Building a MUX-DEMUX Circuit Lab

Overview:
In this lab you will learn how to model a combinatorial circuit using mixed-modeling style of Verilog HDL.

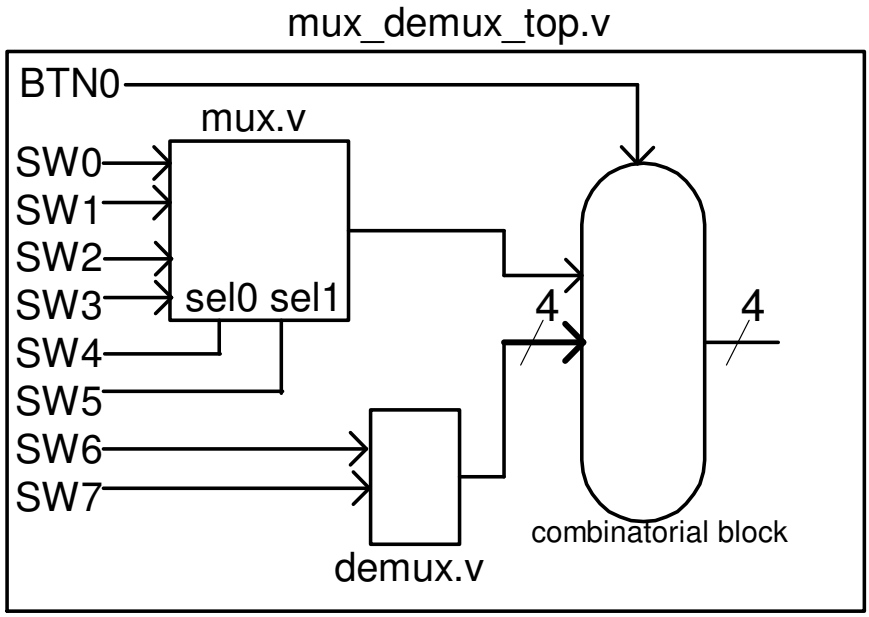
Outcome:
You will understand how to model a combinatorial circuit using various modeling styles available in Verilog HDL. You will learn how to create a model using ISE create project wizard. You will instantiate lower-level models to create a bigger model. You will use ISE simulator to simulate the 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.

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 multiplexer-demultiplexer circuit using gate-level, data-flow, and behavioral modeling styles. The multiplexer you will design will be 4-to-1 and the demultiplexer will be 2-to-4, requiring eight inputs which you will provide using switches. You will use BTN0 to select either output of the multiplexer or demultiplexer. When depressed you will output demultiplexer output otherwise multiplexer output.

Implementation:
The circuit to be designed consists of two functional blocks: mux and demux. They can be modeled using behavioral modeling style. The output selection can be implemented using continuous assignment statements. The hierarchical block diagram of the complete system is shown below.



