

Mini Project - Digital System Design: EE311

Roll Number: 16EE234

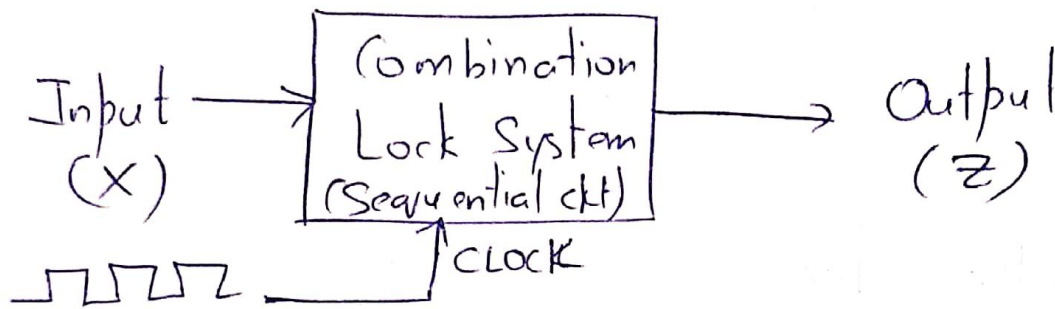
Name: Megh Manoj Bhalerao

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.



$X \rightarrow$ How is X calculated?

$\rightarrow X$ is calculated based on the entered code and the already existing code (which is the correct code for the machine):

$X = 0 \rightarrow$ If code is ~~same~~ different (No match)
 $X = 1 \rightarrow$ If code is same, i.e. it matches.

How is X implemented?

$X \rightarrow$ 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)

