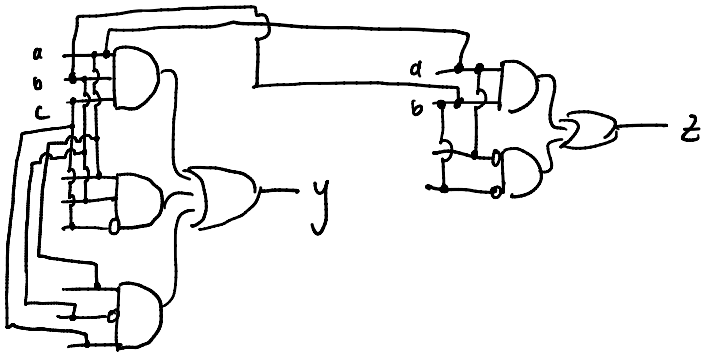
Name: _____ ID: _____

Instructions: Only neat, **hand-written** answers will be accepted (except for the sections 7b and 7c where you can use a computer). This homework assignment is **individual**. Use SystemVerilog for your answers.

1. (5%) Draw a schematic of the logic defined in the following Verilog code.

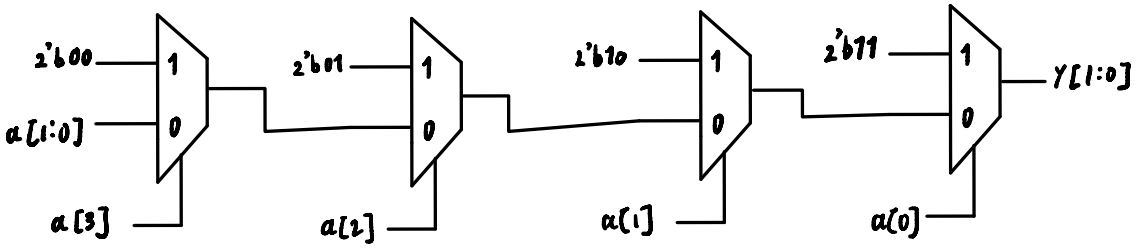
```
module exercise1 (input  a, b, c,
                  output y, z);

  assign y = a & b & c  |  a & b & ~c  |  a & ~b & c;
  assign z = a & b  |  ~a & ~b;
endmodule
```



2. (5%) Draw a schematic of the logic defined in the following Verilog code.

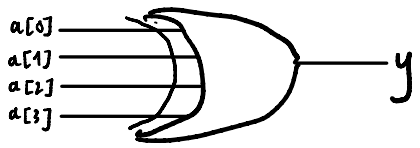
```
module exercise2 (input      [3:0] a,
                  output reg [1:0] y);

  always @(*)
    if     (a[0]) y = 2'b11;
    else if(a[1]) y = 2'b10;
    else if(a[2]) y = 2'b01;
    else if(a[3]) y = 2'b00;
    else          y = a[1:0];
endmodule
```

From Lecture 2 Page 64, Inferred Result is Priority Logic



3. (5%) Draw a schematic of the logic defined in the following Verilog code.

```
module ex3(input [3:0] a, output y);
      assign y = ^a;
endmodule
```



4. (10%) Write HDL code that implements a multiplexer, 8 input to 1 output, all 32-bit wide.

```
module mux( input logic [31:0] in0, in1, in2, in3, in4, in5, in6, in7,
            input logic [2:0] sel,
            output logic [31:0] out);

  always_comb begin
    case (sel)
      3'b000 : out = in0;
      3'b001 : out = in1;
      3'b010 : out = in2;
      3'b011 : out = in3;
      3'b100 : out = in4;
      3'b101 : out = in5;
      3'b110 : out = in6;
      3'b111 : out = in7;
      default : out = 32'b0;
    endcase
  end
endmodule
```

5. (10%) Write HDL code that implements a Priority Encoder with 8 inputs of 1 bit each and 1 output of 3 bits.

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

  always_comb begin
    casez (in)
       8'b1???????: out = 3'b111; // don't care about anything but in[7]=1 for priority logic
       8'b01??????: out = 3'b110;
       8'b001?????: out = 3'b101;
       8'b0001????: out = 3'b100;
       8'b00001???: out = 3'b011;
       8'b000001??: out = 3'b010;
       8'b0000001?: out = 3'b001;
       8'b00000001: out = 3'b000;
       default: out = 3'bxxx;
    endcase
  end
endmodule
```

6. (20%) Write HDL code to synthesize the following circuits:

a. 8-bit register.

