Set-D

20BDS0405.

1) Design a sequential circuit using RS-FF that has 1 data input (w) and 1 data output (z). The output z will become 1 if in the last 3 clock cycles the number of 1s on the input w is greater than 1. Draw the FSM diagram and write/simulate the Verilog code to verify it.

↳ Sol:-

For above condition, we are going to implement a Mealy state machine having four states S_0, S_1, S_2, S_3 and do the state assignment as:- $S_0 = 00$

$S_1 = 01$

$S_2 = 10$

$S_3 = 11$

Following is the state diagram to achieve the required output i.e. '1' if 2 or more preceeding bits are '1' in a sequence.

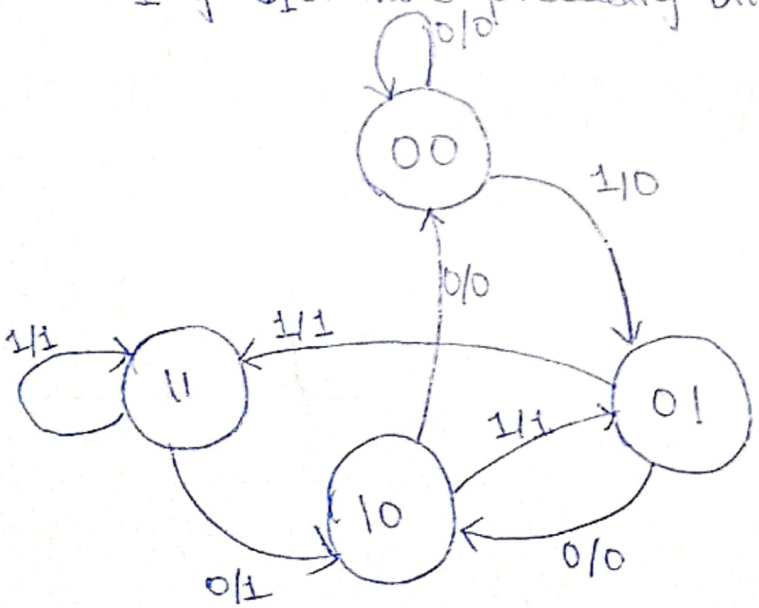


Fig:- State diagram of given problem.

Here, we want to implement it using SR flip flops.
So, the excitation table of SR flip flop is:-

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

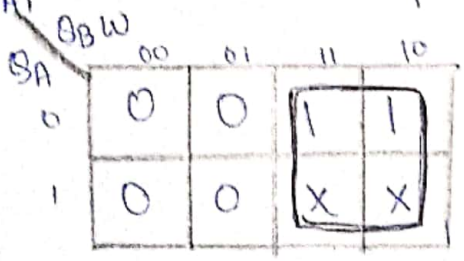
20BDS0405.

State table:-

Present State		Input	Next State		Output	flip flop input - func			
Q_A	Q_B	W	Q_A^+	Q_B^+	Z	S_A	R_A	S_B	R_B
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	0	X	1	0
0	1	0	1	0	0	1	0	0	1
0	1	1	1	1	1	1	0	X	0
1	0	0	0	0	0	0	1	0	X
1	0	1	0	1	1	0	1	1	0
1	1	0	1	0	1	X	0	0	1
1	1	1	1	1	1	X	0	X	0

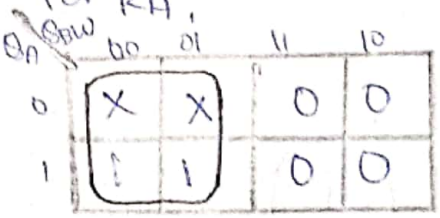
Now, following are the respective KMaps of S_A, R_A, S_B, R_B and Z with respect to Q_A, Q_B and W.

