ECE 332

Digital Electronics and Logic Design Lab

Laboratory and Experiment Guide
Spring 2007

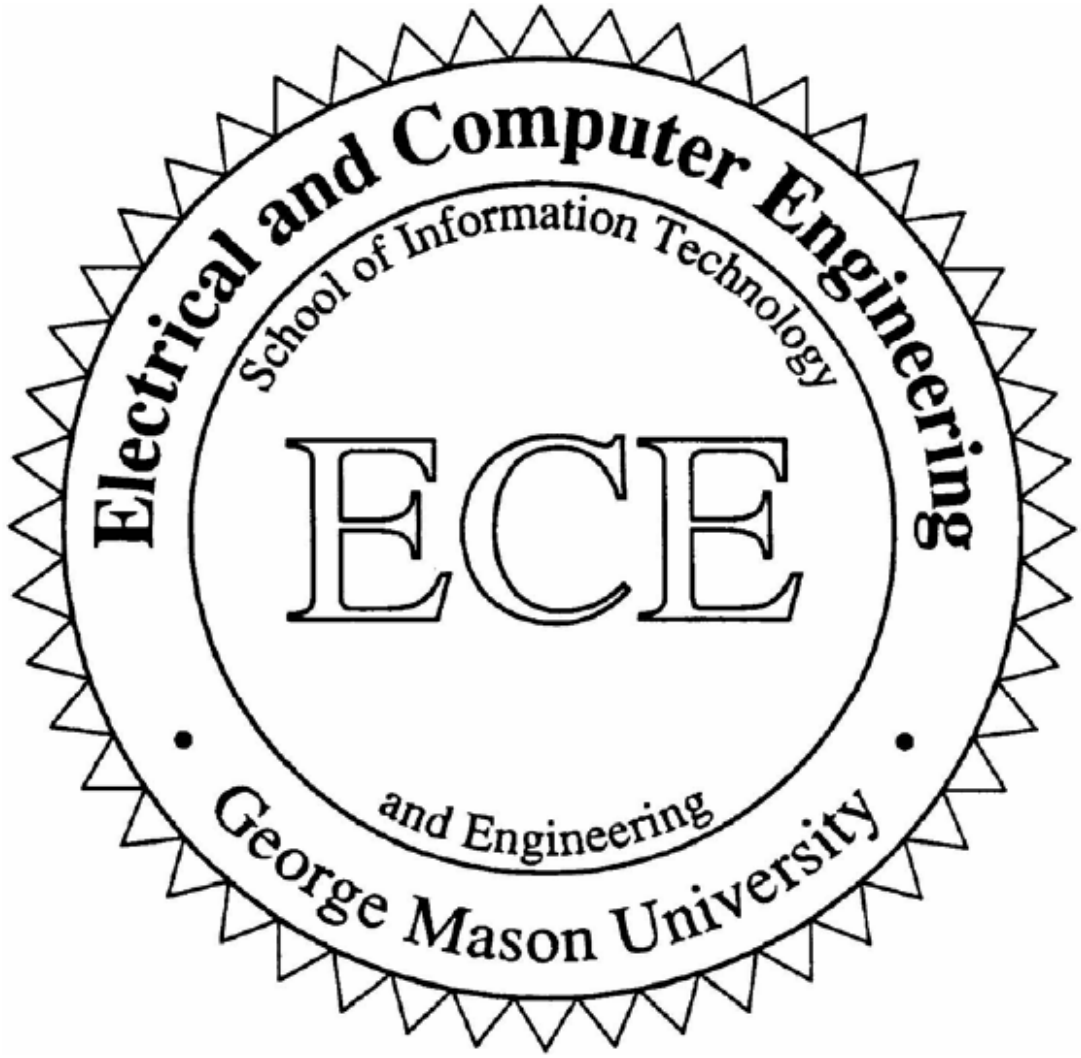


Table of Contents

Electronics Safety……..3
Laboratory Information…….4
Laboratory Rules……...5
Lab Tips…….6
Lab Report Outline……....8
Parts List……9
Aldec Active HDL Tutorial…….10
Resources…….27

Spring 2007

Experiment #	Date Completed	Student Signature	TA Signature
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
Bonus			

Electronics Safety

Exercise of good judgment and knowledge will ensure you a safe laboratory experience.

Do not defeat any safety device such as a fuse or circuit breaker, by shorting across it or by using a higher amperage fuse than that specified by the manufacturer.

Avoid direct contact with any voltage source. Do not wear rings, watches, bracelets, or dangling necklaces while working on equipment. Do not grasp any exposed metal in your circuit when the power is on.

Keep hands dry. Water and perspiration increase conductivity.

Wear shoes with insulating soles.

Measure voltages with one hand held behind you or in your pocket.

Avoid eye injury when cutting off excessive wire lengths. Point the wires downward toward your table top so the cut pieces cannot fly toward your eyes or another person's.

Shut off the power when connecting components or test equipment to a circuit. Double check your wiring before you apply power.

Make sure your circuit is properly grounded. Beware of a possible floating ground. It is a good idea to connect all grounds together before applying power.

To prevent power terminals from shorting, keep the leads coming from those terminals apart.

Your exercise of common sense, safety precautions and knowledge will help you avoid the dangers of electricity. The amount of current required to become lethal depends upon:

1. the person involved and state of health,
2. area of the body involved,
3. length of time the shock is received, and
4. type of electrical current.

Severe electrical shock will cause burns and/or paralysis. A small current passing through the chest can kill. With even minor electrical shock, some people react by going into traumatic shock.

In case of accident, turn off power immediately and call 911. If you suspect someone is touching a "live" wire, do not touch them. Use something non-conductive to push, rather than pull, them away from the wire. An injured person should be kept lying down until medical personnel arrive, and should be kept warm to help prevent traumatic shock. Be sure nothing is done to cause further injury.

Laboratory Information

Sue Davies
Instructional Laboratories Manager
Room 120D, Science and Technology I
(703) 993-1608
email - sdavies@gmu.edu

Purchase all laboratory kits from the above. Replacement parts for all of the labs are available. Special order kits will be filled, time permitting. Room 120D houses a fairly complete inventory available to the University community. It is non profit, operated as a service to our students. It means considerable savings when purchasing parts for your Engineering senior projects. Juniors, NOW is the time to begin planning your senior project.

Also located in the Manager's Office is a reference library including data books (not available in the main library), magazines, hardware, software and equipment catalogs. There are reference books of manufacturer information (address, phone no., etc.). Magazines may be checked out. Data books may only leave the office if you first leave a deposit.

Equipment repair is handled from this office. Please notify me of malfunctioning equipment. Leave a note with symptoms or bring the equipment to my office.

All Teaching Assistants have mailboxes in ST II, in the hallway near room 208. Leave messages there, not under the lab doors. Many students use the labs and your TA likely will not receive material pushed under doors. Hand assignments and anything of value directly to the TA's since the mailboxes aren’t in a secure area. Email is the best way to contact your TA. Be sure to activate your University account to receive all official notices.

Teaching Assistants' office hours will be posted in ST I on the room 2 entry way bulletin board, and room 120D entry door. In ST II, hours will be posted on the bulletin board across from room 230 (ECE office), and on the door of rooms 203 and 265. Whenever anyone is having office hours, it is also open lab time in that area.

The Engr. Computation and Test Lab needs ECE and CpE student volunteers to be Lab Monitors. By signing up for three hours per week to open and monitor the lab, volunteers obtain a door code and priority on one of the newest computers there. They have 24 hr./day, 7days/week access. This requires serious commitment for the semester. Successful volunteers put this work on their resumes.

Laboratory Rules

1. There will be **NO FOOD OR DRINKS** in the laboratory at any time. Students will be held liable for any damage to equipment resulting from abuse of this rule.

2. Students are not allowed in the laboratories without a Lab Instructor or Lab Monitor present, unless signed in with the Lab Manager. Open lab times for make-up or project work will be posted. When a Teaching Assistant is holding office hours, he/she is also monitoring an open lab which any student may use. ECE/CpE students have priority in the Computation and Test Lab, Room 265, ST II.

3. If you suspect a problem with the equipment, notify the TA or Room Monitor. Then, leave a note on it with a brief description of the problem/symptoms, or bring the equipment to the Lab Manager, Room 120D, ST I.

4. Handle equipment with care. Equipment out for repair means less available for your use.

5. **YOU** are responsible for leaving your workstation clean and in good condition when you leave. Failure to do this will negatively impact your final lab grade.

Before leaving:

a. Hang up all test leads neatly under appropriate connector combination.

b. Tidy workstation.

c. Throw away all trash.

d. TURN OFF equipment and lab table SWITCHES.

This is a non-smoking university. This building has **NO** designated smoking areas so you must go outdoors if you choose to smoke.

Lab Tips

1. Instruction manuals for the laboratory equipment may be checked out in room 120D, S&T I. The oscilloscope manual is available for purchase in the book store. It is essential that you become familiar with the correct way to use the basic equipment in your first lab course. Wire cutters/strippers are available for sale or you may bring your own.

2. Twenty-two gauge wire (22AWG) is the best size to use with the trainers and solderless breadboards. Solid wire only, never stranded, is used. There are spools of wire cabled to the back shelf or table. You will need to cut some and strip the insulation at both ends.

   Keep jumper wires short. Strive for a neat and logical layout. This will make troubleshooting easier and successful circuits more likely. (See ex. posted in lab.)

   Strip no more than approximately 3/8 inch of insulation from your jumper wires. Exposed wire increases the risk of short circuits.

3. Using more than one color of wire will help you debug your circuits. Normally RED and BLACK are reserved for power and ground.

4. Probe tips should not be inserted into the solderless breadboard. Wrap a wire around your probe hook tip twice for stability and insert the other end into the connector block. Alligator clips on the equipment leads also need a wrapped wire for connection to the trainers and breadboards.

5. Always ground your probe, but keep the ground wire short. Use the method of wire wrapping described in the previous tip to attach to the alligator clip of the probe ground wire.

6. For any potentiometers (such as most multiturn) which require adjusting, a trimpot tool is available for purchase from room 120D, S & T I. It is included in appropriate kits.

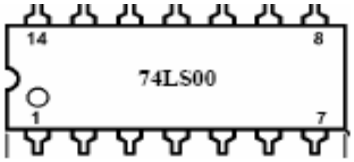
7. Most of the chips used in your lab are not overly static sensitive. However, you should observe some precautions when handling them. If you were issued a tube, it protects the chips from static charges, and is sturdy enough to provide protection to the delicate, metal pin legs.

8. Keep your chips away from magnets, motors and high temperatures. Don’t leave them in your car in the sun or extreme cold. They do best in the moderate temperatures most humans prefer.

9. Bent pins may be gently straightened with fingernails or needlenose pliers. If the pins break, you will need to purchase a new IC.

10. A small, narrow-blade, flathead screwdriver is useful in removing not so sensitive chips from breadboards. Using a side-to-side rocking motion as you insert the blade under the chip, keep a finger lightly on top of the chip to prevent it suddenly popping up on one end, bending the pins.

11. To locate pin 1 on an integrated circuit (IC, chip), look for one of the following:



• A semicircle at one end, cut into the top layer of the chip - With this at the top, pin 1 is at the left of the half circle.
• A tiny spot in the corner of the chip beside pin 1 -There may or may not be other marks on the chip.
Always count pins from pin 1 around the chip so that the last pin is straight across from pin 1. Common chip sizes are 8 pin, 14, 16, 18, 20, 24, 28, and 40 pin. Your TTL ICs (Transistor-Transistor Logic Integrated Circuits) will be in dual inline packages DIP).

12. Chip leads are slightly flared to help hold them in printed circuit boards while being soldered. You will need to reduce the flare to allow them to be inserted into the breadboard. Use a pair of needlenose pliers, or press the leads against a table top, while rotating the body of the chip towards the lead points to reduce the lead angle of all evenly. The pin legs should be at a 90 degree angle to the plastic package base. Don’t bend too far; there is no easy correction.

Lab Report Outline

The lab reports for the experiments in the ECE 332 Lab (as applicable) must contain the following information:

- Title Page
  - Title & Number of the Experiment
  - Course & Section Number
  - Name
  - Date
  - Signed Honor Code Pledge

- Brief Description of the Experiment
  - Approach used if the experiment is a word problem
  - Description of the techniques (QM, K-maps etc.) used to arrive at the final logic expression

- Software Implementation of the Logic Expression
  - Compiled version of the VHDL source code
  - Compiled version of the test bench
  - Simulation results (Timing Diagrams)

- Hardware Implementation of the Logic Expression
  - List of IC chips used to implement the logic expression
  - Detailed circuit diagram (including pin numbers) of the IC interconnections
  - Print outs of the outputs of the experiment

- Conclusion
  - Comparison between the hardware and software implementations

A lab report template and sample report (in MS Word format or PDF) can be found on the ECE 332 Administration web page.

