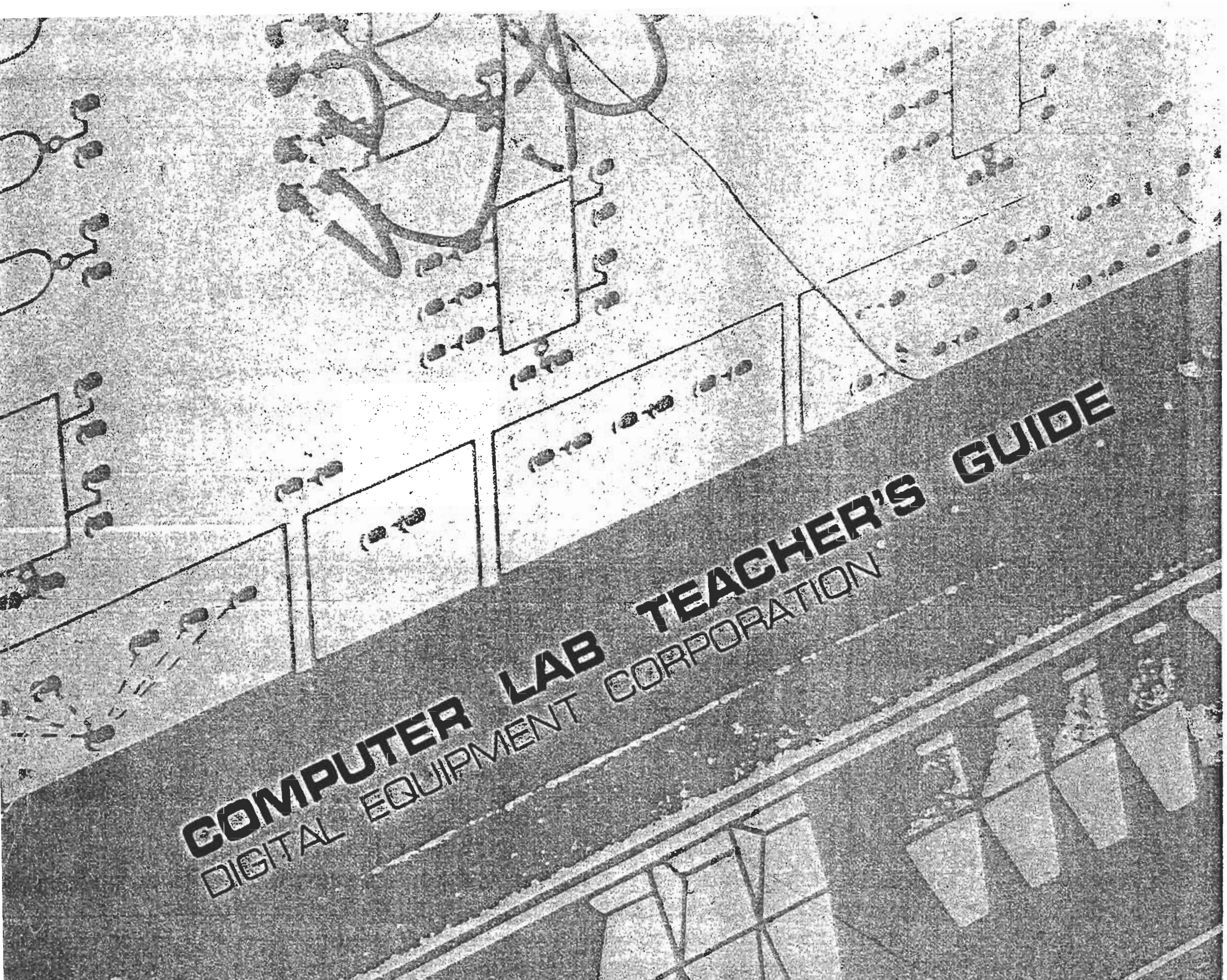
COMPUTER LAB TEACHER'S GUIDE

DIGITAL EQUIPMENT CORPORATION



B-400

COMPUTER LAB TEACHER'S GUIDE

DIGITAL EQUIPMENT CORPORATION ▫ MAYNARD, MASSACHUSETTS

This edition of the COMPUTER LAB Teacher's Guide is intended to be used with the COMPUTER LAB Workbook, 1st Edition.

FOREWORD

This Teacher's Guide accompanies the Digital Equipment Corporation COMPUTER LAB Workbook and provides the instructor with complete answers to all the Workbook questions. Each answer is discussed in detail and the basic logic principles involved are fully explained. The Guide includes a discussion of the intent and educational approach of each chapter, an outline of the major subjects covered by each chapter and a complete experiment list.

Supplemental technical information, which the instructor can use as optional course material or as additional background information for himself, is included on many of the subjects. The Guide also includes course plans and suggested examinations for students at the Computer Science and Computer Technology levels and a Workbook-textbook cross-reference list.

My appreciation goes to all those who have helped in the writing, preparation and revision of this Guide. In particular, I would like to acknowledge the assistance given by: Mr. John Hughes, author of the COMPUTER LAB Workbook, for his technical assistance; Mr. Edward Converse for his editorial review; and Mrs. Leslie Pavelka for her editorial comments and technical contribution.

Lawrence F. DeAngelo
Training Department

CONTENTS

CHAPTER GUIDE

CONTENTS (Cont)

CONTENTS (Cont)

CHAPTER 1

THE BINARY CONCEPTS

INTENT AND APPROACH

The objectives of this chapter are as follows:

a. to show how two-state devices can be used to implement logical functions;

b. to give the student a thorough understanding of the following logic functions and gates: AND, OR, Inversion, Negated Input OR, and NAND;

c. to give the student adequate knowledge of the COMPUTER LAB to accomplish the experiments in this chapter;

d. to give the students a thorough understanding of binary numbers and decimal-binary conversions. Binary fractions and octal numbers may be introduced at the teacher's discretion. Supplementary information is presented on pages 4 to 13.

These objectives are accomplished by showing the correlation of basic two-state logic with practical situations; developing the student's repertoire of logical functions; and having him become familiar with the implementation of these functions on the COMPUTER LAB.

Once the student understands basic logic functions, he uses these simple functions to construct more complex devices. By constructing such devices as a Binary to Decimal Converter and a Decimal to Binary Converter, the student not only reinforces his knowledge of the COMPUTER LAB gates, but also exercises a greater understanding of binary numbers and conversions.

OUTLINE

I. INTRODUCTION (Pages 1-2)

- A. Two-State Devices
  - 1. Memory
  - 2. No memory
- B. Logical Decisions
  - 1. Yes or No conditions
  - 2. Symbolic representation
  - 3. Truth table

II. GATES (Pages 2-5)

- A. OR Gate
  - 1. Symbol (military standard 806B)
  - 2. Truth table

SUPPLEMENTARY INFORMATION

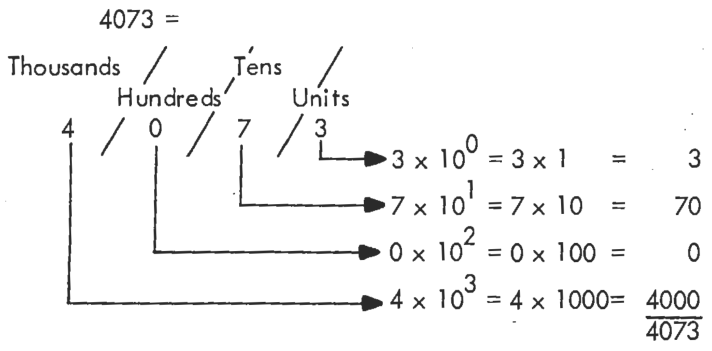
I. NUMBER SYSTEMS

Basic Concepts

A number is assigned a value which depends on its numerical symbol and its position in the whole number. The decimal number system uses ten symbols representing the quantities 0 through 9. Other numbers are constructed by assigning different values (or weights) to the position of the symbol relative to the decimal point. For example, the number 008. represents eight units, while the number 080. represents a quantity of eighty units, and the number 800. represents a quantity of eight hundred units.

The base or radix of a number system is the amount of symbols or numbers a system possesses including zero. A base 8 system has eight digits, 0-7. A base 2 system has two digits, 0 and 1.

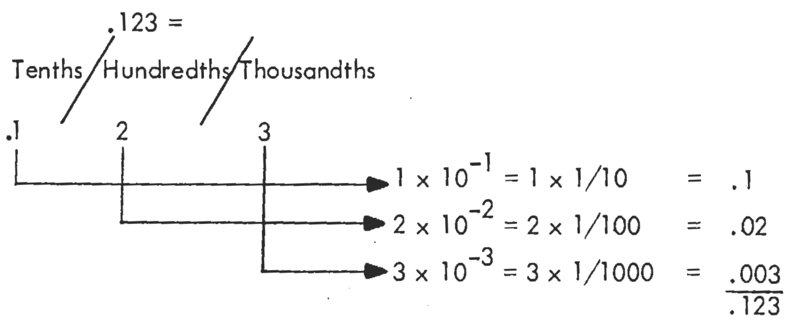
The following diagram illustrates the integral part of the decimal system's positional notation.



4073 =
Thousands
Hundreds
Tens
Units
4 / 0 / 7 / 3
$3 \times 10^0 = 3 \times 1 = 3$
$7 \times 10^1 = 7 \times 10 = 70$
$0 \times 10^2 = 0 \times 100 = 0$
$4 \times 10^3 = 4 \times 1000 = 4000$
4073

The value of the decimal number is equal to the sum of the products.

The following diagram illustrates the fractional part of the decimal system's positional notation.



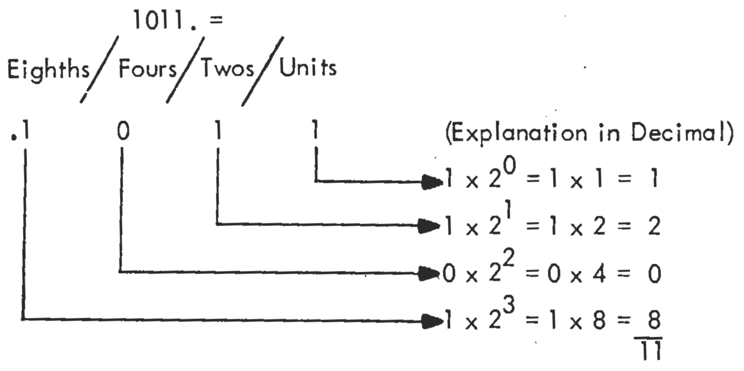
.123 =
Tenths
Hundredths
Thousandths
.1 2 3
$1 \times 10^{-1} = 1 \times 1/10 = .1$
$2 \times 10^{-2} = 2 \times 1/100 = .02$
$3 \times 10^{-3} = 3 \times 1/1000 = .003$
.123

The value of the decimal number is equal to the sum of the products.

Binary Numbers

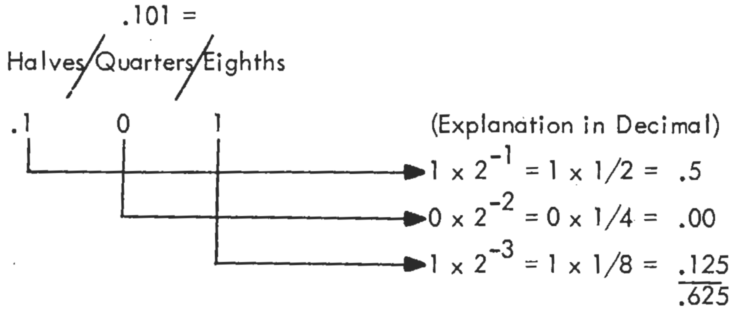
The Binary number system uses two symbols, 0 and 1, and its positional weights are powers of 2.

The following diagram illustrates the integral part of the binary system's positional notation.


1011. =
Eighths/ Fours/ Twos/ Units
.1 0 1 1
(Explanation in Decimal)
$1 \times 2^0 = 1 \times 1 = 1$
$1 \times 2^1 = 1 \times 2 = 2$
$0 \times 2^2 = 0 \times 4 = 0$
$1 \times 2^3 = 1 \times 8 = 8$
11

The decimal equivalent of binary 1011 is 11.

The following diagram illustrates the fractional part of the binary system's positional notation.


.101 =
Halves/Quarters/Eighths
.1 0 1
(Explanation in Decimal)
$1 \times 2^{-1} = 1 \times 1/2 = .5$
$0 \times 2^{-2} = 0 \times 1/4 = .00$
$1 \times 2^{-3} = 1 \times 1/8 = .125$
.625

The decimal equivalent of the binary number .101 is .625.

Binary To Decimal Conversion

Binary positional weights, like decimal positional weights, can be extended as far as desired. Therefore, using the information illustrated above, any binary number can be converted to decimal by multiplying each digit by its positional coefficient, and then adding all the products.

The following example illustrates binary to decimal conversion.

Binary	Decimal
110110.1011 =	54.6875

Integral Part Conversion

$1\ 1\ 0\ 1\ 1\ 0_{(2)}$ ← (Base 2)

$$\begin{aligned}
0 \times 2^0 &= 0 \times 1 &= 0\\
1 \times 2^1 &= 1 \times 2 &= 2\\
1 \times 2^2 &= 1 \times 4 &= 4\\
0 \times 2^3 &= 0 \times 8 &= 0\\
1 \times 2^4 &= 1 \times 16 &= 16\\
1 \times 2^5 &= 1 \times 32 &= \underline{32}\\
&& 54_{(10)}
\end{aligned}$$

← (Base 10)

Fractional Part Conversion

$.1\ 0\ 1\ 1_{(2)}$

$$\begin{aligned}
1 \times 2^{-1} &= 1 \times 1/2 &= .5\\
0 \times 2^{-2} &= 0 \times 1/4 &= .00\\
1 \times 2^{-3} &= 1 \times 1/8 &= .125\\
1 \times 2^{-4} &= 1 \times 1/16 &= \underline{.0625}\\
&& .6875_{(10)}
\end{aligned}$$

Decimal to Binary Conversion - Integers

Subtraction of Powers Method - To convert any decimal whole number to binary, subtract the highest possible power of 2 from the decimal number and place a 1 under the appropriate binary number. Repeat this subtraction process until the decimal number is reduced to 0. If, after the first subtraction, the next lower power of 2 cannot be subtracted, place a 0 in the appropriate weighting position.

The following example illustrates whole number decimal to binary number conversion using the subtraction of powers method.

$39_{(10)} = ?\ \text{binary}_{(2)}$

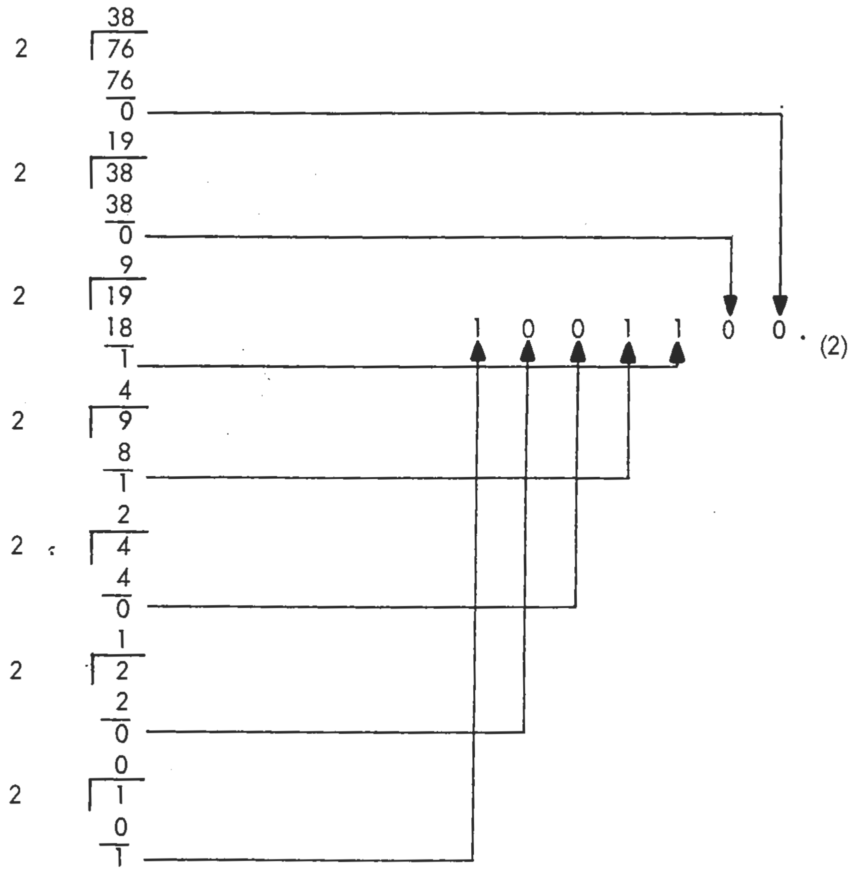
$$\begin{array}{r} 39\\ \underline{-32}\\ 7 \end{array} \quad \begin{array}{r} 7\\ \underline{-4}\\ 3 \end{array} \quad \begin{array}{r} 3\\ \underline{-2}\\ 1 \end{array} \quad \begin{array}{r} 1\\ \underline{-1}\\ 0 \end{array}$$

2^5	2^4	2^3	2^2	2^1	2^0	Power of 2
32	16	8	4	2	1	Positional Value
1	0	0	1	1	1	Binary Number

Division Method - To convert any decimal whole number to binary, divide the decimal number by 2. If there is a remainder, the least significant digit of the partially formed binary number is a 1. If the remainder is a 0, place a 0 in the least significant digit of the binary number. Divide the quotient from the first division by 2 and repeat this process until the quotient has been reduced to 0. If there is a remainder, record a 1; if there is no remainder, record a 0.

The following example illustrates whole number decimal to binary conversion using the division method.

$76_{(10)} = ?\ \text{Binary}_{(2)}$



Decimal to Binary Conversion - Fractional Numbers

Subtraction of Powers Method- To convert a decimal fraction to a binary number, subtract the highest possible negative power of 2 that is contained in the decimal fraction. After each subtraction, record a 1 under the power of 2 subtracted; when no subtraction is possible record a 0. This process gives the equivalent number in binary.

The following example illustrates conversion of a decimal fraction to binary.

$$.750 = ? \text{ Binary}$$

$$\begin{array}{r} .750 \\ \underline{.500} \\ .250 \end{array} \qquad\qquad \begin{array}{r} .250 \\ \underline{.250} \\ .000 \end{array}$$

2^{-1}	2^{-2}	2^{-3}	2^{-4}	Power of 2
.5	.25	.125	.0625	Positional Value
.1	1	0	0	Binary Number

The largest negative power of 2 contained in the decimal fraction 0.750 is 2^{-1}, which is equivalent to the decimal 0.5. Subtract 0.5 from 0.750 and record a 1 in the 2^{-1} column. Subtract 0.25 from 0.250 and record a 1 under the 2^{-2} column. 2^{-2} is equal to 0.25. Since no further subtraction can be performed, all other columns are filled with 0s.

<u>Successive Multiplication Method</u> – A decimal fraction may be converted to binary by using successive multiplication by 2. Double the decimal number. If the product is greater than 1, record a 1 and delete the integral part of the product; if the product is less than 1, record a 0. Write the digits of the binary number from left to right, away from the binary point. Continue multiplication by 2 until the fractional product is 0 or the desired length of the binary number is attained.

$$.6875_{(10)} = ? \text{ Binary}$$

$$\begin{array}{r} .6875 \\ \underline{2} \\ 1.3750 \end{array} \qquad \begin{array}{r} 0.3750 \\ \underline{2} \\ 0.7500 \end{array} \qquad \begin{array}{r} .7500 \\ \underline{2} \\ 1.5000 \end{array} \qquad \begin{array}{r} 0.5000 \\ \underline{2} \\ 1.0000 \end{array} \qquad .0000$$

$$.1\ 0\ 1\ 1_{(2)}$$

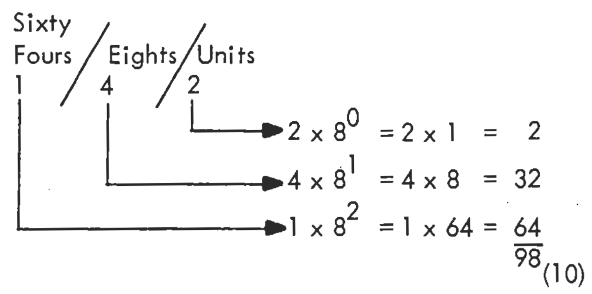
Octal Numbers

It is evident that the binary number system, although appropriate for digital computers, is cumbersome for human usage. The octal number system helps bridge the gap between the computer's base 2 language and our base 10 language. The base 8 or octal number system utilizes the digits 0-7 in forming numbers.

Position Notation - The following illustrates the position notation system of a base 8 system:

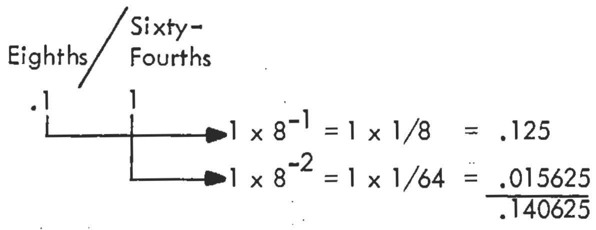
$$142.11_{(8)} = 98.140625_{(10)}$$

Integer Part



$$\begin{aligned} 2 \times 8^0 &= 2 \times 1 = 2 \\ 4 \times 8^1 &= 4 \times 8 = 32 \\ 1 \times 8^2 &= 1 \times 64 = \underline{64} \\ & \quad 98_{(10)} \end{aligned}$$

Fractional Part



$$\begin{aligned} 1 \times 8^{-1} &= 1 \times 1/8 = .125 \\ 1 \times 8^{-2} &= 1 \times 1/64 = \underline{.015625} \\ & \quad .140625 \end{aligned}$$

Conversion

Binary to Octal - The ease in converting from binary to octal and the fact that octal numbers can be easily handled make octal a convenient intermediate number system.

In the octal number system, any number between 0 and 7 can be represented by three binary digits as shown below.

Octal	Binary	Octal	Binary	Octal	Binary
0	000	3	011	6	110
1	001	4	100	7	111
2	010	5	101		

We can represent any binary number as an octal number by starting at the least significant digit end and grouping the binary number into groups of three bits, and then converting these 3-bit groups into octal notation.

001	010	011	111	101 .	(2)
1	2	3	7	5 .	(8)

The conversion from octal is the reverse of the above procedure; convert the octal digits into a 3-bit binary number.

<u>Decimal to Octal Conversion</u> - Decimal numbers can be converted into octal numbers in the same manner as decimals are converted into binary. The following examples illustrate decimal to octal conversion using all methods illustrated previously.

a. Subtraction of Powers - Integers

To convert a decimal whole number to its octal equivalent, subtract the highest possible power of 8 from the decimal number (it can be a multiple from 1 to 7 times the power). Record the number obtained under the appropriate power of 8. Repeat this procedure until the decimal number has been reduced to 0.

$$1090_{(10)} = \text{? Octal}_{(8)}$$

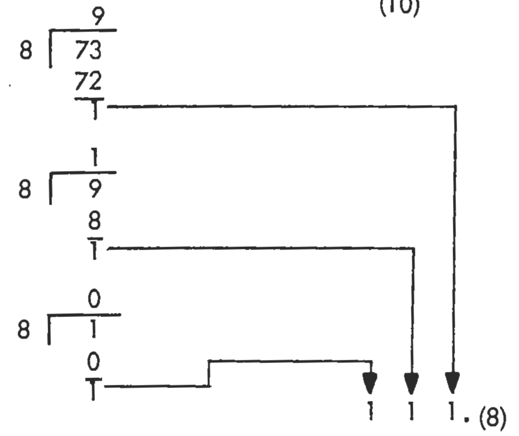
$$\begin{array}{r} 1090 \\ \underline{-1024} \\ 66 \end{array} = 2 \times 8^3 \qquad \begin{array}{r} 66 \\ \underline{-64} \\ 2 \end{array} = 1 \times 8^2 \qquad \begin{array}{r} 2 \\ \underline{-2} \\ 0 \end{array} = 2 \times 8^0$$

8^3	8^2	8^1	8^0	Power of 8
512	64	8	1	Positional Value
2	1	0	2	Octal Number

b. Division Method - Integers

To convert a decimal number to its octal equivalent by the division method, divide the decimal number by 8 and write the remainder down as the least significant digit of the octal number. Continue dividing the quotients by 8 and recording the remainders until the quotient has been reduced to 0. The first remainder will be the octal number's least significant digit, and each successive remainder will be more significant.

$$73_{(10)} = \text{? Octal}$$



c. Subtraction of Powers - Fractions

To convert a decimal number fraction to an octal fraction, subtract the highest possible negative power of 8 from the decimal number. In each subtraction, record the coefficient under the appropriate power of 8. When no subtraction is possible, a 0 is recorded.

$$.265625_{(10)} = ?\ \text{Octal} \qquad \begin{array}{rl} .265625 & \\ -\underline{.250} & = 2 \times 8^{-1} \\ .015625 & \\ -\underline{.015625} & = 1 \times 8^{-2} \\ .000000 & \end{array}$$

8^{-1}	8^{-2}	8^{-3}	Power of 8
.125	.015625	.001953125	Positional Value
.2	1	0	Octal Number

d. Successive Multiplication Method - Fractions

To convert a decimal number fraction to an octal fraction, successive multiplication by 8 may be used. The decimal fraction is multiplied by 8 and the whole number part of the fraction is extracted to become part of the octal number; if no whole number is obtained, a 0 is placed in the next octal number position. The octal fraction is formed starting from the octal point and proceeding to the right. Continue the multiplication process until the desired number length is obtained or until the product is 0.

$$.0625_{(10)} = ?\ \text{Octal}_{(8)}$$

$$\begin{array}{r} .0625 \\ \underline{8} \\ 0.5000 \end{array} \qquad \qquad \begin{array}{r} .5000 \\ \underline{8} \\ 4.0000 \end{array}$$

$$.0\ 4_{(8)}$$

<u>Computer Representation of Fractional Numbers</u>

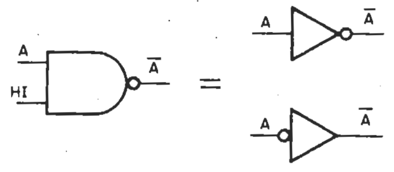
The COMPUTER LAB Workbook does not present material on binary fractions because, in many computers, the position of the binary point in the computer storage devices is not a function of the computer logic but a programming consideration. The programmer can consider the binary point to be present at any position in the binary number. The computer logic is not affected by the programmer's choice. However, material on binary fractions may be presented in order to complete the study of the binary number system.

II. INVERTING LOGIC ADVANTAGES

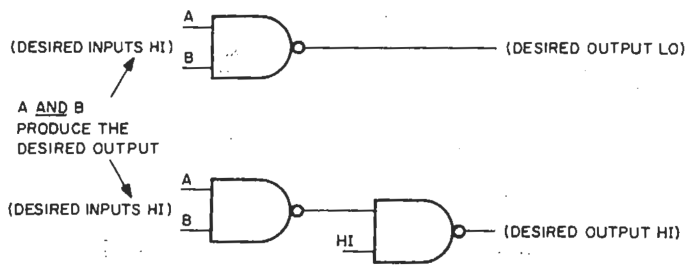
Inverting logic (NAND/Negated Input OR logic) is usually preferred over non-inverting AND/OR logic for the following reasons:

a. An inverting logic element such as the COMPUTER LAB NAND gate is more versatile than a non-inverted AND or OR gate. An inverting logic element can be used to perform all logic functions. An example of the "universal" aspect of inverting logic is demonstrated below using the COMPUTER LAB NAND gate.

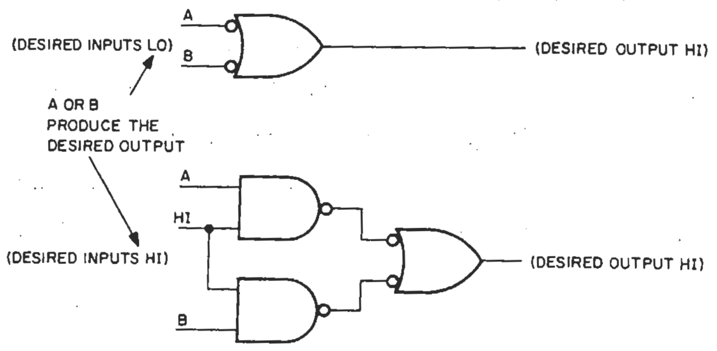
INVERSION



AND FUNCTION

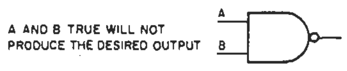


OR FUNCTION



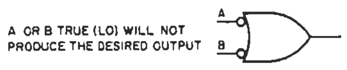
NAND FUNCTION

In the following logic diagram, a HI is considered the true or desired level at the input and the output.



NOR FUNCTION

In the following logic diagram, a LO is considered the true or desired level at both the input and the output.



b. An inverting logic element has an electronic advantage over a noninverting logic element. A noninverting logic element does not usually have amplification in each gate. Inverting logic has amplification in each gate and loading problems are simplified.

ANSWER KEY

Experiment 1.1

A	B	LAMP INDICATOR
LO	LO	HI
LO	HI	HI
HI	LO	HI
HI	HI	LO

Figure 1.20 NAND Gate Truth Table

Experiment 1.2

A	B	C	OUTPUT
LO	LO	LO	HI
LO	LO	HI	HI
LO	HI	LO	HI
LO	HI	HI	HI
HI	LO	LO	HI
HI	LO	HI	HI
HI	HI	LO	HI
HI	HI	HI	LO

Figure 1.22 3-Input NAND Gate Truth Table

Experiment 1.3

A	B	C	D	OUTPUT
LO	LO	LO	LO	HI
LO	LO	LO	HI	HI
LO	LO	HI	LO	HI
LO	LO	HI	HI	HI
LO	HI	LO	LO	HI
LO	HI	LO	HI	HI
LO	HI	HI	LO	HI
LO	HI	HI	HI	HI
HI	LO	LO	LO	HI
HI	LO	LO	HI	HI
HI	LO	HI	LO	HI
HI	LO	HI	HI	HI
HI	HI	LO	LO	HI
HI	HI	LO	HI	HI
HI	HI	HI	LO	HI
HI	HI	HI	HI	LO

Figure 1.24 4-Input NAND Gate Truth Table

Experiment 1.4

INPUT	OUTPUT
LO	HI
HI	LO

Figure 1.27 Inverter Truth Table

Experiment 1.5

A	B	C	OUTPUT
LO	LO	LO	HI
LO	LO	HI	HI
LO	HI	LO	HI
LO	HI	HI	HI
HI	LO	LO	HI
HI	LO	HI	HI
HI	HI	LO	HI
HI	HI	HI	LO

Figure 1.29 Truth Table for 4-Input NAND Gate Used as 3-Input NAND Gate

Questions:

1.a.

INPUTS		OUTPUT
A	B	
HI	LO	HI
HI	HI	LO

2-Input NAND Gate Used as an Inverter

INPUTS		OUTPUT
A	B	
LO	LO	HI
LO	HI	HI
HI	LO	HI
HI	HI	LO

Original truth table (Figure 1.20)

1.b. The first two conditions on the initial truth table (Figure 1.20) cannot occur because input A is connected to a constant HI.

2. If input A were connected to HI, the first eight conditions could not occur.

3. A 3-input NAND gate can be used as an inverter by connecting two inputs together and then connecting one of the two to a HI.

4. By connecting two inputs together or by connecting one input to a HI.

5. The truth table for both the NAND and Negated Input OR are identical.

6.

A.	150	F.	286
B.	51	G.	153
C.	247	H.	170
D.	65	I.	438
E.	8	J.	219

7.

A.	10010100	F.	111110100000
B.	100010101	G.	110001
C.	110101	H.	1101101011
D.	100000000	I.	1011110
E.	1000000000	J.	1110101

8. Recorded indications and Figure 1.30 binary count sequence should be identical. Light on indicates a binary 1. Light out indicates a binary 0.

9.a. A LO level.

9.b. The Decimal to Binary Encoder requires an OR function. The COMPUTER LAB uses NAND Gates (Negated Input OR). These gates require a LO input to be used as an OR function. Therefore a switch to be encoded must be in the LO position.

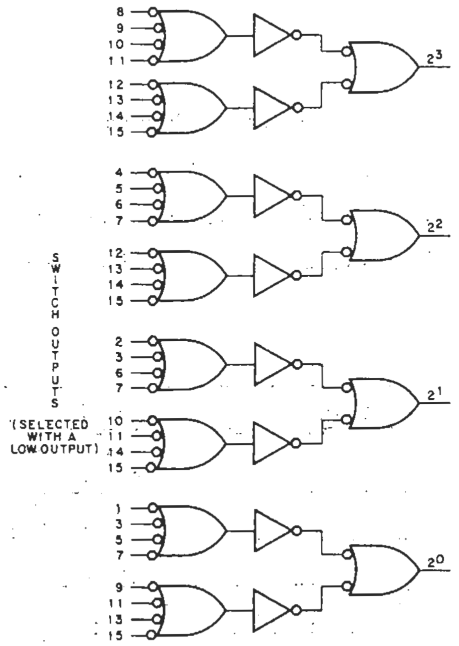
10.a. HI

10.b. Only one decimal input can be decoded at one time, so all inputs other than the one being decoded must be disabled or HI.

11. Gate 2^2 produces a HI output when decimal number 4 or 5 or 6 or 7 is selected. Gate 2^1 produces a HI output when decimal number 2 or 3 or 6 or 7 is selected. Gate 2^0 produces a HI output when decimal number 1 or 3 or 5 or 7 is selected.

12.

2^3	2^2	2^1	2^0	DECIMAL
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15



Decimal to Binary Encoder, Numbers 0-15

Experiment 1.7

2^2	2^1	2^0	$\overline{2^2}$	$\overline{2^1}$	$\overline{2^0}$	Decimal
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	2
0	1	1	1	0	0	3
1	0	0	0	1	1	4
1	0	1	0	1	0	5
1	1	0	0	0	1	6
1	1	1	0	0	0	7

Figure 1.35 Binary to Decimal Decoder Operating Truth Table

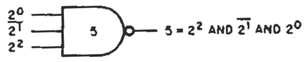
Questions

13.

Each gate in the Binary to Decimal Decoder is searching for a unique binary number. When the gate detects that number with 3 HI inputs, its output will indicate that the number is present with a low level. The following truth table shows the conditions required to enable each gate. The inverters at the bottom of Figure 1.33 are used to provide a high signal when the corresponding bit in a binary number is 0.

Gate	True HI Input Conditions	Output
0	$\overline{2^0}$ and $\overline{2^1}$ and $\overline{2^2}$	LO
1	2^0 and $\overline{2^1}$ and $\overline{2^2}$	LO
2	$\overline{2^0}$ and 2^1 and $\overline{2^2}$	LO
3	2^0 and 2^1 and $\overline{2^2}$	LO
4	$\overline{2^0}$ and $\overline{2^1}$ and 2^2	LO
5	2^0 and $\overline{2^1}$ and 2^2	LO
6	$\overline{2^0}$ and 2^1 and 2^2	LO
7	2^0 and 2^1 and 2^2	LO

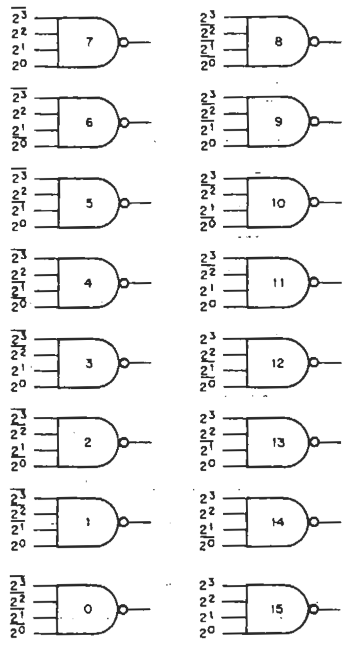
14.



Decode – Binary Number 101 into Decimal Number 5

15.

2^3	2^2	2^1	2^0	DECIMAL
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15


$\overline{2^3}$ 2^2 2^1 2^0 7
2^3 $\overline{2^2}$ $\overline{2^1}$ $\overline{2^0}$ 8
$\overline{2^3}$ 2^2 2^1 $\overline{2^0}$ 6
2^3 $\overline{2^2}$ $\overline{2^1}$ 2^0 9
$\overline{2^3}$ 2^2 $\overline{2^1}$ 2^0 5
2^3 $\overline{2^2}$ 2^1 $\overline{2^0}$ 10
$\overline{2^3}$ 2^2 $\overline{2^1}$ $\overline{2^0}$ 4
2^3 $\overline{2^2}$ 2^1 2^0 11
$\overline{2^3}$ $\overline{2^2}$ 2^1 2^0 3
2^3 2^2 $\overline{2^1}$ $\overline{2^0}$ 12
$\overline{2^3}$ $\overline{2^2}$ 2^1 $\overline{2^0}$ 2
2^3 2^2 $\overline{2^1}$ 2^0 13
$\overline{2^3}$ $\overline{2^2}$ $\overline{2^1}$ 2^0 1
2^3 2^2 2^1 $\overline{2^0}$ 14
$\overline{2^3}$ $\overline{2^2}$ $\overline{2^1}$ $\overline{2^0}$ 0
2^3 2^2 2^1 2^0 15

NOTE:
BINARY NUMBER SELECTED WITH SWITCH OUTPUT HI. DECIMAL OUTPUT INDICATED BY LIGHT OUT.

Decode 0000 → 1111 to Decimal

16. The COMPUTER LAB electronic devices are inherently two state devices with two logic levels LO and HI which can be used to represent a base 2 digit.

17.a.

Device 1	Device 2	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
LO	LO	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
LO	HI	1	1	2	2	3	3	0	0	2	2	3	3	0	0	1	1	3	3	0	0	1	1	2	2
HI	LO	2	3	1	3	1	2	2	3	0	3	2	0	3	1	0	3	0	1	2	1	2	0	1	0
HI	HI	3	2	3	1	2	1	3	2	3	0	0	2	1	3	3	0	1	0	1	2	0	2	0	1

17.b. 0123 - Binary.

17.c. It follows the same organization as the decimal system.

17.d.

Device 1	Device 2	Device 3	Device 4
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	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

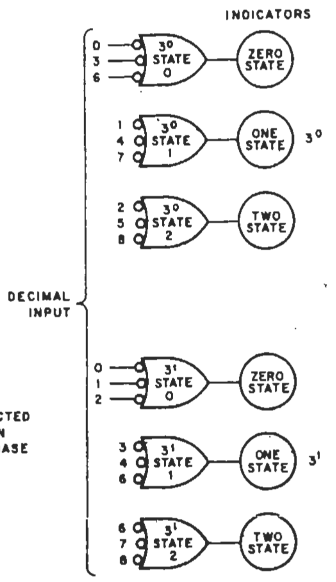
18.

3^1	3^0	Decimal
0	0	0
0	1	1
0	2	2
1	0	3
1	1	4
1	2	5
2	0	6
2	1	7
2	2	8

19.

3^1	3^0	DECIMAL
0	0	0
0	1	1
0	2	2
1	0	3
1	1	4
1	2	5
2	0	6
2	1	7
2	2	8

NOTE:
DECIMAL NUMBER SELECTED WITH A LOW. LIGHT ON INDICATES SELECTED BASE THREE NUMBER.



Decimal to Base 3 Encoder Numbers 0-8

20. All functions, inputs, and outputs on the COMPUTER LAB are two state; therefore, to use the COMPUTER LAB to represent number systems with bases larger than two requires more logic functions.

21.a. A Negated Input AND.

21.b. NOR and Negated Input AND have identical truth tables.

CHAPTER 2
BASIC LOGIC GATES

INTENT AND APPROACH

The objectives of this chapter are as follows:

a. to complete the student's knowledge of the basic rules necessary to operate the COMPUTER LAB;

b. to teach the operation and applications of the AND/NOR gate;

c. to demonstrate the techniques of developing non-inverting, functions from inverting logic gates.

Since the COMPUTER LAB is an electronic device, a certain amount of electronic knowledge is required in order to operate the device. Chapter 2 provides this electronic information in a brief and simple manner. For courses that are electronically oriented, this area can be expanded easily by referring to Appendix C in the Workbook.

The student should understand the operation of the AND/NOR gate after performing Experiment 2.1. Practical applications of the AND/NOR gate are illustrated by presenting the NOR gate, Exclusive OR Gate, and Comparator logic configurations. Later in the chapter these logic configurations of NOR, Exclusive OR, and comparison are combined to demonstrate more complex logical functions such as an Equality Detector and a Parity Bit Generator.

In order to fully understand the equality detector logic, an explanation of inverting to non-inverting gate conversion is included in Chapter 2. In explaining the Equality Detector, the teacher should point out that the output configuration of Figure 2.15 (4-Bit Equality Detector), consisting of a NAND followed by inversion, is an AND function.

OUTLINE

I. INTRODUCTION (Page 23)

II. BASIC RULES FOR USING COMPUTER LAB FUNCTIONS (page 23-24)

   A. Levels

      1. Definition
      2. HI
      3. LO
      4. COMPUTER LAB source

   B. Pulses

      1. Definition
      2. HI pulse
      3. LO pulse

4. Source of pulses on COMPUTER LAB
   a. Pulser switch
   b. Clock
5. Symbolic representation

C. Input Loading
   1. Definition of one unit load
   2. Load units for COMPUTER LAB functions
   3. Determining total input load

D. Output Drive
   1. Definition of fan-out
   2. COMPUTER LAB gate fan-out

E. Output Connections
   1. Outputs to inputs
   2. Outputs to outputs

F. The Clock
   1. Frequency ranges
   2. Frequency adjustments

G. Ground Outputs
   1. Location on COMPUTER LAB
   2. Purpose

III. THE AND/NOR GATE (Pages 24-32)

A. Use of the AND/NOR Gate

B. Symbol

C. Truth table

D. Experiment 2.1: AND/NOR Gate (Page 25)
   1. Circuit construction
   2. Verification of operation

E. Experiment 2.2: NOR Gate Application of AND/NOR Gate (Pages 25-26)
   1. Circuit construction
   2. Truth table completion
   3. Logic diagram
   4. NOR Gate symbol
   5. Negated Input AND Gate symbol

F. Questions 1-2 (Page 26)

G. Experiment 2.3: AND/NOR Comparator (Pages 27-28)
   1. Use
   2. Circuit construction
   3. Verification of operation
   4. Logic diagram
   5. Truth table

H. Question 3 (Page 27)

I. Experiment 2.4: AND/NOR Gate Used for Exclusive OR (Page 27)
   1. Operation of Exclusive OR Gate
   2. Truth table completion
   3. Logic diagram
   4. Circuit construction
   5. Verification of operation

J. Question 4 (Page 28)

K. Experiment 2.5: Non-Inverting Gates (Pages 28-29)
   1. Circuit construction
   2. Construction of truth table
   3. Comparison of truth table with AND Gate truth table
   4. Logic diagram
   5. AND Gate truth table

L. Questions 5-9 (Page 29)

M. Experiment 2.6: The Equality Detector (Pages 29-30)
   1. Logic diagram
   2. Switch usage for Equality Detector
   3. Circuit construction
   4. Verification of operation

N. Questions 10-11 (Page 31)

O. Experiment 2.7: Parity Bit Generator (Page 31)
   1. Use of parity bit
   2. Practical application
   3. Truth table completion
   4. Circuit construction
   5. Verification of operation
   6. Logic diagram (Page 32)

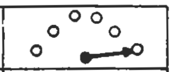
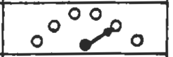
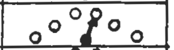
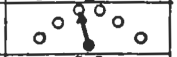
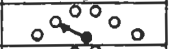
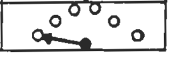
P. Questions 12-13 (Page 32)

Q. Supplementary Questions 14-17 (Page 32)

SUPPLEMENTARY INFORMATION

COMPUTER LAB Clock

The COMPUTER LAB clock has several available frequency ranges. The following chart illustrates the approximate pulse repetition rates of the COMPUTER LAB clock.

