CHAPTER 7

7.1 **(a)** 16 K × 8 ⇒ Address lines = 14, Data lines = 8

(b) 512 K × 16 ⇒ Address lines = 19, Data lines = 16

(c) 32 M × 32 ⇒ Address lines = 25, Data lines = 32

(d) 8 G × 16 ⇒ Address lines = 33, Data lines = 16

7.2 **(a)** 16 K Bytes

(b) 1024 K Bytes

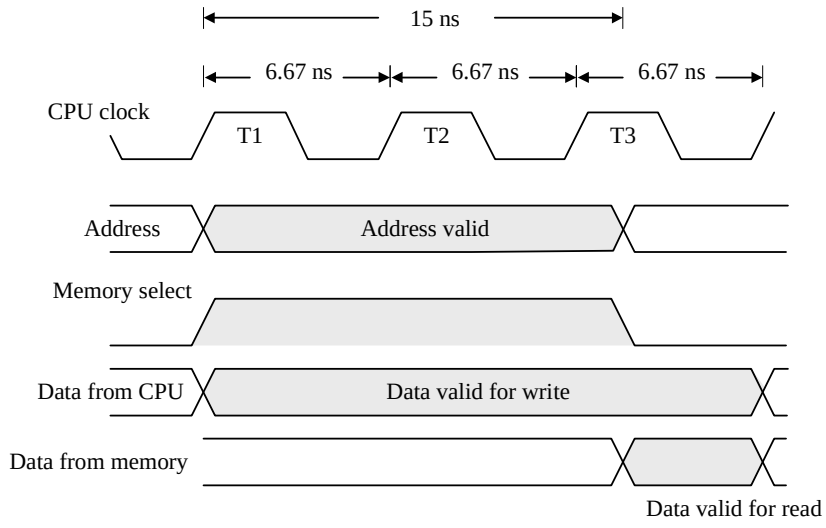
(c) 128 M Bytes

(d) 16 G Bytes

7.3

1024 × 16 memory	⇒	Address line	= 10, Data line = 16
word: no. 875	⇒	Address bits	= 11011011011
data 46654	⇒	16-bit data	= 1011 0110 0011 1110
			= $B63E_{16}$

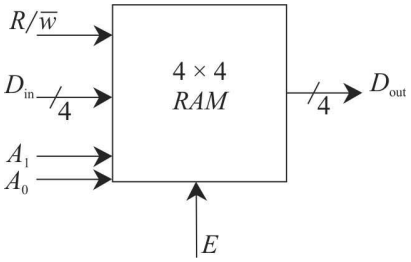
7.4 f_{CPU} = 150 MHz, $T_{CPU} = 1/f_{CPU} = 6.67^{-9}$ Hz^{-1}

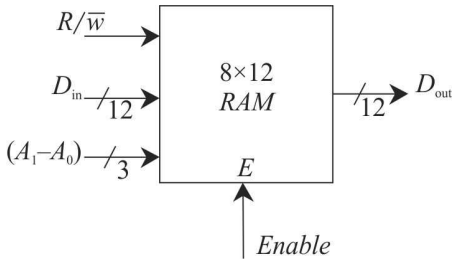


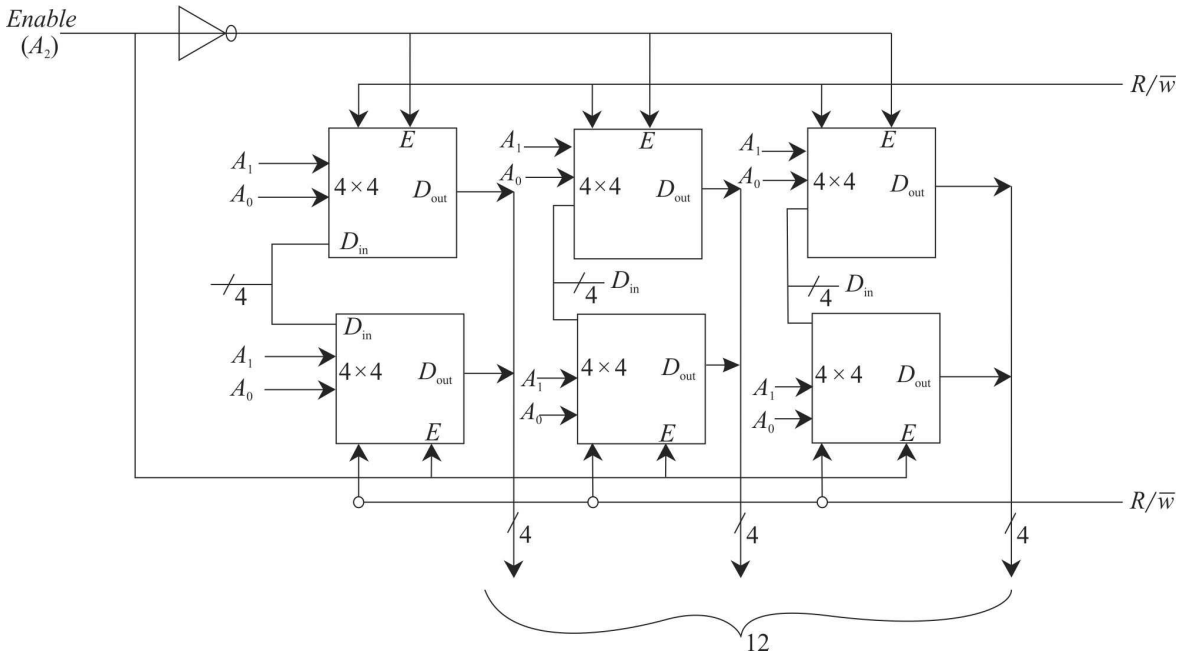
7.5

Pending

7.6







7.7 **(a)** 64 K = $2^{16} = 2^8 \times 2^8 = 256 \times 256$

Each decoder is 8 : 256

Decoder requires 512 AND gates, each with 8 inputs.

