

**CSEE 4280 – ADVANCED LOGIC DESIGN  
SPRING 2017**

**LAB 1 – Complete by Friday Jan 5 2018.**

**This is a practice lab, no lab report is due for this lab. Each of you (not as a team) needs to finish everything during the class on your own.**

ISE Tutorial

Contents

Part 0 : Preparations - Lab Schedule

Part 1 : Introduction to ISE

Getting Started : Using the **Project Manager**

Part 2 : Learning how to use **ECS**: Building the ZERO component

Part 3 : Learning how to use **iSim**: Simulating the ZERO component.

Part 4 : Hierarchical Design: Building the **MUX8** component

**0. PREPARATIONS**

**Lab Schedule**

Xilinx can be downloaded for free and installed on your computers and is also available in the Room 209.

**Your working directory**

Your working directory will be the path for you to store all your work when you are working on your project in the lab. Make sure that all your work, finished or unfinished, is copied onto your portable storage before you delete them from the lab machine's hard drive. **It is strongly recommended to make multiple backups for your work throughout the semester.**

**1. INTRODUCTION**

**Xilinx ISE 14.x**, which you will be using for this course is a very large piece of software with very many capabilities. In CSEE 4280, we will use just a new feature of these capabilities, different from what you learned in CSEE 4270.

**ISE** has a modular structure. It consists of several modules each with a specific function. One of the modules manages all the other modules by coordinating their activities. This is the **Project Navigator**. In this class we shall be using four other modules: **Engineering Capture System (ECS)**, **HDL Bench** and **iSim**.

### Project Navigator

**Project Navigator** is the user interface that helps you manage the entire design process including design entry, simulation, synthesis, implementation and finally download the configuration onto your FPGA (Field Programmable Gate Array) chip. Many files are generated by **ISE** and it is important that they are all kept in their native directory. Moving them around manually will cause **Project Manager** to lose track of their location. **Project Manager** will consequently gripe about that.

### **Design Entry**

#### Engineering Capture System (ECS)

The **Engineering Capture System (ECS)** is a graphical user interface (GUI) that allows you to create, view, and edit schematic drawings and symbols. You can use ECS to create a top-level schematic and use it also to define the lower levels of the design. You can then translate the schematics created by ECS for simulation and synthesis. **Note: This is a different approach from the Verilog entries you used in CSEE 4270. The reason why we choose to use ECS over Verilog is because of the complexity of the toy processor.**

### **Simulation**

#### HDL Benchner

**HDL Benchner** automates verification of schematics created within ISE. Design sources are imported, a waveform is created, and stimulus is specified by filling in the WaveTable spreadsheet-like cells. Outputs may be auto-simulated via a command from ISE. HDL Benchner constrains the test run to a specific sequence of events, initial conditions, and user determined results. With HDL Benchner you can quickly validate if your design functions as intended.

#### iSim

**iSim** is actually third-party software (it is made by ModelTechnologies) which has been integrated into the **ISE** design flow. It is a very powerful tool for simulating your designs.

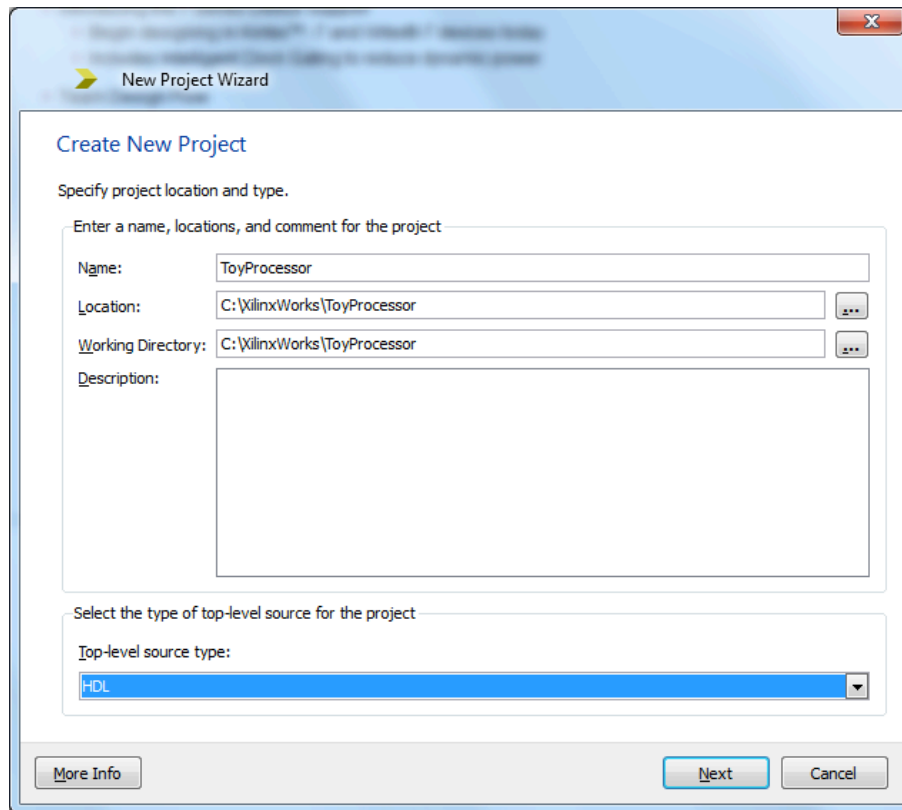
So now that you are acquainted with a brief overview of ISE, let's get started.

**We are going to build several components: a ZERO, a MUX, and an adder/subtractor Arithmetic Logic Unit (ALU). We will start with the ZERO component.**

We will now familiarize ourselves with ISE.

**Getting Started:** Using the **Project Manager**

1. **Start → Programs → Xilinx ISE 14.x → Project Manager.**
2. From the **File** menu select **New Project**. A **New Project** dialog box pops up. Make sure the **Project Location** is the directory where you want to put your project files. Make sure it is set to your storage device. This project is going to be called **ToyProcessor**. Click **Next**.



**Create New Project**

Specify project location and type.

Enter a name, locations, and comment for the project

Name: ToyProcessor

Location: C:\XilinxWorks\ToyProcessor

Working Directory: C:\XilinxWorks\ToyProcessor

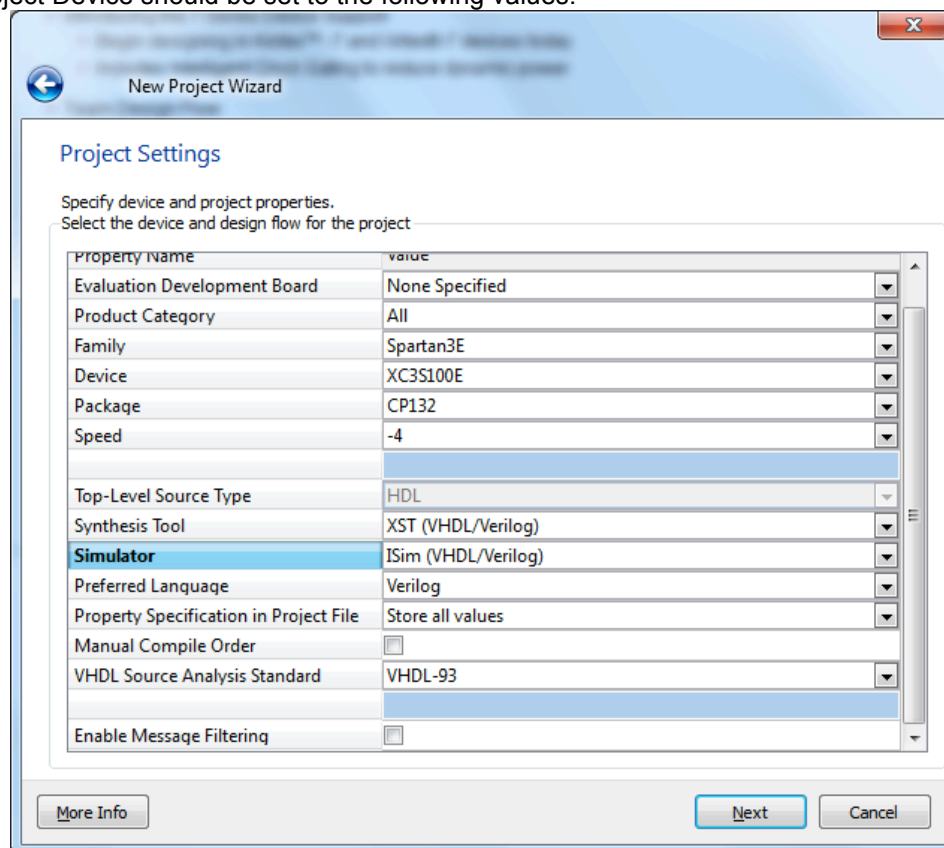
Description:

Select the type of top-level source for the project

Top-level source type: HDL

More Info Next Cancel

The Project Device should be set to the following values:



**Project Settings**

Specify device and project properties.

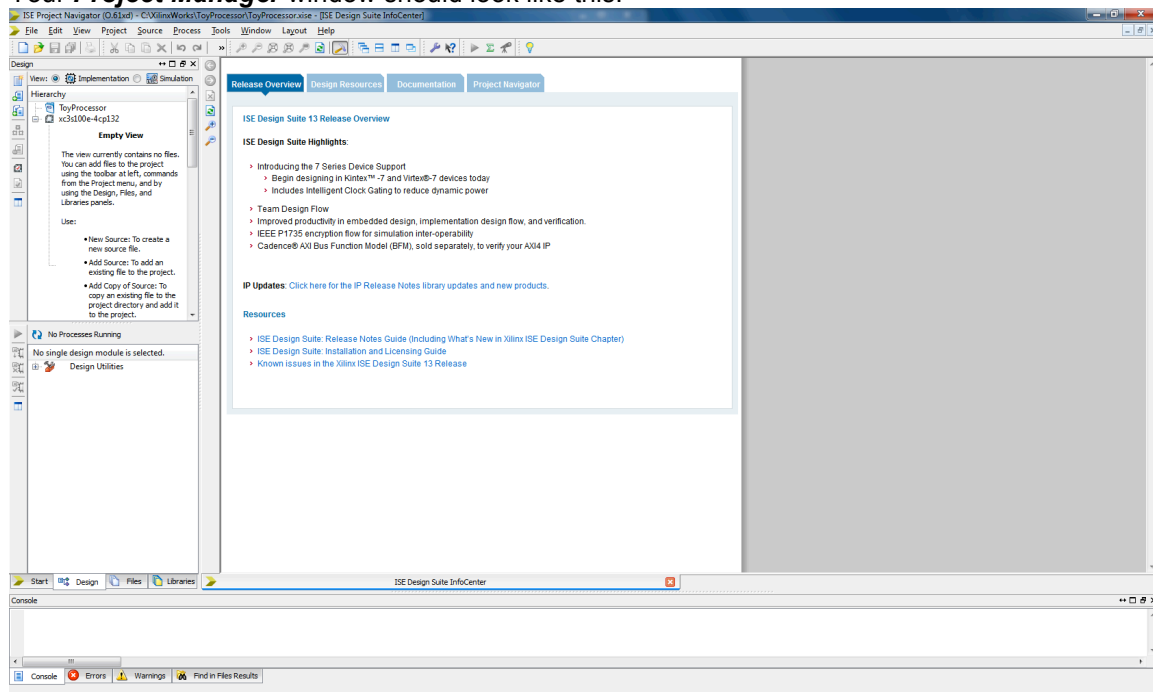
Select the device and design flow for the project

Property name	value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S100E
Package	CP132
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
<b>Simulator</b>	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	<input type="checkbox"/>

More Info Next Cancel

*New project dialog box*

Click **Next**, **Next**, **Next** and **Finish** to create the project.  
Your **Project Manager** window should look like this:



*Project Manager showing a new project.*

## Part 2

Learning how to use **ECS**: Building the ZERO component

In the first part of this lab, you will build the zero component. This component takes in an 8-bit number and generates the signal ZERO. **This signal is 1 if the 8-bit input is all 0s, otherwise it is 0.** So, it is really just an 8-input NOR gate.

## Design

ISE uses **ECS** for drawing schematics. To start up **ECS**, go to File->New. Alternatively right-click on the project name, ToyProcessor, and select New Source.

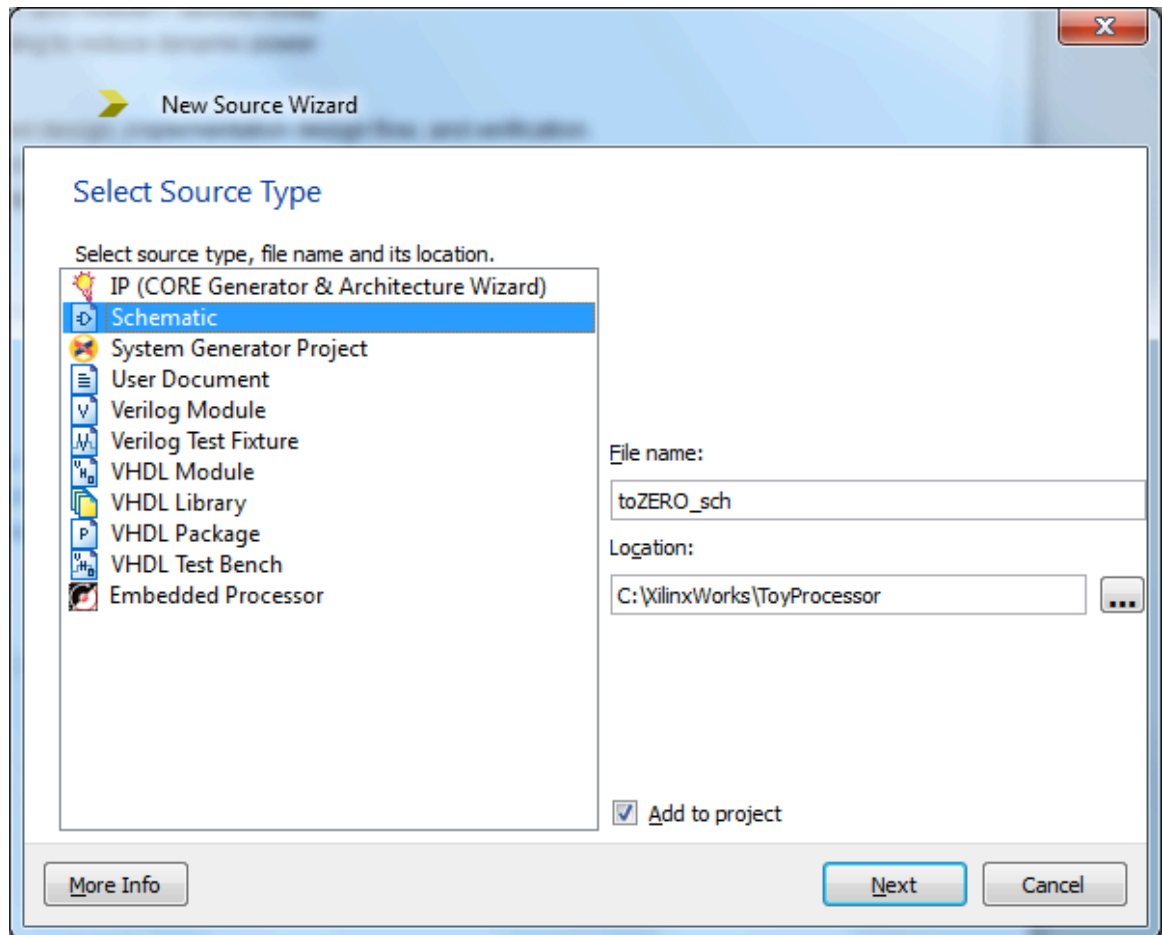
## ECS Hints

The **ECS** schematic capture program is designed around the user selecting the action they wish to perform followed by the object the action is to be performed on. In general most Windows applications currently operate by selecting the object and then the action to be performed on that object. Understanding this fundamental philosophy of operation makes learning ECS a much more enjoyable experience.

