XAPP196 (v1.0) December 15, 2000

Interfacing a Virtex-E Device to a Pentium Processor

Summary

This application note describes a reference design for a Virtex™-E FPGA interface to an Intel Pentium™ processor. The Pentium I system bus, design concerns, and possible applications of this design are discussed. Additionally, the differences between the Pentium I, II, and III busses are discussed. For more information specific to the Intel Pentium family of processors, see the Intel developer website (http://developer.intel.com/).

Design Discussion

General

The design was synthesized using FPGA Express 3.4 (Verilog) and implemented using Foundation ISE version 3.1i.

Internally, the FPGA design has been constrained to run at 100 MHz. However, the Intel Pentium processor system bus runs at only 66 MHz. To account for this discrepancy, two asynchronous FIFOs are used allowing the internal design to run at a higher frequency than the bus (Figure 3). Data is taken from the bus at 66 MHz, processed at a higher speed, and returned to the processor over the bus at 66 MHz. For more information, see the Pentium data sheet at (**http://developer.intel.com/design/intarch/pentium/pentium.htm**).

The Pentium I and II system busses run at 66 MHz. The Pentium III system bus runs at either 100MHz or 133 MHz. These processors are used when fast bus transactions are necessary. A discussion of how to adapt the reference design to the Pentium II and III architecture is included on page 4.

The reference design was created using an Intel Pentium 166 MHz embedded processor, and a Spartan™-II XC2S50. This design can also be implemented in a Virtex-E XCV50E. Figure 1 shows the block diagram used and the design interface connections.

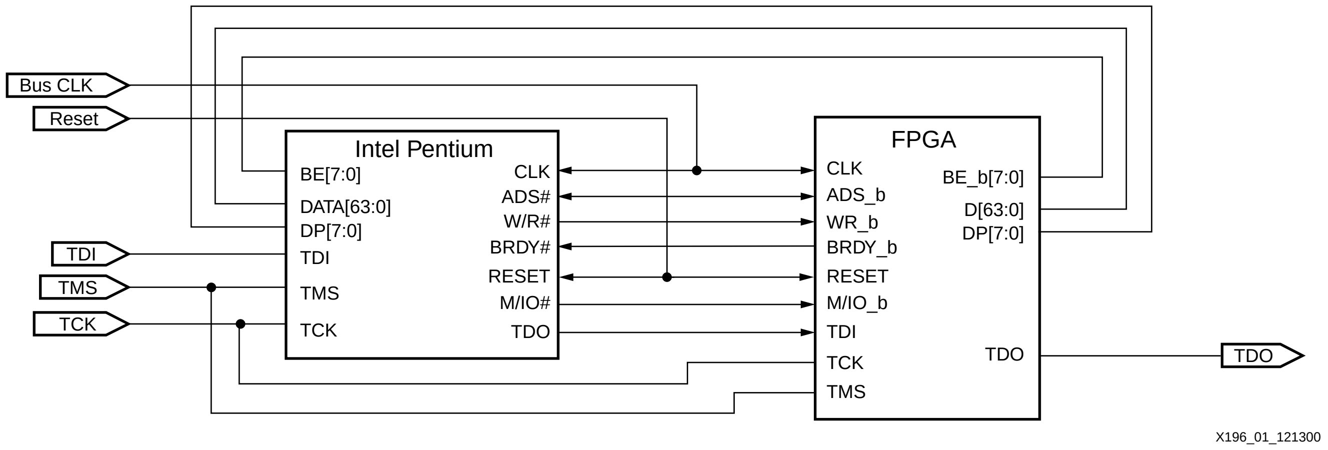

Figure 1: **Virtex or Spartan-II FPGA to Pentium Processor Interface**

Signal Description

Table 1 lists all the command signals associated with the design and bus transactions.

