Finite State Machine

G V V Sharma*

Contents

***Abstract*—This manual explains state machines by deconstructing a decade counter.**

1 The Decade Counter

The block diagram of a decade counter (repeatedly counts up from 0 to 9) is available in Fig. 0. The *incrementing* decoder and *display* decoder are part of *combinational* logic, while the *delay* is part of *sequential* logic.

2 Finite State Machine

1. Fig. 1 shows a *finite state machine* (FSM) diagram for the decade counter in Fig. 0. s_0 is the state when the input to the incrementing decoder is 0. The *state transition table* for the FSM is Table 0 in [1] where the present state is denoted by the variables W, X, Y, Z and the next state by A, B, C, D.
2. The FSM implementation is available in Fig. 2. The *flip-flops* hold the input for the time that is given by the *clock*. This is nothing but the implementation of the *Delay* block in Fig. 0.

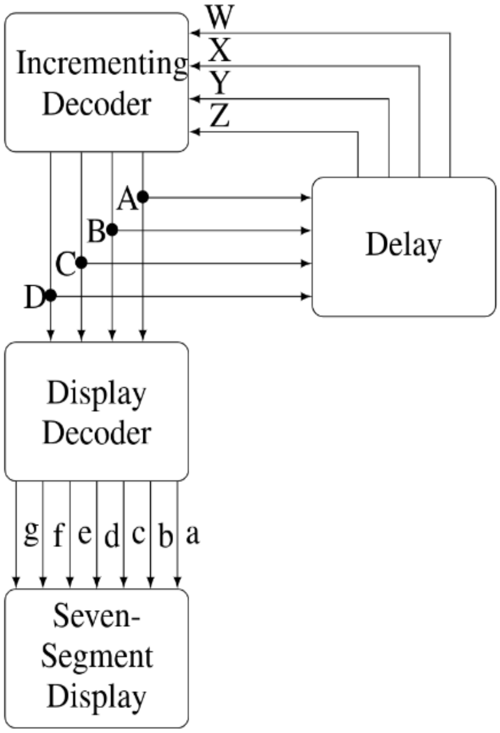


Fig. 0: The decade counter

3. The hardware cost of the system is given by

$$\text{No. of D Flip-Flops} = \lceil \log_2 (\text{No. of States}) \rceil \tag{2.1}$$

For the FSM in Fig. 1, the number of states is 9, hence the number flipflops required = 4.
4. Draw the state transition diagram for a decade down counter (counts from 9 to 0 repeatedly) using an FSM.
Solution: Refer 4

5. Write the state transition table for the down

*The author is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in. All content in this manual is released under GNU GPL. Free and open source.

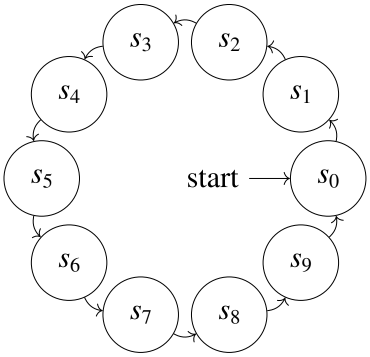


Fig. 1: FSM for the decade counter.

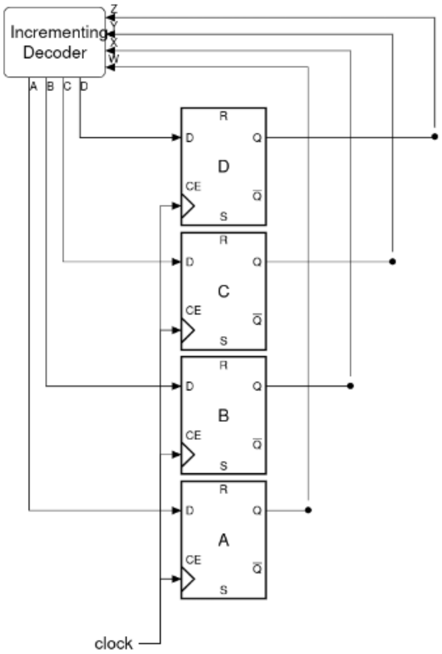


Fig. 2: Decade counter FSM implementation using D-Flip Flops.

