

Building Combinatorial Circuit Using Behavioral Modeling Lab

Overview:

In this lab you will learn how to model a combinatorial circuit using behavioral modeling style of Verilog HDL. You will model a combinatorial circuit that monitors 8 switch input and output non-zero value (equal to the switch number that is ON + 2) when one and only one switch is true on two seven-segment display.

Outcome:

You will understand how to develop behavioral constructs to model a combinatorial circuit in Verilog HDL. You will use ISE simulator to simulate the design. You will add user constraint file (ucf) to assign pins so the design can be targeted to National Instruments (NI) Digital Electronics FPGA Board. You will implement the design and create a bitstream file using ISE's implementation tools. Once bitstream is created, you will download using ISE's iMPACT program and verify the design functionality.

Background:

The Behavioral style modeling in Verilog HDL provides two statement constructs. They are *initial* statement and *always* statement. Only reg data type can be assigned a value in either of these statements. Both of these statements can be a single statement or may contain a block of statements. Inside the block the order in which the statements are executed will be based on the order in which they are encountered, i.e. they are executed sequentially. The assignment statements used inside the block are called blocking procedural assignment statements. They may optionally have a delay. The delay can be inter-statement delay or intra-statement delay. The inter-statement delay is the delay by which a statement's execution is delayed, where as intra-statement delay is the delay between computing the value of the right-hand side expression and its assignment to the left-hand side.

The initial statement, as the name indicates, executes once and only at the beginning of a simulation execution. It is used to generate stimulus in testbench. There can be multiple of initial statements. They execute concurrently. During synthesis, synthesis tool may flag error when it encounters initial statement.

The always statement always executes in a loop. In simulation mode, it executes as long as the simulation is active. It can be called repeatedly when certain condition is met. Synthesis tool will generate hardware based on the way the always statement is written.

The syntax for initial block statement is:

```
initial
begin
    Blocking procedural statements
    ...
end
```

The syntax for always block statement is:

```
always @ (sensitivity list)
```

```
begin
  Blocking procedural statements
  ...
end
```

The sensitivity list may comprise of signals (separated by or keyword) which are monitored (checked) inside the block statement if you are modeling a combinatorial logic or it can be edge sensitive signal list (separated by or keyword) if you are modeling a sequential logic.

As an example, if you want to model a 2-to-2 multiplexer, the model may look like

always @ (a or b or sel)

```
begin
  if (sel==1'b1)
    y = a;
  else
    y = b;
end
```

In the above example, a or b or sel forms a sensitivity list. Note that they all have to appear in the list as they are read (either through source operand or checking using if statement) inside the block. You also must provide all possible alternatives in case of conditional statements like if-else. If you don't include all the signals in the sensitivity list or if you don't include all possible input combination (for example you leave out else part in the above example) then a latch will be inferred which can cause mismatch between what was expected and what got inferred.

References:

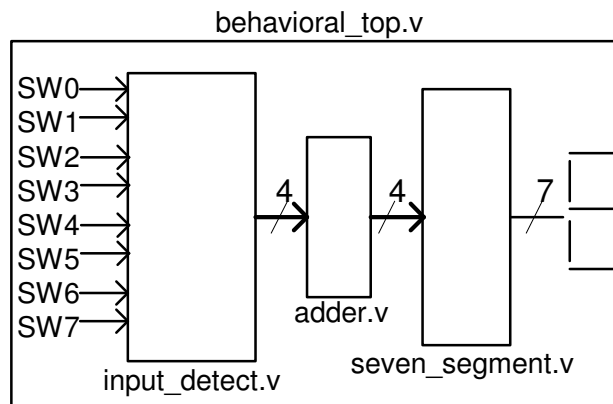
1. National Instruments' Digital Electronics FPGA Board user manual
2. Verilog HDL books
 - Stephen Brown, Zvonko G. Vranesic, "Fundamentals of Digital Logic with Verilog Design", 2002
 - Zainalabedin Navabi, "Verilog Digital Systems Design: RT Level Synthesis, Testbench, and Verification", 2005
 - Samir Paltinkar, "Verilog HDL: A Guide to Digital Design and Synthesis", 2003
 - Joseph Cavanagh, "Verilog HDL: Digital Design and Modeling", 2007
 - Michael D. Ciletti, "Modeling, Synthesis, and Rapid Prototyping with Verilog HDL", 2003
 - Douglas J. Smith, "HDL Chip Design: A Practical Guide for Designing, Synthesizing and Simulating ASICs and FPGAs using VHDL or Verilog", 1996
3. On-line references:
 - Verilog HDL Reference Card: http://www.stanford.edu/class/ee183/handouts_win2003/VerilogQuickRef.pdf