Table 1: **Command Signals for Design and Bus Transactions**

Signal Name	Direction	Function
CLK	Input	System Bus Clock
ADS_b	Input	Address Strobe. Signals a new transaction when pulsed low.
BE_b	Input	Byte enables for the data bus.
BRDY_b	Output	FPGA acknowledge signal. Signals the FPGA has sent valid data or has read data from the processor.
D	Input/Output	64-bit data bus
DP	Input/Output	8-bit parity bus covering the data bus
MIO_b	Input	Transaction is a Memory (1) or an I/O (0) access
Reset	Input	System reset signal
WR_b	Input	Read / Write flag. 1 = Write to FPGA, 0 = Read from FPGA
TDI, TCK, TMS	Input	Standard JTAG pins for JTAG configuration and readback.
TDO	Output	Standard JTAG pins for JTAG configuration and readback.

Notes:
1. All signals with "_b" after the signal name are active low.

Design Hierarchy

Figure 2 shows the design hierarchy broken into two parts: the user application and the Intel interface (designed to get the data to and from the processor). The user application is any number of Xilinx DSP Cores, memories, or any other function. Data is presented for the User application in the InFifo, and transmits what is in the OutFifo to the processor when requested.

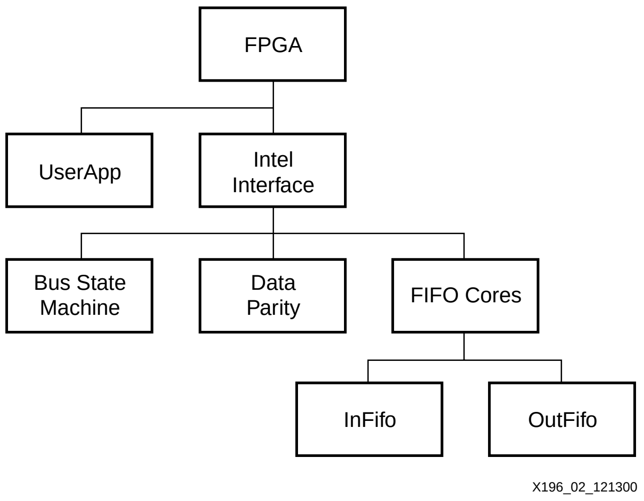

Figure 2: **Design Hierarchy**

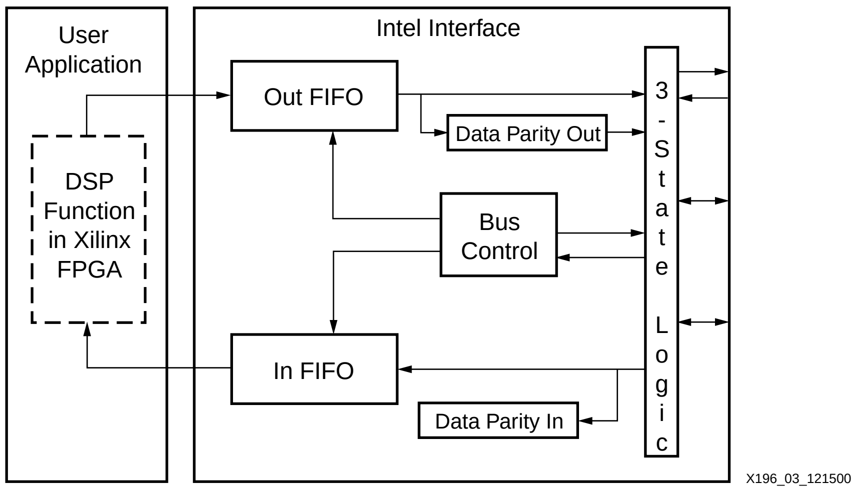

Figure 3: **Functional Block Diagram for FPGA Design**

Pentium Interface

The Pentium interface is the top-level module for a self-contained design that allows interfacing to the Pentium processor. The Pentium interface uses two FIFOs to control the flow of data between the user application and the Pentium processor. The user application loads data from the Input FIFO and sends data back to the processor through the Output FIFO. The design runs faster because of these FIFOs. The BE (Byte Enable) signals are captured and used to create valid data in the FIFO.

