FPGA Programming

Class 2: Design Flow

September 12, 2017
Louis W. Giokas

DesignNews CEC CONTINUING EDUCATION CENTER

Presented by:
Digi-Key ELECTRONICS ROHDE&SCHWARZ

This Week's Agenda

Monday	FPGA Device Description
Tuesday	Design Flow
Wednesday	HDL
Thursday	Synthesis and Layout
Friday	Programming the Chip

Course Description

We start with an introduction to the class of devices called Field Programmable Gate Arrays (FPGAs). The layout and design of several types and critical parameters will be described and discussed. It is important to understand the way the device is constructed to develop effective algorithms.

The device we will be using this week will be the Microsemi IGLOO2. We will also discuss other devices and their structure.

We will introduce two common Hardware Description Languages (HDL), but give examples in one (Verilog).

Today’s Agenda

- Overview
- Algorithm Development
- Design Creation
- Synthesis
- Refinement
- Conclusion/Next Class

Overview

My IGLOO2 Board

Overview

- FPGA programming proceeds through a number of paths that are somewhat different from regular computer programming
- I will attempt to give a feel for the overall process of taking a concept to a FPGA implementation
- Vendor tools help make the process easier and, in general, to integrate the tools into a process flow that is somewhat automated

Overview

- Design is best done at the block, or module level, with well defined interfaces
  - Algorithms are then implemented in HDL (sometimes user written, sometime outside IP)
- This high level approach gives the development tools system significant information that can be used in layout and synthesis

