CS223

Digital Design

Section 2

Lab 3 – Preliminary Report

Seçkin Alp Kargı

22001942

CS

06.05.2023

Behavioral System Verilog Module for 1 to 2 Decoder

```
`timescale 1ns / 1ps
module decode12
(
  input logic [0:0] in,
  output logic [1:0] out
);

  always_comb begin
    if (in == 1'b0) begin
      out = 2'b10;
    end else if (in == 1'b1) begin
      out = 2'b01;
    end else begin
      out = 2'b00;
    end
  ends
endmodule
```

Test Bench for 1 to 2 Decoder

```
`timescale 1ns / 1ps

module decode12test;

  logic [0:0] in;

  logic [1:0] out;

reg clk;

  decode12 dc12

(

    .in(in),

    .out(out)

  );

  always #10ns clk = 1'b0;

  always #10ns clk = ~clk;

  initial begin

    #0ns in = 1'b0;

    #20ns in = 1'b1;

    #40ns in = 1'bx;

    #60ns in = 1'b0;

    #80ns in = 1'b1;

    #100ns in = 1'bz;

    #120ns $finish;

  end

endmodule
```

Structural System Verilog Module for 2 to 4 Decoder using 3 x 1 to 2 Decoder

```
`timescale 1ns / 1ps

module decode24

 (

  input logic [1:0] in,

  output logic [3:0] out

);

  decode12 dc12a (

    .in(in[0]),

    .out({out[0], out[1]})

  );

  decode12 dc12b (

    .in(in[1]),

    .out({out[2], out[3]})

  );

  decode12 dc12c (

    .in(1'b0),

    .out({out[0], out[2]})

  );


endmodule
```

Test Bench for 2 to 4 Decoder

```
`timescale 1ns / 1ps

module decode24test;

 logic [1:0] in;

 logic [3:0] out;

 decode24 dc24

(

   .in(in),

   .out(out)

 );

 always #10ns clk = 1'b0;

 always #10ns clk = ~clk;

 initial begin

   #0ns in = 2'b00;

   #20ns in = 2'b01;

   #40ns in = 2'b10;

   #60ns in = 2'b11;

   #80ns in = 2'b00;

   #100ns in = 2'b11;

   #120ns $finish;

 end

endmodule
```

Behavioral System Verilog Module for 2 to 1 Multiplexer

```
`timescale 1ns / 1ps
module mux21
(
 input logic [1:0] in,
 input logic select,
 output logic out
);

 always_comb begin
   if (select == 1'b0) begin
     out = in[0];
   end else begin
     out = in[1];
   end
 end
endmodule
```

Test Bench for 2 to 1 Multiplexer

```
`timescale 1ns / 1ps

module mux21test;

  logic [1:0] in;

  logic select;

  logic out;

  mux21 m21

(

    .in(in),

    .select(select),

    .out(out)

  );

  always #10ns clk = 1'b0;

  always #10ns clk = ~clk;

  initial begin

    #0ns in = 2'b00;

    #20ns in = 2'b01;

    #40ns in = 2'b10;

    #60ns in = 2'b11;

    #80ns in = 2'b00;

    #100ns in = 2'b11;

    #120ns $finish;

  end

endmodule
```

