ISE Tutorial

Contents

Part 0 : Preparations for working in CO-40 Computer Engineering Lab

Part 1 : Expected duration – 10 minutes
Introduction to ISE
Getting Started : Using the ***Project Manager***

Part 2 : Expected duration – 20 minutes
Learning how to use ***ECS*****:** Building the ZERO component

Part 3 : Expected duration – 20 minutes
Learning how to use ***ModelSim*****:** Simulating the ZERO component.

Part 4 : Expected duration – 20 minutes
Hierarchical Design: Building and simulating the one-bit full adder component.

Part 5 : Expected duration – 20 minutes
Hierarchical Design: Building and simulating the logic extender component.

Part 6 : Expected duration – 20 minutes
Hierarchical Design: Building and simulating the arithmetic extender component.

Part 7 : Expected duration – 20 minutes
Building and simulating the ALU : Everything comes together

0. PREPARATIONS

Getting access to CO-40
Most of the lab hours will be spent in DL-120 or CO-40. DL-120 is accessible after hours with your student ID card. CO-40 is the Computer Engineering lab located on the ground floor close to Davis Auditorium. You need to get your student ID card registered in order to gain access to this lab. The person in charge of giving you authorization is Ed Jackson in room CO 47 in the basement corridor between Becton and Watson. Go to his office in person with your student ID and he will give you access to CO-40. He can be reached Monday to Friday 9:00AM-5:00PM.

Mapping your pantheon drive.
Pantheon drive is your own storage space provided by Yale ITS. To access your pantheon drive from the computers in CO-40, go to Start->Run, then type \\pantheon.its.yale.edu\netID, where netID is your unique netID. Your work should be stored in your pantheon drive. Every time you need to work on the lab, just drag your work from your pantheon account into the C:\XilinxWorks folder and use this as your working directory.

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. Please don't save any work in the desktop when you are working on the lab. Every file on the desktop will have a space in the path name. Xilinx can't read a file from a path name with a space in it. Please use C:\XilinxWorks as your working directory. You should not leave your files in C:\XilinxWork in the lab computers after you leave the lab. Make sure that all your work, finished or unfinished, is copied onto your own pantheon drive before you delete them from the lab machine's hard drive. You can then also access it from the machines in DL120.

1. INTRODUCTION

Xilinx ISE 6.2i, which you will be using for this course is a very large piece of software with very many capabilities. In EENG348, we will use just a few of these capabilities.

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 Manager***. In this class we shall be using four other modules: ***Engineering Capture System (ECS)***, ***StateCad***, ***HDL Bencher*** and ***ModelSim***.

Project Manager
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 (Fully Programmable Gate Array) chip. Very 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.

StateCad

StateCAD is a graphical entry tool that allows engineers to express their ideas as state diagrams. State diagrams are used to represent systems which undergo discrete state changes, e.g. a traffic light turns red for 10 seconds, and then yellow for 5 secs, before green for 15 seconds.

Simulation

HDL Bencher

HDL Bencher 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 Bencher constrains the test run to a specific sequence of events, initial conditions, and user determined results. With HDL Bencher you can quickly validate if your design functions as intended.

ModelSim

ModelSim 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.

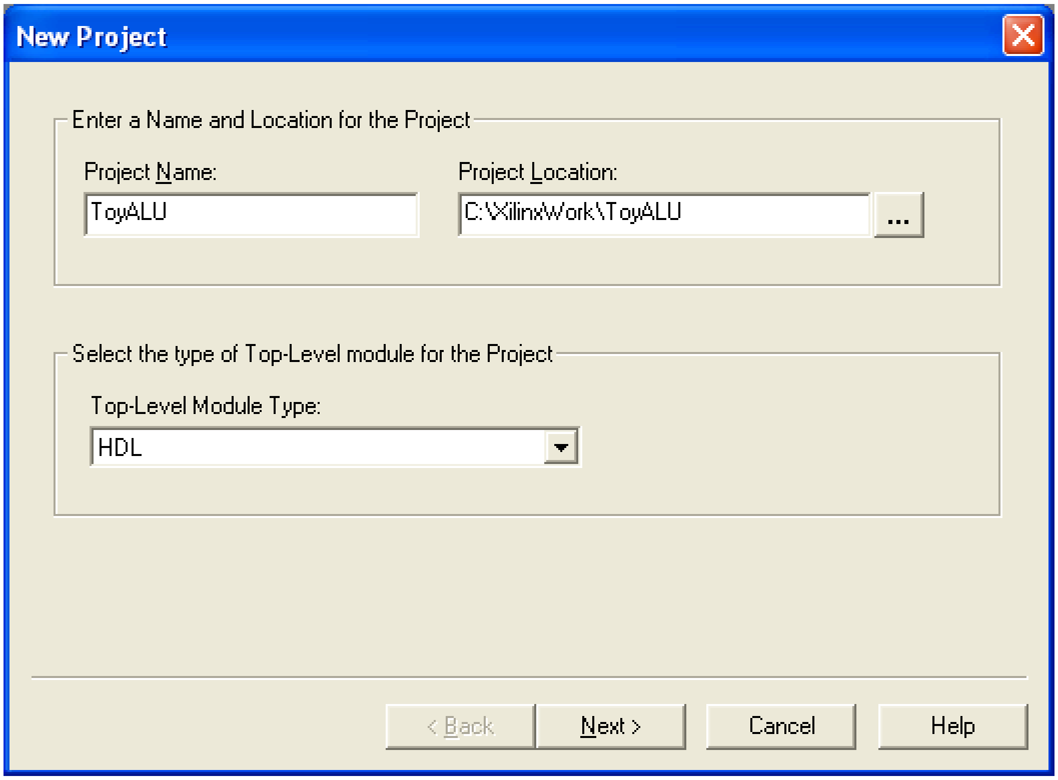
We will now familiarize ourselves with ISE.

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

1. ***Start →Programs →Xilinx ISE 6 →Project Navigator***.
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 C:\XilinxWork if you are working in C0-40. Also, if you are working in C0-40, please remember to save your work on your pantheon drive once you are about to leave, and delete your work in XilinxWork folder on C drive. This project is going to be called ***ToyALU***. Please keep in mind Xilinx doesn't like space in directory or file names.

The Project Device should be set to the following values:

Property Name	Value
Device Family	Spartan2
Device	xc2s50
Package	pq208
Speed Grade	-6
Design Flow	XST VHDL



Enter a Name and Location for the Project

Project Name: ToyALU

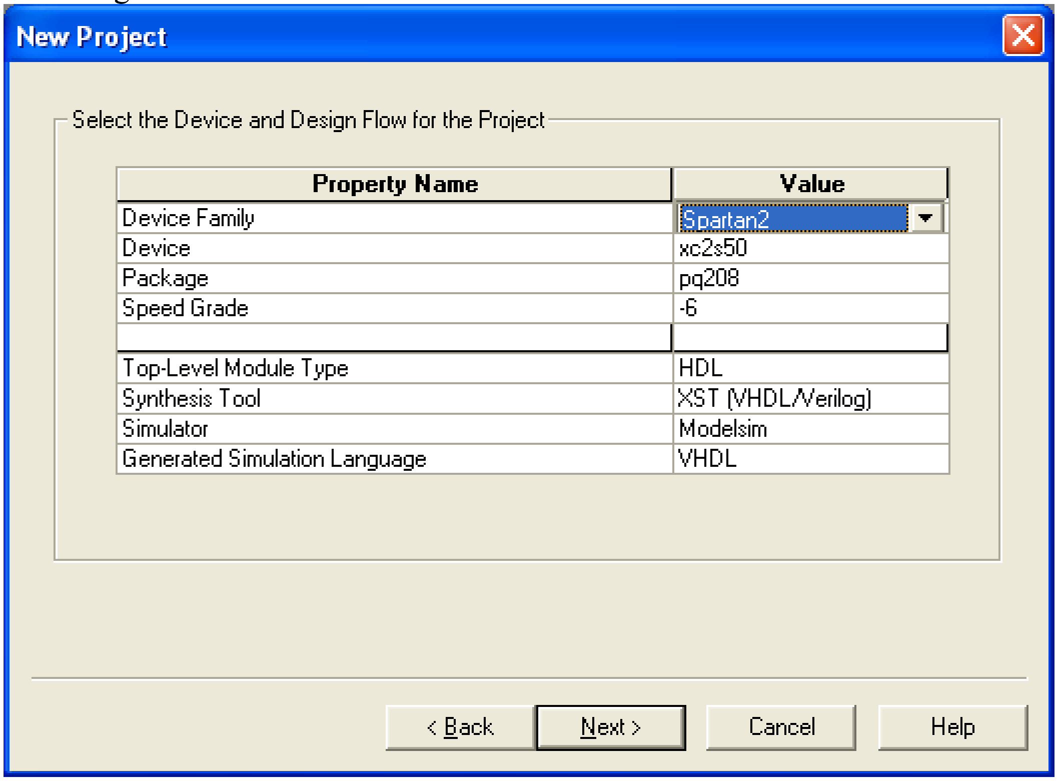
Project Location: C:\XilinxWork\ToyALU

Select the type of Top-Level module for the Project

Top-Level Module Type: HDL

New project dialog box

Click ***Next***, you should get the following window, please set the appropriate values according to the table above:

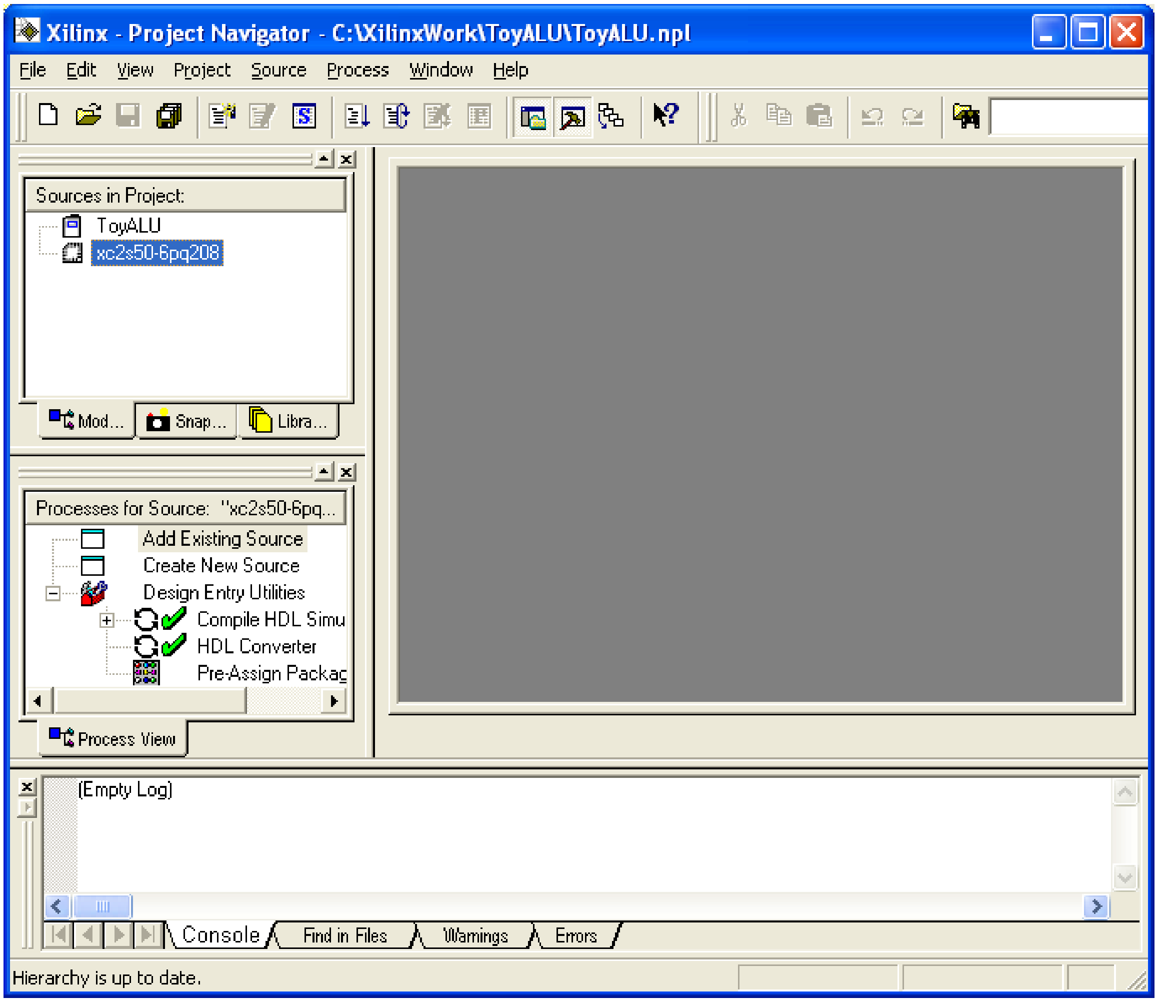


Select the Device and Design Flow for the Project

Property Name	Value
Device Family	Spartan2
Device	xc2s50
Package	pq208
Speed Grade	-6
Top-Level Module Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	Modelsim
Generated Simulation Language	VHDL

New project dialog box

Click *Next, Next, Next,* make sure you have all values set correctly before you click ***Finish*** to create the project. Your ***Project Manager*** window should look like this:



Project Manager showing "Sources in Project—toZERO, xc2s200e XST VHDL"

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

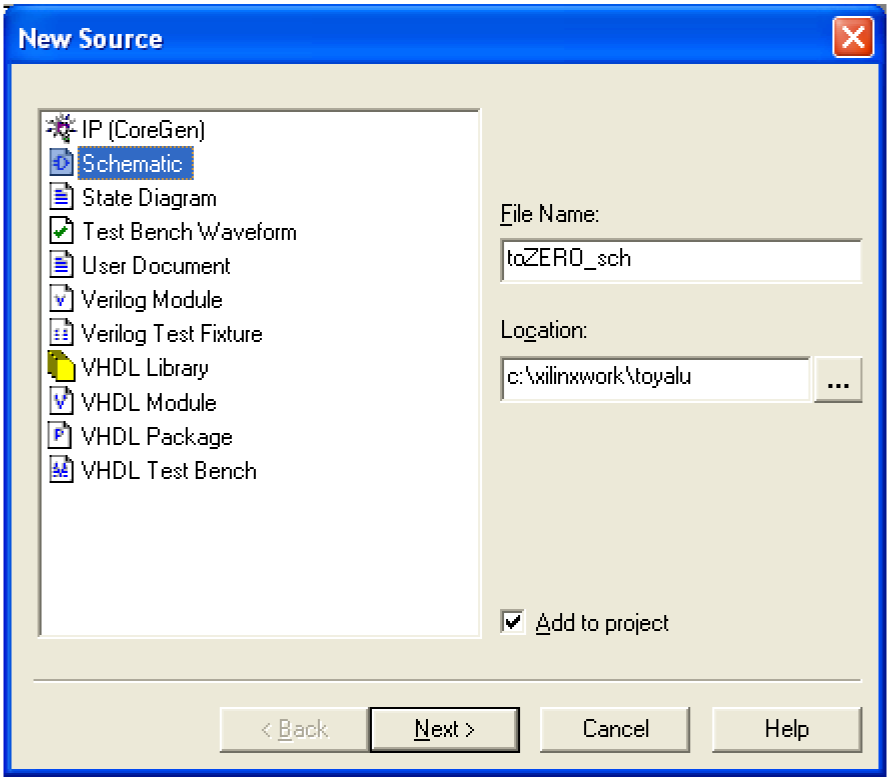
ISE uses ***ECS*** for drawing schematics. To start up ***ECS,*** go to Project->New Sources. Alternatively right-click on the project name, ToyALU, and select New Source.

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.**
   NB: Make sure the ***Add to Project*** checkbox is checked. This adds **toZERO_sch** to the list of sources in ***Project Manager.***



File Name: toZERO_sch

Location: c:\xilinxwork\toyalu

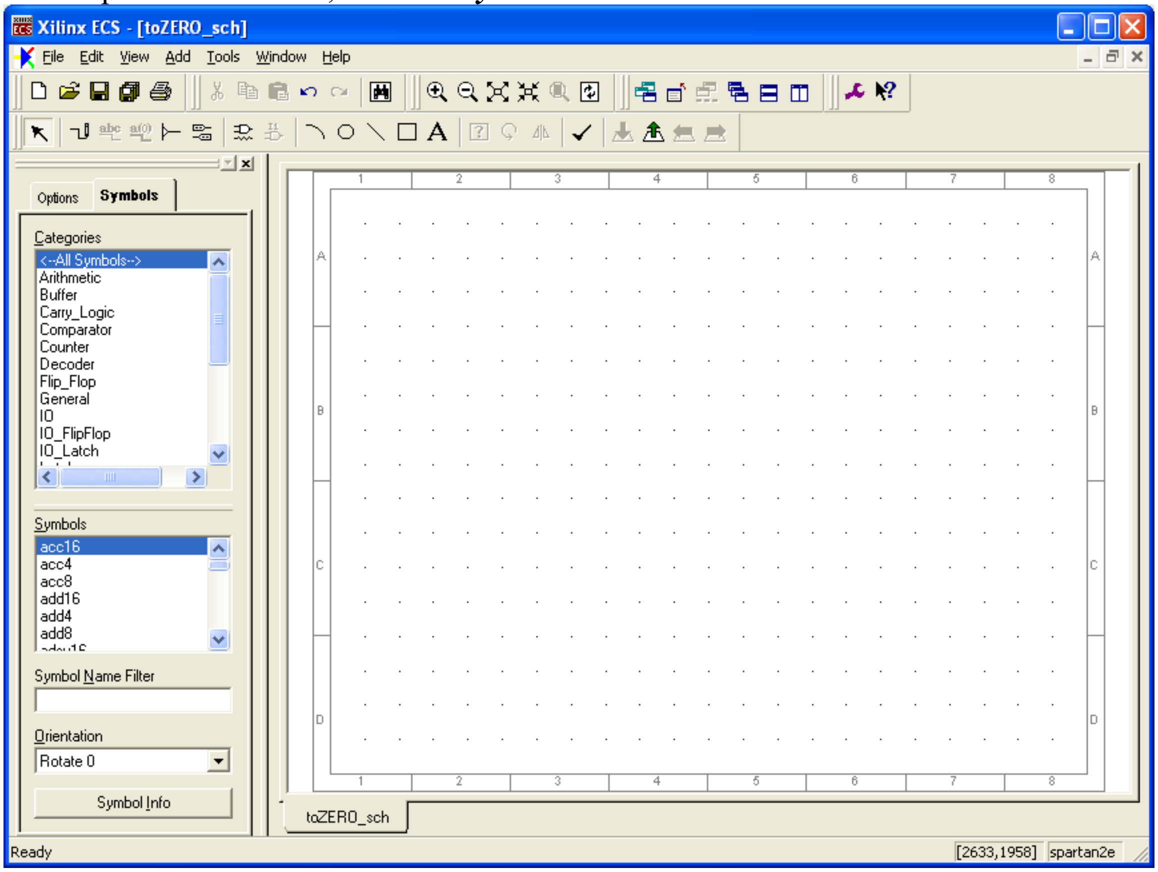
☑ Add to project

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 ***Symbols*** tab.



