Mini Project - Digital System Design: EE311

Roll Number: 16EE234

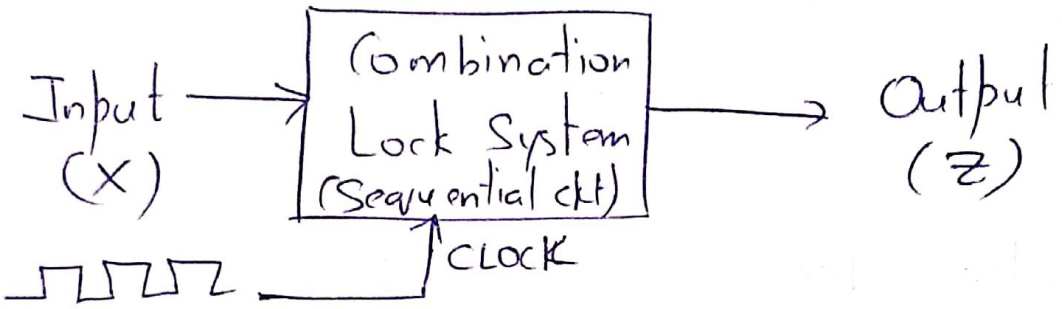
Name: Megh Manoj Bhalerao

Question # 3

Design a controller for a three-number digital combination lock. The lock has a 4-bit register that accepts entered numbers (0-9) from the keypad. Each number will be stable for a minimum of 200ms from the time a number valid signal NV is asserted after the entered number has become stable. The system has a 1kHz clock available. If correct sequence of three numbers is entered an output signal UNLK is asserted for 1 second. If three incorrect sequences of numbers are entered successively, the control circuit should become inactive for approximately 5 minutes. Show all important design steps and state any reasonable assumptions made.

Solution:

⊗ Block diagram – Simplified Version.



X → How is X calculated?

→ X is calculated based on the entered code and the already existing code (which is the correct code for the machine).

X = 0 → If code is ~~same~~ different (No match)

X = 1 → If code is same, i.e it matches.

How is X implemented?

