COMBINATIONAL DATAPATHS

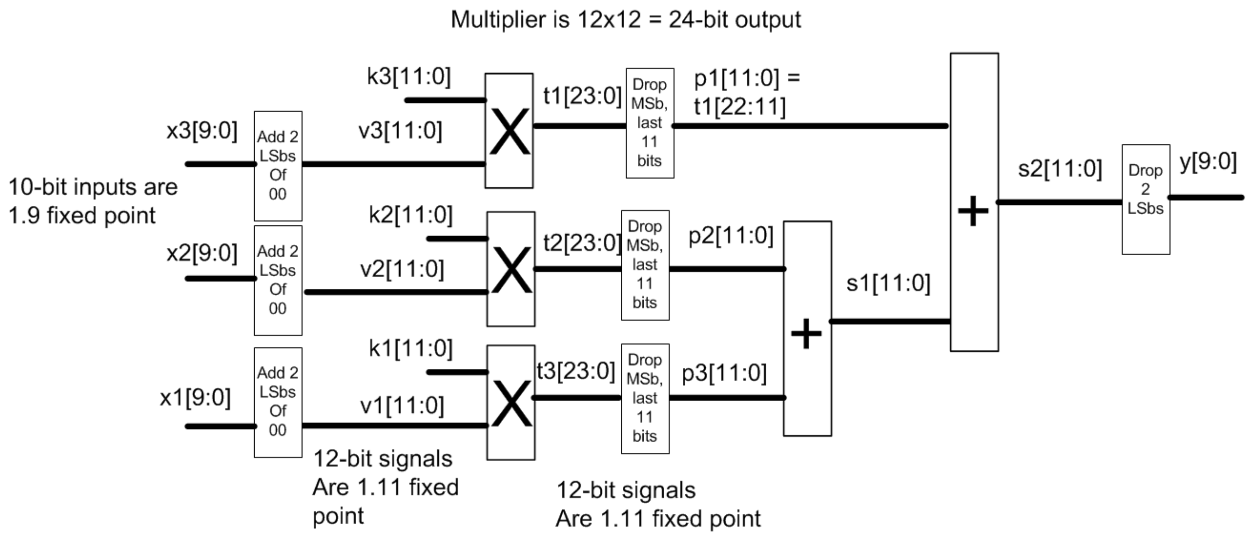
This lab has you implement a combinational datapath containing adders and multipliers. You will run both behavioral and post place & route simulations, run the timing analyzer, and view the placed/routed design in the FPGA viewer.

BACKGROUND

Review the notes on multiplication and fixed-point representation.

TASK DESCRIPTION

You are to create a Verilog module named *lab3dpath* that implements the following schematic:



Busses k1, k2, k3 are constants, and their values are specified in the *lab3_computations.xlxs* file that is in the lab3 zip archive.

You will implement two different versions of this datapath, one will use LUT-based multipliers and the other will use hardmacro multipliers.

Procedure

1. Download the zip archive associated with the lab.
2. This contains the following files:
    - *lab3dpath.v* -- complete this module
    - *tb_lab3dpath.v* – test bench
    - *multadd_vectors.txt* – input vector file for testbench.

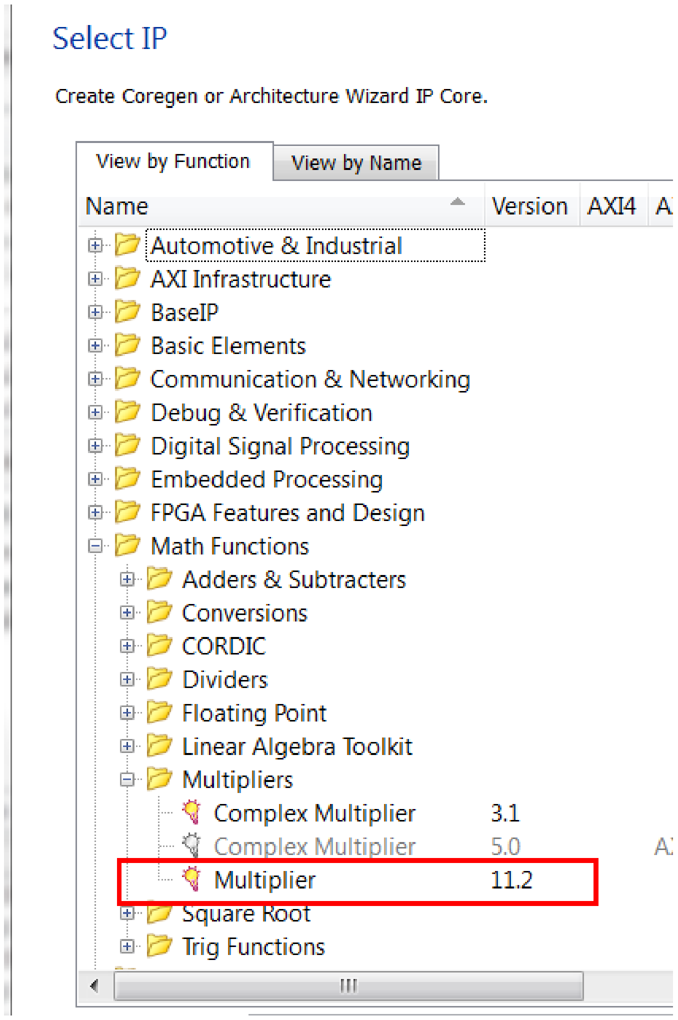
- *lab3_computations.xlxs* – spreadsheet file that contains the test vectors in decimal form, shows the intermediate calculations in hex and decimal. You can use this to help debug your design.
- *report.doc* – report file that needs to be filled out. You should open this file, and fill out the requested information as your perform the remaining steps.