***ECS** schematic with the **Symbols** tab selected*

2. Type **nor** in the ***Symbol Name Filter*** to display symbols that begin with "nor". We want **nor8.**
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.

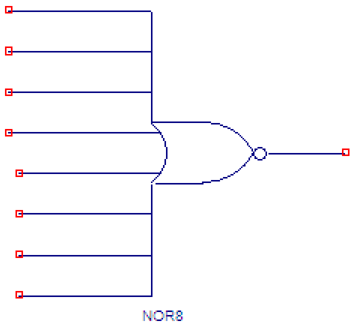
ECS : Adding wires

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:



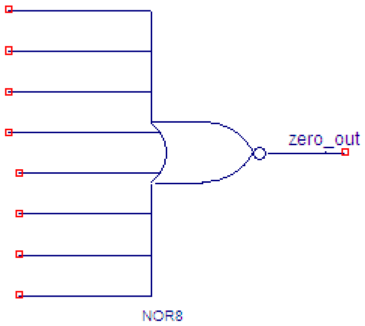
nor8 *gate with ports extended by nets (wires)*

ECS **: Adding Net Names**

5. Select the ***Add Net Names*** tool abc from the ***Drawing Toolbar***. Type **zero_out** in the textbox and then place the net name on the end of the **nor8** output net by clicking it.

   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.*

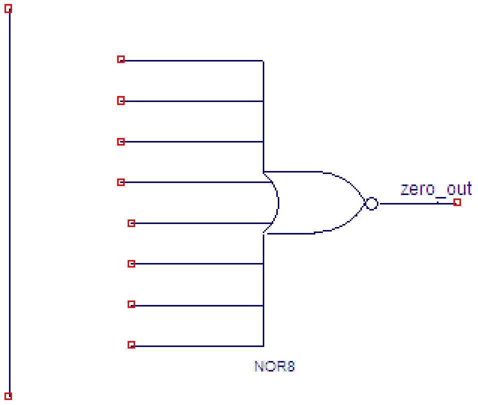
6. Your schematic should look similar to the following figure:



Naming the output net

ECS **: Adding a bus and bus taps**

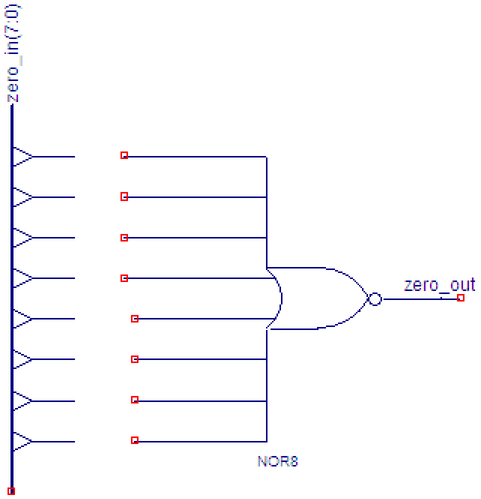
7. Draw a vertical net to the left of the net extensions of the **nor8** input. It should look something like this. The vertical net should be some distance away from the terminal points of the input nets of the **nor8** gate.



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

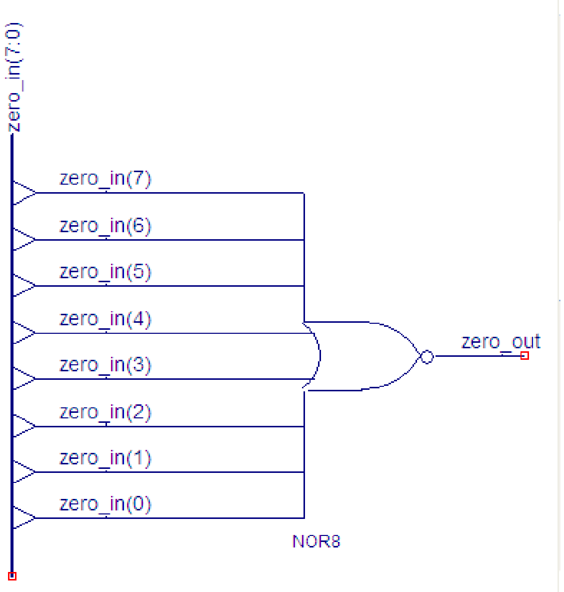
8. Name the vertical net **zero_in(7:0).** The net name indicates that this is a bus with 8 ports (from 0 to 7). Bus nets in ***ECS*** are thicker than single nets.

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

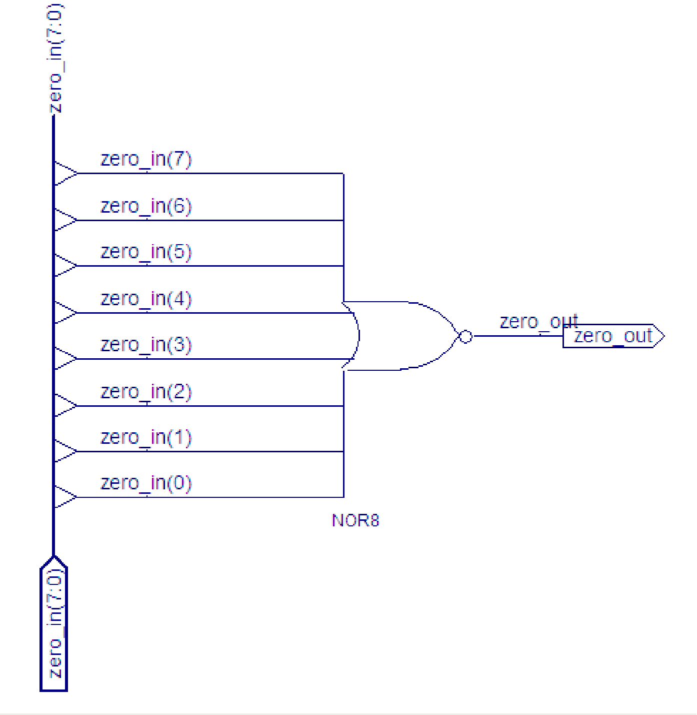
11. Use nets to connect the bus taps to corresponding inputs for simplicity and clarity of design.
12. Add net names to each of the connecting nets. Start with net name **zero_in(7)** and select ***Decrement the name*** in the ***After naming the branch*** groupbox. Alternatively you can start from **zero_in(0)** and **Increment 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 **Schematic Check Errors** dialog box that pops up should report no errors. 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
Learning how to use ***ModelSim*****:** Simulating the ZERO component.

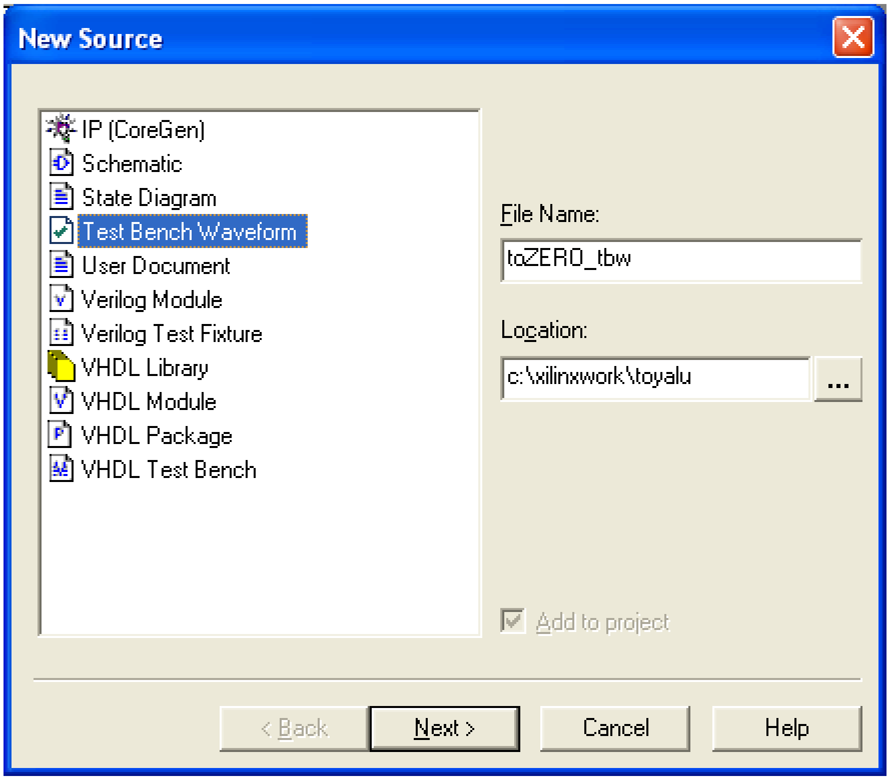
The simulation process consists of two steps. It is handled by two software components of the ***ISE*** suite. The first step involves creating a test bench waveform. This is a test waveform consisting of test signals against which you will test the functionality of your logic circuit.