(b) 36,952 = 1001 0000 0101 1000

X = 1001 0000 = 144

Y = 0101 1000 = 88

7.8 **(a)** $\dfrac{512\text{ K}}{64\text{ K}} = 8$ chips

(b) 512 K = 2^{19} $\Rightarrow$ 19-bit address lines.

64 K = 2^{16} $\Rightarrow$ 16-lines are connected to each chip

and remaining (19 − 16) = 3 are for

selecting the chips.

(c) 19 − 16 = 3 lines with 3 × 8 decoder.

7.9 Capacity of memory = $2^{15+16} = 2^{31}$ words = 2G.

7.10 Bit position = 1 2 3 4 5 6 7 8 9 10 11 12 13 14

P_1 P_2 0 P_4 1 1 0 P_8 1 0 1 0 1 1

P_1 = XOR of (3, 5, 7, 9, 11, 13) = XOR of (0, 1, 0, 1, 1, 1) = 0

P_2 = XOR of (3, 6, 7, 10, 11, 14) = XOR of (0, 1, 0, 0, 1, 1) = 0

P_4 = XOR of (5, 6, 7, 12, 13, 14) = XOR of (1, 1, 0, 0, 1, 1) = 0

P_8= XOR of (9, 10, 11, 12, 13, 14) = XOR of (1, 0, 1, 0, 1, 1) = 0

Composite 14-bit code word = 00 0011 0010 1011

7.11 Bit position 1 2 3 4 5 6 7 8 9 10 11 12 13

P_1 P_2 1 P_4 1 0 1 P_8 1 0 1 1 0

P_1 = XOR of (3, 5, 7, 9, 11, 13) = XOR of (1, 1, 1, 1, 1, 0) = 1

P_2 = XOR of (3, 6, 7, 10, 11) = XOR of (1, 0, 1, 0, 1) = 1

P_4 = XOR of (5, 6, 7, 12, 13) = XOR of (1, 0, 1, 1, 0) = 1

P_8 = XOR of (9, 10, 11, 12, 13) = XOR of (1, 0, 1, 1, 0) = 1

13-bit code word = 1111101110110

7.12 Bit position 1 2 3 4 5 6 7 8 9 10 11 12 13

P_1 P_2 P_4 P_8

(a) 0 1 1 1 0 0 1 0 1 0 1 0 0

9-bit data word = 100110100

(b) 1 1 1 1 0 1 0 1 0 0 1 1 1

9-bit data word = 101000111

(c) 1 0 1 0 1 0 0 1 1 1 0 1 0

9-bit data word = 110011010

(d) Bit position 1 2 3 4 5 6 7 8 9 10 11 12 13

1 0 1 1 1 0 1 1 1 0 1 1 0

C_1 = XOR (1, 3, 5, 7, 9, 11, 13) = XOR (1, 1, 1, 1, 1, 1, 0) = 0

C_2 = XOR (2, 3, 6, 7, 10, 11) = XOR (0, 1, 0, 1, 0, 1) = 1

C_4 = XOR (4, 5, 6, 7, 12, 13) = XOR (1, 1, 0, 1, 1, 0) = 0

C_8 = XOR (8, 9, 10, 11, 12, 13) = XOR (1, 1, 0, 1, 1, 0) = 0

$C_8C_4C_2C_1 = 0010 \Rightarrow$ bit 2 with error.

So, 9-bit data word 110110110

(e)

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	13
	0	0	0	0	1	0	0	0	1	0	1	0	1

C_1 = XOR (1, 3, 5, 7, 9, 11, 13) = XOR (0, 0, 1, 0, 1, 1, 1) = 0

C_2 = XOR (2, 3, 6, 7, 10, 11) = XOR (0, 0, 0, 0, 0, 1) = 1

C_4 = XOR (4, 5, 6, 7, 12, 13) = XOR (0, 1, 0, 0, 0, 1) = 0

C_8 = XOR (8, 9, 10, 11, 12, 13) = XOR (0, 1, 1, 0, 0, 1) = 0

$C_8C_4C_2C_1 = 1010 \rightarrow$ bit 10 with error.

So, 9-bit data word → 010011101

7.13 (**a**) 25 bits ⇒ Check bits K = 5 ⇒ 6 parity bits

+1 bit

(**b**) 55 bits ⇒ Check bits K = 6 ⇒ 7 parity bits

+1 bit

(c) 100 bits ⇒ Check bits K = 7 ⇒ 8 parity bits

+1 bit

7.14

Bit position	1	2	3	4	5	6	7	8	9
	P_1	P_2	P_3	P_4	D_5	D_6	D_7	P_8	P_9
	0	1	0	0	1	1	0	1	1

(a) P_1 = XOR of (3, 5, 7, 9) = XOR of (0, 1, 0, 1) = 0

P_2 = XOR of (3, 6, 7) = XOR of (0, 1, 0) = 1

P_3 = XOR of (5, 6, 7) = XOR of (1, 1, 0) = 0

P_4 = XOR of (9) = XOR of (1) = 1

Ans: 010011011

(b) C_1 = XOR of (1, 3, 5, 7, 9) = XOR of (0, 0, 1, 0, 1) = 0

C_2 = XOR of (2, 3, 6, 7) = XOR of (1, 0, 1, 0) = 0

C_4 = XOR of (4, 5, 6, 7) = XOR of (0, 1, 1, 0) = 0

C_8 = XOR of (8, 9) = XOR of (1, 1) = 0

$C = C_8C_4C_2C_1 = 0000$

(c) 9-bit composite word = 010010011

C_1 = XOR of (1, 3, 5, 7, 9) = XOR of (0, 0, 1, 0, 1) = 0

C_2 = XOR of (2, 3, 6, 7) = XOR of (1, 0, 0, 0) = 1

C_4 = XOR of (4, 5, 6, 7) = XOR of (0, 1, 0, 0) = 1

C_8 = XOR of (8, 9) = XOR of (1, 1) = 0

$C_8C_4C_2C_1 = 0110 \Rightarrow$ Error in bit 6 i.e., $D_6 = 1$

(d) Composite word : 0 1 0 0 1 1 0 1 1 P_{10}

$P_{10} = 1$.

