Tutorial on
FPGA Design Flow
based on
Aldec Active HDL

Ver 1.3
This tutorial assumes that you have basic knowledge on how to use ActiveHDL and its functional simulation. The example codes used in this tutorial can be obtained from http://ece.gmu.edu/coursewebpages/ECE/ECE448/S09/experiments/448_lab3.htm.

The current version of the tutorial was tested using the following tools:

**CAD Tool**
- ActiveHDL Version : 7.3
  - Synthesis Tool
    - Synplicity Synplify PRO Version : 8.6
    - ISE&Webpack Synthesis&Implementation Version : 9.1
- Implementation Tool
  - Xilinx ISE/WebPack Version : 9.1

**FPGA Board**
- Celoxica RC10
# Table of Contents

1. **Project Settings**

2. **Synthesis**
   - 2.1 Synthesis using Synplicity Synplify Pro
     - 2.1.1 Synthesis Options
     - 2.1.2 Synthesis Report Analysis
     - 2.1.3 RTL & Technology View
   - 2.2 Synthesis using Xilinx XST
     - 2.2.1 Synthesis Options
     - 2.2.2 Synthesis Reports Analysis
   - 2.3 Post–Synthesis Simulation

3. **Implementation**
   - 3.1 Implementation Options
   - 3.2 Implementation Reports Analysis
   - 3.3 Post-Implementation Simulation

4. **Uploading Bitstream to FPGA Board**
1. **Project Settings**

Start the workspace normally, but make sure you select *Create an Empty Design with Design Flow*. Then press **Next**. You will see a picture similar to the one shown on the next page.
Verify that **Flow Settings** are defined as followed:

- **Synplicity Synplify Pro**
- **ISE&Webpack Synthesis&Implementation**

**Implementation Tool**
- **Xilinx ISE/WebPack**
- **Xilinx9x SPARTAN3**

If not, click at the **Flow Settings** button and adjust appropriately.

Also choose,

- **Block Diagram Configuration**
  - Default HDL Language
  - Default HDL Language
  - VHDL

Once done, select **Next ➔ Finish**
Now you should see a familiar empty space with a Flow panel on the right side. If you do not see the Flow panel on the right side as shown in the picture, you can press **Alt+3** or **View ➔ Flow** from the top menu bar to open the panel.

Specify the new design name. Download to your hard drive all VHDL files provided to you at the website for lab3 demo.

Add and compile all files from lab3 demo. Then, test your design if it works correctly in the functional simulation as you would normally do. If you are following the tutorial by using lab3demo, make sure you change the *slow_clock_period* located inside *Lab3Demo_package.vhd* to a number suitable for simulation. It will take a long time to simulate otherwise.
2. Synthesis

Synthesis can be done using two different tools: Synplicity Synplify Pro and Xilinx XST. The former can be used only in school, the latter at home and in school. Please follow Sections 2.1 and 2.3 if you are using Synplify Pro, and Sections 2.2 and 2.3 if you are using Xilinx XST.

2.1 Synthesis using Synplicity Synplify Pro

2.1.1 Synthesis Options

Click at the options button next to the synthesis icon. Under Synthesis Options select Update synthesis order. Arrange your files in the order from the bottom to the top of the design hierarchy. Exclude your non-synthesizable files, such as your testbench. Also select a correct Top-level Unit, which is Lab3_demo in this example.

Make sure that your settings under General tab are as follows:
- Family : Xilinx9x Spartan3
- Device : 3s1500fg320
- Speed Grade : -4
- Run Mode : Batch

Then, select settings tab and choose the frequency of your device to 48 MHz instead of Auto Constraint. Press OK and click at the synthesis button. After synthesis, you can view the report by selecting the reports button located to the left of the synthesis button.
2.1.2 Synthesis Report Analysis

Minimum clock period (requested and estimated), slack (requested clock period minus estimated clock period), and resource utilization can be found from the log file generated after synthesis. To view the log file, click at the reports button next to the synthesis icon.