On-Off Switch (Fully Counterclockwise)	3/4 Clock-wise turn	1/2 Clock-wise turn	1/4 Clock-wise turn	Fully Clockwise	Coarse Frequency Control
5.5 µs	2.3 µs	.88 µs	.56 µs	.31 µs	
49 µs	21 µs	8 µs	4.1 µs	1.92 µs	
1.06 ms	.52 ms	148 µs	80 µs	42 µs	
25 ms	10 ms	3.51 ms	1.96 ms	.96 ms	
.62 s	.3 s	90 ms	50 ms	25 ms	
1.25 s	.6 s	170 ms	82 ms	44 ms	

Free Run Frequency 10 mc

µs = microseconds
ms = milliseconds

ANSWER KEY

Experiment 2.2

A	B	OUTPUT
LO	LO	HI
LO	HI	LO
HI	LO	LO
HI	HI	LO

Figure 2.6 NOR Truth Table

(Equal to Diagram, Question 1a, Page 27)

Questions

1.a.

A	B	OUTPUT
LO	LO	HI
LO	HI	LO
HI	LO	LO
HI	HI	LO

1.b. The negated input AND gate and the NOR gate are different interpretations or applications of the same electronic element. In the application of the negated input AND gate, the symbol tells us that two LO inputs are required in order to obtain a HI out. In the application of the NOR gate, the symbol tells us that any HI input yields a LO output. Both applications are performed by the same electronic device.

2.

A	B	C	OUTPUT
LO	LO	LO	HI
LO	LO	HI	LO
LO	HI	LO	HI
LO	HI	HI	LO
HI	LO	LO	HI
HI	LO	HI	LO
HI	HI	LO	LO
HI	HI	HI	LO

3. Refer to the diagrams below.

Gate E is searching for the condition where A is HI and B is LO. When this condition exists Point 1 will go HI. This condition will cause gate G to be enabled and the output (Point 3) will go LO, indicating that A and B are not the same. Gate F looks for the other condition where A and B are not the same, namely when A is LO and B is HI. If this condition exists, gate F will be enabled and Point 2 will go HI. This situation will cause gate G to be enabled and Point 3, or the output, will go LO. If A or B are not different, equality is indicated and gates E, F, and G will not be enabled and Point 3 will be HI.

3.

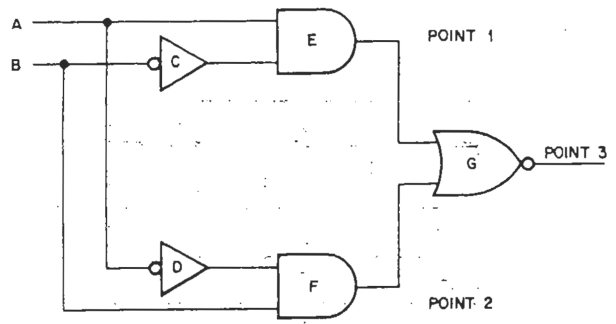


Figure 2.13 AND/NOR Comparator

INPUT CONDITION		POINT	POINT	POINT
A	B	1	2	3
LO	LO	LO	LO	HI
LO	HI	LO	HI	LO
HI	LO	HI	LO	LO
HI	HI	LO	LO	HI

Experiment 2.4

A	B	A Exclusive OR B
LO	LO	LO
LO	HI	HI
HI	LO	HI
HI	HI	LO

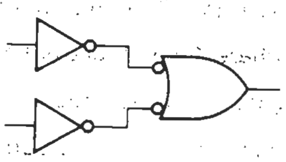
Figure 2.13 Exclusive OR Truth Table

Questions

4. Comparator function

5.

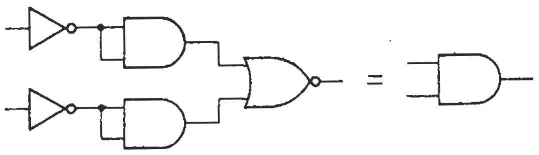
INPUT		OUTPUT
A	B	
LO	LO	LO
LO	HI	HI
HI	LO	HI
HI	HI	HI



OR Function

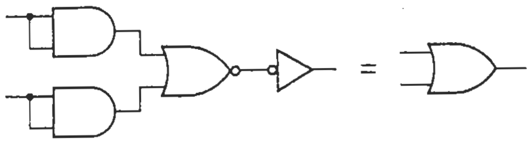
(NAND = Negated Input OR)

6.



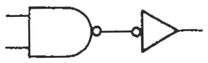
AND Function

7.



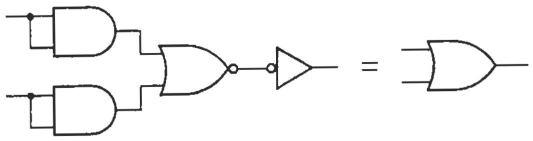
OR Function

8.



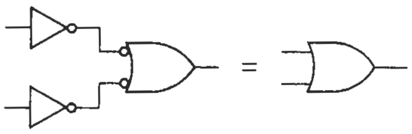
AND Function

9. Suggested Method



OR Function

Alternate Method

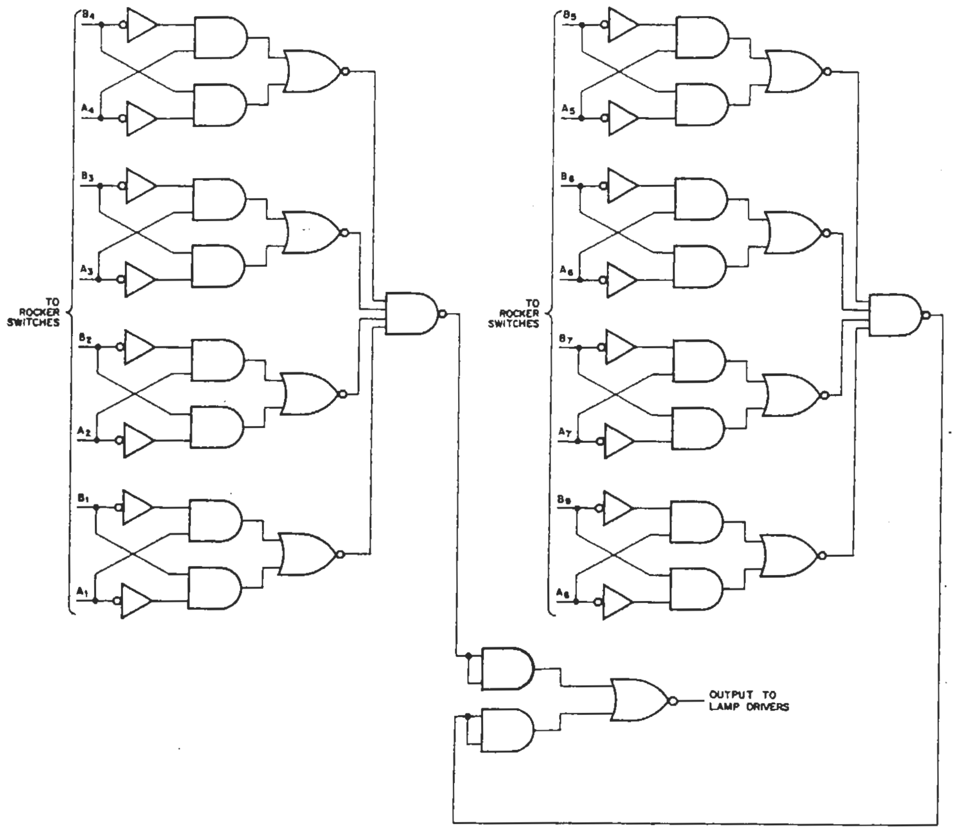


10. The 4-Bit Equality Detector is composed of four identical sections that are comparing corresponding bits of two binary numbers. The outputs of each of the identical sections are ANDed together in an output section that consists of a NAND gate followed by an inverter. Since the operation of each 2-bit Comparator is identical, an explanation of the operation of one comparator section follows.

When A-4 and B-4 are equal, the outputs of both AND gates of the AND/NOR element are LO, and the resultant, output from the output NOR is HI. When A-4 and B-4 are not equal, one of the AND gate outputs is HI and the resultant output from the output NOR is LO.

When all of the 2-bit comparator sections indicate equality, the output AND function (NAND followed by an inverter) will yield a HI output.

11.



Equality Detector For Two 8-Bit Switch Register

BINARY NUMBER				PARITY BIT
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Figure 2.17 Parity Bit Truth Table

12. Interpreting the logic of the Parity Bit Generator, Figure 2.18, in terms of not adding a parity bit, yields the following explanation. Do not add a parity bit (output LO) for the following conditions:

Both Equal				Both Unequal			
2^0	2^1	2^2	2^3	2^0	2^1	2^2	2^3
0	0	1	1	0	1	0	1
1	1	0	0	1	0	1	0
0	0	0	0	0	1	1	0
1	1	1	1	1	0	0	1

The first set of Exclusive OR Functions looks at pairs of bits in the number being examined. If there are an uneven number of 1's in these pairs of bits a HI is passed on to the second Exclusive OR Function (output logic).

The following diagram illustrates the parity bit generator output logic. With either gate A or gate B enabled (HI output), the resultant NOR out is LO, indicating do not add parity.

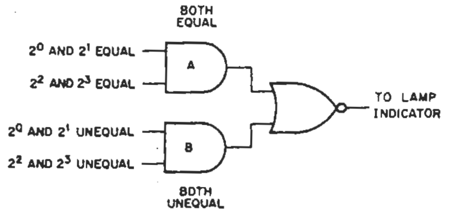
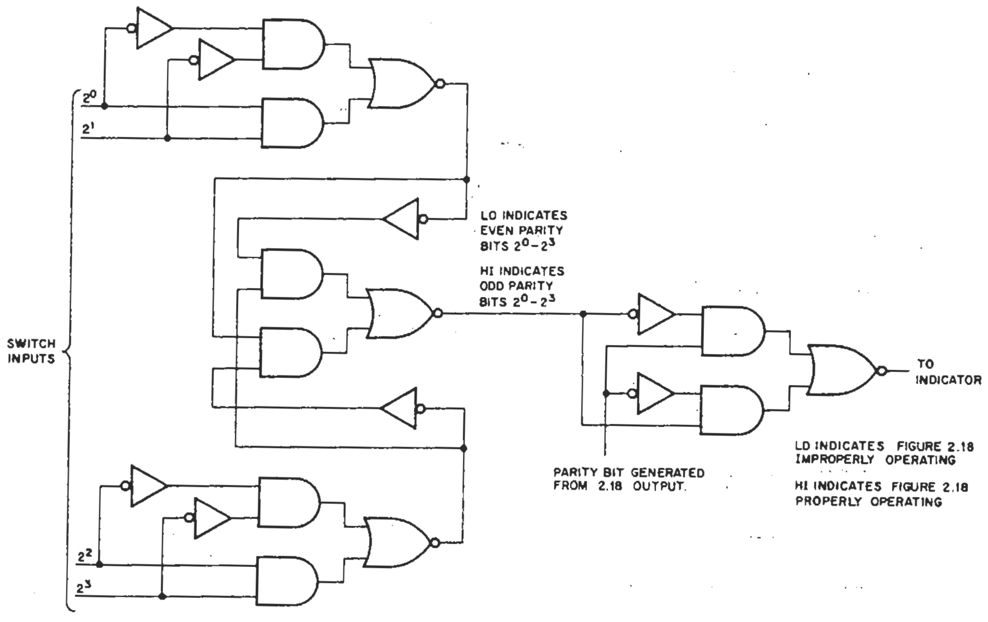


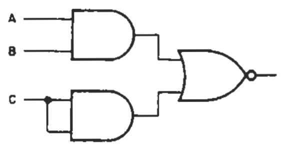
Figure 2.18 Parity Bit Generator Output Logic

13. The following 5-Bit Parity Checker is a separate set of logic that determines if Figure 2.18 is operating properly. The 5-Bit Parity Checker does not generate a parity bit but a Properly Operating" or "Improperly Operating" signal.


2^0
2^1
2^2
2^3
SWITCH INPUTS
LO INDICATES EVEN PARITY BITS 2^0-2^3
HI INDICATES ODD PARITY BITS 2^0-2^3
PARITY BIT GENERATED FROM 2.18 OUTPUT.
TO INDICATOR
LO INDICATES FIGURE 2.18 IMPROPERLY OPERATING
HI INDICATES FIGURE 2.18 PROPERLY OPERATING

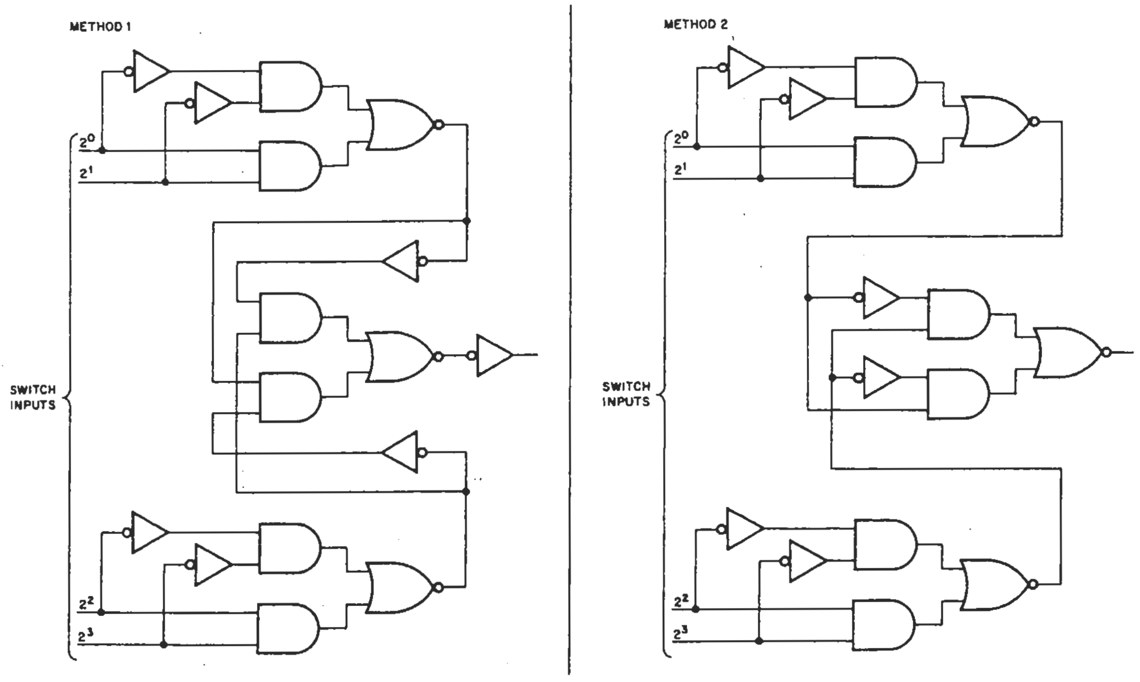
5-Bit Parity Checker

14.



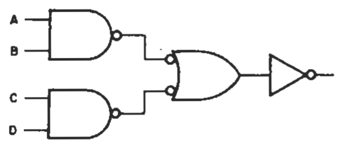
AND/NOR Configuration Where Output LO if Inputs
A and B = HI, or Input C is HI

15.



4-Bit Odd Parity Generators

16.



AND/NOR Function Using NAND Gates

17. No. But the chances of two bits changing is much less than one bit changing.

CHAPTER 3
FLIP-FLOPS

INTENT AND APPROACH

The objectives of this chapter are as follows.

a. to introduce the flip-flop as a digital storage element;

b. to give the students a thorough understanding of the operation of the R-S Flip-Flop, the D Type Flip-Flop, and the Master-Slave J-K Flip-Flop;

c. to demonstrate a practical application of flip-flops by having the student construct a serial shift register.

Because the flip-flop is such an important logic device and is implemented in many forms in modern computers, Chapter 3 presents a detailed analysis of different types of flip-flops.

Chapter 3 begins with the simplest form of flip-flop, the R-S Flip-Flop, and uses this basic form to develop more complex flip-flops such as the D Type and the J-K Flip-Flop.

In this chapter the student builds each type of flip-flop and demonstrates to himself the advantages and disadvantages of each.

Since the flip-flop is generally part of a register, a practical application of flip-flops is demonstrated by having the student build a serial shift register. The teacher may elect to present additional applications of flip-flops by presenting parallel shift registers and ring counters. Additional information is contained in the Supplementary Information section, see pages 38 through 40.

Care must be taken to impress upon the student the fact that he must not simply wire the experiments but must also understand logically how each configuration operates. Having the students construct the 4-Bit J-K Shift Register without the aid of the Workbook provides an exercise in learning shift register logic.

OUTLINE

I. INTRODUCTION (Page 33)
   A. Non-memory Devices
   B. Memory Devices
   C. Flip-Flop
      1. Output conditions
         a. "1" condition
         b. "0" condition
      2. Truth table

II. THE R-S FLIP-FLOP (RESET-SET FLIP-FLOP) (Pages 33-35)

A. Inputs
   1. SET
   2. RESET
B. Outputs
   1. One
   2. Zero
C. Responses to Input Pulses
D. Logic Diagram
E. Operation
   1. Latched "1" condition
   2. Latched "0" condition
   3. Indeterminate condition
F. Experiment 3.1: R-S Flip-Flop (Page 34-35)
   1. Circuit construction
   2. Truth table
   3. Verification of operation
G. Questions 1-3 (Page 35)

III. CLOCKED R-S FLIP-FLOP (Pages 35-37)

A. Pulse Sources
   1. Clock
   2. Pulser switches
B. Steering Network
   1. HI SET
   2. HI RESET
C. Logic Diagram
D. Experiment 3.2: Clock R-S Flip-Flop (Page 36-37)
   1. Truth table completion
   2. Circuit construction
   3. Verification of operation
E. Questions 4-6 (Page 36)

IV. D TYPE FLIP-FLOP (Pages 37-38)

A. Inputs
B. Operation

- C. Experiment 3.3: D Type Flip-Flop (Page 37)
    1. Introduction
    2. Truth table completion
    3. Circuit construction
    4. Verification of operation
    5. Logic diagram
- D. Questions 7-9 (Page 38)

V. MASTER-SLAVE J-K FLIP-FLOP (Pages 38-40)

- A. Unique features of J-K Flip-Flops
- B. Operation of COMPUTER LAB (J-K) Flip-Flops
- C. Logic Diagram
- D. Experiment 3.4: J-K Flip-Flop (Page 39)
    1. Truth table completion
    2. Circuit construction (as per Figure 3.8)
    3. Verification of operation
    4. Examination of transition time
    5. Operation of COMPUTER LAB Flip-Flop
    6. Verification of operation of COMPUTER LAB flip-flop
- E. Questions 10-13 (Page 40)

VI. J-K SHIFT REGISTER (Pages 40-43)

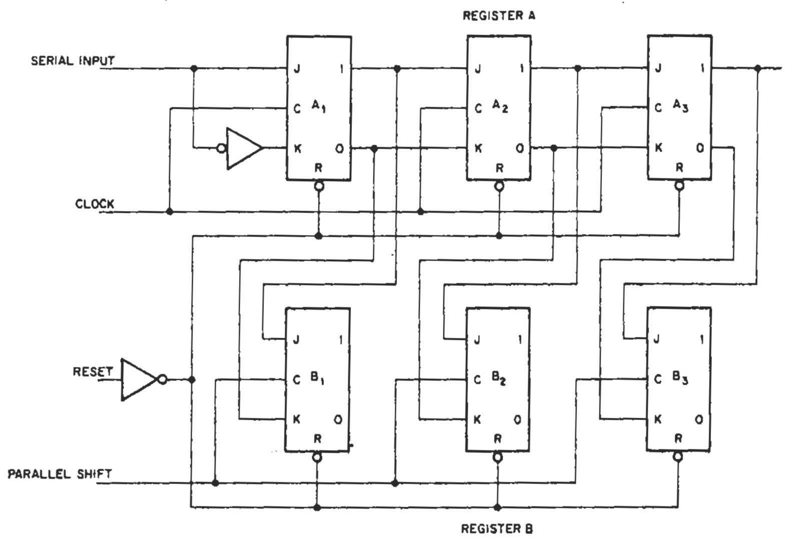
- A. Introduction
- B. Operation of J-K Shift Register
- C. Logic Diagram
- D. Experiment 3.5: 4-Bit Shift Register (Page 41)
    1. Truth table completion
    2. Circuit construction
    3. Verification of operation
- E. Questions 14-15 (Pages 41-42)
- F. Supplementary Questions 16-21 (Pages 42-43)

SUPPLEMENTARY INFORMATION

I. PARALLEL TRANSFER

In Chapter 3 serial transfer is illustrated. In a serial transfer, a number enters the register one bit at a time. A second method of moving information in a computer is called parallel transfer. In a parallel transfer, all bits of a number are transferred simultaneously.

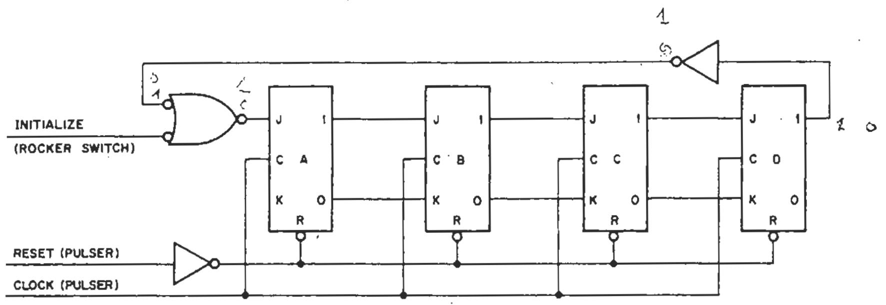
The following diagram illustrates a parallel transfer from Register A to Register B. Information is shifted into Register A in the serial mode.



At the completion of the serial transfer, the flip-flops in Register A are "steering" Register B to some desired state. Assuming Register A contains the binary number 101, flip-flop A_1 would be steering flip-flop B_1 to the "1" state, flip-flop A_2 would be steering B_2 to the "0" state, and A_3 would be steering B_3 to the "1" state. At the lagging edge of the parallel shift pulse, both Register A and Register B would contain the binary number 101. The state of Register B flip-flops prior to the parallel transfer is insignificant because whatever is present in Register A prior to the transfer will be "jammed" into Register B.

II. RING COUNTERS

Basic Ring Counter



The Ring Counter is used primarily for control applications. The counter is a serial shift register with the output flip-flop connected back to the input. The least significant bit (flip-flop A) is initially set to the "1" state. Each succeeding clock pulse after initialization will shift the 1 to the right. The content of the most significant bit (flip-flop D) is always shifted into flip-flop A.

The Ring Counter is operated as follows:

Step 1: Reset all flip-flops to the "0" state by pressing the RESET pulser.

Step 2: Initialize the counter by placing the rocker switch input to flip-flop A to a LO and providing a clock pulse. This step will cause flip-flop A to go to the "1" state.

Step 3: Return the rocker switch to a HI.

Step 4: Press the clock pulse in order to provide shift pulses.

Each clock pulse will now cause the counter to shift the 1 initially present in flip-flop A to the right. The 1 will shift from A to B, B to C, C to D, and D to A. The following table illustrates the counting sequence for the Ring Counter.

FLIP-FLOPS				
D	C	B	A	
0	0	0	1	(Initialized)
0	0	1	0	
0	1	0	0	
1	0	0	0	

Recycles

Because only one flip-flop is in the "1" state at a given time, each flip-flop represents one state or count of the counter. Decoding (reading) the counter state is far less complex than reading other counter types such as a binary up counter.

Switch-Tail Ring Counter

A Switch-Tail Ring Counter is a modified Ring Counter. The Switch-Tail Ring Counter is a serial shift register with the most significant 1 output steering the least significant 0 input and the most significant 0 output steering the least significant 1 input.

(Refer to Figure 3.15, Page 44 in the COMPUTER LAB Workbook.) The Switch-Tail Ring Counter is used for control applications and provides twice the number of counts that a ring counter provides. Decoding the Switch-Tail Ring Counter's present state or count is more complex than decoding the Ring Counter's present state. However, decoding the Switch-Tail Ring Counter's count is less complex than decoding other types of counters; i.e., the Binary Up Counter.

The following table illustrates the counting sequence for the Switch-Tail Ring Counter.

Flip-Flops				Count	
D	C	B	A		
0	0	0	0	0	(Initialized)
0	0	0	1	1	
0	0	1	1	2	
0	1	1	1	3	
1	1	1	1	4	
1	1	1	0	5	
1	1	0	0	6	
1	0	0	0	7	

Recycles

Determining (decoding) the counter's present count requires examination of at least two flip-flop outputs. The chart below illustrates the decoding process.

Count	
0	$\overline{A}\,\overline{D}$
1	$A\overline{B}$
2	$B\overline{C}$
3	$C\overline{D}$
4	AD
5	$\overline{A}B$
6	$\overline{B}C$
7	$\overline{C}D$

ANSWER KEY

Questions

1. Referring to Figure 3.2, and assuming the flip-flop is in the "1" state, if both the SET and RESET inputs are made HI, gate A will have as its inputs a HI from the SET input and a LO from the 0 output on the flip-flop. As a result of this input condition, gate A will be enabled and its output will remain a HI. Gate B will have as its inputs a HI from the RESET input and a HI from the 1 output. As a result of this input condition, gate B will be disabled and its output will be LO and will hold gate A enabled.

2.a Referring to Figure 3.2, when the flip-flop receives a LO RESET pulse, gate B is enabled and its output goes HI. With the SET input HI and the output from gate B HI, gate B is disabled and its output goes LO. The LO output from gate A, fed back to gate B, keeps gate B enabled and the flip-flop in the "0" state after the RESET has returned to the HI condition.

2.b The results of operating the SET and RESET rocker switches simultaneously are inconsistent.

2.c The recurrence of one particular output state is dependent on mechanical and/or electronic considerations. If both the SET and RESET inputs are enabled at the same time, the flip-flop goes into an indeterminate state where both outputs are HI. The flip-flop will go to the "1" or the "0" state, depending upon which input remains last. If both inputs are removed simultaneously, the flip-flop will assume some state that depends upon which gate disables first. Internal electronic considerations will determine which gate will disable first.

3. The fan-out of each of the R-S flip-flop outputs is nine unit loads. Each output must drive one internal gate input, therefore decreasing its fan-out by one.

Experiment 3.2

Initial Conditions		Signal Inputs		After Clock Pulse	
1 Output	0 Output	Set	Reset	1 Output	0 Output
LO	HI	LO	LO	LO	HI
LO	HI	LO	HI	LO	HI
LO	HI	HI	LO	HI	LO
LO	HI	HI	HI	Indeterminate	

Figure 3.5 Clocked R-S Flip-Flop Truth Table

Initial Conditions		Signal Inputs		After Clock Pulse	
1 Output	0 Output	Set	Reset	1 Output	0 Output
HI	LO	LO	LO	HI	LO
HI	LO	LO	HI	LO	HI
HI	LO	HI	LO	HI	LO
HI	LO	HI	HI	Indeterminate	

Figure 3.5 Clocked R-S Flip-Flop Truth Table (Cont)

Questions

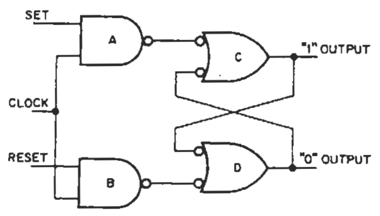
4.a A clocked R-S flip-flop transition takes place on the leading edge of a HI pulse.

4.b On the leading edge of a clock pulse, either gate A or gate B will be enabled (Figure 3.4) causing the flip-flop to change state.

5. An indeterminate resultant output condition in an R-S flip-flop will occur after clock-pulse time if RESET and SET inputs are simulataneously HI previous to clock pulse.

This indeterminate condition results because, during clock-pulse time, the outputs from gate A and gate B (Figure 3.4) go LO simultaneously, causing the flip-flop to produce HI outputs and placing the flip-flop in an illegal state. After clock pulse time, outputs from both gate A and gate B simultaneously go HI and the resultant state is unpredictable and dependent on the speed of the gates in the flip-flop.

6.



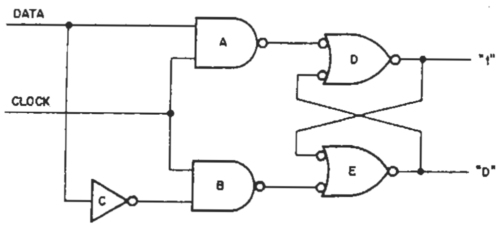
Gate A combines the SET input and clock input to place the flip-flop into the "1" state. Gate B combines the RESET input and clock input to place the flip-flop into the "0" state. Gate C provides the 1 output and also a feedback to the input to gate D. When the flip-flop is in the "1" state, the output of gate C is a HI. When the flip-flop is in the "0" state, the output of gate C is a LO, keeping gate D "latched." Gate D provides the 0 output and also a feed back to the input of gate C. When the flip-flop is in the "0" state, the output of gate D is a HI. When the flip-flop is in the "1" state, the output of gate D is a LO, keeping gate C "latched."

Experiment 3.3

Initial Conditions		D Input	After Clock Pulse	
1 Output	0 Output		1 Output	0 Output
LO	HI	LO	LO	HI
HI	LO	LO	LO	HI
LO	HI	HI	HI	LO
HI	LO	HI	HI	LO

Figure 3.7 D-Type Flip-Flop Truth Table

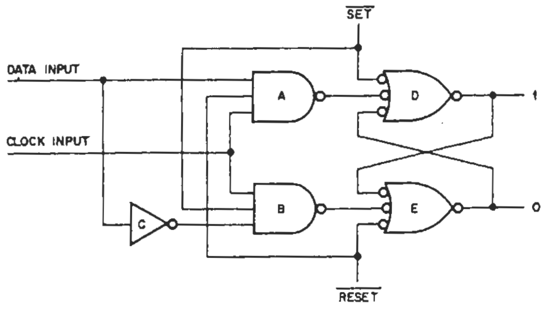
7.



Gates D and E act as an R-S flip-flop controlled by gates A and B. A LO output from gate A places the flip-flop in the "1" state. A LO output from gate B places the flip-flop in the "0" state. When there is a HI on the input data line, a clock pulse will be steered through gate A. When the clock pulse occurs, gate A will have two high inputs which will set the flip-flop composed of gates D and E. A LO on the data link will cause the output inverter C to be HI and at clock-pulse time gate B will have two HI inputs and reset the flip-flop.

8. Because there is only one conditioning input (data input) there can be no indeterminate state in the operation of a D-Type flip-flop.

9.



D Type Flip-Flop with Direct SET and RESET Inputs

Experiment 3.4

Initial Conditions				Final Conditions	
Outputs		Inputs		Outputs	
1	0	J	K	1	0
LO	HI	LO	LO	LO	HI
LO	HI	LO	HI	LO	HI
LO	HI	HI	LO	HI	LO
LO	HI	HI	HI	HI	LO
HI	LO	LO	LO	HI	LO
HI	LO	LO	HI	LO	HI
HI	LO	HI	LO	HI	LO
HI	LO	HI	HI	LO	HI

Figure 3.9 Master-Slave J-K Flip-Flop Truth Table

Questions

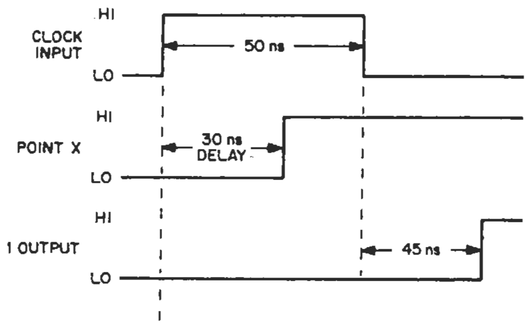
10. The J-K flip-flop has NO indeterminate state. (Refer to Figure 3.8.) When the flip-flop is in the "1" state, the 1 output is fed back, enabling gate B; the 0 output is fed back, disabling gate A. When the flip-flop is in the "0" state, the 1 output is fed back to gate B, disabling it; the 0 output is fed back to gate A, enabling it. The feedback arrangement allows only one of the steering gates (A or B) to be enabled at one time. This prevents a condition that would attempt to place the flip-flop in the "1" and "0" states simultaneously, resulting in an indeterminate condition.

11. Assuming that the J-K flip-flop is initially in the "0" condition and the J input is enabled and the K input is disabled, the following occurs during clock pulse time (Figure 3.8).

a. At the leading edge of the clock pulse: Gate B is disabled by the LO from the 1 output and the K input. Gate A is enabled by the J input, the 0 output, and the clock pulse. The resulting LO out of gate A causes the master flip-flop (gates C and D) to go to the "1" state, placing a HI at point X. The inverter (I) output is disabling the slave steering gates, E and F, and the slave flip-flop remains in the "0" state.

b. At the trailing edge of the clock pulse: Gate F is disabled by LO from gate D. Gate E is enabled by point X being HI and the output of the inverter (I). Gate E being enabled results in a low output causing the slave flip-flop to go to the "1" state. No further change can occur in the master flip-flop after the clock input goes LO because the clock input disabled both gate A and gate B.

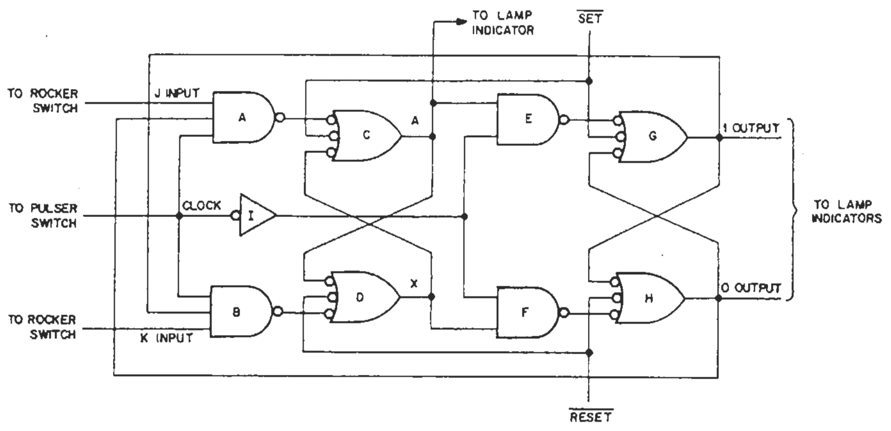
12.



NOTE
J-K flip-flop delay is not precise and varies around 30 ns.

Figure 3.10 J-K Flip-Flop Timing Diagram

13.



J-K Flip-Flop with Direct SET and RESET

NOTE

Both the master and the slave flip-flops must receive direct set of reset inputs to insure that when the direct set or direct reset leave, the master flip-flop does not return the slave to the original condition.

Experiment 3.5

	Flip-Flop A	Flip-Flop B	Flip-Flop C	Flip-Flop D
Initial Condition	"0"	"0"	"0"	"0"
After Clock Pulse				
1	"1"	"0"	"0"	"0"
2	"0"	"1"	"0"	"0"
3	"0"	"0"	"1"	"0"
4	"0"	"0"	"0"	"1"
5	"0"	"0"	"0"	"0"

Figure 3.12 Four-Bit Shift Register Truth Table (Assumes Data Input Goes to Zero After First Clock Pulse)

14.

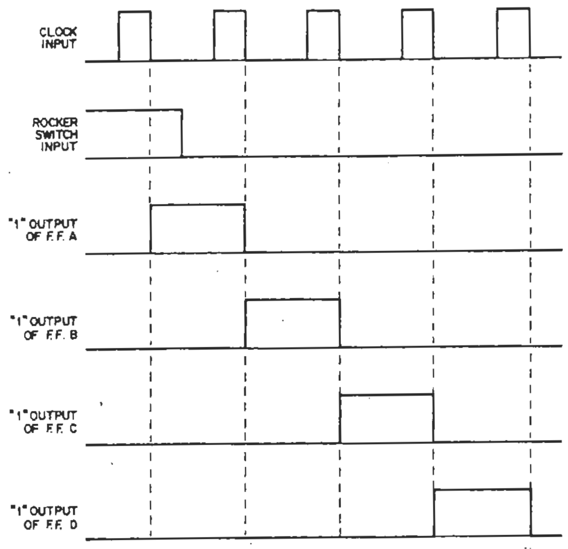
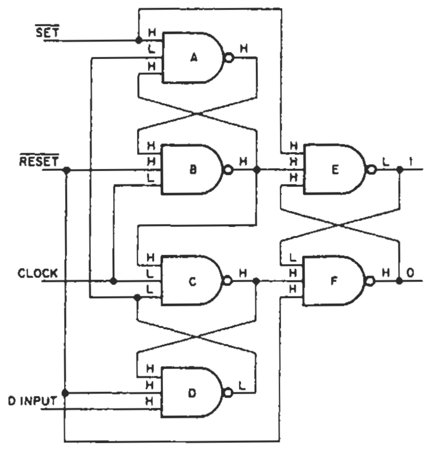


Figure 3-13 Four-Bit Register Timing Diagram

15. Refer to Figure 3.11 and assume that all flip-flops are initially in the "0" condition and the rocker switch to the J input of flip-flop A is providing a HI level. When the first clock pulse occurs, flip-flop A will go to the "1" condition because its J input will be enabled with a HI level and its K input will be disabled with a LO. Flip-flop B will remain in the "0" condition because its J input is disabled with a LO from flip-flop A and its K input is enabled with a HI from flip-flop A. Similarly, flip-flop C will remain in the "0" condition as will flip-flop D. If the rocker switch is placed in the LO condition before the next clock pulse, flip-flop A will go to the "0" condition on the next clock pulse. Flip-flop B will go to the "1" condition because its J input is enabled by the 1 output of flip-flop A and its K input is disabled by the 0 output of flip-flop A. Flip-flops C and D will remain in the "0" condition because their K inputs are enabled with a HI level and their J inputs are disabled with a LO level. Successive clock pulses will shift the 1, which was initially fed into flip-flop A on the first clock pulse, to flip-flop B, then to flip-flop C, and finally to flip-flop D.



16.a. (1) Referring to the above drawing of the D Type Flip-Flop, and assuming initially that it is in the "0" state, there would be a HI on the output of gate F, and a LO on the output of gate E. Assume that the D input is HI, the clock input is LO, the $\overline{\text{RESET}}$ input is HI, and the $\overline{\text{SET}}$ input is HI.

16.a.(2) If we want the flip-flop to make a 0 to 1 transition, we would place the D input at a HI. Gate D now has a HI input from the D input line, and a HI input from the $\overline{\text{RESET}}$ input. Gate C has a LO clock input; thus gate C has a HI output which is fed to gate D. Therefore, all inputs to gate D are HI, and the output from gate D is LO. Gate D's output is fed to gate C, and to the input of gate A. Gate B has a HI input from the $\overline{\text{RESET}}$, and a LO clock input, causing a HI out of gate B. The HI is fed to gate C and to gate A. Gate A has a HI input from the output of gate B, a LO input from the output of gate D, and a HI from the $\overline{\text{SET}}$ input. With this input configuration, the output from gate A is a HI, which is fed into gate B.

16.a (3) At clock pulse time, gate C will get a HI from the leading edge of the clock, but, since the output of gate D is a LO, there will be no change in the output of gate C. Gate B, with the clock pulse HI, will now have all HI inputs, and the output of gate B will go LO. The LO out of gate B will be fed to gate A and gate E where it will cause the output R-S Flip-Flop (composed of gates E and F) to change state. The output of gate E will go HI and will be fed to gate F. All HIs into gate F will produce a LO output which is fed to gate E, maintaining the output of gate E at a HI or in the "1" state.

16.b The D input has no effect on the flip-flop after the leading edge of the clock pulse has occurred. This characteristic is called data lockout. Data lockout prevents a change in the data line from affecting the state of the flip-flop after the leading edge of the clock pulse.

(1) For instance, if the data input were HI, at clock pulse time the output of gate B would go LO, causing the output R-S flip-flop to go to the "1" state. Also, the output of gate B, which is a LO, is fed to the input of gate C, keeping one input of gate C LO, and the output HI. Thus, if the data input were to go LO during clock pulse time, it would have no effect on gate C, and, similarly, no effect on the state of the flip-flop.

(2) If the data input were made LO with the desire to place the flip-flop in the "0" state, at clock-pulse time the output of gate C would go LO, causing the output R-S flip-flop to go to the "0" state. Also, the LO output of gate C, is fed to gate D maintaining the HI output of gate D; thus a data input change from LO to HI would have no effect on the state of the flip-flop.

17. The $\overline{\text{RESET}}$ input going from HI to LO causes the output R-S flip-flop to go to the "0" state by placing a LO on the input of gate F. In order to protect the flip-flop from indeterminate conditions, the $\overline{\text{RESET}}$ must perform multiple functions.

Condition 1: Assuming that the data input line is HI, the clock pulse is present, and $\overline{\text{RESET}}$ goes LO, the flip-flop would be attempting to go to both the "0" and "1" states. To prevent this condition, $\overline{\text{RESET}}$ is fed to gate B causing gate B output to go HI, and preventing an indeterminate condition.

The $\overline{\text{SET}}$ input, with a HI to LO transition, is fed to gate E, and a LO input to gate E causes the output R-S flip-flop to go to the "1" state.

