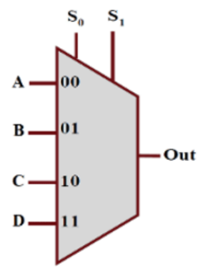
Foundations of Computer Systems Design Lab (CS2310)
Lab 4: Multiplexers and Demultiplexers

3rd September 2024

In this lab, we will learn how to implement multiplexers and demultiplexers in Verilog and use smaller multiplexers and demultiplexers to build bigger ones. In addition, we will also learn to combine multiple Verilog modules to form one large top-level module.

1 Designing a Multiplexer

Design a 16x1 MUX using 4x1 MUXes. The circuit and truth table for a 4x1 MUX is given below:



S_1	S_0	A	B	C	D	Out
0	0	0	x	x	x	0
0	0	1	x	x	x	1
0	1	x	0	x	x	0
0	1	x	1	x	x	1
1	0	x	x	0	x	0
1	0	x	x	1	x	1
1	1	x	x	x	0	0
1	1	x	x	x	1	1

Figure 1: 4-to-1 multiplexer

Write Verilog code that implements this 4x1 MUX and use it to create a 16x1 MUX that receives 16 inputs and a 4-bit select line to select one of the 16 inputs and redirect it to the output. The module signature should be:

```
module mux_16x1(output out, input [15:0] in, input [3:0]sel);
```

Note: Use structural modeling only.
Implement the select line as a 2-bit wire.

2 Designing a Demultiplexer

Design a 1x8 demultiplexer using 4x1 and 2x1 demultiplexers. The circuits and truth tables for 1x2 and 1x4 DeMUXes are given below:

Select	Input	Outputs	
S	D	Y_2	Y_1
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

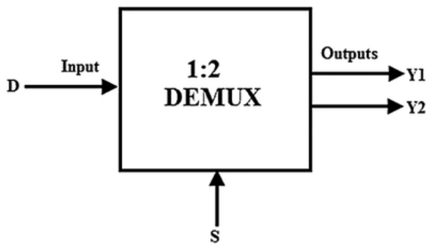
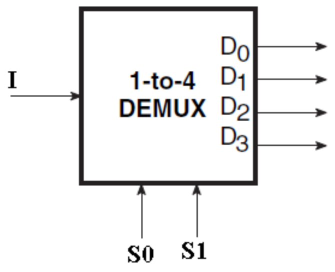


Figure 2: 1-to-2 demultiplexer



I	Select		O/P			
	S0	S1	D_0	D_1	D_2	D_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Figure 3: 1-to-4 demultiplexer

Write Verilog code that implements these 1x2 and 1x4 DeMUXes and use these modules to create a 1x8 DeMUX that receives 1-bit input and a 3-bit select line and produces an 8-bit output. The module signature should be:

```
module demux_1x8(output [7:0] out, input in, input [2:0] sel);
```

Note: Use structural modeling only.
Implement the select line as a 2-bit wire.

3 Design a 7-Segment Display

Design a circuit that gets a 2-bit (4-bit for bonus) binary value as an input and generates 7 control signals that drive the 7-segment display. Each of the 7 segments is labeled using the letters a, b, c, d, e, f, and g. Each of the hexadecimal digits can be represented by making particular segments glow as shown below:

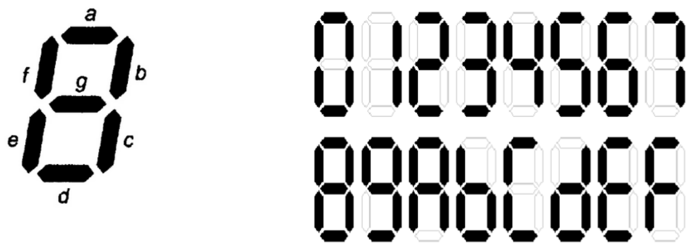


Figure 4: 7-segment display

You can implement the 7-segment decoder only for 2-bit inputs (which maps to digits 0 to 3) in the lab. The implementation for 4-bit inputs (which maps to digits 0 to F) will be treated as a bonus.

Write a Verilog module that combines a 4x1 MUX and a 1x4 DeMUX along with this binary-hexadecimal decoder module according to the block diagram given below. This module's inputs include four 2-bit (4-bit for bonus) data lines (i.e., a[1:0], b[1:0], c[1:0], d[1:0]), 2-bit select line (i.e. sel[1:0]) to select from the data for binary-hexadecimal decoder, and a 1-bit enable signal (i.e. en). The module's output includes one 7-bit data line for seven segments (i.e., y[6:0]) and four 1-bit enable signals (i.e., en_a, en_b, en_c, en_d).

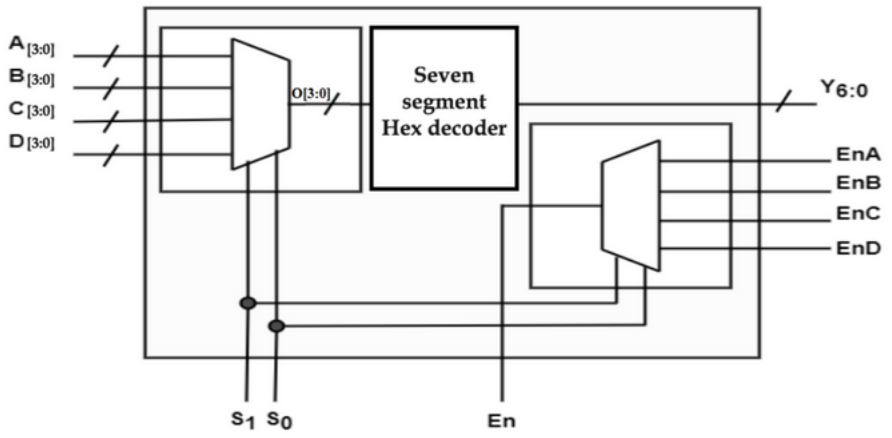


Figure 5: Top level module

The module signature should be:

```
module top(output[6:0] y,output en_a,en_b,en_c,en_d,
           input[1:0] a,b,c,d,input[1:0] sel,input en);
```

Note: You can use behavioral modeling for the 2-bit (4-bit for bonus)

MUX.
The MUX is just to select one of the 4 inputs for the 7-segment decoder. The use of the MUX and DeMux would be relevant in a bigger circuit (not presented here). Here, we will just focus on how to assemble different modules together in a circuit.