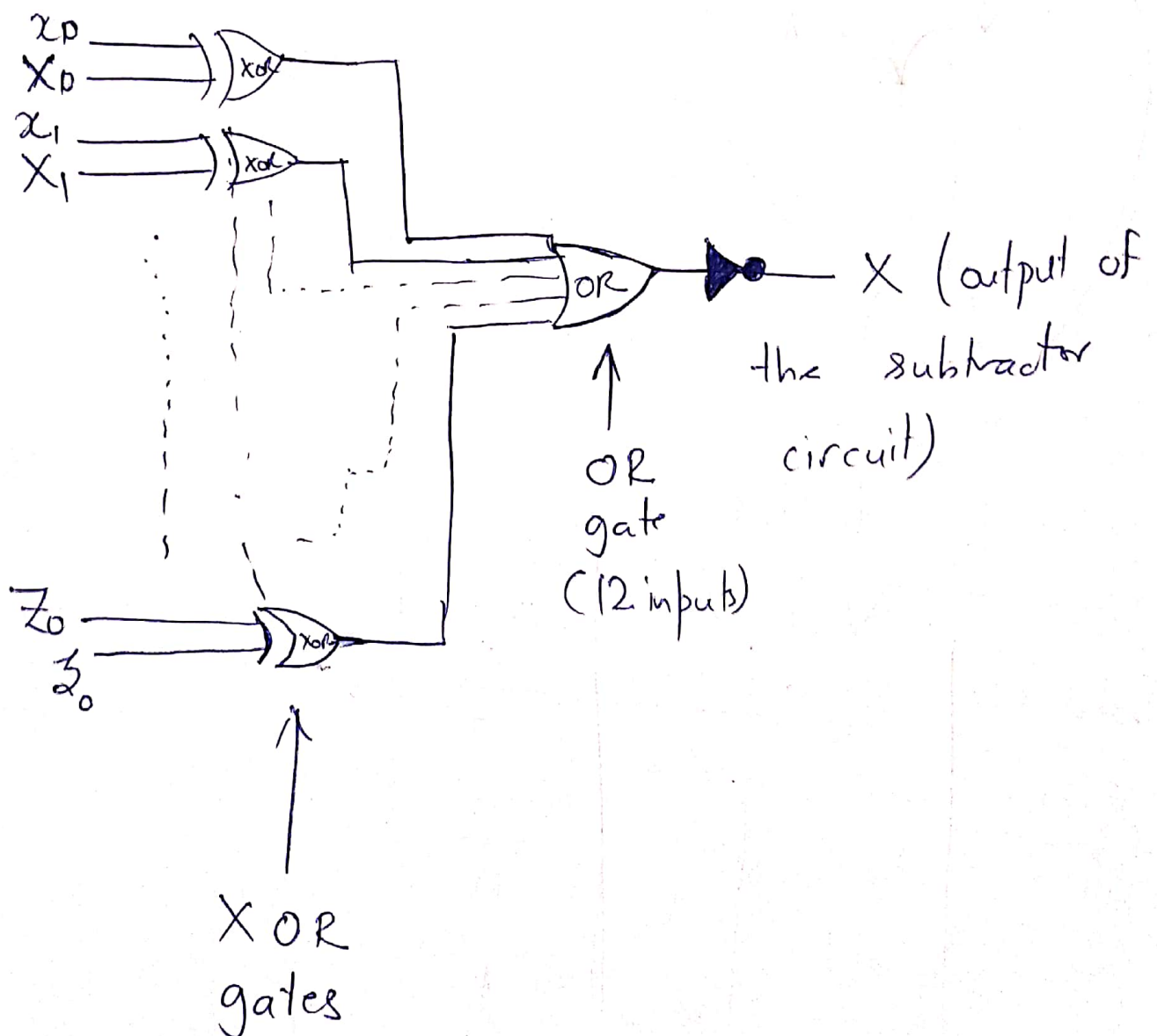
Input \rightarrow

X_3	X_2	X_1	X_0	Y_3	Y_2	Y_1	Y_0	Z_3	Z_2	Z_1	Z_0
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

Actual Combinator \rightarrow

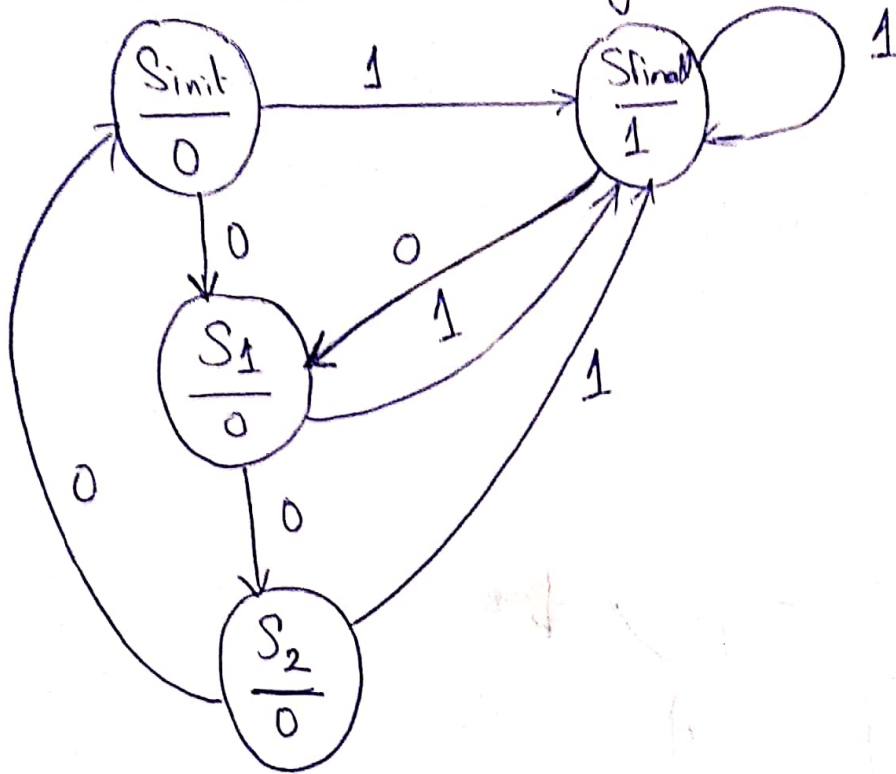
x_3	x_2	x_1	x_0	y_3	y_2	y_1	y_0	z_3	z_2	z_1	z_0
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

Need to have 12 subtractors (XOR gates) to compare (& even if 4 is not some the output 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		Output (Z)
	X=0	X=1	
Sinit	S ₁	Sfinal	0
S ₁	S ₂	Sfinal	0
S ₂	Sinit	Sfinal	0
Sfinal	S ₁	Sfinal	1