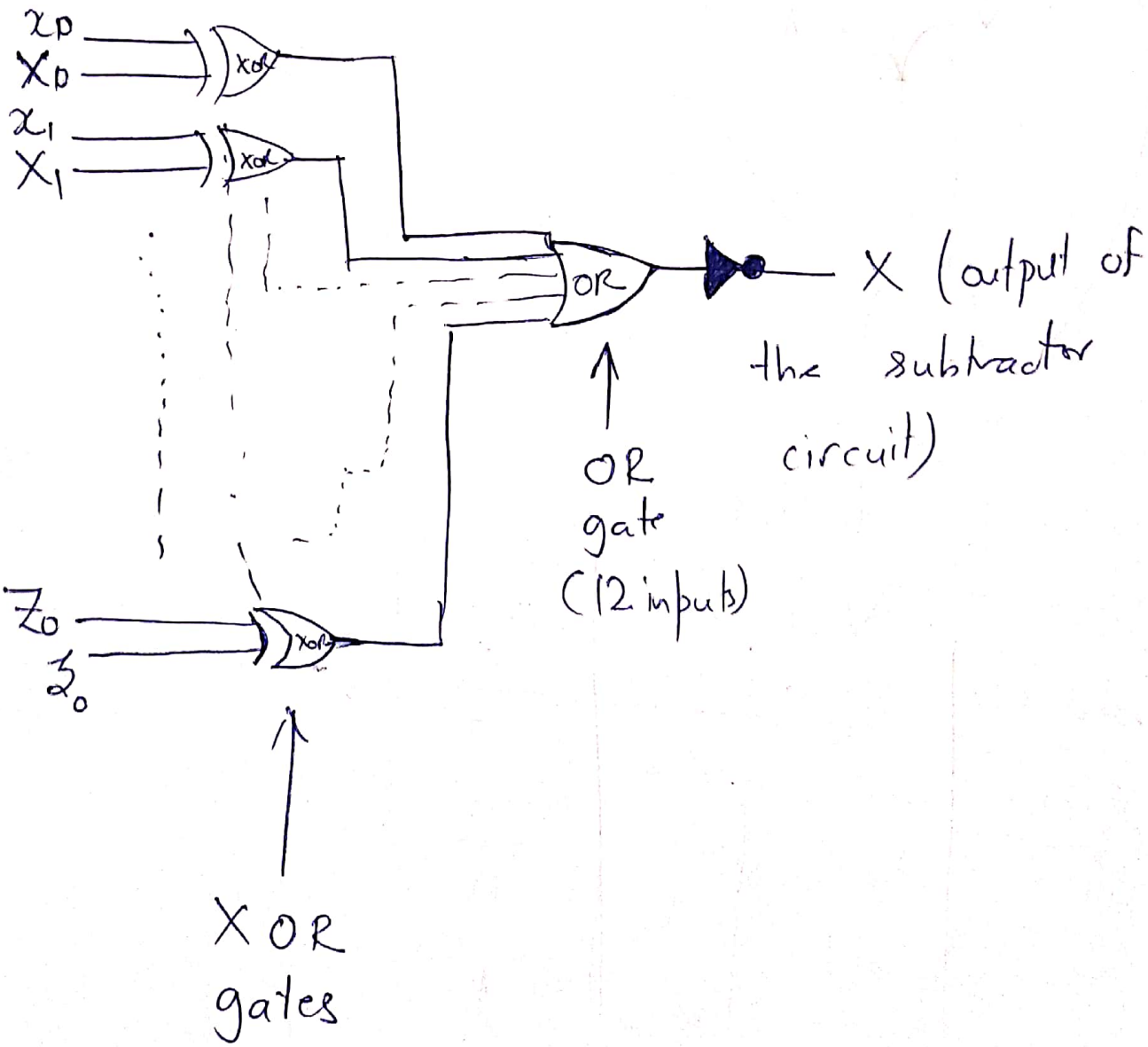
X → can be implemented using a simple subtractor circuit which calculates the difference between the given input and the true key.

Input stored as 4 bit registers (but total 3 digits)

