The Essentials of Computer Organization and Architecture

Linda Null and Julia Lobur
Jones and Bartlett Publishers, 2003

Chapter 3 Instructor's Manual

Chapter Objectives

Chapter 3, Boolean Algebra and Digital Logic, is a classic presentation of digital logic and how it relates to Boolean algebra. This chapter covers both combinational and sequential logic in sufficient detail to allow the reader to understand the logical makeup of more complicated MSI (medium scale integration) circuits (such as decoders). More complex circuits, such as buses and memory, are also included. We have included optimization and Kmaps in a special "Focus On" section.

This chapter should be covered after Chapter 1, but before Chapters 4 through 11.

Lectures should focus on the following points:

- **Boolean algebra.** Boolean algebra is a very natural way to represent digital information, and thus is an important concept to study if one wishes to understand computers. The common Boolean functions AND, OR and NOT should be covered, as well as truth table and Boolean identities.
- **Logic gates.** Boolean algebra allows us to represent expressions in an abstract form. Digital circuits are built using logic gates, the basic building blocks for digital systems. It is important to understand how Boolean expressions are physically implemented.
- **Digital components.** The complexity of a Boolean expression (and its corresponding implementation) has a direct impact on the complexity of the resulting digital circuit. Therefore, it is important to understand how Boolean algebra relates to digital components.
- **Combinational circuits.** Combinational circuits represent a large portion of the actual components used in today's machines. Adders, decoders, multiplexers, and parity checkers are just a few examples. Understanding these simple circuits goes a long way towards understanding the overall system.
- **Sequential circuits.** These circuits allow the computer to retain, or remember, values. Clocks are used to synchronize these circuits. SR, JK and D flip-flops can be used to build a large number of components, particularly registers and memory.

Required Lecture Time

Chapter 3 can typically be covered in 6 lecture hours, depending on how detailed one wishes to go into Kmaps. We suggest that the focus be on understanding the relationship between Boolean expression and digital circuits. Kmaps are on method that can be used to simplify Boolean expressions, resulting in less complex circuits. If time permits, Kmap simplification can be covered, and will most likely require an additional 1-2 hours of lecture.

Lecture Tips

The material in this chapter is not intended to be a thorough coverage of digital logic. Many schools require students to take a separate course in digital logic and design. However, this chapter covers adequate concepts necessary to understand how computer systems work in general.

The focus for this chapter should be on the relationship of Boolean expressions and digital logic circuits, since there is a one-to-one correspondence between a Boolean expression and its digital representation. Boolean identities can be used to reduce these expressions, and thus to minimize both combinational and sequential circuits. However, most "real-life" circuits are quite large and complex, and specialized software is used for simplification. However, going over these simple examples will give students an understanding of how components are designed, how they work, and why it is important to simplify the components.

If teachers are interested in doing more with digital logic, we suggest using a digital logic simulator (such as Diglog) to allow the students to design and implement different circuits. Sample Diglog assignments are given at the end of this chapter of the instructor's manual.

Answers to Exercises

- 1. Construct a truth table for the following:
  - a. *xyz + (xyz)'*
  - b. *x(yz'+x'y)*

Ans.

a.

x	*y*	*z*	*xyz*	*(xyz)'*	*xyz + (xyz)'*
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	0	1

b.

x	*y*	*z*	*yz'*	*x'y*	*(yz' + x'y)*	*x(yz '+ x'y)*
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	1	1	0
0	1	1	0	1	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	1	1
1	1	1	0	0	0	0

2. Construct a truth table for the following:

a. $xyz + x(yz)' + (xyz)'$ b. $(x+y)(x+z)(x'+z)$

Ans.

a.

x	y	z	xyz	x(yz)'	(xyz)'	Sum
0	0	0	0	0	1	1
0	0	1	0	0	1	1
0	1	0	0	0	1	1
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	0	1	1	1
1	1	1	1	0	0	1

b.

x	y	z	x+y	x + z	x' + z	(x + y)(x + z)(x' + z)
0	0	0	0	0	1	1
0	0	1	0	1	1	1
0	1	0	1	0	1	1
0	1	1	1	1	1	1
1	0	0	1	1	0	0
1	0	1	1	1	1	1
1	1	0	1	1	0	0
1	1	1	1	1	1	1

---

3. Using DeMorgan's Law, write an expression for the complement of *F* if $F(x,y,z) = x(y' + z)$.

Ans.

$F(x,y,z) = x(y' + z)$
$F'(x,y,z) = (x(y' + z))'$
$= x'+(y' + z)'$
$= x' + yz'$

---

4. Using DeMorgan's Law, write an expression for the complement of *F* if $F(x,y,z) = xy + x'z + yz'$.

Ans.

$F(x,y,z) = xy + x'z + yz'$
$F'(x,y,x) = (xy + x'z + yz')'$
$= (xy)'(x'z)'(yz')'$
$= (x' + y')(x + z')(y' + z)$ *(not simplified)*

---

5. Using DeMorgan's Law, write an expression for the complement of F if
$F(w,x,y,z) = xyz' (y'z + x)' + (w'yz + x')$.

Ans.

$F(w,x,y,z) = xyz'(y'z + x)'+(w'yz + x)$
$F'(w,x,y,z) = (xyz'(y'z + x)'+(w'yz + x))'$
$= (xyz')'+((y'z + x)')'(w'yz + x)'$
$= (xyz')'+((y'z + x)(w'yz + x')')$
$= x'+y'+z+((y'z + x)((w'yz)'x'')$
$= x'+y'+z+((y'z + x)((w+y' + z')x))$

---

6. a. Use the Boolean identities to prove the Absorption Laws.
b. Prove DeMorgan's Laws are valid.
(Blue indicates a correction was made from what is in the textbook.)

Ans.

a. $x(x + y) = xx + xy$ *Distributive*
$= x + xy$ *Idempotent*
$= x(1 + y)$ *Distributive*
$= x(1)$ *Null*

	=	*x*	*Identity*
x + xy	=	*x(1 + y)1*	*Distributive*
	=	*x(1)*	*Null*
	=	*x*	*Identity*

b. Using a truth table:

x	*y*	*(xy)'*	*x' + y'*
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

x	*y*	*(x+y)*	*(x+y)'*	*x'*	*y'*	*x'y'*
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

---

7. Is the following distributive law valid or invalid? Prove your answer.

x XOR (y AND z) = (x XOR y) AND (x XOR z)

Ans.

False. One method of proof uses a truth table. A more challenging approach using identities employs the relation: *a* XOR *b* = *ab' + a'b*

---

8. Show that *x = xy + xy '*

a. Using truth tables b. Using Boolean identities

Ans.

a.

x	*y*	*xy*	*xy'*	*xy + xy'*
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

The final column is simply x.

b.

xy + xy'	=	*x(y + y')*	*Distributive*
	=	*x(1)*	*Inverse*
	=	*x*	*Identity*

---

9. Show that *xz = (x + y)(x+y')(x' + z)*

a. Using truth tables b. Using Boolean identities

Ans.

a.

x	*y*	*x+y*	*x + y'*	*x' + z*	*(x + y)(x + y')(x' + z)*	*xz*
0	0	0	1	1	0	0
0	1	1	0	1	0	0
1	0	1	1	0	0	0
1	1	1	1	1	1	1

b.

	(x + y)(x + y')(x '+ z)	
=	*x(x + y')(x' + z)+y(x + y')(x' + z)*	*Distributive*
=	*x(xx' + xz + x'y' + y'z) + y(xx' + xz + x'y' + y'x)*	*Distributive*
=	*x(0 + xz + x'y' + y'z) + y(0 + xz + x'y' + y'x)*	*Inverse*
=	*x(xz + x'y' + y'z) + y(xz + x'y' + y'x)*	*Identity*
=	*xxz + xx'y' + xy'z + xyz + x'yy' + yy'z*	*Distributive*
=	*xxz + xy'z + xyz*	*Inverse*
=	*xz + xy'z + xyz*	*Idempotent*
=	*xz(1 + y' + y)*	*Distributive*
=	*xz(1)*	*Null*
=	*xz*	*Identity*

---

10. Simplify the following functional expressions using Boolean algebra and its identities. List the identity used at each step.

    a. F*(x,y,z) = x'y + xyz' + xyz*
    b. F*(w,x,y,z) = (xy' + w'z)(wx' + yz')*
    c. F*(x,y,z) = (x + y)'(x' + y')'*

Ans.

a. *x'y + xyz' + xyz*

=	*x'y + xy(z' + z)*	*Distributive*
=	*x'y + xy(1)*	*Inverse*
=	*x'y + xy*	*Identity*
=	*(x' + x)y*	*Distributive*
=	*(1)y*	*Inverse*
=	*y*	*Identity*

b. *(xy' + w'z)(wx' + yz')*

=	*xy'wx' + xy'yz' + w'zwx' + w'zyz'*	*Distributive*
=	*(xx')y'w + (y'y)xz' + (w'w)zx' + (zz')w'y*	*Associative*
=	*(0)y'w + (0)xz' + (0)zx' + 0(w'y)*	*Inverse*
=	*0 + 0 + 0 + 0*	*Null*
=	*0*	*Idempotent*

c. *(x + y)'(x'+y')'*

=	*(x + y)' (x''y'')*	*DeMorgan*
=	*(x'y')(x''y'')*	*DeMorgan*
=	*(x'y')(xy)*	*Double Complement*
=	*(x'x)(y'y)*	*Associative*
=	*(0)(0)*	*Inverse*
=	*0*	*Idempotent*

♦ 11. Simplify the following functional expressions using Boolean algebra and its identities. List the identity used at each step.