In order to protect the flip-flop from an indeterminate condition caused by the $\overline{\text{SET}}$ input attempting to set the flip-flop and the data input and clock pulse attempting to reset the flip-flop simultaneously, the $\overline{\text{SET}}$ input performs multiple functions. Besides being a direct input to gate E, $\overline{\text{SET}}$ also feeds gate A and, when $\overline{\text{SET}}$ is LO, results in a HI out of gate A. During clock pulse time, gate B will produce a LO, keeping the output of gate C HI (disables gate C).

Condition 2: An understanding of the purpose of the RESET input to gate D can best be obtained by observing the results when $\overline{\text{RESET}}$ is disconnected from gate D and the following is assumed:

a. $\overline{\text{RESET}}$ LO

b. $\overline{\text{SET}}$ HI

c. Clock pulse LO

d. $\overline{\text{RESET}}$ input to gate D disconnected (center input). Gate D center input connected to a constant HI.

e. Data input goes LO 20-ns prior to $\overline{\text{RESET}}$ returning to HI (see timing diagram).

f. Clock pulse goes HI at lagging edge of $\overline{\text{RESET}}$ (see timing diagram).

With a LO data input to gate D and the above listed conditions true, after 20 ns the output of gate D will go HI (assuming maximum gate delay). At this time the following conditions exist:

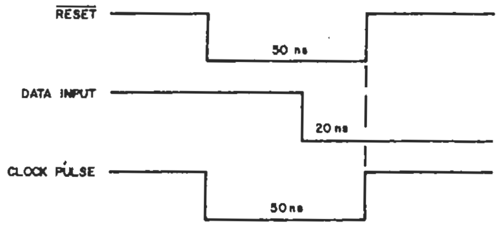
a. Gate A output is still HI (will change to LO 20-ns later).

b. $\overline{\text{RESET}}$ is HI

c. Clock is HI

Because of the above conditions, the output of gate B will go LO, placing the flip-flop in the "1" state, regardless of the fact that the data input was LO 20-ns previous to clock time (a manufacturer's specification for data changes prior to clock input) which should have placed the flip-flop into the "0" state.

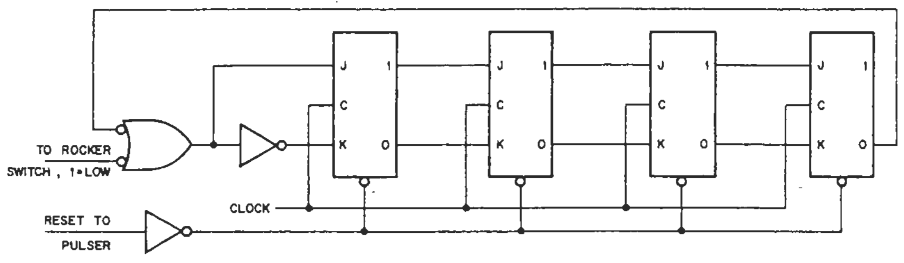
With $\overline{\text{RESET}}$ (which is normally a 50-ns pulse) reconnected to gate D, the output of gate D will go HI in 20 ns and the output of gate A will go HI in 40 ns after the leading edge of $\overline{\text{RESET}}$. This guarantees that on the lagging edge of $\overline{\text{RESET}}$, if the clock pulse goes HI, the flip-flop will not go into an opposite state indicated by the D input.



Timing Diagram

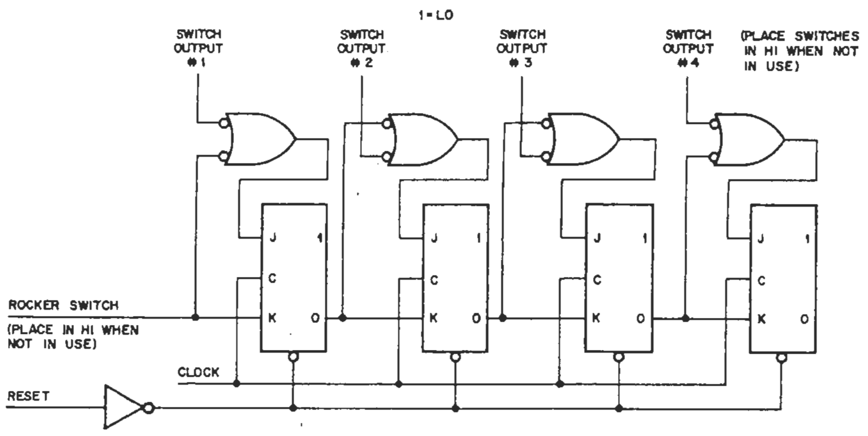
18. Refer to the J-K Flip-Flop with Direct SET and RESET on page 46. On the J-K Flip-Flop, if the $\overline{\text{RESET}}$ and J inputs are both enabled prior to a clock pulse, after the clock pulse has passed, the flip-flop will be in the "0" state. Previous to the clock pulse, both the master and slave flip-flops are being held in the "0" state by $\overline{\text{RESET}}$. On the leading edge of the clock pulse, gate A's output goes LO, placing the master flip-flop in an indeterminate state. On the trailing edge of the clock pulse, the output of gate A goes HI and the master flip-flop returns to the "0" state because $\overline{\text{RESET}}$ remains LO. The lagging edge of the clock pulse has no effect on the slave flip-flop because the master flip-flop is in the "0" state at this time.

19.



Ring Counter with Rocker Switch and RESET Inputs

20.



Parallel Transfer - Serial Shift

21.

D	C	B	A	
0	0	0	0	(Initialized)
0	0	0	1	
0	0	1	1	
0	1	1	1	
1	1	1	1	
1	1	1	0	
1	1	0	0	
1	0	0	0	
0	0	0	0	

Switch-Tail Ring Counter Truth Table

CHAPTER 4
BOOLEAN ALGEBRA TO GATING NETWORKS

INTENT AND APPROACH

The objectives of this chapter are as follows:

a. to explain the basic principles of Boolean algebra;

b. to demonstrate how Boolean functions can be implemented using gating networks; and

c. to explain how gating networks can be simplified by using Boolean algebra techniques.

In this chapter the basic laws and identities of Boolean algebra are presented. The Workbook uses truth table techniques to demonstrate the validity of these Boolean principles and supplementary material provided in this Teacher's Guide allows the teacher the option of selecting two other methods of Boolean algebra proof, the Venn diagram or the logic diagram proof.

In order to demonstrate how Boolean functions can be translated into gating networks, Chapter 4 illustrates the Boolean expressions for the COMPUTER LAB gates. In Section V a sample problem is given, translated into a Boolean expression, simplified, and then constructed using COMPUTER LAB gates. Additional material concerning the conversion of Boolean expressions to gates and gates to Boolean expressions is found in the supplementary material for this chapter.

Simplification techniques are explained by having the student first simplify an expression using Boolean algebra and then construct a truth table which yields the possibilities of further simplification. When these tasks are completed, the student is instructed to translate the Boolean expression into a gating circuit, which could present further means of simplification.

In order to give the student an opportunity to solve problems relating to the material presented in this chapter on a practical logic circuit, the last section of the chapter introduces an Equality and Relative Magnitude Detector.

Additional material on Boolean algebra is presented in Appendix B. Appendix B explains the Karnaugh Mapping technique of simplifying Boolean expressions.

OUTLINE

I. INTRODUCTION (Page 45)
   A. Two-state Algebra
   B. Advantages and Use
      1. Facility in working with logic functions

2. Aid in reducing complexity of gating network
    a. Reduction in cost
    b. Increase in speed
3. Techniques for reduction of gating networks

II. BOOLEAN OPERATORS (Pages 45-48)

A. Algebraic Signs Used to Express Logic Relations

B. Characteristics

1. Composition of Boolean expressions

2. Possible values of variables and expressions

C. Identification by Construction of Truth Table

D. Relationship of Boolean Symbols, Functions, and Logic Symbols

E. Truth Tables for Basic Boolean Operators

F. Combination of Operators to Give More Complex Functions

1. NAND

2. Negative Input OR

3. AND/NOR

4. NOR

5. Negated Input AND

6. Inverter

III. BOOLEAN LAWS (Page 48)

A. Commutative

B. Associative

C. Distributive

D. Truth Table Proof

IV. BOOLEAN IDENTITIES AND DEMORGAN'S THEOREM (Pages 48-52)

A. Basic Boolean Identities with One Variable

B. Identities with Two or More Variables

C. Sample Proofs of Identities

1. Proofs using one variable identities

2. Proofs using truth tables

D. Statement of DeMorgan's Theorem

1. Equivalence of Negated Input AND Gate and NOR Gate

2. Equivalence of Negated Input OR Gate and NAND Gate

E. Algebraic Simplification Techniques

1. Multiplying out

2. Factoring

F. Questions 1-3 (Page 52)

V. BOOLEAN FUNCTIONS TO GATING NETWORKS (Pages 52-60)

A. Translation of Logic Requirement into a Boolean Statement

1. Sample problem

2. Assignment of names to variables

B. Questions 4-5 (Page 52)

C. Simplification of Boolean functions

D. Question 6 (Page 53)

E. Truth Table Reduction

F. Question 7 (Page 56)

G. Hardware Considerations

1. Expression of function in terms of AND, OR, and INVERT functions only

2. Translation into functions available on COMPUTER LAB

3. Further simplification by direct circuit simplification

H. Experiment 4.1: Gating Circuit Simplification (Page 59-60)

1. Determine Boolean equation for each illustrated network

2. Circuit construction of unsimplified network

3. Construction of truth table

4. Simplification of Boolean equations

5. Reduction of circuit to minimum number of gates

6. Construction of simplified circuit

7. Verification of operation

VI. EQUALITY AND RELATIVE MAGNITUDE DETECTOR (Pages 61-64)

A. Capabilities

B. Most Significant Unequal Bit

C. Explanation of Operation of Equality Detector Section

D. Explanation of Operation of Relative Magnitude Section

E. Logic Diagram of Equality Detector

F. Logic Diagram of Relative Magnitude Detector

G. Experiment 4.2: Equality and Relative Magnitude Detector (Page 64)

1. Circuit construction

2. Verification of operation

H. Questions 8-12 (Page 64)

I. Supplementary Questions 13-17 (Page 64)

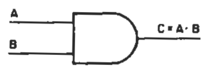
SUPPLEMENTARY INFORMATION

I. BOOLEAN EXPRESSIONS AND LOGIC DIAGRAMS

Simple Boolean Expressions

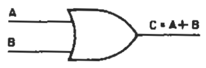
Boolean algebra provides symbolic notation for the logical operations of AND, OR, and NOT.

We can represent the logical operation of the following logic element with the notation $A \cdot B = C$. A and B are inputs, $\cdot$ is the Boolean symbol for AND, and C is the output.



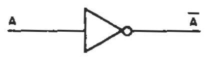
By using the expression $A \cdot B$ (which may be shortened to AB), we are saying that the output at point C will be true when A and B are true.

An OR function (illustrated below) is represented by the expression $A + B = C$.

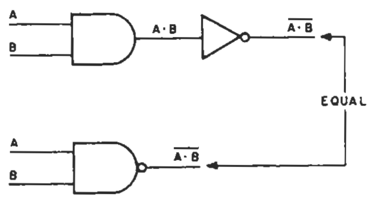


By using the expression $A + B = C$, we are saying the output at point C will be true when either A or B are true. The symbol for OR is + in Boolean notation.

If the input to a logical inverter is A, the output can be represented as $\overline{A}$ (read not A.).



The resulting Boolean expression for a NAND function is shown below.

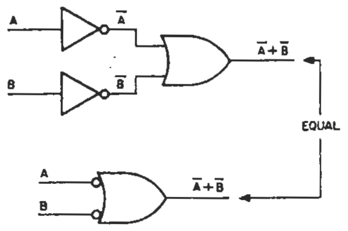


The AND symbol may be eliminated and NAND may be represented as $\overline{AB}$

NAND Symbol

The expression $\overline{AB}$ is read "not A and B." When using this expression we are saying the output will be false when both A and B are true.

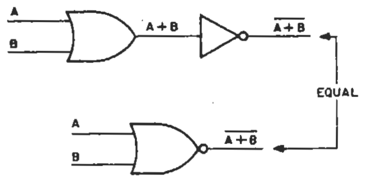
The resulting Boolean expression for a Negated Input OR function is shown below.



Negated Input OR

The expression $\overline{A} + \overline{B}$ is read "not A or not B." When using this expression we are saying when A is not true or B is not true, the output will be true.

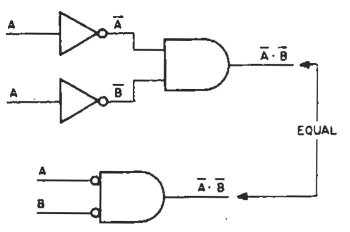
The resultant Boolean expression for a NOR function is shown below.



NOR Symbol

The expression $\overline{A + B}$ is read "not A or B." When using this expression we are saying the output is false when either A or B is true.

The resultant Boolean expression for a Negated Input AND function is shown below.


A
$\overline{A}$
A
$\overline{B}$
$\overline{A}\cdot\overline{B}$
EQUAL
A
B
$\overline{A}\cdot\overline{B}$

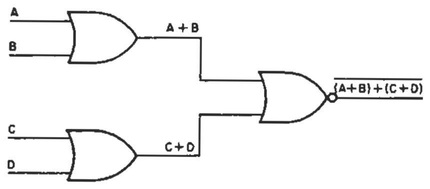
Negated Input AND

The $\cdot$ may be eliminated and the expression becomes $\overline{A}\overline{B}$

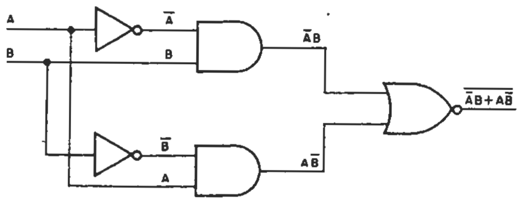
The expression $\overline{A}\cdot\overline{B}$ is read "A not and B not." When using this expression we are saying the output is true when both inputs are false.

Complex Boolean Expressions

More complex logic diagrams with Boolean notations are shown below.



Multivariable inputs are grouped using parentheses, brackets, or vinculums.

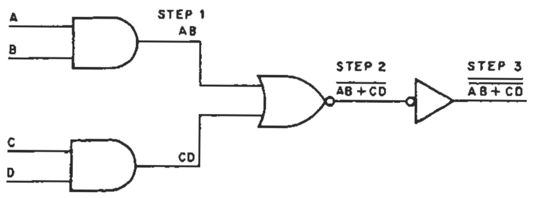


ANDed inputs to an OR function need not be grouped with parentheses.

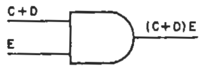
Rules for Finding Output Expressions

The following rules summarize how to find the output expression for a logic diagram

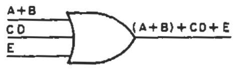
a. Begin at the left of the diagram and find the output expression for each logic symbol.



b. An input expression to any symbol may be represented by two or more letters. These letters should remain grouped in the output expression.



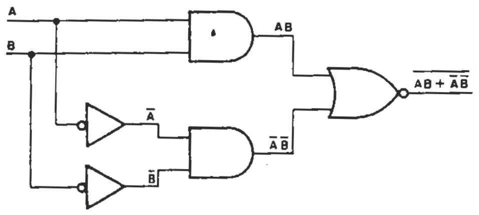
c. Parentheses, brackets, and the vinculum are used as grouping signs. An ANDed input to an OR function requires no grouping sign.

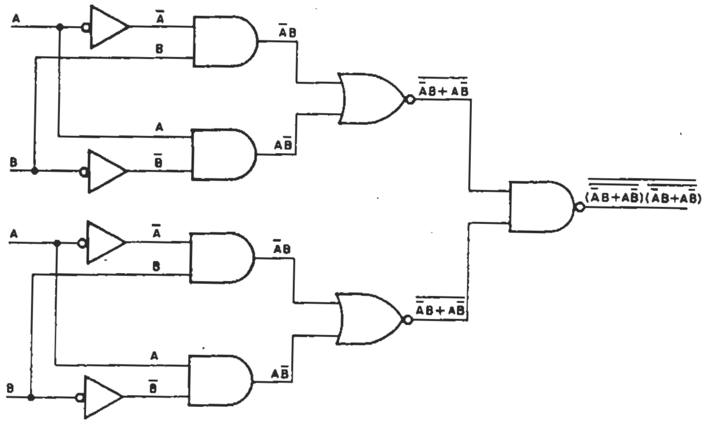


Student Problems

The following logic diagrams may be used as student problems.

Label the output expressions for each gate shown.





A
$\bar{A}$
B
$\bar{A}B$
$\overline{\bar{A}B+A\bar{B}}$
A
$A\bar{B}$
B
$\bar{B}$
$\overline{\overline{(\bar{A}B+A\bar{B})}\,\overline{(\bar{A}B+A\bar{B})}}$
A
$\bar{A}$
B
$\bar{A}B$
$\overline{\bar{A}B+A\bar{B}}$
A
$A\bar{B}$
B
$\bar{B}$

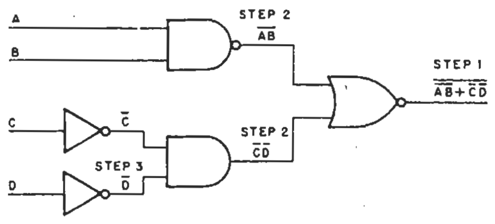
Rules For Forming Logic Diagrams

The following rules summarize how to form a logic diagram from a Boolean expression.

a. Begin constructing the diagram at the right and work left until all inputs are single letters.

b. Never separate letters within a group until the group has been separated from the other groups in the expression.

c. If a vinculum extends over more than one letter, use an inversion symbol to remove it.

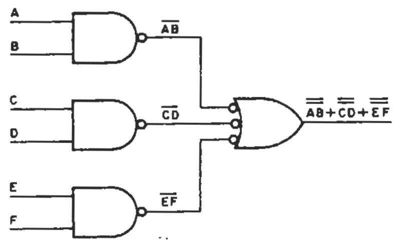


Student Problem

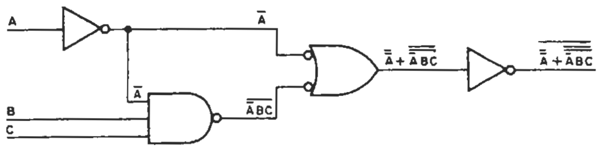
The following examples may be used as student problems.

Draw the logic diagrams for the following Boolean expressions:

$$\overline{\overline{AB}} + \overline{\overline{CD}} + \overline{\overline{EF}}$$

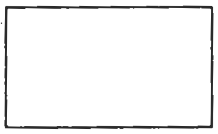

A
B
$\overline{AB}$
C
D
$\overline{CD}$
$\overline{\overline{AB}}+\overline{\overline{CD}}+\overline{\overline{EF}}$
E
F
$\overline{EF}$

$$\overline{\overline{\overline{A}} + \overline{\overline{\overline{A}BC}}}$$

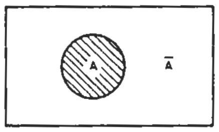

A
$\overline{A}$
$\overline{\overline{A}}+\overline{\overline{\overline{A}BC}}$
$\overline{\overline{\overline{A}}+\overline{\overline{\overline{A}BC}}}$
$\overline{A}$
B
C
$\overline{\overline{A}BC}$

II. VENN DIAGRAMS

Venn diagrams graphically represent logical statements using geometric diagrams. A rectangle is used to represent a class or group possessing certain defined characteristics.

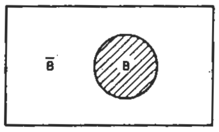

CLASS TO BE REPRESENTED

A subclass is represented by a circle within the rectangle. Variable A, a subclass of the rectangle, is illustrated below. The shaded area represents the fact that we are concerned only with the variable A. $\overline{A}$ is the unshaded area.



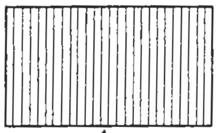
SUBCLASS A

Subclass B is represented by a circle in the rectangle shown below. The shaded area denotes that we are concerned with the variable B. $\overline{B}$ is the unshaded area.



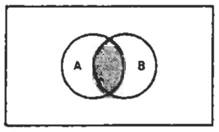
SUBCLASS B

A rectangle completely shaded illustrates that we are concerned with all members of the rectangle class; such a situation is numerically represented by 1.


1

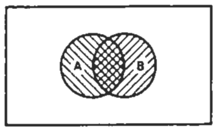
Because we have defined all possible points of consideration as a rectangle, when we have no points of consideration, there can be no rectangle. Such a situation is represented numerically by zero (0).

The expression A·B (A and B) is represented by the shaded area in the diagram below and is obtained by superimposing the diagram for A on the diagram for B and shading only those areas common to both. Thus A·B denotes only those elements which are present in both class A and class B.



A·B

The expression A + B (A or B) is represented graphically by the shaded area in the diagram below. It is obtained by superimposing the diagram for A on the diagram for B and shading A and B in their entirety. Thus the expression A + B includes all elements of class A, all elements of class B, and the elements common to both classes.



A+B

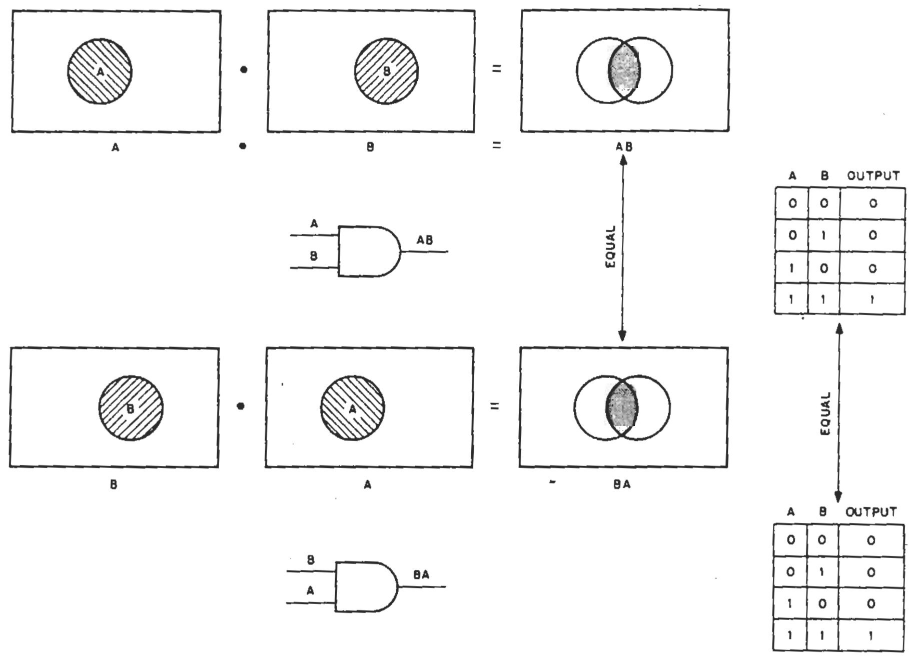
Summary of Boolean Laws

a. Commutative Law: AB = BA - Part 1

A + B = B + A - Part 2

When inputs to a logic symbol are ANDed or ORed, the order in which they are written does not affect the value of the output.

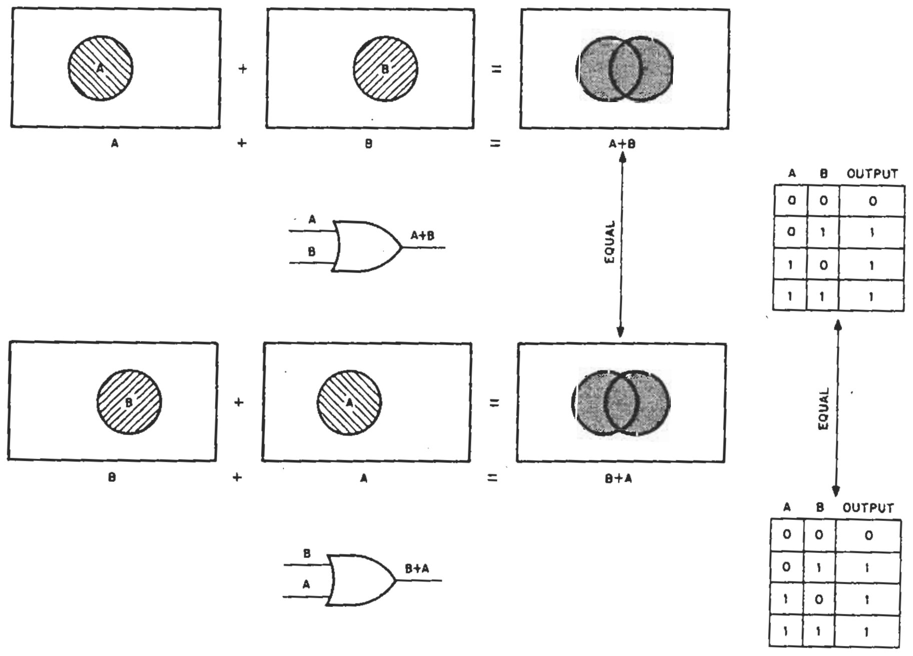
AB = BA   Part 1



A	B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

A	B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

A + B = B + A   Part 2



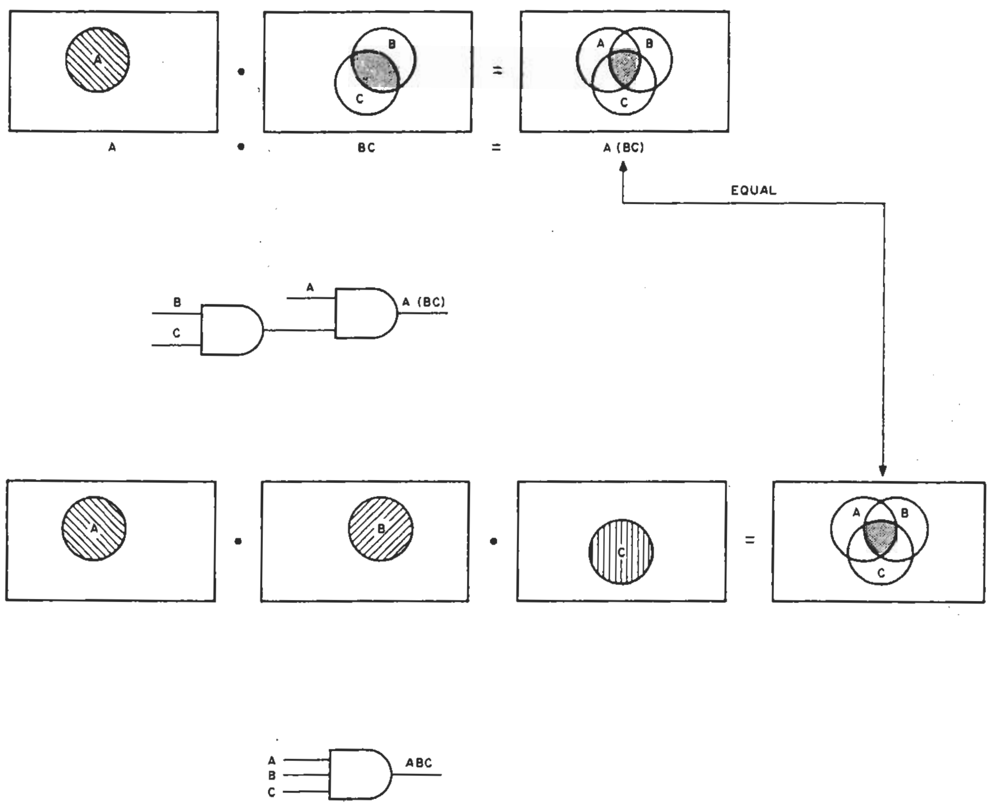
A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

b. Associative Law: A(BC) = ABC - Part 1

A + (B + C) = A + B + C - Part 2

A (BC) = ABC Part 1

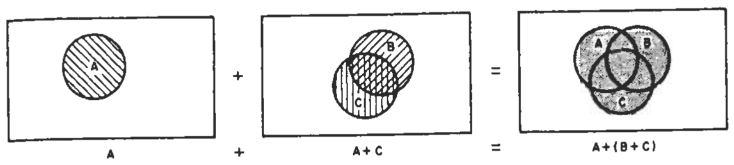


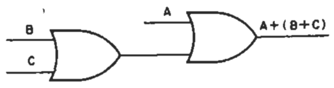
A	B	C	OUTPUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

EQUAL

A	B	C	OUTPUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

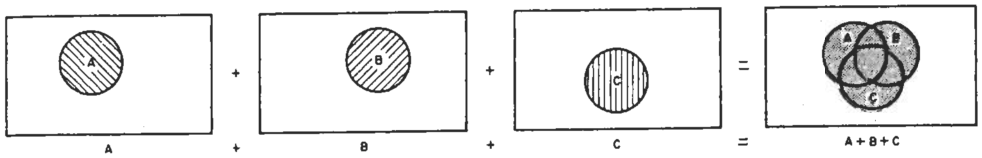
A + (B + C) = A + B + C   Part 2



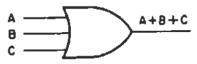


A	B	C	OUTPUT
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

EQUAL



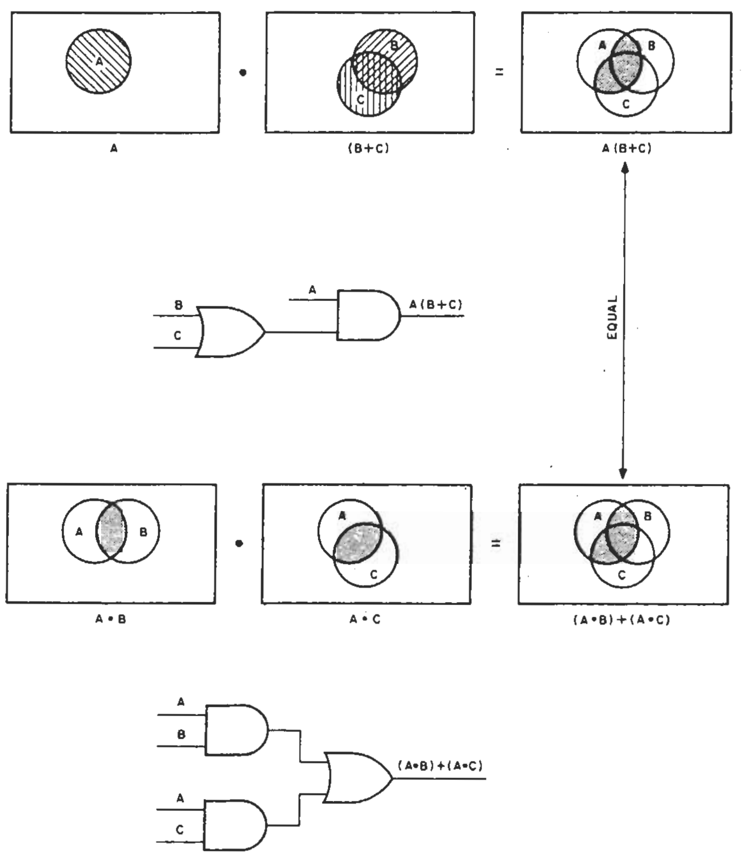
A	B	C	OUTPUT
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



c. Distributive Law: $A(B + C) = (A \cdot B) + (A \cdot C)$ - Part 1

$A + BC = (A + B)(A + C)$ - Part 2

$A(B + C) = (A \cdot B + (A \cdot C))$ Part 1

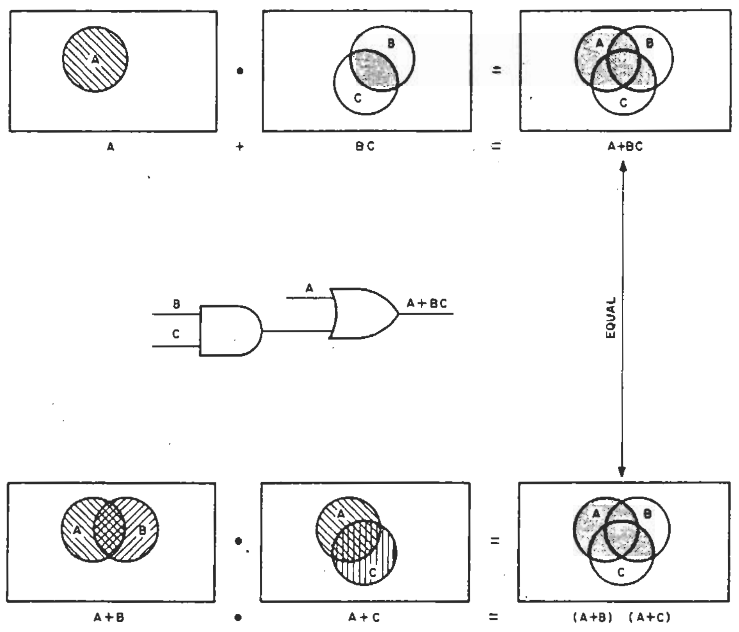


A	B	C	OUTPUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

EQUAL

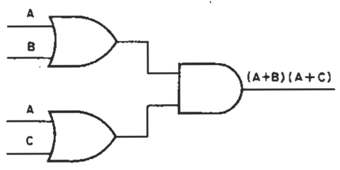
A	B	C	OUTPUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

A + BC = (A + B) (A + C)   Part 2



A	B	C	OUTPUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

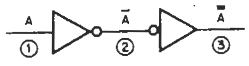
A	B	C	OUTPUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



Summary of Boolean Identities

Identity 1: $\overline{\overline{A}} = A$

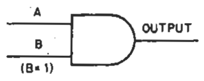
(Page 48, COMPUTER LAB Workbook) A double bar over any variable or group of variables can be eliminated. This operation is illustrated below. As illustrated in the truth table, if point 1 is a LO, point 3 will be LO. If point 1 is HI, point 3 is HI. Thus, point 1, or A, is equal to point 3, or $\overline{\overline{A}}$.



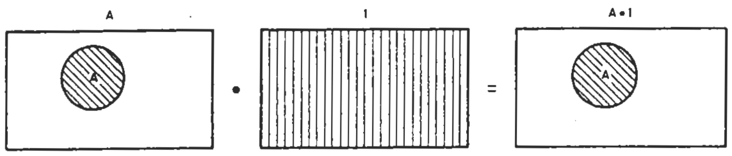
①	②	③
LO	HI	LO
HI	LO	HI

Identity 2: $A \cdot 1 = A$

When using the symbol 1 we are saying that all conditions concerned are always true. If we AND A with 1, the resultant output condition is dependent on A.



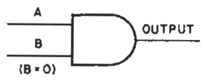
A	B	OUTPUT
0	1	0
1	1	1



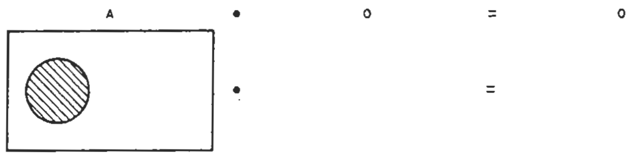
(When ANDing Rectangles, Shade Common Areas.)

Identity 3: $A \cdot 0 = 0$

When using the symbol 0 we are saying that we are not concerned with any conditions or all conditions are false. If we AND A with 0, the resulting output condition is 0.

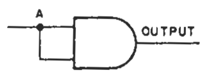


A	B	OUTPUT
0	0	0
1	0	0

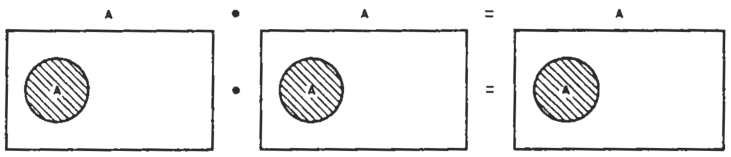


Identity 4: $A \cdot A = A$

A ANDed with A is illustrated below as connecting the inputs to an AND gate to a common source (A). If A is true, the output is true. If A is false, the output is false.

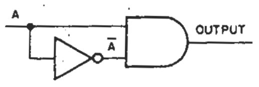


A	A	OUTPUT
1	1	1
0	0	0

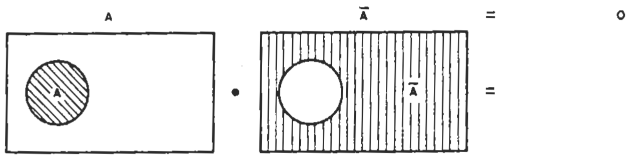


Identity 5: $A \cdot \overline{A} = 0$

If A is ANDed with $\overline{A}$, the resultant output is 0. Since A and $\overline{A}$ cannot be true simultaneously, the AND function can never be enabled.



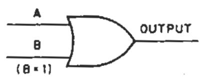
A	$\overline{A}$	OUTPUT
1	0	0
0	1	0



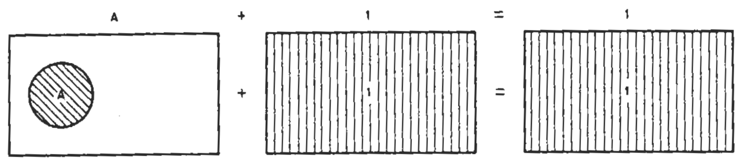
(When ANDing rectangles, shade only common points.)

Identity 6: $A + 1 = 1$

If A is ORed with 1, the resultant output is always true. Since the input labeled 1 is always true in the OR gate illustrated below, the output must always be true.



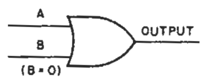
A	B	OUTPUT
0	1	1
1	1	1



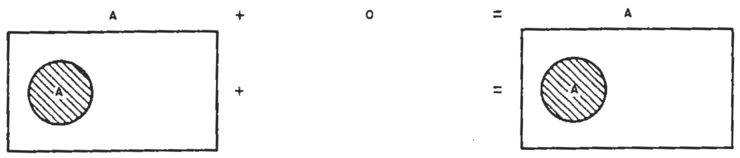
(When rectangles are ORed, shade all points of consideration.)

Identity 7: $A + 0 = A$

A, ORed with an input that is never true (0), is controlled by A.

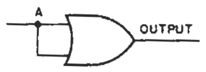


A	B	OUTPUT
1	0	1
0	0	0

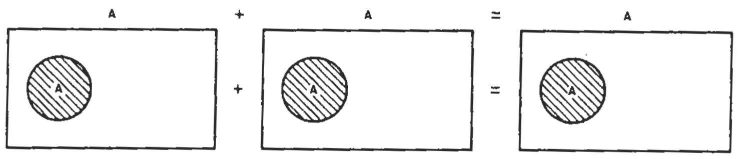


Identity 8: $A + A = A$

A ORed with A is illustrated below as an OR gate which has inputs originating from a single source. If A is true, the output is true; if A is false, the output is false.

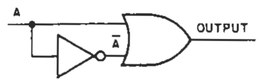


A	A	OUTPUT
1	1	1
0	0	0

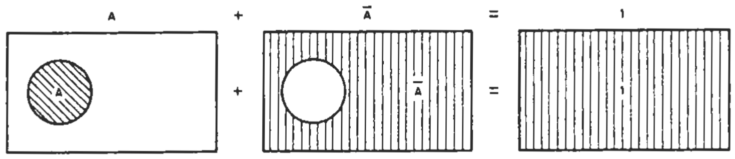


Identity 9: $A + \overline{A} = 1$

If A is ORed with $\overline{A}$ the output will always be true, because either A or $\overline{A}$ must be true.

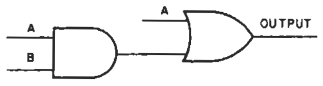


A	$\overline{A}$	OUTPUT
1	0	1
0	1	1

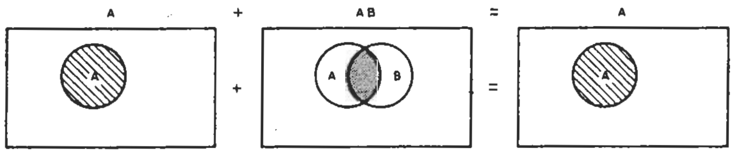


Identity 10: $A + AB = A$

When A is ORed with AB, the resultant output is controlled by A. In the logic diagram below, the output will be true if one or both of the inputs to the OR gate are true. A HI at A will directly enable the OR gate; a LO at A will cause both inputs to the OR gate to go LO, thus disabling it. The state of the B input will not affect the output.



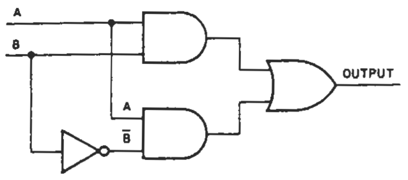
A	B	OUTPUT
0	0	0
0	1	0
1	0	1
1	1	1



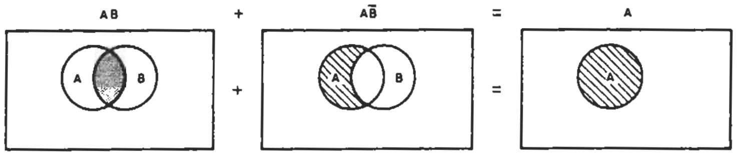
$A + AB$		
$= A(1 + B)$	Distributive Law	Numbers refer to identities
$= A(1)$	#6, $1 + B = 1$	
$= A$	#2, $A \cdot 1 = A$	

Identity 11: $AB + A\overline{B} = A$

When AB and $A\overline{B}$ are ORed, the output is dependent upon the value of A. Because either B or $\overline{B}$ must be true, one of the AND gates will be enabled (and therefore the OR gate will be enabled) when A is true. When A is false, the function cannot be enabled.



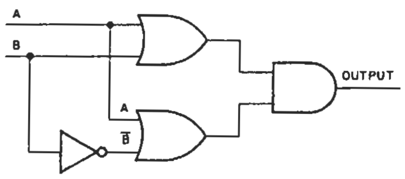
A	B	OUTPUT
0	0	0
0	1	0
1	0	1
1	1	1



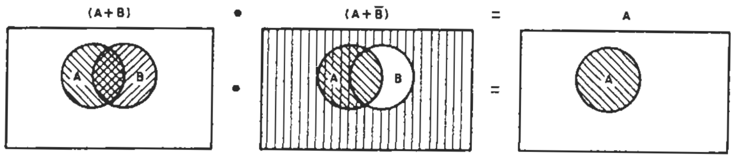
$$\begin{aligned} & AB + A\overline{B} \\ &= A(B + \overline{B}) && \text{Distributive Law} \\ &= A(1) && \#9, A + \overline{A} = 1 \\ &= A && \#2, A \cdot 1 = A \end{aligned}$$

Identity 12: $(A + B)(A + \overline{B}) = A$

The logic diagram below illustrates the expression $(A + B)(A + \overline{B})$. The output of this function is dependent upon the value of A. At a given time either B or $\overline{B}$ must be true. When A is HI both OR gates will be enabled, the AND gate will be enabled, and the output will be HI. When A is LO, only one of the inputs to the AND gate is present and the output will be LO.