Problem Statement:

Design a combinatorial circuit that monitors 8 switch input and output non-zero value (equal to the switch number that is ON + 2), when one and only one switch is true, on two seven-segment display.

Implementation:

The circuit to be designed consists of eight inputs. When one and only one of its input is true, the input number + 2 must be displayed. Thus the possible output value will be 2 to 9. This output value is to be displayed on a seven-segment display. The binary output needs to be converted to seven-segment compatible output. This will require additional conversion block which takes binary input of 2 to 9 and converts it to seven segment compatible outputs. The hierarchical block diagram of the complete system is shown below.



Each lower-level block will be modeled using behavioral modeling style where as top-level model will use module instantiation and connect them using structural modeling style.

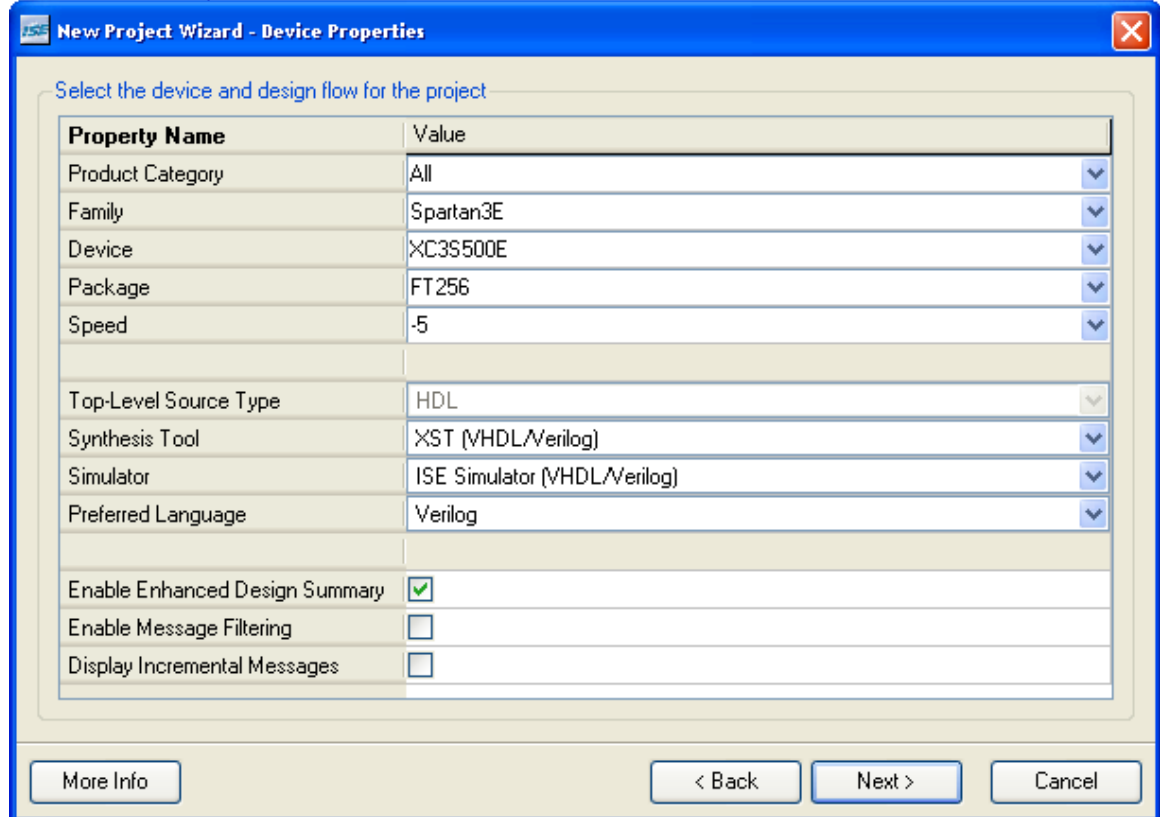
Procedure:

Extract resources.zip file in c:\NI\Verilog_Labs folder

1. Create a ISE project

- Launch ISE: Select **Start** → **Programs** → **Xilinx ISE Design Suite 10.1** → **ISE** → **Project Navigator**
- In the Project Navigator, select **File** → **New Project**. The New Project Wizard opens
- For Project Location, use the “...” button to browse to C:\NI\Verilog_labs, and then click **OK**
- For Project Name, type *behavioral_lab*
- Click **Next**
- Select the following options and click **Next**
 - ✓ Device Family: **Spartan3E**
 - ✓ Device: **xc3s500E**
 - ✓ Package: **ft256**
 - ✓ Speed Grade: **-5**
 - ✓ Synthesis Tool: **XST (VHDL/Verilog)**

- ✓ Simulator: **ISE Simulator (VHDL/Verilog)**
- ✓ Preferred Language: **Verilog**

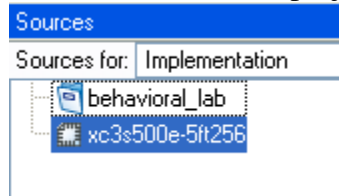



Select the device and design flow for the project

Property Name	Value
Product Category	All
Family	Spartan3E
Device	XC3S500E
Package	FT256
Speed	-5
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISE Simulator (VHDL/Verilog)
Preferred Language	Verilog
Enable Enhanced Design Summary	<input checked="" type="checkbox"/>
Enable Message Filtering	<input type="checkbox"/>
Display Incremental Messages	<input type="checkbox"/>

More Info < Back Next > Cancel

- The **Create New Source** dialog will appear. Click **Next**
- A *Add Existing Sources* form will be displayed. Click **Next** as we do not want to add
- Click **Finish**. An project will be created.



- Click  or **File** → **New** to create a *blank text* file.
- Enter the following model for the input detect unction using behavioral modeling style

```

1  module input_detect (
2      input SW0,
3      input SW1,
4      input SW2,
5      input SW3,
6      input SW4,
7      input SW5,
8      input SW6,
9      input SW7,
10     output reg good, // if input is good then good=1 else =0
11     output reg [3:0] input_out // output 4-bit value representing switch number
12 );
13
14     always @ (SW7 or SW6 or SW5 or SW4 or SW3 or SW2 or SW1 or SW0)
15     begin
16         #2 good = 1'b1;
17         case ({SW7,SW6,SW5,SW4,SW3,SW2,SW1,SW0})
18             8'b00000001 : #2 input_out = 4'h0;
19             8'b00000010 : #2 input_out = 4'h1;
20             8'b00000100 : #2 input_out = 4'h2;
21             8'b00001000 : #2 input_out = 4'h3;
22             8'b00010000 : #2 input_out = 4'h4;
23             8'b00100000 : #2 input_out = 4'h5;
24             8'b01000000 : #2 input_out = 4'h6;
25             8'b10000000 : #2 input_out = 4'h7;
26             default : begin
27                 #2 input_out = 4'h0;
28                 #2 good = 1'b0;
29             end
30         endcase
31     end
32
33 endmodule
34
35

```

- Save the file as **input_detect.v** and notice the text changes into context driven (Verilog language sensitive) format
- Close the file
- Create a new blank text file
- Enter the following model to add 2 to the input data and output

```

1  module adder (
2      input [3:0] datain,
3      output reg [3:0] dataout
4  );
5
6
7      always @ (datain)
8      begin
9          #2 dataout = datain + 2;
10     end
11
12 endmodule
13
14

