BBM233: Logic Design Lab
2020 Fall
Lab Experiment 4
Combinational Circuits in Verilog

Şevval Atmaca, 21827115

December 5, 2020

1 Problem Statement

In this experiment, we are designing and simulating combinational circuits in Verilog HDL. Combinational circuits are circuits whose outputs, at any instant of time, depend only on the present inputs (the combinational circuits do not use any memory elements). That is, the previous inputs or state of the circuit do not have any effect on its present state. Also a decoder is a combinational circuit.

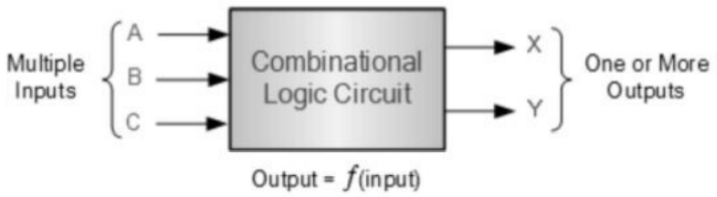

Figure 1: Combinational Circuit

I was trying to make 3x8 decoder using combinational circuit.Then I was making larger decoder circuit (4x16 decoder) using 3x8 decoder.

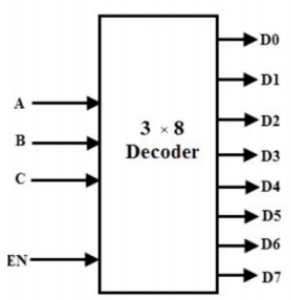

Figure 2: Decoder 3x8

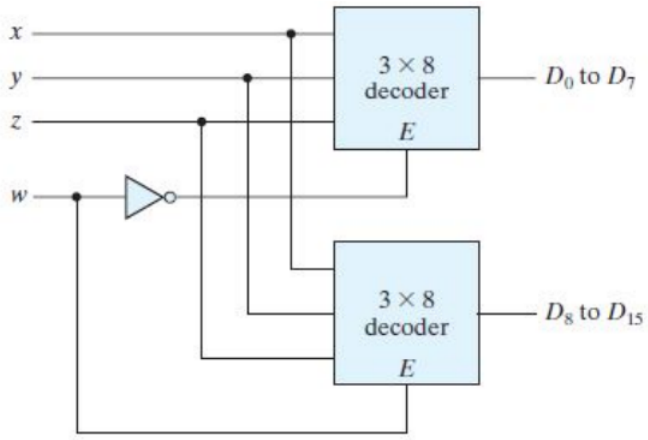

Figure 3: Decoder 4x16

2 Codes

```
`timescale 1ns / 1ps

module decoder_3x8_tb;
    //inputs
    reg [2:0] input_stimuli;
    reg En;
    //outputs
    wire [7:0]D;

    decoder_3x8 UUT( .A(input_stimuli[2]),.B(input_stimuli[1]),.C(input_stimuli[0]),.En(En),.D(D));
    //stimuli
    initial
     begin
      En=1'b0; input_stimuli = 3'b0;
      #100 En=1'b0; input_stimuli = 3'b001;
      #100 En=1'b1; input_stimuli = 3'b0;
        for(integer i = 0; i<7; i=i+1)begin
            #100 En = 1'b1 ;input_stimuli = input_stimuli + 1;
        end
        #100 $finish;
     end
endmodule
```

Figure 4: decoder3x8 tb

```
`timescale 1ns / 1ps


module decoder_3x8(
    //inputs
    input A,
    input B,
    input C,
    input En,//enable
    //outputs
    output [7:0]D
    );// behavoral
    assign D[0]= En & ~A & ~B & ~C;
    assign D[1]= En & ~A & ~B & C;
    assign D[2]= En & ~A & B & ~C;
    assign D[3]= En & ~A & B & C;
    assign D[4]= En & A & ~B & ~C;
    assign D[5]= En & A & ~B & C;
    assign D[6]= En & A & B & ~C;
    assign D[7]= En & A & B & C;
endmodule
```