1/26/2007Parts List

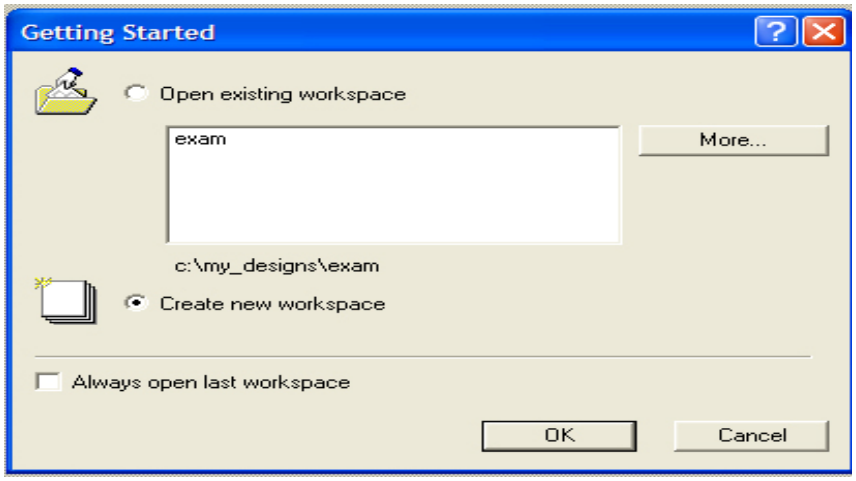
ECE 332 Laboratory Parts List

- 1 74LS00 Quad 2 input NAND gate
- 1 74LS02 Quad 2 input NOR gate
- 1 74LS04 Hex Inverter
- 1 74LS08 Quad 2 input AND gate
- 1 74LS10 Triple 3 input NAND gate
- 1 74LS11 Triple 3 input AND gate
- 1 74LS32 Quad 2 input OR gate
- 2 74LS73 Dual JK Flip-flop with clear
- 1 74LS86 Quad 2 input XOR
- 1 74LS151 8 to 1 Multiplexer
- 1 74LS155 Dual 2/4 Demultiplexer (Decoder)
- 1 74LS157 Quad 2 to 1 Multiplexer
- 1 74LS163 4-Bit Binary Counter
- 1 74LS283 4-Bit Adder
- 1 74C08 Quad 2-input AND gate

ALDEC ACTIVE HDL TUTORIAL

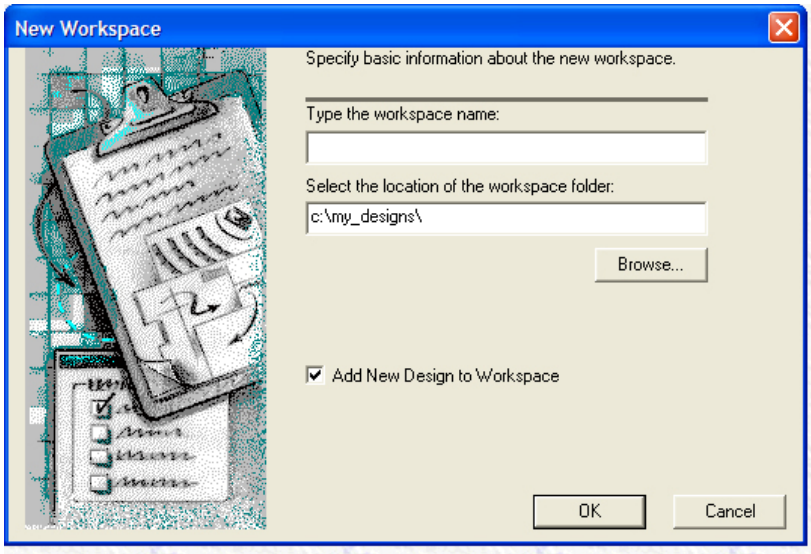
Start-up

1. Double click on the Active HDL icon on the desktop; it comes up with a getting started window.

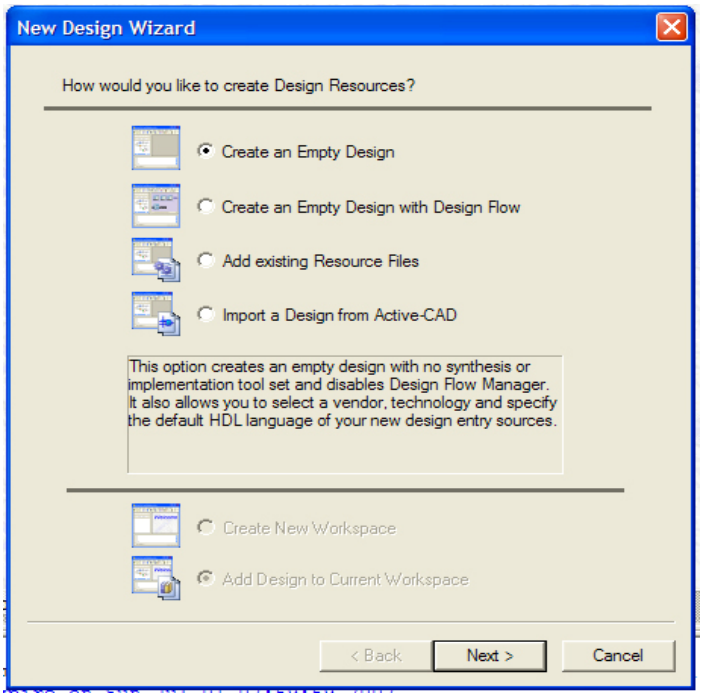


a. Select **create new workspace** and click **OK**, creates new workspace in your directory.
b. Selecting **open existing workspace** gives you the option to choose from your previous Workspace's.
c. Always open last workspace takes you to the last workspace.

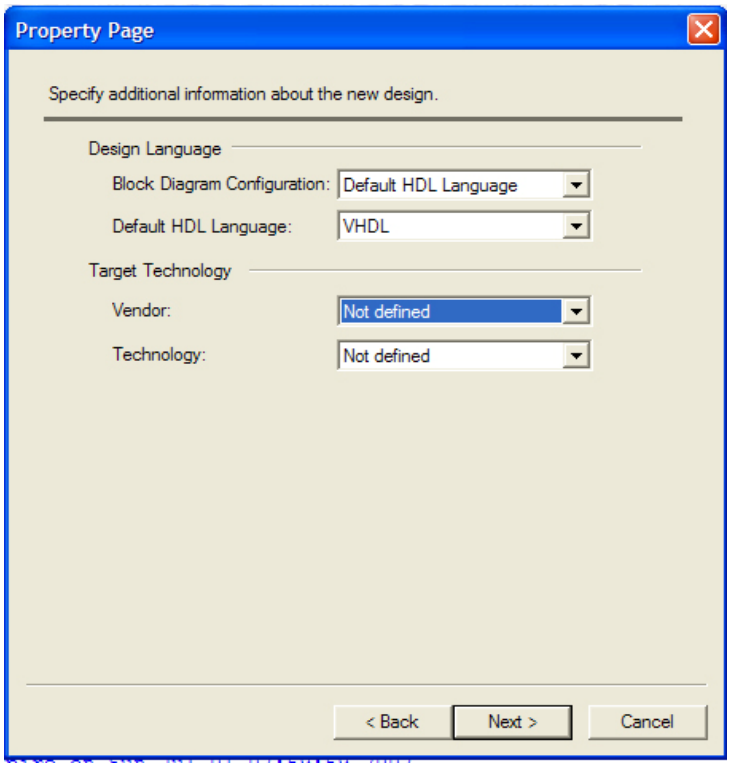
2. Type the workspace name you want to create and click **OK** and the default location of your workspace will be your **network drive\my_designs.**



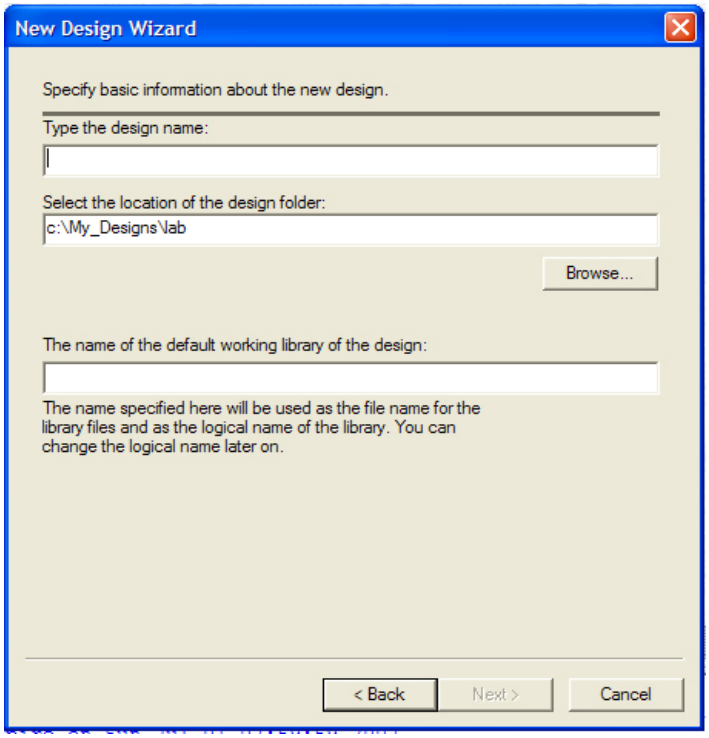
3. Select **create an empty design**, creates a new design in your workspace and click **Next.**



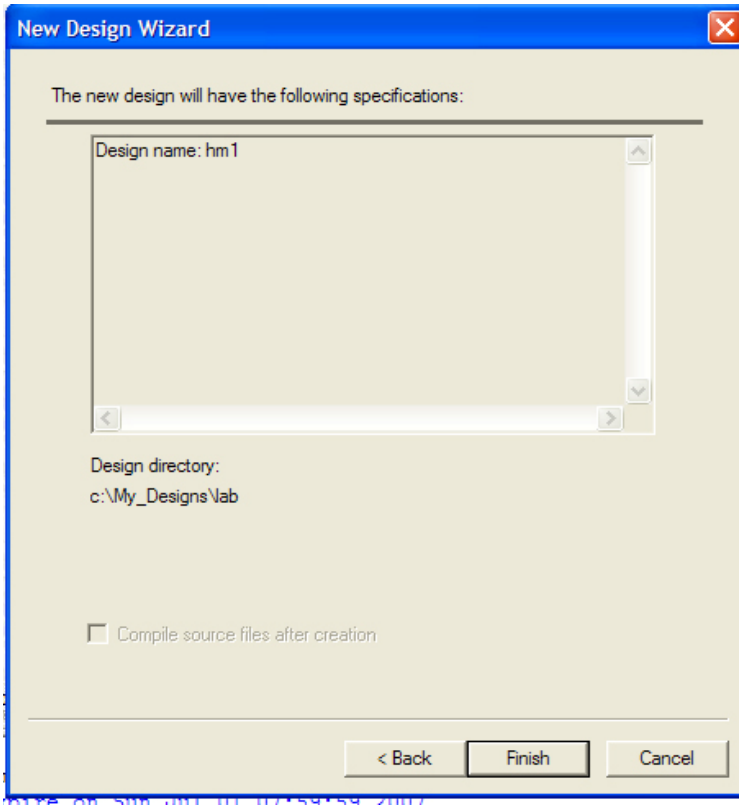
4. Choose block diagram configuration as **Default HDL language** and Default HDL language as **VHDL**; Leave the Target Technology blanks as not defined; Click **Next.**



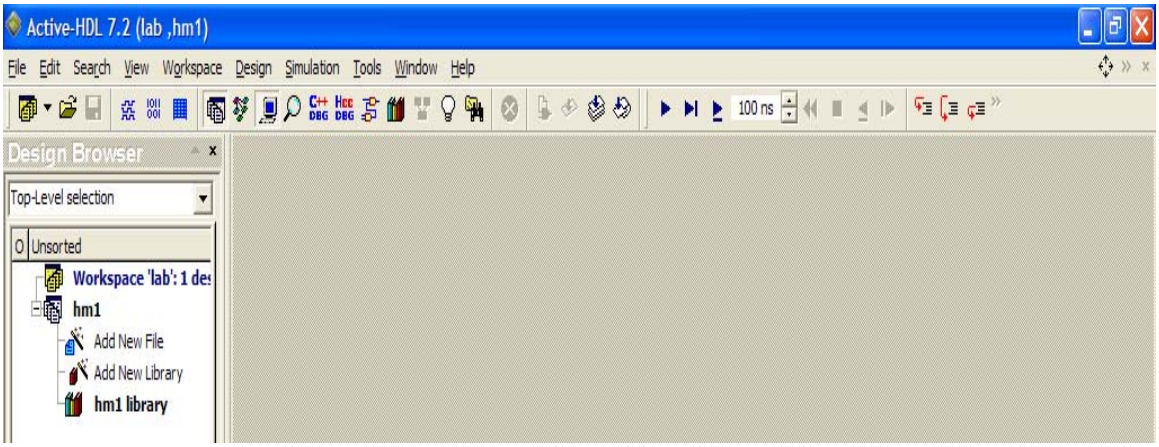
5. Type the design name you want to create and Click **Next** and the default location of your design will be your **network drive\my_designs\workspace name.**