For S_A:



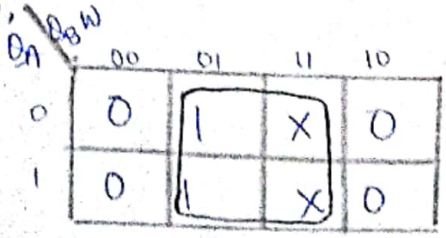
Here, $S_A = Q_B$

For R_A,



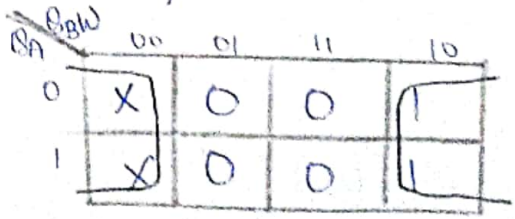
Here, $R_A = \overline{Q_B}$

For S_B,



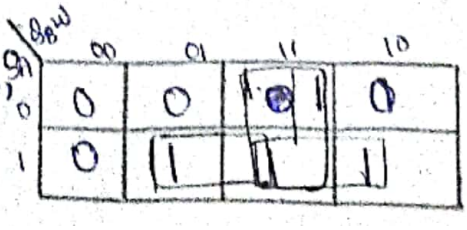
Here, $S_B = W$

For R_B,



$R_B = \overline{W}$

For Z,



$Z = Q_AW + Q_BW + Q_AQ_B$

Following is the circuit for its implementation:-

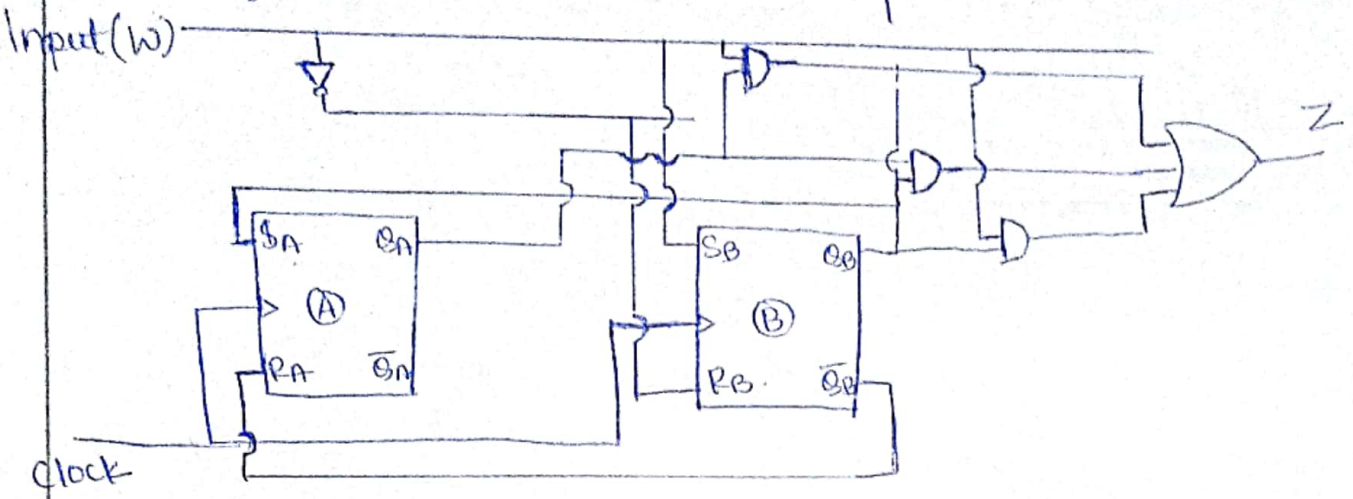


Fig:- ~~Alarm~~ Circuit to detect more than 1 highs in last 3 bits with overlap.

Note:- The verilog code and simulation is attached in the Labpact.
- As Experiment 9 (Digital Simulation Using ModelSim).

2) Design the 4 bit 2421 code to 53-1-1 code and write verilog HDL using case statement.

↳ Sol:

Following table shows the conversion of 2421 code to 53-1-1.

Let the bits representing 2421 are $A_0A_1A_2A_3$ and 53-1-1 are $B_0B_1B_2B_3$

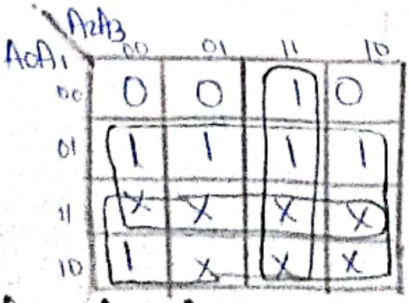
Here, the maximum number represented by the system corresponds to 8 in BCD.