    ♦ a. *x'yz + xz*
    ♦ b. *(x + y)'(x' + y')'*
    ♦ c. *(x'y''z)'*

Ans.

a. *x'yz + xz*

=	*x'yz + xz*(1)	*Identity*
=	*x'yz + xz(y + y')*	*Inverse*
=	*x'yz + xyz + xy'z*	*Distributive and Commutative*
=	*x'yz + (xyz + xyz) + xy'z*	*Idempotent*
=	*(x'yz + xyz) + (xyz + xy'z)*	*Associative*
=	*(x' + x)yz + (y + y')xz*	*Distributive (two applications)*
=	(1)*(yz)* + (1)*xz*	*Inverse*
=	*yz + xz*	*Identity*

b.	*(x+y)'(x' + y')'*	=	*(x'y')(x''y'')*	*DeMorgan's*
		=	*(x'y')(xy)*	*Double Complement*
		=	*(x'x)(y'y)*	*Commutative and Associative*
		=	(0)(0)	*Inverse*
		=	0	*Idempotent*
c.	*(x'x''y)'*	=	*(x'xy)'*	*Double complement*
		=	*(0y)'*	*Inverse*
		=	(0)'	*Null*
		=	1	*Defn. of complement*

---

12. Simplify the following functional expressions using Boolean algebra and its identities. List the identity used at each step.

    a. *(ab+c+df)ef*
    b. *x + xy*
    c. *(xy' + x'z)(wx'+yz')*

Ans.

a.	*(ab + c + df)ef*	=	*abef + cef + dfef*	*Distributive*
		=	*abef + cef + deff*	*Commutative*
		=	*abef + cef + def*	*Idempotent*
b.	*x + xy*	=	*x(1 + y)*	*Distributive*
		=	*x(1)*	*Null*
		=	*x*	*Idempotent*

c. *(xy' + x'z)(wx' + yz')*

= xy'wx' + xy'yz' + x'zwx' + x'zyz'	*Distributive*
= xx'y'w + y'yxz' + x'x'wz + zz'x'y	*Commutative*
= (xx')y'w + (y'y)xz' + (x'x')wz + (zz')x'y	*Associative*
= (0)y'w + (0)xz' + x'wz + (0)x'y	*Inverse*
= 0 + 0 + x'wz + 0	*Null*
= x'wz	*Identity*

---

13. Simplify the following functional expressions using Boolean algebra and its identities. List the identity used at each step.

    a. *xy + xy'*
    b. *x'yz + xz*
    c. *wx + w(xy + yz')*

Ans.

a.	*xy + x'y*	=	*x(y + y')*	*Distributive*
		=	*x(1)*	*Inverse*
		=	*x*	*Identity*

b.	*x'yz + xz*	=	*x'yz + xz(1)*	*Identity*
		=	*x'yz + xz(y + y')*	*Inverse*
		=	*x'yz + xzy + xzy'*	*Distributive*
		=	*x'yz + (xzy + xzy) + xzy'*	*Idempotent*
		=	*(x'yz + xzy) + (xzy + xzy')*	*Associative*
		=	*(x'yz + xyz) + (xyz + xy'z)*	*Commutative*
		=	*(x' + x)yz + xz(y + y')*	*Distributive*
		=	*(1)yz + xz(1)*	*Inverse*
		=	*yz + xz*	*Identity*

c. $wx + w(xy + yz')$ = $wx + wxy + wyz'$ *Distributive*
= $wx(1 + y) + wyz'$ *Distributive*
= $wx(1) + wyz'$ *Null*
= $wx + wyz'$ *Identity*

---

14. Use any method to prove the following either True for False.

$yz + xyz' + x' y' z = xy + x' z$

Ans.

Using identities:

$yz + xyz' + x'y'z$ = $(x +x')yz + xyz' + x'y'z$
= $xyz + x'yz + xyz' + x'y'z$
= $(x'y'z + x'yz) + (xyz + xyz')$
= $x'z(y' + y) + xy\ (z + z')$
= $x'z + xy$

Using truth tables:

x	y	z	yz	xyz'	$x'y'z$	$yz + xyz' + x'y'z$	xy	$x'z$	$xy + x'z$
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	0	1	1
0	1	0	0	0	0	0	0	0	0
0	1	1	1	0	0	1	0	1	1
1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	1	0	0	1	0	1	1	0	1
1	1	1	1	0	0	1	1	0	1

♦ 15. Using the basic identities of Boolean algebra, show that:

$x(x' +y) = xy$

Ans.

$x(x' + y)$ = $xx' + xy$
= $0 + xy$
= xy

---

*16. Using the basic identities of Boolean algebra, show that:

$x + x' y = x + y.$

Ans.

$x + x'y$ = $x(y + y') + x'y$
= $xy + xy' + x'y$
= $xy + xy' +x'y +xy$
= $x(y + y') + y(x' + x)$
= $x(1) + y(1)$
= $x + y$

♦ 17. Using the basic identities of Boolean algebra, show that:

$xy + x'z + yz = xy + x' z$

Ans.

$$\begin{aligned} xy + x'z + yz &= xy + x'z + (1)yz \\ &= xy + x'z + (x + x')yz \\ &= xy + x'z + xyz + x'yz \\ &= (xy + xyz) + (x'z + x'yz) \\ &= xy(1 + z)+x'z(1 + y) \\ &= xy + x'z \end{aligned}$$

♦ 18. The truth table for a Boolean expression is shown below. Write the Boolean expression in sum-of-products form.

x	*y*	*z*	*F*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Ans.

$F(x,y,z) = x'y'z + x'yz' + xy'z + xyz'$

---

19. The truth table for a Boolean expression is shown below. Write the Boolean expression in sum-of-products form.

x	*y*	*z*	*F*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Ans.

$F(x,y,z) = x'y'z' + x'yz + xyz'$

---

20. Draw the truth table and rewrite the expression below as the complemented sum of two products:

$xz' + y'z + x' y$

Ans.

The truth table for $xz' + y'z + x'y$ is:

x	y	z	xz'	$y'z$	$x'y$	$xz' + y'z + x'y$
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	0

The complemented sum of two products forms is:

$$(x'y'z' + xyz)'$$

---

21. Given the Boolean function: $F(x,y,z)=x' y + xyz'$
    a. Derive an algebraic expression for the complement of F. Express in sum-of-products form.
    b. Show that $FF' = 0$.
    c. Show that $F + F' = 1$.

Ans.

a. $(x'y + xyz')' = xy' + xz + x'y' + y' + y'z$ (not simplified)

b. $FF' = (x'y + xyz')(xy' + xz + x'y' + y' + y'z)$
$= x'yxy' + x'xyz + x'x'yy' + x'yy' + x'yy'z + xxyy'z' + xyz'xz + xyz'x'y' + xyz'y' + xyz'y'z$
$= 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0$
$= 0$

c. $F + F' = (x'y + xyz') + (xy' + xz + x'y' + y' + y'z)$

x	y	z	$x'y$	xyz'	xy'	xz	$x'y'$	y'	$y'z$	$F + F'$
0	0	0	0	0	0	0	1	1	0	1
0	0	1	0	0	0	0	1	1	1	1
0	1	0	1	0	0	0	0	0	0	1
0	1	1	1	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	1	0	1
1	0	1	0	0	1	1	0	1	1	1
1	1	0	0	1	0	0	0	0	0	1
1	1	1	0	0	0	1	0	0	0	1

---

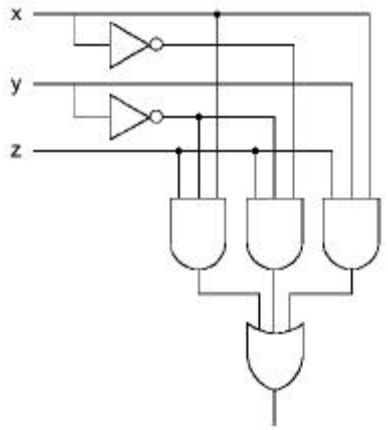
22. Given the function: $F(xy,z)= xy'z + x'y'z + xyz$
    a. List the truth table for F.
    b. Draw the logic diagram using the original Boolean expression
    c. Simplify the expression using Boolean algebra and identities.
    d. List the truth table for your answer in part c.
    e. Draw the logic diagram for the simplified expression in part c.