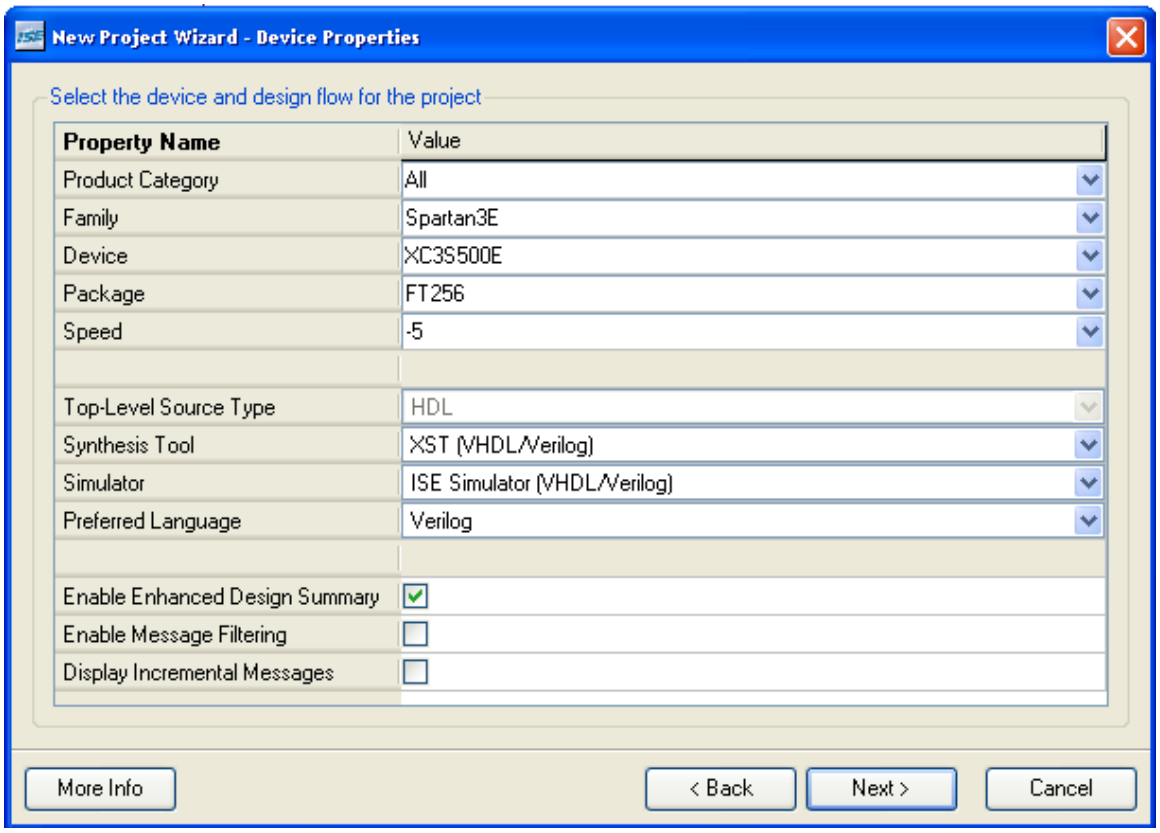
Procedure:

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

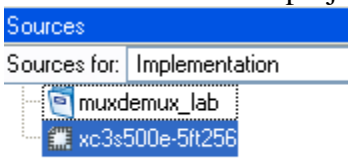
1. Create a ISE project

- 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 ***muxdemux_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**



- 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 *mux* function using behavioral modeling style

```
module mux (
   input [3:0] muxdatain,
   input [1:0] muxsel,
   output reg mux_out
   );

   always @(muxdatain or muxsel)
   case (muxsel)
         2'b00: mux_out = muxdatain[0];
         2'b01: mux_out = muxdatain[1];
         2'b10: mux_out = muxdatain[2];
         2'b11: mux_out = muxdatain[3];
         default : mux_out = 1'b0;
   endcase

endmodule

```

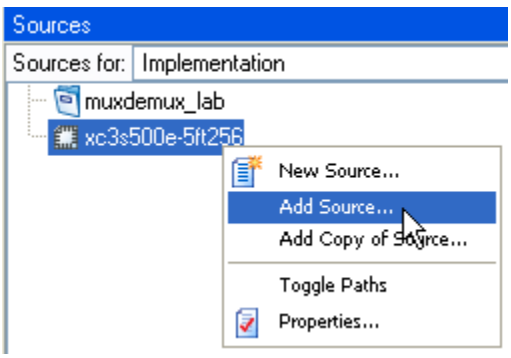
- Save the file as **mux.v** and notice the text changes into context driven (Verilog language sensitive) format
- Close the file
- Similarly, create **demux.v** file with the content as shown below

```
module demux (
   input [1:0] demuxdatain,
   output reg [3:0] demux_out
   );

   always @(demuxdatain)
   case (demuxdatain)
         2'b00: demux_out = 4'b0001;
         2'b01: demux_out = 4'b0010;
         2'b10: demux_out = 4'b0100;
         2'b11: demux_out = 4'b1000;
         default: demux_out = 4'b0000;
   endcase

endmodule