The second step involves the actual running of a simulation using the test signals created in the test bench.

If you are using *ModelSim* in C0-40 lab for the first time, please remember to run the License Wizard before running *ModelSim*. The License Wizard is located at start->All programs->ModelSim XE II 5.7g->Licensing Wizard. The license is located at C:\Documents and Settings\All Users\Documents\license.dat on C0-40 machines. You may have to run it twice in order to activate *ModelSim*. Follow the prompts.

HDL Bencher **: Creating a Test bench waveform**

1. First, we create a test bench waveform from ***Project Navigator*** which we will modify in HDL Bencher. Highlight **toZERO_sch.sch** in the ***Sources in Project*** window.
2. Select ***Project*** → ***New Source***. In the ***New*** dialog box, select ***Test Bench Waveform*** as the source type. Name it **toZERO_tbw.tbw**. (***Remember, somehow Xilinx doesn't like it when you name 2 files with the same name even though they have different file extensions***.)



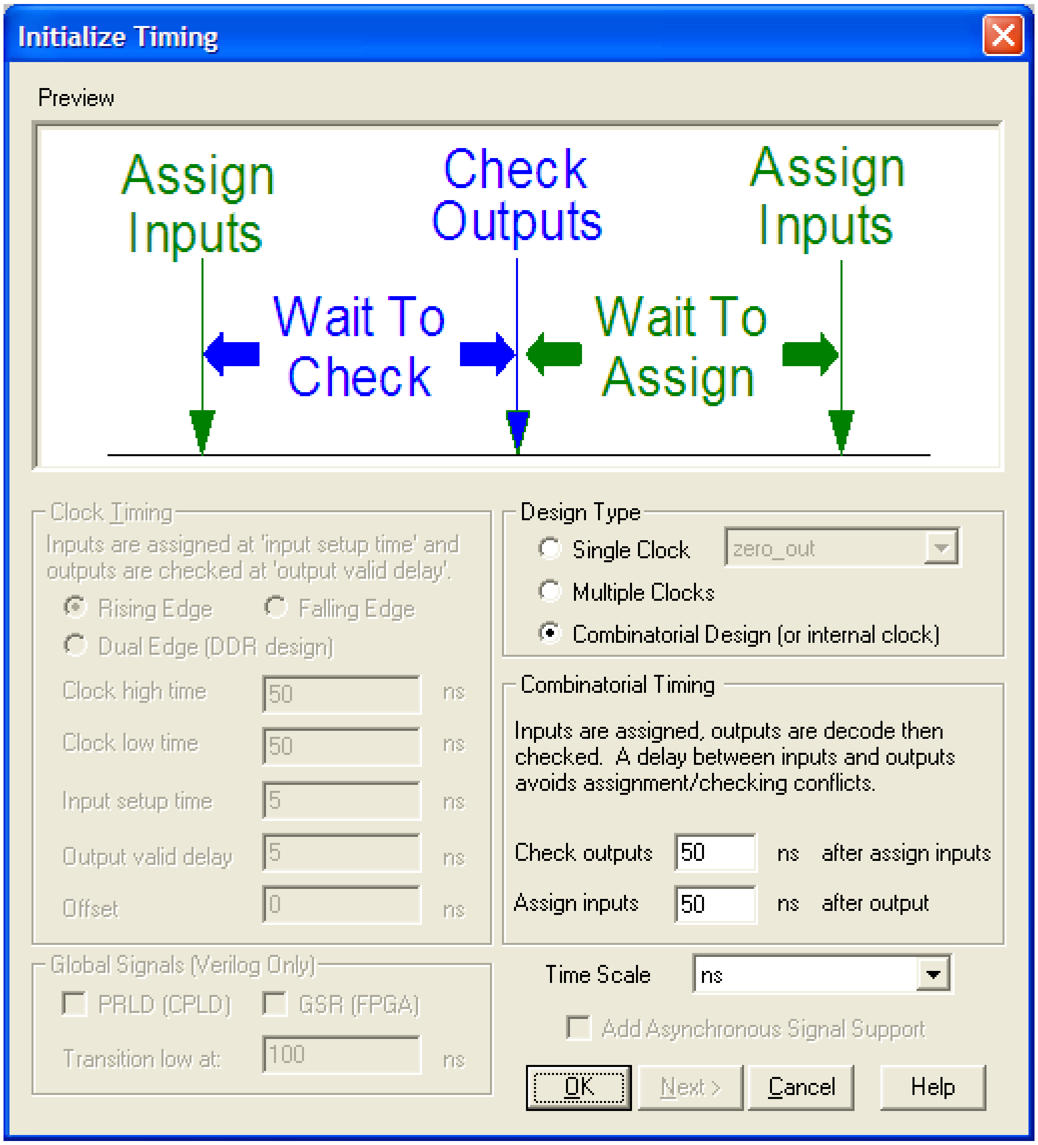
File Name: toZERO_tbw

Location: c:\xilinxwork\toyalu

3. Click ***Next*** and ***Associate with Source*** **toZERO_sch.**

Make sure you have already run the license wizard twice for ModelSim. ***HDL Bencher*** will be launched and ready for timing requirements to be entered. We will now specify the timing parameters used during simulation. The **clock high time** and **clock low time** together define the clock period (period = high time + low time) the design operates with. The **Input setup time** defines when inputs must be valid. The **Output valid delay** defines the time after active clock edge when the outputs must be valid.

For this tutorial, you will not change any of the default timing constraints. The default Initialize Timing settings are the following:
Design type: **Combinatorial Design (or internal clock)**
Check outputs time: **50 ns**
Assign inputs: **50 ns**
Time scale: **ns**



Clock Timing

Inputs are assigned at 'input setup time' and outputs are checked at 'output valid delay'.

● Rising Edge ○ Falling Edge

○ Dual Edge (DDR design)

Clock high time 50 ns

Clock low time 50 ns

Input setup time 5 ns

Output valid delay 5 ns

Offset 0 ns

Global Signals (Verilog Only)

□ PRLD (CPLD) □ GSR (FPGA)

Transition low at: 100 ns

Design Type

○ Single Clock zero_out

○ Multiple Clocks

● Combinatorial Design (or internal clock)

Combinatorial Timing

Inputs are assigned, outputs are decode then checked. A delay between inputs and outputs avoids assignment/checking conflicts.

Check outputs 50 ns after assign inputs

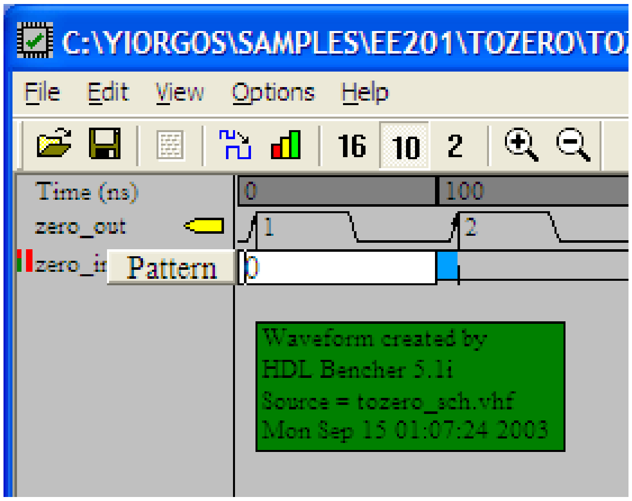
Assign inputs 50 ns after output

Time Scale ns

□ Add Asynchronous Signal Support

Default timing constraints

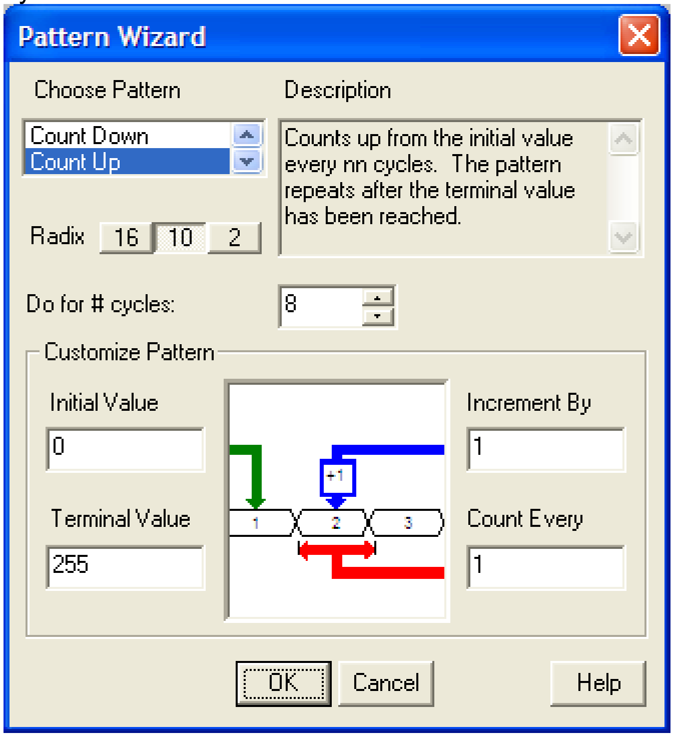
4. Click ***OK*** to accept the default timing constraints.
5. Double-click the first cell of the signal waveform **zero_in[7:0].** A button labeled **Pattern** will be displayed. Click it.