2421 Code				53-1-1 code				
A_0	A_1	A_2	A_3	B_0	B_1	B_2	B_3	BCD
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1	1
0	0	1	0	0	1	1	0	2
0	0	1	1	1	0	1	1	3
0	1	0	0	1	0	1	0	4
1	0	1	1	1	0	0	0	5
1	1	0	0	1	1	1	1	6
1	1	0	1	1	1	1	0	7
1	1	1	0	1	1	0	0	8

All other cases simply don't exist. Hence, can be treated as don't cares.

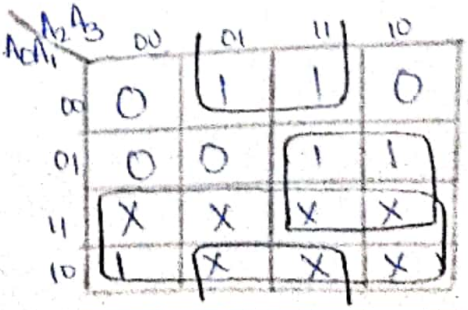
The K-Map for B_0, B_1, B_2, B_3 are :-

For B_0,



$B_0 = A_2A_3 + A_0 + A_1$

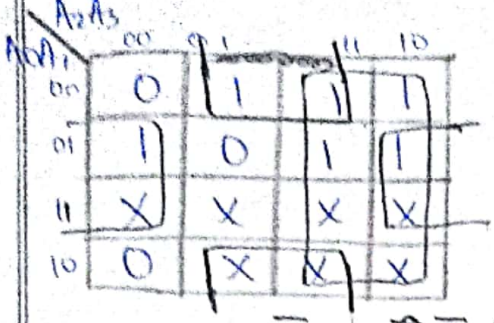
For B_1,



$B_1 = A_0 + \bar{A}_1A_3 + A_1 \cdot A_2$

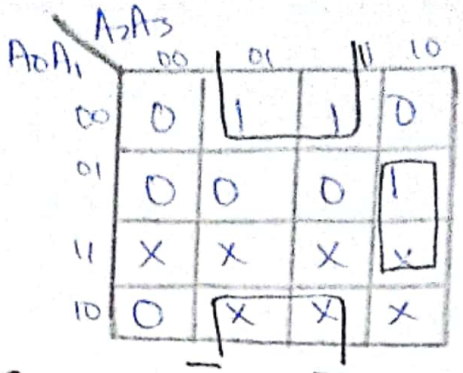
20BDS0405

For B_2,



$B_2 = A_2 + A_1\bar{A_3} + \bar{A_1}A_3$

For B_3,



$B_3 = A_1A_2\bar{A_3} + \bar{A_1}A_3$

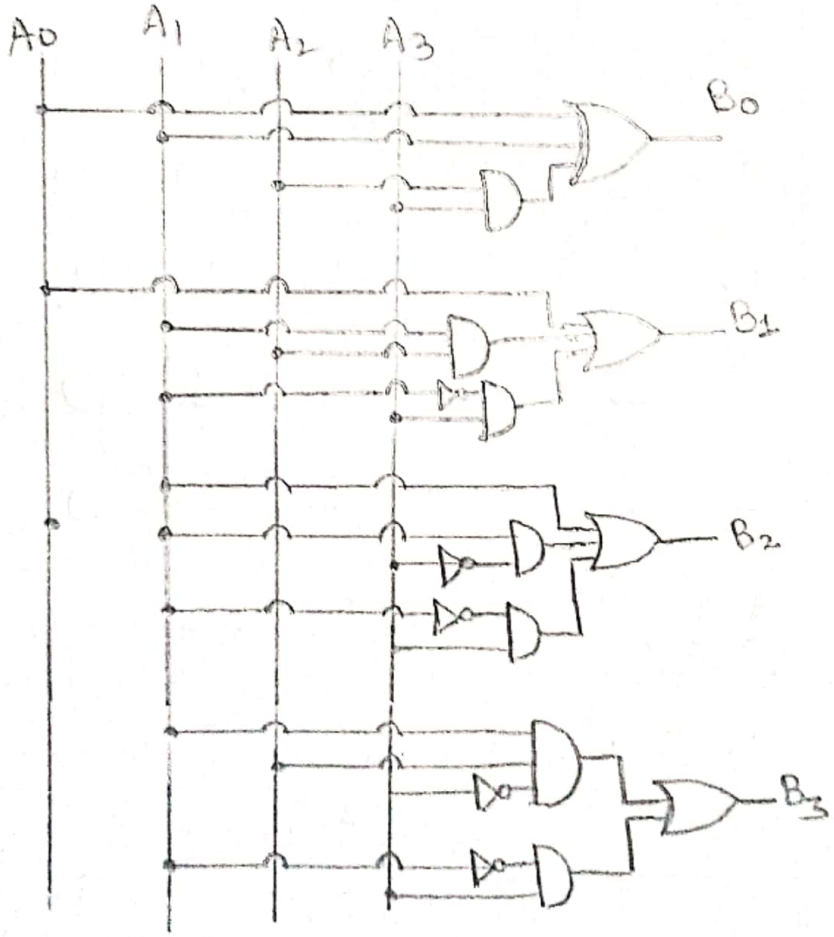
Hence the required functions to be simulated in verilog are:-

