

Fundamentals of FPGA Design



Xilinx is disclosing this Document and Intellectual Property (hereinafter "the Design") to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED "AS IS" WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems ("High-Risk Applications"). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

© 2006 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. PowerPC is a trademark of IBM, Inc. All other trademarks are the property of their respective owners.

Fundamentals of FPGA Design

Contents

Agenda	1a-1
Basic FPGA Architecture Review.....	1b-1
Xilinx Tool Flow	1c-1
Overview	1c-4
ISE	1c-11
Summary.....	1c-29
Architecture Wizard and PACE	1d-1
Architecture Wizard	1d-7
PAGE	1d-16
I/O Layout	1d-27
Summary.....	1d-30
Lab 1: Xilinx Tool Flow	
Introduction	1e-1
Lab.....	1f-1
Review.....	1g-1
Lab 2: Architecture Wizard and PACE	
Introduction	1h-1
Lab.....	1i-1
Review.....	1j-1
Reading Reports.....	2a-1
Introduction	2a-7
Area Goals	2a-9
Performance Goals	2a-21
Summary.....	2a-36
Global Timing Constraints.....	2b-1

Introduction	2b-7
Global Constraints	2b-18
The Constraints Editor	2b-31
Summary.....	2b-36
Lab 3: Global Timing Constraints	
 Introduction	2c-1
 Lab.....	2d-1
 Review.....	2e-1
Synchronous Design Techniques.....	3a-1
 Hierarchical Design.....	3a-7
 Synchronous Design for Xilinx FPGAs	3a-14
 Summary.....	3a-33
Implementation Options	3b-1
 Basic Software Options	3b-7
 Accessing Advanced Software Options	3b-19
 Summary.....	3b-25
Lab 4: Implementation Options	
 Introduction	3c-1
 Lab.....	3d-1
 Review.....	3e-1
Detailed Design Description	3f-1

Appendices

Troubleshooting Tips.....	A-1
FPGA Design Checklist.....	B-1
Flow Diagram	C-1
Glossary	D-1
Additional Xilinx Courses & Services.....	E-1

Education Services Quick Reference

Fundamentals of FPGA Design

From the Xilinx Education Services *Fundamentals of FPGA Design* course.
For more information on Xilinx courses, please visit www.xilinx.com/education

Commonly Used Implementation Options	
Option	Description
Translate	Macro Search Path (use if you have netlists located in other directories)
MAP	Pack I/O Registers/Latches into IOBs <ul style="list-style-type: none">- Inputs- Outputs- Inputs & Outputs Off (driven by synthesized netlist) Trim Unconnected Signals<ul style="list-style-type: none">- Checked (default) – will trim sourceless and loadless logic
PAR	Overall Effort Level (to improve both placer and router effort)

Implementation Reports	
Report	Description
Map Report	Design summary (how many device resources are used).
PAR Report	Lists any unrouted nets.
Post-MAP Static Timing Report	Contains actual block delays and NO net delays.
Post-PAR Static Timing Report	Contains actual block delays and ACTUAL net delays.
Pad Report	Lists device pinout.

Global Timing Constraints	
Constraint	Description
Period	Covers purely synchronous paths (i.e., flops to flops).
Pad-to-Pad	Covers <i>purely combinatorial paths</i> between input and output paths.
Offset In/Out	Specifies the internal delay—the time required for data to go from the input pin to the input register and time required for data to go from an output register to an output pin. Accounts for clock delays and skew.
Diagram	<p>The diagram illustrates a timing constraint between two flop stages. It shows a signal path starting with a PADA block, followed by a first flop (FLOP) with inputs D and Q, and a second flop (FLOP) with inputs D and G. The output of the second flop is labeled OUT1. A CLK1 signal is shown driving the first flop. The timing constraints are labeled as follows: OFFSET IN between the PADA and the first flop; PERIOD between the first and second flop; and OFFSET OUT between the second flop and OUT1. The diagram also includes a BUFG block connected to the CLK1 signal.</p>

Education Services Quick Reference

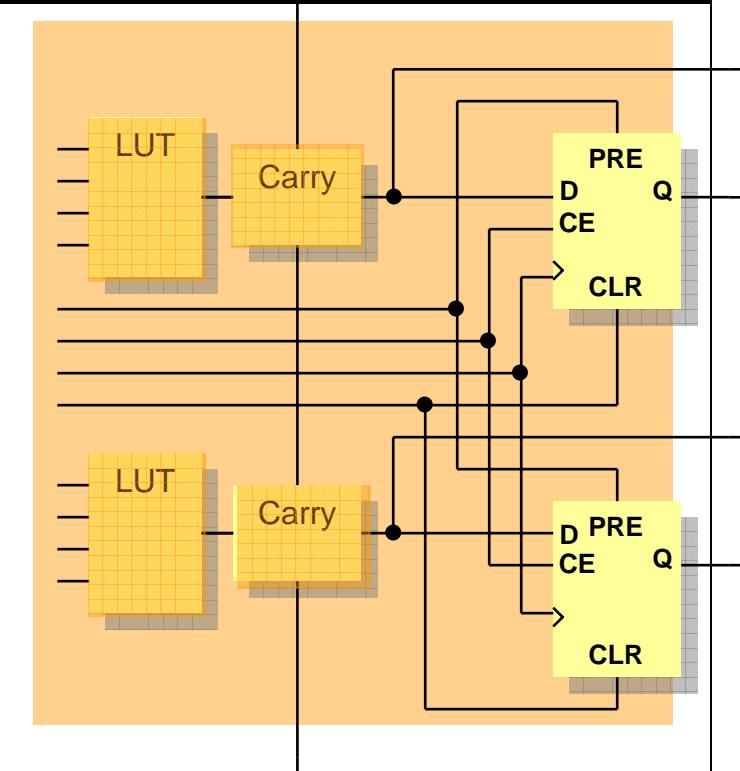
Fundamentals of FPGA Design

From the Xilinx Education Services *Fundamentals of FPGA Design* course.
For more information on Xilinx courses, please visit www.xilinx.com/education

Virtex™-4 Architecture

All Xilinx FPGAs have the same basic building blocks:

- CLBS, which are made of slices
 - Each slice contains the following:
 - 2 look-up tables (LUTS)
 - 2 sequential elements (flops or latches)
 - Carry logic
 - Multiplexers to cascade LUTs
- Global routing resources



Additional Virtex-4 Resources

- Block RAM
- DSP48 arithmetic blocks
- DDR registers in I/O blocks
- Digital Controlled Impedance (DCI)
- Digital Clock Managers (DCM)
 - Delay-locked Loop (DLL)
 - Digital Frequency Synthesizer (DFS)
 - Digital Phase Shifter (DPS)
- Phase Matched Clock Dividers (PMCD)
- I/O SERDES in each IO Tile
- Global Clock multiplexes to switch between clocks

Additional Virtex-4 FX Resources

- RocketIO™ Multi-Gigabit Transceiver (MGT)
- IBM PowerPC™ RISC processor blocks
- Ethernet Media Access Controller (EMAC)

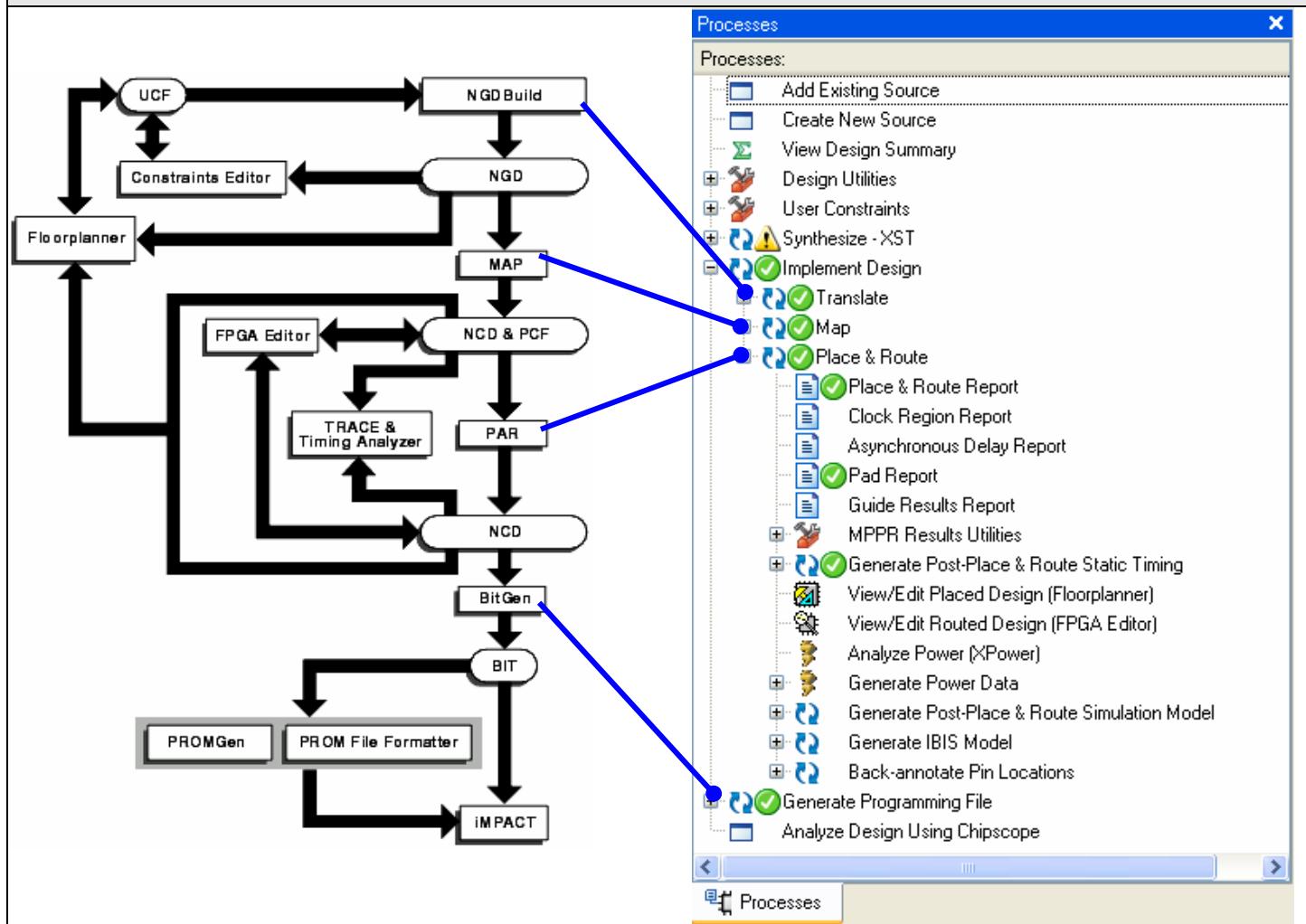
Education Services Quick Reference

Fundamentals of FPGA Design

From the Xilinx Education Services *Fundamentals of FPGA Design* course.
For more information on Xilinx courses, please visit www.xilinx.com/education

Implementation Design Flow Diagram

*More information and details on each process and available options are in the *Development System Reference Guide*



Fundamentals of FPGA Design

Course Agenda

©2006 Xilinx, Inc. All Rights Reserved

NOTES

An Ineffective Design Approach Can Give You Flat Results

For best results...

- Design synchronously
- Optimize the design
- Use HDL coding styles
- Use the Xilinx ISE™ tools
- Simulate efficiently
- Use synthesis options



NOTES

An ineffective design approach can give you flat results. Without the right knowledge and tools, your designs may not reach peak performance, and you may experience longer design times and higher costs.

An Effective Approach Will Obtain Optimum Results

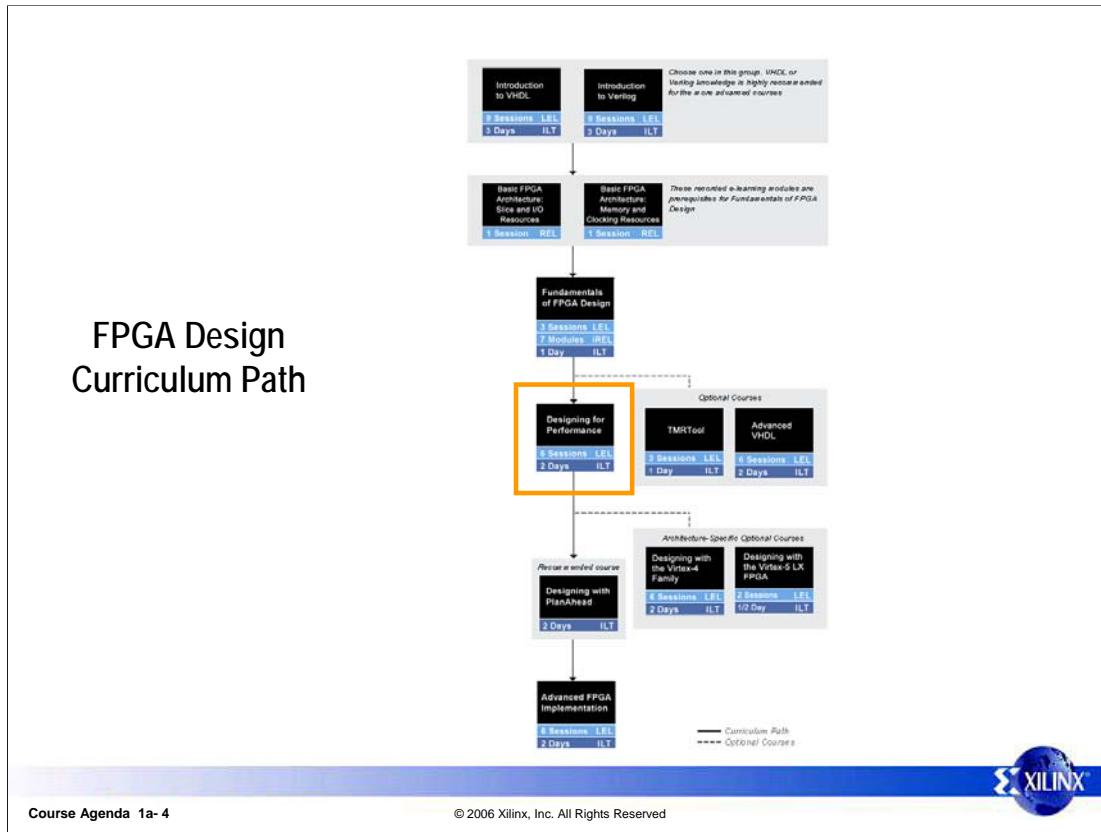
Follow the practices described in this course for...

- The best design speed
- Good device utilization
- Fast simulation and debugging capabilities
- Minimal Place & Route iterations
- Very portable HDL code
- Understandable synthesis and implementation results
- Good results, quickly
- Overall, the best results possible



NOTES

An effective approach will obtain optimum results.

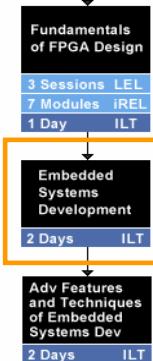


NOTES

For information on how to use these resources in your design (such as whether to instantiate or to infer these resources), refer to the “Synthesis Techniques” module in the Designing for Performance course.

Embedded Systems Design Curriculum Path

*Basic understanding of C Programming required



Type of training
CLASSROOM
WEB-BASED

Delivery methods
ILT - Instructor-led training
LEL - Live e-learning
REL - Recorded e-learning



Course Agenda 1a- 5

© 2006 Xilinx, Inc. All Rights Reserved

NOTES

Use of the PowerPC™ processor is covered in the *Embedded Systems Development* course.

Lab Instructions

- Below each general instruction for a given procedure, you will find accompanying step-by-step directions and illustrated figures that provide more detail for performing the general instruction.
 - General Instruction
 - Step-by-step detailed directions on how to perform the general instruction
- If you feel confident about a specific instruction, feel free to skip the step-by-step directions and move on to the next general instruction in the procedure.



NOTES

Perform Labs in a Virtual Environment with Toolwire

- Secure servers in Windows or UNIX
- Xilinx lab files
- Lab instructions in your workbook
- Other course reference material



Course Agenda 1a- 7

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

With Toolwire, you can perform Xilinx learning labs in a virtual environment and receive online hands-on experience with the Xilinx software tools. Toolwire provides secure servers running all of the necessary software in either a Windows or UNIX environment. Xilinx lab files are provided in your personal account directory, and lab instructions are included with the lecture slides. Other course reference materials, such as user guides and data sheets, may be included as well.

Toolwire Performance

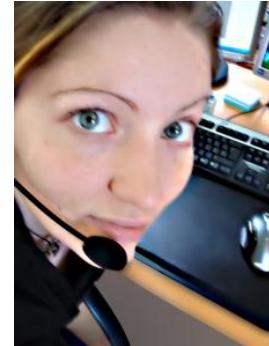
- Software environment is completely set up and tested!
- Easier to fix and maintain!
- Note that Local facility LAN bandwidth and ISP may affect performance
 - May be slower than running software on a local machine, especially when using graphic-intensive applications



NOTES

When to Get Help with Toolwire

- . If application seems to freeze
- . If Toolwire loses connection
 - Normally, exiting Toolwire and logging back in retains all work



NOTES

Course Objectives

After completing this course, you will be able to:

- Use the Xilinx Project Navigator to implement and simulate an FPGA design
- Read reports and determine whether your design goals were met
- Use the Architecture Wizard to create DCM instantiations
- Use the PACE tool to assign pin locations
- Use the Xilinx Constraints Editor to enter global timing constraints
- Locate and modify the implementation options



NOTES

Agenda

Session 1

- Basic FPGA Architecture Review
- Xilinx Tool Flow
- Architecture Wizard and PACE
- Lab 1: Xilinx Tool Flow Lab
- Lab 2: Architecture Wizard and PACE Lab

Session 2

- Reading Reports
- Global Timing Constraints
- Lab 3: Global Timing Constraints Lab

Session 3

- Synchronous Design Techniques
- Implementation Options
- Lab 4: Implementation Options Lab



NOTES

The course workbook includes appendix items, such as a glossary, a flow diagram, troubleshooting tips, information on where to learn more, and a listing of other Xilinx courses.

Prerequisites

- Basic HDL knowledge (VHDL or Verilog)
- Digital design knowledge and experience
- Completion of the Xilinx Basic FPGA Architecture RELs
 - Slice and I/O Resources
 - Memory and Clocking Resources



NOTES

Basic VHDL Knowledge:

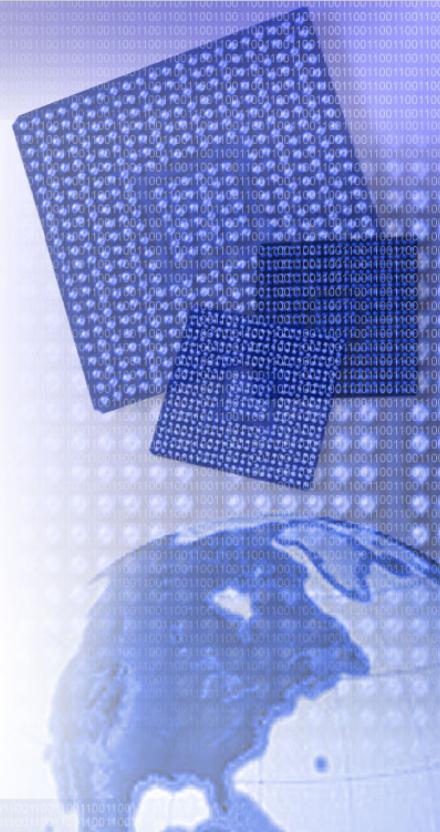
Entity/architecture declarations
IF/THEN statements
CASE statements
Clocked processes
Component instantiation

Basic Verilog Knowledge:

Module declaration
IF/THEN statements
CASE statements
Clocked ALWAYS blocks
Component instantiation



Review of Basic FPGA Architecture



NOTES

This Basic FPGA Architecture module introduces the various resources available in Xilinx devices and specifically discusses the resources in the Virtex™-4 device.

Objectives

After completing this module, you will be able to:

- Identify the basic and advanced architectural resources of the Virtex™-4 FPGA

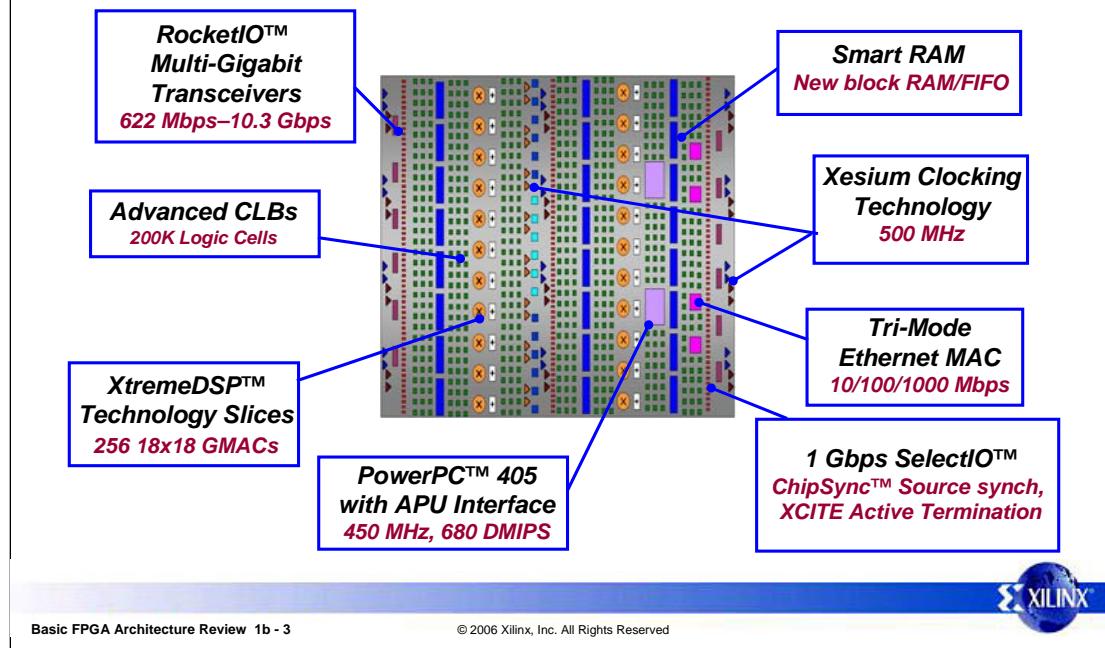


NOTES

Note, this module addresses the primary resources available in the Xilinx Virtex-4 FPGA family. It is intended as a review of the Basic FPGA Architecture recorded e-Learnings (RELs) that are required for this course. If you have not had an opportunity to review them, please do so at your earliest convenience.

For more information on how to use these resources in your design (such as whether to instantiate or to infer these resources), refer to the *HDL Coding Style* module in the *Designing for Performance* course.

Virtex-4 Architecture Has the Most Advanced Feature Set



NOTES

All Xilinx FPGAs contain the same basic resources. Slices, which are grouped into Configurable Logic Blocks, or CLBs, contain combinatorial logic and register resources. Input/Output Blocks, or IOBs interface between the FPGA and the outside world. Programmable interconnect is how the slices and IOBs communicate with each other. Other resources include memory, DSP slices, clock management components, and IP cores.

Choose the Platform that Best Fits the Application

	LX	FX	SX
Resource			
Logic	14K–200K LCs	12K–140K LCs	23K–55K LCs
Memory	0.9–6 Mb	0.6–10 Mb	2.3–5.7 Mb
DCMs	4–12	4–20	4–8
DSP Slices	32–96	32–192	128–512
SelectIO	240–960	240–896	320–640
RocketIO	N/A	0–24 Channels	N/A
PowerPC	N/A	1 or 2 Cores	N/A
Ethernet MAC	N/A	2 or 4 Cores	N/A

 XILINX

Basic FPGA Architecture Review 1b - 4 © 2006 Xilinx, Inc. All Rights Reserved

NOTES

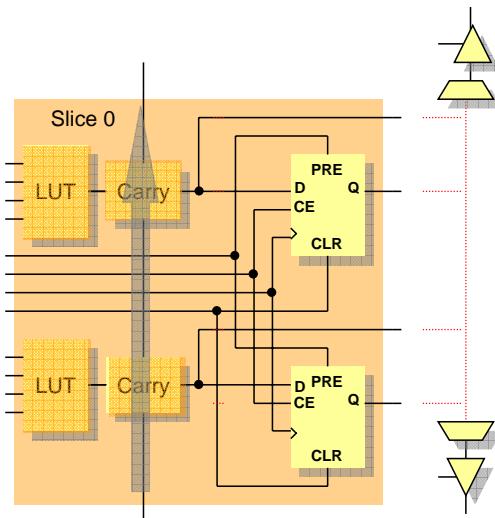
This table shows the three distinct Virtex™-4 platforms. Each platform contains a different mixture of resources, which gives you the most flexibility to select the right device for your application.

The LX family is focused on logic (slices), with a modest amount of memory and DSP Slices.

The FX family contains the RocketIOTM, PowerPCTM, and Ethernet MAC cores.

The SX family is focused on signal processing, and therefore contains more DSP slices than similar-sized LX and FX devices.

Viewing a Simplified Slice Structure



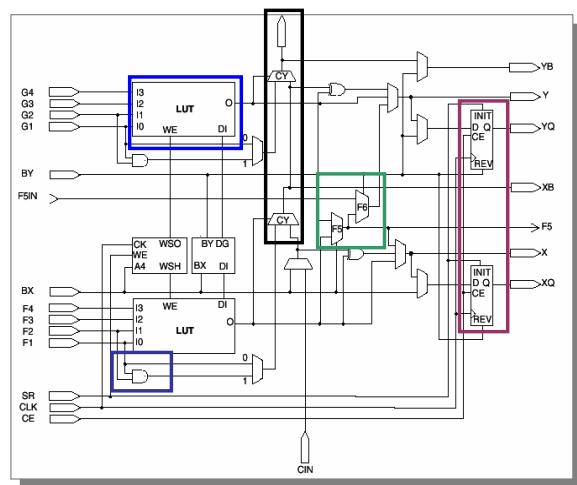
NOTES

Here we see a simplified slice structure. The major parts of a slice include two Look-Up Tables (or LUTs), two sequential elements, and carry logic. Each slice has four outputs. There are two registered outputs and two non-registered outputs. Carry logic runs vertically, in the upward direction only. There are two independent carry chains per CLB. The sequential elements can be programmed to be either registers or latches.

Knowledge Check

Name each slice resource and explain why it is beneficial

Raise your hand if you think you know the answer



Basic FPGA Architecture Review 1b - 6

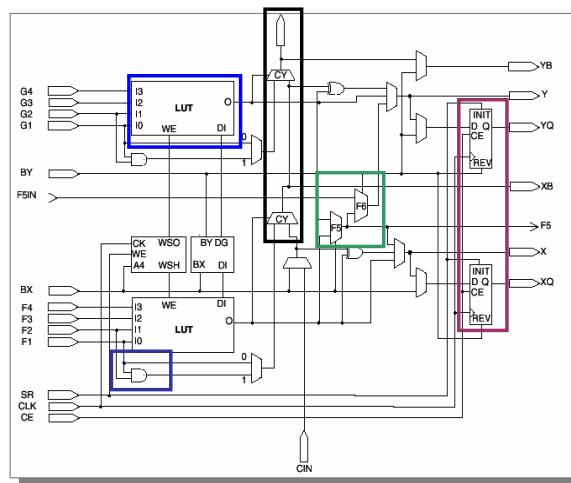
© 2006 Xilinx, Inc. All Rights Reserved

 XILINX

NOTES

Answers

- LUTs
- Carry logic
- Registers

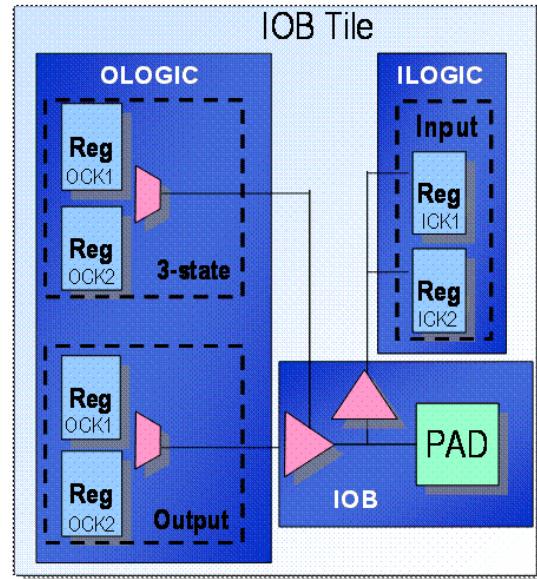


NOTES

A LUT performs any combinatorial function up to 16 inputs, implements a 16-bit shift register, and implements a 16-bit RAM. Carry Logic implements arithmetic functions (adders, subtractors, comparators, counters, etc.), registers can implement with S/R being synchronous or asynchronous (can also be a latch), Mult-And gate does a two-bit multiplication. Dedicated multiplexers implement large multiplexers quickly.

Viewing the IOB Element

- Input path
 - Two DDR registers
- Output path
 - Two DDR registers
 - Two 3-state enable DDR registers
- Separate clocks and clock enables for I and O
- Set and reset signals are shared



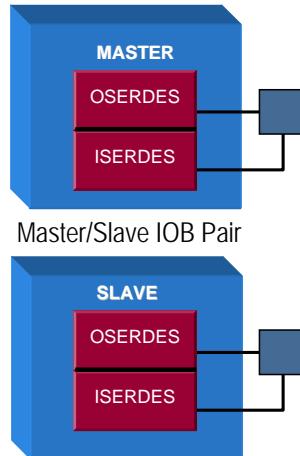
NOTES

You are not required to use the registers in the IOB in Double Data Rate mode.

Clocking the DDR registers: You can also use any pair of DCM outputs that are 180 degrees out of phase (CLK90 / CLK270, CLK2X / CLK2X180, CLKFX / CLKFX180).

The I/O SERDES Combination: Most Useful Development

1-10 bit Parallel/Serial and Serial/Parallel Conversion



Master/Slave IOB Pair



NOTES

The OSERDES/ISERDES combination is probably one of the most useful and radical developments in the Virtex™-4 FPGA. Many issues that you would traditionally address by using the FPGA fabric are handled within these blocks, such as training pattern recognition, clock-to-data alignment, and crossing of clock domains between high-speed serial I/O and slower parallel domains.

Coupled with regional and I/O clocks, which we will discuss later in this module, this feature is very well suited for source-synchronous applications.

The I/O SERDES capability includes input and output SERDES. There is an OSERDES parallel-to-serial converter for both OQ and TQ. It is arranged as master and slave IOB pair to allow for differential inputs or larger single-ended bit widths.

Versatile SelectIO Supports 32 Standards in Virtex-4 Devices

- LVCMOS (3.3-V, 2.5-V, 1.8-V, 1.5-V)
- LVPECL
- PCI, PCI-X
- GTL, GTL+
- HSTL (1.8 V, 1.5 V; Classes I, II, III, IV)
 - Supports differential HSTL
- SSTL (2.5 V, 1.8 V; Classes I, II)
 - Supports differential SSTL
- LVDS, Bus LVDS, Extended LVDS
- HyperTransport™ (LDT)

Easier
and
More
Flexible
I/O Design!



NOTES

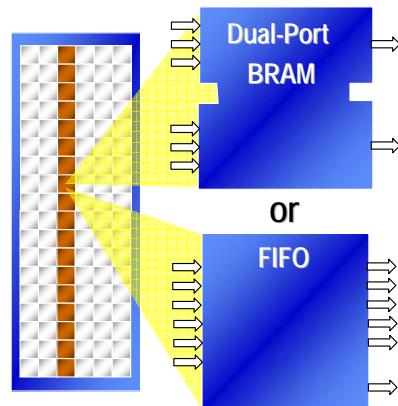
The versatile SelectIO™ standard supports 32 standards available in Virtex™-4 devices including those you see listed here. The SelectIO standard allows direct connections to external signals of varied voltages and thresholds. This optimizes the speed/noise tradeoff and saves having to place interface components onto your board. The differential signaling standards include LVDS, LVPECL. The single-ended I/O standards include LVTTL, LVCMOS, PCI, and HSTL.

There are three ways to use I/O standards in your design:

- 1) Use the PACE tool to assign standards
- 2) Use the IO_STANDARD attribute in your source code
- 3) Instantiate the I/O buffer in your design

Virtex-4 BRAM/FIFO Technology is Redefining On-Chip Memory

- New features
 - Pipeline option for 500 MHz operation
 - Integrated logic for fast, efficient FIFOs
 - Integrated cascade logic
 - . Create 32k x 1 from two 16k x 1 BRAMs
 - Byte-Write enable
 - . Enhances PowerPC™ memory interfacing
- Compatible with Virtex™-II Pro block RAM
- Independent read and write port widths



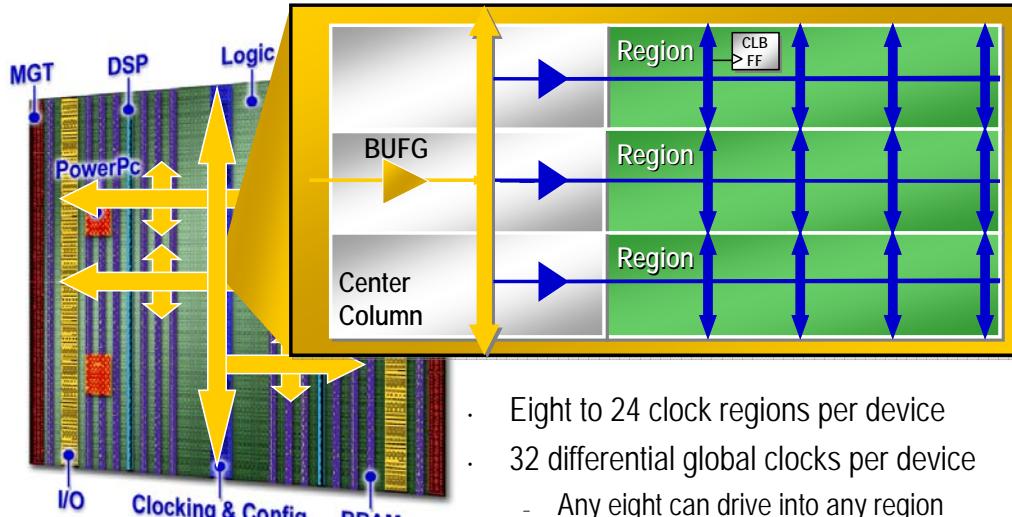
NOTES

Virtex™-4 BRAM/FIFO technology is redefining on-chip memory. The technology's new features include a pipeline option for 500 MHz operation, integrated logic for fast, efficient FIFOs, and integrated cascade logic, which lets you create 32k x 1 from two 16k x 1 BRAMs. Byte-Write enable is another new feature and enhances PowerPC memory interfacing. The Virtex™-4 device is compatible with the Virtex-II Pro block RAM and it has independent read and write port widths.

Block SelectRAM™ memory is available in 18 kbit blocks. This includes 16 kbytes of data, plus one parity bit for each data byte. This dual-port memory has separate clocks for each port and different read and write aspect ratios for each port. An optional output register is included for increased performance. The output latches include a synchronous reset input. Cascade pins are included to link two blocks of RAM with no speed penalty.

Regarding aspect ratios: In Virtex-4 devices, the data in the block RAM can be accessed using different data widths for each port and read/write operation. This means that you can write to Port A in bytes (plus a parity bit), read from Port A serially (with no access to the parity bits), write to Port B in 16-bit words (plus two parity bits), and read from Port B in 32-bit words (plus four parity bits).

Virtex-4 Devices Offer Abundant and Flexible Global Clocking



- Eight to 24 clock regions per device
- 32 differential global clocks per device
 - Any eight can drive into any region

Basic FPGA Architecture Review 1b - 12

© 2006 Xilinx, Inc. All Rights Reserved



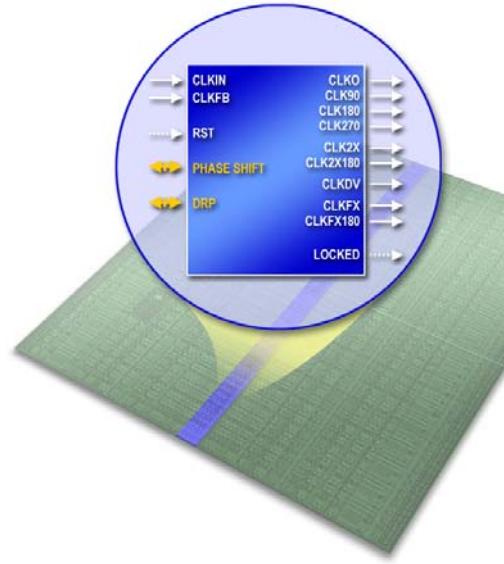
NOTES

The Virtex™-4 devices offer abundant and flexible global clocking. There are thirty-two dedicated global clock multiplexers called BUFGCTRLs. They are located in the center I/O column. They can be driven by a clock input pad, a DCM, or local routing. The global buffer output is distributed differentially to support high clock speeds. Global clock multiplexers provide traditional clock buffer function (BUFG), global clock enable capability (BUFGCE), and glitch-free switching between clock signals (BUFGMUX).

Up to eight clock nets can be used in each clock region of the device. Any eight can drive into ANY region. A clock region is half of the die in width and 16 CLBs tall. 16 CLBs is equivalent to 32 Slices, or 32 IOB tiles, or four block RAMs, or eight DSP Slices.

Each Device Holds Up to 20 DCMs Providing DLL, DFS, & DPS Functions

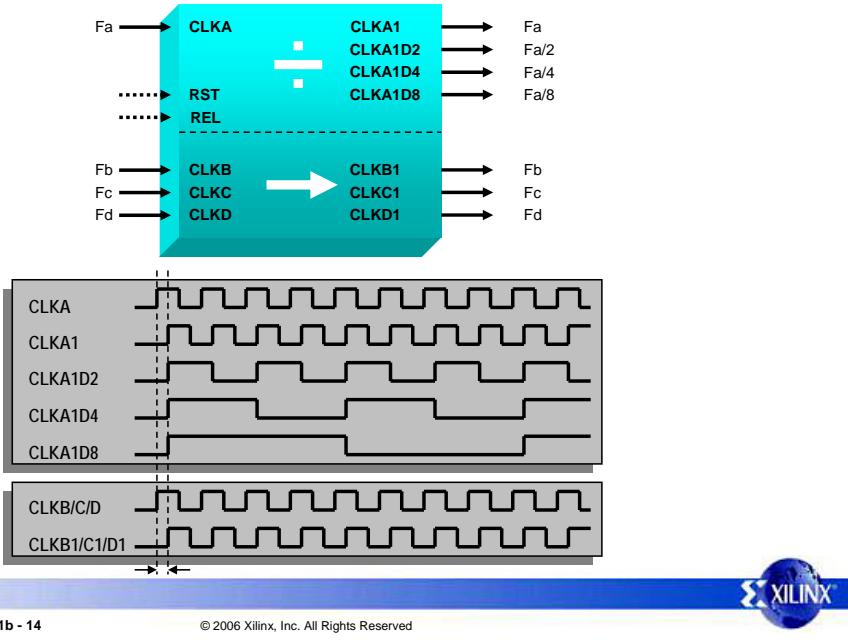
- Zero delay buffer
- Frequency synthesis
 - Simultaneous multiply and divide
- 0, 90,180, and 270-degree shifted out clocks
- Phase Shift control
 - Specified as clock period fraction
 - Direct control of delay line taps
 - Affects all DCM outputs
- Dynamic Reconfiguration Port (DRP)
 - Adjust Multiply/Divide and Phase Shift values without reconfiguring device
 - . DCM reset required for M/D change



NOTES

There are up to twenty DCMs per device. They are located in the center column and are driven by clock input pads. DCMs provide Delay-Locked Loop (DLL), Digital Frequency Synthesis (DFS), and Digital Phase Shifting (DPS) functions. DCM outputs can drive global clock buffers, PMCDs, other DCMs, or local routing.

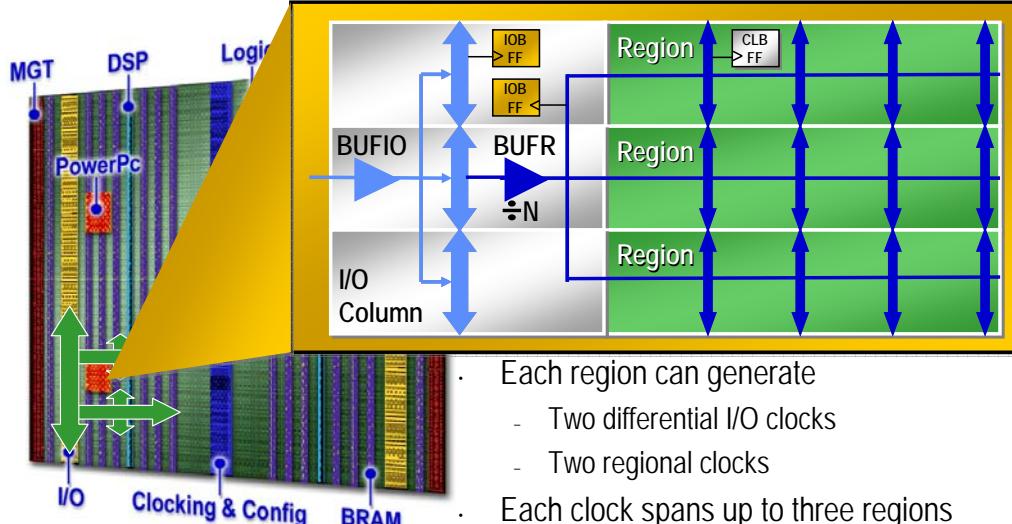
The Phase-Matched Clock Divider (PMCD) is Located with the DCMs



NOTES

The phase-matched clock divider, or PMCD, is located in the center column with the DCMs. It can be driven by a clock input pad, DCM output, or the CLKA1D8 output of another PMCD. It creates edge-aligned divided versions of the CLKA input. Other clock inputs are not divided, but are delayed by the same amount to maintain phase relationships.

Local Clock Routing Resources Include Two I/O Clock Buffers



- Each region can generate
 - Two differential I/O clocks
 - Two regional clocks
- Each clock spans up to three regions
 - One above and one below

Basic FPGA Architecture Review 1b - 15

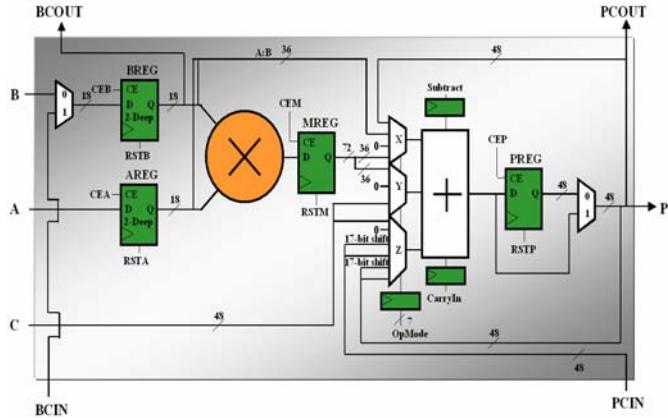
© 2006 Xilinx, Inc. All Rights Reserved



NOTES

Local clock routing resources include two I/O clock buffers in each clock region (BUFIO). These clock signals are distributed differentially to support high clock speeds. They only drive IOB and BUFR components. They are driven by “clock-capable” I/O pins. There are two differential pairs of clock-capable I/O in each clock region. These pins have lower capacitance to support high-speed clock signals. Clock-capable I/O pins can be used as regular I/O pins, except that they do not support LVDS output signals. There are also two regional clock buffers in each clock region (BUFR). The BUFR resource can perform simple clock division to support SERDES applications. They can be driven by a regular input pin, BUFIO or local routing. BUFIO and BUFR components can drive loads in their own clock region, plus the adjacent regions above and below.

DSP48 Slice Has 18x18 2s Complement Multiplier and 48-Bit Accumulator



NOTES

The DSP48 slice has an 18x18 2s complement multiplier and dynamic user-controlled operating modes (OPMODEs). It also has a 3-input, 48-bit adder/subtractor. Cascade pins are included to support complex functions with no speed penalty. Symmetric rounding support is also included. There are optional pipeline registers at several points within the DSP48 slice to maximize performance.

DSP designers who traditionally use the FPGA fabric for arithmetic applications will find that much of their job is done for them internally to this block. All they need to do is configure the block by using OPMODE inputs, which control the flow of data in the block. OPMODE is dynamic, primarily to enable resource sharing.

Other Advanced Features

- Other Advanced Features
 - PowerPC™ 405 processor with enhanced features
 - High Speed Serial I/O
 - Tri-Mode Ethernet MAC
- Each is fully integrated into the FX subfamily



NOTES

For more information about the PowerPC and Xilinx Embedded solutions, you may wish to review the contents of the Embedded and Advanced Embedded courses. Likewise, refer to the course on the Serial Gigabit Transceivers (MGTs) for information about targeting those dedicated resources.

Summary

- . Virtex™-4 memory resources include the following:
 - Distributed SelectRAM™ resources and distributed SelectROM (uses CLB LUTs)
 - 18-kb block SelectRAM resources
 - Dedicated FIFO logic
- . Virtex™-4 clocking resources include the following:
 - Up to 32 global clocks per device
 - Delay-Locked Loop (DLL)
 - Digital Frequency Synthesizer (DFS)
 - Digital Phase Shifter (DPS)
 - Phase Matched Clocked Dividers (PMCD)
 - Local Clock Routing



NOTES

Summary

- . The Virtex™-4 devices provide the following dedicated hardware resources:
 - DSP Slices
 - Serial Gigabit Transceivers (MGT)
 - PowerPC™ microprocessor with an enhanced APU
 - Tri-Mode Ethernet MAC
- . Spartan™-3 FPGAs are designed for low cost and have a reduced subset of resources



NOTES

Where Can I Learn More?

- www.xilinx.com/mysupport.htm
 - Check out the Virtex™-4, Spartan™-3, and Spartan™-3E user guides and data sheets
- www.xilinx.com/education-home.htm
 - Learn about the Designing for Performance course
 - Review the Basic FPGA Architecture RELs
 - Learn about RELs and other courses that may help you



NOTES



Xilinx Tool Flow

©2006 Xilinx, Inc. All Rights Reserved

NOTES

This Xilinx Tool Flow module will show you the steps of the Xilinx design process and how to implement and simulate an FPGA design by using default software options.

Objectives

After completing this module, you will be able to:

- List the steps of the Xilinx design process
- Implement and simulate an FPGA design using the default software options



NOTES

Note, this module addresses the various resources available on Xilinx devices, and it specifically discusses the resources on the Virtex™-4 device. For information on how to use these resources in your design (such as whether to instantiate or to infer these resources), refer to the “HDL Coding Style” module in the Designing for Performance course.

Outline

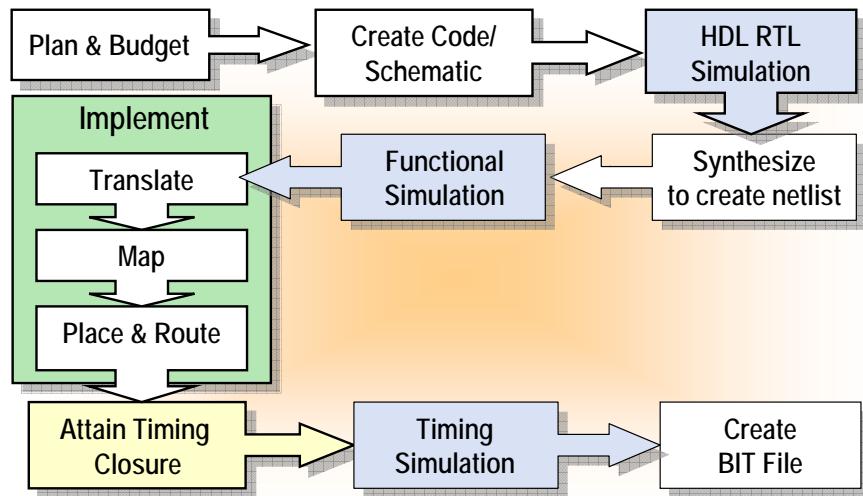


- . Overview
- . ISE
- . Summary



NOTES

Implementing a Design into a Xilinx Device



Xilinx Tool Flow 1c - 4

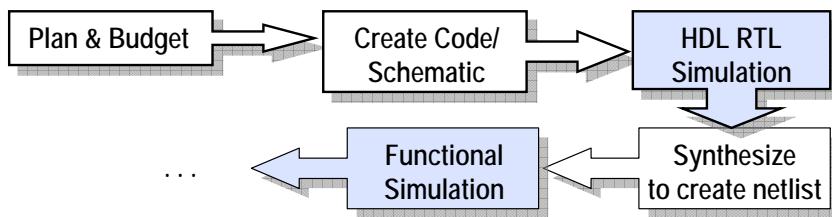
© 2006 Xilinx, Inc. All Rights Reserved



NOTES

Here you see the Xilinx design flow depicting the major stages of implementing a design into a Xilinx device. The implementation stage consists of three steps, which will be discussed later in this module. Although simulation points can happen in other parts of the design cycle, the three simulation points in the above diagram are the Xilinx-recommended simulation points. You will hear more details on Timing Closure in just a bit.

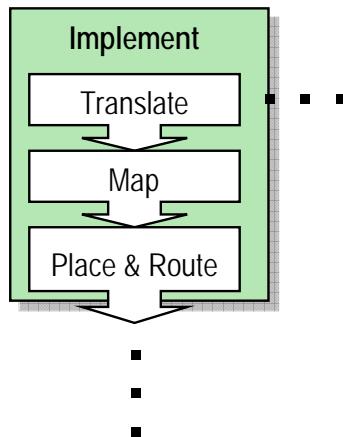
Two Design Entry Methods Include HDL or Schematic



NOTES

Let's take a closer look at the first five steps of the Xilinx design flow. First you need to plan and budget your project. Next, note that there are two design-entry methods: HDL or schematic. The Architecture Wizard, CORE Generator™ system, and StateCAD tools are available to assist in design entry. Whichever method you use, you will need a tool to generate an EDIF or NGC netlist to bring into the Xilinx implementation tools. Popular synthesis tools include: Synplify, Precision, FPGA Compiler II, and XST. Then, simulate the design to ensure that it works as expected!

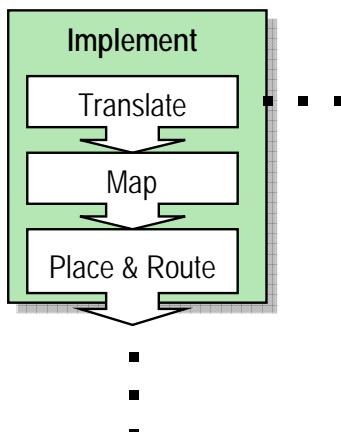
After Generating a Netlist, Implement the Design



NOTES

Once you generate a netlist, you can implement the design. There are several outputs of implementation. They include: Reports, Timing Simulation Netlists, Floorplan files, FPGA Editor files, and more!

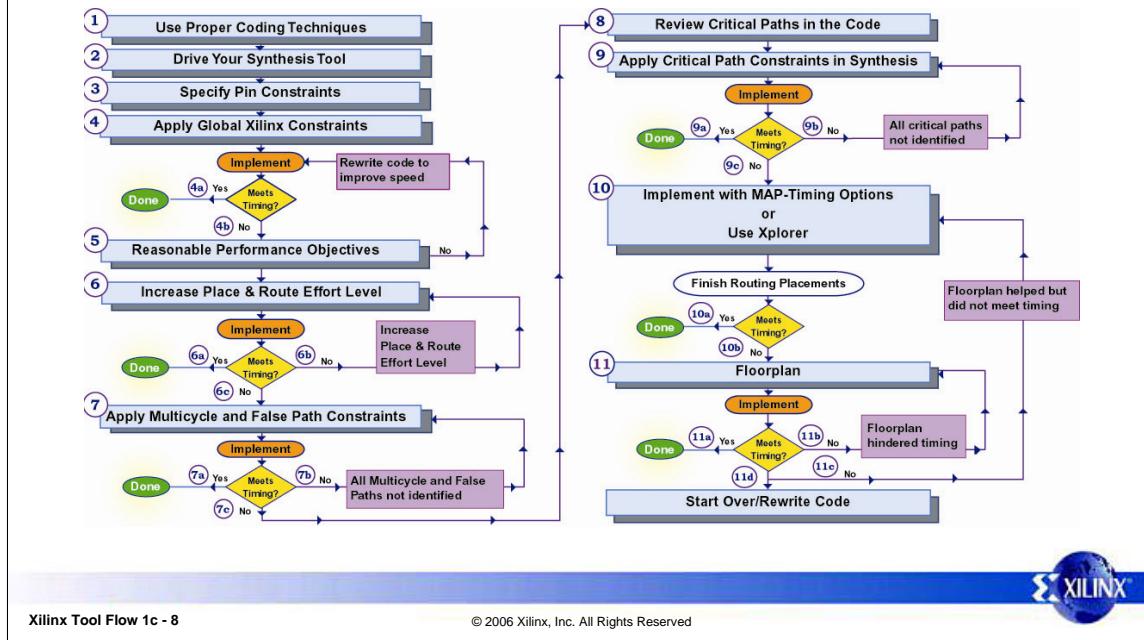
Implementation Includes Many Phases



NOTES

When you take a closer look at the Implementation phase, you realize it is more than just *Place & Route*. Implementation includes many phases. The **Translate** phase is where you merge multiple design files into a single netlist. The **Map** phase is where you group logical symbols from the netlist (gates) into physical components (slices and IOBs). The **Place & Route** phase is where you place components onto the chip, connect the components, and extract timing data into reports. Each phase generates files that allow you to use other Xilinx tools, such as Floorplanner, FPGA Editor, and XPower software.

Taking a Look at the Timing Closure Flow



NOTES

After implementation, you will want to attain timing closure. Here you see the steps involved in the timing closure flow. You will cover steps 3, 4, and 6 in this module. To learn more about steps 1, 2, 5, 7, 8, 9, and 10 enroll in the *Designing for Performance* course. To learn more about the final step, step 11, see the “Floorplanner: Effective Layout” module in the *Advanced FPGA Implementation* course.

Where to find details on each numbered step:

1. “Synthesis Techniques” module in the *Designing for Performance* course.
2. “Synthesis Techniques” module in the *Designing for Performance* course.
3. “Architecture Wizard and PACE” module in this course.
4. “Global Timing Constraints” module in this course.
5. “Reading Reports” module in this course. More details in the “Achieving Timing Closure” module in the *Designing for Performance* course.
6. “Implementation Options” module in this course.
7. “Timing Groups and OFFSET Constraints” and “Path-Specific Timing Constraints” modules in the *Designing for Performance* course.
8. “Synthesis Techniques” module in the *Designing for Performance* course. Third-party synthesis vendors also offer training.
9. “Synthesis Techniques” module in the *Designing for Performance* course. Third-party synthesis vendors also offer training.
10. “Advanced Implementation Options” module in the *Designing for Performance* course.
11. “Floorplanner: Effective Layout” module in the *Advanced FPGA Implementation* course.

After Implementation, Create a File Called a Bitstream



NOTES

Once a design is implemented, you must create a file that the FPGA can understand. This file is called a bitstream: a BIT file (.bit extension). The BIT file can be downloaded directly into the FPGA, or the BIT file can be converted into a PROM file, which stores the programming information.

Knowledge Check

Can you describe the three primary implementation phases?

*Raise your hand if you
think you know the
answer*



NOTES

Answers

- The three primary implementation phases:
 - **Translate:** Merges multiple design files into a single netlist
 - **Map:** Groups logical symbols from the netlist (gates) into physical components (slices and IOBs)
 - **Place & Route:** Places components onto the chip, connect the components, and extracts timing data into reports



NOTES

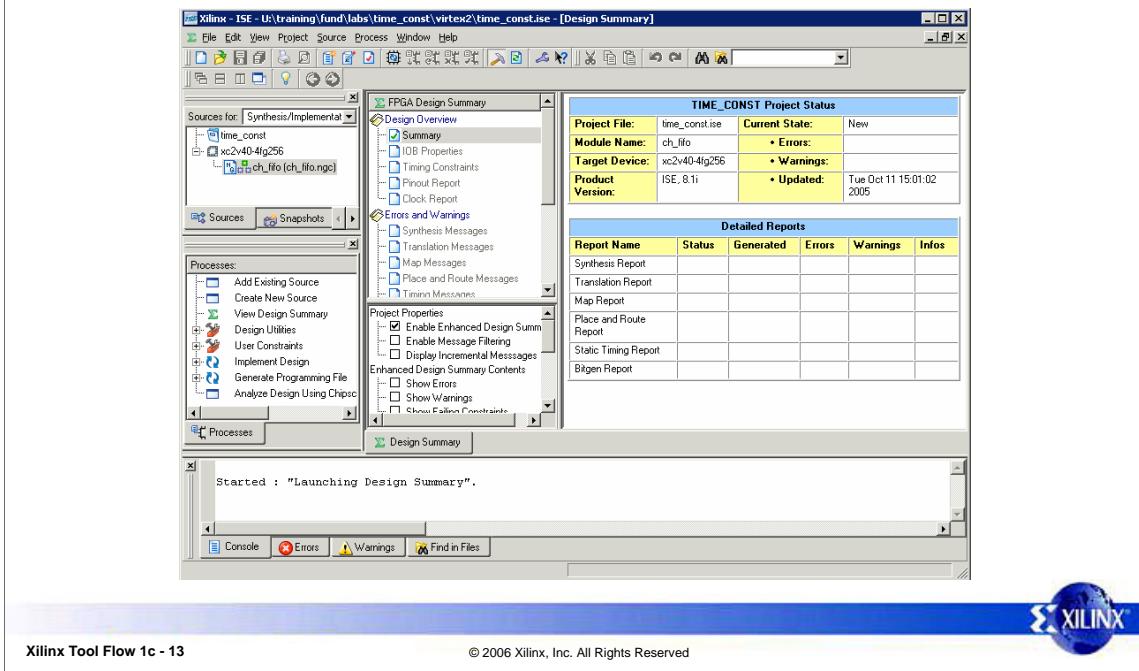
Outline

- Overview
- **ISE**
- Summary



NOTES

Project Navigator is the Graphical Interface to the ISE Tool Suite

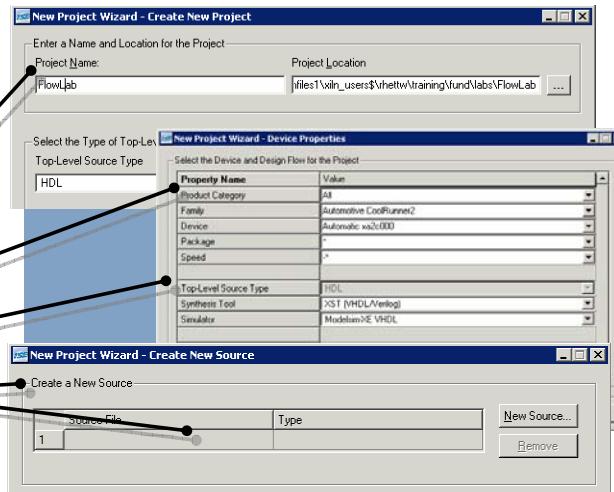


NOTES

Here you see the ISE™ Project Navigator, the main graphical interface to the ISE tool suite, design entry, and implementation tools. ISE is an acronym for Integrated Software Environment. The ISE™ software is the Xilinx tool suite that includes design entry tools (PC only), implementation tools, and timing analysis tools. The ISE software also interfaces with third-party tools for synthesis and simulation. You can implement your design with a simple double-click and fine-tune with easy-to-access software options. There are four main windows in the Project Navigator. In the upper-left corner is the Sources in Project window. Directly below that window is the Processes for Current Source window. To the right is the main working window and Design Summary window, and below is the Message (or Console) window. The ISE software includes XST synthesis, CORE Generator™ tool, Architecture Wizard, ECS schematic editor, StateCAD state diagram editor, and HDL Bencher waveform editor (for testbench creation). The ISE software runs on Windows, UNIX, and Linux operating systems.

Creating a Project

- Select File → New Project
- New Project Wizard guides you through the process
 - Project name and location
 - Target device
 - Software flow
 - Create or add source files

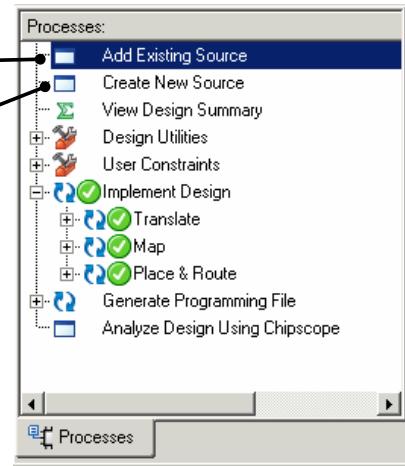


NOTES

To create a project in the Project Navigator, select **File**, then **New Project**. The new Project Wizard guides you through the process.