Minimum clock period can be found under Performance Summary section of the report. Respectively, one can determine the critical path by looking at the Worst Path Information. The report provides you with the 5 worst critical paths in your design. Looking at the critical paths can give you an idea of which portions of your code to change in order to improve the circuit performance.

Similarly, the resource utilization is located at the bottom of the log file. The report tells you the amount of resources the FPGA needs for the design.

Example Report: Timing

<table>
<thead>
<tr>
<th>Starting Clock</th>
<th>Requested Frequency</th>
<th>Estimated Frequency</th>
<th>Requested Period</th>
<th>Estimated Period</th>
<th>Slack</th>
<th>Clock Type</th>
<th>Clock Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab1_demo_clock</td>
<td>40.0 MHz</td>
<td>119.3 MHz</td>
<td>20.003</td>
<td>7.730</td>
<td>12.260</td>
<td>inferred</td>
<td>inferred_clkgroup_1</td>
</tr>
</tbody>
</table>

Example Report: Resource Utilization

---

Resource Usage Report for Lab3_demo

Mapping to part: xc3s50pq208-5

Cell usage:

- FDC: 27 uses
- FDCX: 4 uses
- UDF: 2 uses
- VERIFY I: 74 uses
- XORCY: 25 uses
- LUT1: 26 uses
- LUT2: 15 uses
- LUT3: 5 uses
- LUT4: 12 uses

I/O ports: 9
I/O primitives: 8
IBUF: 1 use
IBUF: 7 uses

I/O Register bits: 0
Register bits not including I/Os: 31 (2%)

Global Clock Buffers: 1 of 0 (12%)

Total load per clock:
Lab1_demo_clock: 31

Mapping Summary:
Total LUTs: 58 (3%)
Example Report: Worst Path Information

Worst Path Information
************************

Path information for path number 1:
- Requested Period: 20.833
- Setup time: 0.524
- Required time: 20.399
- Propagation time: 7.214
- Slack (critical): 13.096

Number of logic level(s): 6
Starting point: slow_clock_gen.counter[10] / 0
Ending point: counting.count_sig[0] / CE
The start point is clocked by lab5_dclk/clock [rising] on pin C
The end point is clocked by lab5_dclk/clock [rising] on pin C

<table>
<thead>
<tr>
<th>Instance / Net Name</th>
<th>Type</th>
<th>Pin Name</th>
<th>Pin Dir</th>
<th>Delay</th>
<th>Arrival Time</th>
<th>No. of Fan Out(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>slow_clock_gen.counter[10]</td>
<td>FDC</td>
<td>0</td>
<td>Out</td>
<td>0.626</td>
<td>0.626</td>
<td>-</td>
</tr>
<tr>
<td>slow_clock_gen.clk_div.mul_counter[12]</td>
<td>LUT3</td>
<td>10</td>
<td>In</td>
<td>-</td>
<td>1.340</td>
<td>-</td>
</tr>
<tr>
<td>slow_clock_gen.clk_div.mul_counter[12]</td>
<td>LUT3</td>
<td>0</td>
<td>Out</td>
<td>0.504</td>
<td>1.844</td>
<td>-</td>
</tr>
<tr>
<td>mul_counter[14]</td>
<td>FDC</td>
<td>-</td>
<td>-</td>
<td>0.515</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>slow_clock_gen.clk_div.mul_counter[17]</td>
<td>LUT4</td>
<td>10</td>
<td>In</td>
<td>-</td>
<td>2.159</td>
<td>-</td>
</tr>
<tr>
<td>slow_clock_gen.clk_div.mul_counter[17]</td>
<td>LUT4</td>
<td>0</td>
<td>Out</td>
<td>0.504</td>
<td>2.663</td>
<td>-</td>
</tr>
<tr>
<td>mul_counter[20]</td>
<td>FDC</td>
<td>-</td>
<td>-</td>
<td>0.515</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>slow_clock_gen.clk_div.mul_counter[20]</td>
<td>LUT4</td>
<td>10</td>
<td>In</td>
<td>-</td>
<td>2.578</td>
<td>-</td>
</tr>
<tr>
<td>slow_clock_gen.clk_div.mul_counter[20]</td>
<td>LUT4</td>
<td>0</td>
<td>Out</td>
<td>0.504</td>
<td>3.082</td>
<td>-</td>
</tr>
<tr>
<td>N_15</td>
<td>FDC</td>
<td>-</td>
<td>-</td>
<td>0.515</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>slow_clock_gen.clk_div.mul_counter[23]</td>
<td>LUT4</td>
<td>0</td>
<td>Out</td>
<td>0.504</td>
<td>4.501</td>
<td>-</td>
</tr>
<tr>
<td>G_5</td>
<td>FDC</td>
<td>-</td>
<td>-</td>
<td>0.515</td>
<td>-</td>
<td>13</td>
</tr>
<tr>
<td>G_5</td>
<td>LUT2</td>
<td>11</td>
<td>In</td>
<td>-</td>
<td>5.865</td>
<td>-</td>
</tr>
<tr>
<td>G_5</td>
<td>LUT2</td>
<td>0</td>
<td>Out</td>
<td>0.504</td>
<td>6.369</td>
<td>-</td>
</tr>
<tr>
<td>counting.count_sig[0]</td>
<td>FDC</td>
<td>-</td>
<td>-</td>
<td>0.645</td>
<td>-</td>
<td>4</td>
</tr>
</tbody>
</table>