$B_0 = A_0 + A_1 + A_2A_3$

$B_1 = A_0 + A_1A_2 + \bar{A_1}A_3$

$B_2 = A_2 + A_1\bar{A_3} + \bar{A_0}\bar{A_1}A_3$

$B_3 = A_1A_2\bar{A_3} + \bar{A_1}A_3$



20BDS0405

Verilog Code:-

```
// To Convert from 2421 code to 53-1-1 Code.
// Consider 2421 as Code A and 53-1-1 as code B for simplicity.


module A2B (input [3:0] A, output [3:0] B);
assign B[0] = A[0] | A[1] | (A[2] & A[3]);
assign B[1] = A[0] | (A[1] & A[2]) | (~A[1] & A[3]);
assign B[2] = A[2] | (A[1] & ~A[3]) | (~A[0] & ~A[1] & A[3]);
assign B[3] = (A[1] & A[2] & ~A[3]) | (~A[1] & A[3]);
endmodule.
```

Note:- The verilog simulation along with above code is attached in lab part as Experiment 7 (Digital simulation with Modelsim).

③ How to Construct an asynchronous MOD-5 counter.
MOD-7 Counter
MOD-12 Counter.

(a) MOD-5 Counter.

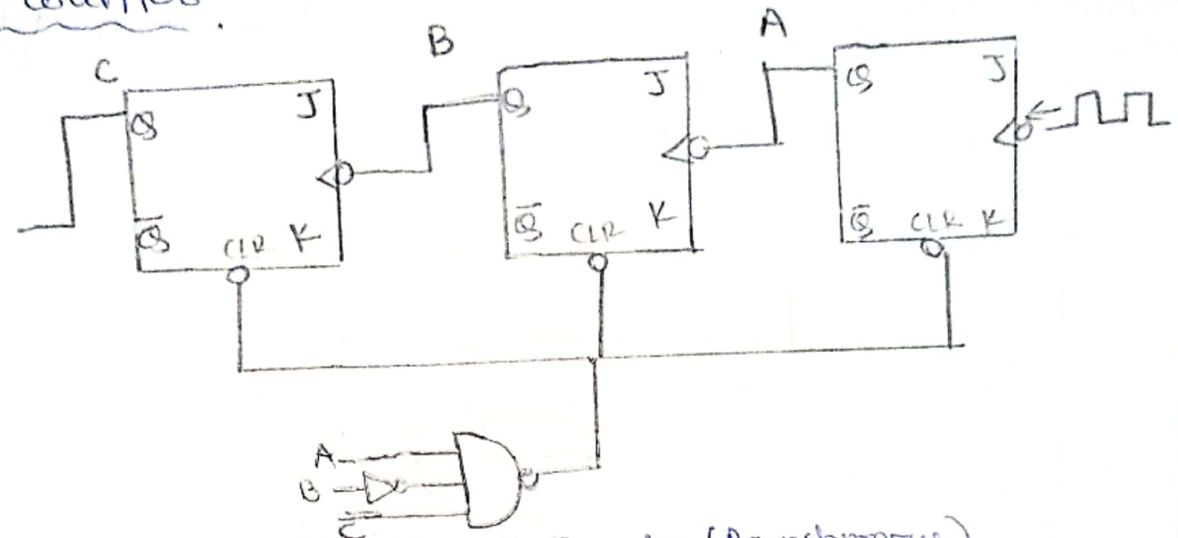


Fig:- MOD-5 Counter (Asynchronous)

The mod-5 counter can be designed like this. Here, we have 3 asynchronous JK flip flops whose outputs is fed to a NAND gates by complementing the middle one. Hence, MOD-5 counter can be realized.