A	B	OUTPUT
0	0	0
0	1	0
1	0	1
1	1	1

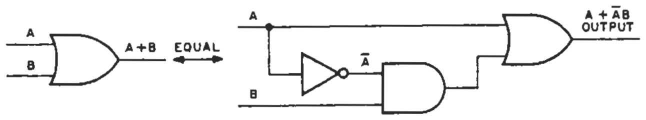


$(A + B)(A + \overline{B})$

$= A + (B \cdot \overline{B})$	Distributive Law
$= A + 0$	#5, $A \cdot \overline{A} = 0$
$= A$	#7, $A + 0 = A$

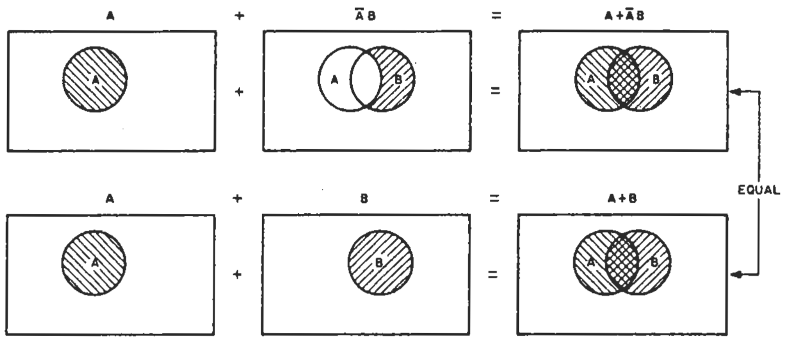
Identity 13: $A + \overline{A}B = A + B$

When A and $\overline{A}B$ are ORed, the output depends upon the values of A and B. The following logic diagram illustrates the expression $A + \overline{A}B$. At any given time, either A or $\overline{A}$ must be true. If A is HI, the OR gate will be enabled and the output will be HI. If A is LO and B is HI, the AND gate will be enabled by two HIs and, therefore, the output of the OR gate will be HI. If A and B are LOs, the output will be LO.



EQUAL

A	B	$A+\overline{A}B$	A+B
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

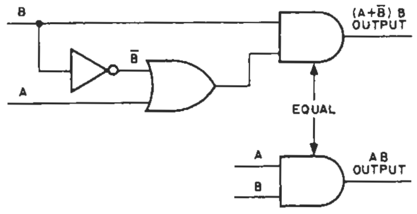


$A + \overline{A}B$

$= (A + \overline{A})(A + B)$	Distributive Law
$= 1(A + B)$	#9, $A + \overline{A} = 1$
$= A + B$	#2, $A \cdot 1 = A$

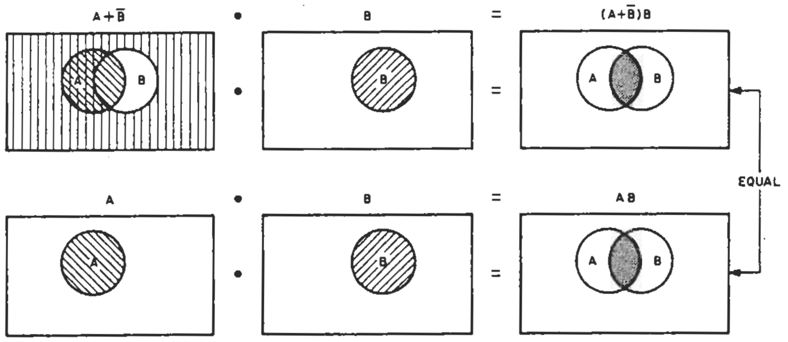
Identity 14: $(A + \overline{B})B = AB$

The logic diagram for $(A + \overline{B})B$ is illustrated below. Both inputs to the AND gate must be HI to enable the gate. Only when both the A and the B inputs are true will the output be true.



EQUAL

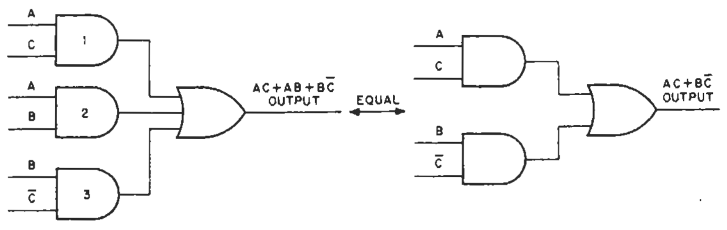
A	B	$(A+\overline{B})B$	AB
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1



$$\begin{aligned} &(A + \overline{B})B \\ &= AB + \overline{B}B && \text{Distributive Law} \\ &= AB + 0 && \#5,\ A\overline{A} = 0 \\ &= AB && \#7,\ A + 0 = A \end{aligned}$$

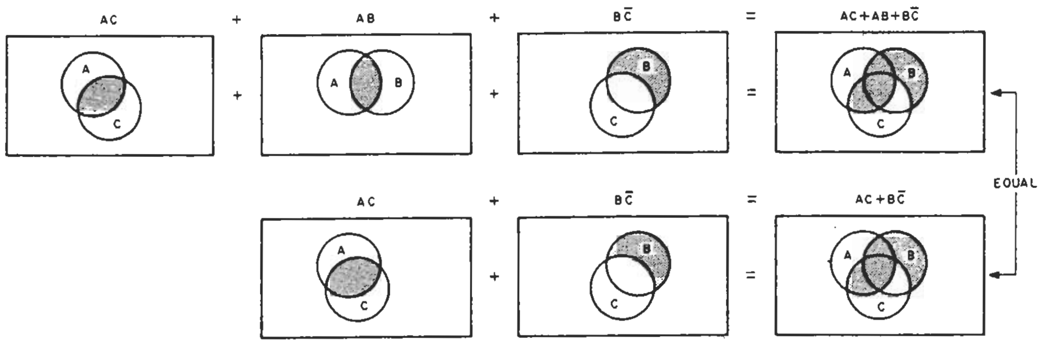
Identity 15: $AC + AB + B\overline{C} = AC + B\overline{C}$.

The logic diagram for $AC + AB + B\overline{C}$ follows. At any given time, either C or $\overline{C}$ must be true. To get a HI output out of the OR gate, at least one of the inputs from the AND gates must be HI. When A and B are true, the input to the OR gate from gate 2 would make the final output true. However, when A and B are true, either AC or $B\overline{C}$ must also be true. As only one HI input is required to enable the OR gate, the input AB is not necessary to the determination of the output.



EQUAL

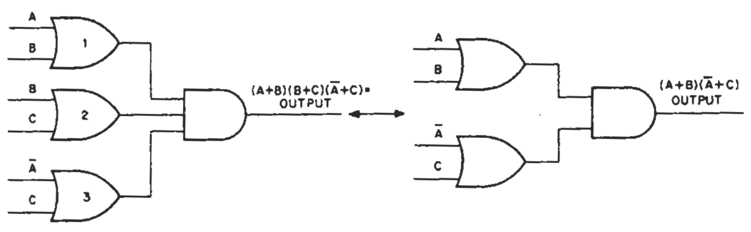
A	B	C	AC+AB+B$\overline{C}$	AC+B$\overline{C}$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	0	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



$AC + AB + B\overline{C}$
$= AC + AB(1) + B\overline{C}$ #2, $A \cdot 1 = A$
$= AC + AB(C + \overline{C}) + B\overline{C}$ #9, $A + \overline{A} = 1$
$= AC + ABC + AB\overline{C} + B\overline{C}$ Distributive Law
$= AC\,(1 + B) + B\overline{C}\,(A + 1)$ Distributive Law
$= AC\,(1) + B\overline{C}\,(1)$ #6, $A + 1 = 1$
$= AC + B\overline{C}$ #2, $A \cdot 1 = A$

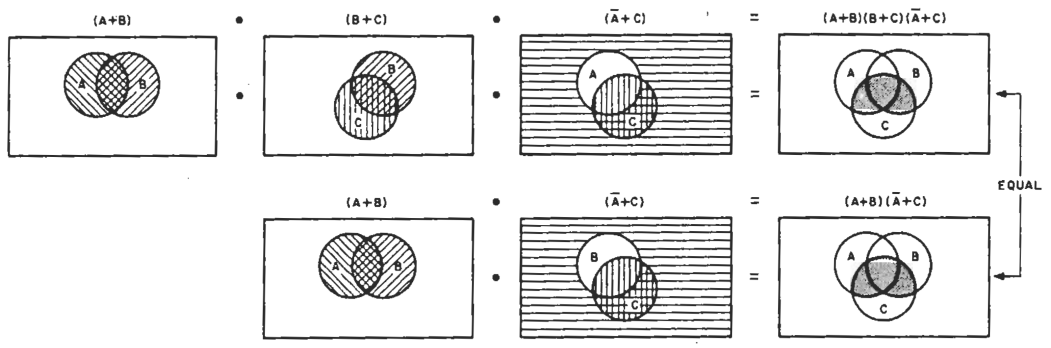
Identity 16: $(A+B)\,(B+C)\,(\overline{A}+C) = (A+B)\,(\overline{A}+C)$

The logic diagram which follows illustrates the expression $(A+B)\,(B+C)\,(\overline{A}+C)$. For the output to be true, all inputs to the AND gate must be true. Each of these inputs will be HI when at least one of the inputs to each OR gate is HI. However, A and $\overline{A}$ cannot be HI simultaneously. If A is HI, gate 1 will be enabled; $\overline{A}$ will be LO and input C must therefore be HI to enable gate 3 (Cs input being HI would also enable gate 2). If A is LO, input B must be HI if gate 1 is to be enabled (a HI B will also enable gate 2). Thus (B+C) will be necessarily true when $(A+B)\,(\overline{A}+C)$ is true, and therefore is redundant.



EQUAL

A	B	C	(A+B)(B+C)(Ā+C)	(A+B)(Ā+C)
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1



$(A+B)(B+C)(\bar{A}+C)$	
$= \overline{\bar{A}\bar{B} + \bar{B}\bar{C} + A\bar{C}}$	DeMorgan's Theorem
$= \overline{\bar{A}\bar{B} + \bar{B}\bar{C}(1) + A\bar{C}}$	#2, A·1 = A
$= \overline{\bar{A}\bar{B} + \bar{B}\bar{C}(A+\bar{A}) + A\bar{C}}$	#9, A + Ā = 1
$= \overline{\bar{A}\bar{B} + \bar{B}\bar{C}A + \bar{B}\bar{C}\bar{A} + A\bar{C}}$	Distributive Law
$= \overline{\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B} + A\bar{C}\bar{B} + A\bar{C}}$	Commutative
$= \overline{(\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}) + (A\bar{C}\bar{B} + A\bar{C})}$	Associative
$= \overline{\bar{A}\bar{B}(\bar{C}+1) + A\bar{C}(\bar{B}+1)}$	Distributive
$= \overline{\bar{A}\bar{B}(1) + A\bar{C}(1)}$	#6, A + 1 = 1
$= \overline{\bar{A}\bar{B} + A\bar{C}}$	#2, A·1 = A
$= (\overline{\bar{A}\bar{B}})(\overline{A\bar{C}})$	DeMorgan's Theorem
$= (A+B)(\bar{A}+C)$	DeMorgan's Theorem

ANSWER KEY

Questions

1.

A	B	C	A+B+C	A·B·C	$A \oplus B \oplus C$	$A \equiv B \equiv C$
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
1	0	0	1	0	1	0
1	0	1	1	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

2.

											(12)	(11)	(15)	
A	B	C	AC	AB	$\bar{C}$	$\bar{B}$	$A\bar{B}$	$B\bar{C}$	A+B	$A+\bar{B}$	$(A+B)(A+\bar{B})$	$AB+A\bar{B}$	$AC+AB+B\bar{C}$	$AC+B\bar{C}$
0	0	0	0	0	1	1	0	0	0	1	0	0	0	0
0	0	1	0	0	0	1	0	0	0	1	0	0	0	0
0	1	0	0	0	1	0	0	1	1	0	0	0	1	1
0	1	1	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	1	1	1	0	1	1	1	1	0	0
1	0	1	1	0	0	1	1	0	1	1	1	1	1	1
1	1	0	0	1	1	0	0	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	1	1	1	1	1	1

11. $AB+A\bar{B} = A$ 12. $(A+B)\,(A+\bar{B}) = A$ 15. $AC+AB+B\bar{C} = AC+B\bar{C}$

3. $(A + \overline{B})\,B = AB$

$AB + B\overline{B}$	Distributive Law
$B\overline{B} = 0$	Identity 5
$AB + 0 = AB$	Identity 7

4. F = Bowler wins game when true
$\overline{F}$ = Bower loses game when true
A = Pin A up when true
B = Pin B up when true
C = Pin C up when true

$F = A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}\,\overline{C}B + \overline{A}\,\overline{B}\,\overline{C}$
$\overline{F} = \overline{A}BC + AB\overline{C} + AC\overline{B} + ABC$

5.a. Assume that all factory lights are turned on previous to 5:00 p.m.

A = Factory A lights out (5:30)	$\overline{A}$ = Factory A lights on
B = Factory B lights out (6:00)	$\overline{B}$ = Factory B lights on
C = Factory C lights out (5:00)	$\overline{C}$ = Factory C lights on
D = Factory D lights out (5:15)	$\overline{D}$ = Factory D lights on
E = Factory E lights out (5:45)	$\overline{E}$ = Factory E lights on

F_A = True function between 5:15 and 5:45 p.m.
$F_A = (D)\,(\overline{E})$

5.b. F_B = True function between 5:45 to 6:00 p.m.
$F_B = E\overline{B}$

5.c. F_C = True function between 5:00 and 5:15 p.m. and 5:45 and 6:00 p.m.
$F_C = C\overline{D} + E\overline{B}$

6.a. $\overline{A} \cdot \left\{\overline{[BC+D]\ \overline{A}}\right\}$

$\overline{A + [BC+D]\ \overline{A}}$	DeMorgan's Theorem
$\overline{A + \overline{A}BC+\overline{A}D}$	Distributive Law
$\overline{A + BC+\overline{A}D}$	Identity #13
$\overline{A + BC+D}$	Identity #13
$\overline{A} \cdot \overline{BC} \cdot \overline{D}$	DeMorgan's Theorem

6.b. $\overline{\overline{A}\ \overline{B} + \overline{(A+\overline{B})}}$

$A + B\,(A+\overline{B})$	DeMorgan's Theorem
$A + AB + B\overline{B}$	Distributive Law
$A + AB$	Identity #5
A	Identity #10

6.c. $(A + AB) + \overline{A}B$

$A + \overline{A}B$	Identity #10
$A + B$	Identity #13

6.d	$(\overline{A} + \overline{B})\,(A\overline{B})$	
	$A\overline{A}\overline{B} + A\overline{B}\overline{B}$	Distributive Law
	$A\overline{B}$	Identity #4
6.e.	$[(AB + A\overline{B}) + AB] + \overline{A}B$	
	$[AB + A\overline{B} + AB] + \overline{A}B$	Associative Law
	$AB + AB + A\overline{B} + \overline{A}B$	Associative Law
	$AB + A\overline{B} + \overline{A}B$	Identity #8
	$A(B + \overline{B}) + \overline{A}B$	Distributive Law
	$A + \overline{A}B$	Identity #2 and #9
	$A + B$	Identity #13

7.

A	B	C	D	$\overline{A}$	$\overline{B}$	$\overline{C}$	$\overline{D}$	7A	7B	7C	7D
0	0	0	0	1	1	1	1	0	1	1	1
0	0	0	1	1	1	1	0	0	1	1	1
0	0	1	0	1	1	0	1	0	0	1	1
0	0	1	1	1	1	0	0	0	0	1	1
0	1	0	0	1	0	1	1	0	1	1	1
0	1	0	1	1	0	1	0	0	1	0	1
0	1	1	0	1	0	0	1	1	0	1	0
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	0	1	1	1	0	0	1	1
1	0	0	1	0	1	1	0	1	0	1	1
1	0	1	0	0	1	0	1	0	1	1	1
1	0	1	1	0	1	0	0	1	1	1	1
1	1	0	0	0	0	1	1	0	0	1	0
1	1	0	1	0	0	1	0	1	0	0	0
1	1	1	0	0	0	0	1	1	1	1	1
1	1	1	1	0	0	0	0	1	1	0	1

7.a. $(\overline{A}BC\overline{D}) + (\overline{A}BCD) + (A\overline{B}\overline{C}D) + (A\overline{B}CD) + (AB\overline{C}D) + (ABC\overline{D}) + (ABCD) = BC + AD$

7.b. $\overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}C + ABC = \overline{A}\overline{C} + AC$

7.c. $\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \overline{A}B\overline{C}\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}C\overline{D} +$
$A\overline{B}CD + AB\overline{C}\overline{D} + ABC\overline{D} + A\overline{B}\overline{C}D = \overline{B} + \overline{D}$

7.d. $\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + A\overline{B}C + ABC = \overline{B} + AC + \overline{A}\overline{C}$

Experiment 4.1 Gating Circuit Simplification

Figure 4.18

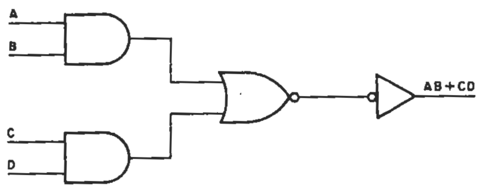
The unsimplified circuit output expression is $\overline{\overline{A\,(\overline{A\overline{B}})} \cdot \overline{CD}}$ and it can be simplified as follows:

$A\,(\overline{A\overline{B}}) + CD$

$A\,(\overline{A} + B) + CD$

$A\overline{A} + AB + CD$

$AB + CD$



(1 = HI)

A	B	C	D	$\overline{\overline{A\,(\overline{A\overline{B}})} \cdot \overline{CD}}$	AB + CD
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

Figure 4.19

The unsimplified circuit output expression is

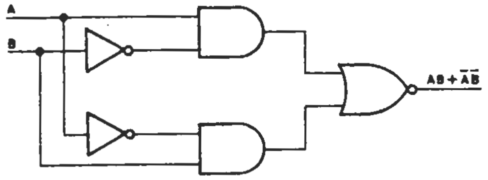
$(AB + \overline{A}\,\overline{B}) + \overline{\overline{A}B + A\overline{B}}$ and it can be simplified as follows:

$AB + \overline{A}\,\overline{B} + (A + \overline{B})\ (\overline{A} + B)$

$AB + \overline{A}\,\overline{B} + A\overline{A} + AB + \overline{A}\,\overline{B} + B\overline{B}$

$AB + \overline{A}\,\overline{B} + AB + \overline{A}\,\overline{B}$

$AB + \overline{A}\,\overline{B}$

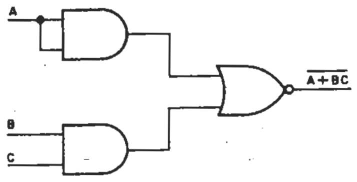


A	B	$(AB + \overline{A}\,\overline{B}) + \overline{\overline{A}B + A\overline{B}}$	$AB + \overline{A}\,\overline{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	1	1

Figure 4.20

The unsimplified circuit output expression is $\overline{\overline{A}BC + A}$ and it can be simplified as follows:

$\overline{A + BC}$



A	B	C	$\overline{\overline{A}BC + A}$	$\overline{A + BC}$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

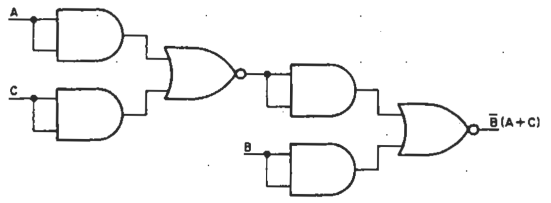
Figure 4.21

The unsimplified circuit output expression is $\overline{B}\,(\overline{B}C + A\overline{B})$ and it can be simplified as follows:

$\overline{B}\overline{B}C + A\overline{B}\overline{B}$

$\overline{B}C + A\overline{B}$

$\overline{B}\,(A + C)$



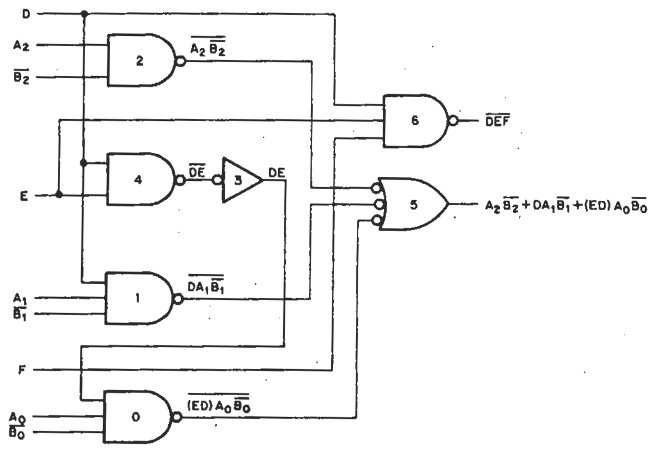
A	B	C	$\overline{B}\,(\overline{B}C + A\overline{B})$	$\overline{B}\,(A + C)$
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	0	0	0
0	0	1	1	1
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0

8. (Refer to Figures 4.22 and 4.23.) The Equality and Relative Magnitude Detector is divided into two sections. The first section determines if corresponding bits of two numbers being compared are equal. The second section determines which of the two numbers is greater if an inequality exists.

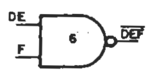
The 3-bit Equality Detector of Figure 4.22 is divided into three comparators. Outputs D, E, and F, when HI, indicate $A_2 = B_2$, $A_1 = B_1$, and $A_0 = B_0$, respectively.

The Relative Magnitude Detector has two outputs: $\overline{A = B}$ and $A > B$. If the number A is not equal to the number B, $\overline{A = B}$ will be HI. If A is greater than B, the output $A > B$ will be HI. If A is not greater than B, the output $A > B$ will be LO. Gate 2 inputs are A_2, HI when A_2 is equal to a 1, and $\overline{B_2}$, LO when B_2 is equal to a 1. The output of gate 2 will be LO when $A_2 > B_2$. Gate 1 inputs are $A_2 = B_2$, A_1, and $\overline{B_1}$. If $A_2 = B_2$ (input D is HI), and $A_1 = B_1$ (input E is HI), gate 0 will be enabled by gate 3. If $A_0 > B_0$, gate 0's output will be LO. Gate 5's output will be HI if $A_2 > B_2$, or $A_1 > B_1$, or $A_0 > B_0$. Gate 6's output will be LO if $A = B$ ($A_2 = B_2$, $A_1 = B_1$, and $A_0 = B_0$).

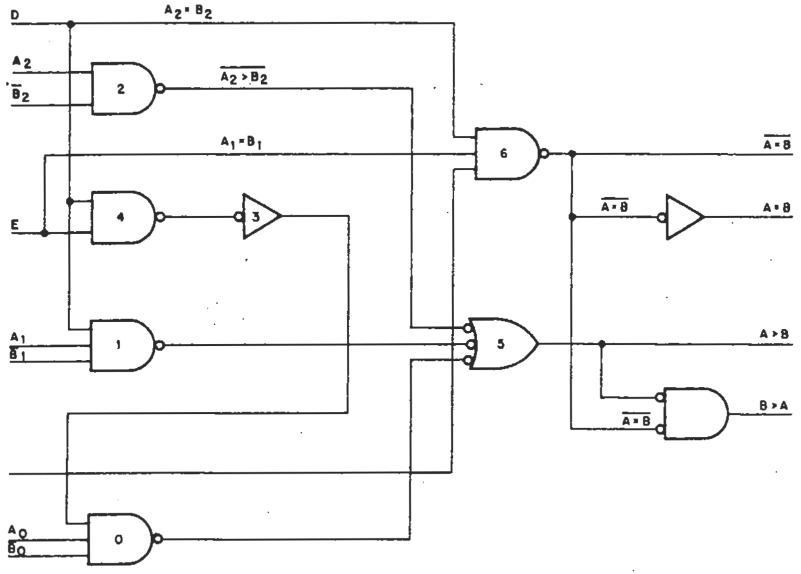
9.


D
A2
$\overline{B_2}$
2
$\overline{A_2 \overline{B_2}}$
6
$\overline{DEF}$
4
$\overline{DE}$
3
DE
E
5
$A_2\overline{B_2} + DA_1\overline{B_1} + (ED)A_0\overline{B_0}$
1
$\overline{DA_1\overline{B_1}}$
A1
$\overline{B_1}$
F
0
$\overline{(ED)A_0\overline{B_0}}$
A0
$\overline{B_0}$

10. Refer to diagram, Question 9.

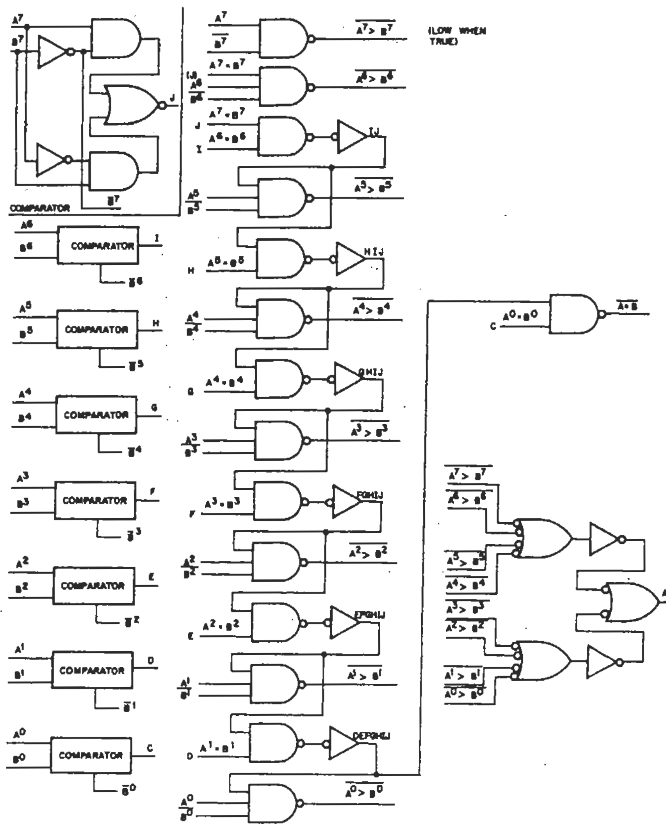


11.



Equality and Relative Magnitude Extension with Outputs A = B, A > B, B > A

12.



8-Bit Equality and Relative Magnitude Detector

Supplementary Questions

13.

A	B	C	$\overline{A}$	A+B+C	A·B·C	A⊕B⊕C	A=B=C
0	0	0	1	0	0	0	1
0	0	1	1	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	1	1	0	0	0
1	0	0	0	1	0	1	0
1	0	1	0	1	0	0	0
1	1	0	0	1	0	0	0
1	1	1	0	1	1	1	1

14. $A > B = A_2\overline{B}_2 + DA_1\overline{B}_1 + (ED)A_0\overline{B}_0$

15.

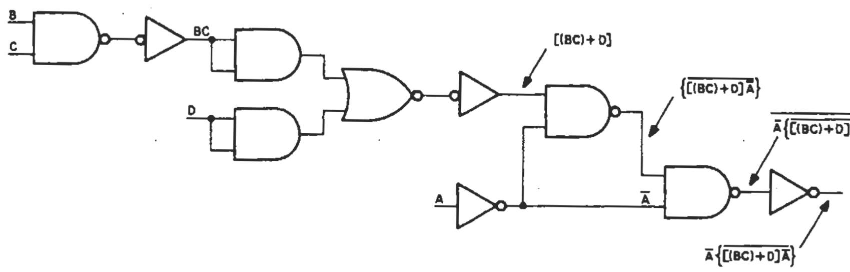
(LSD) (MSD) (HI = 1)

A_0	B_0	A_1	B_1	$A_0 = B_0$	$A_1 = B_1$	$A > B$	$\overline{A = B}$
0	0	0	0	1	1	0	0
0	0	0	1	1	0	0	1
0	0	1	0	1	0	1	1
0	0	1	1	1	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	1
1	0	0	0	0	1	1	1
1	0	0	1	0	0	0	1
1	0	1	0	0	0	1	1
1	0	1	1	0	1	1	1
1	1	0	0	1	1	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	1	0	0

16. In the Equality and Relative Magnitude Detector, Figures 4.22 and 4.23, the maximum delay for output $\overline{A = B}$ is 45×10^{-9} seconds, and the maximum delay for output $A > B$ is 90×10^{-9} seconds.

17.a. Unsimplified

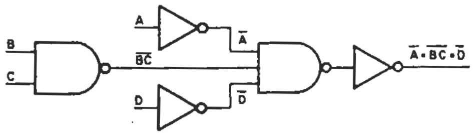
$$\overline{A}\bullet\left\{\overline{\left[(BC) + D\right]\bullet\overline{A}}\right\}$$



B
C
BC
D
A
$[(BC)+D]$
$\{\overline{[(BC)+D]\overline{A}}\}$
$\overline{\overline{A}\{\overline{[(BC)+D]\overline{A}}\}}$
$\overline{A}$
$\overline{A}\{\overline{[(BC)+D]\overline{A}}\}$

Simplified

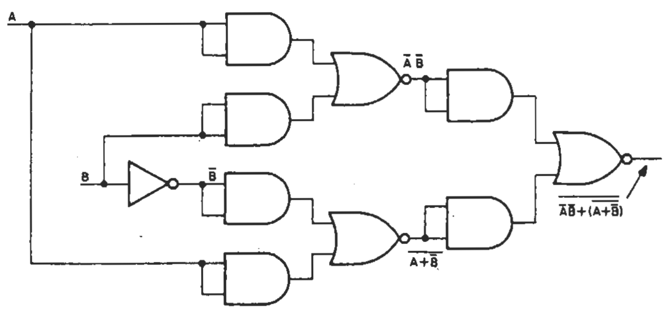
$$\overline{A} \bullet \overline{BC} \bullet \overline{D}$$



	Cost	Maximum Delay
Unsimplified	\$81.20	105 ns
Simplified	48.80	45 ns

17.b. Unsimplified

$$\overline{\overline{A}\overline{B} + \overline{(A + \overline{B})}}$$


A
B
B̄
ĀB̄
$\overline{A+\overline{B}}$
$\overline{\overline{A}\overline{B}+\overline{(A+\overline{B})}}$

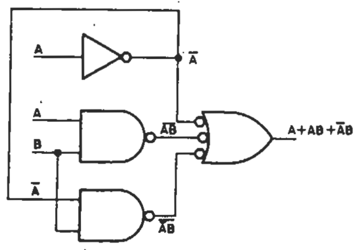
Simplified

A

	Cost	Maximum Delay
Unsimplified	$59.60	45 ns
Simplified	0	0

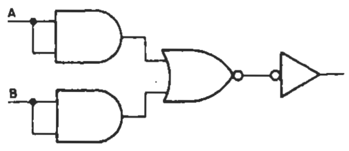
17.c. Unsimplified

$$A + AB + \overline{A}B$$



Simplified

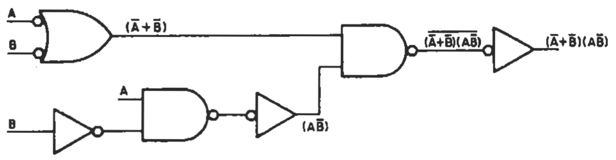
$A + B$



	Cost	Maximum Delay
Unsimplified	$30.60	45 ns
Simplified	26.00	30 ns

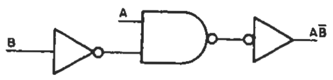
17.d. Unsimplified

$(\overline{A} + \overline{B})\ (A\overline{B})$


A
B
$(\overline{A}+\overline{B})$
A
B
$(A\overline{B})$
$\overline{(\overline{A}+\overline{B})(A\overline{B})}$
$(\overline{A}+\overline{B})(A\overline{B})$

Simplified

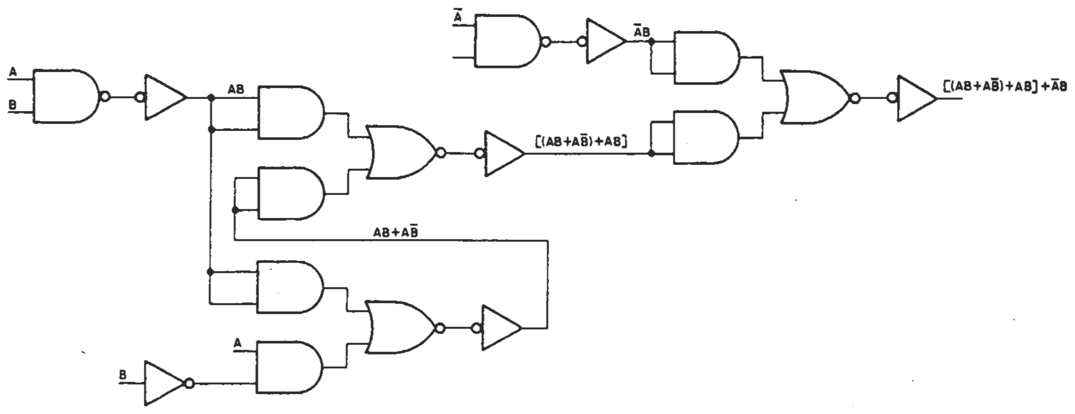
$A\overline{B}$



	Cost	Maximum Delay
Unsimplified	$55.20	75 ns
Simplified	27.60	45 ns

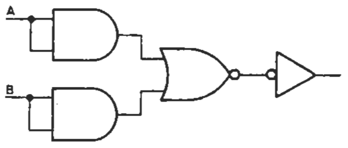
17.e. Unsimplified

$$\left[(AB + A\bar{B}) + AB\right] + \bar{A}B$$



Simplified

A + B



	Cost	Maximum Delay
Unsimplified	$124.00	120 ns
Simplified	26.00	30 ns

CHAPTER 5

BINARY COUNTERS

INTENT AND APPROACH

The objectives of this chapter are to teach the operation of synchronous and asynchronous counters and counter design.

In Chapter 5 the operation of various counter circuits is explained in detail. The explanations begin with a basic Asynchronous Binary Up Counter and proceed to a complex Recycling Modulo N Synchronous Up Counter. Through the experiments provided in the chapter, the student completes his understanding of each type of counter presented. With the exception of the more basic design problems, such as designing an 8-Bit Asynchronous Up Counter, it is suggested that the teacher select questions that require modifications of the explained logic circuits rather than those which have the student formulate completely new logic designs.

As explained in Chapter 5, there are rigorous mapping techniques, described in the reference texts for the COMPUTER LAB Workbook, which can be used to design counter circuits. The approach developed in Chapter 5 should be a design technique that contains the following steps.

1. List the count sequence or control requirements.
2. If the design is a counter, determine if it should be synchronous or asynchronous.
3. Taking into consideration the logic designs illustrated in Chapter 5, accomplish the requirements of steps 1 and 2.
4. Relying upon the knowledge of gate simplification gained from Chapter 4, simplify the gating circuitry.
5. Construct the logic circuit and verify its correct operation.

The approach outlined above requires a thorough knowledge of gates, flip-flops, counters, and simplification techniques. Additional techniques, such as mapping, should be presented as supplementary to the suggested method since the objectives of this chapter are not only to teach counter design, but also to reinforce knowledge acquired in all previous chapters.

OUTLINE

I. INTRODUCTION (Page 65)
   A. Binary Count Sequence
   B. Representation of Number Bits with J-K Flip-Flops

II. ASYNCHRONOUS COUNTERS (Pages 65-68)

A. Divisibility of Clock Pulses

B. Timing Diagram

C. Logic Diagram of 4-Bit Up Counter

D. Explanation of Operation of Up Counter

E. Experiment 5.1: Asynchronous Binary Up Counter (Page 67)

1. Circuit construction
2. Operation of counter
3. Truth table construction
4. Verification of operation
5. Modification of circuit to use COMPUTER LAB Clock
6. Explanation of term "asynchronous"

F. Experiment 5.2: Modified Asynchronous Binary Counter (Page 67)

1. Circuit construction
2. Operation of counter
3. Truth table construction

G. Questions 1-4 (Pages 67-68)

III. SYNCHRONOUS COUNTERS (Pages 68-69)

A. Comparison of Asynchronous and Synchronous Counters

B. Logic Diagram

C. Explanation of Operation of Synchronous Counters

D. Experiment 5.3: Synchronous Binary Up Counter (Page 68)

1. Circuit construction
2. Operation of counter
3. Truth table construction
4. Verification of operation

E. Questions 5-7 (Page 68)

IV. SYNCHRONOUS UP/DOWN COUNTER (Page 70)

A. Gating Networks

1. Up counting network
2. Down counting network

B. Experiment 5.4: Synchronous Binary Up/Down Counter (Page 70)

1. Logic diagram
2. Circuit construction

ANSWER KEY

Experiment 5.2

2^3	2^2	2^1	2^0	DECIMAL
0	0	0	0	0
1	1	1	1	15
1	1	1	0	14
1	1	0	1	13
1	1	0	0	12
1	0	1	1	11
1	0	1	0	10
1	0	0	1	9
1	0	0	0	8
0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1
0	0	0	0	0

Questions

1.

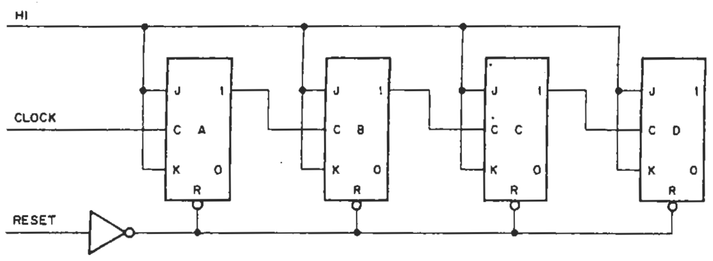


Figure 5.3 Asynchronous Up-Counter

1* A	2*	B		C		D		Clock	State Of			
								Pulses	L.S.B.	Counter		M.S.B.
MFF	SFF	MFF	SFF	MFF	SFF	MFF	SFF		A	B	C	D
0	0	0	0	0	0	1	1		0	0	0	1
1	0	0	0	0	0	1	1	1				
1	1	1	0	0	0	1	1		1	0	0	1
0	1	1	0	0	0	1	1	2				
0	0	1	1	1	0	1	1		0	1	0	1
1	0	1	1	1	0	1	1	3				
1	1	0	1	1	0	1	1		1	1	0	1
0	1	0	1	1	0	1	1	4				
0	0	0	0	1	1	0	1		0	0	1	1
1* Master Flip-Flop 2* Slave Flip-Flop												

The Asynchronous Binary Up Counter of Figure 5.3 consists of four J-K master-slave flip-flops. Each flip-flop is connected as a complementing flip-flop with both J and K connected to a constant HI; thus at clock time, each flip-flop changes state or complements. Each flip-flop consists of a master flip-flop and a slave flip-flop. At the leading edge of the clock pulse, the master flip-flop complements; on the trailing edge of the clock pulse, the slave flip-flop complements and the output also complements.

Assume that flip-flops A, B, and C are in the "0" state, and that flip-flop D is in the "1" state. On the leading edge of the next clock pulse (clock pulse 1), flip-flop A's master flip-flop complements or goes to the "1" state. Flip-flops B through D do not change state. On the trailing edge of clock pulse 1, slave flip-flop A, or output flip-flop A, will complement; thus flip-flop A will now be in the "1" state. The change in flip-flop A's output from LO to HI will cause the master flip-flop of flip-flop B to change but will produce no change in flip-flop B's output flip-flop. Therefore, there will be no change in flip-flop C and no change in flip-flop D. The state of the counter, starting from the least significant bit, is now 1001.

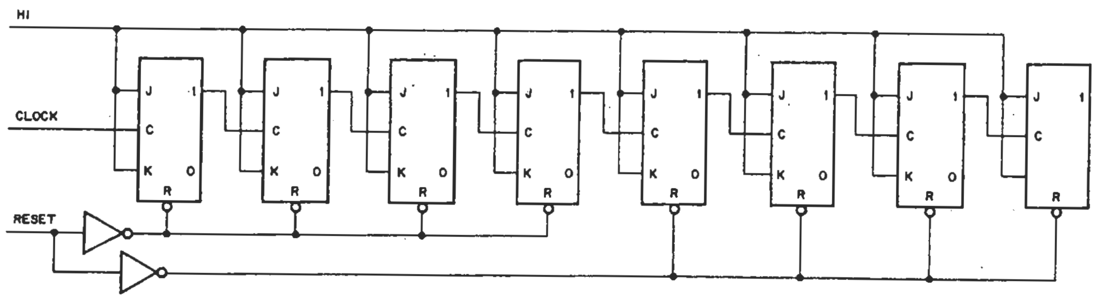
On the leading edge of clock pulse 2, the master flip-flop of flip-flop A will complement, going to the "0" state. However, there will be no change in the output of flip-flop A, and no change in flip-flops B through D. On the trailing edge of clock pulse 2, flip-flop A's output flip-flop will go to the "0" state, and the 1 output will go from HI to LO, causing the slave flip-flop of flip-flop B to go to the "1" state or to complement. Flip-flop B's 1 output going from LO to HI will cause the master flip-flop of flip-flop C to go to the "1" state, but there will be no output change from flip-flop C. With no output change from flip-flop C, there will be no output change in flip-flop D. The state of the counter, starting from the least significant bit, is now 0101.

At the leading edge of clock pulse 3, master flip-flop A will go to the "1" state. There will be no change in flip-flops B, C, or D. At the trailing edge of clock pulse 3, the slave or output of flip-flop A will switch to the "1" state. Flip-flop A's output change from LO to HI will cause master flip-flop B to go to the "0" state, but there will be no change in flip-flop B's output; thus there will be no change in flip-flop C or flip-flop D. The asynchronous up counter, starting from the least significant bit, is now 1101.

On the leading edge of clock pulse 4, master flip-flop A goes to the "0" state. There is no change in flip-flop B, C, or D. On the trailing edge of clock pulse 4, the output flip-flop A goes to the "0" state. With the change in the 1 output of flip-flop A from HI to LO, the slave flip-flop B goes to the "0" state. This causes the 1 output from flip-flop B to also go from HI to LO, which, in turn, will cause the slave flip-flop of flip-flop C to go to the "1" state. With flip-flop C going to the "1" state, its 1 output goes from LO to HI, causing the master flip-flop of flip-flop D to go to the "0" state. The count of the counter, starting from the least significant bit, is now 0011.