Ans.

a.

x	*y*	*z*	*xy'z*	*x'y'z*	*xyz*	*F*
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	0
1	0	1	1	0	0	1
1	1	0	0	0	0	0
1	1	1	0	0	1	1

b. Logic diagram for *xy'z + x'y'z + xyz*

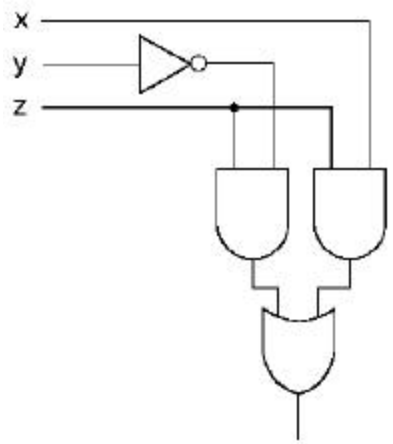


c. *xy'z + x'y'z + xyz* = *(xy'z + xy'z) + x'y'z + xyz*
= *(xy'z + x'y'z) + (xy'z + xyz)*
= *(x + x')y'z + (y' + y)xz*
= *y'z + xz*

d.

x	*y*	*z*	*y'z*	*xz*	*F*
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	1	1	1

e. Logic diagram for $y'z + xz$:

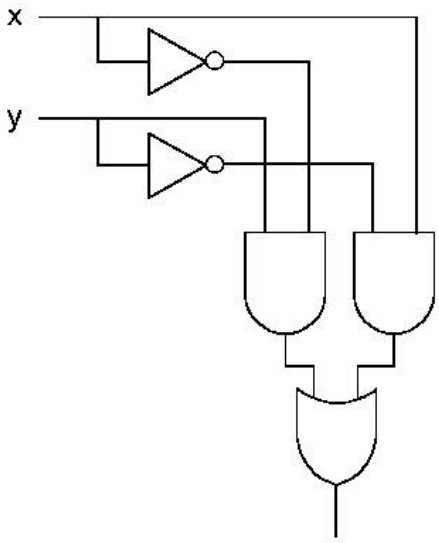


---

23. Construct the XOR operator using only AND, OR and NOT gates.

Ans.

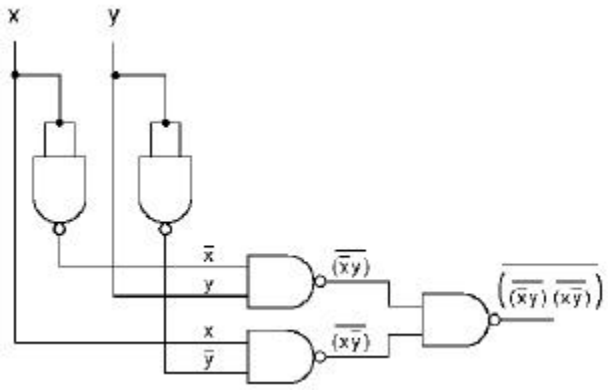
$x \text{ XOR } y = x'y + xy'$



---

*24. Construct the XOR operator using only NAND gates.
Hint: x XOR $y = ((x\ '\ y)'\ (xy'\ )'\ )'$

Ans.



---

25. Design a circuit with three inputs (x,y and z) representing the bits in a binary number, and three outputs (a,b and c) also representing bits in a binary number. When the input is 0, 1, 2 or 3, the binary output should be one less than the input. When the binary input is 4,5,6, or 7, the binary output should be one greater than the input. Show your truth table, all computations for simplification and the final circuit.

Ans.

The truth table is:

x	y	z	a	b	c
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	1	1	1

This simplifies to: $a = x$

$b = x'yz + xy'z + xyz' + xyz = yz + xz + xy$

$c = x'yz' + xy'z' + xyz' + xyz = xz' + xy + yz'$

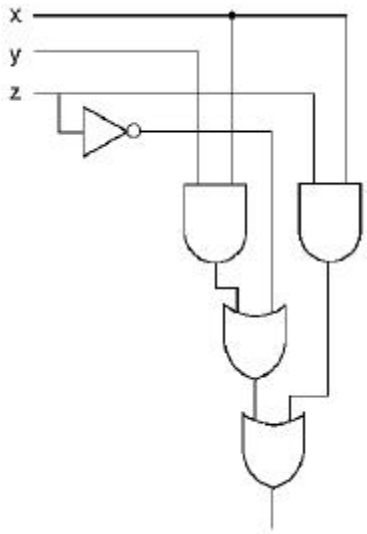
(Final circuit not drawn.)

---

26. Draw the combinational circuit that directly implements the Boolean expression:

F*(x,y,z)= xz + (xy + z')*

Ans.

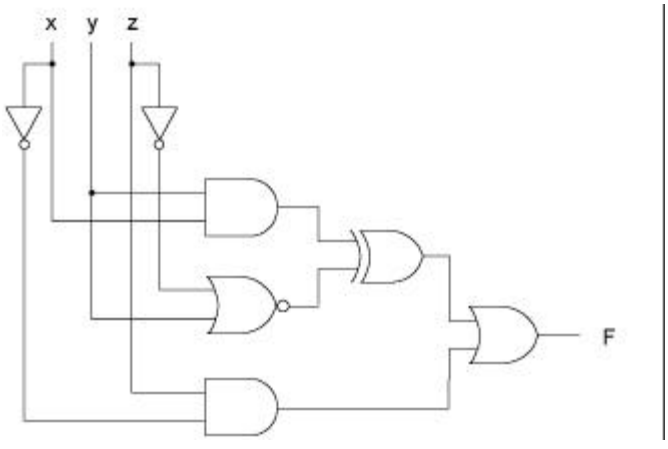


---

27. Draw the combinational circuit that directly implements the Boolean expression:

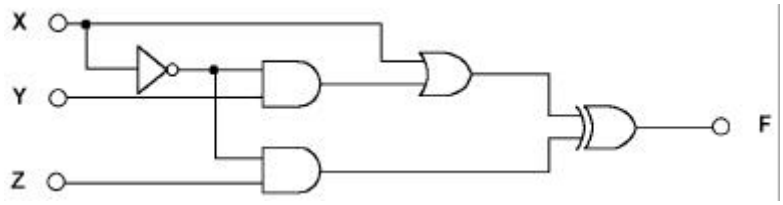
F*(x,y,z)=(xy XOR (y+z')') + x'z*

Ans.



---

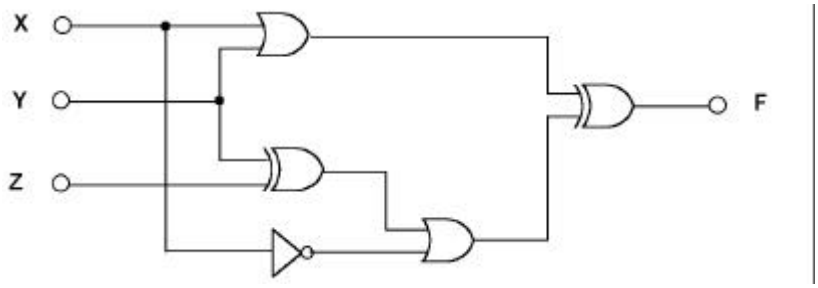
28. Find the truth table that describes the following circuit:



Ans.

x	*y*	*z*	*F*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

♦ 29. Find the truth table that describes the following circuit:

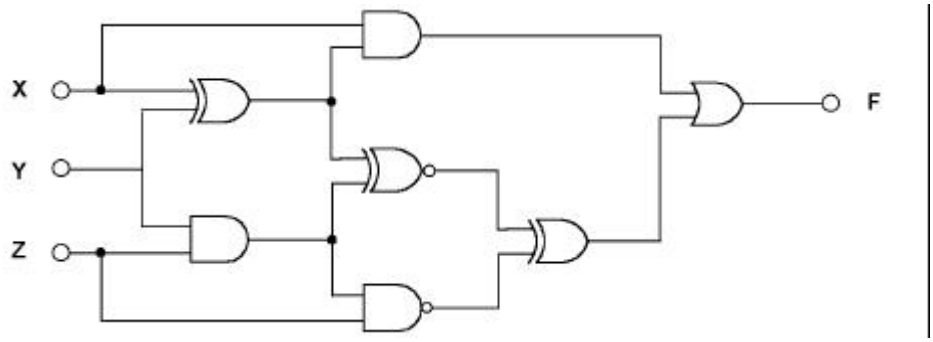


Ans.

x	*y*	*z*	*F*
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

---

30. Find the truth table that describes the following circuit:



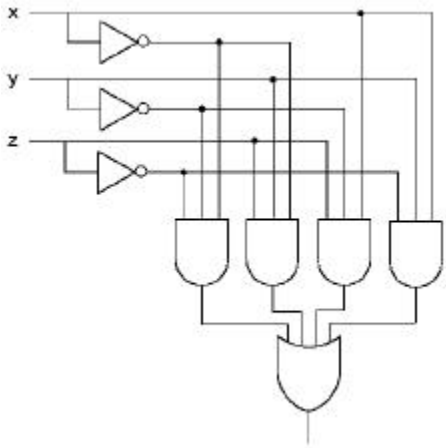
Ans.

