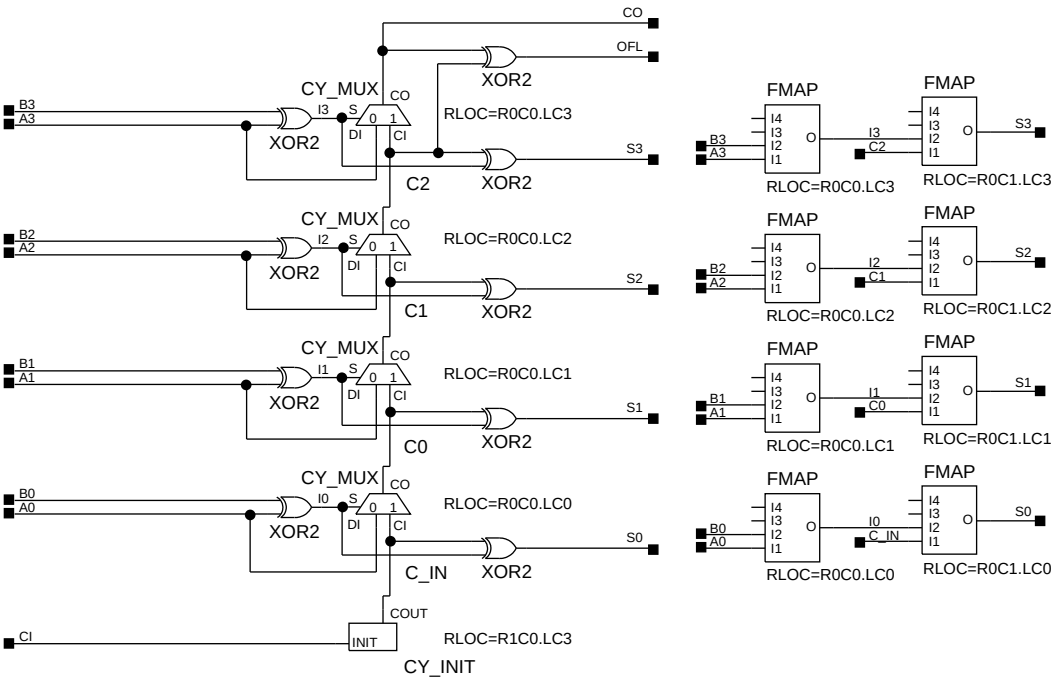
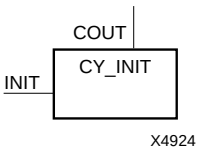


Figure 3-1 4-Bit Adder Schematic

CY_INIT

Initialization Stage for Carry Chain (Macro)



CY_INIT is used in conjunction with the CY_MUX element to implement a high-speed carry function. The CY_INIT element is placed in any logic cell of the CLB immediately below the CLB containing the first carry element (CY_MUX), within the same column of CLBs. The INIT input is driven from the Direct Input (DI) within LC3. The CY_INIT output CO drives the C_{in} input of the first LC in the carry chain. The CO output reflects the state of the DI input. Figure 3-1 illustrates the application of the CY_INIT element for a 4-bit adder.

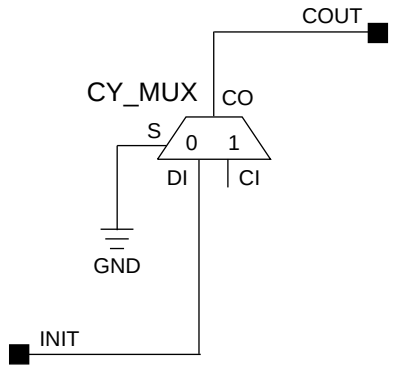


Figure 3-2 CY_INIT Implementation

Buffers

None

Comparators

None

Counters

None

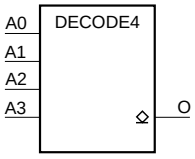
Data Registers

None

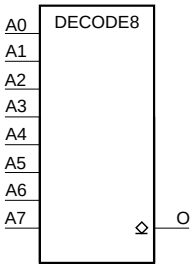
Decoders

DECODE4, DECODE8, DECODE16, DECODE32, and DECODE64

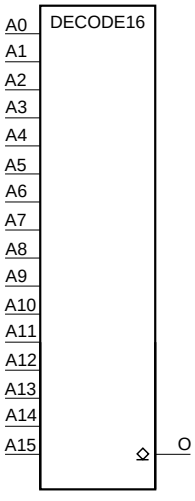
4-, 8-, 16-, 32-, and 64-Bit Active Low Decoders (Macros)



X3907



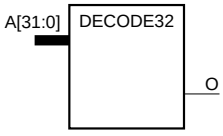
X3908



X3909

These decoders are implemented by cascading CY_MUX elements driven from Lookup Tables (LUTs). When one or more of the inputs (A_n) are Low, the output (O) is Low. When all of the inputs are High, the output is High. Patterns can be decoded by adding inverters to inputs.

