CHAPTER 5

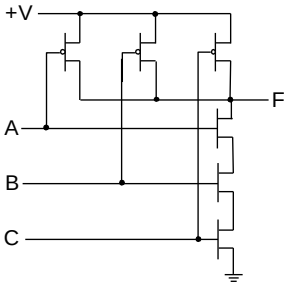
5-1.*

a) $F = (\overline{A} + B)CD$

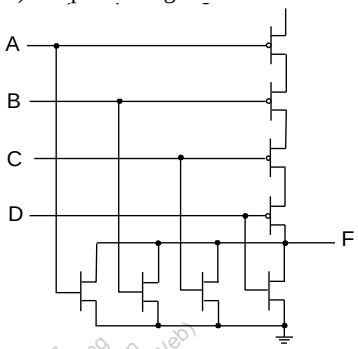
b) $G = (A + \overline{B})(\overline{C} + D)$

5-2.

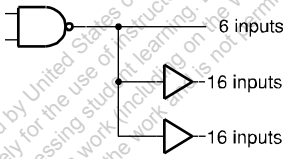
a) 3-input NAND gate



b) 4-input NOR gate

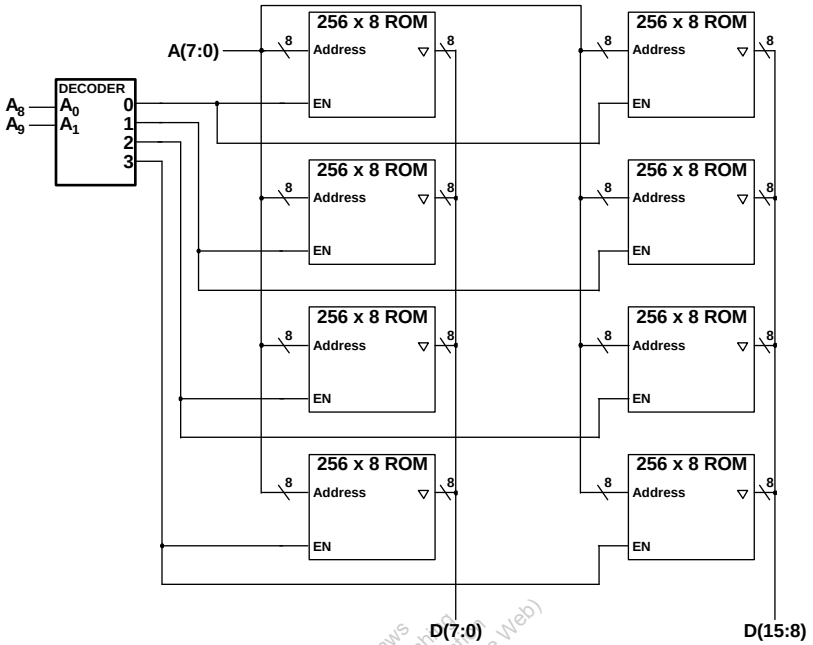


5-3.



5-4.

a)



b) 4K×32/(256×8) = 64 ROM chips

5-5.* (Errata: Change "32 X 8" to "64 X 8" ROM)

IN	OUT	IN	OUT	IN	OUT	IN	OUT
000000	0000 0000	010000	0001 0110	100000	0011 0010	110000	0100 1000
000001	0000 0001	010001	0001 0111	100001	0011 0011	110001	0100 1001
000010	0000 0010	010010	0001 1000	100010	0011 0100	110010	0101 0000
000011	0000 0011	010011	0001 1001	100011	0011 0101	110011	0101 0001
000100	0000 0100	010100	0010 0000	100100	0011 0110	110100	0101 0010
000101	0000 0101	010101	0010 0001	100101	0011 0111	110101	0101 0011
000110	0000 0110	010110	0010 0010	100110	0011 1000	110110	0101 0100
000111	0000 0111	010111	0010 0011	100111	0011 1001	110111	0101 0101
001000	0000 1000	011000	0010 0100	101000	0100 0000	111000	0101 0110
001001	0000 1001	011001	0010 0101	101001	0100 0001	111001	0101 0111
001010	0001 0000	011010	0010 0110	101010	0100 0010	111010	0101 1000
001011	0001 0001	011011	0010 0111	101011	0100 0011	111011	0101 1001
001100	0001 0010	011100	0010 1000	101100	0100 0100	111100	0110 0000
001101	0001 0011	011101	0010 1001	101101	0100 0101	111101	0110 0001
001110	0001 0100	011110	0011 0000	101110	0100 0110	111110	0110 0010
001111	0001 0101	011111	0011 0001	101111	0100 0111	111111	0110 0011