```

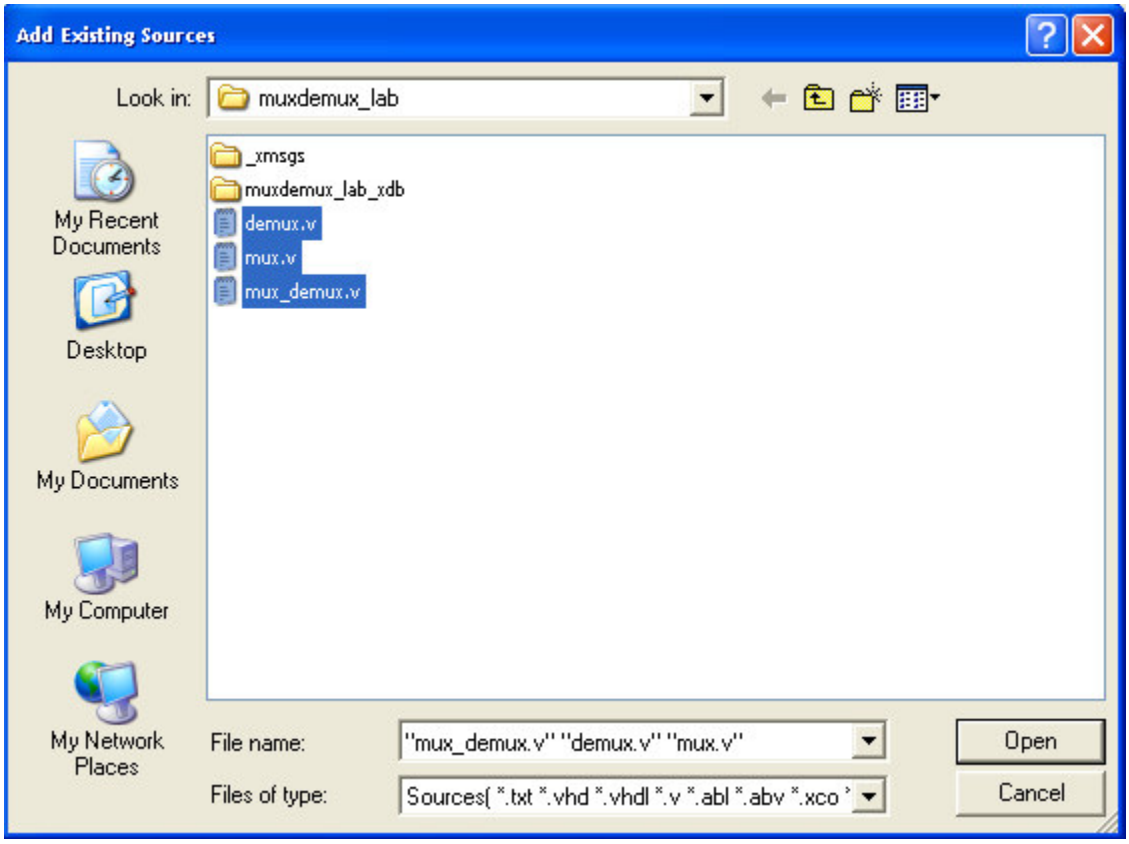
- Save and close the file
- Create a top-level model and enter the following code which instantiates *mux* and *demux* models and adds data-flow modeling statements to complete the design

```
 1  module mux_demux_top (
 2     input [3:0] muxdatain,
 3     input [1:0] muxsel,
 4     input [1:0] demuxdatain,
 5     input funcsel,
 6     output [3:0] muxdemuxout
 7     );
 8
 9  wire mux_out;
10  wire [3:0] demux_out;
11
12  assign muxdemuxout[0] = funcsel ? demux_out[0] : mux_out;
13  assign muxdemuxout[1] = funcsel ? demux_out[1] : mux_out;
14  assign muxdemuxout[2] = funcsel ? demux_out[2] : mux_out;
15  assign muxdemuxout[3] = funcsel ? demux_out[3] : mux_out;
16
17  mux M1 (
18     .muxdatain(muxdatain),
19     .muxsel(muxsel),
20     .mux_out(mux_out)
21     );
22
23  demux D1 (
24     .demuxdatain(demuxdatain),
25     .demux_out(demux_out)
26     );
27
28  endmodule
29
```