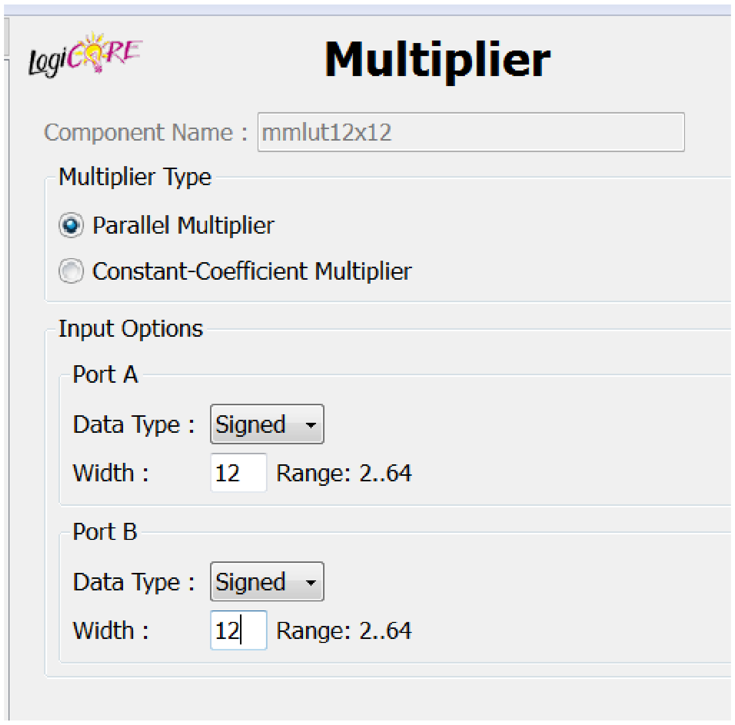
3. Create a project named *lab3_part1* and add the *lab3dpath.v* file to it as the top module. Then, generate a test fixture for this file named *tb_lab3dpath.v*, and copy the contents of the original *tb_lab3dpath.v* into it. When you create the project, you do not have to add the UCF file to it as was done for Lab #1 as we are not going to download this file into the Basys board.
4. Copy the *multadd_vectors.txt* file into the project directory. This is needed before you simulate your design.
5. Implement your Verilog code in the *lab3dpath.v* file and test it with the *tb_lab3dpath.v* using a behavioral simulation. You are finished when all of the vectors pass as indicated by messages printed to the simulation console pane. **This implementation will use LUT-based multipliers**. See the section titled *Multiplier Implementation* to see how to generate the multiplier using CoreGen. NOTE: the Coregen tool only has to be run ONE TIME to generate the multiplier implementation. You will instantiate this module three times in your design since it uses three multipliers (you can call the instances U1, U2, U3 or whatever you want, like 'Sally', 'Monica', and 'Betsy' or 'Sam', 'Charles' and 'Xavier', or "WTH", "IDK", "LOL",… you get the picture). The Verilog module file (*somename*.v) is generated in the 'ipcore_dir' folder – open this with a text editor so that you can discover the pin names that CoreGen used for this module. You need this information when you write the statement that instantiates this in your top-level module.
6. Once the simulation is working, use the 'Implement Top Module' command to map this design to the FPGA. You do not have to download it into your board.
7. Record the number of 4-input LUTs and the %utilization for this design (see the 'Design Summary' page).
8. Run the 'Timing Analyzer' tool, and record the maximum pad-to-pad delay along with the starting pin and ending pin. The 'pad-to-pad' delay is the delay from input pin to an output pin. Since this is a purely combinational design, all input pins have paths to one or more output pins. The 'Timing Analyzer' tool will list the pad-to-pad paths in terms of longest path to shortest path. See the section titled 'Timing Analysis' for details on running the Timing Analyzer.
9. Run a 'Post-Place & Route' simulation. This simulation includes delays from the placed/routed design. See the section titled 'Post-Route Simulation' for details on how to do this. Change the radix of all signals in the waveform window to 'hexadecimal' via the right-click menu. In the waveform window, zoom in on any applied vector (you choose). Place a marker at beginning of a new vector (inputs have changed). Then use the vertical cursor to measure the delay to when the 'y' output stabilizes with its correct value. Take a screenshot of this and include this in your report.
10. Create new project and call it *lab3_part2*. The only difference between this project and the previous project is that you will use a hardmacro multiplier instead of a LUT-based multiplier (you can copy your solution for *lab3dpath.v* and just modify it to use the hardmacro multiplier module after you generate it). Repeat all of the steps for this project that you did for the first project, except do not run a 'Post-Place & Route' simulation. Instead, start the FPGA Editor, zoom-in, pan around, and find one of the hardmacro multipliers. They will be self-evident, as they will be the only rectangular blocks with 24

inputs and 24 outputs (also, when you click on it, the name will be something like 'MULT18x18_X0Y3" or similar). Take a screenshot of this, and include it in your report.

Multiplier Implementation

The Xilnx Coregen tool will be used to generate the multiplier module. See the Lab 2 writeup about how to start up the Xilinx Coregen tool. Generate a LUT-based signed 12x12=24 bit multiplier. See the screenshots below for the correct options to generate this.