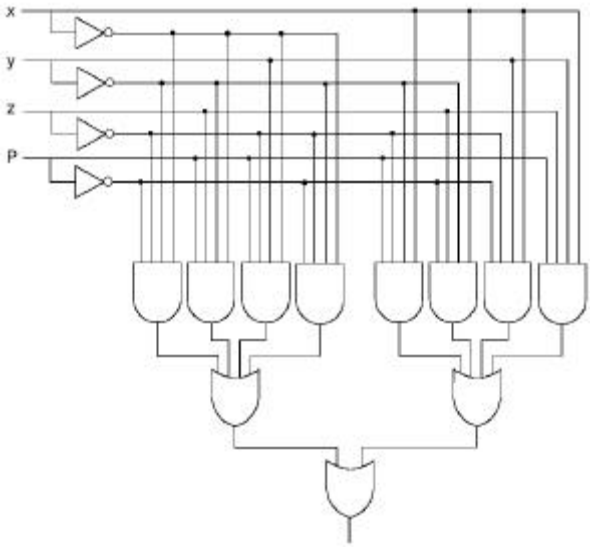
x	*y*	*z*	*F*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

---

31. Draw circuits to implement the parity generator and parity checker shown in Tables 3.11 and 3.12, respectively.

Ans.

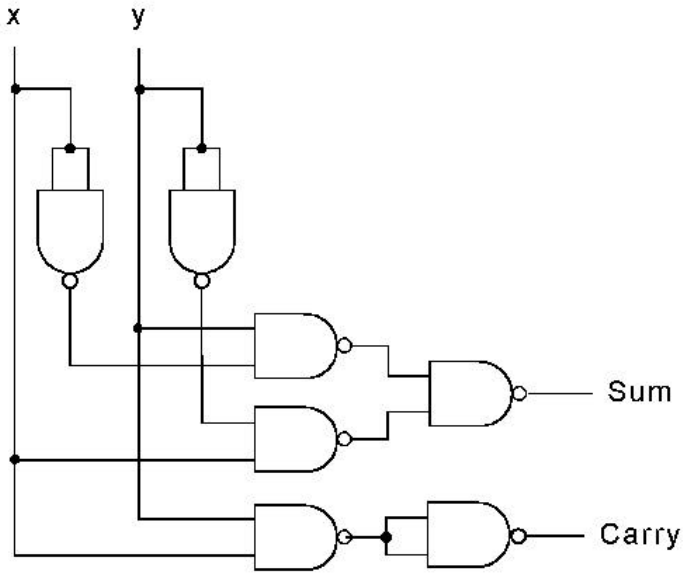




---

32. Draw a half adder using only NAND gates.

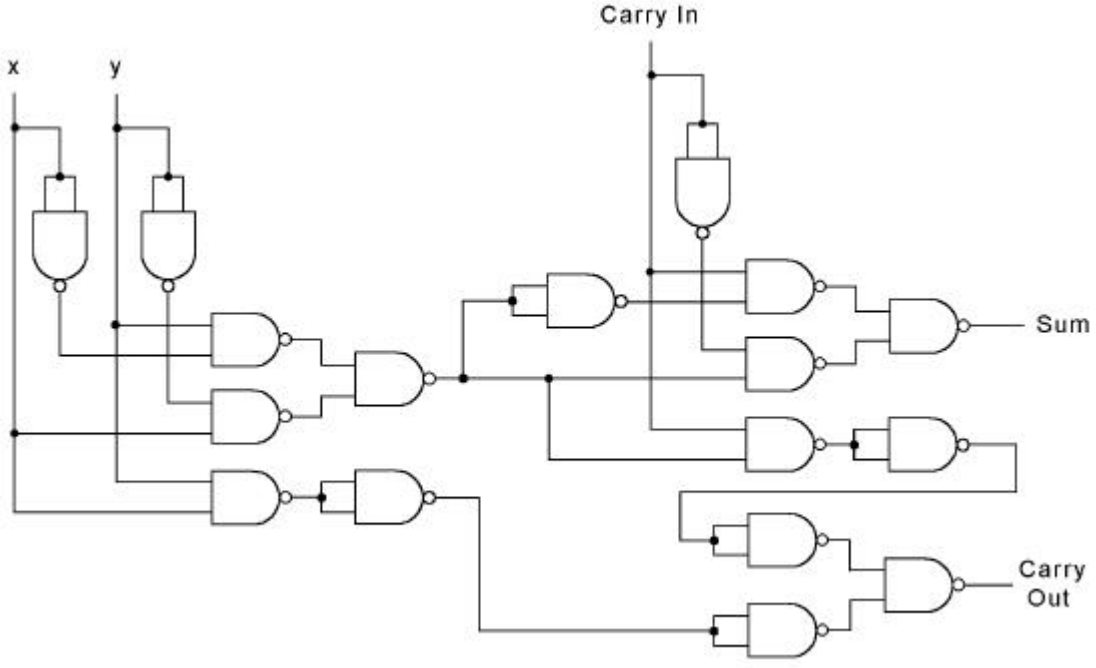
Ans.



---

33. Draw a full adder using only NAND gates.

Ans.



---

34. Tyrone Shoelaces has invested a huge amount of money into the stock market and doesn't trust just anyone to give him buying and selling information. Before he will buy a certain stock, he must get input from three sources. His first source is Pain Webster, a famous stock broker. His second source is Meg A. Cash, a self-made millionaire in the stock market, and his third source is Madame LaZora, world famous psychic. After several months of receiving advice from all three, he has come to the following conclusions:

a. Buy if Pain and Meg both say yes and the psychic says no.
b. Buy if the psychic says yes.
c. Don't buy otherwise.

Construct a truth table and find the minimized Boolean function to implement the logic telling Tyrone when to buy.

Ans.

Pain (x)	*Meg (y)*	*Madame (z)*	*Buy?*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The "buy" function is:
$x'y'z + x'yz + xy'z + xyz' + xyz$
$= z + xy$

♦ *35. A very small company has hired you to install a security system. The brand of system that you install is priced by the number of bits encoded on the proximity cards that allow access to certain locations in a facility. Of course, this small company wants to use the fewest bits possible (spending the least amount of money as possible) yet have all of their security needs met. The first thing that you need to do is to determine how many bits each card requires. Next, you have to program card readers in each secured location so that they respond appropriately to a scanned card.

This company has four types of employees and five areas that they wish to restrict to certain employees. The employees and their restrictions are as follows:

a. The Big Boss needs access to the executive lounge and the executive washroom.
b. The Big Boss's secretary needs access to the supply closet, employee lounge, and executive lounge.
c. Computer room employees need access to the server room and the employee lounge.
d. The janitor needs access to all areas in the workplace.

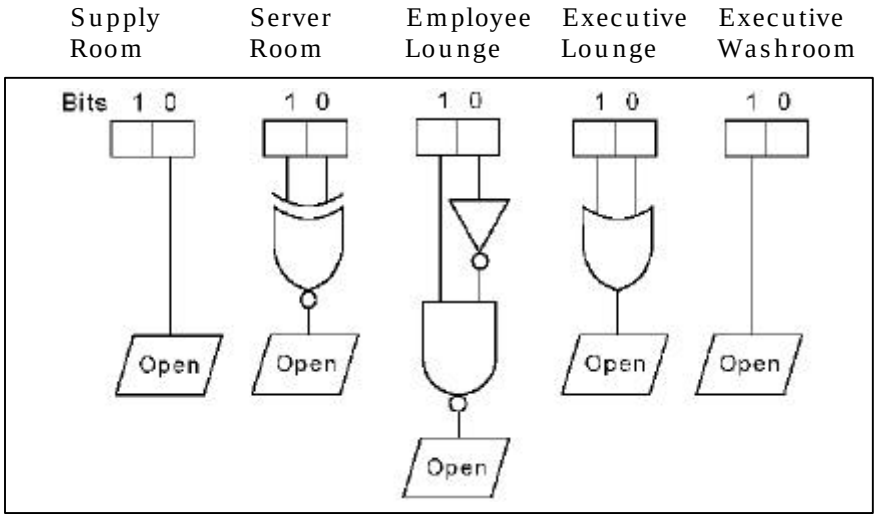
Determine how each class of employee will be encoded on the cards and construct logic diagrams for the card readers in each of the five restricted areas.