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

$Q_0 Q_1$	$Q_0^+ Q_1^+$		Output (Z)
	X=0	X=1	
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

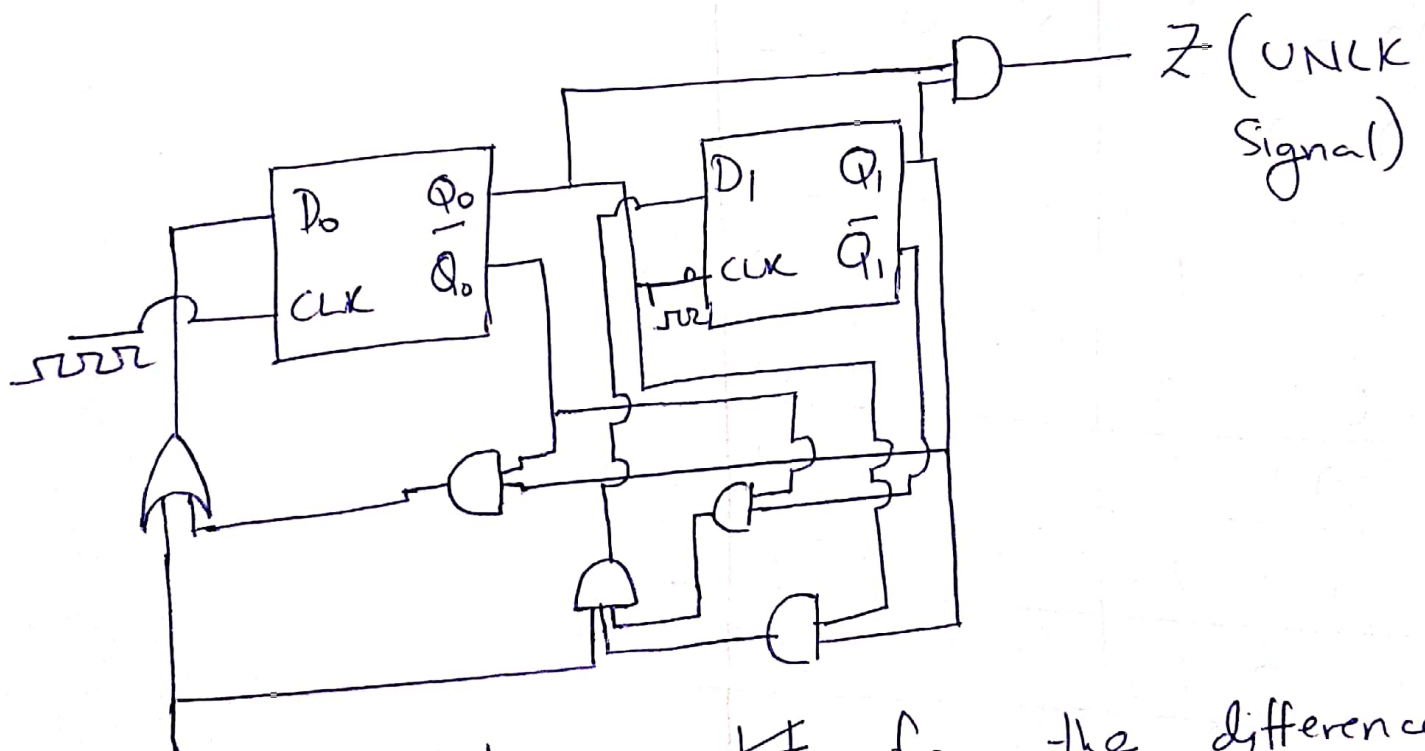
On simplifying the truth tables:

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

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

$$Z = Q_0 Q_1$$

Sequential circuit Using D-Flip Flops:



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

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

- ④ Since we have to wait for 1 second after $Z=1$ start counting after $Z=1$ for 1000 cycles of the clock

Since $f = \frac{1}{1000} \text{ Hz}$



- ⑤ We also have to wait for 5 mins after 3 incorrect Inputs \rightarrow i.e 300×1000 clock cycles \rightarrow hence we need to

used ~~$\text{int}(3000 \times 1000)$~~ $\text{int}(\log_2(300 \times 1000)) + 1$

counters = 19 ~~counters~~ Flip flops.