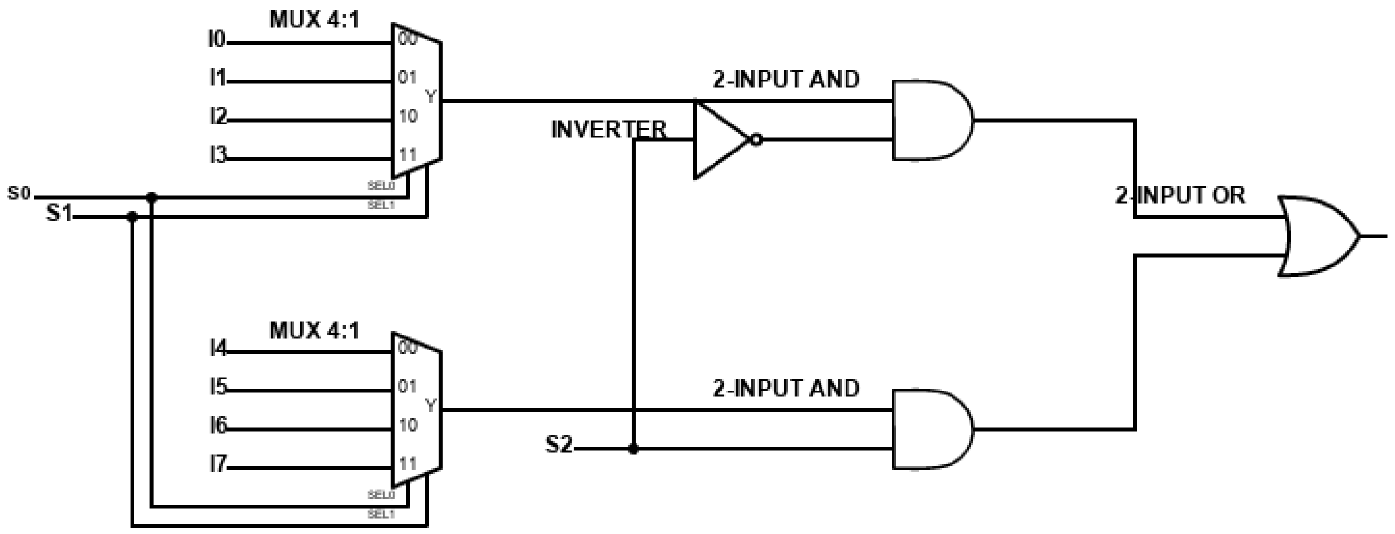
Structural System Verilog Module for 4 to 1 multiplexer using 3 x 2 to 1 multiplexer

```
`timescale 1ns / 1ps
module mux41
 (
  input logic [3:0] in,
  input logic [1:0] select,
  output logic out
);
  logic [1:0] select1, select2;
  logic [1:0] in1, in2, in3;
  assign in1 = in[3:2];
  assign in2 = in[1:0];
  assign in3 = in[3:2];
  mux21 m21a (
    .in({in1, in2}),
    .select(select[1]),
    .out(out1)
  );
  mux21 m21b (
    .in({in3, in[1:0]}),
    .select(select[1]),
    .out(out2)
  );
  mux21 m21c (
    .in({out1, out2}),
    .select(select[0]),
    .out(out)
  );
endmodule
```

Test Bench for 4 to 1 Multiplexer

```
`timescale 1ns / 1ps
module mux41test;
 logic [3:0] in;
 logic [1:0] select;
 logic out;
 mux41 dut (
   .in(in),
   .select(select),
   .out(out)
 );
 always #10ns clk = 1'b0;
 always #10ns clk = ~clk;
 initial begin
   in <= 4'b0000;
   select <= 2'b00;
   #20ns in <= 4'b0001;
   #40ns in <= 4'b0010;
   #60ns in <= 4'b0100;
   #80ns in <= 4'b1000;
   #100ns in <= 4'b1100;
   #120ns $finish;
 end
 always @(posedge clk) begin
   case ($time)
     50: select = 2'b00;
     100: select = 2'b01;
   endcase
 end
endmodule
```

Block Diagram of 8-to-1 MUX by using two 4-to-1 MUX modules, two AND gates, an INVERTER, and an OR gate



Behavioral System Verilog Module for And Gate

```
module and2
(
  input logic x,
  input logic y,
  output logic z
);
  assign z = x & y;
endmodule
```

Behavioral System Verilog Module for Or Gate

```
module or2
(
  input logic x,
  input logic y,
  output logic z
);
  assign z = x | y;
endmodule
```

Structural System Verilog Module for 8 to 1 Mux

```
`timescale 1ns / 1ps

module mux81

(

 input logic [7:0] in,

 input logic [2:0] select,

 output logic out

);

 logic outmux1, outmux2, outand1, outand2;

 mux41 m41a

(

  .in(in[3:0]),

  .select(select[1:0]),

  .out(outmux1)

 );

 mux41 m41b

 (

  .in(in[7:4]),

  .select(select[1:0]),

  .out(outmux2)

 );

 and2 a1

 (

  .x(select[2]),

  .y(outmux1),

  .z(outand1)

 );

 and2 a2

 (

  .x(~select[2]),
    .y(outmux2),
    .z(outand2)
  );
  or2 o1 (
    .x(outand1),
    .y(outand2),
    .z(out)
  );
endmodule
```

SystemVerilog module for F(A,B,C,D)=Σ(0,2,5,6,8,10,12,13,15) function, using only one 8-to-1 multiplexer and a 2-to-4 decoder.