C_1 = XOR of (1, 3, 5, 7, 9) = XOR of (1, 1, 1, 0, 1) = 0

C_2 = XOR of (2, 3, 6, 7, 10) = XOR of (1, 1, 1, 0, 1) = 0

C_4 = XOR of (4, 5, 6, 7) = XOR of (0, 1, 1, 0) = 0

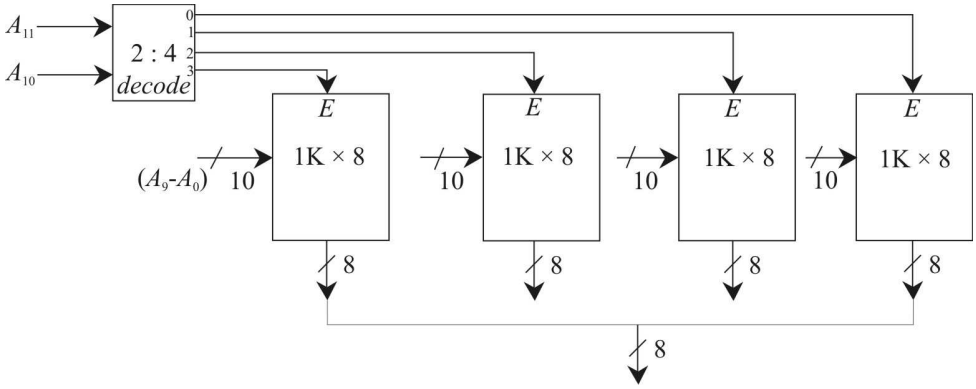
C_8 = XOR of (8, 9, 10) = XOR of (1, 1, 1) = 1

$P = 0$

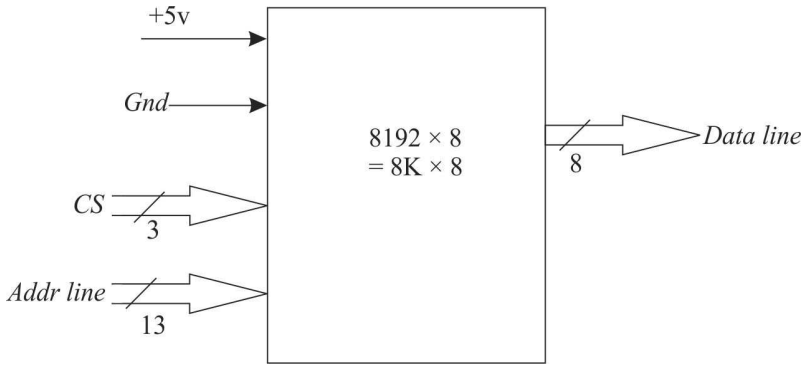
Error in P_1 and D_3, so the composite word
1110110111

$C_8C_4C_2C_1 = 1000$ and $P = 0 \Rightarrow$ indicates double error.

7.15 To construct 4K × 8 ROM from 1K × 8 we need 4 chips.



7.16



Total number of pins = Addr line + Data line + CS + 2
= 13 + 8 + 3 + 2
= 26 pins

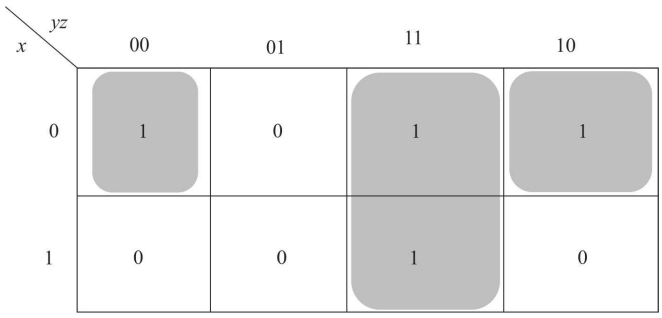
7.17

Input Address	Output of ROM			
$I_5 I_4 I_3 I_2 I_1$	$D_6 D_5 D_4$	$D_3 D_2 D_1$	D_0 (2^0)	Decimal
0 0 0 0 0	0 0 0	0 0 0	0, 1	0, 1
0 0 0 0 1	0 0 0	0 0 1	0, 1	2, 3
…	…	…	…	…
…	…	…	…	…
0 1 0 0 0	0 0 1	0 1 1	0, 1	16, 17
0 1 0 0 1	0 0 1	1 0 0	0, 1	18, 19
…	…	…	…	…
…	…	…	…	…
1 1 1 1 0	1 1 0	0 0 0	0, 1	60, 61
1 1 1 1 1	1 1 0	0 0 1	0, 1	62, 63