Inputs				Outputs
A_0	A_1	...	A_{n-1}	O
1	1	1	1	1
0	X	X	X	0
X	0	X	X	0
X	X	X	0	0



X4977

X4978

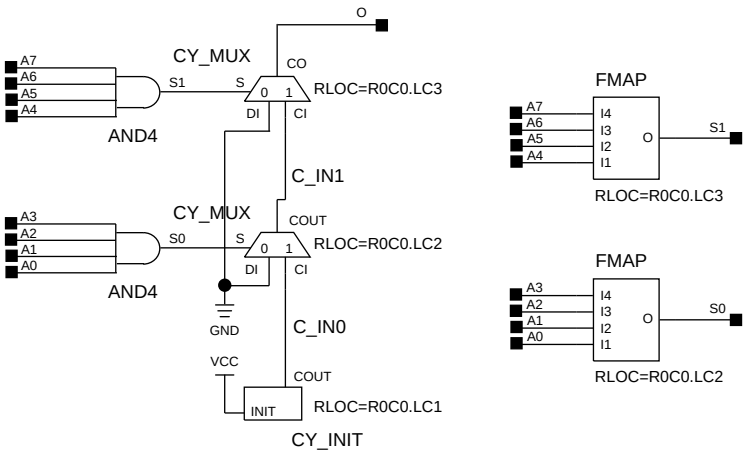
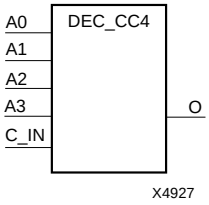


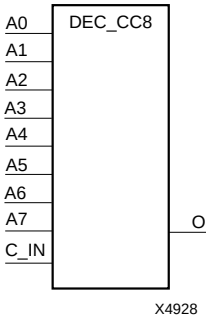
Figure 3-3 DECODE8 XC5200 Implementation

DEC_CC4, DEC_CC8, and DEC_CC16

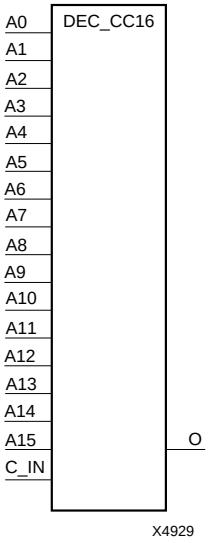
4-, 8-, and 16-Bit Active Low Decoders (Macros)



These decoders are used to build wide decoder functions. They are implemented by cascading CY_MUX elements driven from Lookup Tables (LUTs). The C_IN pin can only be driven by a CY_INIT or by the output (O) of a previous decode stage. When one or more of the inputs (A_n) are Low, the output is Low. When all the A_n inputs are High and the C_IN input is High, the output is High. Patterns can be decoded by adding inverters to inputs.



Inputs					Outputs
A_0	A_1	…	A_{n-1}	C_IN	O
1	1	1	1	1	1
X	X	X	X	0	0
0	X	X	X	X	0
X	0	X	X	X	0
X	X	X	0	X	0