2. The modified counter in Figure 5.4 acts as a down counter.

3.

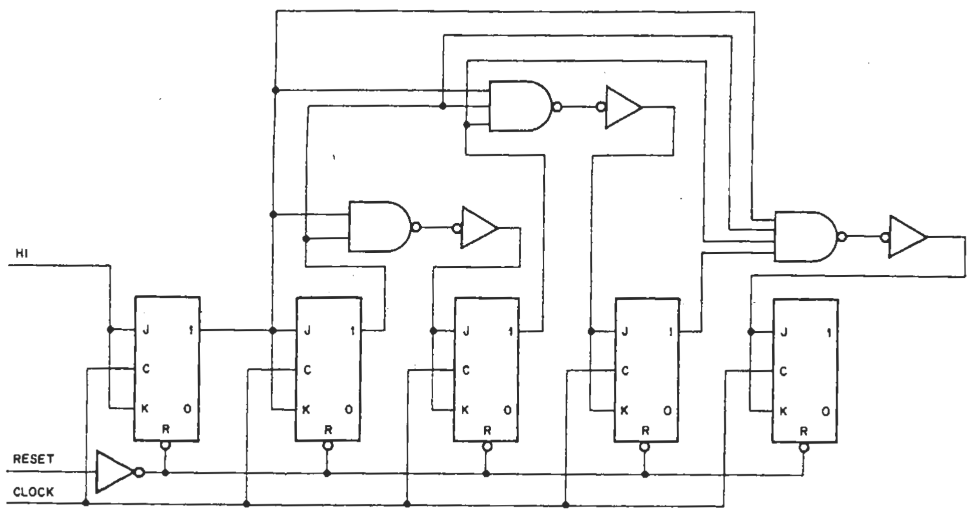


8-Bit Asynchronous Binary Up Counter

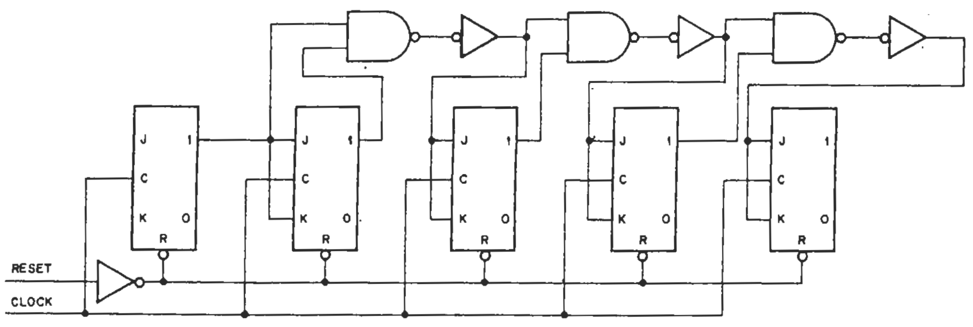
4. In the Asynchronous Binary Counter of Question 3, if the J and K inputs of the 2^0 flip-flop are connected to a rocker switch, when the rocker switch is placed in the HI position, the counter will be enabled and will count. With the rocker switch in the LO position, the

2^0 flip-flop will be disabled and the counter will be disabled; the counter will not function. Thus, connected in this manner, the rocker switch acts as a count enable switch.

5. The diagrams below illustrate two up counter configurations. The unsimplified counter can be operated faster than the simplified counter.

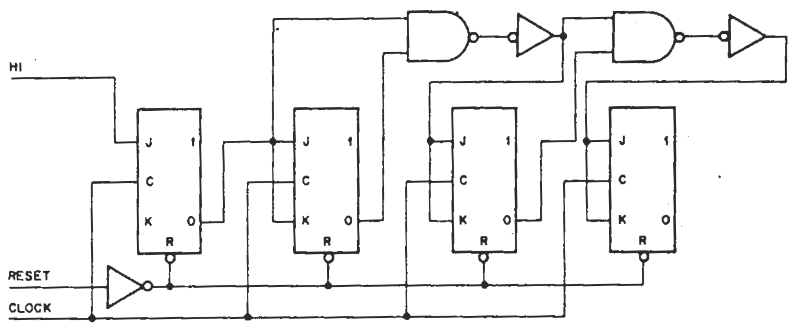


Unsimplified



Simplified

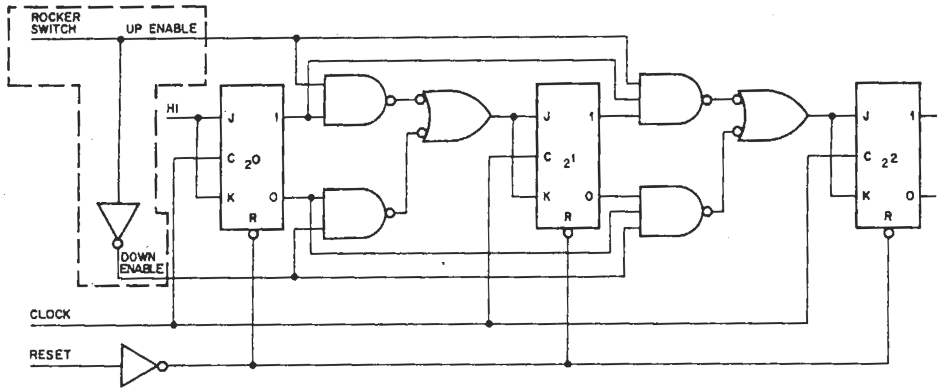
6.



4-Bit Synchronous Down Counter

7. The disadvantage of the asynchronous counter as compared to a synchronous counter is that, since information must ripple through from one flip-flop to another, the asynchronous counter has a higher propagation delay. The asynchronous counter is simpler to construct, compared to the synchronous counter, because the synchronous counter requires complex anticipatory logic.

8.

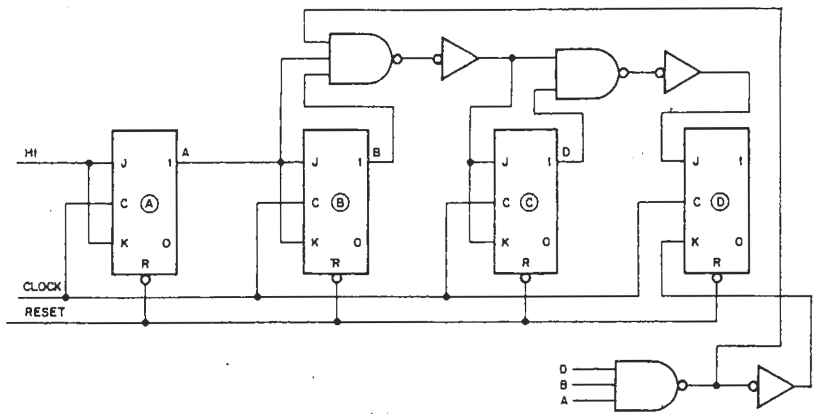


9. If a change of direction is made while the clock input is still HI, and propagation delay permits, the 2^1 flip-flop will complement on the trailing edge of the clock pulse; the 2^2 flip-flop will complement if the 2^0 and the 2^1 flip-flops were the same while the clock pulse was HI. In general, if a change of direction is made while the clock input is still HI, a false number will reside in the counter after the trailing edge of the clock pulse.

10. $0001_{(2)}$

11. If the Up Enable and Down Enable lines of the Synchronous Up/Down Counter are simultaneously enabled, all flip-flops which have all less significant bits (flip-flops) the same will complement with each clock pulse.

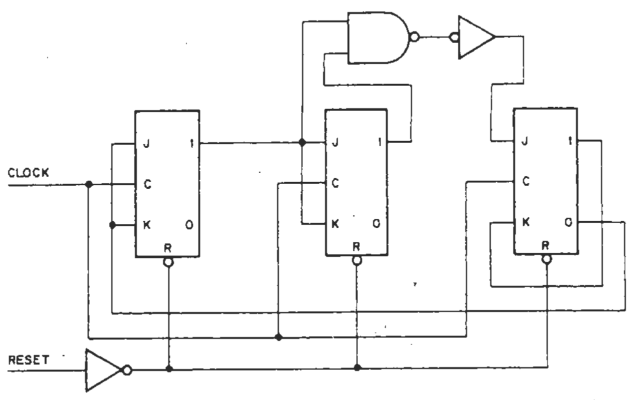
12.



Modulo 12 Counter

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
0	0	0	0

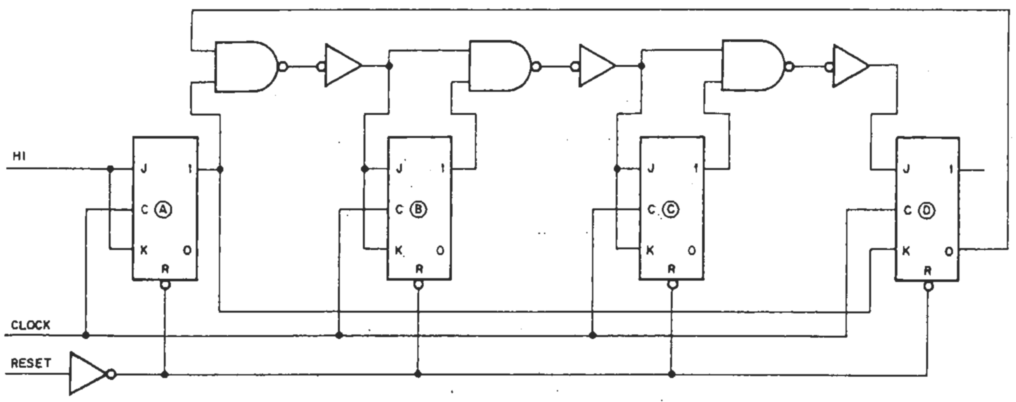
13.



Modulo 5 Counter

C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
0	0	0

14.



Modulo 10 Counter

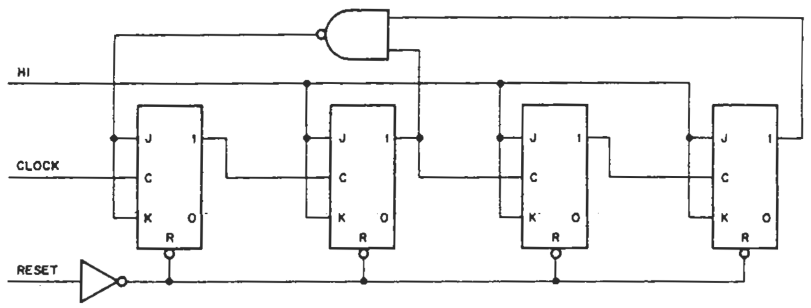
D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0

(Continued on next page)

Modulo 10 Counter (Cont)

D	C	B	A
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
0	0	0	0

15.



Self-Stopping Modulo 11 Counter

16.

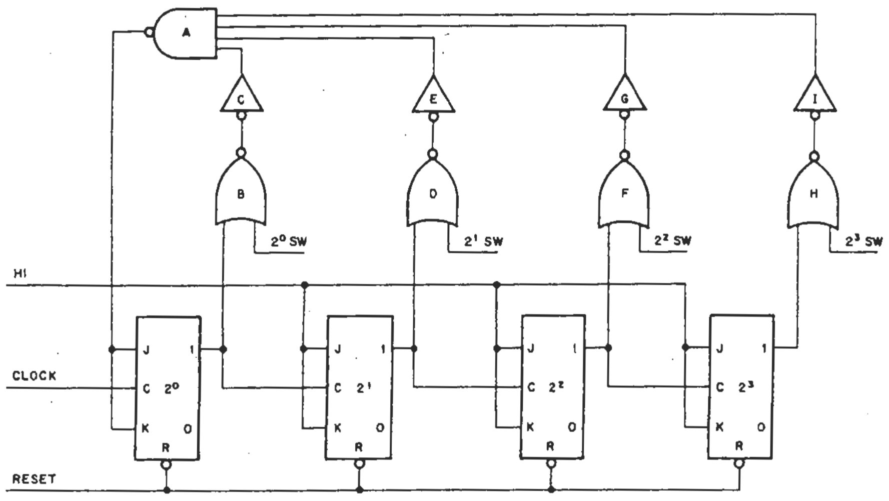


Figure 5.10 Self-Stopping Asynchronous Binary Up-Counter (Counts with Reset Pulser Depressed)

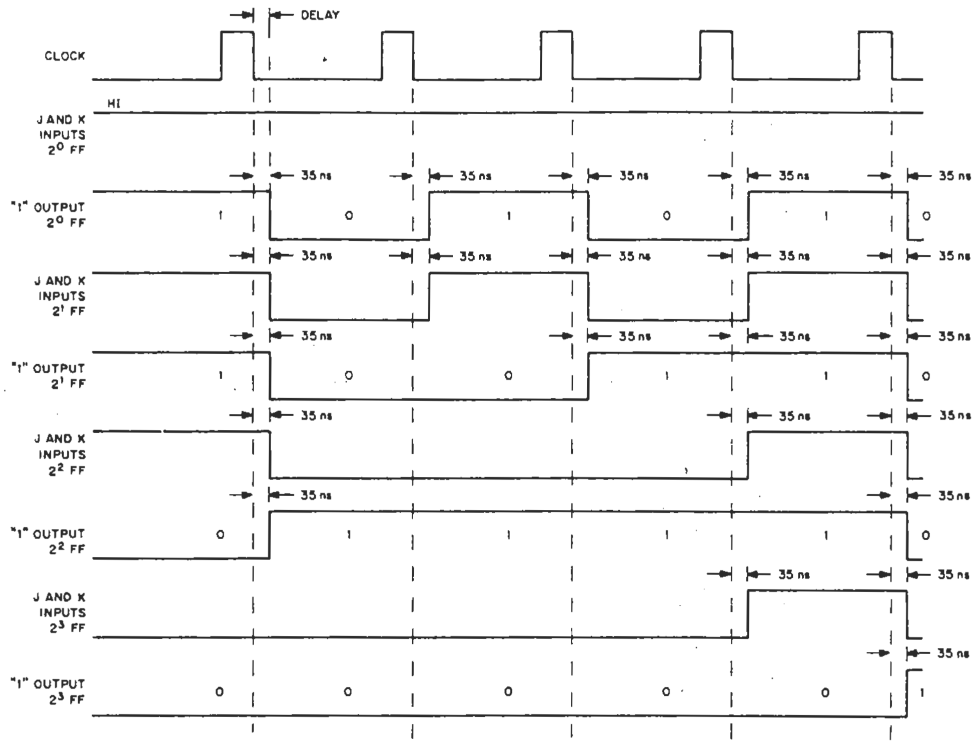
(Refer to Figure 5.10.) Flip-flops 2^0 through 2^3 make up an asynchronous up counter. As long as the J-K inputs to the 2^0 flip-flop are connected to a HI, clocking the 2^0 flip-flop will cause the counter to count. The J-K inputs of the 2^0 flip-flop are fed by gate A. When gate A's inputs are all HI, the counter is disabled. If any of the gate A inputs are LO, the counter is enabled. The counter is designed to stop at a number which is selected by a 4-bit switch register. The maximum number the counter will reach can be selected by changing 2^0 through 2^2 switches. If a switch is set LO, the corresponding bit in the number will be a 1 at the maximum count. If a switch is set HI, the corresponding bit in the number will be 0 at the maximum count. For example, if the 2^0 switch is set HI, 2^1 switch LO, 2^2 switch HI, and 2^3 switch HI, with 2^0 switch HI, the output from gate B is LO, and the output from gate C is HI. The 2^2 switch being HI produces, in the same manner, a HI out of gate G, and the 2^3 switch being HI produces a HI from gate I. Thus all inputs to gate A are HI except the output from gate E. With the 2^1 switch LO, the output of gate E will be LO until the counter reaches, starting from the least significant digit, a count of 0100. When 2^1 goes to the "1" state, the HI out of the 1 output of the 2^1 flip-flop will produce a HI out of gate E. With HIs out of gates C, E, G, and I, the output from gate A is LO, and the counter stops because the J-K input of 2^0 flip-flop is disabled. Assuming that all flip-flops were initially in the "0" state, when the RESET pulser switch is depressed, the counter will count to the maximum number indicated on the switch register. When the RESET pulser is released, the counter will reset to all 0s.

Supplementary Questions:

17. Question 17 is a supplementary experiment which asks the student to construct Figure 5.11, Recycling Modulo N Synchronous Binary Up-Counter. To determine the student's knowledge of the recycling modulo N counter, he might be asked the following question: "What is the purpose of connecting the K input of the 2^0 flip-flop to a constant HI rather than having the J and K inputs of the 2^0 flip-flop connected to a common gate input as in Figure 5.10, the Self-Stopping Asynchronous Binary Up-Counter?

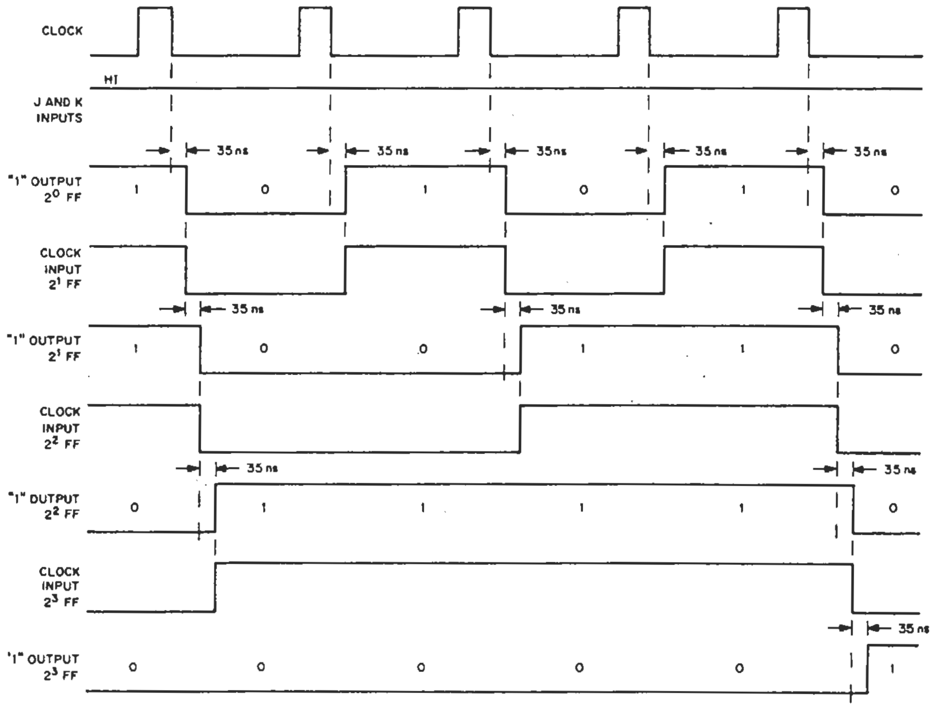
The answer is that when J and K of the 2^0 flip-flop are connected together as in Figure 5.10, if the counter stops at a number where 2^0 is in the "1" state at RESET time, 2^0 will not RESET, and 2^1 through 2^3 will RESET to 0. This means that the new up count will start from a count of 1.

18.



Synchronous Up-Counter Timing Diagram

19.



Asynchronous Up-Counter Timing Diagram

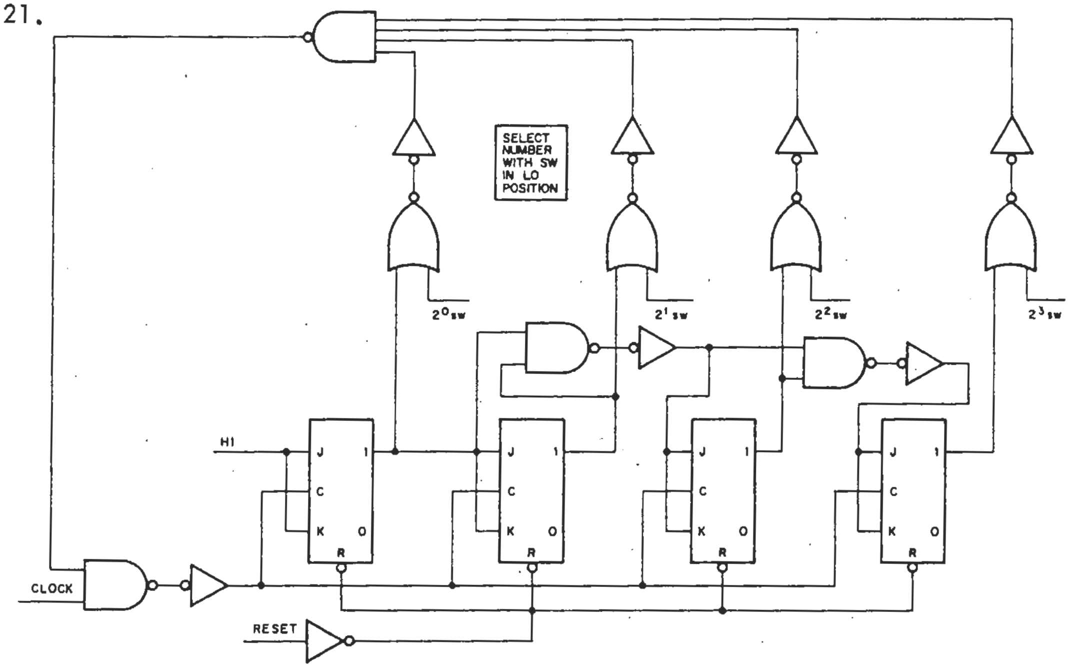
Sampling the outputs of the Asynchronous Up-Counter of Figure 5.3, 40-ns after clock pulse produces an erroneous count. This faulty count is caused by sampling the count before the ripple, caused by flip-flop delay, has time to settle.

20. Refer to Figure 5.6, Synchronous Binary Up/Down Counter. In considering the time that is required for the counter to respond to a change in counting directions, two timing considerations must be made.

1. The first situation occurs if the enable inputs are changed during a clock pulse, in this case the counter will count incorrectly as explained in the answer to Question 9; therefore, a change in direction should occur only when the clock input is LO.

2. The other situation is the amount of delay in the gates before a change is seen in the counter. Assuming the ENABLE pulses were changed during the time between clock pulses, a minimum delay of 30 ns must be allowed for a change in counting direction.

21.



4-Bit, Self-Stopping, Synchronous Binary Up-Counter

CHAPTER 6
SERIAL ADDITION

INTENT AND APPROACH

The objectives of this chapter are to explain serial binary addition logic and to introduce binary subtraction, multiplication, and division principles.

In many computer applications, arithmetic operations other than addition are a programming function. Because of this, Chapter 6 stresses addition logic and introduces only the principles involved in performing other arithmetic functions.

Initially, the chapter presents the rules of binary addition and the requirements for addition logic. After discussing these rules and requirements, a "Block Diagram," or functional diagram, of a Serial Adder is developed.

On page 82 of the COMPUTER LAB Workbook, the logic diagram for the Serial Adder is illustrated. The teacher should thoroughly explain the operation of the input register, the adder gates, and the carry circuit before asking the student to build the adder.

Referring to the truth table of Figure 6.1, the teacher should develop the Boolean expressions for the sum logic and for the carry logic. Emphasis should be placed upon an explanation of the carry logic, which is not as easily understood as the sum logic.

In Section III of Chapter 6, a method of subtraction using 2's complement arithmetic is developed; Experiment 6.2 provides practical experience in using the serial adder for subtraction.

A simple way to accomplish multiplication is introduced in Experiment 6.3. Supplementary information for this chapter provides additional information on complementing math, on adder logic, and on binary subtraction, multiplication, and division.

OUTLINE

I. BINARY ADDITION (Page 79)
   A. Rules for Addition of Binary Numbers
   B. Binary Addition Truth Table

II. SERIAL ADDITION (Pages 79-84)

- A. Explanation of Serial Process of Addition
- B. Logic Diagram of Serial Adder
- C. Explanation of Operation and Use of Serial Adder
- D. Simplified Serial Adder
    1. Operation
    2. Functional block diagrams
- E. Experiment 6.1: Serial Adder (Page 83)
    1. Logic diagram
    2. Lamp and switch usage diagram
    3. Circuit construction
    4. Directions for operation
    5. Verification of operation according to test additions
- F. Questions 1-7 (Pages 83-84)

III. BINARY SUBTRACTION (Pages 84-86)

- A. Methods of Binary Subtraction
    1. General rules
    2. Two's complement arithmetic
        - a. Procedures
        - b. Example
- B. Experiment 6.2: Subtraction Using the Serial Adder (Page 84-85)
    1. Instructions for operating adder
    2. Verification of operation
- C. Questions 8-10 (Page 85)
- D. Experiment 6.3: Multiplication Using the Serial Adder (Page 85)
    1. Logic diagram
    2. Circuit modification
    3. Operating instructions
    4. Verification of operation according to test problem
- E. Questions 11-12 (Page 85)
- F. Supplementary Questions 13-15 (Page 85)

SUPPLEMENTARY INFORMATION

I. BINARY MATH

The rules for binary addition are as follows:

$$\begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +0 \\ \hline 1 \end{array} \qquad \begin{array}{rl} 1 & \text{(Carry)} \\ 1 & \\ +1 & \\ \hline 10 & \text{(0 and a carry of 1)} \end{array}$$

Examples: 1)

$$\begin{array}{rrrl} 1 & 0 & 1 & (5) \\ +0 & 1 & 0 & (2) \\ \hline 1 & 1 & 1 & (7) \end{array}$$

2)

$$\begin{array}{rrrl} 1 & 0 & 0 & (4) \\ +0 & 1 & 1 & (3) \\ \hline 1 & 1 & 1 & (7) \end{array}$$

3)

$$\begin{array}{rrrl} & 1 & & \\ 1 & 0 & 1 & (5) \\ + & 0 & 1 & (1) \\ \hline 1 & 1 & 0 & (6) \end{array}$$

4)

$$\begin{array}{rrrrl} 1 & 1 & 1 & & \\ 0 & 1 & 1 & 1 & (7) \\ + & 0 & 1 & 1 & (3) \\ \hline 1 & 0 & 1 & 0 & (10) \end{array}$$

5)

$$\begin{array}{rrrrrl} 1 & 1 & & & & \\ & 1 & 1 & 1 & & \\ & & 1 & 0 & 1 & (5) \\ & & 1 & 1 & 1 & (7) \\ & & 1 & 0 & 0 & (4) \\ \hline 1 & 0 & 0 & 0 & 0 & (16) \end{array}$$

The rules for binary subtraction are as follows:

$$\begin{array}{r} 1 \\ -1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ -0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ -0 \\ \hline 0 \end{array} \qquad \begin{array}{rl} (1) & \text{(Borrow of 1)} \\ 0 & \\ -1 & \\ \hline 1 & \end{array}$$

Rule 4 indicates that a borrow of 1 from a more significant position becomes twice its original value when moved to the right. The 1 borrowed becomes 10 (2_{10}). Therefore rule 4 becomes

$$\begin{array}{rl} 10 & (2_{10}) \\ -\ 1 & (1_{10}) \\ \hline 1 & (1_{10}) \end{array}$$

Examples:

1)

$$\begin{array}{rrrl} 1 & 0 & 1 & (5) \\ -1 & 0 & 1 & (5) \\ \hline 0 & 0 & 0 & (0) \end{array}$$

2)

$$\begin{array}{rrrl} 1 & 1 & 1 & (7) \\ -0 & 1 & 1 & (3) \\ \hline 1 & 0 & 0 & (4) \end{array}$$

3) Problem

$$\begin{array}{cc} 1 & 0 \\ & 1 \\ \hline \end{array}$$

Solution

$$\begin{array}{cc} (1) \curvearrowright & 10 \\ \cancel{1} & 0 \\ & 1 \\ \hline 0 & 1 \end{array}$$

(Same as Rule 4)

4)

$$\begin{array}{ccccl} 1 & 0 & 0 & 1 & (9) \\ & & 1 & 1 & (3) \\ \hline \end{array} \qquad \begin{array}{cccc} (1) \curvearrowright & 10 & & \\ \cancel{1} & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ \hline \end{array} \qquad \begin{array}{cccc} & (1) \curvearrowright & 10 & 0 \\ & 1\cancel{0} & & \\ \cancel{1} & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ \hline \end{array} \qquad \begin{array}{ccccl} & 1 & 10 & & \\ \cancel{1} & 0 & 0 & 1 & \\ 0 & 0 & 1 & 1 & \\ \hline 0 & 1 & 1 & 0 & (6) \end{array}$$

5)

$$\begin{array}{rl} 11010 & (26) \\ 111 & (\ 7) \\ \hline \end{array} \qquad \begin{array}{ccccc} & (1) \curvearrowright & 10 & & \\ 1 & \cancel{1} & 0 & 1 & 0 \\ & & 1 & 1 & 1 \\ \hline \end{array} \qquad \begin{array}{ccccc} & & 10 & (1) \curvearrowright & 10 \\ 1 & \cancel{1} & 0 & \cancel{1} & 0 \\ & & 1 & 1 & 1 \\ \hline \end{array}$$

$$\begin{array}{cccccl} & & (1) \curvearrowright & & & \\ & & 1\cancel{0} & 10 & 10 & \\ 1 & \cancel{1} & 0 & \cancel{1} & 0 & \\ & & 1 & 1 & 1 & \\ \hline 1 & 0 & 0 & 1 & 1 & (19) \end{array}$$

The rules for binary <u>multiplication</u> are as follows:

$$\begin{array}{r} 0 \\ \times 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ \times 1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ \times 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ \times 1 \\ \hline 1 \end{array}$$

Examples:

1)

$$\begin{array}{rcccccl} & & & 1 & 1 & 1 & (7) \\ & & \times & & 1 & 1 & (3) \\ \hline & & & 1 & 1 & 1 & \leftarrow \\ & & 1 & 1 & 1 & & \leftarrow \\ \hline 1 & 0 & 1 & 0 & 1 & & (21) \leftarrow \end{array}$$

First Partial Product

$$\text{Step 1:} \quad \begin{array}{rccc} & 1 & 1 & 1 \\ \times & & & 1 \\ \hline & 1 & 1 & 1 \end{array}$$

Second Partial Product

$$\text{Step 2:} \quad \begin{array}{rccc} & 1 & 1 & 1 \\ \times & & & 1 \\ \hline & 1 & 1 & 1 \end{array}$$

Addition

$$\text{Step 3:} \quad \begin{array}{cccccc} & & & 1 & 1 & 1 \\ + & & 1 & 1 & 1 & \\ \hline & 1 & 0 & 1 & 0 & 1 \end{array}$$

2)

$$\begin{array}{cccccccl} & & & 1 & 0 & 1 & 0 & (10) \\ & & \times & & 1 & 0 & 1 & (5\ ) \\ \hline & & & 1 & 0 & 1 & 0 & \\ & & 0 & 0 & 0 & 0 & & \\ 1 & 0 & 1 & 0 & & & & \\ \hline 1 & 1 & 0 & 0 & 1 & 0 & & (50) \end{array}$$

3)

$$\begin{array}{cccccccl} & & & 1 & 1 & 1 & 0 & (14) \\ & & \times & & 1 & 1 & 0 & (\ 6) \\ \hline & & & 0 & 0 & 0 & 0 & \\ & & 1 & 1 & 1 & 0 & & \\ & 1 & 1 & 1 & 0 & & & \\ \hline 1 & 0 & 1 & 0 & 1 & 0 & 0 & (84) \end{array}$$

The rules for binary division are as follows:

$$\begin{array}{r} 1 \\ 1\overline{)1} \end{array} \qquad \begin{array}{r} 0 \\ 1\overline{)0} \end{array} \qquad \begin{array}{r} \text{Undefined} \\ 0\overline{)1} \end{array} \qquad \begin{array}{r} \text{Undefined} \\ 0\overline{)0} \end{array}$$

Examples:

1)
$$\begin{array}{r} 10 \\ 101\overline{)1010} \\ \underline{101} \\ 0000 \\ \underline{0000} \end{array}$$

2)
$$\begin{array}{r} 110 \\ 110\overline{)100100} \\ \underline{110} \\ 0110 \\ \underline{110} \\ 0 \\ \underline{0} \end{array}$$

3)
$$\begin{array}{r} 110 \\ 101\overline{)11110} \\ \underline{101} \\ 101 \\ \underline{101} \\ 0 \\ \underline{0} \end{array}$$

II. TWO'S COMPLEMENT NUMBERS

Using three bits, the highest number that can be represented is the decimal number 3. The chart shown below illustrates all possible positive and negative numbers that can be represented using 2's complement arithmetic.

Sign Bit			Decimal	
0	1	1	3	Positive Numbers
0	1	0	2	
0	0	1	1	
0	0	0	0	
1	1	1	-1	Negative Numbers
1	1	0	-2	
1	0	1	-3	

III. ADDER LOGIC

(Refer to Figure 6.5 and Figure 6.1 in the COMPUTER LAB Workbook.)

The sum input to flip-flop B can be expressed using the following Boolean expression:

A_0 = Least significant incident bit

B_0 = Least significant accumulator bit

C = Carry flip-flop

$$S \text{ (HI Pin J, Flip-flop } B_2) = \underset{\text{Gate 4}}{\overline{A}_0 B_0 \overline{C}} + \underset{\text{Gate 2}}{A_0 \overline{B}_0 \overline{C}} + \underset{\text{Gate 3}}{\overline{A}_0 \overline{B}_0 C} + \underset{\text{Gate 1}}{A_0 B_0 C}$$

The carry logic can be developed by referring to Figure 6.1 and can be expressed as follows:

$$\text{CARRY} = A_0B_0\overline{C} + \overline{A}_0B_0C + A_0\overline{B}_0C + A_0B_0C$$

An alternate method is illustrated in Figure 6.5. This method concerns itself with when the carry flip-flop should change state. Referring to the truth table of Figure 6.1, the carry flip-flop goes to the "1" state when A_0 and B_0 are in the "1" state; when A_0 and B_0 are in the "0" state, the carry flip-flop goes to the "0" state.

The resultant Boolean expression for the J input is $A_0 \cdot B_0$, and the resultant Boolean expression for the K input is $\overline{A}_0 \cdot \overline{B}_0$.

ANSWER KEY

Questions

1. Refer to Figure 6.4, Serial Adder. Read in from the switch register to the incident register (A_0-A_2) of Figure 6.4 is a parallel 1's transfer. In the switch register, when a switch is in a HI position, it represents a 0; when a switch is in the LO position, it represents a 1. When a switch is in the LO position, it places a HI on the J input to a corresponding flip-flop. Prior to performing the actual transfer, all flip-flops in the incident are in the "0" state. At clock pulse time, those flip-flops with a HI on the J input will switch to the "1" state, accomplishing the transfer. At transfer time, $\overline{\text{RESET}}$ and $\overline{\text{NEGATE ENABLE}}$ must be HI; $\overline{\text{READ ENABLE}}$ must be LO. $\overline{\text{NEGATE ENABLE}}$ is used for subtraction and $\overline{\text{READ ENABLE}}$ prevents the clock pulse from affecting the accumulator register during switch register to incident register transfers.

2. Refer to Figure 6.4, Serial Adder. Both the incident and accumulator registers are serial shift registers. In each register a more significant flip-flop in a particular state ("1" or "0") is "steering" the next less significant flip-flop to the same state.

In the incident register, A_2 flip-flop's input is always arranged so that on the next clock pulse it will go to the "0" state. Assume that the information transferred from the switch register placed the following number in the incident register: $A_2 = 1$, $A_1 = 0$, $A_0 = 1$; also assume that $\overline{\text{RESET}}$ was HI, $\overline{\text{NEGATE ENABLE}}$ was HI, and $\overline{\text{READ ENABLE}}$ was HI. A_2's 1 output would be HI and A_2's 0 output would be LO. The A_2 0 output would be placing a HI on the J input of flip-flop A_1, "steering" A_1 to the "1" state. Prior to clock pulse time, A_1's 1 output would be LO and A_1's 0 output would be HI. A_1's 1 output would be "steering"

A_0 to the "0" state. At clock pulse time, A_2 would switch to the "0" state, A_1 to the "1" state, and A_0 to the "0" state. Thus a "0" has switched into the A_2 flip-flop and the previous information has shifted up one position. This shifting process will continue after each clock pulse. After three clock pulses, A_2 through A_0 will be in the "0" state and all the original information will have been transferred out of the register.

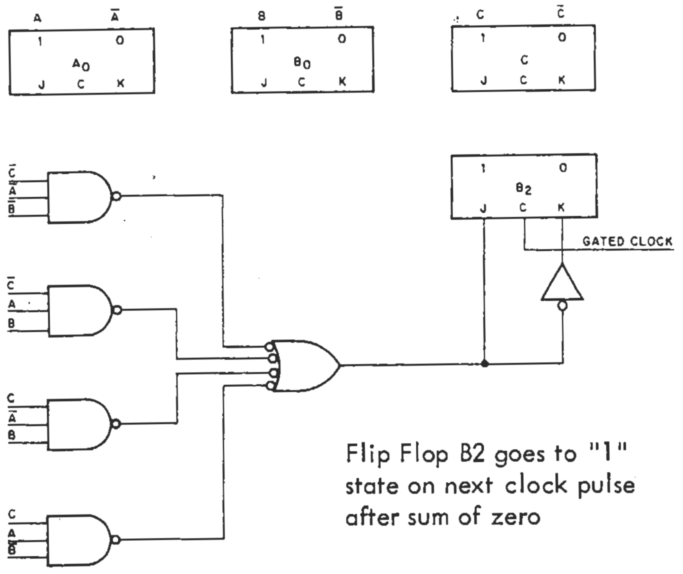
In the accumulator register, B_2's input is a result of the output of the adder circuit (gates 1 to 4). Assume that the adder output (gates 1 to 4 output) is always a 0 and that the original number in the accumulator is $B_2 = 1$, $B_1 = 0$, and $B_0 = 0$; $\overline{\text{RESET}}$ is HI, $\overline{\text{NEGATE ENABLE}}$ is HI, and $\overline{\text{READ ENABLE}}$ is HI. B_2's 1 output is HI and B_2's 0 output is LO. B_2's 1 output is "steering" B_1 to the "1" state. B_1, being in the "0" state, is steering B_0 to the "0" state.

At clock pulse time, B_2 will switch to the "0" state (assuming zero sum), B_1 will switch to the "1" state and B_0 remains in the "0" state. This shifting process will continue after each clock pulse. After three clock pulses, B_2 through B_0 will be in the "0" state and all the original information will have been transferred out of the register.

The output configuration of each flip-flop in the incident register differs from that of the accumulator; the accumulator register is only a serial shift register whereas the incident register provides not only serial transfer but also a parallel transfer from the switch register and a complementing function controlled by $\overline{\text{NEGATE ENABLE}}$. Since the incident register uses an inverting gate between bits, the J input gate is enabled by the 0 output of the previous bit and the K input gate is enabled by the 1 output of the previous bit. In the accumulator register, the 1 output enables the J input of the next bit and the 0 output enables the K output in the next bit as in a normal shift register.

3. The incident register fills with 0s as the information shifts out because A_2's input configuration always steers it to the "0" state. If A_2 is initially set to the "1" state by a parallel transfer from the switch register, A_2's 0 output, passed through a Negated Input OR gate, will place a HI on the K input of flip-flop A_2, causing the flip-flop to go to the "0" state after the next clock pulse. If the flip-flop is initially in the "0" state and $\overline{\text{RESET}}$ and $\overline{\text{NEGATE ENABLE}}$ are HI, the 0 output of A_2 would be HI. Both inputs J and K of A_2 would be LO. At clock pulse time, A_2 would remain in the "0" condition.

4. ZERO SUM = $\bar{C}\cdot\bar{A}_0\bar{B}_0 + \bar{C}\cdot A_0 B_0 + C\cdot\bar{A}_0 B_0 + C\cdot A_0\bar{B}_0$



Flip Flop B2 goes to "1" state on next clock pulse after sum of zero

5. The following chart illustrates the new condition of the carry flip-flop after clock pulse time. The carry flip-flop will go to the "1" state on the next clock pulse after both the incident and accumulator registers are in the "1" state (see point Ⓐ). The carry flip-flop will go to the "0" state on the next clock pulse after both the incident and accumulator registers are in the "0" state (see point Ⓑ).

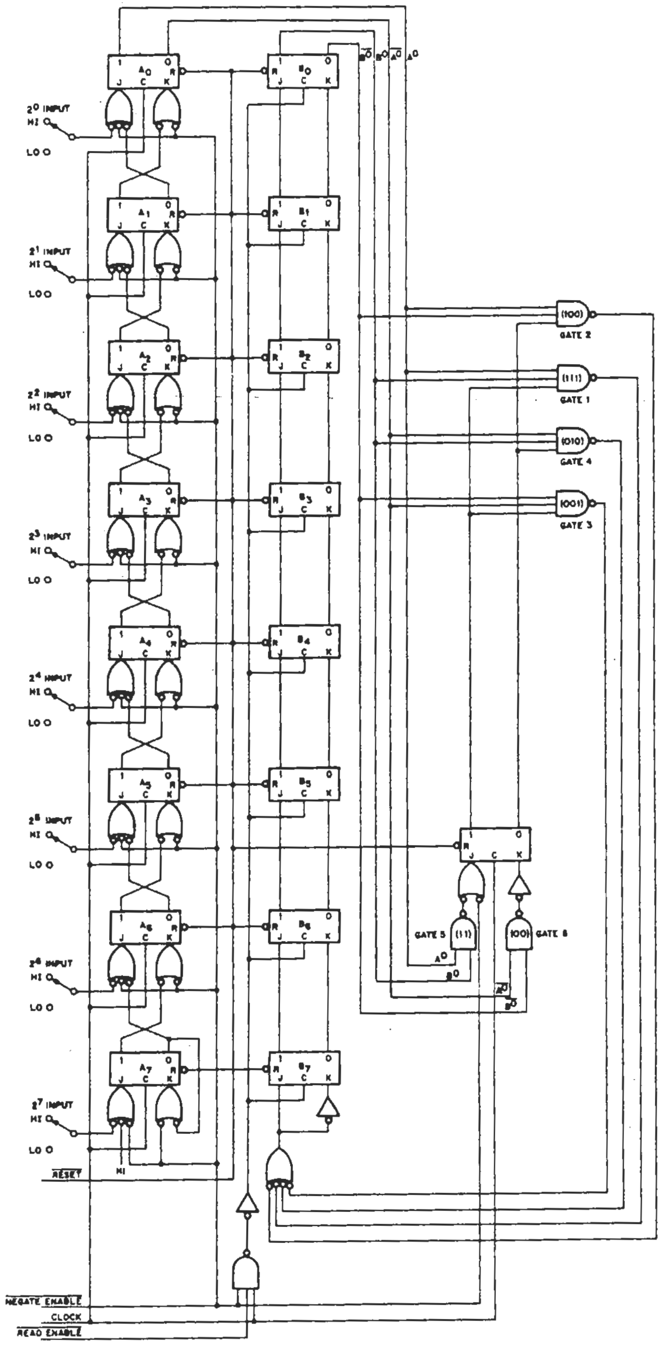
Incident Register A_0	Accumulator Register B_0	Carry Flip-Flop Previous to Clock Pulse C	Carry Flip-Flop After Clock Pulse– New Condition C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Carry Flip-Flop Change Ⓑ

Carry Flip-Flop Change Ⓐ

6. The carry flip-flop must be reset when the incident and accumulator registers are reset to remove any previous carry information that might produce erroneous information from new additions. If a "1" were present in the carry flip-flop at the beginning of a new addition, the final answer would be in error by plus one.