Ans.

The values assigned for the inputs (the card encoding), determine the exact design of each reader. One encoding is shown in the table below.

Encoding	Employee Class	Has authorization to enter:				
		Supply Room	Server Room	Employee Lounge	Executive Lounge	Executive Washroom
00	IT Workers		X	X		
01	Secretary	X		X	X	
10	Big Boss				X	X
11	Janitor	X	X	X	X	X

Based on this coding, the card reader for the server room can be implemented as shown below:



---

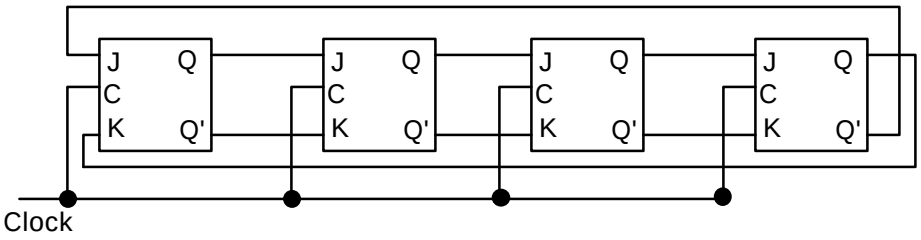
36. How many 256x8 RAM chips are needed to provide a memory capacity of 4096 bytes?
    a. How many bits will each memory address contain?
    b. How many address lines must go to each chip?
    b. How many lines must be decoded for the chip select inputs? Specify the size of the decoder.

Ans.

To get 4096 bytes of memory, one would need 16 256x8 RAM chips (256x16=4096).

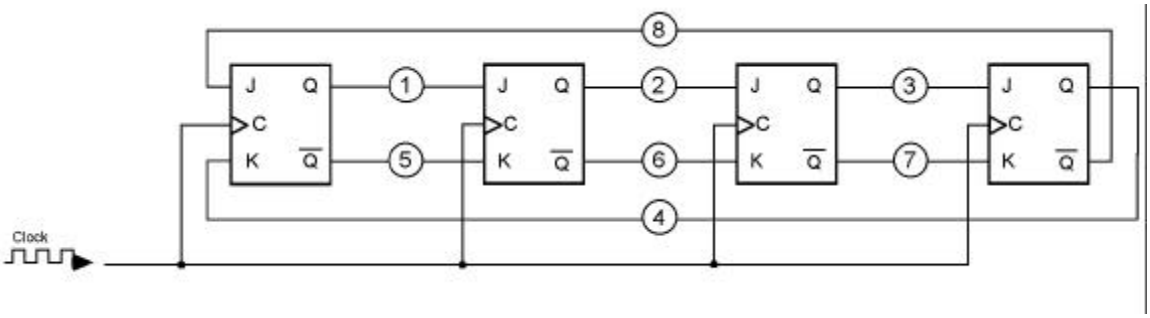
a. 8
b. 8
c. 4, which implies a 4-to-16 decoder.

♦ *37. Investigate the operation of the following circuit. Assume an initial state of 0000. Trace the outputs (the Qs) as the clock ticks and determine the purpose of the circuit. You must show the trace to complete your answer.



Ans.

Start by numbering the lines between the flip-flops as shown:



Complete the following chart for $t = 0$ through 8:

t	Lines that are "off"	Lines that are "on"
0	1,2,3,4	5,6,7,8
1	5,2,3,4	1,6,7,8
2	5,6,3,4	1,2,7,8
3	5,6,7,4	1,2,3,8
4	5,6,7,8	1,2,3,4
5	1,2,3,8	5,2,3,4
6	1,2,7,8	5,6,3,4
7	1,2,3,8	5,6,7,4
8	1,2,3,4	5,6,7,8

This counts as follows:
0000 1000 1100 1110 1111 0111
0011 0001 0000

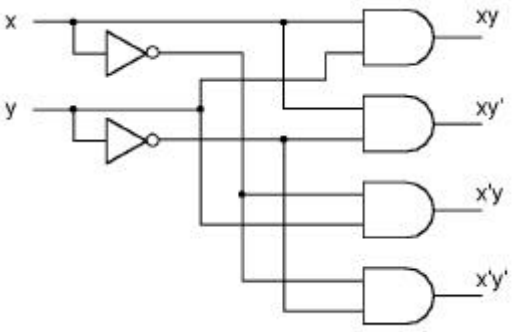
---

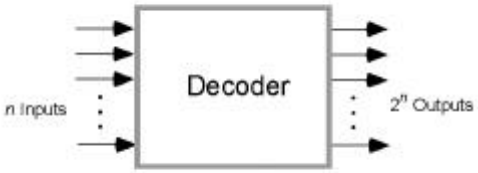
38. Describe how each of the following circuits works and indicate typical inputs and outputs. Also provide a carefully labeled "black box" diagram for each.

a) Decoder b) Multiplexer

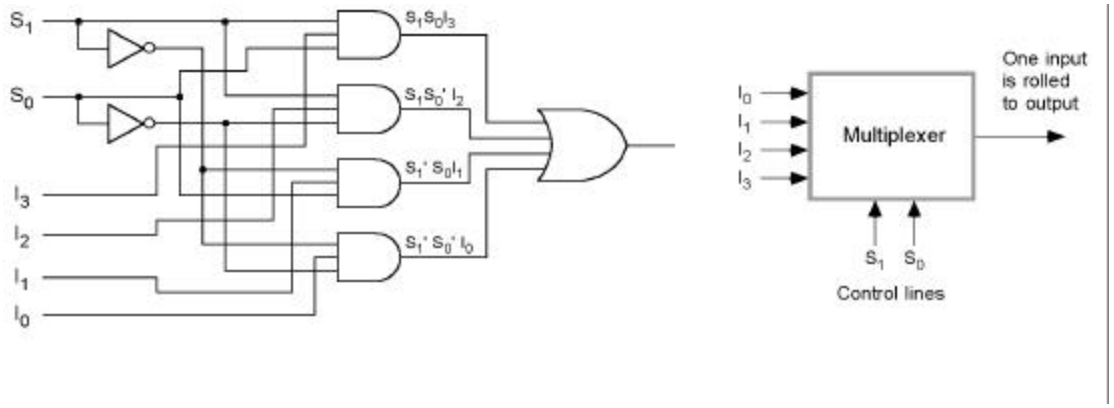
Ans.

a. A decoder takes n inputs and uses those inputs to select exactly one of (typically) 2^n outputs.

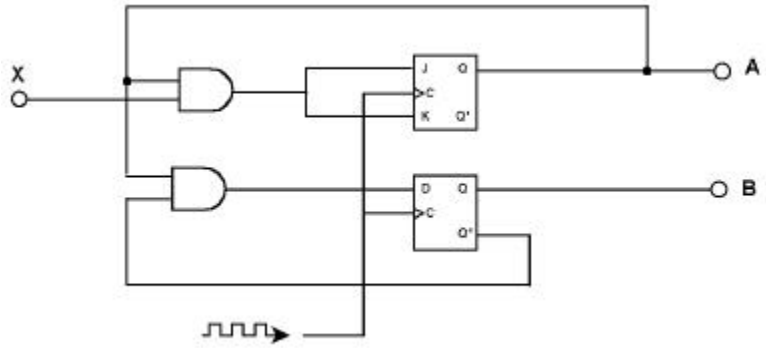




b. A multiplexer uses n control lines to select one of its input lines to route through to the output.



♦ 39. Complete the truth table for the following sequential circuit:

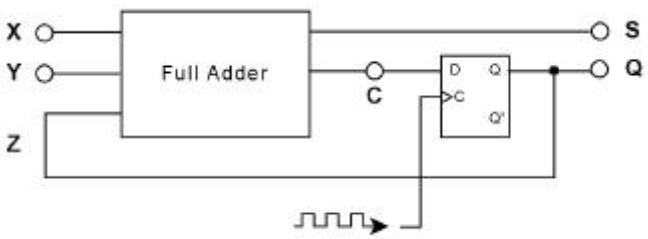


Ans.