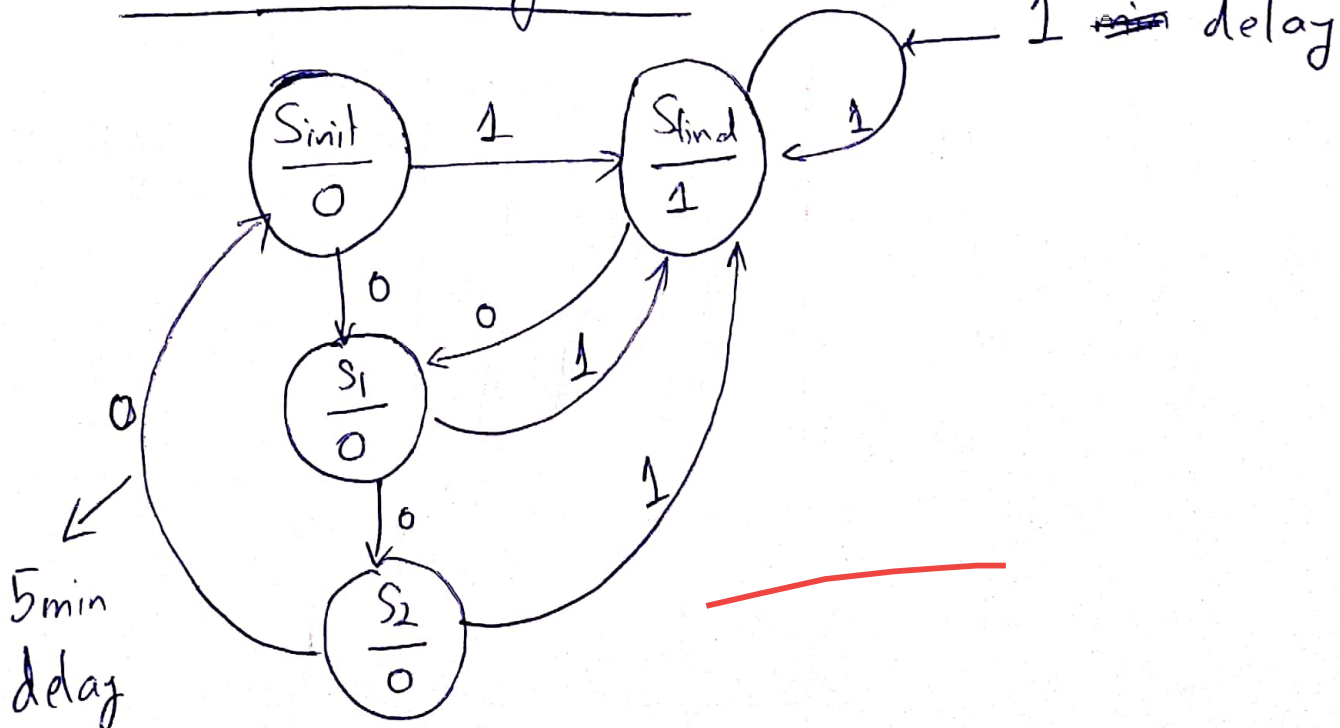
Implementation of delay circuits:

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

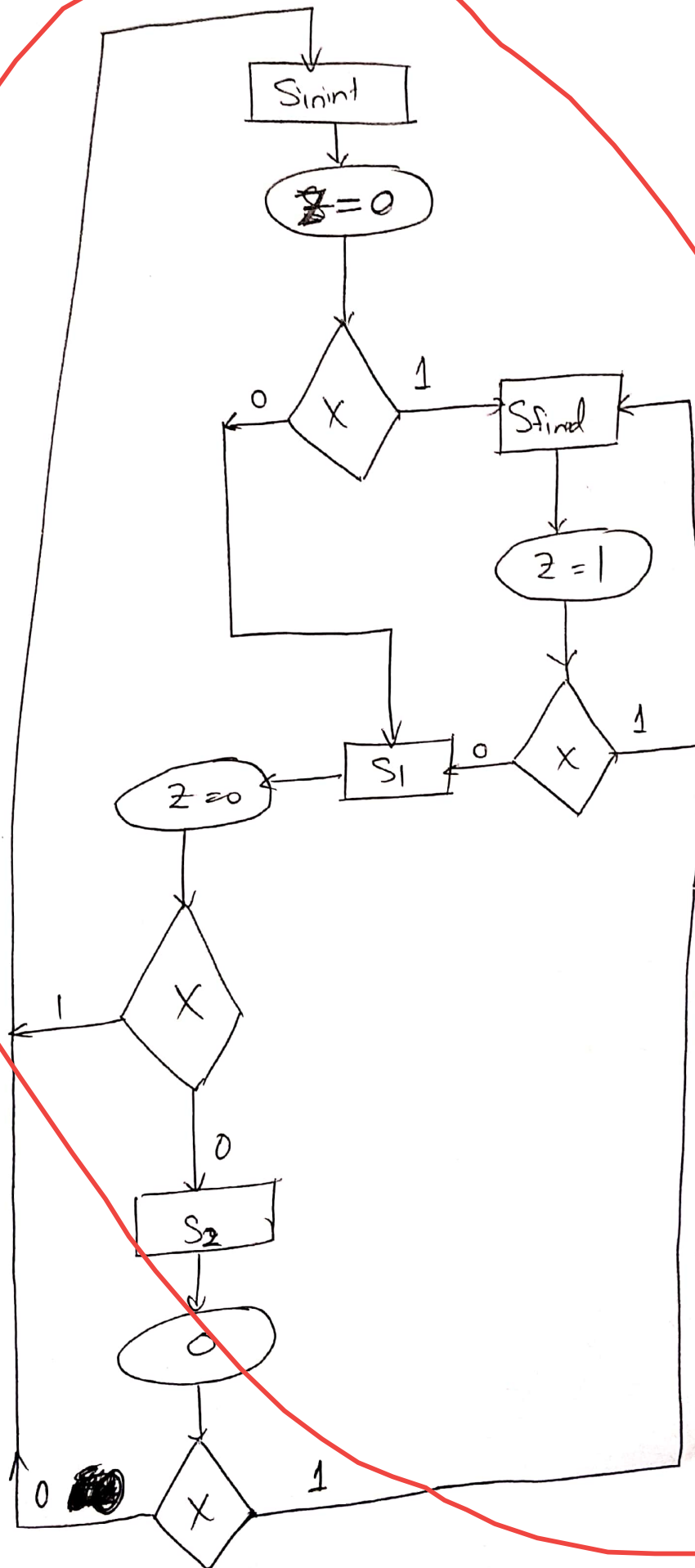
Drawing the state table again:

States	Next state		Output (Z)
	X=0	X=1	
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

State diagram :

