# Lab 09 – Worksheet

| Name:Syed Asghar Abbas Zaidi | ID:07201 | Section:T6 |
| --- | --- | --- |

## Task a. 4:1 MUX

Complete the truth table provided in Table 9.2. As an example, when the Select line input is S1:0=00, output will be the binary number stored in A3:0

Table 9.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0111 | X | X | X | 0111(7) |
| 0 | 1 | X | 0010 | X | X | 0010(2) |
| 1 | 0 | X | X | 0000 | X | 0000(0) |
| 1 | 1 | X | X | X | 0001 | 0001(1) |

*Provide appropriately commented code for designed module*

| module mux\_4\_1(  input [3:0] A,  input [3:0] B,  input [3:0] C,  input [3:0] D,  input [1:0] S,  output [3:0] O  );  assign O[3:0] = S[1] ? (S[0] ? D : C) : ( S[0] ? B : A );  endmodule |
| --- |

*Write a testbench to thoroughly test designed mux\_4\_1 module.*

| `timescale 1ns / 1ps  module Multiplex\_Test();  reg [3:0] A;  reg [3:0] B;  reg [3:0] C;  reg [3:0] D;  reg [1:0] S;  wire [3:0] O;  mux\_4\_1 module\_u\_test (A,B,C,D,S,O);  initial begin  //is there a way to not write A,B,C,D again and again?  #100 A = 3'b111;  B = 3'b010;  C = 3'b000;  D = 3'b001;  S = 2'b00;  #100 A = 3'b111;  B = 3'b010;  C = 3'b000;  D = 3'b001;  S = 2'b01;    #100 A = 3'b111;  B = 3'b010;  C = 3'b000;  D = 3'b001;  S = 2'b10;    #100 A = 3'b111;  B = 3'b010;  C = 3'b000;  D = 3'b001;  S = 2'b11;    end  endmodule |
| --- |

*Attach screenshot of Simulation output- make sure to scale properly for visibility of all case.*

|  |
| --- |

## Task b. 1:4 DEMUX

Complete the truth table provided in Table 9.4. As an example, when the Select line input is S1:0=00, output EnA will be 0 and the rest of the outputs will remain high.

Table 9.4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| X | X | 1 | 1 | 1 | 1 | 1 |

*Provide appropriately commented code for designed module*

| module demux\_1\_4(  input En,  input [1:0] S,  output EnA,  output EnB,  output EnC,  output EnD  );  assign EnA = En ? (1'b1) : (S[1] ? (S[0] ? 1'b1 : 1'b1) : ( S[0] ? 1'b1 : 1'b0 ));  assign EnB = En ? (1'b1) : (S[1] ? (S[0] ? 1'b1 : 1'b1) : ( S[0] ? 1'b0 : 1'b1 ));  assign EnC = En ? (1'b1) : (S[1] ? (S[0] ? 1'b1 : 1'b0) : ( S[0] ? 1'b1 : 1'b1 ));  assign EnD = En ? (1'b1) : (S[1] ? (S[0] ? 1'b0 : 1'b1) : ( S[0] ? 1'b1 : 1'b1 ));  endmodule |
| --- |

*Add your testbench here*

| `timescale 1ns / 1ps  module demux\_4\_Simulation();  reg En;  reg [1:0] S;  wire EnA;  wire EnB;  wire EnC;  wire EnD;  demux\_1\_4 module\_u\_test (En,S,EnA,EnB,EnC,EnD);  initial begin  //is there a way to not write A,B,C,D again and again?  #100 En = 1'b0;  S = 2'b00;  #100 En = 1'b0;  S = 2'b01;  #100 En = 1'b0;  S = 2'b10;  #100 En = 1'b0;  S = 2'b11;    #100 En = 1'b1;  S = 2'b00;  #100 En = 1'b1;  S = 2'b01;  #100 En = 1'b1;  S = 2'b10;  #100 En = 1'b1;  S = 2'b11;  end  endmodule |
| --- |

*Attach screenshot of waveform results here*

|  |
| --- |

## Exercise

Write a Verilog module topLevelModule that combines MUX (**mux\_4\_1**) and DeMUX (**demux\_1\_4**) modules created in this lab and binary-hexadecimal decoder module already created in Lab 6, according to the block diagram of Figure 9.2 Top Level Module. This module’s inputs includes four 4-bit data lines (i.e., A[3:0], B[3:0], C[3:0], D[3:0]), 2-bit select line (i.e. S[1:0]) to select from the data for binary-hexadecimal decoder, and a 1-bit active low enable signal (i.e. En). The modules output includes one 7-bit data line for seven segments (i.e., Y [6:0]) and four 1-bit Enable signals (i.e., EnA, EnB, EnC, EnD).