Creating and Adding Source Files

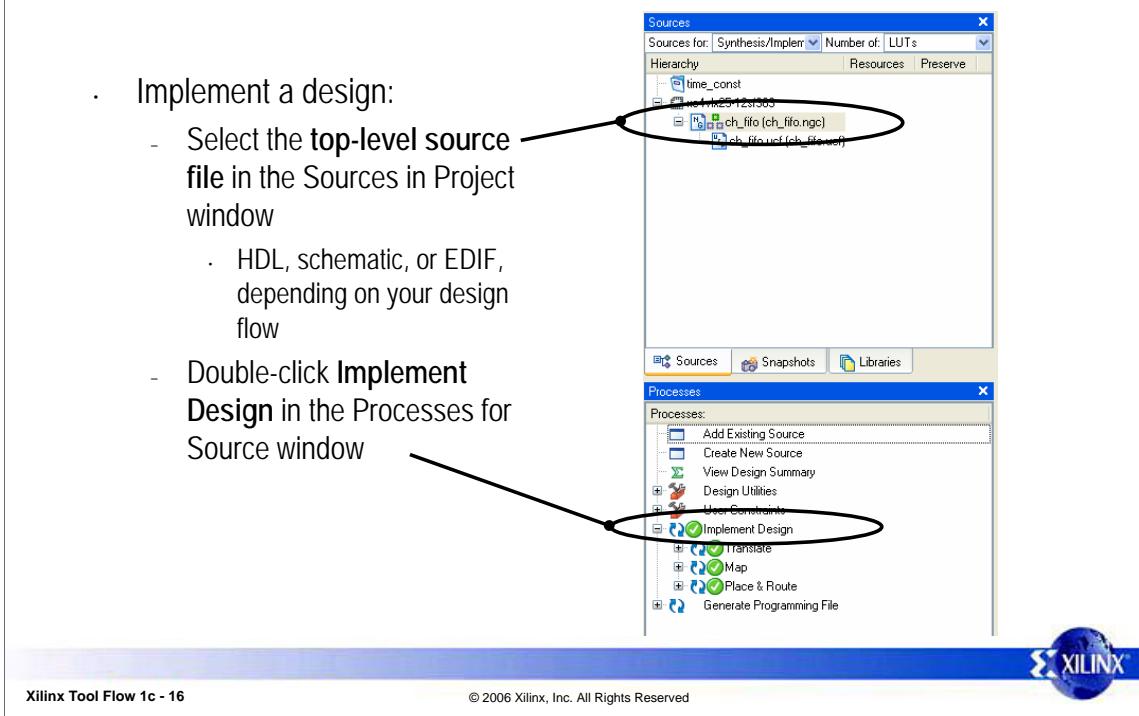
- Double-click **Add Existing Source** to include an existing source file
- Double-click **Create New Source** and choose the type of file to create a new source file
 - HDL
 - IP
 - Schematic
 - State Diagram
 - Testbench
 - Constraints



NOTES

The Processes for Source window is where you can create and add source files. Double-click Add Existing Source to include an existing source file. To create new source files, double-click Create New Source and choose the type of file to create a new source file. Choices include HDL, IP, schematic, state diagram, testbench, and constraints.

Implementing a Design



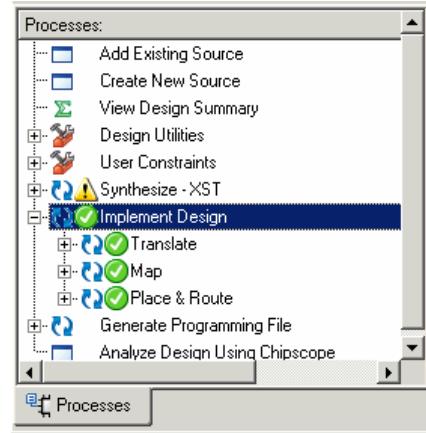
NOTES

To implement a design, select the top-level source file in the Sources in Project window. Choices include HDL, schematic, or EDIF, depending on your design flow. You can select a lower-level file in the Sources in Project window to perform resource estimation, then double-click Implement Design in the Processes for Source window.

Checking the Implementation Status

- . The ISE™ software will run all of the necessary steps to implement the design
 - Synthesize HDL code
 - Translate
 - Map
 - Place & Route

 = process was completed successfully
 ! = warnings
 ? = a file that is out of date
 X = errors

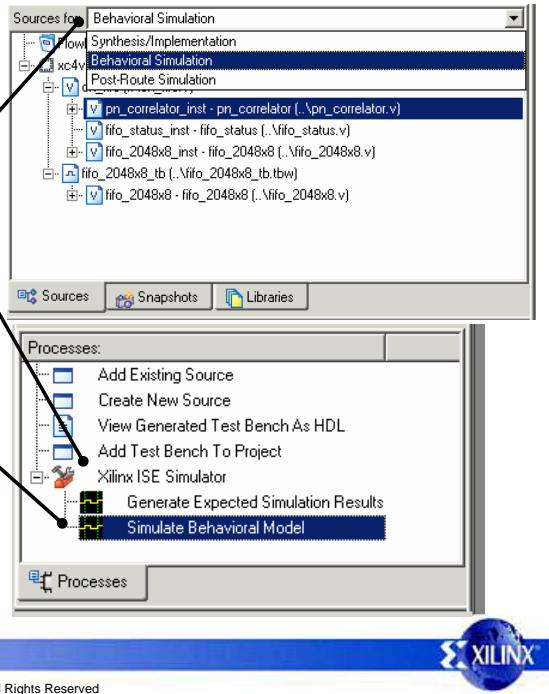


NOTES

The ISE™ software will run all of the necessary steps to implement the design. It will synthesize HDL code, translate, Map, and Place & Route. Progress and status are indicated by icons. A green check mark indicates that the process was completed successfully. A yellow exclamation point (!) indicates warnings. A yellow question mark (?) indicates a file that is out of date. And a red X indicates errors.

Simulating a Design

- Simulate a design:
 - Select Sources for: Behavioral Simulation
 - Expand Xilinx ISE Simulator in the Processes for Source window
 - Double-click Simulate Behavioral Model or Simulate Post-Place & Route Model
 - . You can also simulate after Translate or after Map



Xilinx Tool Flow 1c - 18

© 2006 Xilinx, Inc. All Rights Reserved

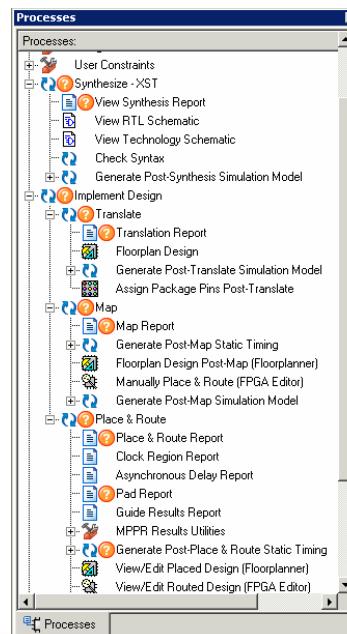


NOTES

To simulate a design, select Sources for Behavioral Simulation from the drop-down box. Next, select a testbench file in the Sources for Behavioral Simulation window. As you select different files in this window, the Processes window adjusts accordingly to show you the processes available for that source. Expand Xilinx ISE Simulator in the Processes for Source window. Then double-click Simulate Behavioral Model or Simulate Post-Place & Route Model. You can also simulate after Translate or after Map. The ISE™ Project Navigator can also be integrated with ModelSim.

Viewing Subprocesses

- Expand each process to view subtools and subprocesses
 - Translate
 - Floorplan
 - Assign package pins
 - Map
 - Analyze timing
 - Place & Route
 - Analyze timing
 - Floorplan
 - FPGA Editor
 - Analyze power
 - Create simulation model



Xilinx Tool Flow 1c - 19

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

To view subtools and subprocesses, expand each process. For Translate, you see Floorplan and Assign package Pins. For Map, you see Analyze Timing. For Place & Route, you see Analyze Timing, Floorplan, FPGA Editor, Analyze Power, and Create Simulation Model. There are many subprocesses for each main process. Not all of these subprocesses are necessary for a successful design implementation. Some of the more popular subprocesses include the Floorplanner, FPGA Editor, and Timing Analyzer. The Floorplanner physically constrains or places parts of your design. This tool can be accessed after MAP or after PAR. More information is available in the *Advanced FPGA Implementation* course.

The FPGA Editor is used after PAR. This tool displays the actual placement and routing of a completed design. One popular use for the FPGA Editor is to route internal signals out to a pin for testing. More information on this feature is available at www.xilinx.com and in the *Advanced FPGA Implementation* course. The Timing Analyzer is also available after MAP and PAR. The MAP and PAR static timing reports can be viewed by using the Timing Analyzer. The Timing Analyzer performs static timing analysis. Various reports can be generated, including customized reports. More information on the Timing Analyzer is available in the *Designing for Performance* course.

The Design Summary Displays Design Data

- Quick View of Reports, Constraints
- Project Status
- Device Utilization
- Design Summary Options
- Performance and Constraints
- Reports

FPGA Design Summary

Project File: FlowLab.sce **Current State:** Placed and Routed

Module Name: ch_00 **Target Device:** xc4vlx15-12f36I

Product Version: ISE, 8.1 **• Errors:** No Errors **• Warnings:** 7 Warnings **• Updated:** Wed Oct 12 10:09:46 2005

Device Utilization Summary			
	Used	Available	Utilization
Number of Slice Flip Flops	80	12,288	1%
Number of 4 input LUTs	136	12,288	1%
Logic Distribution			
Number of occupied Slices	86	6,144	1%
Number of Slices containing only related logic	86	86	100%
Number of Slices containing unrelated logic	0	66	0%
Total Number 4 input LUTs	137	12,288	1%
Number used as logic	136		
Number used as route-thru	1		
Number of bonded IOBs	18	240	7%
Number of BUFG,BUFGCTRLs	2	32	6%
Number used as BUFQs	2		
Number used as BRAMCBR16s	0		
Number of FIFO16s/RAM16s	1	48	2%
Number used as FIFO16s	0		
Number used as RAM16s	1		
Total equivalent gate count for design	1,639		
Additional JTAG gate count for IOBs	664		

Performance Summary

Final Timing Score: 0 **Pinout Data:** Pinout Report

Routing Results: All Signals Completely Routed **Clock Data:** Clock Report

Timing Constraints: All Constraints Met

Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Address Report	Current	Tue Oct 11 15:48:49 2005	0	73 Warnings	10 Infos
Translation Report	Current	Tue Oct 11 15:48:54 2005	0	0	0
Timing Report	Pending	Y... Mon 11 14:49:08 2005	n/a	n/a	9 Infos

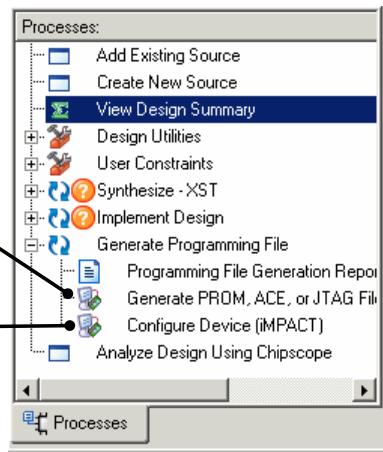
Xilinx Tool Flow 1c - 20 © 2006 Xilinx, Inc. All Rights Reserved

NOTES

The new Design Summary screen displays important design data in a single page, easy-to-read format. Information from the Map and Timing reports are displayed, as well as links to all software reports.

Programming the FPGA

- There are two ways to program an FPGA
 - Through a PROM device
 - . You must generate a file that the PROM programmer can understand
 - Directly from the computer
 - . Use the iMPACT configuration tool



NOTES

Knowledge Check

Select all the statements that describe the ISE™ Project Navigator?

- Graphical interface for implementing a design
- Can be used to edit HDL source files
- Shows specific processes that are available for each source
- Can be used to create schematics
- Displays design summary
- Can be used as a real-time hardware debugger
- Used for invoking other Xilinx tools (PACE, Constraints Editor, etc.)

*Place a check next to the
correct answers*



NOTES

Answers

- . Statements that describe the ISE™ Project Navigator
 - Graphical interface for implementing a design
 - Can be used to edit HDL source files
 - Shows specific processes that are available for each source
 - Can be used to create schematics
 - Displays design summary
 - Can be used as a real-time hardware debugger
 - Used for invoking other Xilinx tools (PACE, Constraints Editor, etc.)



NOTES

Outline

- Overview
- ISE
- **Summary**



NOTES

Summary

- Implementation means more than Place & Route
- Xilinx provides a simple *pushbutton* tool to guide you through the design process



NOTES

Where Can I Learn More?

- . Complete design flow tutorials
 - www.xilinx.com → Documentation → Download → Tutorials
- . On Implementation: Development System Reference Guide
 - www.xilinx.com → Documentation → Software Manuals
 - Documentation may also be installed on your local computer
- . On simulation: ISE Simulator Help
 - www.xilinx.com → Documentation → Software Manuals
- . Configuration Problem Solver
 - www.xilinx.com → Support → Problem Solvers → Configuration Problem Solver



NOTES



Architecture Wizard and PACE

© 2006 Xilinx, Inc. All Rights Reserved

NOTES

This Architecture Wizard and PACE module will show you at least two uses for the Architecture Wizard and two features of PACE, and how to create quality pin assignments for Xilinx FPGAs.

Objectives

After completing this module, you will be able to:

- List at least two uses for the Architecture Wizard
- Identify two features of PACE
- Create quality pin assignments for Xilinx FPGAs



NOTES

Outline

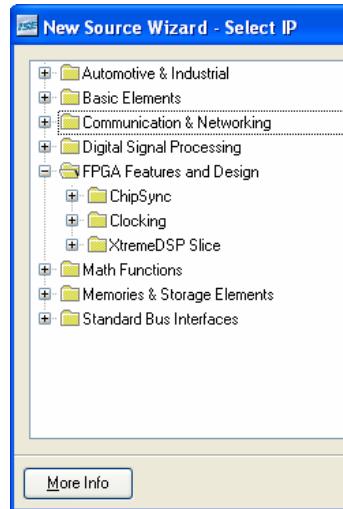
- . Architecture Wizard
 . PACE
 . I/O Layout
 . Summary



NOTES

The Architecture Wizard Contains Several Wizards

- Double-click Create New Source
 - Select IP (COREGen & Architecture Wizard), and click Next



Architecture Wizard and PACE 1d- 4

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

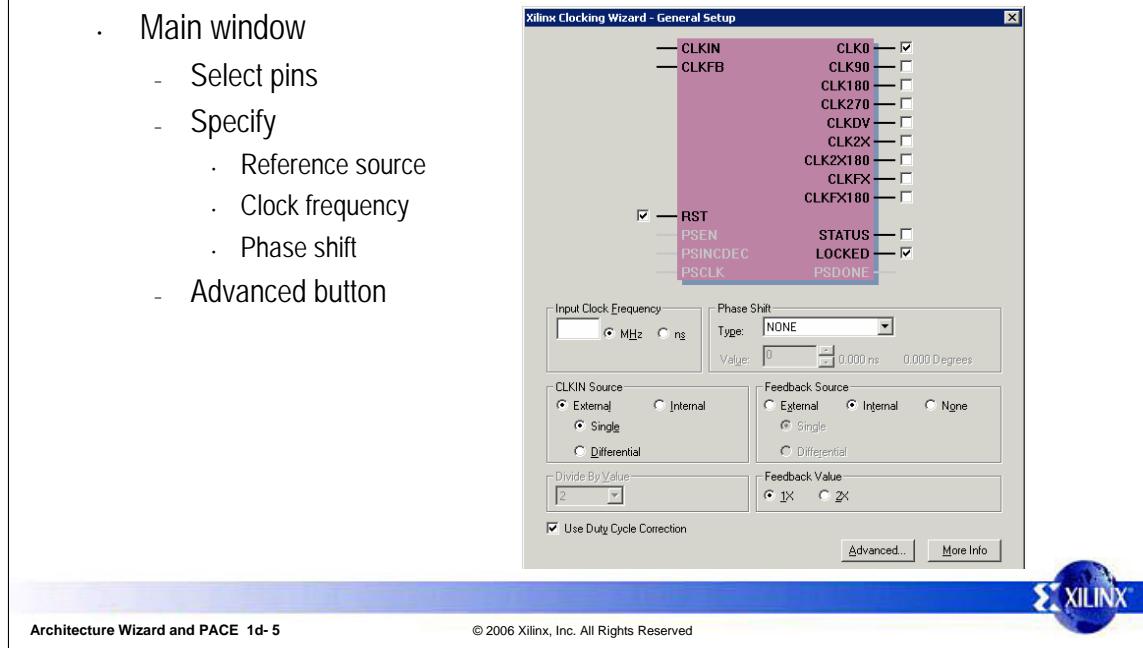
The Architecture Wizard contains several wizards. They include the Clocking Wizard, RocketIO™ Wizard, ChipSync™ Wizard, and more. To open the New Source Wizard and select IP, double-click **Create New Source**. Select **IP (COREGen & Architecture Wizard)**, and click **Next**.

*The RocketIO™ MGT is only available for projects targeting the Virtex™-II Pro, Virtex™-II Pro X, or Virtex™-4 devices. The ChipSync Wizard is only available for projects targeting the Virtex-4 device family. Additional wizards exist for Virtex-4 designs, providing access to special I/O and DSP functions.

The other folders, shown in the window, will launch the CORE Generator™ system. To learn more about the CORE Generator system, refer to the “CORE Generator System” module in the *Designing for Performance* course.

The Clocking Wizard Helps You Define the DCM

- Main window
 - Select pins
 - Specify
 - . Reference source
 - . Clock frequency
 - . Phase shift
 - Advanced button



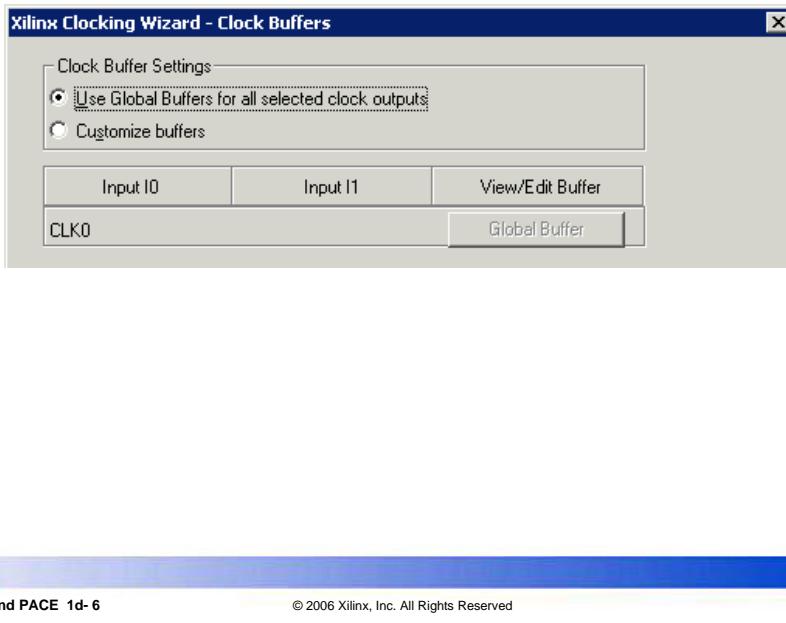
NOTES

The Clocking Wizard takes you through several dialog boxes to define the DCM that you want to create.

In the first dialog box, specify the pins, clock frequency, and phase shift that you require. This dialog box also has an Advanced button, which allows you to access additional options. These advanced options include the following:

- 1) Wait for the DCM lock before the DONE signal goes high
(STARTUP_WAIT attribute, default is FALSE)
- 2) Divide the input clock frequency by 2
(CLKIN_DIVIDE_BY_2 attribute, default is FALSE)
- 3) Align the DCM output clocks and the clock input pin of the FPGA
(DESKEW_ADJUST attribute, default is SYSTEM_SYNCHRONOUS)

Specify Types of Buffers You Want with Each Clock Signal



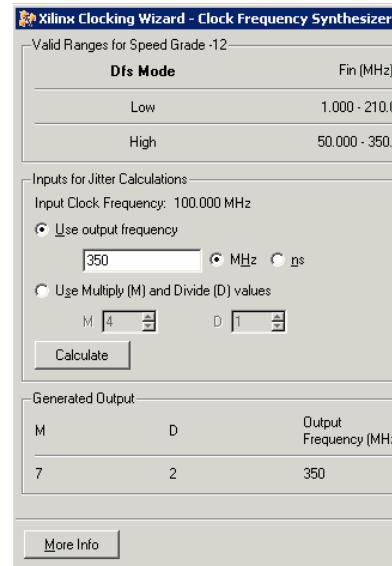
NOTES

In this dialog box, you can specify the types of buffers that you want with each clock signal. By default, the tools will use a BUFG. If you want to use a BUFGMUX, BUFGCE, or other low-skew lines, you must select the Customize radial button.

Adding buffers: Connect clock buffers (BUFG, BUFGMUX) to the selected output pins of the DCM.

Specify FX Frequency and Calculate Jitter

- . Frequency synthesizer
 - Select M/D value
 - OR
 - Specify frequency
- . Calculate button for jitter
 - Period jitter is evaluated for CLKFX output



Architecture Wizard and PACE 1d- 7

© 2006 Xilinx, Inc. All Rights Reserved

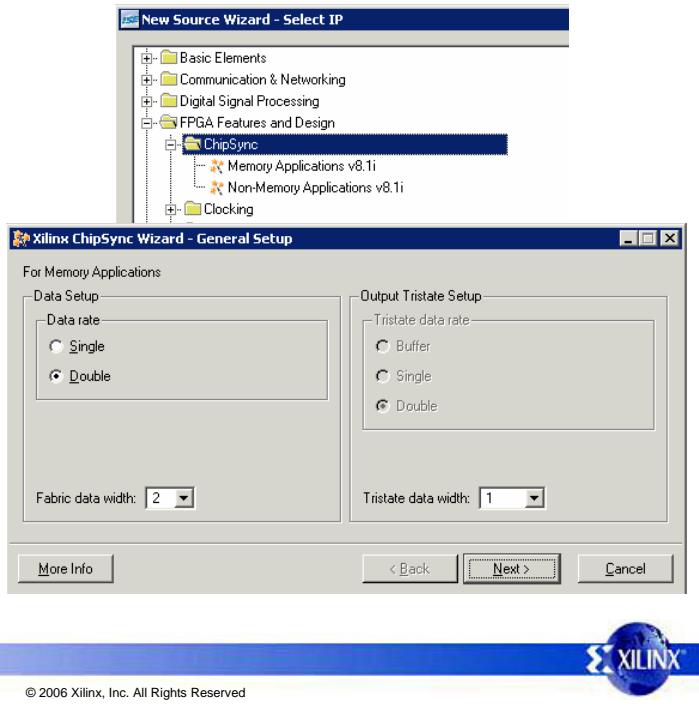


NOTES

This last dialog box appears only if the CLKFX output was selected in the first dialog box. This window displays the valid input and output frequencies for the two different modes (high and low). Then, it allows you to enter values to calculate jitter.

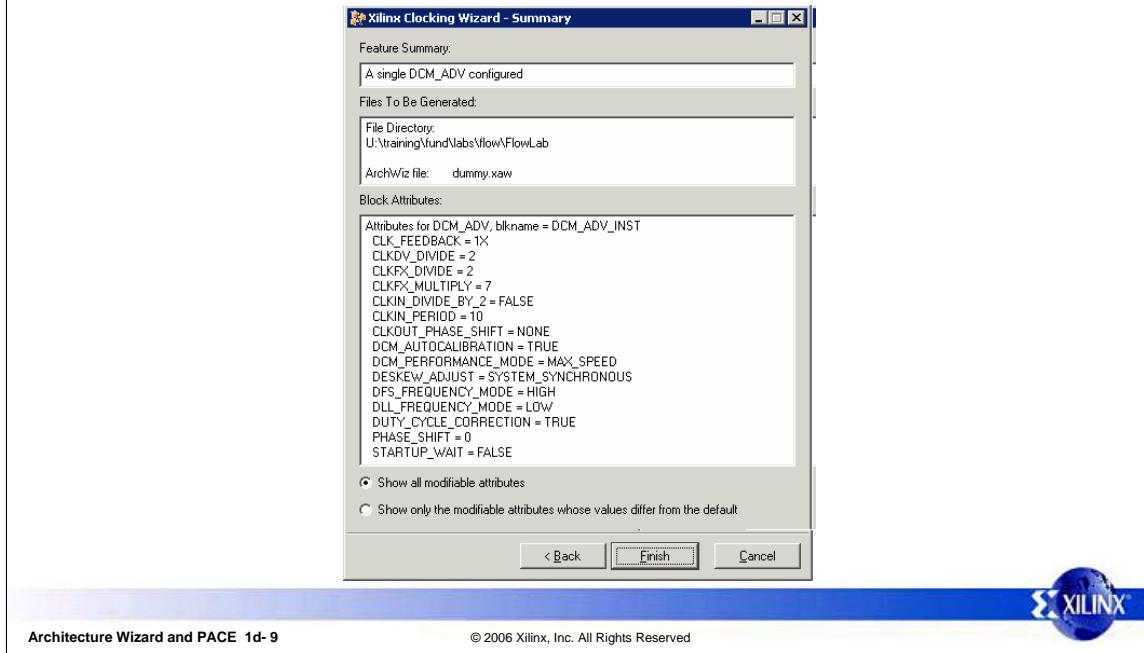
Viewing the ChipSync Wizard

- Configures the ISERDES and OSERDES components
- Select memory or networking application
- Set options
 - Data width
 - SDR or DDR



NOTES

By Default, Attributes are Written into the HDL Files



NOTES

By default, attributes are written into the HDL files. For example: Verilog with Synplicity.

Simulation attributes are passed through 'defparam'

```
defparam DCM_INST.CLKDV_DIVIDE=2;
```

Synthesis attributes are passed within metacomments

```
/*synthesis xc_props="CLK_FEEDBACK=1x,
CLKDV_DIVIDE=2, ....";*/
```

Alternatively, the UCF flow can be used to customize the components.

The Wizard creates a file named <component_name>_arwz.ucf. Just cut and paste from this file to the main UCF file. Remove attributes from the generated HDL file.

The Architecture Wizard will write code with attributes that your synthesis tool can understand. The synthesis or design flow selected for the project determines the type of attributes. This flow is selected when you open a new project. To change synthesis tools or design flows in the Project Navigator, right-click the **device and design flow** line in the Sources for Current Source window, and then click **Properties**. This provides you with the option to change synthesis tools. For example, for an Exemplar flow, the Exemplar attributes will be used:

```
// exemplar attribute DCM_INST CLKDV_DIVIDE 2
```

UCF = User Constraints File

Knowledge Check

Can you list the advantages of using the Architecture Wizard in your design?

*Raise your hand if you
think you know the
answer*



NOTES

Answers

Advantages of using the Architecture Wizard in your design:

- Not required to create the instantiation on your own
- Automatically creates usable source code
- Synthesis and simulation attributes are already written
- Easy-to-use dialog boxes



NOTES

Outline

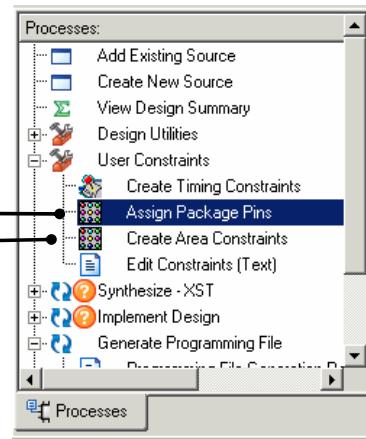
- Architecture Wizard
- • **PACE**
- I/O Layout
- Summary



NOTES

PACE is Your Pinout and Area Constraints Editor

- Assign Package Pins
- Create Area Constraints



NOTES

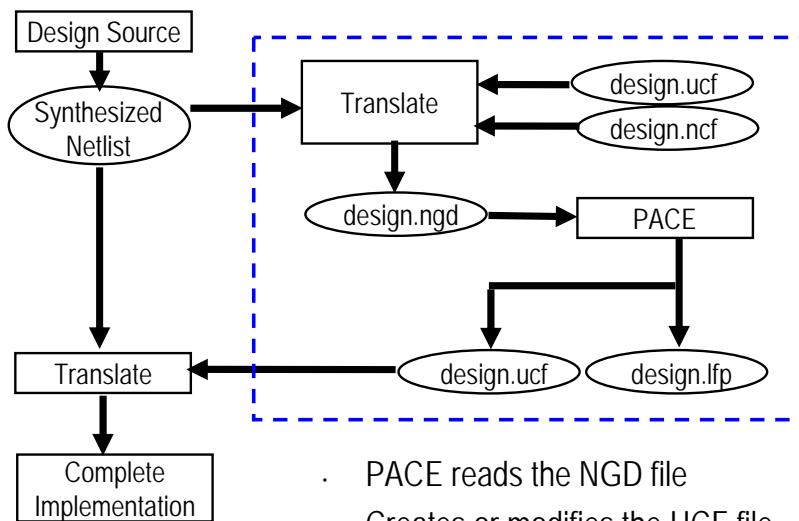
PACE is your Pinout and Area Constraints Editor. You can assign package pins. Here, you can also assign I/O locations, specify I/O banks and I/O standards, prohibit I/O locations, and verify the pin type to the logic assignment. You can also perform a DRC check to prevent illegal placements.

You can create area constraints. Here, you can create area constraints for logic and display I/Os on the periphery to show connectivity. Begin floorplanning early in the design flow. Also verify area constraints.

DRC = Design Rule Check

The PACE DRC check detects illegal placements only. For example, it will flag an error if you place pins using incompatible SelectIO™ standards onto locations that share supply voltages.

Design Flow with PACE (Existing Design)



NOTES

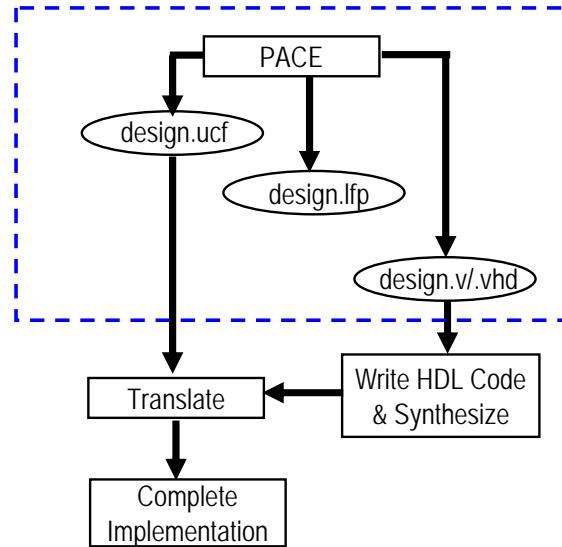
PACE reads the NGD file and creates or modifies the UCF file.

LFP = Logical FloorPlan file. This file is used with the PACE tool only. This file contains color settings and other information specific to the PACE tool.

NGD = Native Generic Design. This file is the output file of NGDBuild, which combines all of the design netlist and constraints files.

Design Flow with PACE (New Design)

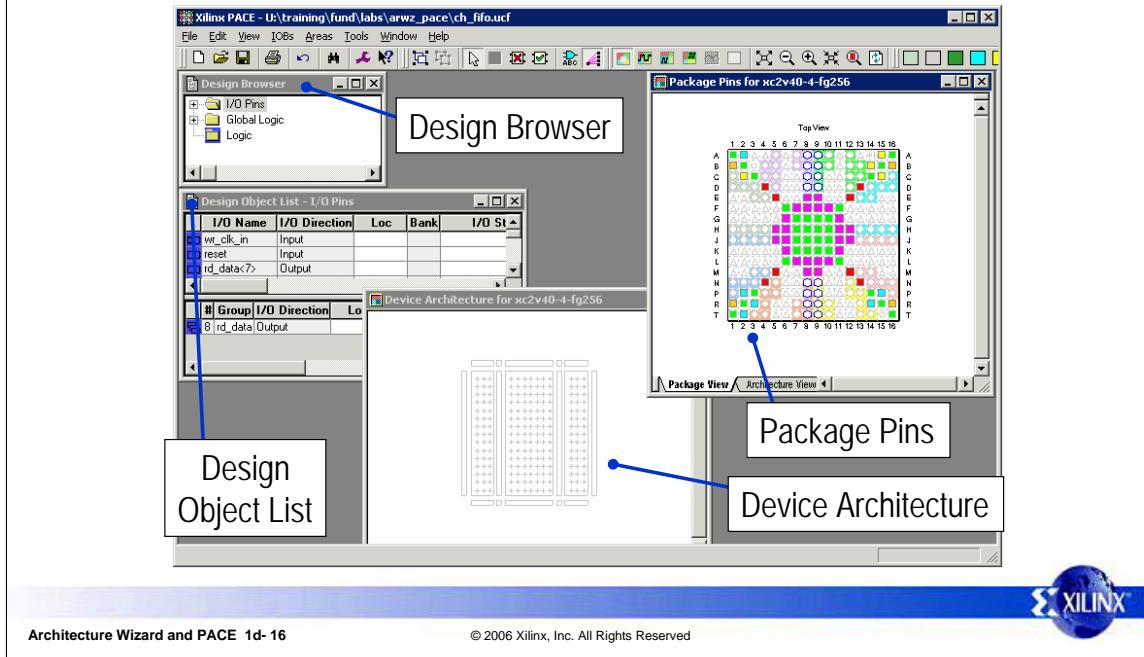
- PACE creates the top-level HDL file
- Creates the UCF file



NOTES

PACE creates the top-level HDL file. It contains port information only. Use this file as a starting point for writing the remaining HDL code. PACE also creates the UCF file.

Viewing the PACE GUI



NOTES

The Design Browser navigates parts of your design. Items cannot be added or removed from this window. The list allows you to display different parts of your design in the Design Object List (DOL) window.

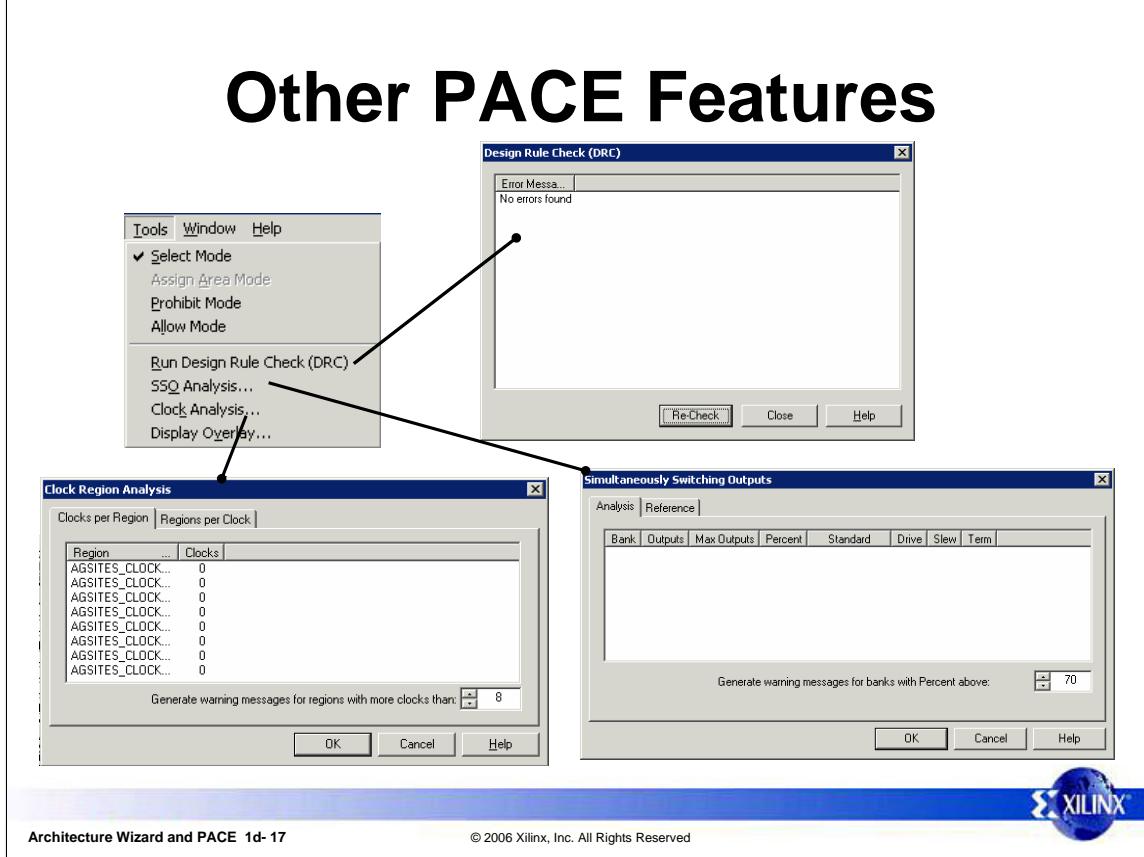
Depending on what is selected in the Design Hierarchy window, the DOL window will display signal names or components from your design. This list can be sorted by various grouping methods: alphabetical order, location, banks, I/O direction, or whatever categories are available.

The Device Architecture window displays a graphical view of your device resources. Assigned pins are denoted by colors in this window.

The Package Pin window provides a graphical view of the package pins. This window is color-coded so that it is easy to identify pins in the same bank, and it is easy to identify ground and power pins. Use the Package Pin Legend box to identify pins in the Package Pin window.

You can enter pin locations into the Design Object List window, or you can drag-and-drop I/Os from the Design Browser or Design Object List windows into the Device Architecture or Package Pin windows.

Other PACE Features



NOTES

Tools menu:

Run Design Rule Check (DRC): Checks that all pins assigned to each bank use compatible I/O standards

SSO Analysis: Flags potential ground bounce problems caused by pin assignments

Clock Analysis: Displays clock distribution to each clock region

Import and export pin assignments:

Easy communication between PACE and board layout tools

File → Import and File → Export

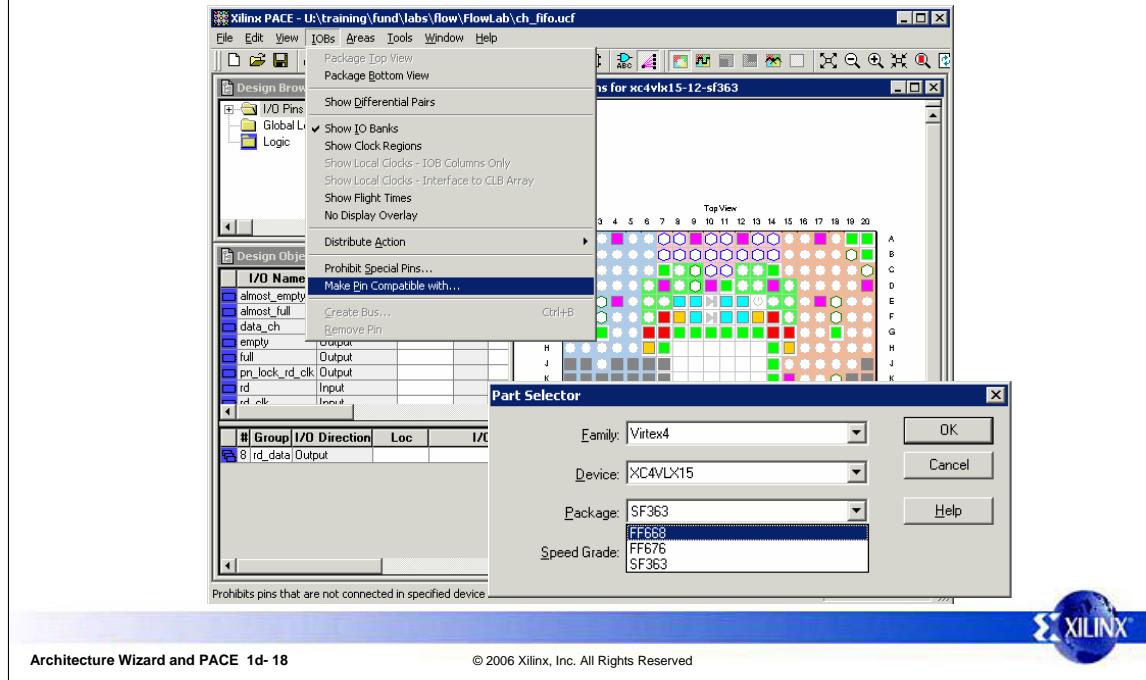
Uses comma separated (CSV) text file format

The PACE DRC checks for incompatible I/O standards that have been assigned to the same I/O bank. A message window lists each error and the cause (incompatible VCCO, VREF, etc.). SSO analysis opens a dialog that shows the recommended SSO guidelines and how the current layout compares to those guidelines.

Clock analysis shows how each global clock net is distributed across the device. There are limits regarding how many global clock lines can be used in each clock region. If the current layout violates these limits, you will be warned that the I/O layout must be changed (or else some clocks may need to use local routing resources). You can also import pin assignments from a CSV file. To see the required format, open **PACE**, make a **pin assignment**, and export the **data**. The file will include the expected column headers.

To import a pin assignment, the CSV file must contain at least the Pin Number and Signal Name columns. All other columns are optional.

Viewing Package Migration



NOTES

Use this feature when a design may move to a different device or package. PACE can display incompatible pins to help with pinout planning. Menu command: **IOBs** → **Make Pin Compatible With**

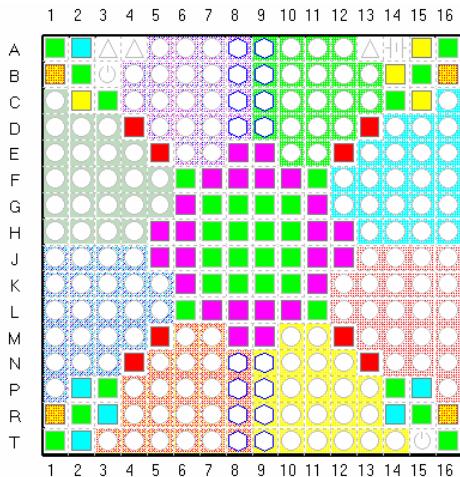
The PACE DRC checks for incompatible I/O standards that have been assigned to the same I/O bank. A message window lists each error and the cause (incompatible VCCO, VREF, etc.).

SSO analysis opens a dialog that shows the recommended SSO guidelines and how the current layout compares to those guidelines.

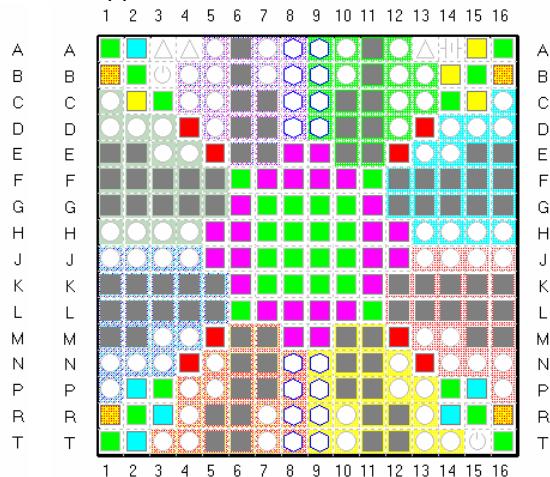
Clock analysis shows how each global clock net is distributed across the device. There are limits regarding how many global clock lines can be used in each clock region. If the current layout violates these limits, you will be warned that the I/O layout must be changed (or else some clocks may need to use local routing resources).

Viewing Pin Compatibility

Original Package View of 2V250-FG256



Modified Package View after Pin compatibility is applied with 2V40-FG256



Architecture Wizard and PACE 1d- 19

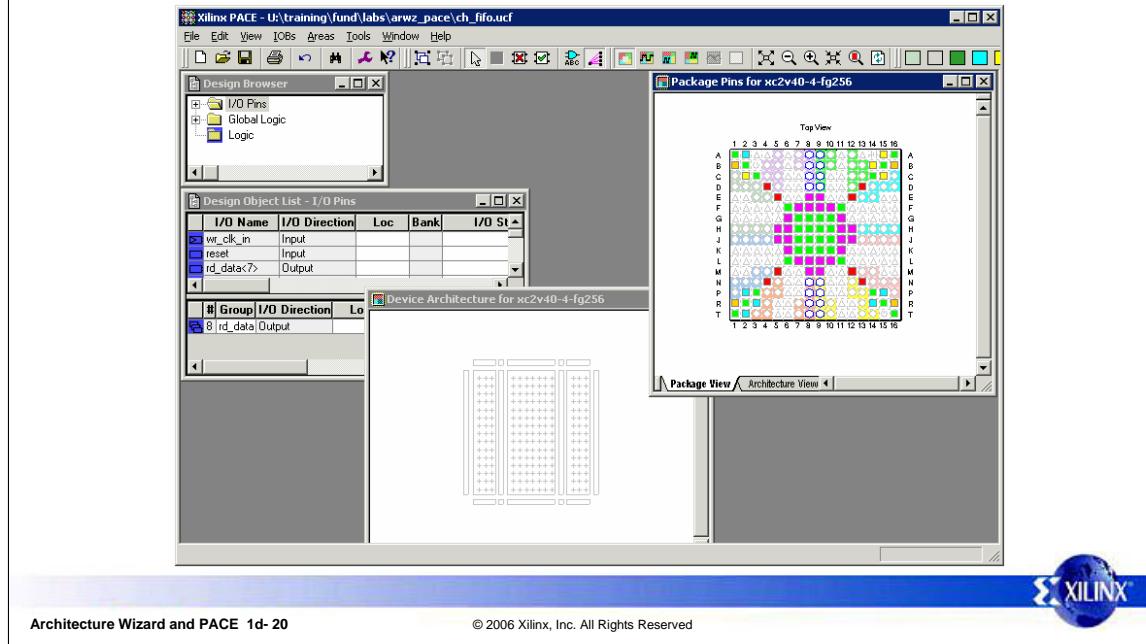
© 2006 Xilinx, Inc. All Rights Reserved



NOTES

This view shows a 2V250-FG256 package pin view after you have set the pins to be compatible with 2V40-FG256. The grayed-out pin locations should not be used when targeting the larger 2V250 device to maintain pin compatibility with the smaller 2V40 device.

Use PACE to Assign Pins and Create Area Constraints



NOTES

Use PACE to assign pins and create area constraints for an existing design. After creating HDL code, launch **PACE**. Assign pin locations and area constraints. Select **File → Save** to save the UCF file.

Plan your pinout and create a skeleton top-level HDL file for a new design.

In PACE, click **File → New** and choose **Create New UCF and Design**.

Enter file names for the new UCF and the new top-level HDL file (VHDL or Verilog).

Enter pin names and locations in the PACE GUI.

Select **File → Save** to save the UCF and HDL files.

Knowledge Check

Can you list at least two features available in PACE?

*Raise your hand if you
think you know the
answer*



NOTES

Answers

Features available in PACE:

- Drag-and-drop I/Os and area locations onto device package and layout windows
- DRC checking
- Enter I/O standards
- Prohibit and Allow sites
- Easy package migration



NOTES

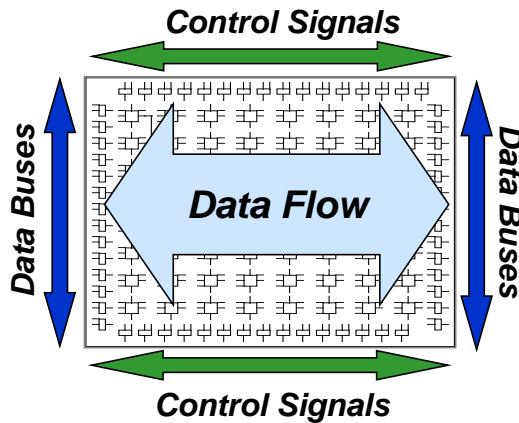
Outline

- Architecture Wizard
- PACE
- · **I/O Layout**
- Summary



NOTES

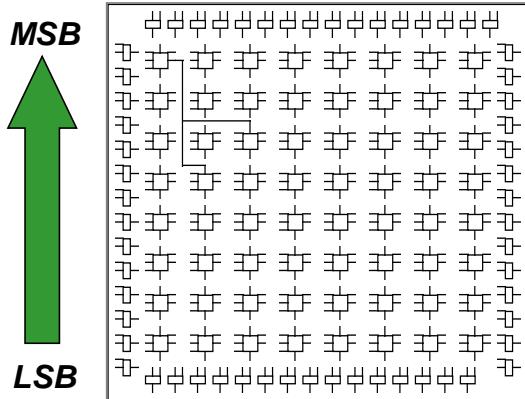
I/O Layout Guidelines



NOTES

I/O pins for control signals are on the top or the bottom of the die. Control signals are usually routed vertically. I/O pins for data buses are on the left or the right edge of the die. FPGA architecture favors horizontal data flow.

Data Bus Layout



NOTES

Arithmetic functions with more than five bits use carry logic. Carry chains require a specific vertical orientation since the LSB is always at the bottom. Likewise, your pin assignments should follow this same pattern. Note that this does not mean that each bit of the bus needs to be next to each other. In fact, Xilinx recommends dispersing other signal between the bits if possible. But, make sure that your bit ordering follows this pattern, LSB beneath the MSB and the bits are in order.

Outline

- Architecture Wizard
- PACE
- I/O Layout
-  · Summary



NOTES

Summary

- . The Architecture Wizard consists of several wizards, including:
 - Clocking Wizard
 - RocketIO™ Wizard
- . PACE can easily assign I/O locations and create area constraints for logic

NOTES

Where Can I Learn More?

- . Architecture Wizard
 - *More Info* buttons in dialog boxes
- . PACE Online Help



NOTES



Xilinx Tool Flow Lab

Introduction

©2006 Xilinx, Inc. All Rights Reserved

NOTES

Lab Instructions

- . Below each general instruction for a given procedure, you will find accompanying step-by-step directions and illustrated figures that provide more detail for performing the general instruction.
 - General Instruction
 - . Step-by-step detailed directions on how to perform the general instruction
- . If you feel confident about a specific instruction, feel free to skip the step-by-step directions and move on to the next general instruction in the procedure.



NOTES

Perform Labs in a Virtual Environment with Toolwire

- Secure servers in Windows or UNIX
- Xilinx lab files
- Lab instructions in your workbook
- Other course reference material



NOTES

With Toolwire, you can perform Xilinx learning labs in a virtual environment and receive online hands-on experience with the Xilinx software tools. Toolwire provides secure servers running all of the necessary software in either a Windows or UNIX environment. Xilinx lab files are provided in your personal account directory, and lab instructions are included with the lecture slides. Other course reference materials, such as user guides and data sheets, may be included as well.

Toolwire Performance

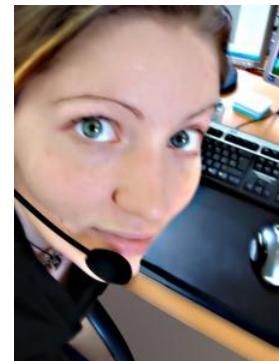
- Software environment is completely set up and tested!
- Easier to fix and maintain!
- Note that Local facility LAN bandwidth and ISP may affect performance
 - May be slower than running software on a local machine, especially when using graphic-intensive applications



NOTES

When to Get Help with Toolwire

- If application seems to freeze
- If Toolwire loses connection
 - Normally, exiting Toolwire and logging back in retains all work



NOTES

Objectives

After completing this lab, you will be able to:

- Create a new project in the ISE™ Project Navigator
- Add design files to a project
- Use the ISE Simulator to simulate a design
- Use default software options to implement a design



NOTES

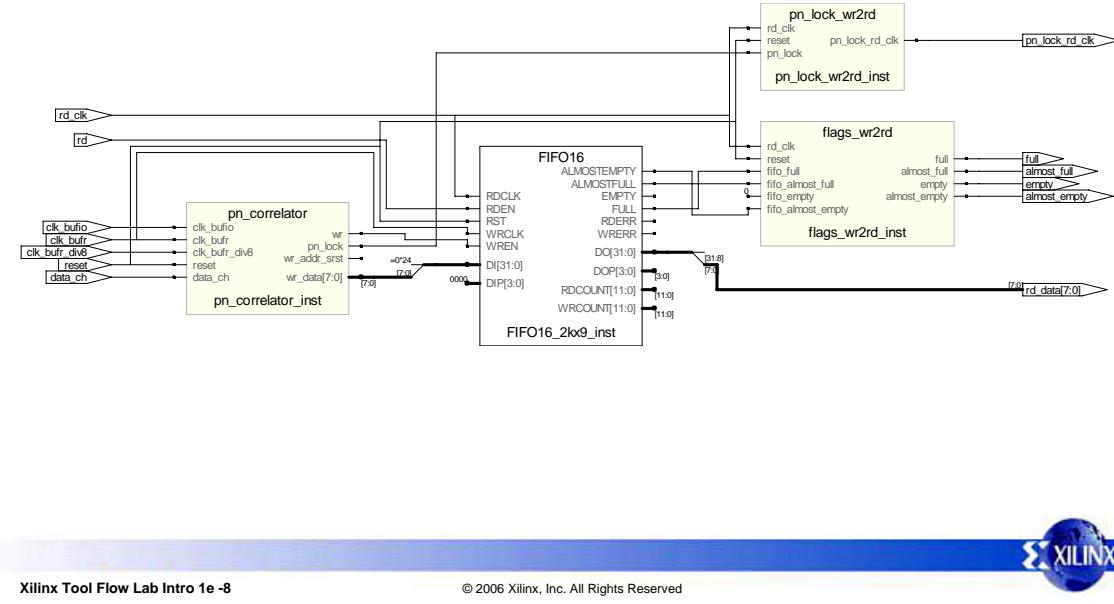
Introduction

- . This lab illustrates how to:
 - Create a new design with ISE™ Project Navigator
 - Specify project attributes
 - Add design files
 - Use ISE Simulator for behavioral simulation
 - Implement a design
 - Navigate the ISE Project Navigator software



NOTES

Channel FIFO Design



NOTES

This is a block diagram for the lab design, which you will use in this course.

The PN Correlator searches for a specific pattern (PN code) in the input data stream. Once the PN code is found, the following 255 bytes of data are stored in the FIFO. After the 255 bytes of data are stored, the correlator begins to search for the PN code again.

The FIFO Status block generates read/write addresses for the FIFO and creates flags that indicate how much data has been stored in the FIFO. These flags are used by the Data Control block to decide from which channel to read.

The FIFO block is a simple dual-port RAM that stores the data from the PN Correlator block.

For more information about the design, refer to the “Detailed Design Description for Labs” section of this course workbook or the HDL source code for this design.

General Flow

- Step 1: Create a new project
- Step 2: Add design files
- Step 3: Simulate design
- Step 4: Implement design



NOTES

Xilinx Tool Flow Lab



Xilinx Tool Flow Lab

Introduction

This lab introduces the ISE™ software.

Objectives

After participating in this demonstration, you will be able to:

- Create a new project in the ISE Project Navigator
- Add design files to a project
- Use the ISE Simulator to simulate a design
- Use the default software options to implement a design

Procedure

This demonstration comprises four primary steps: you will create a new project, add design files to the project, simulate the design, and finally implement the design.

For each procedure within a primary step, there are general instructions (indicated by the  symbol). These general instructions only provide a broad outline for performing the procedure. Below these general instructions, you will find accompanying step-by-step directions and illustrated figures that provide more detail for performing the procedure. If you feel confident about completing a procedure, you can skip the step-by-step directions and move on to the next general instruction.

Note: If you are not using **Toolwire** to perform this lab, all software programs, files, and projects will be located on the C:\ drive instead of R:\.

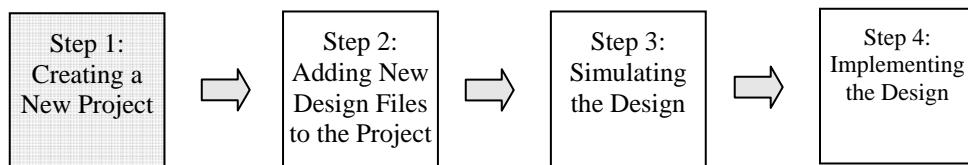
Note: If you are unable to complete the lab at this time, you can download the original lab files for this module from the Xilinx FTP site at [ftp://ftp.xilinx.com/pub/documentation/education/fpga13000-82-xlnx_lab_files.zip](http://ftp.xilinx.com/pub/documentation/education/fpga13000-82-xlnx_lab_files.zip). These are the original lab files and do not contain any work you may have previously completed.

Creating a New Project

Step 1

For each procedure within a primary step, there are general instructions (indicated by the  symbol). These general instructions only provide a broad outline for performing the procedure. Below these general instructions, you will find accompanying step-by-step directions and illustrated figures that provide more detail for performing the procedure. If you feel confident about completing a procedure, you can skip the step-by-step directions and move on to the next general instruction.

General Flow for this Lab:



Log into the Toolwire server, which provides the software and files you will need for all of the labs and demonstrations in this course.

- ①** Open a **Web browser** and go to <http://dcm.toolwire.com/login>
- ②** Enter the **Username** and **Password** provided by the instructor, and click **Login**

Toolwire will launch a virtual Windows desktop. Notice that your screen appears to have two Windows taskbars. The lower taskbar is for your local computer. The upper taskbar is part of the Toolwire desktop. Always use the upper taskbar during labs and demonstrations.



Launch the ISE™ Project Navigator and create a new design project.



1. What is the purpose of creating a project in ISE Project Navigator? That is, what will the project include?
-
-

- ①** Select **Start → Programs → Xilinx ISE 8.2i → Project Navigator**



Some popups may appear with messages regarding reading a network directory or running WebUpdate. These messages appear because we are running the tools on the Toolwire servers, and they can be ignored. Dismiss the popups to continue.

- ② In the Project Navigator, select **File → New Project**

The New Project Wizard opens (**Figure 1f-1**)

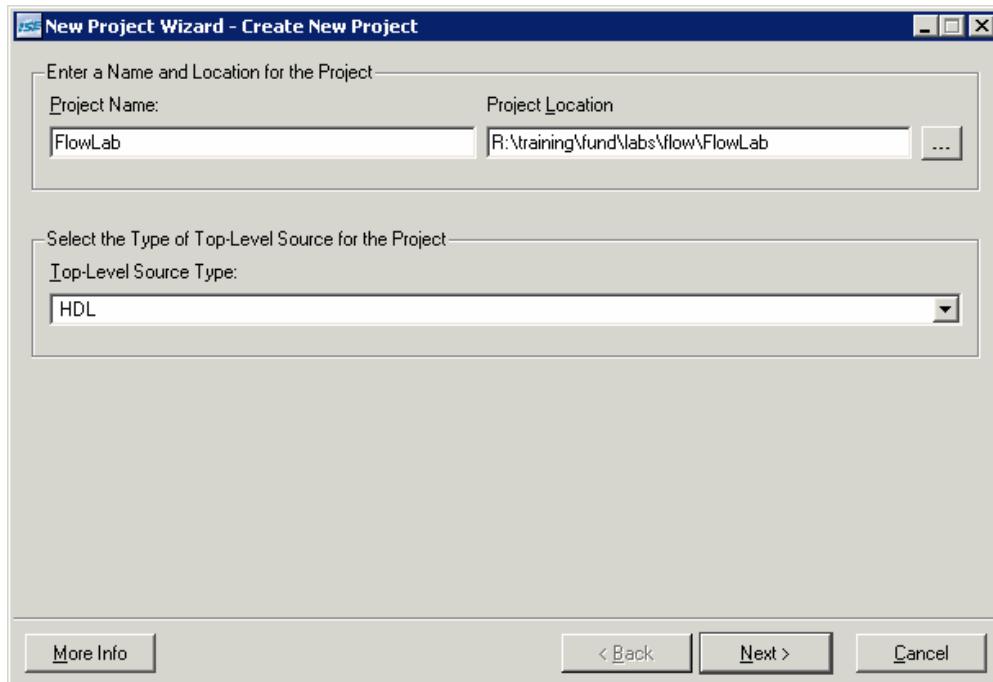


Figure 1f-1. New Project Wizard

- ③ For Project Name, type ***FlowLab***
- ④ For Project Location, use the “...” button to browse to **R:\training\fund\labs\flow** and click **OK**



Make certain that the Project Location is **R:\training\fund\labs\flow**, without another subdirectory named **\FlowLab** after it. This is extremely important in the Toolwire environment for running the simulation executable.

- ⑤ Click Next

The Device and Design Flow dialog appears (**Figure 1f-2**)

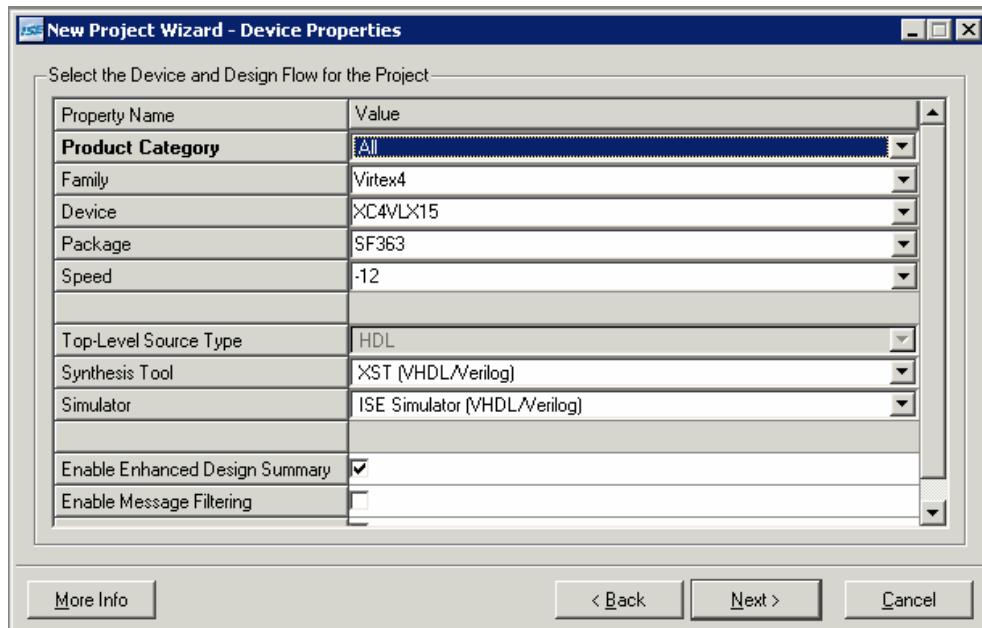


Figure 1f-2. Device and Design Flow Dialog

- ⑥ Select the following options, and click Next:

Step 1: Product Category: All

Step 2: Device Family: Virtex4

Step 3: Device: XC4VLX15

Step 4: Package: SF363

Step 5: Speed Grade: -12

Step 6: Synthesis Tool: XST (VHDL/Verilog)

Step 7: Simulator: ISE Simulator (VHDL/Verilog)

Step 8: Enable Enhanced Design Summary: Checked

The Create New Source dialog appears (**Figure 1f-3**). You can use this dialog to create a new HDL source file by defining the module name and ports. All of the source files have been created for you in this project.