Input → | X_3 | X_2 | X_1 | X_0 | Y_3 | Y_2 | Y_1 | Y_0 | Z_3 | Z_2 | Z_1 | Z_0 |

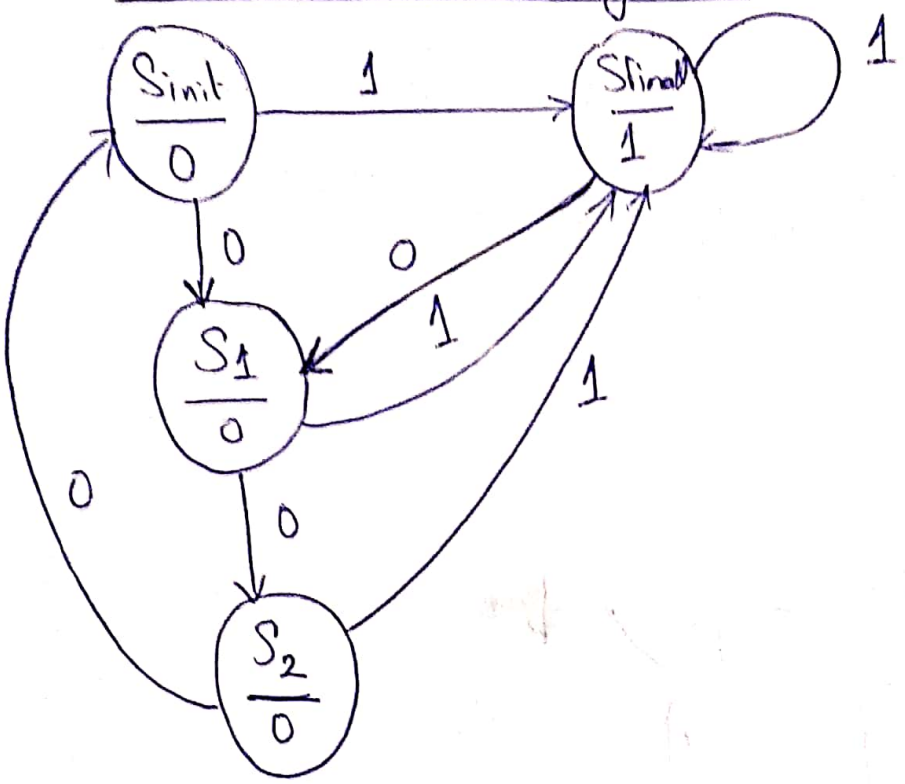
Adual Combination → | x_3 | x_2 | x_1 | x_0 | y_3 | y_2 | y_1 | y_0 | z_3 | z_2 | z_1 | z_0 |