Data Parity Generation

The Data Parity Generation module calculates the data parity for data currently being transmitted to the processor. When a write transaction occurs, this module processes the current 64-bit data frame and generates an even parity bit check to be transmitted on DP[7:0].

The Data Parity from the processor is stored in the register DataParity_in. This design does a simple parity check and flags an error (ERROR) signal if the parity check fails.

FIFO Cores

The FIFO Cores module contains all of the CoreGen generated FIFO cores used in the design.

INFIFO

InFifo is a 64 bit wide by 15 bit deep Asynchronous FIFO generated by CoreGen.

OUTFIFO

OutFifo is a 64 bit wide and 15 bit deep Asynchronous FIFO generated by CoreGen.

Async_FIFO

The Aysnc_FIFO module (not shown in hierarchy) was produced by CoreGen and is used by both InFifo and OutFifo.

User_App

User_App is where the user places the design. It can be any function or Xilinx Core. In the reference design, the FIFOs are merely connected together, no example designs are included.

State Machine

Figure 4 shows the state machine for this design.

- IDLE: The design stays in this state until ADS_b goes low. This signals a new transaction.

WR_b tells whether this is a read or write. Then, the design checks the status of the appropriate FIFO to determine if it is ready for a transaction. If the read FIFO is not FULL, or the write FIFO is not EMPTY, it can begin the action. Otherwise, it holds BRDY_b high and inserts wait states on the bus (see Figure 5).

- READ: Processor transfers data to the FPGA. On the cycle after ADS_b assertion, the FPGA reads data off the bus, put there by the processor.
- WRITE1: FPGA writes data to the processor. This cycle allows one clock for the data to become valid from the FIFO.
- WRITE2: This cycle completes the FPGA write to the bus. Once valid data is present on the FIFO output, this data is transfer to the bus along with the parity bits calculated at that time.

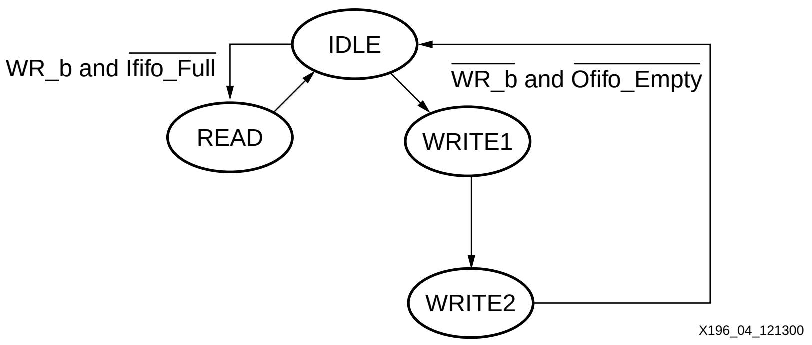
IDLE
WR_b and $\overline{\text{Ififo\_Full}}$
$\overline{\text{WR\_b}}$ and $\overline{\text{Ofifo\_Empty}}$
READ
WRITE1
WRITE2
X196_04_121300

