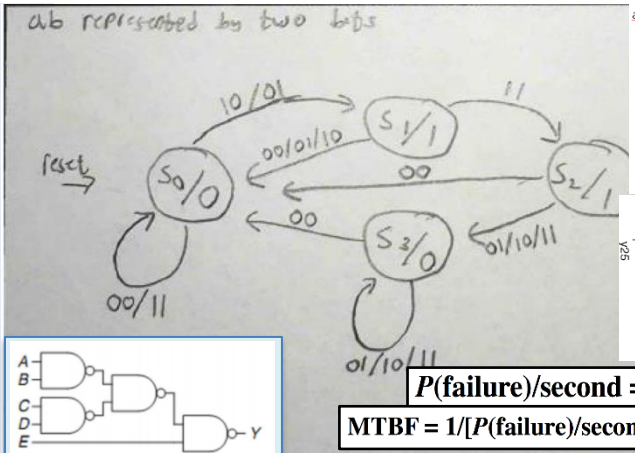
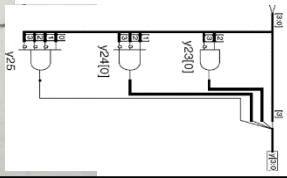
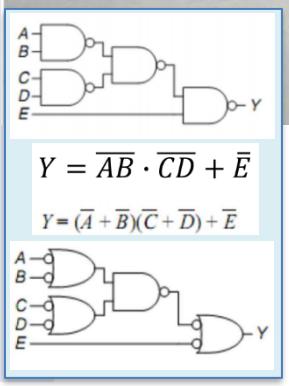
```
module fsm2(input  logic clk, reset,
            input  logic a, b,
            output logic y);

  logic [1:0] state, nextstate;

  parameter  S0 = 2'b00;
  parameter  S1 = 2'b01;
  parameter  S2 = 2'b10;
  parameter  S3 = 2'b11;

  always_ff @(posedge clk, posedge reset)
    if (reset) state <= S0;
    else       state <= nextstate;

  always_comb
    case (state)
      S0: if (a ^ b) nextstate=S1;
          else       nextstate=S0;
      S1: if (a & b) nextstate=S2;
          else       nextstate=S0;
      S2: if (a | b) nextstate=S3;
          else       nextstate=S0;
      S3: if (a | b) nextstate=S3;
          else       nextstate=S0;
    endcase

  assign y=(state==S1) | (state==S2);
endmodule
```




```
always_comb
  casez(a)
    4'b1???: y = 4'b1000;
    4'b01??: y = 4'b0100;
    4'b001?: y = 4'b0010;
    4'b0001: y = 4'b0001;
    default: y = 4'b0000;
  endcase
```





$$Y = \overline{AB} \cdot \overline{CD} + \bar{E}$$

$$Y = (\bar{A} + \bar{B})(\bar{C} + \bar{D}) + \bar{E}$$

$$P(\text{failure})/\text{second} = (NT_0/T_c)\, e^{-(T_c - t_{setup})/\tau}$$

$$\text{MTBF} = 1/[P(\text{failure})/\text{second}] = (T_c/NT_0)\, e^{(T_c - t_{setup})/\tau}$$

Suppose: T_c = 1/500 MHz = 2 ns τ = 200 ps

T_0 = 150 ps t_{setup} = 100 ps

N = 1 events per second

What is the probability of failure? MTBF?

$$P(\text{failure}) = (150\text{ ps}/2\text{ ns})\, e^{-(1.9\text{ ns})/200\text{ ps}}$$

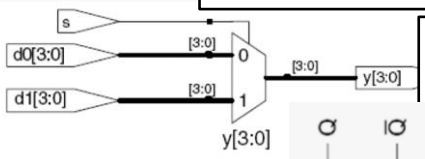
$$= 5.6 \times 10^{-6}$$

$$P(\text{failure})/\text{second} = 10 \times (5.6 \times 10^{-6})$$

$$= 5.6 \times 10^{-5} / \text{second}$$

$$\text{MTBF} = 1/[\text{P(failure)/second}] \approx 5 \text{ hours}$$

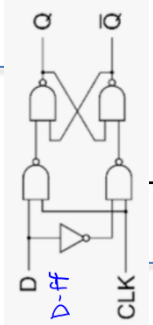
```
module mux2(input  logic [3:0] d0, d1,
            input  logic       s,
            output logic [3:0] y);
   assign y = s ? d1 : d0;
endmodule
```