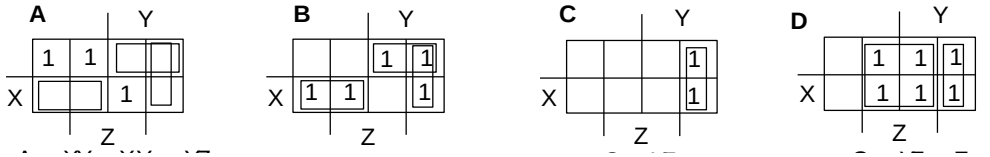
5-6.

a) 16 + 16 + 1 = 33 address bits and 16 + 1 = 17 output bits, 8G × 17
b) 8 + 8 + 1 + 1 = 18 address bits and 8 + 1 = 9 output bits, 256K×9
c) 4 × 4 = 16 address bits and 14 output bits are needed, 64K × 14
d) 4+4 = 8 address bits and 8 output bits, 256× 8

5-7.

Input			Output			
X	Y	Z	A	B	C	D
0	0	0	1	0	0	0
0	0	1	1	0	0	1
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	0	1	0	0
1	0	1	0	1	0	1
1	1	0	0	1	1	1
1	1	1	1	0	0	1

5-8.



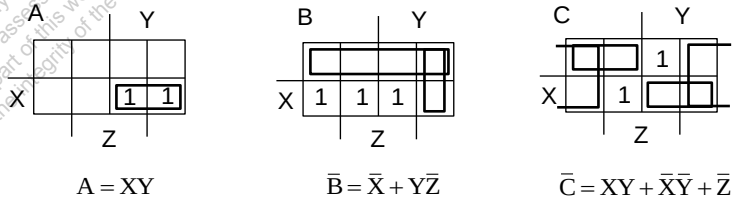
$\bar{A} = X\bar{Y} + \bar{X}Y + Y\bar{Z}$ $B = X\bar{Y} + \bar{X}Y + Y\bar{Z}$ $C = Y\bar{Z}$ $C = Y\bar{Z} + Z$

By using $\bar{A}$ instead of A and $Y\bar{Z}$ instead of Y in D, $Y\bar{Z}$ can be shared by all four functions. Further, since A is the complement of B, terms $X\bar{Y}$ and $\bar{X}Y$ can be shared between $\bar{A}$ and B. Thus, only four product terms $Y\bar{Z}$, $X\bar{Y}$, $\bar{X}Y$, and Z are required. An inversion must be programmed for A.

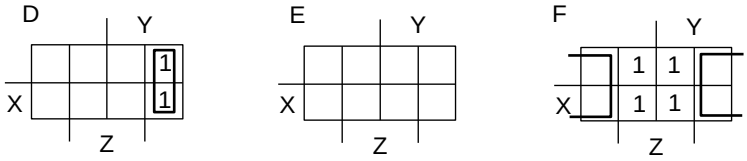
5-9.