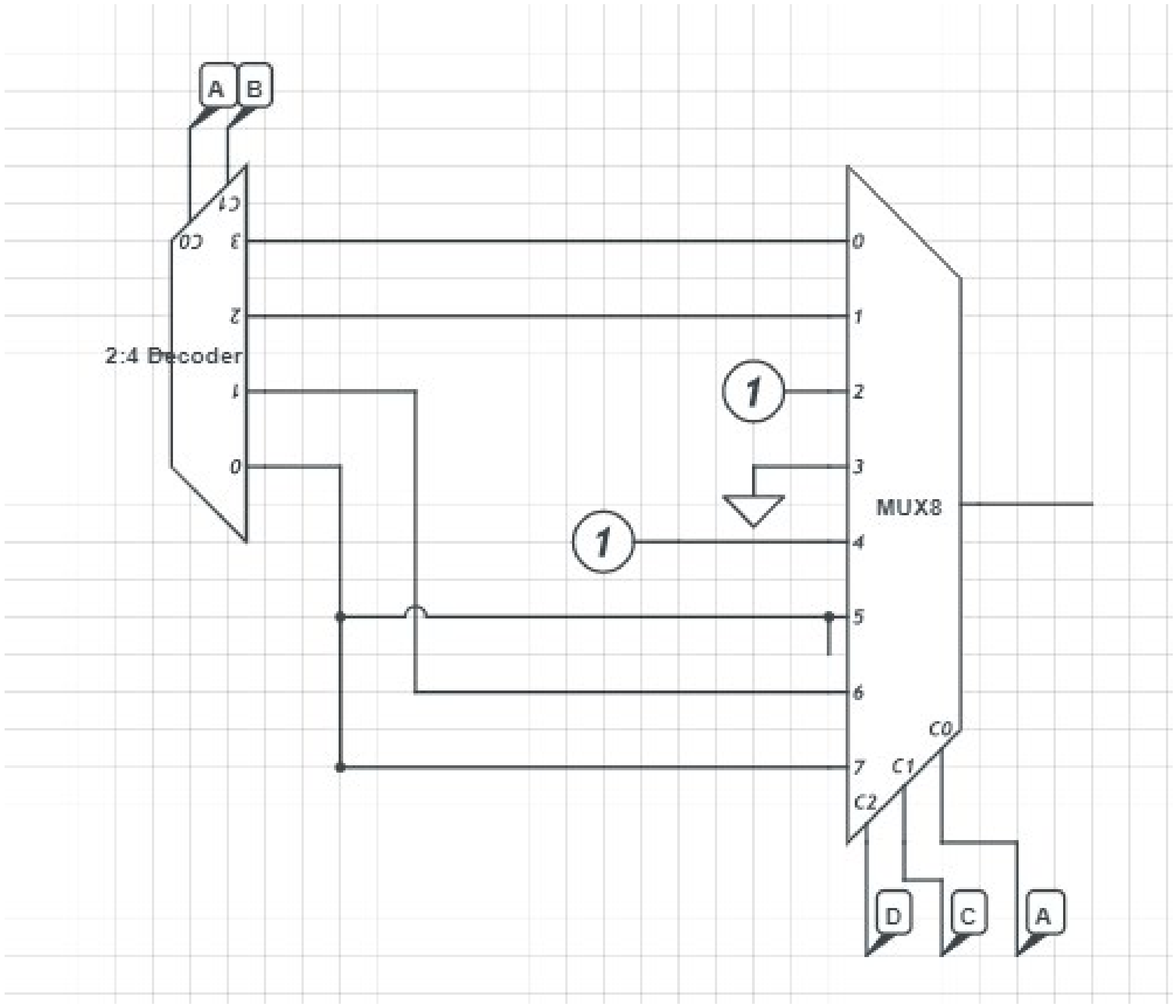
```
module func
(
 input logic A, B, C, D,
 output logic Y
);
 logic out1,out2,out3,out4;
 decode24 decoder
 (
   .in({A, B}),
   .out({out1,out2,out3,out4})
 );
 logic [7:0] in;
 assign in[0] = out1;
 assign in[1] = out2;
 assign in[2] = 1'b1;
 assign in[3] = 1'b0;
 assign in[4] = 1'b1;
 assign in[5] = out4;
 assign in[6] = out3;
 assign in[7] = out4;
 mux81 mux
(
   .in(in),
   .select({A,B,C}),
   .out(Y)
 );
Endmodule
```

TestBench 8 to 1 Multi

```
`timescale 1ns / 1ps
module functest;
  logic [3:0] A, B, C, D;
  logic Y;
  func fun
(
    .A(A),
    .B(B),
    .C(C),
    .D(D),
    .Y(Y)
  );
  initial begin
A = 0; B = 0; C = 0; D = 0; #10;
A = 0; B = 0; C = 0; D = 1; #10;
A = 0; B = 1; C = 1; D = 1; #10;
A = 0; B = 1; C = 1; D = 0; #10;
A = 0; B = 0; C = 1; D = 0; #10;
A = 0; B = 1; C = 0; D = 0; #10;
A = 1; B = 0; C = 0; D = 0; #10;
A = 0; B = 0; C = 1; D = 1; #10;
A = 0; B = 1; C = 1; D = 0; #10;
A = 1; B = 1; C = 0; D = 0; #10;
A = 1; B = 1; C = 0; D = 1; #10;
A = 1; B = 1; C = 1; D = 1; #10;
  end
endmodule
```

Block diagram

I couldnot find decoder I flip multiplexer 3 is out1 2 is out 2 1 is out 3 0 is out 4


A
B
C1
C0
3
2
2:4 Decoder
1
0
0
1
1
2
3
MUX8
1
4
5
6
C0
7
C1
C2
D
C
A