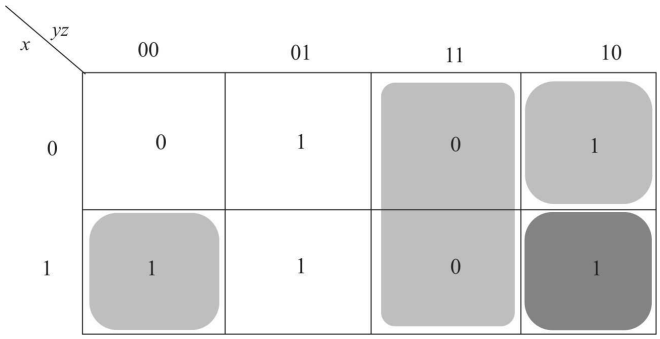
7.18 **(a)** 5-bit binary multiplier:
Size of ROM = 2^{10} × 10 = 1 K × 10 ROM
(b) 5-bit adder-subtractor:
Size of ROM = 2^{11} × 6 = 2 K × 6 ROM

(c) Quadruple 4 × 1 mux:
Size of ROM = $2^{19} \times 4$ = 512 K × 4 ROM
(d) 5 inputs 7 outputs $2^5 \times 7$ 32 × 7 ROM

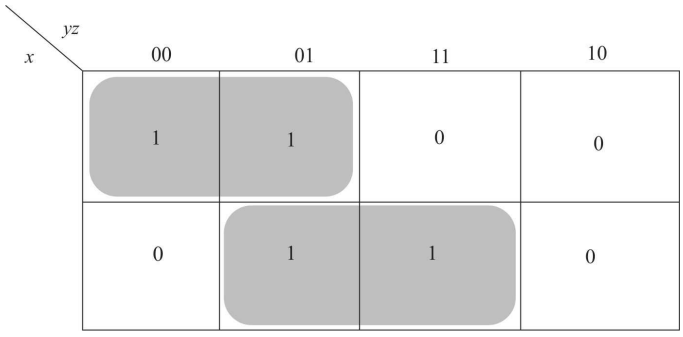
7.19



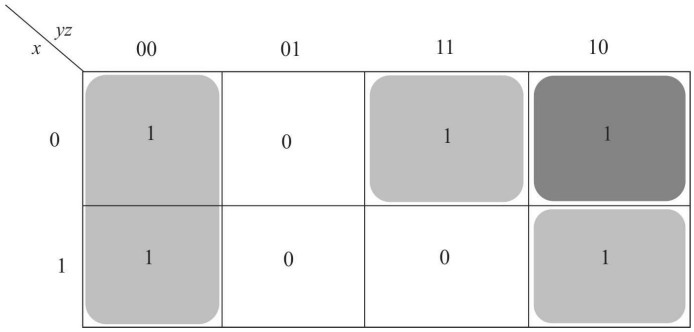
$A(x, y, z) = \Sigma(0, 2, 3, 7)$
$= x'z' + yz$
A' $= xy' + xz' + y'z$



$B(x, y, z) = \Sigma(1, 2, 4, 5, 6)$
$A(x, y, z) = y'z + yz' + xz'$



$C(x, y, z) = \Sigma(0, 1, 5, 7)$
$C(x, y, z) = x'y' + xz$

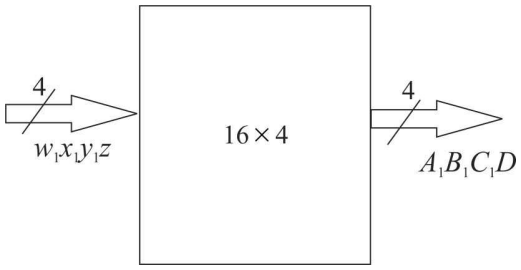


$D(x, y, z) = \Sigma(0, 2, 3, 4, 6)$
$D(x, y, z) = z' + x'y$
$D' = Z(x + y') = xz + y'z$

PLA programming table:

		Inputs			Outputs			
	Product term	*x*	*y*	*z*	*A*	*B*	*C*	*D*
xy'	1	1	0	-	1	-	-	-
xz'	2	1	-	0	1	1	-	-
$y'z$	3	-	0	1	1	1	-	1
yz'	4	-	1	0	-	1	-	-
$x'y'$	5	0	0	-	-	-	1	-
xz	6	1	-	1	-	-	1	1
					C	*T*	*T*	*C*

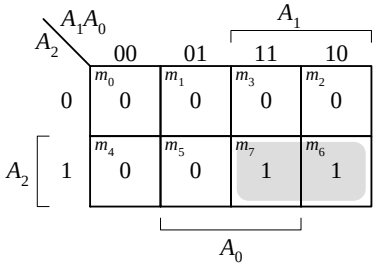
7.20



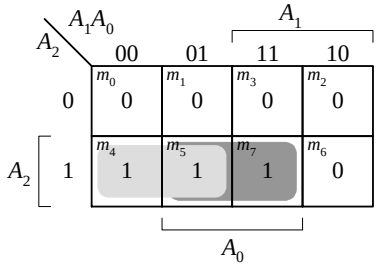
Inputs				Outputs			
w	*x*	*y*	*z*	*A*	*B*	*C*	*D*
0	0	0	0	0	1	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	1	1	0
0	0	1	1	1	0	0	1
0	1	0	0	0	1	0	1
0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0
0	1	1	1	1	0	1	1
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	0	1	0	0
1	0	1	1	1	0	0	1

1	1	0	0		0	1	0	1
1	1	0	1		1	0	1	0
1	1	1	0		0	1	1	0
1	1	1	1		1	0	0	1

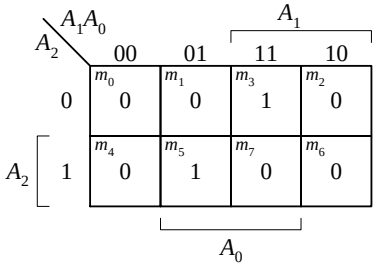
7.21 Note: See truth table in Fig. 7.12(b).