## Project Manager : Adding a New Source

1. From the **Project** menu select **New Source** → **Schematic** and give it the name **toZERO\_sch.sch**.


**Note:** Make sure the **Add to Project** checkbox is checked. This adds **toZERO\_sch** to the list of sources in **Project Manager**.

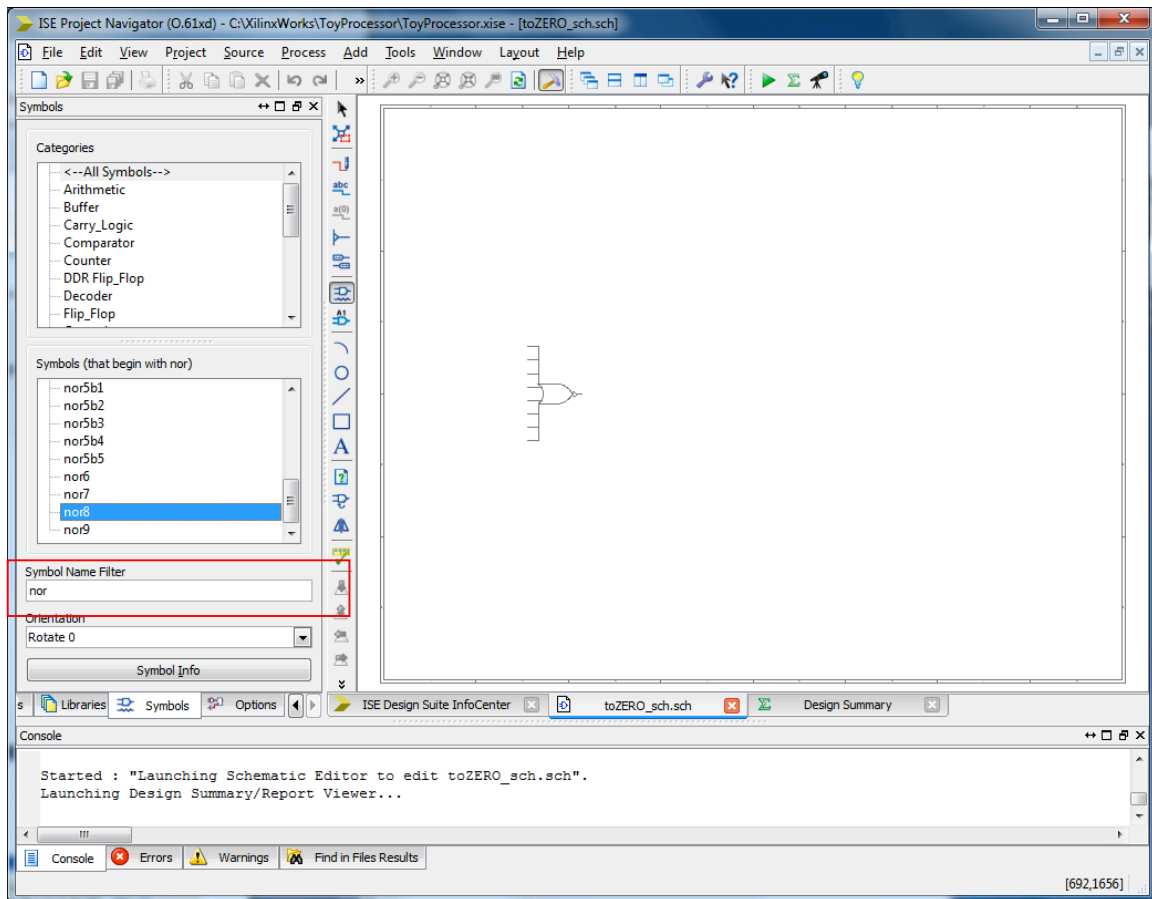


*New Source: Schematic : toZERO\_sch.sch*

2. Click **Next** then **Finish**. The **ECS Schematic Editor** window will now open

#### **ECS: Adding Symbols to the schematic**


1. In the open **ECS** window, click the **Add Symbol** tool. 
2. Type **nor** in the **Symbol Name Filter** to display symbols that begin with "nor". We want **nor8**.



**ECS schematic with the *New Symbol* tool selected**

3. Click the **nor8** gate and place it on the schematic drawing sheet using the mouse. Press **Esc** to exit from the symbol placement mode.

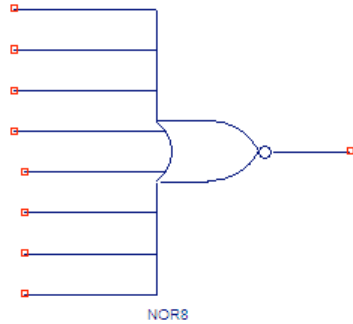
### ECS : Adding wires

4. Select the **Add Wire**  tool from the **Drawing Toolbar**. Add a hanging wire extending outwards from each of the ports (or pins) of the **nor8** symbol.

**Note:** To add a hanging wire click on the symbol pin to start the wire, once at each vertex and then double-click at the location you want the wire to terminate.


**Note:** Click once on the symbol pin, once at each vertex and once on the destination pin to add a wire between two pins. ECS will let the user know that a net can be attached to a port by highlighting it with a red square.

Your schematic should look like this:



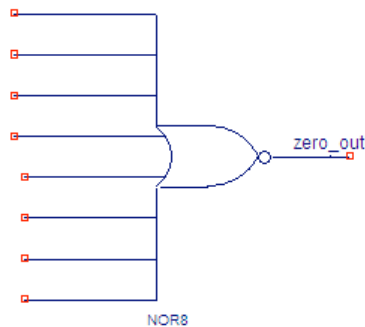
*nor8 gate with ports extended by nets (wires)*

### ECS: Adding Net Names

5. Select the **Add Net Names** tool  from the **Drawing Toolbar** ("net" is synonym for "wire"). Type **zero\_out** in the **Name** textbox in the options tab of the processes window (on the left) and then place the net name on the end of the **nor8** output net by clicking it. (Net names can only be connected to the hanging wire)

**Note:** To add net names to wires that will be connected to your FPGA I/Os, place the net name on the end of the hanging wire. Press **Esc** to exit net naming mode.