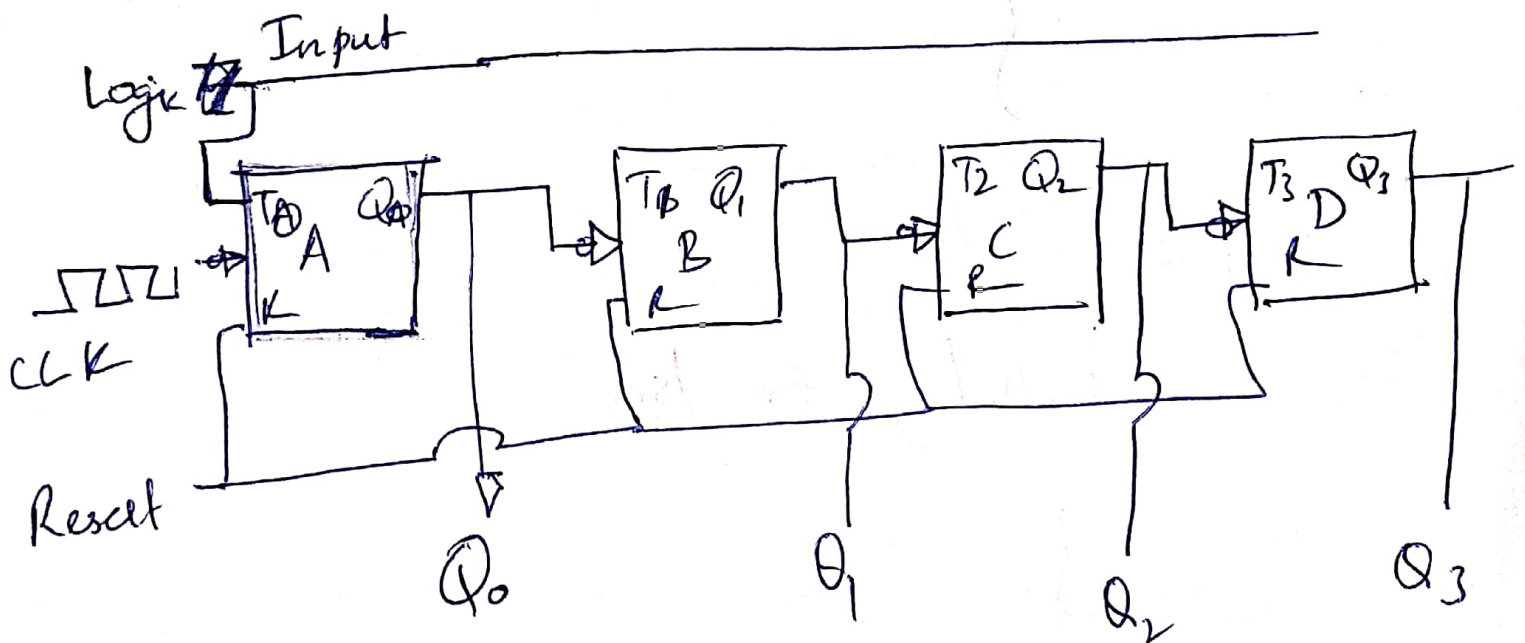


ASTM3



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 300×1000

= 100100100 1111 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 $S_2 \rightarrow$ then delay for 5 mins as mentioned previously