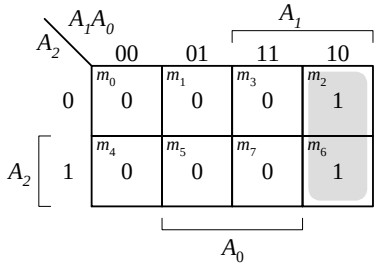
$F_1 = A_2A_1$
$F'_1 = A'_2 + A'_1$



$F_2 = A_2A'_1 + A_2A_0$
$F_2' = A'_2 + A_1A'_0$



$F_3 = A'_2A_1A_0 + A_2A'_1A_0$
$F_3' = A'_0 + A'_2A'_1 + A_2A_1$



$F_4 = A_1A'_0$
$F'_4 = A'_1 + A_0$

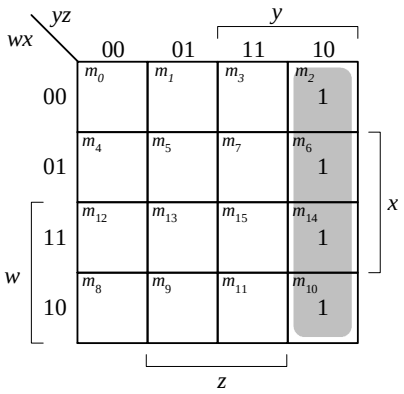
	Product term	Inputs $A_2A_1A_0$	Outputs F_1	F_2	F_3	F_4
A_2A_1	1	1 1 -	1	-	-	-
A'_2	2	0 - -	-	1	-	-
$A_1A'_0$	3	- 1 0	-	1	-	1
$A'_2A_1A_0$	4	- 1 1	-	-	1	-
$A_2A'_1$	5	1 0 1	-	-	1	-
			T	*C*	*T*	*T*

Alternative: F'_1, F'_2, F_3, F_4
(5 terms)

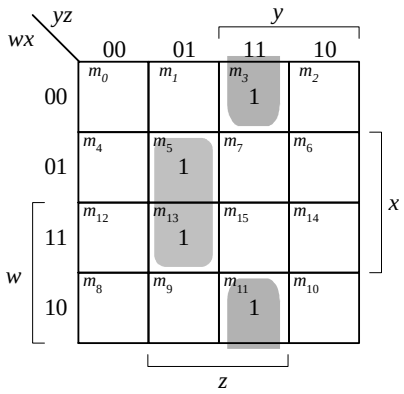
7.22

Decimal		w	x	y	z	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0	0	0	1
2	4	0	0	1	0	0	0	0	0	0	1	0	0
3	9	0	0	1	1	0	0	0	0	1	0	0	1
4	16	0	1	0	0	0	0	0	1	0	0	0	0
5	25	0	1	0	1	0	0	0	1	1	0	0	1
6	36	0	1	1	0	0	0	1	0	0	1	0	0
7	49	0	1	1	1	0	0	1	1	0	0	0	1
8	64	1	0	0	0	0	1	0	0	0	0	0	0
9	81	1	0	0	1	0	1	0	1	0	0	0	1
10	100	1	0	1	0	0	1	1	0	0	1	0	0
11	121	1	0	1	1	0	1	1	1	1	0	0	1
12	144	1	1	0	0	1	0	0	1	0	0	0	0
13	169	1	1	0	1	1	0	1	0	1	0	0	1
14	196	1	1	1	0	1	1	0	0	0	1	0	0
15	225	1	1	1	1	1	1	1	0	0	0	0	1

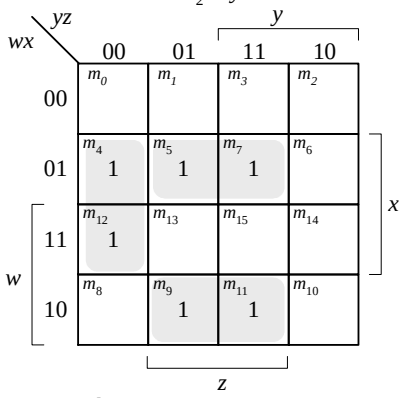
Note: b_0 = z, and b_1 = 0. ROM would have 4 inputs and 6 outputs. A 4 x 8 ROM would waste two outputs.