```
module reg8 (input logic clk, input logic rst, input logic [7:0] d , output logic [7:0] q);

    always_ff @(posedge clk) begin
     if(rst)
       q <= 8'b0
     else
       q <=d;
    end
endmodule
```

b. 9-bit Register with Asynchronous Reset

```
module reg9 (input logic clk, input logic rst, input logic [8:0] d, output logic [8:0] q);

    always_ff(posedge clk, posedge rst) begin
      if (rst)
        q <= 9'b0
      else
        q <= d
    end
endmodule
```

c. N-bit Register with Synchronous Reset where N is a parameter

```
module regN #(parameter n=8) (input logic clk, input logic rst, input logic [n-1:0] d, output logic [n-1:0] q);

    always_ff @(posedge clk) begin
     if(rst)
       q <= '0;
     else
       q <=d;
    end
endmodule
```

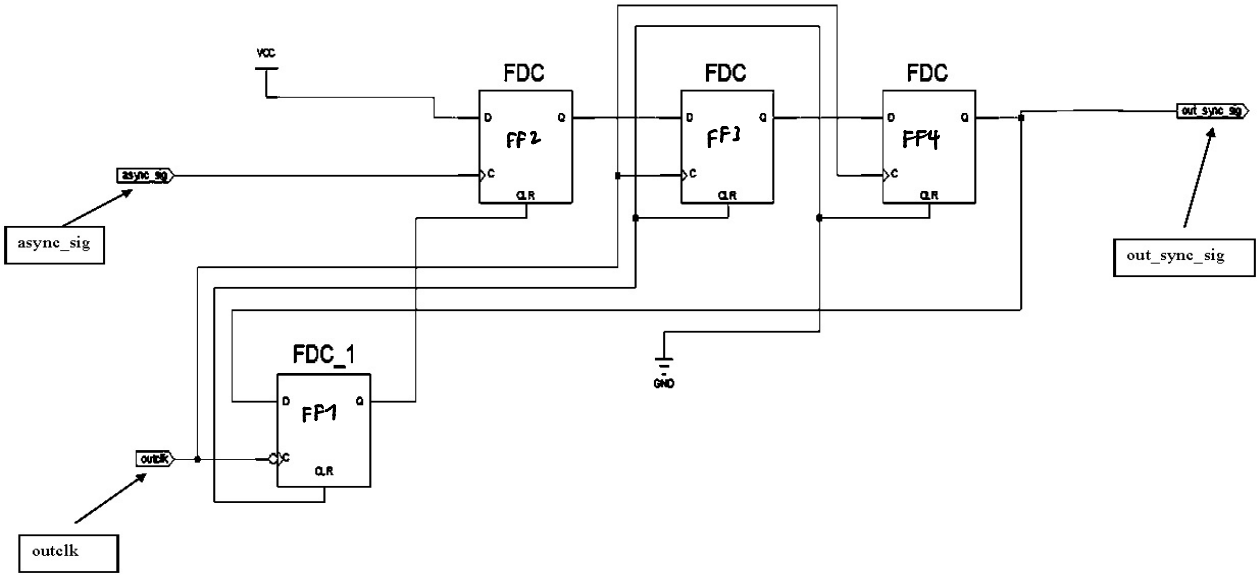
d. N-bit register with Enable and Asynchronous reset
where N is a parameter

```
module regNER #(parameter n=8) (input logic [n-1:0] d, input logic clk, input logic rst, input logic en, output logic [n-1:0] q);

    always_ff (posedge clk, posedge rst) begin
        if (rst) begin
         q <= '0;
        end else if (en) begin
          q <= d;
        end
endmodule
```

e. 8-bit latch