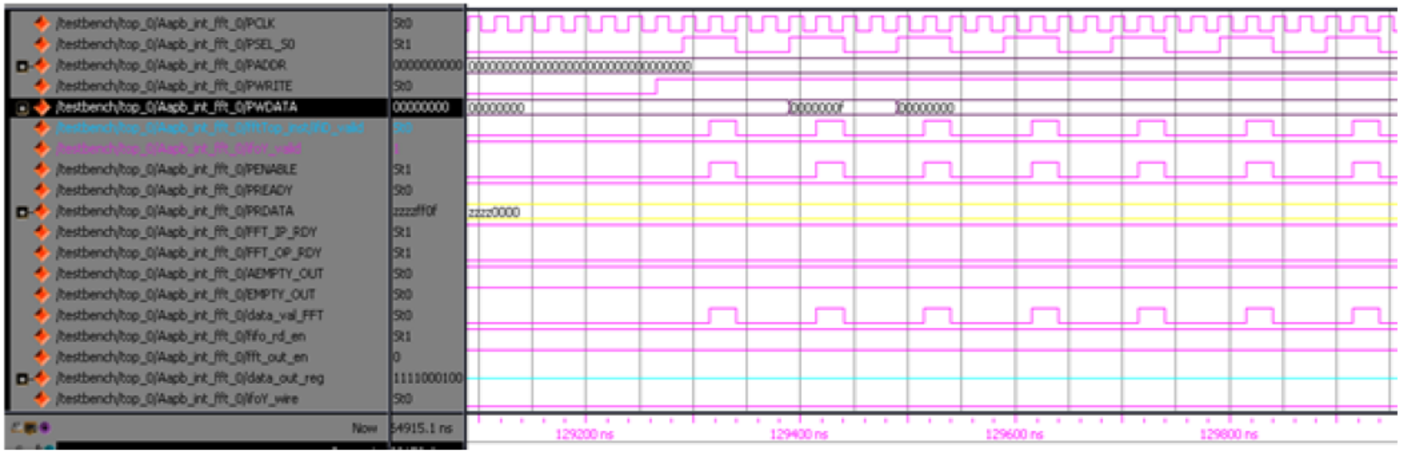
Algorithm Development

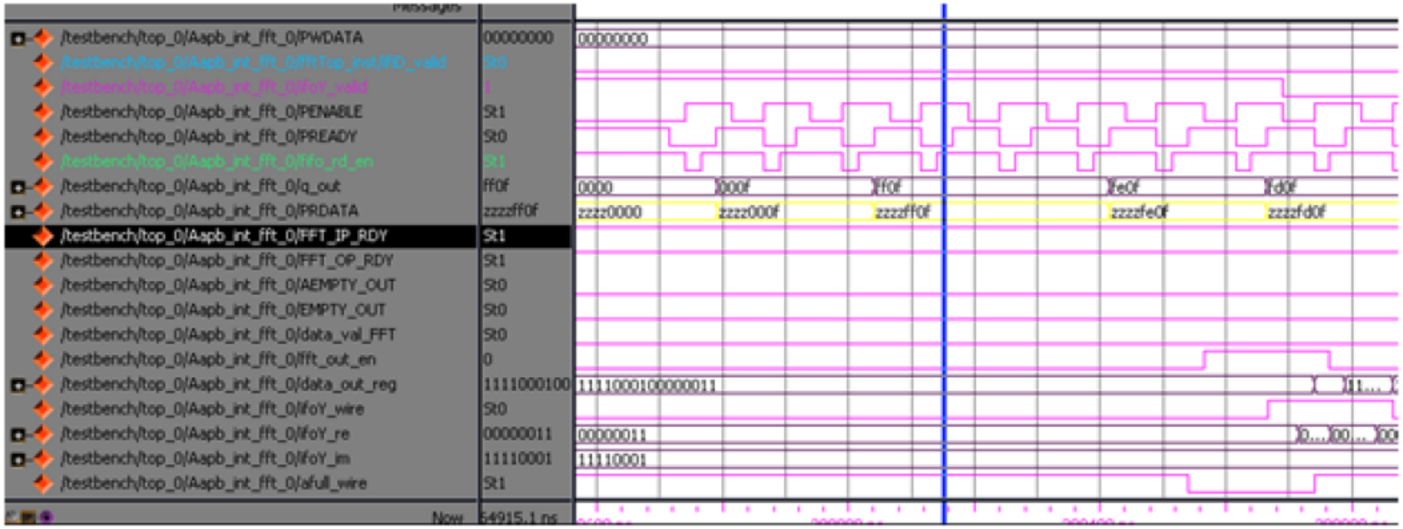
- We start with the “lowest” level of the “program” design task
  - The algorithms are what distinguish one design from another
  - High performance designs can often become valuable IP
- The algorithms we are talking about include things like FFTs and Matrix multiplication, etc.

Algorithm Development

- Writing the code (HDL) for a function such as a FFT can be arduous
  - Need to define both the mathematical functions as well as all the control lines and data lines required
  - Understanding the mathematical algorithm is important, and using the most efficient for your application can yield great efficiencies in your FPGA application

Algorithm Development


/testbench/top_0/Aapb_int_fft_0/PCLK St0
/testbench/top_0/Aapb_int_fft_0/PSEL_S0 St1
/testbench/top_0/Aapb_int_fft_0/PADDR 0000000000
/testbench/top_0/Aapb_int_fft_0/PWRITE St0
/testbench/top_0/Aapb_int_fft_0/PWDATA 00000000
/testbench/top_0/Aapb_int_fft_0/fftTop_inst/iFD_valid St0
/testbench/top_0/Aapb_int_fft_0/ifoY_valid 1
/testbench/top_0/Aapb_int_fft_0/PENABLE St1
/testbench/top_0/Aapb_int_fft_0/PREADY St0
/testbench/top_0/Aapb_int_fft_0/PRDATA zzzzff0f
/testbench/top_0/Aapb_int_fft_0/FFT_IP_RDY St1
/testbench/top_0/Aapb_int_fft_0/FFT_OP_RDY St1
/testbench/top_0/Aapb_int_fft_0/AEMPTY_OUT St0
/testbench/top_0/Aapb_int_fft_0/EMPTY_OUT St0
/testbench/top_0/Aapb_int_fft_0/data_val_FFT St0
/testbench/top_0/Aapb_int_fft_0/fifo_rd_en St1
/testbench/top_0/Aapb_int_fft_0/fft_out_en 0
/testbench/top_0/Aapb_int_fft_0/data_out_reg 1111000100
/testbench/top_0/Aapb_int_fft_0/ifoY_wire St0
Now 54915.1 ns
00000000000000000000000000000000
00000000
0000000f
00000000
zzzz0000
129200 ns
129400 ns
129600 ns
129800 ns


