FPGAs 2

Some of the Slides picked from Xilinx Educational Resources
FPGA Design Flow

1. Functional Simulation of your Design
2. Adding Design Constraints
3. Synthesizing and Optimizing your Design
4. Evaluating your Design Size and Performance
5. Placing and Routing your Design
6. Generating a Bitstream
7. Timing Simulation of your Design
8. Evaluating your Design's Coding Style and System Features
9. Static Timing Analysis
10. Downloading to the Device, In-System Debugging
Functional Simulation

• Verify syntax and functionality

• Perform Separate Simulations
  – With larger hierarchical Hardware Description Language (HDL) designs, perform separate simulations on each module before testing your entire design.
  – Easier to debug your code.

• Verify entire module
  – Once each module functions as expected, create a test bench to verify that your entire design functions as planned.
Setting Constraints

- Allows you to control timing optimization
- Uses synthesis tools and implementation processes more efficiently
- Helps minimize runtime and achieve your design requirements
- You can add the following constraints:
  - Clock frequency or cycle and offset
  - Input and Output timing
  - Path timing
  - Global timing
Synthesis

• We usually design using register-transfer-level (RTL) Verilog
  – Higher level of abstraction than gates
• Synthesis tool translates to a circuit of gates that performs the same function
• Specify to the tool
  – the target implementation fabric
  – constraints on timing, area, etc.
• Post-synthesis verification
  – synthesized circuit meets constraints
Physical Implementation

• Implementation fabrics
  – Application-specific ICs (ASICs)
  – Field-programmable gate arrays (FPGAs)
• Floor-planning: arranging the subsystems
• Placement: arranging the gates within subsystems
• Routing: joining the gates with wires
• Physical verification
  – physical circuit still meets constraints
  – use better estimates of delays
Pin Assignment

• The process of assigning design ports to FPGA IO pins, requires:
  • Configuring direction (input/output/inout)
  • Defining signaling standard for each of the pins
Pin Assignment Related Windows

- **I/O Port** lists all top-level ports
- **I/O Port Properties** allows configuring pins
- **Device View** shows resources and area constraints
- **Package View** shows pins, I/O banks are color-coded
I/O Layout Guidelines

- I/O for control signals on the top or the bottom
  - Signals are routed vertically
- I/O for data buses on the left or the right
  - FPGA architecture favors horizontal data flow
Data Bus Layout

- Arithmetic functions with more than five bits use carry logic
- Carry chains require specific vertical orientation
Reading Reports

• 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
Device Utilization Summary

Get quick access to used and available resources through the FPGA Design Summary.

Design Utilization Summary

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>141</td>
<td>9,312</td>
<td>1%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>281</td>
<td>9,312</td>
<td>3%</td>
</tr>
<tr>
<td>Number of occupied Slices</td>
<td>162</td>
<td>4,656</td>
<td>3%</td>
</tr>
<tr>
<td>Number of Slices containing only related logic</td>
<td>162</td>
<td>162</td>
<td>100%</td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic</td>
<td>0</td>
<td>162</td>
<td>0%</td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>291</td>
<td>9,312</td>
<td>3%</td>
</tr>
</tbody>
</table>

- Number used as logic: 177
- Number used as a route-thru: 10
- Number used for Dual Port RAMs: 16
- Number used for 102x1 RAMs: 52
- Number used as Shift registers: 36
- Number of bonded IOBs: 21
- Available: 232
Device Utilization Summary

Get quick access to used and available resources from the Map report.

Access the Map report through the Detailed Reports.
Post-Map Static Timing Report

Evaluate logic delays to see if you should proceed to place & route
Analyze Logic Timing

Look at critical paths to determine if timing is reasonable

Select critical paths

Source and destination registers of path illustrated

Detailed listing of logic and estimated routing delays

Total delay
Post-Place & Route Static Timing Report

Determine if the constraints were met

Access Post-Place & Route Static Timing report
Timing Summary

Provides statistics on average routing delays and performance versus constraints

Access Place & Route report through Detailed Reports

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Period Requirement</th>
<th>Actual Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0_clk</td>
<td>10.000ns</td>
<td>2.025ns</td>
</tr>
<tr>
<td>T0_Insc_clkgen_CLKD_BUF</td>
<td>10.000ns</td>
<td>N/A</td>
</tr>
</tbody>
</table>

All constraints were met.
INFO: Timing: 2761 - N/A entries in the Constraints list may indicate that the constraint does not cover any paths or that it has no requested value.

Period Requirement for global clock is 10 ns
Actual Period of 2.025 ns
Timing Constraints

What effects do timing constraints have on your project?

- The implementation tools do not attempt to find the Place & Route that will obtain the best speed
  - Instead, the implementation tools try to meet your performance expectations

- Performance expectations are communicated with timing constraints
  - Timing constraints improve the design performance by placing logic closer together so that shorter routing resources can be used
  - Note: The Constraints Editor refers to the Xilinx Constraints Editor