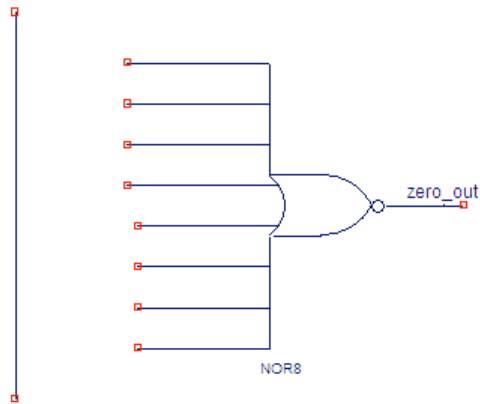
6. Your schematic should look similar to the following figure:




*Naming the output net*

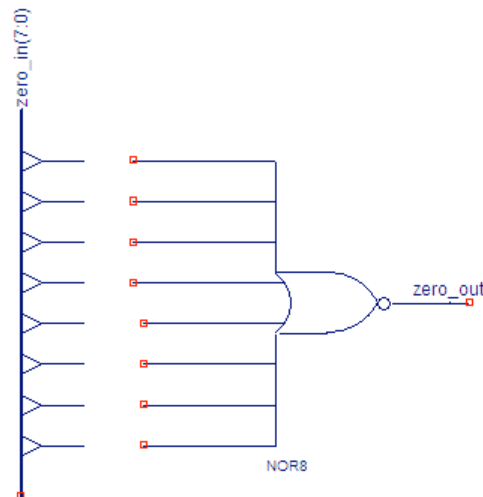
### ECS : Adding a bus and bus taps

7. Draw a vertical net to the left of the net extensions of the **nor8** input. It should look something like this. The vertical net should be some distance away from the terminal points of the input nets of the **nor8** gate.



*Vertical net to be designated as bus.  
Note that it extends beyond the top and bottom input nets.*

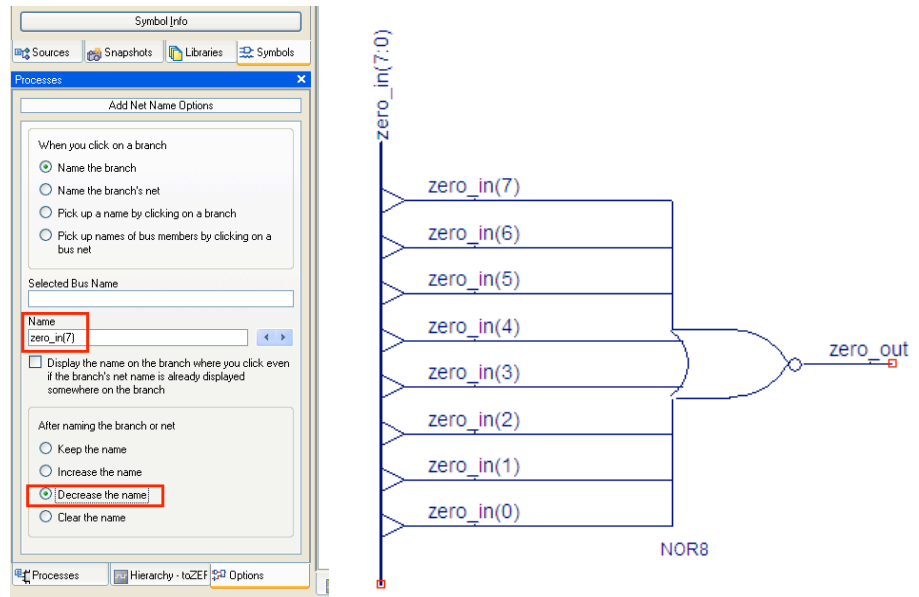
8. Name the vertical net **zero\_in(7:0)**. The net name indicates that this is a bus with 8 ports (from 0 to 7). Bus nets in **ECS** are thicker than single nets.
9. Click the bus tap button  on the toolbar.
10. Place bus taps on the bus (you can change the alignment by selecting the correct orientation from the **Add Bustap Options**) so that they roughly align with the **nor8** input nets. The following shows how your schematic should look like:



*Bus taps connected to bus and aligned with inputs*


11. Use nets to connect the bus taps to corresponding inputs for simplicity and clarity of design.
12. Add net names to each of the connecting nets. Start with net name **zero\_in(7)** and select **Decrease the name** in the **After naming the branch** groupbox. Alternatively you can start from **zero\_in(0)** and **Increase the name**. Your schematic should look like this:

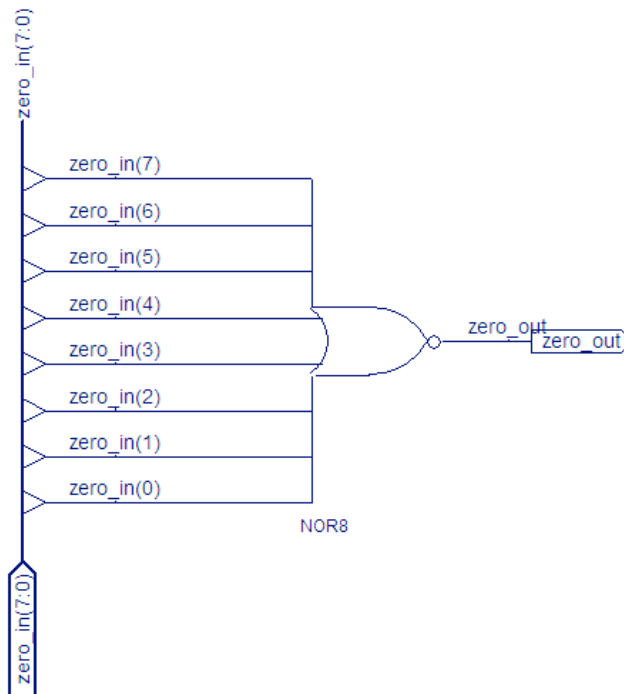





Connected bus taps with correctly assigned net names.

### ECS : Adding I/O Markers

13. Select the **Add I/O Marker**  tool from the **Drawing Toolbar**.
14. With the **Input** type selected, click and drag your mouse around (as if you were drawing a square or circle) the (lower) end of the bus **zero\_in(7:0)**. Repeat for the output **zero\_out** but select **Output** type. Your completed schematic should look like the following figure:



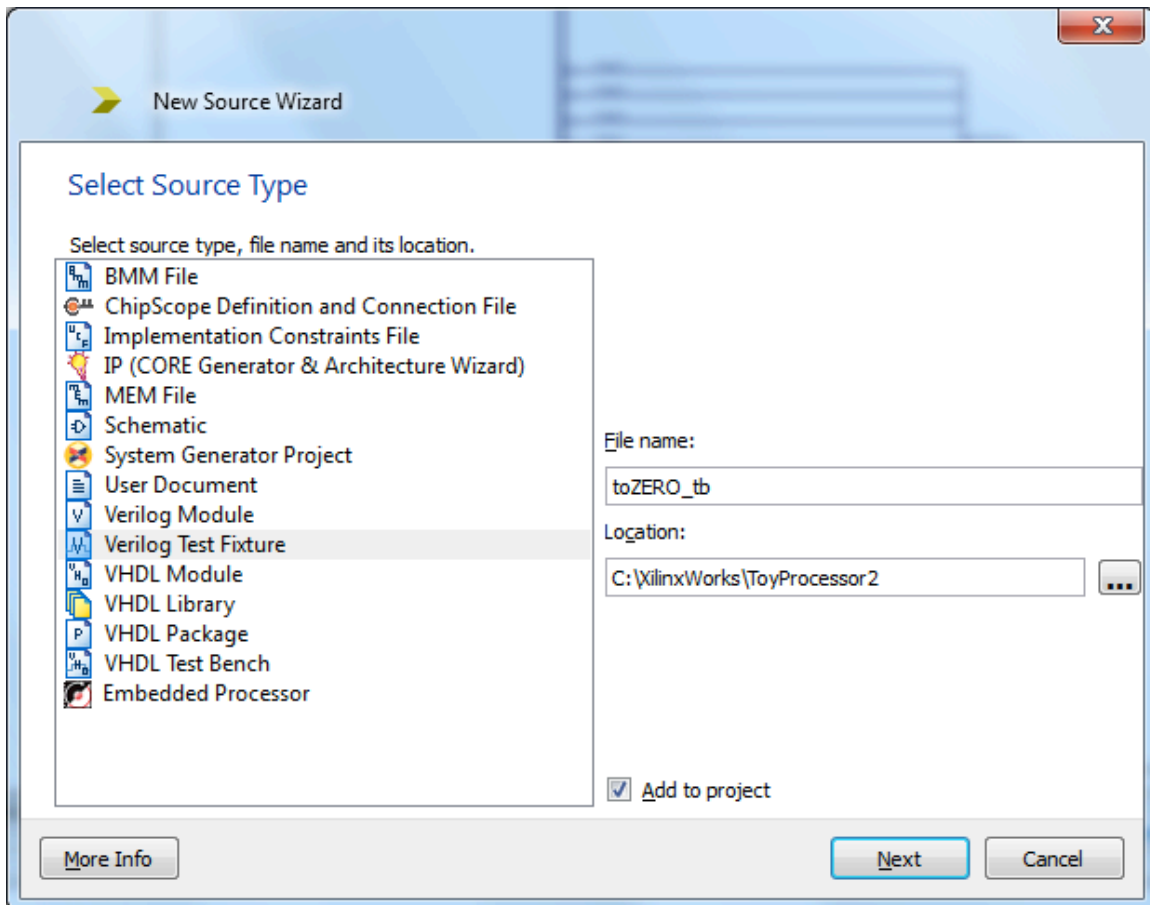
Adding I/O markers

15. Click the check button  on the drawing toolbar. The **Transcript** dialog box at the bottom of the program should report no errors.
16. Save the design and exit the schematic editor. The entire design can now be simulated.

**General Note:** You can rotate the symbols by using the **Orientation** drop-down menu on the **Symbols** tab. Remember to extend ports with wires before attempting to add **I/O** markers to avoid frustration.

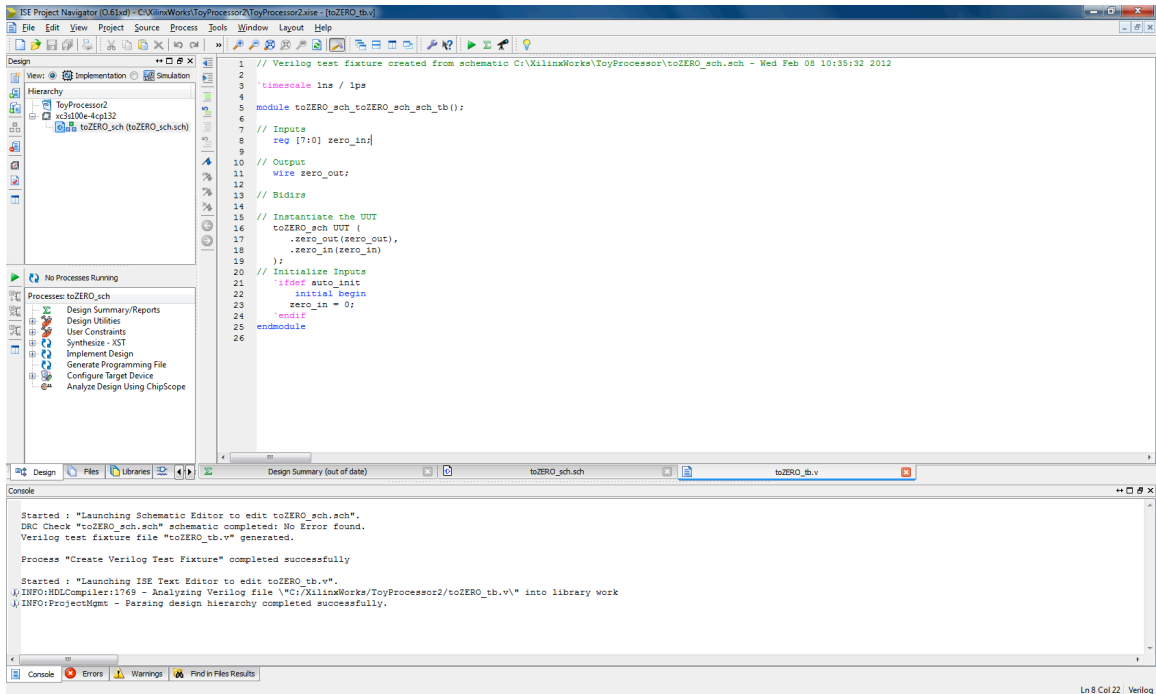
### Part 3

1. First, we create a Verilog test fixture from **Project Navigator**. Highlight **toZERO\_sch.sch** in the **Sources in Project** window.
2. Right-click on the project (ToyProcessor) and select **New Source**. In the **New** dialog box, select **Verilog Test Fixture** as the source type. Name it **toZERO\_tb**



Click **Next** ->**Next** ->**Finish**

The default verilog test file will show up automatically. (shown below)



Default Verilog test code

The default test file shows the components in the designed module. We need to modify the default code to create the test file. The modified code shows below:

**`timescale 1ns/1ps**

```

module toZERO_tbw_tb_0;
    reg [7:0] zero_in = 8'b00000000;           // Initialize the input ports to 0;
    wire zero_out;

```

```

toZERO_sch UUT (
    .zero_in(zero_in),
    .zero_out(zero_out));

```

**initial begin**

```

    // ----- Current Time: 200ns
    #200;           // pause 200 ns
    zero_in = 8'b00000001; //change the input ports to 00000001
    // -----
    // ----- Current Time: 300ns
    #100;
    zero_in = 8'b00000010;
    // -----
    // ----- Current Time: 400ns
    #100;
    zero_in = 8'b00000011;
    // -----
    // ----- Current Time: 500ns
    #100;

```

```

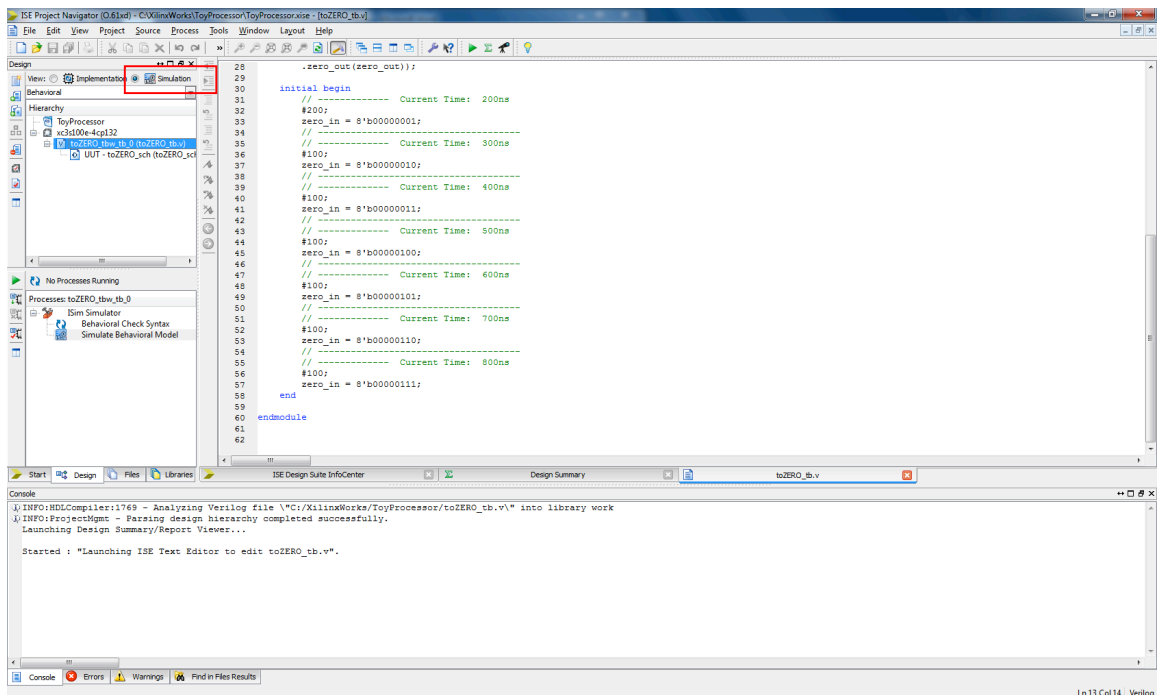
zero_in = 8'b00000100;
// -----
// ----- Current Time: 600ns
#100;
zero_in = 8'b00000101;
// -----
// ----- Current Time: 700ns
#100;
zero_in = 8'b00000110;
// -----
// ----- Current Time: 800ns
#100;
zero_in = 8'b00000111;
end

endmodule

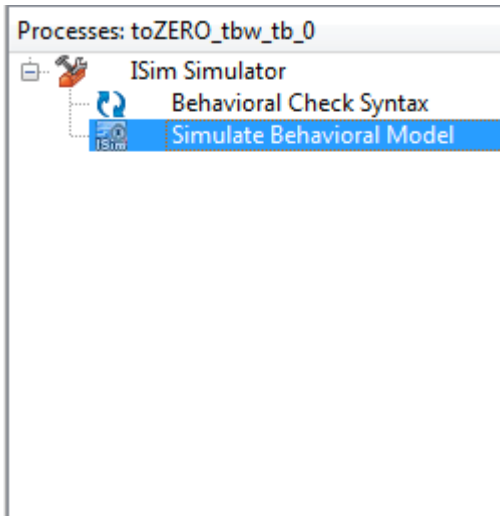
```