Messages
/testbench/top_0/Aapb_int_fft_0/PWDATA 00000000
/testbench/top_0/Aapb_int_fft_0/fftTop_inst/iFD_valid St0
/testbench/top_0/Aapb_int_fft_0/ifoY_valid 1
/testbench/top_0/Aapb_int_fft_0/PENABLE St1
/testbench/top_0/Aapb_int_fft_0/PREADY St0
/testbench/top_0/Aapb_int_fft_0/fifo_rd_en St1
/testbench/top_0/Aapb_int_fft_0/q_out ff0f
/testbench/top_0/Aapb_int_fft_0/PRDATA zzzzff0f
/testbench/top_0/Aapb_int_fft_0/FFT_IP_RDY St1
/testbench/top_0/Aapb_int_fft_0/FFT_OP_RDY St1
/testbench/top_0/Aapb_int_fft_0/AEMPTY_OUT St0
/testbench/top_0/Aapb_int_fft_0/EMPTY_OUT St0
/testbench/top_0/Aapb_int_fft_0/data_val_FFT St0
/testbench/top_0/Aapb_int_fft_0/fft_out_en 0
/testbench/top_0/Aapb_int_fft_0/data_out_reg 1111000100
/testbench/top_0/Aapb_int_fft_0/ifoY_wire St0
/testbench/top_0/Aapb_int_fft_0/ifoY_re 00000011
/testbench/top_0/Aapb_int_fft_0/ifoY_im 11110001
/testbench/top_0/Aapb_int_fft_0/afull_wire St1
Now 54915.1 ns
00000000
0000
000f
ff0f
fe0f
fd0f
zzzz0000
zzzz000f
zzzzff0f
zzzzfe0f
zzzzfd0f
1111000100000011
11...
00000011
0...
00...
11110001

Presented by:

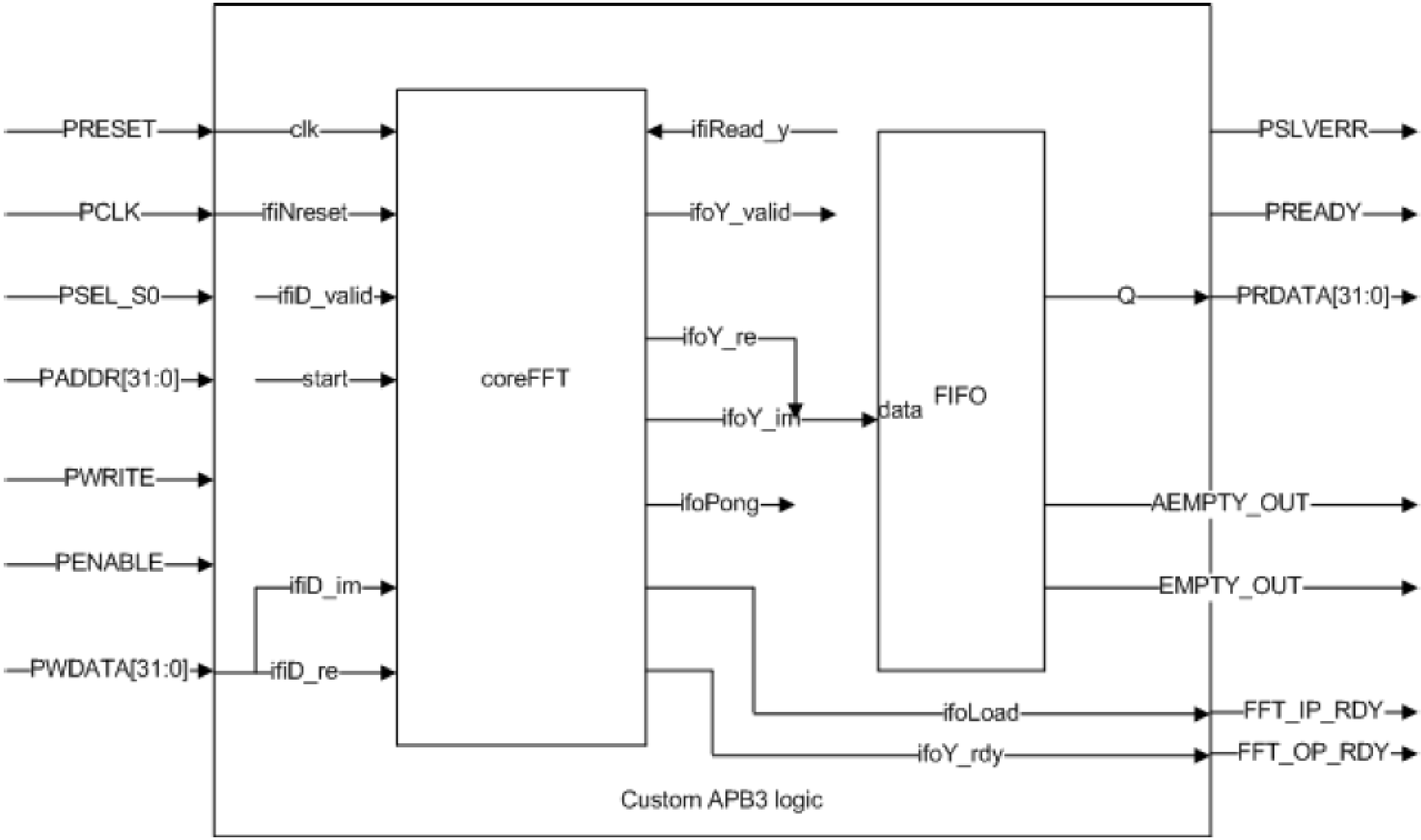
Design Creation

- By design creation I am referring to the overall structure of the FPGA program
  - We have the algorithms we want to use
  - We know the sources and syncs of the data
  - We know the size and time limitations we are working under
- The next step is to connect all of these in an optimal manner

Design Creation

- Development environments provide visual tools to lay out the design and to specify the signals and data paths
- Simulation tools are used to test the configuration
  - Correctness, Timing, Signals
- This is where a test-bench is created
  - Allows correctness to be assessed