C:\YIORGOS\SAMPLES\EE201\TOZERO\TO
File Edit View Options Help
16 10 2
Time (ns)
0
100
zero_out
1
2
zero_in
Pattern
0

Waveform created by
HDL Bencher 5.1i
Source = tozero_sch.vhf
Mon Sep 15 01:07:24 2003

Creating a test pattern

6. Set **Choose Pattern** to ***Count Up***. **Do for # of cycles** should be set to 8, incrementing by 1 with an initial value of 0. The ***Pattern Wizard*** will look as follows:



Choose Pattern: Count Up

Counts up from the initial value every nn cycles. The pattern repeats after the terminal value has been reached.

Radix 16 10 2

Do for # cycles: 8

Initial Value: 0

Terminal Value: 255

Increment By: 1

Count Every: 1

Pattern Wizard

7. Click **OK** to accept the pattern. **Zero_in[7:0]** will be set to 0 to 7 from the first to the eight cycle respectively.
8. Save the test bench. You may close ***HDL Bencher***. We are now ready to simulate our schematic.

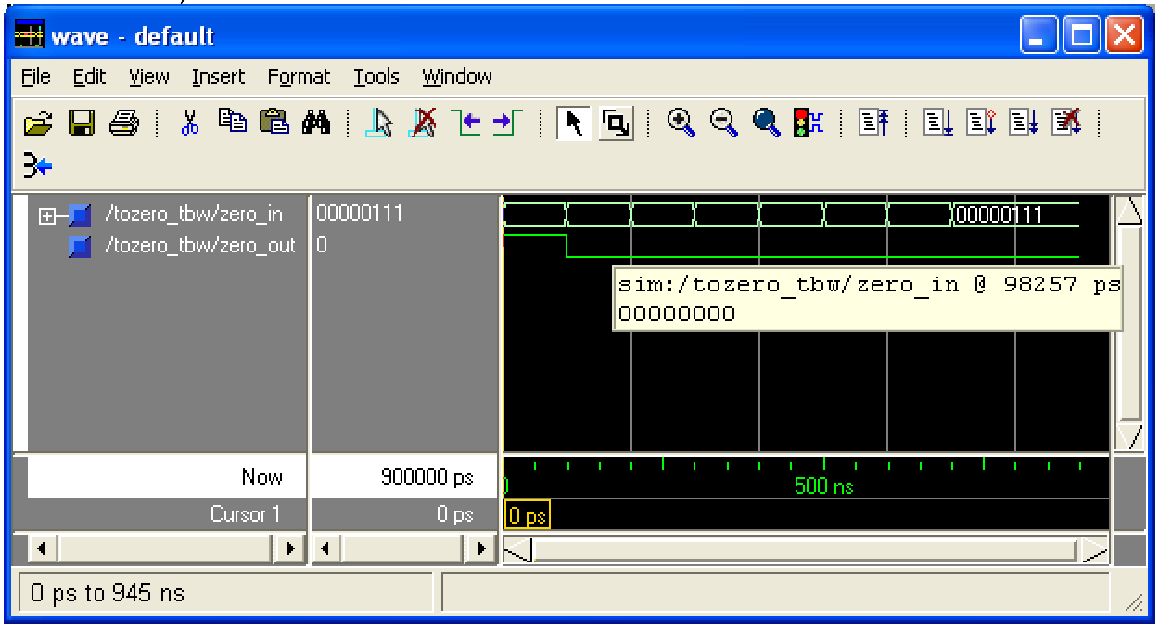
ModelSim : Running the simulation

9. In ***Project Manager*** highlight **toZERO_tbw.tbw.** The ***Processes for Current Source*** will display an expandable toolbox tree ***ModelSim* Simulator.**

10. Expand ***ModelSim* Simulator** and double-click **Simulate Behavioral Model.** This will run the ***ModelSim*** simulator process.

    Note: ModelSim *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.*

11. Your output should look like this (you may have to select view->Zoom->Zoom Full to get the whole view):



wave—*simulation result.* ***Zero_out*** *is high when* ***zero_in*** *is zero. Otherwise it is low.*

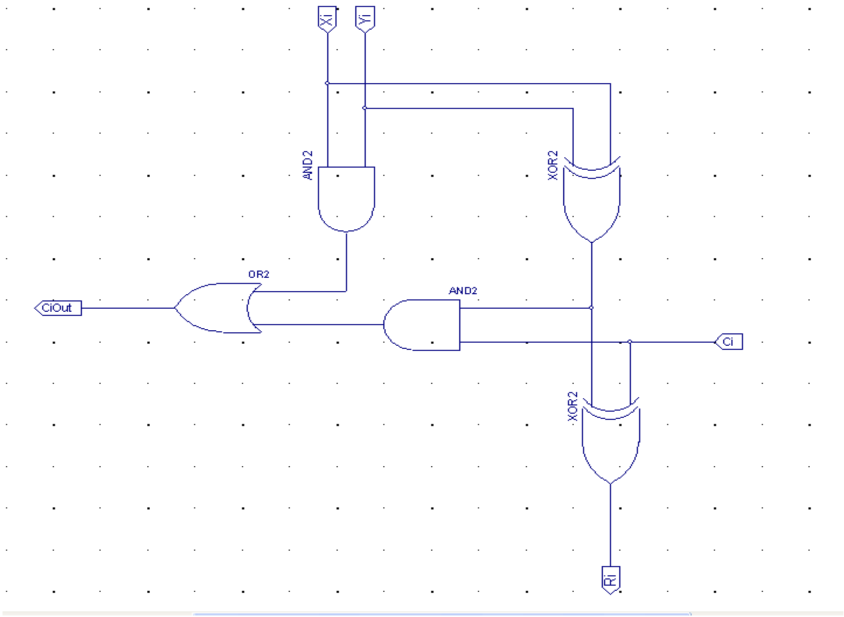
CONGRATULATIONS! You have finished your very first project!

Part 4
Hierarchical Design: Building the one-bit full adder component

Next we are going to build a single bit full adder from which we will build an ALU later. This is called hierarchical design.

1. Add a **New Source** to the existing project in ***Project Manager*** of type **Schematic**. Call it **one_bit_fa.sch.**

2. In ***ECS***, draw the following schematic:

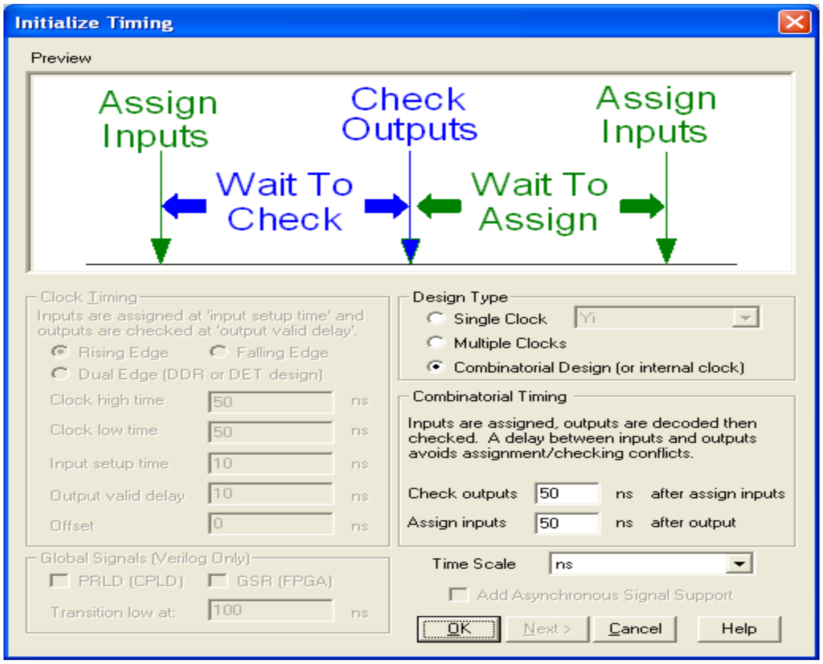


1-bit full adder. Signals: input->Xi, Yi, and Ci; output->Ri and CiOut

3. Save the schematic. Create a test bench for the schematic called **fa_tbw.tbw** and associate it with **one_bit_fa.**

   Note: *Because there is no clock signal in this design, our test bench will use the following timing constraints (i.e. the timing constraints of a combinatorial circuit):*

   Design type: ***Combinatorial Design (or internal clock)***
   Check outputs time: ***50 ns***
   Assign inputs: ***50 ns***
   Time scale: ***ns***



Initialize Timing

Preview

Clock Timing
Inputs are assigned at 'input setup time' and outputs are checked at 'output valid delay'.
● Rising Edge ○ Falling Edge
○ Dual Edge (DDR or DET design)
Clock high time 50 ns
Clock low time 50 ns
Input setup time 10 ns
Output valid delay 10 ns
Offset 0 ns

Global Signals (Verilog Only)
□ PRLD (CPLD) □ GSR (FPGA)
Transition low at: 100 ns

Design Type
○ Single Clock Yi
○ Multiple Clocks
● Combinatorial Design (or internal clock)

Combinatorial Timing
Inputs are assigned, outputs are decoded then checked. A delay between inputs and outputs avoids assignment/checking conflicts.
Check outputs 50 ns after assign inputs
Assign inputs 50 ns after output