counter.
Solution:
Without Don't Care: Refer 5
With Don't Care: Refer 5

6. Obtain the state transition equations with and without don't cares.
Solution:
1. Without DON'T CARE:
from Fig. 6

$$A = Y'X'W' + Z'W' \tag{2.2}$$

from Fig. 6

$$B = Z'XW + Z'YX'W' + ZY'X'W' \tag{2.3}$$

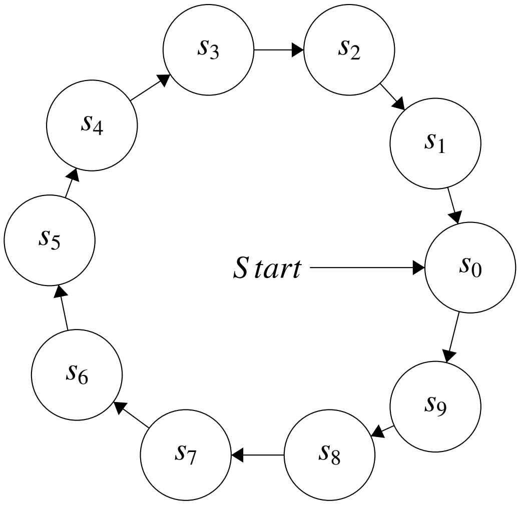


Fig. 4: FSM for Down Counter

Z	Y	X	W	**D**	**C**	**B**	**A**
0	0	0	0	**1**	**0**	**0**	**1**
1	0	0	1	**1**	**0**	**0**	**0**
1	0	0	0	**0**	**1**	**1**	**1**
0	1	1	1	**0**	**1**	**1**	**0**
0	1	1	0	**0**	**1**	**0**	**1**
0	1	0	1	**0**	**1**	**0**	**0**
0	1	0	0	**0**	**0**	**1**	**1**
0	0	1	1	**0**	**0**	**1**	**0**
0	0	1	0	**0**	**0**	**0**	**1**
0	0	0	1	**0**	**0**	**0**	**0**

TABLE 5: Down Counter State Transition Table Without Don't Care

from Fig. 6

$$C = ZY'X'W' + Z'YW + Z'YX \tag{2.4}$$

from Fig. 6

$$D = Z'Y'X'W' + ZY'X'W \tag{2.5}$$

2. With DON'T CARE:
from Fig. 6

$$A = W' \tag{2.6}$$

from Fig. 6

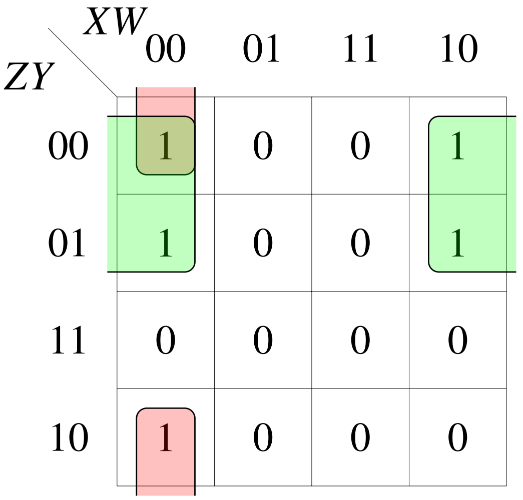
$$B = XW + YX'W' + ZX'W' \tag{2.7}$$

from Fig. 6

$$C = ZW' + YW + YX \tag{2.8}$$

Z	Y	X	W	**D**	**C**	**B**	**A**
0	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	0	0	0	1	1	1
0	1	1	1	0	1	1	0
0	1	1	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	0	0	0	0	1	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0
1	0	1	0	-	-	-	-
1	0	1	1	-	-	-	-
1	1	0	0	-	-	-	-
1	1	0	1	-	-	-	-
1	1	1	0	-	-	-	-
1	1	1	1	-	-	-	-

TABLE 5: Down Counter State Transition Table With Don't Care



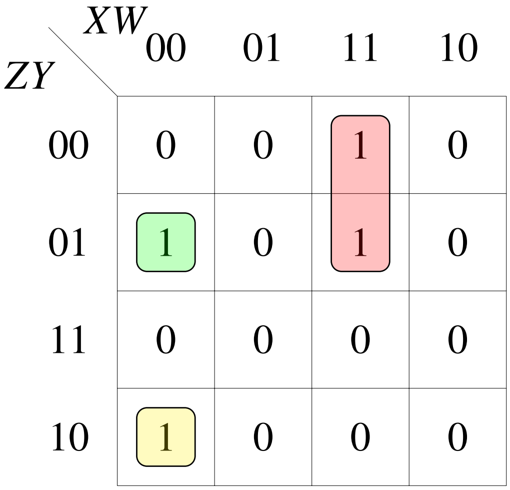
ZY \ XW	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	0	0
10	1	0	0	0

Fig. 6: K-map for *a*.