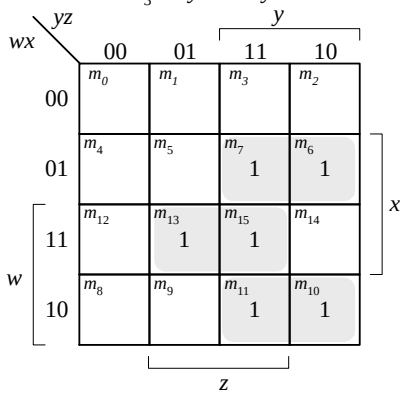
$b_2 = yx'$



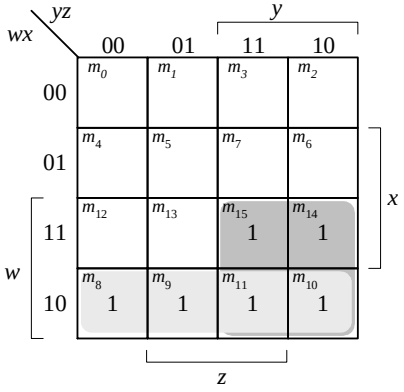
$b_3 = xy'z + x'yz$



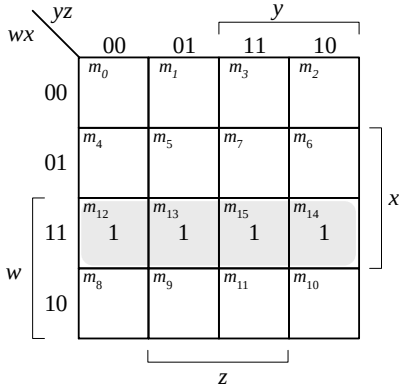
$b_4 = w'xz + xy'z' + wx'z$



$b_5 = w'xy + wxz + wx'y$



$b_6 = wy + wx'$



$b_7 = wx$

7.23

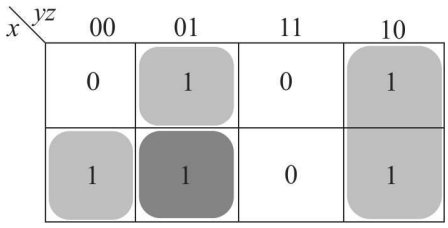
From Fig. 4-3:
w = A + BC + BD
w' = A'B' + A'C'D'
x = B'C + B'D + BC'D'
x' = B'C'D' + BC BD
y = CD + C'D'
y' = C'D + CD'
z = D'
z' = D
Use w, x', y, z (7 terms)

	Product term	Inputs *A B C D*	Outputs F_1 F_2 F_3 F_4
A	1	1 - - -	1 - - -
BC	2	- 1 1 -	1 1 - -
BD	3	- 1 - 1	1 1 - -
B'C'D'	4	- 0 0 0	- 1 - -
CD	5	- - 1 1	- - 1 -
C'D'	6	- - 0 0	- - 1 -
D'	7	- - - 0	- - - 1
			T C T T

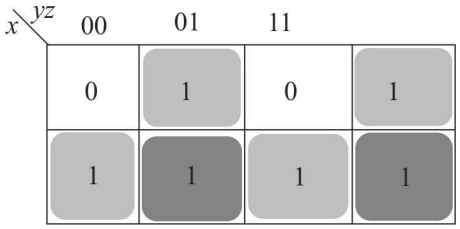
7.24

Product term	AND Inputs *A B C D*	Outputs
1	1 - - -	*w = A + BC + BD*
2	- 1 1 -	
3	- 1 - 1	
4	- 0 1 -	*x = B'C + B'D + BC'D'*
5	- 0 - 1	
6	- 1 0 0	
7	- - 1 1	*y = CD + C'D'*
8	- - 0 0	
9	- - - -	
10	- - - 0	*z = D'*
11	- - - -	
12	- - - -	

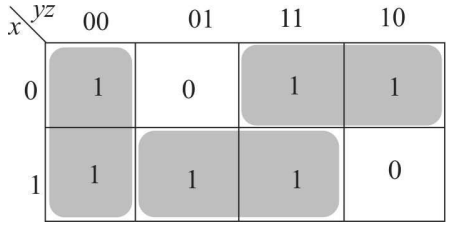
7.25



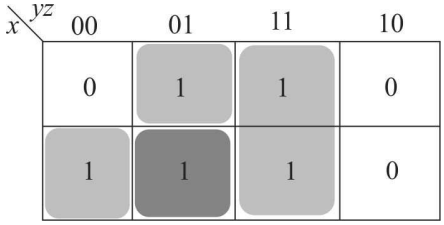
$A = \Sigma(1, 2, 4, 5, 6)$
$A = xy' + y'z + yz'$



$B = \Sigma(1, 2, 4, 5, 6, 7)$
$B = x + y'z + yz'$
$= xy + xy' + y'z + yz'$
$= A + xy$

