Tutorial on FPGA Design Flow based on Aldec Active HDL Ver 1.5
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
- Synthesis Tool
  - ISE&Webpack Synthesis&Implementation
- Implementation Tool
  - Xilinx ISE/WebPack

**FPGA Board**
- Digilent Basys2

**Combinations of tools supported as of Fall 2010 are as follows:**

**At home:**
- Aldec Active-HDL Student Edition ver. 7.2
- Xilinx ISE/Webpack 9.1 or 10.1

**At GMU:**
- Aldec Active-HDL ver. 8.2
- Xilinx ISE/Webpack 9.1, 10.1, 11.4, or 12.1.
# Table of Contents

1. **Project Settings**

2. **Synthesis**
   - 2.1 Synthesis using Xilinx XST
     - 2.1.1 Synthesis Options
     - 2.1.2 Synthesis Reports Analysis
   - 2.2 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
Create new workspace and choose Create an Empty Design with Design Flow.

Then press Next. You will see a picture similar to the one shown below.

Verify that Flow Configuration Settings are defined as followed:

- **Synthesis Tool**
  - *Xilinx ISE/WebPack <version number>* XST

- **Implementation Tool**
  - *Xilinx ISE/WebPack <version number>*

- **Default Family**
  - *Xilinx<version number>*x SPARTAN3

If not, click at the Flow Configuration Settings button and adjust appropriately.
Also choose,

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

Once done, select **Next** ➔ **Finish**
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 (5). It will take a long time to simulate otherwise.
Now you should see a screen divided into several parts, 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.
2. Synthesis

Synthesis can be done using Xilinx XST.

2.1 Synthesis using Xilinx XST

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 testbench. Also select a correct **Top-level Unit**, which is Lab3_demo in the example you follow.

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

- **Family**: Xilinx11x Spartan3
- **Device**: 3s50p2q208
- **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**.
2.1.2 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

--XILINX Synthesis--
No Partitions were found in this design.

Final Results
RTL Top Level Output File Name : \input\_design.ncd
Top Level Output File Name : \input\_design.ncd
Output Format : nnc
Optimization Goal : speed
Java Hierarchy : no

Design Statistics
\# JOB : 9
Cell Group :
\# XILS : 104
\# AND : 1
\# INV : 1
\# IOTI : 11
\# IOII : 19
\# IOTII : 1
\# IOTT : 16
\# MUX : 19
\# MCC : 1
\# XOR : 32
\# FlipFlops/latches : 37
\# DCC : 13
\# DFF : 4
\# Clock Buffers : 1
\# DFF : 1
\# MUX : 1

Device utilization summary:
--------
Selected Device : 3s50pgq108-4

Number of Slices: 45
Number of Slice Flip Flops: 37
Number of 4 input LUTs: 81
Number of I/Os: 9
Number of bonded I/Os: 9
Number of OLRBs: 1

Partition Resource Summary:
# Example Report: Minimum Clock Period and Critical Path

### Timing Summary:

- **Speed Grade:** –
- **Minimum period:** 7.497ns (Maximum Frequency: 133.10MHz)
- **Maximum output setup time before clock:** No path found
- **Maximum output hold time after clock:** No path found

### Timing Details:

All values displayed in nanoseconds (ns)

| Timing constraints | Default period manager for Clock 'clock'
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock period:</td>
<td>7.497ns (frequency: 133.10MHz)</td>
</tr>
<tr>
<td>Total number of paths / destination ports:</td>
<td>1000 / 41</td>
</tr>
</tbody>
</table>

### Delay:

- **Delay:** 7.497ns (Levels of Logic = 33)
- **Source:** slow_clock_gen/counter_31 (FF)
- **Destination:** clock_rising