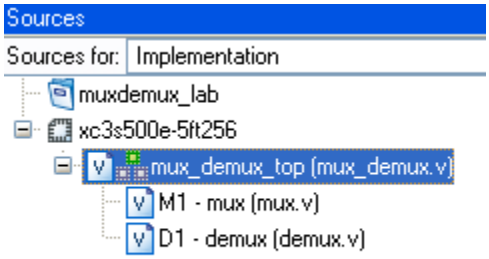
- Save the model as **mux_demux.v** and close the file
- Note that even though the files have been created, they are not automatically added to the project as they were created as blank text files
- To add the files, select the chip in the *Sources* window and right-click and then select **Add Source…**



- Select all three verilog files (mux.v, demux.v, and mux_demux.v) and click **Open**



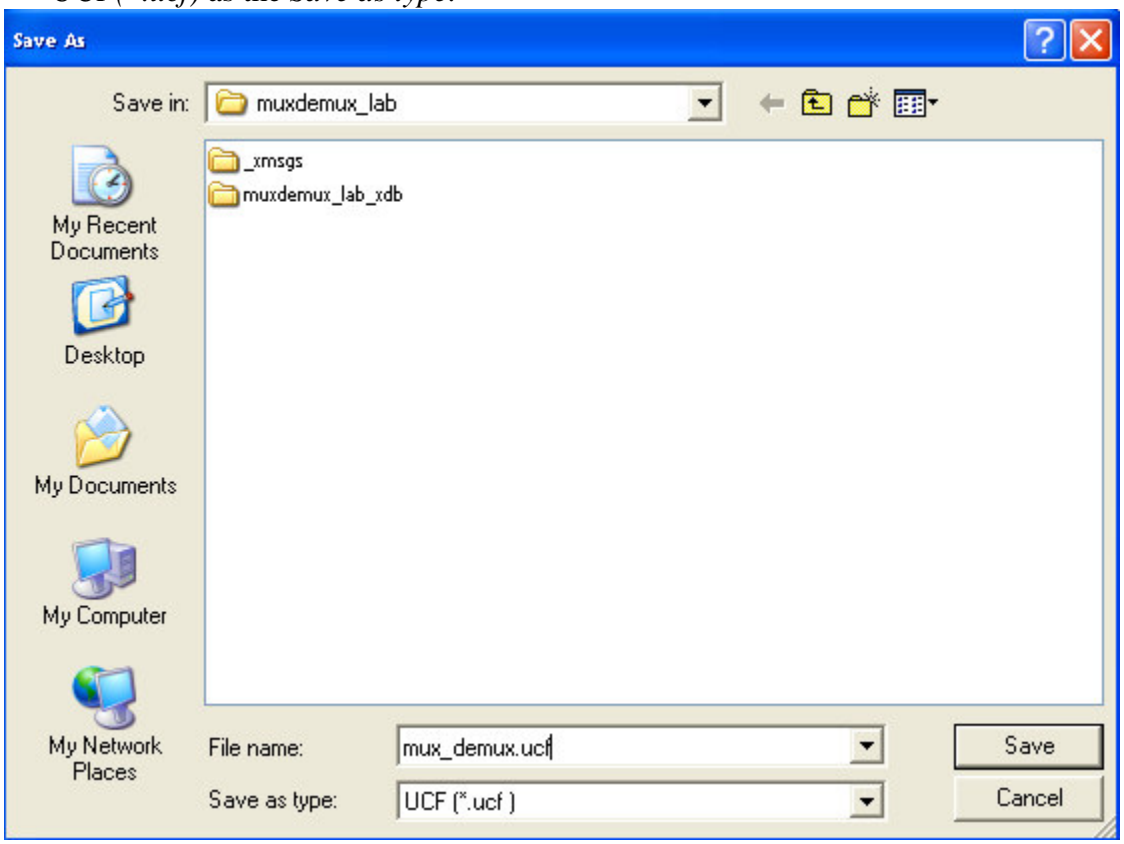
- Click **OK** to add the three files
- Expand the *mux_demux* entry in *Sources* window and observe the lower-level modules