### *iSim* : Running the simulation

1. Switch to the simulation tab for test



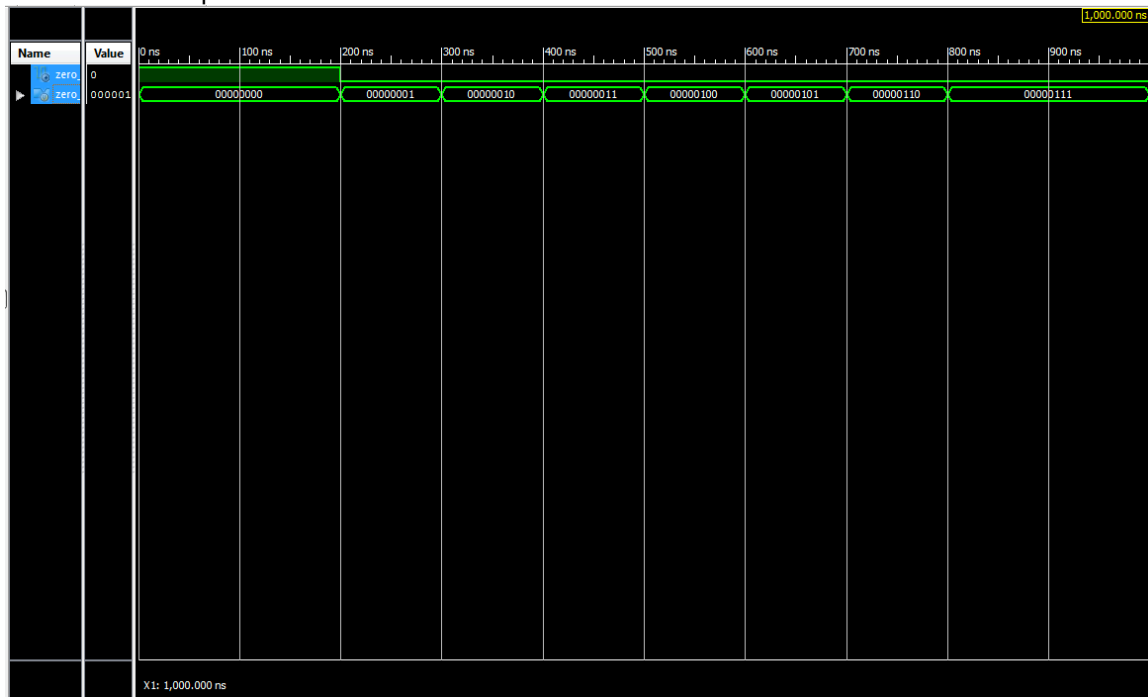
2. In **Project Manager** highlight toZERO\_tb.v.
3. Expand **iSim Simulator** and double-click **Simulate Behavioral Model**. This will run the *iSim* simulator process.



**Note:** *iSim* opens a multiplicity of windows on successful completion of a simulation process. Each window describes the test signals. You may examine them for more information about the signals.

What is the expected output?

4. Your output should look like this:



**wave**—simulation result. **Zero\_out** is high when **zero\_in** is zero. Otherwise it is low.

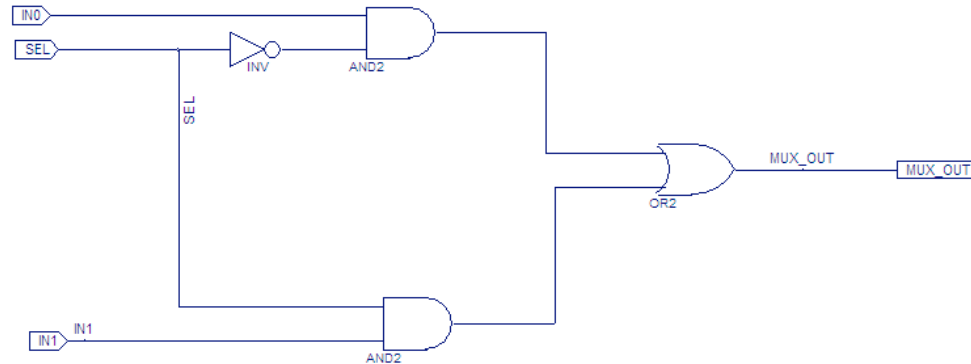
CONGRATULATIONS! You have finished your very first project!

#### Part 4

Hierarchical Design: Building the MUX8 component

Next we are going to build a single bit 2-to-1 multiplexer (or mux) from which we will build an 8-bit 2-to-1 mux. This is called hierarchical design.

1. Add a **New Source** to the existing project in **Project Manager** of type **Schematic**. Call it **mux.sch**.
2. In **ECS**, draw the following schematic:



1-bit multiplexer. Signals: IN0, IN1, SEL and MUX\_OUT

3. Save the schematic. Create a test bench for the schematic called **mux\_tb**
4. Modify the default test file to the following:

```
`timescale 1ns/1ps
```

```
module mux_tbw_tb_0;
  reg IN0 = 1'b0;
  reg IN1 = 1'b0;
  reg SEL = 1'b0;
  wire MUX_OUT;
```

**//Initialize inputs**

```
  mux UUT (
    .IN0(IN0),
    .IN1(IN1),
    .SEL(SEL),
    .MUX_OUT(MUX_OUT));
```

```
  initial begin
```

```
    // ----- Current Time: 500ns
    #500;
```