			Next State	
A	B	X	A	B
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	0
1	1	1	0	0

---

40. Complete the truth table for the following sequential circuit:

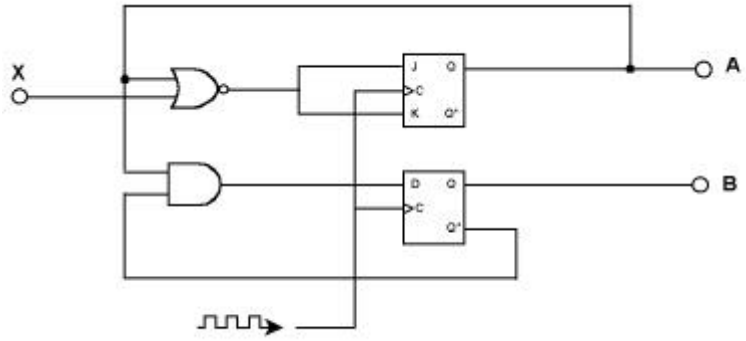


Ans.

			Next State	
X	Y	Z(Q)	S	C(Q)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

---

41. Complete the truth table for the following sequential circuit:

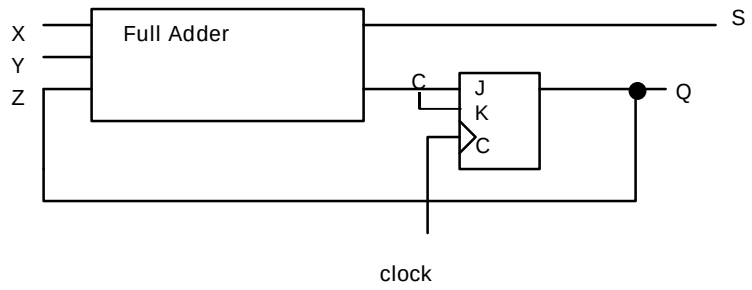


Ans.

			Next State	
A	B	X	A	B
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	0

---

*42. A sequential circuit has one flip-flop; two inputs X and Y; and one output, S. It consists of a full-adder circuit connected to a JK flip-flop, as shown below. Fill in the characteristic table for this sequential circuit by completing the *Next State* and *Output* columns.



Ans.

Present State Q(t)	Inputs X	Y	Next State Q(t + 1)	Output S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	1

---

*43. A Mux-Not flip-flop (MN flip-flop) behaves as follows: If M = 1, the flip-flop complements the current state. If M = 0, the next state of the flip-flop is equal to the value of N.

a. Derive the characteristic table for the flip-flop.
b. Show how a JK flip-flop can be converted to a MN flip-flop by adding gate(s) and inverter(s).

Ans.

a. The characteristic table can be thought of as either of the two following:

M	N	Q(t+1)
0	0	0
0	1	1
1	0	Q'(t)
1	1	Q'(t)

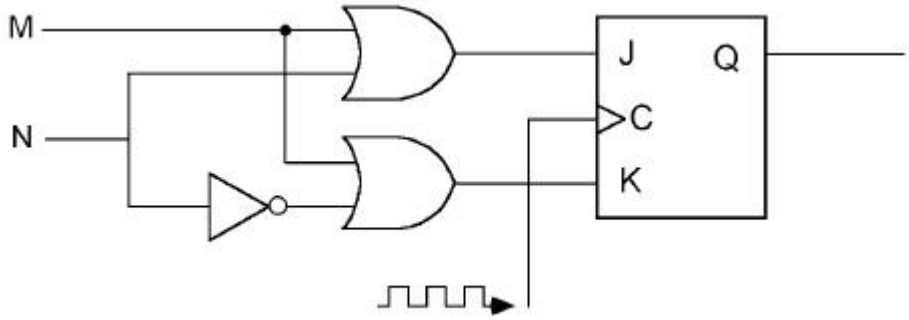
M	N	Q(t)	Q(t+1)
0	0	X*	0
0	1	X	1
1	X	0	1
1	X	1	0

* (where X means either 0 or 1)

b. To convert a JK flip-flop to an MN flip-flop, we must express J and K in terms of M and N, as follows (remember with a JK flip-flop, 01 as input means reset, 10 means set, 00 means no change, and 11 means toggle):

M	N	J	K
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	1

From the above we can see that J is M + N and K is M + N'. So we have:



---

44. List the steps necessary to read a word from memory in the 4x3 memory circuit shown in Figure 3.25.

Ans.

1. An address is asserted on S_0 and S_1.
2. Write enable is set to low.
3. The decoder using the control lines enables only one AND gate, selecting a given word in memory.
4. The current value stored in that word is put on the output lines

---

Chapter 3A: Focus On Kmaps

1. Write a simplified expression for the Boolean function defined by each of the following Kmaps.

a.

X \ YZ	00	01	11	10
0	0	1	1	0
1	1	0	0	1

b.

X \ YZ	00	01	11	10
0	0	1	1	1
1	1	0	0	0

c.

X \ YZ	00	01	11	10
0	1	1	1	0
1	1	1	1	1

Ans.

a. *x'z + xz'*

b. *x'z + x'y + xy'z'*

c. x + y' + z

---

2. Create the Kmaps and then simplify for the following functions:

a. F(*x,y,z*) = *x'y'z' + x'yz + x'yz'*

b. F(*x,y,z*) = *x'y'z' + x'yz' + xy'z' + xyz'*

c. F(*x,y,z*) = *y'z' + y'z + xyz'*

Ans.

a. *x'y'z' + x'yz + x'yz'*

Simplifies to:
x'y + x'z'

X \ YZ	00	01	11	10
0	1	0	1	1
1	0	0	0	0

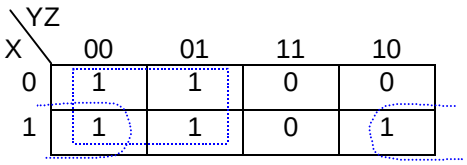
b. *x'y'z' + x'yz' + xy'z' + xyz'*

Simplifies to:
z'

X \ YZ	00	01	11	10
0	1	0	0	1
1	1	0	0	1

c. *y'z' + y'z + xyz'*

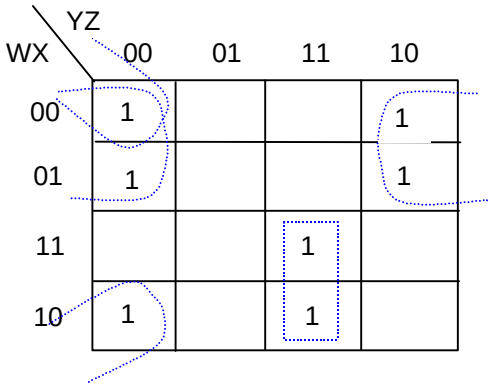
Simplifies to:
y' + xz'



---

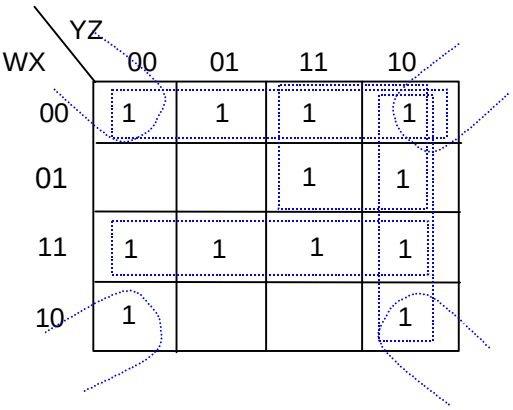
3. Write a simplified expression for the Boolean function defined by each of the following Kmaps.

a.



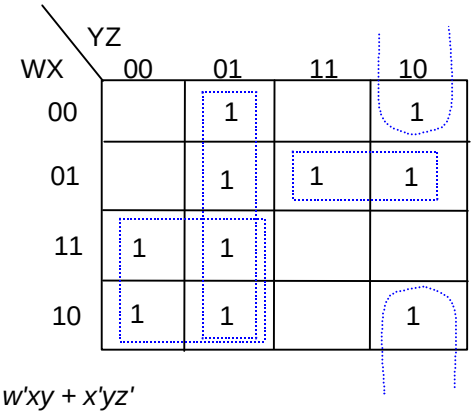
Ans.
w'z' + w'y'z' + wyz

b.



Ans.
w'x' + wx + w'y + yz' + x'z'

c.