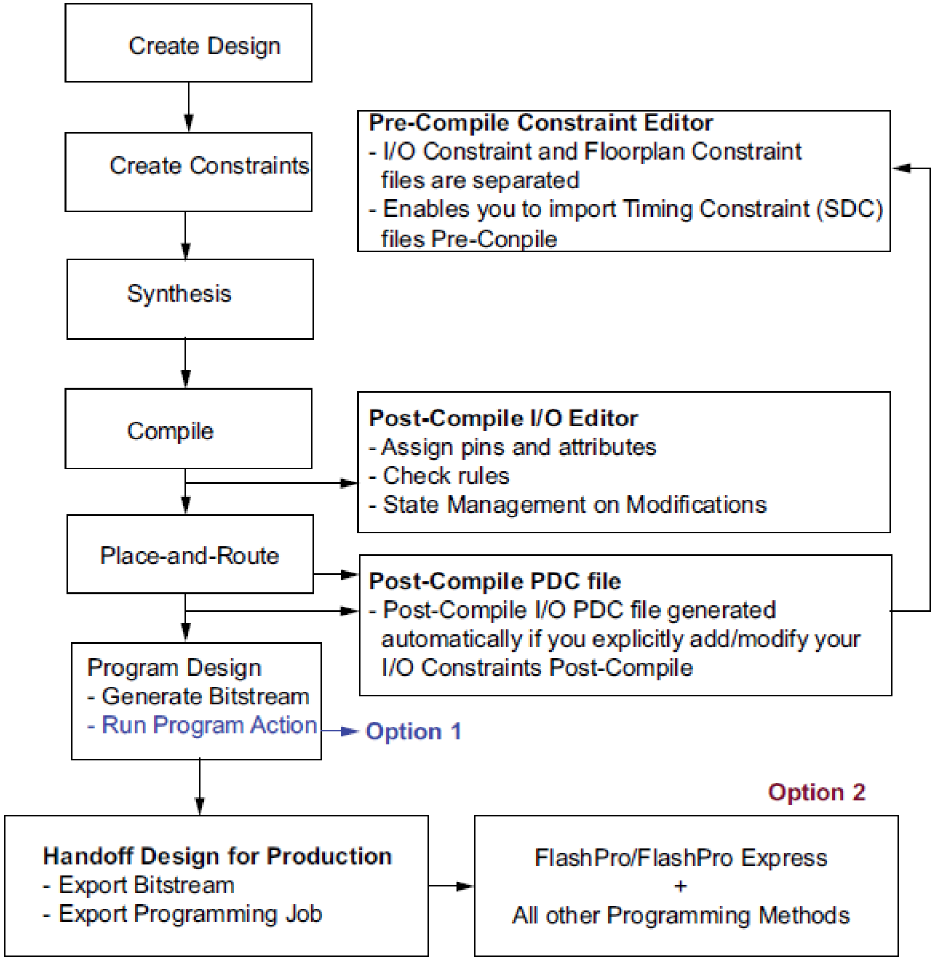
Design Creation



Synthesis

- Synthesis is the process used to lay out the design on the device
- A key feature of synthesis is the application of constraints
- There are typically multiple ways to do this as defined by the vendor

Synthesis



Create Design

Create Constraints

Synthesis

Compile

Place-and-Route

Program Design
- Generate Bitstream
- Run Program Action

Pre-Compile Constraint Editor
- I/O Constraint and Floorplan Constraint files are separated
- Enables you to import Timing Constraint (SDC) files Pre-Conpile

Post-Compile I/O Editor
- Assign pins and attributes
- Check rules
- State Management on Modifications

Post-Compile PDC file
- Post-Compile I/O PDC file generated automatically if you explicitly add/modify your I/O Constraints Post-Compile