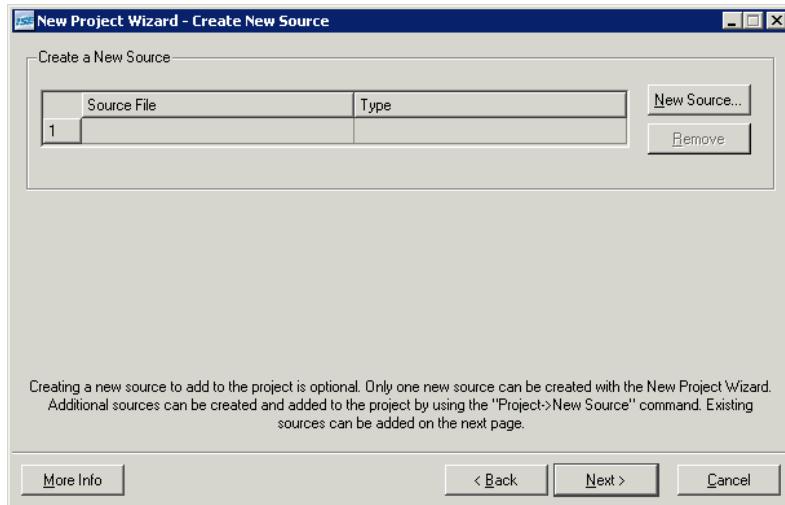


Figure 1f-3. Create New Source Dialog

⑦ Click Next

The Add Existing Sources dialog appears (**Figure 1f-4**).

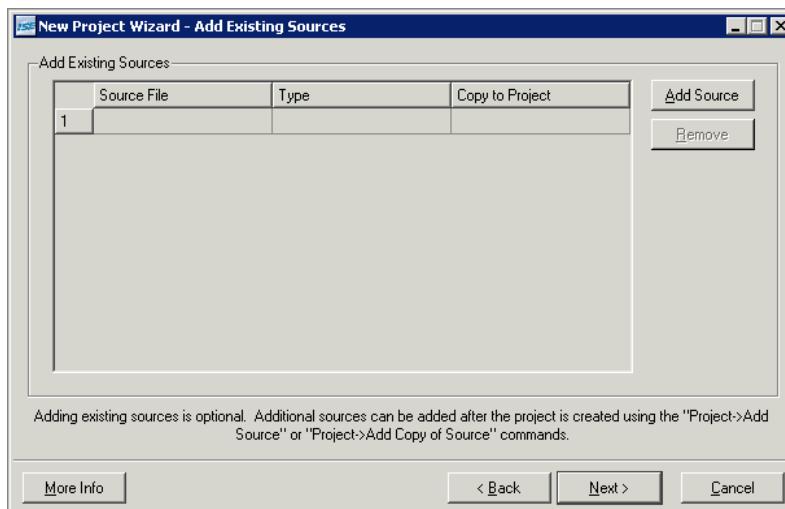
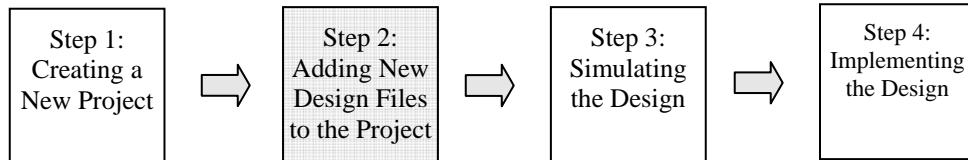


Figure 1f-4. Add Existing Sources Dialog

Adding New Design Files to the Project

Step 2

General Flow for this Lab:



Add HDL and UCF source files into the project.

- ① Click **Add Source** and browse to the *R:\training\fund\labs\flow* folder
- ② To select all of the files, click the **first file** and, while pressing the **Shift** key, click the **last file** as shown in **Figure 1f-5**. Click **Open**

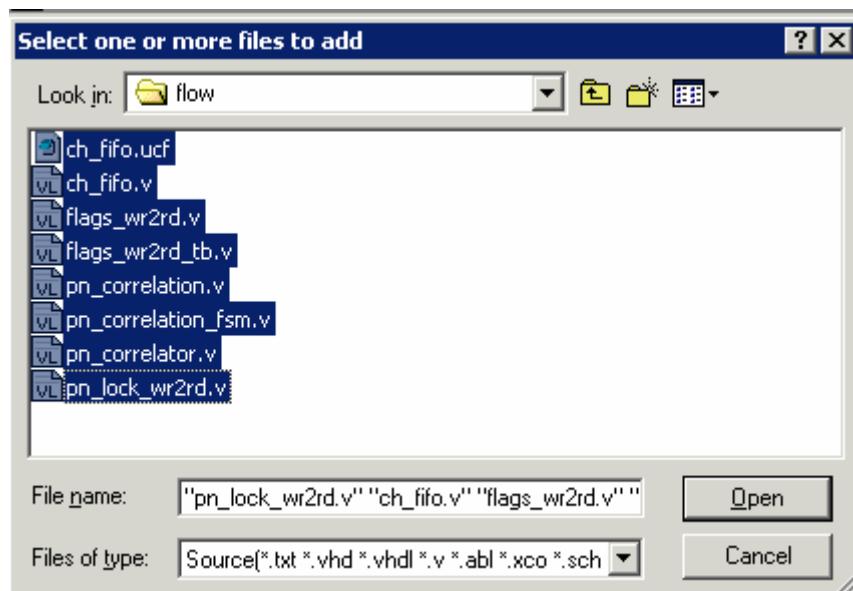


Figure 1f-5. Adding Existing Sources to Project

- ③ Click **Next**, and click **Finish**

- ④ In the Adding Source Files box, leave everything at their default value (**Figure 1f-6**). Click OK

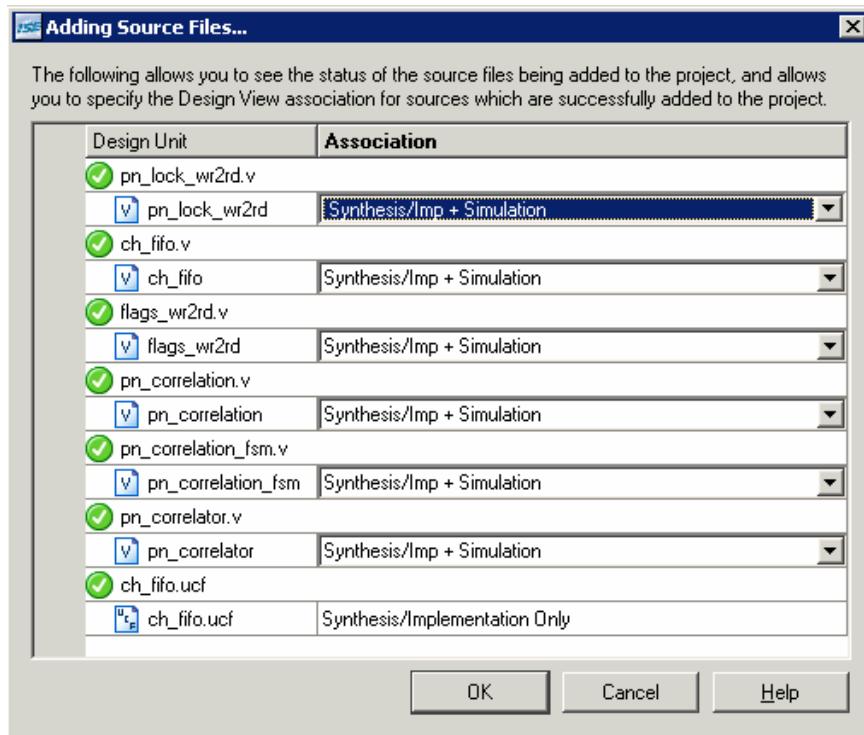


Figure 1f-6. Adding Source Files...Design Unit Association

The ISE™ software will process all of the files and then will determine the design hierarchy. The design files and hierarchy are reflected in the Sources for: window.

- ⑤ In the Sources for: window, expand *ch_fifo.v* to view the design hierarchy (**Figure 1f-7**)

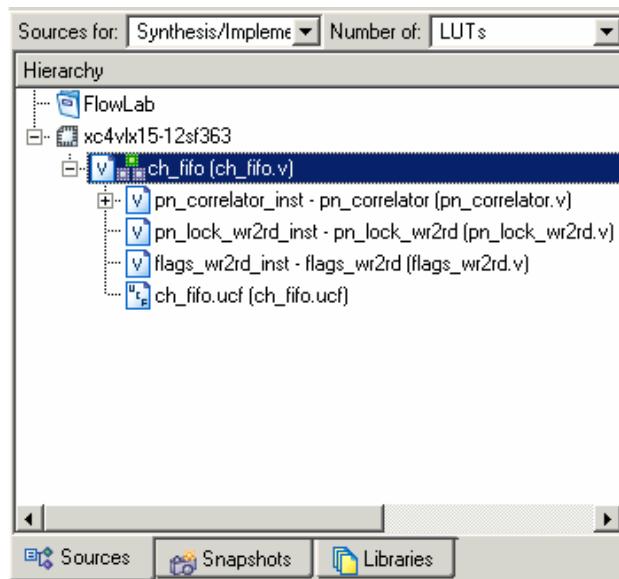


Figure 1f-7. Sources for: Window

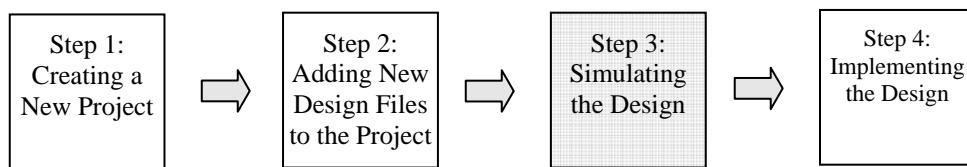


2. What information can you gather from the Sources window?
-
-

Simulating the Design

Step 3

General Flow for this Lab:



Open the testbench flags_wr2rd_tb and view the simple testbench stimulus. The block flags_wr2rd is used to cross clock domains from the slower wr_clk domain to the faster rd_clk domain. Then run a behavioral simulation using this testbench and examine the results.

- ① In the Sources for: window, click on **Behavioral Simulation** (**Figure 1f-8**). This will make the testbench associated with the design appear in the Sources for: window

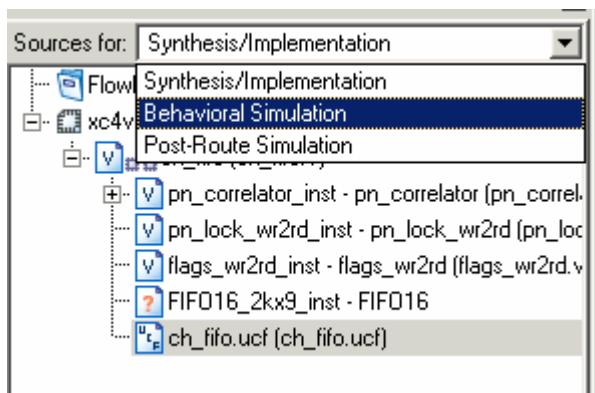


Figure 1f-8. Sources for: Behavioral Simulation

- ② In the Sources for: window, double-click **flags_wr2rd_tb**. Browse this simple testbench. If you have any questions, ask the instructor
- ③ Make sure that the testbench is still selected. In the Processes window, expand the **Xilinx ISE Simulator** toolbox, and double-click **Simulate Behavioral Model**

- ④ Click the **Simulation tab** to view the simulation results. Confirm the following:

Inputs fifo_empty, fifo_almost_empty, fifo_full, fifo_and almost_full are synchronous to wr_clk.

Outputs empty, almost_empty, full, and almost_full follow the pattern of their respective inputs, but synchronous to rd_clk

Zoom and pan to confirm that the module simulated correctly

- ⑤ Close the **simulator windows**. Click **Yes** to confirm that you want to end the simulation

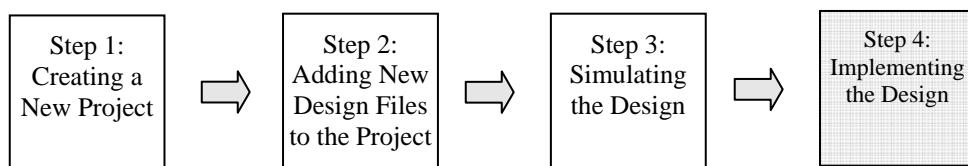


3. Why did we include running a simple simulation in this lab?
-
-

Implementing the Design

Step 4

General Flow for this Lab:



Implement the design. During implementation, some reports will be created. You will look more closely at some of these reports in the next module.

- ① In the Sources for: drop-down box, select **Synthesis/Implementation**
- ② Select the top-level design file **ch_fifo.v**

- ③ In the Processes window, expand **Implement Design** to view the underlying processes. Double-click **Implement Design** (Figure 1f-9)

Notice that the tools run all of the processes required to implement the design. In this case, the tools run Synthesis before going into Implementation.

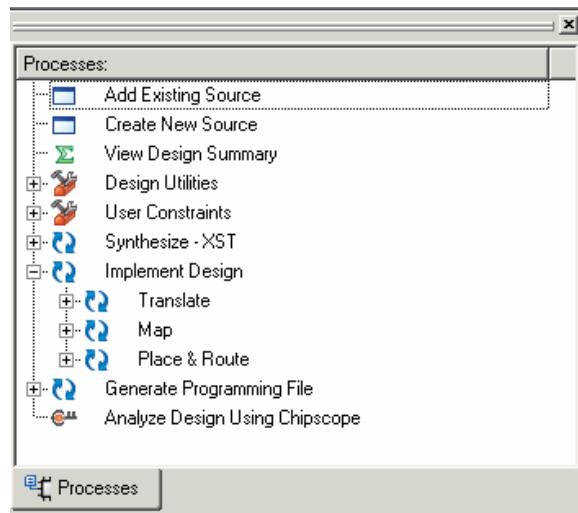


Figure 1f-9. Processes for Source Window

After each stage is complete, a symbol will appear next to each stage:

- ✓ Green check mark for successful
- ! Yellow exclamation point for warnings
- ✗ Red X for errors

For this particular design, there may be a yellow exclamation point (warnings) for some steps. The warnings here are okay to ignore.

- ④ Read some of the messages in the message window located across the bottom of the Project Navigator window

- 5 When implementation is complete, double-click **View Design Summary** in the Processes window (**Figure 1f-10**)

The screenshot shows the Xilinx ISE Tools interface. The top part is the 'Processes' window, which lists various design steps. The 'View Design Summary' option is highlighted. Below it is the main workspace displaying the 'FPGA Design Summary' report.

FPGA Design Summary

- Design Overview**
 - Summary
 - IOB Properties
 - Timing Constraints
 - Pinout Report
 - Clock Report
- Errors and Warnings**
 - Synthesis Messages
 - Translation Messages
 - Map Messages
 - Place and Route Messages
 - Timing Messages
 - Bitgen Messages
 - All Current Messages
- Detailed Reports**
 - Synthesis Report
 - Translation Report
 - Map Report
 - Place and Route Report
 - Static Timing Report
 - Bitgen Report
- Secondary Reports**
 - Xplorer Report

Project Properties

- Enable Enhanced Design Summary
- Enable Message Filtering
- Display Incremental Messages
- Enhanced Design Summary Contents
 - Show Partition Data
 - Show Errors
 - Show Warnings
 - Show Falling Constraints
 - Show Clock Report

FLOWLAB Project Status

Project File:	FlowLab.ise	Current State:	Placed and Routed
Module Name:	ch_fifo	• Errors:	No Errors
Target Device:	xc4vlx15-12sf363	• Warnings:	20 Warnings
Product Version:	ISE 8.2i	• Updated:	Wed Jun 21 10:16:31 2006

FLOWLAB Partition Summary

No partition information was found.

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	24	12,288	1%	
Number of 4 input LUTs	72	12,288	1%	
Logic Distribution				
Number of occupied Slices	41	6,144	1%	
Number of Slices containing only related logic	41	41	100%	
Number of Slices containing unrelated logic	0	41	0%	
Total Number of 4 input LUTs	72	12,288	1%	
Number of bonded IOBs	20	240	8%	
Number of BUF6/BUF6CTRLs	4	32	12%	
Number used as BUFGs	4			
Number used as BUFCTRLs	0			
Number of FIFO16/RAM16s	1	48	2%	
Number used as FIFO16s	1			
Number used as RAM16s	0			
Number of ISERDESs	2	320	1%	
Total equivalent gate count for design	66,198			
Additional JTAG gate count for IOBs	960			

Performance Summary

Final Timing Score:	0	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports

Figure 1f-10. Design Summary



4. What information can you gather from the Design Summary window?
-
-

- ⑥ Leave the Project Navigator open so that you can open the report files during the next module

Conclusion

In this demonstration, you completed the major stages of the ISE™ design flow: creating a project, adding source files, simulating the design, and implementing the design.

In the next module, you will examine some of the software reports and determine how the design was implemented and whether your design goals for area and performance were met.

A Answers

1. What is the purpose of creating a project in ISE™ Project Navigator? That is, what will the project include?

The project itself specifies the top-level design attributes. For example: target device, input design type, simulator, project location, synthesis tool, speed grade, package, and input design files.

2. What information can you gather from the Sources window?

It indicates the project name, target device, and the input files. Additionally, the input files are arranged hierarchically. Additionally, any missing or unknown blocks are listed with a question mark .

3. Why did we include running a simple simulation in this lab?

Behavioral simulation is an important element to include in the design process. The ISE software includes a simulator for this purpose.

4. What information can you gather from the Design Summary window?

Project Status, Device Utilization information, Performance Summary, and direct access to design reports.



Xilinx Tool Flow Lab

Review



NOTES

Review Questions

- What information is included in an ISE™ Project Navigator project?
- How do you implement the design?
- Where can you find information about the implementation?



NOTES

Answers

- What information is included in an ISE™ Project Navigator project?
 - Target device
 - Input files
 - Simulator
 - Synthesis tool
- How do you implement the design?
 - Double-click **Implement Design**
- Where can you find information about the implementation?
 - Design Summary



NOTES

Summary

- The ISE™ Project Navigator includes all the information and attributes required to implement the design
- To implement the design, double-click **Implement Design**
- Use default implementation options for a baseline implementation
- Simulation and synthesis tools are integrated into Project Navigator, providing a simplified design environment



NOTES



Architecture Wizard and PACE Demo/Lab

Introduction

© 2006 Xilinx, Inc. All Rights Reserved

NOTES

Objectives

After completing this lab, you will be able to:

- Use the Architecture Wizard to configure a DCM component
- Instantiate the DCM component into the design
- Use PACE to assign pin locations
- Implement the design and confirm that the pin assignments were used

NOTES

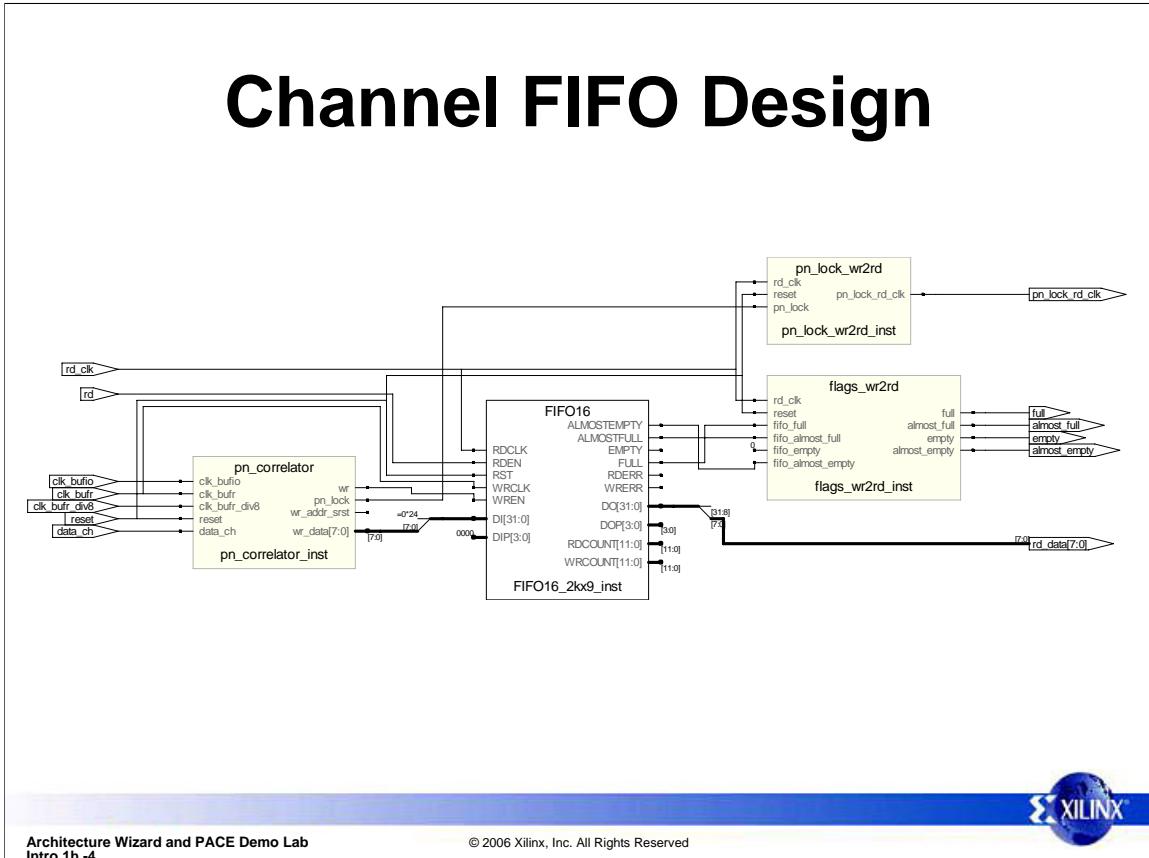
Introduction

- . This lab illustrates how to:
 - Use Architecture Wizard to create IP
 - Use Architecture Wizard to design clock resources
 - Customize a core
 - Instantiate an Architecture Wizard core
 - Use PACE for specifying pin assignments



NOTES

Channel FIFO Design



NOTES

This is a block diagram for the lab design, which you will use in this course.

The PN Correlator searches for a specific pattern (PN code) in the input data stream. Once the PN code is found, the following 255 bytes of data are stored in the FIFO. After the 255 bytes of data are stored, the correlator begins to search for the PN code again.

The FIFO Status block generates read/write addresses for the FIFO and creates flags that indicate how much data has been stored in the FIFO. These flags are used by the Data Control block to decide from which channel to read.

The FIFO block is a simple dual-port RAM that stores the data from the PN Correlator block.

For more information about the design, refer to the “Detailed Design Description for Labs” section of this course workbook or the HDL source code for this design.

General Flow

- Step 1: Use Architecture Wizard to configure clock resources
- Step 2: Instantiate clock core into the design
- Step 3: Use PACE to assign pin location constraints
- Step 4: Implement design

NOTES

Architecture Wizard and PACE Lab

Architecture Wizard and PACE Lab

Introduction

This lab introduces the Architecture Wizard and PACE.

Objectives

After completing this lab, you will be able to:

- Use the Architecture Wizard to configure a DCM component
- Instantiate the DCM component into the design
- Use PACE to assign pin locations
- Implement the design and confirm that the pin assignments were used

Procedure

This demonstration comprises four primary steps: You will use the Architecture Wizard to configure a DCM, instantiate the DCM into the design, use PACE to assign pin locations, and, finally, implement the design.

For each procedure within a primary step, there are general instructions (indicated by the  symbol). These general instructions only provide a broad outline for performing the procedure. Below these general instructions, you will find accompanying step-by-step directions and illustrated figures that provide more detail for performing the procedure. If you feel confident about completing a procedure, you can skip the step-by-step directions and move on to the next general instruction.

Note: If you are not using **Toolwire** to perform this demonstration, all software programs, files, and projects will be located on the C:\ drive instead of R:\.

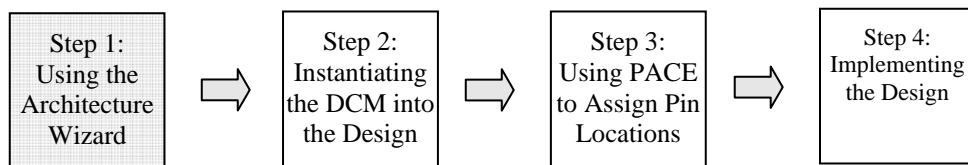
Note: If you wish to review this software demonstration at a later time, you can download the original files from the Xilinx FTP site at ftp://ftp.xilinx.com/pub/documentation/education/fpga13000-82-xlnx_lab_files.zip. These are the original lab files and do not contain any work you may have previously completed.

Using the Architecture Wizard

Step 1

For each procedure within a primary step, there are general instructions (indicated by the  symbol). These general instructions only provide a broad outline for performing the procedure. Below these general instructions, you will find accompanying step-by-step directions and illustrated figures that provide more detail for performing the procedure. If you feel confident about completing a procedure, you can skip the step-by-step directions and move on to the next general instruction.

General Flow for this Lab:



Open an existing project.

- ① If you have closed the ISE™ Project Navigator, select **Start → Programs → Xilinx ISE 8.2i → Project Navigator**
- ② Select **File → Open Project** in the Project Navigator
- ③ Browse to **R:\training\fund\labs\arwz_pace** and select **arwz_pace.ise**
- ④ Click **Open**



This version of the design is missing a DCM component. Use the Architecture Wizard to configure a DCM component to meet your design needs.



1. Why are you using Architecture Wizard to configure the DCM component? That is, why not just instantiate the DCM primitive?
-
-

- ① In the Processes window, double-click **Create New Source**

If you do not see the Create New Source process, ensure that an HDL source file is selected in the Sources in Project window.

- ② In the New Source window, select **IP (COREGen & Architecture Wizard)** and enter **my_dcm** as the file name
- ③ Click **Next**

- ④ In the Select IP window, expand **FPGA Features and Design**, expand **Clocking**, expand **Virtex-4**, and select **Single DCM ADV v8.2i** (Figure 1i-1)

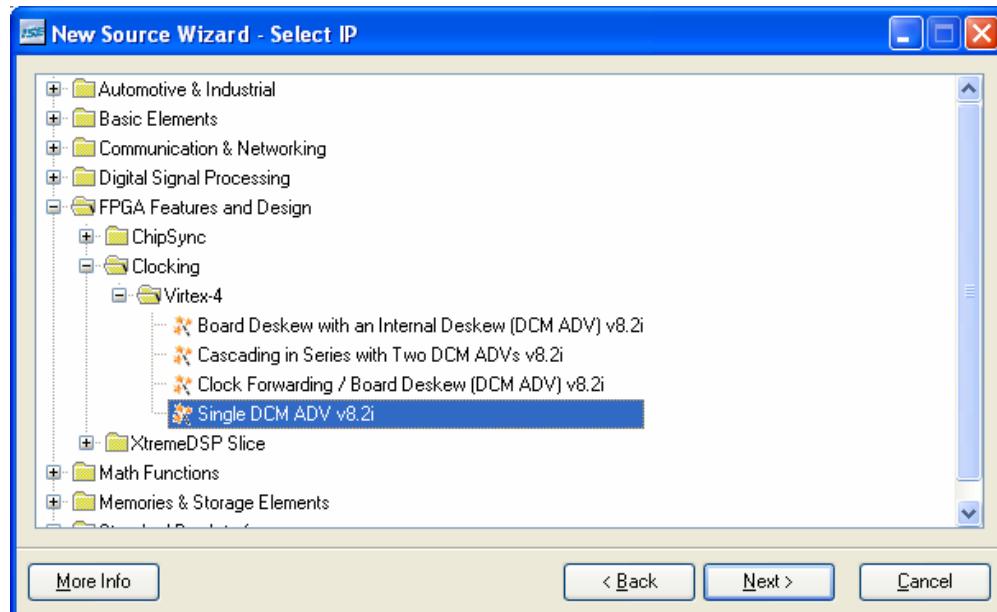


Figure 1i-1. Architecture Wizard Selection Box

- ⑤ Click **Next**, and click **Finish**

- ⑥ In the Xilinx Clocking Wizard – General Setup window, set the following options (as shown in **Figure 1i-2**):

- RST, CLK0, CLK2X, and LOCKED boxes: **Checked**
- Add PMCD after DCM: **Checked**
- Input Clock Frequency: **50 MHz**

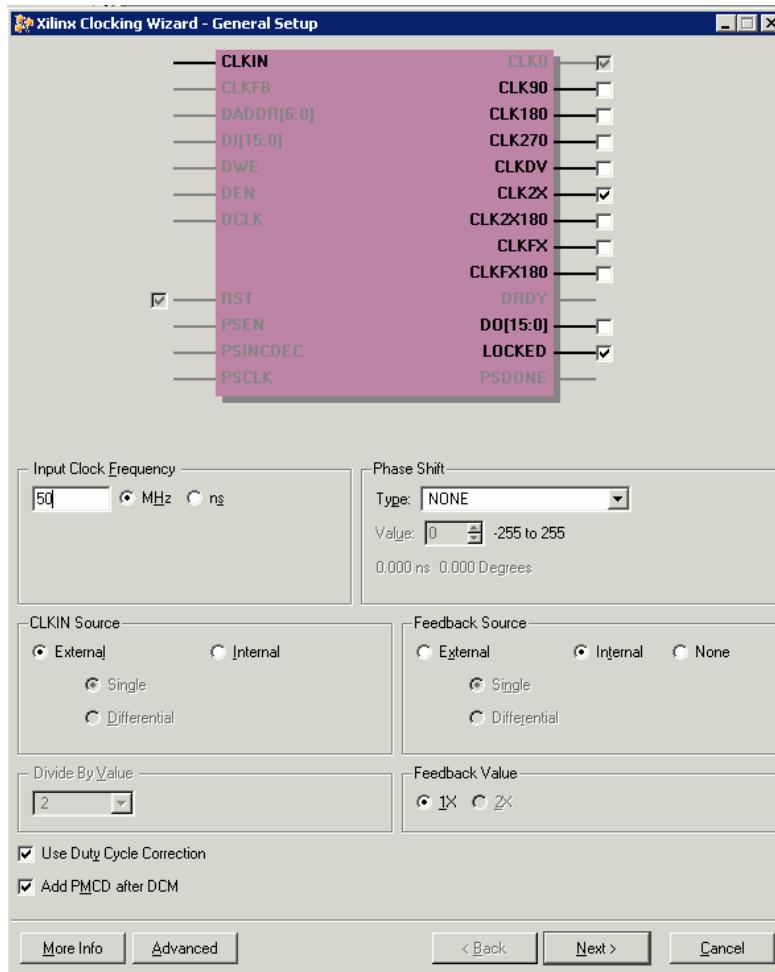


Figure 1i-2. Xilinx Clocking Wizard – General Setup Window

Note: Specifying a frequency in the Clocking Wizard is done so that the wizard knows whether the DCM is in High or Low frequency mode. Based on the input frequency, the wizard also calculates output frequencies to ensure these frequencies are within the specified range. In addition, if the CLKFX output is used, the input frequency is used to calculate the output frequency at the CLKFX and to calculate estimated jitter.

- ⑦ Click **Next**

- ③ To customize the PMCD, set the following (as shown in **Figure 1i-3**), and click **Next**:

- Single PMCD: **Selected**
- CLKA input: **CLK0**
- CLKA1D8: **Checked**
- CLKB input: **Checked**
- CLKB input: **CLK2X**

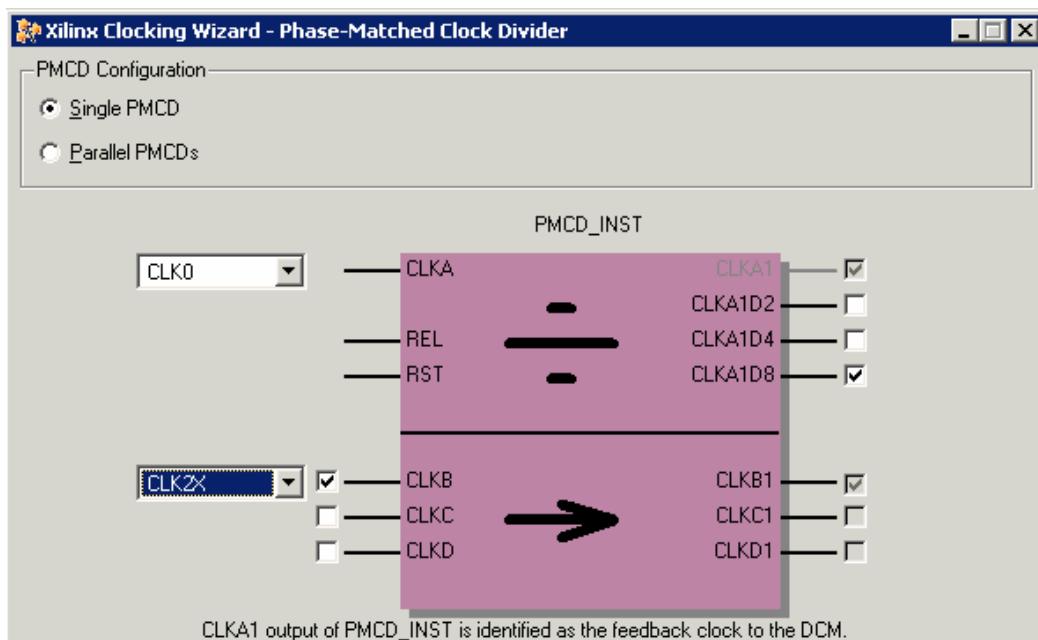


Figure 1i-3. Xilinx Clocking Wizard – Phase-Matched Clock Divider

- 9 In the Xilinx Clocking Wizard – Clock Buffers window (**Figure 1i-4**), keep the defaults and click **Next**, and click **Finish**

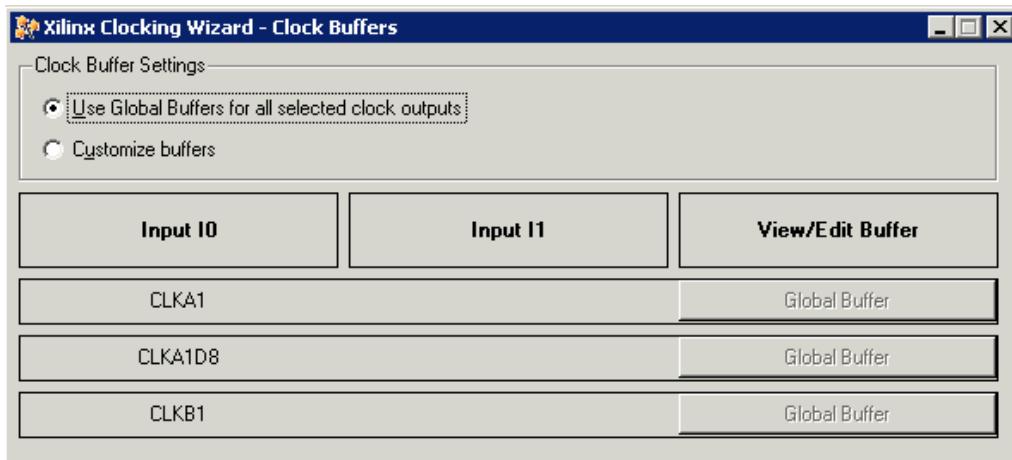


Figure 1i-4. Xilinx Clocking Wizard – Clock Buffers Window

Notice that a new file is added to the Sources in Project window (*my_dcm.xaw*). This source file will not be included in the design hierarchy until the component has been instantiated into one of the HDL source files.

- 10 With the *my_dcm.xaw* file selected, go to the **Processes** window, and double-click **View HDL Source** to examine the source code generated by the Architecture Wizard



If the file does not appear in the text editor, double-click **View HDL Source** again.

This file contains the following instantiated components: an IBUFG, a DCM, a PMCD, and three BUFGs.

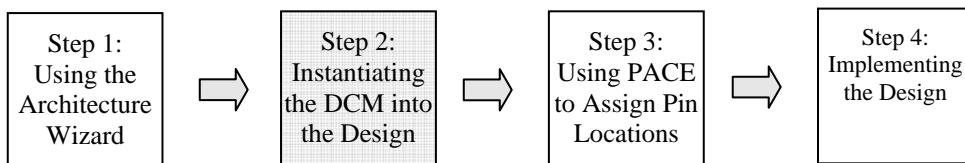
The input clock *CLKIN_IN* drives the IBUFG, which is connected to the DCM. The three output clocks are driven onto BUFG components.

All of the DCM attributes are passed through VHDL generics. In Verilog, meta-comments attach the attributes to the DCM.

Instantiating the DCM into the Design

Step 2

General Flow for this Lab:





Now that you have created the necessary files, you can instantiate the DCM component into your design. Copy and paste the text from the Instantiation Template into *ch_fifo.vhd* and connect the signals.



2. The core was created for us, so is the DCM already connected to the design? Why must the DCM core be instantiated into this design?
-
-

- ① In the Sources in Project window, double-click *ch_fifo.v* to open the source code in the text editor
- ② Select *my_dcm.xaw* in the Sources in Project window
- ③ In the Processes for Source window, double-click **View HDL Instantiation Template** to open the instantiation template in the text editor



If the template does not appear in the text editor, double-click **View HDL Instantiation Template** again.

- ④ In the Instantiation Template, copy the **component instance (begin at my_dcm and end after the close parenthesis)** and paste into *ch_fifo.v* under the comment *// Instantiate the DCM module here*
- ⑤ Complete the instantiation by filling in the port connections as follows:

```
my_dcm instance_name (
    .CLKIN_IN(wr_clk),
    .RST_IN(reset),
    .CLKIN_IBUFG_OUT(),
    .CLK0_CLKA1D8_OUT(wr_clk_div8_bufg),
    .CLK0_CLKA1_OUT(wr_clk_bufg),
    .CLK2X_CLKB1_OUT(rd_clk_bufg),
    .LOCKED_OUT(dcm_lock)
);
```

Note: The *clkin_ibufg_out* port is an output that is present to support designs that use the RocketIO™ transceivers. We will leave it unconnected here.

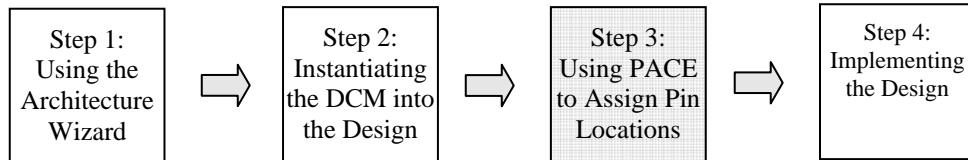
- ⑥ Click **File → Save** to save the file

Notice that the source file *my_dcm.xaw* is now inserted into the correct place in the design hierarchy.

Using PACE to Assign Pin Locations

Step 3

General Flow for this Lab:



You have completed the design by adding a DCM.

In this demonstration, you will only make a few pin assignments. Usually, you will use PACE to assign all of the pins in your design.



3. Why are you making pin assignments? Why not simply let the implementation tools make those assignments?

-
-
- ➊ In the Sources for: window, select the top-level design file **ch_fifo.v**
 - ➋ In the Processes window, expand **User Constraints**, and double-click **Assign Package Pins** to open PACE
 - ➌ In the Design Object List window, scroll down until you see the *rd_data** signals
 - ➍ In the Architecture window, zoom into the upper-left corner of the die (so you can view the pin numbers)

- 5 Drag and drop the I/O signal from the Design Object List window to the following locations in the Device Architecture window (**Figure 1i-5**):

- rd_data<0>: **C17**
- rd_data<1>: **B17**
- rd_data<2>: **C16**
- rd_data<3>: **C15**
- rd_data<4>: **B16**
- rd_data<5>: **A16**
- rd_data<6>: **A15**
- rd_data<7>: **B15**

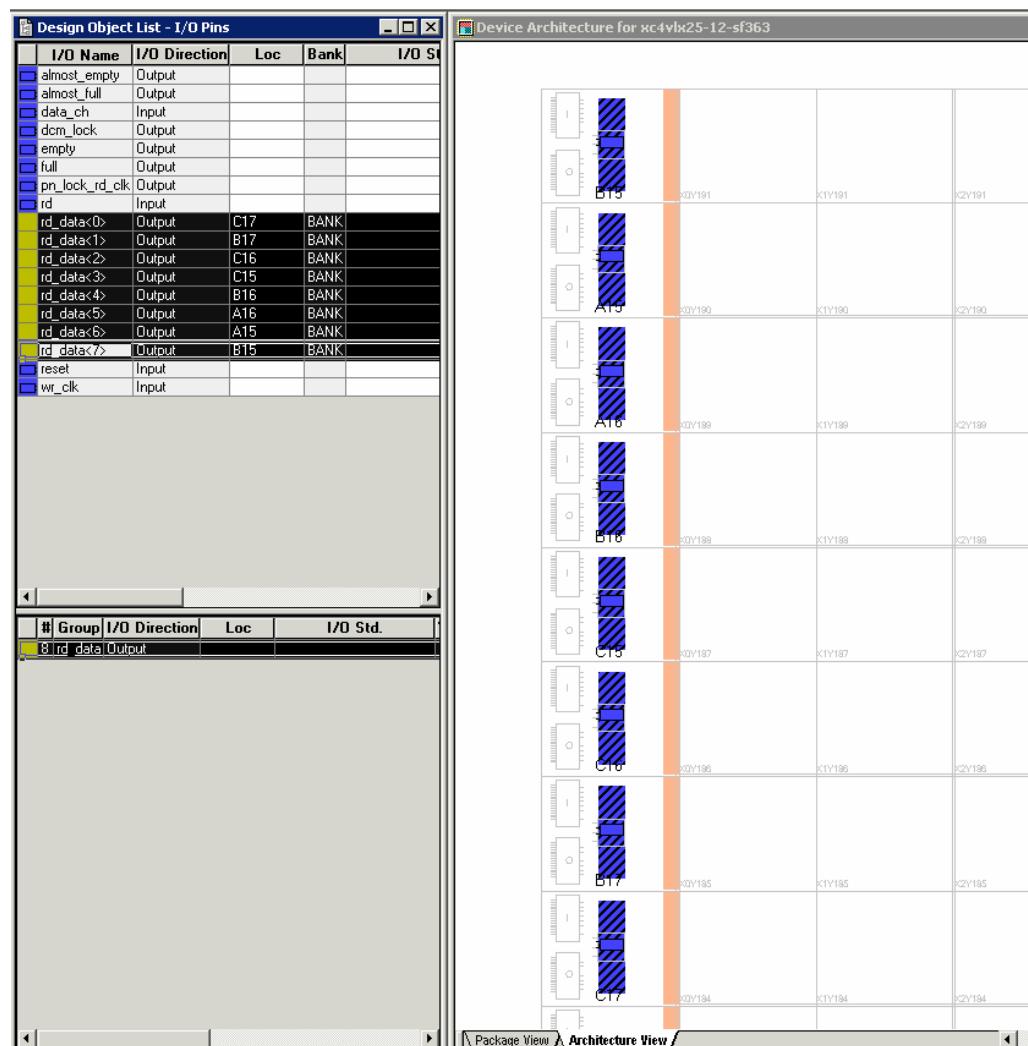


Figure 1i-5. Design Object List – I/O Pins Window



4. Why did we make the pin placements in that order on the die?



View your pin assignments in relation to the package.

- ① At the bottom of the Device Architecture window, select the **Package View** tab

Note the location of the pins in the upper-right corner of the die.



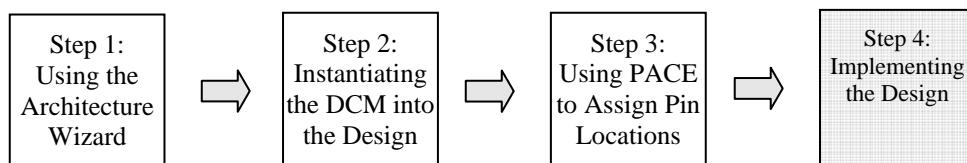
- ⑤ From this view (Package View), can you tell whether all the *rd_data* signals are in the same bank?
-
-

- ② Click **each colored I/O pin**. The corresponding pin will be selected in the Design Object List window
- ③ Click **File → Save** to save these pin placements
- ④ In the dialog, select **XST Default: <>** as the I/O Bus Delimiter, and click **OK**
- ⑤ Click **File → Exit** to close PACE
- ⑥ In the Processes for Source window of the Project Navigator, expand **User Constraints**, and double-click **Edit Constraints (Text)** to view the constraints created in the *ch_fifo.ucf* file through PACE. View the text version of the UCF file to confirm that the constraints were written to the file

Implementing the Design

Step 4

General Flow for this Lab:



Implement the design and review the PAD report to confirm that the pin assignments were obeyed.

- ① Make sure that the top-level design file *ch_fifo.v* is selected in the Sources for: window
- ② In the Processes window, double-click **Implement Design**
- ③ In the Processes window, double-click **View Design Summary**
- ④ In the FPGA Design Summary pane on the left, click **Pinout Report**
- ⑤ In the Pinout Report, click the **Signal Name column header** to view the design signals



6. Do the pin numbers for the *rd_data* bits match your specifications in PACE?
-

Conclusion

In this demonstration, you used the Architecture Wizard to configure a DCM component, and you instantiated the component into the design. You used PACE to make pin assignments. Finally, you implemented the design and examined the PAD report to confirm that the correct pin assignments were used.

 **Answers**

1. Why are we using Architecture Wizard to configure the DCM component? That is, why not just instantiate the DCM primitive?

The Architecture Wizard provides an easy to use interface for configuring the DCM attributes. Additionally, it can be used to configure its use with a PMCD and global clock buffers. All of these will come in a single core for instantiation.

2. The core was created for us, so is the DCM already connected to the design? Why must the DCM core be instantiated into this design?

No, it is not automatically inserted into the design. Therefore, you must instantiate it into the design to use that particular architecture resource.

3. Why are you making pin assignments? Why not simply let the implementation tools make those assignments?

Most FPGA designs have pinout requirements before the design is complete. PACE makes it easy to assign pins and to check that the chosen pinout conforms to all DRC rules regarding I/O standards and I/O banks. Your design will have specific board level requirements that will usually dictate many of the pin location constraints. Allowing the implementation tools to perform this function is unlikely to produce results that are compatible with your design requirements.

4. Why did we make the pin placements in that order on the die?

Following bit ordering creates an optimal alignment of bits for the internal fabric as suggested in the presentation. Note that you could still have inserted other signals between these bits.

5. From this view (Package View), can you tell whether all the *rd_data* signals are in the same bank?

Yes. The banks are color coded, and all *rd_data* bits have the same color bank.

6. Do the pin numbers for the *rd_data* bits match your specifications in PACE?

Yes.



Architecture Wizard and PACE Demo/Lab

Review

© 2006 Xilinx, Inc. All Rights Reserved

NOTES

Review Questions

- What are the advantages of using Architecture Wizard to create a clocking core?
- What are some advantages of using PACE for making pin assignments?



NOTES

Answers

- What are the advantages of using Architecture Wizard to create a clocking core?
 - Easy to customize and specify attributes in graphical interface
 - It creates a simple core for instantiation that can include IBUFG, DCM, PMCD, and BUFGCTRL components
 - Saves time
- What are some advantages of using PACE for making pin assignments?
 - Color coded banks
 - Die location easily identified
 - Design rule checker
 - Simultaneous switching output check



NOTES

Summary

- Architecture Wizard provides a simple graphical user interface for specifying attributes and core customization
- PACE provides a graphical user interface with color coding, key, and availability to all I/O attributes
- PACE includes design rule checker to verify that your constraints comply with the architecture's capabilities



NOTES



Reading Reports

©2006 Xilinx, Inc. All Rights Reserved

NOTES

This Reading Reports module will show you how to determine whether a design met your area goals and performance goals. You should have the ISE™ Project Navigator open from the Xilinx Tool Flow Demo/Lab so you can open the report files you created during that lab.

Objectives

After completing this module, you will be able to:

- Determine whether a design met your area goals
- Determine whether a design met your performance goals



NOTES

Outline



- Introduction
- Area Goals
- Performance Goals
- Summary

NOTES

Introduction

- After you have implemented your design, how can you tell whether the implementation was successful?
- First and foremost, how do you define a successful design?
- Answer: A successful design:
 - Fits into the device
 - Achieves performance goals (meets your Timing Constraints)



NOTES

Outline

- Introduction
- • **Area Goals**
- Performance Goals
- Summary



NOTES

Determining Whether Area Goals Were Met

The screenshot shows the Xilinx ISE Design Suite interface. The main window displays the 'Project Manager' with a tree view of source files and a 'Design Summary' tab selected. To the right, the 'Device Utilization Summary (Estimated values)' is shown, listing logic utilization details. Below the main window is a 'Detailed Reports' table.

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Wed Jun 21 10:15:12 2006	0	19 Warnings	10 Infos
Translation Report	Current	Wed Jun 21 10:15:24 2006	0	0	0
Map Report	Current	Wed Jun 21 10:15:35 2006	0	1 Warning	4 Infos
Place and Route Report	Current	Wed Jun 21 10:16:19 2006	0	0	0
Static Timing Report	Current	Wed Jun 21 10:16:29 2006	0	0	1 Info
Bitgen Report					

Reading Reports 2a - 6 © 2006 Xilinx, Inc. All Rights Reserved

NOTES

How do you know whether the design fits into the device? Information can be found in the Design Summary, Map Report, or the Place & Route Report. How do you know whether there is enough room in the device for more logic? If there is enough room, exactly how much space is available? Information can be found in the Design Summary, Map Report, or the Place & Route Report. If the design fits into the device, was it able to route completely? Information can be found in the Place & Route Report.

Remember that you can review the Synthesis Report to determine whether your area goals can be met as well.

Reviewing the Map Report

- . Map Report
 - <design>.mrp

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Wed Jun 21 10:15:12 2006	0	19 Warnings	10 Infos
Translation Report	Current	Wed Jun 21 10:15:24 2006	0	0	0
Map Report	Current	Wed Jun 21 10:15:35 2006	0	1 Warning	4 Infos
Place and Route Report	Current	Wed Jun 21 10:16:19 2006	0	0	0
Static Timing Report	Current	Wed Jun 21 10:16:29 2006	0	0	1 Info
Bitgen Report					

Reading Reports 2a - 7

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

Let's take a look at the Map Report contents.

Read the Map Report to See How Your Design Will Fit into the FPGA

Table of Contents

-
- Section 1 - Errors
- Section 2 - Warnings
- Section 3 - Informational
- Section 4 - Removed Logic Summary
- Section 5 - Removed Logic
- Section 6 - IOB Properties
- Section 7 - RPMs
- Section 8 - Guide Report
- Section 9 - Area Group Summary
- Section 10 - Modular Design Summary
- Section 11 - Timing Report
- Section 12 - Configuration String Information
- Section 13 - Additional Device Resource Counts



NOTES

In the Map report you will find the command line options for the map program. You will see the design summary and a list of the number of device resources required. Errors and warnings are shown here. Section 4 has a Removed Logic Summary with a list of logic that was removed due to sourceless or loadless nets. The IOB properties indicate whether an I/O flip-flop is used and provides a list of attributes on each I/O pin.

It is useful to review the Map report before proceeding to Place & Route, especially for large or high-speed designs. You want to ensure the device has enough resources. Review this report before or during Place & Route. The Map Report is very useful because it is your first chance to see how easily your design will fit into the target FPGA. In general, designs that use more of the resources of the chip will take longer to Place & Route. If the report indicates that your design may not meet your design goals, you can stop the implementation process immediately by using the menu command **Process → Stop**.

Reviewing the Map Report

- How many slices were used (Number of Occupied Slices)?
- How many IOBs were used?
- Were there any errors, warnings, or informational messages?



NOTES

Refer to the Map report you created in the Xilinx Tool Flow Demo/Lab to see how many slices were used, how many IOBs were used, and whether there were any errors, warnings, or informational messages.

Reviewing the Map Report

- . How many slices were used (Number of Occupied Slices)?
 - 41
- . How many IOBs were used?
 - 20
- . Were there any errors, warnings, or informational messages?
 - 0 errors
 - 1 warning
 - 4 informational messages



NOTES

Reviewing the Place & Route Report

<design>.par

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Wed Jun 21 10:15:12 2006	0	19 Warnings	10 Infos
Translation Report	Current	Wed Jun 21 10:15:24 2006	0	0	0
Map Report	Current	Wed Jun 21 10:15:35 2006	0	1 Warning	4 Infos
Place and Route Report	Current	Wed Jun 21 10:16:19 2006	0	0	0
Static Timing Report	Current	Wed Jun 21 10:16:29 2006	0	0	1 Info
Bitgen Report					

Reading Reports 2a - 11

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

Now, let's take a look at the Place & Route Report contents.

Read the Place & Route Report: Errors, Warnings, Timing Summary

```
par -w -intstyle ise -ol std -t 1 ch_fifo_map.ncd ch_fifo.ncd ch_fifo.pcf
Device Utilization Summary:

  Number of BUFGs           4 out of 32    12%
  Number of FIFO16s         1 out of 48    2%
  Number of External IOBs   20 out of 240   8%
    Number of LOCed IOBs    0 out of 20    0%
  Number of ISERDESS        2 out of 320   1%
  Number of OLOGICs         4 out of 320   1%
  Number of Slices          41 out of 6144  1%
    Number of SLICEMs        0 out of 3072  0%

Generating "PAR" statistics.

*****
Generating Clock Report
*****
```



NOTES

In PAR Report we can find the...command line options for the PAR program, the device utilization summary (which is similar to the Design Summary in the Map Report), and a list of any unrouted nets. The timing summary provides statistics on average routing delays and performance versus constraints if the design contains timing constraints.

Reviewing the Place & Route Report

Device Utilization Summary:

Number of BUFGs	4 out of 32	12%
Number of FIFO16s	1 out of 48	2%
Number of External IOBs	20 out of 240	8%
Number of LOCed IOBs	0 out of 20	0%
Number of ISERDESSs	2 out of 320	1%
Number of OLOGICs	4 out of 320	1%
Number of Slices	41 out of 6144	1%
Number of SLICEMs	0 out of 3072	0%

- Does the Device Utilization Summary agree with the Map Report?
- Any unrouted signals?



NOTES

Refer to the Place & Route Report you created in the Xilinx Tool Flow demo.

Reviewing the Place & Route Report

- Does the Device Utilization Summary agree with the Map Report?
 - Yes
- Any unrouted signals?
 - No



NOTES

Outline

- Introduction
 - Area Goals
 - **Performance Goals**
 - Summary
- 



NOTES

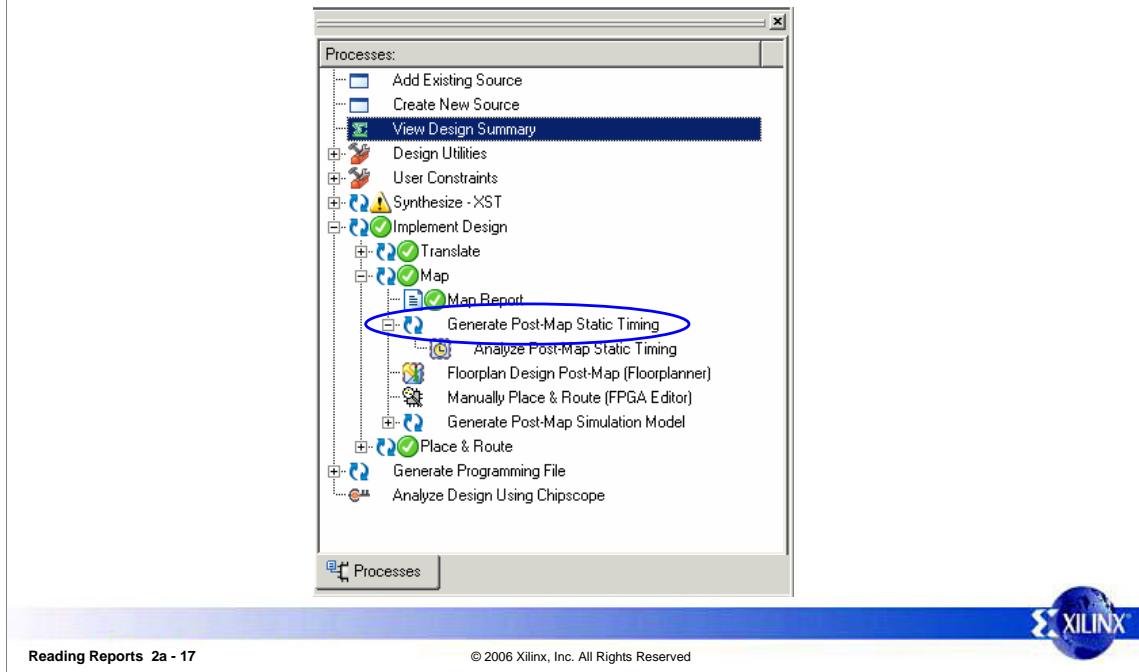
Are Your Timing Constraints Realistic?



NOTES

Now that you have entered constraints, how do you know whether the constraints were met?

Determining Whether Performance Goals Are Realistic



NOTES

Where would you look to find reasonable values for your timing constraints?

The Post-Map Static Timing Report Indicates Reasonable Constraints

Processes:

- Add Existing Source
- Create New Source
- View Design Summary
- Design Utilities
- User Constraints
- Synthesize -XST
- Implement Design
- Translate
- Map
- Map Report
- Generate Post-Map Static Timing
- Analyze Post-Map Static Timing
- Floorplan Design Post-Map (Floorplanner)
- Manually Place & Route (FPGA Editor)

Slack: 9.207ns (requirement - (data path - clock path skew + uncertainty))

Source: flags wr2rd inst/almost full p0 (FF)
Destination: flags wr2rd inst/almost full (FF)

Requirement: 10.000ns

Data Path Delay: 0.793ns (Levels of Logic = 0)

Clock Path Skew: 0.000ns

Source Clock: rd_clk_BUFGP rising at 0.000ns

Destination Clock: rd_clk_BUFGP rising at 10.000ns

Clock Uncertainty: 0.000ns

Data Path: flags wr2rd inst/almost full p0 to flags wr2rd inst/almost full

Delay type	Delay(ns)	Logical Resource(s)
Tck0	0.258	flags wr2rd inst/almost full p0
net (fanout=1)	e 0.100	flags wr2rd inst/almost full p0
Tdock	0.435	flags wr2rd inst/almost full
Total	0.793ns	(0.693ns logic, 0.100ns route) (87.4% logic, 12.6% route)

XILINX

Reading Reports 2a - 18

© 2006 Xilinx, Inc. All Rights Reserved

NOTES

You would look at the Post-Map Static Timing Report. The Post-Map Static Timing Report indicates whether your constraints are reasonable. It contains actual block delays and **0.1-ns net delays**. Note that the net in this report has an “e” next to it. This denotes that it is an estimated net delay.