Multiplier

Component Name : mmlut12x12

Multiplier Type

[✓ Parallel Multiplier]

Constant-Coefficient Multiplier

Input Options

Port A

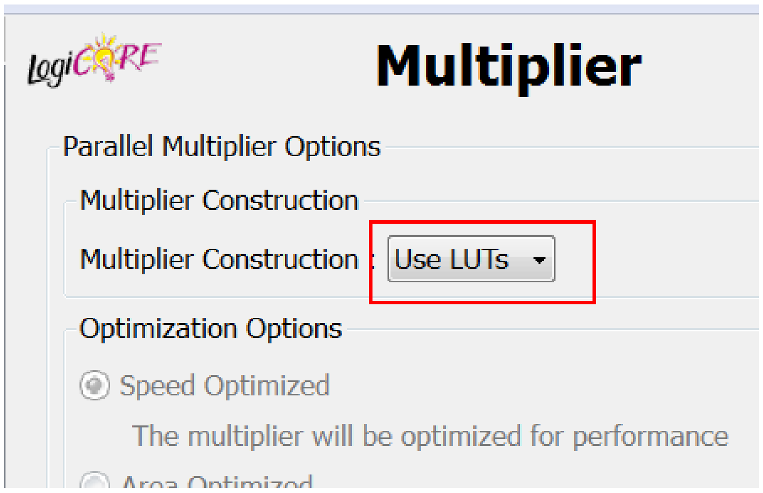
Data Type : Signed

Width : 12 Range: 2..64

Port B

Data Type : Signed

Width : 12 Range: 2..64



Multiplier

Parallel Multiplier Options

Multiplier Construction

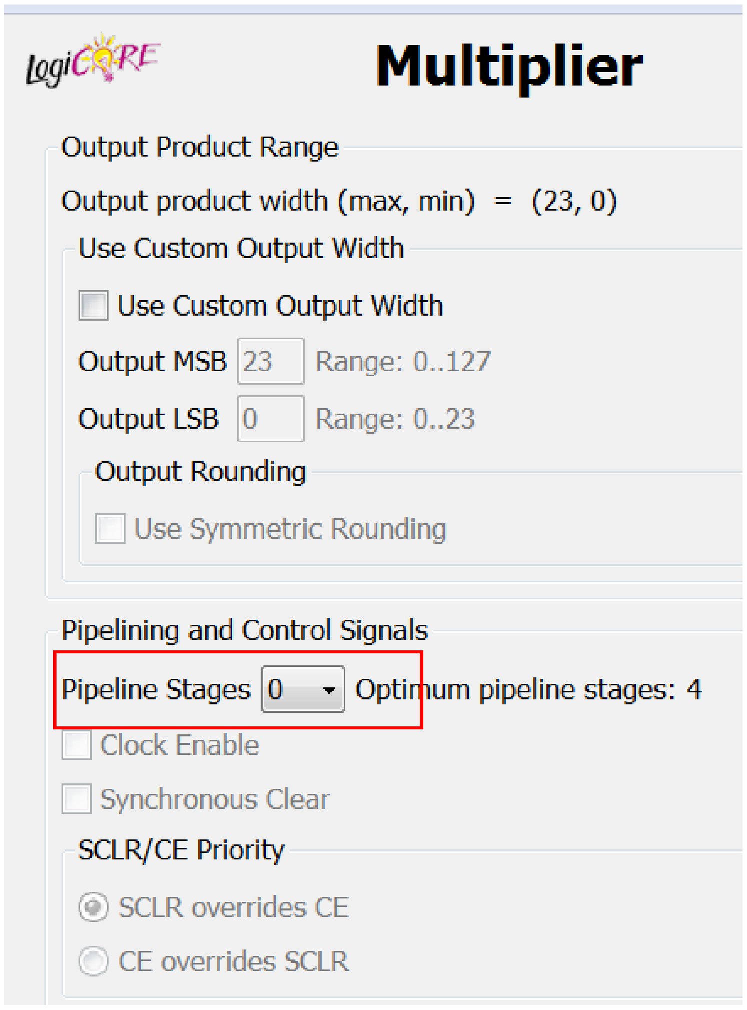
Multiplier Construction : Use LUTs

Optimization Options

Speed Optimized

The multiplier will be optimized for performance

Area Optimized


Multiplier

Output Product Range

Output product width (max, min) = (23, 0)

Use Custom Output Width

- [ ] Use Custom Output Width

Output MSB 23 Range: 0..127

Output LSB 0 Range: 0..23

Output Rounding

- [ ] Use Symmetric Rounding

Pipelining and Control Signals

Pipeline Stages 0 Optimum pipeline stages: 4

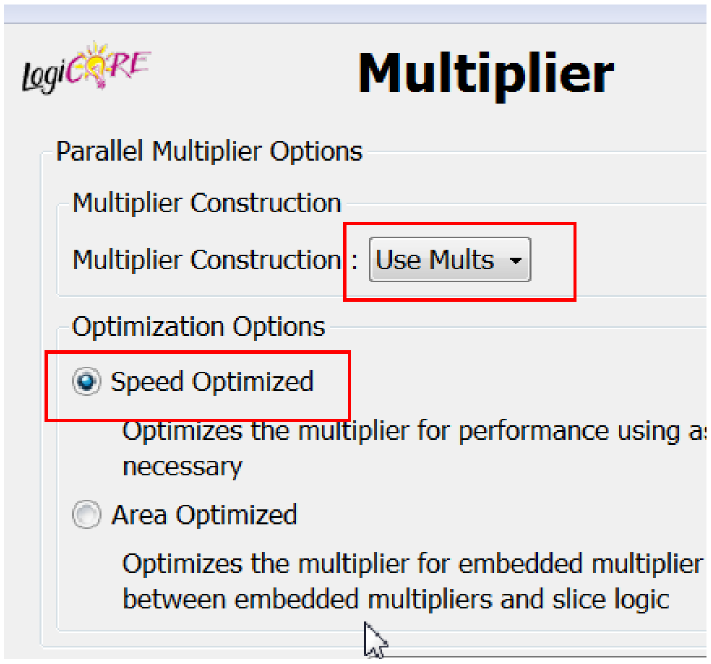
- [ ] Clock Enable
- [ ] Synchronous Clear

SCLR/CE Priority

● SCLR overrides CE

○ CE overrides SCLR

To generate a multiplier using a hardmacro multiplier block, use the following option in the second screen:


LogiCORE Multiplier

Parallel Multiplier Options

Multiplier Construction

Multiplier Construction : Use Mults

Optimization Options

[✓ Speed Optimized]

Optimizes the multiplier for performance using a
necessary

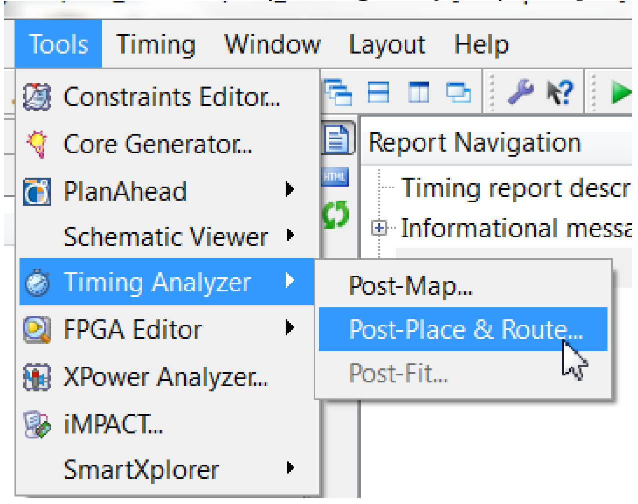
Area Optimized

Optimizes the multiplier for embedded multiplier
between embedded multipliers and slice logic

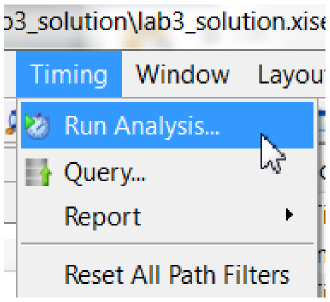
Timing Analysis

The timing analysis is run after you have successfully implemented the design via 'Process | Implement Top Module'.

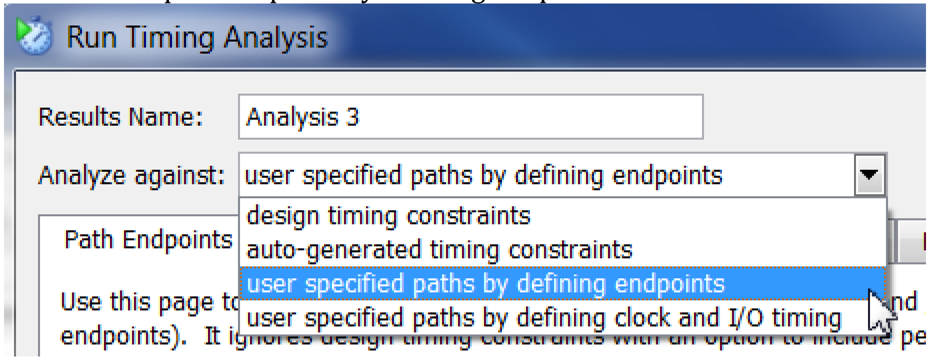
To run the timing analysis tool, click on 'Implementation' tab in the View Window. From the top menu, execute Tools | Timing Analyzer | Post-place & Route.



After this is finished, the display window will contain the some data from the timing tool, but all this did was find all of the paths in the design. We will still need to perform some analysis on these paths. From the top menu, execute Timing | Run Analysis.



This pops up the 'Run Timing Analysis' window. Use the 'Analyze against' pulldown menu, and select 'user specified paths by defining endpoints'.