- Create a new blank file and enter the following location constraints

```
1   NET "muxdatain[0]"  LOC = "J11" | IOSTANDARD = LVCMOS33 ; #switch[0]
2   NET "muxdatain[1]"  LOC = "J12" | IOSTANDARD = LVCMOS33 ; #switch[1]
3   NET "muxdatain[2]"  LOC = "H16" | IOSTANDARD = LVCMOS33 ; #switch[2]
4   NET "muxdatain[3]"  LOC = "H13" | IOSTANDARD = LVCMOS33 ; #switch[3]
5   NET "muxsel[0]"  LOC = "G12" | IOSTANDARD = LVCMOS33 ; #switch[4]
6   NET "muxsel[1]"  LOC = "E14" | IOSTANDARD = LVCMOS33 ; #switch[5]
7   NET "demuxdatain[0]"  LOC = "D16" | IOSTANDARD = LVCMOS33 ; #switch[6]
8   NET "demuxdatain[1]"  LOC = "B16" | IOSTANDARD = LVCMOS33 ; #switch[7]
9   NET "funcsel"  LOC = "C13" | IOSTANDARD = LVCMOS33 ;  # BTN[0]
10  NET "muxdemuxout[0]"  LOC = "C11" | IOSTANDARD = LVCMOS33 ; #LED[0]
11  NET "muxdemuxout[1]"  LOC = "D11" | IOSTANDARD = LVCMOS33 ; #LED[1]
12  NET "muxdemuxout[2]"  LOC = "B11" | IOSTANDARD = LVCMOS33 ; #LED[2]
13  NET "muxdemuxout[3]"  LOC = "A12" | IOSTANDARD = LVCMOS33 ; #LED[3]
14
```

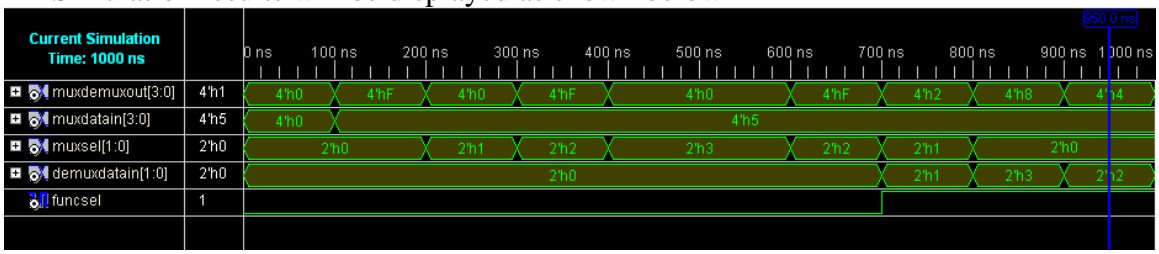
- Save and close the file, giving **mux_demux.ucf** as the filename and *UCF(*.ucf)* as the *Save as type:*



- Add the ucf file to the project

2. ***Simulate the design using ISIM***
   - Right-click on the **mux_demux** entry in *Sources* window, right-click and select **Add Copy of Source…**
   - Browse to C:\NI\Verilog_labs\resouces\muxdemux_lab and select **mux_demux_top_tb.v**. Notice that the Sources for window changes to Behavioral Simulation from Implementation

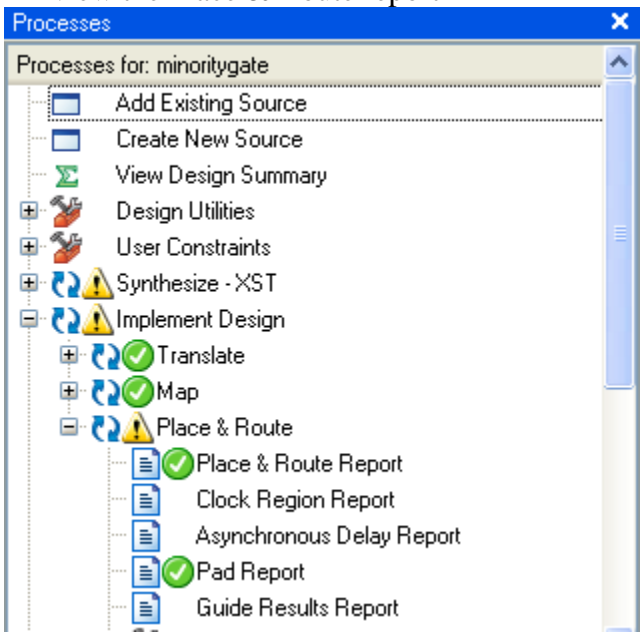
- Select **mux_demux_top_tb** in the *Sources* window, expand the **Xilinx ISE Simulator** process in *Processes* window, and double-click **Simulator Behavioral Model**
- The model will be compiled and the simulator will be run
- Simulation results will be displayed as shown below



