**DECODERS IN**

**VERILOG**

**LAB # 0****8**

**Fall 2023**

**CSE-304L**

**Computer Organization & Architecture Lab**

Submitted by: **AIMAL KHAN**

Registration No.: **21PWCSE1996**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”



Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr. Bilal Habib**

Sunday, December 17, 2023

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**ASSESSMENT RUBRICS COA LABS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LAB REPORT ASSESSMENT** | | | | |
| **Criteria** | **Excellent** | **Average** | **Nill** | **Marks Obtained** |
| 1. **Objectives of Lab** | All objectives of lab are properly covered  [Marks 10] | Objectives of lab are partially covered  [Marks 5] | Objectives of lab are not shown  [Marks 0] |  |
| 1. **MIPS instructions with**   **Comments and proper indentations.** | All the instructions are well written with comments explaining the code and properly indented  [Marks 20] | Some instructions are missing are poorly commented code  [Marks 10] | The instructions are not properly written  [Marks 0] |  |
| 1. **Simulation run without error and warnings** | The code is running in the simulator without any error and warnings  [Marks 10] | The code is running but with some warnings or errors.  [Marks 5] | The code is written but not running due to errors  [Marks 0] |  |
| 1. **Procedure** | All the instructions are written with proper procedure  [Marks 20] | Some steps are missing  [Marks 10] | steps are totally missing  [Marks 0] |  |
| 1. **OUTPUT** | Proper output of the code written in assembly  [Marks 20] | Some of the outputs are missing  [Marks 10] | No or wrong output  [Marks 0] |  |
| 1. **Conclusion** | Conclusion about the lab is shown and written  [Marks 20] | Conclusion about the lab is partially shown  [Marks 10] | Conclusion about the lab is not shown [Marks0]  [Marks 0] |  |
| 1. **Cheating** |  |  | Any kind of cheating will lead to 0 Marks |  |
| Total Marks Obtained: \_\_\_\_\_\_\_\_\_\_  Instructor Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |

**Decoders in Verilog**

Objectives:

* Create and test various decoders in Verilog.

Tasks:

**Task 1**: Implement 2X4 Decoder using gate level modeling in Verilog.

**DUT Code:**

module Decoder\_2x4(A, B, E, I);

input A, B, E;

output [3: 0]I; // 3 inputs

wire nA, nB;

not n1(nA, A);

not n2(nB, B);

and a1(I[0], nA, nB, E);

and a2(I[1], nA, B, E);

and a3(I[2], A, nB, E);

and a4(I[3], A, B, E);

endmodule;

**Test Code:**

module Decoder\_2x4\_tb;

reg A, B, E;

wire [3:0] I;

// Instantiate the Decoder\_2x4 module

Decoder\_2x4 uut (

.A(A),

.B(B),

.E(E),

.I(I)

);

// Clock generation (not used in this example)

reg clk = 0;

always #5 clk = ~clk;

// Testbench stimulus

initial begin

// Test case 1: Enable (E) is high, A=0, B=0

A = 0;

B = 0;

E = 1;

#10 $display("Input: A=%b, B=%b, E=%b, Output: I=%b", A, B, E, I);

// Test case 2: Enable (E) is high, A=1, B=0

A = 1;

B = 0;

#10 $display("Input: A=%b, B=%b, E=%b, Output: I=%b", A, B, E, I);

// Test case 3: Enable (E) is high, A=0, B=1

A = 0;

B = 1;

#10 $display("Input: A=%b, B=%b, E=%b, Output: I=%b", A, B, E, I);

// Test case 4: Enable (E) is high, A=1, B=1

A = 1;

B = 1;

#10 $display("Input: A=%b, B=%b, E=%b, Output: I=%b", A, B, E, I);

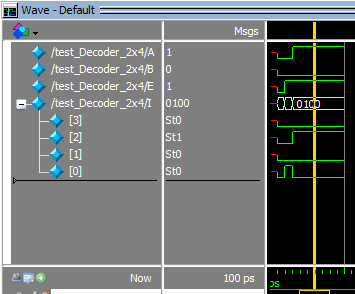
// End simulation

$finish;

end

endmodule;

**Output:**

****

**Task 2**: Implement 3X8 Decoder using gate level modeling in Verilog.

**DUT Code:**

module Decoder\_3x8 ( select, enable, out);

input wire[2:0] select;

input wire enable;

output wire[7:0] out;

assign out[0] = (enable & ~select[0] & ~select[1] & ~select[2]);

assign out[1] = (enable & ~select[0] & ~select[1] & select[2]);

assign out[2] = (enable & ~select[0] & select[1] & ~select[2]);

assign out[3] = (enable & ~select[0] & select[1] & select[2]);

assign out[4] = (enable & select[0] & ~select[1] & ~select[2]);

assign out[5] = (enable & select[0] & ~select[1] & select[2]);

assign out[6] = (enable & select[0] & select[1] & ~select[2]);

assign out[7] = (enable & select[0] & select[1] & select[2]);

endmodule;

**Test Code:**

module test\_Decoder\_3x8;

reg [2:0] select;

reg enable;

wire [7:0] out;

// Instantiate the 3x8 Decoder

Decoder\_3x8 dut (

.select(select),

.enable(enable),

.out(out)

);