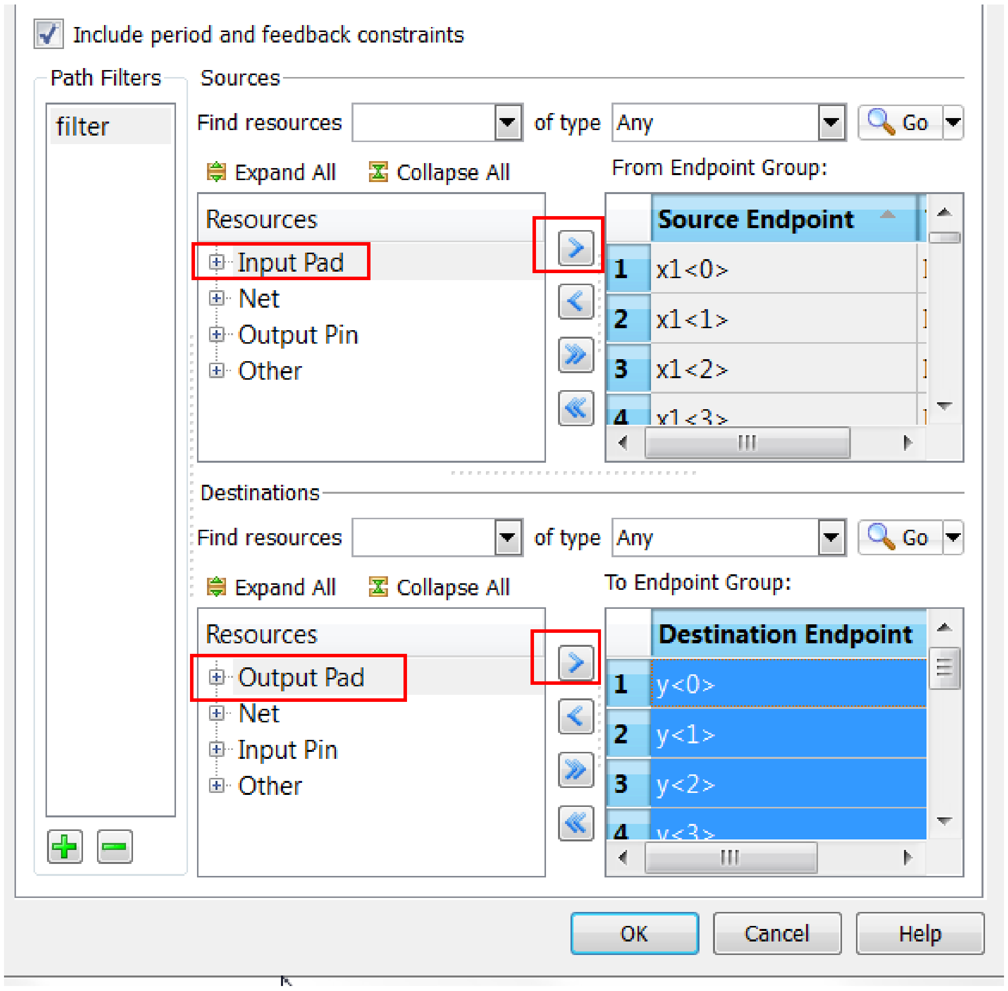
Results Name: Analysis 3

Analyze against: user specified paths by defining endpoints

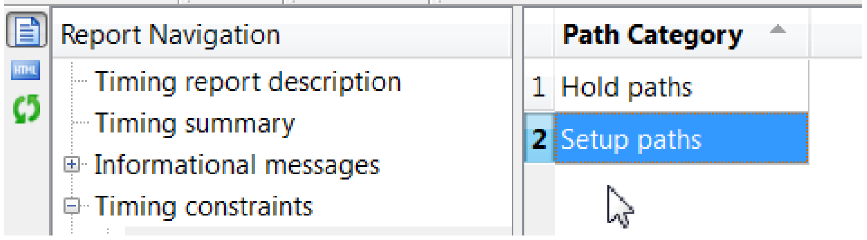
design timing constraints
auto-generated timing constraints
user specified paths by defining endpoints
user specified paths by defining clock and I/O timing

Use this page to
endpoints). It ignores design timing constraints with an option to include pe

This changes the window to allow you to define starting and ending points for paths to be analyzed. Click on 'Input pad', and then click on the '>' symbol to put all input pads as source inputs. Click on 'Output pad', then click on the '>' symbol to put all output pads in the Destination Endpoint window. This causes the tool to analyze all input pad to output pad paths in the design. Click on 'OK' to proceed.



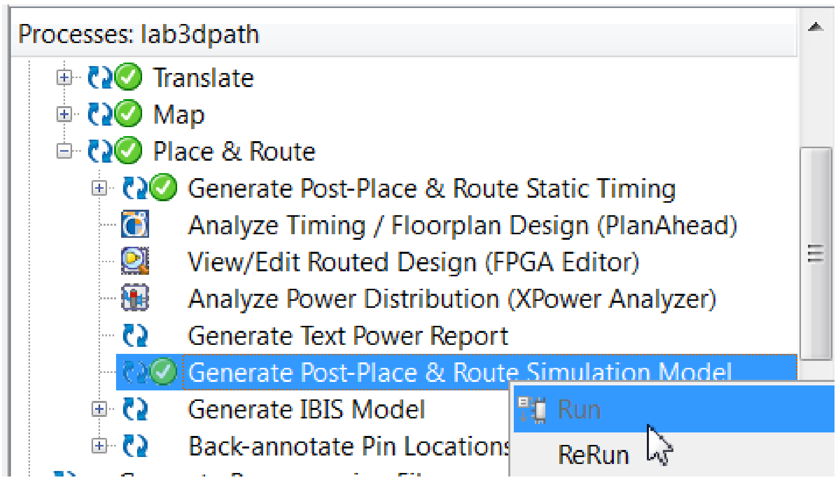
The timing report window changes, and you will now see two choices: *Setup paths* and *Hold Paths*. The *Setup paths* are the longest paths, click on this (the *Hold Paths* are the shortest paths).



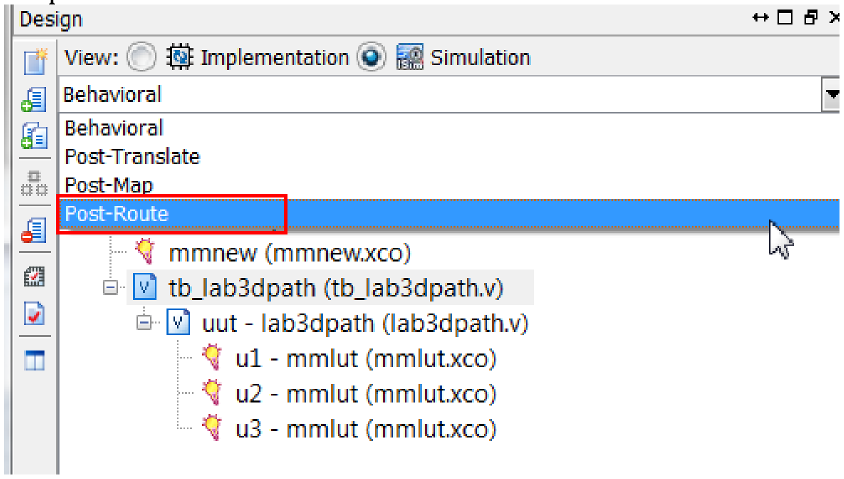
The window will show the delay of the longest path, and trace this path from input to output. The listing of the delay path will alternate between gate delays (these will have various labels) and 'net' delays (routing delays). Routing delays in an FPGA can be significant, and as long or longer than the gate delays.