6. Design name and design directory will be displayed proceed further by clicking **Finish.**

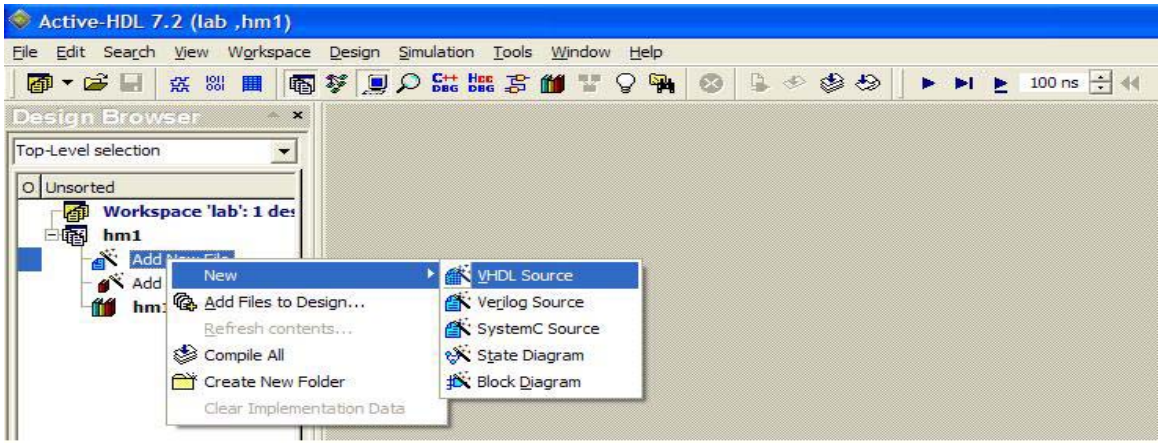


7. To the left workspace and new design are displayed.

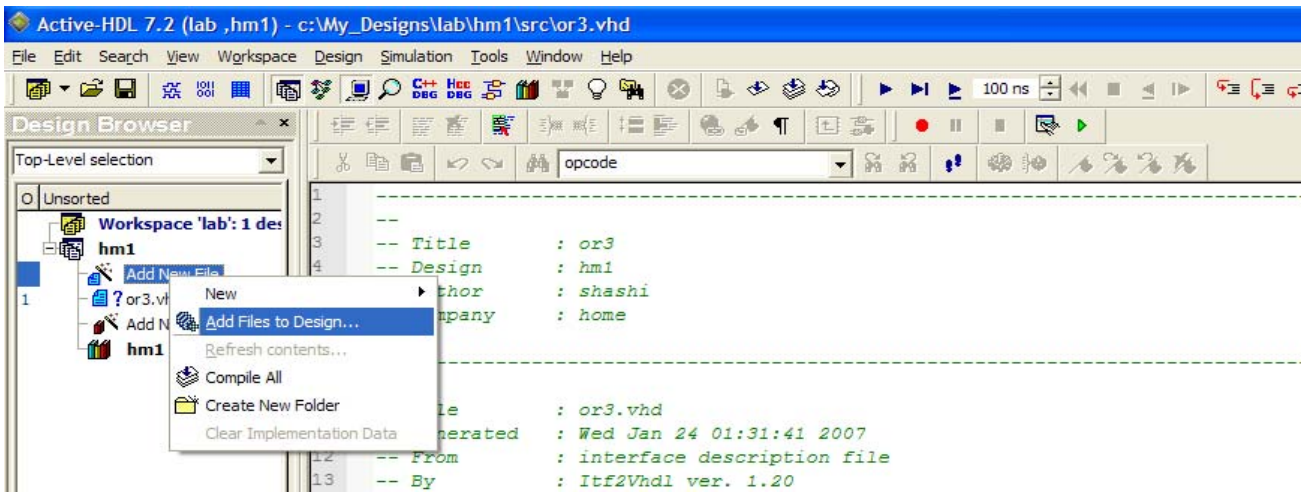


8. To start with your VHDL source right click on **Add New File > New > VHDL source,** then **GO TO 9.**

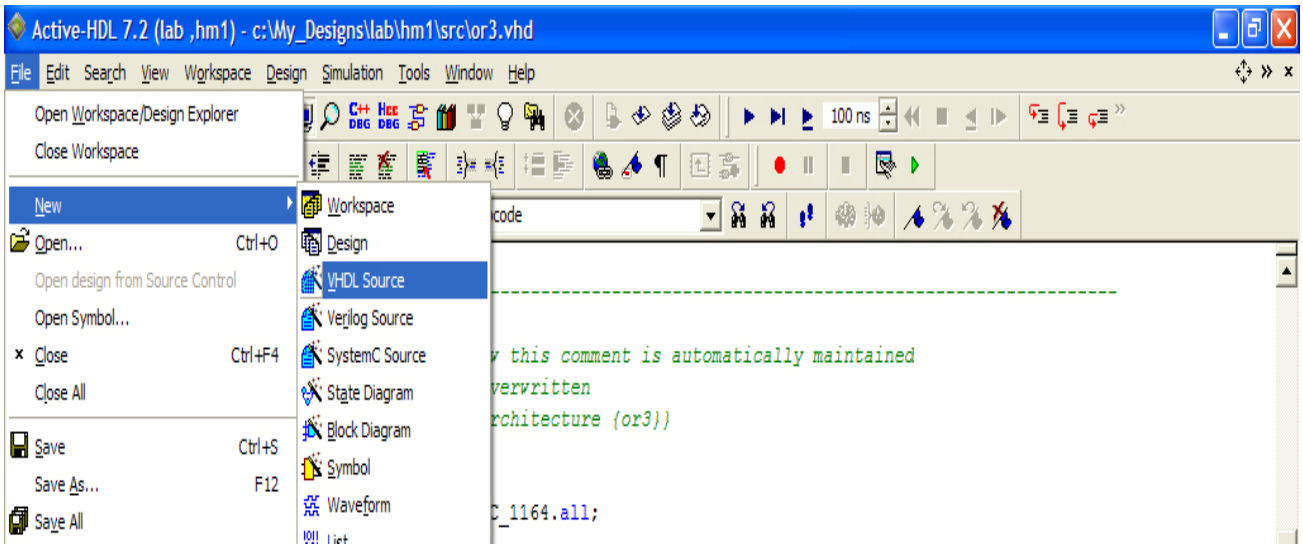
To add existing files to your design **GO TO 8A.**



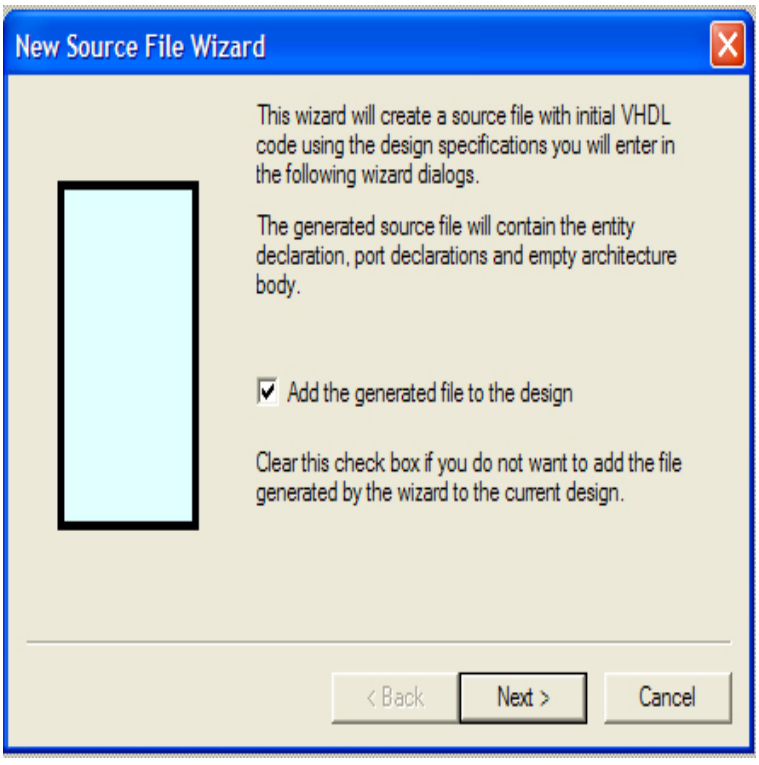
8A. To add existing files to your design, select **Add New File> Add Files to Design.**



8B. New VHDL source can also be added by following up **File > New > VHDL source.**

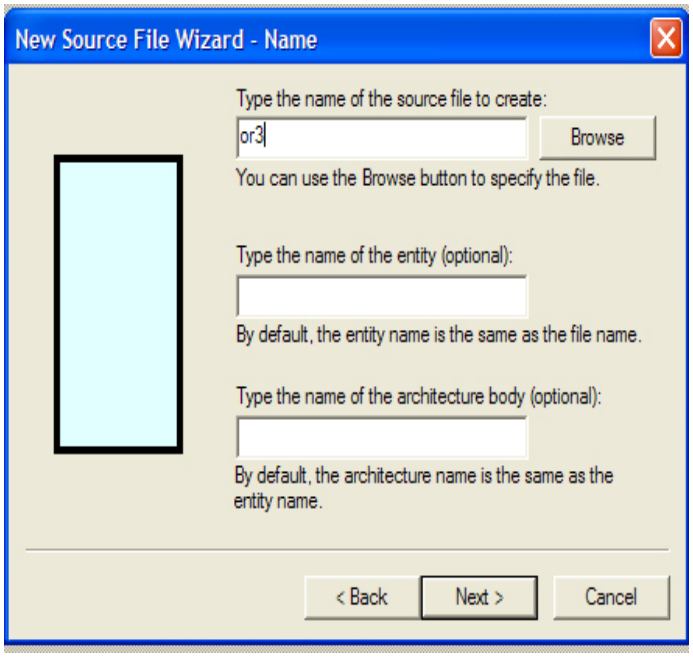


9. Click **Next** on the new source file wizard.

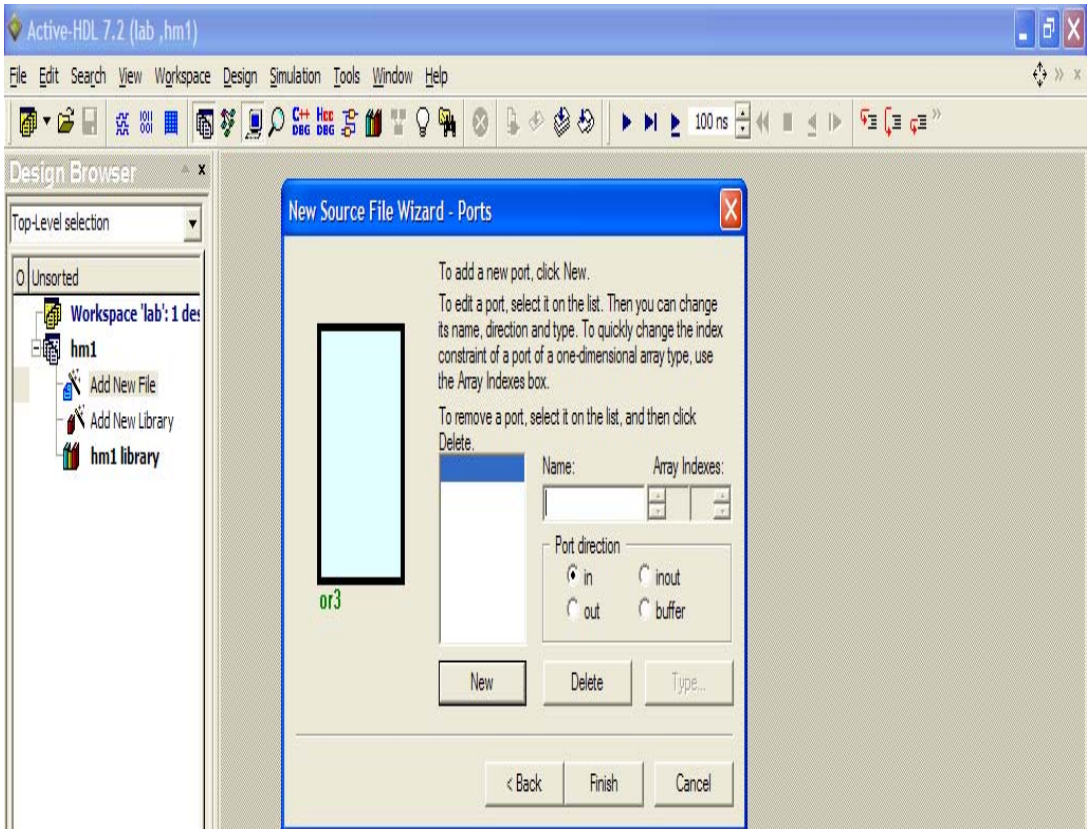


.