Option 1

Option 2

Handoff Design for Production
- Export Bitstream
- Export Programming Job

FlashPro/FlashPro Express
+
All other Programming Methods

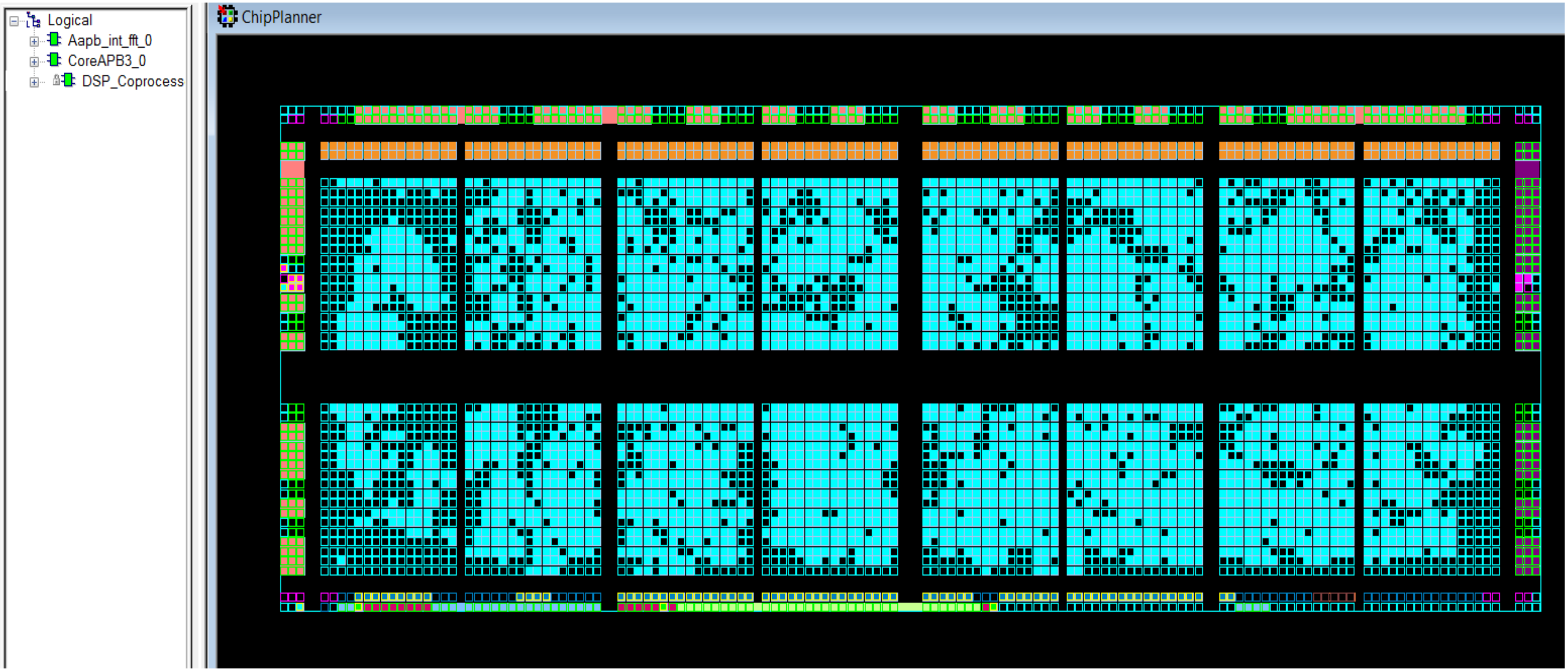
Synthesis

- While we have tested and simulated the design in the tools, we also need to verify the design in the synthesis process
  - Layout may have some impact on operation and signals
  - When multiple functional blocks are placed on the same FPGA, we need to verify the design as a whole

Refinement

- There are a number of issues to be addresses after the design has been synthesized
  - These may lead us to modify the design (at any level) to achieve our goals
- Issues may include timing, area and power
- Tools are generally provided to be able to observe these items and determine where we can make changes

Refinement



Presented by:

Refinement

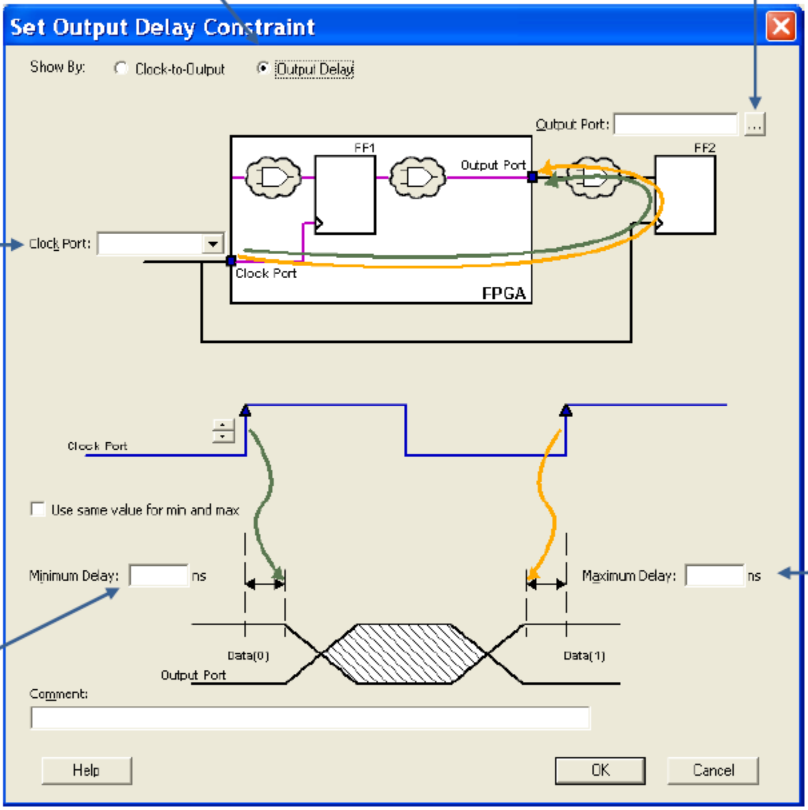
Select Output Delay

Select Output Port

Select Clock Port

Maximum Delay

Minimum Delay



Set Output Delay Constraint

Show By: ○ Clock-to-Output ● Output Delay

Output Port:

Clock Port:

□ Use same value for min and max

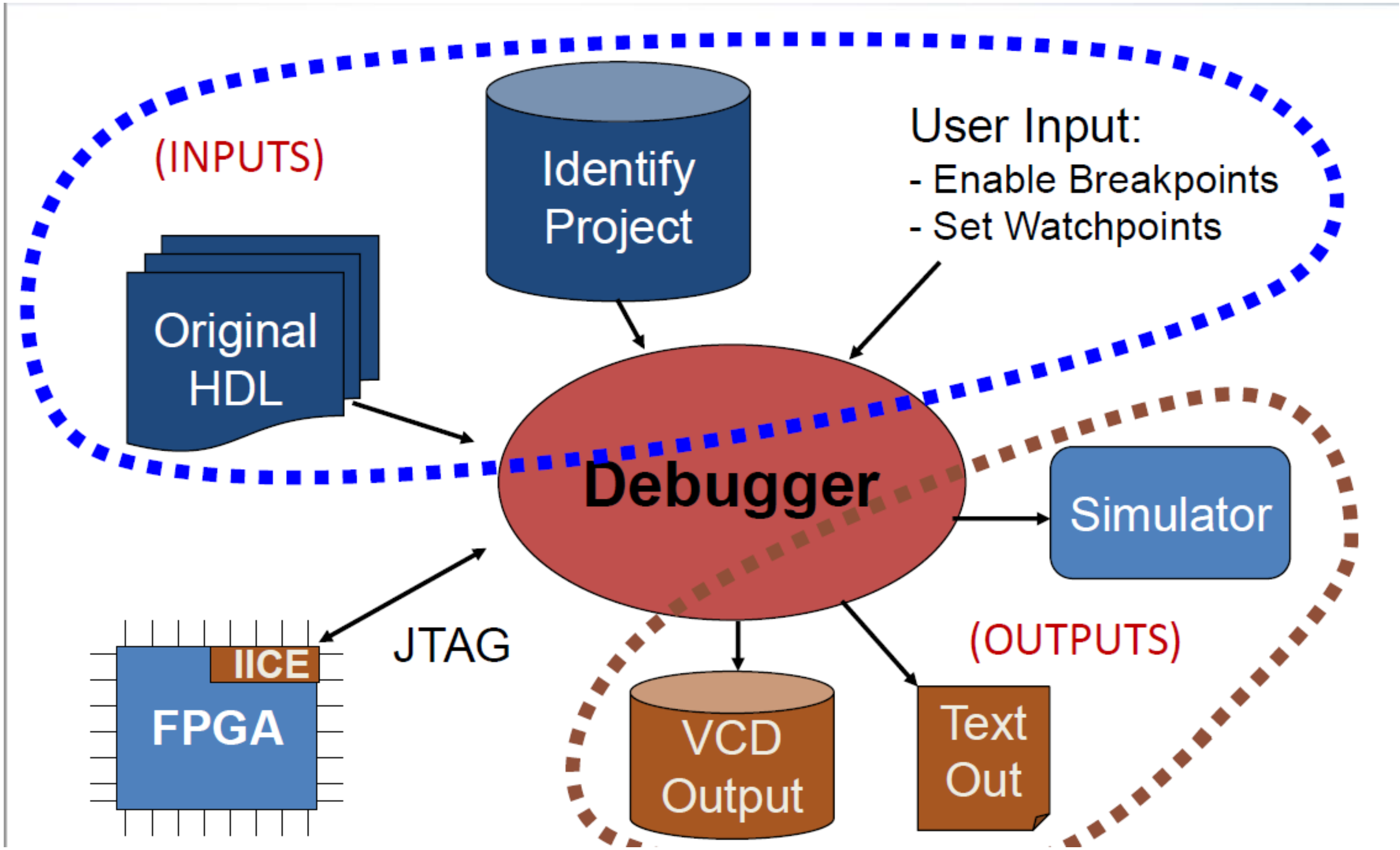
Minimum Delay: ns

Maximum Delay: ns

Comment:

Presented by:

Refinement


(INPUTS)
Identify Project
User Input:
- Enable Breakpoints
- Set Watchpoints
Original HDL
Debugger
Simulator
IICE
FPGA
JTAG
(OUTPUTS)
VCD Output
Text Out

Conclusion/Next Class

- We have discussed the design flow for FPGA development at a high level
- We have looked at some of the major issues and tools that are used
- Tomorrow we will discuss HDL (Verilog) in more detail