7a.



Eight Bit Serial Adder

7b. NOTE

1. Step 4, page 83, should be changed to: " Pulsing the clock pulser switch eight times will shift the number from the incident register to the accumulator register by adding the incident number to the 0s initially in the accumulator."

2. Step 6, page 83, should be changed to: " Pulse the clock pulser switch eight times to add the numbers in the incident and accumulator registers. The sum will appear in the accumulator register after the addition.

8. Refer to Figure 6.4, Serial Adder. When a number is negated it is complemented and increased by 1 (incremented). Assume that the decimal number 2 is in the incident register, placing A_2 in the "0" state, A_1 in the "1" state, and A_0 in the "0" state. If the $\overline{\text{NEGATE ENABLE}}$ switch is placed in the LO position, inputs J and K of A_2 through A_0 and input J of the carry flip-flop go to a HI. If the clock pulser switch is then pulsed one time, all incident register flip-flops complement and the carry flip-flop is placed in the "1" state. The decimal number 5 is now in the incident register ($101_{(2)}$) and a carry of 1 is present in the carry flip-flop. When the incident register information is added to the accumulator (all 0s) the sum of 5 + 0 + 1 or 6 will appear in the accumulator.

9. The adder of Figure 6.4 can be used to determine the absolute value of a 2's complement negative number as follows:

a. Clear the incident and accumulator registers and the carry flip-flop by a momentary LO $\overline{\text{RESET}}$ signal. Restore the output of the $\overline{\text{RESET}}$ switch tō HI.

b. Read 110 (2's complement negative number) into the incident register by setting the input switch register to 110 and the $\overline{\text{READ ENABLE}}$ switch to LO and then pulsing the clock pulser switch once. Restore all switches to HI after read in.

c. Take the $\overline{\text{NEGATE ENABLE}}$ switch LO and pulse the clock pulser once. Restore $\overline{\text{NEGATE ENABLED}}$ to a HI.

d. Pulse the clock pulser switch three times to add the incident register, the accumulator register, and the carry flip-flop. The absolute value of 110 will appear in the accumulator.

The above procedure placed the number 110 in the incident register, and complemented the incident number and the carry. The result in the incident register was 001. By adding the 001 to the accumulator (000) and the carry (1), the 2's complement of 110 appears in the accumulator (010). The 2's complement of a negative 2's complement number is the absolute value of the number.

(1)	$110_{(2)}$	- 2's complement negative number
(2)	001	- Complement of (1)
	$\underline{001}$	- Plus 1
	$\overline{010}$	- Complement + 1 or 2's complement of 110

10. The Serial Adder can not represent numbers with a value greater than 3. Each number represented requires a sign bit plus two bits.

11. Refer to Figure 6.6. Assume that the number 010 has been placed in the register, and that $\overline{\text{NEGATE ENABLE}}$ is HI and $\overline{\text{RESET}}$ is HI. A_2 is in the "0" state and is steering A_1 to the "0" state. A_1 is in the "1" state and is steering A_0 to the "1" state. A_0 is in the "0" state and is steering A_2 to the "0" state. After the first clock pulse, A_2 will remain in the "0" state, A_1 will switch to the "0" state, and A_0 will switch to the "1" state. With the next clock pulse, information will continue to switch from A_2 to A_1, A_1 to A_0, and A_0 to A_2. After every third clock pulse, the original data configuration will return to the register.

12. If five successive additions of the number $010_{(2)}$ are attempted using the modified serial adder of Experiment 6.3, an incorrect answer will appear in the accumulator because multiplying 2 times 5 results in a number which exceeds the capacity of the accumulator ($1010_{(2)}$).

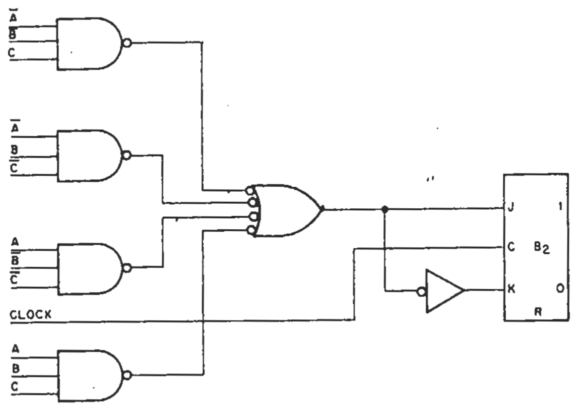
13.

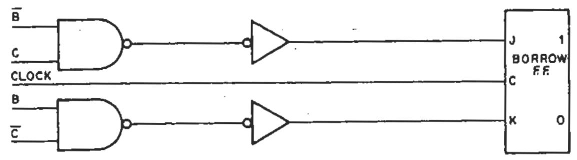
INPUTS			OUTPUTS	
Borrow Flip-Flop A Borrow (In)	Incident B A_N Bit (Minuend)	Accumulator C B_N Bit (Subtrahend)	Difference	Borrow (Out)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

14. Difference = $\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

Borrow Flip-Flop goes to the "1" state = $\bar{B}C$

Borrow Flip-Flop goes to the "0" state = $B\bar{C}$





15. Just as multiplication can be performed on the serial adder by successive additions, division can be performed by successive subtraction and testing for 0 or a negative result.

CHAPTER 7
PARALLEL ADDITION

INTENT AND APPROACH

The objectives of this chapter are as follows:

a. to teach the principles of two step parallel addition;

b. to demonstrate the use of the Two Step Parallel Adder in subtraction;

c. to explain the principles of one step addition.

In Chapter 7 the explanation of a two step parallel adder begins with the organization of the adder. A "block diagram" is used to demonstrate the adder's functional operation. In conjunction with the block diagram, a truth table which illustrates all possible adder conditions is presented.

Since the adder logic is duplicated for every bit position of an adder circuit, an explanation of only one adder stage (bit position) is given.

In Experiment 7.1, the student is required to build a Three Stage Adder. Test additions for the adder are provided on page 92 in the COMPUTER LAB Workbook. These additions check both the half-add and the carry logic.

Subtraction using the Two Step Adder is explained in Section III of Chapter 7. Since both methods are found in practical computer logic, both 2's complement and 1's complement subtraction are covered. In this chapter, as in all of the chapters, questions were selected to emphasize major points in the operation and application of particular logic configurations. The teacher should explain thoroughly the answers to every question in the chapter. Detailed explanations are found in the Answer Key of the Teacher's Guide.

OUTLINE

I. INTRODUCTION (Page 87)

  A. Comparison of Serial and Parallel Adders in Terms of Speed

  B. Comparison of Serial and Parallel Adders in Terms of Cost

II. TWO STEP PARALLEL ADDER (Pages 87-92)

  A. Block Diagram of Parallel Adder

1. Half-add
2. Operation truth tables
3. Carry

B. Logic Diagram: One Bit of Two Step Parallel Adder

C. Operation of Two Step Parallel Adder

D. Questions 1-2 (Pages 89-90)

E. Experiment 7.1: Two Step Parallel Adder (Pages 90-92)
1. Logic diagram
2. Circuit construction
3. Lamp and switch usage
4. Operating instructions
5. Verification of operation according to test problems

F. Questions 3-5 (Page 92)

III. SUBTRACTION USING THE TWO STEP PARALLEL ADDER (Pages 92-94)

A. Experiment 7.2: Two's Complement Subtraction (Page 92)

B. Questions 6-7 (Page 92)

C. Experiment 7.3: One's Complement Subtraction (Page 93)
1. Explanation
2. Sign bit
3. End around carry

D. Questions 8-11 (Page 93)

E. Experiment 7.4: One Step Parallel Addition (Page 93)
1. Use
2. Logic diagram
3. Circuit construction
4. Verification of operation

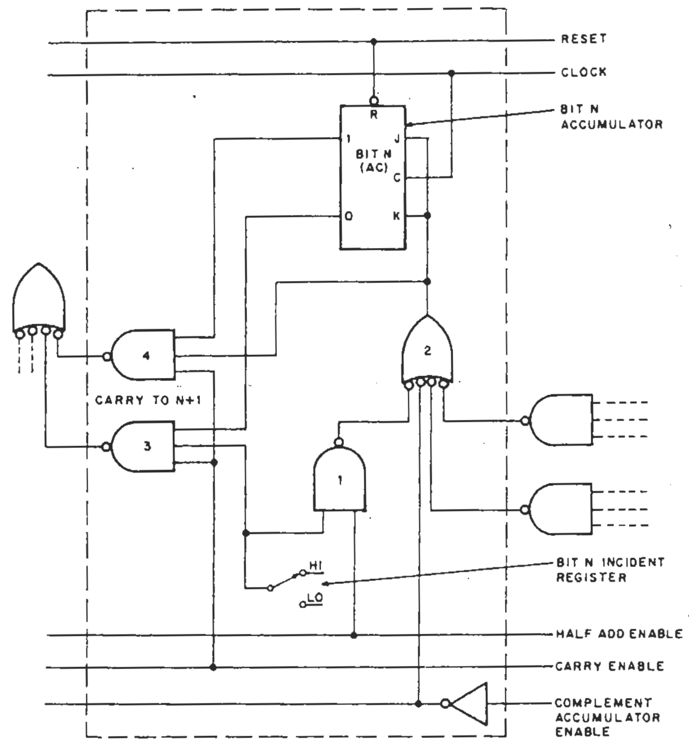
F. Questions 12-13 (Page 94)

G. Supplementary Questions 14-15 (Page 94)

ANSWER KEY

Questions

1.



Two Step Parallel Adder

Refer to the diagram of a Two Step Parallel Adder. The half-add function in the Two Step Parallel Adder has two conditions. The first condition occurs when the incident bit is a "1." If the incident bit is a "1" and HALF-ADD ENABLE is HI, gate 1 will be enabled and will place a HI on both J and K of the accumulator Bit N flip-flop, through gate 2. The next clock pulse will complement the Bit N flip-flop.

The second condition occurs when the incident bit is a "0." With the incident bit a "0", gates 1 and 2 are disabled; this places a LO on J and K of the accumulator Bit N flip-flop The next clock pulse will not complement the Bit N flip-flop.

2. Refer to diagram of a Two Step Parallel Adder.

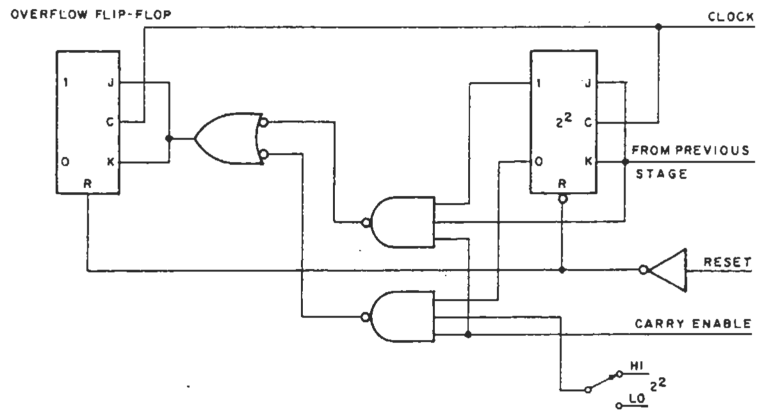
(AC Bit N in "0." state)

$$\text{Carry} = \overbrace{\text{AC Bit}_N \cdot \text{Gate } 2 \cdot \text{CARRY ENABLE}}^{\text{Gate 4}} + \overbrace{\overline{\text{AC Bit}_N} \cdot \text{INC Bit}_N \cdot \text{CARRY ENABLE}}^{\text{Gate 3}}$$

Gate 3 passes carry information from Bit N to Bit N + 1 if the incident bit is a 1 and the accumulator bit is a 0 and Carry Enable is present.

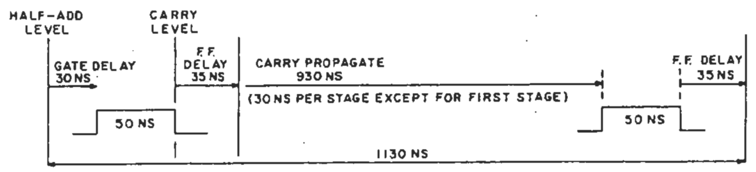
Gate 4 passes carry information on to the next bit if there is a carry from the previous bit (Bit N - 1) and if the accumulator bit is in the "1" condition and Carry Enable is present.

3.



4.a From the leading edge of the HALF-ADD level to 35 ns after the trailing edge of the clock pulse occurring during carry time, a 3-bit addition would take 260 ns.

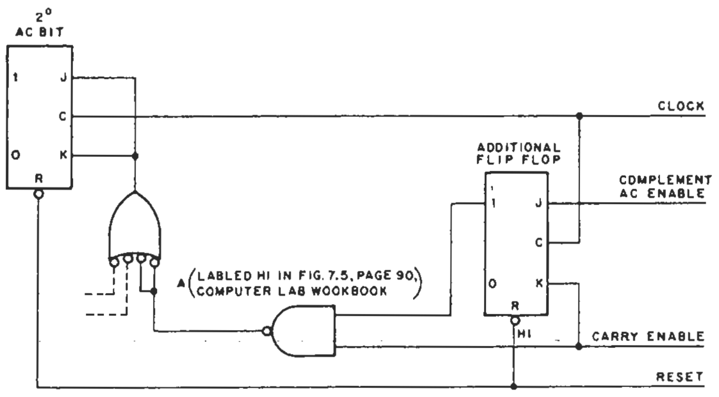
4.b From the leading edge of the HALF-ADD level to 35 ns after the trailing edge of the clock pulse occurring during carry time, a 32-bit addition would take 1130 ns.



5. The RESET input originates from a pulser. During RESET time, when the pulser is pressed, the input to the RESET line inverter is HI and the output of the inverter is LO, resetting the adder flip-flops. During the time when the RESET pulser is not pressed, the output of the pulser is LO; the inverter is required to invert the pulser output to a HI in order not to reset the adder flip-flops.

In order to standardize the rocker switch enable control lines, COMPLEMENT AC ENABLE, CARRY ENABLE, and HALF-ADD ENABLE are all energized with their associated rocker switches in the HI position. COMPLEMENT AC ENABLE requires a LO level input at the adder's Negated Input OR function, and thus requires inversion from the rocker HI output.

6.



7. Using the control system developed in Question 6, the adder can be used to determine the absolute value of a 2's complement negative number as follows:

a. Depress the RESET pulser once to clear the accumulator.

b. Set a 2's complement negative number in the incident register.

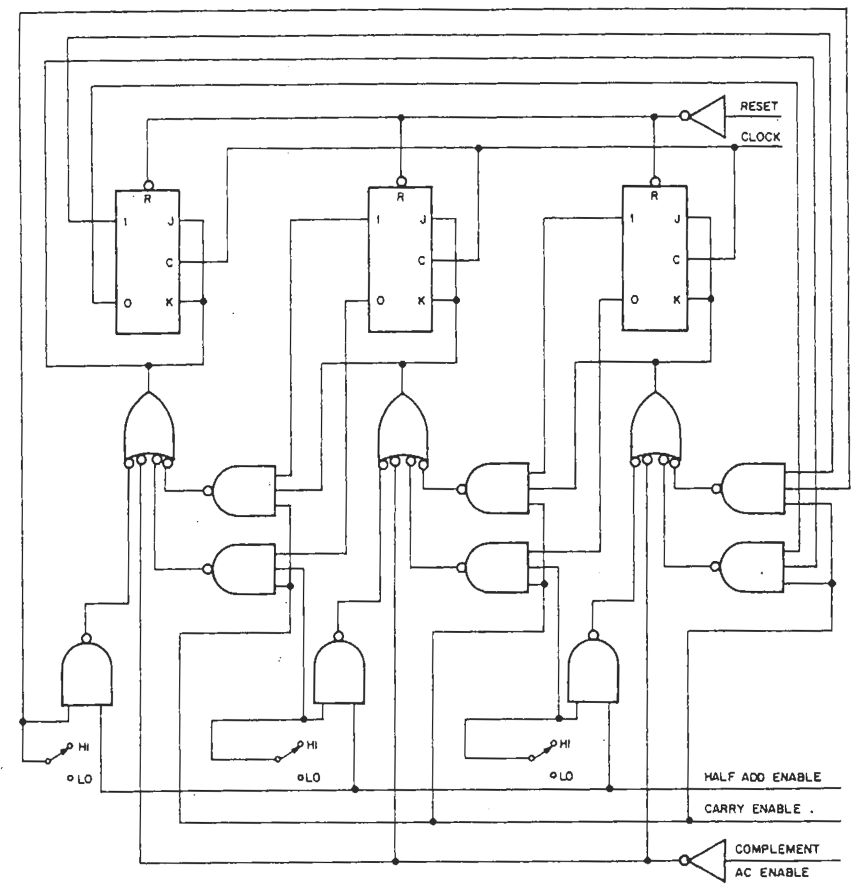
c. Read the 2's complement negative number into the accumulator by putting the HALF-ADD ENABLE switch in the HI position and depressing the clock pulser once. Make sure that the other two function enabling switches are in the disable position (LO) before pulsing the clock switch. Return HALF-ADD ENABLE to LO position.

d. Place the COMPLEMENT AC ENABLE switch to the HI position and generate a clock pulse. The information in the accumulator will be complemented. Return COMPLEMENT ENABLE to LO position.

e. Place the CARRY ENABLE switch in the HI position (with the new control system this provides a LO at Point A). Generate a clock pulse. This action will provide the absolute value of the original 2's complement negative number in the accumulator and reset the control flip-flop.

If the negative number is already present in the accumulator, the absolute value can be obtained by accomplishing steps d and e.

8.



Parallel Adder With End-Around Carry Logic

9.a Two numbers which will generate overflow must not be added in the above adder. Overflow in the above figure will cause the least significant digit to complement and produce erroneous results.

9.b One's complement subtraction procedure:

(1) Depress the RESET pulser.

(2) Set the subtrahend into the incident register.

(3) Read the number into the accumulator by putting the HALF-ADD ENABLE switch in the HI position and depressing the clock pulser once. Make sure that the other two function enabling switches are in the disable position (LO) before pulsing the clock switch.

(4) Set the minuend in the switch register.

(5) Place COMPLEMENT AC ENABLE in the HI position and press the clock pulse pulser. This action complements the accumulator. Replace COMPLEMENT AC ENABLE in the LO position.

(6) Half-add by putting the HALF-ADD ENABLE switch in the HI position and depressing the clock pulser once. Again be sure that half-add is the only function enabled when the clock pulser is depressed. After half-adding, return the HALF-ADD ENABLE switch to the LO position.

(7) Carry by putting the CARRY ENABLE switch in the HI position and depressing the clock pulser once. Carry must be the only function enabled when the clock pulse occurs. The remainder will appear in the accumulator.

10. To subtract the incident register from the accumulator register, negate the number in the accumulator, add it to the incident register, and then negate the final difference stored in the accumulator.

11. It is easier to obtain the 1's complement of a number than the 2's complement. However, because there are two 0s in the 1's complement representation, it is sometimes troublesome in computer applications.

The 2's complement logic is simpler than the 1's complement logic because 2's complement logic does not require an end-around carry.

Doing 1's complement arithmetic in a serial adder requires more steps than does 2's complement arithmetic. In doing 1's complement arithmetic, the minuend is loaded into the AC and the subtrahend is loaded into the incident register. The incident register is complemented, then added to the accumulator. An additional step is now required to add any possible end-around carry bit stored in the carry flip-flop to the accumulator. With 2's complement arithmetic, no end-around carry has to be considered; therefore, the additional step is not necessary.

12. From the trailing edge of an ADD CLOCK, 65 ns is required for the first stage to settle and produce carry out. (At the trailing edge of the same ADD CLOCK the incident information can be changed.) Carry also requires 30 ns to propagate through each additional stage. The last stage only requires 15 ns to apply carry in to its AC flip-flop.

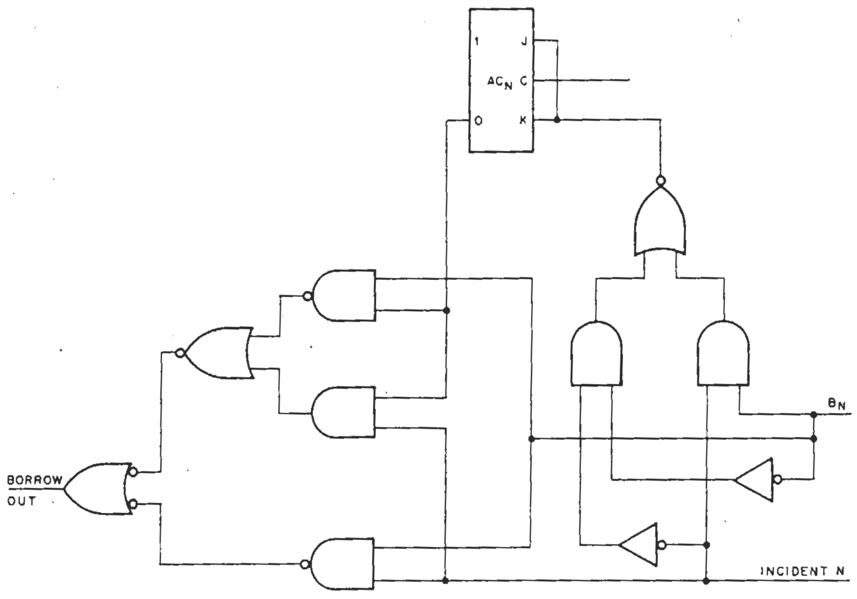
For 12 bits, starting from the trailing edge of an ADD CLOCK, 380 ns are required before another ADD CLOCK can be applied. This addition rate is 2.6 MHz. This time assumes carry out of the last stage is not used.

13. $C_N + 1 = \overline{\bar{B}_N \cdot \bar{A}_N + \bar{C}_N \cdot (\overline{A_N \cdot B_N + \bar{B}_N \cdot \bar{A}_N})}$

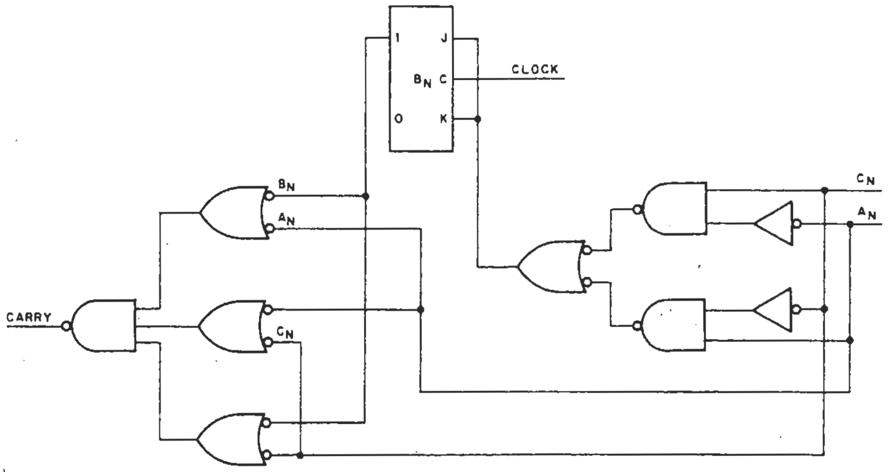
$J \cdot K = \overline{\bar{A}_N \cdot \bar{C}_N + A_N \cdot C_N}$

14.

Inputs			Outputs	
Borrow (In)	AC	Incident	Difference	Borrow (Out)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1



15.a



15.b 9 NAND gates.

CHAPTER 8

BINARY CODED DECIMAL OPERATIONS

INTENT AND APPROACH

The objectives of this chapter are as follows:

a. to explain the principles of BCD counters;

b. to teach BCD counter design;

c. to explain the principles of BCD addition;

d. to explain the principles of BCD subtraction.

In Chapter 8, various BCD codes are demonstrated. The codes most frequently used in the computer industry are further developed in the chapter.

This chapter assumes that the student possesses a thorough knowledge of the information presented in previous chapters and does not give detailed explanations for each logic configuration. The teacher should require the student to provide explanations for each of the logic circuits presented.

In designing the logic circuits in this chapter, the student should follow the procedures previously outlined in the intent and approach section for Chapter 5.

BCD addition is presented in Section III of Chapter 8 in two parts, serial addition and paralled addition. An introduction to BCD addition can be obtained by explaining the material preceding Experiment 8.5. If a more thorough explanation is required, Experiment 8.5 should be performed by the student, and parallel BCD addition should be explained.

Before attempting to understand the principles of BCD subtraction covered in Section IV of Chapter 8, the student should be thoroughly familiar with what constitutes a 9's complement and a 10's complement number. The relationship between 1's and 2's complements and 9's and 10's complements should be discussed.

OUTLINE

I. INTRODUCTION (Page 95)

   A. Necessity of Coding

   B. Common Types of Codes (Counting Sequences)

      1. 8421

      2. Excess 3

      3. 2421

4. 5421
5. 5311
6. 74 (-2) (-1)

II. BCD COUNTERS (Pages 95-101)

A. Experiment 8.1: The 8421 Counter (Pages 95-96)
1. Explanation
2. Logic diagram
3. Circuit construction
4. Verification of operation according to count sequence

B. Questions 1-3 (Page 96)

C. Experiment 8.2: The Excess 3 Counter (Page 96)
1. Operation
2. Logic diagram
3. Circuit construction
4. Verification of operation according to count sequence

D. Questions 4-7 (Page 96)

E. Experiment 8.3: The 2421 Counter (Page 96)
1. Operation
2. Logic diagram
3. Circuit construction
4. Verification of operation according to count sequence

F. Questions 8-10 (Page 96)

G. Experiment 8.4: The 5421 Counter (Page 96)
1. Operation
2. Logic diagram
3. Circuit construction
4. Verification of operation according to count sequence

H. Questions 11-12 (Page 101)

III. BCD ADDITION (Pages 101-103)

A. 8421

B. Excess 3

C. Experiment 8.5: Serial BCD Addition (Supplementary) (Pages 101-102)
1. Logic diagram
2. Circuit construction

D. Questions 13-16 (Pages 102-103)

E. Parallel BCD Addition

1. Operation

2. Block diagram of an Excess 3 parallel adder

F. Questions 17-18 (Page 103)

IV. BCD SUBTRACTION (Pages 103-104)

A. 9's Complement Arithmetic

1. Forming the 9's complement of a decimal

2. Sign bit

3. Rules for 9's complement addition

4. Example

B. 10's Complement Arithmetic

1. Formation of 10's complement negative

2. Discard carry

C. Supplementary Questions 19-24 (Page 104)

ANSWER KEY

Questions

1. Refer to Figure 8.2, page 97, COMPUTER LAB Workbook.

8421 Count Sequence

2^3	2^2	2^1	2^0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

The 8421 counter of Figure 8.2 is a Synchronous Up-Counter with a break point at 1001; it then recycles to 0000.

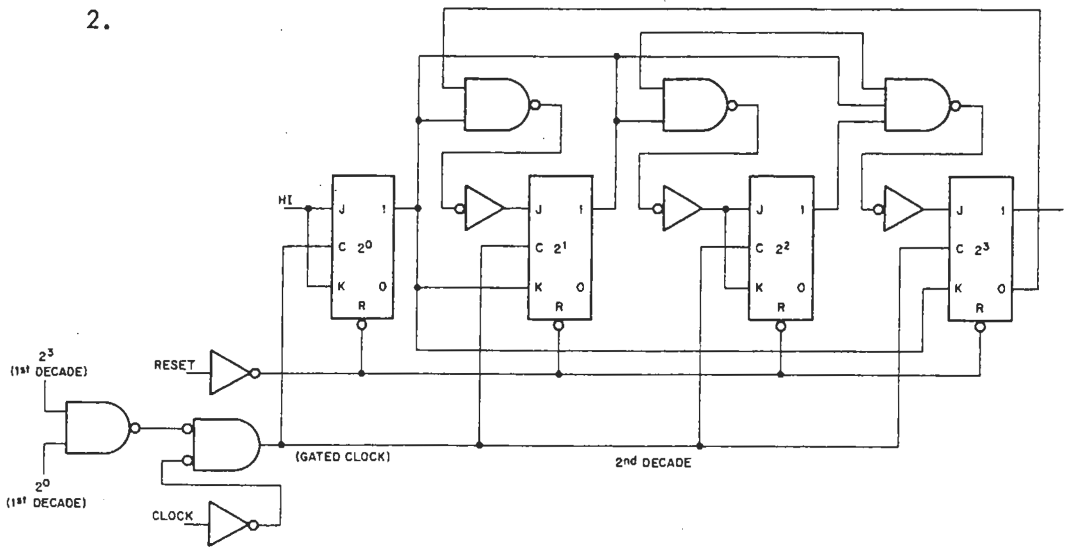
The 2^0 flip-flop complements at each clock pulse. With the 2^3 flip-flop in the "0" state, the 2^1 flip-flop will complement on the next clock pulse after 2^0 flip-flop goes to the "1" state.

The 2^2 flip-flop complements on the next clock pulse after both 2^0 and 2^1 are in the "1" state.

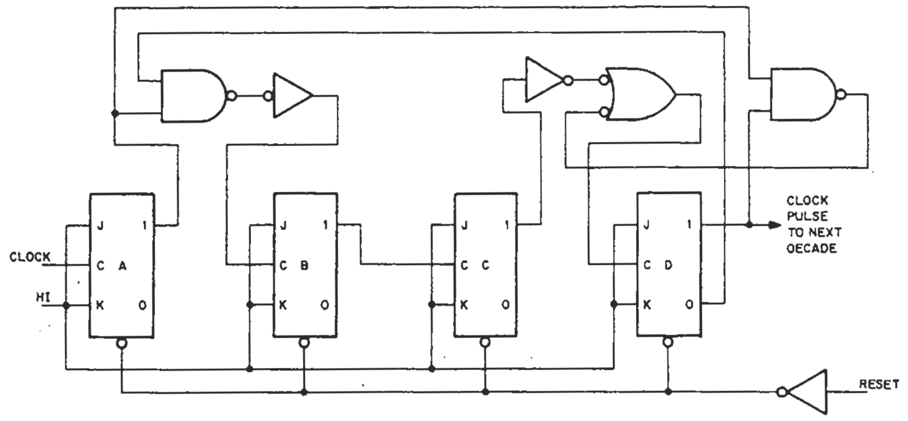
The 2^3 flip-flop will switch to the "1" state on the next clock pulse after 2^0, 2^1, and 2^2 are in the "1" state. The 2^3 flip-flop's K Input will go HI after the 2^0 flip-flop goes to the "1" state.

The counter breakpoint is reached at a count of 1001. At this count, the 2^1 flip-flop is disabled by 2^3's 0 output. 2^3's K input is enabled by the 2^0 flip-flop's 1 output. The next clock pulse will place the 2^0 and 2^3 flip-flops in the "0" state.

2.

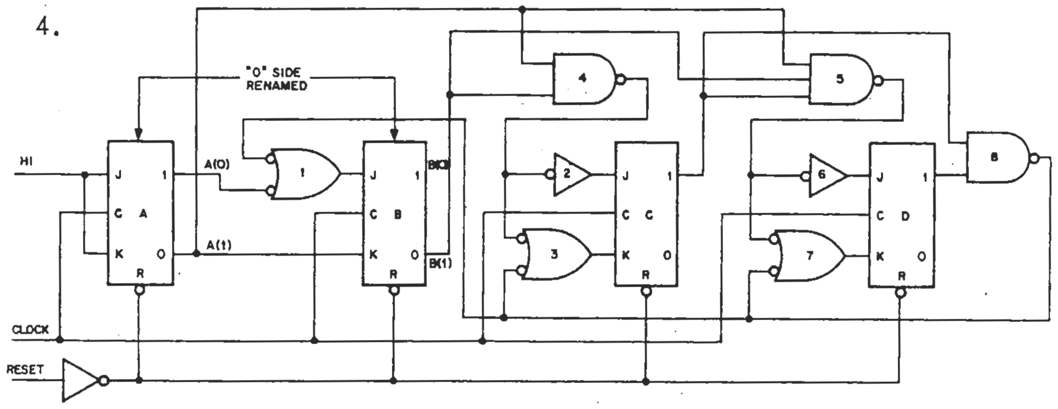


3.



Asynchronous 8421 Up Counter

4.



Synchronous Excess 3 Code Up-Counter

D	C	B	A
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0

Excess 3 Count Sequence

Since the synchronous Excess 3 up counter should start from a count of 0011 and progress to 1100, correct initial reset can be obtained by reversing and renaming the two least significant flip-flop outputs as shown in the drawing for the synchronous Excess 3 Code Up-Counter.

In the Excess 3 counter, flip-flop A complements with each clock pulse. When flip-flop A goes to the "1" state, its 1 (Renamed) output places a HI on the K input of flip-flop B. Flip-flop A going to the "1" state also places a HI on the J input of flip-flop B through gate 1. As a result of the preceding conditions, flip-flop B will complement on the next clock pulse after flip-flop A goes to the "1" state. Flip-flop B will also switch to the "1" state on the next clock pulse after C, and D goes to the "1" state, because gates 8 and 1 are enabled placing a HI on the J input of flip-flop B.

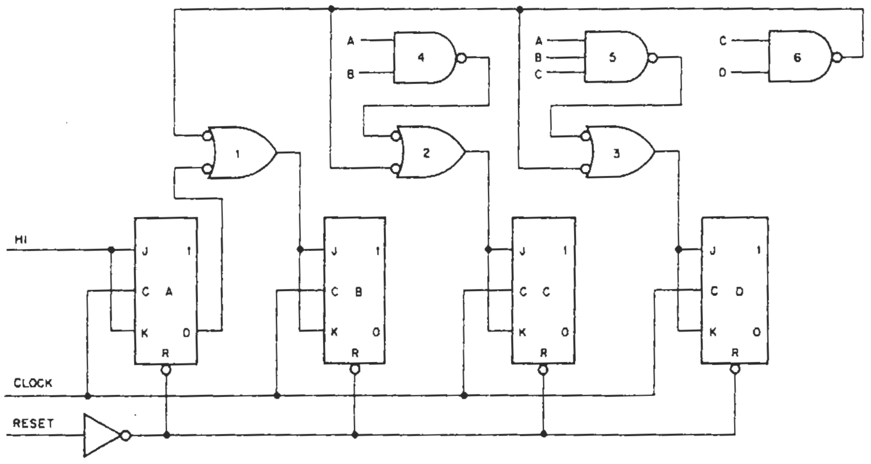
Flip-flop C, controlled by gates 2, 3, and 4, complements on the next clock pulse after flip-flops A and B go to the "1" state. Flip-flop C goes to the "0" state on the next clock pulse after flip-flops C and D are in the "1" state. Flip-flop D, controlled by gates 5, 6, and 7, complements on the next clock pulse after flip-flops A, B, and C are in the "1" state. Flip-flop D goes to the "0" state on the next clock pulse after flip-flop C and D are in the "1" state.

The counter follows a binary counting sequence from 0011 to 1100. With flip-flops C and D in the "1" state (1100) gate 8 is enabled, causing the following to occur on the next clock pulse:

(1) Flip-flop B goes to the "1" state through gate 1
(2) Flip-flop C goes to the "0" state through gate 3
(3) Flip-flop D goes to the "0" state through gate 7.

On the next clock pulse after a count of 1100, flip-flop A complements because its J and K inputs connect to a HI.

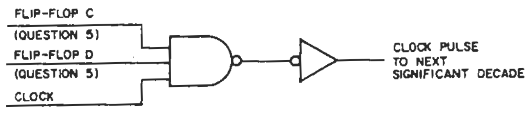
5.



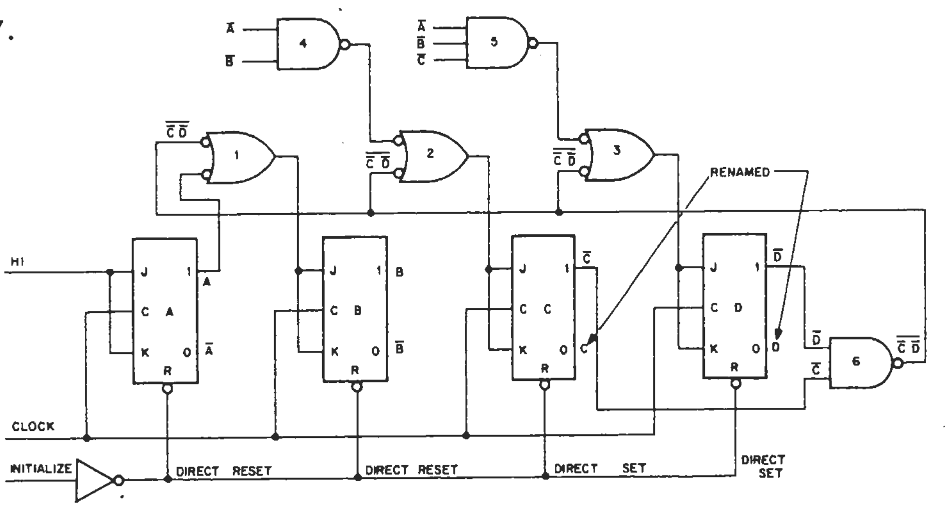
Synchronous Excess 3 Code Up-Counter

D	C	B	A
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0

6.



7.



Synchronous Excess 3 Code Down-Counter

D	C	B	A
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1

Excess 3 Down Counter

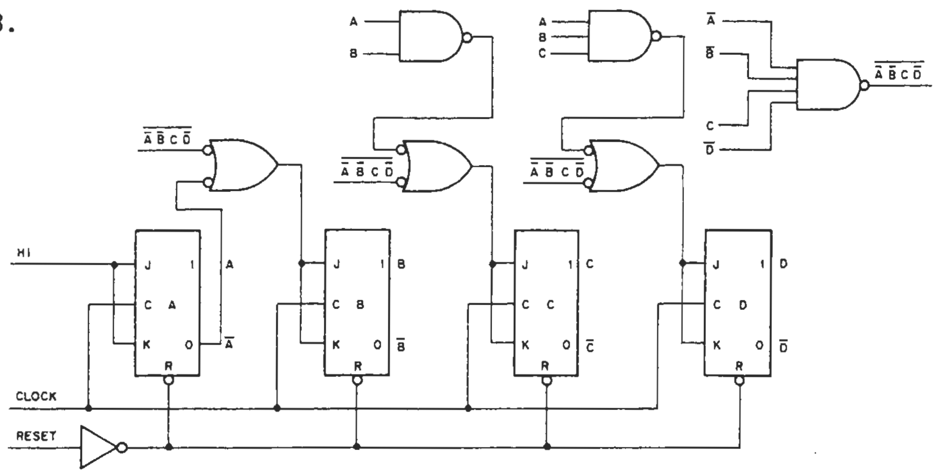
Complement D = $\overline{A}\,\overline{B}\,\overline{C} + \overline{C}\,\overline{D}$

Complement C = $\overline{C}\,\overline{D} + \overline{A}\,\overline{B}$

Complement $\overline{B}$ = $\overline{C}\,\overline{D} + \overline{A}$

Complement A = each clock pulse

8.



SIMPLIFIED 2421 COUNTER

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

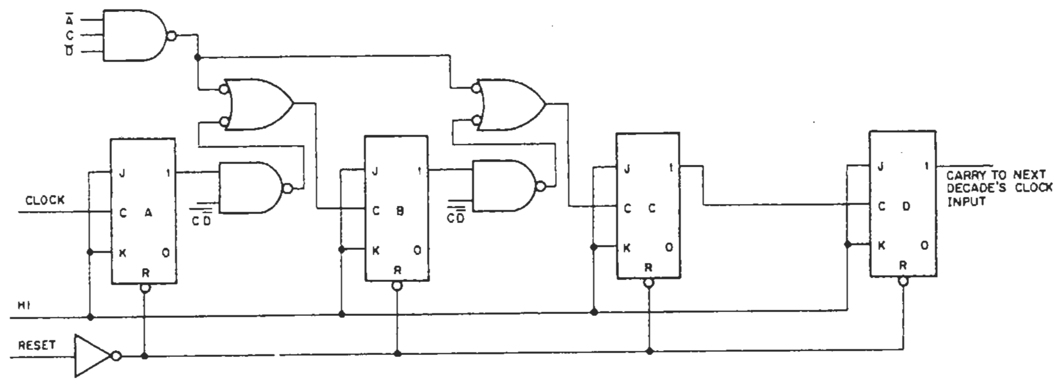
2421 Counter

Change B = $A + \overline{A}\overline{B}C\overline{D}$

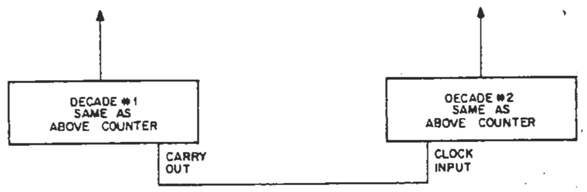
Change C = $AB + \overline{A}\overline{B}C\overline{D}$

Change D = $ABC + \overline{A}\overline{B}C\overline{D}$

9.



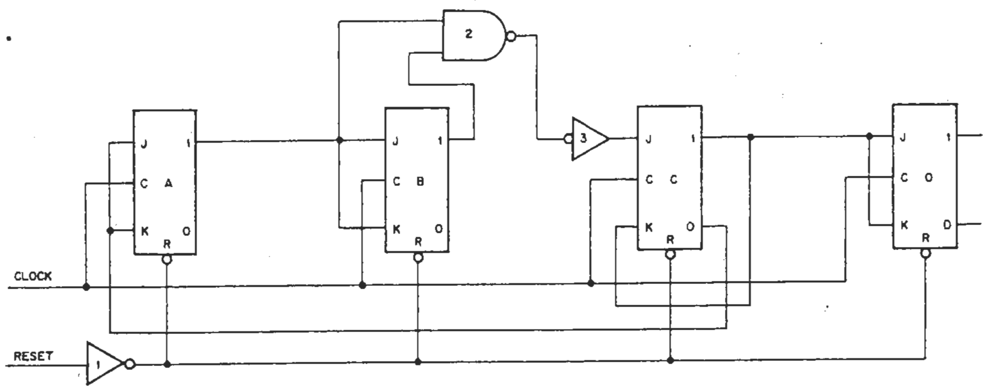
Asynchronous 2421 Counter



DECADE #1 SAME AS ABOVE COUNTER
DECADE #2 SAME AS ABOVE COUNTER
CARRY OUT
CLOCK INPUT

10. 440 ns (from the trailing edge of the input clock pulse, and the counter going from all 1s to all 0s.