Post-Place & Route Simulation

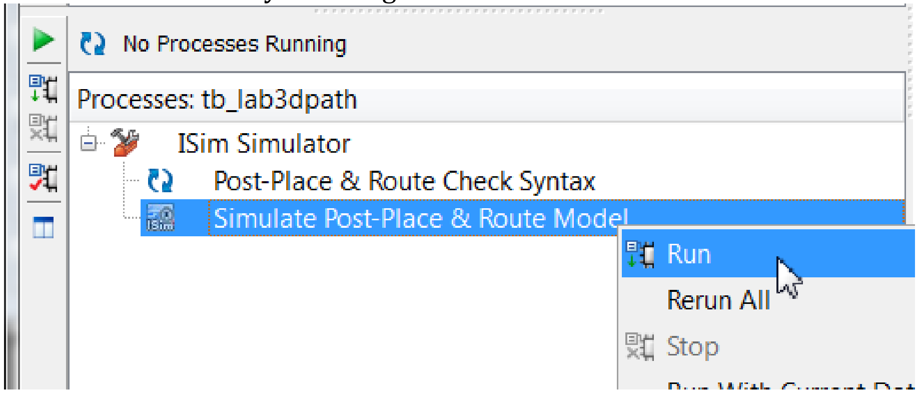
To run a Post-place & Route simulation, click on the 'Implementation' tab in the View window. Then in the Processes Window, expand the 'Place & Route' process by clicking on the '+' symbol, then run the 'Generate Post-Place & Route Simulation Model". You should get a message saying that process completed successfully.


Processes: lab3dpath
Translate
Map
Place & Route
Generate Post-Place & Route Static Timing
Analyze Timing / Floorplan Design (PlanAhead)
View/Edit Routed Design (FPGA Editor)
Analyze Power Distribution (XPower Analyzer)
Generate Text Power Report
Generate Post-Place & Route Simulation Model
Generate IBIS Model
Back-annotate Pin Locations
Run
ReRun

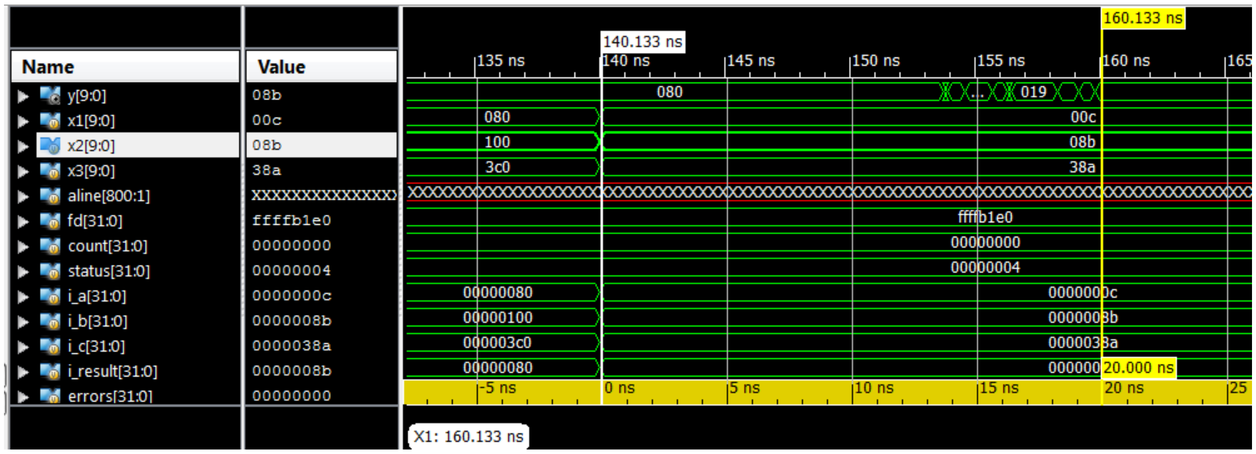
Then, click on the 'Simulation' radio button in the View window, and choose 'Post-Route' from the drop-down menu.



Run the simulation by selecting the 'Simulate Post-Place & Route Model' in the Processes window.