Need to have 12 subtractors (XOR gates) to compare (& even if 1 is not same the ouput should be False i.e $x=0$)



Now we have to implement the main logic of the circuit of the digital combination lock. Hence we use the state diagram to obtain the following.

Moore state diagram



State table:

State	Next state x=0	Next state x=1	Output (z)
S_{init}	S_1	S_{final}	0
S_1	S_2	S_{final}	0
S_2	S_{init}	S_{final}	0
S_{final}	S_1	S_{final}	1

Converting the state diagram and encoding the states into binary symbols:-

Q_0Q_1	$Q_0^+Q_1^+$ x=0	$Q_0^+Q_1^+$ x=1	Output (z)
00	01	11	0
01	10	11	0
10	00	11	0
11	01	11	1

⇓ Truth table

Q_0	Q_1	X	Q_0^+	Q_1^+	Z
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	1	1	1

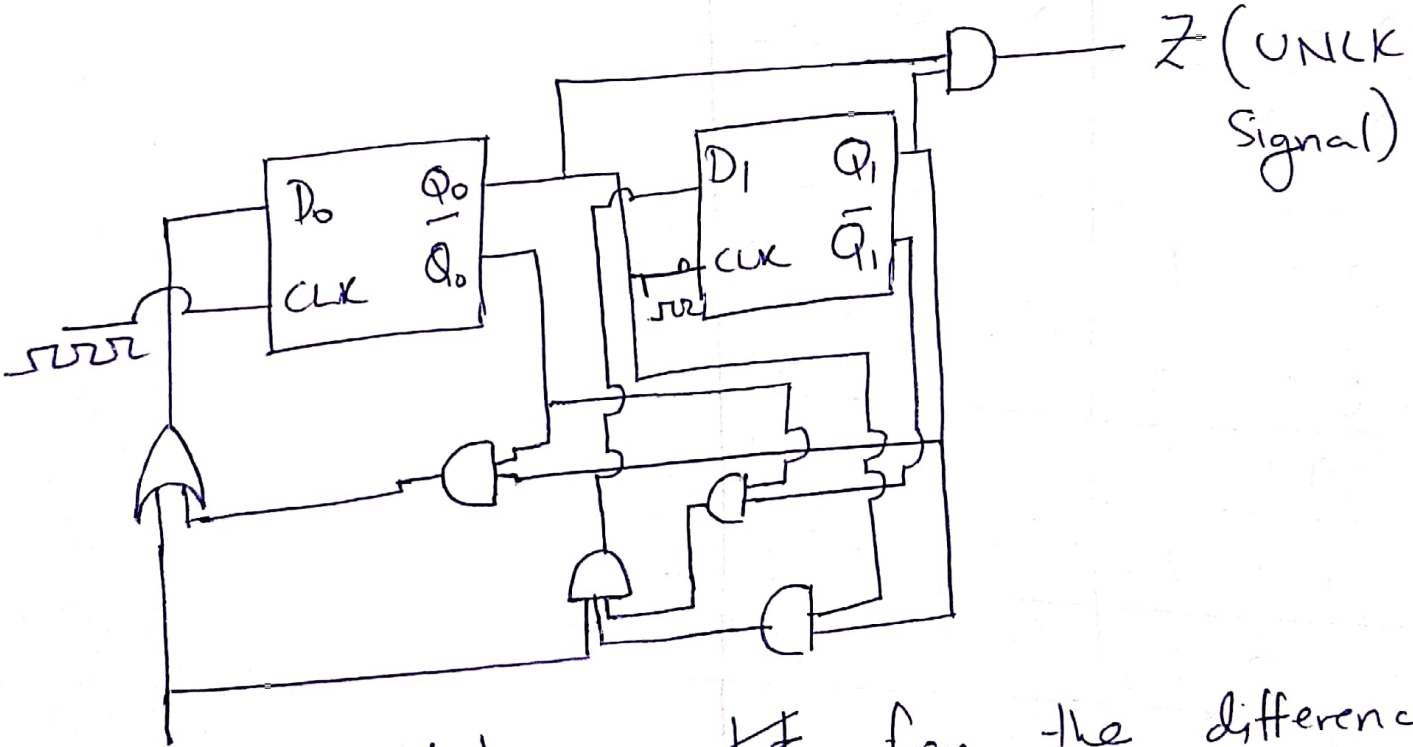
On simplifying the Truth tables:

$$Q_0^+ = X + \bar{Q}_0 Q_1 \longrightarrow D_0$$

$$Q_1^+ = X + \bar{Q}_0 \bar{Q}_1 + Q_0 Q_1 \longrightarrow D_1$$

$$Z = Q_0 Q_1$$

Sequential circuit Using D - Flip Flops:



X (This input comes ~~bet~~ from the difference between stored input and given input)

We use a counter circuit to count upto the given number:

(*) Since we have to wait for 1 second after Z=1 start counting after Z=1 for 1000 cycles of the clock

Since f: 1000 Hz

(*) We also have to wait for 5 mins after 3 incorrect Inputs → i.e 300 x 1000 clock cycles → hence we need to used $int(300 \times 1000)$ ~~int(3000 x 1000)~~ $int\left(\log_2(300 \times 1000)\right) + 1$

counters = 19 ~~counters~~ Flip flops.

Implementation of delay circuits:

⊛ We have some delay circuits between the states (as mentioned in the question)

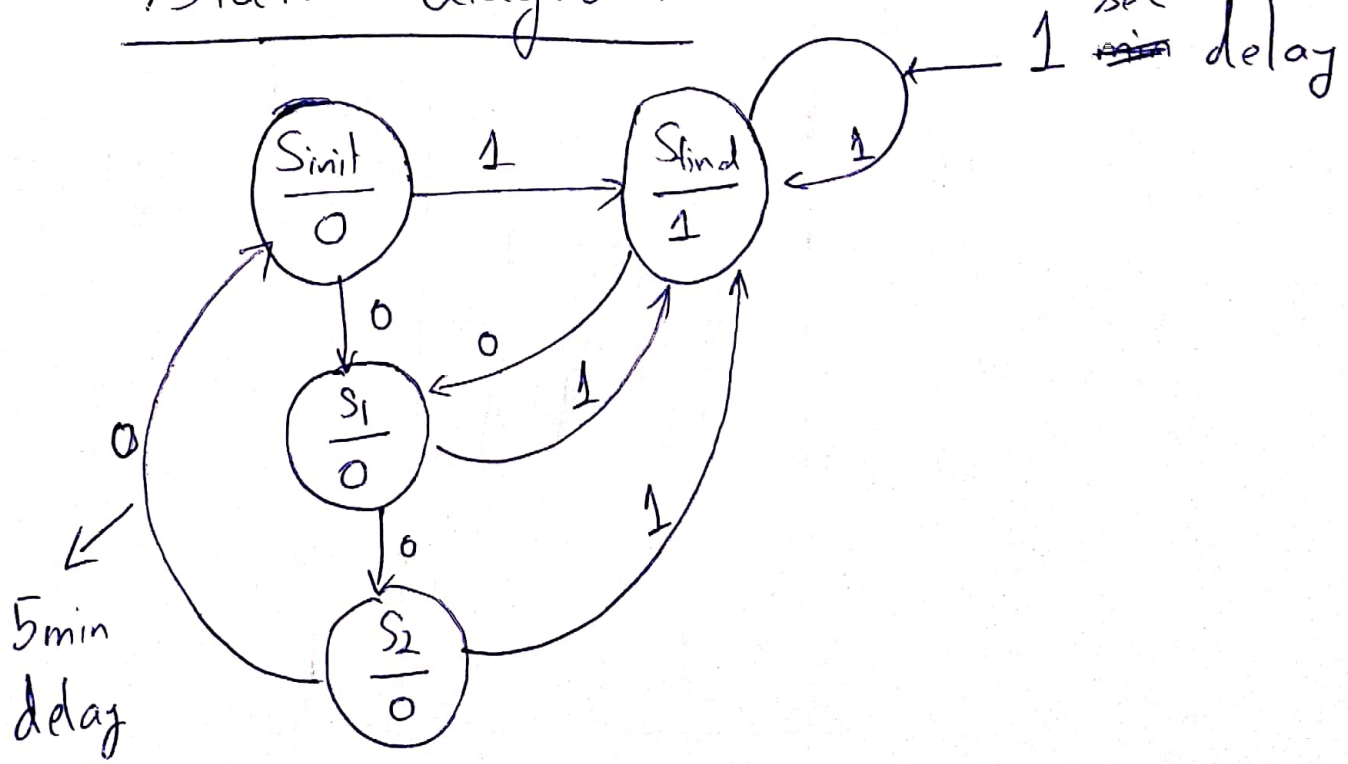
Drawing the state table again:

States	Next state X=0	Next state X=1	Output (Z)
S_{init}	S_1	S_{final}	0
S_1	S_2	S_{final}	0
S_2	S_{init}	S_{final}	0
S_{final}	S_1	S_{final}	1