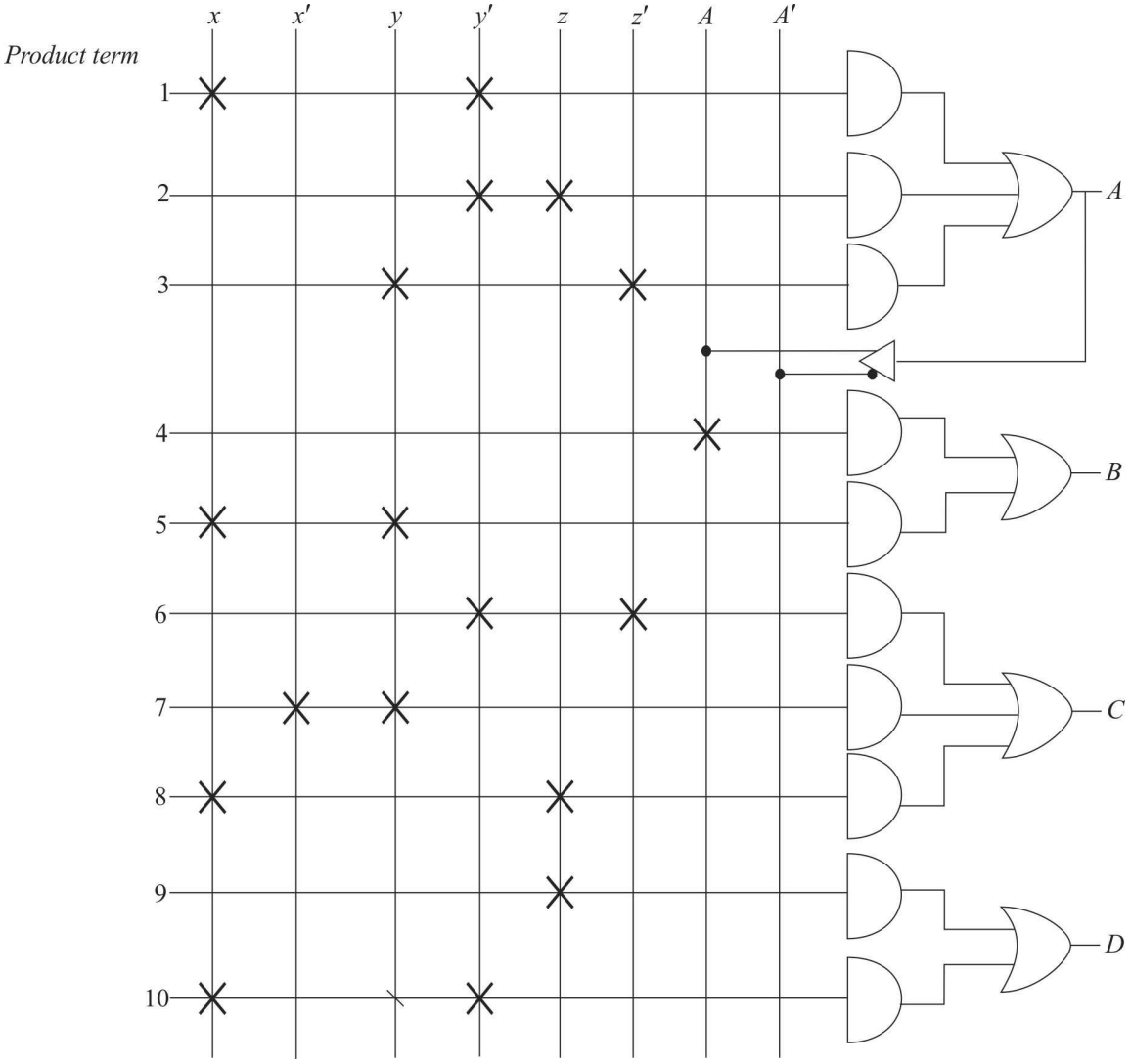


$C = \Sigma(0, 2, 3, 4, 5, 7)$
$C = y'z' + x'y + xz$



$D = \Sigma(1, 3, 4, 5, 7)$
$D = z + xy'$

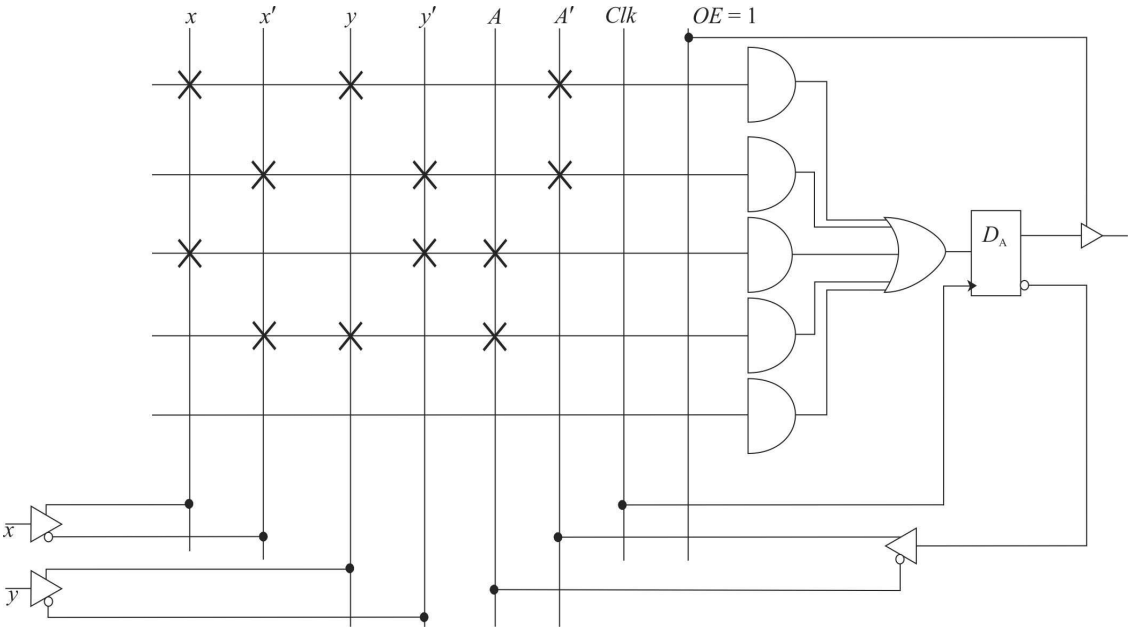
		Inputs				Outputs
	Product term	x	y	z	A	
xy'	1	1	0	-	-	
$y'z$	2	-	0	1	-	$A = xy' + y'z + yz'$
yz'	3	-	1	0	-	
A	4	-	-	-	1	$B = A + xy$
xy	5	1	1	-	-	
$y'z'$	6	-	0	0	-	
$x'y$	7	0	1	-	-	$C = y'z' + x'y + xz$
xz	8	1	-	1	-	
z	9	-	-	1	-	$D = z + xy'$
xy'	10	1	0	-	-	