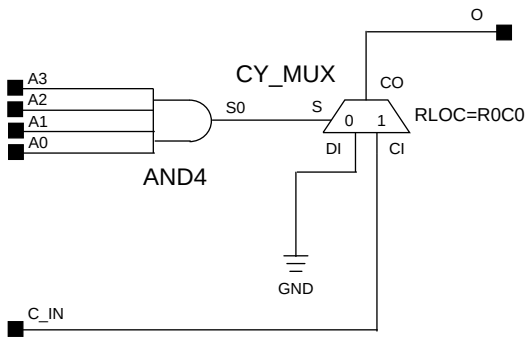


Figure 3-4 DEC_CC4 Implementation

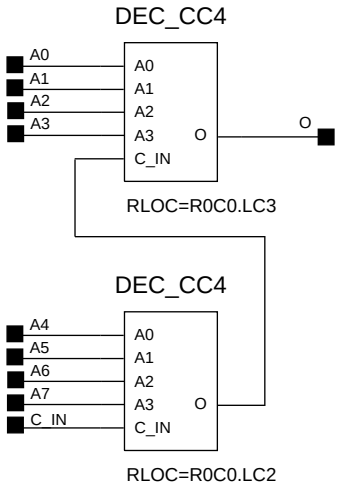


Figure 3-5 DEC_CC8 Implementation

Encoders

None

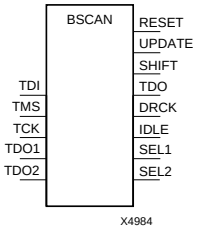
Flip-Flops

None

General

BSCAN

Boundary Scan Logic Control Circuit (Primitive)



The BSCAN symbol indicates that boundary scan logic should be enabled after PLD configuration is complete. It also provides optional access to some special features of the XC5200 boundary scan logic. An overview of the boundary scan interface follows; for complete details, refer to the application note “Boundary Scan in XC4000 Devices” in the Programmable Logic Data Book.

To indicate that boundary scan remain enabled after configuration, connect the BSCAN symbol to the TDI, TMS, TCK, and TDO pads shown in Figure 3-6. The other pins on BSCAN do not need to be connected, unless those special functions are needed. A signal on the TDO_1 input is passed to the external TDO output when the USER1 instruction is executed; the SEL_1 output goes High to indicate that the USER1 instruction is active. The TDO_2 and SEL_2 pins perform a similar function for the USER2 instruction. The DRCK output provides access to the data register clock (generated by the TAP controller). The IDLE output provides access to a version of the TCK input, which is only active while the TAP controller is in the Run-Test-Idle state. The RESET, UPDATE, and SHIFT pins represent the decoding of the corresponding state of the boundary scan internal state machine.

If boundary scan is used only before configuration is complete, do not include the BSCAN symbol in the design, so the TDI, TMS, TCK, and TDO pins can be used for user functions.

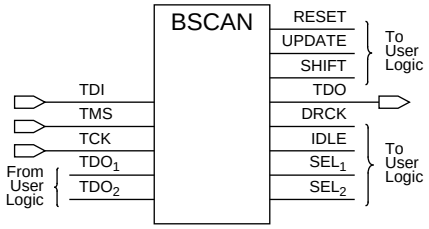
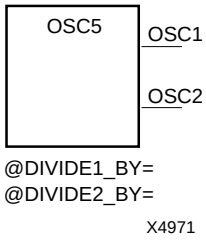


Figure 3-6 Boundary Scan Representation

OSC5

Internal Multiple-Frequency Clock-Signal Generator (Macro)



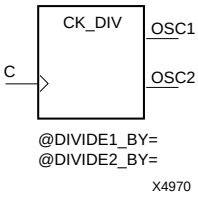
OSC5 provides internal clock signals in applications where timing is not critical. The available frequencies are determined by FPGA device components which are process dependent. Therefore, the available frequencies vary from device to device. Only one OSC5 may be used per design. The OSC5 is not available if the CK_DIV element is used.

The clock frequencies of the OSC1 and OSC2 outputs are determined by specifying the DIVIDE1_BY=n_1 attribute for the OSC1 output, and the DIVIDE2_BY=n_2 attribute for the OSC2 output. n_1 and n_2 are integer numbers by which the internal 16-MHz clock is divided to produce the desired clock frequency. The available frequency options appear in the table below.

n_1	OSC1 Frequency	n_2	OSC2 Frequency
4	4 MHz	2	8 MHz
16	1 MHz	8	2 MHz
64	250 kHz	32	500 kHz
256	63 kHz	128	125 kHz
		1024	16 kHz
		4096	4 kHz
		16384	1 kHz
		65536	244 Hz

CK_DIV

Internal Multiple-Frequency Clock Divider (Macro)



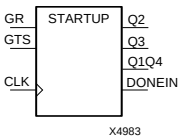
CK_DIV divides a user-provided external clock signal with different frequency options on either or both of the outputs. Only one CK_DIV may be used per design. The CK_DIV is not available if the OSC5 element is used.