Figure 5: decoder3x8

```
`timescale 1ns /1ps

module decoder_4x16_tb;
    reg [3:0] input_stimulus;
    wire [15:0] output_D;

    decoder_4x16 UUT(.x(input_stimulus[2]),.y(input_stimulus[1]),.z(input_stimulus[0]),.w(input_stimulus[3]),.output_D(output_D));

    //stimule
    initial
     begin
      input_stimulus = 4'b0;
        for(integer i = 0; i<15; i=i+1)begin
            #100 input_stimulus = input_stimulus + 1;
        end
        #100 $finish;
     end
endmodule
```

Figure 6: decoder4x16 tb

```
`timescale 1ns / 1ps
`include "decoder_3x8.v"

module decoder_4x16(
    //inputs
    input x,
    input y,
    input z,
    input w,

    //outputs
    output [15:0]output_D
    );
    //two decoders
    decoder_3x8 A1(.A(x), .B(y), .C(z), .En(~w), .D(output_D[7:0]));
    decoder_3x8 A2(.A(x), .B(y), .C(z), .En(w), .D(output_D[15:8]));
endmodule
```

Figure 7: decoder4x16

3 Waveforms

Figure 8: decoder 3x8

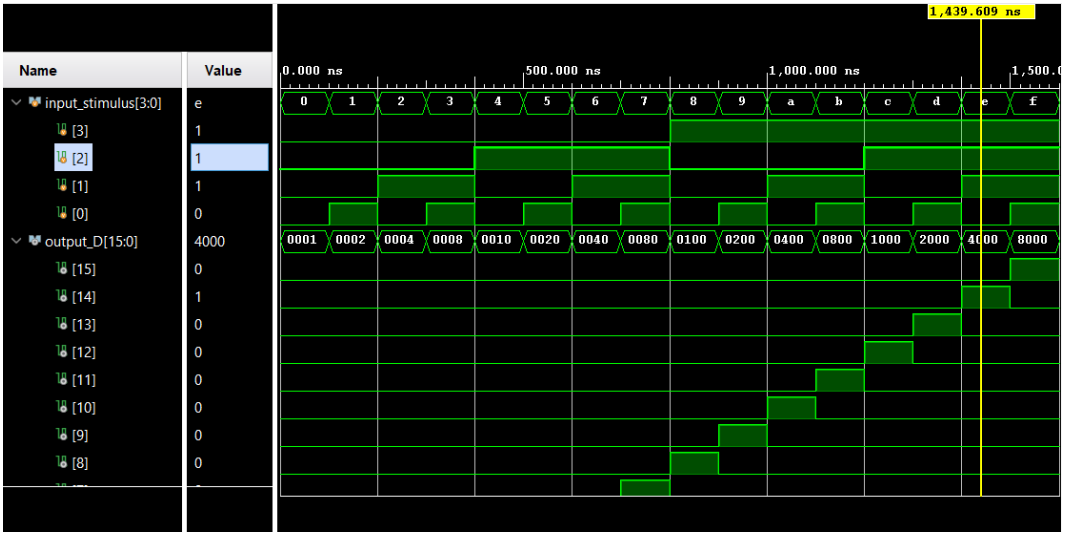

Figure 9: decoder 4x14

4 Codes

In decoder 3x18, when enable is 0 in the 3x8 decoder, all outputs are 0 regardless of the value of other inputs. And accordingly when I look at my waveform it is 0 proves that it is true. And if enable is 1 and we compare the other values according to the truth table, then according to the value of the input, that output should be 1 and the others should be. This proves if I examine input.

In the decoder is 4x16, if we think that there are 2 decoder 3x8.
Then, the first one w is connected to the "not gate", so when w is 1, then enable is 0 and all outputs must be zero. When w is 0, enable must be 1, the output value will be 1 according to the input value (x, y, z) and others zero.
In the second one, there is no "not gate" in enabled. If enable(w) is 0, all outputs must be 0. If enable(w) is 1, the output value will be 1 according to the input value (x, y, z) and others zero. It proves when I examine all inputs on waveform

References

- lesson slides