What is reasonable? Use the 60/40 rule, which states that if less than 60 percent of the timing budget is used for logic delays, the Place & Route tools should be able to meet the constraint easily. If between 60 to 80 percent of the timing budget is used for logic delays, the software run time will increase. And if greater than 80 percent of the timing budget is used for logic delays, the tools may have trouble meeting your goals

Evaluate and Handle Unreasonable Constraints

Slack:	0.828ns (requirement - (data path - clock path skew + uncertainty))
Source:	pn_correlator_inst/pn_correlation_fsm_inst/pn_addr_cntr_1 (FF)
Destination:	pn_correlator_inst/pn_correlation_fsm_inst/pn_addr_cntr_9 (FF)
Requirement:	3.000ns
Data Path Delay:	2.172ns (Levels of Logic = 3)
Clock Path Skew:	0.000ns
Source Clock:	clk_bufr_BUFGP rising at 0.000ns
Destination Clock:	clk_bufr_BUFGP rising at 20.000ns
Clock Uncertainty:	0.000ns
Data Path: pn_correlator_inst/pn_correlation_fsm_inst/pn_addr_cntr_1 to pn_correlator_inst/pn_correlation_fsm_inst/pn_addr_cntr_9	
Delay type	Delay(ns) Logical Resource(s)

Tcko	0.258 pn_correlator_inst/pn_correlation_fsm_inst/pn_addr_cntr_1
net (fanout=8)	e 0.100 pn_correlator_inst/pn_correlation_fsm_inst/pn_addr_cntr<1>
Tilo	0.146 pn_correlator_inst/pn_correlation_fsm_inst/ n002312
net (fanout=3)	e 0.100 pn_correlator_inst/pn_correlation_fsm_inst/ n0023_map1085
Ti5x	0.428 pn_correlator_inst/pn_correlation_fsm_inst/ n0027_1.G pn_correlator_inst/pn_correlation_fsm_inst/ n0027_1
net (fanout=2)	e 0.100 pn_correlator_inst/pn_correlation_fsm_inst/ n00271
Tilo	0.146 pn_correlator_inst/pn_correlation_fsm_inst/ n0005<9>76
net (fanout=1)	e 0.100 pn_correlator_inst/pn_correlation_fsm_inst/ n0005<9> map1059
Tsck	0.794 pn_correlator_inst/pn_correlation_fsm_inst/pn_addr_cntr_9

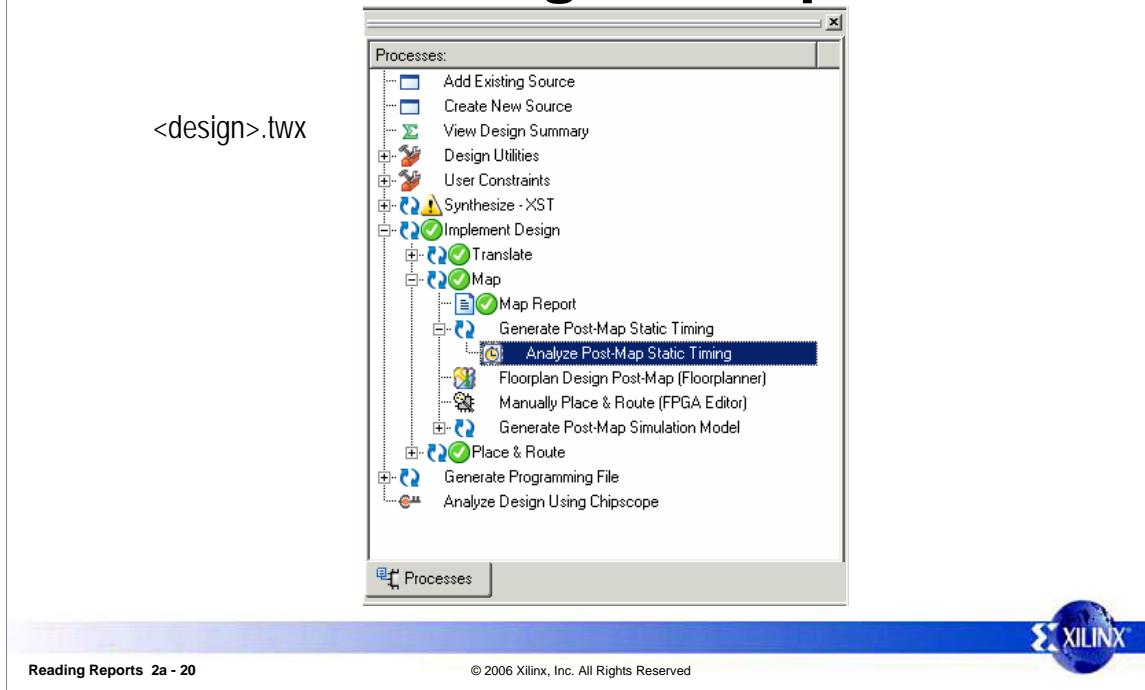
Total	2.172ns (1.772ns logic, 0.400ns route) (81.6% logic, 18.4% route)



NOTES

Evaluate the constraint. Is the design over-constrained? If it is, consider using advanced constraints. If it is not, redesign the logic or pipeline the path to reduce the number of logic levels in long paths. *Over-constrained* means that some paths are being constrained to a value that is tighter than necessary. For example, a multicycle path between two flip-flops that is covered by the global PERIOD constraint would be an over-constrained path.

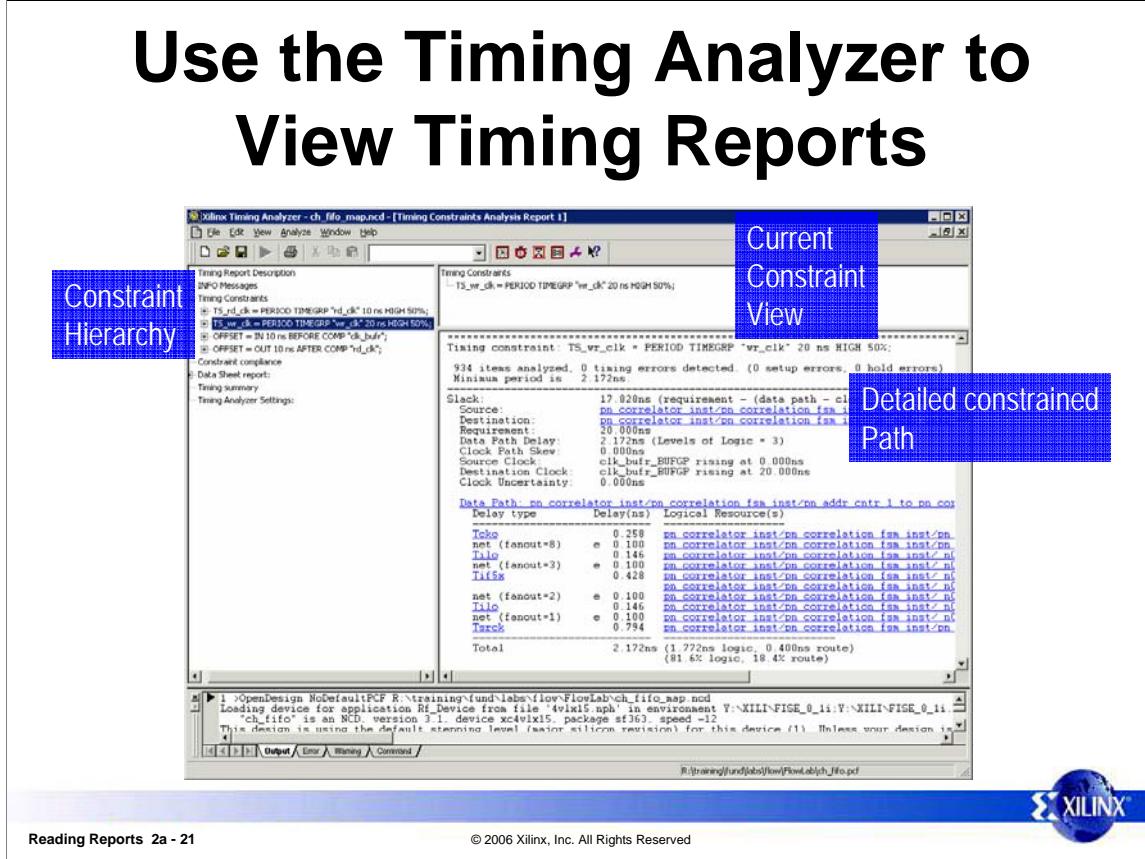
The Post-Map Static Timing Report is Created During the Map Process



NOTES

The Post-Map Static Timing Report is created during the Map process. It is not created by default. Double-click the process to create this report. Double-click the report to view it in the Timing Analyzer.

Use the Timing Analyzer to View Timing Reports



NOTES

Use the Timing Analyzer to view Timing Reports. Using the Timing Analyzer to create custom reports is covered in the *Designing for Performance* course. Note that you can create Timing Reports for designs without constraints. Xilinx, however, recommends using timing constraints to meet performance goals.

Use the Process Properties dialog to choose formatting options for each timing report. The example reports in this section were created by using the default options. You can generate timing reports even if your design does not contain any constraints. Instead of analyzing each constraint, the reports will analyze each clock signal for maximum frequency and the longest net delay. For more control over the timing reports, use the Timing Analyzer to create customized reports. The format of customized reports is similar to the reports discussed in this section.

Reading Timing Reports

The screenshot shows the "Timing Report Description" window from Xilinx ISE. The left sidebar lists navigation options: INFO Messages, Timing Constraints, Constraint compliance, Data Sheet report, Timing summary, and Timing Analyzer Settings. The main pane displays timing constraint details and analysis results.

Timing constraint: TS_wr_clk = PERIOD TIMEGRP "wr_clk" 20 ns HIGH 50%; 934 items analyzed. 0 timing errors detected. (0 setup errors, 0 hold errors) Minimum period is 2.172ns.

Slack: 17.828ns (requirement - (data path - clock path skew + t_{pd}))

Source:	pn correlator inst/pn correlation fsm inst/pn addr cntr
Destination:	pn correlator inst/pn correlation fsm inst/pn addr cntr
Requirement:	20.000ns
Data Path Delay:	2.172ns (Levels of Logic = 3)
Clock Path Skew:	0.000ns
Source Clock:	clk_bufr_BUFGP rising at 0.000ns
Destination Clock:	clk_bufr_BUFGP rising at 20.000ns
Clock Uncertainty:	0.000ns

Data Path: pn correlator inst/pn correlation fsm inst/pn addr cntr 1 to pn correlator inst/pn correlation fsm inst/pn addr cntr 2

Delay type	Delay(ns)	Logical Resource(s)
T _{ck0}	0.258	pn correlator inst/pn correlation fsm inst/pn
net (fanout=8)	e 0.100	pn correlator inst/pn correlation fsm inst/pn
T _{cl0}	0.146	pn correlator inst/pn correlation fsm inst/pn
net (fanout=3)	e 0.100	pn correlator inst/pn correlation fsm inst/pn
T _{if5x}	0.428	pn correlator inst/pn correlation fsm inst/pn

Timing summary:

Timing errors: 0 Score: 0

Constraints cover 973 paths, 0 nets, and 418 connections

NOTES

The contents of the Timing Report include the command line options for the trce program. The Timing Constraints section includes a summary of each timing constraint as well as details on paths that fail to meet constraints.

The Data Sheet section provides information on setup and hold, clock-to-pad, timing between clock domains, and pad-to-pad delay. It is organized in an easy-to-read table format. The Timing Summary section provides the number of errors and Timing Score.

The most important sections of a timing report are the *detailed descriptions of paths that fail to meet constraints* and the *Timing Score* (located at the end of the report). The detailed descriptions indicate why a particular path is failing to meet its constraint. You can use this information to determine how to reduce the path delay (changing software options for implementation or synthesis or both, adjusting the constraint, redesigning the logic, etc.). The Timing Score is the total number of picoseconds by which all constraints were missed. If all constraints are met, the Timing Score will be 0. If some constraints are missed, the Timing Score provides an indication of how far the design is from meeting all constraints.

Were the Constraints Met?



NOTES

Now that you have entered constraints, how do you know whether the constraints were met?

Post-Place & Route Static Timing Report: Were Constraints Met?

<design>.twx

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Wed Jun 21 10:15:12 2006	0	19 Warnings	10 Infos
Translation Report	Current	Wed Jun 21 10:15:24 2006	0	0	0
Map Report	Current	Wed Jun 21 10:15:35 2006	0	1 Warning	4 Infos
Place and Route Report	Current	Wed Jun 21 10:16:19 2006	0	0	0
Static Timing Report	Current	Wed Jun 21 10:16:29 2006			
Bitgen Report					

Processes:

- > Implement Design
 - +> Translate
 - +> Map
 - > Place & Route
 - +> Place & Route Report
 - +> Clock Region Report
 - +> Asynchronous Delay Report
 - +> Pad Report
 - +> Guide Results Report
 - +> MPPR Results Utilities
 - > Generate Post-Place & Route Static Timing
 - +> Analyze Post-Place & Route Static Timing
 - +> Generate Primetime Netlist

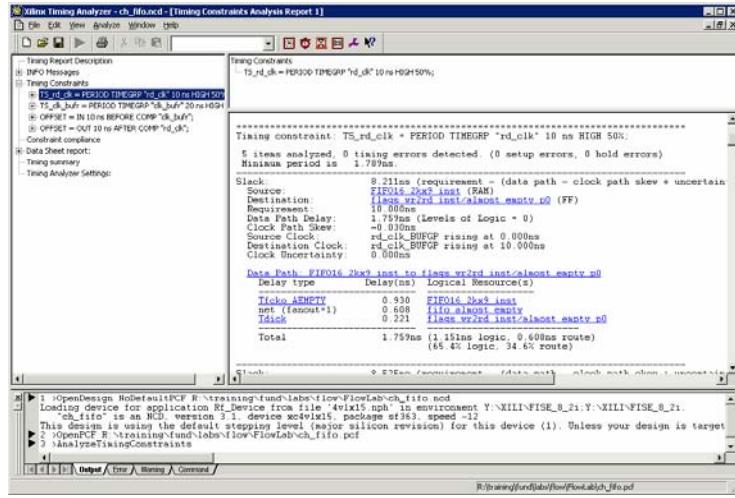
XILINX

Reading Reports 2a - 24 © 2006 Xilinx, Inc. All Rights Reserved

NOTES

You can review the Place & Route Report or Post-Place & Route Static Timing Report to determine whether the constraints were met. The Post-Place & Route Static Timing Report indicates whether your constraints were actually met. It is created by default and contains actual block delays and actual net delays calculated from Place & Route. It is used for static timing analysis after Place & Route. If you open the Static Timing Report from the Design Summary View, it will open a text-based report. To open the Post-Place & Route Static Timing Analysis Report in Timing Analyzer, expand **Implement Design**, expand **Place & Route**, expand **Generate Post-Place & Route Static Timing**, and double-click **Post-Place & Route Static Timing Report**.

Reviewing the Post-Place & Route Static Timing Report



- What are the minimum clock periods for this design?



Reading Reports 2a - 25

© 2006 Xilinx, Inc. All Rights Reserved

NOTES

What are the minimum clock periods for this design?

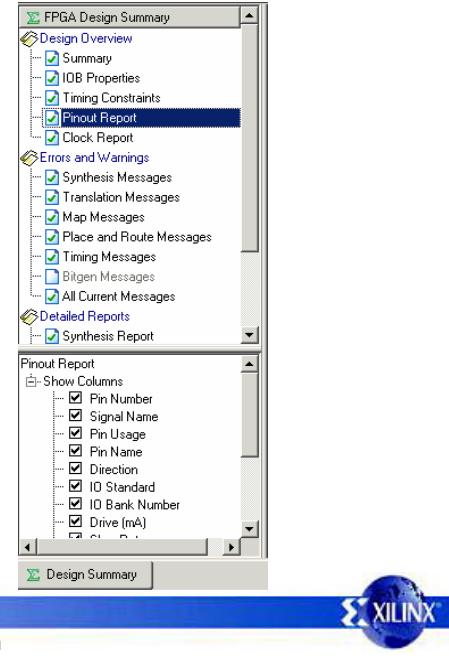
Reviewing the Post-Place & Route Static Timing Report

- What are the minimum clock periods for this design?
 - PERIOD for rd_clk: 1.789 ns
 - PERIOD for clk_bufr: 3.258 ns

NOTES

Read the Pad Report to Get Pinout Information

- Plain text Pad Report
 - <design>_pad.txt
- Pad Report
 - <design>.pad, <design>_pad.csv
- What pin location was assigned to the signal *reset*?
- What signal is on pin V18?



NOTES

You can read the Pad Report to get pinout information to start the board layout. Clicking Pinout Report will open an XML version of the report file. The report can also be opened from the Processes screen. You can sort the data by various fields. For example, you can sort by pin number, signal name, I/O standard, or data flow direction.

Reviewing the Pad Report

- . What pin location was assigned to the signal *reset*?
 - R1
- . What signal is on pin V18?
 - Empty



NOTES

Outline

- Introduction
- Area Goals
- Performance Goals
- **Summary**



NOTES

Summary

- A successful implementation means that your design meets your area and performance objectives
- The Map Report and Place & Route Report provide resource utilization and availability
- The Post-Map Static Timing Report gives you information to create reasonable timing constraints
- The Post-Place & Route Static Timing Report informs whether your timing constraints were met
- The Design Summary screen provides quick access to many reports



NOTES

Where Can I Learn More?

- Development System Reference Guide
 - www.xilinx.com → Documentation → Software Manuals



NOTES



Global Timing Constraints

© 2006 Xilinx, Inc. All Rights Reserved

NOTES

Objectives

After completing this module, you will be able to:

- Apply global timing constraints to a simple synchronous design
- Use the Constraints Editor to specify global timing constraints

NOTES

Outline



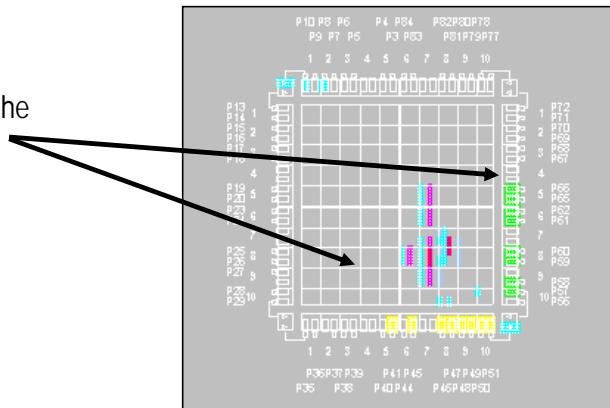
- . Introduction
- . Global Constraints
- . The Constraints Editor
- . Summary



NOTES

Designs Without Timing Constraints Produce Inconclusive Results

Note the logical structure of the placement and pins

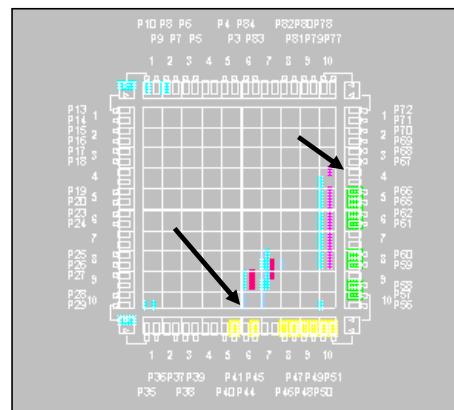


NOTES

Let's take a look at a design that had no timing constraints or pin assignments. Note the logical structure of the placement and pins. The implementation tools do a good job of placing and routing the design without using timing constraints. From this example, you can see that the logic is grouped closely to provide a good internal frequency and minimize clock skew. Likewise, the inout pins have a logic grouping because the design has no pin assignments. Note that the tools do not anticipate any ground-bounce problems. The tools do not know when the signals will be toggled or how many signals may change simultaneously. Be certain that you try to disperse switching outputs simultaneously to help avoid any problems with ground bounce.

Designs with Timing Constraints Provide Conclusive Timing Results

Note that the logic is placed closer to the I/O pins

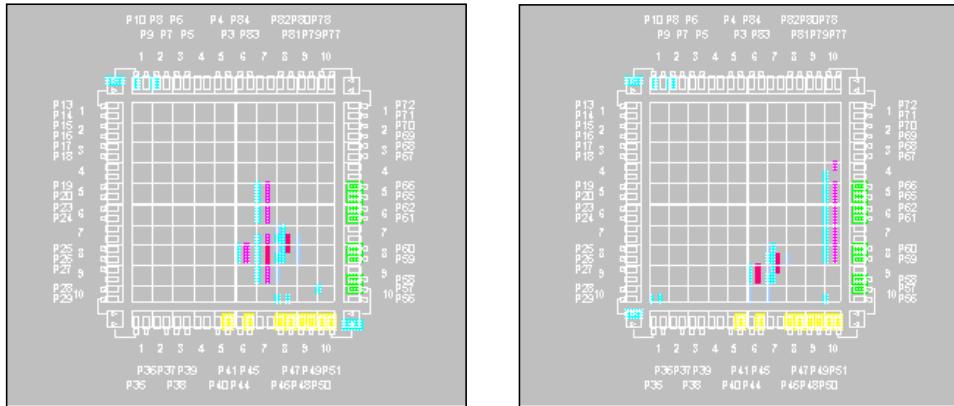


NOTES

Here is the same design with global timing constraints. In this design, inout flip-flops were not used. Moving the logic closer to the pins improves on-chip and off-chip timing. Note that the logic is placed closer to the inout pins due to input and output timing constraints (Offset In/Out constraints) in addition to a Period global timing constraint. Timing constraints are used to communicate performance objectives to the software—which in turn is used to place and route the design such that it meets those objectives.

Logic Placement Can Be Very Different with Timing Constraints

- Without global timing constraints
- With global timing constraints



Global Timing Constraints - 2b - 6

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

When timing constraints are used, the placement and routing solution can be very different. In this case, the pin placement is almost identical to the original implementation without timing constraints; however, the placement of the logic is very different. The net result is a faster performing design. Remember, timing constraints are used to communicate performance objectives to the software.

Timing Constraints Define Your Performance Objectives

```
=====
Timing constraint: OFFSET = OUT 6 ns AFTER COMP "clk_p" TIMEGRP "CLK0_GRP" ;
4 items analyzed, 0 timing errors detected.
Minimum allowable offset is 5.097ns.
```

```
=====
Slack: 0.903ns (requirement - (clock arrival + clock path + data path))
Source: clk_p (PAD)
Destination: tx_data<0> (PAD)
Source Clock: clk0 rising at 0.000ns
Requirement: 6.000ns
Data Path Delay: 5.667ns (Levels of Logic = 0)
Clock Path Delay: -0.570ns (Levels of Logic = 3)
=====
```

```
=====
Timing constraint: OFFSET = OUT 10 ns AFTER COMP "clk_p" TIMEGRP "CLK180_GRP" ;
4 items analyzed, 0 timing errors detected.
Minimum allowable offset is 9.097ns.
```

```
=====
Slack: 0.903ns (requirement - (clock arrival + clock path + data path))
Source: clk_p (PAD)
Destination: tx_data<0> (PAD)
Source Clock: clk0 falling at 4.000ns
Requirement: 10.000ns
Data Path Delay: 5.667ns (Levels of Logic = 0)
Clock Path Delay: -0.570ns (Levels of Logic = 3)
=====
```



NOTES

Timing constraints should be used to define your performance objectives. Tight timing constraints will increase your compile time. Unrealistic constraints will cause the implementation tools to stop. Use the Synthesis Report or the Post-Map Static Timing Report to determine whether your constraints are realistic. The Post-Place & Route Static Timing Report provides a design performance summary, plus detailed information about paths that failed to meet your constraints. After implementing, review the Post-Place & Route Static Timing Report to determine whether your performance objectives were met. If constraints were not met, use the Timing Report to determine the cause. You can learn more about interpreting Timing Reports in the “Achieving Timing Closure” module in the *Designing for Performance* course.

Inout Pads and Synchronous Elements Make Path End Points

- I/O pads
- Synchronous elements

Flip-flops

Latches

RAMs

NOTES

There are two types of path end points: inout pads and synchronous elements such as flip-flops, latches, and RAMs.

Creating a Timing Constraint is a Two-Step Process

Step 1: Create groups of path end points

Step 2: Specify a timing requirement between the groups

NOTES

Creating a timing constraint is done in two steps. First, you need to create groups of path end points, and then you need to specify a timing requirement between the groups. Global constraints use default groups of path end points such as all flip-flops, or all inout pads, and so on.

Knowledge Check

- . Can you tell us what the question is?
 - Answer is: drives packing, placement, and routing

*Raise your hand if you
think you know the
answer*



NOTES

Answers

- The answer is: drives packing, placement, and routing
 - What is the question?
 - . Why do we apply timing constraints?

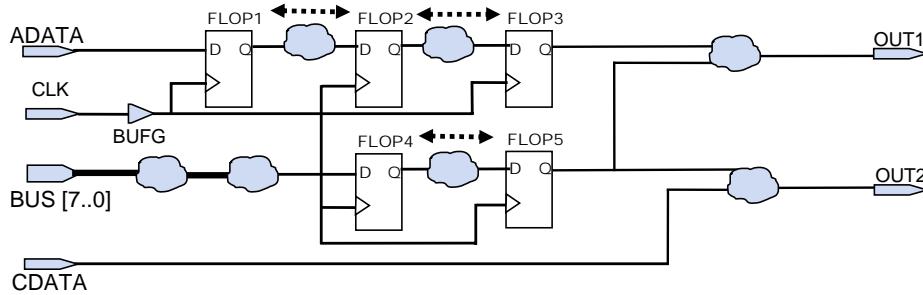
NOTES

Outline

- Introduction
- • **Global Constraints**
- The Constraints Editor
- Summary

NOTES

Period Constraints Cover Paths Between Synchronous Elements



Global Timing Constraints - 2b - 13

© 2006 Xilinx, Inc. All Rights Reserved

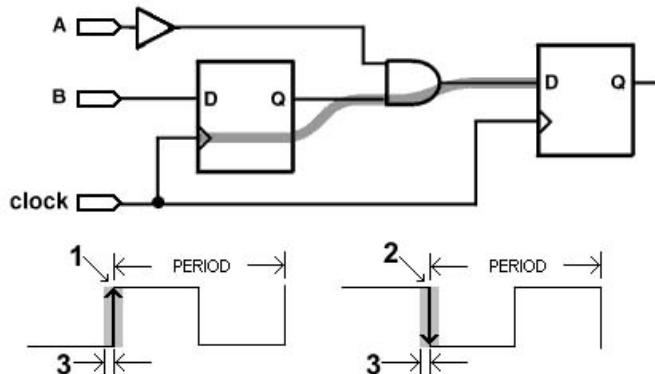


NOTES

PERIOD constraints cover paths between synchronous elements clocked by the reference net. Synchronous elements include flip-flops, latches, and synchronous RAM components.

PERIOD Constraints Use the Most Accurate Timing Information

- ✓ Clock skew between the source and destination flip-flops
- ✓ Synchronous elements clocked on the negative edge
- ✓ Unequal clock duty cycles
- ✓ Clock input jitter

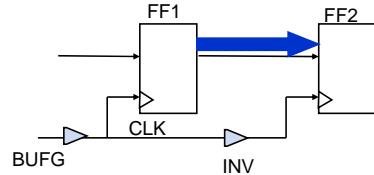


NOTES

PERIOD constraints use the most accurate timing information so they can automatically account for clock skew between the source and destination flip-flops, synchronous elements clocked on the negative edge, unequal clock duty cycles, and clock input jitter. These features make constraining your design much easier and give the tools the most flexibility in meeting all of your timing constraints.

An Example of the PERIOD Constraint

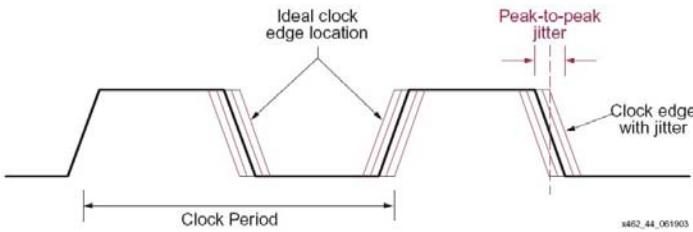
- Assume:
 - 50 percent duty cycle on CLK
 - PERIOD constraint of 10 ns
 - Because FF2 will be clocked on the falling edge of CLK, the path between the two flip-flops will be constrained to 50 percent of 10 ns = 5 ns



NOTES

Let's assume that there is a 50-percent duty cycle on CLK and a PERIOD constraint of ten nanoseconds. Because FF2 will be clocked on the falling edge of CLK, the path between the two flip-flops will actually be constrained to 50-percent of ten nanoseconds equals five nanoseconds. We will discuss clock input jitter in just a bit. Note that whenever you use the DCM component, the PERIOD constraint is placed on the input clock signal going into the DCM component. The PERIOD constraint will be propagated according to the DCM outputs.

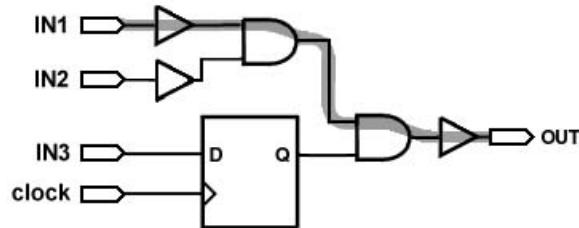
Clock Input Jitter is One Source of Clock Uncertainty



NOTES

Clock input jitter is one source of clock uncertainty. Clock uncertainty is automatically subtracted from PERIOD constraint setup paths and OFFSET IN constraint setup paths. Clock uncertainty is automatically added to PERIOD constraint hold paths, OFFSET IN constraint hold paths, and OFFSET OUT constraint paths.

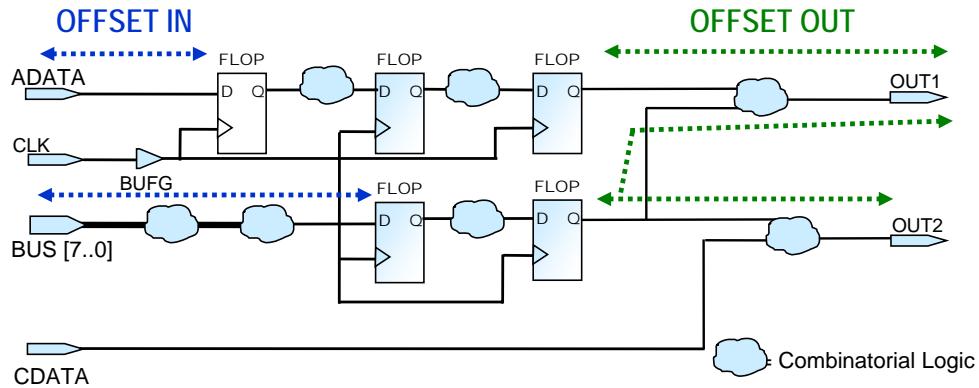
Pad to Pad Constraints Cover Purely Combinatorial Paths



NOTES

Purely combinatorial delay paths do not contain any synchronous elements. Purely combinatorial delay paths start and end at inout pads and are often left unconstrained. The pad-to-pad constraint is not used very often. Most designs do not contain any purely combinatorial delay paths. However, if your design does, you may wish to use it.

Offset In/Out Constrains I/O Pads To/From Synchronous Elements



Global Timing Constraints - 2b - 18

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

OFFSET constraints cover paths from Input pads to synchronous elements (OFFSET IN) and synchronous elements to output pads (OFFSET OUT).

Global OFFSET constraints are associated with a clock net, which identifies the synchronous elements that will be considered as valid path end points for the constraint. Synchronous elements include flip-flops, latches, and synchronous RAM components. Note that even after specifying an OFFSET IN, an OFFSET OUT, and a PERIOD constraint, it is still possible to have unconstrained paths, such as purely combinational paths. Remember that the OFFSET IN/OUT constraints do not cover paths between synchronous elements or purely combinational delay paths.

Offset Constraints Account for Delay and Jitter

```
-----  
Slack: -0.468ns (requirement - (data path - clock path - clock arrival + uncertainty))  
Source: wr_enl (PAD)  
Destination: wr_addr[2] (FF)  
Destination Clock: wclk rising at 0.000ns  
Requirement: 4.000ns  
Data Path Delay: 3.983ns (Levels of Logic = 2)  
Clock Path Delay: -0.485ns (Levels of Logic = 3)  
Clock Uncertainty: 0.000ns  
Data Path: wr_enl to wr_addr[2]  
-----
```

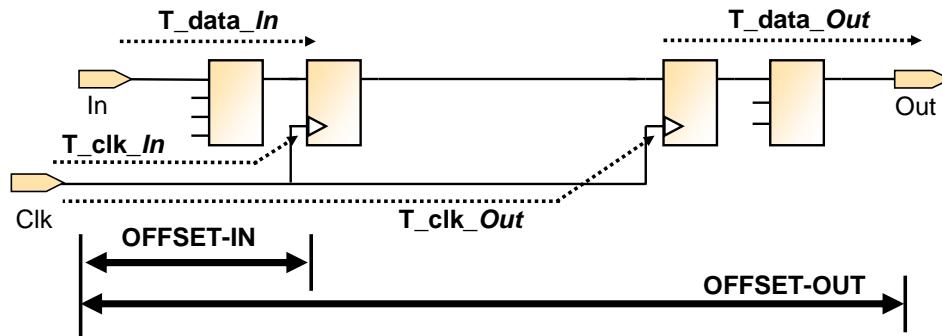


NOTES

OFFSET constraints automatically account for the clock distribution delay. They provide the most accurate timing information, increase the amount of time for input signals to arrive at synchronous elements, and reduce the amount of time for output signals to arrive at output pins. OFFSET constraints also account for clock input jitter. They use the jitter defined on the associated PERIOD constraint. The OFFSET IN and OUT constraints take the clock distribution delay into account. Therefore, the tools evaluate the effective input path as the data input delay minus the clock delay. Likewise, the effective output path is the data output delay plus the clock delay. Because the clock distribution delay is tangible in these cases, the input delays are reduced, and the output delays are increased when the clock distribution delay is taken into account. Because the tools account for the clock distribution delay, you can specify your pin-to-pin timing goals without knowing what the internal clock distribution delay will be. When analyzing timing for OFFSET constraints, the Xilinx tools examine the associated PERIOD constraint to check for clock input jitter.

Offset Constraints Account for Clock Delay

- OFFSET IN = $T_{\text{data_In}} - T_{\text{clk_In}}$
- OFFSET OUT = $T_{\text{data_Out}} + T_{\text{clk_Out}}$



NOTES

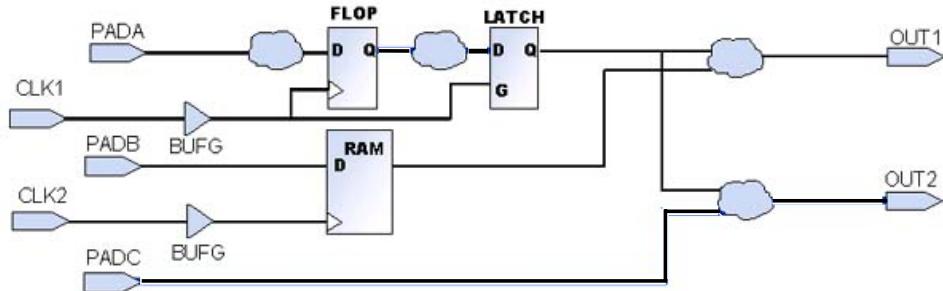
This example shows how the tools use the clock distribution delay to calculate the effective OFFSET IN and OFFSET OUT delay. Because the input datapath and clock path are in parallel for input paths, the tools subtract the clock distribution delay from each input path. Likewise, because the output datapath and the clock path are in series for output paths, the tools add the clock distribution delay to each output path. Therefore, having a positive clock distribution delay helps your input times, but it hurts your output times. To compensate, the Virtex™ devices include a DLL to remove the clock distribution delays from your input and output paths. The DLL is very important for good chip-to-chip timing.

Remember that the DLL does not remove clock skew. The DLL only removes clock distribution delay. The global routing resources are designed to minimize clock skew, but they cannot remove it entirely.

Note that the datapaths include the IOB, net, logic, and setup times or hold times associated with the registers.

Knowledge Check

- Which paths are constrained by a pad-to-pad constraint?



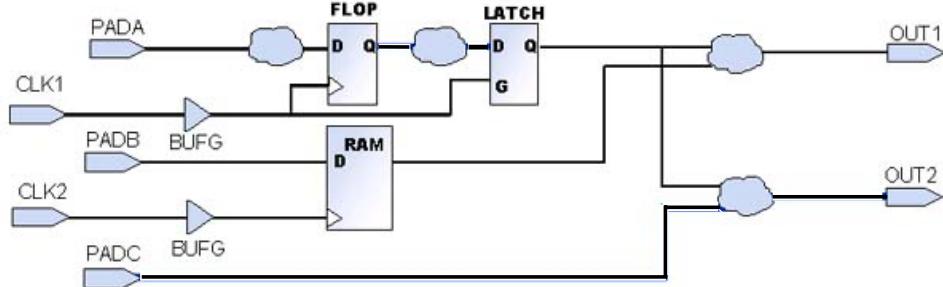
*Raise your hand if you
think you know the
answer*



NOTES

Answers

- PADC to OUT2 are paths constrained by a pad-to-pad constraint.



NOTES

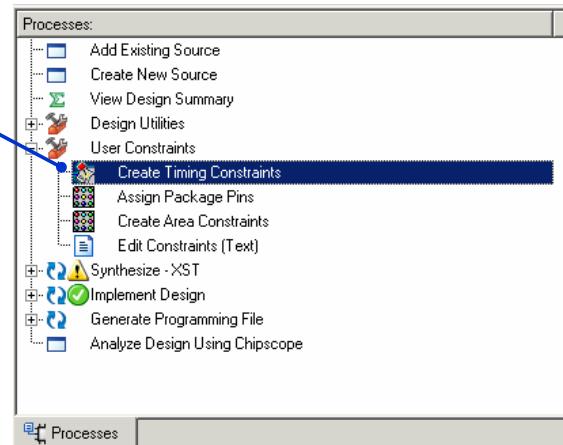
Outline

- Introduction
 - Global Constraints
 - **The Constraints Editor**
 - Summary
- 

NOTES

Launching the Constraints Editor

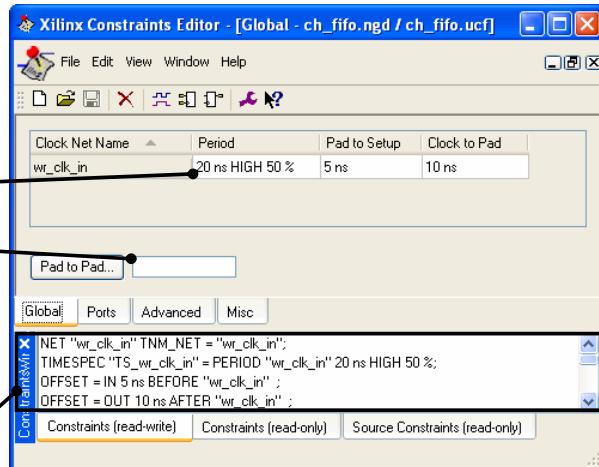
- Expand User Constraints in the Processes for Source window
- Double-click Create Timing Constraints



NOTES

Entering PERIOD and Pad-to-Pad Constraints

- PERIOD and Pad-to-Pad constraints can be entered on the Global tab
- Double-click here to make a PERIOD constraint
- Global Pad-to-Pad constraint
- Constraints can be deleted by selecting the **constraint in the text window** and pressing <Delete>



NOTES

To open the Period Constraint options box, double-click **under the Period column heading**. You can also single-click **in the Period box** and enter a value. If you single-click and enter a value, the constraint will have default units of “ns” and a 50-percent duty cycle. The units and duty cycle can be customized if you double-click **in the Period box**. As constraints are made with the Constraints Editor, they are added to the UCF file. Use any text editor to edit the User Constraints File.

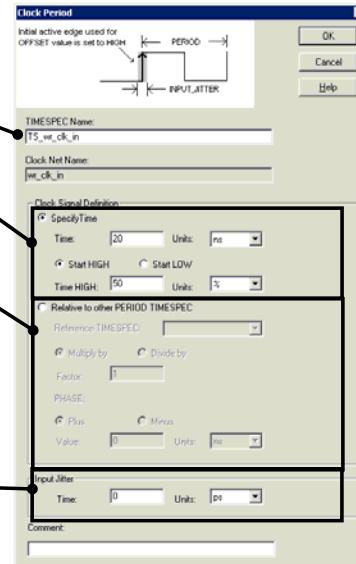
The window at the bottom of the Constraints Editor shows the following:

- UCF constraints (read/write). You can edit these constraints with the Constraints Editor.
- UCF constraints (read only). You cannot edit these constraints with the Constraints Editor. You must open the UCF directly.
- Source Constraints. You use your synthesis tool to create these constraints, and you cannot edit them with the Constraints Editor. You must modify these constraints with your synthesis tool and resynthesize your design.

Note: Right-click **each read/write constraint** to delete, dock, disable, or hide each constraint.

PERIOD Constraint Options

- **TIMESPEC Name**
- Specific constraint value
 - Active clock edge
 - Duty cycle
- Relative to other PERIOD TIMESPEC
 - Useful for designs with multiple clock signals
 - Can define both frequency and phase relationships
- **Input Jitter**



NOTES

The default name for a PERIOD constraint is TS_<clk_net_name>. The default active clock edge is a rising edge, and the default duty cycle is 50 percent.

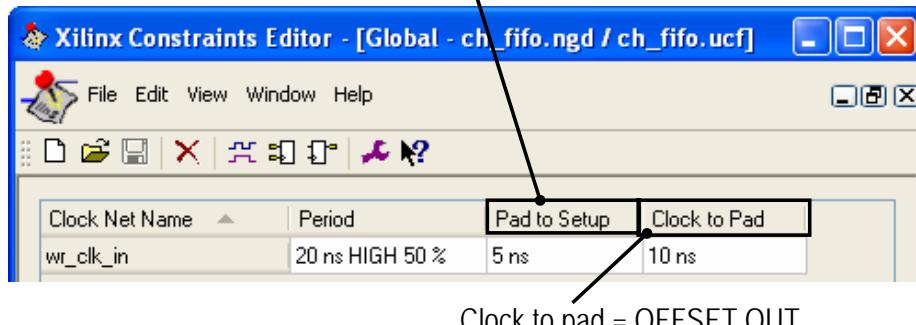
If your design uses a DCM, simply define a PERIOD constraint on the clock input to the DCM. The software will push the constraint onto all used clock outputs of the DCM, using relative PERIOD constraints to define the frequency and phase relationships.

If you specify a value for input jitter on this clock, the jitter is used to calculate worst-case timing. When checking setup times, jitter is subtracted from the clock period. When checking hold times and clock-to-out times, jitter is added to the clock period.

Entering OFFSET Constraints

- Global OFFSET IN and OFFSET OUT constraints can be made on the Global tab

Pad to Setup = OFFSET IN



NOTES

Remember that these constraints will optimize all input paths and output paths to the design. Pin-specific OFFSET IN/OUT constraints can be entered from the Ports tab. These constraints are covered in the “Timing Groups and OFFSET Constraints” module in the *Designing for Performance* course.

Outline

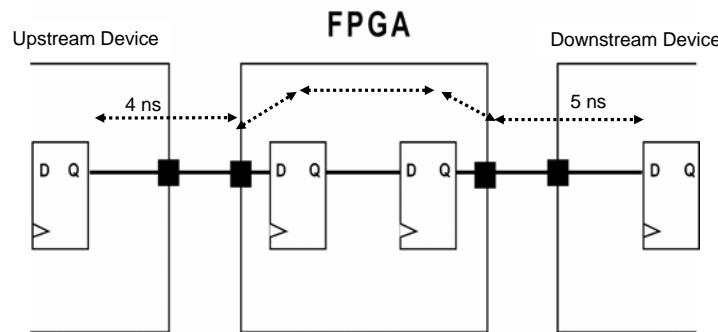
- Introduction
- Global Constraints
- The Constraints Editor
- · Summary



NOTES

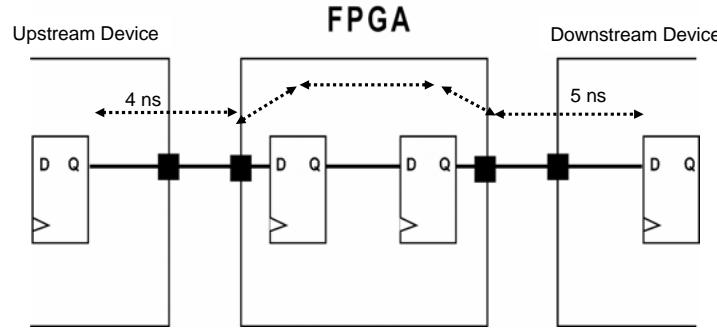
Review Question

- Given the system diagram below, what values would you put in the Constraints Editor so that the system will run at 100 MHz?
 - Assume no clock skew between devices



NOTES

Review Question



PERIOD = ____ ns , OFFSET IN = ____ ns, and OFFSET OUT = ____ ns

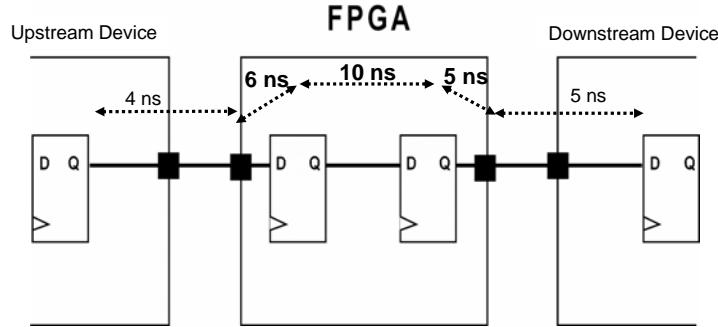
*Raise your hand if you think you know
the answer and use the text tool to type
the answer onto the whiteboard*



NOTES

Answer

- Given the system diagram below, what values would you put in the Constraints Editor so that the system will run at 100 MHz?



- Answer: PERIOD = 10 ns , OFFSET IN = 6 ns, and OFFSET OUT = 5 ns

NOTES

Because you want the entire system/board to run at 100 MHz (10 ns), the delays from one synchronous element to another synchronous element must be no more than 10 ns. Therefore, the PERIOD constraint, which is from one synchronous element to another, will be 10 ns.

The OFFSET IN and OFFSET OUT constraints must be calculated because the values are related to the upstream and downstream device.

For OFFSET IN, we need the delay from the synchronous element in the upstream device to the synchronous element in the FPGA to be 10 ns. The delay through the synchronous element in the upstream device plus the trace delay is 3 ns, as shown in the figure. Hence, the delay within the FPGA can be no more than 10 ns to 3 ns, which gives you a final value of 7 ns.

A similar calculation is completed for the OFFSET OUT constraint.

Summary

- Performance expectations are communicated with timing constraints
- PERIOD constraints cover delay paths between synchronous elements
- OFFSET constraints cover delay paths from input pins to synchronous elements and from synchronous elements to output pins
- Use the Constraints Editor to create timing constraints

NOTES

Where Can I Learn More?

- Timing presentation on the Web at www.xilinx.com
 - Support → Documentation → Technical Tips → Timing & Constraints → Getting Started → The Timing Presentation
- Timing Improvement Wizard on the Web at www.xilinx.com
 - Support → Problem Solvers
- Timing Closure TechXclusive article on the Web at www.xilinx.com
 - Support → TechXclusives



NOTES

You can learn more about the Xilinx design process on www.xilinx.com. For more information on implementation, check out the software manuals. Note that documentation may also be installed on your computer. Remember, you can go to the Configuration Problem Solver for help with debugging configuration problems. ISIM Online Help provides more information on simulation.

Where Can I Learn More?

The screenshot shows the Xilinx Support website interface. At the top, there's a navigation bar with links for US Site, 日本サイト, 中文网站, Documentation, Download, Buy Online, and Login. Below the navigation is the Xilinx logo and a search bar labeled "Enter Search Terms" with a "Search" button. A dropdown menu for "Answers Database" has "Advanced Search" selected.

The main content area is titled "Support by Device Family" and shows tabs for Virtex-4, Spartan-3 LX, Virtex-2 Pro, Virtex-2, and CoolRunner-II. A dropdown menu for "Device List" is open. On the left, there's a sidebar for "Virtex-4 Documentation" with sections for Hardware Documentation (Data Sheets, User Guides, etc.), Configuration Solutions Documentation (System ACE Solutions, Configuration PROCESSES, Configuration Examples), and Reference Software Documentation (ISE Software Manuals, EDK User Guides, System Generator User Guide, ChiselScope User Manual). Below this is a section for "Virtex-4 Documentation Alerts/Changes" with three items: IC105025 - Virtex Stepping Methodology: New Initial Release, WP233 - IEEE 802.17 Resilient Packet Ring Networks Enable, and Virtex4GIP for Virtex-4 FPGAs User Guide: Typographical Updates. There's also a link to "Subscribe to Alerts".

In the center, there's a box titled "Support Quicklinks" containing a list of links:

- > [Answer Browser](#) - Search our Answers Database
- > [WebCase](#) - Need a resolution to your problem? Submit a WebCase
- > [Software Manuals](#) - Find technical documentation and literature
- > [Forums](#) - Share your design techniques, opinions, and ideas
- > [Tech Tips](#) - Discover technical information on Xilinx products
- > [Problem Solvers](#) - Troubleshoot by answering a series of questions
- > [TechXclusives](#) - Read technical articles from Xilinx experts
- > [Alerts](#) - Subscribe to Alerts and stay in tune with new or changed content

At the bottom of the page, there's a blue footer bar with the text "Global Timing Constraints - 2b - 34" on the left and "© 2006 Xilinx, Inc. All Rights Reserved" on the right. To the right of the footer is the Xilinx logo.

NOTES

To learn more about global timing constraints, go to www.xilinx.com and click **Support**. Here you will find Support Quicklinks. Click **Software Manuals**, **Tech Tips**, **Forums**, **TechXclusives**, or **Problem Solvers** to find information on timing constraints. To learn more about timing constraints in addition to what you may find on the Support site, register for Xilinx courses on the Education Services site.



Global Timing Constraints Lab

Introduction

©2006 Xilinx, Inc. All Rights Reserved

NOTES

Objectives

After completing this lab, you will be able to:

- Use the Xilinx Constraints Editor to enter global timing constraints
- Use the Post-Map Static Timing Report to verify that the timing constraints are realistic
- Use the Post-Place & Route Static Timing Report to determine the delay of the longest constrained path for each timing constraint



NOTES

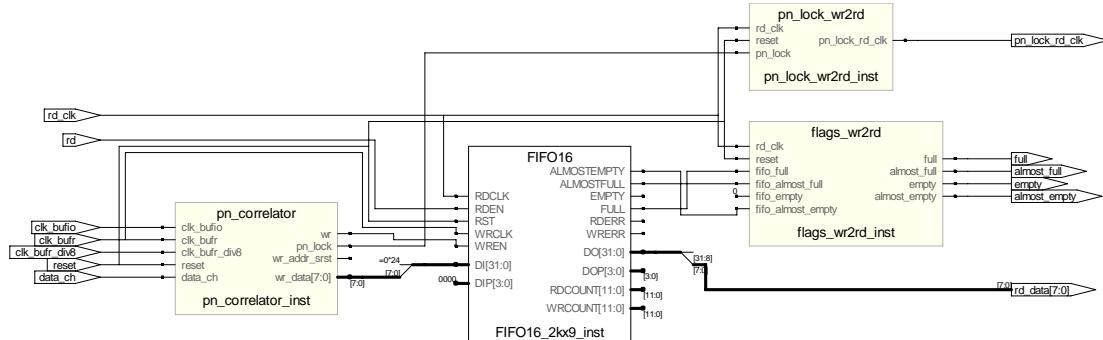
Introduction

- This lab illustrates how to use the global timing constraints in the Constraints Editor
- You will enter the global timing constraints for an existing project (the flow project)
- You will then use reports to analyze the design and the constraints



NOTES

Channel FIFO Design



Global Timing Constraints Lab Intro 2c -4

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

This is a block diagram for the lab design, which you will use in this course.

The PN Correlator searches for a specific pattern (PN code) in the input data stream. Once the PN code is found, the following 255 bytes of data are stored in the FIFO. After the 255 bytes of data are stored, the correlator begins to search for the PN code again.

The FIFO Status block generates read/write addresses for the FIFO and creates flags that indicate how much data has been stored in the FIFO. These flags are used by the Data Control block to decide from which channel to read.

The FIFO block is a simple dual-port RAM that stores the data from the PN Correlator block.

For more information about the design, refer to the “Detailed Design Description for Labs” section of this course workbook or the HDL source code for this design.

General Flow

- Step 1: Open the project
- Step 2: Enter the global timing constraints
- Step 3: Implement the design and analyze the timing



NOTES

Global Timing Constraints Lab

Global Timing Constraints Lab

Introduction

In this lab, you will use the global timing constraints to improve the system clock frequency of an existing project. You also will use the Post-Map Static Timing Report and the Post-Place & Route Static Timing Report to analyze the performance of the design.

Objectives

After completing this lab, you will be able to:

- Use the Xilinx Constraints Editor to enter the global timing constraints
- Review the Post-Map Static Timing Report and verify that the timing constraints are realistic
- Use the Post-Place and Route Static Timing Report to determine the delay of the longest constrained path for each timing constraint

Design Description and Other Information

The Anatomy of a Timing Report

Timing reports detail why your timing constraints failed and which paths passed or failed.

When you open a timing report, the Timing Analyzer utility generates your timing report.

The Timing Analyzer GUI contains three windows (**Figure 2d-1**). You can use the Hierarchical Browser, on the left, to navigate through large reports easily.

The Path Detail window is in the lower-right corner. The Path Detail window contains the actual text of the timing report.

The top window displays the section of the timing report that you are currently viewing in the Path Detail window.

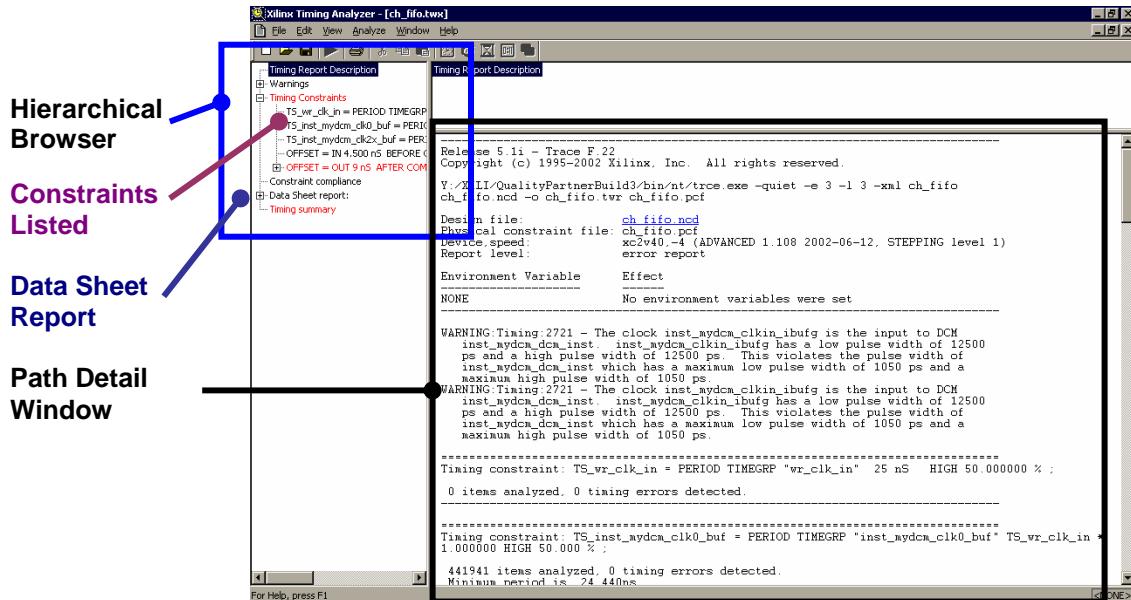


Figure 2d-1. The Timing Analyzer GUI

The Detailed Path Analysis (**Figure 2d-2**) contains information about the delay path, including the following:

- Slack—the difference between the constraint and the actual length of the path (negative slack indicates that the path failed to meet the constraint)
- Path source and destination
- List of incremental delays along the path (abbreviations correspond to data sheet information)
- Fanout of each net in the delay path
- Total delay on the path
- Percentage breakdown between logic and routing—this provides you with an idea of whether your delay path was poorly placed

The constraint, number of paths analyzed, and number of errors

Longest path (least slack)—summary of path delay information

Detailed path description

Total delay (split into logic and routing)

```

=====
Timing constraint: OFFSET = OUT 9 nS AFTER COMP "wr_clk_in";
13 items analyzed, 8 timing errors detected.
Minimum allowable offset is -0.431ns

Slack: -0.431ns (requirement - (clock arrival + clock path + data path))
Source: wr_clk_in (PAD)
Destination: rd_data<0> (PAD)
Source Clock: xd_clk rising at 0.000ns
Requirement: 9.000ns
Data Path Delay: 10.079ns (Levels of Logic = 1)
Clock Path Delay: -0.648ns (Levels of Logic = 3)
Timing Improvement Wizard
Clock Path: wr_clk_in to fifo_2048x8_inst_fifo_bram_B
Delay type   Delay(ns) Logical Resource(s)
Tq10o          0.825  wr_clk_in inst_mydcm_c1kin_ibufg_inst
net (fanout=1) 0.798  inst_mydcm_c1kin_ibufg
Tdcmin0        -4.186  inst_mydcm_dcm_inst
net (fanout=1) 0.852  inst_mydcm_c1k2x_buf
Tq10o          0.589  inst_mydcm_c1k2x_bufq_inst_GCLKMUX
net (fanout=27) 0.474  inst_mydcm_c1k2x_bufq_inst_rd_clk
Total          -0.648ns (-2.772ns logic, 2.124ns route)

Data Path: fifo_2048x8_inst_fifo_bram_B to rd_data<0>
Delay type   Delay(ns) Logical Resource(s)
Tbcko          2.647  fifo_2048x8_inst_fifo_bram_B
net (fanout=1) 1.325  rd_data_0obuf
Tioop          6.107  rd_data_0obuf
rd_data<0>
Total          10.079ns (8.754ns logic, 1.325ns route)
(86.9% logic, 13.1% route)

```

Figure 2d-2. Detailed Path Analysis

Procedure

In this lab, you enter the global timing constraints. The design is the same design as in the demonstrations.

This lab comprises three primary steps: You will open the project, enter the global timing constraints, and, finally, implement the design and analyze the timing.

For each procedure within a primary step, there are general instructions (indicated by the symbol). These general instructions only provide a broad outline for performing the procedure. Below these general instructions, you will find accompanying step-by-step directions and illustrated figures that provide more detail for performing the procedure. If you feel confident about completing a procedure, you can skip the step-by-step directions and move on to the next general instruction.

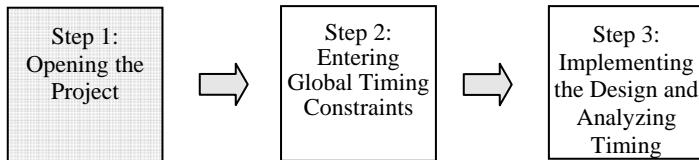
Note: If you are not using **Toolwire** to perform this lab, all software programs, files, and projects will be located on the C:\ drive instead of R:\.

Note: If you are unable to complete the lab at this time, you can download the original lab files for this module from the Xilinx FTP site at ftp://ftp.xilinx.com/pub/documentation/education/fpga13000-82-xlnx_lab_files.zip. These are the original lab files and do not contain any work you may have previously completed.

Opening the Project

Step 1

General Flow for this Lab:



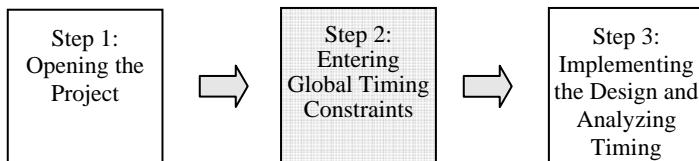
- ◆ Launch the ISE™ Project Navigator and open the *time_const* project located at *R:\training\fund\labs\time_const* or *R:\training\fund\labs\time_const_s3*.

- ① Select Start → Programs → Xilinx ISE 8.2i → Project Navigator
- ② Select File → Open Project in the Project Navigator
- ③ Browse to one of the following directories: *R:\training\fund\labs\time_const* (Virtex-4) or *R:\training\fund\labs\time_const_s3* (Spartan-3), and select **time_const.ise**
- ④ Click Open

Entering the Global Timing Constraints

Step 2

General Flow for this Lab:



- ◆ Launch the Constraints Editor.



1. What effect do timing constraints have on the implementation tools? What do the tools do if there are no constraints applied?
-
-

- ① In the Sources for: window, select the top-level design file *ch_fifo.ngc*

- ② In the Processes window, expand **User Constraints** and double-click **Create Timing Constraints** (Figure 2d-3)

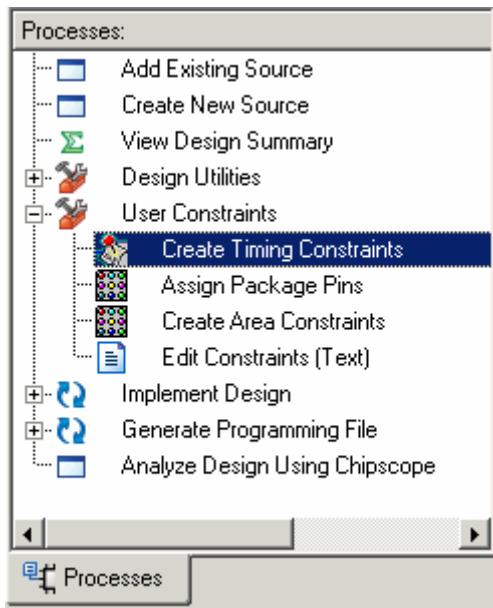


Figure 2d-3. Processes for Source Window

The project does not currently have a UCF file associated with it. The Project Navigator will offer to create one automatically.

- ③ Click **Yes** to create a new UCF file called *ch_fifo.ucf* and add it to the project

- ④ When the Constraints Editor opens, click the **Global** tab (**Figure 2d-4**)

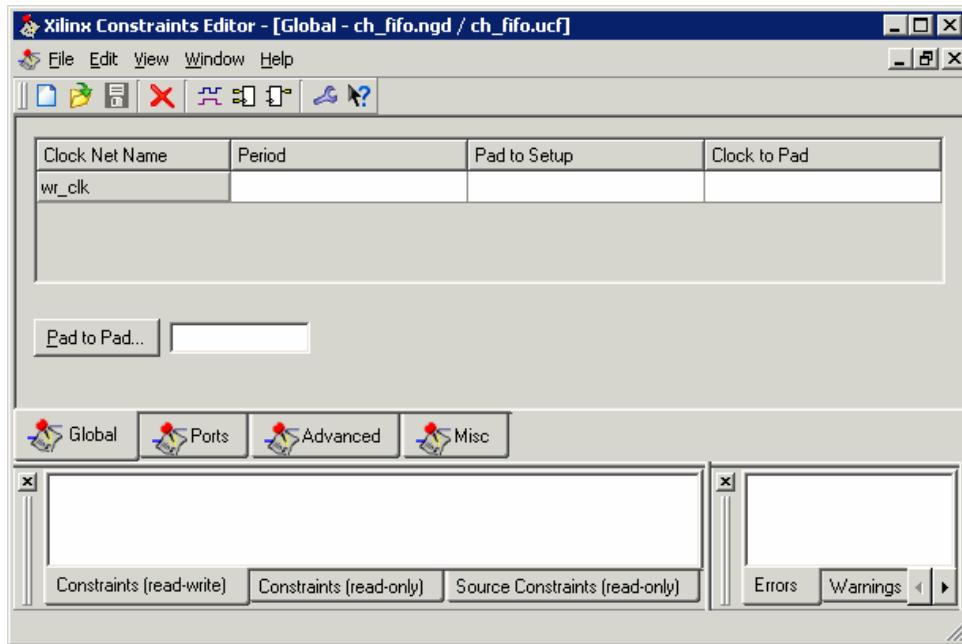


Figure 2d-4. Global Tab



Enter a PERIOD constraint of 20 ns for *wr_clk*. For SpartanTM-3 users, the clock signal is named *wr_clk_in*.