*******************************************************

Total path delay (propagation time + setup) of 7.738 is 4.174(53.9%) logic and 3.564(46.1%) route.
2.1.3 RTL & Technology View

Investigation of the internal structure of your design after synthesis can be done by looking at the RTL and Technology views of your circuit.

RTL view is the schematic representation of the design in terms of generic logic components that are independent of the target technology (specific Xilinx FPGA), for example, in terms of multiplexers, adders, comparators, registers, counters, and logic gates.

Technology view is the schematic representation of the design in terms of components available in the target technology (specific Xilinx FPGA), for example, in terms of LUTs, flip-flops, fast carry logic, I/O blocks.

Hence, Technology view generally has more detailed/bigger diagram than RTL view. Viewing either one of them can be done by using Synplify Pro, which can be opened using RTL schematic button of the Flow panel.

Below is the basic layout of Synplify Pro.
Once open, selecting *.srs file will open RTL view of your design. Similarly, selecting *.sms file will open Technology view of your design.

RTL View:

Technology View:
Navigating through the components can be done by a **right-click** on any blank area and selecting **Push/Pop Hierachy**. Your mouse icon should now change from a cross sign to arrow sign, allowing you to click and navigate through the component, if possible.

Investigating the critical path of your circuit can be done by selecting **HDL-Analyst → Technology → Hierachichal Critical Path**. Once clicked, a page similar to the one below will be shown for lab3 demo.

A zoomed diagram of LUT3_01 is shown below. The red number on top of the component is showing delay and slack time of this circuit.

Lastly, clicking directly on the component in this view can take you straight to the vhdl source code.
2.2 Synthesis using Xilinx XST

2.2.1 Synthesis Options

Click at the options button next to the synthesis icon. Under Synthesis Option select Update synthesis order. Arrange your files in the order from the bottom to the top of the design hierarchy. Exclude your non-synthesizable files, such as testbench. Also select a correct Top-level Unit, which is Lab3_demo in this example.

Make sure that your settings under General tab are as follows:

- Family: Xilinx9x Spartan3
- Device: 3s1500fg320
- Speed Grade: -4

Under Std Synthesis and Adv Synthesis tabs, you can adjust optimization goal of the synthesis tool for various results. Most notably, you can tell the synthesis tool to optimize for either area or speed. To select either one of them, choose Std Synthesis → Optimization Goal → select Speed or Area.
2.2.2 Synthesis Report Analysis

Minimum clock period, critical path and resource utilization can be found from the log file generated after synthesis. To view the log file, click at the reports button next to the Synthesis icon.

Minimum clock period, maximum frequency and critical path can be found under Timing Summary section. Looking at the critical paths can give you an idea of which portions of your code to change in order to improve the circuit performance.