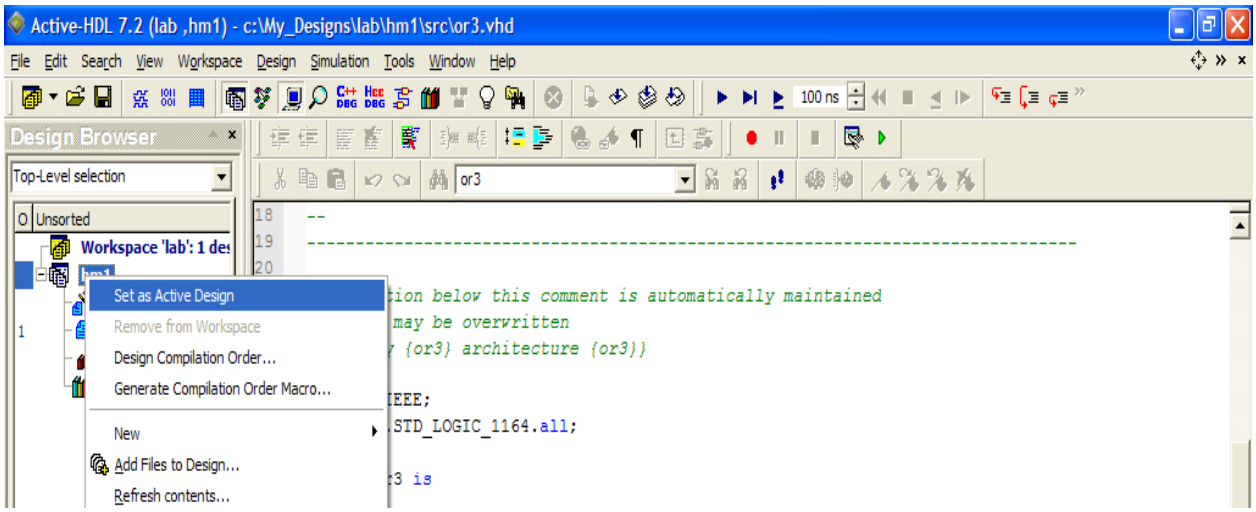
10. Type the source file name you want to create; you can give a different name for entity and architecture. Click **Next.**



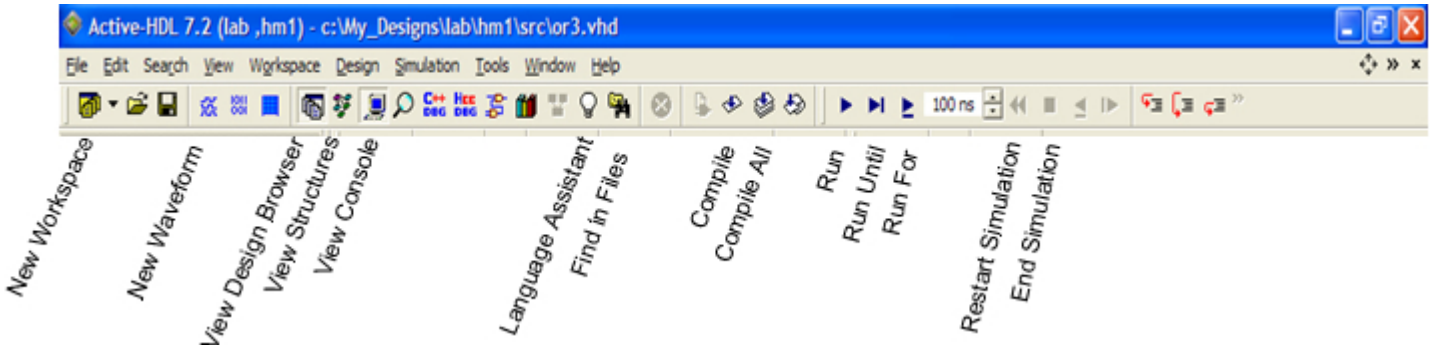
11. Select **New** in the New Source File Wizard-Ports, start declaring your input and output ports by naming them, choosing the port direction, choose the type by indicating the array indexes. Click **Finish**.

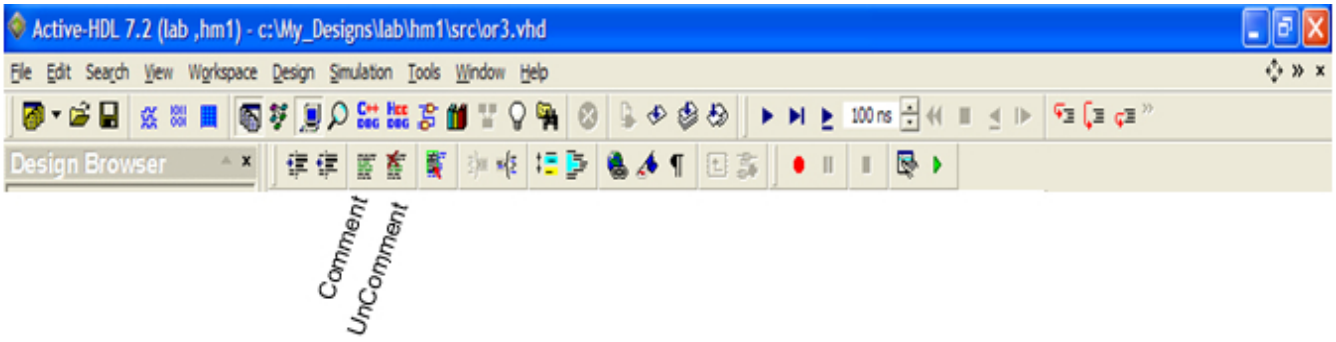


12. If you have multiple designs in the same workspace you need to set an active design and this can be done by a right click on your design and select **Set as Active Design.**

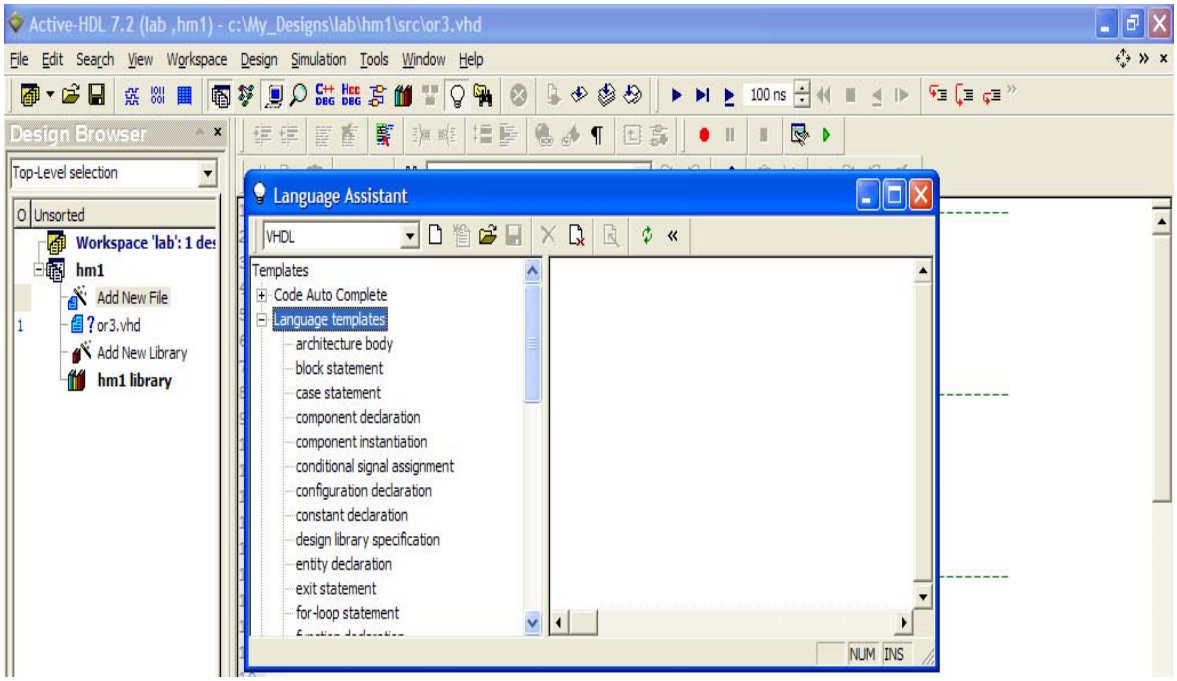


Main Window Toolbars



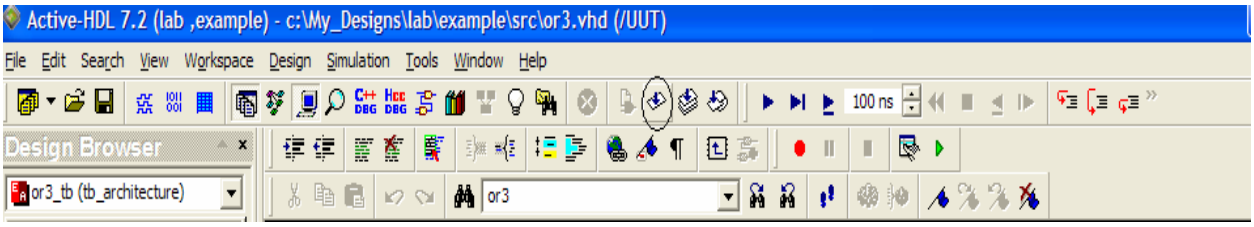


Language Assistant helps you with the syntax of the VHDL data types.



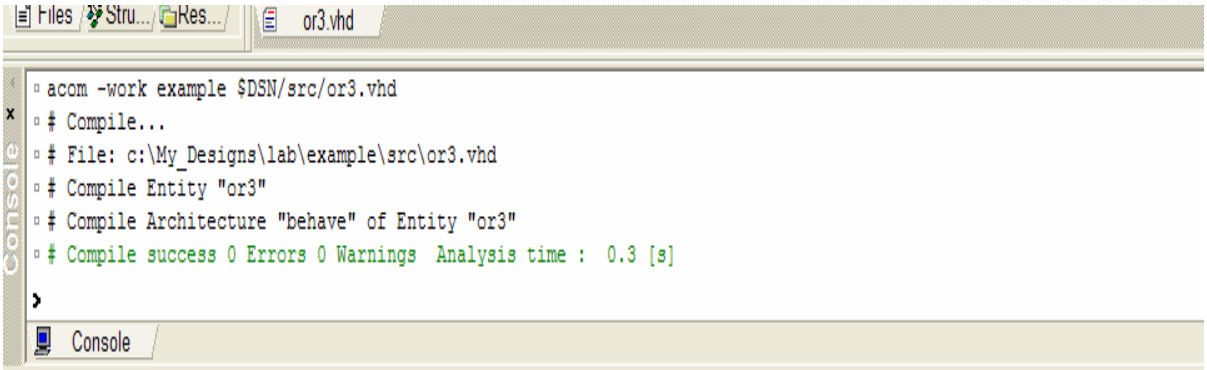
Compiling the Design

Once the design is ready, compile it either by clicking on the toolbar as shown or hit **F11** or choose **Design > Compile.**



Console window

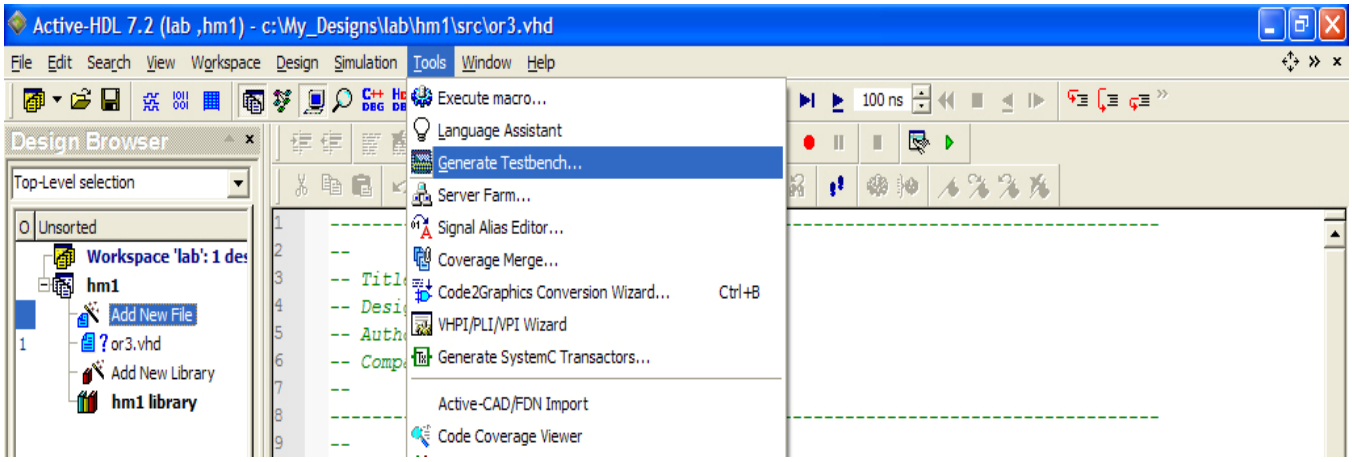
Console window appears at the bottom of the page with a **success** message if design is free of errors.



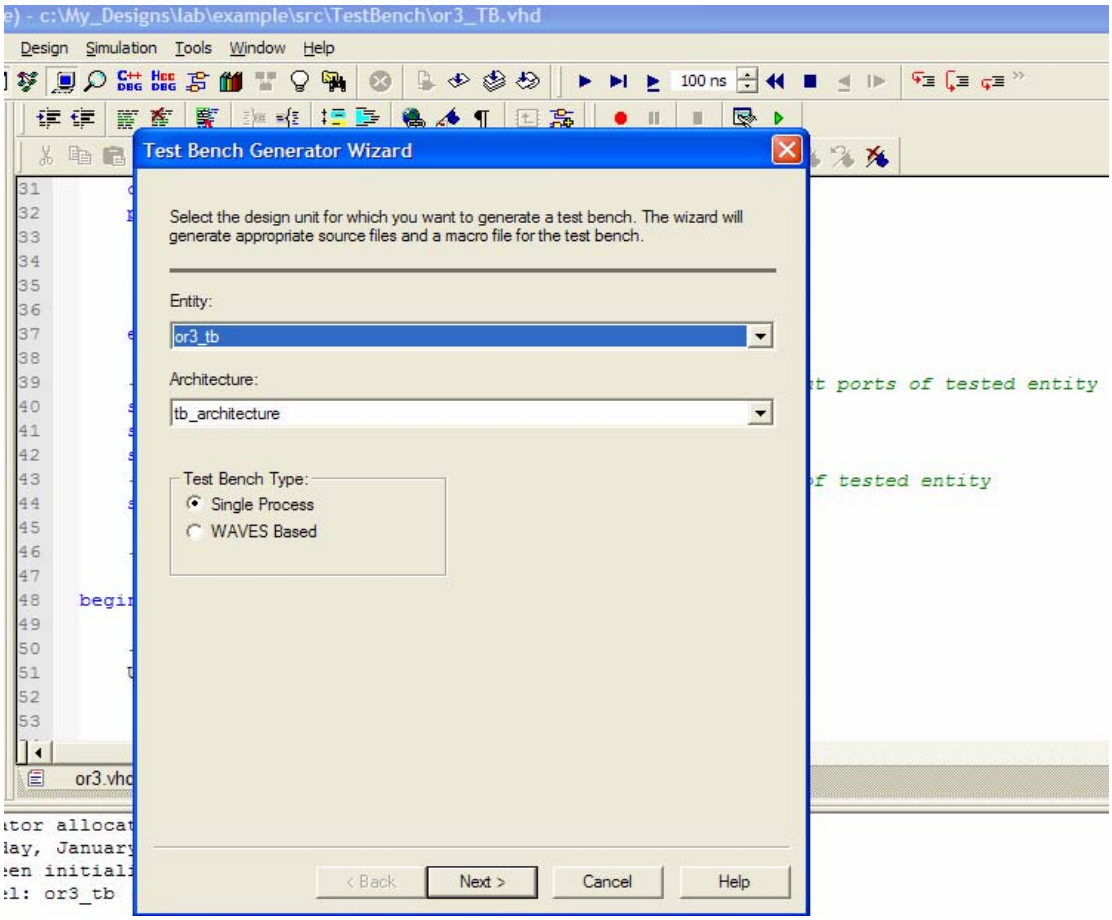
Generating Testbench

NOTE: To add your testbench, click **Add New File>Add Files to Design** as in 8A.