The clock frequencies of the OSC1 and OSC2 outputs are determined by specifying the DIVIDE1_BY=n_1 attribute for the OSC1 output, and the DIVIDE2_BY=n_2 attribute for the OSC2 output. n_1 and n_2 are integer numbers by which the clock input (C) is divided to produce the desired output clock frequency. The available combinations of n_1 and n_2 appear in the table below.

n_1	n_2
4	2
16	8
64	32
256	128
	1024
	4096
	16384
	65536

STARTUP

User Interface to Global Clock, Reset, and 3-State Controls (Macro)



STARTUP is used for Global Reset, global 3-state control, and the user configuration clock. The Global Reset (GR) input, when High, resets every flip-flop in the device. Following configuration, the global 3-state control (GTS), when High, forces all the I/O block (IOB) outputs into High impedance mode, which isolates the device outputs from the circuit but leaves the inputs active.

The configuration clock input (CLK) must be connected to a user clock if the start-up of the device is synchronized with the user clock. Also, "user clock" must be selected in the MakeBits program.

The STARTUP outputs (Q2, Q3, Q1Q4, and DONEIN) display the progress/status of the start-up process following the configuration.

I/O Flip-Flops

None

I/O Functions

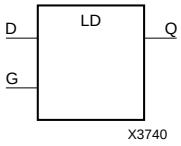
None

Input Latches

None

Latches

LD



Transparent Data Latch (Macro)

The data output (Q) of the latch reflects the data (D) input while the gate (G) input is High. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains Low.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs		Outputs
G	**D**	**Q**
1	0	0
1	1	1
0	X	No Chg
↓	D	d

d = state of input one setup time prior to High-to-Low gate transition

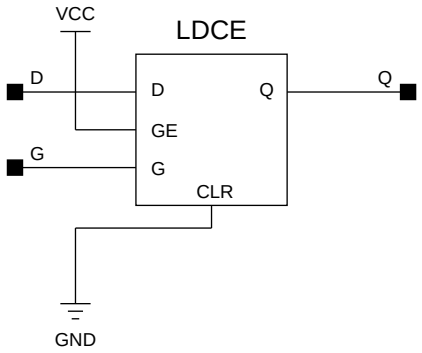
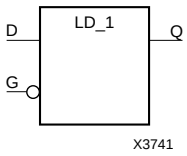


Figure 3-7 LD Implementation

LD_1

Transparent Data Latch with Inverted Gate (Macro)



The data output (Q) of the latch reflects the data (D) input while the gate (G) input is Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs		Outputs
G	**D**	**Q**
0	0	0
0	1	1
1	X	No Chg
↑	D	d

d = state of input one setup time prior to Low-to-High gate transition

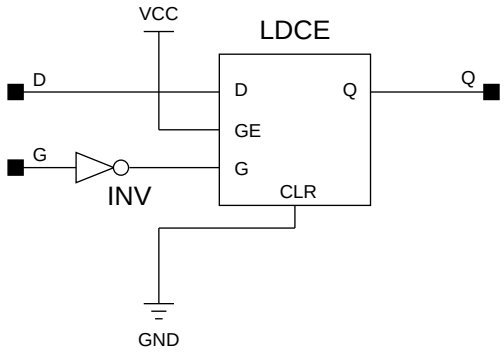
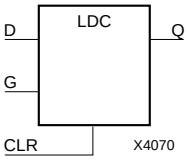


Figure 3-8 LD_1 Implementation

LDC

Transparent Data Latch with Asynchronous Clear (Macro)



When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. Q reflects the data (D) input while the gate (G) input is High and CLR is Low. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains low.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs			Outputs
CLR	**G**	**D**	**Q**
1	X	X	0
0	1	0	0
0	1	1	1
0	0	X	No Chg
0	↓	D	d

d = state of input one setup time prior to High-to-Low gate transition

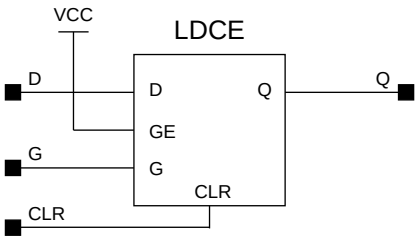
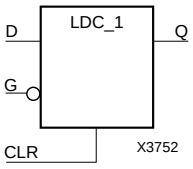


Figure 3-9 LDC Implementation

LDC_1

Transparent Data Latch with Asynchronous Clear and Inverted Gate Input (Macro)