*Provide appropriately commented code for your design module*

| `timescale 1ns / 1ps  module topLevelModule(  input [3:0] A,  input [3:0] B,  input [3:0] C,  input [3:0] D,  input [1:0] S,  input En,  output [6:0] Y,  output EnA,  output EnB,  output EnC,  output EnD  );  wire [3:0] O;  mux\_4\_1 Mux(A,B,C,D,S,O);  demux\_1\_4 DeMux(En,S,EnA,EnB,EnC,EnD);  SevenSegment SS (O,Y);  endmodule  module mux\_4\_1(  input [3:0] A,  input [3:0] B,  input [3:0] C,  input [3:0] D,  input [1:0] S,  output [3:0] O  );  assign O[3:0] = S[1] ? (S[0] ? D : C) : ( S[0] ? B : A );  endmodule  module demux\_1\_4(  input En,  input [1:0] S,  output EnA,  output EnB,  output EnC,  output EnD  );  assign EnA = En ? (1'b1) : (S[1] ? (S[0] ? 1'b1 : 1'b1) : ( S[0] ? 1'b1 : 1'b0 ));  assign EnB = En ? (1'b1) : (S[1] ? (S[0] ? 1'b1 : 1'b1) : ( S[0] ? 1'b0 : 1'b1 ));  assign EnC = En ? (1'b1) : (S[1] ? (S[0] ? 1'b1 : 1'b0) : ( S[0] ? 1'b1 : 1'b1 ));  assign EnD = En ? (1'b1) : (S[1] ? (S[0] ? 1'b0 : 1'b1) : ( S[0] ? 1'b1 : 1'b1 ));  endmodule  module SevenSegment(  input [3:0] D,  output [6:0] S);    assign S[0] = ((~D[3])&(~D[2])&(~D[1])&D[0]) | ((~D[3])&(D[2])&(~D[1])&(~D[0])) | ((D[3])&(~D[2])&(D[1])&D[0]) | ((D[3])&(D[2])&(D[1])&D[0]);  assign S[1] = ((D[2])&(D[1])&(~D[0])) | ((D[3])&(D[1])&(D[0])) | ((D[3])&(D[2])&(~D[0])) | ((~D[3])&(D[2])&(~D[1])&D[0]);  assign S[2] = ((D[3])&(D[2])&(~D[0])) | ((D[3])&(D[2])&(D[1])) | ((~D[3])&(~D[2])&(D[1])&(~D[0]));  assign S[3] = ((D[2])&(D[1])&D[0]) | ((~D[3])&(~D[2])&(~D[1])&(D[0])) | ((~D[3])&(D[2])&(~D[1])&(~D[0])) | ((D[3])&(~D[2])&(D[1])&(~D[0]));  assign S[4] = ((~D[3])&D[0]) | ((~D[2])&(~D[1])&(D[0])) | ((~D[3])&(D[2])&(~D[1]));  assign S[5] = ((~D[3])&(~D[2])&D[0]) | ((~D[3])&(~D[2])&(D[1])) | ((~D[3])&(D[1])&D[0]) | ((D[3])&(D[2])&(~D[1])&D[0]);  assign S[6] = ((~D[3])&(~D[2])&(~D[1])) | ((~D[3])&(D[2])&(D[1])&(D[0])) | ((D[3])&(D[2])&(~D[1])&(~D[0]));  endmodule |
| --- |

*Add your testbench here*

| *`timescale 1ns / 1ps*  *module topLevelModuleSimulation();*  *reg [3:0] A;*  *reg [3:0] B;*  *reg [3:0] C;*  *reg [3:0] D;*  *reg [1:0] S;*  *reg En;*  *wire [6:0] Y;*  *wire EnA;*  *wire EnB;*  *wire EnC;*  *wire EnD;*  *topLevelModule module\_u\_test (A,B,C,D,S,En,Y,EnA,EnB,EnC,EnD);*  *initial begin*  *//is there a way to not write A,B,C,D again and again?*  *#100 A = 3'b111;*  *B = 3'b010;*  *C = 3'b000;*  *D = 3'b001;*  *S = 2'b00;*  *En = 1'b00;*    *#100 A = 3'b111;*  *B = 3'b010;*  *C = 3'b000;*  *D = 3'b001;*  *S = 2'b01;*  *En = 1'b00;*    *#100 A = 3'b111;*  *B = 3'b010;*  *C = 3'b000;*  *D = 3'b001;*  *S = 2'b10;*  *En = 1'b00;*    *#100 A = 3'b111;*  *B = 3'b010;*  *C = 3'b000;*  *D = 3'b001;*  *S = 2'b11;*  *En = 1'b0;*    *#100 A = 3'b111;*  *B = 3'b010;*  *C = 3'b000;*  *D = 3'b001;*  *S = 2'b00;*  *En = 1'b1;*    *end*  *endmodule* |
| --- |

*Attach screenshot of waveform results here*

|  |
| --- |

## Assessment Rubrics

**Marks Distribution:**

|  |  | **LR2**  **Code** | **LR5**  **Results** | **LR7**  **Viva (Performance)** | **LR9**  **Report** |
| --- | --- | --- | --- | --- | --- |
| **In-lab** | **Task a** | 10 points | 10 points | 10 points | 20 Points |
| **Task b** | 10 points | 10 points |
| **Exercise (In-Lab)** |  | 20 points | 10 Points |
|  |  | 40 | 30 | 10 | 20 |
| **Total** |  | | | | 100 Points |

**Marks Obtained:**

|  |  | **LR2**  **Code** | **LR5**  **Results** | **LR7**  **Viva (Performance)** | **LR9**  **Report** |
| --- | --- | --- | --- | --- | --- |
| **In-lab** | **Task a** |  |  |  |  |
| **Task b** |  |  |
| **Exercise (In-Lab)** |  |  |  |
| **Total** |  | | | |  |