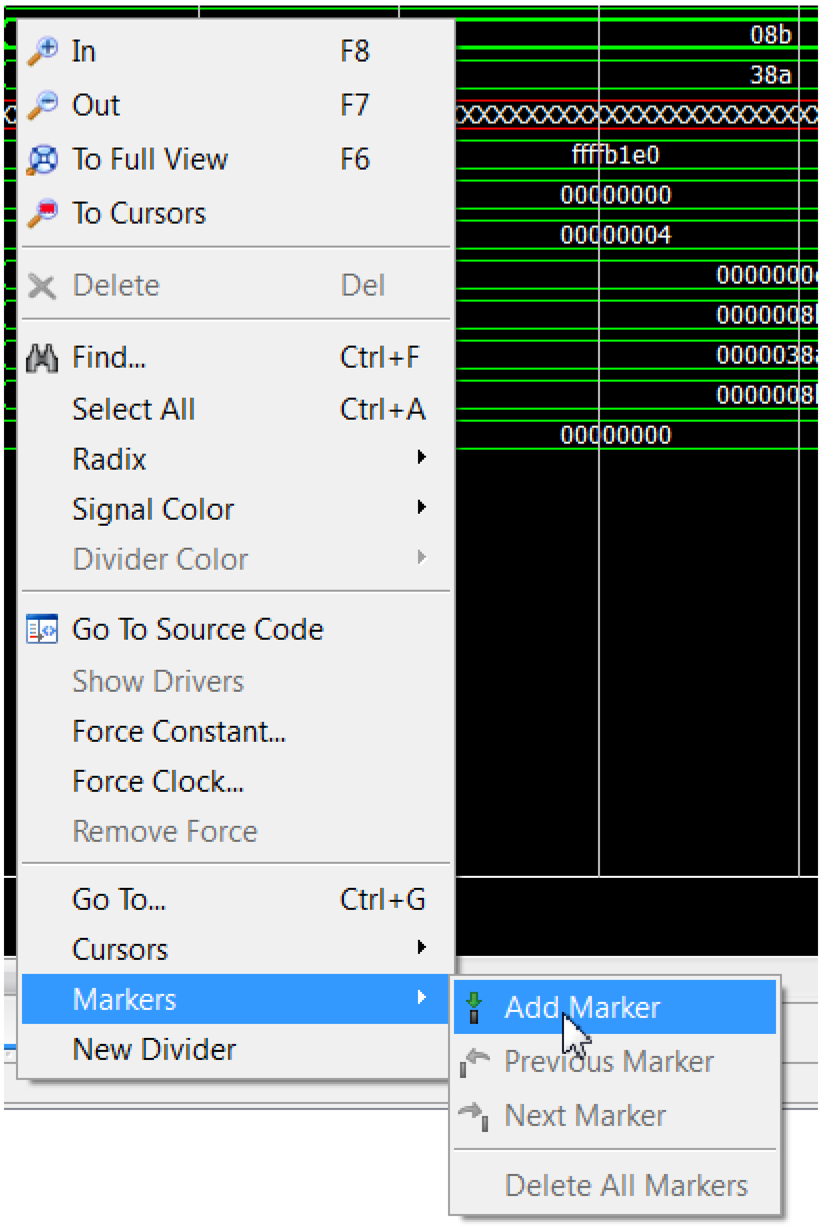


This simulation contains estimated delays for the design. You can tell that this is a simulation with real delays as signals will not change all at the same time as they do in a behavioral design – different paths have different delays, and a bus value will bounce around until all delays have settled out. In the screenshot below, I have zoomed into a vector that has *x1*= 0x080, *x2*=0x100, *x3*=0x38a. Note that it takes about 20 ns for the *y* output to settle out to its final value of 0x08b.



Name	Value
y[9:0]	08b
x1[9:0]	00c
x2[9:0]	08b
x3[9:0]	38a
aline[800:1]	XXXXXXXXXXXXXXX
fd[31:0]	ffffb1e0
count[31:0]	00000000
status[31:0]	00000004
i_a[31:0]	0000000c
i_b[31:0]	0000008b
i_c[31:0]	0000038a
i_result[31:0]	0000008b
errors[31:0]	00000000

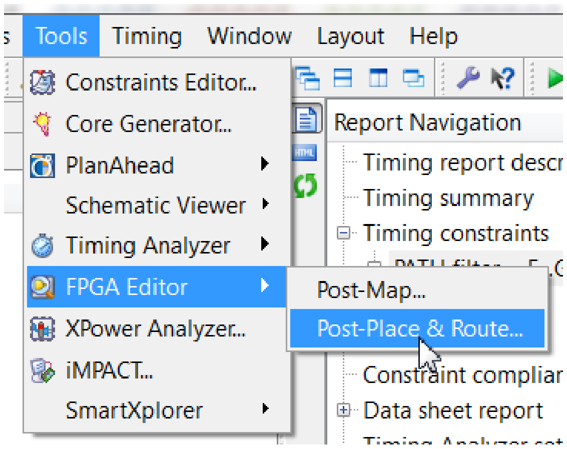
To make a propagation delay measurement like this on the waveforms, add a 'Marker' as shown below from the right-click menu.



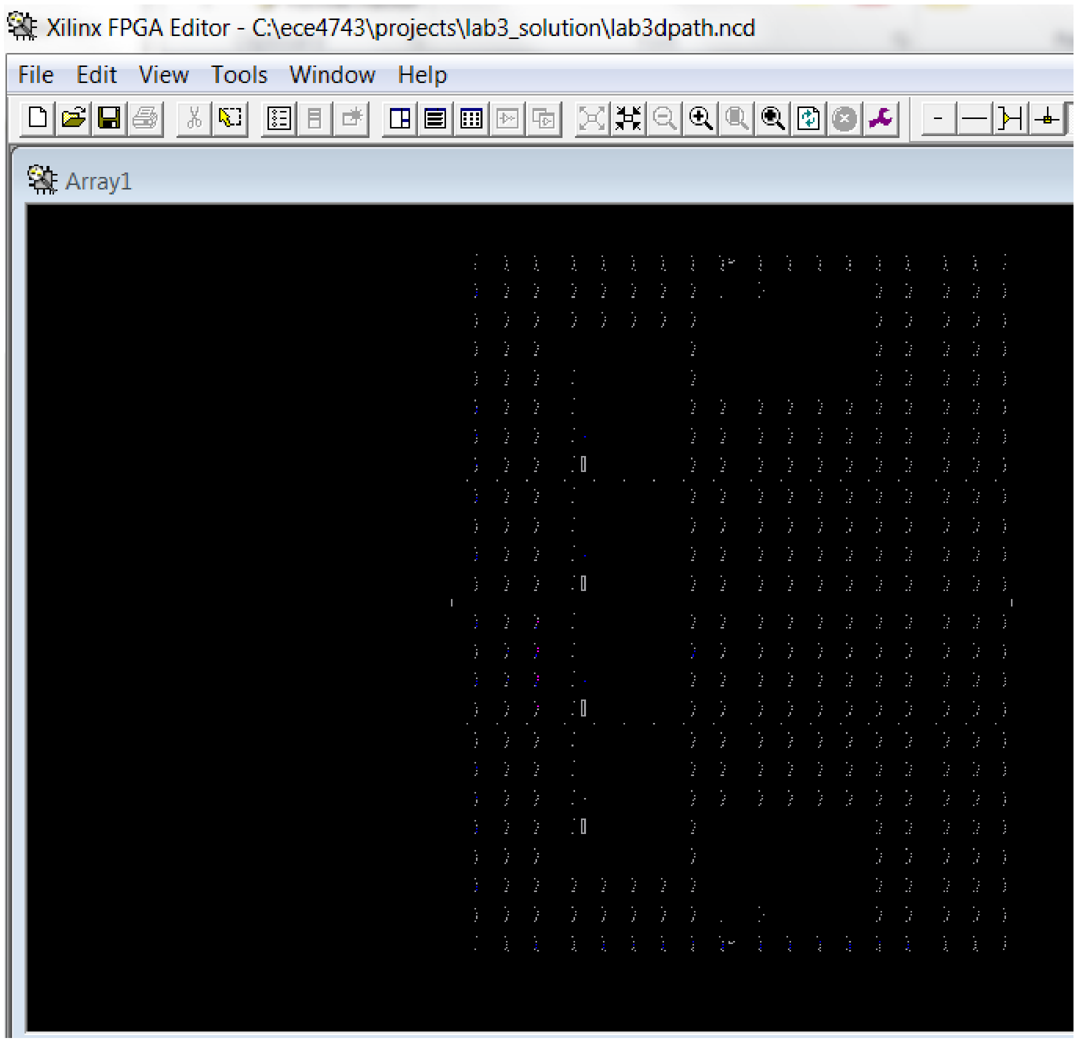
Place the marker where you wish to start the measurement. Then drag the original vertical cursor to the right of this marker – as you drag the vertical cursor, you will notice that the time delta between this cursor and the marker is displayed on the bottom horizontal time scale.