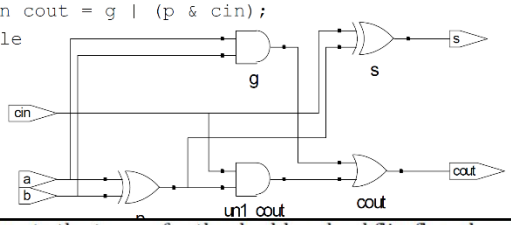
Resettable D Flip-Flop

```
module flopr(input  logic       clk,
             input  logic       reset,
             input  logic [3:0] d,
             output logic [3:0] q);

  // synchronous reset
  always_ff @(posedge clk)
    if (reset) q <= 4'b0;
    else       q <= d;

endmodule
```

```
  // asynchronous reset
  always_ff @(posedge clk, posedge reset)
    if (reset) q <= 4'b0;
    else       q <= d;
```

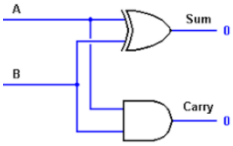



```
module fulladder(input  logic a, b, cin,
                 output logic s, cout);
  logic p, g;    // internal nodes

  assign p = a ^ b;
  assign g = a & b;

  assign s = p ^ cin;
  assign cout = g | (p & cin);
endmodule
```

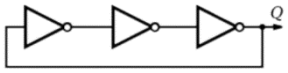


Verilog design

```
module half_adder(
input a,b,
output sum,carry);

  assign sum = a^b;
  assign carry = a & b;

endmodule
```

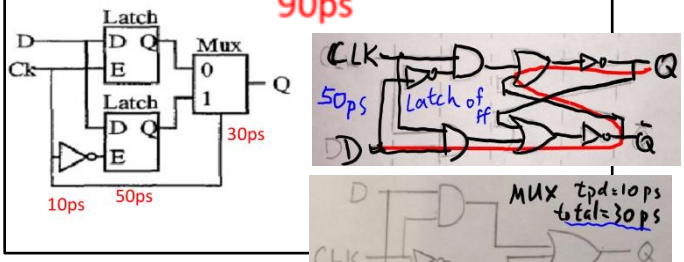


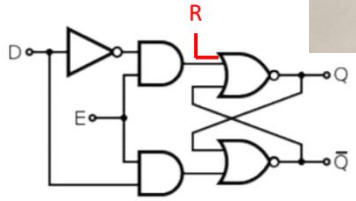
Inputs		Outputs	
CK	D	Q	$\bar{Q}$
0	X	No change	
1	0	0	1
1	1	1	0




```
module latch(input logic clk,
             input  logic [3:0] d,
             output logic [3:0] q);
  always_latch
    if (clk) q <= d;
endmodule
```

5. Compute the t_{pcq_deff} for the double-edged flip-flop above





Design an asynchronously resettable D latch using logic gates.

Range of frequencies:

From $\frac{1}{2Nt_{pd}}$ to $\frac{1}{2Nt_{cd}}$.

~	NOT
*, /, %	mult, div, mod
+, -	add,sub
<<, >>	shift
<<<, >>>	arithmetic shift
<, <=, >, >=	comparison
==, !=	equal, not equal
&, ~&	AND, NAND
^, ~^	XOR, XNOR
|, ~|	OR, NOR
?:	

S	R	Q(t)	Q(t+Δ)	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	not allowed
1	1	1	X	

```
module xor_4(input logic [3:0] a,
             output logic      y);

  assign y = ^a;
endmodule
```

```
module rslatch1 (output wire q, qbar,
                 input logic r, s);
  nand n0(q, qbar, r);
  nand n1(qbar, q, s);
endmodule
```

$$Q(t + \Delta) = S + R'Q(t)$$

```
nodule rslatch2 (output logic q, qbar,
                 input logic r, s);
ılways @(r, s)
  unique case ({r, s})
    2'b00: {q, qbar} <= 2'b11;
    2'b01: {q, qbar} <= 2'b10;
    2'b10: {q, qbar} <= 2'b01;
    default ;
  endcase
endmodule
```

Q(t)	S=0, R=0	S=0, R=1	S=1, R=0	S=1, R=1
0	0	0	X	1
1	1	0	X	1