### Data Paths:

- **Data Path:** slow_clock_gen/counter_1 to slow_clock_gen/counter_31

<table>
<thead>
<tr>
<th>Cell</th>
<th>input</th>
<th>output</th>
<th>delay</th>
<th>logical name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA</td>
<td>2</td>
<td>2,216</td>
<td>0.720</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>LUT1</td>
<td>5</td>
<td>6,658</td>
<td>0.051</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX1</td>
<td>5</td>
<td>6,652</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX2</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX3</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX4</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX5</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX6</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX7</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX8</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX9</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX10</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX11</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX12</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX13</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX14</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX15</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX16</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX17</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX18</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX19</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX20</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX21</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX22</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX23</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX24</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX25</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX26</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX27</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
<tr>
<td>MUX28</td>
<td>5</td>
<td>6,656</td>
<td>0.056</td>
<td>slow_clock_gen/counter_1</td>
</tr>
</tbody>
</table>

```python
# Example code

def report_min_clock_period(critical_path):
    # Code to calculate and display minimum clock period
    # and critical path details

report_min_clock_period()```

---

15
2.2 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 : Xilinx11x Spartan3
Device : 3s50pq208
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.

![Implementation Options](image)

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

You can also 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
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 the Windows menu. The path used in the ECE labs at GMU is 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 ALTER analysis for clock "clock_C"

34 items analyzed, 0 timing errors detected.
Maximum allowable offset is: 13.75ns.

| Offset: 13.75ns (clock path + data path + uncertainty) |
| Source: counting/count.sig(0) (FF) |
| Destination: 5.Seq(11) (PAD) |
| Source Clock: clock_C rising |
| Data Path Delay: 11.75ns (Levels of Logic = 3) |
| Clock Path Delay: 2.016ns (Levels of Logic = 2) |
| Clock Uncertainty: 0.000ns |

Clock Path: clock to counting/count.sig(0)

<table>
<thead>
<tr>
<th>Delay type</th>
<th>Delay(ns)</th>
<th>Logical Resource(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tclock</td>
<td>0.829</td>
<td>clock</td>
</tr>
<tr>
<td>net (fanout=1)</td>
<td>0.001</td>
<td>clock_iobuf1/IIOFG</td>
</tr>
<tr>
<td>Ttile</td>
<td>0.401</td>
<td>clock_iobuf1/IIOFG</td>
</tr>
<tr>
<td>net (fanout=19)</td>
<td>0.785</td>
<td>clock_c</td>
</tr>
</tbody>
</table>

Total Delay: 2.016ns (1.253ns logic, 0.763ns route) (61.0% logic, 39.0% route)

Data Path: counting/count.sig(0) to 5.Seq(11)

<table>
<thead>
<tr>
<th>Delay type</th>
<th>Delay(ns)</th>
<th>Logical Resource(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tclock</td>
<td>0.720</td>
<td>counting/count.sig(0)</td>
</tr>
<tr>
<td>net (fanout=12)</td>
<td>1.441</td>
<td>counting(0)</td>
</tr>
<tr>
<td>Ttile</td>
<td>0.408</td>
<td>seven_seq 0.0 $4df1 1 0 2</td>
</tr>
<tr>
<td>net (fanout=2)</td>
<td>0.585</td>
<td>seven_seq 0.0 $45</td>
</tr>
<tr>
<td>Ttile</td>
<td>0.000</td>
<td>seven_seq 0.0 $51</td>
</tr>
<tr>
<td>net (fanout=1)</td>
<td>2.442</td>
<td>N 3</td>
</tr>
<tr>
<td>Ttile</td>
<td>5.131</td>
<td>5.Seq(0)</td>
</tr>
<tr>
<td>5.Seq(11)</td>
<td></td>
<td>5.Seq(11)</td>
</tr>
</tbody>
</table>

Total Delay: 11.735ns (7.067ns logic, 4.668ns route) (61.2% logic, 38.8% 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 the Adept program as shown in the picture above. When the program is opened, a device will be shown if it is connected and recognized. Select the bit file by clicking Browse and finding the appropriate file. Click Program to program the file device.
Good luck! Have fun debugging =)