When the asynchronous clear input (CLR) is High, it overrides the other inputs (D and G) and resets the data (Q) output Low. Q reflects the data (D) input while the gate (G) input and CLR are Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs			Outputs
CLR	**G**	**D**	**Q**
1	X	X	0
0	0	0	0
0	0	1	1
0	1	X	No Chg
0	↑	D	d

d = state of input one setup time prior to Low-to-High gate transition

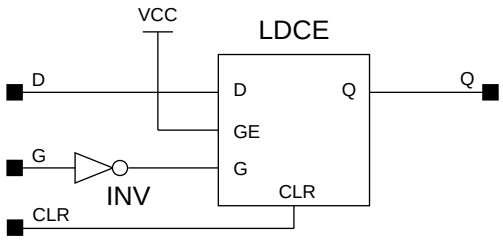
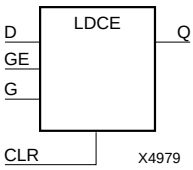


Figure 3-10 LDC_1 Implementation

LDCE

Transparent Data Latch with Asynchronous Clear and Gate Enable (Primitive)



When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. Q reflects the data (D) input while the gate (G) input and gate enable (GE) are High and CLR is Low. If GE is Low, data on D cannot be latched. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G or GE remains low.

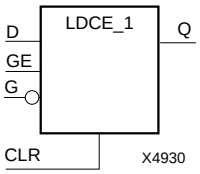
The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs				Outputs
CLR	**GE**	**G**	**D**	**Q**
1	X	X	X	0
0	0	X	X	No Chg
0	1	1	0	0
0	1	1	1	1
0	1	0	X	No Chg
0	1	↓	D	d

d = state of input one setup time prior to High-to-Low gate transition

LDCE_1

Transparent Data Latch with Asynchronous Clear and Gate Enable and Inverted Gate Input (Macro)



When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. Q reflects the data (D) input while the gate (G) input and CLR are Low and gate enable (GE) is High. If GE is Low, data on D cannot be latched. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High or GE remains Low.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs				Outputs
CLR	**GE**	**G**	**D**	**Q**
1	X	X	X	0
0	0	X	X	No Chg
0	1	0	0	0
0	1	0	1	1
0	1	1	X	No Chg
0	1	↑	D	d

d = state of input one setup time prior to Low-to-High gate transition

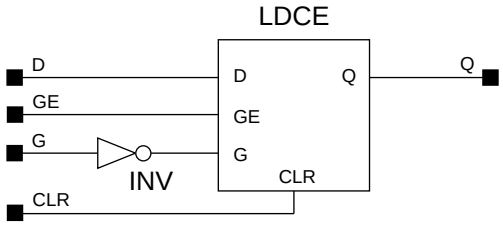
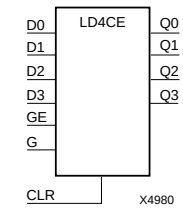
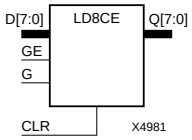


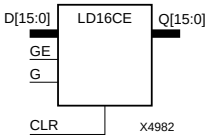
Figure 3-11 LDCE_1 Implementation

LD4CE, LD8CE, and LD16CE

Transparent Data Latches with Asynchronous Clear and Gate Enable (Macros)







LD4CE, LD8CE, and LD16CE have 4, 8, and 16 transparent data latches, respectively. When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) outputs Low. Q reflects the (D) inputs while the gate (G) input is High, gate enable (GE) is High, and CLR is Low. If GE is Low, data on D cannot be latched. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G or GE remains Low.

The latch is reset, output Low, when power is applied or when global reset (GR) is active.

Inputs				Outputs
CLR	**GE**	**G**	**Dn**	**Qn**
1	X	X	X	0
0	0	X	X	No Chg
0	1	1	0	0
0	1	1	1	1
0	1	0	X	No Chg
0	1	↓	Dn	dn