Find the truth table and K-maps:

X	Y	Z	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1



$A = XY$ $\bar{B} = \bar{X} + Y\bar{Z}$ $\bar{C} = XY + \bar{X}\bar{Y} + \bar{Z}$

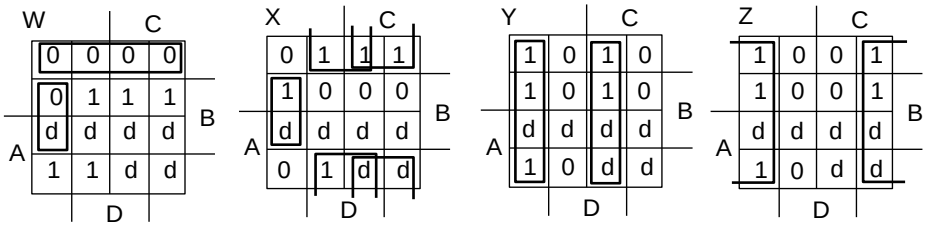


$D = Y\bar{Z}$ $E = 0$ $\bar{F} = \bar{Z}$

Implementation of A, D, and E requires only two terms, XY and $Y\bar{Z}$. Straightforward implementation of B, C, and F requires four terms, XY, XYZ, XYZ, and Z. By implementing $\bar{B}$, $\bar{C}$, and $\bar{F}$, only three additional terms $\bar{X}$, $\bar{X}\bar{Y}$, and $\bar{Z}$ are required. So we form the solution using five product terms: XY, $Y\bar{Z}$, $\bar{X}\bar{Y}$, $\bar{X}$, and $\bar{Z}$. The solution is described by the equations given with the six K-maps.

5-10.

The values given in the four K-maps come from Table 4-4 on page 224.

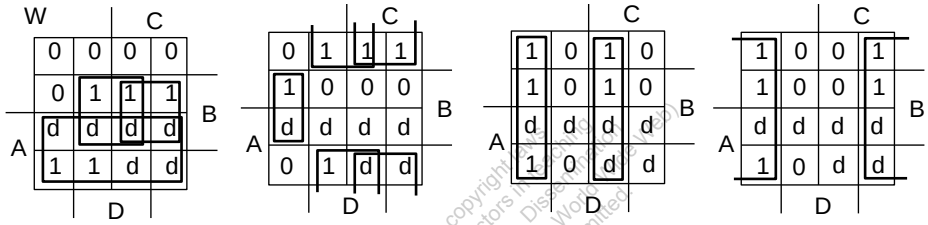


$\bar{W} = \bar{A}\bar{B} + B\bar{C}\bar{D}$ $X = B\bar{C}\bar{D} + \bar{B}C + \bar{B}D$ $Y = CD + \bar{C}\bar{D}$ $Z = \bar{D}$

In this case, shared terms are limited. One such term $B\bar{C}\bar{D}$ is generated in $\bar{W}$.

5-11.*

Assume 3-input OR gates.

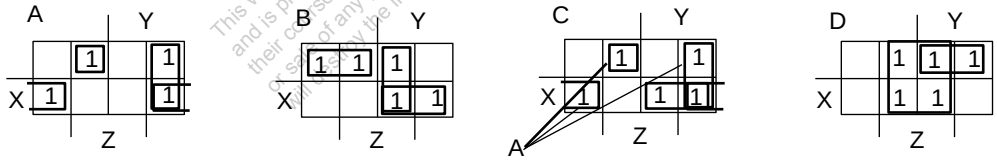


$W = A + BC + BD$ $X = B\bar{C}\bar{D} + \bar{B}C + \bar{B}D$ $Y = CD + \bar{C}\bar{D}$ $Z = \bar{D}$

Each of the equations above is implemented using one 3-input OR gate. Four gates are used.

5-12.

Figure 5-10 uses 3-input OR gates.