Time Scale ns

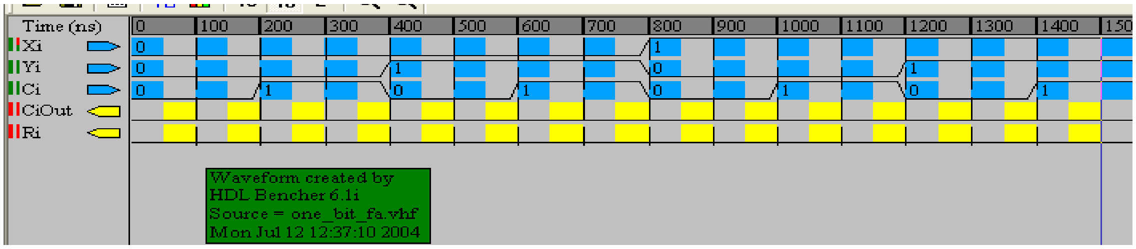
□ Add Asynchronous Signal Support

OK Next > Cancel Help

4. Set your simulation signal against the following table:

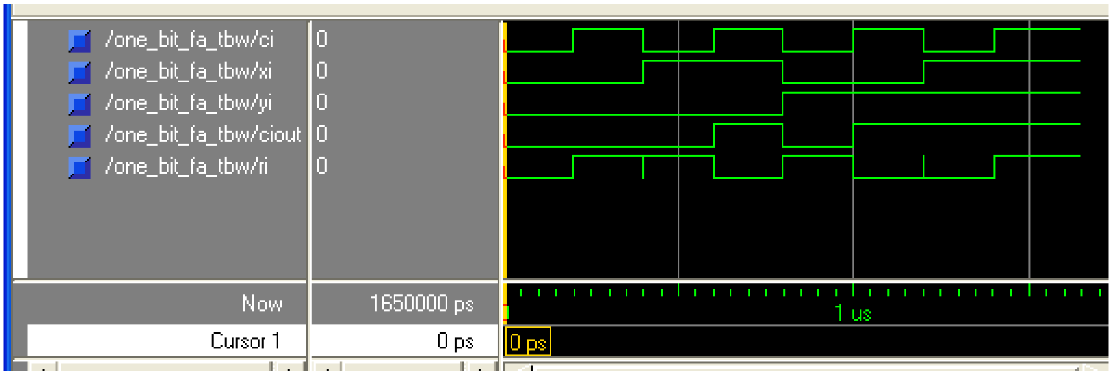
Xi	Yi	Ci	Ri	CiOut
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

5. Click the small square at a particular time interval to toggle the signal value (eg. Xi Yi and Ci) between 0 and 1



Waveform created by
HDL Bencher 6.1i
Source = one_bit_fa.vhf
Mon Jul 12 12:37:10 2004

one_bit_fa test bench waveform

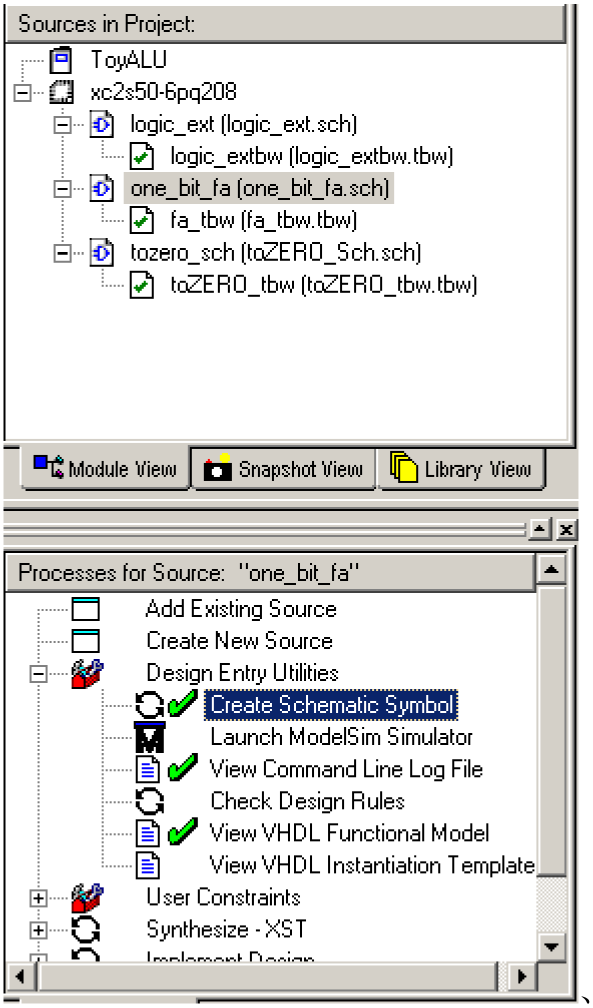


```
/one_bit_fa_tbw/ci     0
/one_bit_fa_tbw/xi     0
/one_bit_fa_tbw/yi     0
/one_bit_fa_tbw/ciout  0
/one_bit_fa_tbw/ri     0

Now                    1650000 ps
Cursor 1               0 ps
```

Simulation result for 1-bit full adder

6. Create a symbol from it using ***Project Manager.***



Sources in Project:

- ToyALU
- xc2s50-6pq208
  - logic_ext (logic_ext.sch)
    - logic_extbw (logic_extbw.tbw)
  - one_bit_fa (one_bit_fa.sch)
    - fa_tbw (fa_tbw.tbw)
  - tozero_sch (toZERO_Sch.sch)
    - toZERO_tbw (toZERO_tbw.tbw)

Processes for Source: "one_bit_fa"

- Add Existing Source
- Create New Source
- Design Entry Utilities
  - Create Schematic Symbol
  - Launch ModelSim Simulator
  - View Command Line Log File
  - Check Design Rules
  - View VHDL Functional Model
  - View VHDL Instantiation Template
- User Constraints
- Synthesize - XST
- Implement Design

Note: *To create a symbol, highlight the name of the schematic file and double-click* ***Create Schematic Symbol*** *from the* ***Design Entry Utility*** *in the* ***Process View*** *tab. This creates a black-box type symbol for the 1-bit full adder. The default name given to it by* ***Project Manager*** *is* ***one_bit_fa.sym.*** *Or you can use the Symbol Wizard under Tools in schematic view of the component.*

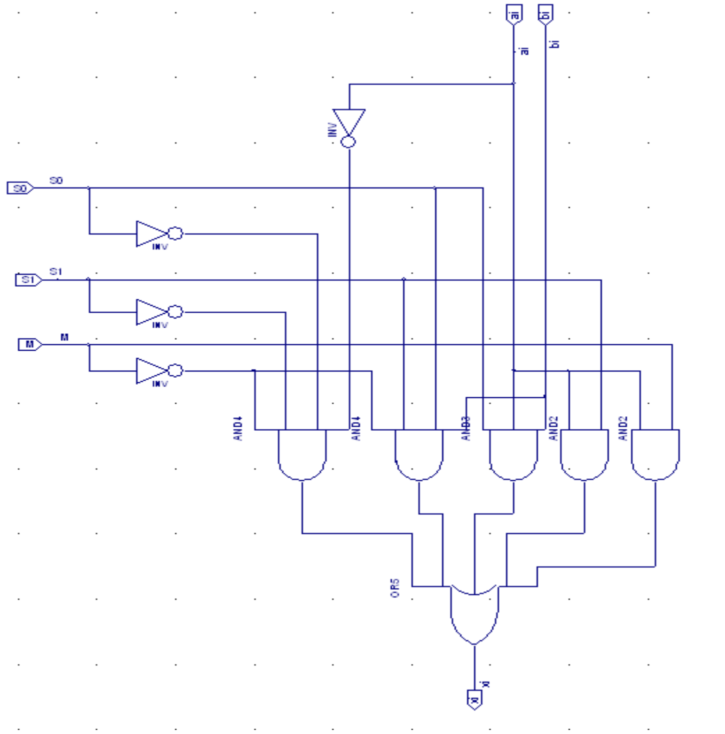
Part 5

Hierarchical Design: Building a logic extender component

Next we are going to build a single logic extender. Its functions are described by the following table:

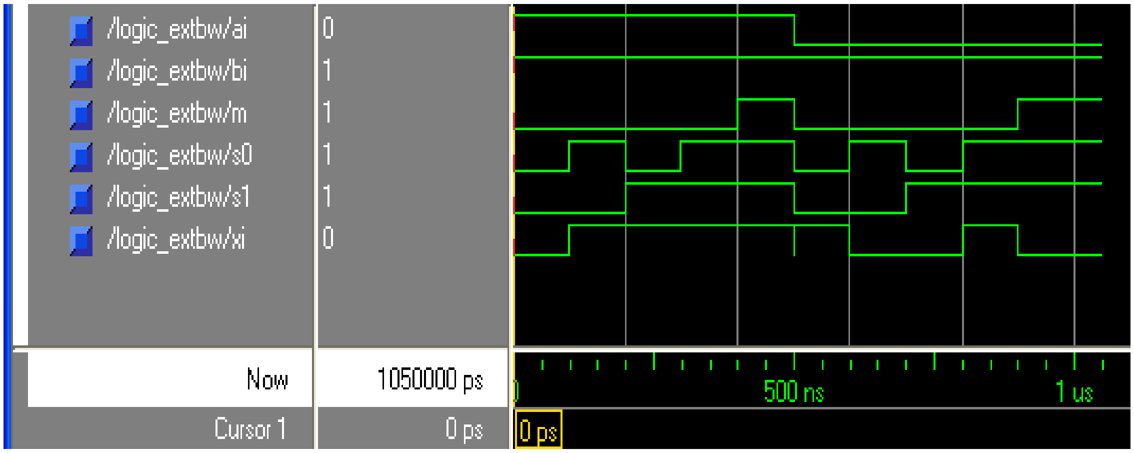
M	S1	S0	Fun. Name	F	x_i
0	0	0	Complement	A’	a_i’
0	0	1	AND	A AND B	$a_i b_i$
0	1	0	Identity	A	a_i
0	1	1	OR	A OR B	$a_i + b_i$
1	X	X	X	X	a_i

1. With ***Project Manager*** still open, create a **New Source** of type schematic. Call it **logic_ext.sch**. It is going to be a logic extender, a basic component of our ALU.
2. Draw the following schematic:



logic extender

3. Create a test bench called **logic_extbw.tbw.**
4. Simulate and make sure your component functions as specified by the table above.