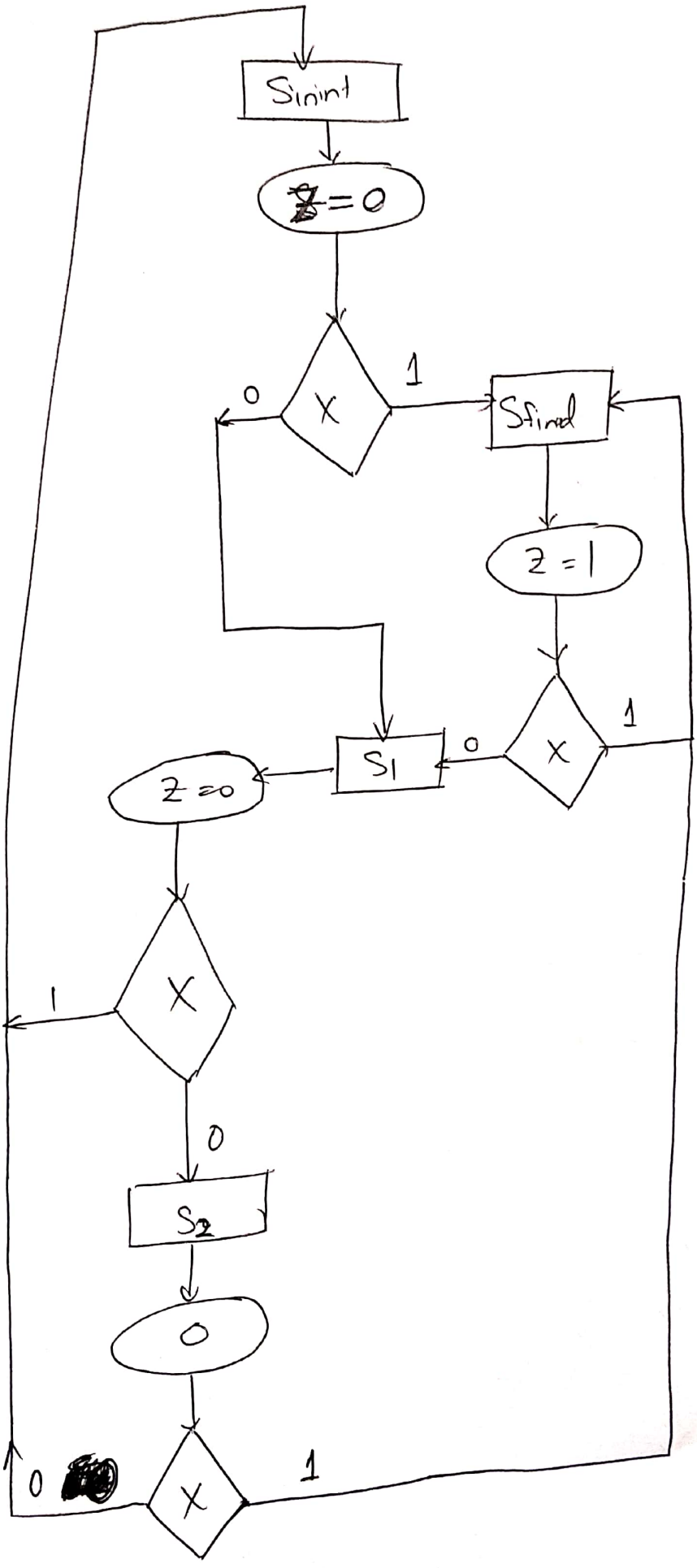
5min

Delay = 1 sec

State diagram:



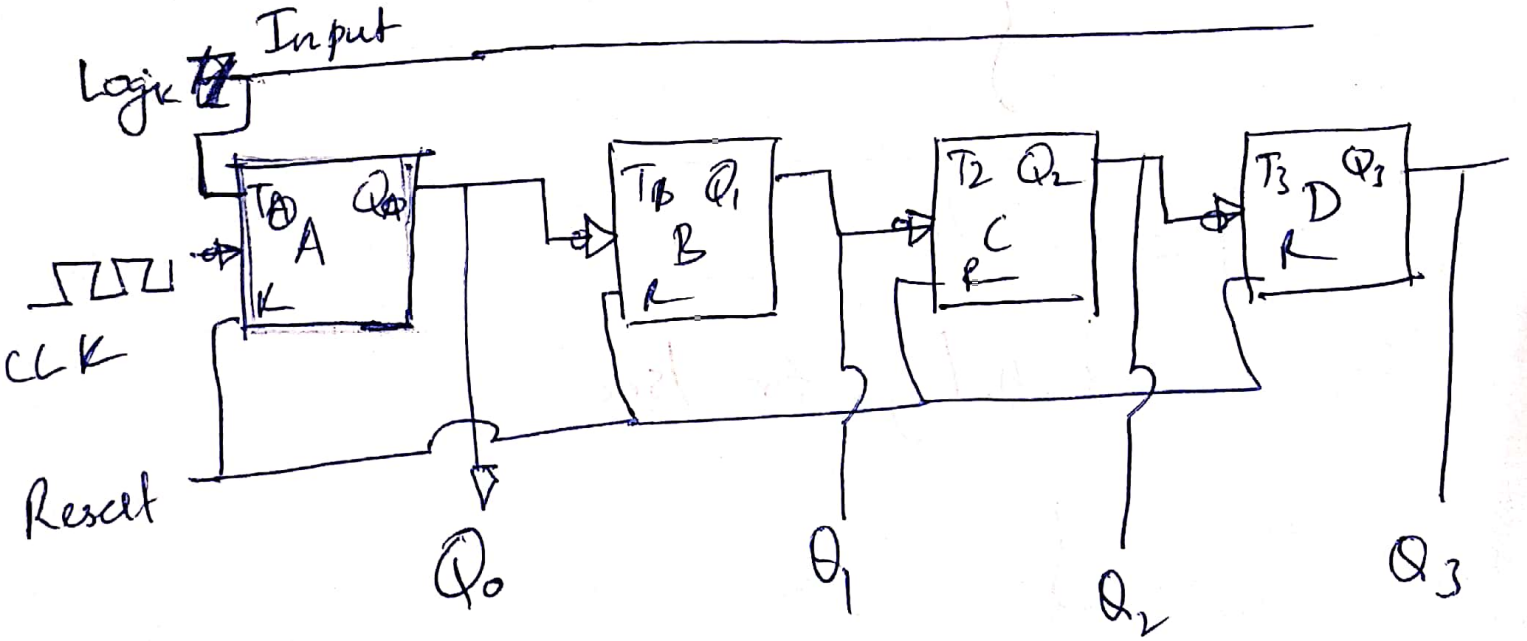
ASM3



Hence, we have to design 19 bit upcounter ~~when~~ to implement delay of 1 sec & max of 5 mins:-

Since 19 is a very high number:

Let's see the design using 4 bit asynchronous Counter: (T-Flip flops)



Q Binary Representation of 300x1000

= 100100100 11111 00000

Take Normal of 1's bit and complement of 0's bits and AND them.