```

- Save the file as **adder.v** and close it
- Create a new blank text file
- Enter the following model to take the binary input and output for hex display. Also check if the current data is good data or not. If not, then display 0

```

1  module seven_segment (
2      input good,
3      input [3:0] datain,
4      output reg [6:0] segments
5  );
6
7      always @ (good or datain)
8      begin
9          if (good)
10             begin
11                 case (datain)
12                     4'd2 : #2 segments = 7'b1011011;
13                     4'd3 : #2 segments = 7'b1001111;
14                     4'd4 : #2 segments = 7'b1100110;
15                     4'd5 : #2 segments = 7'b1101101;
16                     4'd6 : #2 segments = 7'b1111101;
17                     4'd7 : #2 segments = 7'b0000111;
18                     4'd8 : #2 segments = 7'b1111111;
19                     4'd9 : #2 segments = 7'b1100111;
20                     default : #2 segments = 7'b0000000;
21                 endcase
22             end
23             else
24                 begin
25                     #2 segments = 7'b0111111;
26                 end
27             end
28         end
29     endmodule
30

```

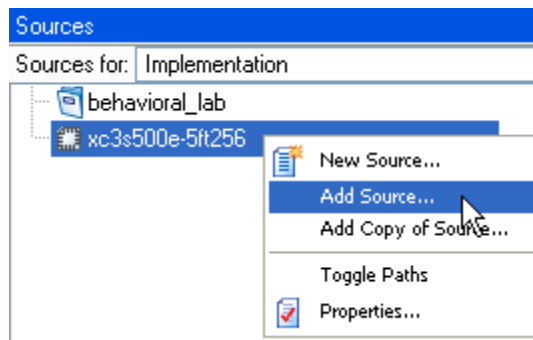
- Save the file as **seven_segment.v** and close it
- Create a new blank text file
- Enter the following model to instantiate the three lower-level modules

```

1  module behavioral_top (
2      input SW0,
3      input SW1,
4      input SW2,
5      input SW3,
6      input SW4,
7      input SW5,
8      input SW6,
9      input SW7,
10     output [6:0] segments
11 );
12
13 wire good;
14 wire [3:0] datain, dataout;
15
16 input_detect I1 (
17     .SW0(SW0),
18     .SW1(SW1),
19     .SW2(SW2),
20     .SW3(SW3),
21     .SW4(SW4),
22     .SW5(SW5),
23     .SW6(SW6),
24     .SW7(SW7),
25     .good(good),
26     .input_out(datain)
27 );
28 adder A1 (
29     .datain(datain),
30     .dataout(dataout)
31 );
32 seven_segment S1 (
33     .good(good),
34     .datain(dataout),
35     .segments(segments)
36 );
37 endmodule
38

```

- Save the file as **behavioral_top.v** and close it
- Note that even though the files have been created, they are not automatically added to the project as they were created as blank text files
- To add the files, select the chip in the *Sources* window and right-click and then select **Add Source...**