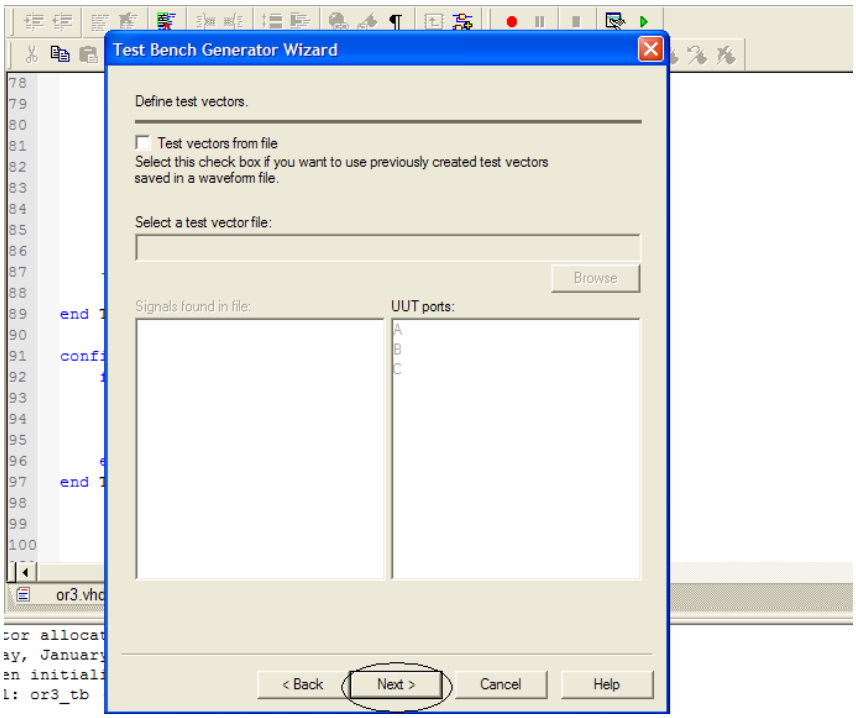
Testbench for the design can be generated as **Tools > Generate Testbench.**



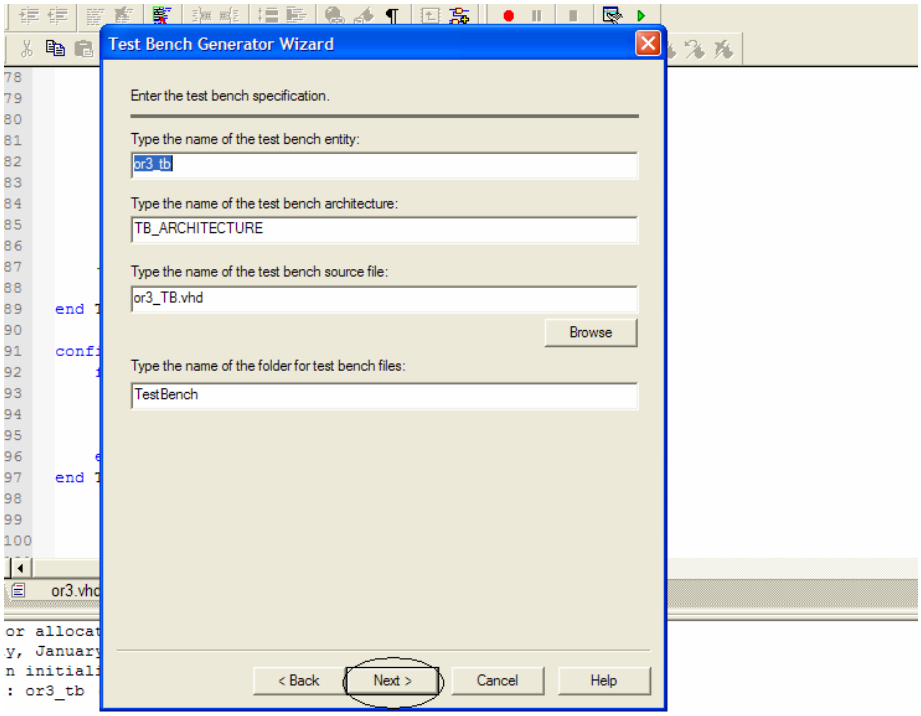
Select the Design Unit (Entity and Architecture) for which testbench should be generated and select testbench type as **Single Process;** Proceed by clicking **Next.**



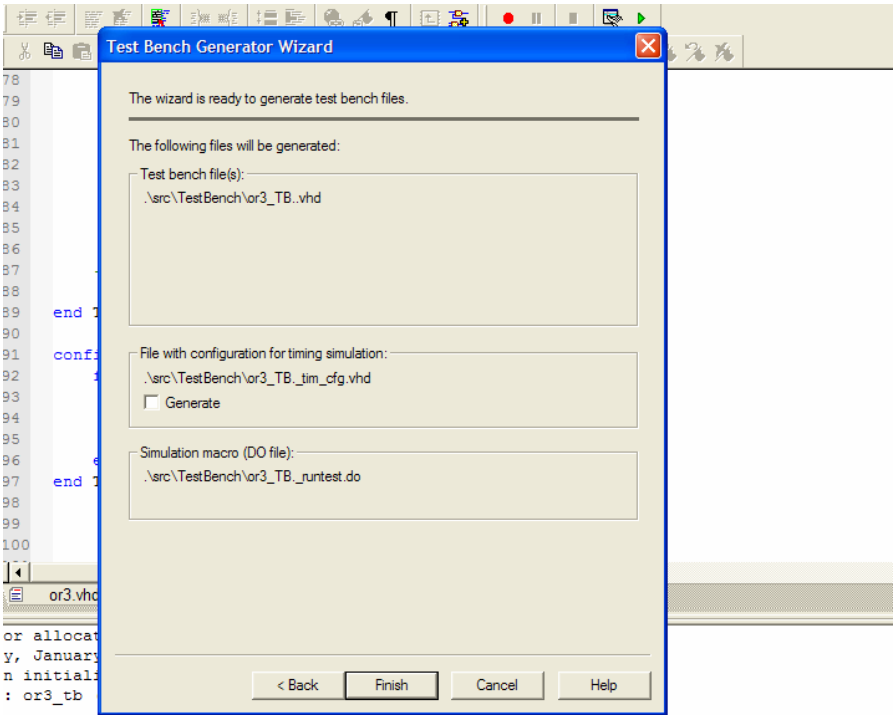
Testbench generator wizard appears with all your input ports under UUT ports; proceed By clicking **Next.**



Click **Next** as the tool generates the names for all the design units, you can change them (Not Recommended).



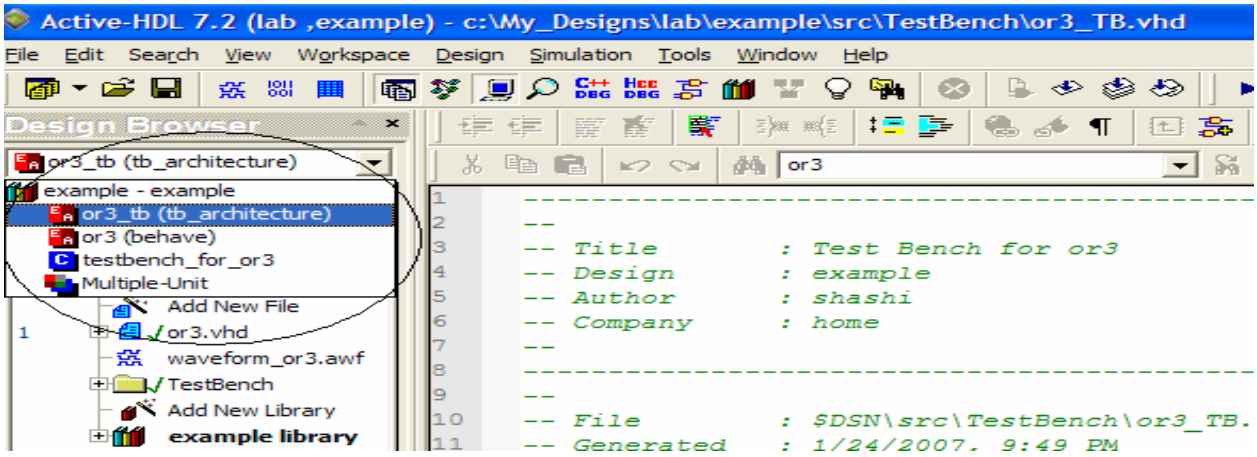
Click **Finish** to proceed, the window below shows the files generated by the Testbench.



NOTE: Add your test conditions to the Testbench, **Hit F11** to compile.

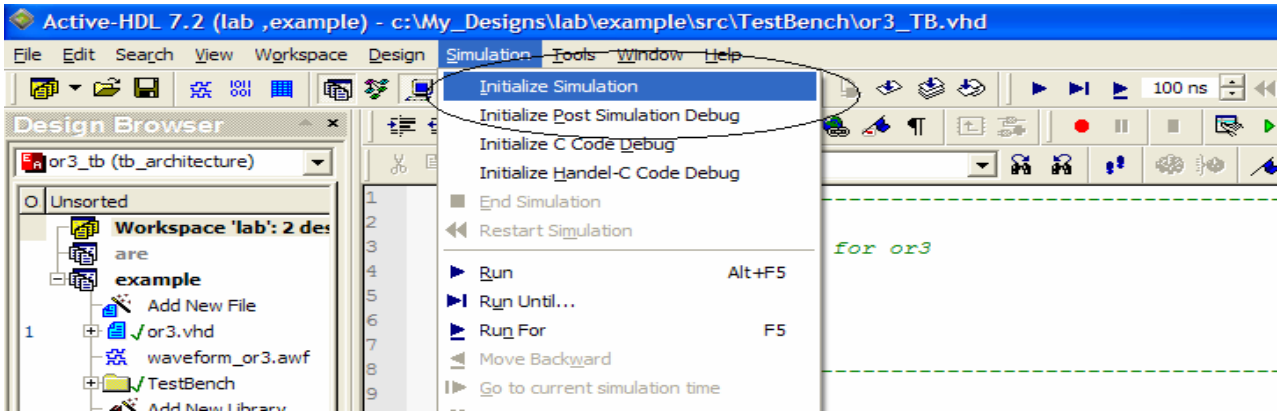
Getting Started with the Simulation

Before You simulate, select the **Top level** unit as your testbench as shown below in the design browser.

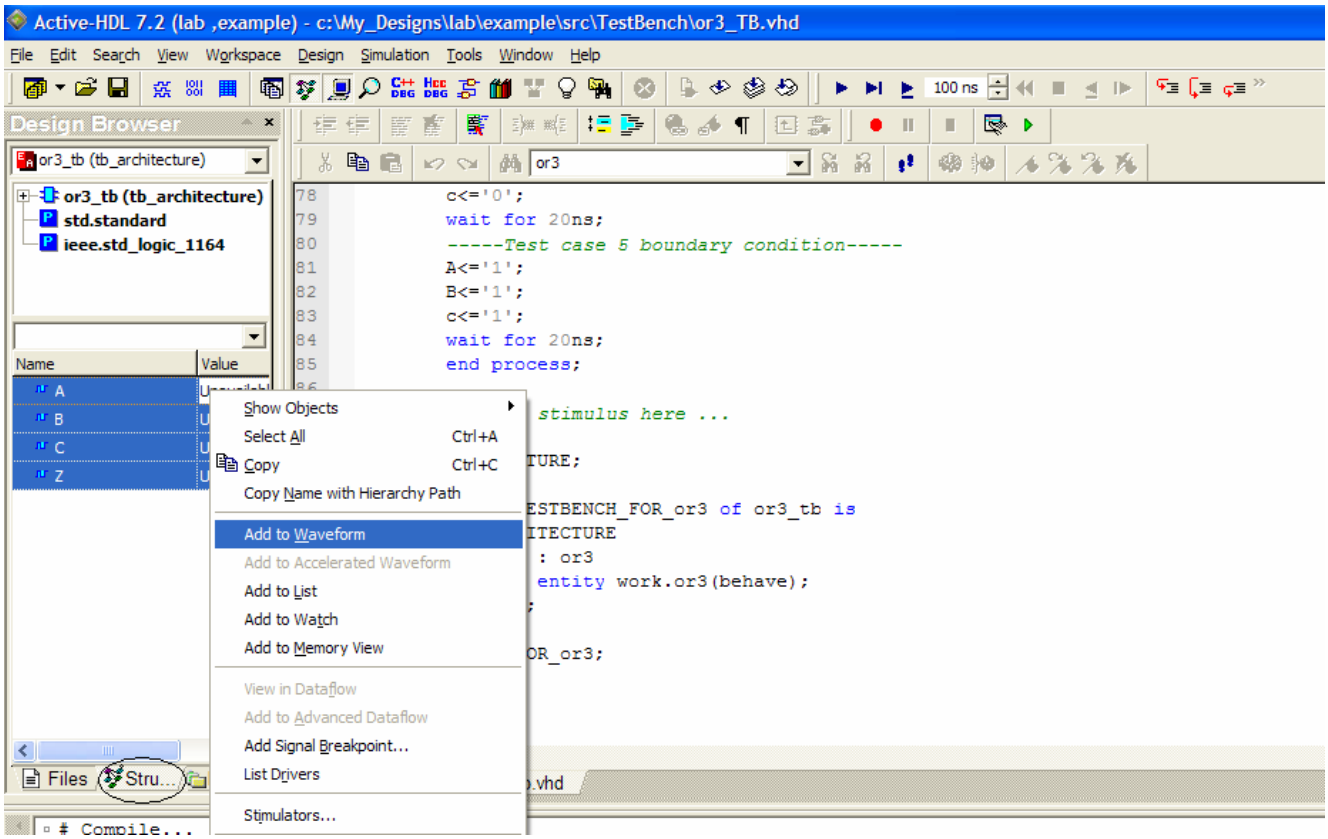


NOTE: All files must be compiled before you select **Top level** unit.