Resource utilization is located in the Final Report section.

Example Report: Resource Utilization

```
kinson Report
Final Results
Top Level Output File Name   : lab3 demo
Output Format               : .nc
Optimization Goal           : speed
Keep Hierarchy             : no

Design Statistics
# I/Os                      : 9

* Call Usage :                  
  # EELS                     : 155
  # END                      : 1
  # DNN                      : 7
  # LUT1                     : 1
  # LUT2                     : 4
  # LUT3                     : 35
  # LUT4                     : 9
  # MUXCY                    : 49
  # VCC                      : 1
  # XORCY                    : 32
  # FlipFlops/Latches       : 97
  # FDC                      : 37
  # Clock Buffers           : 1
  # BUFGF                    : 1
  # IO Buffers               : 9
  # ISEUF                    : 1
  # OSUF                     : 7

Device utilisation summary:

Selected Device : Ss1500fpg220-4

Number of Slices: 25 out of 13312 0%
Number of Slice Flip Flops: 37 out of 25624 0%
Number of 4 input LUTs: 86 out of 25624 0%
Number of I/Os: 9
Number of bonded I/Os: 9 out of 221 4%
Number of OClkS: 1 out of 6 12%
```
Example Report: Minimum Clock Period and Critical Path

Timing Detail:

Timing constraint: Default period analysis for clock 'clock'

Clock period: 30.00ns (frequency: 99.9999Hz)

Total number of path / destination pairs: 12000 / 33

Delay:
- 10.00ns (levels of Logic = 46)

Sources:
- slow_clock_gen/counter_16 (PP)

Destination:
- slow_clock_gen/counter_31 (PP)

Source Clock:
- clock rising

Destination Clock:
- clock rising

Path Path: slow_clock_gen/counter_16 to slow_clock_gen/counter_31

<table>
<thead>
<tr>
<th>Cell</th>
<th>IC1</th>
<th>Front</th>
<th>Delay</th>
<th>Delay Logical Name (Net Name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDTC</td>
<td>&lt;0</td>
<td>2</td>
<td>0.721</td>
<td>slow_clock_gen/counter_16 (slow_clock_gen/counter_16)</td>
</tr>
<tr>
<td>102T</td>
<td>ID=0</td>
<td>1</td>
<td>0.151</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
<tr>
<td>HSTC1</td>
<td>CT=0</td>
<td>1</td>
<td>0.634</td>
<td>slow_clock_gen/Monaur_counter_cmp_gcl0080 cyc0</td>
</tr>
</tbody>
</table>

15
2.3 Post-Synthesis Simulation

Click at the options button next to the post-synthesis simulation icon. Remove the default input file, and select your testbench as an input file by clicking at the button close to the cross sign (marked by a dot). Then, select Recompile Files. Once done, choose the appropriate top-level unit, which is lab3demo_tb.vhd in this example.

Press OK, and then select post-synthesis simulation. Now you should see timing waveforms similar to the ones obtained during functional simulation. The difference is that the components and signals are now mapped into appropriate FPGA hardware.
3 Implementation

3.1 Implementation Options

Click at the options button next to the implementation icon. Select the correct Netlist File which is a file with the same name as your top level VHDL file and the extension .edf. It is normally located in the synthesis folder of your workspace. Use this file to implement your design. Choose the correct FPGA Family, Device and Speed Grade, the same as used during the Synthesis phase:

In our example these are:
Family : Xilinx9x Spartan3
Device : 3s1500fg320
Speed Grade : -4

Under Constraint File, select Custom constraint file. Browse to your .ucf for the lab, lab3_demo.ucf in our example. Then, navigate to the BitStream tab by clicking at the right arrow at the top right hand
corner. Under General tab of BitStream deselect Do Not Run Bitgen. This will create bitstream, .bit, which you can upload to FPGA.

Also, under Post-Map STR, Post-PAR STR, and Simulation tabs make sure that your device speed grade is set to 4.

Similar to synthesis option for Xilinx XST as synthesis tool, you can specify the implementation tool to use a certain optimization goal. To do this, go to Advanced Map → Optimization Goal → select either Area or Speed.