(b) MOD-7 Counter.

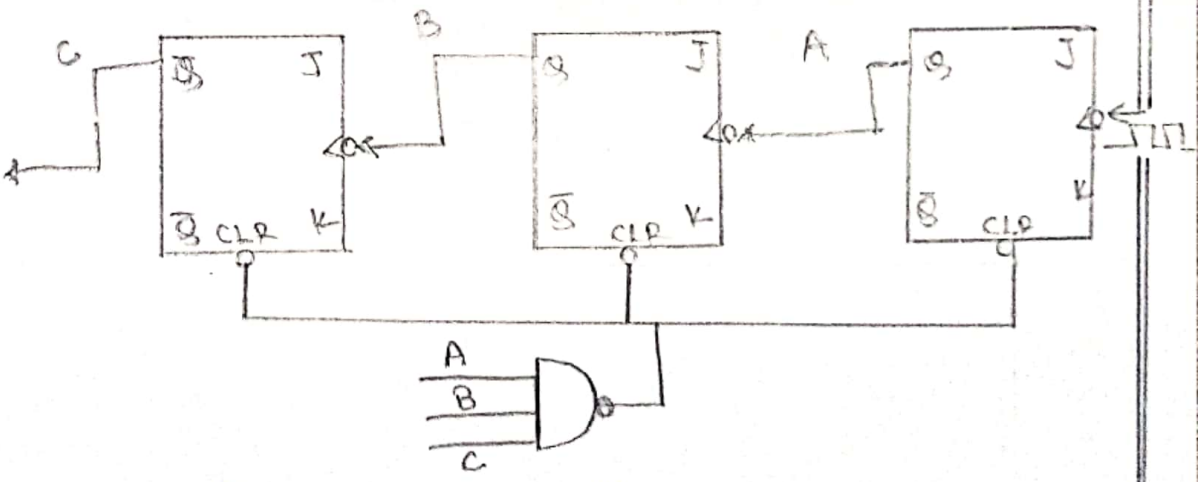


Fig:- MOD-7 Counter (Asynchronous).

In this way, the MOD-7 counter can be designed by feeding the outputs of each flip flops to a 3 input nand gate and then subsequently feeding into the clear inputs of the flip-flops. When A,B,C = 1,1,1, NAND gives low output and the flip-flops are cleared thus resetting count to 0. In this way, it can count upto 6 as a MOD-7 Counter.

(c) MOD-12 Counter:

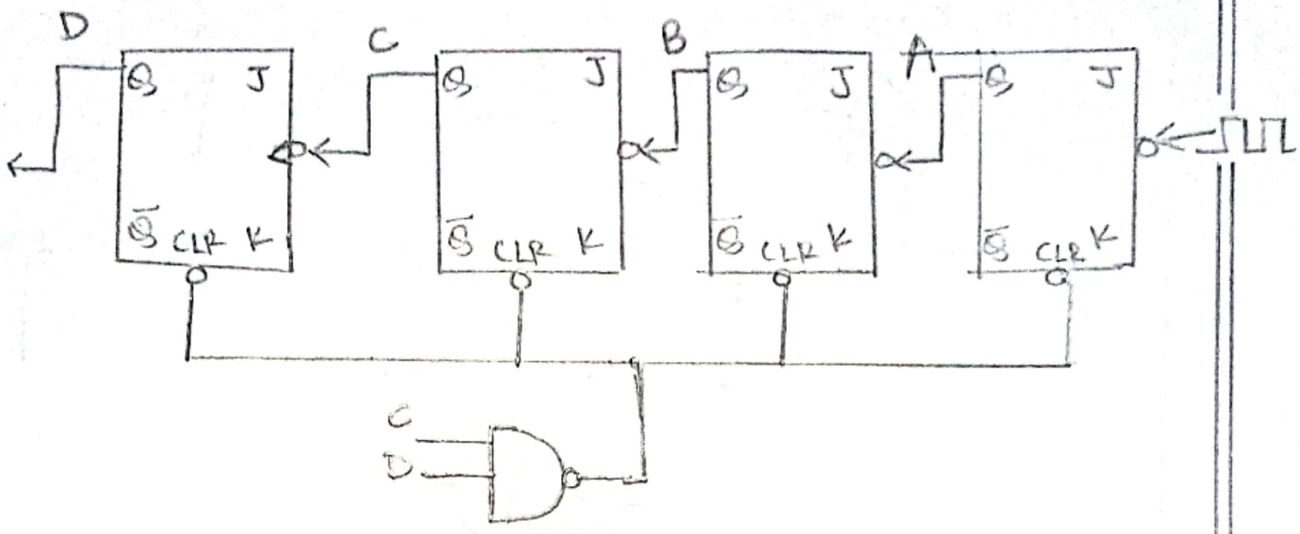


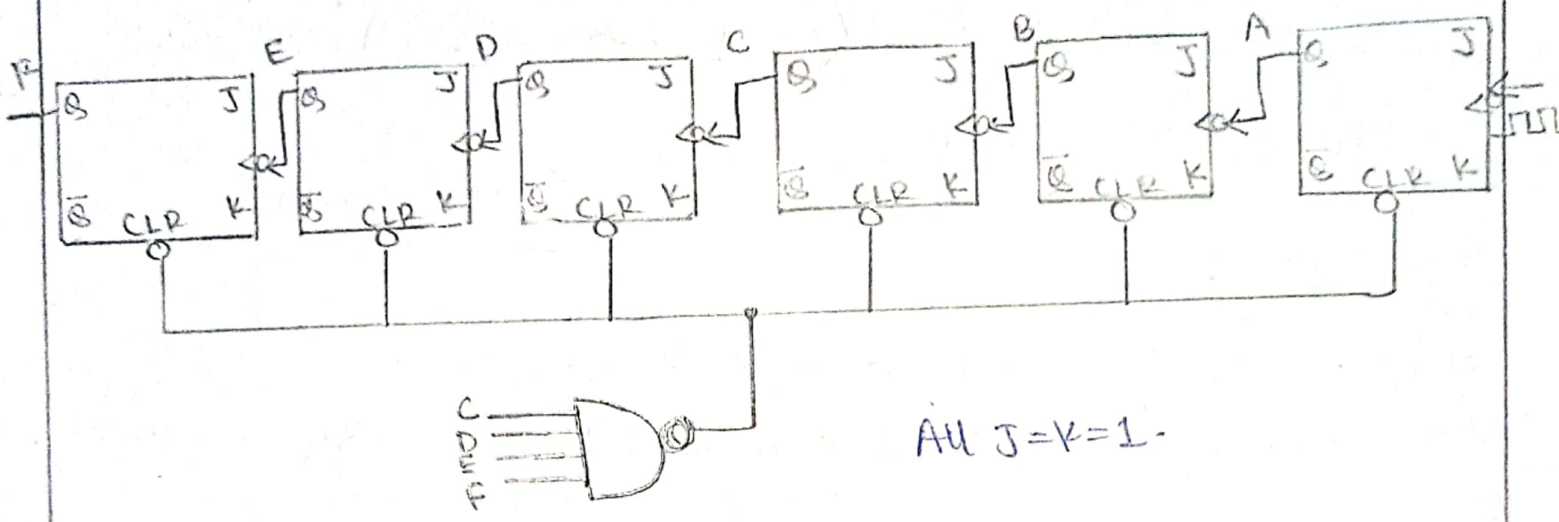
Fig:- MOD-12 Counter (Asynchronous Counter).

Here, we have implemented MOD-12 asynchronous counter by using 4 bits counter (4 flip-flops). The outputs C and D are fed to NAND such that when both C and D are 1,1, the NAND gives low output thus clearing all the flip-flops and the counting process starts over again and reaches upto 11 and again clears at 12 and the cycle continues.

In this way, MOD-12 asynchronous counter can be constructed.

20BDS0405.

4) The following is a MOD-? Counter.



All J=K=1.

Ans:- The above given counter is a MOD-60 counter. Without NAND gate on clear, it could count upto 63 as a MOD-64 counter But, since NAND is applied and input is given from C, D, E, F, the whole circuit will set to clear when C, D, E, f = 1.

i.e. $2^5 + 2^4 + 2^3 + 2^2 = 60$

Hence, the counter resets at 60 by counting upto 59.

So, It is a MOD-60 counter.