AND this and Z to activate ~~the~~ the counter

→ We also have if S_2 → then delay for 5 mins as mentioned previously

→ S_2 → 10 → ~~$Q_0\bar{Q}_0$~~ $Q_0\bar{Q}_1$

if $(Q_0\bar{Q}_1) == 1$

{ Delay for 5mins

}

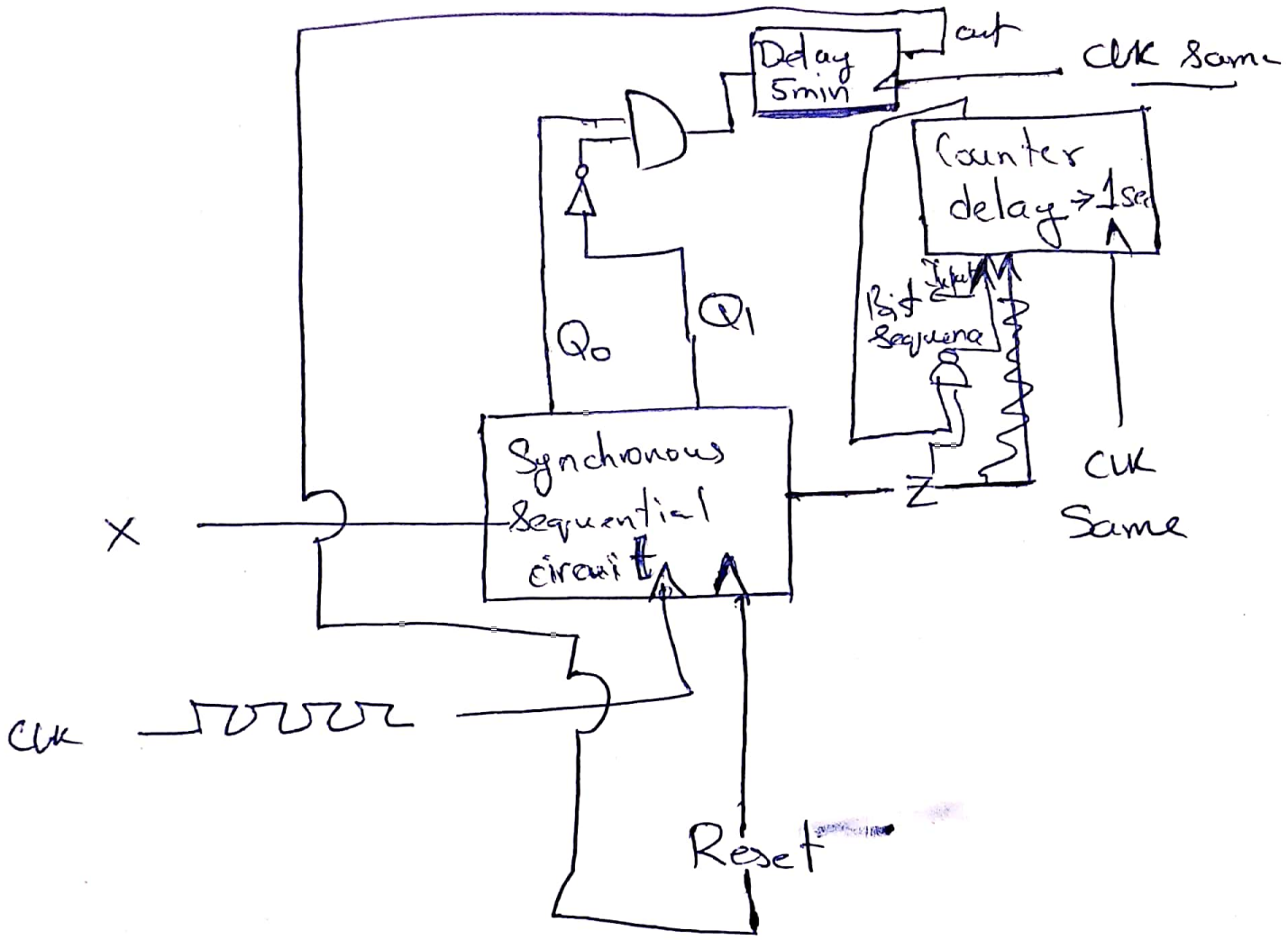
if $(Z == 1)$

{ Delay for 1sec

}

The complete block diagram looks like:

P.T.O.



Assumptions:

(1) No input is given when the system is in delay state

Verilog code:

```
module lock (clk, d, UNLK, LK);

  input clk;
  input [3:0] d
  output UNLK, LK;
  reg [3:0] r, P1, P2, P3
  reg [1:0] i
  reg [2:0] state, next state
  reg NV, LK, UNLK
//time scale : 1ms/1ms
initial
    begin
      P1 = 4'b0010 ; //password is set to 118
      P2 = 4'b0101 ;
      P3 = 4'b1000;
      state=0; nextstate=0;
    UNLK =0; LK=0;
      NV=0;
      i=0 ; end
```

```
2): begin
      if (q == p3) begin
       UNLK =1; #60000      // 60,000ms = 1second
       UNLK = 0;
      next state =0;
    end
    else begin
        i=i+1;
        if (i==3) begin
        LK=1 ; #300000   = 5mins
        LK=0 ;
        i=0
     end
     next stak =0;
  end
 end

3: begin
     next state = 4;
   end

4: begin
     i=i+1;
     if (i==3) begin
     LK =1; # 300,000
     LK = 0;
     i = 0;
   end
  nextstate =0;
```

2.

```
always @(q) // change in q  main loop

  begin
     case (state)

       0: begin
            if (q == p1) begin
            nextstate = 1;
            end
          else begin
           nextstate =3;
           end
         end

  1: begin
       if (q == p2)  begin
          nextstate = 2;
        end
     else begin
       nextstate= 4;

   end
end
```

2.

```
    end
  endcase
 end
end module
```

// Change in q is ignored during above delays as they are declared inside always block.

```
always @ (d)  // gen NV signal for 200 ns after i/p
  begin
    if ( d != 4'b0000)
      begin
        NV = 1; #200  // timescale is (ns/1ns)
        NV = 0;       // # 200 implies 200ns
      end
  end

always @ (negedge clock)
  begin
    state <= nextstate;
    if (NV != 1 && d != 4'b0000)
      begin
        q <= d
      end
  end
```