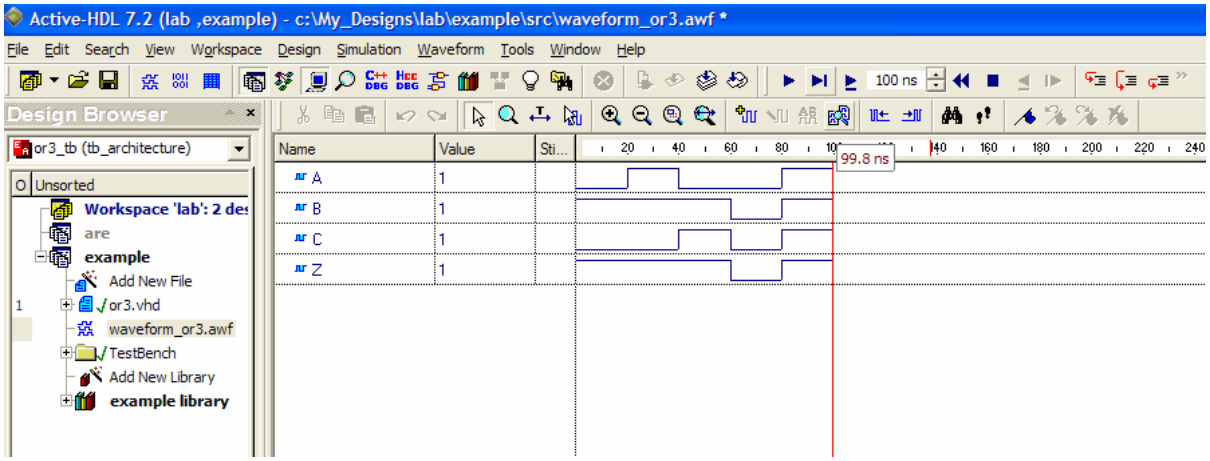
Initialize simulation by selecting **Simulation > Initialize Simulation.**



Click **Structures** (circled below) to see all the ports, select them and add to the waveform as shown below.



Hit F5 to run or select **Simulation > Run For.**



Wave Window Toolbar

Creating/Editing VHDL Files

Or3.vhd

```
------------------------------------------------------------------
-----------------
--
-- Title       : or3
-- Design      : example
-- Author      : shashi
-- Company     : home
--
------------------------------------------------------------------
-----------------
--
-- File        : or3.vhd
-- Generated   : Wed Jan 24 21:40:37 2007
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.20
--
------------------------------------------------------------------
-----------------
--
-- Description :
--
------------------------------------------------------------------
-----------------

--{{ Section below this comment is automatically maintained
--   and may be overwritten
--{entity {or3} architecture {behave}}

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity or3 is
_ port(
_ A : in STD_LOGIC;
_ B : in STD_LOGIC;
_ C : in STD_LOGIC;
_ Z : out STD_LOGIC
_     );
end or3;

architecture behave of or3 is
begin

Z <= A or B or C;_

end behave;
```

Or3_TB.vhd

```
-------------------------------------------------------------------
-----------------
--
-- Title       : Test Bench for or3
-- Design      : example
-- Author      : shashi
-- Company     : home
--
-------------------------------------------------------------------
-----------------
--
-- File        : $DSN\src\TestBench\or3_TB.vhd
-- Generated   : 1/24/2007, 9:49 PM
-- From        : $DSN\src\or3.vhd
-- By          : Active-HDL Built-in Test Bench Generator ver.
1.2s
--
-------------------------------------------------------------------
-----------------
--

-- Description: Automatically generated Test Bench for or3_tb
--
-------------------------------------------------------------------
-----------------

library ieee;
use ieee.std_logic_1164.all;

_-- Add your library and packages declaration here ...

entity or3_tb is
end or3_tb;

architecture TB_ARCHITECTURE of or3_tb is
_-- Component declaration of the tested unit
_component or3
_port(
_A : in std_logic;
_B : in std_logic;
_C : in std_logic;
_Z : out std_logic );
_end component;

_-- Stimulus signals - signals mapped to the input and inout
ports of tested entity
_signal A : std_logic;
```

```
_signal B : std_logic;
_signal C : std_logic;
_-- Observed signals - signals mapped to the output ports of
tested entity
_signal Z : std_logic;

_-- Add your code here ...

begin

_-- Unit Under Test port map
_UUT : or3
_port map (
_A => A,
_B => B,
_C => C,
_Z => Z
_);
_process
_begin
_----Test case 1-----
_A<='0';
_B<='1';
_c<='0';
_wait for 20ns;__
_-----Test case 2-----
_    A<='1';
_B<='1';
_c<='0';
_wait for 20ns;
_-----Test case 3-----
_    A<='0';
_B<='1';
_c<='1';
_wait for 20ns;_
_-----Test case 4  boundary condition-----
_    A<='0';
_B<='0';
_c<='0';
_wait for 20ns;_

_-----Test case 5 boundary condition-----
_    A<='1';
_B<='1';
_c<='1';
_wait for 20ns;_
_end process;

_-- Add your stimulus here ...
```

```
end TB_ARCHITECTURE;


configuration TESTBENCH_FOR_or3 of or3_tb is
_for TB_ARCHITECTURE
_for UUT : or3
_use entity work.or3(behave);
end for;
_end for;
end TESTBENCH_FOR_or3;
```

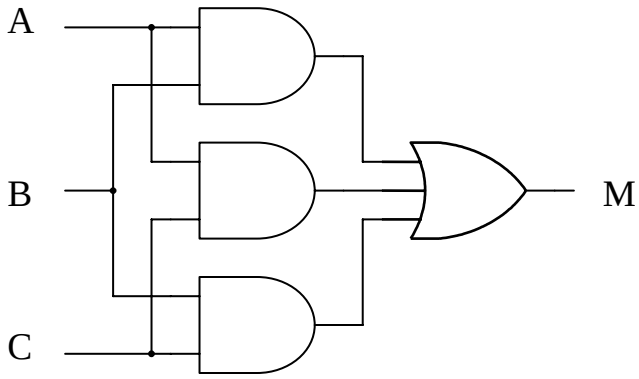
Resources

- ➔ Find and explore all relevant web pages starting at
  http://ece.gmu.edu/~jkaps/courses/ece331/index.html

- ➔ Find the additional VHDL files for future experiments at

- ➔ Find an additional Tutorial for VHDL at
  http://ece.gmu.edu/labs/Active_HDL.pdf

Experiment 1



Given the circuit diagram above:

1. Write an algebraic expression of the output as a function of the input signals.
2. Working VHDL code for this experiment is provided, so you can concentrate on using Model*Sim* and verifying simulation function equivalency:
   - Go to the ECE 332 Experiments page (http://ece.gmu.edu/~rhayne/ece332/experiments.htm) and save the source code for **ece332_gates.vhd**, **lab1.vhd**, and **lab1_testbench.vhd** into your `ece332_labs` directory (created when you completed the Mentor Graphics Tutorial).
3. Using `vsim` (as you did in the tutorial), import all three of your new vhdl files, compile them (ece332_gates first, then lab1, then lab1_testbench), and load the testbench.
4. Simulate and print the resulting waveforms for all possible inputs.
5. Draw a detailed diagram showing all the pin connections.
6. Implement the circuit in hardware, using the TTL devices from the parts list.

lab1.vhd

```
-------------------------------------------------------------
-- Author             : Dr. Ron Hayne
-- Date               : July 11, 2004
-- Course             : ECE332
-- File Name          : lab1.vhd
-- Design Units       : Lab1_Circuit
-- Purpose of Code    : Entity description and multiple
--                      architectures for lab1.
-- Hardware modeled   : Circuit diagram given for lab1.
-- Model Limits       : None known.
-- Known Errors       : None known.
-- Design Library     : work
-- Dependencies       : ECE332_Gates
-- Environment:
--   Simulator        : Mentor Graphics (ModelSim) V5.4d
--   Platform         : Unix (SunOS 5.8)
--
-- Change List (most recent on top)
--  Date    Who  What
-- -------  ---  ----
-- 11Jul04  rjh  Creation
-------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

entity Lab1_Circuit is
  port( A : in std_logic;
        B : in std_logic;
        C : in std_logic;
        M : out std_logic );
end Lab1_Circuit;

-------------------------------------------------------

architecture BEHAVE1 of Lab1_Circuit is
begin
  M <= (A and B) or (A and C) or (B and C);
end BEHAVE1;

-------------------------------------------------------
```

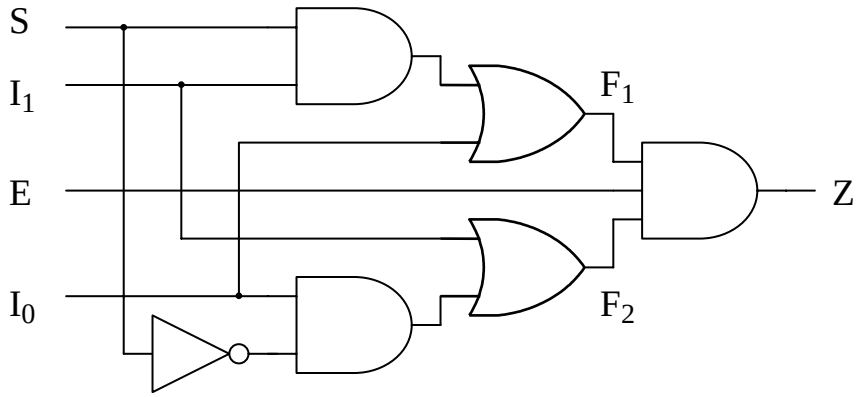
```
architecture BEHAVE2 of Lab1_Circuit is
begin
  process(A, B, C)
  begin
    if (A = '1' and B = '1') then
      M <= '1';
    elsif (A = '1' and C = '1') then
      M <= '1';
    elsif (B = '1' and C = '1') then
      M <= '1';
    else
      M <= '0';
    end if;
  end process;
end BEHAVE2;

-------------------------------------------------------------

architecture BEHAVE3 of Lab1_Circuit is
  signal Input: std_logic_vector(2 downto 0);
begin
  Input <= A & B & C;
  with Input select
    M <= '1' when "011",
         '1' when "101",
         '1' when "110",
         '1' when "111",
         '0' when others;
end BEHAVE3;

-------------------------------------------------------------

use work.ECE332_Gates.all;

architecture STRUCTURE of Lab1_Circuit is
  signal AB    : std_logic;
  signal AC    : std_logic;
  signal BC    : std_logic;
  signal OR1a  : std_logic;
begin
  AND1: AND_74LS08 port map(A, B, AB);
  AND2: AND_74LS08 port map(A, C, AC);
  AND3: AND_74LS08 port map(B, C, BC);
  OR1: OR_74LS32 port map(AB, AC, OR1a);
  OR2: OR_74LS32 port map(OR1a, BC, M);
end STRUCTURE;
```

lab1_testbench.vhd

```
------------------------------------------------------------------
-- Author            : Dr. Ron Hayne
-- Date              : July 11, 2004
-- Course            : ECE332
-- File Name         : lab1_testbench.vhd
-- Design Units      : Lab1_Testbench
-- Purpose of Code   : This is a top level testbench for
--                     Lab1_Circuit
-- Hardware modeled  : This testbench creates an instance of the
--                     Lab1_Circuit and then drives the inputs
--                     and checks the results.
-- Model Limits      : VHDL-93 Syntax (report)
-- Known Errors      : None known
-- Design Library    : work
-- Dependencies      : Lab1_Circuit
-- Environment:
--   Simulator       : Mentor Graphics (ModelSim) V5.4d
--   Platform        : Unix (SunOS 5.8)
--
-- Change List (most recent on top)
--  Date    Who  What
-- -------  ---  ----
-- 11Jul04  rjh  Creation
------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity Lab1_Testbench is
end Lab1_Testbench;

architecture Testbench of Lab1_Testbench is

  -- Signal Declarations
  signal Input : std_logic_vector(2 downto 0) := "000";
  signal M     : std_logic;

  -- Component Declarations
  component Lab1_Circuit
    port( A : in std_logic;
          B : in std_logic;
          C : in std_logic;
          M : out std_logic );
  end component;
```

```
begin
  -- Component Instantiation
  Tested_Circuit : Lab1_Circuit port map( A => Input(2),
                                          B => Input(1),
                                          C => Input(0),
                                          M => M );

  -- Cycle through test vectors and evaluate the results
  process
  begin
    wait for 100 ns;
    case Input is
      when "011" | "101" | "110" | "111" =>
        assert (M = '1')
          report "Test Failed";
      when others =>
        assert (M = '0')
          report "Test Failed";
    end case;
    Input <= Input + "001";
  end process;
end Testbench;
```

Experiment 2



Given the circuit diagram above:

1. Write an algebraic expression of the output as a function of the input signals.
2. Fill in the truth table to determine the input/output relationship of the funcion.
3. Write a VHDL behavioral description to implement the circuit, including the following
   a. Entity declaration
   b. Behavioral architecture body
   c. **lab2_testbench.vhd** (provided on the ECE 332 Experiments page)
4. Simulate and print the resulting waveforms for all possible inputs.
5. Draw a detailed diagram showing all the pin connections.
6. Implement the circuit in hardware, using the TTL devices from the parts list.
7. Compare the results of the two approaches with respect to the timing characteristics of the outputs.