- Select all the verilog model files (behavioral_top.v, input_detect.v, adder.v, and seven_segment.v) and click **Open**
- Click **OK** to add the file
- Create a new blank file and enter the following location constraints

```

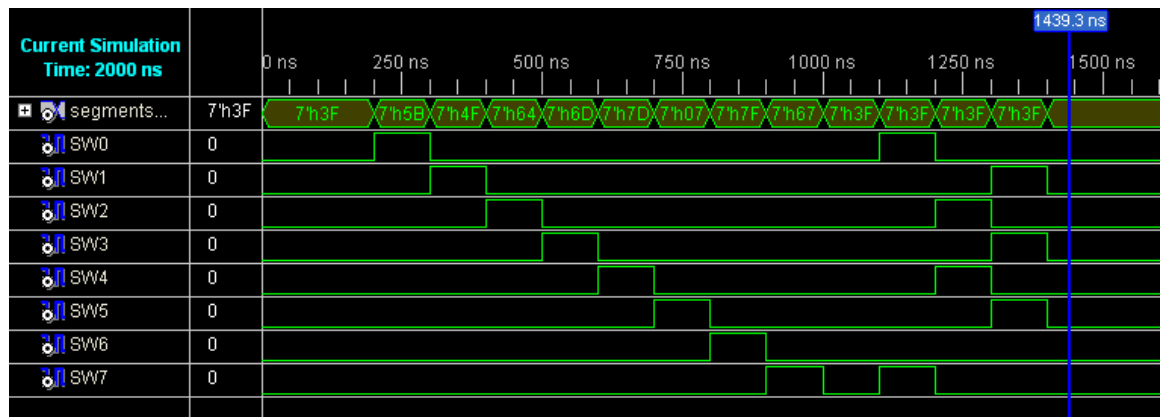
1  NET "SW0" LOC = "J11" | IOSTANDARD = LVCMOS33 ; #switch[0]
2  NET "SW1" LOC = "J12" | IOSTANDARD = LVCMOS33 ; #switch[1]
3  NET "SW2" LOC = "H16" | IOSTANDARD = LVCMOS33 ; #switch[2]
4  NET "SW3" LOC = "H13" | IOSTANDARD = LVCMOS33 ; #switch[3]
5  NET "SW4" LOC = "G12" | IOSTANDARD = LVCMOS33 ; #switch[4]
6  NET "SW5" LOC = "E14" | IOSTANDARD = LVCMOS33 ; #switch[5]
7  NET "SW6" LOC = "D16" | IOSTANDARD = LVCMOS33 ; #switch[6]
8  NET "SW7" LOC = "B16" | IOSTANDARD = LVCMOS33 ; #switch[7]
9  NET "segments[0]" LOC = "E3" | IOSTANDARD = LVCMOS33 ; #SegA
10 NET "segments[1]" LOC = "E1" | IOSTANDARD = LVCMOS33 ; #SegB
11 NET "segments[2]" LOC = "G5" | IOSTANDARD = LVCMOS33 ; #SegC
12 NET "segments[3]" LOC = "D1" | IOSTANDARD = LVCMOS33 ; #SegD
13 NET "segments[4]" LOC = "E4" | IOSTANDARD = LVCMOS33 ; #SegE
14 NET "segments[5]" LOC = "C1" | IOSTANDARD = LVCMOS33 ; #SegF
15 NET "segments[6]" LOC = "C2" | IOSTANDARD = LVCMOS33 ; #SegG
16

```

- Save and close the file, giving **behavioral_top.ucf** as the filename and *UCF(*.ucf)* as the *Save as type*:
- Add the ucf file to the project

2. Simulate the design using ISE

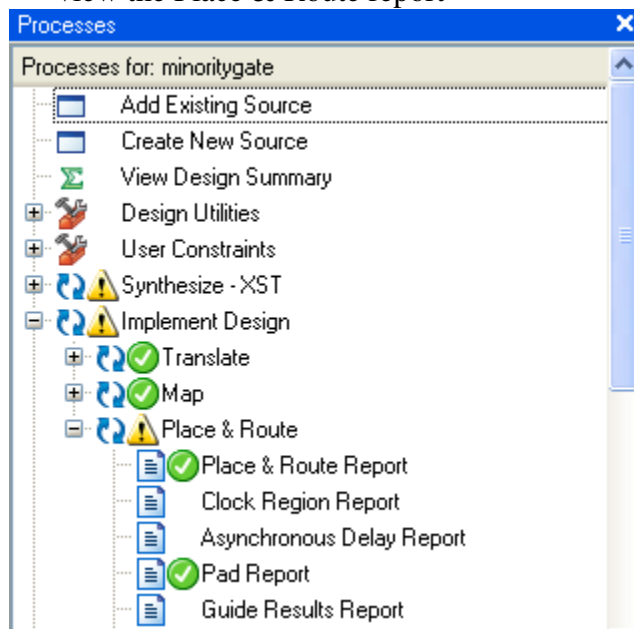
- Right-click on the **dataflow** entry in *Sources* window, right-click and select **Add Copy of Source...**
- Browse to C:\NI\Verilog_labs\resources\behavioral_lab and select **behavioral_top_tb.v**. Notice that the *Sources for* window changes to Behavioral Simulation from Implementation
- Select **behavioral_top_tb** in the *Sources* window, expand the **Xilinx ISE Simulator** process in *Processes* window, and double-click **Simulator Behavioral Model**
- The model will be compiled and the simulator will be run
- Simulation results will be displayed as shown below