7.26 $D_A = (x \oplus y \oplus A)'$

$= ((x \oplus y)' A' + (x \oplus y) A)$

$= xyA' + x'y'A' + xy'A + x'yA$

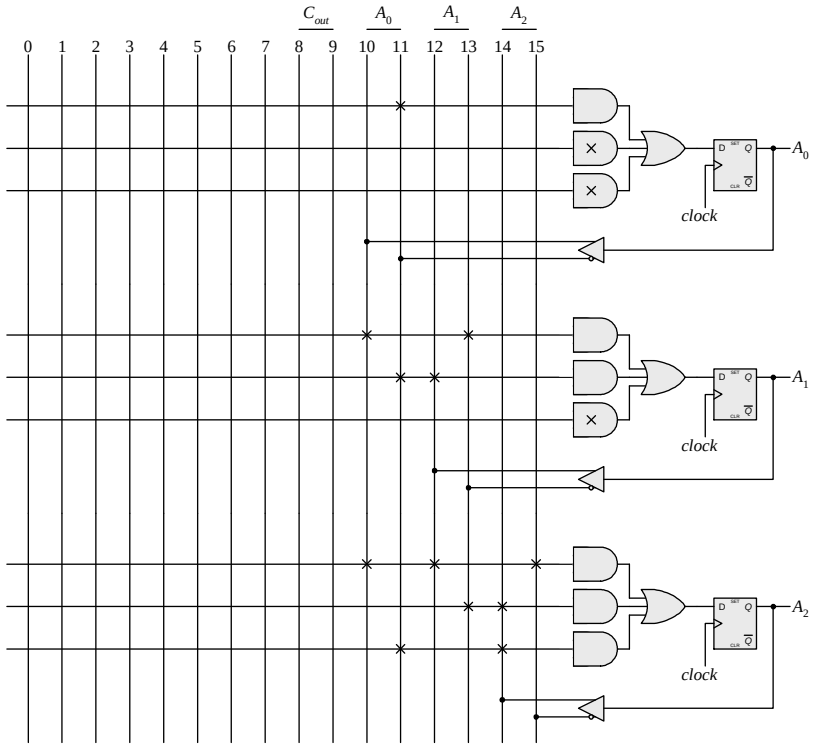


7.27

The results of Prob. 6.17 can be used to develop the equations for a three-bit binary counter with D-type flip-flops.

$DA_0 = A'_0$
$DA_1 = A'_1A_0 + A_1A'_0$
$DA_2 = A'_2 A_1A_0 + A_2A'_1 + A_2A'_0$

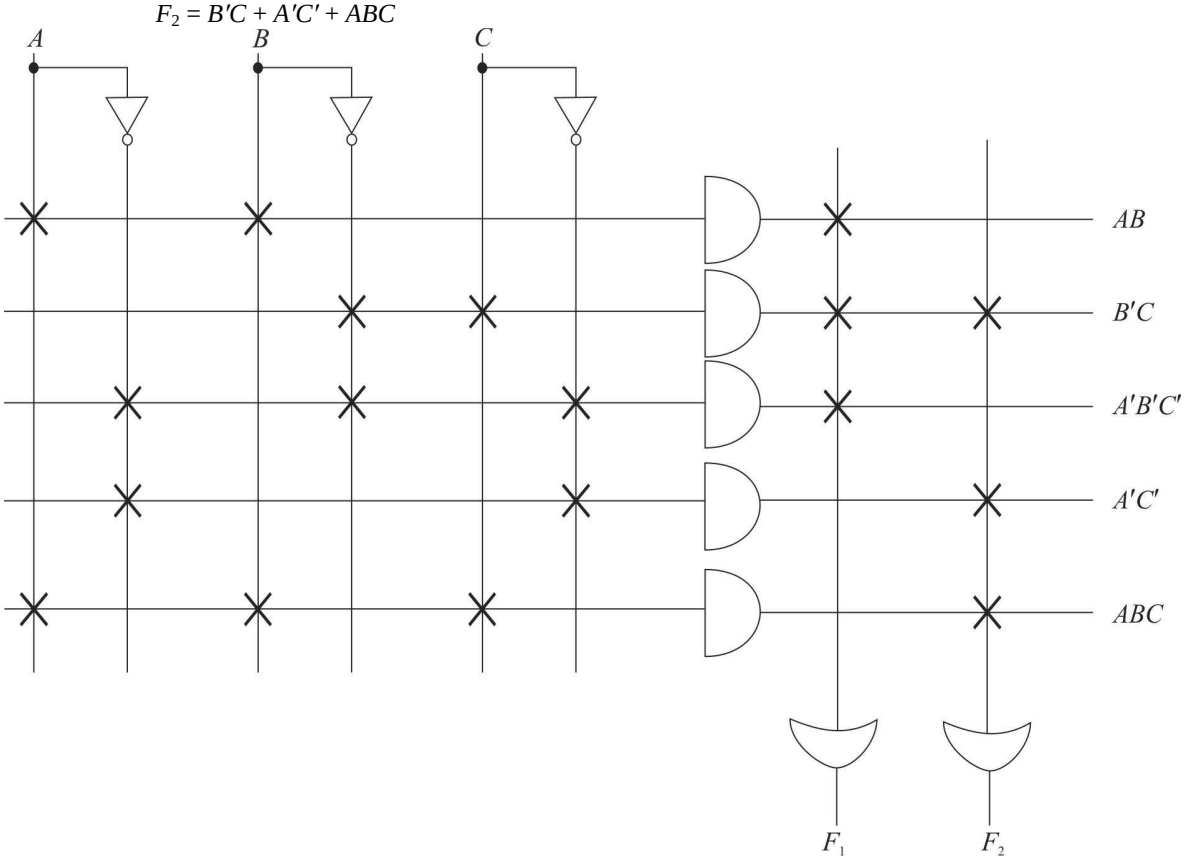
$C_{out} = A_2A_1A_0$



7.28

$F_1 = AB + B'C + A'B'C'$

$F_2 = B'C + A'C' + ABC$



7.29 PLA programming table:

		Inputs			Outputs
	Product term	x	y	A	D_A
xyA'	1	1	1	0	1
$x'y'A'$	2	0	0	0	1
$xy'A$	3	1	0	1	1
$x'yA$	4	0	1	1	1