Figure 4: **State Machine for Intel Interface**

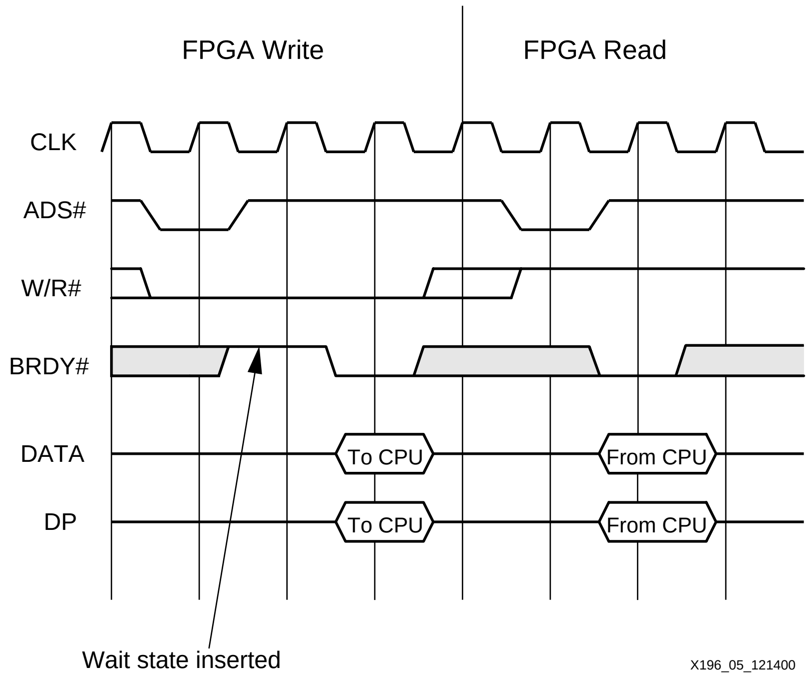

Figure 5: **Non-pipelined Read and Write with Wait States**

Additional Features of the Pentium Architecture

There are several features of the Intel Pentium architecture not shown in this design. For more information, see the Intel Pentium datasheet and Embedded Pentium processor family developers manual.

ADDRESS LINES

Utilize address lines to switch between multiple applications or Cores on the FPGA. Address lines can also enable the FPGA to act as common memory. Additionally, use the FPGA in conjunction with a Xilinx PCI Core to interface with the PCI bus.

PIPELINED READ AND WRITE

The Pentium processor has the ability to pipeline certain bus transactions when fast transfer rates are desired.

INTERRUPTS

Hardware Interrupts are used to facilitate transfer of data between the FPGA and the processor.

Differences Between the Pentium I, II, and III Architectures

This section covers the differences in signals and architecture between the Pentium II or III and the Pentium I processor. The Pentium II and III system bus architecture are identical.

The signaling on the bus is essentially the same. One difference is that the Pentium III System Bus runs at 100 MHz or 133 MHz, depending on device. The Pentium II and Pentium I run at 66 MHz. Several signal names are also different. Table 2 shows the corresponding Pentium and Pentium III signal names.

Table 2: **Pentium Signal Names**

Pentium I Signals	Pentium III Signals
CLK	BCLK
M/IO#, W/R#, D/C#	REQ[4:0]#, RS[2:0]#
BRDY#	TRDY#, DRDY#
DP[7:0]	DEP[7:0]#

All other Pentium signals are the same in the Pentium II/III architecture. For more information, see the P6 Family of Processors Hardware Developer's Manual.

Achieving Speed

The following techniques are used to achieve the maximum system clock speed when integrating the user design with the Intel interface.

Delay-Locked Loops (DLL)

A DLL is used to eliminate clock skew on a clock as it propagates through the design. This gives the user more of the clock period to utilize because there is less clock skew.

Another feature of the DLL that can be utilized is its ability to multiple, divide and phase shift clocks. This allows the designer to generate a wide variety of clocks for his design based off of the system clock.

For more detailed information, refer to **XAPP 132**, ***Using the Virtex Delay-Locked Loop***. For more information on Synplify, refer to **http://www.synplicity.com**.

Registering I/Os

Registered I/Os are used to minimize the delay in capturing data entering the chip, or hold data constant when exiting the chip. Registered I/Os are registers located in the Input Output Block (IOB). Using registered I/Os guarantees the shortest path between an I/O and a register.

