FPGA Design
Part VI - Advanced Topics

Thomas Lenzi
High-Speed Design Considerations
Primitives

• It is possible to implement entities that represent the building blocks of the FPGA.

• Those entities are called Primitives.

• Each FPGA family has its own set of primitives which are listed in user guides.

• Using primitives can be useful in certain cases when a specific function is needed (clock multiplexing, …) or when you want to optimise your design beyond what ISE does.
Propagation Delays

• Every component in the FPGA adds delay to the signal propagation.

• If you are working at high speed, the delay between the entry and exit point of your logic should be shorter than the period of the clock.

• If this is not the case, you will encounter undefined behaviour.
Smaller Logic

- To run at high speed, you need to break down your logic.

- Instead of comparing two 32 bits numbers in one clock cycle, which requires 11+2+1 LUTs (cascaded), you could spread the comparison over three clock cycles.
Buffering

• Insert DFFs in the logic to allow for faster clock speeds.

• If each inverter adds a delay of 5 ns to the logic, your clock will only be able to run at a period of 10 ns in the first case, while it can run at a period of 5 ns in the second one.
Example: 2x8bits comparator

• Stage 1: 
  5 LUTs, 1 DFF

```vhdl
process(clk)
begin
  if (rising_edge(clk)) then
    if (a = b) then
      dout <= '1';
    else
      dout <= '0';
    end if;
  end if;
end process;
```

Timing summary:

Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)
Constraints cover 0 paths, 0 nets, and 0 connections

Design statistics:

Minimum period: 3.192ns(1)  (Maximum frequency: 313.283MHz)
Example: 2x8bits comparator

• Stage 1:
  4 LUTs, 4 DFFs

• Stage 2:
  1 LUT, 1 DFF

```-- Stage 1
stagela_lut : lut4
generic map(init => x"9009")
port map(i0 => a(0), i1 => b(0), i2 => a(1), i3 => b(1), o => stage_1(0));

stagelb_lut : lut4
generic map(init => x"9009")
port map(i0 => a(2), i1 => b(2), i2 => a(3), i3 => b(3), o => stage_1(1));

stage1d_buf : fdce
generic map(init => '0')
port map(C => clk, D => stage_1(3), Q => stage_1_buf(3), CE => '1', CLR => '0');
```

Timing summary:
--------------
Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)
Constraints cover 0 paths, 0 nets, and 0 connections

Design statistics:
Minimum period:  1.596ns(1)  (Maximum frequency: 626.566MHz)
Pipelining

- Splitting your processes in multiple tasks doesn’t mean your code will run less efficiently.
- You can for example implement pipelining, which allow two events to be processed at the same time by your machine.
How to Pipeline

• Do not make a blocking state machine in which only one event can be processed.

• Transitions between states should be done at each clock cycle so that events can follow one another.

• One process per action can help you visualise the workflow.

• Process data even if the input data is not valid. Let the machine run.

• Use a flag that is carried along the pipeline to indicate if the data is valid.
Multiplexing

- If pipelining is not an option (task too complex to breakdown), you can consider multiplexing.

- In multiplexing, a system is duplicated N times (N is related to the execution time of the process). At each cycle, a different system is fed with information.
Timing Analysis
Timing Constraints

• You can set constraints on the clock’s period in a UCF file.

• The tools will translate the design so that they can match the constraints. If the constraints cannot be matched, an error occurs.

```plaintext
NET "clk" TNM_NET = "clk";
TIMESPEC TS_clk = PERIOD "clk" 20 ns;
```
• ISE can generate timing reports that analyse your logic and give you information about the delays, skews, ...

• To start a timing analysis, go to “Tools > Timing Analyser > Post Place & Route…”. This option will take into account the path’s delay. The “Post Map” option will only compute the components delay.
Floorplanning
Floorplanning

- Floorplanning allows you to define regions of the FPGA in which the Xilinx tools should place certain components.

- You can even go as far as placing the LUTs, DFFs, … manually.
Hierarchical Design

• When synthesising your design, you can tell ISE to keep the hierarchy of your design.

• This means that the entities will keep their name and be easier to place on the FPGA. You will be able to place certain components in given regions.
Problem

- Optimise the logic shown on the left for area by placing the components manually.

- You should first translate it to building blocks.
Ressources


• UCF to use: http://iihe.ac.be/~tlenzi/challenge.ucf