Hacettepe University

Computer Engineering Department

BM233 Logic Design Lab - 2021 Fall

Verilog Assignment 1

December 6, 2021

Student name:
Metin Eren Oktay

Student Number:
2210356137

1 Problem Definition

We are required to create a combinational circuit using a 2x4 decoder and a 4x1 MUX in Verilog to learn basics of Verilog and Vivado.

2 Verilog Code Solutions

2.1 Decoder 2x4

```verilog
module decoder_2x4(
    input[1:0] A,
    output[3:0] D
    );
    assign D[3] = (A[0]) && (A[1]);
    assign D[2] = (!(A[0])) && (A[1]);
    assign D[1] = (A[0]) && (!(A[1]));
    assign D[0] = (!(A[0])) && (!(A[1]));
endmodule
```

2.2 Mux 4x1

```verilog
module mux_4x1(
    input[3:0] i,
    input[1:0] s,
    output F
);
    wire s1neg, s0neg, T1, T2, T3, T4;
    not (s0neg, s[0]), (s1neg, s[1]), (i1neg, i[1]), (i2neg, i[2]);
    and (T1, i[0], s0neg, s1neg);
    and (T2, i[1], s[0], s1neg);
    and (T3, i[2], s0neg, s[1]);
    and (T4, i[3], s[0], s[1]);
    or(F, T1, T2, T3, T4);
endmodule
```

2.3 Circuit

```verilog
`include "dec.v"
`include "mu.v"
module circuit(
    input a,
    input b,
    input c,
    input d,
    output F
);
    wire [3:0] Carries;
    decoder_2x4 DE(.A({a,b}), .D(Carries));
    mux_4x1 MU(.i(Carries), .s({c,d}), .F(F));
endmodule
```

3 Testbench Waveforms

3.1 Decoder 2x4 TB

```verilog
module decoder_2x4_tb;
    reg [1:0] A;
    wire [3:0] D;
    decoder_2x4 uut(.A(A), .D(D));
    initial begin
        A[1] = 1'b0; A[0] = 1'b0;
        #5 A[1] = 1'b0; A[0] = 1'b1;
        #5 A[1] = 1'b1; A[0] = 1'b0;
        #5 A[1] = 1'b1; A[0] = 1'b1;
        #5 $finish;
    end
endmodule
```

3.2 MUX 4x1 TB

```
module mux_4x1_tb;
    reg [3:0] i;
    reg [1:0] s;
    wire F;
    mux_4x1 uut(.i(i), .s(s), .F(F));
    initial begin
        s[1]=1'b0; s[0]=1'b0; i[3]=1'b0; i[2]=1'b0; i[1]=1'b0; i[0]=1'b1;
        #5 s[1]=1'b0; s[0]=1'b0; i[3]=1'b1; i[2]=1'b1; i[1]=1'b1; i[0]=1'b0;
        #5 s[1]=1'b0; s[0]=1'b1; i[3]=1'b0; i[2]=1'b0; i[1]=1'b1; i[0]=1'b0;
        #5 s[1]=1'b0; s[0]=1'b1; i[3]=1'b1; i[2]=1'b1; i[1]=1'b0; i[0]=1'b1;
        #5 s[1]=1'b1; s[0]=1'b0; i[3]=1'b0; i[2]=1'b1; i[1]=1'b0; i[0]=1'b0;
        #5 s[1]=1'b1; s[0]=1'b0; i[3]=1'b1; i[2]=1'b0; i[1]=1'b1; i[0]=1'b1;
        #5 s[1]=1'b1; s[0]=1'b1; i[3]=1'b1; i[2]=1'b0; i[1]=1'b0; i[0]=1'b0;
        #5 s[1]=1'b1; s[0]=1'b1; i[3]=1'b0; i[2]=1'b1; i[1]=1'b1; i[0]=1'b1;
        #5 $finish;
    end
endmodule
```

3.3 Circuit TB

```
module circuit_tb;
    reg a;
    reg b;
    reg c;
    reg d;
    wire F;
    circuit uut(.a(a), .b(b), .c(c), .d(d), .F(F));
    initial begin
        a = 1'b0; b = 1'b0; c = 1'b0; d = 1'b0;
        #5 a = 1'b0; b = 1'b0; c = 1'b0; d = 1'b1;
        #5 a = 1'b0; b = 1'b0; c = 1'b1; d = 1'b0;
        #5 a = 1'b0; b = 1'b0; c = 1'b1; d = 1'b1;
        #5 a = 1'b0; b = 1'b1; c = 1'b0; d = 1'b0;
        #5 a = 1'b0; b = 1'b1; c = 1'b0; d = 1'b1;
        #5 a = 1'b0; b = 1'b1; c = 1'b1; d = 1'b0;
        #5 a = 1'b0; b = 1'b1; c = 1'b1; d = 1'b1;
        #5 a = 1'b1; b = 1'b0; c = 1'b0; d = 1'b0;
        #5 a = 1'b1; b = 1'b0; c = 1'b0; d = 1'b1;
        #5 a = 1'b1; b = 1'b0; c = 1'b1; d = 1'b0;
        #5 a = 1'b1; b = 1'b0; c = 1'b1; d = 1'b1;
        #5 a = 1'b1; b = 1'b1; c = 1'b0; d = 1'b0;
        #5 a = 1'b1; b = 1'b1; c = 1'b0; d = 1'b1;
        #5 a = 1'b1; b = 1'b1; c = 1'b1; d = 1'b0;
        #5 a = 1'b1; b = 1'b1; c = 1'b1; d = 1'b1;
        #5 $finish;
    end
endmodule
```