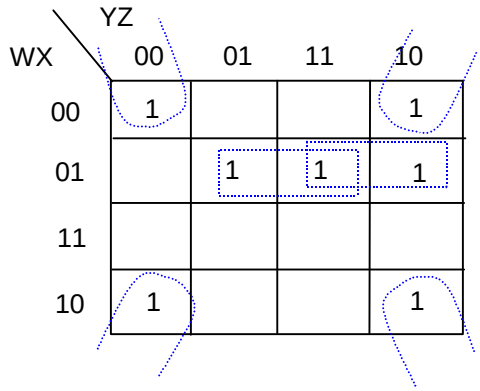
Ans. *wy' + y'z + w'xy + x'yz'*

---

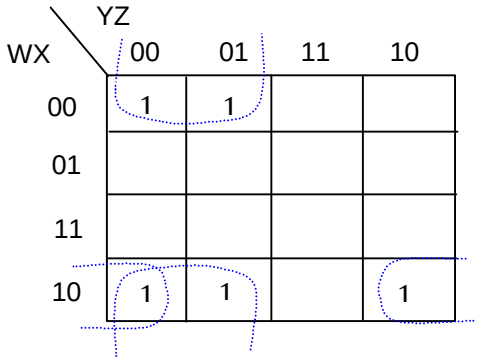
4. Create the Kmaps and then simplify for the following functions:
   a. F(*w,x,y.z*) = *w'x'y'z' + w'x'yz' + w'xy'z + w'xyz + w'xyz' + wx'y'z' + wx'yz'*
   b. F(*w,x,y,z*) = *w'x'y'z' + w'x'y'z + wx'y'z + wx'yz' + wx'y'z'*
   c. F(*w,x,y,z*) = *y'z + wy' + w'xy + w'x'yz' + wx'yz'*

Ans.

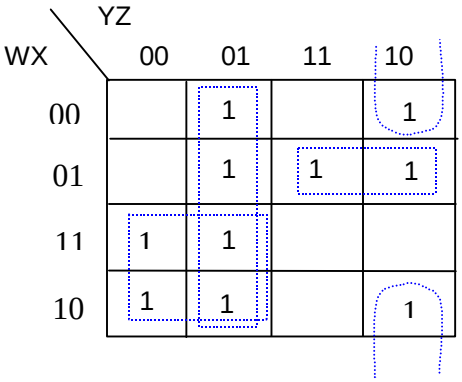
a. *w'xz + w'xy + x'y'z' + x'yz*



b. *x'y' + wx'z'*

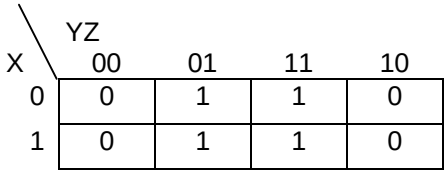


c. *y'z + wy' + w'xy + x'yz'*



---

5. Given the following Kmap, show algebraically (using Boolean identities) how the 4 terms reduce to 1 term.

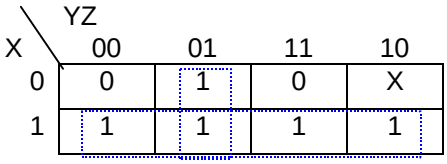


Ans. *x'y'z+x'yz+xy'z+xyz* = *x'z(y'+y)+xz(y'+y)*
= *x'z+xz*
= *z(x'+x)*
= *z*

---

6. Write a simplified expression for the Boolean function defined by each of the following Kmaps.

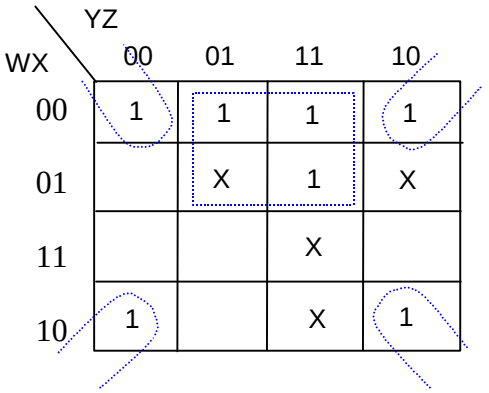
a.



Ans.

$x + y'z$ (We don't want to include the "don't care" as it doesn't help us.)

b.



Ans..

$x'z' + w'z$

Sample Exam Questions

1. Using DeMorgan's Law, write an expression for the complement of F if

   $F(x,y,z) = x(y' + z)$

Ans.

$F' = x' + yz'$

---

2. Use any method to prove the following either TRUE or FALSE. You must show ALL work.

   $yz + xyz' + x'y'z = xy + x'z$

Ans.

Use either truth tables or Boolean algebra. This is true.

---

3. Draw the combinational circuit that directly implements the Boolean expression:

   $F(x,y,z) = (xy \text{ XOR } (y + z')') + x'z$

Ans.

Circuit not drawn.

---

4. Which of the following Boolean expressions is not logically equivalent to all the others?

   A. $((w' + x')(w' + y)(w' + z))'$
   B. $w(x + y' + z')$
   C. $w + x + y' + z$
   D. $wx + wy' + wz'$

Ans.

C is not the same.

---

5. Identify whether each of the following is based on combinational logic (**C**) or sequential logic (**S**).

_____	a. SR latch	_____	e. ALU
_____	b. register	_____	f. memory
_____	c. decoder	_____	g. half-adder
_____	d. clocked D latch	_____	h. a digital circuit to implement $F(x,y,z) = x'y + y'z'$

Ans..

a. S b. S c. C d. S e. C f. W g. C h. C

---

6. Is the following distributive law valid? Prove your answer.

a XOR bc = (a XOR b)(a XOR c)

Ans.

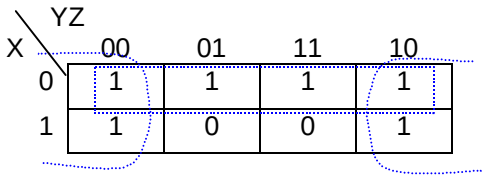
No. This can be shown using truth tables.(Not true for 101 and 110)

---

7. Write a simplified expression for the Boolean function defined by the following K-map:

X \ YZ	00	01	11	10
0	1	1	1	1
1	1	0	0	1

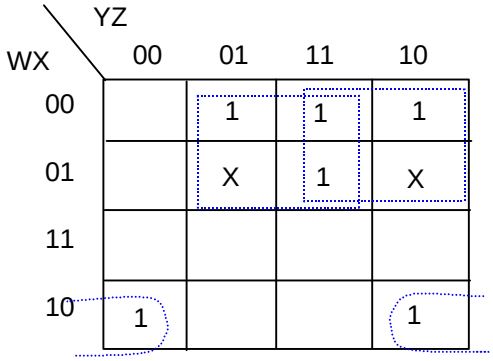
Ans:



---

8. Write a simplified expression for the Boolean function defined by the following K-map:

WX \ YZ	00	01	11	10
00		1	1	1
01		X	1	X
11				
10	1			1



Ans.

F = w'z + w'y + wx'z'

---

9. Given the following K-map:

WX \ YZ	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	0	0	0
10	1	0	0	1

a. Construct the truth table for F.

b. Using the K-map, write the minimal sum of products form for F (minimized)

c. Using the K-map, write the minimal sum of products form for F'.

Ans.

a.

w	*x*	*y*	*z*	*F*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

b. *F = x'z' + w'xz + w'xy OR x'z' + w'xz + w'yz'*

c. *To find F', circle the zeros, resulting in: F' = wx + wz + xy'z' + x'z*

TRUE OR FALSE.

_____ 1. The inputs of a half-adder include a carry-in and a single bit from both numbers we wish to sum.

_____ 2. An AND gate is the same as an OR gate with its inputs complemented.

_____ 3. Taking a Boolean expression's dual is the same as applying DeMorgan's Law.

_____ 4. Any Boolean function can be generated using only AND and OR gates, assuming we have access to the variables and their complements (i.e., we have X and X').

_____ 5. A clocked (or synchronous) SR flip-flop holds the present state when the clock is inactive regardless of the changes on the flip-flop's S and R inputs.

_____ 6. A certain "black box" has 4 inputs and 16 outputs. Only one of the 16 outputs is asserted (set to 1) for each unique combination of the inputs. This circuit is called a multiplexer.

_____ 7. A 16-input multiplexer can select between any one of 16 inputs and requires 16/2=8 "select" or control lines.

Ans.

1. F 2. F 3. F 4. T 5. T 6. F 7. F

Sample Digital Logic Homework Assignments

1. You have been hired by a home security company to design and implement a home alarm system. The logic of the system is as follows: once the alarm system has been armed, it is to sound if the front door is opened, the back door is opened or either of two windows is opened (you can assume there are only two windows).