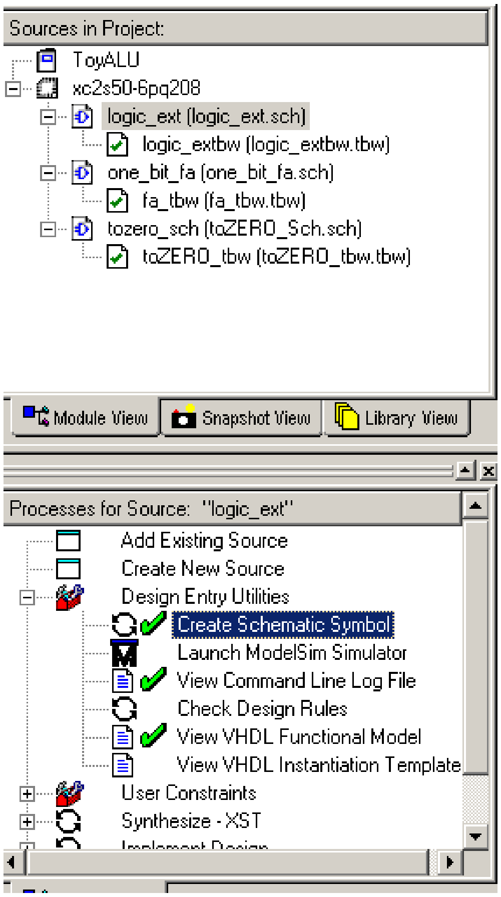


```
/logic_extbw/ai   0
/logic_extbw/bi   1
/logic_extbw/m    1
/logic_extbw/s0   1
/logic_extbw/s1   1
/logic_extbw/xi   0

          Now   1050000 ps
     Cursor 1         0 ps
```

A sample simulation result

5. Close **ModelSim** and create a symbol from it using ***Project Manager.***


Sources in Project:
ToyALU
xc2s50-6pq208
logic_ext (logic_ext.sch)
logic_extbw (logic_extbw.tbw)
one_bit_fa (one_bit_fa.sch)
fa_tbw (fa_tbw.tbw)
tozero_sch (toZERO_Sch.sch)
toZERO_tbw (toZERO_tbw.tbw)
Module View
Snapshot View
Library View
Processes for Source: "logic_ext"
Add Existing Source
Create New Source
Design Entry Utilities
Create Schematic Symbol
Launch ModelSim Simulator
View Command Line Log File
Check Design Rules
View VHDL Functional Model
View VHDL Instantiation Template
User Constraints
Synthesize - XST
Implement Design

***Note:** To create a symbol, highlight the name of the schematic file and double-click **Create Schematic Symbol** from the **Design Entry Utility** in the **Process View** tab. This creates a black-box type symbol for the logic extender. The default name given to it by **Project Manager** is **logic_ext.sym.** Or you can use the Symbol Wizard under Tools in schematic view of the component.*

Part 6

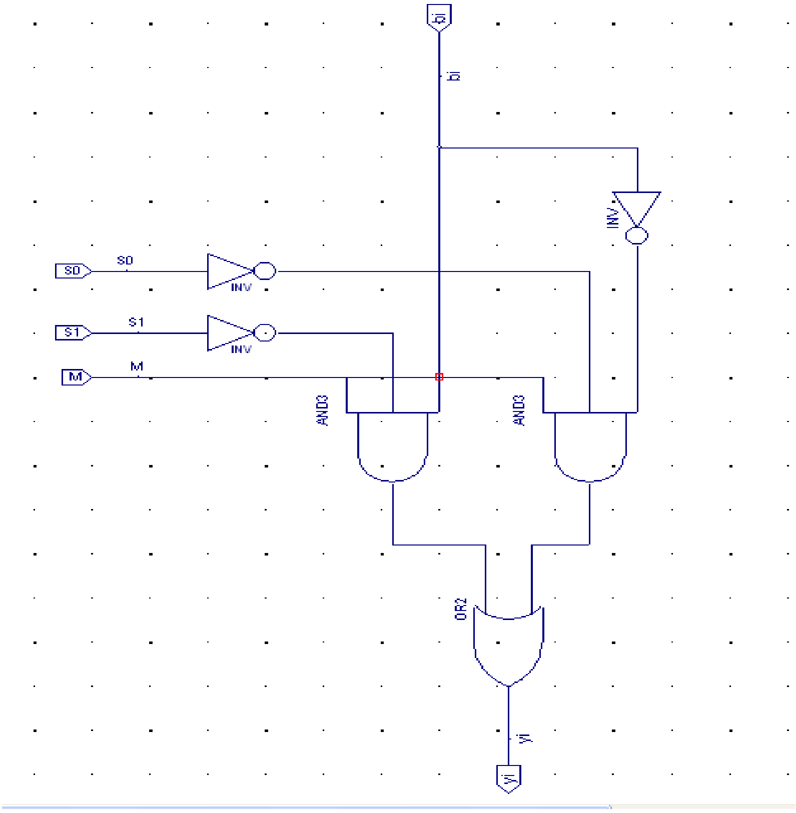
Hierarchical Design: Building a arithmetic extender component

Now we are going to build a single arithmetic extender. Its functions are described by the following table:

M	S1	S0	Fun. Name	F	X	Y	C0
1	0	0	Decrement	A – 1	A	All 1’s	0
1	0	1	Add	A + B	A	B	0
1	1	0	Subtract	A + B’ + 1	A	B’	1
1	1	1	Increment	A + 1	A	All 0’s	1

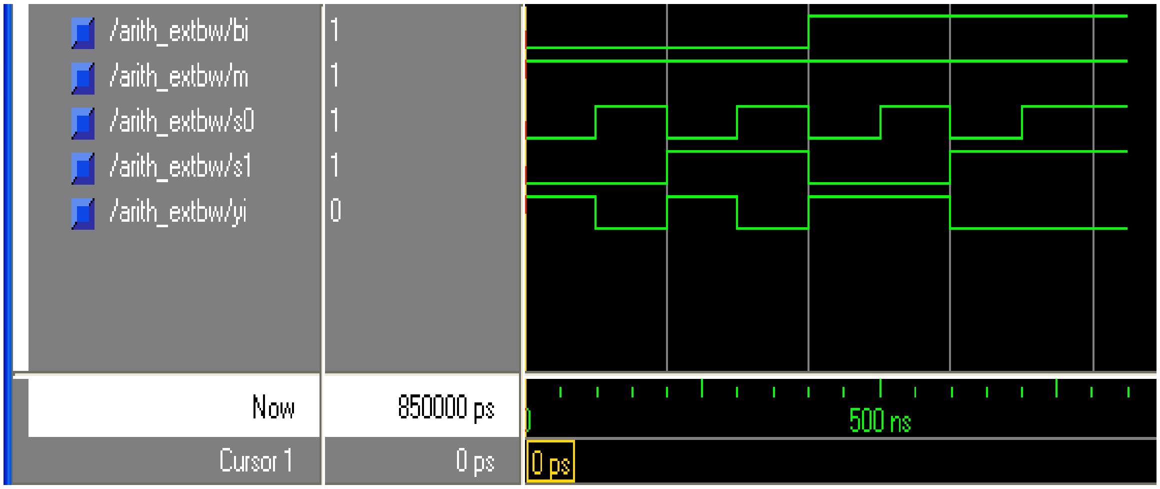
1. With ***Project Manager*** still open, create a **New Source** of type schematic. Call it **arith_ext.sch**. It is going to be the arithmetic extender, a basic component of our ALU.

2. Draw the following schematic:



arithmetic extender

3. Run “check schematic” **.**
4. Create a test bench called **arith_extbw.tbw.**
5. Simulate and make sure your component functions as specified by the table above.