**// First signal starts after 500ns, where IN0 becomes**

**1**

```
    IN0 = 1'b1;
    // -----
    // ----- Current Time: 600ns
    #100;
    IN1 = 1'b1;
    // -----
    // ----- Current Time: 700ns
    #100;
    IN0 = 1'b0;
    // -----
    // ----- Current Time: 800ns
    #100;
    IN0 = 1'b1;
    // -----
```

```

// ----- Current Time: 1000ns
#200;
IN0 = 1'b0;
// -----
// ----- Current Time: 1100ns
#100;
IN0 = 1'b1;
// -----
// ----- Current Time: 1300ns
#200;
IN0 = 1'b0;
SEL = 1'b1;
// -----
// ----- Current Time: 1400ns
#100;
IN0 = 1'b1;
// -----
// ----- Current Time: 1500ns
#100;
SEL = 1'b0;
// -----
// ----- Current Time: 1600ns
#100;
IN0 = 1'b0;
IN1 = 1'b0;
// -----
// ----- Current Time: 1700ns
#100;
IN0 = 1'b1;
SEL = 1'b1;
// -----
// ----- Current Time: 1900ns
#200;
IN0 = 1'b0;
SEL = 1'b0;
// -----
// ----- Current Time: 2000ns
#100;
IN0 = 1'b1;
// -----
// ----- Current Time: 2100ns
#100;
SEL = 1'b1;
// -----
// ----- Current Time: 2200ns
#100;
IN0 = 1'b0;
IN1 = 1'b1;
// -----
// ----- Current Time: 2300ns
#100;
IN0 = 1'b1;
SEL = 1'b0;
// -----
// ----- Current Time: 2400ns
#100;

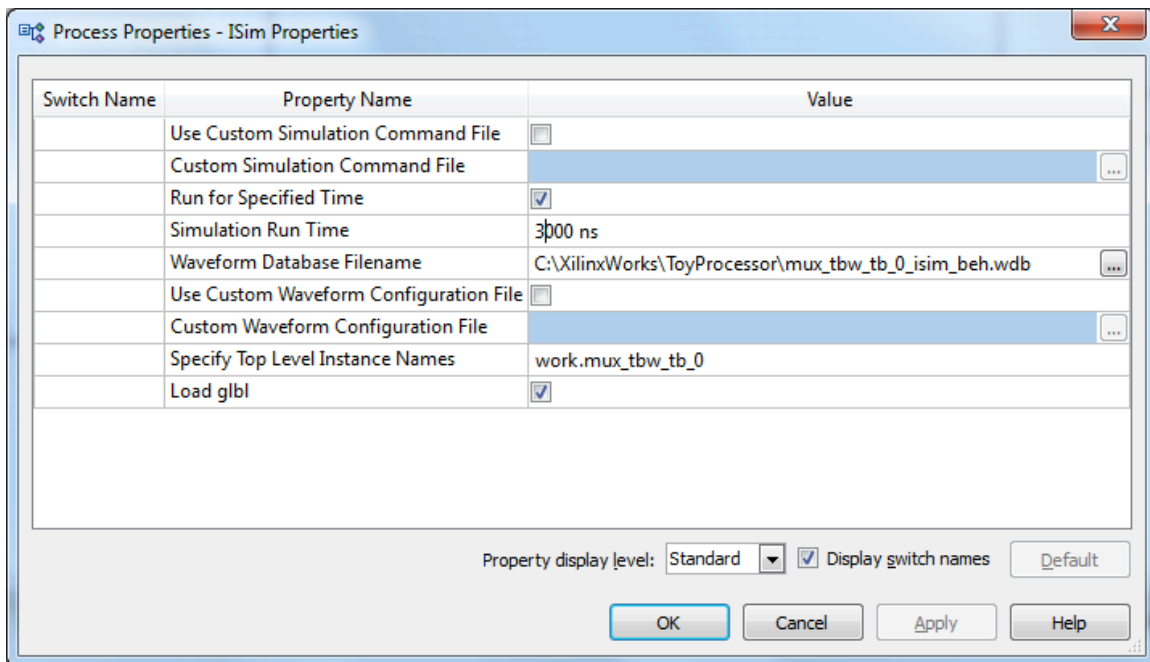
```

```

SEL = 1'b1;
// -----
// ----- Current Time: 2500ns
#100;
IN0 = 1'b0;
// -----
// ----- Current Time: 2600ns
#100;
IN0 = 1'b1;
IN1 = 1'b0;
// -----
// ----- Current Time: 2700ns
#100;
SEL = 1'b0;
// -----
// ----- Current Time: 2800ns
#100;
IN0 = 1'b0;
// -----
// ----- Current Time: 2900ns
#100;
SEL = 1'b1;
end
endmodule

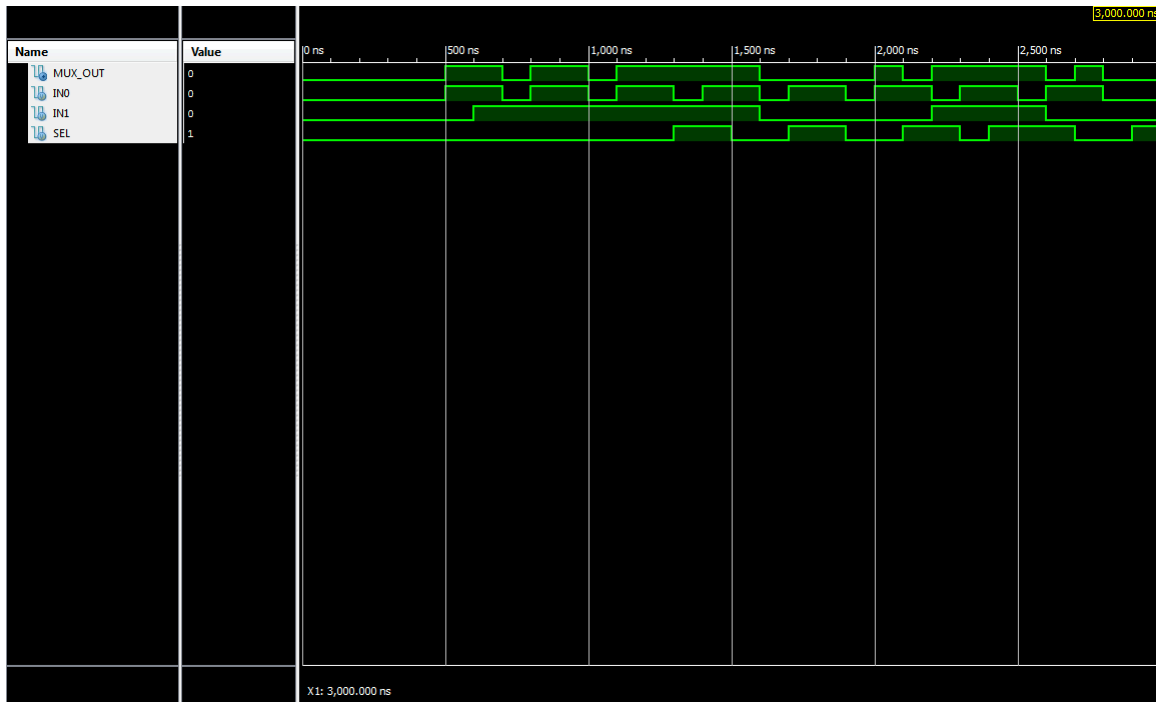
```

- Adjust the simulation run time by right clicking the “Simulation Behavior Model” -> “ISim Properties”, change it from 1000ns to 3000ns

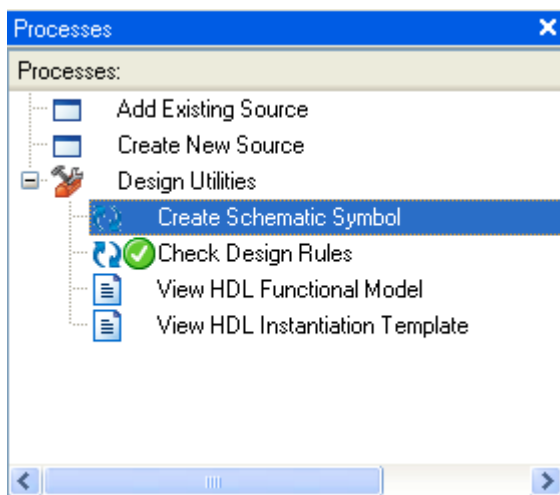




6. Make sure that your simulation verifies that if SEL is high (i.e. SEL = 1) MUX\_OUT = IN1, and if SEL is low (i.e. SEL = 0) then MUX\_OUT = IN0.