from Fig. 6

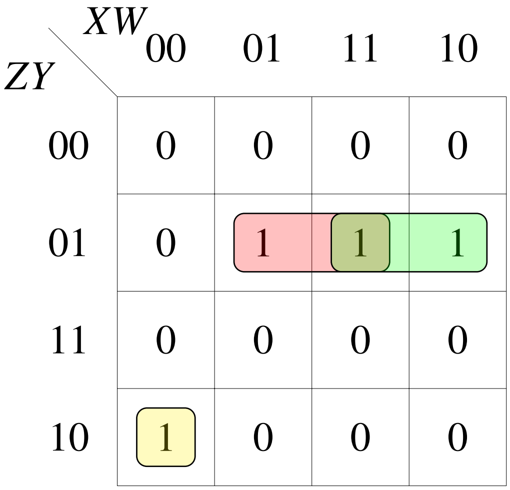
$$D = Z'Y'X'W' + ZW \tag{2.9}$$

7. Verify your design using an arduino.
8. Repeat the above exercises by designing a circuit that can detect 3 consecutive 1s in a bitstream.
   Solution:
   Finite State Machine Diagram can seen in Fig.8



ZY \ XW	00	01	11	10
00	0	0	1	0
01	1	0	1	0
11	0	0	0	0
10	1	0	0	0

Fig. 6: K-map for *b*.



ZY \ XW	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	0	0	0	0
10	1	0	0	0

Fig. 6: K-map for *c*.

Transition Table can be seen in Table. 8
from Fig. 8

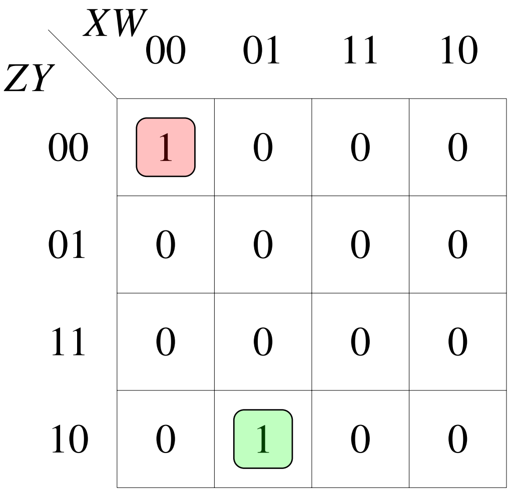
$$D_a = Ax + Bx \tag{2.10}$$

from Fig. 8

$$D_b = Ax + B'x \tag{2.11}$$

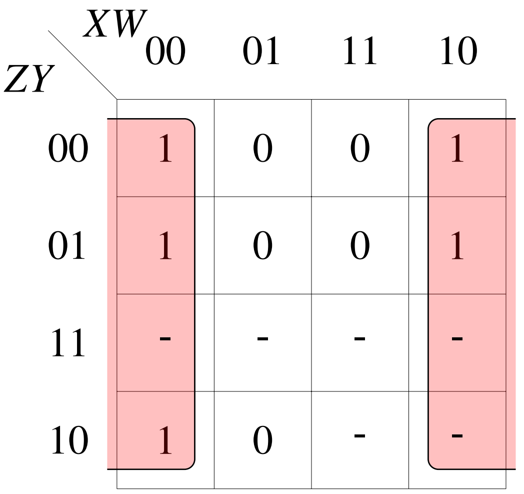
from Fig. 8

$$y = AB \tag{2.12}$$



$ZY \backslash XW$	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	1	0	0

Fig. 6: K-map for *d*.

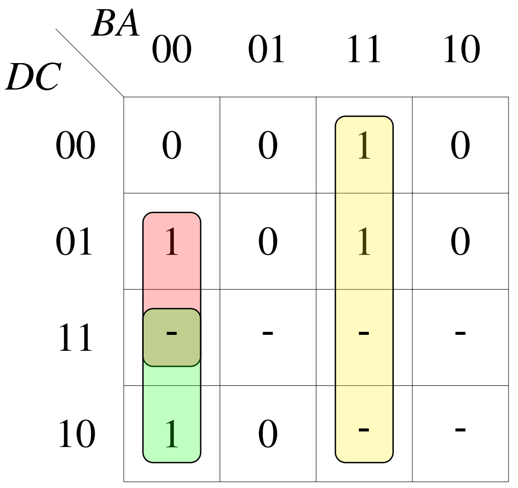


$ZY \backslash XW$	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	-	-	-	-
10	1	0	-	-

Fig. 6: K-map for *a* with don't care.