11.



5421 BCD Counter

The 5421 counter shown above has two break points. One break point occurs after a count of 0100 and the other occurs after a count of 1100.

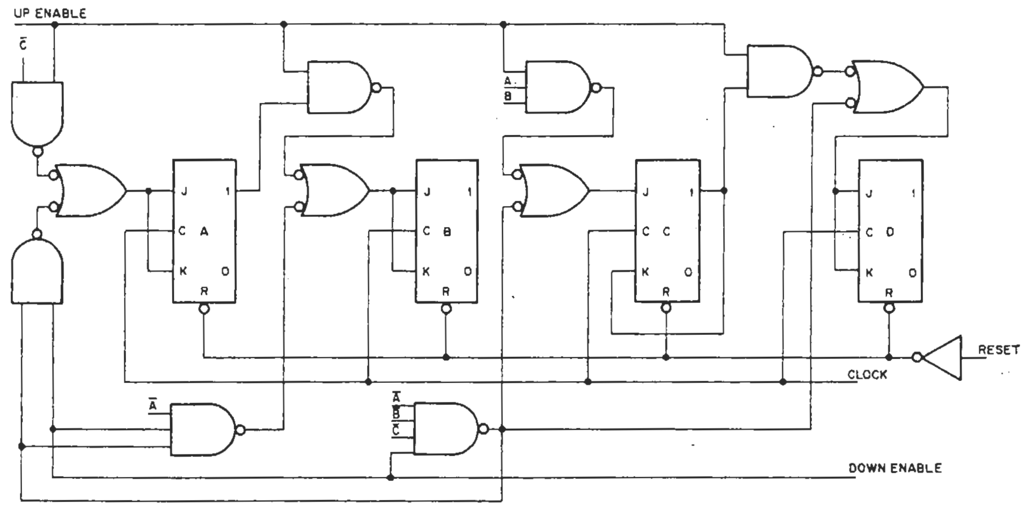
Flip-flop A complements at each clock pulse until C goes to the "1" state.

Flip-flop B complements on the clock pulse after flip-flop A is in the "1" state. Flip-flop C goes to the "1" state on the clock pulse after flip-flops A and B are in the "1" state. Flip-flop C goes to the "0" state on the clock pulse after flip-flop C was in the "1" state. Flip-flop D complements on the next clock pulse after flip-flop C is in the "1" state.

If the counter is reset, it starts from a count of 0000 and proceeds sequentially to a count of 0100. On the next clock pulse, flip-flop C goes to the "0" state because its 1 output has been keeping flip-flop C's K input HI. Flip-flop D complements because the 1 output of flip-flop C has been keeping the J and K inputs of flip-flop D HI. Flip-flop B does not change state because flip-flop A is in the "0" state disabling the J and K inputs of flip-flop B. Flip-flop A does not change state because flip-flop C was in the "1" state, placing a LO from the 0 side of flip-flop C on the J and K inputs of flip-flop A.

The counter continues sequentially from 1000 to 1100 as a binary counter. On the next clock pulse flip-flop D complements because of a HI supplied from the 1 output of flip-flop C. Flip-flop C goes to the "0" state because its 1 output is fed back to its K input. Flip-flop B does not change because both its J and K inputs receive LO's supplied from gate A. Flip-flop A does not change because both its J and K inputs are disabled with a LO from the 0 output of flip-flop C.

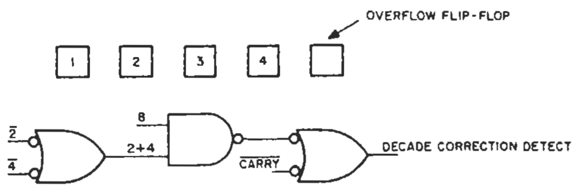
12.



5421 Up/Down Counter

13. In 8421 code, the sum will be correct if it does not exceed 9. If the decimal sum is between 10 and 15, it is necessary to add 6 to the binary sum and generate a carry to the next decade. If the decimal sum exceeds 15, a carry signal is generated by the initial addition, but the correction factor 6 must still be added to the binary sum.

Accumulator Register Flip-Flops

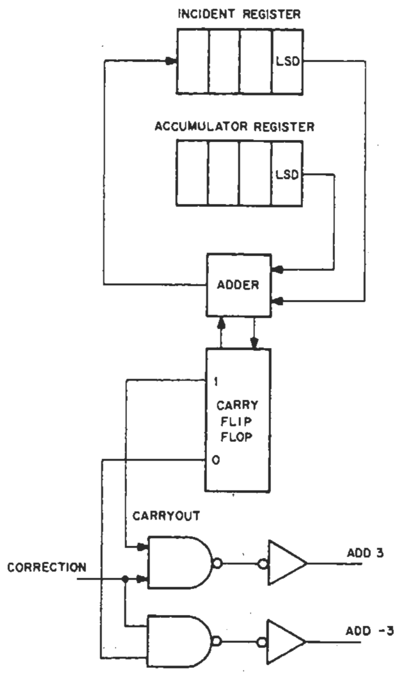


8421 Adder Correction Control

14.	7	0111	5	0101
	+2	0010	+1	0001
	9	1001	6	0110
Correction factor		(NO)		(NO)
	8	1000	9	1001
	+6	0110	+8	1000
	14	1110	17	0001
Correction factor		0110		0110
		0100		0111
		carry to 10' decade		carry to 10' decade

The decade-carry detect circuit should detect sums over 9 and the logic should provide for a carry to the most significant decade when a sum of greater than 9 occurs.

15.



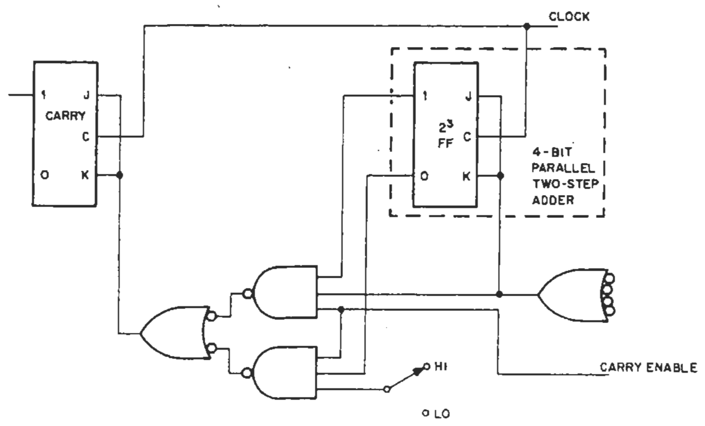
NOTE: CORRECTION MUST BE ENABLED AFTER THE INFORMATION IN THE ACCUMULATOR REGISTERS IS ADDED

16.

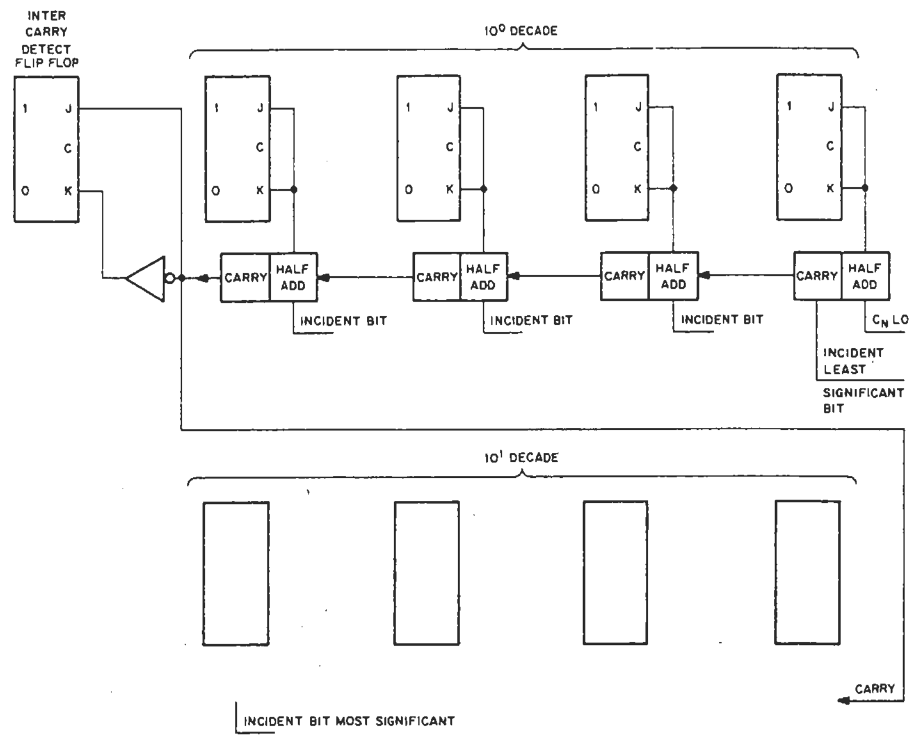
```
        7       1010      5     1000      8      1011                9   1100
       +2        101     +1      100     +6      1001              +8   1011
        9       1111      6     1100    +14      0100   (+Carry)  17   0111 (+Carry)
Correction factor-0011          -0011           +0011                   +0011
                1100            1001             0111                    1010
```

The decade-carry detect circuit should detect sums over 9 and the logic should provide for a carry to a more significant decade when a sum of greater than 9 occurs.

17.



18.



INTER CARRY DETECT FLIP FLOP
10^0 DECADE
1
J
C
O
K
CARRY
HALF ADD
INCIDENT BIT
C_N LO
INCIDENT LEAST SIGNIFICANT BIT
10^1 DECADE
CARRY
INCIDENT BIT MOST SIGNIFICANT

19. 9's Complement Subtraction:

A. $\begin{array}{r} 045 \\ (-)032 \\ \hline 013 \end{array}$ $\qquad$ $\begin{array}{r} 999 \\ 032 \\ \hline 967 \end{array}$ $\qquad$ $\begin{array}{r} 045 \\ 967 \\ \hline 1\ |012 \\ \hookrightarrow 1 \\ \hline 013 \end{array}$

B. $\begin{array}{r} 089 \\ (-)036 \\ \hline 053 \end{array}$ $\qquad$ $\begin{array}{r} 999 \\ 036 \\ \hline 963 \end{array}$ $\qquad$ $\begin{array}{r} 089 \\ 963 \\ \hline 1\ |052 \\ \hookrightarrow 1 \\ \hline 053 \end{array}$

C. $\begin{array}{r} 031 \\ (-)025 \\ \hline 006 \end{array}$ $\qquad$ $\begin{array}{r} 999 \\ 025 \\ \hline 974 \end{array}$ $\qquad$ $\begin{array}{r} 031 \\ 974 \\ \hline 1\ |005 \\ \hookrightarrow 1 \\ \hline 006 \end{array}$

10's Complement Subtraction:

A.
$$\begin{array}{r} 045 \\ (-)032 \\ \hline 013 \end{array} \qquad \begin{array}{r} 999 \\ 032 \\ \hline 967 \\ +\ 1 \\ \hline 968 \end{array} \qquad \begin{array}{r} 045 \\ 968 \\ \text{Carry} \leftarrow \boxed{013} \end{array} \leftarrow \text{Answer (Carry Ignored)}$$

B.
$$\begin{array}{r} 089 \\ (-)036 \\ \hline 053 \end{array} \qquad \begin{array}{r} 999 \\ 036 \\ \hline 963 \\ +\ 1 \\ \hline 964 \end{array} \qquad \begin{array}{r} 089 \\ 964 \\ 1\,\boxed{053} \end{array}$$

C.
$$\begin{array}{r} 031 \\ (-)025 \\ \hline 006 \end{array} \qquad \begin{array}{r} 999 \\ 025 \\ \hline 974 \\ +\ 1 \\ \hline 975 \end{array} \qquad \begin{array}{r} 031 \\ 975 \\ 1\,\boxed{006} \end{array}$$

20.

	4	6	9's Complement	5	3
Excess 3	0111	1001		1000	0110

To find the 9's complement of an excess 3 number, complement all digits.

21.

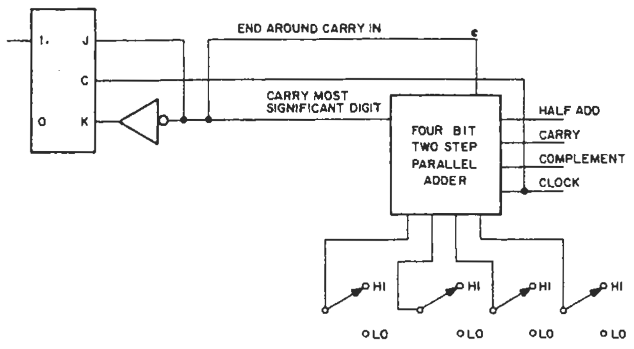
	3	0	10's Complement	7	0
Excess 3	0110	0011		1010	0011

To find the 10's complement of an excess 3 number find the 9's complement and add one count to the entire number in excess 3 fashion, not in binary fashion and not to each decade.

Decimal	Excess 3	9's Comp, Dec.	9's Comp, XS 3	10's Comp, Dec.	10's Comp, XS 3
+58	0 1000 1011	-41	1 0111 0100	-42	1 0111 0101
+59	0 1000 1100	-40	1 0111 0011	-41	1 0111 0100
+60	0 1001 0011	-39	1 0110 1100	-40	1 0111 0011
+61	0 1001 0100	-38	1 0110 1011	-39	1 0110 1100

Examples of 9's and 10's Complement

22.



23. Addition of 8421-coded numbers has the disadvantage that a carry signal can be generated during the correction process. For this reason, each decade in the adder has to be corrected individually.

24.

$$\text{8421 Code}\quad \overbrace{0100}^{4}\ \overbrace{0110}^{6} \xrightarrow{\text{9's Complement}} \overbrace{0101}^{5}\ \overbrace{0011}^{3} \xrightarrow{\text{10's Complement}} \overbrace{0101}^{5}\ \overbrace{0100}^{4}$$

$$\text{8421 Code}\quad \overbrace{0011}^{3}\ \overbrace{0000}^{0} \xrightarrow{\text{9's Complement}} \overbrace{0110}^{6}\ \overbrace{1001}^{9} \xrightarrow{\text{10's Complement}} \overbrace{0111}^{7}\ \overbrace{0000}^{0}$$

Excess 3 is more convenient for BCD arithmetic operations than 8421 because negation may be accomplished simply.

CHAPTER 9
CODE CONVERSION AND DECODING

INTENT AND APPROACH

Chapter 9 groups the BCD codes introduced in Chapter 8 into main classifications and explains the relationships, applications, and advantages of these codes.

In Section VI of this chapter, code conversion is introduced. The student is given examples of code converters and is required to design various code converters. The methods of code converter design outlined in this chapter follow the pattern of previous design methods. The student should initially determine the circuit requirements; in the case of a converter, the initial requirements are what code conditions must be detected. Secondly, the student must determine what final state should be enabled in the converter when the convert clock pulse occurs.

Experiment 9.1 The 2421 to 8421 Converter and Experiment 9.2 The 5421 to 8421 Converter present thorough examples of code conversion. Questions 6 and 7 on page 113, COMPUTER LAB Workbook, should not be assigned to the student until he understands the basic principles and design method of the converters discussed in Experiments 9.1 and 9.2. Supplementary questions are available on page 117 of the COMPUTER LAB Workbook.

OUTLINE

I. INTRODUCTION (Page 105)

II. DECIMAL CODES (Pages 105-108)
   A. Code Possibilities
   B. Desirable Features
   C. Count Sequences
   D. 8421 Code
   E. Excess 3 Code
   F. 2421 Code
   G. Pictorial Representation of 8421, 2421, and Excess 3 Codes
      1. Unused segments
      2. Symmetry
   H. Counting

I. Arithmetic Operations

1. Symmetrical code groups

2. 8421 code

J. Up/Down Counting

K. Digital-to-Analog Conversion

L. Summary

III. REFLECTIVE CODES (Pages 108-109)

A. Advantages

B. Gray Binary Code Count Sequence

C. Applications

IV. ERROR-DETECTING CODES (Page 109)

A. Parity Bit

B. Hamming Code

V. MISCELLANEOUS CODES (Page 110)

A. Codes for Specific Applications

B. ASCII Code

VI. CONVERTERS (Pages 110-117)

A. Introduction

B. Method of Design

C. Experiment 9.1: The 2421 to 8421 Converter (Pages 110-111)

1. Comparison of code sequences

2. Circuit construction

3. Logic diagram (Page 112)

4. Operating instructions

5. Verification of operation according to test problems

D. Questions 1-3 (Pages 111-113)

E. Experiment 9.2: The 5421 to 8421 Converter (Page 113)

1. Corresponding code states

2. Identification

3. Changes

4. Logic diagram (Page 114)

5. Circuit construction

6. Verification of operation according to test problems

F. Questions 4-7 (Page 113)

G. Experiment 9.3: Gray to Binary Converter (Pages 115-117)

1. General conversion rule
2. Count sequences
3. Circuit construction
4. Logic diagram
5. Verification of operation according to test problems

H. Questions 8-9 (Page 117)

I. Supplementary Questions 10-11 (Page 117)

VII. BINARY TO BCD CONVERSION (Page 117)

A. Simultaneous Count Method

B. Converting Individual Decades

C. Supplementary Questions 12-13 (Page 117)

ANSWER KEY

Questions

1. Referring to Figure 9.5 (page 112 in the COMPUTER LAB Workbook), the operation of the 2421-8421 converter can be explained by discussing what each gate detects, and what each gate does.

Gate E detects a 2421 code for 5 (flip-flop D = 1 and flip-flop C = 0). When gate E is enabled, the next clock pulse will cause flip-flop D to go to the "0" state, flip-flop C to go to the "1" state, and flip-flop B to go to the "0" state.

Gate F detects 2421 codes for 6 and 7 (flip-flop D = 1 and flip-flop B = 0). When gate F is enabled, the next clock pulse will cause flip-flop D to go to the "0" state and flip-flop B to go to the "1" state.

Gate G detects 2421 codes for 8 and 9 (flip-flops D, C, and B in the "1" state). When gate G is enabled, the next clock pulse will cause flip-flops B and C to go to the "0" state.

2. Gate E is detecting the decimal number 5 in the 2421 code. The number 5 is the only 2421 code sequence that has C input = 0 and D input = 1.

Gate F is detecting the decimal numbers 6 and 7 in the 2421 code. The numbers 6 and 7 are the only 2421 code sequence that has B input = 0 and D input = 1.

3. The 2421 number is read into the 2421-8421 Converter by parallel transfer. Prior to clock pulse time, the switch inputs are set to the input 2421 configuration. A "1" is equal to a LO switch position; "0" is equal to a HI switch position. Placing a switch in the LO position "steers" an associated flip-flop to the "1" state by placing a HI on the J input. At clock pulse time, all flip-flops with their corresponding switch inputs LO will change to the "1" state. This type of parallel transfer is called a 1's transfer and requires a cleared receiving register prior to the transfer.

4. Referring to Figure 9.7 (page 114 in the COMPUTER LAB Workbook), the operation of the 5421-8421 Converter can be explained by discussing what each gate detects, and what each gate accomplishes when enabled.

Gate I detects the 5421 code for 9 (flip-flop D = 1 and flip-flop C = 1). When gate D is enabled, the next clock pulse will cause flip-flop C to go to the "0" state and flip-flop A to go to the "1" state.

Gate H detects the 5421 code for 8 (flip-flops D, B, and A in the "1" state). When gate H is enabled, the next clock pulse will cause flip-flops B and A to go to the "0" state.

Gate G detects the 5421 code for 7 (flip-flops D and B in the "1" state and flip-flop A in the "0" state). When gate F is enabled, the next clock pulse will cause flip-flops A and C to go to the "1" state and flip-flop D to go to the "0" state.

Gate F detects the 5421 code for 6 (flip-flops D and A in the "1" state and flip-flop B in the "0" state). When gate F is enabled, the next clock pulse will cause flip-flops B and C to go to the "1" state and flip-flops A and D to go to the "0" state.

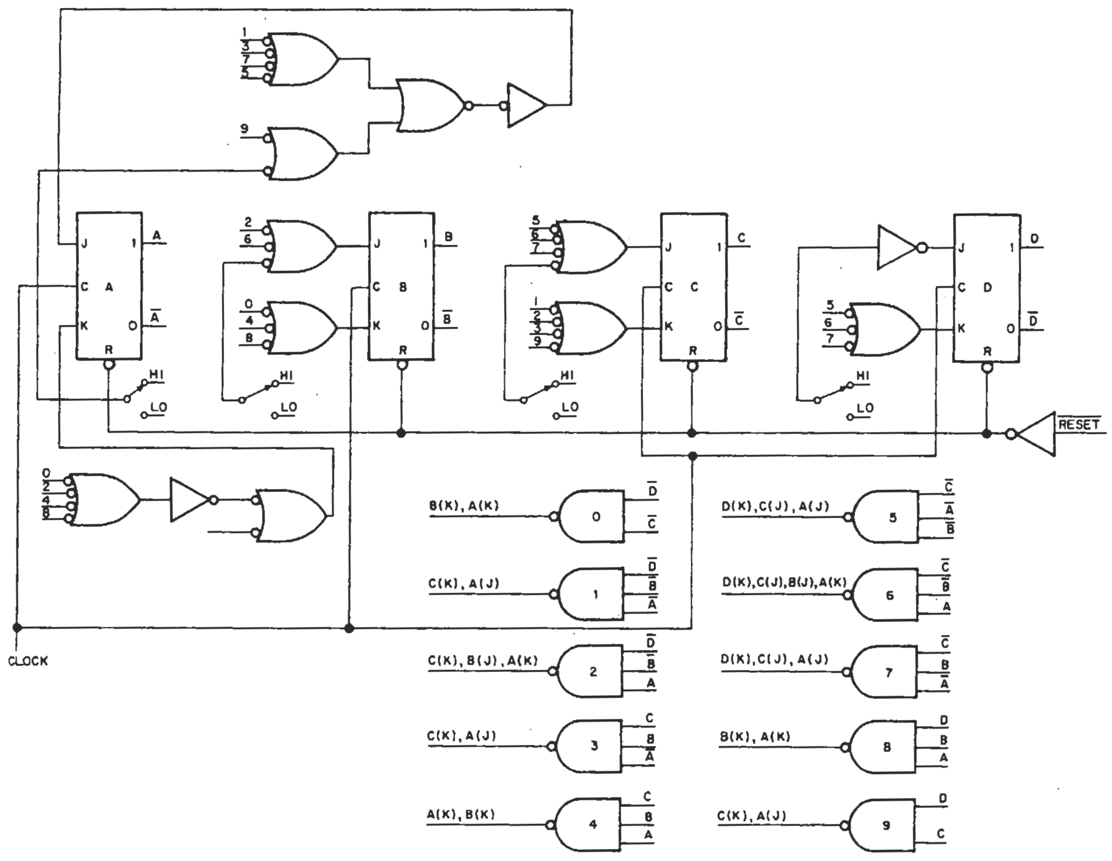
Gate E detects the 5421 code for 5 (flip-flop D in the "1" state and flip-flops C, B, and A in the "0" state). When gate E is enabled, the next clock pulse will cause flip-flop D to go to the "0" state and flip-flops A and C to go to the "1" state.

5. After the information has been transferred from the switch register to the accumulator register, the switches should be placed in the HI position. Placing the switches in the HI position prevents the switch contents from affecting the J inputs of the flip-flops during the conversion from 5421 to 8421.

6.

Excess 3				8421				Decimal
D	C	B	A	D	C	B	A	
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	1	1
0	1	0	1	0	0	1	0	2
0	1	1	0	0	0	1	1	3
0	1	1	1	0	1	0	0	4
1	0	0	0	0	1	0	1	5
1	0	0	1	0	1	1	0	6
1	0	1	0	0	1	1	1	7
1	0	1	1	1	0	0	0	8
1	1	0	0	1	0	0	1	9

	Identify By	Change To
0	D = 0, C = 0	B = 0, A = 0
1	D = 0, B = 0, A = 0	C = 0, A = 1
2	D = 0, B = 0, A = 1	C = 0, B = 1, A = 0
3	C = 1, B = 1, A = 0	C = 0, A = 1
4	C = 1, B = 1, A = 1	A = 0, B = 0
5	C = 0, B = 0, A = 0	D = 0, C = 1, A = 1
6	C = 0, B = 0, A = 1	D = 0, C = 1, B = 1, A = 0
7	C = 0, B = 1, A = 0	D = 0, C = 1, A = 1
8	D = 1, B = 1, A = 1	B = 0, A = 0
9	D = 1, C = 1	C = 0, A = 1

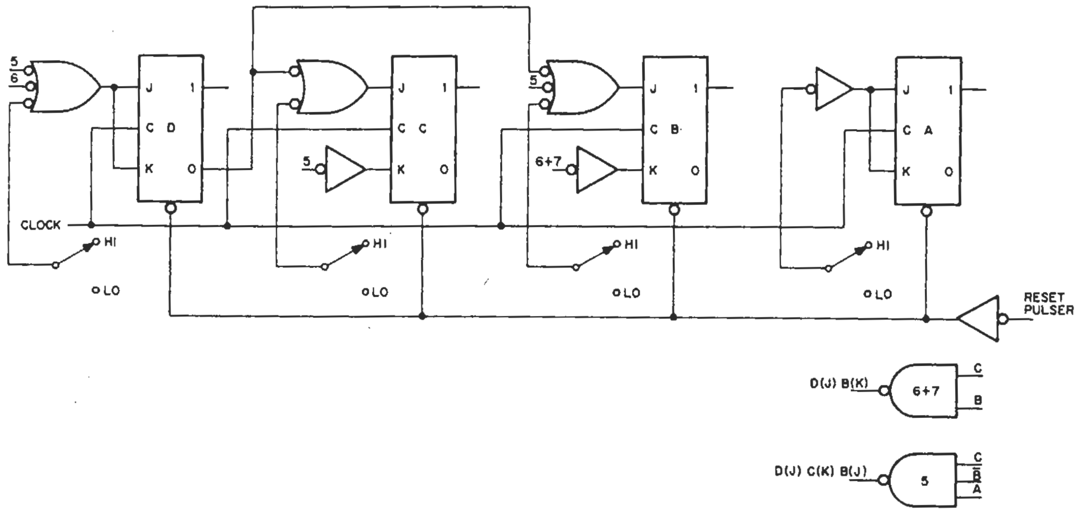


Excess 3 to 8421 Code Converter

7. 8421 to 2421 Code Converter

8421				2421				
D	C	B	A	D	C	B	A	Decimal
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	0	2
0	0	1	1	0	0	1	1	3
0	1	0	0	0	1	0	0	4
0	1	0	1	1	0	1	1	5
0	1	1	0	1	1	0	0	6
0	1	1	1	1	1	0	1	7
1	0	0	0	1	1	1	0	8
1	0	0	1	1	1	1	1	9

	Identify By	Change To
5	$C\bar{B}A$	D = 1, C = 0, B = 1
6	CB	D = 1, B = 0
7	CB	D = 1, B = 0
8	D	C = 1, B = 1
9	D	C = 1, B = 1



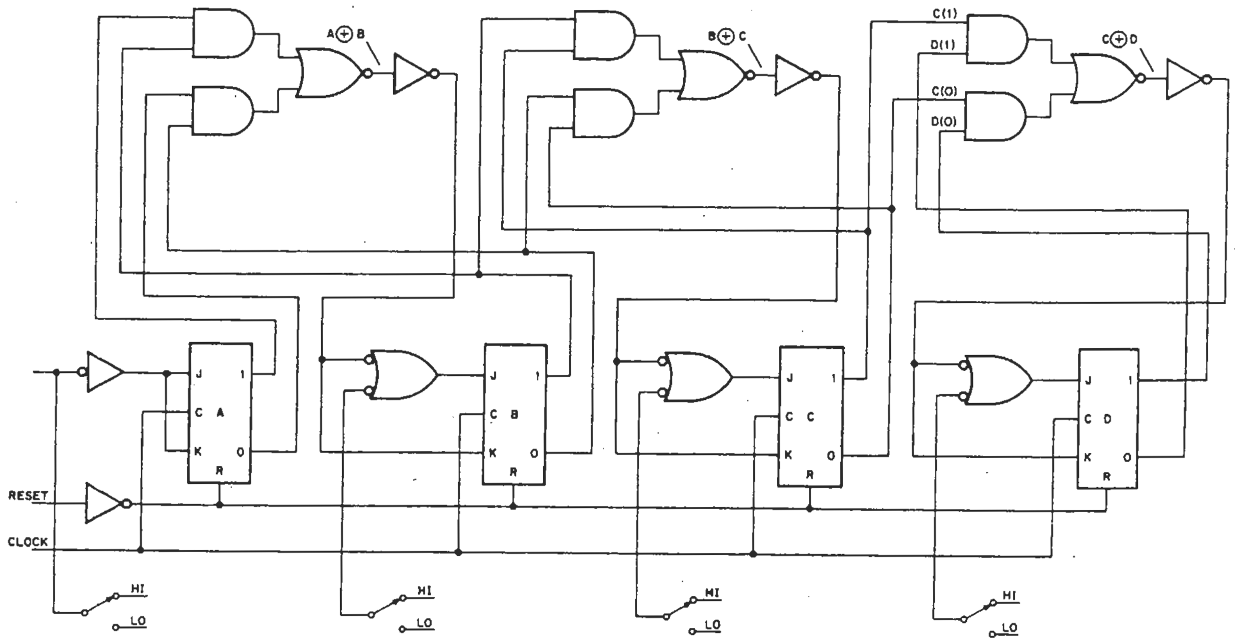
8. $E = A = B$
$F = E \oplus C$
$G = F \oplus D$

9. From the trailing edge of the "transfer in" clock pulse to the leading edge of the "convert" clock should be a minimum of 140 ns (include 35 ns flip-flop delay).

10.a The most significant digit is identical in standard binary and reflected gray code.

10.b If the most significant binary digit is a 1 before conversion, complement the next least significant binary digit to get its equivalent gray code value.

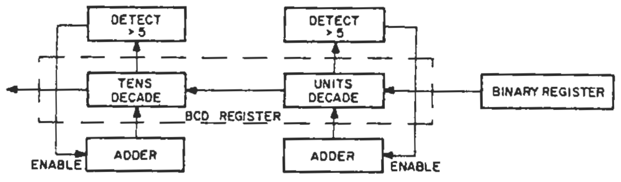
10.c $A(1) = A(0)$
$B(1) = B(0) \oplus A(0)$
$C(1) = C(0) \oplus B(0)$
$D(1) = D(0) \oplus C(0)$



Binary to Gray Converter

Questions 12 and 13

Section VII Computer Lab Workbook, Binary To BCD Conversion is in error, and will be revised at a later date. The following material explains one method of binary to BCD conversion and answers questions 12 and 13.



The technique is to shift the number serially from the binary register into the BCD register, shifting the high order binary bit into the low order BCD bit. (Sufficient BCD decades must be provided to hold the BCD equivalent of the binary value.) In addition to simply shifting, each decade value must be tested to see if the value it holds is 5 or greater. For each decade holding a 5 or greater, its value must be increased by 3. The function of checking the value and adding 3 occurs prior to each shift. (Since at least 3 shifts must be made before the least significant decade could contain a 5 or larger value, the tests prior to the first 3 shifts need not be made.)

The logic sequence goes as follows:

```
3 shifts left
   test decades for ≥ 5
   add 3 to "guilty" decades
 shift left          — — — — — —⎫
   test decades for ≥ 5         ⎪  repeat for each
   add 3 to "guilty" decades    ⎬  remaining bit in
 shift left          — — — — —  ⎭  the binary register
```

Note the following example:

	BCD		BINARY	
100's	10's	1's		
			11010010	
		110	10010	3 shifts left
		1001	10010	add 3 to 1's
	1	0011	0010	shift left
	10	0110	010	shift left
	10	1001	010	add 3 to 1's
	101	0010	10	shift left
	1000	0010	10	add 3 to 10's
1	0000	0101	0	shift left
1	0000	1000	0	add 3 to 1's
10	0001	0000		shift left

finally $210_{10} = 11010010_2$

A Binary to Excess 3 conversion could be constructed by adding 3 to the information obtained in each decade of the BCD Register of the Binary To BCD (8421) Converter illustrated above.

CHAPTER 10
SYSTEM CONSIDERATIONS

INTENT AND APPROACH

The objectives of Chapter 10 are to explain logic control, synchronization and frequency considerations.

Under control considerations, an Adder Control for the Two Step Parallel Adder of Chapter 7 is explained and a control design method is outlined.

Synchronization is discussed using the Synchronous Up/Down Counter developed in Chapter 5. Experiment 10.2 illustrates how a J-K flip-flop can be used to control and synchronize the Up/Down Counter.

Section IV, System Frequency, requires the use of an oscilloscope. In this section, counter frequencies up to 10 MHz are measured and individual J-K flip-flop delays are determined.

OUTLINE

B. Experiment 10.2: Synchronizers (Page 123)

1. Logic diagram (Page 125)

2. Circuit construction

C. Questions 5-7 (Pages 123-126)

IV. SYSTEM FREQUENCY (Pages 126-127)

A. Timing Diagrams

B. Experiment 10.3: Maximum Frequency (Page 126)

1. Circuit construction (8-Bit Counter)

2. Operation of counter at various speeds

C. Questions 8-11 (Page 126)

ANSWER KEY

Questions:

1. Figure 10.3 (page 122 in the COMPUTER LAB Workbook) is a 2-Bit Asynchronous Self-Stopping Counter. After four counts or a count of $11_{(2)}$, the counter will stop. At a count of $01_{(2)}$, HALF ADD ENABLE goes HI. At a count of $10_{(2)}$, CARRY ENABLE goes HI.

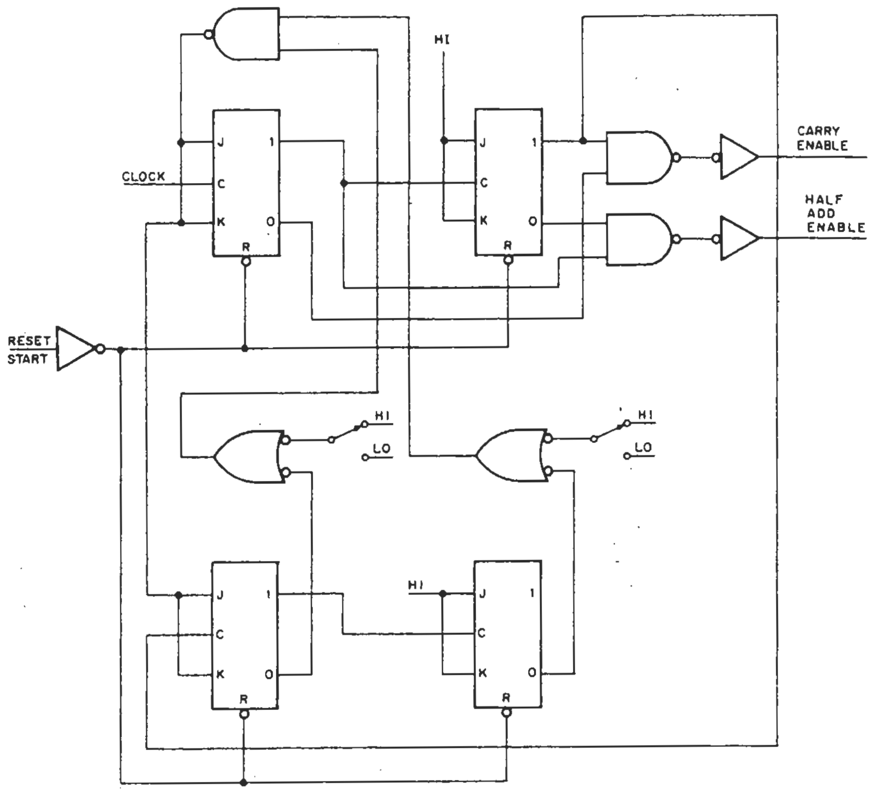
Assuming that the counter has been reset to $00_{(2)}$, both gates A and B are disabled and both CARRY ENABLE and HALF ADD ENABLE are LO. The first clock pulse causes the least significant flip-flop to complement (left flip-flop). With a count of $01_{(2)}$ in the counter, gate A is enabled. The next clock pulse causes the counter to go to a count of $10_{(2)}$, enabling gate B and CARRY ENABLE. The third clock pulse causes the counter to go to a count of $11_{(2)}$, disabling gate B, gate A, CARRY ENABLE, and HALF ADD ENABLE. The counter going to a count of $11_{(2)}$ also enables the NAND gate feeding J and K of the least significant flip-flop, halting the counter by disabling the J and K inputs of the least significant bit with a LO level.

Control action may be resumed by resetting the counter.

2. The $00_{(2)}$ condition of the counter is not used as a control operation because the $00_{(2)}$ condition is not synchronized with the clock.

If the $00_{(2)}$ were an enabling condition, the length of the enabled condition would vary according to how long a RESET signal was present.

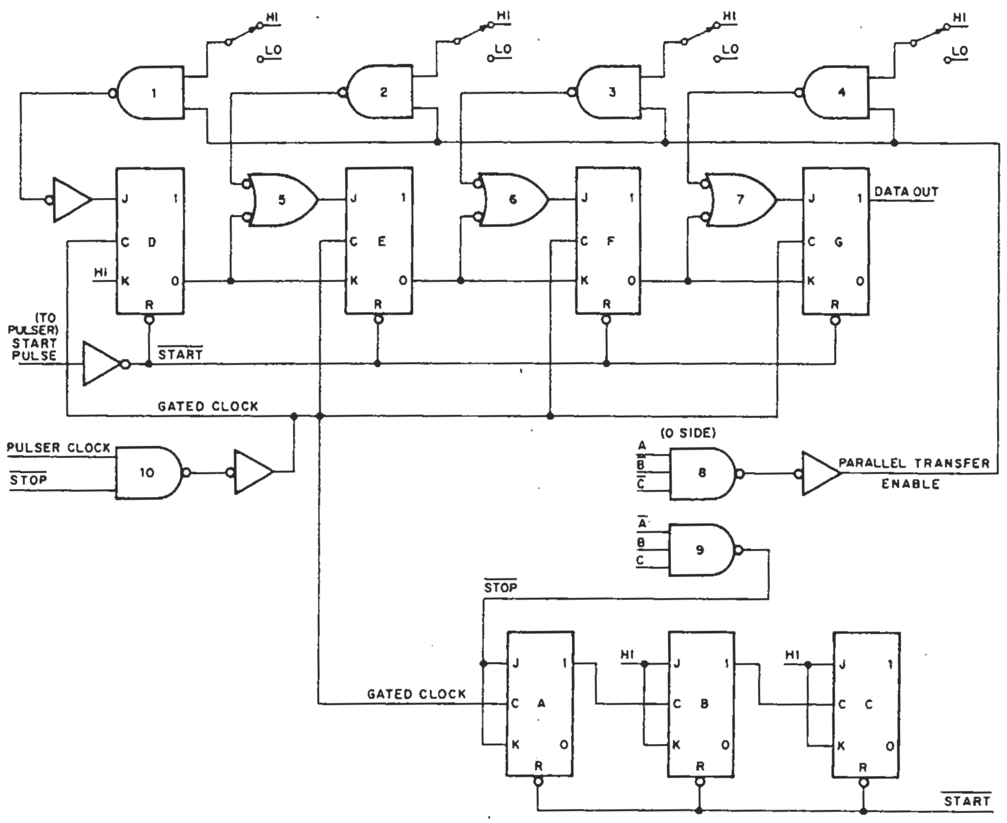
3.