FPGA Editor

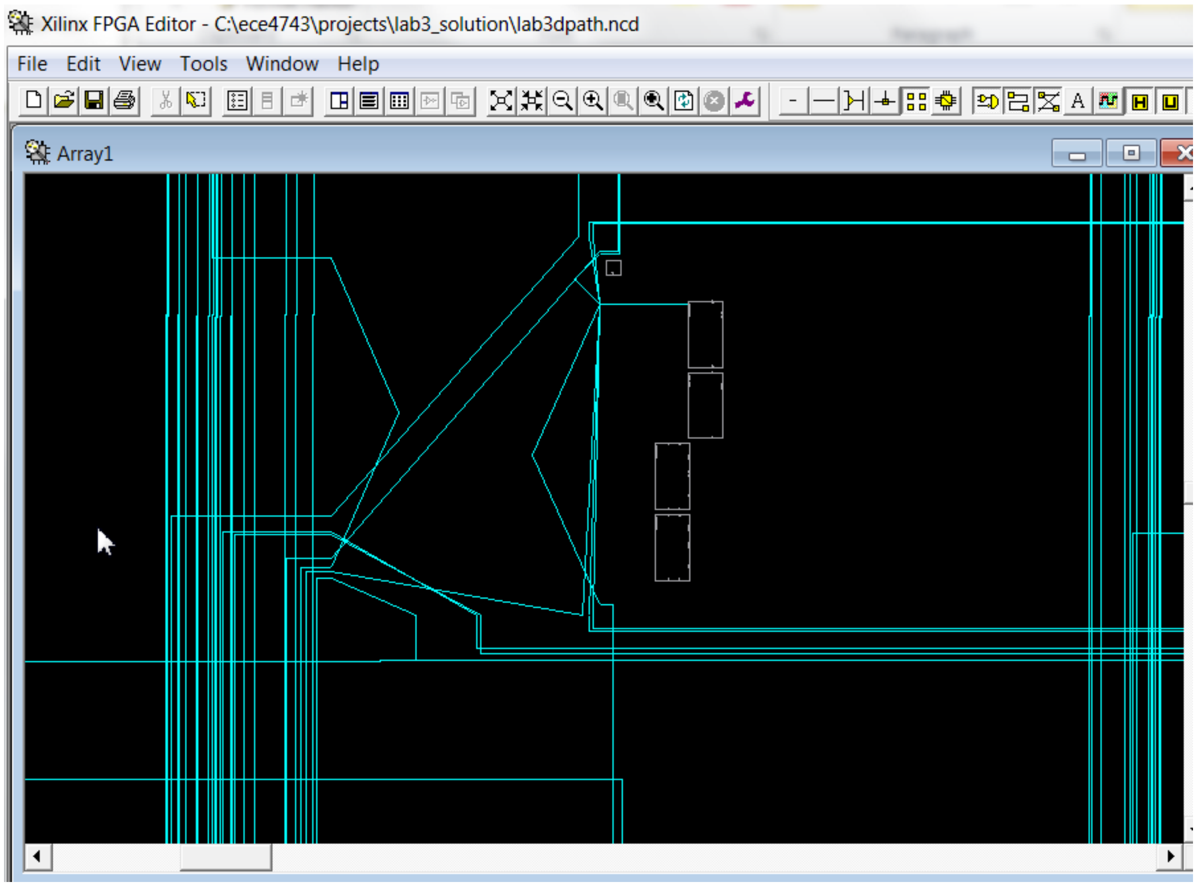
After the design has been implemented (Placed & Routed), use the command 'Tools | FPGA Editor | Post Place & Route' start up the FPGA editor. This tool allows you to see the routing as it appears on the FPGA substrate.



When the tool starts up, you will see a window similar to:


Xilinx FPGA Editor - C:\ece4743\projects\lab3_solution\lab3dpath.ncd
File
Edit
View
Tools
Window
Help
Array1

You can zoom in and see routes going to CLBs and hardmacros:


Xilinx FPGA Editor - C:\ece4743\projects\lab3_solution\lab3dpath.ncd
File Edit View Tools Window Help
Array1

GRADING

The grading point distribution is specified in the report document.

SUBMISSION INSTRUCTIONS

Create a directory named 'lab3_*netid*', i.e., (lab3_rbr5).

Copy the lab3_part1, lab3_part2 directories to this directory.
Copy your completed report.doc to this directory.

From Windows, select the 'lab3_netid' folder, and from the right-click menu, use 'Send To Compressed (zipped) Folder' command to produce a ZIP archive named 'lab2_netid.zip'.

Upload your 'lab3_*netid*.zip' file to Blackboard using the Lab3 assignment link.