To use the registered I/O with a device based on the Virtex architecture (Virtex, Spartan-II, or Virtex-E devices), all of the flip-flops in the same IOB must use the same clock and reset signals. No logic is permitted between the flip flop and the I/O pad, because there is no logic in the IOB and all logic must be implemented via a Configuration Logic Block (CLB) outside of the IOB. To pull the registers into the IOBs, use the map option:

```
-pr [ i | o | b ]
```

where `i` = input, `o` = output, and `b` = both.

For more information regarding map options, refer to *Development System Reference Guide* at **http://toolbox.xilinx.com/docsan/3_1i**.

SelectI/O™ Resource

The SelectI/O resource allows one to specify the use of different I/O standards with Virtex-based families. The Intel Pentium uses the 3.3V LVTTL standard and it is the default standard for the Spartan-II and Virtex families. Since in this design the 3.3V LVTTL standard is used, no additional logic or changes needed to be made.

For the Pentium II and III 100 MHz bus, a mixture of AGTL and CMOS voltage standards are used. The I/O standard on the FPGA is changed to accommodate the appropriate signaling levels. Also, when the FPGA interfaces with RAM or other components, these standards are used. For more information on how to infer buffers in the appropriate synthesis vendor documentation.

For more information on using the SelectI/O resource refer to **XAPP133**, ***Using the Virtex SelectI/O Resource***.

Area Constraints and Pin Constraints

To aid the placer in achieving optimal speed, it helps to place area and pin constraints on the design. It also helps to make it more modular in nature and easier to integrate into a system. These constraints can easily be added in using the Xilinx Floorplanner (area) and FPGA Express Constraints Editor (pin).

Area constraints are used to lock into a single column the block RAM used in the asynchronous FIFOs. Area constrain the rest of the design to a location on the edge of the die, as close to the block RAM as possible. This makes the paths as short as possible, reducing delay between elements.

Pin constraints are also used to lock the data lines to specific pin locations around the die. Ground bounce is minimized by separating out the bus. By placing the data, clock and control lines on the appropriate side of the device, you can utilize the FPGA dedicated routing. The CLBs (Configuration Logic Blocks) have horizontal direct connects to the adjacent CLB. These are used to flow the data lines through the device. The FPGA also has long lines that run vertically along the columns of the device. These long lines are well suited for control signals to the data paths.

Synthesis Settings

The design used a Foundation ISE project targeted at a Spartan-II XC2S50 TQ144 -6 device. The synthesis tool was FPGA Express Verilog. The design was simulated in ModelSim EE 5.4a with the Xilinx Libraries compiled. Any synthesis tool capable of supporting Verilog can be used with this design. A Virtex XCV50 TQ144 -6 can be substituted for the Spartan-II device.

Files Synthesized

Listed below are the design files.

- `Tb_main.tf` - Test fixture for Verilog, with sample vectors to test the design.
- `Interface.v` - Top level of the Intel processor interface design.
- `StateMach.v` - State machine for the design.
- `DataParity.v` - Data parity checks on both input and output.
- `Myfifo.v` - FIFO instantiation file.

Implementation Options

The two implementation options used to maximize performance and speed are:

`Map -pr b` (pack *both* input and output registers into the IOB)

`Par -ol 5` (overall Effort Level = 5)

Design Notes

1. The Verilog reference design XAPP196.zip is provided "as-is."
2. This design assumes only a processor, the FPGA, and memory is on the system bus. No arbitration is necessary in this configuration. In order to have the FPGA act as a Master, or have more than one processor on the system bus, the arbitration signals should be used. This is not desirable, as it will take more cycles and logic to arbitrate the bus. For embedded applications, only use one processor, the FPGA and memory. See Chapter 6 in the Embedded Pentium Processor Family Developers Manual for more information on arbitration.
3. In the state machine, the FPGA asserts wait cycles on the system bus if the InFifo is full or the OutFifo is empty. This may delay the processor, so another mechanism to check the status of the FIFO should be designed.
4. This design ignores all bus error signals generated by the processor.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/15/00	1.0	Initial Xilinx release.