4 Waveforms

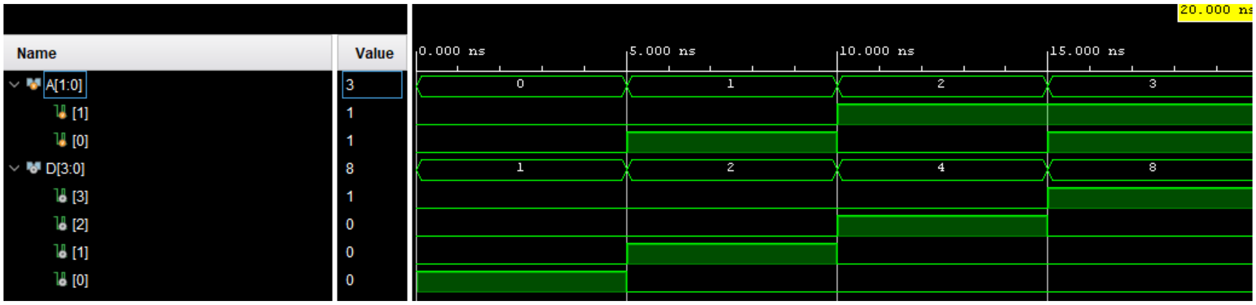


Figure 1: Decoder 2x4

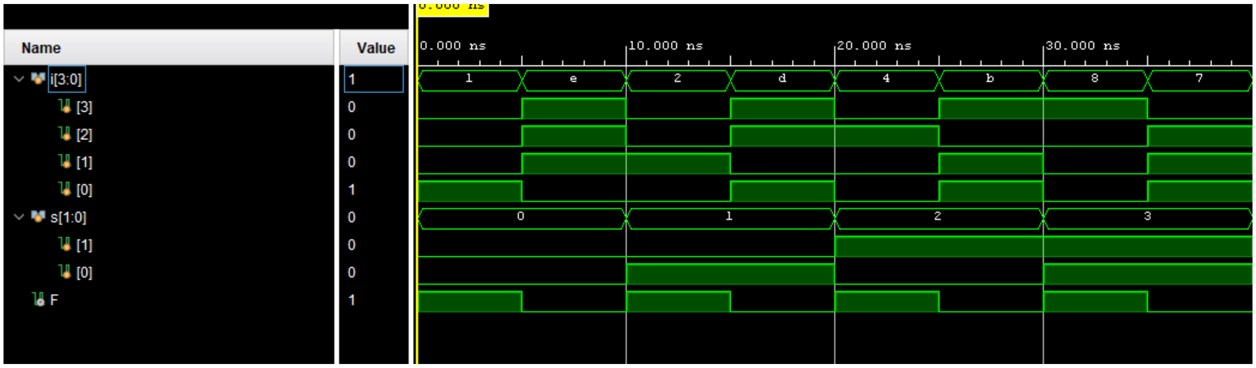


Figure 2: MUX 4X1

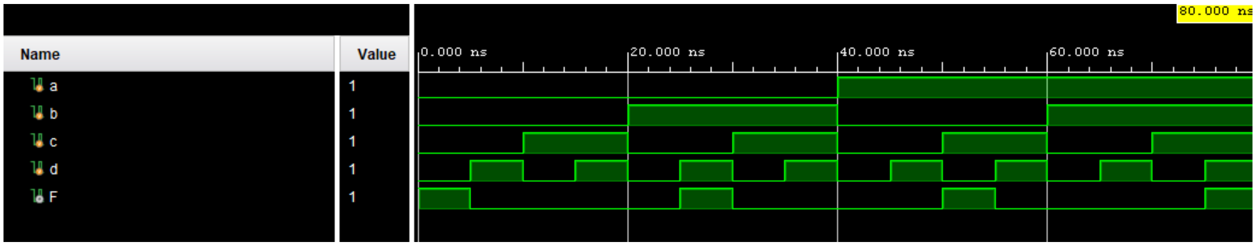
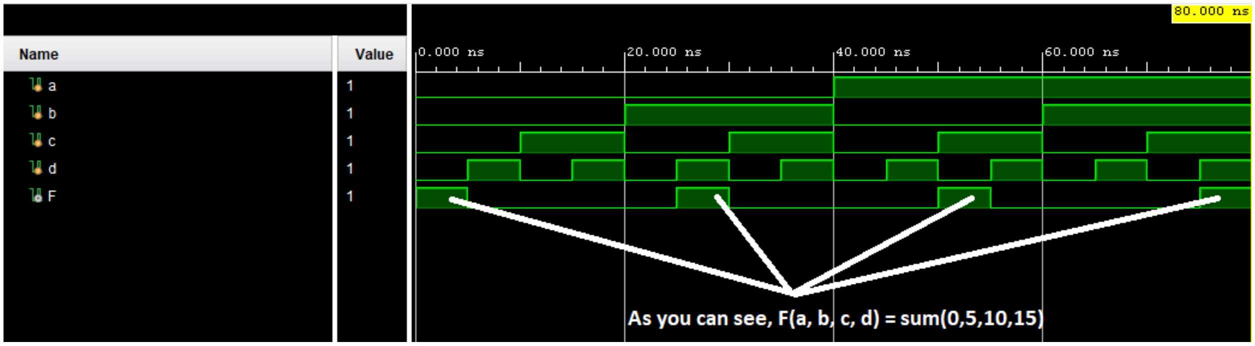


Figure 3: Circuit

5 Correctness of the Circuit

I designed a 2x4 decoder, a 4x1 multiplexer and combined these to implement a combinational circuit. As you can see below; giving the combinational circuit, you can get accurate outputs.



As you can see, F(a, b, c, d) = sum(0,5,10,15)

Figure 4: Decoder 2x4

References

- Mux implementation reference