ECE5260: HARDWARE DESCRIPTION LANGUAGE

Take Home Assignment 01

NAME : MUNSIF M.F.A

REG No. : EG/ 2021/ 4684

SEMESTER: 05

DATE : 26 /07 /2024

Q1.

a)

Total States = 2^3

= 8

Decimal	Binary (Q2 Q1 Q0)
0	000
1	001
2	010
3	011
4	100
5	101
6	110

b) State Diagram of 3-bit up-down ripple counter

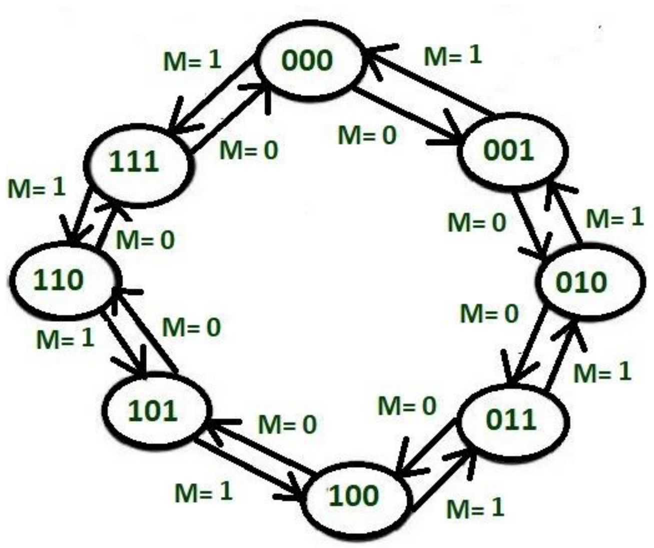


Figure 1: State Diagram of 3-bit up-down ripple counter

c)

```
module up_down_counter (
    input clk,
    input reset,
    input up_down,
    output reg [2:0] count
);
always @(posedge clk ) begin
    if (reset==1) begin
        count <= 3'b000;
    end else begin
        if (up_down==1) begin
            count <= count + 1;
        end else begin
            count <= count - 1;
        end
    end
end
endmodule
```

d)

```
module up_down_counter_tb;
reg clk;
reg reset;
reg up_down;
wire [2:0] count;

up_down_counter uut (
    .clk(clk),
    .reset(reset),
    .up_down(up_down),
    .count(count)
);

always #5 clk = !clk;
initial begin
    $dumpfile("up_down_counter.vcd");
    $dumpvars(0, up_down_counter_tb);


    clk = 0;
    reset = 1;
    up_down = 1;
    #10 ;
    reset =0;
    up_down = 1;
    #100;
     up_down = 0;
    #100;
    $stop;
end
endmodule
```

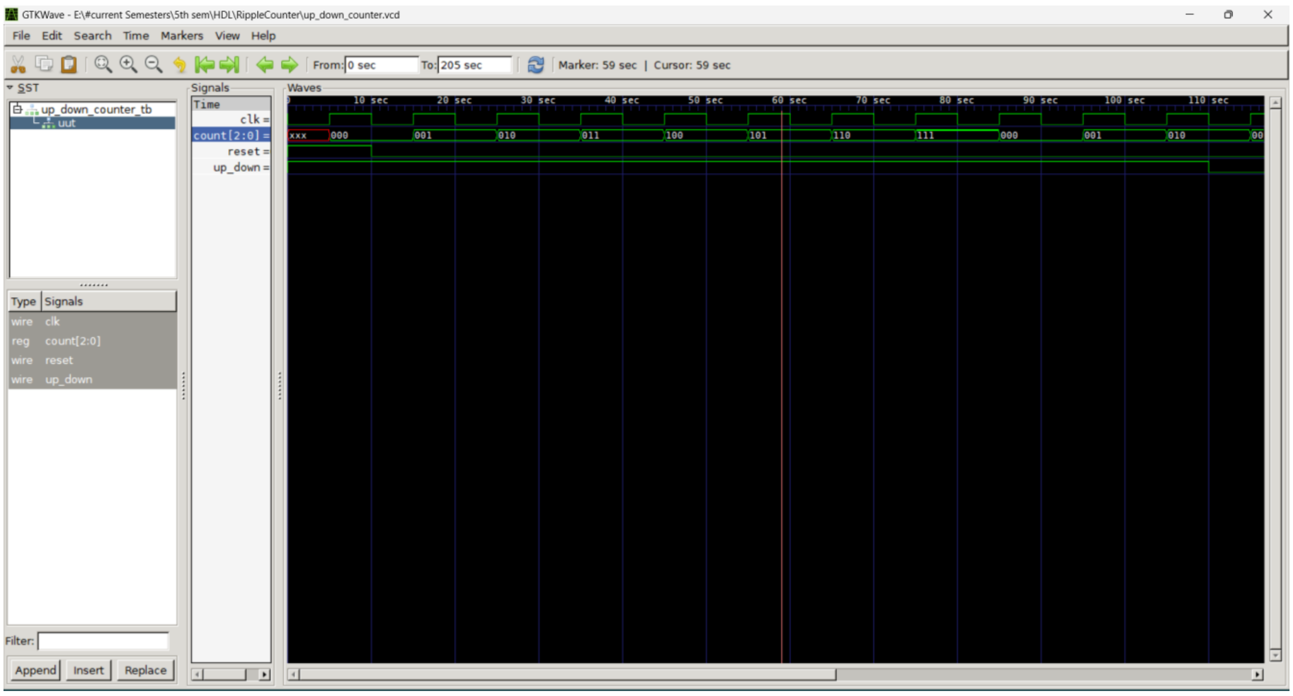


Figure 2 : Output for the test bench code

Q2.

