Karnaugh Map and Finite State Machine

G V V Sharma*

Contents

***Abstract*—This manual explains Karnaugh maps (K-map) and 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. 1. The *incrementing* decoder and *display* decoder are part of *combinational* logic, while the *delay* is part of *sequential* logic.

2 Incrementing Decoder

The incrementing decoder in Fig. 1 takes the numbers $0, 1, \ldots, 9$ in binary as inputs and generates the consecutive number as output. The corresponding truth table is available in Fig. I.

*The author is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in.

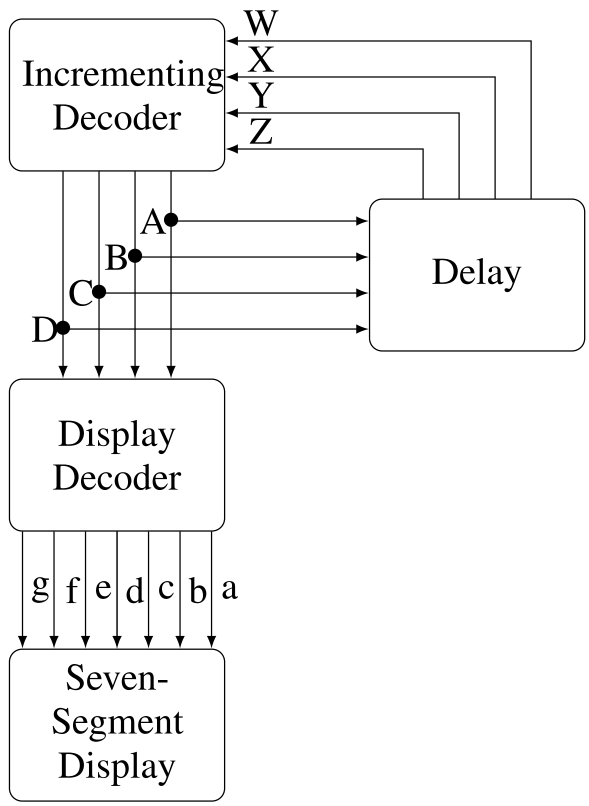

Fig. 1: The decade counter

3 Karnaugh Map

Using Boolean logic, output A in Table I can be expressed in terms of the inputs W, X, Y, Z as

$$A = W'X'Y'Z' + W'XY'Z' + W'X'YZ' + W'XYZ' + W'X'Y'Z \quad (1)$$

3.1 K-Map for A

The expression in (1) can be minimized using the K-map in Fig. 2. In Fig. 2, the *implicants* in boxes 0,2,4,6 result in $W'Z'$. The implicants in boxes 0,8

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

TABLE I

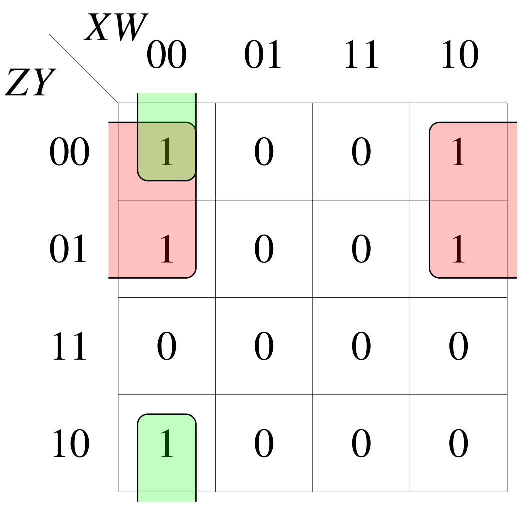

Fig. 2: K-map for A.

result in $W'X'Y'$. Thus, after minimization using Fig. 2, (1) can be expressed as

$$A = W'Z' + W'X'Y' \tag{2}$$

Problem 1. Using the fact that

$$\begin{aligned} X + X' &= 1 \\ XX' &= 0, \end{aligned} \tag{3}$$

derive (2) from (1) algebraically.

3.2 K-Map for B

From Table I, using boolean logic,

$$B = WX'Y'Z' + W'XY'Z' + WX'YZ' + W'XYZ' \tag{4}$$

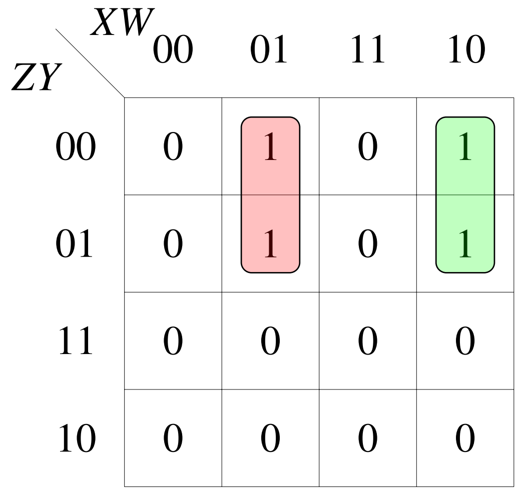

Fig. 3: K-map for B.

Problem 2. Show that (4) can be reduced to

$$B = WX'Z' + W'XZ' \tag{5}$$

using Fig. 3.

Problem 3. Derive (5) from (4) algebraically using (3).