Press OK, and then select implementation.

### 3.2 Implementation Reports Analysis

Similarly to synthesis, you can access the generated reports by clicking the reports button, near the implementation icon. Unlike synthesis log, implementation log is divided into several smaller reports, which are named differently. Below is a list of reports in which you can find the most useful information about your design after implementation, such as resource utilization, maximum clock frequency, and critical path:

**Resource Utilization:**
- Map: See Design Summary
- Place & Route: See Device Utilization Summary

Note: Place & Route provides overall information about the design after placing and routing. Map provides a more detailed summary of resource utilization.

**Minimum Clock period (Maximum Frequency):**
- Post-Place & Route Static Timing Report

This file describes the worst case scenario in terms of minimum clock period. However, since the implementation tools do not provide complete information, please refer to Timing Analysis below for a more detailed report.

Note: Post-Map Static Timing Report can be ignored because it provides timing report before placing & routing, and thus cannot correctly predict interconnect delays.

Pad file provides the mapping between FPGA pins and ports of your top-level unit (obtained based on the user constraint file .ucf). Please double check this report before running your design on the FPGA board.

Example: Mapping between the FPGA pin P10 and the clock input of the Lab3_Demo unit; the two neighboring pins P9 and P11 are marked as UNUSED

```
P9| |DIFFM| IO_L32P_5/GCLK2|UNUSED||5|| | | | | | | | | |
P10| |CLOCK| IO_L32P_4/GCLK2| INPUT| LVCKGS25| 4|| | |NONE| |LOCATED| |NO|NONE|
P11| |DIFFM| IO_L32P_4/D3|UNUSED||4|| | | | | | | | | |
```
Example Report: Minimum Clock Period

Clock to Setup on destination clock clock
-------------------------------+-----------------------------+-----------------------------+-----------------------------+-----------------------------+-----------------------------+-----------------------------+-----------------------------+
|-----------|----------|----------|----------|----------|----------|----------|----------|
Source Clock | 0.570 | | | | | | |
clock | 5.570 | | | | | | |

Example Report: Resource Utilization

**Device Utilization Summary:**

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of BUFGMUXes</td>
<td>1 out of 8 12%</td>
</tr>
<tr>
<td>Number of External IOBs</td>
<td>9 out of 221 4%</td>
</tr>
<tr>
<td>Number of I/OCEd IOBs</td>
<td>9 out of 9 100%</td>
</tr>
<tr>
<td>Number of Slices</td>
<td>30 out of 13312 1%</td>
</tr>
<tr>
<td>Number of SLICEMs</td>
<td>0 out of 6656 0%</td>
</tr>
</tbody>
</table>

**Design Summary**

Number of errors: 0
Number of warnings: 0

**Logic Utilization:**

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>31 out of 26,624 1%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>31 out of 26,624 1%</td>
</tr>
</tbody>
</table>

**Logic Distribution:**

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of occupied Slices</td>
<td>30 out of 13,312 1%</td>
</tr>
<tr>
<td>Number of Slices containing only related logic</td>
<td>30 out of 30 100%</td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic</td>
<td>0 out of 30 0%</td>
</tr>
</tbody>
</table>

*See NOTES below for an explanation of the effects of unrelated logic*

Total Number of 4 input LUTs: 56 out of 26,624 1%
Number used as logic: 31
Number used as a route-thru: 25
Number of bonded IOBs: 9 out of 221 4%
Number of OCLRs: 1 out of 3 12%

Total equivalent gate count for design: 554
Additional JTAG gate count for IOBs: 432
Peek Memory Usage: 153 MB
Total REAL time to MAP completion: 4 secs
Total CPU time to MAP completion: 1 secs

Timing Analysis (Clock period, Maximum Frequency and Critical Path)

For the detailed analysis of critical path and minimum clock period (or maximum frequency) a separate timing analyzer provided by Xilinx should be used. To generate the report, select **Analysis ➔ Static Timing Analyzer** from the **Flow** panel. This will open Xilinx Timing Analyzer. You can also navigate to the program from Windows menu by **Start ➔ All Programs ➔ VLSI Tools ➔ Xilinx ISE ➔ Accessories ➔ Timing Analyzer**.
Once the program is opened, select **Open**, choose netlist file located in /implement/ver1/rev1 of your workspace, *.ncd, and press **OK**. Selecting **Analyze against Auto Generated Design Constraints** will generate a static timing report.