```
module latch8 (input logic clk , input logic [7:0] d , output logic [7:0] q);

    always_latch
      if (clk)
        q <= d;
endmodule
```

(45%) 7. Look at the diagram below and answer the questions that follow.



The "clr" of Flip-Flops is asynchronous and active high. Note the inversion (circle) on the "C" (clock) input of FDC_1.

(a) (25%) Write HDL code that will result in the synthesis of this circuit.

(b) (20%) (use a computer for this section): In Quartus, create a project and synthesize your code (you do not need to do a full compilation, just analysis and synthesis). Submit a simulation showing that your circuit is operating (if you don't understand what it is supposed to do, look at its inputs and outputs and devise a simulation that will cause those inputs and outputs to toggle).

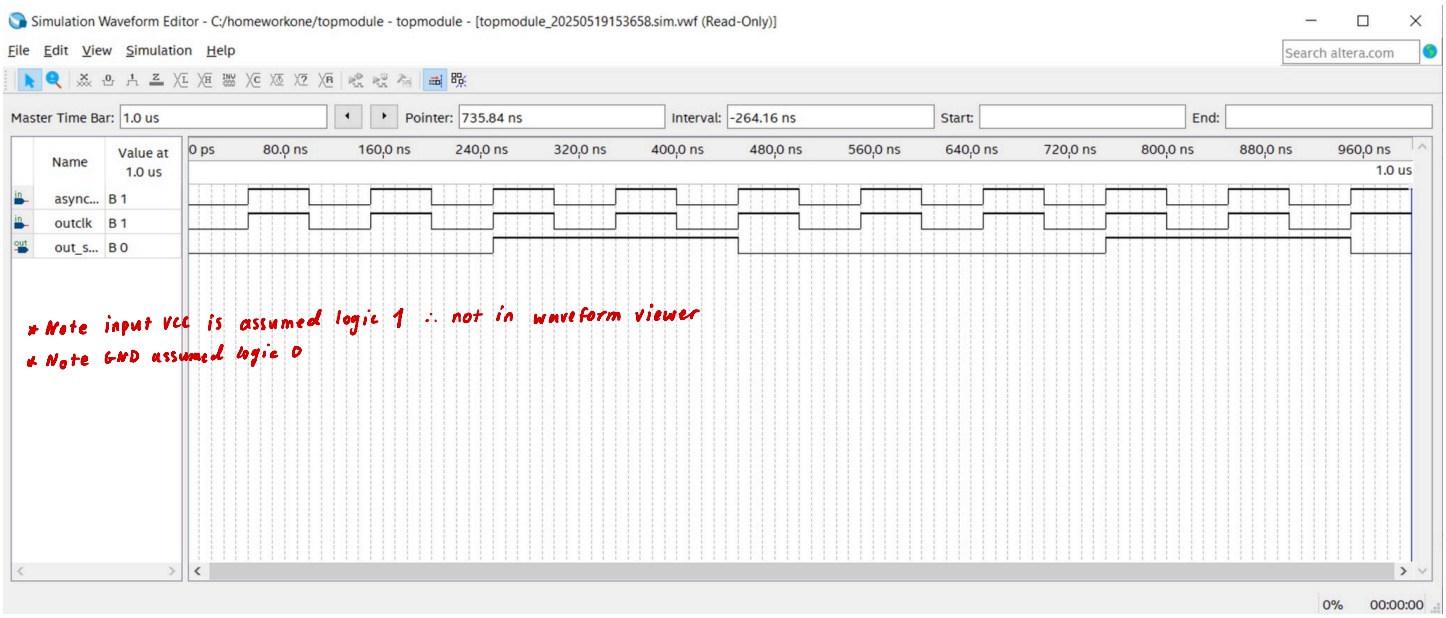
(c) (10% Bonus) What does the circuit do? What could it be useful for? Explain how it works (an annotated simulation may be helpful, you can use a computer for this section).

7)

a)

```
module topmodule(input async_sig, input logic outclk, output logic out_sync_sig);
    logic d1, d2, clr1;

    idff ff1 (.clk(outclk), .clr(1'b0), .d(out_sync_sig), .q(clr1));
    my_dff ff2 (.clk(async_sig), .clr(clr1), .d(1'b1), .q(d1));
    my_dff ff3(.clk(outclk), .clr(1'b0), .d(1'b1), .q(d1));
    my_dff ff4(.clk(outclk), .clr(1'b0), .d(d2), .q(out_sync_sig));
endmodule

module my_dff(input logic clk, input logic clr, input logic d, output logic q);

    always_ff@(posedge clk or posedge clr) begin
        if (clr)
            q <= 1'b0;
        else
            q <= d;
    end
endmodule

module idff (input logic clk, input logic clr, input logic d, output logic q);

    always_ff@(negedge clk or posedge clr) begin
        if (clr)
            q <= 1'b0;
        else
            q <= d;
    end
endmodule
```

b)



c) seperate document