// Clock generation (not used in this example)

reg clk = 0;

always #5 clk = ~clk;

// Testbench stimulus

initial begin

// Test case 1

select = 3'b000;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 2

select = 3'b001;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 3

select = 3'b010;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 4

select = 3'b011;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 5

select = 3'b100;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 6

select = 3'b101;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 7

select = 3'b110;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 8

select = 3'b111;

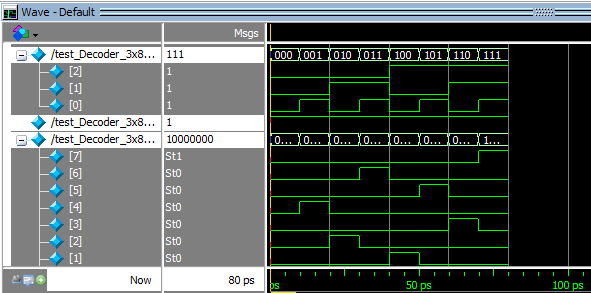
enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

end

endmodule;

**Output:**

****

**Task 3**: Implement 3X8 Decoder using 2X4 Decoder.

**DUT Code:**

module Decoder\_3x8\_using\_2x4 ( select, enable, out);

input wire[2:0] select;

input wire enable;

output wire[7:0] out;

wire [3:0] decoder1\_out;

wire [3:0] decoder2\_out;

Decoder\_2x4 decoder1 (

.select(select[1:0]),

.enable(enable),

.out(decoder1\_out)

);

Decoder\_2x4 decoder2 (

.select(select[2:1]),

.enable(enable),

.out(decoder2\_out)

);

assign out = {decoder2\_out, decoder1\_out};

endmodule;

**Test Code:**

module test\_Decoder\_3x8\_using\_2x4;

reg [2:0] select;

reg enable;

wire [7:0] out;

// Instantiate the 3x8 Decoder

Decoder\_3x8\_using\_2x4 dut (

.select(select),

.enable(enable),

.out(out)

);

// Clock generation (not used in this example)

reg clk = 0;

always #5 clk = ~clk;

// Testbench stimulus

initial begin

// Test case 1

select = 3'b000;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 2

select = 3'b001;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 3

select = 3'b010;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 4

select = 3'b011;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 5

select = 3'b100;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 6

select = 3'b101;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 7

select = 3'b110;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 8

select = 3'b111;

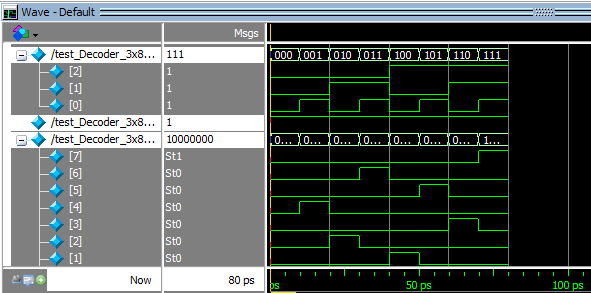
enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

end

endmodule;

**Output:**

****

**Task 4**: Implement 4X16 Decoder using 3X8 Decoder.

**DUT Code:**

module Decoder\_4x16\_using\_3x8 (select,enable,out);

input wire[3:0] select;

input wire enable;

output wire[15:0] out;

wire [7:0] decoder1\_out;

wire [7:0] decoder2\_out;

Decoder\_3x8 decoder1 (

.select(select[2:0]),

.enable(enable),

.out(decoder1\_out)

);

Decoder\_3x8 decoder2 (

.select(select[3:1]),

.enable(enable),

.out(decoder2\_out)

);

assign out = {decoder2\_out, decoder1\_out};

endmodule;

**Test Code:**

module test\_Decoder\_4x16\_using\_3x8;

reg [3:0] select;

reg enable;

wire [15:0] out;

// Instantiate the 4x16 Decoder

Decoder\_4x16\_using\_3x8 dut (

.select(select),

.enable(enable),

.out(out)

);

// Clock generation (not used in this example)

reg clk = 0;

always #5 clk = ~clk;

// Testbench stimulus

initial begin

// Test case 1

select = 4'b0000;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 2

select = 4'b0001;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 3

select = 4'b0010;

enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

// Test case 16

select = 4'b1111;

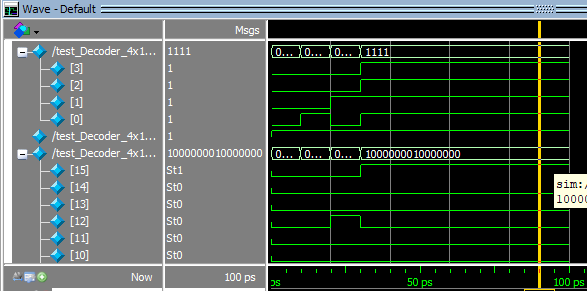
enable = 1;

#10 $display("Input: select=%b, enable=%b, Output=%b", select, enable, out);

end

endmodule;

**Output:**

****

Reference:

To view my codes, please refer to my [GitHub Account.](https://github.com/aimalexe/DCSE/tree/main/semester_5_(fall-23)/computer_organization_and_architechure_lab/lab_reports/)

Conclusion:

In conclusion, I have learned how to create and use decoders in Verilog.

The End.