- ❶ Under the Period column heading, double-click in the **Period column box that corresponds to the *wr_clk* signal**

Spartan-3 users: The clock signal is named *wr_clk_in*

This opens the Clock Period dialog and allows you to enter a PERIOD constraint (Figure 2d-5).

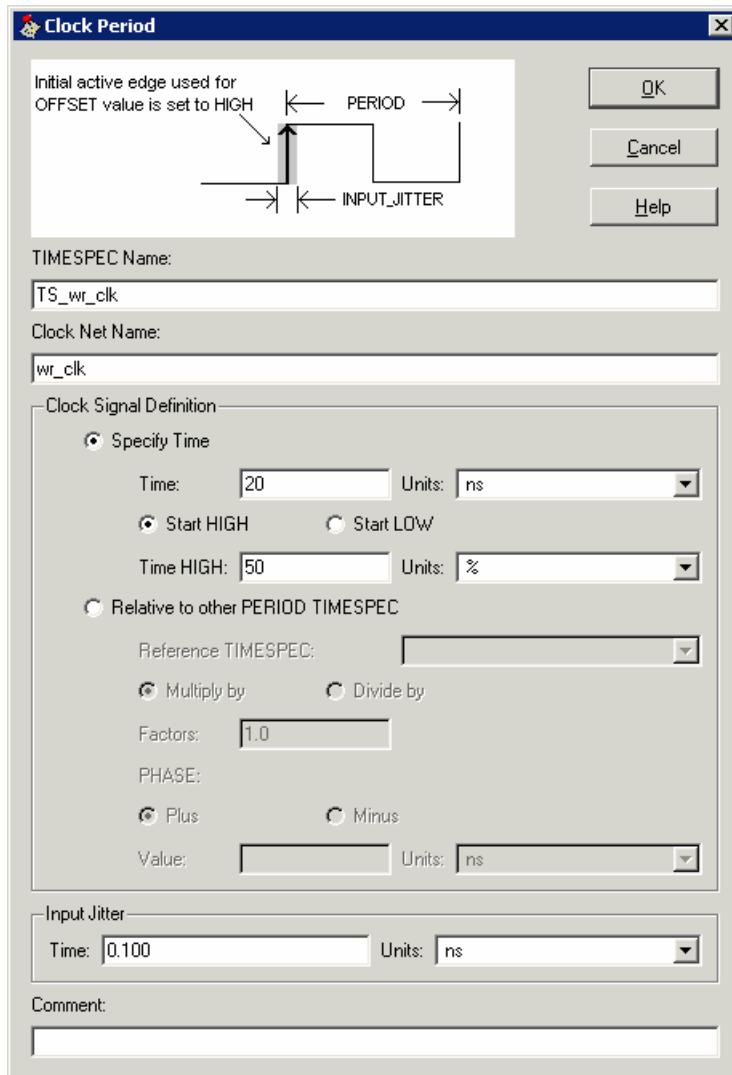


Figure 2d-5. Clock Period Dialog

- ② Enter a Specific Time of **20 ns** and an Input Jitter Time of **0.100 ns**. Click **OK**

➡ Enter an OFFSET IN constraint of *5 ns* and an OFFSET OUT constraint of *6 ns* for *wr_clk*. Then save the constraints and exit the Constraints Editor.

- ① Click the **box under the Pad to Setup column heading**, type **5**, and press **<Enter>**. This is the OFFSET IN constraint (**Figure 2d-6**)
- ② Click the **box under the Clock to Pad column heading**, type **6**, and press **<Enter>**. This is the OFFSET OUT constraint
- ③ Select **File → Save**
- ④ Select **File → Exit** to exit the Constraints Editor

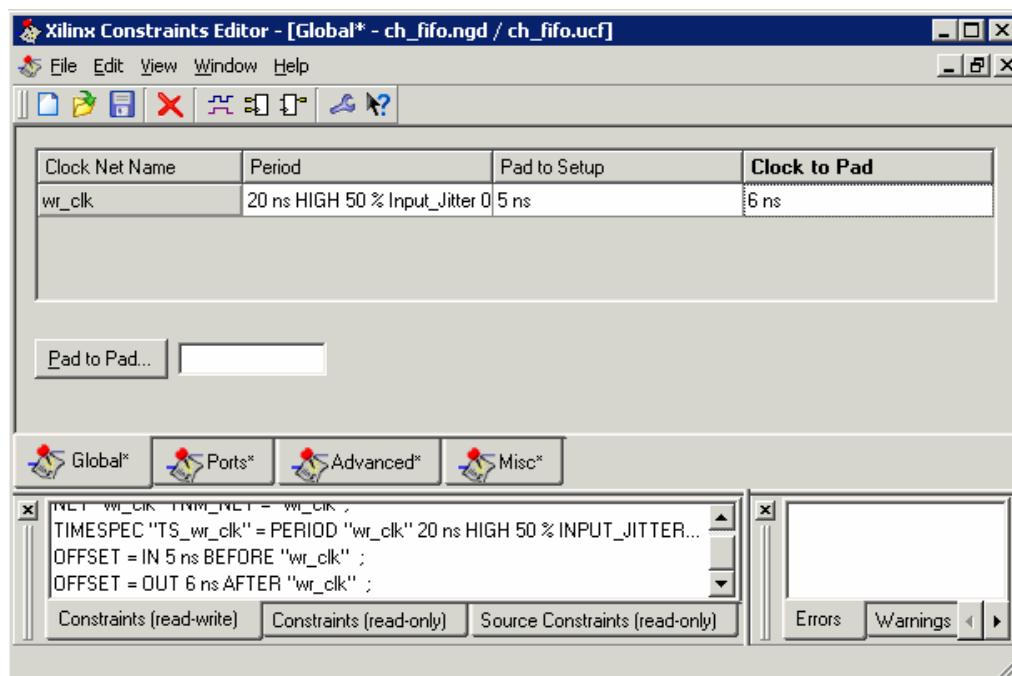


Figure 2d-6. Constraints Editor Global Tab

Implementing the Design and Analyzing the Timing Step 3

General Flow for this Lab:



► Implement the design. Look through the Post-Map Static Timing Report and the Post-Place & Route Static Timing Report to complete Chart 1 and Chart 2 in this section.

- ❶ In the Sources window, select **ch_fifo.ngc**. In the Processes for Source window, expand the **Implement Design** process, and expand the **Map** process

! If you do not see the Implement Design process, make sure that **ch_fifo.ngc** is selected in the Sources in Project window.

- ❷ Expand the **Generate Post-Map Static Timing** process
- ❸ Double-click **Analyze Post-Map Static Timing**

Performing these steps implements the design through MAP, generates the Post-Map Static Timing Report, and opens the report in the Timing Analyzer. Use the report to verify that your timing constraints are realistic and to avoid wasting Place & Route time.

Note: The PERIOD constraint was attached to the input clock net, which drives a DCM. During Translate, this constraint is moved to the DCM outputs and renamed. The CLK0 output of the DCM still has a 20-ns constraint.

- ?**
2. Given the above information, what constraint, do you suppose, is applied to the CLK2X output of the DCM? What about the CLK1AD8 output of the PMCD, which is driven by CLK0 (Virtex™-4 design only)?
-
-

Virtex-4 device users: When filling in the charts, locate the PERIOD constraint that was placed on the CLKA1 (=CLK0 or wr_clk) and CLKB1 (CLK2X) output of the PMCD. The other Period constraint on the PMCD's CLKA1D8 output drives the input SERDES but otherwise will not constrain any paths.

Spartan™-3 device users: When filling in the charts, locate the PERIOD constraints that were placed on the CLK0 and CLK2X outputs of the DCM.



3. Complete the row titled Post-Map in the following chart:

Chart 1	CLK0_CLKA1 PERIOD constraint	CLK2X_CLKB1 PERIOD constraint	OFFSET IN constraint	OFFSET OUT constraint
Constraint	20 ns	10 ns	5 ns	6 ns
Post-Map				

Compare your answers with those in the Answers section at the end of this lab.



4. Given the Post-Map timing results, is it likely that timing will be met after Place & Route? Why or why not?

④ Exit the Timing Analyzer

- ⑤** In the Processes window, expand the **Place & Route** process and the **Generate Post-Place & Route Static Timing** process
- ⑥** Double-click **Analyze Post-Place & Route Static Timing**

The Spartan™-3 design will fail during the Place & Route process because the OFFSET OUT constraint cannot be met, even if all routing delays were 0 ns.



5. Complete the row titled Post-PAR in the following chart (Virtex™-4 design only):

Chart 2	CLK0_CLKA1 PERIOD constraint	CLK2X_CLKB1 PERIOD constraint	OFFSET IN constraint	OFFSET OUT constraint
Constraint	20 ns	10 ns	5 ns	6 ns
Post-PAR				

Compare your answers with those in the Answers section at the end of this lab.



6. Compare the Post-Map results with the Post-Par results. By approximately what factor did the two differ? Is this about what you expected? Did anything surprise you?

Additional Exercises (Optional)



Spartan-3 Device users: Explore the three types of hyperlinks in the Post-Map Timing Report.

Conclusion

In this lab, you used the Xilinx Constraints Editor to enter the global timing constraints. You also reviewed the Post-Map and Post-Place & Route Timing Reports.

Timing constraints are the best way to communicate your performance expectations to the implementation tools.

You must verify that your timing constraints are realistic while the implementation tools are placing and routing your design for the first time. You can get an estimate of the timing performance from the Post-Map Static Timing Report.

After implementation is complete, timing constraints must be verified with the Post-Place & Route Static Timing Report or a custom timing report generated by the Timing Analyzer.

A Answers

Lab answers listed represent sample solutions only. Your results may differ depending on the version of the software, service pack, or operating system that you are using.

- What effect do timing constraints have on the implementation tools? What do the tools do if there are no constraints applied?

A: Constraints drive packing, placement, and routing results. Without constraints, the tools place and route the design in a minimal amount of time which will not necessarily provide the best results.

- Given the above information, what constraint, do you suppose, is applied to the CLK2X output of the DCM? What about the CLK1AD8 output of the PMCD, which is driven by CLK0 (Virtex™-4 design only)?

A: The tools automatically apply the correct Period constraint value, based on the input clock constraint. Thus, the CLK2X output will have a 10-ns Period constraint applied. The PMCD's CLKA1D8 output will have a 160-ns Period constraint applied.

- Complete the row titled Post-Map in the following chart:

Chart 1	CLK0_CLKA1 PERIOD constraint	CLK2X_CLKB1 PERIOD constraint	OFFSET IN constraint	OFFSET OUT constraint
Constraint	20 ns	10 ns	5 ns	6 ns
Post-Map (Virtex-4)	~2.3	~1.5	~2.9	~3.0
Post-Map (Spartan-3)	~14.9	~4.6	~3.3	~7.4

- Given the Post-Map timing results, is it likely that timing will be met after Place & Route? Why or why not?

A: For the Virtex™-4 design, given that the Post-Map estimates are all around 50 percent or less of the available budget, there is a good chance that we will meet timing. For the Spartan™-3 design, the OFFSET OUT constraint is already failing. It also looks like there may be problems with the main PERIOD constraint.

- Complete the row titled Post-PAR in the following chart:

Chart 2	CLK0_CLKA1 PERIOD constraint	CLK2X_CLKB1 PERIOD constraint	OFFSET IN constraint	OFFSET OUT constraint
Constraint	20 ns	10 ns	5 ns	6 ns
Post-PAR (Virtex-4)	~4.4	~2.1	~4.0	~5.2

6. Compare the Post-Map results with the Post-PAR results. By approximately what factor did the two differ? Is this about what you expected? Did anything surprise you?

A: For CLK0_CLKA1 = 1.91, for CLK2X_CLKB1 = 1.4, for Offset In = 1.38, for Offset Out = 1.73. The average factor = 1.6. Post-PAR timing results normally will result in approximately 1.5 to 2x the delay as reported in Post-Map results. This is not always the case. For example, a path that contains carry logic would not double the delays. However, this is a good rule of thumb to go by.



Global Timing Constraints Lab

Review



NOTES

Review Questions

- In this lab, you used a PERIOD constraint. Now that the design has met this constraint, will the entire system (board) be able to run at this speed?
- What are some things you must consider when determining the system frequency?



NOTES

Answers

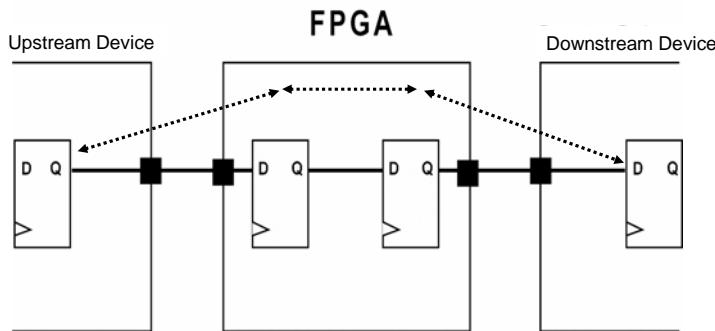
- . In this lab, you used a PERIOD constraint. Now that the design has met this constraint, will the entire system (board) be able to run at this speed?
 - Not necessarily
- . What are some things you must consider when determining the system frequency?
 - You must find the longest path between synchronous elements in the entire system. This may be a path within the FPGA, a path within another device, or a path between two devices



NOTES

Explanation

- Data traveling from synchronous element to synchronous element, on the entire board, determines system performance. The Xilinx tools only know information about data traveling within the FPGA. The tools do not know the external board delays
- The system frequency will be the slowest of the three paths shown below:



NOTES

Summary

- Use the Constraints Editor to enter the global timing constraints (PERIOD and OFFSET) with ease
- Use the Post-Place & Route Static Timing report to see whether your constraints were met



NOTES



Synchronous Design Techniques

This Synchronous Design Techniques module will show you how to use the hierarchy effectively and increase the circuit reliability and performance by applying synchronous design techniques.

Objectives

After completing this module, you will be able to:

- Use the hierarchy effectively
- Increase the circuit reliability and performance by applying synchronous design techniques

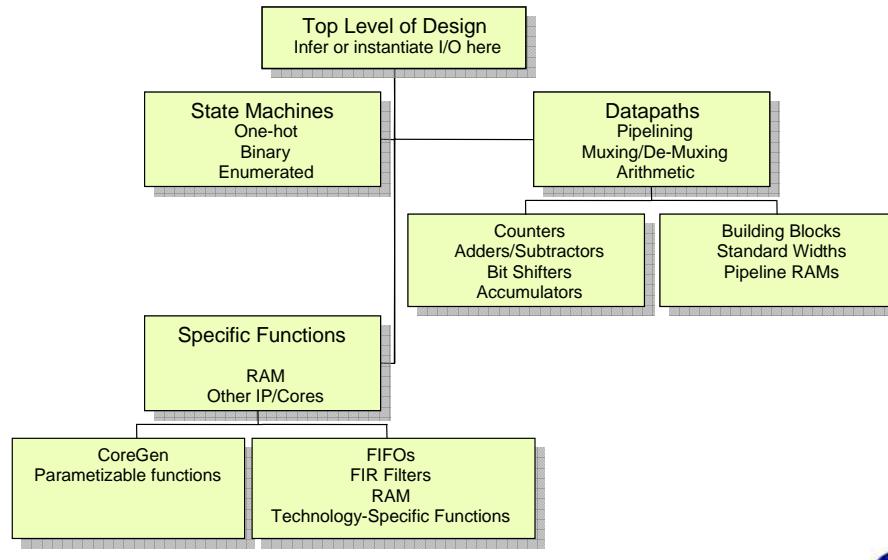


Outline

- . Hierarchical Design
. Synchronous Design for
Xilinx FPGAs
. Summary

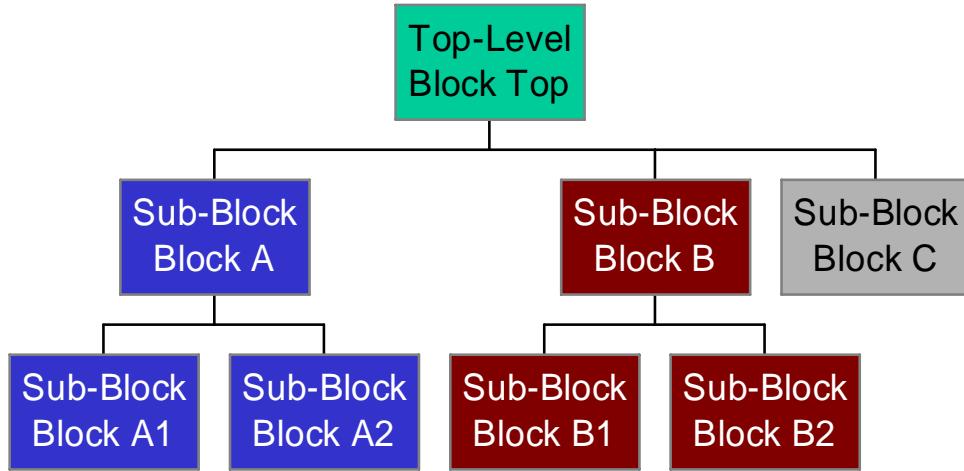


Hierarchy Leads to Greater Design Readability, Reuse, and Debug



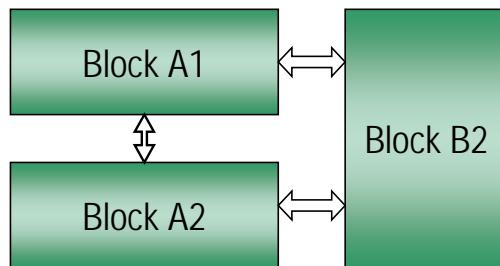
Using hierarchy leads to greater design readability, reuse, and debug. A *hierarchical* design is composed of several large blocks. Each block may be composed of smaller blocks, and so on. Designs that do not have this type of structure are called *flat* designs. We will use the term *block* in this module to mean a macro or symbol on a schematic, or an entity in VHDL, or a module in Verilog. One way to divide your design into hierarchical blocks is by logic type, as this drawing illustrates. You will see a bit later why this type of division is recommended for HDL designs.

Design Readability and Reusability are Benefits of Hierarchy



Design readability is a benefit of using hierarchy. It is easier to understand the design functionality and data flow, and it is easier to debug. Using hierarchy also makes it easy to reuse parts of a design. You have been introduced to some of the benefits of hierarchical design. Next, we will go into more detail on each benefit. One major benefit of hierarchical design, not mentioned here, is team-based design.

Follow Xilinx Design Readability Tips



Follow the Xilinx design readability tips. Choose hierarchical blocks that have logical data flow between blocks and a minimum of routing between blocks. Choose descriptive labels for blocks and signals. Keep clock domains separated, which makes the interaction between clocks very clear. Keep the length of each source file manageable, which makes it easier to read, synthesize, and debug. The file length recommendation is strictly for readability of your code. Your synthesis vendor may have additional recommendations on block size. Use descriptive labels. Good naming conventions help you locate trouble spots in your design during simulation and implementation. For example, *UI* for an instantiated component may not be as useful as a name such as *FIFO_CTRL*.

Follow Xilinx Design Reuse Tips

- Synchronous Design Techniques ✓
 - Partitioning hierarchy
 - Reliability
 - Ease of integration
- Readability and coding style ✓
 - Future modifications/enhancements
- Modular ✓
 - Scalable solutions provide flexibility for future use
- Verification & test bench ✓
 - Allow ample time and resources on the verification effort



Follow the Xilinx design reuse tips. Build a set of blocks, such as register banks, FIFOs, other standard functions, and custom functions commonly used in your applications that are available to all designers. Name blocks by function and target the Xilinx family, which will make it easy to locate the block you want. For example: REG_4X8_S3 (bank of four 8-bit registers targeting Spartan-3). Store in a directory that is separate from the Xilinx tools to prevent accidental deletion when updating tools.

Knowledge Check

- Can you list two benefits of hierarchical design?

*Raise your hand if you
think you know the
answer*



Answers

- . Two benefits of hierarchical design
 - Design readability
 - Design reuse



Outline

- Hierarchical Design
- · **Synchronous Design
for Xilinx FPGAs**
- Summary



Synchronous Circuits are More Reliable

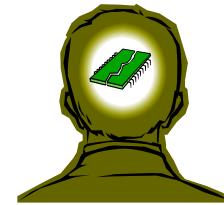
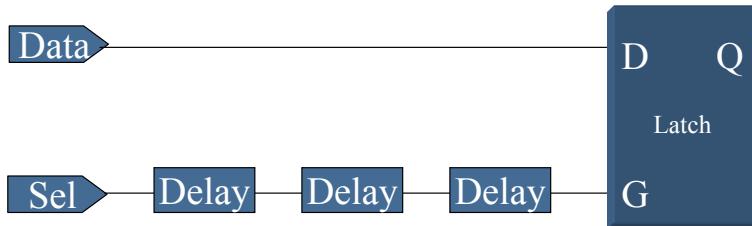
- A fully synchronous design uses only one clock and one clock edge
- Synchronous Design Methodology:
 - One clock, one edge (all FFs use rising or falling edge)
 - Use D-type flip-flops
 - Partition hierarchy along functional lines while minimizing the interaction of blocks
 - Register the outputs of each behavioral block
 - In place of multiple clocks - use clock enables
 - Synchronize asynchronous signals to the "single" clock (synchronization circuits)
 - Do NOT create:
 - . Gated, derived, or divided clocks
 - . Local asynchronous set/reset



Synchronous circuits are more reliable. Events are triggered by clock edges that occur at well-defined intervals. Outputs from one logic stage have a full clock cycle to propagate to the next stage. Skew between data arrival times is tolerated within the same clock period.

- Synchronous designs are inherently reliable
- Static timing analysis is fast and reliable
- Fewer clocks provide for easier design and debug
- High performance through pipelining
- Interface timing is more predictable

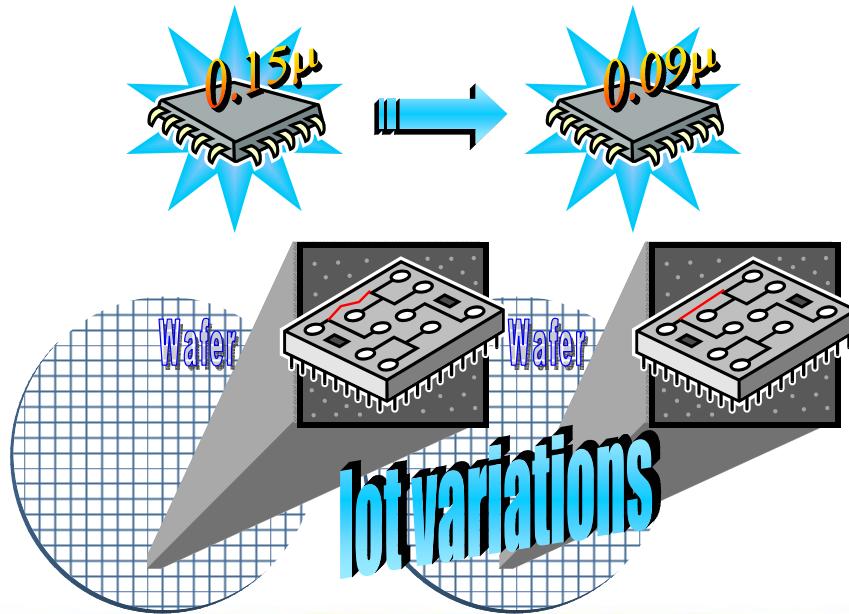
Asynchronous Circuits are Less Reliable



Asynchronous circuits are less reliable. Delay may need to be a specific amount (for example, 12 ns). Multiple delays may need to hold a specific relationship (for example, DATA arrives 5 ns before SELECT). For reliability reasons, Xilinx strongly recommends synchronous design. Asynchronous circuits may work correctly in one device but fail in another *identical* device due to normal variations of delays inside the chip.

Asynchronous Design

Case Studies



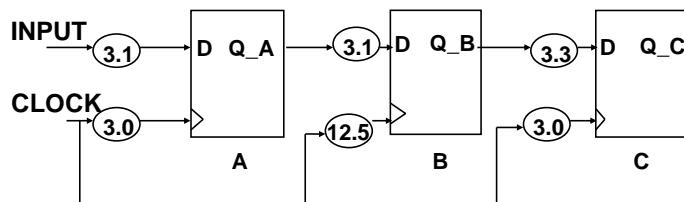
Synchronous Design Techniques 3a- 13

© 2006 Xilinx, Inc. All Rights Reserved

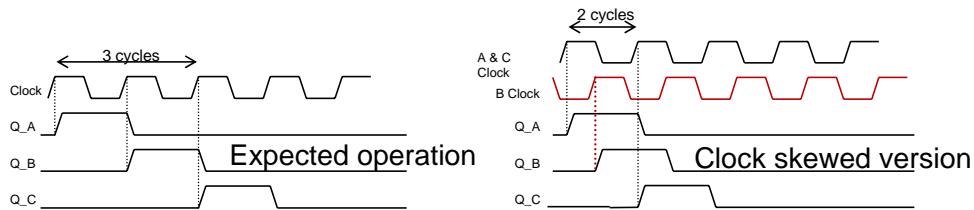


Let's review a case study. The design I created two years ago no longer works. What did Xilinx change in their FPGAs? SRAM process improvements and geometry shrinks increase speed. Normal variations between wafer lots. My design passes a timing simulation test but fails in-circuit. Is the timing simulation accurate? **YES**. Timing simulation uses worst-case delays. Actual board-level conditions are usually better.

The Clock Signal Controls Events in Synchronous Circuits

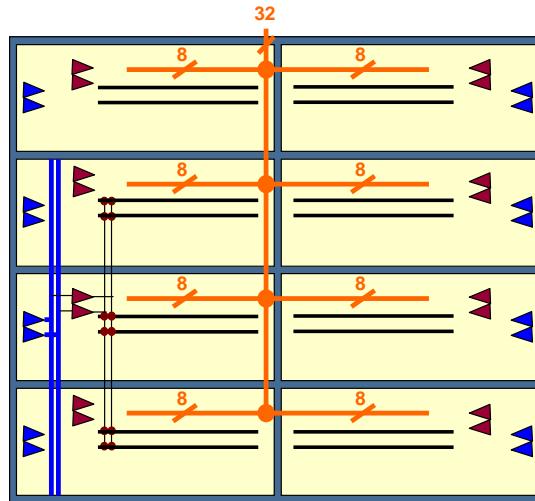
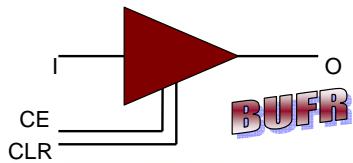
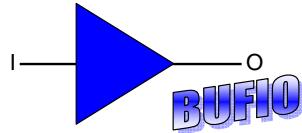
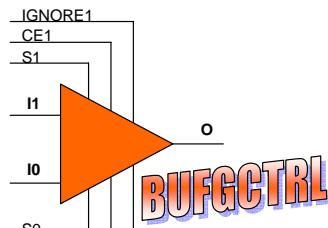


- This shift register will not work because of clock skew!



In synchronous circuits, events are controlled by the clock signal. If the clock arrives at different flip-flops at different times, your circuit may not function correctly. This is an example of a clock skew problem.

Use Dedicated Clock Buffers to Reduce Clock Skew



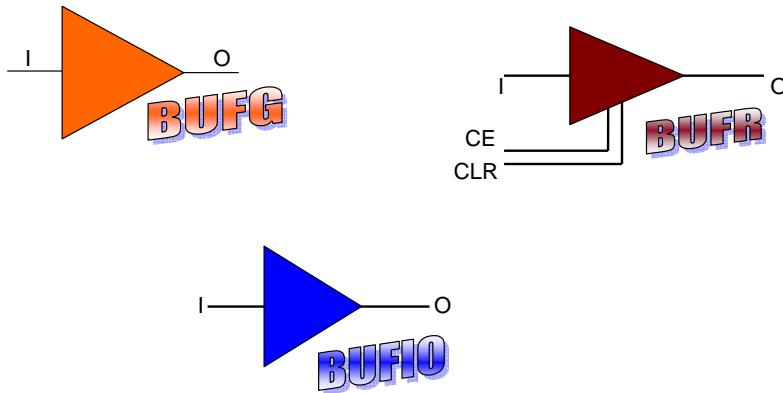
Synchronous Design Techniques 3a- 15

© 2006 Xilinx, Inc. All Rights Reserved



Global buffers are connected to dedicated routing. This routing network is balanced to minimize skew. All Xilinx FPGAs have global buffers. Virtex™-II and Virtex™-II Pro devices have sixteen BUFGMUXes. Spartan™-3 devices have eight BUFGMUXes. Virtex™-4 devices have 32 BUFGCTRLs (in orange), two BUFIO per clock region (in blue), and two BUFR per clock region (in maroon). You can always use a BUFG symbol, and the software will choose an appropriate buffer type. All major synthesis tools can infer global buffers onto clock signals that come from off-chip.

Using Clock Buffers



Most synthesis tools will automatically infer a BUFG on global input clocks. Clock signals must come from a top-level port. Clocks that you would like placed on the BUFIO or BUFR must be instantiated. Internally generated clocks are not automatically placed on a BUFG. The easiest way to design your clocking “core” is with Architecture Wizard. Note that these templates, and templates for other BUFGMUX configurations, are available in the ISE™ language templates. See the sample instantiation of BUFGMUX (Verilog) in your workbook.

Sample instantiation of BUFGMUX (Verilog)

```
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);

  component BUFGMUX
  port (I0 : in std_logic;
        I1 : in std_logic;
        S : in std_logic;
        O : out std_logic);
  end component;

  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      );
```

Knowledge Check

- Why should you use global buffers for your clock signals?

*Raise your hand if you
think you know the
answer*



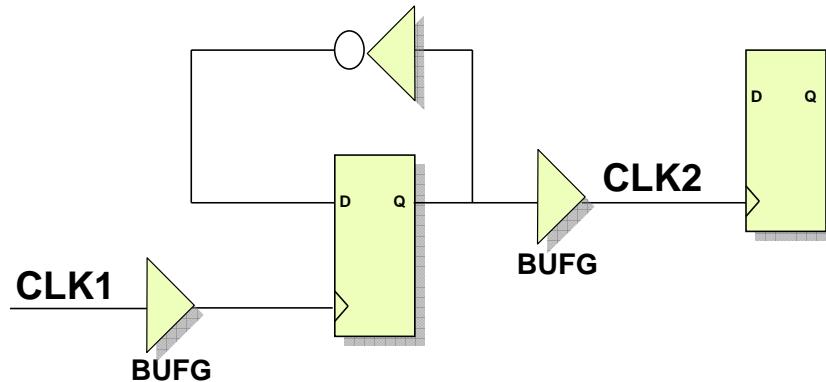
Answers

- Use global buffers for your clock signals to reduce clock skew



Traditional Clock Divider

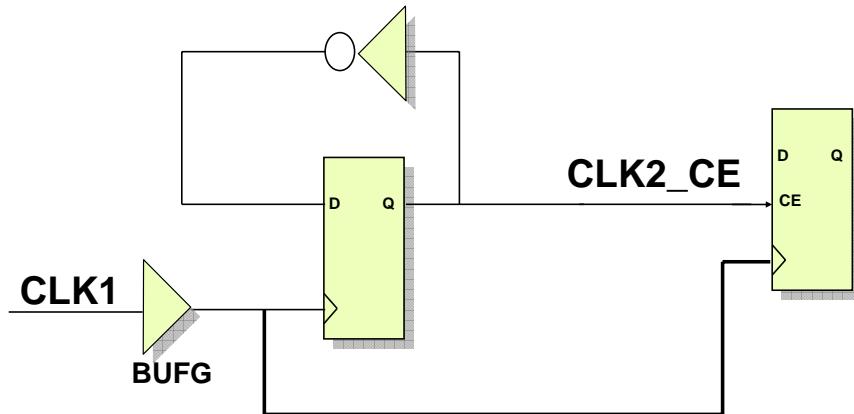
- Introduces clock skew between CLK1 and CLK2
- Uses an extra BUFG to reduce skew on CLK2



This is a typical circuit that divides CLK1 by two. This circuit has a potential problem: skew between CLK1 and CLK2. If this skew does not cause a problem in your design, it is okay to use this circuit.

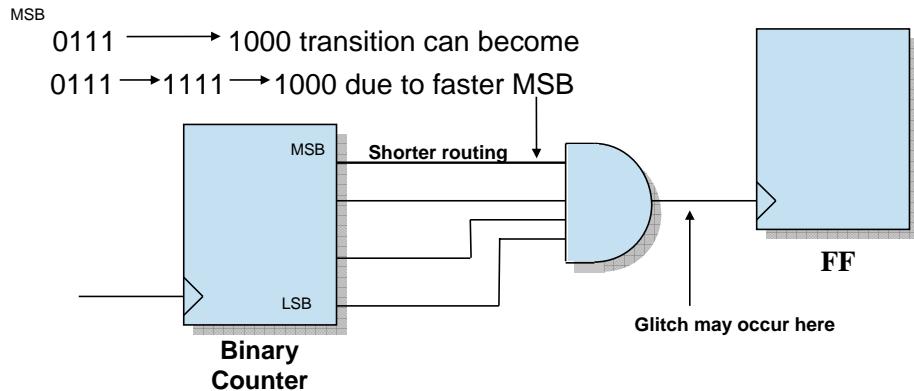
Recommended Clock Divider

- No clock skew between flip-flops



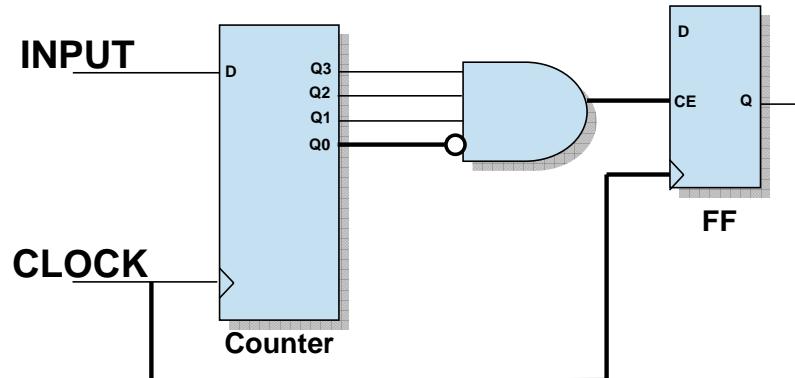
This implementation is very similar, but it eliminates the skew because all flip-flops are clocked with CLK1. Note, if you use a Virtex™-II or Spartan™-3 device, you can use the DCM to generate a divided clock with no skew. You can use the DLL to generate a divided clock with no skew.

Avoid Clock Glitches



Because flip-flops in today's FPGAs are very fast, they can respond to very narrow clock pulses. Never source a clock signal from combinatorial logic, which is also known as "gating the clock." Again, because it is such an important signal, you must avoid glitches on the clock. In the past, flip-flops were slower and would ignore brief glitches on the clock. The flip-flops of today are so fast that they can react to clock glitches that are less than 1 ns wide.

Avoid Clock Glitches



This circuit creates the same function, but it creates the function without glitches on the clock.

Notice two things:

1. The output of the AND gate is connected to the clock enable of the flip-flop instead of being connected to the clock pin.
2. The AND gate is decoding the 1110 state of the counter. The flip-flop will be enabled when the counter reaches 1111.

Coding Clock Enables

VHDL

```
FF_AR_CE: process(CLK)
begin
  if (CLK'event and CLK = '1') then
    if (ENABLE = '1') then
      Q <= D_IN;
    end if;
  end if;
end process
```

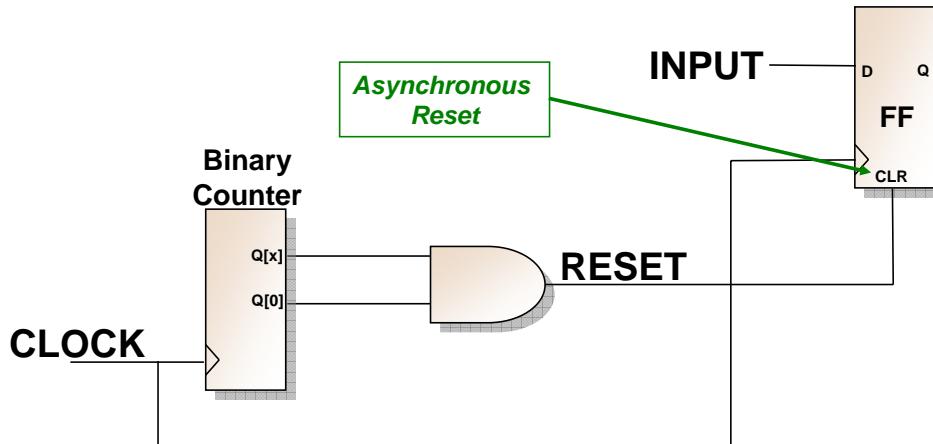
Verilog

```
always @(posedge CLOCK)
  if (ENABLE)
    Q = D_IN;
```



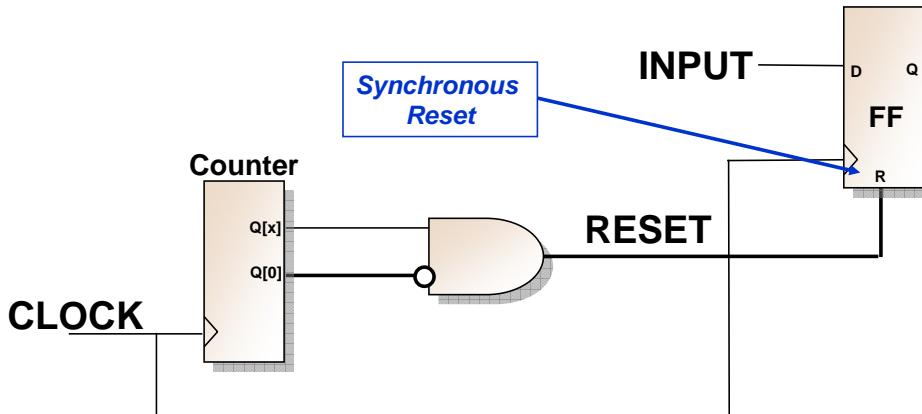
If ENABLE is not a top-level port, write the code for ENABLE in another process. This makes the code more readable and helps the synthesis tools create better netlists. The control signal precedence is Reset, Set, and Enable. Use this order in your code. Control signal precedence: If you do not follow the correct order of precedence in your HDL code, the synthesis tool may build extra logic. For example, if your code gives precedence to *set* over *reset*, the synthesis tool must add inverters to the D and Q pins of the flip-flop and swap the set and the reset signal connections.

Avoid Set/Reset Glitches



Glitches on asynchronous set and asynchronous reset inputs can lead to incorrect circuit behavior. Other than the clock, there may be other signals in your design that must be glitch-free. One example is asynchronous sets or resets. In this circuit, a glitch could occur on RESET during the $01 \rightarrow 10$ transition of the counter output ($01 \rightarrow 11 \rightarrow 10$).

Avoid Set/Reset Glitches



Convert to synchronous set and synchronous reset when possible. Convert to synchronous reset by placing a different library component on your schematic (FDR and FDS have synchronous control; FDC and FDP have asynchronous control) or by modifying your HDL code. Notice that the lower AND gate is decoding the 10 state of the counter instead of the 11 state; therefore, the flip-flop resets when the counter transitions to 11.

Coding Synchronous Flip-Flops

- Asynchronous Reset

```
always @(posedge CLOCK or posedge RESET)
if (RESET)
    Q = 0;
else
    Q = D_IN;
```

- Synchronous Reset

```
always @(posedge CLOCK)
if (RESET)
    Q = 0;
else
    Q = D_IN;
```



Asynchronous Reset (VHDL)
process(CLK, RESET)
begin
if (RESET = '1') then
 Q <= '0';
elsif (CLK'event and CLK = '1') then
 Q <= D_IN;
end if;
end process

Synchronous Reset (VHDL)
process(CLK)
begin
if (CLK'event and CLK = '1') then
 if (RESET = '1') then
 Q <= '0';
 else
 Q <= D_IN;
 end if;
end if;
end process

Knowledge Check

- What is an alternative to gating a clock?

*Raise your hand if you
think you know the
answer*



Answers

- An alternative to gating a clock...
 - Use a clock enable

Outline

- Hierarchical Design
- Synchronous Design for Xilinx FPGAs
- · Summary



Summary

- Proper use of the hierarchy aids design readability and debug
- Synchronous designs are more reliable than asynchronous designs
- FPGA design tips
 - Global clock buffers and DCMs eliminate skew
 - Avoid glitches on clocks by not building with asynchronous sets and asynchronous resets

Where Can I Learn More?

- Application notes
 - www.xilinx.com → Documentation → Application Notes
- Software Documentation: Development System Reference Guide
 - www.xilinx.com → Documentation → Software Manuals
- Software Documentation: Libraries Guide
 - www.xilinx.com → Documentation → Software Manuals
- Software Documentation: XST User Guide
 - www.xilinx.com → Documentation → Software Manuals
 - Or check the documentation for your synthesis tool
- Software Documentation: Synthesis and Simulation Design Guide
 - www.xilinx.com → Documentation → Software Manuals
- Answer Record database: Search by FPGA family or software tool
 - www.xilinx.com/support → Answer Browser (under Support Quicklinks)
 - www.xilinx.com/xlnx/xil_ans_browser.jsp





Implementation Options

© 2006 Xilinx, Inc. All Rights Reserved

NOTES

Objectives

After completing this module, you will be able to:

- Use the ISE™ GUI to access basic software options
- Describe the effects of the basic implementation options
- Access advanced software options



NOTES

Outline



- Basic Software Options
- Accessing Advanced Software Options
- Summary

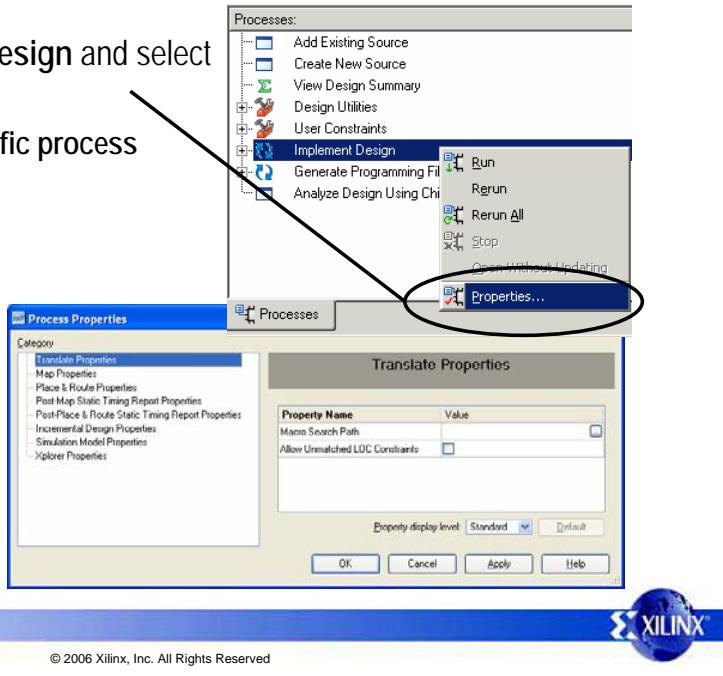


NOTES

Accessing Implementation Options

- Right-click **Implement Design** and select **Properties**

- Or right-click the specific process



- This dialog has six tabs

NOTES

Here we look at accessing the implementation options.

Open the **Process Properties** window from the Project Navigator menus:

- Click **Process → Properties**.
- If the Properties option is unavailable, ensure that **Implement Design** is selected in the Processes for Current Source window.

Create the Post-Map Static Timing Report*

- Expand the **Map** process, expand the **Post-Map Static Timing** process, and double-click **Post-Map Static Timing Report**.

*Create a Simulation Model

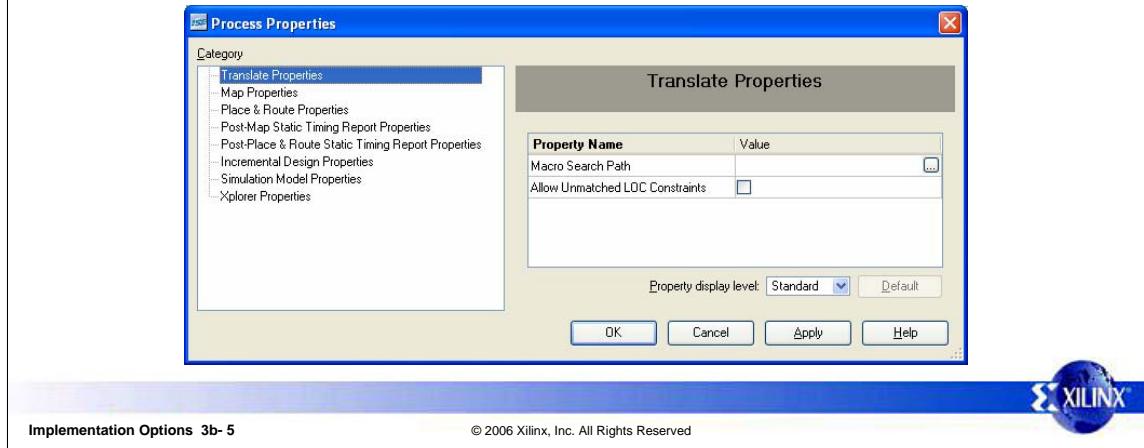
*The Incremental Design and Simulation Model processes are not run during a typical implementation.

- Click the **Place & Route Properties** tab, and check the **Generate Post-Place & Route Simulation Model** option.

To learn more about Incremental Design, refer to the “Incremental and Modular Design” module in the *Advanced FPGA Implementation* course.

Translate Properties

- Macro Search Path
 - Tells the tools where to search for netlists
- Allow Unmatched LOC Constraints
 - Ignores invalid LOC constraints instead of issuing an error



NOTES

Here is a look at the Translate properties. Use the Macro Search Path to enter a list of directories where netlists are located. Each directory needs to be separated by a semicolon (;).

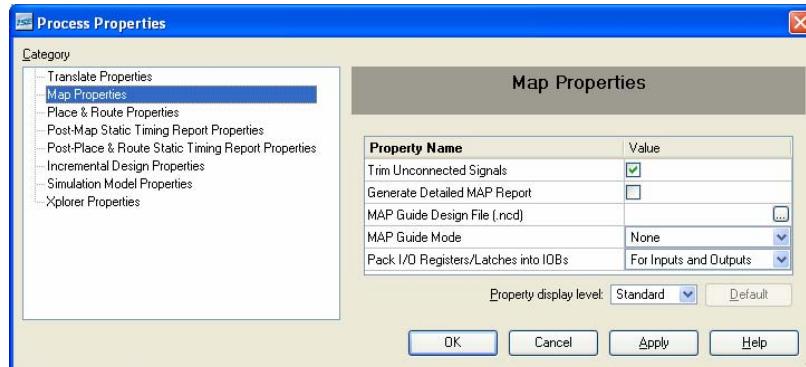
For example: *C:\mylibrary\netlists; D:\yourlibrary\netlists*

Several designers can use the Macro Search Path to work on different parts of the same design. The Macro Search Path also allows you to have your own library of netlists for commonly used components, such as cores.

Allowing Unmatched LOC Constraints is useful when you want to target a new device or package but have not created a new pinout yet.

Map Properties

- Trim Unconnected Signals
 - Uncheck for incomplete designs
- Generate Detailed MAP Report
- MAP Guide Design File
- MAP Guide Mode
- Use RLOC Constraints
- Pack I/O Registers/Latches into IOBs



Implementation Options 3b- 6

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

Here is a look at the Map properties. The Trim Unconnected Signals option is on by default. It will trim away any sourceless or loadless nets.

Pack I/O Registers/Latches into IOBs is an important option. By default, it will pack registers and latches into both the input and the output blocks. Keep this option as is; you save CLB resources and increase the speed of your data coming on and off of the device.

Knowledge Check

- Can you name two ways for accessing the properties for the Map process?

*Raise your hand if you
think you know the
answer*



NOTES

Answers

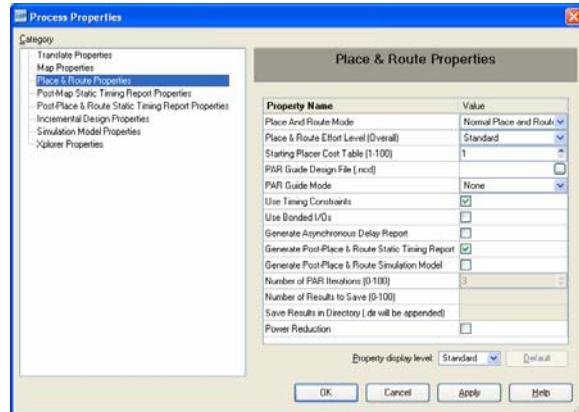
- Two ways for accessing the properties for the Map process
 - Right-click the **Implement Design** process, click **Properties**, and click the **Map Properties** tab
 - Expand the **Implement Design** process, right-click **Map**, and click **Properties**



NOTES

Place & Route Properties

- Place & Route Effort Level
- Starting Placer Cost Table
- Place & Route Mode
- PAR Guide File/Mode
- Use Timing Constraints
- Use Bonded I/Os
- Generate Asynchronous Delay Report
- Generate Post-Place & Route Static Timing Report
- Generate Post-Place & Route Simulation Model



Implementation Options 3b- 9

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

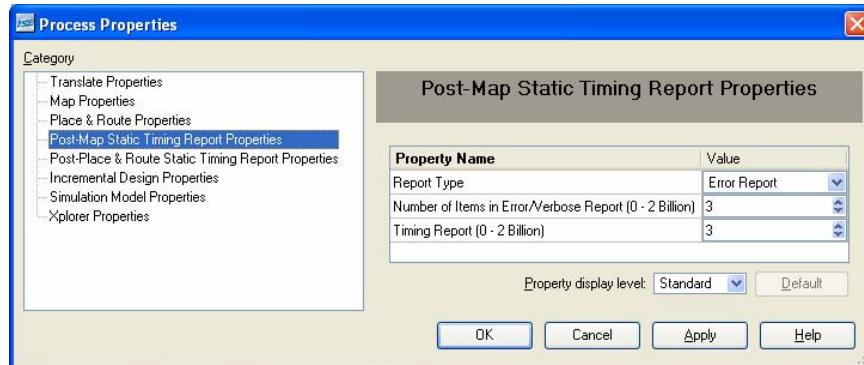
Here you see the Place & Route properties. The Place & Route Effort Level affects the placement time. *Standard* provides the fastest run time with the lowest effort—good for simple and small designs. *High* increases the effort level and incurs the longest run time.

The Starting Placer Cost Table ranges from 1 to 100. You can specify a starting point for placement. The “Advanced Implementation Options” module in the *Designing for Performance* course covers cost tables in greater detail.

Use the Use Bonded I/Os property to specify whether PAR can place internal I/O logic into bonded I/O sites in which the I/O pad is not used. Use this option when your design is very full, when it cannot complete placement or routing, or when it cannot meet your timing constraints. If you use this option, you must check the Pad Report to see which bonded I/O sites are used so that you can make sure those I/O pins are not connected to external, power, or ground signals.

Post-Map Static Timing Report Properties

- Report Type
- Number of Items in Error/Verbose Report
- Timing Report (Number of Items)



Implementation Options 3b- 10

© 2006 Xilinx, Inc. All Rights Reserved

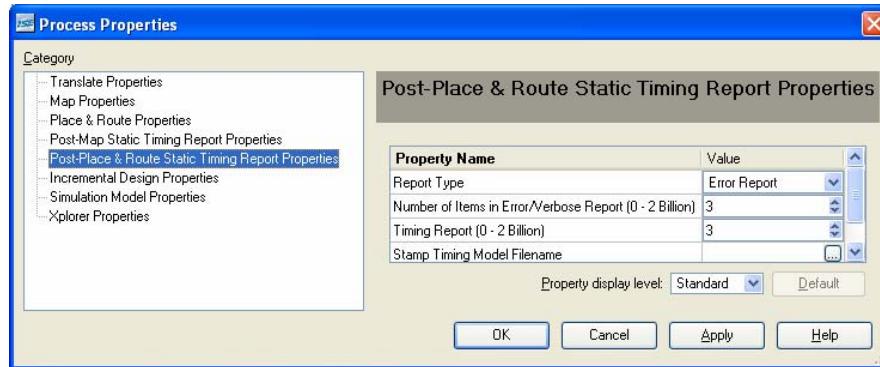


NOTES

The Post-Map Static Timing Report properties are shown here. These options are identical to the options in the Post-Place & Route Timing Report.

Post-Place & Route Static Timing Report Properties

- Report Type
- Number of Items in Error/Verbose Report
- Stamp Timing Model Filename
- Timing Specification Interaction Report file
- Timing Report



Implementation Options 3b- 11

© 2006 Xilinx, Inc. All Rights Reserved



NOTES

Here you see the Post-Place & Route Static Timing Report properties. Report type *Error* lists detailed delay information on timing errors and failed paths. *Verbose* lists detailed delay information for the longest paths covered by each constraint—even if the paths met the constraint.

The next two options on this tab are the same. These options control the number of paths listed in detail for each timing constraint. The default is 3.

Stamp Timing Model Filename: Specify the filename of the stamp file that you will use during post-route timing. TRACE generates the *stampfile.mod* and *stampfile.data* STAMP timing models file.

Note: The Timing Specification Interaction Report file is an obsolete option. Using this option will create a blank report file.

Knowledge Check

- What are some prominent options?

*Raise your hand if you
think you know the
answer*



NOTES

Answers

- Some prominent options
 - Translate: Macro search path
 - MAP: Pack registers into I/Os
 - PAR: Effort level, Starting cost table
 - Timing: Report types, Number of items reported



NOTES

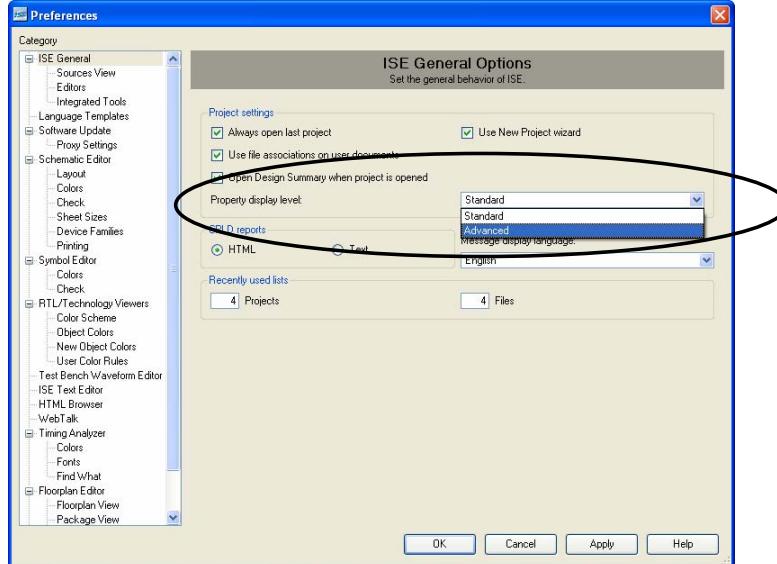
Outline

- Basic Software Options
- • **Accessing Advanced Software Options**
- Summary



NOTES

Viewing Advanced Options: Globally



Implementation Options 3b- 15

© 2006 Xilinx, Inc. All Rights Reserved

NOTES

Demo

Click **Edit → Preferences** in the ISETM Project Navigator

Select a Property display level of **Advanced**

All Process Property dialogs will now display all possible software options

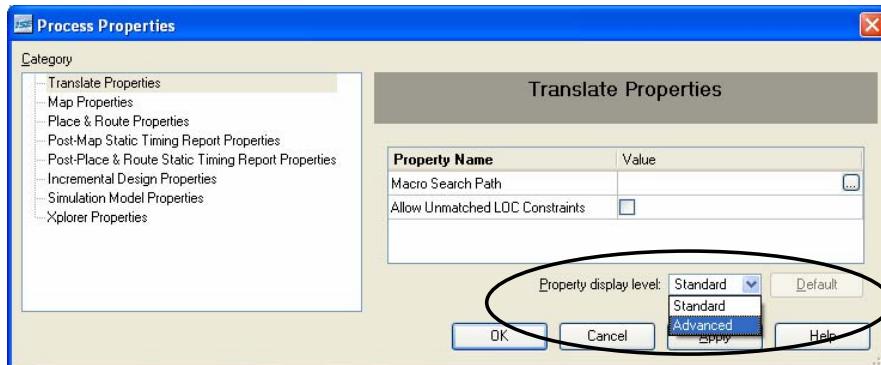
You can also select the display level in individual properties dialogs

Demo

Follow the directions listed above to change from the Standard process properties to the Advanced properties.

Open a properties dialog and point out the **Property display level** option at the bottom of the window.

Viewing Advanced Options



Implementation Options 3b- 16

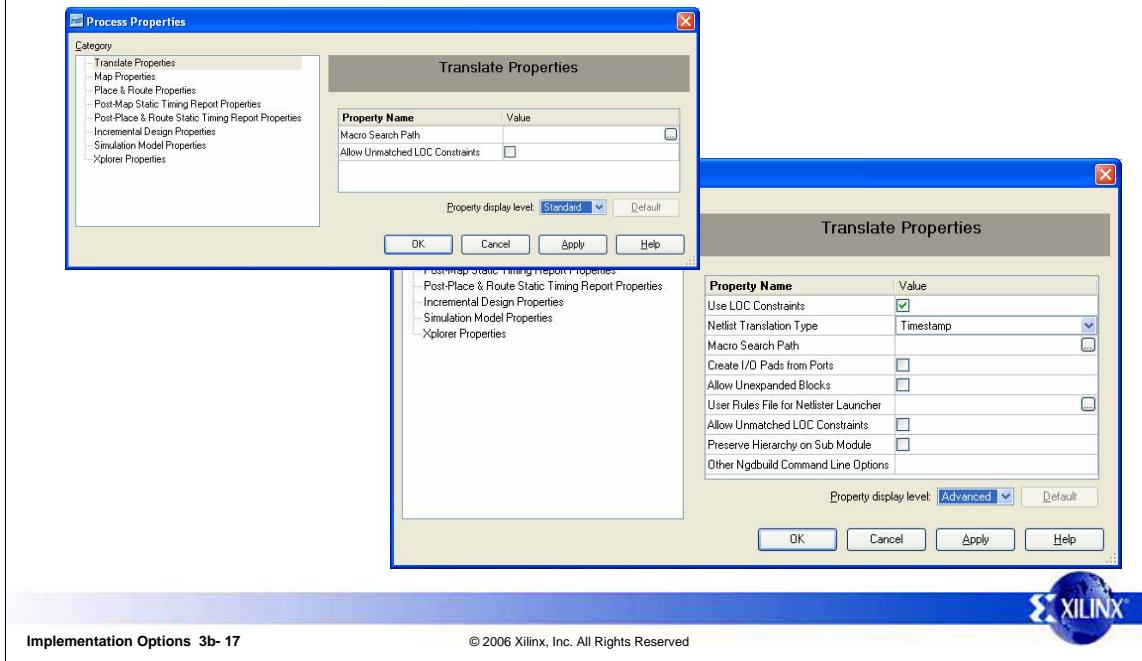
© 2006 Xilinx, Inc. All Rights Reserved



NOTES

You can also access advanced Process properties directly from the Process Properties window. In the lower-right corner of the window for Property display level, select **Advanced** in the drop-down box.

Standard Versus Advanced Properties



NOTES

The Advanced properties provides additional options for each of the implementation phases. For example, sampleadvanced translate option allows Unexpanded Blocks (for implementing partially completed designs).

Sample Advanced MAP option: Timing-Driven Packing. Sample Advanced PAR option offers Extra Effort Level, and the Sample Advanced Timing Report option provides a report on Unconstrained Paths.

Knowledge Check

- How are the Advanced process properties accessed?

*Raise your hand if you
think you know the
answer*



NOTES

Answers

- Access the Advanced process properties:
 - In the ISE™ Project Navigator, click **Edit → Preferences** and set the Property display level to **Advanced**
 - Or select the **Property display level** in a properties dialog



NOTES

Outline

- Basic Software Options
- Accessing Advanced Software Options
- · Summary



NOTES

Summary

- Software options are easy to access
- Software options are also called properties
- Basic or *standard* properties are the most commonly used options
- View Advanced properties to access all of the software options



NOTES

Where Can I Learn More?

- Online Help
 - Click the Help button on any Properties dialog box
 - . Help pages show Standard and Advanced options
- Software Documentation: Development System Reference Guide
 - www.xilinx.com → Documentation → Software Manuals



NOTES

You can learn more about implementation options with online help. Click the **Help** button on any Properties dialog box. The Help pages will show Standard and Advanced options. You may also check out the Development System Reference Guide software documentation on www.xilinx.com.