Dn = referenced input, for example, D0, D1, D2

Qn = referenced output, for example, Q0, Q1, Q2

dn = referenced input state, one setup time prior to High-to-Low gate transition

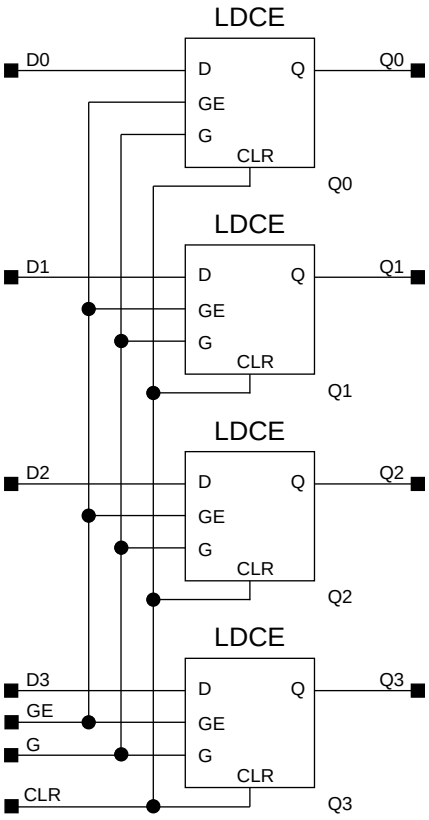


Figure 3-12 LD4CE Implementation

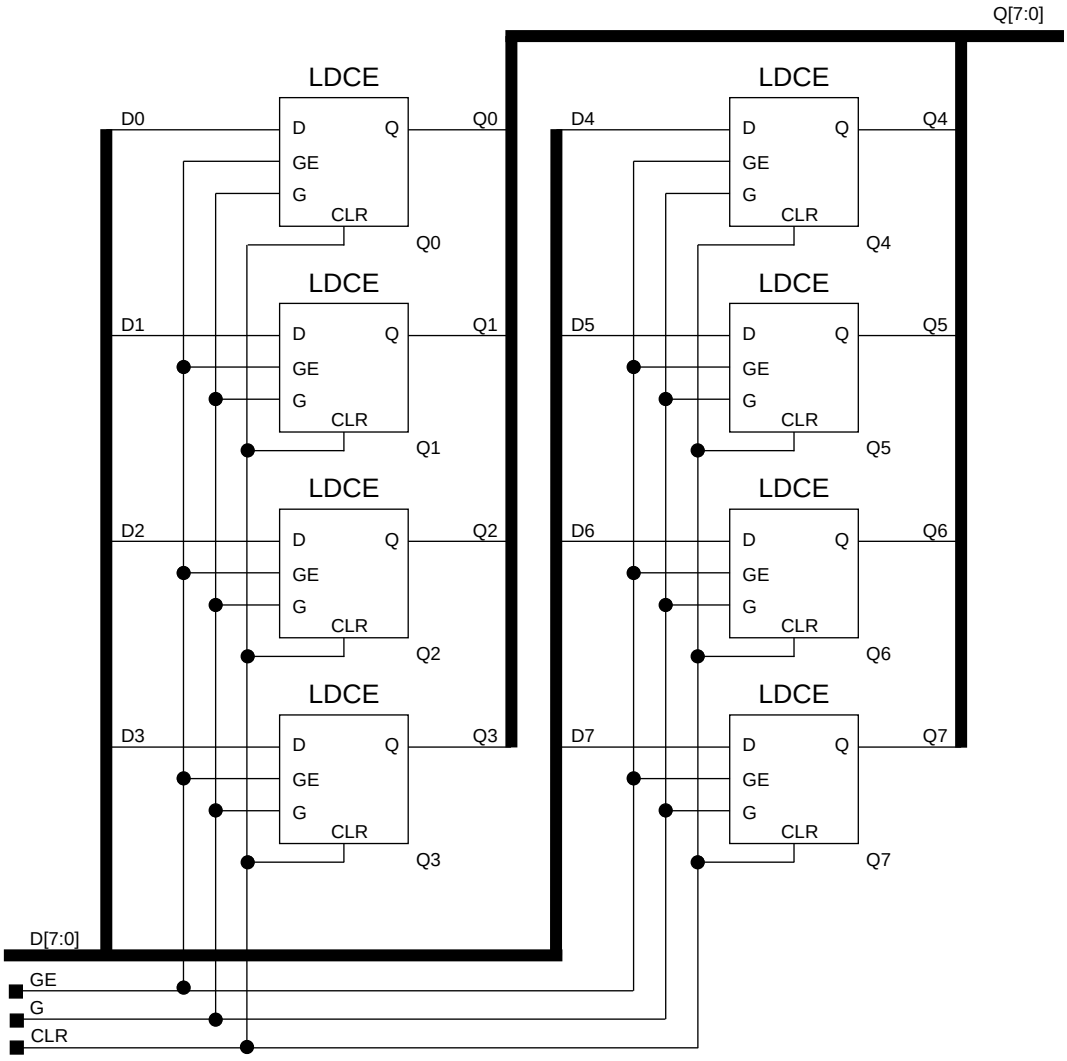


Figure 3-13 LD8CE Implementation

Logic Primitives

AND

12- and 16-Input AND Gates with Non-Inverted Inputs (Macros)

The 12- and 16-input AND functions are available only with non-inverting inputs. To invert some or all inputs, use external inverters.