- Analyze and understand the simulation results
- Close the simulator

3. Implement the design

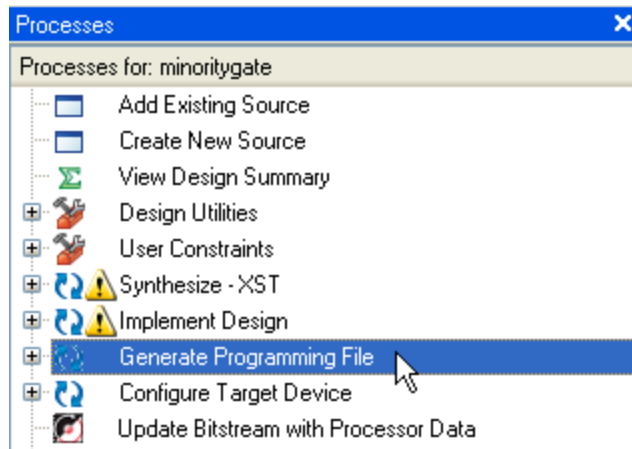
- Select **implementation** in *Sources for* window
- Select **behavioral_top** module in *Sources* window and double-click on **Implement Design** process in *Processes* window. This will go through Synthesis, and Implementation stages
- When the implementation is completed, expand Implement Design process to view the Place & Route report



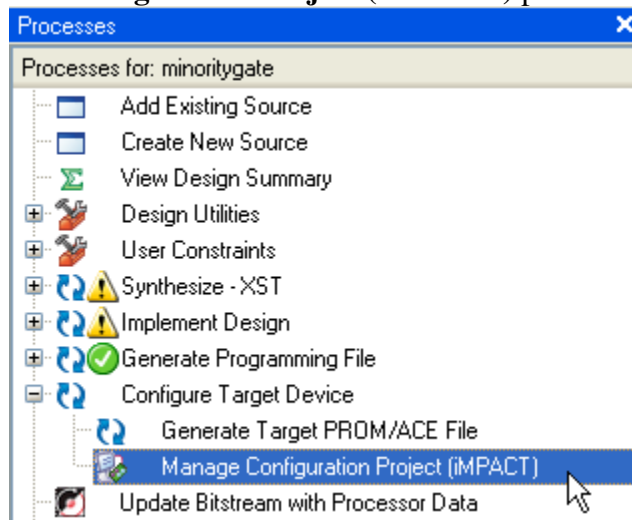
- Double-click on the Place & Route report to view the report. Look at the resource utilization and note that **13** slices are being used
- You can see similar information by clicking on Design Summary tab and looking at the various information

4. Verify the design in hardware

- Select **behavioral_top** in *Sources* window and double-click on **Generate Programming File** process to generate the bit file for the design



- Expand **Configure Target Device** process and double-click on **Manage Configuration Project (iMPACT)** process



- Connect the board with the USB-JTAG cable
- Power ON the board
- Click Finish to use the JTAG chain
- Select behavioral_top.bit file to be assigned to xc3s500e device and click **Open**
- Click **Bypass** button for *xcf04s* and then **OK** to use FPGA device programming
- Right-click on the FPGA and select **Program**
- This will program the FPGA and DONE light will lit on the board
- Once programmed successfully, verify the functionality by using SW0 thru SW7 and monitoring left 7-segment display output
 - SW0 thru SW7 are input, with SW0 being the least significant
 - To test the design, first set all switches to OFF position and observe that the 7-segment display shows 0. This is because none of the switch is ON. Next, try to turn on a switch and observe the 7-segment display shows the switch number + 2 value. If you turn ON another switch then the display will change to 0 as more than one switch is ON

- Now try various input combination and verify that the design is behaving correctly
- Once confirmed the functionality, power down the board and close ISE saving project changes

Conclusion:

In this lab exercise you learned how to model a combinatorial circuit using behavioral style of Verilog HDL. You learned two fundamental procedural statements supported in the behavioral style. You also learned what a sensitivity list in always statement is and how to model a combinatorial logic. You were able to simulate the design and then verify the complete design in hardware board.