**Example Report: Clock period, Maximum Frequency and Critical Path**

```
Timing constraint: Default OFFSET OUT AFTER analysis for clock 'clock_c'
34 items analyzed, 0 timing errors detected.
Maximum allowable offset is 13.75ns.

<table>
<thead>
<tr>
<th>Offset</th>
<th>13.75ns (clock path + data path + uncertainty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>counter/cout sig[0] (FF)</td>
</tr>
<tr>
<td>Destination</td>
<td>counter/cout sig[0] (PAD)</td>
</tr>
<tr>
<td>Source Clock</td>
<td>clock_c sizing</td>
</tr>
<tr>
<td>Data Path Delay</td>
<td>11.75ns (Levels of Logic = 2)</td>
</tr>
<tr>
<td>Clock Path Delay</td>
<td>2.015ns (Levels of Logic = 2)</td>
</tr>
<tr>
<td>Clock Uncertainty</td>
<td>0.000ns</td>
</tr>
</tbody>
</table>

**Clock Path: clock_to_counter/cout sig[0]**

<table>
<thead>
<tr>
<th>Delay type</th>
<th>Delay(ns)</th>
<th>Logical Resource(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>0.029</td>
<td></td>
</tr>
<tr>
<td>net (fanout=1)</td>
<td>0.001</td>
<td>clock_to_IBUF</td>
</tr>
<tr>
<td>net (fanout=19)</td>
<td>0.785</td>
<td>clock_c</td>
</tr>
</tbody>
</table>

**Total:** 2.015ns (3.230ns logic, 0.786ns route) (61.0% logic, 39.0% route)

**Data Path: counter/cout sig[1] to S.Send[0]**

<table>
<thead>
<tr>
<th>Delay type</th>
<th>Delay(ns)</th>
<th>Logical Resource(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>0.720</td>
<td></td>
</tr>
<tr>
<td>net (fanout=12)</td>
<td>1.644</td>
<td>count[0]</td>
</tr>
<tr>
<td>T100</td>
<td>0.638</td>
<td>s.env sd 0-0.144</td>
</tr>
<tr>
<td>net (fanout=2)</td>
<td>0.595</td>
<td>s.env sd 0-0.34</td>
</tr>
<tr>
<td>net (fanout=1)</td>
<td>0.608</td>
<td>s.env sd 0-0.3</td>
</tr>
<tr>
<td>net (fanout=1)</td>
<td>2.442</td>
<td>s.Send[1][1]</td>
</tr>
<tr>
<td>net (fanout=1)</td>
<td>5.131</td>
<td>s.Send[1][2]</td>
</tr>
</tbody>
</table>

**Total:** 11.75ns (7.007ns logic, 4.666ns route) (60.2% logic, 39.8% route)
3.3 Post-Implementation Simulation

Click at the **options** button next to the **timing simulation** icon. Select your testbench as the Top-Level Unit. Afterwards, select **timing simulation**, which will generate timing waveforms based on your netlist after implementation. You should notice slight timing delays compared to the waveforms from your post-synthesis simulation & functional simulation.
4. Uploading Bitstream to FPGA Board

Before uploading Bit file, make sure that you change your constant values in all your files to proper values, and re-synthesize/re-implement all the files. In particular, in our example, please change the value of the constant slow_clock_period in the Lab3Demo_package.vhd.

Select FTU3 program as shown in the picture above. When the program is opened, a device will be shown if it is connected and recognized. Select your FPGA and click open. Then, Clear FPGA and select the bit file located in implement/ver1 of your workspace. Upload (Configure) the code and test your design whether it works correctly on the FPGA board.

Good luck! Have fun debugging =)