a)

```
module mux4to1(
    input wire [3:0] data_in,  // 4-bit data input
    input wire [1:0] sel,       // 2-bit select input
    output wire out             // 1-bit output
);
    assign out = (sel == 2'b00) ? data_in[0] :
                 (sel == 2'b01) ? data_in[1] :
                 (sel == 2'b02) ? data_in[2] :
                                  data_in[3];
endmodule
```

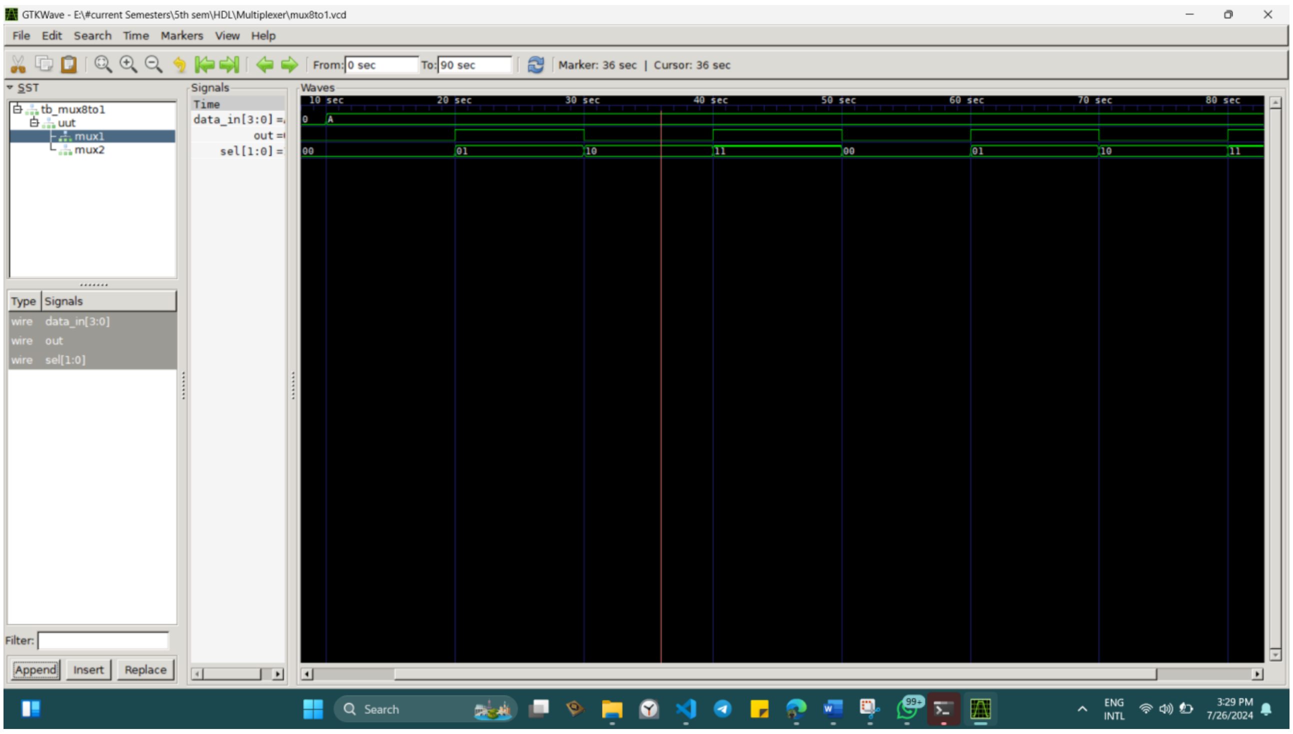
b)

```
module mux8to1(
    input wire [7:0] data_in,
    input wire [2:0] sel,
    output wire out
);
    wire out1, out2;

    mux4to1 mux1 (
        .data_in(data_in[3:0]),
        .sel(sel[1:0]),
        .out(out1)
    );

    mux4to1 mux2 (
        .data_in(data_in[7:4]),
        .sel(sel[1:0]),
        .out(out2)
    );

    assign out = (sel[2] == 1'b0) ? out1 : out2;
endmodule
```

c)

```
module tb_mux8to1;
    reg [7:0] data_in;
    reg [2:0] sel;
    wire out;
    mux8to1 uut (
        .data_in(data_in),
        .sel(sel),
        .out(out)
    );
    initial begin
        $dumpfile("mux8to1.vcd"); // Create a dump file
        $dumpvars(0, tb_mux8to1); // Dump all variables in tb_mux8to1
        data_in = 8'b00000000;
        sel = 3'b000;
        #10 data_in = 8'b10101010; sel = 3'b000;  // Expected output: 0
        #10 data_in = 8'b10101010; sel = 3'b001;  // Expected output: 1
        #10 data_in = 8'b10101010; sel = 3'b010;  // Expected output: 0
        #10 data_in = 8'b10101010; sel = 3'b011;  // Expected output: 1
        #10 data_in = 8'b10101010; sel = 3'b100;  // Expected output: 1
        #10 data_in = 8'b10101010; sel = 3'b101;  // Expected output: 0
        #10 data_in = 8'b10101010; sel = 3'b110;  // Expected output: 1
        #10 data_in = 8'b10101010; sel = 3'b111;  // Expected output: 0
        #10 $finish;
    end
    initial begin
       $monitor("Time = %0t | data_in = %b | sel = %b | out = %b", $time, data_in, sel,
out);
    end
endmodule
```

d)


GTKWave - E:\#current Semesters\5th sem\HDL\Multiplexer\mux8to1.vcd

From: 0 sec To: 90 sec Marker: 36 sec | Cursor: 36 sec

Figure 3 : for 8 to 1 multiplexer