Truth Table

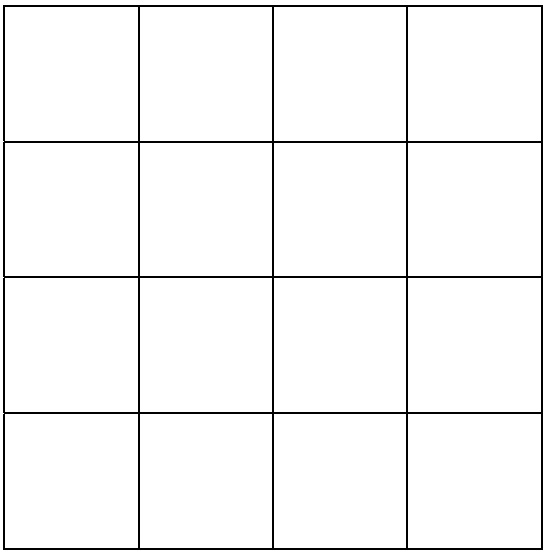
E	S	I_1	I_0	F_1	F_2	Z
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Experiment 3

Given the following function:

$$f(S,E,A,L) = \sum m(0,2,3,5,8,9,11,13,15) + d(1,4,10)$$

1. Using a K-map, determine a minimal sum-of-products covering of the function, using "don't cares" as appropriate.
2. Draw a gate level circuit diagram implementing the K-map derived function, using only NAND gates.
3. Write a VHDL structural description to implement your circuit at the gate level, using the pre-defined gate libraries. Be sure to including the following:
   a. Entity declaration
   b. Structural architecture body
   c. An appropriate testbench
4. Simulate and print the resulting waveforms for all possible inputs.
5. Draw a detailed diagram showing all the pin connections.
6. Implement the circuit in hardware, using the TTL devices from the parts list.
7. Compare the results of the two approaches with respect to the timing characteristics of the outputs.

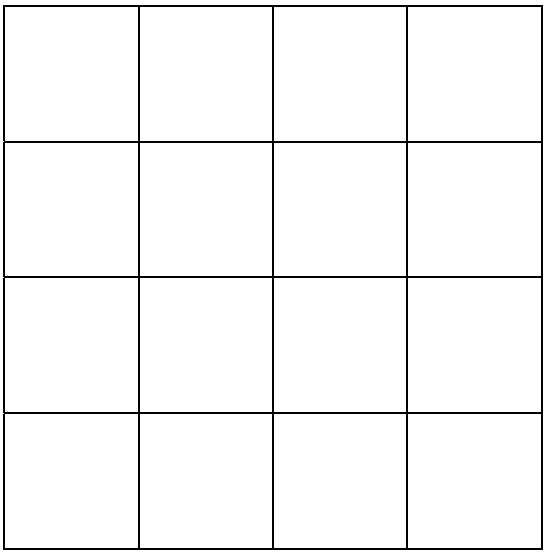


Experiment 3

Given the following function:

$$f(S,E,A,L) = \sum m(0,2,3,5,8,9,11,13,15) + d(1,4,10)$$

1. Using a K-map, determine a minimal sum-of-products covering of the function, using "don't cares" as appropriate.
2. Draw a gate level circuit diagram implementing the K-map derived function, using only NAND gates.
3. Write a VHDL structural description to implement your circuit at the gate level, using the pre-defined gate libraries. Be sure to including the following:
   a. Entity declaration
   b. Structural architecture body
   c. An appropriate testbench
4. Simulate and print the resulting waveforms for all possible inputs.
5. Draw a detailed diagram showing all the pin connections.
6. Implement the circuit in hardware, using the TTL devices from the parts list.
7. Compare the results of the two approaches with respect to the timing characteristics of the outputs.



Experiment 4

Due to uncertain economic times, you as the chief engineer of Patriot Enterprises, have decided to diversify your product line. Your first circuit is to be a combinational circuit to illuminate three lights indicating the status of the oil level in an automobile engine. The engine is capable of holding 7 liters of oil, but has a tendency to pull oil into the intake manifold when the level is equal to or greater than 6 liters. At the other extreme, it has been determined that an oil level of less than 2 liters does not insure that there is a sufficient supply of oil to the oil pressure pump. For proper, safe operation, the engine should have an oil level between these two extremes. In order to simplify the information for the average driver, you have decided to make an indicator with three lights Red, Green, Yellow. These lights are to be illuminated under the following conditions:

- Red Light: Oil level greater than 5 or less than 2 liters.
- Yellow Light: Oil level greater than 4 or less than 3 liters.
- Green Light: Oil level equal to 3 or 4 liters.

More than one light may be on at a time.

You have also found that there is a commercially available oil level indicator that has a binary output proportional to oil level as in the table below:

mt	Output Signal Lines (Oil Level)			Oil Level (liters)
	A	B	C	
	0	0	0	0
	0	0	1	1
	0	1	1	2
	0	1	0	3
	1	1	0	4
	1	1	1	5
	1	0	1	6
	1	0	0	7

1. Design a combinational circuit to perform the functions described above.
2. Minimize the functions using any appropriate method.
3. Write a VHDL behavioral description to implement the circuit, including the following
   a. Entity declaration
   b. Behavioral architecture body
   c. An appropriate testbench
4. Simulate and print the resulting waveforms for all possible inputs.
5. Draw a detailed diagram showing all the pin connections.
6. Implement the circuit in hardware, using the TTL devices from the parts list.
7. Compare the results of the two approaches with respect to the timing characteristics of the outputs.

1. Design a combinational circuit to perform the functions described above.
2. Minimize the functions using any appropriate method.
3. Write a VHDL behavioral description to implement the circuit, including the following
   a. Entity declaration
   b. Behavioral architecture body
   c. An appropriate testbench
4. Simulate and print the resulting waveforms for all possible inputs.
5. Draw a detailed diagram showing all the pin connections.
6. Implement the circuit in hardware, using the TTL devices from the parts list.
7. Compare the results of the two approaches with respect to the timing characteristics of the outputs.

7. Exercise the redundant terms with the test bench and verify that the static 1-hazard has been removed.
8. Implement *f1* with physical gates and capture the transient 0 which occurs when *C* switches from a 1 to a zero.

Experiment 6

MSI Circuit for Cold Switching Signals

It is sometimes awkward in an electronic instrument to have the switches on the front panel directly control the routing of signals (called ``hot switching") among circuits for several reasons. In hot switching, the signals have to be routed to the front panel for switching which introduces delays in the signal due to the length of the wire and possibly noise, not to mention the greater number of wires which must leave one board and be routed back to the board. The single largest source of failures in electronic circuits is connections.

A technique for time-division multiplexing signals can be used to alleviate this problem and serve to introduce "cold switching." In cold-switching, front panel switches indirectly redirect data signals by sending control signals from the switches to a circuit on the signal processing board which actually routes the signals from/to the desired processing circuit. This means that the data signal never leaves the circuit board and is therefore incurs no additional delay and is less susceptible to noise. Not only that, but by this method, 3 control lines can redirect 8 data lines thereby reducing the number of wires which must be used to connect the front panel to a signal processing board. The example presented here is used for routing digital signals, but the process is directly analogous for the routing of analog signals using digital control signals.

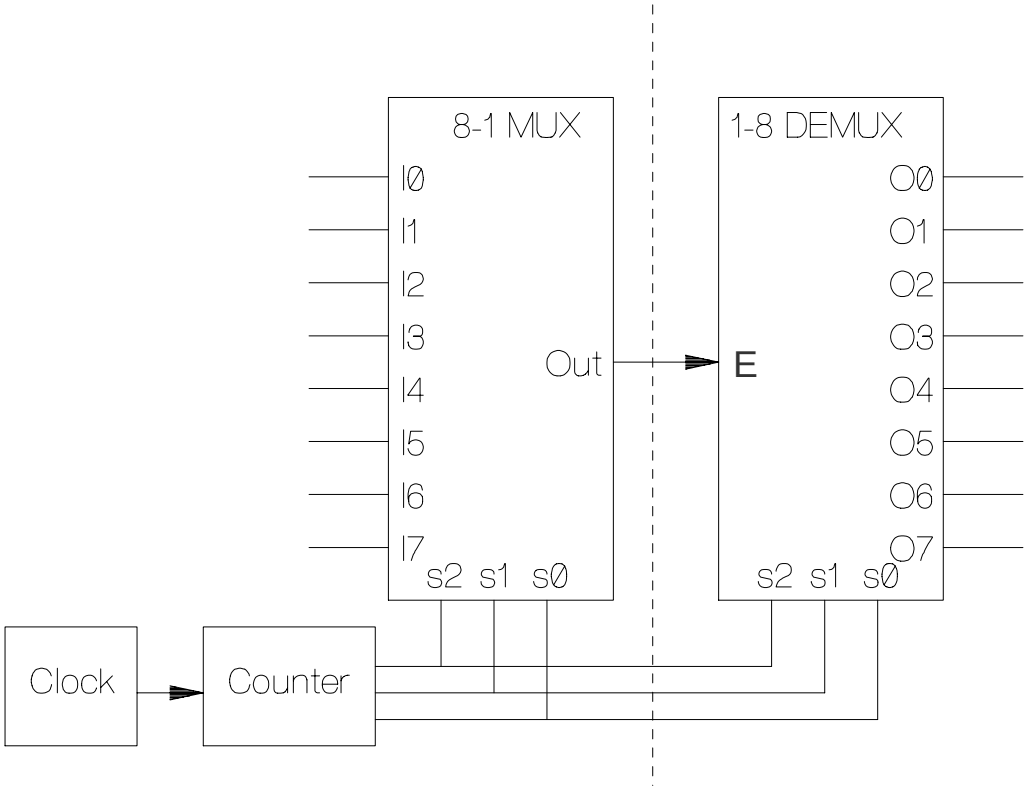
Time-division multiplexing means that multiple signals are transmitted over the same wire, but not at the same time as can be done with frequency-division multiplexing or other methods. The signal which is to be sent over the single data wire is specified by a set of n control signals, where 2^n is the number of signals which can be multiplexed. At the other end of the multiplexed signal data wire is a demultiplexer which selects which of the 2^n output signal lines to route the signal to. It is assumed here that the same address (control signals) is applied to both the multiplexer and demultiplexer.

With reference to the above problem do the following

1. Simulate and verify a multiplexer:
   a. Write a structural VHDL description of an 8-to-1 multiplexer.
   b. Demonstrate using a test bench that the output signal is the same as the selected input signal. Do this for all eight input signal lines.
2. Simulate and verify a demultiplexer:
   a. Write a structural VHDL description of a 1-to-8 demultiplexer.
   b. Demonstrate using a test bench that the input signal is routed to the correct output. Do this for all eight output signal lines.
3. Simulate and verify the complete TDM circuit:
   a. Write a structural VHDL description of an 8-to-1 multiplexer connected to a 1-to-8 demultiplexer.

b. Write a testbench with a counter to drive the control lines of both the multiplexer and the demultiplexer. Demonstrate the proper operation of the digital TDM circuit by verifying that the correct input signal is routed to the proper output.

4. Build an MSI implementation of the complete TDM circuit using parts from the ECE332 parts list and compare it to the TDM simulation.



Experiment 7

Integrated Circuit Technology Lab

As a digital systems designer, you will need to be aware of the electrical characteristics of various logic families, because a design must be logically *and* electrically correct to ensure proper operation. Even if a design is logically correct, it may fail to operate correctly if the design violates electrical limitations and usage rules of a particular logic family.