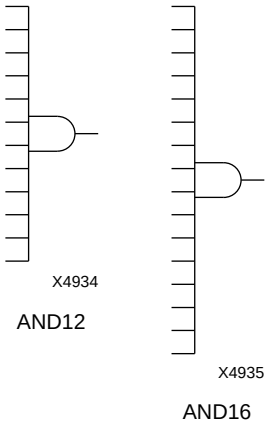


Figure 3-14 AND Gate Representations

OR

12- and 16-Input OR Gates with Non-Inverted Inputs (Macros)

The 12- and 16-input OR functions are available only with non-inverting inputs. To invert some or all inputs, use external inverters.

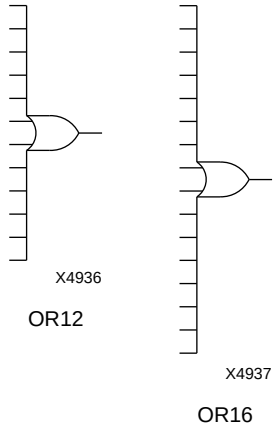


Figure 3-15 OR Gate Representations

NAND

12- and 16-Input NAND Gates with Non-Inverted Inputs (Macros)

The 12- and 16-input NAND functions are available only with non-inverting inputs. To invert some or all inputs, use external inverters.

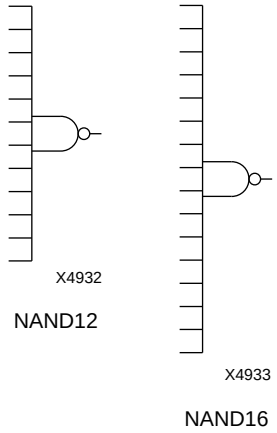


Figure 3-16 NAND Gate Representations

NOR

12- and 16-Input NOR Gates with Non-Inverted Inputs (Macros)

The 12- and 16-input NOR functions are available only with non-inverting inputs. To invert some or all inputs, use external inverters.

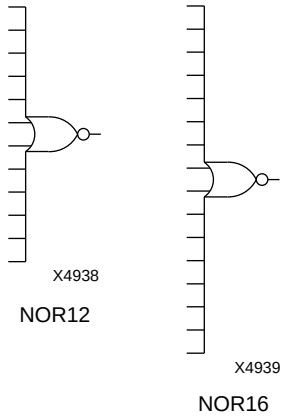
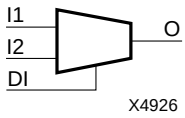


Figure 3-17 NOR Gate Representations

F5_MUX

2-to-1 Lookup Table Multiplexer (Primitive)



F5_MUX provides a multiplexer function within half of a CLB. The output from the Lookup Table (LUT) in LC1 is connected to the I_1 input of the F5_MUX and the output from the LUT in LC0 is connected to the I_2 input. The Direct Input (DI) of LC0 is connected to the DI input of the F5_MUX. The output (O) reflects the state of the selected input. When Low, DI selects I_1; when High, DI selects I_2. Similarly, the F5_MUX can connect to the LUTs in LC2 and LC3. The F5_MUX is also used within the F5MAP element to implement any 5-input function within the top or bottom half of a CLB.

Inputs			Outputs
DI	**I_1**	**I_2**	**O**
0	1	X	1
0	0	X	0
1	X	1	1
1	X	0	0

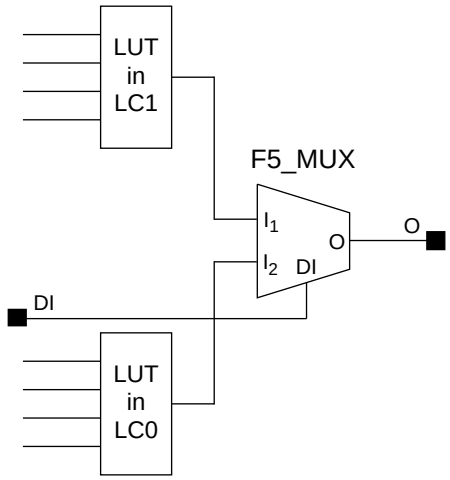
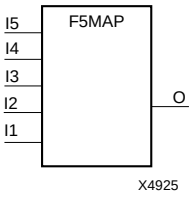


Figure 3-18 F5_MUX Representation

Map Elements

F5MAP

5-Input Function Partitioning Control Symbol



The F5MAP symbol is used to control logic partitioning of 5-input functions into the top or bottom half of a CLB. The F5MAP symbol is not a substitute for logic. It is used in addition to combinatorial gates for mapping control.