   Design the necessary circuit using DIGLOG to implement the situation described above. Your circuit should have five inputs (A = alarm, F = front door, B = back door, W1 = window 1 and W2 = window 2). A = 1 means the system is armed; A = 0 means it is disarmed. F = 1 means the front door is open; F = 0 means it is closed. (Similarly for the back door and the windows.) There should be one output, S. When S = 1 the alarm should sound; S = 0 means the alarm is silent. *Please use these letters to indicate the inputs and the output so all projects are consistent.*

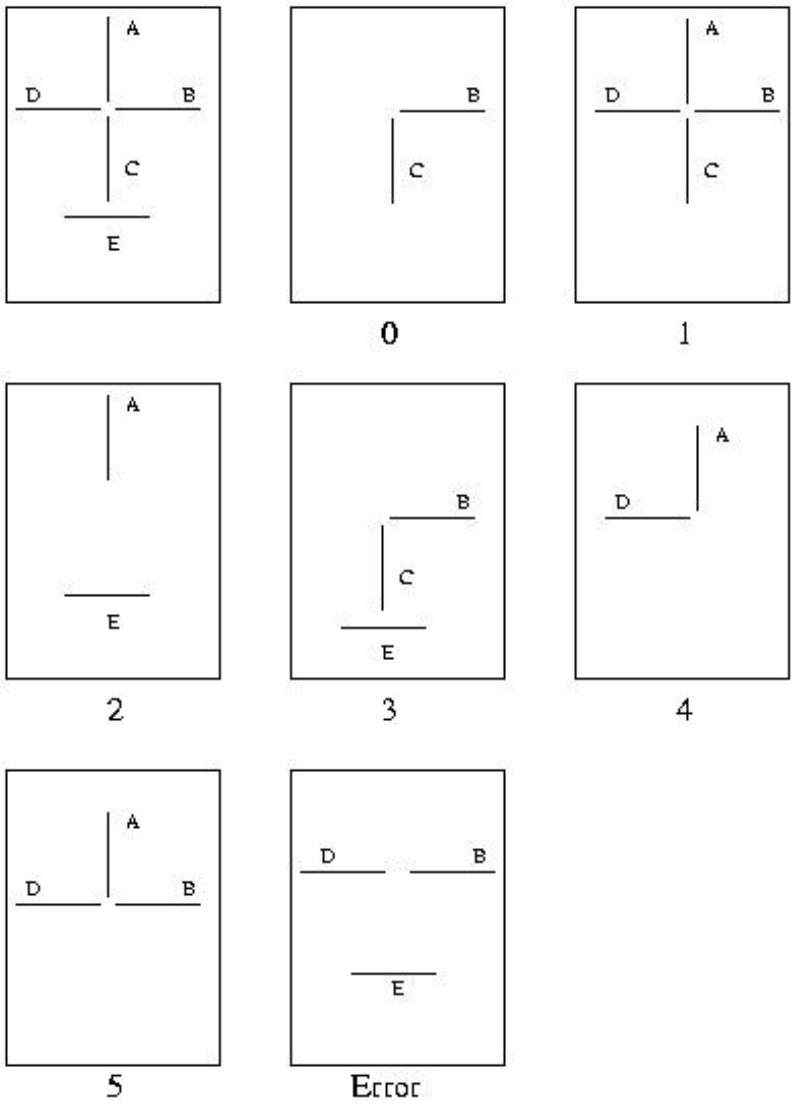
   Be VERY careful to get the correct function for your five inputs before simplifying and designing the circuit with DIGLOG. **You should minimize the circuit.** Your inputs and output should be labeled (in DIGLOG, not by hand).

   When your circuit has been designed using DIGLOG, you should do three things:
   - Hand in the truth tables and any work you used to simplify the circuit.
   - Mail a copy of the final circuit file
   - Hand in a copy of the printed circuit

2. It is the year 2080 and you have been hired by the inhabitants of planet Foobar to create calculators (their technology is a little behind the times). The Foobarians only have two fingers and a thumb on each hand, and therefore, they count in something similar to base 6. Your part of the FBCP (Foobarian Calculator Project) is to design the necessary electronics to display the digits (the Foobarians call them "fidgets") of their number system. In the figure below, you will see the layout your design team has decided to use for the LED display device on the calculators, as well as the symbols that represent the equivalent of our numbers zero through five and the Foobarian symbol for "error."

   Design the necessary circuit using DIGLOG to implement the LED as specified in the figure. Since we only have 6 fidgets to display, and you have decided to use binary numbers to represent values internally, three inputs will suffice. For consistency, please call the inputs X,Y, and Z (with Z representing the rightmost bit in the binary representation of the fidget). Since 3 inputs yields 8 possible fidgets (and the values 6 and 7 as inputs have no meaning), your circuit should display the Foobarian symbol for error (see figure) when those binary values are input into your circuit. Your circuit should have an output for each segment of the LED. You should end up with one large circuit that correctly "lights up" the LED when given the fidget value. You should arrange your output LEDs as the actual FBCP LED display.

   Be VERY careful to get the correct function for your inputs before simplifying and designing the circuit with DIGLOG. **You should minimize the circuit.** You should investigate each output function AND its complement to determine the best combination of functions and terms ("best" defined to be minimal number). Your equations should all be left in "sum of products" form (factoring out terms in the equation results in an additional layer of circuits that requires more time to display the fidgets).



3. Your job for this assignment is to design a controller for Dino Clean Cars, an automated car wash. Users drive up to the car wash and insert combinations of Trex tokens and/or Raptor tokens (the Trex token is worth twice what the Raptor token is worth; the tokens themselves represent a monetary amount that changes as the car wash raises prices). Tokens are purchased at a special machine outside the car wash. If users insert tokens and then press the start button, their cars will be washed, and possibly waxed and dried. The various services available cost the following:

   - **wash only**: 1Trex token OR 2 Raptor tokens
   - **wash and wax**: 1 Trex token and 1 Raptor token OR 3 Raptor tokens
   - **wash, wax and dry**: 2 Trex tokens OR 1 Trex token and 2 Raptor tokens OR 4 Raptor tokens

   The car wash controller has two inputs (call them A and B) to encode the following:

A	B	Description
0	0	Nothing is happening
0	1	Raptor token is inserted
1	0	Trex token is inserted
1	1	Start button is pushed

There will be three outputs: W (for wash), X (for wax) and D (for dry). When all of these are 0, the car wash will do nothing. When the correct number of tokens has been inserted AND the start button has been pressed, the car wash will perform the desired action. If more tokens are inserted than are necessary, the car wash will NOT give back the extras (so, for example, if someone puts in 5 Raptor tokens and then presses the START button, this person will receive a wash, wax and dry, but won't get back the additional token). Since we only have 2 inputs (that recognize Raptor and Trex tokens), no other tokens are recognized (no actual coins or bills either).

Design this using D flip-flops. You will need to look at the state for each flip-flop. Let T represent the state that a Trex token has been inserted and R represent the state that a Raptor token has been inserted. Then there are really 4 inputs we need to consider: A, B, T and R (we must worry about the past state to determine the future state). We therefore end up with the following partial table (where D_T is the input of the D flip-flop for Trex tokens, D_R is the input of the D flip-flop for Raptor tokens, T is the output of the Trex token D flip-flop, and R is the output of the Raptor token D flip-flop:

INPUTS	OUTPUTS	INPUTS FOR D FLIP-FLOPS	
A B T R	W X D	D_T D_R	What Happens
0 0 0 0	0 0 0	0 0	(nothing is happening)
0 0 0 1	0 0 0	0 1	(Raptor token previously inserted, nothing happening now)
0 0 1 0	0 0 0	1 0	(Trex token previously inserted, nothing happening now)
...			
0 1 0 0	0 0 0	0 1	(no tokens previously inserted, user inserts Raptor token, D_R now stores this fact)
...			
0 1 0 1	0 0 0	1 0	(Raptor token previously inserted, user inserts another Raptor token, D_T now stores this fact)
0 1 1 0	0 0 0	1 1	(Trex token previously inserted, user inserts Raptor token, D_T and D_R both store this info
...			
1 1 1 0	1 0 0	0 0	(Trex token previously inserted, user presses start button, car will be washed, D flip flops reset)

You will need to add a 3rd D flip-flop to represent the state that 2 Trex tokens (or 4 Raptor tokens) have been inserted. Call this D flip-flop D_{2T}. When you are done filling in the table, you will need to find the resulting equations for each of the necessary values (W, X, D, D_T, D_R and D_{2T}). When using D flip-flops, please be sure to hook the set (top line) of the flip flop to V_{dd} and the reset (bottom line) to a switch (so they can be reset easily). For the clock to each flip-flop, use a pulse switch (so the clock can be pulsed to test your circuits). Use the DNEG or JKNEG components. W, X and D should be expressed in terms of A, B, T and R. To allow the clock to be pulsed so changes can be observed, please use flip-flops for W, X and D as well.

If you need OR or AND gates with more than 2 or 3 inputs, remember they are available in larger sizes. If, for example, you can find 5 input gate and an 8 input gate, but nothing in between, use the larger one and connect the inputs you don't need to the appropriate value of 0 or 1 (ground or V_{dd}).