ECE 337: ASIC Design Lab 2

Week of 8/26/2013

What's involved in Lab 2

- Coding and Verifying the Sensor Detector
- Viewing the Schematic forms of the Sensor Detector
- Synthesis Optimization
- Coding and Verifying the Building blocks from Lab 1

Dataflow Coding Example

```
module multiply
(
  input  wire [3:0] a,
  input  wire [3:0] b,
  output wire [7:0] p
);
   wire [7:0] st1;
   wire [7:0] st2;
   wire [7:0] st3;

   assign st1 = (b[0]== 1'b1) ? {4'b0000,a}: 8'b00000000;
   assign st2 = (b[1]== 1'b1) ? ({3'b000,a,1'b0} + st1): st1;
   assign st3 = (b[2]== 1'b1) ? ({2'b00,a,2'b00} + st2): st2;
   assign P   = (b[3]== 1'b1) ? ({1'b0,a,3'b000} + st3): st3;
endmodule
```

Behavioral Coding Example

```
module multiply
(
   input  wire [3:0] a,
   input  wire [3:0] b,
   output reg  [7:0] p
);
   reg [2:0] i;
   reg [8:0] shifted_a;

   always @ (a,b)
   begin
      shifted_a = {4'b0000,a};
      p = 8'b00000000;
      for (i = 0; i < 4; i = i + 1)
      begin
         if (b[i] == 1b'1) p = p + shifted_a;
         shifted_a = {shifted_a[6:0],1b0};
      end
   end
endmodule
```

Structural Coding

- Use to break up design into sub-blocks
- Use **port map** to connect corresponding wires
- Modelsim will interface with the latest version compiled into your work library.

Structural Coding Example

```
Module half_adder
(
  input  wire a,
  input  wire b,
  output wire s,
  output wire c_out
);

  xor sum_gen (s, a, b);
  and carry_gen (c_out, a, b);
endmodule
```

Design Compiler Commands

```
# Step 1:  Read in the source file
analyze -format sverilog -lib WORK {sensor_b.sv}       # parse the code
# map code into technology-independent logic gates
elaborate protein5_d -lib WORK -update

# Step 2: Set design constraints
# Uncomment below to set timing, area, power, etc. constraints
# set_max_delay <delay> -from "<input>" -to "<output>"
# set_max_area <area>
# set_max_total_power <power> mW

# Step 3: Compile the design, map to target cell library
compile -map_effort medium
```

Design Compiler Commands Continued

```
# Step 4: Output reports
current_design .
report_timing -path full -delay max -max_paths 1 -nworst 1
        > reports/$current_design.rep
report_area >> reports/$current_design.rep
report_power -hier >> reports/$current_design.rep

# Step 5: Output final VHDL and Verilog files
# create structural-style code of individual gates
#   textual description of gate-level design schematic
write -format verilog -hierarchy -output "mapped/$current_design.v"
echo "\nScript Done\n"
echo "\nChecking Design\n"
check_design
exit
```

Modifying The Makefile Synthesis

- All of the previous synthesis commands are stored in the variable “SYN_CMDS” as a contiguous string.
  - The “\” symbol in make causes the current line to continue/wrap onto the following text line
  - The “\n” character “terminates” the command line
  - The combination of “\n\” at the end of each “line” keeps the whole variable definition as one string
    - This must be maintained for it to work
- The $(if… make function adds in a clock constraint if the “CLOCK_NAME” and “CLOCK_PERIOD” variables are populated

Using Subversion

- Import to SVN
  - svn import Lab2 $rdir/Lab2/trunk –m “import Lab2”
- Delete existing
  - rm –rf Lab2
- Check Out from SVN
  - svn co $rdir/Lab2/trunk Lab2
- Check In to SVN
  - svn add source/* scripts/* --force
  - svn ci –m “modified/added ? in Lab2”