→ $S_2 \rightarrow 10 \rightarrow \cancel{Q_0 \bar{Q}_1} Q_0 \bar{Q}_1$

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

{ Delay for 5 mins

}

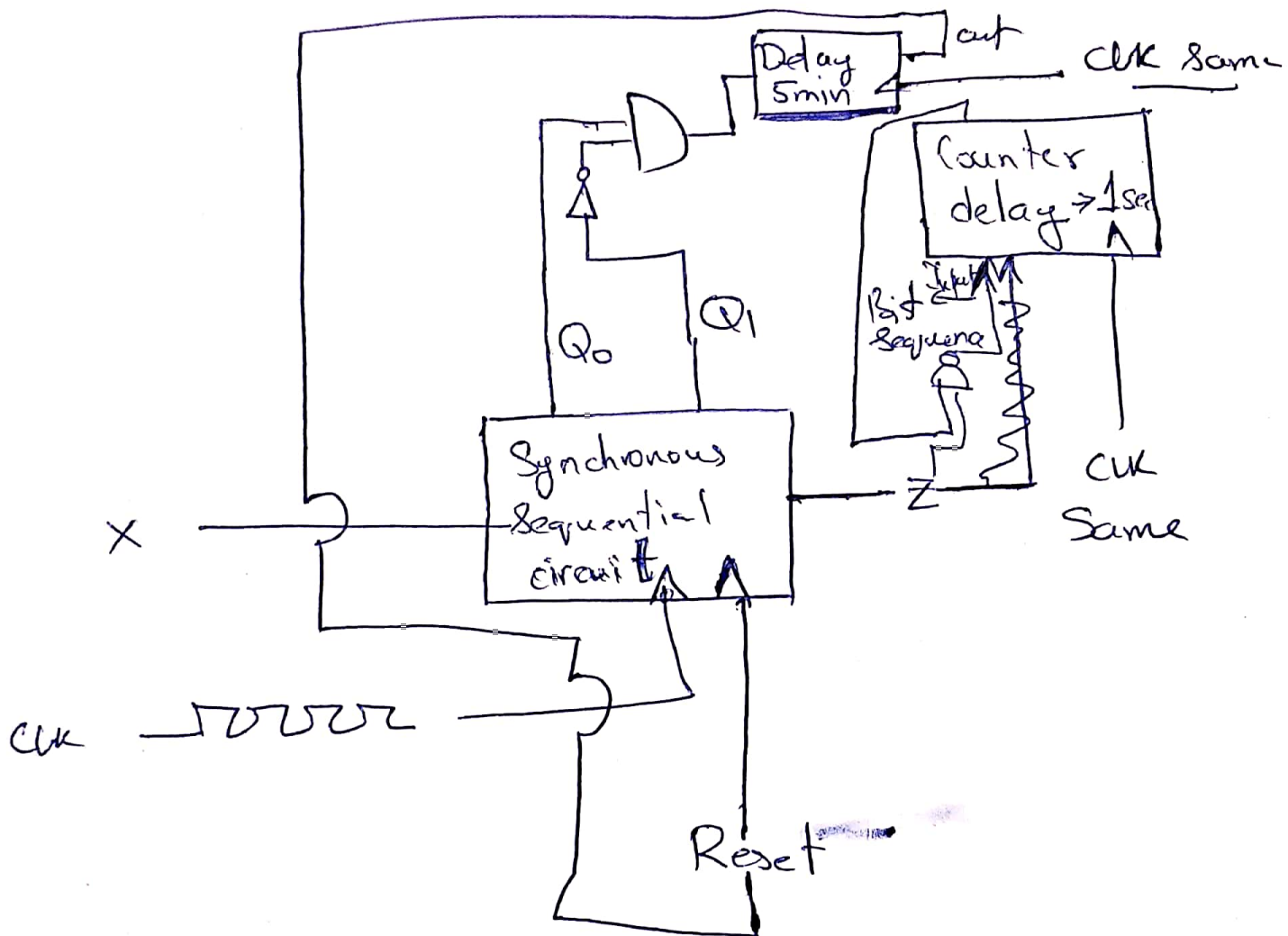
if $(Z == 1)$

{ Delay for 1 sec

}

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, p;
```

```
    reg [1:0] i;
```

```
    reg [2:0] stat, next_stat;
```

```
    reg NV, LK, UNLK;
```

```
//time scale : 1ms/1ns
```

```
initial
```

```
begin
```

```
    p1 = 4'b0010; //password is set to 118
```

```
    p2 = 4'b0101;
```

```
    p3 = 4'b1000;
```

```
    stat = 0; next_stat = 0;
```

```
    UNLK = 0; LK = 0;
```

```
    NV = 0;
```

```
    r = 0; end
```

2) : begin

if ($q_1 == p_3$) begin

UNLK = 1; #60000

UNLK = 0;

next state = 0;

end

else begin

i = i + 1;

if ($i == 3$) begin

LK = 1 ; #300000 = 5 mins

LK = 0 ;

i = 0

end

next state = 0;

end

end

3: begin

next state = 4;

end

4: begin

i = i + 1;

if ($i == 3$) begin

LK = 1; # 3 00,000

LK = 0;

i = 0;

end

next state = 0;

// 60,000ms = 1 second

2, always @(q) // change in q main loop

begin

case (state)

0: begin

if ($q == p_1$) begin

nextstate = 1;

end

else begin

nextstate = 3;

end

end

1: begin

if ($q == p_2$) begin

nextstate = 2;

end

else begin

nextstate = 4;

end

end

```

    end
  endcase
end
end module

```

// change in q is ignored during above delays as delay are declared inside always block.

```

always @ (d) // gen NV signal for 200ms after i/p

```

```

  begin

```

```

    if (d != 4'b0000)

```

```

      begin

```

```

        NV = 1; #200 // timescale is ns/1ms

```

```

        NV = 0;

```

```

      end

```

```

        // #200 implies 100ms

```

```

    end

```

```

always @ (negedge dock)

```

```

  begin

```

```

    state <= nextstate;

```

```

    if (NV != 1 & d != 4'b0000)

```

```

      begin

```

```

        q <= d

```

```

      end

```

```

    end

```