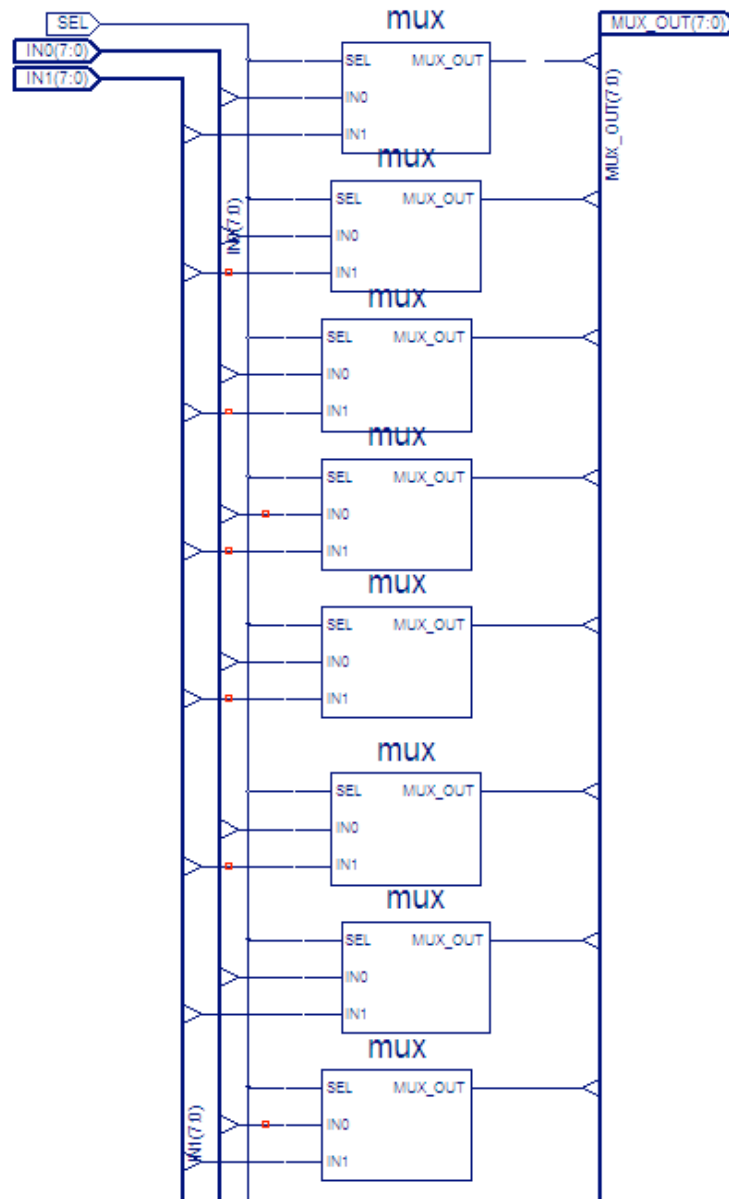
7. Highlight **mux.sch**, make sure the **Sources for** pull-down menu is on **Synthesys/Implementation** and double-click **Create Schematic Symbol** in the **Processes** tab. This creates a black-box type symbol for the 1-bit multiplexer. The default name given to it by **Project Manager** is **mux.sym**.



#### ECS : Hierarchical 8-bit mux

8. Add a **New Source** to the project of type schematic and call it **mux8\_sch.sch**.
9. On the **Symbols** tab and under its **Categories** in **ECS**, there will be the name of the project directory. Select it.
10. The symbol **mux** should be listed in the **Symbols** listbox. This is the symbol we created in **Project Manager**, **mux.sym**.

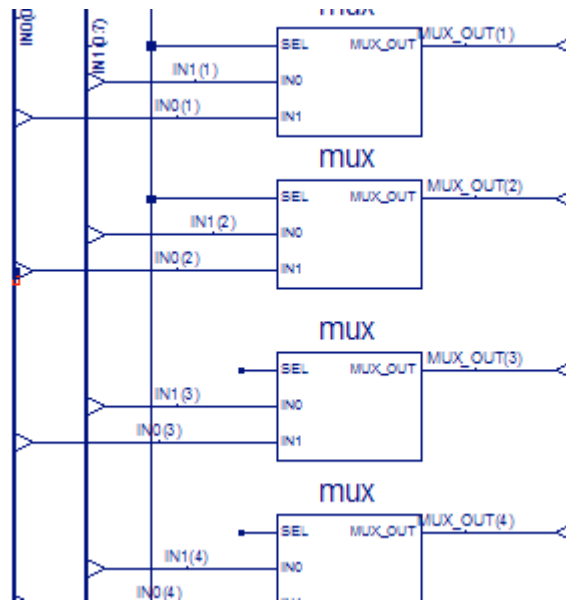
11. Place eight instances of **mux** on the schematic drawing sheet and connect them as follows:



*mux8\_sch.sch*

**Note:** Each IN0 of **mux** is connected to a single bus tap on the 8-bit bus IN0(7:0). Each IN1 of **mux** is also connected to a single bus tap on the 8-bit bus IN1(7:0). Each SEL of all the **muxes** is joined to a single net called SEL. Each MUX\_OUT of **mux** is connected to a different bus tap on the 8-bit bus MUX\_OUT(7:0).

**IMPORTANT:** Note that the net names of each of the bus tap nets are not indicated. They must be appropriately named from the most significant bit to the least. The format for naming bus tap nets is IN0(0), IN1(1) ..... IN1(0)IN1(1) ..... MUX\_OUT(0) MUX\_OUT(1).... Part of the View is as follows



**Part View of the 8 bit MUX Schematic**

12. Save **mux8\_sch.sch** after verifying that there are no **Schematic Check Errors**.
13. Use the following test bench to verify that **mux8** works correctly.

```
`timescale 1ns/1ps
```

```
module mux8_tbw_tb_0;
  reg [7:0] IN0 = 8'b00000000;
  reg [7:0] IN1 = 8'b00000000;
  reg SEL = 1'b0;
  wire [7:0] MUX_OUT;
```

```
  mux8_sch UUT (
    .IN0(IN0),
    .IN1(IN1),
    .SEL(SEL),
    .MUX_OUT(MUX_OUT));
```

```
  initial begin
    // ----- Current Time: 100ns
    #100;
    IN1 = 8'b01011111;
    // -----
    // ----- Current Time: 200ns
    #100;
    SEL = 1'b1;
    IN0 = 8'b01011111;
    IN1 = 8'b11001000;
    // -----
    // ----- Current Time: 300ns
    #100;
    IN1 = 8'b10010001;
    // -----
    // ----- Current Time: 400ns
    #100;
    IN0 = 8'b11001000;
```

```

IN1 = 8'b00011101;
// -----
// ----- Current Time: 500ns
#100;
SEL = 1'b0;
IN1 = 8'b11101010;
// -----
// ----- Current Time: 600ns
#100;
IN0 = 8'b10010001;
IN1 = 8'b01110011;
// -----
// ----- Current Time: 700ns
#100;
SEL = 1'b1;
IN1 = 8'b01110100;
// -----
// ----- Current Time: 800ns
#100;
IN0 = 8'b00011101;
IN1 = 8'b10101000;
// -----
// ----- Current Time: 1000ns
#200;
IN0 = 8'b11101010;
end

endmodule

```

Note: This is it for today. I hope you understand the way we can use **Engineering Capture System (ECS)** to design components. This is in addition to the HDL Verilog method we learned last semester in CSEE 4270. In this class, we will mostly use ECS, but sometimes Verilog, to design the Toy Processor. You will be able to design more complex components and test bench them before moving onto next steps.