At the schematic level, any 5-input logic function can be implemented using gates, and mapped into half of a single CLB by using the F5MAP symbol. The signals that are the inputs and outputs of the 5-input function must be labelled and connected to appropriate pins of the F5MAP symbol, or the F5MAP signals and logic signals must have identical labels. The symbol can have unconnected pins, but all signals on the logic group to be mapped must be specified on a symbol pin.

The use of the F5MAP forces any 5-input function to be implemented by two Lookup Tables (LUTs), the Direct Input (DI), and the F5_MUX primitive, all contained within adjacent CLB logic cells LC0 and LC1 or LC2 and LC3. Figure 3-19 illustrates the connections within a CLB.

An F5MAP primitive example is shown in Figure 3-20.

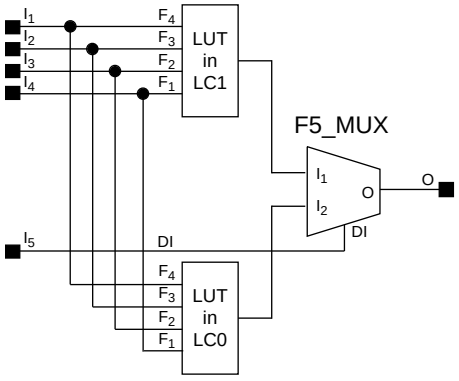


Figure 3-19 Two LUTs in Parallel Combined to Create a 5-Input Function

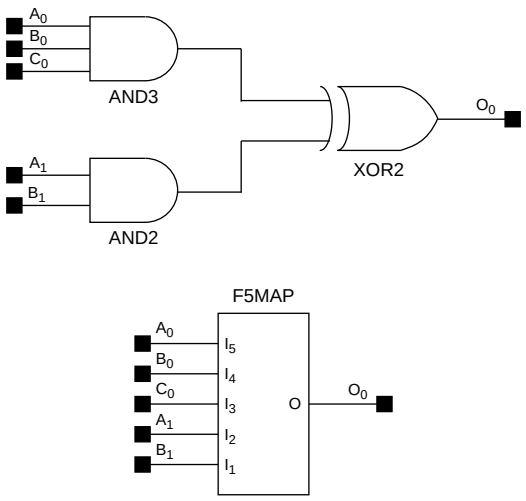


Figure 3-20 F5MAP Primitive Example

Multiplexers

None

Shift Registers

None

Shifters

None

Chapter 4

Attributes and Constraints

The following attributes and constraints are new or changed for the XC5200.

- DIVIDE1_BY
- DIVIDE2_BY
- NODELAY
- RLOC

DIVIDE1_BY and DIVIDE2_BY

DIVIDE1_BY=*value* and DIVIDE2_BY=*value* attributes are user-defined parameters added to the schematic for the OSC5 and CK_DIV symbols.

These attributes have the following syntax:

```
DIVIDE1_BY={4|16|64|256}
DIVIDE2_BY={2|8|32|128|1024|4096|16384|65536}
```

For the OSC5 symbol, these attributes define the amount by which the internal 16-MHz clock is to be divided for the OSC1 and / or OSC2 outputs. The DIVIDE1_BY attribute applies to the OSC1 output and the DIVIDE2_BY attribute applies to the OSC2 output.

For the CK_DIV symbol, the attributes define the amount by which the user specifies that clock input (C) is to be divided.

NODELAY

The default configuration of flip-flops and data latches includes an input delay that results in no external hold time on the input data path. However, this delay can be removed by placing the NODELAY attribute on input buffers (IBUFs), resulting in a smaller setup time but a positive hold time.

The syntax of the NODELAY attribute is the following:

NODELAY

RLOC

RLOC=*value* is a user-, system-, or library-defined parameter added to schematic symbols to control relative placement of the elements contained in the schematic. The RLOC parameter has the following syntax:

RLOC=R*row#*C*column#*[.*extension*]

where the row and column numbers of the LCA grid array can be any positive integer or zero.

The optional *.extension* further denotes positioning of FMAPs, flip-flops, latches, and CY_MUX symbols within the appropriate Logic Cell of the CLB. Its value can be LC0, LC1, LC2, or LC3.

The RLOC value cannot specify a range or a list of several locations; it must specify a single location.

For more details on the RLOC constraints, refer to the 1994 *XACT Libraries Guide*.