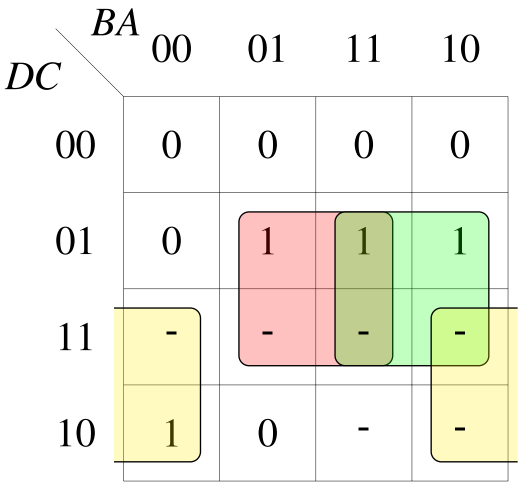
References

[1] G. V. V. Sharma. Karnaugh Map. [Online]. Available: https://github.com/gadepall/arduino/raw/master/ide/kmap/gvv_kmap.pdf



$DC \backslash BA$	00	01	11	10
00	0	0	1	0
01	1	0	1	0
11	-	-	-	-
10	1	0	-	-

Fig. 6: K-map for *b* with don't care.

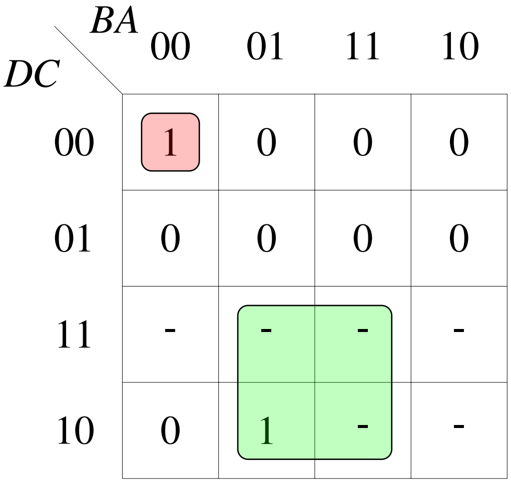


$DC \backslash BA$	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	-	-	-	-
10	1	0	-	-

Fig. 6: K-map for *c* with don't care.

Present State		Input	Next State		Output
A	B	x	A'	B'	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

TABLE 8: Transition Table for Bitstream



DC \ BA	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	-	-	-	-
10	0	1	-	-

Fig. 6: K-map for d with don't care.

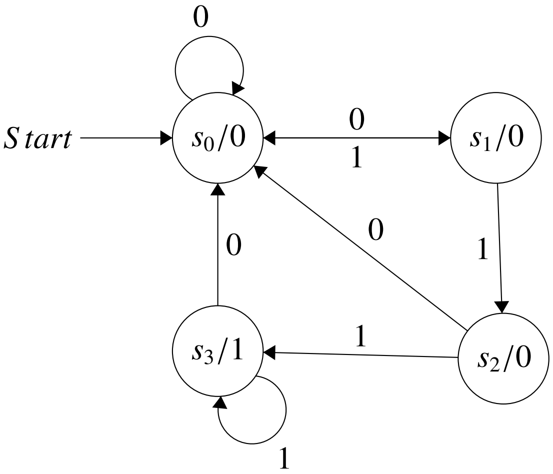
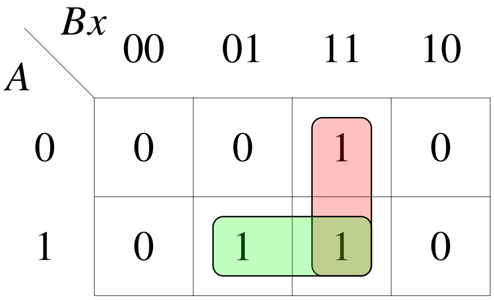
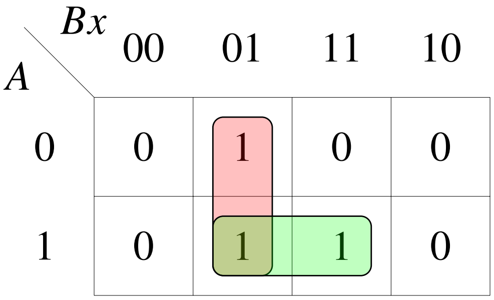


Fig. 8: FSM for Bitstream



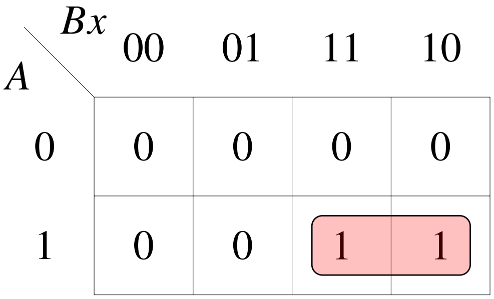
A \ Bx	00	01	11	10
0	0	0	1	0
1	0	1	1	0

Fig. 8: K-map for D_a.



A \ Bx	00	01	11	10
0	0	1	0	0
1	0	1	1	0

Fig. 8: K-map for D_b.



A \ Bx	00	01	11	10
0	0	0	0	0
1	0	0	1	1

Fig. 8: K-map for y.