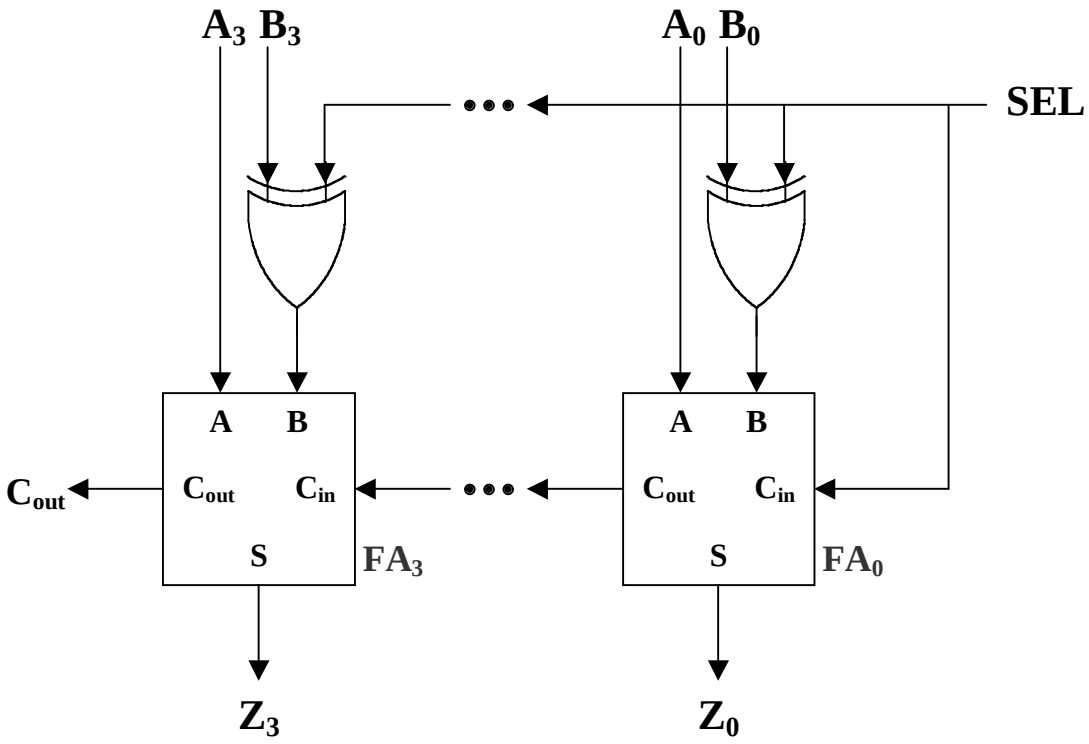
In this lab you will do a comparative study of the 74LS00 Quad 2-input AND gate and the 74C00, a CMOS version of the same device.

1. Using the respective datasheets for the two devices, determine the following electrical characteristics and compare the results for the TTL and CMOS families.
   a. Noise Margin
   b. Fan-out
   c. Propagation Delay
2. Use a dual-trace oscilloscope to observe the actual propagation delays for the two devices. Inject a square wave into the selected channel and compare the input signal with the output signal by displaying them simultaneously on the two channels. Record your observations and comment on the difference in behavior between the TTL and CMOS devices, as well as the differences in observed behavior with that determined in part 1c above.
3. Based on your observed results in part 2 above, write VHDL **package** containing behavioral models for both the TTL and CMOS devices. Use a **generic** clause to model the propagation delay.
4. Write a VHDL structural model and associated testbench that will instantiate each device model with the appropriate propagation delay. Simulate your VHDL model to demonstrate that you have correctly modeled the behavior observed in part 2 above.

Hint: For the CMOS gate, output currents are named I_{source} and I_{sink}. Make sure you use the V_{cc} = 5V parameters.

Experiment 8

4-bit Adder-Subtractor



The circuit diagram above implements a 4-bit adder-subtractor:

$$SEL = 0 \Rightarrow Z = A + B$$

$$SEL = 1 \Rightarrow Z = A + \overline{B} + 1 = A - B$$

1. Write a VHDL **package** containing behavioral descriptions of an *XOR* gate and a *Full Adder*.
2. Write a VHDL structural description to implement the circuit, including the following:
   a. Entity declaration
   b. Structural architecture body using **generate** statements for the regular structures

c. A testbench which demonstrates correct operation of the circuit with an example of each of the following cases:

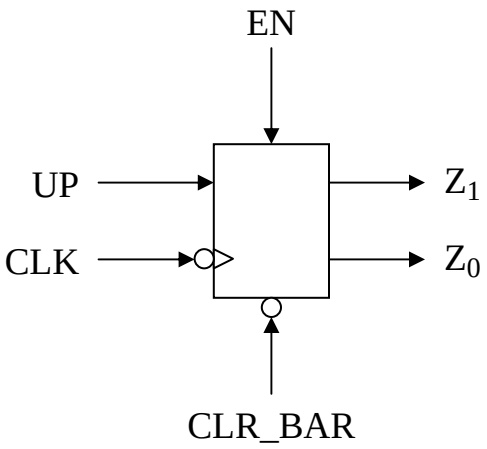
A	**B**	**A + B**	**A - B**
4	2		
2	5		
-3	-4		
-2	6		
6	3		

3. Simulate and print the resulting waveforms for the cases listed above.
4. Design an MSI implementation of the circuit using a 4-bit adder and draw a detailed diagram showing all the pin connections.
5. Implement the circuit in hardware, using the TTL devices from the parts list and verify its operation using the test vectors above.
6. Compare the results of the two approaches with respect to the timing characteristics of the outputs.

Experiment 9

Synchronous Sequential Design

Design a synchronous sequential Moore machine which implements a 2-bit binary counter with the following characteristics:



- An active-high enable, EN
  - When EN = 1, the counter changes state
  - When EN = 0, the counter maintains its current state
- An active-high control signal, UP
  - When UP = 1, the counter counts up in the sequence 00, 01, 10, 11, 00, …
  - When UP = 0, the counter counts down in the sequence 00, 11, 10, 01, 00, …
- An active-low asynchronous clear, CLR_BAR
  - When CLR_BAR = 0, the count is reset to 00
- A falling-edge triggered clock, CLK

To complete this experiment do the following:

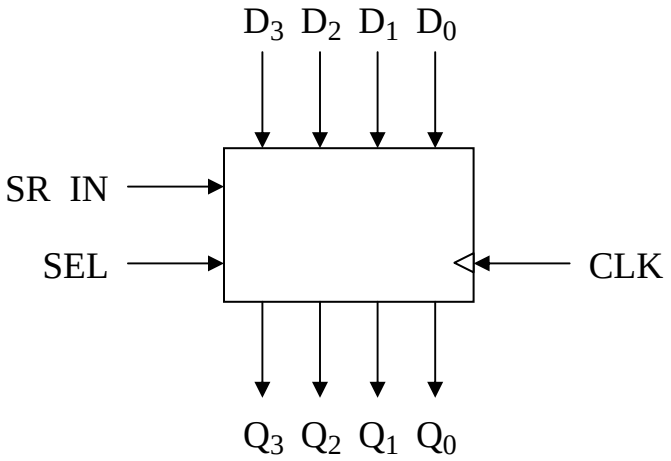
1. Draw a state diagram which represents the Moore machine described above.
2. Fill in the state table below for the Next States and determine Excitation Variables necessary to implement the state machine using JK flip-flops.
3. Minimize the Boolean equations for the Excitation Variables.
4. Draw a detailed circuit diagram for the state machine, showing all pin connections.
5. Implement your circuit in hardware, using TTL parts, and verify its operation.
6. Write a VHDL structural description to implement the circuit, including the following:
   a. Entity declaration.
   b. Structural architecture body using components from the **ece332_gates** package.
   c. A testbench which demonstrates correct operation of the circuit.
7. Simulate your VHDL model and print the resulting waveforms.
8. Compare the results of the two implementations.

	Inputs		Present State		Next State		Excitation Variables				Outputs	
mt	EN	UP	Q_1	Q_0	Q_1^+	Q_0^+	J_1	K_1	J_0	K_0	Z_1	Z_0
0	0	0	0	0							0	0
1	0	0	0	1							0	1
2	0	0	1	0							1	0
3	0	0	1	1							1	1
4	0	1	0	0							0	0
5	0	1	0	1							0	1
6	0	1	1	0							1	0
7	0	1	1	1							1	1
8	1	0	0	0							0	0
9	1	0	0	1							0	1
10	1	0	1	0							1	0
11	1	0	1	1							1	1
12	1	1	0	0							0	0
13	1	1	0	1							0	1
14	1	1	1	0							1	0
15	1	1	1	1							1	1

Experiment 10

Shift Register

Design a *4-bit parallel input / parallel output shift register* with the following characteristics:



- An mode control signal, SEL
  - When SEL = 0, the register loads D_i
  - When SEL = 1, the register shifts right (and Q_3 receives SR_IN)
- A rising-edge triggered clock, CLK

To complete this experiment do the following:

1. Draw the circuit diagram for the *shift register* described above, using D flip-flops and 2 to 1 multiplexers.
2. Implement your circuit in hardware, using TTL parts, and verify its operation.
3. Write a VHDL behavioral description which implements this *shift register*, including the following:
   a. Entity declaration
   b. Dataflow architecture body
   c. An appropriate testbench
4. Simulate your VHDL model and print the resulting waveforms.
5. Compare the results of two implementations.

Experiment 11

Asynchronous Finite State Machine

Given the following characteristic equations for the next state variables of an asynchronous finite state machine:

$$Y_1 = x_1\bar{x}_2 + y_2\bar{x}_1 + y_1\bar{y}_2x_2$$

$$Y_2 = x_1 + y_2\bar{x}_2$$

1. Determine the transition table and identify the stable states.
2. Identify any oscillations and races and indicate their types.
3. Write a VHDL behavioral description which implements the characteristic equations and includes the following:
   a. Entity declaration
   b. Behavioral architecture body
   c. An appropriate testbench
4. Simulate and Print the resulting waveforms for all possible inputs and state transitions.
   a. Set the propagation delays of Y1 and Y2 to demonstrate normal operation.
   b. Adjust the propagation delays to demonstrate the results of a critical race.
5. Draw a detailed diagram showing all the pin configurations.
6. Implement the above circuit in hardware, using the TTL devices from the parts list.
7. Compare the results of two approaches with respect to the timing characteristics of the outputs.

Experiment 12

Aquarium Controller

As chief engineer of $\overline{\text{AQUA}}$ (Pronounced "AquaNot") Aquarium Maintenance Engineering, Inc., you have decided to develop an automatic device to control the acidity (pH) and Calcium content of the water in fish aquariums. You also want to use this device to automatically feed the fish on a regular basis.

You have decided to purchase devices that measure the pH and Calcium content and use them as inputs to your system. The pH measuring device has an output pH which is asserted as (1) when the pH is too high requiring the introduction of a measured amount of buffering agent into the aquarium to lower the pH. The Calcium measuring device has an output (pC) which is asserted as (0) when the calcium level is too low requiring the introduction of a measured amount of Calcium into the aquarium. You will use these two measurement devices as inputs to your design to determine whether to actuate the output devices.

The first of the three output devices introduces a measured amount of buffering solution (B) into aquarium to lower the pH of the water. The second output device introduces a measured amount of calcium (C) into the water to raise Calcium levels. The simultaneous introduction of the buffering solution and calcium cannot be allowed since they may cancel the effects of each other. You have decided that pH control is more important than Calcium control, so that if both pH and pC sensors indicated that the buffer and calcium must be introduced at the same time, you will only introduce the buffer into the water during that cycle.

There is, of course, a third (output) device (F) which dumps food into the aquarium. Through experience you have found that you cannot put fish food into the tank at the same time that you put in any chemicals. Experience has also shown that, if possible, it is more important to control pH and pC than to feed the fish. it is well known that the breed of fish you have in the tank cannot miss more than two feeding cycles before they are adversely affected.

Remembering that this type of problem can be reasonably solved using a synchronous finite state machine of the Mealy type, it further seems reasonable that this can be done in three states where the first one (01) indicates that the fish were fed on the last cycle, the second (10) indicates that the fish have missed one feed cycle and the third (11) state which indicates that the fish have missed a second feed cycle.

With reference to the above word problem do the following:

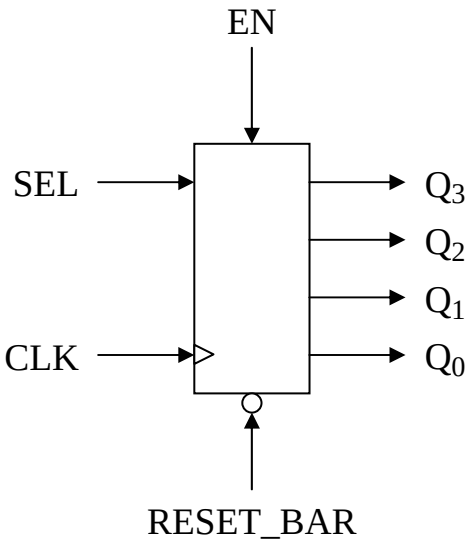
1. Draw a properly labeled and complete state diagram of the Mealy machine to implement the aquarium controller.
2. Fill in a State/Output table for the Mealy machine.

3. Write a VHDL Behavioral Description which implements the finite state machine, including the following
   a. Entity declaration
   b. Behavioral architecture body
   c. An appropriate testbench
4. Simulate and print the resulting waveform for all possible inputs and state transitions.
5. Determine whether the FSM is self-starting and verify this by starting the FSM in state 0.

Bonus Experiment

Variable Modulus Counter

Design a *variable modulus counter* with the following characteristics:



- An active-high enable, EN
  - When EN = 1, the counter changes state
  - When EN = 0, the counter maintains its current state
- A modulus control signal, SEL
  - When SEL = 1, the counter counts up in the sequence 6, 7, … ,10, 11, 6, … (modulus = 6)
  - When SEL = 0, the counter counts up in the sequence 3, 4, … ,10, 11, 3, … (modulus = 9)
- An active-low synchronous reset, RESET_BAR
  - When RESET_BAR = 0, the count is reset to its lowest value, depending on the modulus (controlled by SEL)
- A rising-edge triggered clock, CLK

To complete this experiment do the following:

1. Draw the circuit diagram for the *variable modulus counter* described above.
2. Implement your circuit in hardware, using TTL parts, and verify its operation.
3. Write a VHDL structural description to implement the circuit, including the following:
   a. Entity declaration.
   b. Structural architecture body using components from the **ece332_gates** package.
   c. A testbench which demonstrates correct operation of the circuit.
4. Simulate your VHDL model and print the resulting waveforms.
5. Write a VHDL behavioral architecture to implement the *variable modulus counter*.
6. Simulate your VHDL model and print the resulting waveforms.
7. Compare the results of the three implementations.