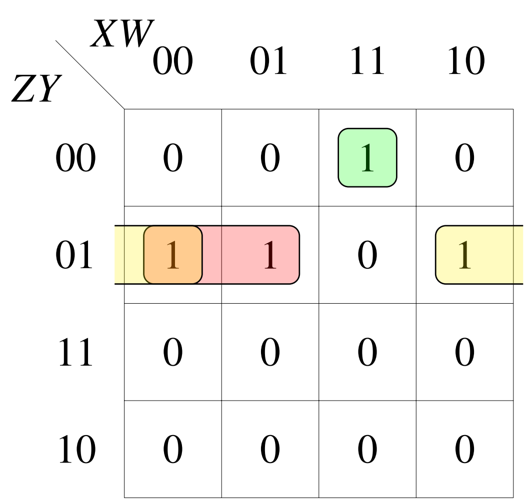

Fig. 4: K-map for C.

3.3 K-Map for C

From Table I, using boolean logic,

$$C = WXY'Z' + W'X'YZ' + WX'YZ' + W'XYZ' \quad (6)$$

Problem 4. Show that (6) can be reduced to

$$C = WXY'Z' + X'YZ' + W'YZ' \quad (7)$$

using Fig. 4.

Problem 5. Derive (7) from (6) algebraically using (3).

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

Fig. 5: K-map for D.

3.4 K-Map for D

From Table I, using boolean logic,

$$D = WXYZ' + W'X'Y'Z \quad (8)$$

Problem 6. Minimize (8) using Fig. 5.

4 Don't Care Conditions

4 binary digits are used in the incrementing decoder in Fig. 1. However, only the numbers from 0-9 are used as input/output in the decoder and we *don't care* about the numbers from 10-15. This phenomenon can be addressed by revising Table I to obtain Table II.

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

ZY \ 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 cares.

The revised K-maps for A,B,C,D are now available in Figs. 6-9 resulting in

$$A = W' \quad (9)$$
$$B = WX'Z' + W'X \quad (10)$$
$$C = X'Y + W'Y + WXY' \quad (11)$$
$$D = W'Z + WXY \quad (12)$$

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

Fig. 7: K-map for B with don't cares.

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

Fig. 8: K-map for C with don't cares.

which are simpler than (2), (5), (7) and (8).

5 Display Decoder

Table III is the truth table for the display decoder in Fig. 1.

Problem 7. Use K-maps to obtain the minimized expressions for a, b, c, d, e, f, g in terms of A, B, C, D with and without don't care conditions.

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

Fig. 9: K-map for D with don't cares.

D	C	B	A	**a**	**b**	**c**	**d**	**e**	**f**	**g**	Decimal
0	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	0
0	0	0	1	**1**	**0**	**0**	**1**	**1**	**1**	**1**	1
0	0	1	0	**0**	**0**	**1**	**0**	**0**	**1**	**0**	2
0	0	1	1	**0**	**0**	**0**	**0**	**1**	**1**	**0**	3
0	1	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**0**	4
0	1	0	1	**0**	**1**	**0**	**0**	**1**	**0**	**0**	5
0	1	1	0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	6
0	1	1	1	**0**	**0**	**0**	**1**	**1**	**1**	**1**	7
1	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	8
1	0	0	1	**0**	**0**	**0**	**1**	**1**	**0**	**0**	9

TABLE III: Truth table for display decoder.

6 Finite State Machine

Fig. 10 shows a *finite state machine* (FSM) diagram for the decade counter in Fig. 1. s_0 is the state when the input to the incrementing decoder is 0. The *state transition table* for the FSM is Table I where the present state is denoted by the variables W, X, Y, Z and the next state by A, B, C, D. The FSM implementation is available in Fig. 11. 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. 1. The hardware cost of the system is given by

$$\text{No. of D Flip-Flops} = \lceil \log_2 (\text{No. of States}) \rceil \quad (13)$$

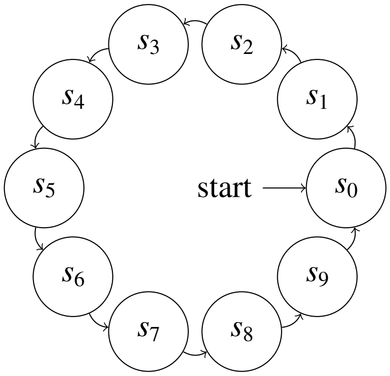

Fig. 10: FSM for the decade counter.

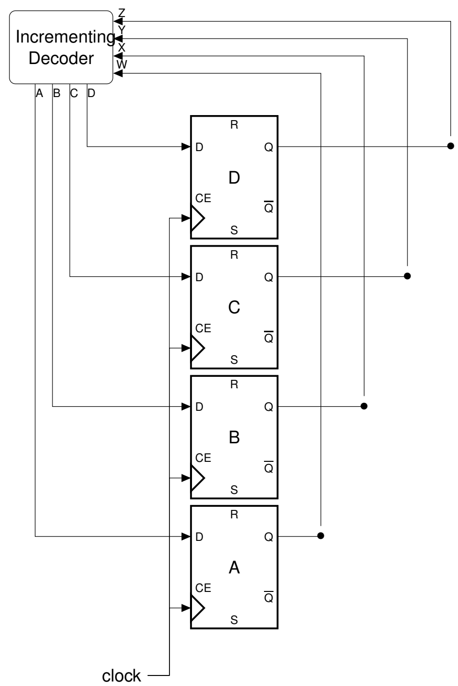

Fig. 11: Decade counter FSM implementation using D-Flip Flops.

For the FSM in Fig. 10, the number of states is 9, hence the number flipflops required = 4.

Problem 8. Design a decade down counter (counts from 9 to 0 repeatedly) using an FSM.