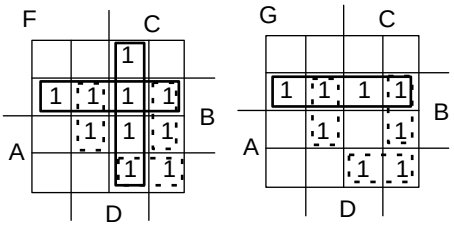


$A = X\bar{Z} + Y\bar{Z} + \bar{X}\bar{Y}Z$ $B = XY + YZ + \bar{X}\bar{Y}$ $C = A + XY$ $D = \bar{X}Y + Z$

A, B, and D each require three or fewer product terms so can be implemented with 3-input OR gates. C requires four terms so cannot be implemented with a 3-input OR gate. But because the first PAL device output can used as an input to implement other functions it can be assigned to A and A can then be used to implement C using just two inputs of a 3-input OR gate.

5-13.

Figure 5-10 uses 3-input OR gates.



Straightforward implementation of F requires five prime implicants and of G requires four prime implicants, but only 3 inputs are available on the PAL OR gates. So sum-of-products that can be factored from F and G or both and implemented by the other PAL cells are needed. A single sum of products that will work is $H = A\bar{B}C + B\bar{C}D + BC\bar{D}$. The terms of H are shown with dotted lines on the K-maps. Using H:

$F = H + CD + \bar{A}B$

$G = H + \bar{A}B$

There are other possible functions for H and corresponding results for F and H.

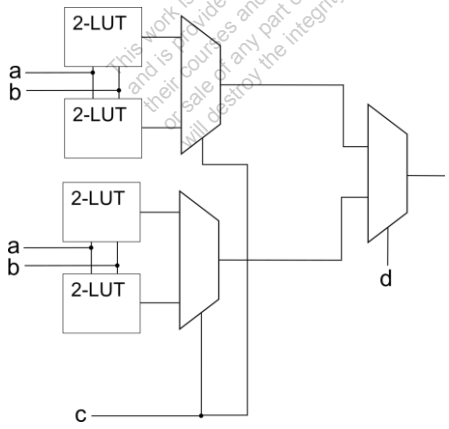
5-14.

a) $F(A, B, C) = C(\bar{A}\bar{B} + B) + \bar{C}(\bar{A}\bar{B} + A)$

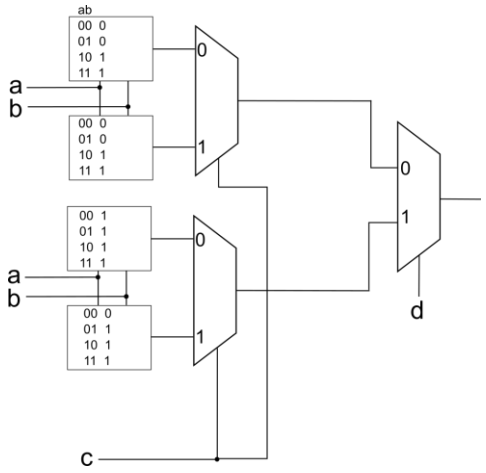
b) $F(A, B, C) = C(\bar{A}B + A\bar{B}) + \bar{C}(\bar{A} + B)$

5-15.

a)



b) Using Shannon's expansion theorem, $F(A, B, C, D) = D(C(A+B)+\overline{C}(1)) + \overline{D}(C(A)+\overline{C}(A))$. Then using the 4-LUT from part a, the function can be implemented as:



5-16.

Assuming that the upper input of each mux is selected with a 0:

a) MUX2: 0, MUX3: 1, MUX4: 0
b) MUX2: 1, MUX3: 1, MUX4: 1
c) MUX2: 1, MUX3: 1, MUX4: 0

5-17.

Invert the b input so that the upper 2-LUT is equal to $f(a, b) = a \oplus \bar{b}$ and the lower 2-LUT is equal to $f(a, b) = a\bar{b}$

5-18.

The state machine is a Moore machine since the output Z depends only the current state. Using a state assignment of 0 for State0 and 1 for State1, then Z is the same as the state of the flip-flop.
Then the configuration bits 0:10 = 1001 0011 111 (assuming that the a input of the 2-LUTs is the most significant bit of the address and that a = in1 and b = in2).