Implementation Options Lab

Introduction

©2006 Xilinx, Inc. All Rights Reserved

NOTES

Objectives

After completing this lab, you will be able to:

- Use I/O configuration options in PACE to improve design performance
- Adjust process properties to improve the design performance



NOTES

Introduction

- This lab introduces the implementation process properties and the PACE I/O configuration options
- You will modify process properties and I/O configuration options to improve the design performance



NOTES

General Flow

- Step 1: Analyze the design performance
- Step 2: Increase the Place & Route effort
- Step 3: Change the I/O configuration options
- Step 4: Implement the design and analyze the timing



NOTES

Implementation Options Lab

Implementation Options Lab

Introduction

Global timing constraints have been applied to the lab design. In this lab, you implement the design with default software options, and you evaluate the design performance versus constraints. You then change the software options (also called *process properties*) and I/O configuration options to improve the performance of the design.

Objectives

After completing this lab, you will be able to:

- Adjust process properties to improve the design performance
- Use I/O configuration options in PACE to improve the design performance

Procedure

You will implement the project using the constraints file provided (*myucf.ucf*) and the default software options. Then you will improve the design performance by adjusting the process properties and the I/O configuration options.

This lab comprises four primary steps: You will analyze the design performance, increase the Place & Route effort, change the I/O configuration options, and, finally, implement the design and analyze the timing.

For each procedure within a primary step, there are general instructions (indicated by the  symbol). These general instructions only provide a broad outline for performing the procedure. Below these general instructions, you will find accompanying step-by-step directions and illustrated figures that provide more detail for performing the procedure. If you feel confident about completing a procedure, you can skip the step-by-step directions and move on to the next general instruction.

Note: If you are not using **Toolwire** to perform this lab, all software programs, files, and projects will be located on the C:\ drive instead of R:\.

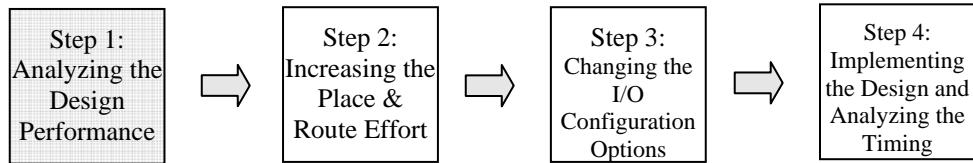
Note: If you are unable to complete the lab at this time, you can download the original lab files for this module from the Xilinx FTP site at ftp://ftp.xilinx.com/pub/documentation/education/fpga13000-82-xlnx_lab_files.zip. These are the original lab files and do not contain any work you may have previously completed.

Analyzing the Design Performance

Step 1

For each procedure within a primary step, there are general instructions (indicated by the ➤ symbol). These general instructions only provide a broad outline for performing the procedure. Below these general instructions, you will find accompanying step-by-step directions and illustrated figures that provide more detail for performing the procedure. If you feel confident about completing a procedure, you can skip the step-by-step directions and move on to the next general instruction.

General Flow for this Lab:



- Open the project in the *R:\training\fund\labs\impl_opt* or *R:\training\fund\labs\impl_opt_s3* directory.
- ❶ Select **Start → Programs → Xilinx ISE 8.2i → Project Navigator**
 - ❷ In the Project Navigator, select **File → Open Project**
The Open Project window opens.
 - ❸ Browse to one of the following directories: *R:\training\fund\labs\impl_opt* (Virtex™-4) or *R:\training\fund\labs\impl_opt_s3* (Spartan™-3)
 - ❹ Double-click *impl_opt.ise* (Virtex-4) or *ImpOpt.ise* (Spartan-3)
- Open the Post-Place & Route Static Timing report, and answer Question 1 (on the following page).
- ❶ In the Sources in Project window, select *ch_fifo.ngc*
 - ❷ In the Processes for Source window, expand the **Implement Design** process, and expand the **Place & Route** process
 - ❸ Expand the **Generate Post-Place & Route Static Timing** process

- ④ Double-click **Analyze Post-Place & Route Static Timing** (Figure 3d-1)



Figure 3d-1. Post-Place & Route Static Timing Report



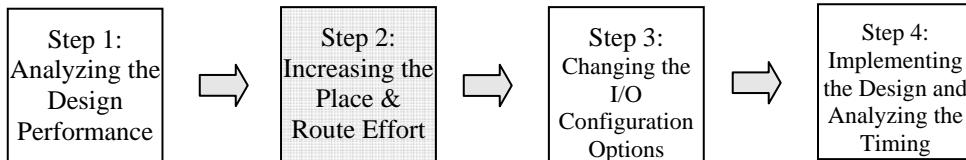
1. How many timing constraints failed? How many paths failed to meet their constraint (number of timing errors)? What is the Timing Score for this implementation? How can you use this information?

- ⑤ Exit the **Timing Analyzer**

Increasing the Place & Route Effort

Step 2

General Flow for this Lab:



Because the design does not meet timing constraints, first try increasing the Place & Route effort level. Increase the Place & Route Effort Level to *High* and run the Place & Route process again.



2. Why are we increasing the Place & Route effort level as the first option? What else could be done to optimize the timing?

- ① In the Processes for Source window, right-click **Place & Route**, and select **Properties** from the menu
- ② In the Process Properties dialog box, click the **drop-box next to Place & Route Effort Level (Overall)**
- ③ Click the **arrow** and select **High** (Figure 3d-2)

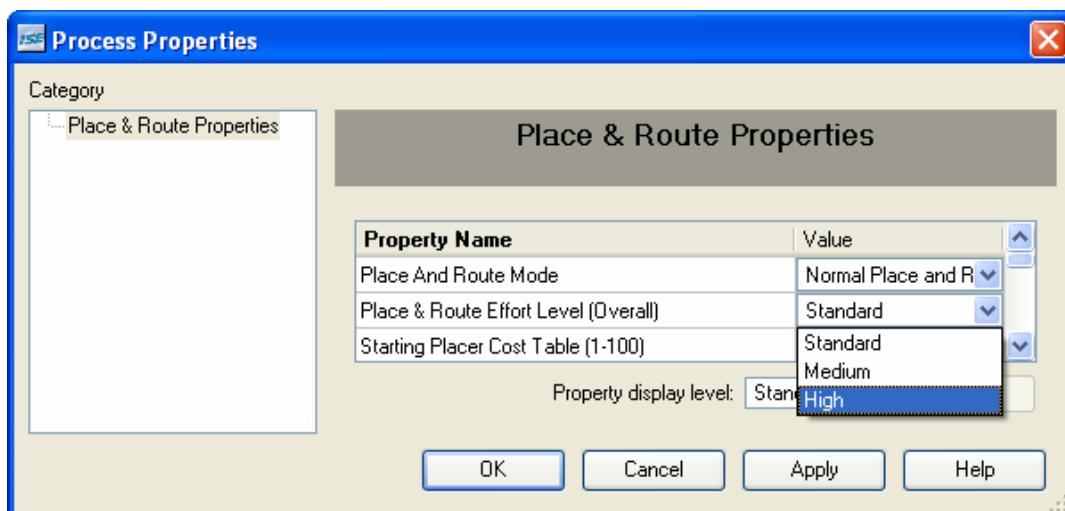


Figure 3d-2. Process Properties

④ Click OK

The green check mark next to the Place & Route process changes to a question mark. The question mark indicates that the process must be run again to bring the design up to date.

► Open the Post-Place & Route Static Timing Report and answer the questions below.

- ❶ In the Processes for Source window, expand the **Implement Design** process, and expand the **Place & Route** process
- ❷ Expand the **Generate Post-Place & Route Static Timing** process
- ❸ Double-click **Analyze Post-Place & Route Static Timing**



3. How many timing constraints failed? How many paths failed to meet their constraint (number of timing errors)? What is the Timing Score for this implementation? Did the increased effort level improve the implementation results?
-
-



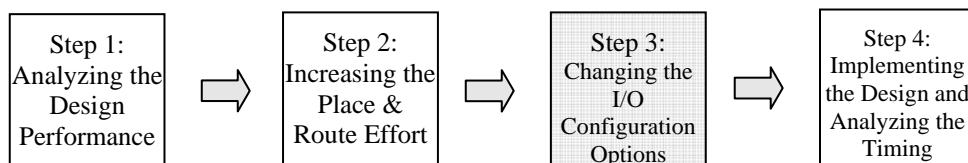
4. Which specific paths for the Offset Out constraint are not being met?
-
-

④ Exit the Timing Analyzer

Changing the I/O Configuration Options

Step 3

General Flow for this Lab:

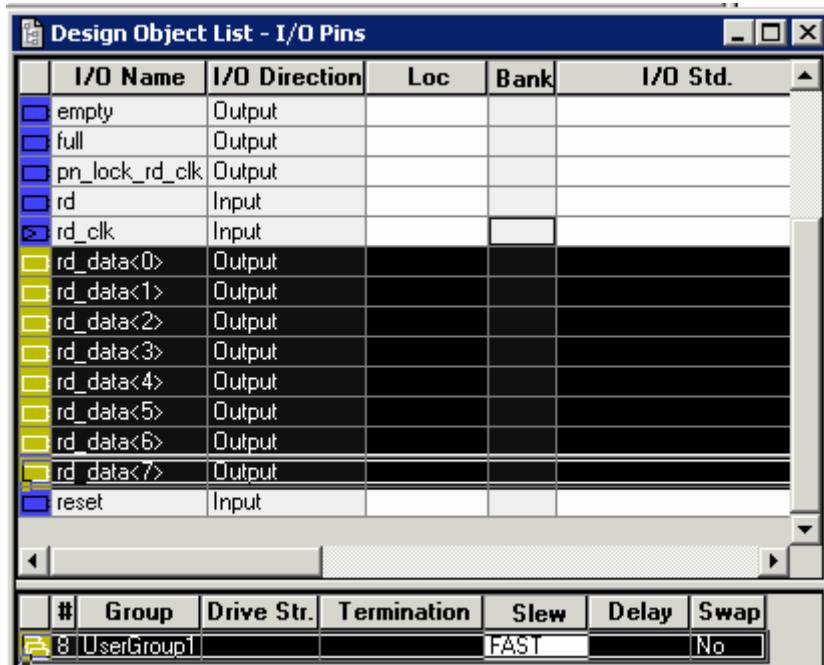


Because the OFFSET OUT constraint is not being met, increase the slew rate on these paths. Use PACE to set the output slew rate on the `rd_data[7:0]` outputs to FAST.

- ① In the Processes for Source window, expand **User Constraints**, and double-click **Assign Package Pins**
- ② In the Design Browser, select **I/O Pins**
- ③ In the Design Object List window, select **rd_data<0>** and then use shift-click to select **rd_data<7>**
- ④ Right-click the selected **rd_data I/O** and select **Group**

Grouping I/O pins together can make it easier to assign attributes.

- ⑤ For the UserGroup1 group in the Design Object List window, scroll to the right and click the **Slew column** in the drop-down list, and select **FAST** (Figure 3d-3). Press <**Enter**> to confirm the new slew rate



The screenshot shows the 'Design Object List - I/O Pins' window. The main table lists various I/O pins with their names, directions, locations, banks, and standard settings. The 'rd_data' pins (rd_data<0> through rd_data<7>) are grouped together. Below the main table is a smaller table for the UserGroup1 entry, where the 'Slew' column is set to 'FAST'. The 'Group' column shows 'UserGroup1'.

I/O Name	I/O Direction	Loc	Bank	I/O Std.
empty	Output			
full	Output			
pn_lock_rd_clk	Output			
rd	Input			
rd_clk	Input			
rd_data<0>	Output			
rd_data<1>	Output			
rd_data<2>	Output			
rd_data<3>	Output			
rd_data<4>	Output			
rd_data<5>	Output			
rd_data<6>	Output			
rd_data<7>	Output			
reset	Input			

#	Group	Drive Str.	Termination	Slew	Delay	Swap
8	UserGroup1			FAST		No

Figure 3d-3. Setting FAST Slew Rate

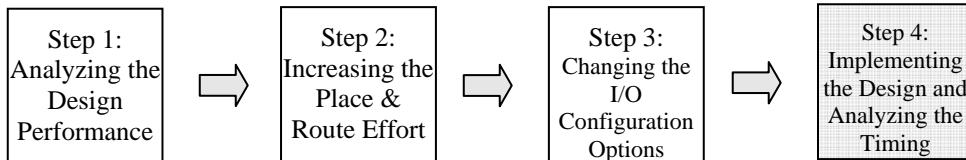
- ⑥ Click **File → Save** to save the file
- ⑦ Exit **PACE**
- ⑧ In the Processes for Source window, expand **User Constraints**, and double-click **Edit Constraints (Text)** to view the new slew rate constraints



You must see the text **SLEW = FAST** constraint for each of the **rd_data** signals. If you do not see this text, repeat Steps 1-7. Ensure that you press <**Enter**> after selecting the FAST slew rate.

Implementing the Design and Analyzing the Timing Step 4

General Flow for this Lab:



Implement the design with the new slew rates, and view the Post-Place & Route Static Timing report to see whether the design meets constraints.

- ① In the Source window, select **ch_fifo.ngc**. In the Processes for Source window, right-click **Analyze Post-Place & Route Static Timing**, and select **Rerun All**

This will automatically implement the design and open the Post-Place & Route Static Timing Report.



5. Does this design meet all of the timing constraints?
-
-



If timing is still not met, the Project Navigator did not rerun the Translate step.

Workaround: Right-click **Post-Place & Route Static Timing Report**, and select **Rerun All**.

- ② Exit the **Timing Analyzer**
- ③ Exit the **Project Navigator**

Conclusion

In this lab, you evaluated the design with the default options and global timing constraints. Because the design was not meeting timing, you increased the Place & Route effort level to improve the performance of the design.

By reviewing the Post-Place & Route Static Timing Report, you determined that the output delays were the primary cause of the failing paths. By increasing the slew rate to FAST, the design was able to improve the design performance further. The design was able to meet timing constraints with these changes.

A

Answers

Lab answers represent sample solutions only. Your results may differ, depending on the version of the software, service pack, or operating system that you are using.

1. How many timing constraints failed? How many paths failed to meet their constraint (number of timing errors)? What is the Timing Score for this implementation? How can you use this information?

Virtex™-4 device: Failed constraints: 2; Timing errors: 18; Timing Score: 5757

Spartan™-3 device: Failed constraints: 1; Timing errors: 8; Timing Score: 6768

This information is used to help determine the next option to use in meeting timing.

2. Why are we increasing the Place & Route effort level as the first option? What else could be done to optimize the timing?

You are attempting this first because the design is relatively close to meeting timing. If you can meet timing without having to change the design, this might be a useful alternative.

You could try many different things instead of simply increasing the Place & Route effort level. For example, you could pipeline the design, optimize logic in synthesis, and optimize logic in the code. These options are covered in more detail in the next course – *Designing for Performance*.

3. How many timing constraints failed? How many paths failed to meet their constraint (number of timing errors)? What is the Timing Score for this implementation? Did the increased effort level improve the implementation results?

Virtex-4 device: Failed constraints: 2; Timing errors: 14; Timing Score: 5626

Spartan-3 device: Failed constraints: 1; Timing errors: 8; Timing Score: 6768

In the Virtex-4 design, it helped reduce the overall timing score – even if only a little, but it also reduced the number of timing errors and failed constraints. In the Spartan-3 design, this option had no effect.

4. Which specific paths for the Offset Out constraint are not being met?

Looking at the path details, the FIFO driving the *rd_data* bus is the problem.

5. Does this design meet all of the timing constraints?

This design now meets all constraints. There are zero timing errors and the Timing Score is also zero.



Implementation Options Lab

Review

©2006 Xilinx, Inc. All Rights Reserved

NOTES

Summary

- It is easy to access and change the implementation options
- The Implementation options and I/O configurations can help improve your design performance



NOTES

Detailed Design Descriptions for Labs

Detailed Design Descriptions for Labs

Design Description (correlate_and_accumulate)

This design was created to take four input channels of serial signed data clocked at one rate (*wr_clks*) and correlate that data with a PN code. After correlation is achieved, the data is stored in a FIFO to cross-clock domains from the input clock to an internal clock (*rd_clk*).

The data is stored as 8-bit words (or chips). After a PN lock has been achieved, the data is read out of the FIFO and propagated to the output as *final_data*. Finally, the data for each channel is multiplied and accumulated. After 255 chips have been multiplied and accumulated, the data is sent out of the FPGA as *mac_cha*, *mac_chb*, *mac_chc*, and *mac_chd*. The top-level block diagram is shown in **Figure 4-1**.

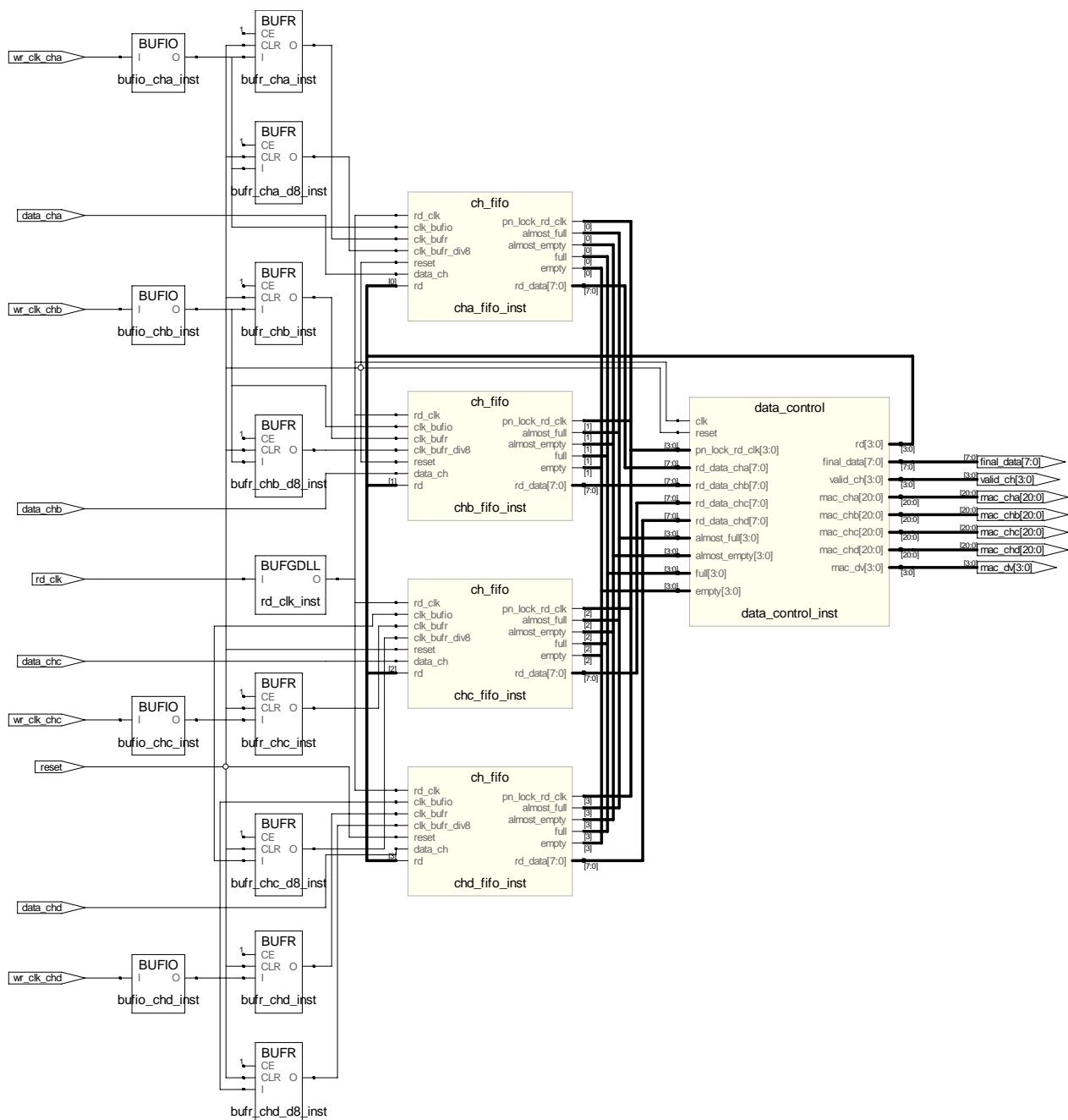


Figure 4-1. Correlate and Accumulate Block Diagram

Channel FIFO (ch_fifo)

There are four **ch_fifo** blocks, as shown in **Figure 4-1**. The **ch_fifo** block is shown in **Figure 4-2**. The FIFO for each channel uses four primary blocks: **pn_correlator**, **FIFO16 primitive**, **pn_lock_wr2rd**, and **flags_wr2rd**.

The **pn_correlator** attempts to find a correlation value (PN code) in a specific location in the data stream. The **pn_correlation** block in the **pn_correlator** (**Figure 4-3**) looks for a PN code match in the input data stream. When a PN code is found, the **pn_correlation_fsm** begins to generate an output wr enable signal for the write port of the FIFO. This port uses the slower **wr_clk**. After a PN code is found, every eight clock cycles (8-bit sample or chip), the sample is stored in memory until it is time to acquire the PN code again.

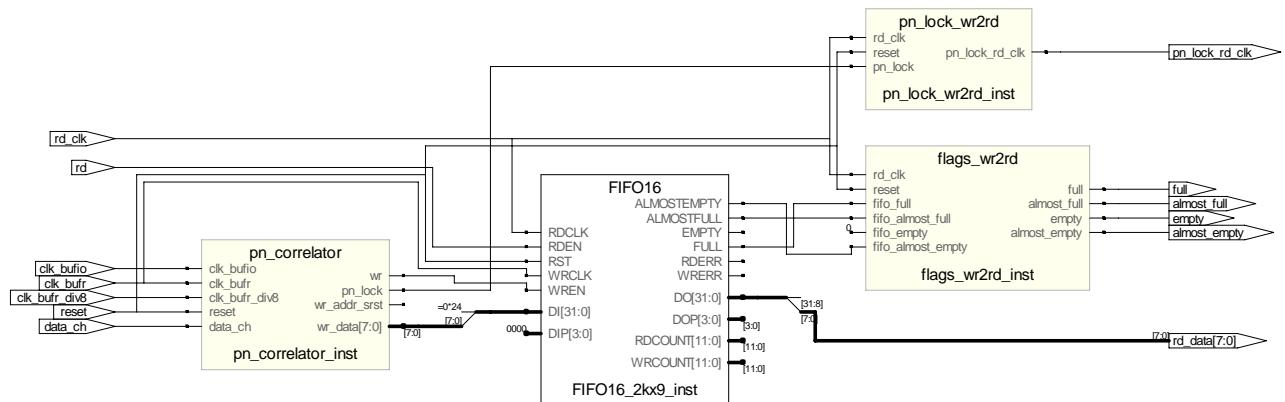


Figure 4-2. Channel FIFO Block

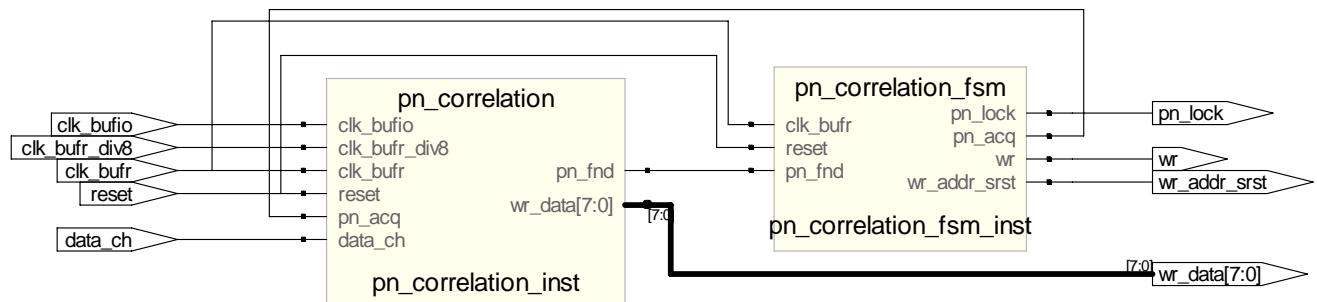


Figure 4-3. PN Correlator

Data Control (data_control)

The **data_control** block (Figure 4-4) reads data from the FIFOs and propagates that data to the **data_output_mux** and **MAC** blocks. The **data_output_mux** block multiplexes the valid data to the outputs of the FGPA (**final_data(7:0)**). The **MAC** block multiplies and accumulates the data from the FIFO. The **MAC** block uses a constant coefficient of -30 as the multiplicand. After each channel has been multiplied and accumulated for 255 chips, the **MAC** value for each **mac_ch** block becomes active.

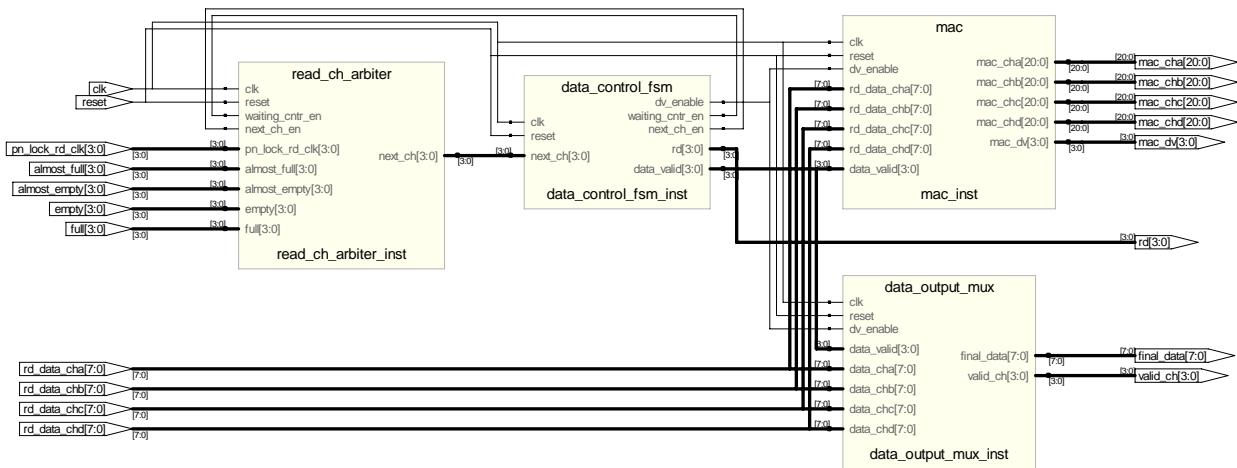


Figure 4-4. Data Control Block

The **data_control_fsm** block controls the reading of data from each **ch_fifo** block and drives the **data_valid** signals driving the **data_output_mux** and the four **mac_ch** blocks in the **MAC** block (Figure 4-6). The **data_control_fsm** block reads from each channel no more than every eight **rd_clks**. Therefore, there are several multi-cycle paths.

The **data_control_fsm** block functions are shown in Figure 4-5. From this state machine diagram, you can see that the signal **rd** is only active once every eight clock cycles. This is part of the reason that there are several multi-cycle paths in this design.

The **read_ch_arbiter** block (Figure 4-4) arbitrates the channel to read from next. It takes into account which channels are *locked*, how long each locked channel has been waiting to be read from, and the flags **full**, **empty**, **almost_full**, and **almost_empty** from the **FIFO16 primitive** (Figure 4-2).

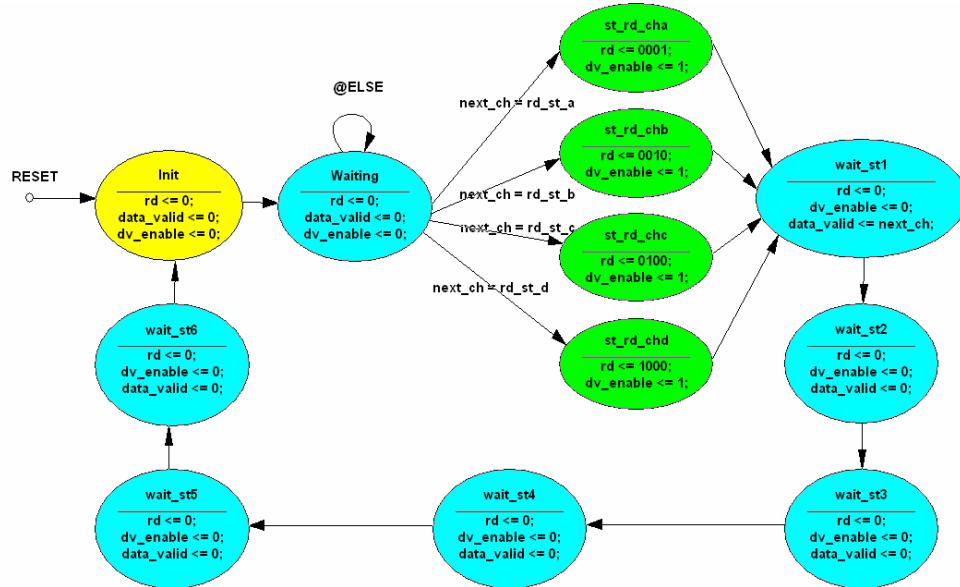


Figure 4-5. Data Control FSM

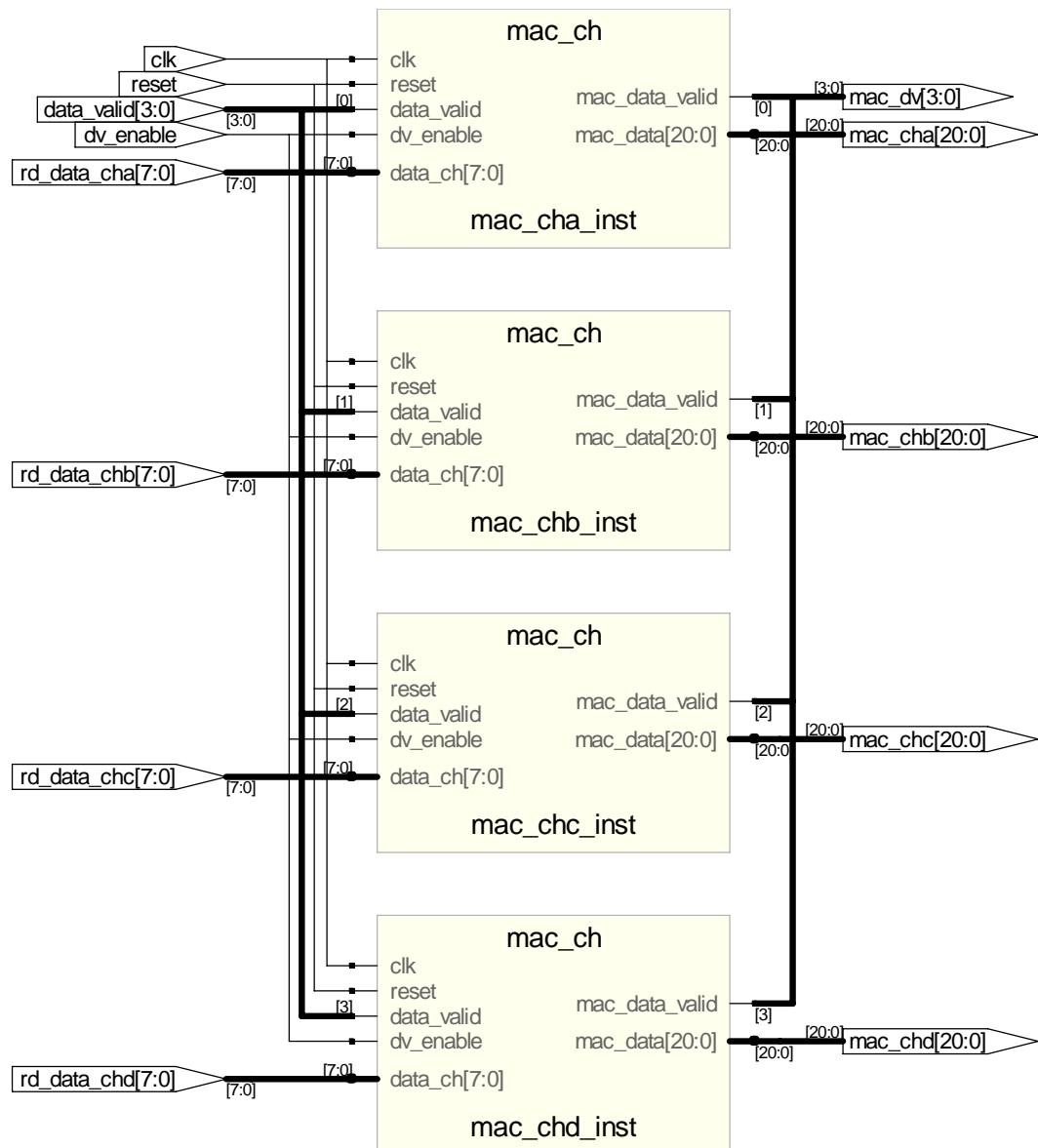


Figure 4-6. Multiply and Accumulate (MAC) Block

Appendices

Troubleshooting Tips

Troubleshooting Tips

Introduction

This is a list of common mistakes made in designs. These mistakes may make your design unreliable or slow. To increase your performance and increase reliability, make sure that your design passes all of these checks.

Reliability

- Using global clock buffers (BUFG) for clock signals.
 - Clocks that do not use global clock buffers will introduce skew.
 - Local clocks can use the BUFIO or BUFR resources in the Virtex™4 FPGA.
- Using only one clock edge for registering data.
 - Using both clock edges is unreliable because one or both clock edges may “drift.” If the clock does drift and you are using only one clock edge, you reduce the risk of a drifting clock edge.
 - This problem can be avoided by allowing the DCM to correct automatically the duty cycle on the clock for 50 percent duty cycle. Otherwise, using only one clock edge is highly recommended.
- No internally generated clock signals except those generated by the DCM.
 - This includes generating gated and divided clocks.
 - Instead, create clock enables or use the DCM to generate different clock signals.
 - For a purely synchronous design, using only one clock, whenever possible, is recommended.
- No internally generated asynchronous control signals, such as reset or set signals.
 - Internally generated asynchronous control signals will glitch.
 - Instead, generate a synchronous set or reset decoded for one clock cycle before it should be applied.
- No multiple clocks without a phase relationship.
 - You may not always be able to avoid this situation; in which case, make sure you are crossing clock domains with a proper synchronization circuit.

- Properly constrain multiple clocks with a phase relationship.
 - Again, you may not always be able to avoid this situation. In fact, many designs may require this. In this situation, make certain that you properly constrain the paths that cross clock domains.
- No internal latches.
 - Internal latches confuse the timing and often introduce another clock signal.
 - Internal latches are essentially considered combinatorial logic when transparent (gate is open), but are a synchronous element when the gate is latched, which confuses the timing analysis.
 - Often, internal latches introduce a gated clock. Gated clocks will glitch, making the design unreliable.

Performance

- Logic-level delays do not exceed 50 percent of the timing budget.
 - The logic-level delays can be found by looking in the logic-level timing report or the post-layout timing report for each path. After the detailed analysis of each path, the Timing Analyzer will generate statistics on the path delays. Look at the total logic-level delays. Does it exceed 50 percent of your timing budget?
- IOB registers.
 - IOB registers provide the fastest clock-to-output delays and input-to-clock delays.
 - There are a few limitations. For input registers, no combinatorial logic can exist between the pad and the register. For output registers, no combinatorial logic can exist between the register and the pad. For 3-stated outputs, all registers in the IOB must use the same clock and reset signals. Also, note that the IOB 3-state register must be active-low to be placed in the IOB (the 3-state buffer is active low with no inverter between the register and the 3-state buffer).
 - You must enable the use of IOB registers. Setting the global implementation options to use IOB registers for inputs, outputs, or both can do this. The default value is *off*.
 - You may also enable the use of IOB registers in your synthesis tool or by specifying in the User Constraint File (UCF). The syntax is: *INST <io_register_name> IOB = TRUE*;
- Fast slew rate on critical outputs.
 - Some I/O standards support a slew rate adjustment (check your data sheet). The fast slew rate decreases the output delay at the expense of increased ground bounce. Therefore, you should use the fast slew rate on a highly selective basis.
 - Never use fast slew rate carelessly—only on output paths having trouble meeting timing constraints. Fast slew rate makes your circuit more susceptible to ground bounce.
- Pipelined logic.
 - If your design allows you to add latency, pipeline the combinatorial logic to increase performance.
 - Xilinx FPGAs are register rich. For every four-input function there is a register. Use those registers to increase throughput at the expense of latency.

- Code optimized for four-input LUT structure.
 - Remember that each LUT can create a four-input combinatorial function. If you need a bigger function, note the number of LUTs that will be required to implement that function.
- Case statements rather than if-then-else statements.
 - Complex if-then-else statements will, generally, create priority-encoded logic, which increases the combinatorial delays on those paths.
 - Case statements used to generate complex logic will, generally, produce parallel logic that will not incur as much delay. For Verilog users, use the compiler directive *synopsys parallel_case*.
- One or more CORE Generator™ software blocks.
 - CORE Generator software blocks are optimized for the Xilinx architecture. Many blocks are customizable in terms of size, width, and pipeline delays.
 - Look at the critical paths in your design. Could you create a core in the CORE Generator software to improve the performance of your critical paths?
- FSM in its own level of hierarchy.
 - To allow the synthesis tool to fully optimize your FSM, the FSM should be in its own block. If this is not true, the synthesis tool can optimize logic around it with the FSM logic.
 - The FSM should not include any arithmetic logic, datapath logic, or other combinatorial functions unrelated to the state machine.
- FSM using two processes or always blocks.
 - Next-state decode logic and output decode logic should be placed in separate processes or always blocks to prevent the synthesis tool from sharing resources between the output and next-state decode logic.
- FSM using one-hot encoding.
 - One-hot encoding generally provides the highest performance state machine in register-rich FPGAs.
- Registered outputs of each leaf-level block
 - A leaf-level block is one that infers logic, whereas a structural-level block only instantiates lower-level blocks (that is, creates hierarchy).
 - If outputs of a leaf-level block are registered, the synthesis tool retains the hierarchy, making it easier to cross-analyze the static timing analysis report with the code.
 - Registering the boundaries allows for a known timing relationship between blocks.
- Taking advantage of the data flow with proper pin-location constraints.
 - The data flow of Xilinx devices runs horizontally, in part, due to the carry chain running vertically. There are other reasons as well: 3-state buffer lines run horizontally and horizontal direct-connections between blocks.
 - To take advantage of the data flow, you should place address and data pins on the left-hand and right-hand edges of the chip. Note that because the carry chain runs up, you should place the LSB on the lower edge. The control signals are placed on the top and bottom of the die.

- Alternative counter styles.
 - Binary counters are slow. If your binary counter is a critical path, consider using an alternative style, such as Linear Feedback Shift Register (LFSR), pre-scalar, or Johnson, for example.
- Design is hierarchical and broken into functional blocks and technology blocks.
 - Designs should be broken into functional blocks, first at the higher-level blocks and then at the lower-level blocks; you should include technology-specific blocks.
 - Design hierarchy should make the design more readable, easier to debug, and easier to reuse.
- Duplicated high-fanout nets.
 - This can be controlled via your synthesis tool. However, you may choose to duplicate the registers for tighter control over duplication.
- Globally constrained design with the four global constraints: Period (for each clock), Offset In, Offset Out, and Pad-to-Pad.
 - You may have other constraints for multicycle paths, false paths, and critical paths. However, you should always start by specifying the four global constraints.

FPGA Design Checklist

FPGA Design Checklist

Introduction

Applying to Xilinx Virtex™ FPGAs, this checklist shows you how to increase the speed, reliability, and performance of your FPGA design. This checklist should also help all FPGA designers better use the FPGA resources and avoid common problems. Note that these tips can vary for CPLDs.

Creating Your HDL Code

- Group your operators for efficient synthesis. Use parentheses to group operators in arithmetic and combinatorial functions.
- Define all outputs, or use a default statement in your combinatorial processes to prevent inferring unnecessary latches in your design.
- Use default statements in registered processes (if this will not change the functionality) to avoid long delays on clock enable signals.
- Consider instantiation of a core if it will take you considerable time to create the same component, or you want the component to have peak performance.
- Instantiate components that cannot be inferred (DCM, block RAM, dynamically addressable SRL, or DSP slice, for example). This requires that you know which resources your synthesis tool will allow you to infer.
- Choose good design hierarchy. After creating functional blocks, separate distinct logic types (control or bus functions, for example). Keep clock domains in separate levels of the hierarchy.
- Register the outputs of each leaf level of hierarchy, including the outputs of state machines. This is a form of pipelining and should only be done when your design can tolerate additional latency.
- Consider pipelining your design to improve your design speed.
- Use case statements rather than if-then statements to synthesize to parallel logic and enable performance over cascading LUTs in series (priority encoders).
- Avoid nesting case and if-then statements. Nesting cascades LUTs in series.

- Put your next-state decoding and output-decoding logic for your state machines in separate process or always blocks to synthesize to faster performing designs.
- Carefully choose your state machine encoding technique. FSMs with less than eight states should be binary encoded. FSMs with between eight and sixteen states should be one-hot encoded. FSMs with more than sixteen states should be gray encoded or custom encoded.

Taking Advantage of the FPGA Architecture

- Verify that carry logic is automatically inferred for all of your arithmetic functions. Carry logic is inferred automatically with most synthesis tools if the design contains arithmetic operators (+, -, <, or >). Do *not* try to create arithmetic functions with XOR and other gates.
- Verify that the F5, F6, F7, and F8 multiplexers are being inferred automatically.
- Verify that the mult_and gate is used in the creation of multipliers.
- Verify that the SRL16 block is used as a pipelined delay element.
- Instantiate the DCM and use its clock multiplication, division, and phase shifting capabilities.
- Instantiate the DSP slice resources for large arithmetic functions, for example. Remember that if you do not use all of it, you are wasting some device resources.
- Instantiate block RAM in your design. Consider using block RAM resources for FIFOs, state machines, multipliers, Constant(k) Coefficient Multipliers (KCMs), and other deep memory functions. Remember that if you do not use all of it, you are wasting some device resources.
- Instantiate distributed RAM in your design when you need wide memory functions. Remember that deep memories that use Distributed RAM may require several LUTs to decode the outputs and add delay to the memory performance.
- Register your design's top-level inputs and outputs with the IOB registers to improve the design's on-chip and off-chip timing. Also, take advantage of the programmable set-up and hold-time capability of the input registers.
- Consider using 3-state buffers when creating wide multiplexers (for example, 32-to-1). This is much faster than using LUTs.
- Use the fast slew rate feature on a pin-by-pin basis to decrease your output transition times.
- Instantiate the BUFI0, BUFR, and BUFG resources when you want to control how clocks are routed in your design.

Building Reliable Designs

- Never gate your clock signals or generate clock signals from within the FPGA. These signals can glitch. Instead, follow the recommended coding style to infer a clock enable.
- Do not design with asynchronous sets or resets. Instead use synchronous sets and resets. If you need an asynchronous set or reset for portability, limit it to one signal and never gate it internally.
- Minimize clock skew by using the global routing resources to distribute your clock signals. Low fanout clocks and local clocks should be routed on the BUFI0 and BUFR resources.
- Understand the reliability of your state machine encoding. A design that has fewer bits that transition at once will function more reliably if the design has asynchronous characteristics.

- Consider alternative counter designs, such as Linear Feedback Shift Register (LFSR), pre-scalar, or Johnson, for example.
- When transmitting data between clock domains, consider using a synchronization circuit (to ensure that the data is transmitted properly) or a FIFO implemented in block RAM.

Avoiding Early Pin Locking

- Avoid locking your pins too early. The implementation tools have the greatest chance of meeting all of your timing objectives when they have the most flexibility to move pins around the die.
- If you must lock your I/O pins early, be certain that you work from your last successful implementation and move only those pins that must be moved.
- Avoid ground bounce by not placing simultaneously switching outputs all in a row. Move them around the die and place them close to GND pins so that you do not suffer from ground bounce.
- Only assign fast slew rate to those I/O pins that must have it to meet their timing constraints. Using fast slew rate carelessly can cause you to suffer from ground bounce.

Using Your Synthesis Tool Properly

- Do not allow optimization across hierarchical boundaries, which makes verification and analysis of your design more difficult, unless there is no other way for you to improve the design's timing.
- Use global timing constraints with your synthesis tool. Synplify will nudge your synthesis results, but most synthesis tools allow for verification of the constrained paths.
- Use your synthesis tool's ability to target specific device resources in your design. This is most often done with a utility (such as Scope in Synplify), and it can be done with most synthesis tools. Placing attributes with your HDL is often essential for inferring the proper device resources.
- Verify the performance of your design with your design's timing estimation capability to prevent you from compiling a netlist that cannot meet your timing objectives.
- Over-constrain the synthesis period constraint by two to three times your true performance objectives (for example, if your goals is 100 MHz, specify 200 to 300 MHz). Synthesis tools with timing-driven synthesis will take slightly more time to synthesize, but they can considerably reduce the number of PAR iterations and the run time for each PAR iteration.
- Do *not* pass on these timing constraints with your netlist (NCF, netlist constraints file). The implementation tools will either stop or incur considerable run times trying to meet these unrealistic constraints.
- In most situations, do not spend the time to make multicycle and false-paths constraints with your synthesis tool. This will create a single constraint for each path, rather than a single constraint that can cover many paths (which is what you can do with the implementation tools). However, if you are LUT limited, the synthesis tool may be able to save some LUT resources when multicycle constraints are applied. Similarly, OFFSET IN and OFFSET OUT constraints will be applied to each output separately rather than as a group.

Using the Xilinx Implementation Tools

- Use global timing constraints to communicate your system timing objectives.
- Use path-specific timing constraints to define false paths and multicycle paths that give the tools more flexibility to meet your global constraints.
- Verify the timing of your critical paths with the Timing Analyzer. Create custom timing reports and generate minimum timing information with this utility.
- To improve the speed of your design, activate the timing-driven packing option in the Mapper.
- To improve the speed of your design, increase the Place & Route Effort Level.
- To improve the speed of your design, activate Extra Effort Level. Be aware that there are two settings. Setting 1 (Normal) allows the implementation tools to increase their PAR time significantly. Setting 2 (Continue on Impossible) allows the implementation tools to run until you stop them (Ctrl + C).
- To further improve the speed of your design, use Multi-Pass Place & Route.
- Use the Post-Map Timing Report (or your synthesis tools estimates) to verify that your timing constraints are realistic. If you use the Post-Map Timing Report, estimate your design performance by using the 60/40 rule: Sixty percent of your timing budget for logic (or less), forty percent of your timing budget for routing.

Flow Diagram

Flow Diagram

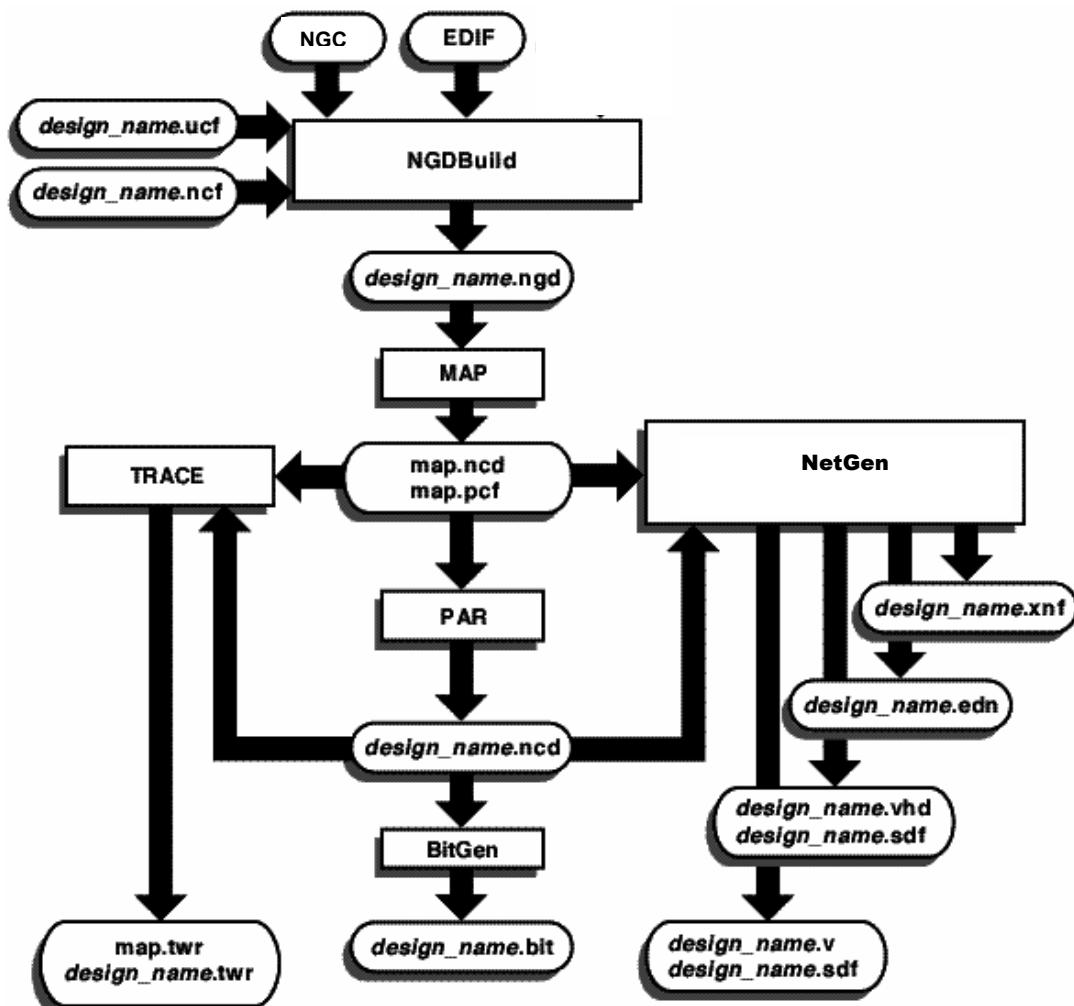
Introduction

This diagram uses the command line syntax for calling the Xilinx tools. The corresponding processes in the Project Navigator are:

NGDBuild = Translate

PAR = Place & Route

TRACE = Generate Post-Map or Post-Place & Route Static Timing



Glossary

Glossary

ABEL

ABEL is a primitive HDL used extensively in CPLD design. ABEL is generally not considered to be as powerful for creating high-level hardware descriptions as VHDL or Verilog.

ADC

Analog-to-Digital Converter. Sampled at various intervals, an analog signal is modeled as a digital signal.

AGP

Advanced Graphics Port. Voltage interface standard for graphics.

Alliance

Alliance is the name for the Xilinx consortium that interacts with third-party vendors to share information to help build seamless interaction between Xilinx tools and other EDA tools. The Alliance software package includes the Xilinx implementation tools, but this package does not include software for simulation, synthesis, or schematic capture. If you are using the Alliance Series™ software from Xilinx, it is assumed that you use a third-party EDA vendor for those applications.

analyze

Analyze is the term used to describe the process of syntax-checking in synthesis.

APU

Auxiliary Processing Unit. Provides a dedicated processing unit for controlling peripherals.

ASIC

Application Specific Integrated Circuit. A chip that is custom-designed for a specific application, rather than a general-purpose chip, such as a microprocessor.

ASSP

Applied Specific Standard Part (or Product). Another name for an ASIC.

ATPG

Automatic Test Pattern Generation. Test vectors are generated and run through the circuitry to test the part.

behavioral

A term often used to describe a form of HDL or simulation. Behavioral HDL is a model of a system that is not necessarily synthesizable. Behavioral simulation is a simulation of the source code (RTL or behavioral).

BGA

Ball Grid Array. A popular surface-mount chip package that uses a grid of solder balls for its connectors. Available in plastic and ceramic varieties, BGA is noted for its compact size, high lead count, and low inductance, which allow lower voltages to be used.

BIST

Built-In Self-Test. Tests functionality of memory resources (specifically RAM).

BitGen

BitGen is the command-line name for the configuration step performed by the Xilinx implementation tools. See Configuration.

bitstream

A bitstream is used to program Xilinx devices. It contains all the information used to configure internally a device's routing and logic resources as created by the designer.

block SelectRAM

Commonly referred to as block RAM. The block RAM in Xilinx is a dedicated block of RAM in Virtex™ devices. Xilinx block RAM can store up to 16 KB of data of varying widths and depths. Block RAM is fully synchronous with various ports for dual-port access. Each block can have individual clocks, enables, resets, data in, data out, and port widths.

BLVDS

Bus LVDS. This standard allows for bidirectional LVDS communication between two or more devices. The external resistor termination is different than the one for standard LVDS.

BSCAN

Boundary Scan. Boundary scan logic is used in manufacturing to test the interconnect in a Printed Circuit Board (PCB).

BSDL

Boundary Scan Description Language. BSDL is a software description of how the boundary scan logic is implemented in an IC. Boundary scan test software accepts BSDL descriptions.

BUFGCE

A Xilinx primitive that is part of clock management in Virtex FPGAs, BUFGCE is used to distribute a high-fanout clock-enabled clock signal. The clock signal is gated (without glitches) with a clock enable signal. When the clock enable is disabled, the clock is also disabled.

BUFGCTRL

A global clock buffer control block, BUFGCTRL is used to control and drive global clocks onto the differential global clock network.

BUFGMUX

A Xilinx primitive that is part of clock management in Virtex™ FPGAs, BUFGMUX is used to switch between clocks without glitches.

BUFIO

Clock buffer which drives clock pins for ILOGIC, OLOGIC, ISERDES, and OSERDES resources. Additionally, the BUFIO can drive the BUFR regional clocking resource.

BUFR

A regional clock buffer, BUFR drives clock pins of logic resources (CLB or RAM, for example) within a clock region.

BUFT

BUFT is the Xilinx primitive that represents a 3-state buffer.

carry logic

Carry logic exists in each slice and is dedicated logic used primarily to implement arithmetic logic functions. Carry logic (or the carry chain) runs vertically in Xilinx devices.

ChipScope ILA

ChipScope™ Integrated Logic Analyzer (ILA) is an add-on Xilinx software for use in place of a logic analyzer to test and capture data within a Xilinx device. ChipScope ILA consists of two primary elements: the ChipScope software, which resides on the PC, and the ChipScope core, which resides inside the chip. The software is used to set up trigger events and capture data. The core that resides in the chip is used to connect to the internal test nodes of the design under test. The information is communicated between the device and the software via a cable connected to the PC and the JTAG ports on the chip.

CLB

Configurable Logic Block. Made up of slices, the Xilinx CLB is where most of the logic is implemented in the FPGA.

CMOS

Complementary Metal Oxide Semiconductor. The most widely used type of integrated circuit for digital processors and memories. CMOS uses Positive Channel MOS (PMOS) and N-Channel MOS (NMOS) transistors wired together in a manner that causes less power to be used than PMOS-only or NMOS-only circuits.

combinational

See combinatorial.

combinatorial

Combinatorial logic is implemented in gates that need to be continually driven. Combinatorial logic is separate from registered logic, which does not need to be continually driven. Combinatorial logic does not hold its value if the signals driving it are not maintained.

compile

Compiling a design can happen at various points in the design process. When you synthesize HDL, you are compiling the code into a hardware netlist. Compile implies transformation of code (or a file) into a different format.

configuration

Configuration is one step in the Xilinx implementation process. During the configuration step, a bitstream is created for programming the device.

core

A core, which can be quickly used without much engineering time or cost, usually refers to IP. A core should be pre-tested for functionality. A core might be referred to as a “plug-n-play” design.

CORE Generator software system

The CORE Generator™ software system is the Xilinx software used to create cores for your design. These ready-made functions can be instantiated directly into your design and can also be simulated in a behavioral environment. The cores range in complexity and price. Most simple functions are free and customizable (block RAM or FIR filters, for example) while other cores (PCI or USB, for example) require a fee.

CPLD

Complex Programmable Logic Device. A programmable logic device that includes a programmable interconnect between the logic blocks. A CPLD is usually characterized as multiple interconnected PALs.

CPU

Central Processing Unit.

CS

Chip Scale package.

CTT

Center Tap Terminated. A voltage interface standard. 3.3-Volt memory bus standard.

DAC

Digital-to-Analog Converter. Converts a digital signal into an analog signal.

daisy chain

A daisy chain is the connection of several Xilinx parts in sequence for implementing each device in sequence through serial configuration.

DCI

Digital Controlled Impedance. DCI in Virtex™ FPGAs provides controlled impedance drivers and on-chip termination for single-ended I/Os—eliminating the need for external resistors and improving signal integrity.

DCM

Digital Clock Manager. The Xilinx DCM has four blocks of clock management: Clock Delay-Locked Loop (DLL); Digital Frequency Synthesizer (DFS); and Digital Phase Shifter (DPS).

DDR

Double Data Rate uses both edges of a clock to capture data.

die

A small piece of silicon wafer, bounded by adjacent scribe lines in the horizontal and vertical directions, that contains the complete device being manufactured. Also called chip and microchip.

DFT

Design For Test. Circuitry included in a design to test the functionality and/or integrity of the internal circuits. The goal is to make the device self-testing.

DLL

Delayed-Locked Loop. Digital version of the PLL. Digital clock locking circuit. Compares two clock signals and aligns the two.

DRP

Dynamic Reconfiguration Port. Allows dynamic changes to specific resources (DCM or MGT, for example). Available through the FPGA fabric.

DSM

Deep Sub-Micron. Also known as second-order and third-order effects, DSM is the effect routing has on the delays and noise in a circuit.

DSP48

The DSP48 slice contains register, multiplier, and arithmetic logic for general arithmetic operations and DSP functions.

EA

Embedded Array ASIC. An ASIC that is a combination of a gate array and a standard cell. The wafer does have a partially fabricated portion (gate array) and a blank portion. Embedded arrays also allow custom macros and memories similar to a standard cell ASIC.

ECO

Engineering Change Order. After the ASIC has been masked, changes to the mask require a “re-spin” for which there is a fee.

EDA

Electronic Design Automation. Using the computer to design and simulate the performance of electronic circuits on a chip.

EDIF

Electronic Design Interchange Format. EDIF is the industry-standard netlist format.

equivalency checking

Also known as formal verification, equivalency checking is used to check the equivalency of a circuit both before and after it has been synthesized.

FG

Fine-pitch ball grid array package.

FIFO

First-In First-Out. A FIFO is generally implemented in a RAM block. A FIFO is used to store data at one rate (clock rate) and read data at a different rate.

FIFO16

Dedicated FIFO resources utilizing the RAMB16 resource for memory storage element.

flash memory

A memory chip that can be rewritten and retain its content without power.

Foundation

Foundation™ software is a Xilinx software package that includes a complete solution for schematic capture, simulation, and implementation of Xilinx devices.

footprint

A footprint represents the package layout. A footprint also refers to the number of pins used as I/O and as power and ground pins.

FPGA

Field Programmable Gate Array. Field reprogrammable integrated circuit.

FPGA Compiler II

FPGA Compiler II is a synthesis tool from Synopsys.

FSM

Finite State Machine. A computing device designed with the operational states required to solve a specific problem. The circuits are minimized, specialized, and optimized for the application. A state machine controls the operation of circuits. It provides outputs that are generated at the proper time to control other logic.

function generator

See LUT.

gate array

An ASIC that uses a partially fabricated wafer that uses only the masking of routing layers to customize its operation. A gate array is characterized by low up-front costs, low development time, low densities, and limited performance. Manufactured in low volumes, it is inexpensive. This form of ASIC is becoming obsolete as other technologies surpass it in size and lower cost.

gating

Gating generally refers to gating a clock. When you gate a clock, the clock is combined with another signal to create a new clock. This is not good design practice. A gated clock will glitch, which will make the design unreliable. Xilinx Virtex™ devices have glitchless clock management resources for you to create a “gated-clock.” See BUFGMUX and BUFGCE.

GDSII

Graphic Design System II. A polygon layout format used in the ASIC design process.

global clock buffer

Global clock buffers are used to drive dedicated clock trees within Xilinx devices. These clock networks are optimized for propagating low-skew, high-frequency clock signals throughout the device. Each part has from four to sixteen global clock buffers. Global clock buffers are also known as BUFGs.

gray code

Gray encoding refers to logic where only one bit changes when it changes state, reducing the glitching caused by binary sequences. Gray code is also generally faster than binary-encoded logic because it requires more registers to represent the state; however, logic needed to decode a state only needs to look at fewer bits to decode the state it is in.

GSR

Global Set Reset. The global set reset is a dedicated routing network used at the end of configuration during the startup sequence. GSR brings the device up in a power-on state. Registers are in their set or reset state, as determined by the code. GSR can be used after configuration to set or reset all synchronous elements in the device. Using GSR in Virtex™-based devices is not recommended because it is very slow.

GTL

Gunning Transceiver Logic Terminated. Voltage interface standard.

GTS

Global Three-State. The GTS net is a dedicated routing network used during configuration to place all I/O pins (all I/O not used during configuration) on the device in a 3-state condition. This net is released during the startup sequence of configuration. Afterwards, it can be used to place all outputs of the operational device in a 3-state condition.

GUI

Graphical User Interface. Interface where a user specifies options that will control how a design is optimized or implemented.

HDL

Hardware Description Language. A language used for modeling, designing, and simulating hardware. The two most common HDL languages are VHDL and Verilog.

HQ

High heat dissipation Quad flat pack package.

HSTL

High-Speed Transceiver Logic. Voltage interface standard. 1.5-Volt bus interface standard.

IBM

International Business Machines, Corp. A standard cell ASIC vendor.