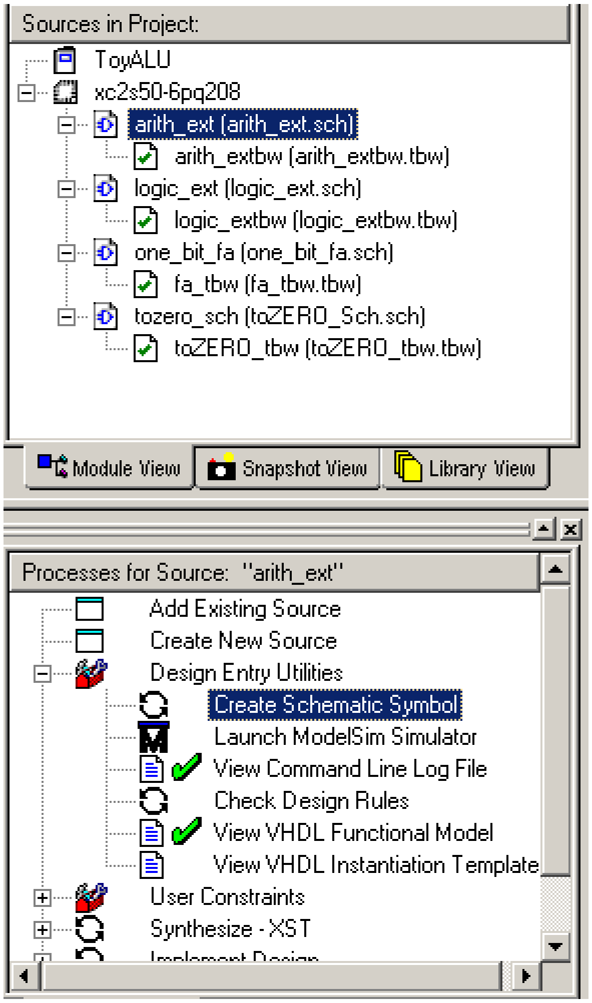


```
/arith_extbw/bi   1
/arith_extbw/m    1
/arith_extbw/s0   1
/arith_extbw/s1   1
/arith_extbw/yi   0

Now          850000 ps
Cursor 1     0 ps
```

A sample simulation result

6. Close **ModelSim** and create a symbol for this module using ***Project Manager.***



Sources in Project:

- ToyALU
- xc2s50-6pq208
  - arith_ext (arith_ext.sch)
    - arith_extbw (arith_extbw.tbw)
  - logic_ext (logic_ext.sch)
    - logic_extbw (logic_extbw.tbw)
  - one_bit_fa (one_bit_fa.sch)
    - fa_tbw (fa_tbw.tbw)
  - tozero_sch (toZERO_Sch.sch)
    - toZERO_tbw (toZERO_tbw.tbw)

Processes for Source: "arith_ext"

- Add Existing Source
- Create New Source
- Design Entry Utilities
  - Create Schematic Symbol
  - Launch ModelSim Simulator
  - ✓ View Command Line Log File
  - Check Design Rules
  - ✓ View VHDL Functional Model
  - View VHDL Instantiation Template
- User Constraints
- Synthesize - XST
- Implement Design

Note: *To create a symbol, highlight the name of the schematic file and double-click **Create Schematic Symbol** from the **Design Entry Utility** in the **Process View** tab. This creates a black-box type symbol for the arithmetic extender. The default name given to it by **Project Manager** is **arith_ext.sym.** Or you can use the Symbol Wizard under Tools in schematic view of the component.*

Part 7
Building the ALU : Everything comes together.

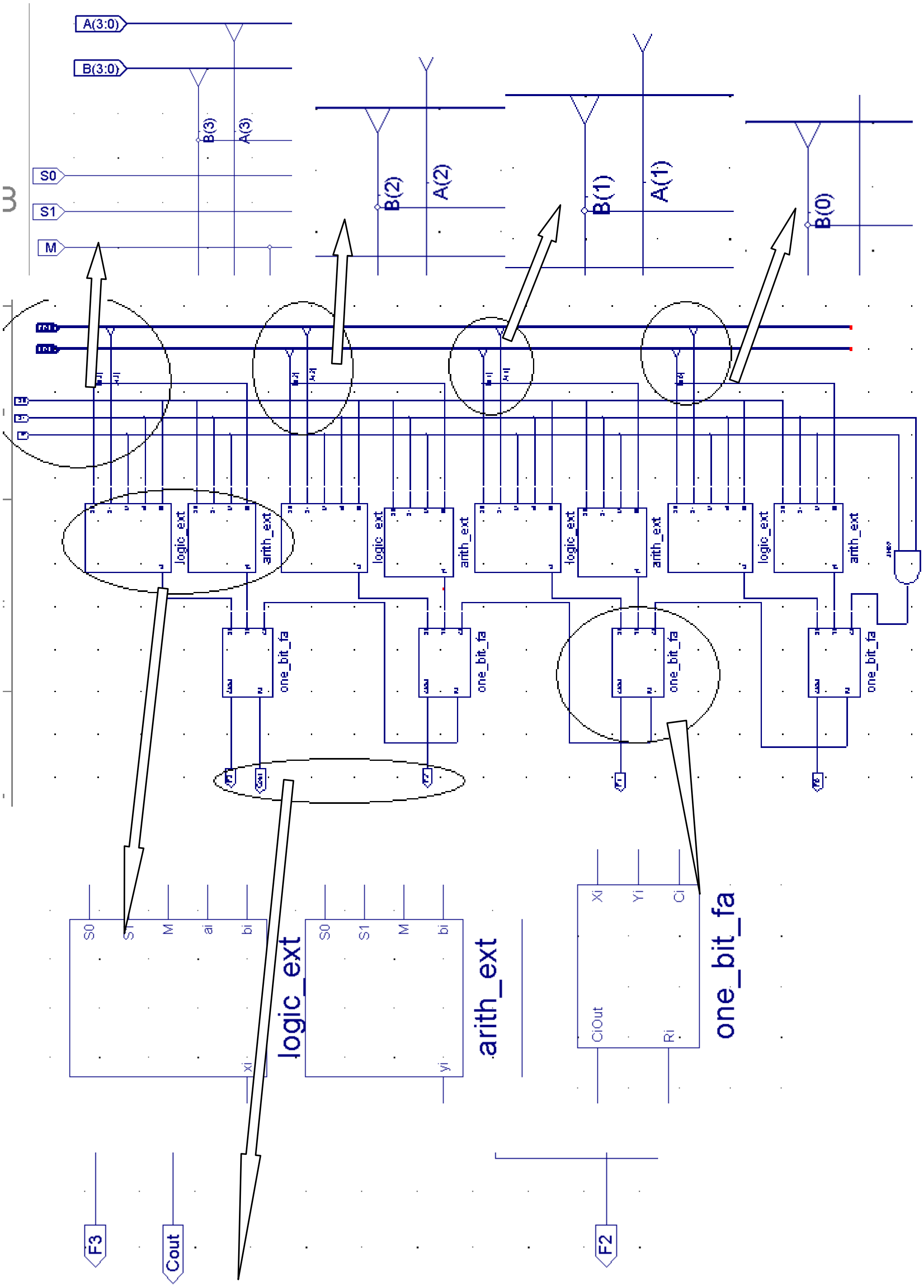
At last, we are going to put everything together and build a 4-bit ALU.

***ECS* : Building the ALU from all the components we have.**

1. Add a **New Source** of type schematic to the project. Name it **alu_sch.sch.**
2. Draw the schematic of a 4-bit ALU.

Note: *SEL is connected to the CARRY of the least significant bit (LSB).*

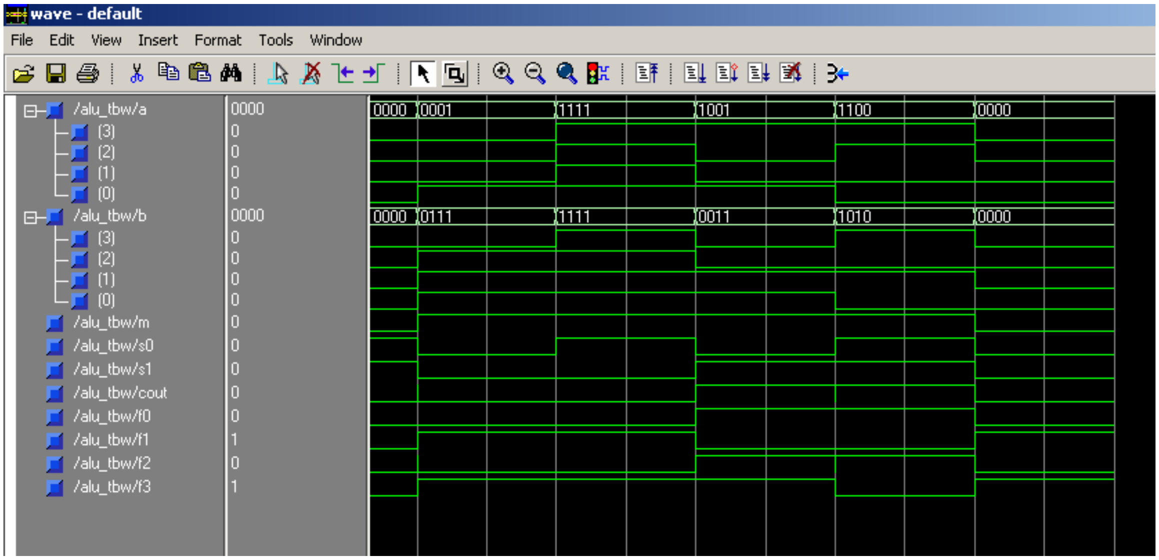
Full 4-bit ALU



The XOR gates allow for the ability to subtract a(3:0) from b(3:0). Create a test bench called **alu_tbw.tbw** to test the functionality of our ALU. You should come up with appropriate input combinations to demonstrate the functionality specified by the 2 tables above.

3. Simulate and make sure your ALU functions correctly. Below is an example:

The complete circuit simulation result



DELIVERABLE: Once you have finished the lab, you should see the Lab TA during his Lab hours (Monday/Thursday, 4-7 pm, in CO-40) and demonstrate your work.

DEADLINE: Thursday, September 23rd, 7pm.