- Analyze and understand the simulation results
- Close the simulator

3. *Implement the design*

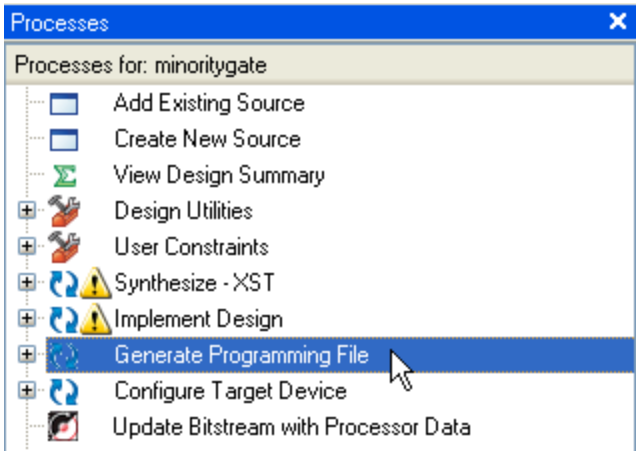
- Select **implementation** in *Sources for* window
- Select **mux_demux_top** module in *Sources* window and double-click on **Implement Design** process in *Processes* window. This will go through Synthesis, and Implementation stages
- When the implementation is completed, expand Implement Design process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



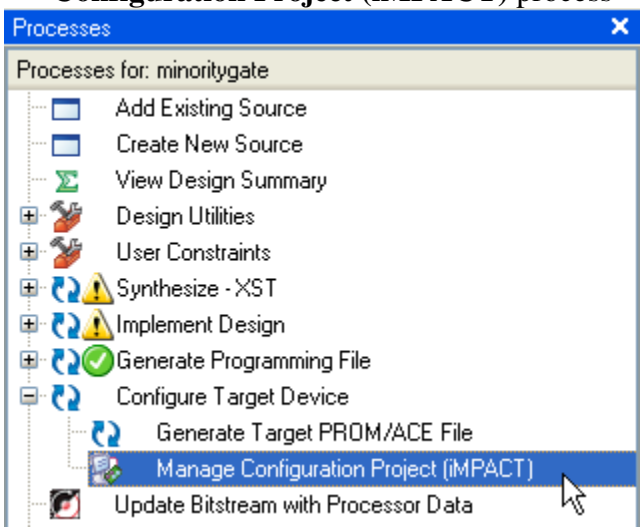
- Double-click on the Place & Route report to view the report. Look at the resource utilization and note that **3** slices are being used
- You can see similar information by clicking on Design Summary tab and looking at the various information

4. *Verify the design in hardware*

- Select **mux_demux_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 *mux_demux.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, push-button 0, and monitoring LD0 thru LD3 output
  - SW0 thru SW3 are mux datain, SW4 and SW5 are mux channel selector, SW6 and SW7 are demux data in
  - To test the design, first set all switches to OFF position. Since BTN0 is de-pressed it is selecting multiplexer functionality. Switch SW0 to 1 and notice that all LEDs (LD3:LD0) are turned ON as mux channel selector (SW5:SW4=00) is selecting channel 0 and by design it is sent to all LEDs
  - Now change settings of mux channel selector to other than 00 and notice that all LEDs are turned OFF. Depending on channel selection now turn

ON the corresponding channel switch and observe that all LEDs are turned ON
    - Now press BTN0 to select de-multiplexing functionality. When pressed observe that LD0 alone is turned ON as SW7:SW6 are 00
    - Change SW7:SW6 settings while BTN0 is pressed and observe the corresponding LED turning ON
- 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 mixed-modeling styles of Verilog HDL. You were able to simulate the design and then verify the complete design in hardware board.