IBIS

Input output Buffer Information Specification. IBIS is a method for providing the input and output device characteristics through V/I data without disclosing any circuit or process information. A behavioral modeling specification suitable for transmission line

simulation of digital systems, IBIS is applicable to most digital components. Xilinx provides IBIS models instead of SPICE models because SPICE models have proprietary information.

IC

Integrated Circuit. The formal name for computer chip. Silicon on which circuits are grown.

ICE

Integrated Circuit Engineering.

ILOGIC

Input Logic. Contains all input register and routing logic for input data.

implementation

Implementation is what Xilinx refers to as several steps that are part of the place & route process. These steps include translate; map; place & route, static timing analysis; and bitstream generation (for programming).

instantiate

Instantiate is a term used in HDL to represent placing a hierarchical block in the code. This is synonymous with placing a logic symbol in a schematic.

Intellectual Property

See IP.

I/O

Input and Output. Referring to the ports of a chip.

IOB

Input/Output Block. The Xilinx IOB interfaces external signals. The Xilinx IOB has a pad, input and output buffers, and registers.

IP

Intellectual Property. A broad category of intangible materials that are legally recognized as proprietary to an organization. In the computer field, hardware circuits, software, and text is copyrighted. Depending on the situation, the algorithms used within hardware circuits and software may also be patentable, and most brand names can be trademarked.

ISE

Integrated Synthesis Environment. Pronounced as “ice”, the ISE™ software package from Xilinx includes a complete solution for simulation, synthesis, and implementation of a Xilinx device.

ISERDES

Input Serializer/Deserializer. Converts incoming serial bitstream to parallel data.

JTAG

Joint Test Action Group. An IEEE standard for boundary scan technology.

latency

The number of clock cycles required to process information.

LeonardoSpectrum

LeonardoSpectrum is a synthesis tool from Mentor Graphics.

LFSR

Linear Feedback Shift Register. An LFSR uses a pseudo-random counting sequence. LFSRs are very useful because they can run at higher clock frequencies than a binary sequence, but they do repeat the sequence. LFSRs are also known as Pseudo-Random Bitstream (PRBS) generators.

Libraries Guide

A software manual that contains a list of all Xilinx macros and primitives listed in alphabetical order. For each macro or primitive, there is a schematic drawing that displays the port names for HDL instantiation, a description of its functionality, and a truth table for expected outputs depending on the inputs.

LM

Layer Metal. The number of metal layers used to provide routing lines in an integrated circuit.

LOC

Location constraint. Location constraints are used to lock pin locations or to place logic in specific locations on the die.

LogiBLOX

LogiBLOX software is used to create small cores for XC4000 and Spartan devices.

LUT

Look-Up Table. A Xilinx LUT is made from SRAM, but it generally functions as a 16 x 1 ROM. Xilinx LUTs use four inputs. A LUT is loaded with the possible logic values based on the 16 possible outputs from a four-input logic function. The LUT is also known as a function generator.

LVCMOS

Low Voltage CMOS.

LVDS

Low Voltage Differential Signaling. A differential I/O standard, LVDS requires that one data bit is carried through two signal lines. As with all differential signaling standards, LVDS has an inherent noise immunity over single-ended I/O standards. The voltage swing between two signal lines is approximately 350 mV. The use of a reference voltage (V REF) or a board termination voltage (V TT) is not required. LVDS requires the use of two pins per input or output. LVDS requires external resistor termination.

LVPECL

Low Voltage Positive Emitter Coupled Logic.

LVTTL

Low Voltage TTL. Voltage interface standard.

macro

The term macro is often used interchangeably with core. See core.

map

Map is one step of the implementation of a Xilinx device. The map step optimizes (if the input netlist comes from a schematic tool) logic in the netlist into Xilinx device resources (LUTs, registers, or 3-state buffers, for example) and packs resources into slices and IOBs.

MP

Micro Processor.

MHz

Mega Hertz.

ModelSim

ModelSim is a behavioral simulation tool from Mentor Graphics.

NCF

Netlist Constraints File. The NCF is created by the synthesis tools to propagate constraints from the synthesis tool to the Xilinx implementation tools.

Netlist

A list of logic gates and their interconnections that make up a circuit. Usually in text format. Most netlists used by Xilinx use the EDIF or XNF format.

NGDBuild

NGDBuild is the command line name for translate. See translate.

NRE

Non-Recurring Engineering costs. Up-front costs paid to the ASIC vendor for developing an ASIC.

nW

Nano Watts.

OFFSET IN

A timing constraint that covers paths from input pads to a synchronous element.

OFFSET OUT

A timing constraint that covers paths from synchronous elements to output pads.

OLOGIC

Output Logic. Contains all output register and routing logic for output data.

one-hot

One-hot encoded logic refers to logic where one bit is active (or high) and all other bits are not active (or low).

optimize

Optimization of hardware is the act of optimizing logic (Boolean) to use hardware resources efficiently.

OSERDES

Output Serializer/Deserializer. Converts outgoing parallel data to a serial bitstream.

pad-to-pad

Timing specification that covers paths from input pads through combinatorial logic to output pads. A pad-to-pad constraint does not cross any registered boundaries.

PAL

Programmable Array of Logic. A type of programmable logic chip (PLD) that contains arrays of programmable AND gates and predefined OR gates (only the AND gate connections are programmable). A PAL is naturally aligned to provide Sum-Of-Products (SOP) logic expressions.

P&R

Place & Route. The act of placing the logic on the die and routing the signals between the logic to meet timing requirements. This is one step of the Xilinx implementation process.

PAR

Place And Route. See P&R.

PCI

Peripheral Component Interconnect. A peripheral bus commonly used in PCs, Macintoshes, and workstations. PCI provides a high-speed datapath between the CPU and peripheral devices (video, disk, or network, for example).

period

Timing specification for synchronous elements to synchronous elements.

pipeline

Pipelining a design is the act of placing registers between combinatorial logic to increase the throughput of the design (clock frequency) at the expense of latency.

PECL

Positive Emitter Coupled Logic. Requires two signal lines for transmitting one data bit. This standard specifies two pins per input or output. The voltage swing between these two signal lines is approximately 850 mV. The use of a reference voltage (V REF) or a board termination voltage (V TT) is not required. The LVPECL standard requires external resistor termination.

pin locking

Pin locking is the act of placing input and output signals on specific pins of the part. Because an FPGA is completely programmable, the designer can place signals on any I/O pin in the device. Note that it is generally recommended to set up the data flow in the device to run horizontally with the Least Significant Bit (LSB) of the buses being placed lower on the left-right hand edges of the device because carry logic (the carry chain) runs vertically and up in the devices.

PLA

Programmable Logic Array. A type of programmable logic chip (PLD) that contains arrays of programmable AND and OR gates (both the AND and OR gate connections are programmable). A PLA is naturally aligned to provide Sum-Of-Products (SOP) logic expressions.

Place & Route

The act of placing the logic on the die and routing the signals between the logic to meet timing requirements. This is one step of the Xilinx implementation process.

pipelining

Insertion of registers between combinatorial logic to increase the throughput (performance or clock frequency) of the circuitry at the expense of latency.

PLD

Programmable Logic Device. Includes FPGAs and CPLDs.

PLL

Phase-Locked Loop. Analog clock locking circuit. Compares two clock signals and aligns the two.

PMCD

Phase-Matched Clock Divider. Creates multiple rising-edge aligned clocks.

PQ

Plastic Quad flat pack package.

PRBS generator

See LFSR.

Precision Synthesis

Precision Synthesis is a synthesis tool from Mentor Graphics.

priority encoded

Refers to a logic structure in which the logic is cascaded to implement the logic function. Generally, cascaded logic is not as efficient as logic implemented in a parallel structure.

Project Navigator

Project Navigator is the GUI in the ISE™ software program. Within this GUI, you can perform synthesis, simulation, and implementation and specify synthesis, simulation, and implementation options.

RAM

Random Access Memory. Read and write memory.

regression

A functional relationship between two or more correlated variables, which is often empirically determined from data and used to predict values of one variable when given values of the others. In hardware, regression testing is used so that not all possible vectors need be applied. For example, for a two-input AND gate, there are four possible logic

vectors that can be applied. Regression testing might only test one or two of those vectors, and if it works, that AND gate can be considered to be working correctly.

revision

A revision in the Xilinx tools represents a change in Xilinx implementation options. Revision creation is controlled by the user.

RLOC

Relative Location Constraints. RLOCs are used to group logic elements together to reduce routing delays in the design. Placing the logic closely together does not allow the place & route step to separate any logic that “belongs” together. This creates an RPM. That is, one piece of the logic is placed in a relative location to another piece of the logic. It is not hard-placed on the die.

RPM

Relationally Placed Macro. A relationally placed macro uses RLOCs to group related logic together to reduce the amount and delays associated with routing.

RTL

Register Transfer Level. A term usually used to describe HDL code that is synthesizable.

SC

Standard Cell ASIC. An ASIC that uses uniform logic cells for tight packing. A standard cell ASIC uses a blank wafer rather than a partially fabricated one. A standard cell ASIC is characterized by high up-front costs, and long development time. It provides for the largest possible density and the highest performance, and, in high volumes, is the most inexpensive and most efficient.

scan

Internal scan chain. Creates an internal shift register to test the functionality of the part.

SDF

Standard Delay Format. The SDF file is used to propagate delays associated with logic and routing in hardware circuitry for timing simulation. An SDF file usually accompanies a structural HDL file used to provide the logic functionality and connectivity.

SelectIO Technology

SelectIO™ technology is Xilinx terminology for the support of a myriad of voltage threshold levels for interfacing external components. The Virtex™-4 FPGA and other families support many different I/O standards.

SelectRAM Memory

SelectRAM™ memory refers to using the internal LUTs as RAM rather than as function generators. SelectRAM memory is also known as distributed RAM or LUT RAM. Each LUT can be configured as a 16 (depth) by 1 (width) RAM. The characteristics of this RAM are synchronous write and asynchronous read. SelectRAM memory can also be used as dual-port RAMs.

SelectROM

SelectROM refers to using the internal LUTs as ROM rather than as function generators. SelectRAM is also known as distributed ROM or LUT ROM. Each LUT can be configured as a 16 (depth) by 1 (width) ROM. The ROM has an asynchronous read.

silicon

The base material used in chips. Its atomic structure and availability make it an ideal semiconductor material. In chip making, silicon is mined from white quartz rocks and put through a chemical process at high temperatures for purification. It is mixed (doped) with other chemicals in a molten state to alter its electrical properties.

simulation

The execution of a concept (design) on a computer to imitate real-world functionality. In hardware systems, simulation is done to model and verify hardware concepts via software before implementing the hardware.

simprims

Simulation primitives. These simulation primitives are used for timing simulation of a design implemented in Xilinx.

skew

Skew represents the difference in time between when a signal reaches different endpoints.

slices

Slices exist inside a CLB. Each slice contains two LUTs and two registers. Other logic that exist inside each slice include multiplexers (F5, F6, F7, and F8 MUXes), routing, and carry logic.

SRAM

Static RAM. A memory chip that requires power to retain its content. Unlike dynamic RAMs, static RAM does not require refresh circuitry.

SRL

Shift Register LUT. The SRL uses the LUT as a shift register. The SRL16 is the macro for the SRL. This shift register can implement a serial shift of up to 16 clock cycles (for each LUT). The SRL can implement only simple shift-register functions of serial-in, serial-out without any reset functionality. However, the SRL can be initialized on power-up with data (using INIT attributed in the UCF). Also, each “register” within the SRL can be dynamically read.

SSTL

Stub Series Transceiver Logic. Voltage interface standard. Memory bus interface standard.

ST

Formerly SGS Thompson.

STA

Static Timing Analysis. Timing analysis used to determine the worst-case delays or performance of a circuit. Usually, the delays are compared to timing objectives provided in the form of timing constraints.

stamp

Stamp models are the industry-standard format for board-level timing verification.

startup

The startup block is used during configuration to control the internal reset, global write enable, and global three-state net. The startup block is called STARTUP_VIRTEX for Virtex™ FPGAs (STARTUP_VIRTEX2 for Virtex-II FPGAs and STARTUP_VIRTEX4 for Virtex-4 FPGAs).

Static Timing Analysis

See STA.

synchronous

Synchronous design implies a design that uses only one clock, one edge of the clock, D-type flip-flops, and the proper use of hierarchy.

Synplify

Synplify is a synthesis tool from Synplicity.

synthesis

Compilation of HDL code (Verilog or VHDL) into a hardware representation of a circuit. Synthesis tools create a netlist, usually in EDIF format.

synthesize

Verb form of synthesis.

TI

Texas Instruments. A standard cell and embedded array ASIC vendor.

timing constraints

Timing constraints are used by the designer to communicate the performance objectives of a design to the Xilinx implementation tools. The Xilinx implementation tools are timing driven in that the tools attempt to place & route the logic to meet timing constraints.

translate

Translate is one step of the implementation of a Xilinx device. The translate step combines all input netlists, checks the constraints, and looks for general problems with the input netlists.

TTL

Transistor Transistor Logic. A digital circuit composed of bipolar transistors wired in a certain manner. TTL logic has been widely used since the early days of digital circuitry. TTL designations may appear on input or output ports of various devices, which indicates a digital circuit in contrast to an analog circuit.

TTM

Time To Market.

TWR

The TWR file is created by the Timing Analyzer, a Xilinx static timing analysis tool. This file contains timing delay information as it pertains to the timing constraints applied to the design.

version

A version in the Xilinx tools represents a change in the input netlist. This generally represents a change in the HDL source code or a resynthesized netlist.

Verilog

A Hardware Description Language (HDL).

VHDL

Very High Speed Integrated Circuit HDL.

VITAL

VHDL Initiative Towards ASIC Libraries. VITAL is the industry standard used for creating timing simulation models for use with VHDL.

UCF

User Constraints File. The UCF is used to communicate constraints to the Xilinx implementation tools. The UCF is Xilinx specific.

UI

User Interface. See GUI.

Unisim

Unified simulation primitives. These simulation files are created for simulating Xilinx primitives instantiated in HDL code.

uP

Micro Processor.

Xilinx CoreLib

Xilinx simulation files for IP created by the CORE Generator™ software system.

XNF

Xilinx Netlist Format. A text format specifically created for Xilinx.

XST

Xilinx Synthesis Technology. XST is a synthesis tool from Xilinx.

Additional Xilinx Courses and Services

Additional Xilinx Courses

Fundamental Courses

- ISE Design Entry
- Fundamentals of FPGA Design
- Fundamentals of FPGA Design Live e-Learning
- Fundamentals of FPGA Design Interactive Recorded e-Learning
- Introduction to Verilog
- Introduction to Verilog Live e-Learning
- Introduction to VHDL
- Introduction to VHDL Live e-Learning
- TMRTTool
- TMRTTool Live e-Learning
- Introduction to AccelDSP

Intermediate Courses

- Designing for Performance
- Designing for Performance Live e-Learning
- Designing with PlanAhead
- Designing with the Virtex-4 Family
- Designing with the Virtex-4 Family Live e-Learning
- Designing with the Virtex-5 LX FPGA
- Designing with the Virtex-5 LX FPGA Live e-Learning
- DSP Design Flow
- Designing with Multi-Gigabit Serial I/O
- Embedded Systems Development
- Signal Integrity for High-Speed Memory and Processor I/O

Advanced Courses

- Advanced Features and Techniques of Embedded Systems Development
- Advanced FPGA Implementation
- Advanced FPGA Implementation Live e-Learning
- Advanced VHDL
- Advanced VHDL Live e-Learning
- DSP Implementation Techniques

Specialized Courses

- Designing a LogiCORE PCI System
- Designing a LogiCORE PCI System Live e-Learning
- Designing a LogiCORE PCI-X System
- Designing a LogiCORE PCI-X System Live e-Learning
- Designing a LogiCORE PCI Express System

Global Services Portfolio

Design Services

Whether it's helping you create your product from an initial concept, implementing your specifications, modifying existing intellectual property (IP) cores, or improving product performance, Xilinx Design Services provides extensive FPGA hardware and embedded software design experience to solve even the most complex design challenge.

MySupport.xilinx.com

MySupport.xilinx.com gives you the same access to tools as on support.xilinx.com but with the added benefit of the ability to personalize your page. The MySupport.xilinx.com web site gives you instant access to information such as datasheets, application notes, Xcell Journal articles, software manuals, the Answer Database, and much more. And, all of this is searchable so you can find every reference to your specific question, quickly. Plus you can easily add bookmarks to retrace your steps as you compile all the answers you need. If that's not enough, you can also customize the layout and the colors to suit your preferences.

Titanium Technical Service

With a dedicated application engineer, Titanium Technical Service will improve your productivity, accelerate your time to market, increase your expertise, minimize your risk, and help you produce the best possible designs. You save development cost, and may fit your design in a lower-cost FPGA.

Embedded Processing QuickStart

With solutions ranging from the soft-core PicoBlaze™ and MicroBlaze™ processors to the embedded PowerPC™ available in the Virtex™-II Pro and Virtex-4 FPGAs, Xilinx is leading the way in embedded processing performance. The Embedded Processing QuickStart! solution delivers individualized service that includes a QuickStart! application engineer at your site for a week. This Xilinx expert will train your hardware team on creating embedded systems, instruct software engineers on how to optimally utilize supporting FPGA features, and help your team maximize performance.

PlanAhead QuickStart

Xilinx has a history of creating multi-platform solutions to meet the needs of almost any system. Our breakthrough Virtex-4 FPGA family is the perfect example of this commitment, consisting of multiple devices and three domain-optimized platforms—LX for logic-intensive designs, SX for high-performance signal processing, and FX for high-speed serial connectivity and embedded processing. The PlanAhead™ QuickStart! makes it easier than ever before to optimize this unique technology in your design. You will receive a QuickStart! engineer at your site for one week. During that period, the Xilinx expert will train and empower your team to complete your project on time and make best use of the Xilinx device you've selected.

Xilinx Productivity Advantage Program

The XPA Program will significantly reduce your risk and increase productivity, while saving you time and money because a single purchase delivers the soft-ware, support, services, and IP, in the quantities you need, at the best value. If you want the assurance of flawless designs, with access to the best engineers and tools in the industry, choose the Xilinx Productivity Advantage Program – and rest easy.



Course Specification

Lab Descriptions

- **Lab 1:** Projects in the Project Navigator – Gain comprehensive hands-on experience with the HDL flow in the ISE software. Create a new project, add source files, synthesize a design, and use the error navigation feature.
- **Lab 2:** Synthesis Options – Modify XST synthesis properties, read synthesis reports to compare the synthesis results, and use the snapshot utility.
- **Lab 3:** ECS – Perform the basic tasks of the schematic editor, such as adding symbols, connecting symbols with wires, naming wires and buses, adding I/O markers, and using the Xilinx CORE Generator™ software system with ECS.
- **Lab 4:** ISE Simulator and StateCAD Tool – Perform the simulation and verification process of the design cycle. Demonstrate how these tools are incorporated into the ISE software.

Level – Fundamental**Course Duration** – 1 day**Price** – No Charge**Course Part Number** – FPGA16000-8-ILT**Who Should Attend?** – Digital designers who use ISE software extensively and who need to learn the major aspects of the new ISE 8.1 product**Prerequisites**

- Basic knowledge of the VHDL or Verilog language
- Basic knowledge of Virtex™ FPGA architecture

Software Tools

- ISE 8.1i

After completing this comprehensive training, you will have the necessary skills to:

- Create a new Project Navigator project in the ISE software
- List the design flows available in the ISE software
- Access and modify XST synthesis options
- Create a schematic design by using the Engineering Capture System (ECS) schematic entry tool
- Create a symbolic state machine by using the StateCAD tool
- Create testbenches and simulate a design by using the ISE Simulator

Course Outline

- Course Agenda
- Projects in the Project Navigator
- **Lab 1:** Projects in the Project Navigator
- HDL Synthesis and XST
- **Lab 2:** XST Synthesis Options
- ECS: Engineering Capture System
- **Lab 3:** ECS
- StateCAD Tool
- ISE Simulator
- **Lab 4:** ISE Simulator and StateCAD Tool
- Additional Features
- Summary

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, the ISE Design Entry course is not run publicly. Please contact your local Field Application Engineer or Sales Representative to request that this course be delivered to you in a private session.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



FPGA13000-82-ILT (v1.0)

Fundamentals of FPGA Design

Course Specification

Lab Descriptions

- **Lab 1:** Xilinx Tool Flow – Create a new project in the ISE Project Navigator and use the Architecture Wizard and PACE tool in the design process. Implement a design by using default software options. The design will be simulated.
- **Lab 2:** Architecture Wizard and PACE – Use the Architecture Wizard to customize a DCM and incorporate the DCM into the design. Use PACE to assign pin locations and implement the design.
- **Lab 3:** Global Timing Constraints – Enter global timing constraints with the Xilinx Constraints Editor. Review the Post-Map Static Timing Report to verify that the timing constraints are realistic. Use the Post-Place & Route Static Timing Report to determine the delay of the longest constrained path for each timing constraint.
- **Lab 4:** Implementation Options – Adjust process properties and I/O configuration options to improve the design performance.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLASS (877-959-2527). To register online, search by Keyword "FPGA" in the Training Catalog at <https://xilinx.onsaba.net/Saba/Web/Main>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

After completing this comprehensive training, you will have the necessary skills to:

- Use Xilinx Project Navigator to implement an FPGA design
- Assign pin locations with the PACE tool
- Create DCM instantiations with the Architecture Wizard
- Read reports to determine whether design goals were met
- Use the Constraints Editor to enter basic global timing constraints
- Locate and modify implementation options

Course Outline

- Course Agenda
- Review of Basic FPGA Architecture
- Xilinx Tool Flow
- **Lab 1:** Xilinx Tool Flow Lab
- Reading Reports
- Architecture Wizard and PACE
- **Lab 2:** Architecture Wizard and PACE Lab
- Global Timing Constraints
- **Lab 3:** Global Timing Constraints Lab
- Implementation Options
- **Lab 4:** Implementation Options Lab
- Synchronous Design Techniques
- Course Summary

* Recorded e-Learning

© 2006 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.



FPGA13000-8.2-LEL (v1.0)

Fundamentals of FPGA Design Live e-Learning

Course Specification

Lab Descriptions

- **Lab 1:** Xilinx Tool Flow – Create a new project in the ISE Project Navigator and use the Architecture Wizard and PACE tool in the design process. Implement a design by using default software options. The design will be simulated.
- **Lab 2:** Architecture Wizard and PACE – Use the Architecture Wizard to customize a DCM and incorporate the DCM into the design. Use PACE to assign pin locations and implement the design.
- **Lab 3:** Global Timing Constraints – Enter the global timing constraints with the Xilinx Constraints Editor. Review the Post-Map Static Timing Report to verify that the timing constraints are realistic. Use the Post-Place & Route Static Timing Report to determine the delay of the longest constrained path for each timing constraint.
- **Lab 4:** Implementation Options – Adjust the process properties and I/O configuration options to improve the design performance.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLASS (877-959-2527). To register online, search by **Delivery Type** "Live e-Learning" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

* Recorded e-Learning

Course Description

The *Fundamentals of FPGA Design Live e-Learning* course delivers high-value training to you remotely via the Internet, with the emphasis on keeping your work schedule and priorities intact. Scheduled over a one-week period, this course comprises a series of three 3-hour sessions (two to three lectures plus a lab combination).

This course covers the ISE™ 8.2 software features, such as the Architecture Wizard and the Pin and Area Constraint Editor (PACE). Other topics include design planning, implementation options, and global timing constraints. For more emphasis on improving the overall design performance, take the follow-up course *Designing for Performance*, which builds on the basic principles covered in this course.

Level – Fundamental

Course Duration – Three 3-hour sessions (two to three lectures plus a lab combination)

Price – \$400 USD or 4 Xilinx Training Credits

Course Part Number – FPGA13000-8.2-LEL

Who Should Attend? – Digital designers who have working knowledge of basic HDL (VHDL or Verilog) and who are new to Xilinx FPGAs

Prerequisites

- Basic FPGA Architecture: Slice and I/O Resources REL*
- Basic FPGA Architecture: Memory and Clocking REL*
- Working HDL knowledge (VHDL or Verilog)
- Digital design experience

Software Tools

- Xilinx ISE 8.2i

After completing this comprehensive training, you will have the necessary skills to:

- Use Xilinx Project Navigator to implement an FPGA design
- Assign pin locations with the PACE tool
- Create DCM instantiations with the Architecture Wizard
- Read reports to determine whether design goals were met
- Use the Constraints Editor to enter basic global timing constraints
- Locate and modify implementation options

Course Outline

Session 1

- Review of Basic FPGA Architecture
- Xilinx Tool Flow
- Architecture Wizard and PACE
- **Lab 1:** Xilinx Tool Flow
- **Lab 2:** Architecture Wizard and PACE

Session 2

- Reading Reports
- Global Timing Constraints
- **Lab 3:** Global Timing Constraints

Session 3

- Synchronous Design Techniques
- Implementation Options
- **Lab 4:** Implementation Options



FPGA13000-8-iREL (v1.0)

Fundamentals of FPGA Design

Interactive Recorded e-Learning

Course Specification

Lab Descriptions

Note: use the free downloadable ISE WebPACK™ solution from www.Xilinx.com or your local install to perform the labs

- **Lab 1:** Xilinx Tool Flow – Create a new project in the ISE Project Navigator and use the Architecture Wizard and PACE tool in the design process. Implement a design by using default software options. The design will be simulated.
- **Lab 2:** Architecture Wizard and PACE – Use the Architecture Wizard to customize a DCM and incorporate the DCM into the design. Use PACE to assign pin locations and implement the design.
- **Lab 3:** Global Timing Constraints – Enter the global timing constraints with the Xilinx Constraints Editor. Review the Post-Map Static Timing Report to verify that the timing constraints are realistic. Use the Post-Place & Route Static Timing Report to determine the delay of the longest constrained path for each timing constraint.
- **Lab 4:** Implementation Options – Adjust the process properties and I/O configuration options to improve the design performance.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Delivery Type** "Live e-Learning" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Level – Fundamental

Course Duration – Seven 15- to 20-minute recorded modules and four self-paced hands-on labs

Price – \$400 USD or 4 Xilinx Training Credits

Course Part Number – FPGA13000-8-iREL

Who Should Attend? – Digital designers who have working knowledge of basic HDL (VHDL or Verilog) and who are new to Xilinx FPGAs

Prerequisites

- Working knowledge of basic HDL (VHDL or Verilog)
- Digital design experience

Software Tools

- Xilinx ISE 8.1i

After completing this comprehensive training, you will have the necessary skills to:

- Use Xilinx Project Navigator to implement an FPGA design
- Assign pin locations with the PACE tool
- Create DCM instantiations with the Architecture Wizard
- Read reports to determine whether design goals were met
- Use the Constraints Editor to enter basic global timing constraints
- Locate and modify implementation options

Course Outline

- Basic FPGA Architecture
- Xilinx Tool Flow
- **Lab 1:** Xilinx Tool Flow
- Reading Reports
- Architecture Wizard and PACE
- **Lab 2:** Architecture Wizard and PACE
- Global Timing Constraints
- **Lab 3:** Global Timing Constraints
- Implementation Options
- **Lab 4:** Implementation Options
- Synchronous Design Techniques

Introduction to Verilog

Course Specification

- Verilog Procedural Statements
- Controlled Operation Statements
- **Lab 4:** n-bit Binary Counter and RTL Verification
- Advanced Language Concepts
- **Lab 5:** Comparator

Day 3

- Tasks and Functions
- **Lab 6:** Arithmetic Logic Unit
- Finite State Machines
- **Lab 7:** Finite State Machine
- Targeting Xilinx FPGAs
- **Lab 8:** Calculator

Lab Description

The labs for this course provide a practical foundation for creating synthesizable RTL code. All aspects of the design flow are covered in the labs. The labs are written, synthesized, behaviorally simulated, and implemented by the student. The focus of the labs is to write code that will optimally infer reliable and high-performance circuits. The labs culminate in a functional calculator that students verify in simulation.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "Language" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

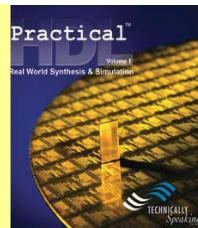
Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Practical HDL is a comprehensive VHDL/Verilog self-paced multimedia training environment.

This tool both reinforces topics covered during the class and offers additional advanced subject matter.

Cost - \$300 USD



Introduction to Verilog Live e-Learning

Course Specification

Session 5

- Verilog Procedural Statements
- Controlled Operation Statements
- **Lab 4:** n-bit Binary Counter and RTL Verification

Session 6

- Advanced Language Concepts
- **Lab 5:** Comparator

Session 7

- Tasks and Functions
- **Lab 6:** Arithmetic Logic Unit

Session 8

- Finite State Machines
- **Lab 7:** Finite State Machine

Session 9

- Targeting Xilinx FPGAs
- **Lab 8:** Calculator

Lab Description

The labs for this course provide a practical foundation for creating synthesizable RTL code. All aspects of the design flow are covered in the labs. The labs are written, synthesized, behaviorally simulated, and implemented by the student. The focus of the labs is to write code that will optimally infer reliable and high-performance circuits. The labs culminate in a functional calculator that students verify in simulation.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "Language" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

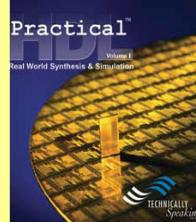
Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Practical HDL is a comprehensive VHDL/Verilog self-paced multimedia training environment.

This tool both reinforces topics covered during the class and offers additional advanced subject matter.

Cost - \$300 USD



Introduction to VHDL

Course Specification

Day 2

- Concurrent and Sequential Statements
- Advanced Process Statements
- **Lab 4:** n-bit Binary Counter and RTL Verification
- Controlled Operation Statements
- **Lab 5:** Comparator
- Behavioral to RTL Coding

Day 3

- Finite State Machines
- **Lab 6:** Arithmetic Logic Unit
- VITAL: VHDL Initiative toward ASIC Libraries
- **Lab 7:** State Machines
- Targeting Xilinx FPGAs
- Functions and Procedures
- **Lab 8:** Calculator

Lab Description

The labs for this course provide a practical foundation for creating synthesizable RTL code. All aspects of the design flow are covered in the labs. The labs are written, synthesized, behaviorally simulated, and implemented by the student. The focus of the labs is to write code that will optimally infer reliable and high-performance circuits. The labs culminate in a functional calculator that students verify in simulation.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "Language" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

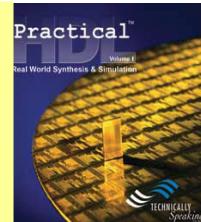
Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Practical HDL is a comprehensive VHDL/Verilog self-paced multimedia training environment.

This tool both reinforces topics covered during the class and offers additional advanced subject matter.

Cost - \$300 USD





LANG11000-7-LEL (v1.0)

Introduction to VHDL Live e-Learning

Course Specification

- Lab 3 – Memory and Record

Session 4

- Concurrent and Sequential Statements
- Advanced Process Statements
- Lab 4 – n-bit Binary Counter and RTL Verification

Session 5

- Controlled Operation Statements
- Lab 5 – Comparator

Session 6

- Behavioral to RTL Coding
- Finite State Machines
- Lab 6 – Arithmetic Logic Unit (ALU)

Session 7

- VITAL: VHDL Initiative toward ASIC Libraries
- Targeting Xilinx FPGAs
- Lab 7 – State Machines

Session 8

- Functions and Procedures
- Lab 8 – Calculator

Lab Descriptions

The labs for this course provide a practical foundation for creating synthesizable RTL code. All aspects of the design flow are covered in the labs. The labs are written, synthesized, behaviorally simulated, and implemented by the student. The focus of the labs is to write code that will optimally infer reliable and high-performance circuits.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Delivery Type** "Live e-Learning" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call +852-2424-5200.

Japan, see the Japanese training schedule at www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



MILAE10000-7-ILT (v1.0)

Course Description

This comprehensive course is a thorough introduction to the Xilinx TMR (XTMR) solution for designs that require Triple Module Redundancy. The XTMR solution incorporates TMRTTool, a proprietary software application that offers total control and flexibility for the TMR process for Xilinx FPGAs.

TMRTTool allows you to easily trade off maximum radiation effect immunity against area, pinout, and board layout consideration.

The XTMR solution consists of TMR and device scrubbing. This combination fully accounts for the unique programmable logic and routing resources in FPGAs, delivering maximum SEU/SET protection. This class covers all those topics.

This one-day course offers valuable hands-on experience, allowing you to evaluate TMR's timing impact, as well as area and pinout considerations. You will also perform design verification to ensure functional integrity for pre- and post-TMR circuits.

Incoming students with little knowledge of SEU/SET considerations will get a thorough overview of how these risks affect technology in general and FPGAs in particular.

Level – Fundamental to Intermediate

Course Duration – 1 day

Price – \$500 USD or 5 Training Credits

Course Part Number – MILAE10000-7-ILT

Who Should Attend? – Any design engineer who creates hardware with TMR requirements. This includes spaced-based deployment or similarly hostile environments

Prerequisites

- Basic digital design knowledge

Software Tools

- TMRTTool 7.1
- ISE™ 7.1i
- ModelSim PE 6.0c

After completing this comprehensive training, you will have the necessary skills to:

- Recognize and address unique FPGA TMR challenges
- Perform comprehensive TMR for Xilinx FPGAs
- Prioritize SEU/SET risks against area and pinout limitations
- Incorporate device scrubbing into TMR strategy
- Create effective timing constraints for TMR design
- Modify testbenches to handle post-TMR circuits
- Incorporate TMRTTool into the standard ISE design flow
- Choose the best overall solution for maximum SEU/SET immunity

Course Outline

- Virtex-II Radiation Effect Summary
- XTMR and Scrubbing Overview
- **Lab 1:** Basic TMRTTool Design Flow
- XTMR and TMRTTool Details
- XTMR and Timing Constraints
- **Lab 2:** Timing Constraints and Design Verification

TMRTTool

Course Specification

- Performance and Application Issues
- **Lab 3:** Performance and Application Issues
- **Lab 4:** TMRTTool Custom Macro Flow

Lab Description

This course is a lab-intensive, one-day workshop that gives you practical hands-on experience with TMRTTool, design verification, timing constraints, and device implementation.

Each lab exercise offers insight to the underlying concepts, while enhancing designer skills and productivity. The exercises are briefly described here.

- **Lab 1:** Basic TMRTTool Design Flow – Incorporate the TMRTTool into the overall ISE design flow, set XTMR options, and export the post-TMR design
- **Lab 2:** Timing Constraints and Design Verification – Update timing constraints for TMR designs, modify the testbench for post-TMR design verification
- **Lab 3:** Performance and Application Issues – Evaluate trade-offs for output registers and bidirectional I/O, and assess impact of half-latch removal
- **Lab 4:** TMRTTool Custom Macro Flow – Create user-defined macros as necessary for critical paths or functional blocks. Replace existing components for TMR circuit. Rerun simulation to ensure pre- and post-TMR logic and functional integrity

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "FPGA" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7750

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



MILAE10000-7-LEL (v1.0)

TMRTTool Live e-Learning

Course Specification

Session 3

- Performance and Application Issues
- **Lab 3:** Performance and Application Issues
- **Lab 4:** TMRTTool Custom Macro Flow

Lab Description

This course is a lab-intensive, one-day workshop that gives you practical hands-on experience with TMRTTool, design verification, timing constraints, and device implementation.

Each lab exercise offers insight to the underlying concepts, while enhancing designer skills and productivity. The exercises are briefly described here.

- **Lab 1:** Basic TMRTTool Design Flow – Incorporate the TMRTTool into the overall ISE design flow, set XTMR options, and export the post-TMR design
- **Lab 2:** Timing Constraints and Design Verification – Update timing constraints for TMR designs, modify the testbench for post-TMR design verification
- **Lab 3:** Performance and Application Issues – Evaluate trade-offs for output registers and bidirectional I/O, and assess impact of half-latch removal
- **Lab 4:** TMRTTool Custom Macro Flow – Create user-defined macros as necessary for critical paths or functional blocks. Replace existing components for TMR circuit. Rerun simulation to ensure pre- and post-TMR logic and functional integrity

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "FPGA" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-602.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7750.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Level – Fundamental to Intermediate

Course Duration – Three 2-hour sessions

Price – \$400 USD or 4 Training Credits

Course Part Number – MILAE10000-7-LEL

Who Should Attend? – Any design engineer who creates hardware with TMR requirements. This includes spaced-based deployment or similarly hostile environments

Prerequisites

- Basic digital design knowledge

Software Tools

- TMRTTool 7.1
- ISE™ 7.1i
- ModelSim PE 6.0c

After completing this comprehensive training, you will have the necessary skills to:

- Recognize and address unique FPGA TMR challenges
- Perform comprehensive TMR for Xilinx FPGAs
- Prioritize SEU/SET risks against area and pinout limitations
- Incorporate device scrubbing into TMR strategy
- Create effective timing constraints for TMR design
- Modify testbenches to handle post-TMR circuits
- Incorporate TMRTTool into the standard ISE design flow
- Choose the best overall solution for maximum SEU/SET immunity

Course Outline

Session 1

- Virtex-II Radiation Effects Summary
- XTMR and Scrubbing Overview
- **Lab 1:** Basic TMRTTool Design Flow

Session 2

- XTMR and TMRTTool Details
- XTMR and Timing Constraints
- **Lab 2:** Timing Constraints and Design Verification



Course Specification

Course Description

Learn how to synthesize an algorithm written in MATLAB into a design that is optimized for a Xilinx FPGA. Find out how to make MATLAB coding changes that improve area and performance. Use the floating-to-fixed point and design exploration features of the AccelDSP Synthesis Tool to achieve maximum results. Merge a synthesized MATLAB block into a larger HDL design or System Generator design.

Level – Fundamental

Course Duration – 2 days

Price – \$1000 USD or 10 Training Credits

Course Part Number – DSP12000-8-ILT

Who Should Attend? – Engineers seeking to develop the necessary skills for designing DSP systems using Xilinx AccelDSP software running with MATLAB

Prerequisites

- Fundamentals of MATLAB and the Filter Design Toolbox
- Basics of digital signal processing theory

Software Tools

- Xilinx ISE™ 8.1i SP1
- Xilinx AccelDSP Synthesis Tool 8.1.690
- MATLAB R14 SP3
- Xilinx System Generator 8.1.1
- Mentor Graphics ModelSim 6.0c PE

After completing this comprehensive training, you will have the necessary skills to:

- Modify a MATLAB script for a DSP algorithm so that it can be synthesized using the AccelDSP Synthesis Tool
- Identify the concepts of quantization as well as specify, monitor, and control bit growth in a MATLAB design
- Modify the MATLAB for a direct form FIR filter into a synthesizable polyphase decimation filter
- Apply coding style changes and AccelDSP directives to optimize a design for performance and efficiency
- Write MATLAB coding changes to add hardware control features to a design
- Merge a synthesized MATLAB block into a larger HDL design
- Export and merge a synthesized MATLAB block into a larger System Generator design

Course Outline

Day 1

- Introduction to AccelDSP Design Flow
- Synthesizable MATLAB
- Quantization
- Multirate Design
- Using AccelWare IP
- Overview of FPGA Architecture

Day 2

- Design Exploration
- Adding Hardware Control
- Coding for Hardware Performance
- Synthesizing Complex Numbers
- Interfacing to System Hardware
- Exporting to System Generator

Lab Descriptions

- **Lab 1: Getting Started** – Learn the basic design flow through the AccelDSP Synthesis Tool.
- **Lab 2: Synthesizable MATLAB** – Modify an unsynthesizable MATLAB design into a design that can be synthesized by AccelDSP.
- **Lab 3: Quantization** – Specify, monitor, and control bit growth in the synthesized design.
- **Lab 4: Multirate Design** – Set up behavioral MATLAB to model the effects of decimation by 2. Create a synthesizable polyphase decimation filter in MATLAB and implement the filter in a Xilinx FPGA.
- **Lab 5: Using AccelWare** – Replace a MATLAB-based polyphase decimation filter with an equivalent Firdecim AccelWare IP block.
- **Lab 6: Design Exploration** – Apply the design exploration features of AccelDSP to optimize a design for area and performance.
- **Lab 7: Adding Hardware Control** – Modify the source MATLAB of a FIR filter to add a serial coefficients load feature.
- **Lab 8: Coding for Hardware Performance** – Learn MATLAB coding techniques to take advantage of even-symmetric coefficients and drive performance over 300 MHz.
- **Lab 9: Synthesizing Complex Numbers** – Explore the methods available for synthesizing designs that use complex numbers.
- **Lab 10: Interfacing to Hardware** – Connect the AccelDSP-generated interface signals to a larger HDL design.
- **Lab 11: Export to System Generator** – Export a MATLAB-based design into a System Generator block and merge the block into a larger System Generator design.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "DSP" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-602.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



Course Specification

Day 2

- Achieving Timing Closure
- **Lab 4:** Review of Global Timing Constraints
- Timing Groups and OFFSET Constraints
- Path-Specific Timing Constraints
- **Lab 5:** Achieving Timing Closure
- Advanced Implementation Options
- **Lab 6:** Designing for Performance
- Power Estimation (Optional)
- **Lab 7:** FPGA Editor Demo (Optional)
- ChipScope™ Pro Analyzer (Optional)
- **Lab 8:** ChipScope Pro Analyzer (Optional)
- Course Summary

Lab Descriptions

- **Lab 1:** CORE Generator Software System – Create a core, instantiate the core into VHDL or Verilog source code, and run behavioral simulation.
- **Lab 2:** Designing Clock Resources – Use the Clocking Wizard to configure DCMs and global clock buffer resources.
- **Lab 3:** Synthesis Techniques – Experiment with different synthesis options and view the results. Versions of this lab are available for Synplicity Synplify Pro, Precision RTL, and Xilinx XST software.
- **Lab 4:** Review of Global Timing Constraints – Use the Constraints Editor to enter global timing constraints.
- **Lab 5:** Achieving Timing Closure – Review timing reports and enter path-specific timing constraints to meet performance goals.
- **Lab 6:** Designing for Performance – Improve performance and maximize results solely with implementation options.
- **Lab 7:** FPGA Editor Demo – Use the FPGA Editor to view a design and add a probe to an internal net.
- **Lab 8:** ChipScope Pro Analyzer – Add an internal logic analyzer to a design to perform real-time debugging.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword "FPGA"** in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-602.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



FPGA23000-8-LEL (v1.0)

Designing for Performance Live e-Learning

Course Specification

Session 2

- Designing with the Digital Clock Manager and PMCD
- **Lab 2 – Designing Clock Resources**

Session 3

- FPGA Design Techniques
- Synthesis Techniques
- **Lab 3 – Synthesis Techniques**

Session 4

- Achieving Timing Closure
- **Lab 4: Review of Global Timing Constraints**

Session 5

- Timing Groups and OFFSET Constraints
- Path-Specific Timing Constraints
- **Lab 5: Achieving Timing Closure**

Session 6

- Advanced Implementation Options
- **Lab 6: Designing for Performance**

Lab Descriptions

- **Lab 1:** CORE Generator Software System – Create a core, instantiate the core into VHDL or Verilog source code, and run behavioral simulation.
- **Lab 2:** Designing Clock Resources – Use the Clocking Wizard to configure the DCMs to generate various clock frequencies and connect the clock signals to global clock buffer resources.
- **Lab 3:** Synthesis Techniques – Experiment with different synthesis options and view the results for Synplicity Synplify Pro, Precision RTL, or Xilinx XST software.
- **Lab 4:** Review of Global Timing Constraints – Use the Constraints Editor to enter global timing constraints.
- **Lab 5:** Achieving Timing Closure – Review timing reports and enter path-specific timing constraints to meet performance goals.
- **Lab 6:** Designing for Performance – Improve performance and maximize results solely with implementation options.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Delivery Type** "Live e-Learning" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

© 2006 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.



FPGA11000-8-ILT

Designing with PlanAhead

Course Specification

- **Lab 8:** Updating the Netlist and Incremental Design
- Floorplanning Strategies
- Course Summary

Lab Descriptions

Note: All labs in this course are also available as self-guided tutorials, which are packaged with the PlanAhead software.

- **Lab 1:** Getting Started with PlanAhead – Illustrates the steps you take to import a synthesized design into the PlanAhead software so that you can begin floorplanning. Also introduces the PlanAhead software environment and views.
- **Lab 2:** Design Analysis and Exploration – Introduces the analysis features of the PlanAhead software that enable early detection of potential design issues, alternate device exploration, initial floorplanning direction, and post-implementation exploration.
- **Lab 3:** Design Partitioning and Top-Level Floorplanning – Introduces the concept of floorplanning. By using automated partitioning tools, you will create a top-level floorplan and experiment with sizing and shaping Pblocks based on the resources assigned to them.
- **Lab 4:** Implementation using ExploreAhead – Introduces the integration of the ISE software implementation tools with the PlanAhead software. Also introduces the new ExploreAhead tool for queuing multiple ISE software runs with varying strategies.
- **Lab 5:** Floorplanning – Describes how to analyze implementation results and to use that information for generating a floorplan aimed at increasing design performance.
- **Lab 6:** Floorplan Tuning – Introduces techniques to help close on timing targets consistently.
- **Lab 7:** Block-Based Design and IP Reuse – Describes the steps to implement a block-based methodology that includes the creation and reuse of an IP module.
- **Lab 8:** Updating the Netlist and Incremental Design – Illustrates the steps needed to update the project top-level netlist and to implement an incremental design change.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "FPGA" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Level – Intermediate

Course Duration – 2 days

Price – \$1000 USD or 10 Training Credits

Course Part Number – FPGA11000-8-ILT

Who Should Attend? – FPGA designers, system architects, and system engineers who are interested in analyzing and driving the physical implementation of their designs to maximize performance and capacity.

Prerequisites

- *Fundamentals of FPGA Design* or equivalent knowledge of the FPGA architecture and the Xilinx ISE™ software flow
- *Designing for Performance* recommended

Software Tools

- Xilinx ISE 8.1i

After completing this comprehensive training, you will have the necessary skills to:

- Import designs into the PlanAhead software project environment
- Analyze design statistics, connectivity, and timing
- Partition and floorplan designs
- Run the Design Rule Checks (DRCs) and implement a design
- Run ExploreAhead to try multiple implementation strategies
- Import and analyze the implementation results to improve the floorplan
- Floorplan to improve performance and consistency
- Use block-based design and create and reuse module-level IP
- Employ incremental design techniques

Course Outline

Day 1

- Course Overview
- **Lab 1:** Getting Started with PlanAhead
- Design Analysis and Exploration
- **Lab 2:** Design Analysis and Exploration
- Design Partitioning and Top-Level Floorplanning
- **Lab 3:** Design Partitioning and Top-Level Floorplanning
- Implementing a Floorplanned Design
- **Lab 4:** Implementation using ExploreAhead

Day 2

- Floorplanning Techniques
- **Lab 5:** Floorplanning
- Tuning a Floorplan for Performance
- **Lab 6:** Floorplan Tuning Lab
- Block-Based Design and IP Reuse
- **Lab 7:** Block-Based Design and IP Reuse
- Netlist Updates and Incremental Design with PlanAhead



V4-23000-8-ILT (v1.0)

Designing with the Virtex-4 Family

Course Specification

Day 2

- Day Two Overview
- I/O and Source-Synchronous Resources
- **Lab 3:** Utilizing Source-Synchronous I/O Resources
- Block RAM Memory Resources
- FIFO16 Memory Resources
- **Lab 4:** Utilizing Block RAM and FIFO16
- XtremeDSP™ Technology Slice
- **Lab 5:** Utilizing XtremeDSP Technology Resources
- Configuration
- Day Two Review

Lab Descriptions

- **Lab 1:** DCM Clocking – Designing a clock management scheme with DCUs and PMCDs.
- **Lab 2:** Clocking Resources – Utilizing global and regional clock networks.
- **Lab 3:** Utilizing Source-Synchronous I/O Resources – Creating a source-synchronous design interface for a network application.
- **Lab 4:** Utilizing Block RAM and FIFO16 – Utilizing new block RAM features and FIFO16-dedicated resources.
- **Lab 5:** Utilizing XtremeDSP Technology Resources – Utilizing the DSP48 block.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword "FPGA"** in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Course Description

Interested in learning how to utilize Virtex™-4 FPGA architectural resources effectively? This course focuses on understanding and utilizing several of the new and enhanced resources found in our newest device. Topics covered include an overview of the Virtex-4 FPGA; the Digital Clock Manager (DCM) and Phase-Matched Clock Divider (PMCD); global and regional clocking techniques; memory and FIFO; and source-synchronous resources. A combination of modules and labs allow for practical hands-on application of the principles taught in this course.

Level – Intermediate

Course Duration – 2 days

Price – \$1000 USD or 10 Training Credits

Course Part Number – V4-23000-8-ILT

Who Should Attend? – Experienced Xilinx users or those who have taken the *Fundamentals of FPGA Design* and *Designing for Performance* courses. Students should have a solid understanding of Virtex-II, Virtex-II Pro, and Virtex-II ProX FPGA architectures, the ISE™ software, timing constraints, and timing closure techniques.

Prerequisites

- *Fundamentals of FPGA Design* course
- *Designing for Performance* course
- Understanding of the Virtex-II, Virtex-II Pro, Virtex-II Pro X FPGA architecture
- Intermediate knowledge of VHDL or Verilog

Software Tools

- Xilinx ISE 8.1i
- Xilinx XST

After completing this comprehensive training, you will have the necessary skills to:

- Describe the Digital Clock Manager (DCM) and Phase-Matched Clock Divider (PMCD) functionality of the Virtex-4 FPGA
- Describe the global and regional clock resources of the Virtex-4 FPGA
- Describe the ILOGIC and OLOGIC blocks
- Describe the ISERDES and OSERDES blocks
- Describe the block RAM features in the Virtex-4 FPGA
- Describe the new FIFO-dedicated resources
- Specify the features of the DSP48 block
- Describe what's new in the configuration of the Virtex-4 FPGA

Course Outline

Day 1

- Introduction
- Product Overview
- DCM Clock Management
- PMCD Clock Management
- **Lab 1:** DCM Clocking
- Clock Networks
- **Lab 2:** Clocking Resources



V4-23000-8-LEL (v1.0)

Designing with the Virtex-4 Family Live e-Learning

Course Specification

Session 4

- I/O and Source-Synchronous Resources
- **Lab 3:** Utilizing Source-Synchronous I/O Resources

Session 5

- Block RAM Memory Resources
- FIFO16 Memory Resources
- **Lab 4:** Utilizing Block RAM and FIFO16

Session 6

- XtremeDSP™ Technology Slice
- **Lab 5:** Utilizing XtremeDSP Technology Resources

Lab Descriptions

- **Lab 1:** DCM Clocking – Designing a clock management scheme with DCMs and PMCDs.
- **Lab 2:** Clocking Resources – Utilizing global and regional clock networks.
- **Lab 3:** Utilizing Source-Synchronous I/O Resources – Creating a source-synchronous design interface for a network application.
- **Lab 4:** Utilizing Block RAM and FIFO16 – Utilizing new block RAM features and FIFO16-dedicated resources.
- **Lab 5:** Utilizing XtremeDSP Technology Resources – Utilizing the DSP48 block.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword "FPGA"** in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-602.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Level – Intermediate

Course Duration – Six 2-hour sessions

Price – \$900 USD or 9 Training Credits

Course Part Number – V4-23000-8-LEL

Who Should Attend? – Experienced Xilinx users or those who have taken the *Fundamentals of FPGA Design* and *Designing for Performance* courses. Students should have a solid understanding of Virtex-II, Virtex-II Pro, and Virtex-II ProX FPGA architectures, the ISE™ software, timing constraints, and timing closure techniques.

Prerequisites

- *Fundamentals of FPGA Design* course
- *Designing for Performance* course
- Understanding of the Virtex-II, Virtex-II Pro, Virtex-II Pro X FPGA architecture
- Intermediate knowledge of VHDL or Verilog

Software Tools

- Xilinx ISE 8.1i
- Xilinx XST

After completing this comprehensive training, you will have the necessary skills to:

- Describe the Digital Clock Manager (DCM) and Phase-Matched Clock Divider (PMCD) functionality of the Virtex-4 FPGA
- Describe the global and regional clock resources of the Virtex-4 FPGA
- Describe the ILOGIC and OLOGIC blocks
- Describe the ISERDES and OSERDES blocks
- Describe the block RAM features in the Virtex-4 FPGA
- Describe the new FIFO-dedicated resources
- Specify the features of the DSP48 block

Course Outline

Session 1

- Introduction
- Product Overview

Session 2

- DCM Clock Management
- PMCD Clock Management
- **Lab 1:** DCM Clocking

Session 3

- Clock Networks
- **Lab 2:** Clocking Resources



V5LX-21000-8-ILT (v1.0)

Designing with the Virtex-5 LX FPGA

Course Specification

Lab Descriptions

The labs will provide practical hands-on application of the principles taught throughout the course.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "FPGA" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-602.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Course Description

Interested in learning how to effectively utilize Virtex™-5 FPGA architectural resources? Targeted towards experienced Xilinx users who have already completed *Fundamentals of FPGA Design* and *Designing for Performance* and have a comprehensive knowledge of Virtex-4 FPGAs, this course focuses on understanding as well as designing into several of the new and enhanced resources found in our newest device.

Topics covered include a Virtex-5 FPGA overview, new LUT, DCM and PLL, global and regional clocking techniques, memory, DSP and arithmetic logic, and source-synchronous resources. A combination of modules and labs allow for practical hands-on application of the principles taught.

Note: Recorded e-Learning modules will also be available. For all other regions, only the recorded e-Learning modules will be available. Also note that the initial course material covers the Virtex-5 LX FPGA platform only. Future revisions will include additional platforms as they become available.

Level – Intermediate

Course Duration – ½ day (will increase to a full day as the other Virtex-5 platforms become available)

Price – \$300 USD or 3 Training Credits

Course Part Number – V5LX-23000-8-ILT

Who Should Attend? – For those who have taken the *Fundamentals of FPGA Design* and *Designing for Performance* courses. A comprehensive knowledge of the Virtex-4 family architecture is also required. This material should be considered a Virtex-5 FPGA update course from the Virtex-4 FPGA family.

Prerequisites

- *Fundamentals of FPGA Design* course
- *Designing for Performance* course
- *Designing with the Virtex-4 Family* course
- Comprehensive knowledge of the Virtex-4 FPGA

Software Tools

- Xilinx ISE 8.1i
- Synplicity Synplify Pro 8.2
- Mentor Graphics Precision 2005b

After completing this comprehensive training, you will have the necessary skills to:

- Describe and utilize new Virtex-5 FPGA resources, including:
 - 6-input LUT
 - DCM and PLL
 - Global and regional clock resources
 - Memory resources
 - Arithmetic and DSP resources

Course Outline

- Virtex-5 FPGA Overview
 - Utilizing Virtex-5 FPGA Logic Resources
 - 6-input LUT
 - Memory
 - DSP
 - Configuration
- Clocking Resources



V5LX-21000-8-LEL (v1.0)

Designing with the Virtex-5 LX FPGA Live e-Learning

Course Specification

Lab Descriptions

The labs will provide practical hands-on application of the principles taught throughout the course.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword "FPGA"** in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-602.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Course Description

Designing with the Virtex-5 Family Live e-Learning delivers high-value training to you remotely via the Internet, with the emphasis on keeping your work schedule and priorities intact. Scheduled over a one-week period, this course comprises a two 2-hour sessions.

Interested in learning how to effectively utilize Virtex™-5 FPGA architectural resources? Targeted towards experienced Xilinx users who have already completed *Fundamentals of FPGA Design* and *Designing for Performance* and have a comprehensive knowledge of Virtex-4 FPGAs, this course focuses on understanding as well as designing into several of the new and enhanced resources found in our newest device. Topics covered include a Virtex-5 FPGA overview, new LUT, DCM and PLL, global and regional clocking techniques, memory, DSP and arithmetic logic, and source-synchronous resources. A combination of modules and labs allow for practical hands-on application of the principles taught.

Note: Recorded e-Learning modules will also be available. For all other regions, only the recorded e-Learning modules will be available. Also note that the initial course material covers the Virtex-5 LX FPGA platform only. Future revisions will include additional platforms as they become available.

Level – Intermediate

Course Duration – Two 2-hour sessions (will increase as the other Virtex-5 platforms become available)

Price – \$300 USD or 3 Training Credits

Course Part Number – V5LX-23000-8-ILT

Who Should Attend? – For those who have taken the

Fundamentals of FPGA Design and *Designing for Performance* courses. A comprehensive knowledge of the Virtex-4 family architecture is also required. This material should be considered a Virtex-5 FPGA update course from the Virtex-4 FPGA family.

Prerequisites

- *Fundamentals of FPGA Design* course
- *Designing for Performance* course
- *Designing with the Virtex-4 Family* course
- Comprehensive knowledge of the Virtex-4 FPGA

Software Tools

- Xilinx ISE 8.1i
- Synplify Synplify Pro 8.2
- Mentor Graphics Precision 2005b

After completing this comprehensive training, you will have the necessary skills to:

- Describe and utilize new Virtex-5 FPGA resources, including:
 - 6-input LUT
 - DCM and PLL
 - Global and regional clock resources
 - Memory resources
 - Arithmetic and DSP resources

Course Outline

- Virtex-5 FPGA Overview
 - Utilizing Virtex-5 FPGA Logic Resources
 - 6-input LUT
 - Memory
 - DSP
 - Configuration
- Clocking Resources

© 2006 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at 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.



DSP10000-8-ILT (v1.0)

DSP Design Flow

Course Specification

- **Lab 2:** Designing a FIR Filter
- HDL Co-Simulation
- **Lab 3:** MAC FIR Filter Verification Using Simultaneous Co-Simulations

Day 2

- Looking Under the Hood
- **Lab 4:** Looking Under the Hood
- Controlling the System
- **Lab 5:** Controlling the System
- Multirate Systems
- **Lab 6:** Designing a MAC-Based FIR Using the DSP48 Slice

Day 3

- Advanced Features
- **Lab 7:** Integrating the ChipScope Pro Analyzer
- **Lab 8:** A System Generator Design as an XPS Peripheral
- **Lab 9:** Multiple Clock Domains Design Using Shared Memories
- **Lab 10:** Improving Design Performance Using Timing Analyzer
- **Lab 11:** Designing Using the PicoBlaze™ MicroController
- **Lab 12:** Creating Parametric Designs

Lab Descriptions

This lab-intensive class gives you hands-on experience by using System Generator for DSP to visualize, simulate, verify, and implement DSP algorithms in Xilinx FPGAs. The labs start at a descriptive level and build on each other. You should expect each successive lesson's challenges to increase. In addition, the labs included in the Advanced Features module provide you experience with other tools such as the ChipScope Pro analyzer and the Embedded Development Kit. System Generator for DSP 8.1 features are identified, including hardware and software co-simulation verification.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword "DSP"** in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



RIO22000-8-ILT (v2.0)

Designing with Multi-Gigabit Serial I/O

Course Specification

Course Description

Learn how to employ RocketIO™ MGT serial transceivers in your Virtex™-II Pro design. Understand and utilize the features of the RocketIO transceiver blocks, such as CRC, 8b/10b encoding, channel bonding, clock correction, and comma detection. Additional highlighted topics include debugging techniques, use of the Architecture Wizard, synthesis and implementation considerations, and standards compliance. This course balances lecture modules and practical hands-on labs.

Level – Intermediate

Course Duration – 2 days

Price – \$1000 USD or 10 Training Credits

Course Part Number – RIO22000-8-ILT

Who Should Attend? – FPGA designers and logic designers

Prerequisites

- Verilog or VHDL experience (or the *Introduction to Verilog* or the *Introduction to VHDL* course)
- Synthesis and simulation experience
- FPGA design experience or the *Fundamentals of FPGA Design* course
- Knowledge of high-speed serial I/O protocols and standards (SONET, Gigabit Ethernet, InfiniBand) is a plus

Software Tools

- ISE 8.1i
- ModelSim PE 6.0

After completing this comprehensive training, you will have the necessary skills to:

- Effectively use all of the advanced RocketIO features, such as CRC, channel bonding, clock correction, comma detection, 8b/10b encoding/decoding, programmable termination, and pre-emphasis
- Utilize the ports and attributes of RocketIO transceivers that control the RocketIO features
- Use the Architecture Wizard to instantiate RocketIO primitives in your design
- Achieve compatibility with high-speed I/O standards by using RocketIO transceivers

Course Outline

Day 1

- Introduction
- Clocking and Resets
- 8b/10b Encoder and Decoder Details
- **Lab 1:** 8b/10b Disparity and Bypass Lab
- Commas and Deserializer Alignment Details
- **Lab 2:** Commas and K-Characters Lab
- Cyclical Redundancy Check Details
- **Lab 3:** Cyclical Redundancy Check Lab
- Clock Correction Details
- **Lab 4:** Clock Correction Lab

Day 2

- Channel Bonding Details
- **Lab 5:** Channel Bonding Lab
- Architecture Wizard Overview
- Implementing a RocketIO Design
- **Lab 6:** Synthesis and Implementation Lab
- IP Overview: Aurora Reference Design
- **Lab 7:** Aurora Protocol Engine Lab
- Common Serial I/O Standards Compliance
- Physical Media Attachment Overview

Lab Descriptions

- **Lab 1:** 8b/10b Disparity and Bypass Lab – Utilize the 8b/10b encoder/decoder and manipulate running disparity. Learn how to bypass the 8b/10b encoder/decoder
- **Lab 2:** Commas and K-Characters Lab – Use programmable comma detection to align a serial data stream
- **Lab 3:** CRC Lab – Modify a design to use the CRC feature for both the user mode and the Fiber Channel mode of CRC
- **Lab 4:** Clock Correction Lab – Utilize the clock correction logic to compensate for frequency differences on the TX and RX side of a link
- **Lab 5:** Channel Bonding Lab – Modify a design to use two transceivers bonded together to form one virtual channel
- **Lab 6:** Synthesis and Implementation Lab – Use the Architecture Wizard to instantiate RocketIO primitives, synthesize a design, and implement the design.
- **Lab 7:** Aurora Protocol Engine Lab – Use the Aurora reference design to send and receive data

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "High-Speed" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548 or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



EMBD21000-8-ILT (v1.0)

Embedded Systems Development

Course Specification

Day 2

- Software Development
- Address Management
- **Lab 4:** Writing Basic Software Applications
- Software Development and Debugging Using SDK
- **Lab 5:** Advanced Software Writing and Debugging Using SDK
- System Simulation
- **Lab 6:** Performing System Simulation

Lab Descriptions

- **Lab 1:** Simple Hardware Design – Create an XPS project by using the Base System Builder to develop a basic hardware system for a target board.
- **Lab 2:** Adding IP to a Hardware Design – Learn to add IP, such as bridges, OPB peripherals, OPB buses, and others, to the basic hardware design.
- **Lab 3:** Adding Custom IP to an Embedded System – Explore adding a custom IP to your design by using the Create and Import Peripheral wizard.
- **Lab 4:** Writing Basic Software Applications – Write a basic C application that utilizes the UART and GPIO.
- **Lab 5:** Advanced Software Writing and Debugging Using SDK – Use the OPB timer and interrupt controller, develop an interrupt service routine, and debug software by using the Software Development Kit (SDK) and debugging tools.
- **Lab 6:** Performing System Simulation – Generate simulation scripts with XPS and perform behavioral simulation.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "Embedded" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



SI20000-6-ILT (v1.0)

Signal Integrity for High-Speed Memory and Processor I/O

Course Specification

Course Description

Learn how signal integrity techniques are applicable to high-speed interfaces between Xilinx FPGAs and semiconductor memories. This course teaches you about high-speed bus and clock design, including transmission line termination, loading, and jitter. You will work with IBIS models and complete simulations using CAD packages. Other topics include managing PCB effects and on-chip termination. This course balances lecture modules and practical hands-on labs.

