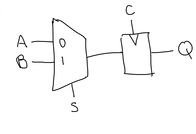
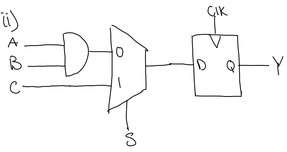
Homework 2

Thursday, July 14, 2022 9:51 AM

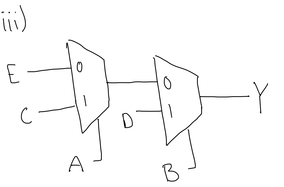
i)



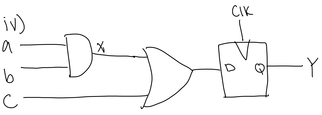
ii)



iii)



iv)



Q2:

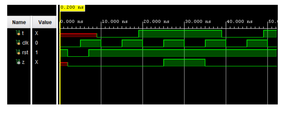
i)

```
module tflipflop(
    input t,
    input clk,
    input rst,
    output z
    );

reg CS, NS;
parameter zero = 0;
parameter one = 1;

always @ (posedge clk, negedge rst) begin
  if (rst == 1'b0) begin NS <= 0; CS <=0; end
  else begin
  CS <= NS; end
  end

always@(CS, t) begin
  case(CS)
    zero: NS = t? one : zero;
   one: NS = t?  zero : one;
   default: NS = zero;
   endcase
   end

assign z = (CS == one);
endmodule
```

ii)

```
module tflipflop_tb;
reg t, clk, rst=1;
wire z;

tflipflop UUT(
    .rst(rst),
    .z(z),
    .t(t),
    .clk(clk));

initial begin
clk = 0;
forever #5 clk = ~clk;
end

initial begin
$monitor("z = ", z);
#2 rst = 0; #5 rst = 1;
#2 t = 0;  #10 t = 1;  #20 t = 0; #10 t = 1;
#10 $finish;
end

endmodule
```

iii) Optional: Simulator Waveform



Q3:

i) Table

CNT[3]	CNT[2]	CNT[1]	CNT[0]
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
0	0	0	0

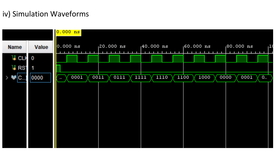
ii) Verilog

```
module QCounter #(parameter wordsize = 4)(
    input CLK,
    input RST,
    output reg [wordsize-1:0]CNT
    );

    always @ (posedge CLK, posedge RST)
                if (RST) CNT<= 0;
                else  begin
                    CNT <= {CNT[wordsize-2:0], ~CNT[3]};
                end

endmodule
```

iii) Test Bench

```
module QCounter_tb;
    reg CLK, RST =1;
    wire [3:0]CNT;

QCounter UUT(
.CLK(CLK),
.RST(RST),
.CNT(CNT)
);

initial begin
CLK=0;
forever #5 CLK=~CLK;
end

initial begin
$monitor("CNT = ", CNT);
#2 RST =0;
#100; $finish;
end
endmodule
```

iv) Simulation Waveforms



Q4:

```
module moore_fsm (
    input D,
    input clk,
    output z
);
parameter a = 2'd0, b = 2'd1, c = 2'd2, d = 2'd3;

reg [1:0] CS, NS;
always @(posedge clk) //edge sensitive behavior
begin CS<= NS; end
always @(CS, D) begin //level-sensitive behavior
    NS = a; //default state
    case (CS)
        a : NS = D ? c:b;
        b : NS = D ? d:c;
        c : NS = D ? d:b;
        d : NS = D ? a:c;
    endcase
end
assign z = (CS == b | c);
endmodule
```