Level – Intermediate

Course Duration – 2 days

Price – \$1000 USD or 10 training credits

Course Part Number – SI20000-6-ILT

Who Should Attend? – Digital designers, board layout designers, or scientists, engineers, and technologists seeking to implement Xilinx solutions. Also end users of Xilinx products who want to understand how to implement high-speed interfaces without incurring the signal integrity problems related to timing, crosstalk, and overshoot or undershoot infractions.

Prerequisites

- Xilinx FPGA design experience preferred (equivalent of *Fundamentals of FPGA Design* course)

Software Tools

- Mentor Graphics HyperLynx®
- Cadence SPECCTRAQuest®

After completing this comprehensive training, you will have the necessary skills to:

- Identify when signal integrity is important and relevant
- Interpret an IBIS model and correct common errors
- Apply appropriate transmission line termination
- Understand the effect loading has on signal propagation
- Mitigate the impact of jitter
- Manage a memory data bus
- Understand the impact of selecting a PCB stackup
- Differentiate between on-chip termination and discrete termination

Course Outline

Day 1

- Introduction
- Transmission Lines
- **Mentor or Cadence Lab 1**
- IBIS Models
- **Mentor or Cadence Lab 2**
- **Mentor or Cadence Lab 3**
- High-Speed Clock Design
- **Mentor or Cadence Lab 4**
- SRAM Requirements
- **Mentor or Cadence Lab 5**

Day 2

- Physical PCB Structure
- On-Chip Termination
- SDRAM Design
- **Mentor Lab 6**
- Managing an Entire Design

Lab Descriptions

Note: Labs feature the Mentor Graphics or Cadence flow. For private training, please specify your flow to your registrar or sales contact. For public classes, flow will be determined by the instructor based upon class feedback.

- **Mentor Lab 1:** Opening the appropriate Mentor simulator
- **Mentor Lab 2:** Hands-on signal integrity observation of reflection and propagation effects
- **Mentor Lab 3:** Using an IBIS simulator to study basic transmission line effects
- **Mentor Lab 4:** Using saved simulation information to perform power calculation. Also, additional clock simulations
- **Mentor Lab 5:** Observing the effects of coupling on transmission lines
- **Mentor Lab 6:** Demonstrating how an SDRAM module can be handled with an EBD model
- **Cadence Lab 1:** Opening the appropriate Cadence simulator
- **Cadence Lab 2:** Analysis of a simple clock net
- **Cadence Lab 3:** Signal integrity effects caused by multidrop clock networks
- **Cadence Lab 4:** Crosstalk analysis
- **Cadence Lab 5:** Address and data analysis

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "High-Speed" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



EMBD33000-8-ILT (v1.0)

Advanced Features and Techniques of Embedded Systems Development

Course Specification

Day 2

- Performance Tuning
- **Lab 4:** Performance Tuning
- Board Support Packages (BSPs)
- Bus Functional Modeling Simulation
- **Lab 5:** BFM Simulation
- Boot Loader
- **Lab 6:** Boot Loading from Flash Memory

Lab Descriptions

- **Lab 1:** Building a Complete Embedded System – Develop hardware that incorporates IP cores to interface to push buttons, switches, LEDs, an LCD display, and serial communication. Develop an application that interacts with switches, push buttons, an LCD display, and serial communication. Generate and download a bitstream onto a hardware board connected to a server.
- **Lab 2:** External Memory Controllers and File Systems – Design a system that includes an On-Chip Peripheral Bus (OPB) DDR IP core. Develop an application that performs file-related tasks on external memory.
- **Lab 3:** Debugging Using the ChipScope Pro Analyzer – Perform simultaneous hardware and software debugging on stack-related errors with the ChipScope™ Pro Analyzer, GDB, and XMD.
- **Lab 4:** Performance Tuning – Profile a simple piece of code, using the SDK profile tool, running on a processor and go through iterative steps of refinement to improve the performance by using caching and porting a repetitive function to hardware.
- **Lab 5:** BFM Simulation – Create an EDK system that includes IBM CoreConnect bus architecture Bus Functional Models (BFM) and Bus Functional Language (BFL) constructs for an OPB IP. Simulate the OPB bus-based design to verify IP functionality.
- **Lab 6:** Boot Loading from Flash Memory – Develop an application that performs desired tasks. Due to application size and resource limitations, store it in flash, load it through a boot loader program, and execute from external memory.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "Embedded" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

© 2006 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.



FPGA33000-8-ILT (v1.0)

Advanced FPGA Implementation

Course Specification

- **Lab 5:** Divide and Conquer Design Techniques
- Section 3: Reduce Debug Time
- FPGA Editor: Viewing and Editing a Routed Design
- **Lab 6:** FPGA Editor
- **Lab 7:** Reduce Clock Period

Lab Descriptions

Note: Labs will be based on Xilinx ISE 8.1i software.

- **Lab 1:** Timing Analyzer, Constraints, and Closure – Create global timing constraints, read timing reports, apply path-specific constraints (multicycle and false paths), and apply advanced implementation options.
- **Lab 2:** UCF – Write constraints directly into a UCF file to guide the performance results of implementation.
- **Lab 3:** Scripting – Write program commands into a batch file to implement the design. Then modify program switches to obtain the greatest possible performance from the design.
- **Lab 4:** RPM – Create an RPM in a UCF file. Use the Timing Analyzer to find a path that is not meeting timing constraints and identify the components of that path. RLOC the components to create the RPM and improve timing for that path.
- **Lab 5:** Divide and Conquer Design Techniques – Use incremental design techniques and Floorplanner for effective implementation of “divide and conquer” techniques.
- **Lab 6:** FPGA Editor – Use the FPGA Editor to view and edit a design. Analyze the contents of a CLB; add a probe; remove, place, and modify components; and analyze long nets.
- **Lab 7:** Reduce Clock Period – Use all of your obtained knowledge to reduce the clock period delay.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword "FPGA"** in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call +852-2424-5200.

Japan, see the Japanese training schedule at www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Level – Advanced

Course Duration – 2 days

Price – \$1000 USD or 10 Training Credits

Course Part Number – FPGA33000-8-ILT

Who Should Attend? – Engineers who seek advanced training in using Xilinx tools to improve FPGA performance and utilization while also increasing productivity

Prerequisites

- *Fundamentals of FPGA Design*
- *Designing for Performance*
- Intermediate knowledge of Verilog or VHDL is strongly recommended
- At least six months' design experience with Xilinx tools and FPGAs

Software Tools

- Xilinx ISE 8.1i
- Synplicity Synplify
- Mentor Precision
- Xilinx XST

After completing this comprehensive training, you will have the necessary skills to

- Create and edit constraints for hand-placing logic and creating timing constraints
- Build Relationally Placed Macros (RPMs) to improve performance on critical paths
- Run the Xilinx Implementation software from the command line
- Use incremental design techniques in addition to Floorplanner to implement a “divide and conquer” methodology
- Optimize post-Place & Route designs in the FPGA Editor for more efficient in-circuit testing and minor modifications

Course Outline

- Introduction
- **Lab 1:** Timing Analyzer, Constraints, and Closure Review
- Section 1: Advanced Implementation Control
- UCF Editing
- **Lab 2:** UCF
- Command Line Implementation
- **Lab 3:** Scripting
- Creating Your Own RPM
- **Lab 4:** RPM
- Section 2: Timing Enhancement, Fortification, and Preservation
- Divide and Conquer Design Techniques
- Floorplanner: Effective Layout



FPGA33000-8-LEL (v1.0)

Advanced FPGA Implementation Live e-Learning

Course Specification

Session 4

- Divide and Conquer Design Techniques

Session 5

- Floorplanner: Effective Layout
- Lab 4 – Divide and Conquer**

Session 6

- FPGA Editor: Viewing and Editing a Routed Design
- Lab 5 – FPGA Editor**

Lab Descriptions

- Lab 1:** UCF – Write constraints directly into a UCF file to guide the performance results of implementation.
- Lab 2:** Scripting – Write program commands into a batch file to implement the design. Then modify program switches to obtain the greatest possible performance from the design.
- Lab 3:** RPM – Create an RPM in a UCF file. Use the Timing Analyzer to find a path that is not meeting timing constraints, and identify the components of that path. RLOC the components to create the RPM and improve timing for that path.
- Lab 4:** Divide and Conquer – Use incremental design techniques and Floorplanner for effective implementation of “divide and conquer” techniques.
- Lab 5:** FPGA Editor – Use the FPGA Editor to view and edit a design. Analyze the contents of a CLB; add a probe; remove, place, and modify components; and analyze long nets.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Delivery Type** "Live e-Learning" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call +852-2424-5200.

Japan, see the Japanese training schedule at www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Course Description

Advanced FPGA Implementation Live e-Learning delivers high-value training to you remotely via the Internet, with the emphasis on keeping your work schedule and priorities intact. Scheduled over a two-week period, this course comprises a series of six 2-hour sessions.

Advanced FPGA Implementation Live e-Learning tackles the most sophisticated aspects of the ISE™ tool suite and includes labs that provide hands-on experience. This course requires *Fundamentals of FPGA Design* and *Designing for Performance* as prerequisites. An intermediate knowledge of Verilog or VHDL is strongly recommended, as is at least six months of design experience with Xilinx tools and FPGAs.

Level – Advanced

Course Duration – Six 2-hour sessions

Price – \$900 USD or 9 Training Credits

Course Part Number – FPGA33000-8-LEL

Who Should Attend? – Customers who seek advanced training in using Xilinx tools to improve FPGA performance and utilization while also increasing productivity

Prerequisites

- Fundamentals of FPGA Design* course
- Designing for Performance* course
- Intermediate knowledge of Verilog or VHDL is strongly recommended
- At least six months of design experience with Xilinx tools and FPGAs

Software Tools

- Xilinx ISE 8.1i
- Synplicity Synplify
- Mentor Precision
- Xilinx XST

After completing this comprehensive training, you will have the necessary skills to:

- Create and edit constraints for hand placing logic and creating timing constraints
- Build Relationally Placed Macros (RPMs) to improve performance on critical paths
- Use incremental design techniques in addition to Floorplanner to implement a "divide and conquer" methodology
- Optimize post-Place & Route designs in the FPGA Editor for more efficient in-circuit testing and minor modifications

Course Outline

Session 1

- Introduction
- UCF Editing
- Lab 1 – UCF**

Session 2

- Command Line Implementation
- Lab 2 – Scripting**

Session 3

- Creating Your Own RPM
- Lab 3 – RPM**



LANG21000-8-ILT (v1.0)

Advanced VHDL

Course Specification

Lab Descriptions

- **Lab 1:** Modeling – Write a hardware model utilizing generics, subprograms, generate statements, and access data types.
- **Lab 2:** Model Testbench – Write a self-testing testbench and simulate model.
- **Lab 3:** Text IO Testbench – Utilize VHDL Text IO operations in a self-testing testbench.
- **Lab 4:** RTL and Scalable Design – Write a reusable and scalable design block by utilizing synchronous design techniques.
- **Lab 5:** FSM and Scalable Design – Write a Finite State Machine (FSM) by utilizing FSM techniques for a high-performance FSM.
- **Lab 6:** Xilinx and Scalable Design – Optimize the design for Xilinx implementation. Simulate and implement the optimized design.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword "Language"** in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Course Description

Increase your VHDL proficiency by learning advanced techniques that will help you write more robust and reusable code. This comprehensive course is targeted toward designers who already have some experience with VHDL. The course highlights modeling, testbenches, RTL/synthesizable design, and techniques aimed at creating parameterizable and reusable designs. The majority of class time is spent in challenging hands-on labs as compared to lecture modules.

Level – Advanced

Course Duration – 2 days

Price – \$1000 USD or 10 Training Credits

Course Part Number – LANG21000-8-ILT

Who Should Attend? – VHDL users with introductory to intermediate knowledge of VHDL

Prerequisites

- *Introduction to VHDL* course or equivalent knowledge of modeling, simulation, and RTL coding
- At least 6 months of coding experience beyond an introductory course

Software Tools

- Xilinx ISE™ 8.1i
- Mentor Graphics ModelSim PE 6.0c

After completing this comprehensive training, you will have the necessary skills to:

- Write efficient and reusable RTL, testbenches, and packages
- Create self-testing testbenches
- Create realistic models
- Use the Text IO capabilities of the VHDL language
- Store data dynamically
- Create parameterized designs

Course Outline

Day 1

- Course Introduction
- Modeling and Simulation I: Subprograms and Attributes
- Modeling and Simulation II: Access Types and Blocks
- **Lab 1:** Modeling
- Testbench Stimulus
- **Lab 2:** Model Testbench
- Utilizing Text IO
- **Lab 3:** Text IO Testbench

Day 2

- RTL Design and Xilinx
- Design Reuse and Parameterized Design
- **Lab 4:** RTL and Scalable Design
- Finite State Machines
- **Lab 5:** FSM and Scalable Design
- Xilinx-Specific Simulation Issues
- **Lab 6:** Xilinx and Scalable Design
- Course Review



LANG21000-8-LEL (v1.0)

Advanced VHDL Live e-Learning

Course Specification

Session 5

- Finite State Machines
- **Lab 5:** FSM and Scalable Design

Session 6

- Simulation Issues Specific to Xilinx
- **Lab 6:** Xilinx and Scalable Design

Lab Descriptions

- **Lab 1:** Modeling – Write a hardware model utilizing generics, subprograms, generate statements, and access data types.
- **Lab 2:** Model Testbench – Write a self-testing testbench and simulate model.
- **Lab 3:** Text IO Testbench – Utilize VHDL Text IO operations in a self-testing testbench.
- **Lab 4:** RTL and Scalable Design – Write a reusable and scalable design block by utilizing synchronous design techniques.
- **Lab 5:** FSM and Scalable Design – Write a Finite State Machine (FSM) by utilizing FSM techniques for a high-performance FSM.
- **Lab 6:** Xilinx and Scalable Design – Optimize the design for Xilinx implementation. Simulate and implement the optimized design.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Delivery Type** "Live e-Learning" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call +852-2424-5200.

Japan, see the Japanese training schedule at www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Course Description

Advanced VHDL Live e-Learning delivers high-value training to you remotely via the Internet, with the emphasis on keeping your work schedule and priorities intact. Scheduled over a two-week period, this course comprises a series of six 2-hour sessions.

Increase your VHDL proficiency by learning advanced techniques that will help you write more robust and reusable code. This comprehensive course is targeted toward designers who already have some experience with VHDL. The course highlights modeling, testbenches, RTL/synthesizable design, and techniques aimed at creating parameterizable and reusable designs. The majority of class time is spent in challenging hands-on labs as compared to lecture modules.

Level – Advanced

Course Duration – Six 2-hour sessions

Price – \$900 USD or 9 Training Credits

Course Part Number – LANG21000-8-ILT

Who Should Attend? – VHDL users with introductory to intermediate knowledge of VHDL

Prerequisites

- *Introduction to VHDL* course or equivalent knowledge of modeling, simulation, and RTL coding
- At least 6 months of coding experience beyond an introductory course

Software Tools

- Xilinx ISE™ 8.1i
- Mentor Graphics ModelSim PE 6.0c

After completing this comprehensive training, you will have the necessary skills to:

- Write efficient and reusable RTL, testbenches, and packages
- Create self-testing testbenches
- Create realistic models
- Use the Text IO capabilities of the VHDL language
- Store data dynamically
- Create parameterized designs

Course Outline

Session 1

- Course Introduction
- Modeling and Simulation I: Subprograms and Attributes
- Modeling and Simulation II: Access Types and Blocks
- **Lab 1:** Modeling

Session 2

- Testbench Stimulus
- **Lab 2:** Model Testbench

Session 3

- Utilizing Text IO
- **Lab 3:** Text IO Testbench

Session 4

- RTL Design and Xilinx
- Design Reuse and Parameterized Design
- **Lab 4:** RTL and Scalable Design



DSP20000-7-ILT (v1.0)

DSP Implementation Techniques for Xilinx FPGAs

Course Specification

Day 3

- One Filter Does Not Make a System
 - Options to be considered with multiple channels
 - Interpolation and decimation
 - Rate changing and its effect on FIR filter choice
 - Filtering algorithms that exploit device architecture
 - Importance of connectivity versus isolated functions
- Do Not Block the Datapath
 - Numeric controlled oscillators and mixers
 - Strategies for FFT implementation
 - Achieving bandwidth requirements of the FFT
 - Using the FPGA as an efficient co-processor

Course Exercises

- MAC Rates and Memory Requirements
- Constructing a 128-Tap FIR Filter
- Fractional Number Formats
- Twos Complement Arithmetic
- Summation by Addition Tree
- Summation by Addition Chain
- Full Adder: How Many Slices?
- Summation Structure Sizes
- Serial Summation Structure
- 8-Bit by 12-Bit Multiplier
- KCM Multipliers
- Distributed RAM for FIFO
- Size Estimates for Delay Structures
- Using the SRL16E as a FIFO
- Creating Larger RAM Structures
- Selecting a MAC FIR Technique
- Parallel FIR Filter Size
- Symmetry, Interpolation, and Phases
- Decimation Filter
- "fs/4" Mixing and Decimation
- Designing a Numeric Controlled Oscillator (NCO)
- FFT: Benchmarks and Transform Time
- Collection Time = Processing Time
- 128-Point FFT in 1.28 μ s

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "DSP" in the Training Catalog at <https://xilinx.onسابا.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

© 2005 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.



PCI28000-63-ILT (v1.0)

Designing a LogiCORE PCI System

Course Specification

Day 2

- User Application Interface for Target
- User Application Interface for Initiator
- Other User Interface Signals
- Practical Design of PCI Agents
- Xilinx PCI 64-Bit and 66 MHz
- Implementing the LogiCORE PCI
- Device-Specific Considerations
- **Lab 1:** Analyzing PCI Bus Transactions
- **Lab 2:** Synthesis and Implementation Using XST

Lab Descriptions

- **Lab 1:** Analyzing PCI Bus Transactions – This lab demonstrates typical PCI bus transactions and the signals associated with each type of bus transaction. The relationship between various signals is identified. Various settings are changed and the behaviors are analyzed after each change is made.
- **Lab 2:** Synthesis and Implementation Using XST – This lab demonstrates the Xilinx PCI design flow, from synthesis to an implementation targeted for a device that supports Xilinx PCI using ISE 6.3i software. The lab uses the example “Ping” design (included with the PCI Core) to target a Virtex-II™ FPGA. Similar steps can be followed for other device families.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "PCI" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Level – Intermediate

Course Duration – 2 days

Price – \$1000 USD or 10 Training Credits

Course Part Number – PCI28000-6.3-ILT

Who Should Attend? – Engineers who are interested in Xilinx PCI products and who want to maximize their productivity

Prerequisites

- Some knowledge of PCI
- Working experience with digital design
- Basic knowledge of Verilog or VHDL
- Experience with either Xilinx Alliance Series™ or Foundation™ ISE software tools

Software Tools

- ISE 6.3i
- ModelSim 5.8c

After completing this comprehensive training, you will have the necessary skills to:

- Describe the basics of the PCI specification
- Select the appropriate PCI solution for a specific application
- Describe the basic LogiCORE™ PCI design flow
- Use available product documentation to successfully complete a LogiCORE PCI user application, including configuration, logic design, implementation, and verification

Course Outline

Day 1

- Introduction
- PCI Local Bus Architecture
- PCI Signals
- Basic Bus Operations: Transactions, Decoding, and Wait States
- Basic Bus Operations: Target, Parity, and Arbitration
- PCI Addressing and Bus Commands
- PCI Configuration
- 64-Bit and 66-MHz PCI
- PCI Timing
- The LogiCORE PCI Solution
- Xilinx 32/33 and 64/66 LogiCORE PCI



PCI28000-63-LEL (v1.0)

Designing a LogiCORE PCI System Live e-Learning

Course Specification

Session 3

- 64-Bit and 66-MHz PCI
- PCI Timing
- The LogiCORE PCI Solution

Session 4

- Xilinx 32/33 and 64/66 LogiCORE PCI
- User Application Interface for Target
- User Application Interface for Initiator

Session 5

- Other User Interface Signals
- Practical Design of PCI Agents
- Xilinx PCI 64-bit and 66 MHz

Session 6

- Implementing the LogiCORE PCI
- Device-Specific Considerations
- **Lab 1:** Analyzing PCI Bus Transactions
- **Lab 2:** Synthesis and Implementation Using XST

Lab Descriptions

- **Lab 1:** Analyzing PCI Bus Transactions – This lab demonstrates typical PCI bus transactions and the signals associated with each type of bus transaction. The relationship between various signals is identified. Various settings are changed and the behaviors are analyzed after each change is made.
- **Lab 2:** Synthesis and Implementation Using XST – This lab demonstrates the Xilinx PCI design flow, from synthesis to an implementation targeted for a device that supports Xilinx PCI using ISE 6.3i software. The lab uses the example “Ping” design (included with the PCI Core) to target a Virtex™-II FPGA. Similar steps can be followed for other device families.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Delivery Type** "Live e-Learning" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

© 2005 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.



PCIX28000-63-ILT (v1.0)

Designing a LogiCORE PCI-X System

Course Specification

Day 2

- The LogiCORE PCI-X Solution
- Xilinx LogiCORE PCI-X
- User Application Interface for Target
- User Application Interface for Initiator
- Design Considerations in LogiCORE PCI-X
- Designing Target Agent Using LogiCORE PCI-X
- Designing Initiator Agent Using LogiCORE PCI-X
- Implementing the LogiCORE PCI-X
- **Lab 2:** Analyzing PCI-X Bus Transactions
- **Lab 3:** Synthesis and Implementation Using XST

Lab Descriptions

- **Lab 1:** Analyzing PCI and PCI-X Transactions – This lab introduces you to the PCI and PCI-X protocol and their typical transactions. The simulation will show different aspects of PCI and PCI-X transactions.
- **Lab 2:** Analyzing PCI-X Bus Transactions – This lab demonstrates typical PCI-X bus transactions and the signals associated with each type of bus transaction. The relationship between various signals is identified. Some settings are changed and the behaviors are analyzed after the change is made.
- **Lab 3:** Synthesis and Implementation Using XST – This lab demonstrates the Xilinx PCI-X design flow, from synthesis to an implementation targeted for a device that supports Xilinx PCI-X using ISE 6.3i SP1 software. The lab uses the simple transactions example design (included with the PCI-X Core) to target a Virtex™-II Pro device. Similar steps can be followed for other device families.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "PCI" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



PCIX28000-63-LEL (v1.0)

Designing a LogiCORE PCI-X System Live e-Learning

Course Description

This course focuses on the PCI-X Addendum to the PCI Local Bus Specification and provides a detailed investigation into the operation of the LogiCORE™ PCI-X solution. Explaining the basic principles and concepts introduced by the PCI-X Addendum, this course also provides an in-depth understanding of the LogiCORE PCI-X solution and how a digital designer may interface this solution to a typical user application to create a flexible PCI-X solution.

Level – Intermediate

Course Duration – Six two-hour sessions scheduled sequentially over a two-week period

Price – \$800 USD or 8 Training Credits

Course Part Number – PCIX28000-6.3-LEL

Who Should Attend? – This course is beneficial to any digital designer tasked with designing a PCI-X system where overall time-to-market is an important factor. This course is also beneficial to any engineer with a general interest in learning about PCI-X and the Xilinx PCI-X solution.

Prerequisites

- Experience with either VHDL or Verilog languages
- Familiar with the Xilinx software tools

Software Tools

- Xilinx ISE™ 6.3
- Mentor Graphics ModelSim 6.0 (SE or PE)

After completing this comprehensive training, you will have the necessary skills to:

- Identify differences between PCI and PCI-X
- Describe the basics of the PCI-X specification
- Describe the basic Xilinx PCI-X design flow
- Identify LogiCORE PCI-X signals
- Understand the behavior of the PCI-X LogiCORE solution and how to interface it to a typical user application
- Design a user application that can accommodate operation in both conventional PCI and PCI-X mode

Course Outline

Session 1

- Course Agenda
- PCI-X Local Bus Architecture
- PCI-X Signals and Terminology
- Basic Bus Operations: Transactions, Decoding, and Wait States

Session 2

- Basic Bus Operations: Terminations, Parity, and Arbitration
- PCI-X Modes

Session 3

- PCI-X Addressing and Bus Commands
- PCI-X Configuration
- 64-Bit Transactions

Session 4

- **Lab 1:** Analyzing PCI and PCI-X Transactions
- The LogiCORE PCI-X Solution

Course Specification

- Xilinx LogiCORE PCI-X
- User Application Interface for Target

Session 5

- User Application Interface for Initiator
- Design Considerations in LogiCORE PCI-X
- Designing Target Agent Using LogiCORE PCI-X
- Designing Initiator Agent Using LogiCORE PCI-X

Session 6

- Implementing the LogiCORE PCI-X
- **Lab 2:** Analyzing PCI-X Bus Transactions
- **Lab 3:** Synthesis and Implementation Using XST

Lab Descriptions

- **Lab 1: Analyzing PCI and PCI-X Transactions** – This lab introduces you to the PCI and PCI-X protocol and their typical transactions. The simulation will show different aspects of PCI and PCI-X transactions.
- **Lab 2: Analyzing PCI-X Bus Transactions** – This lab demonstrates typical PCI-X bus transactions and the signals associated with each type of bus transaction. The relationship between various signals is identified. Some settings are changed and the behaviors are analyzed after the change is made.
- **Lab 3: Synthesis and Implementation Using XST** – This lab demonstrates the Xilinx PCI-X design flow, from synthesis to an implementation targeted for a device that supports Xilinx PCI-X using the ISE 6.3i SP1 software. This lab uses the simple transactions example design (included with the PCI-X Core) to target a Virtex™-II Pro device. Similar steps can be followed for other device families.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Delivery Type** "Live e-Learning" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



PCIE28000-8-ILT (v1.0)

Designing a LogiCORE PCI Express System

Course Specification

Lab Descriptions

- **Lab 1:** Using the Local Link Interface – Introduces the PCI Express core design that will also be used in Lab 2. It familiarizes you with the core user application interface (Local Link) and with modifying the design to change the packets being sent.
- **Lab 2:** Exploring the Configuration Space – Reinforces lessons learned in the previous modules by having you decode configuration packets to understand the requirements in configuring the core. In addition, you will see how to implement user configuration space.
- **Lab 3:** Designing with the PCI Express Core – Takes an in-depth look at designing with the core. You will become familiar with packet ordering and credits available.
- **Lab 4:** PCI Express DMA Design Example – Allows you to see how a simple, single-channel DMA example can be used with the PCI Express Core. You will also become familiar with allocating completion space for inbound completions.
- **Lab 5:** Generating a Xilinx PCI Express Core – Illustrates using the CORE Generator™ software to generate a core. The core is then implemented and you can verify the implementation by studying the various reports created by the Xilinx tools.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword "PCI"** in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

Course Description

By learning PCI Express core protocol fundamentals, designers will gain a working knowledge of how PCI Express can be used in their systems. This course focuses on the PCI Express protocol subjects that designers, using the Xilinx PCI Express core, should understand to complete their designs faster and more easily. Students will also be introduced to each Xilinx PCI Express core product and will gain intimate knowledge of how the PCI Express core operates.

Level – Intermediate

Course Duration – 2 days

Price – \$1000 USD or 10 Training Credits

Course Part Number – PCIE28000-8-ILT

Who Should Attend? – Engineers who seek training in developing the necessary skills for designing PCI Express systems using Xilinx PCI Express cores

Prerequisites

- Basic PCI and/or PCI-X protocol knowledge
- Basic knowledge of Verilog or VHDL
- Basic experience with commonly used simulation tools like ModelSim
- Basic knowledge of Xilinx ISE™ software

Software Tools

- Xilinx ISE 8.1i
- Mentor Graphics ModelSim 6.0c PE

After completing this comprehensive training, you will have the necessary skills to:

- Effectively use the Xilinx PCI Express cores in your own design environments
- Select the appropriate PCI solution for a specific application
- Identify how PCI Express specification requirements apply to using Xilinx PCI Express cores

Course Outline

Day 1

- Overview
- Layers and Channels
- TLP Fields and Packet Routing
- PCI Express Local Link Interface
- **Lab 1:** Using the Local Link Interface
- PCI Express Configuration Space
- **Lab 2:** Exploring the Configuration Space

Day 2

- TLP Request and Completion Packets
- Generating Interrupts
- PCI Express Core Design Considerations
- **Lab 3:** Designing with the PCI Express Core
- PCI Express DMA Design Example
- **Lab 4:** PCI Express DMA Design Example
- Clocking and Other Physical Layer Topics
- Xilinx PCI Express Solutions
- **Lab 5:** Generating a Xilinx PCI Express Core



Xilinx Global Services

Finish Faster

Time is money, and you need every advantage to minimize your R&D design risk. How do you reduce your development time in order to drive tangible bottom-line benefits, such as lower overall project costs, while bringing your products to market ahead of the competition?

Xilinx® Design Services (XDS) will improve the success of your project by augmenting your design team with experts in FPGA and embedded software design techniques and solutions. Led by professionals with proven project management skills, our industry-recognized FPGA hardware engineers, embedded software designers, product specialists, and system architects will make a difference.

Together, your design team and our XDS specialists can get your most advanced projects completed on time, within budget, and with optimal performance.

Xilinx Design Services

Lower Costs and Accelerated Productivity



Helping You Solve Your Design Challenges

Whether it's helping you create your product from an initial concept, implementing your specifications, modifying existing intellectual property (IP) cores, or improving product performance, Xilinx Design Services provides extensive FPGA hardware and embedded software design experience to solve even the most complex design challenge. When you partner with XDS, you benefit from:

- Depth of experience and expertise with Xilinx device and software technologies
- Proven project management success
- Extensive experience with Xilinx embedded tool flows and embedded design for Xilinx platform FPGAs
- Access to source code and special knowledge of Xilinx IP cores
- Detailed understanding of the Xilinx ISE "back-end implementation" tools

"Xilinx suite of solutions provided the least risk to the design and by using Xilinx Design Services, we avoided hiring additional engineers for this project. We were guaranteed an expert design team and they completed the task ahead of schedule."

Pradeep Samudra
VP, Samsung Telecommunications Broadband Network Lab

The Xilinx Design Services Portfolio

The XDS team offers several services to meet your needs across the complete development lifecycle—from requirements capture, through design, implementation and testing, to final acceptance.

System FPGA Design – Depth of experience with Xilinx technologies counts when you need help with system architecture consulting or FPGA logic design. XDS customers benefit from advanced Virtex™-II Pro and Spartan™-3 FPGAs and tools that include ISE, ChipScope™ Pro, and System Generator for DSP.

Design from Specification – XDS has the proven project management success you need in developing projects such as turnkey FPGA designs, ASIC conversions, and the latest applications in DSP, embedded processor, and parallel or high-speed serial I/O design.

Embedded Software – Our team of embedded engineers has extensive experience designing with the PowerPC™ and MicroBlaze™ processors, as well as the Embedded Development Kit. We develop complex embedded software with real-time constraints using hardware/software co-design techniques.

IP Core Modification and Integration – XDS has unrivaled access to source code, plus the experience with Xilinx LogiCORE™ products to successfully modify, integrate, and optimize these tools to meet customer needs for DSP, Processor, or I/O applications.

A Dedicated Applications Engineer – Titanium Technical Service solves your resource requirements with a dedicated applications engineer whose knowledge of the Xilinx implementation tools helps you meet device timing in your design.

The Xilinx Advantage

We focus on doing one thing better than anyone else — implementing your ideas into FPGAs with maximum efficiency and minimum risk.

Customer satisfaction comes first. We work hard to ensure that our commitments are met and to build long-term partnerships with our customers. With partnership comes flexibility, the type you can rely on to respond quickly to emerging business opportunities. You get a *complete* programmable logic design solution:

- Products completed on time
- Developed in a more cost-effective manner
- Featuring higher quality than the competition

Aggressive innovation is what sets Xilinx apart. No other company can provide complete FPGA design solutions supported by as much experience and expertise. Xilinx Global Services helps you fully realize the advantages of programmable logic by providing global technical support and services, education, and a portfolio of Design Services that help you finish faster.

Find Out for Yourself

To find out more about how Xilinx Design Services can improve the success of your project, email us at designservices@xilinx.com or visit our website at support.xilinx.com/xds.



Corporate Headquarters

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: (408) 559-7778
Fax: (408) 559-7114
Web: www.xilinx.com

Europe

Xilinx, Ltd.
Benchmark House
203 Brooklands Road
Weybridge
Surrey KT13 ORH
United Kingdom
Tel: 44-870-7350-600
Fax: 44-870-7350-601
Web: www.xilinx.com

Japan

Xilinx, K.K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific

Xilinx, Asia Pacific
Unit 1201, Tower 6, Gateway
9 Canton Road
Tsimshatsui, Kowloon,
Hong Kong
Tel: 852-2-424-5200
Fax: 852-2-494-7159
E-mail: ask-asiapac@xilinx.com



The Programmable Logic Company™

FORTUNE®
100 BEST COMPANIES TO WORK FOR
2003

© 2003 Xilinx Inc. All rights reserved. The Xilinx name, logo, Spartan-3 and Virtex-II Pro are registered trademarks; ChipScope, LogiCORE, MicroBlaze, and Xtreme DSP are trademarks; and The Programmable Logic Company is a service mark of Xilinx Inc. All other trademarks are the property of their owners.



Xilinx Global Services
Finish Faster

Programmable logic technology is rapidly changing. There are many things you need to know about the devices, the tools, and the services you use to create your systems. How do you find all the answers and get the immediate help you need to complete your design?

The Xilinx MySupport website is your best resource for keeping up to date on the latest product news and technical information. It's also a fast and easy way for you to solve most of the technical problems you might encounter. And, to further improve your productivity – and ultimately save development cost, you can easily personalize this website so you get all the information you need and only the information you need, the way you want it. There is no better way to stay informed or to get help quickly.

MySupport.xilinx.com

Personalized Technical Support



Here's What You'll Find

MySupport.xilinx.com gives you the same access to tools as on support.xilinx.com but with the added benefit of the ability to personalize your page. The MySupport.xilinx.com web site gives you instant access to information such as datasheets, application notes, Xcell Journal articles, software manuals, the Answer Database, and much more. And, all of this is searchable so you can find every reference to your specific question, quickly. Plus you can easily add bookmarks to retrace your steps as you compile all the answers you need. If that's not enough, you can also customize the layout and the colors to suit your preferences. There are many options that you can add to your personalized website, including:

- **Automatic MyAlerts** – Become instantly informed when any new information is posted that meets your specific needs.
- **Support News** – Access the latest information about Xilinx software and IP (cores).
- **What's New** – Access the latest information about products, silicon solutions, and design resources.
- **Tech Tips** – Access the latest technical information about development tools, device families, interface tools, and Virtex-II Pro™ FPGAs.
- **Forums** – Collaborate with other designers in discussion groups or chat rooms; join the popular newsgroup comp.arch.fpga.
- **Answer Database** – Use our Advanced Search or Answer Browser tools to search our database archives, technical tips, application notes, or software manuals. Easily access the latest technical answers from our huge database of over 4,000 answers, indexed in logical categories.
- **Problem Solvers** – Get instant help for installation and configuration, PCI applications, JTAG implementation; automatically troubleshoot your configuration or installation problem using a series of diagnostic questions within an interactive tool that can save you hours of work.

Here's what you will see when you log on to MySupport.xilinx.com...

Support News

- ISE 5.1i Service Pack 2 is now available!
- IP Update #1 is now available for all ISE 5.1i users.
- Check out the latest work of author Peter Alfke: "[Metastability Delay and Mean Time Between Failures in Virtex™-II Pro Flip-Flops](#)".

Xilinx Global Services

- [Virtex-II Pro Training](#) - New Virtex-II Pro Free Technical Lecture
- What's New in ISE 5.1i?** Six new [Free Recorded e-learning sessions](#)
- [Fundamentals of FPGA Design](#) and [Designing for Performance](#) courses have been updated for ISE 5.1i.
- NEW! RocketI/O Multi-Gigabit Transceivers [Technical Training Class](#)

Tech Tips

- Xilinx Development Tools :**
 - Embedded Development Kit
 - Xilinx Synthesis Technology
 - Becoming Familiar with Xilinx Design Tools
- Device Families :**
 - Getting Familiar with Xilinx Devices
 - MicroBlaze Soft Processor
 - Virtex Series
- Interface Tools :**
 - Becoming Familiar with Xilinx Support
 - Formal Verification
 - MicroBlaze Soft Processor
- Virtex-II Pro :**
 - RocketI/O
 - Developer's Kit

What's New

- Products :**
 - IQ Solutions
- Silicon Solutions :**
 - Design Resources
- System Resources :**
 - Application Notes and Reference Designs

My Alerts

- [Tech Tips \(8 items, 8 new\)](#)
11/25/02 11:54:23
- [Products \(14 items, 14 new\)](#)
11/25/02 11:52:04
- [FPGA Data Sheets \(4 items, 4 new\)](#)
11/18/02 11:43:55
- [PROM Data Sheets \(1 items, 1 new\)](#)
11/18/02 11:43:55
- [Most Recent Answers \(2 items, 2 new\)](#)
11/18/02 11:40:55
- [CPLD Data Sheets \(1 items, 1 new\)](#)
11/11/02 11:43:29

Latest Answers

- Answers you last viewed :**
- Most Recent Answers :**
- Popular Answers :**
- [Today's Answers](#)
- [Yesterday's Answers](#)
- [Last 7 Days' Answers](#)

Increase your productivity
 Go to Mysupport.xilinx.com and personalize the web site to receive Automatic alerts.

Corporate

Xilinx, Inc.
 2100 Logic Drive
 San Jose, CA 95124
 Tel: 408-559-7778
 Fax: 408-559-7114
 Web: www.xilinx.com

Europe

Xilinx, Ltd.
 Benchmark House
 203 Brooklands Road
 Weybridge
 Surrey KT13 ORH
 United Kingdom
 Tel: 44-1-870-7350-600
 Fax: 44-1-870-7350-601
 Web: www.xilinx.com

Japan

Xilinx, K. K.
 Shinjuku Square Tower 18F
 6-22-1 Nishi-Shinjuku
 Shinjuku-ku, Tokyo
 163-1118, Japan
 Tel: 81-3-5321-7711
 Fax: 81-3-5321-7765
 Web: www.xilinx.co.jp

Asia Pacific

Xilinx, Asia Pacific
 Unit 1201, 12/F, Tower 6
 Gateway
 9 Canton Road
 Tsimshatsui, Kowloon
 Hong Kong
 Tel: 852-2-424-5200
 Fax: 852-2-494-7159
 E-mail: ask-asiapac@xilinx.com

FORTUNE®
 100 BEST COMPANIES TO WORK FOR
 2003





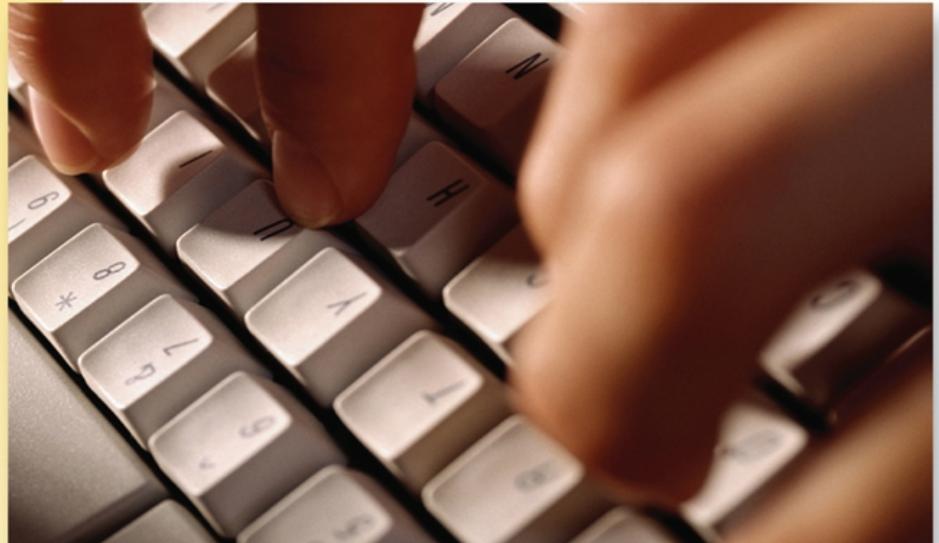
SUPPORT SERVICES
Xilinx Global Services

Design complexity, tough schedules, and constantly changing technology can make it very difficult to succeed in today's fast-paced marketplace. How do you keep up with all the changes and still produce new designs with ever more performance and lower costs?

Our Titanium Technical Service provides you with a dedicated application engineer and is an excellent way to make sure you have everything you need, when you need it. Make use of our in-house experts to reduce the risk complexity and ultimately design cost of your next design – and get your products to market faster than ever before. There is no better way to get the help you need to be more productive.

Titanium Technical Service

It's All About Efficiency



Expertise is the Key

With a dedicated application engineer, Titanium Technical Service will improve your productivity, accelerate your time to market, increase your expertise, minimize your risk, and help you produce the best possible designs. You save development cost, and may fit your design in a lower-cost FPGA. It will also insure your product success by providing you with the best expertise in the industry, including:

- **Dedicated Application Engineers** – Our expert engineers are available, on a contract basis, at your site, at Xilinx, or both. Our engineers can fully understand your needs and requirements, and will help you solve your design problems, quickly. You will achieve the fastest clock speeds and quicker timing closure.
- **Design Flow Coaching** – Our application engineers can provide one-to-one or one-to-many coaching, to help you understand how to use our tools to your best advantage. We'll teach you the latest techniques for floorplanning, timing constraints, timing analysis, and much more.
- **Factory Escalation Process** – We can help you resolve any technical problem through immediate access to our factory personnel. We can also help you accelerate your production and save money.

Example Engagements

Here are several examples of customer engagements where Titanium Technical Service helped our customers meet their design goals.

Example 1

This customer was very experienced with Xilinx products, but was unable to get a design to meet their timing budget. The project was nearing an important deadline requiring the engineer to achieve critical clock speed. Within a week, the Titanium Technical Service Application Engineer got the design to meet the required specifications.

Example 2

This customer was new to Xilinx designs, and was beginning a project with an aggressive timeline. They contracted with Xilinx for a Titanium Technical Service Application Engineer to provide a week of design and tool methodology coaching. The engineer assisted the team with HDL code reviews and HDL simulation, and identified several areas for improvement. The customer met their timeline and deliverables.

Example 3

This customer had many design groups that had varying levels of experience with Xilinx tools. These groups were all doing independent designs and required some degree of coaching regarding their specific needs. Our engineer resided at the customer's site for two weeks and met individually with each of the designers to address their design challenges. In all cases, the Titanium Technical Service Application Engineer increased the engineers' design knowledge and skills, and helped them meet their design goals.

Take the Next Step

Our contract method gives you control over your Titanium-related expenses. There are specific start and end dates written into the contract, and your Titanium Technical Service Application Engineer and Account Manager can provide you with regular status reports, including project plans, pre-scheduled status reports, timeline reviews, and other custom deliverables.

To purchase a Titanium Technical Service contract:

In North America, contact your Xilinx Sales Representative or Xilinx support representative at: 1-800-888-FPGA (3742).

In Europe, call 44-870-7350-532, or email us at eurotitanium@xilinx.com.

In Japan, contact your local distributor.

Corporate

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: 408-559-7778
Fax: 408-559-7114
Web: www.xilinx.com

Europe

Xilinx, Ltd.
Benchmark House
203 Brooklands Road
Weybridge
Surrey KT13 ORH
United Kingdom
Tel: 44-1-870-7350-600
Fax: 44-1-870-7350-601
Web: www.xilinx.com

Japan

Xilinx, K. K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific

Xilinx, Asia Pacific
Unit 1201, 12/F, Tower 6
Gateway
9 Canton Road
Tsimshatsui, Kowloon
Hong Kong
Tel: 852-2-424-5200
Fax: 852-2-494-7159
E-mail: ask-asiapac@xilinx.com

FORTUNE®
2002
100 BEST COMPANIES TO WORK FOR

© 2002 Xilinx Inc. All rights reserved. The Xilinx name and logo are registered trademarks, and The Programmable Logic Company is a service mark of Xilinx Inc.

Printed in U.S.A.





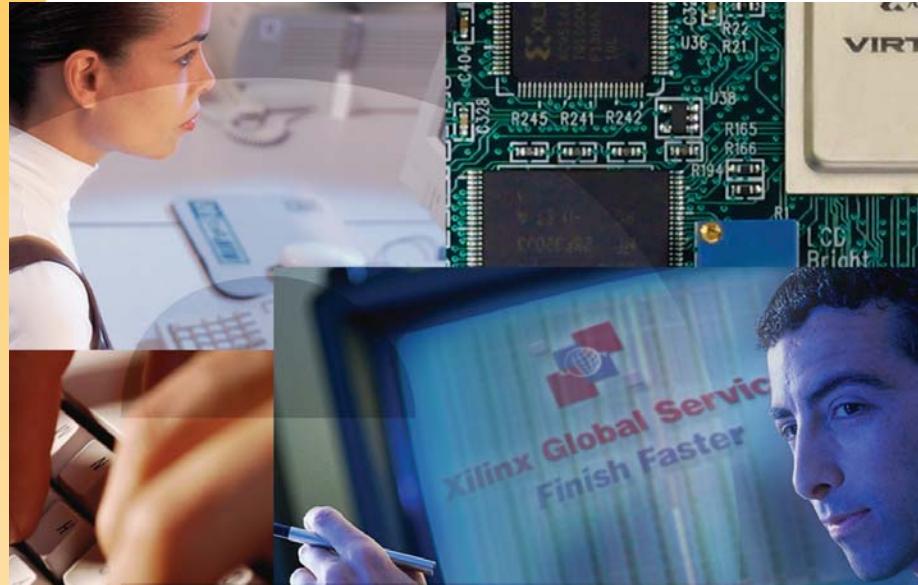
Xilinx Productivity Advantage

Today's designers are reducing system costs, power consumption, and design time by implementing programmable systems that combine hard and soft processors on an FPGA platform. However, co-designing across processor and logic domains can present a significant learning curve for embedded software engineers and logic designers alike.

To help you ramp quickly and hit your project targets with minimal risk, Xilinx offers Embedded Processing QuickStart!, the newest in our line of Xilinx Productivity Advantage bundled solutions.

Embedded Processing QuickStart! is unique in the industry, offering an unprecedented level of on-site design support and training from Xilinx, the leader in embedded processing technology for FPGAs. Designed for the critical initial design phase of your project, Embedded Processing QuickStart! ensures that you and your team will finish On Budget, On Time, Every Time with the Xilinx embedded processing design environment.

Embedded Processing QuickStart!



Get the most from your Embedded Solution

With solutions ranging from the soft-core PicoBlaze™ and MicroBlaze™ processors to the embedded PowerPC™ available in Virtex-II Pro™ and Virtex-4™ FPGAs, Xilinx is leading the way in embedded processing performance.

The Embedded Processing QuickStart! solution delivers individualized service that includes a QuickStart! application engineer at your site for a week. This Xilinx expert will train your hardware team on creating embedded systems, instruct software engineers on how to optimally utilize supporting FPGA features, and help your team maximize performance.

Embedded Processing QuickStart! includes the following:

- Configuration of the Xilinx design environment
- An instructor-led Embedded Systems Development course
- Design architecture/implementation consultation and guidance
- Guidance with system partitioning
- Initial design techniques to enable faster and more effective debug and verification
- Development of a comprehensive training plan

The on-site Xilinx expert provides a variety of services:

- Educates your team on developing procedures to ease verification and debug
- Assists your hardware team in creating a software application in "C"
- Helps your software engineers fully exploit Xilinx technology
- Instructs your team on how to partition between hardware and software for maximum performance

Your Complete Embedded Systems Solution

Embedded Processing QuickStart! provides the end-to-end service you need to create more successful designs.

Configuration of the Xilinx Design Environment

This feature ensures that your team has a properly configured development environment from the start of the project, which is a critical consideration in the design of any embedded system.

Embedded Processing and EDK (Embedded Development Kit)

Training Class

This class brings designers up to speed on the capabilities and characteristics of the Xilinx MicroBlaze 32-bit soft processor core, the hard embedded IBM PowerPC™ core in the Virtex-II Pro/Virtex-4 FPGAs, and the Embedded Development Kit (EDK) design environment. Your team will learn how to develop embedded systems using hard or soft processor cores and a set of soft peripherals.

Design Architecture/Implementation Consultation

The QuickStart! on-site application engineer provides a consultation to ensure that your team achieves optimal performance for the selected technology. By minimizing design risks and reducing the time needed for hardware and software debugging, you will get to market faster. These two areas are improved with proper planning and the solid design methodology provided by the Embedded Processing QuickStart! service.

Guidance with System Partitioning

This feature helps your design team determine which operations are better suited for CPU or FPGA fabric. A properly partitioned design yields a higher performance system by fully utilizing the best features of your selected device.

Comprehensive Training Plan

Xilinx helps you develop a training plan tailored specifically to the needs of your design team, which can include system partitioning, CPU programming, "C" programming, debugging, and verification. This plan helps prevent schedule slips later in the project by ensuring that your team is skilled in the required disciplines. It also helps you maintain a more effective and highly motivated team.

Take the Next Step

To find out more about Embedded Processing QuickStart!, contact your Xilinx representative or go to www.xilinx.com/epq



Corporate Headquarters

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: 408-559-7778
Fax: 408-559-7114
Web: www.xilinx.com

Europe Headquarters

Xilinx, Ltd.
Citywest Business Campus
Saggart,
Co. Dublin
Ireland
Tel: +353-1-464-0311
Fax: +353-1-464-0324
Web: www.xilinx.com

Japan

Xilinx, K. K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific

Xilinx, Asia Pacific Pte. Ltd.
No. 3 Changi Business Park Vista, #04-01
Singapore 486051
Tel: (65) 6544-8999
Fax: (65) 7689-8886
RCD no: 20-0312557-M
Web: www.xilinx.com

Distributed By:



FORTUNE®
2005
100 BEST COMPANIES TO WORK FOR

© 2005 Xilinx Inc. All rights reserved. The Xilinx name and logo are registered trademarks; Virtex-4, Virtex-II Pro, PicoBlaze, and MicroBlaze are trademarks; and The Programmable Logic Company is a service mark of Xilinx Inc. PowerPC is a trademark of International Business Machines Corporation in the United States, or other countries, or both. All other trademarks are the property of their owners.



Xilinx Productivity Advantage

As the industry moves towards the total integration of hardware, complete device utilization and optimization are becoming essential for programmable logic designers. However, these increasing densities present significant design challenges.

Xilinx, a leader in programmable logic design methodologies, maximizes your system performance and minimizes risk with our bundled PlanAhead QuickStart! offering, the newest member of the Xilinx Productivity Advantage bundled solutions portfolio. This service offering is designed to shorten development cycles and limit scheduling challenges associated with maximizing system performance and integration.

PlanAhead QuickStart! is unique in the industry, offering an unprecedented level of support during the most critical phase of a project — the design phase. With this service, Xilinx guarantees risk reduction and faster project completion by providing on-site training and support to ensure that your team is competent and confident with the design environment.

PlanAhead QuickStart!



Get the Most from Your Multi-Platform FPGA

Xilinx has a history of creating multi-platform solutions to meet the needs of almost any system. Our breakthrough Virtex-4™ FPGA family is the perfect example of this commitment, consisting of multiple devices and three domain-optimized platforms — LX for logic-intensive designs, SX for high-performance signal processing, and FX for high-speed serial connectivity and embedded processing.

PlanAhead QuickStart! makes it easier than ever before to optimize this unique technology in your design. You will receive a QuickStart! engineer at your site for one week. During that period, the Xilinx expert will train and empower your team to complete your project on time and make best use of the Xilinx device you've selected. This program includes:

- Configuration of the Xilinx design environment
- An instructor-led floorplanning systems with PlanAhead course
- Design architecture/implementation consultation
- Development of a comprehensive training plan

Your Complete System Floorplanning Solution

The PlanAhead QuickStart! offering includes:

- **Configuration of the Xilinx Design Environment**

This ensures that your team will have a properly configured development environment from the start of the project, a critical consideration when designing with multi-platform FPGAs.

- **System Floorplanning with PlanAhead Training Class***

This class guarantees that your team is fully trained in floor planning systems and the use of the PlanAhead solution. Each student will receive an official certificate of training upon completion.

- **Design Architecture/Implementation Consultation**

This consultation ensures that your design is optimally architected for the selected technology, shortening the implementation phase of your project and getting you to market faster by reducing design risks.

- **Comprehensive Training Plan**

Xilinx will help you develop a training plan tailored specifically to the needs of your design team. This will help prevent schedule slips later in the project by ensuring that the team is skilled in the needed disciplines. It will also help you maintain a more effective and highly motivated team.

Take the Next Step

To find out more about PlanAhead QuickStart!, contact your Xilinx representative or go to www.xilinx.com/paq

* The goal of this offering is to help kick-start the customer's design and not as a vehicle to train large numbers of engineers. The number of students allowed in the class is limited to ten per valid PlanAhead license.

Corporate Headquarters

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: (408) 559-7778
Fax: (408) 559-7114
Web: www.xilinx.com

European Headquarters

Xilinx
Citywest Business Campus
Saggart,
Co. Dublin
Ireland
Tel: +353-1-464-0311
Fax: +353-1-464-0324
Web: www.xilinx.com

Japan

Xilinx, K.K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific

Xilinx Asia Pacific Pte. Ltd.
No. 3 Changi Business Park Vista, #04-01
Singapore 486051
Tel: (65) 6544-8999
Fax: (65) 6789-8886
RCB no: 20-0312557-M
Web: www.xilinx.com

Distributed By:





The Xilinx Productivity Advantage Program

Everything You Need in One Package

To create successful programmable logic designs for increasingly complex design challenges it takes more than devices and raw engineering expertise. You need the best design tools delivered with the training, intellectual property (IP), and support to make your design team as effective as possible. How do you rest assured that your entire design team has all the software and services they need, delivered when they need them?

The Xilinx Productivity Advantage (XPA) Program is your key to success. To help you use our programmable logic devices to their maximum potential, we offer everything you need, delivered in one comprehensive package, and designed to accelerate your productivity.



The Fast Track to Productivity

The XPA Program will significantly reduce your risk and increase productivity, while saving you time and money because a single purchase delivers the software, support, services, and IP, in the quantities you need, at the best value. If you want the assurance of flawless designs, with access to the best engineers and tools in the industry, choose the Xilinx Productivity Advantage Program – and rest easy.

With the XPA Program, you will:

- **Reduce your design risk.** You don't need to face your design challenges alone. Our expert engineers will help you create your design correctly the first time, using a consistent software flow. You can choose premium support services for prioritized assistance or design methodology coaching. Plus, our standard hotline and Web-based support is always available.
- **Increase your productivity and profitability.** You will get more done in less time because the software and services are available to your entire design team. You get to market sooner with a better product, and that means higher profitability as well.
- **Reduce your overall costs.** The XPA Program reduces engineering and administrative costs by improving your efficiency. The packaged solution ensures your engineering team will have everything they need to get started designing immediately. Because the XPA Program can be set up with a single purchase order, it saves time and cuts down on the paperwork required to equip each designer.

The Xilinx Solution

Here's What's Included in the XPA Program:

Support Services – The Fastest Time To Knowledge

- Our premium hotline, Platinum Technical Service, gives you access to senior application engineers who provide expedited call handling and priority case resolution through a dedicated, toll-free number.*
- Our Titanium Technical Service puts a dedicated Xilinx design expert on your team. Either onsite or remote, the Titanium Applications Engineer can help advise your design team on the best strategies to achieve timing closure and design fit in the shortest amount of time.

www.support.xilinx.com/support/gsd



Xilinx Global Services *Finish Faster*

Support

Services

Education Services – Save Time, Get It Right

- Education Services – XPA Program training credits make it easy to attend your choice of public, private or e-learning courses.
- Classes for every level of design experience: tool flow (fundamental), enhance your knowledge (intermediate), or high-speed/high-density (advanced).
- Specialized design tracks which include FPGA Design, DSP, PCI, High Speed Design, Embedded Systems Design, and HDL.
- Lectures and labs delivered by professional instructors with extensive design experience.

www.support.xilinx.com/education

Instructor Led
Public Classes

Private
Classes

Live
On-Line

Software

IP

ISE Alliance™ or ISE Foundation™ Software – Fast And Efficient

- Easy to use, high speed and high-density design features include incremental compile and modular design capabilities.

ISE Alliance	ISE Foundation
Timing driven implementation tools with PACE and XPower	Complete design environment that includes everything in ISE Alliance
EDA libraries and integration capabilities	XST HDL synthesis – Verilog or VHDL
CORE Generator™	Schematic and State Diagram Editors (PC)
Constraints Editor	HDL Bencher™ (PC)
Support for all leading Xilinx devices	Support for all leading Xilinx devices

Optional productivity tools include:

- System Generator for DSP is an automatic tool for DSP creation.
- ChipScope™ Pro performs integrated logic and bus analysis of internal FPGA signals.
- ModelSim Xilinx Edition II is a complete PC HDL simulation environment for Verilog or VHDL.
- Embedded software tools allow you to create, edit, compile, link, load, and debug high-level language code for execution on an embedded processor.

www.xilinx.com/ise



Intellectual Property – Add Engineering Value With Advanced LogiCORE™ Modules

- **DSP cores** – developed to implement system-level algorithms, so you can create the highest performance programmable DSP solution available.
- **Processor, controller, and peripheral cores** – allow you to get the maximum benefit from the Xilinx embedded processors.
- **System I/O cores** – provide connectivity solutions for high-performance programmable data path, control plane, and backplane applications.
- **Proven Interoperability.**

Active XPA Program members receive special pricing on Xilinx LogiCORE products.

www.xilinx.com/ipcenter



Two Choices in the XPA Program:

• The Custom XPA Program

Our specialists consult with you to tailor a custom XPA Program to your needs. The proper choices and quantities of software, support, services, and IP are determined by your requirements.

• The XPA Seat – A Pre-Packaged Solution

If you need a single-license solution, the XPA Seat comes prepackaged as the “XPA Seat” so it's easy to order from your Xilinx distributor. The XPA Seat solution includes:

- One license for ISE Alliance™ software
(part # DS-ISE-ALI-XPA) or ISE Foundation™ software
(part # DS-ISE-FND-XPA).
- Ten training credits.
- One seat of Platinum Technical Service.

Take the Next Step

You'll get to market faster with less cost by improving your productivity – and there is no easier way to become more productive than with the XPA Program. To learn more, call 1-800-888-FPGA (3742), or e-mail us at: fpga@xilinx.com.

www.support.xilinx.com/xpa

Corporate
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: 408-559-7778
Fax: 408-559-7114
Web: www.xilinx.com

FORTUNE 2003
100 BEST COMPANIES TO WORK FOR

Europe
Xilinx, Ltd.
Benchmark House
203 Brooklands Road
Weybridge
Surrey KT13 ORH
United Kingdom
Tel: 44-1-870-7350-600
Fax: 44-1-870-7350-601
Web: www.xilinx.com

Japan
Xilinx, K. K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific
Xilinx, Asia Pacific
Unit 1201, 12/F, Tower 6
Gateway
9 Canton Road
Tsimshatsui, Kowloon
Hong Kong
Tel: 852-2-424-5200
Fax: 852-2-494-7159
E-mail: ask-asiapac@xilinx.com

 **XILINX®**
The Programmable Logic Company™