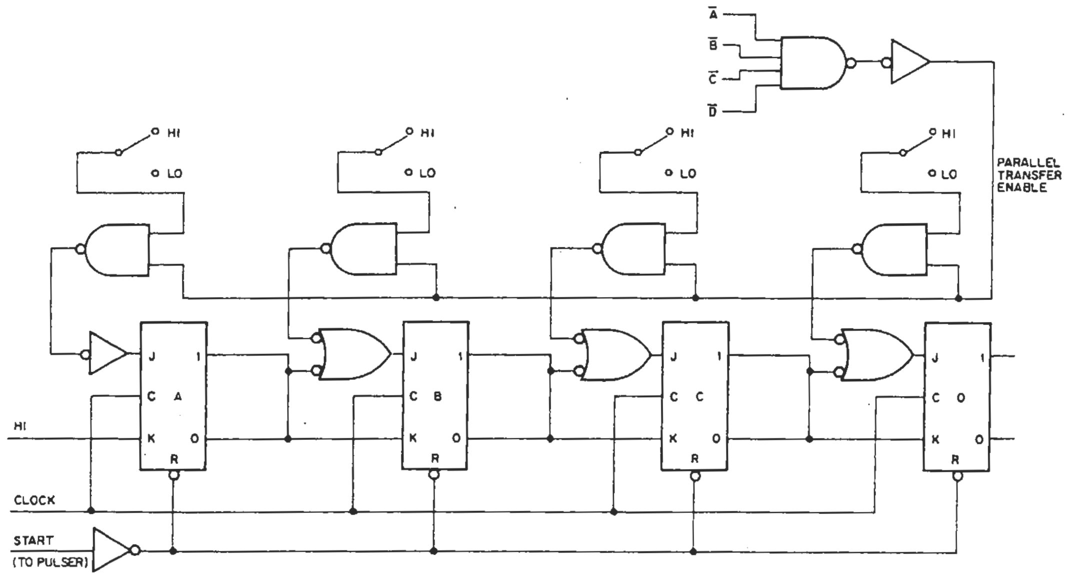
To use the multiplier control, perform the following steps:

1. Hold RESET pressed (Start)
2. Place the multiplicand in the switches
3. Select a multiplier with the control switches LO = 0 HI = 1
4. Release RESET

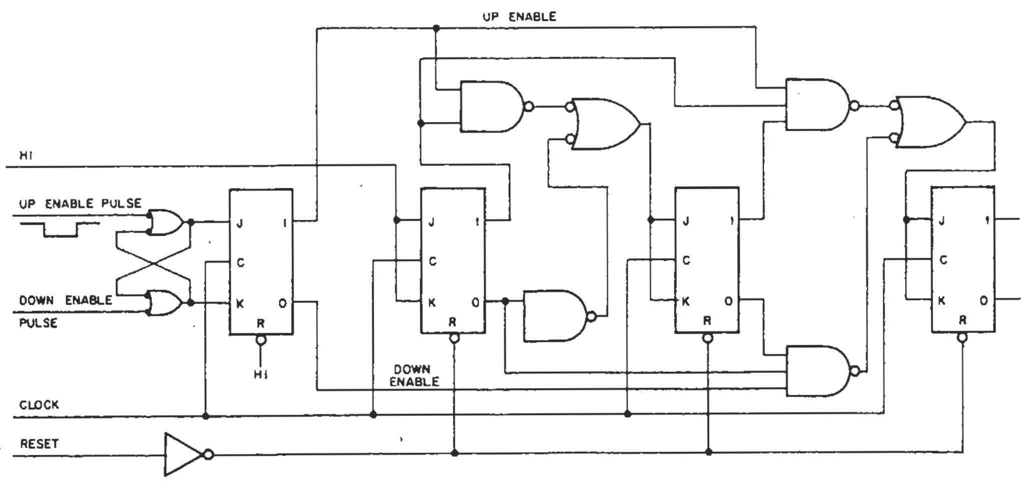
4. Method 1: HI = 1



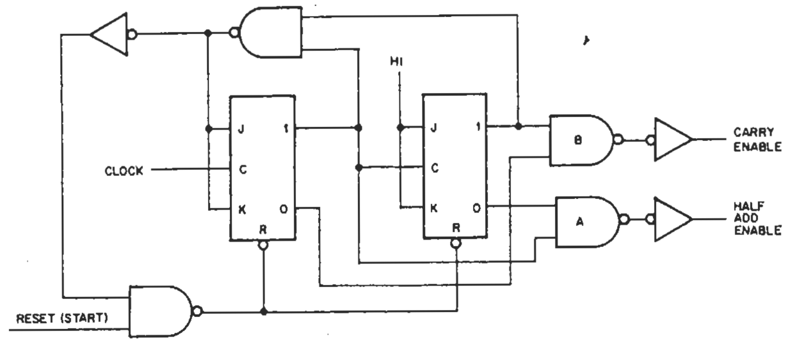
Method 2: HI = 1



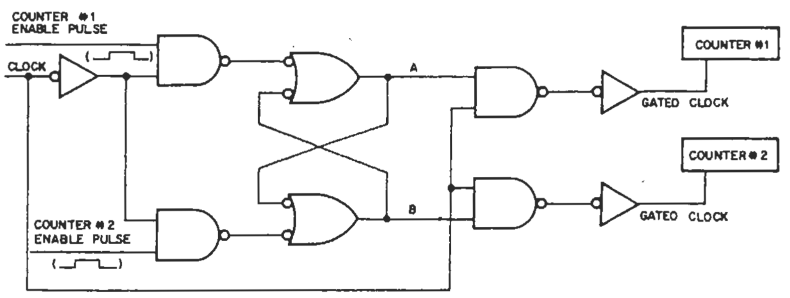
5.



6.



7. (The following logic assumes counter enable pulses will be synchronized with the clock pulse input.)

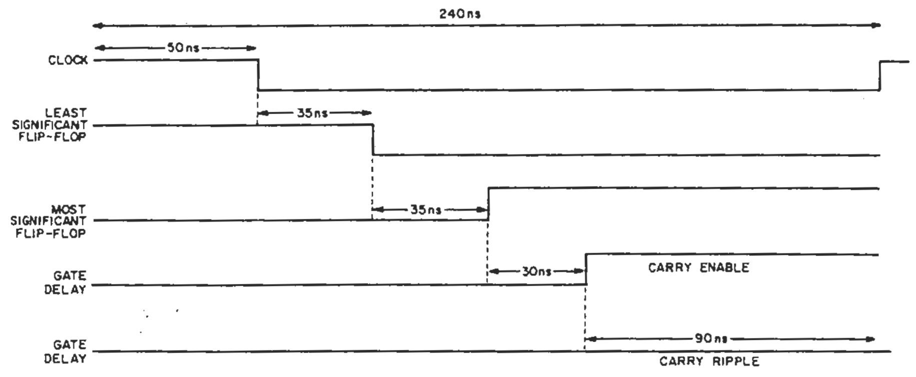


8. No. At higher frequencies the ripple delay is too great and the counter will over count by 1 or 2 counts.

9.a An acceptable range is from 2.9 → 4.4 MHz (8-Bit Counter)

9.b 37 → 22 ns is the corresponding range of propagation delays.

10.a (Using Parallel Adder Figure 10.1)



10.b Approximately 4.1 MHz.

11. The controlled synchronous up/down counter, Figure 10.5, will operate at a maximum frequency of approximately 8.6 MHz.

COURSE PLANNING

I. Academic High School

General Objectives

The general objectives for including computer training in an academic high school program would be to enhance existing programs in mathematics, physics, and logic and to provide the student with the basic knowledge of computer concepts that he will need in both future academic and vocational endeavors. Basically, Computer training provides the college bound student with an opportunity for engineering experience early in his academic career. This experience will enable the student to make a decision concerning engineering as a vocation while he is still in high school where such a choice entails less expense than it would in college.

Possible Course Areas

Computer Science - The Computer Science course is a one or two semester course. The first semester consists of the study of the internal building blocks of a digital computer; the second semester deals with the functional organization of a computer, its operation, and programming.

Boolean Algebra - Using the COMPUTER LAB Workbook material, Boolean algebra can be a one semester course that teaches the basic Boolean algebra laws, theorems, postulates, and simplification techniques. The COMPUTER LAB can be used to demonstrate these Boolean algebra principles.

Number Systems - The COMPUTER LAB can also be a supplement to existing mathematics courses. Binary and octal number systems can be vividly demonstrated by having the student build such devices as counters, decoders, and encoders.

Physics - The COMPUTER LAB can supplement existing courses in physics. Detailed electronic analyses of the COMPUTER LAB elements can be presented and laboratory experiments can then be performed.

Logic - The COMPUTER LAB is a valuable supplement to existing courses in logic. Abstract logical concepts can be demonstrated using COMPUTER LAB electronic devices.

Computer Science Course Outline

Semester One
18 Weeks - 5 Hours/Week

Time	Major Heading	References
Weeks 1 & 2	Introduction to Computers Introduction to the COMPUTER LAB Introduction to Two-State Devices Gates COMPUTER LAB Functions Binary Numbers Decimal-Binary Conversion Octal Numbers Decimal-Octal Conversion	W.B.* PP VII-VIII, Chap. 1 T.G.** PP 1 to 21
Weeks 3 & 4	AND/NOR Gate AND/NOR - NOR Application AND/NOR - Comparator AND/NOR - Exclusive OR Non-Inverting Gates	W.B. Chap. 2 T.G. PP 23 to 33
Weeks 5, 6 & 7	R-S Flip-Flop Clocked R-S Flip-Flop J-K Flip-Flop 4-Bit Shift Register	W.B. Chap. 3 T.G. PP 35 to 52
Weeks 8, 9, & 10	Introduction to Boolean Algebra Boolean Operators Boolean Expressions & Logic Diagrams	W.B. PP 45-48 Figure 4.3 on P 49 T.G. PP 53 to 60, 77 to 89
Weeks 11, 12, & 13	Asynchronous Binary Up Counter Modified Asynchronous Binary Counter Synchronous Binary Up Counter Synchronous Binary Up/Down Counter Synchronous Modulo 6 Binary Counter	W.B. Chap. 5 T.G. PP 91 to 104
Weeks 14, 15, & 16	Binary Addition Serial Addition Binary Subtraction Serial Subtraction	W.B. Chap. 6 T.G. PP 105 to 116
Weeks 17 & 18	Two Step Parallel Addition Single Step Parallel Addition	W.B. PP 87-92, Experiment 7.4 on P 93 T.G. PP 117 to 125

*W.B. - COMPUTER LAB Workbook
**T.G. - COMPUTER LAB Teacher's Guide

Semester Two

18 Weeks - 5 Hours/Week

Time	Major Heading
Week 1	Computer Functional Units Computer Magnetic Core Memories
Week 2	Input-Output Media (example: punched cards, paper tape, magnetic tape) Alpha Numeric Codes (example: Hollerith, 7-bit punched tape, 5 and 8 channel code, 7-bit magnetic tape)
Week 3	Input-Output Devices (example: card readers-punches, paper tape readers-punches, magnetic tape units, cathode ray tubes)
Week 4	Introduction to Digital Computer Programming
Weeks 5-8	Machine Language Programming Computer Operation Logic Diagrams
Weeks 9-12	Computer Conversational Languages
Weeks 13-18	Programming Projects

Computer Science Course

Semester One, Final Exam (Closed Book)

________ 1. In computer terminology, the word accumulator is most likely to be associated with:

A. Memory
B. Arithmetic unit
C. Magnetic tape
D. Card punch

________ 2. The logical NAND function is described in which of the following truth tables (HI = true, LO = false):

A.

Inputs		
A	B	Output
LO	LO	LO
LO	HI	HI
HI	LO	HI
HI	HI	HI

B.

Inputs		
A	B	Output
LO	LO	LO
LO	HI	HI
HI	LO	HI
HI	HI	LO

C.

Inputs		
A	B	Output
LO	LO	HI
LO	HI	HI
HI	LO	HI
HI	HI	LO

D.

Inputs		
A	B	Output
LO	LO	HI
LO	HI	LO
HI	LO	LO
HI	HI	LO

________ 3. A COMPUTER LAB logic element (gates and flip-flops) can drive:

A. 10 loads
B. 20 loads
C. 30 loads
D. 1 load

________ 4. Converting $13_{(10)}$ to binary will produce:

A. 1101
B. 1001
C. 1111
D. 1000

________ 5. The quantity $6737_{(8)}$ is equal to:

A. $4000_{(10)}$
B. $110\ 111\ 011\ 111_{(2)}$
C. $7435_{(10)}$
D. $4095_{(10)}$

________ 6. The following chart illustrates an octal number in a binary form, a 1's complement form, and a 2's complement form. The chart assumes an eight bit register is being used for storage. Select the incorrect section of the chart.

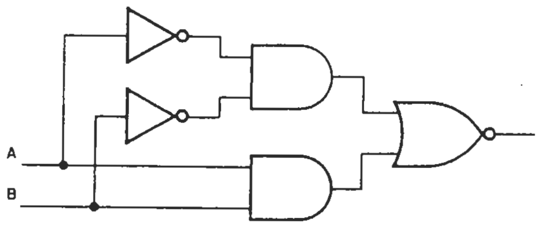
	Number	Binary Form	1's Complement Form	2's Complement Form
A.	$23_{(8)}$	00010011	11101100	11101101
B.	$123_{(8)}$	01010011	10101101	10101110
C.	$17_{(8)}$	00001111	11110000	11110001
D.	$177_{(8)}$	01111111	10000000	10000001

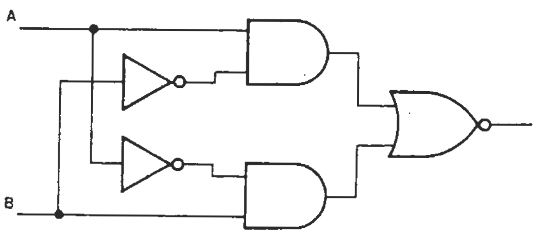
________ 7. Select the correct way in which a computer would perform 2's complement subtraction on the following numbers. Assume the numbers are in the computer in their proper form.

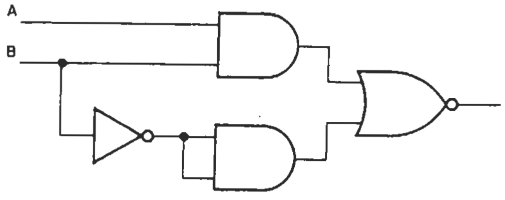
$$\begin{array}{r} 125_{(8)} \\ (-)\text{-}32_{(8)} \\ \hline \end{array}$$

A. 2's complement the subtrahend and directly add
B. 2's complement the subtrahend and minuend and subtract
C. 2's complement the minuend and add
D. 2's complement the minuend and subtract

________ 8. Which of the following logic configurations will produce an HI output when A = B?

A.


B.


C.


________ 9. An Exclusive OR function will yield a true output when:

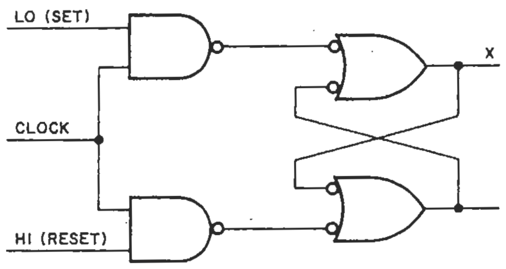
A. all inputs are the same
B. the inputs are different
C. any input is true
D. any input is false

________ 10. The major disadvantage of an R-S flip-flop is:

A. the high cost
B. long propagation delays
C. the very low frequency response
D. the possibility of an indeterminate condition

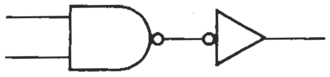
________ 11. Referring to the diagram shown below, point X will be at what level after the next clock pulse?

A. HI
B. Indeterminate
C. LO



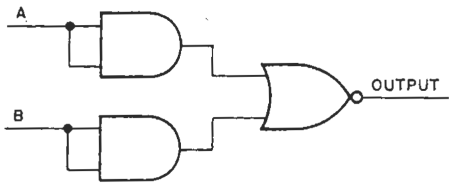
________ 12. The following logic diagram performs:

A. an AND function
B. an OR function
C. an Exclusive OR function
D. a comparator function



________ 13. The following logic diagram performs what logic function?

A. Exclusive OR function
B. Comparator function
C. OR function
D. NOR function



________14. A Master-Slave J-K flip-flop output changes state on the:

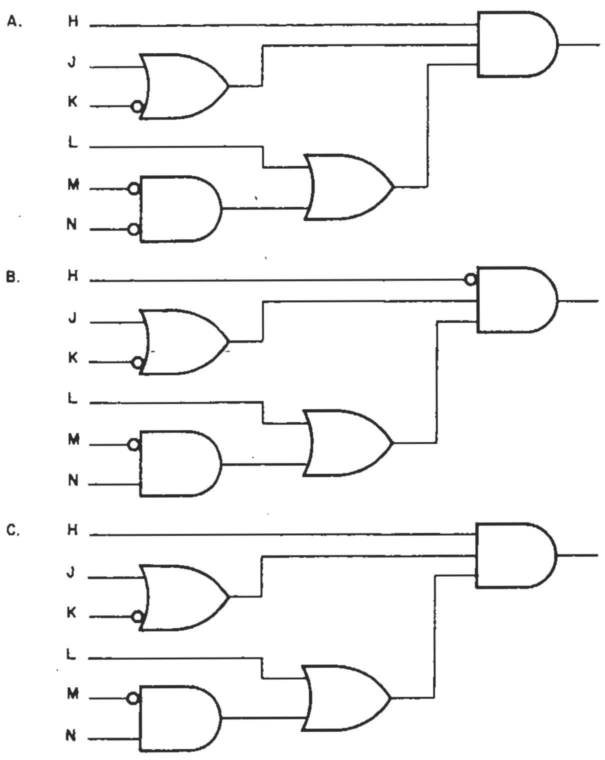
A. leading edge of a clock pulse
B. leading edge of a J input level change
C. trailing edge of a clock pulse
D. leading edge of a K input level change

________15. H $(J+\overline{K})$ $(L+\overline{M}N)$ is best represented by which of the following diagrams:

A. H
J
K
L
M
N

B. H
J
K
L
M
N

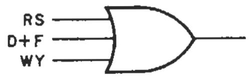
C. H
J
K
L
M
N



________16. What is the best expression for the following logic symbol?

A. $(R\overline{S})$ $(D+F)$ (WY)
B. $R\overline{S} + (D+F)WY$
C. $R\overline{S} + D + F + WY$
D. $R\overline{S} + (D+F) + WY$

RS
D+F
WY

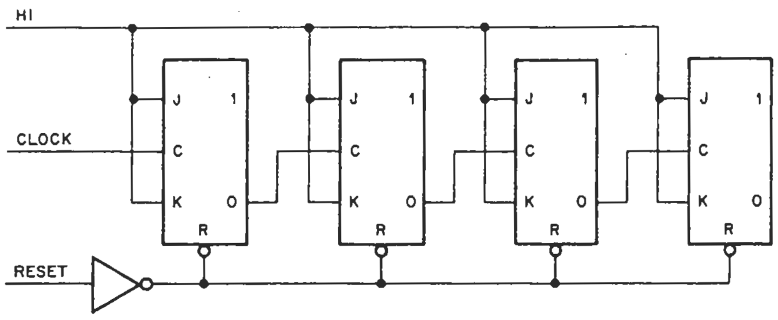


________17. Which of the following terms is <u>not</u> used in Boolean algebra?

A. And
B. Or
C. But
D. Not

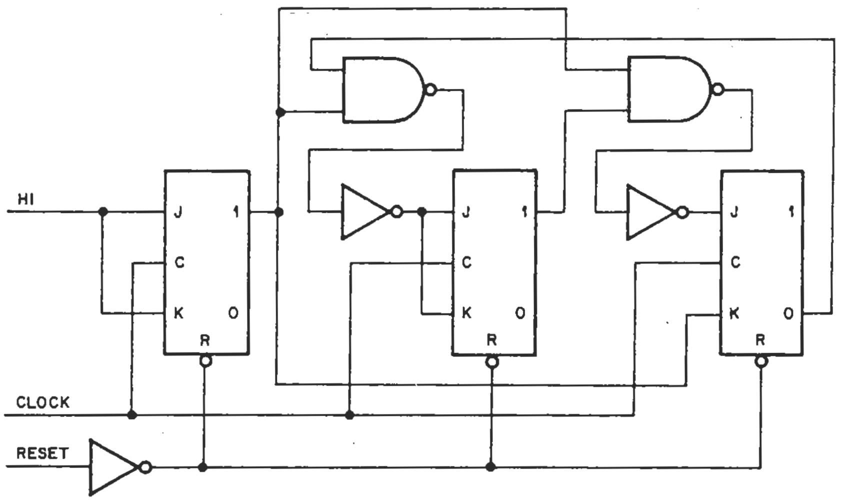
________18. What function does the counter shown below perform?

A. Synchronous Up Counter
B. Synchronous Down Counter
C. Asynchronous Up Counter
D. Asynchronous Down Counter



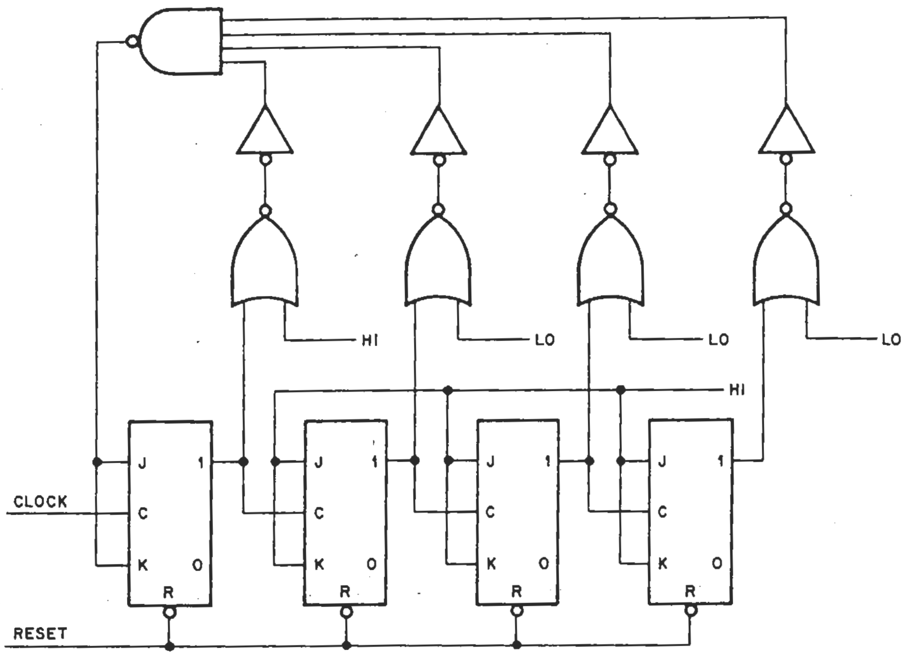
——————19. What is the maximum binary count that the following counter reaches before recycling?

A. 0111
B. 0101
C. 0110
D. 0011



——————20. What number is in the following counter when it stops?

A. 1
B. 7
C. 14
D. 8



________ 21. When adding the binary numbers 101 and 100 in a three-bit serial adder, how many clock pulses are required before the answer appears in the AC?

A. 6
B. 3
C. 2
D. 1

________ 22. In a full adder, a sum is generated when which <u>one</u> of the following conditions is true:

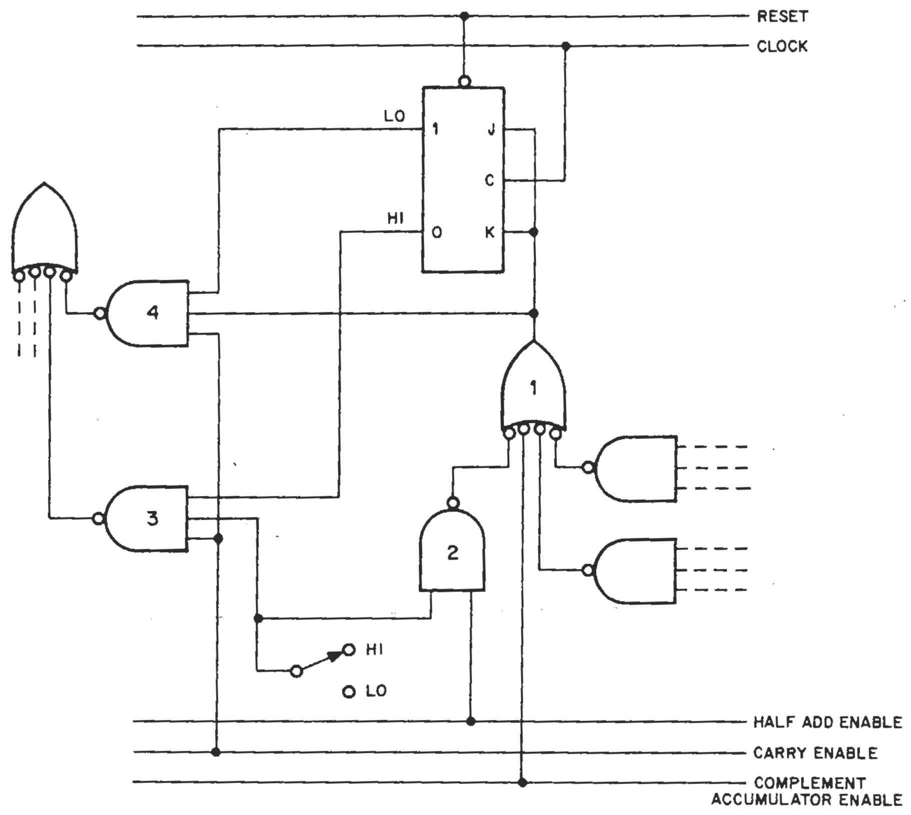
A. Carry in = 1; incident bit = 1; AC = 1
B. Carry in = 0; incident bit = 1; AC = 1
C. Carry in = 1; incident bit = 0; AC = 1
D. Carry in = 0; incident bit = 0; AC = 0

________ 23. In a parallel adder, the half-add function is equivalent to a (an):

A. OR function
B. Exclusive OR function
C. NAND function
D. Negated input AND function

________ 24. In the following two-step adder, assuming that half-add has already occurred, when carry occurs, which of the following statements will be true:

A. AC Bit N will complement
B. Gate 4 will be enabled
C. Gate 3 will be enabled
D. Gate 2 will be enabled



_______ 25. Which one of the following statements is true:

A. Two's complement adder logic is simpler to construct than one's complement adder logic.

B. Doing two's complement arithmetic in a serial adder requires more steps than does one's complement arithmetic.

C. It is easier to obtain the two's complement of a number than the one's complement.

D. End around carry logic is necessary for two's complement addition.

Computer Science Course

Semester One, Final Exam
Answer Key

1.	B	14.	C
2.	C	15.	C
3.	A	16.	D
4.	A	17.	C
5.	B	18.	D
6.	B	19.	B
7.	A	20.	C
8.	B	21.	B
9.	B	22.	A
10.	D	23.	B
11.	C	24.	C
12.	A	25.	A
13.	D		

II. Digital Computer Technology Programs

General Objectives

The objective of the computer technology program is to prepare individuals for employment in the computer field.

In this section, computer technology course material is outlined, and prerequisites and additional material are suggested.

The technology program is divided into instructional blocks. The omission, arrangement and training time of instructional blocks are dependent on the particular educational program.

Possible Educational Programs

Computer Technician Program (Technical Institute) - Students entering this program would generally be recent high school graduates. The computer technician training program should be divided into two areas of instruction. The first area should include the study of basic mathematics and electronic fundamentals. The second area should be the study of computer subject material.

A computer technician needs a basic knowledge of electronic fundamentals and electronic test equipment. However, he doesn't need an extensive electronic background such as that of a general electronic technician. Emphasis should be on solid state devices used in switching applications and oscilloscope training. Less emphasis can be placed on communications and radar.

If possible, the basic mathematics and electronic training should be completed prior to the study of the specific computer material. Only the computer training is described in this manual.

Computer Technician Program (Industrial or Military Training) - Students entering this program need a basic electronic background equivalent to that received in a technical institute. No previous knowledge of computers is required.

The objective of this program is to prepare an individual to enter a training program on a specific computer system as rapidly as possible. The student doesn't receive a broad background similar to that received in the technical institute program; and, in order to improve his background for advancement much future self-study will be required of the student.

Electrical Engineering (College or University) - Computer technology instructional blocks may be selected as part of an Electrical Engineering curriculum. During the course, emphasis should be placed on the supplementary questions found in the Computer Lab Workbook. The supplementary questions stress the design of logic functions.

Computer Programming Training - Some general computer technology concepts are helpful for the general computer programmer. The computer technology material should be presented after the programming students have a good general knowledge of a computer system.

Digital Computer Technology Instruction Blocks

Block	Subject Headings	References
1.	Number Systems Introduction to Computers Number Systems Number System Conversions Binary Math Octal Math	W.B.* PP VII-VIII, Chap. 1 T.G.* PP 1 to 21
2.	Introduction to Logic Introduction to the Computer Lab Logic Gates Computer Lab Functions Decimal to Binary Encoder Binary to Decimal Decoder	W.B.* PP VII-VIII. Chap. 1 T.G.* PP 1 to 21
3.	AND/NOR Gate NOR Application Comparator Application Exclusive OR Application Non-Inverting Gates	W.B. Chap. 2 T.G. PP 23 to 33
4.	Boolean Algebra Introduction Boolean Operators Boolean Expressions and Logic Diagrams Basic Laws Simplification Mapping Techiques Applications to Logic Design	W.B. Chap. 4 T.G. PP 53 to 89 W.B. Appendix B

* W.B. = Computer Lab Workbook
T.G. = Computer Lab Teacher's Guide

Block	Subject Headings	References
5.	<u>Flip-Flops</u>	
	Introduction to Flip-Flops	W.B.* Chap. 5
	R-S Flip-Flop	T.G.* PP 91 to 104
	Clocked R-S Flip-Flop	
	D Type Flip-Flop	
	J-K Flip-Flop	
6.	<u>Transfers</u>	
	Serial Transfer	W.B. PP 40, 41
	Ring Counters	T.G. PP 38 to 40
	Parallel Transfer	
7.	<u>Counter</u>	
	Asynchronous Binary Up-Counter	W.B. Chap. 5
	Asynchronous Binary Down-Counter	T.G. PP 91 to 104
	Synchronous Binary Up-Counter	
	Synchronous Binary Up-Down Counter	
	Synchronous Modulo N Counter	
	Asynchronous Self-Stopping Counter	
	Variable Modules Asyncronous Binary Up-Counter	
8.	<u>Serial Addition</u>	
	Review of Binary Math	W.B. Chap. 6
	Complementary Math	T.G. PP 105 to 116
	Serial Addition Block Diagram	
	Serial Adder	
	Subtraction Using the Serial Adder	
	Multiplication Using the Serial Adder	
9.	<u>Parallel Addition</u>	
	Parallel Adder Block Diagram	W.B. Chap. 7
	Two Step Parallel Addition	T.G. PP 105 to 116
	Single Step Parallel Addition	
	Subtraction Using the Parallel Adder	
10.	<u>Binary Coded Decimal Operations</u>	
	8421 Counter	W.B. Chap. 8
	Excess 3 Counter	T.G. PP 127 to 142
	2421 Counter	
	5421 Counter	
	Serial 3CD Addition	
11.	<u>Code Conversion and Decoding</u>	W.B. Chap. 9
	2421 to 8421 Converter	T.G. PP 143 to 151
	5421 to 8421 Converter	
	Gray to Binary Converter	
12.	<u>System Control</u>	
	Parallel Adder Control	W.B. Chap. 10
	Synchronizers	T.G. PP 153 to 158
	Frequency Consideration	

*W.B. = Computer Lab Workbook
T.G. = Computer Lab Teacher's Guide

Block	Subject Headings	References
13.	<u>Magnetic Core Memory</u>	
	Coincident current memory organization	Digital Computer Fundamentals,
	Core characteristics	Bartee, PP 226-227, 234-235
	Core configurations	
	Memory logic (Block Diagram Analysis)	
14.	<u>Peripheral Equipment</u>	
	Punched Card	Digital Computer Fundamentals,
	Paper Tape	Bartee, PP 257-289
	Magnetic Tape	
	Line Printers	
	Drums	
	Disks	
15.	<u>Introduction to Programming</u>	
	Computer Organization	W.G. PP 145-155
	Introduction to Programming	
	Flow Charting	
	Machine Languages	
	Assemblers	
	Machine Independent Languages	

Suggested Course Arrangements

Computer Technician Program (Technical Institute)

Block number (arranged in order of presentation)	Major Headings	Time (Hours)
1	Number Systems	32
2	Introduction to Logic	16
4	Boolean Algebra	48
3	AND/NOR Gates - Non-Invertive Gates	8
5	Flip-Flops	24
6	Transfers	16
7	Counters	32
10	Binary-Coded Decimal Operations	24
8	Serial Addition	16
9	Parallel Addition	16
11	Code Conversion and Decoding	8
12	System Control	16
13	Magnetic Core Memory	16
14	Peripheral Equipment	48
15	Introduction to Programming	48
	Total	368

Computer Technician Program (Military or Industrial)

Block #	Subject Material	Time (Hours)
1	Number Systems	16
2	Introduction to Logic	8
3	AND/NOR Gate - Inverting Gates	4
5	Flip-Flops	12
6	Transfers	8
7	Counters	16
8	Serial Addition	12
9	Parallel Addition	8
*1 10	Binary-Coded Decimal Operation	8
*1 11	Code Conversion and Decoding	4
12	System Control	8
13	Magnetic Core Memory	8
*2 14	Peripheral Equipment	16
19	Introduction to Programming	32
	Total	160

Electrical Engineering

Blocks	Major Headings	
1	Number Systems	
4	Boolean Algebra	
2	Introduction to Logic	3 Semester Hours
3	AND/NOR Gate - Non-Invertive Gates	
5	Flip-Flops	
6	Transfers	
7	Counters	
8	Serial Addition	3 Semester Hours
9	Parallel Addition	
10	Binary Coded Decimal Operations	
11	Code Conversion and Decoding	
12	System Control	3 Semester Hours
13	Magnetic Core Memory	
14	Peripheral Equipment	3 Semester Hours
15	Introduction to Programming	3 Semester Hours
	Total	15 Semester Hours

*1 Since this course is designed to rapidly prepare students for a specific computer system, this material may not be required.

*2 It may be advantageous to delay teaching peripheral equipment until a specific computer system is taught.

Digital Computer Programmer

Blocks	Major Headings	Hours
1	Number Systems	8
2	Introduction to Logic	14
5	Flip-Flops	4
6	Transfers	2
7	Counters	4
8	Serial Addition	2
9	Parallel Addition	2
14	Peripheral Equipment	4
	Total	40 Hours

Computer Technician Program (Technical Institute)

Computer Technology, Final Exam (Closed Book)

______ 1. In computer terminology, the word accumulator is most likely to be associated with

A. Memory
B. Arithmetic unit
C. Magnetic tape
D. Card Punch

______ 2. Converting $13_{(10)}$ to binary will produce

A. 1101
B. 1001
C. 1111
D. 1000

______ 3. Converting $211_{(8)}$ to decimal will produce

A. 137
B. 106
C. 224
D. 73

______ 4. The logical NAND function is described in which of the following truth tables (HI = true, LO = false):

A.

Inputs		
A	B	Output
LO	LO	LO
LO	HI	HI
HI	LO	HI
HI	HI	HI

B.

Inputs		
A	B	Output
LO	LO	LO
LO	HI	HI
HI	LO	HI
HI	HI	LO

C.

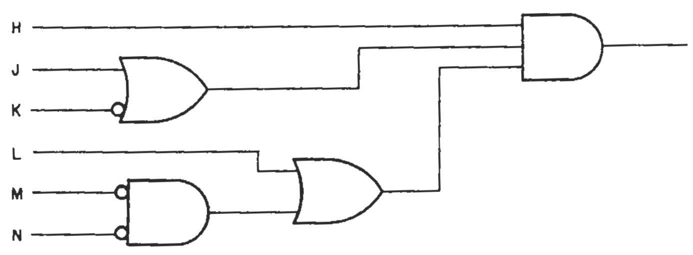
Inputs		
A	B	Output
LO	LO	HI
LO	HI	HI
HI	LO	HI
HI	HI	LO

D.

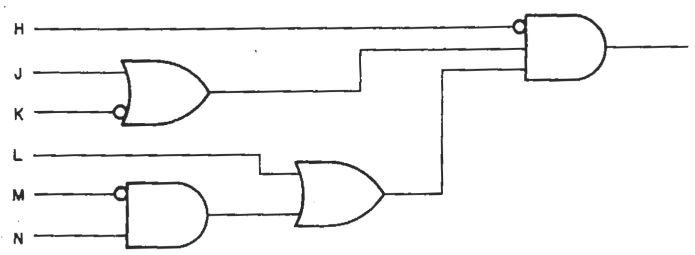
Inputs		
A	B	Output
LO	LO	HI
LO	HI	LO
HI	LO	LO
HI	HI	LO

_______ 5. H (J + $\overline{K}$) (L + $\overline{M}$N) is best represented by which of the following diagrams:

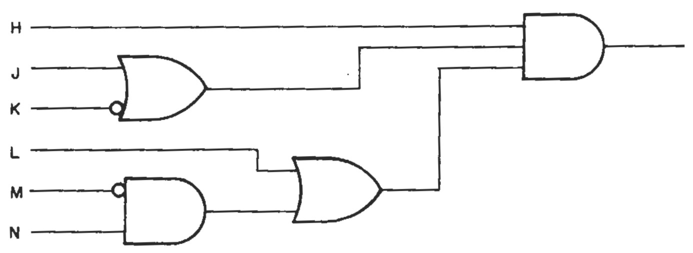
A.



B.



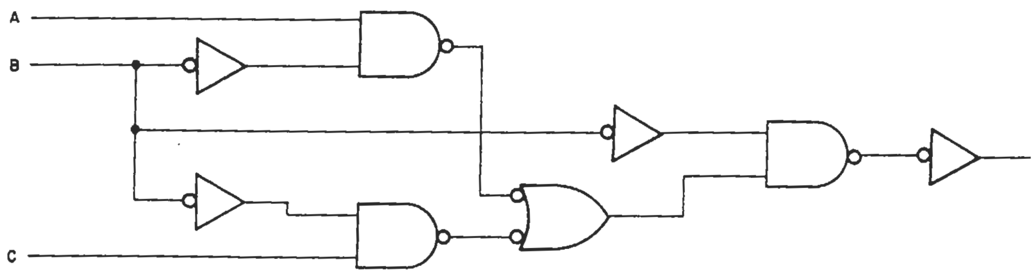
C.



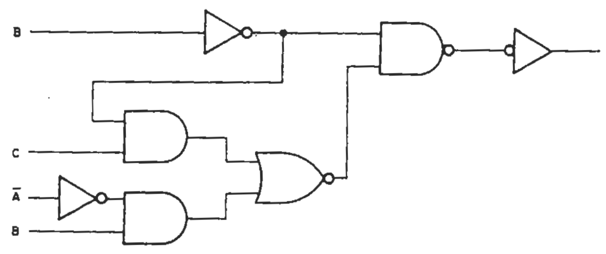
_______ 6. Simplifying the following Boolean expression AB + ABC + AB$\overline{C}$ + $\overline{A}$BC to its simplest form yields

A AB + AB$\overline{C}$
B B(A + C)
C ABC
D AB$\overline{C}$

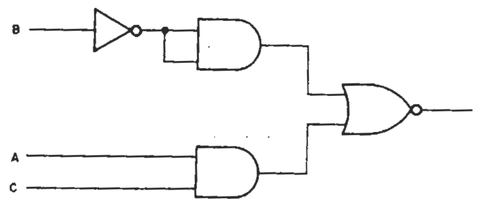
_______ 7. Using gates available on the computer lab, which of the following logic arrangements represents the correct simplification of this diagram.



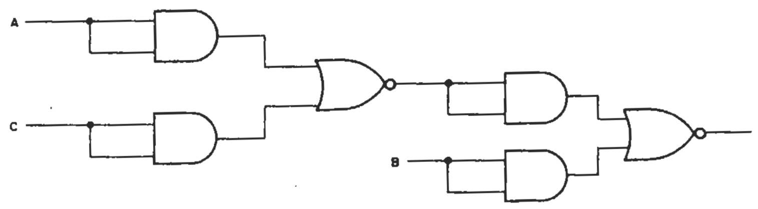
A.



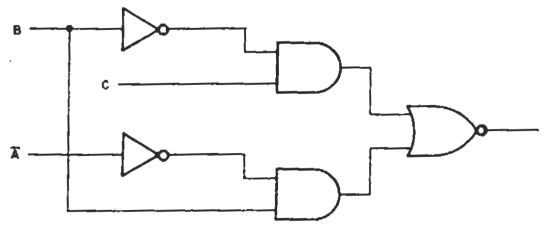
B.



C.

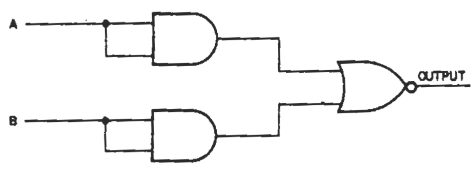


D.

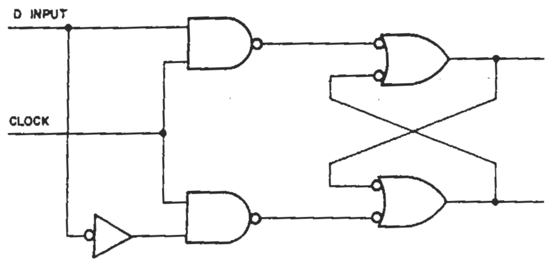


________8. The following logic diagram performs what logic function?

A. Exclusive OR function
B. Comparator function
C. OR function
D. NOR function



________9.



Which of the following best describes the limitations of the simplified version of the D type flip-flop illustrated above as compared to the actual version.

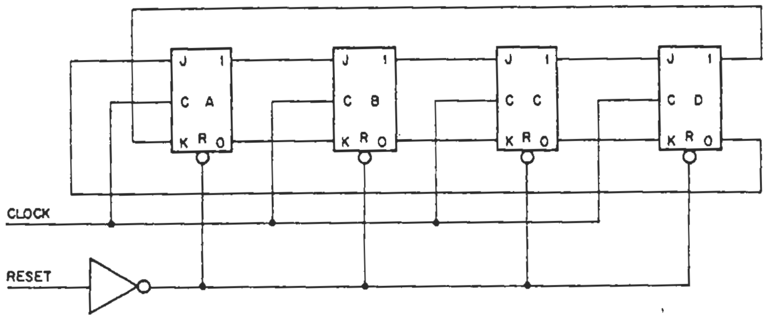
A. The level input in the simplified version has to be present 30 ns previous to clock time.

B. The trailing edge of the clock pulse changes the state of the flip-flop.

C. An indeterminate state arises if both clock and D inputs are HI simultaneously.

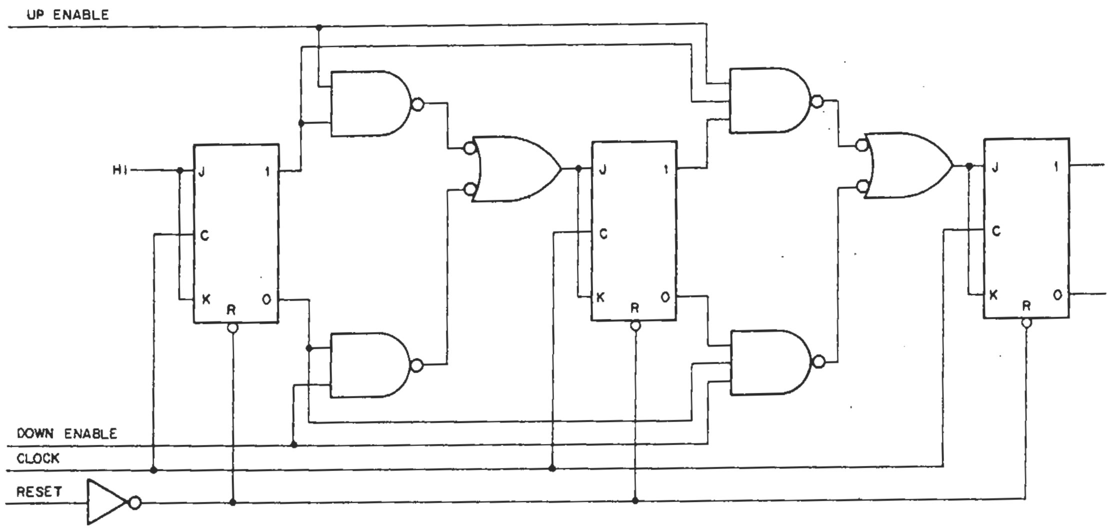
D. When the D input changes while the clock input is still HF the flip-flop "1" output will follow the level change.

________ 10. With the figure below initialized with all zeroes, how many counts are possible.



A. 4
B. 16
C. 8
D. 15

________ 11. What happens if both the up enable and down enable lines of the synchronous up/down counter of the figure below are simultaneously enabled with a HI level at clock pulse time.



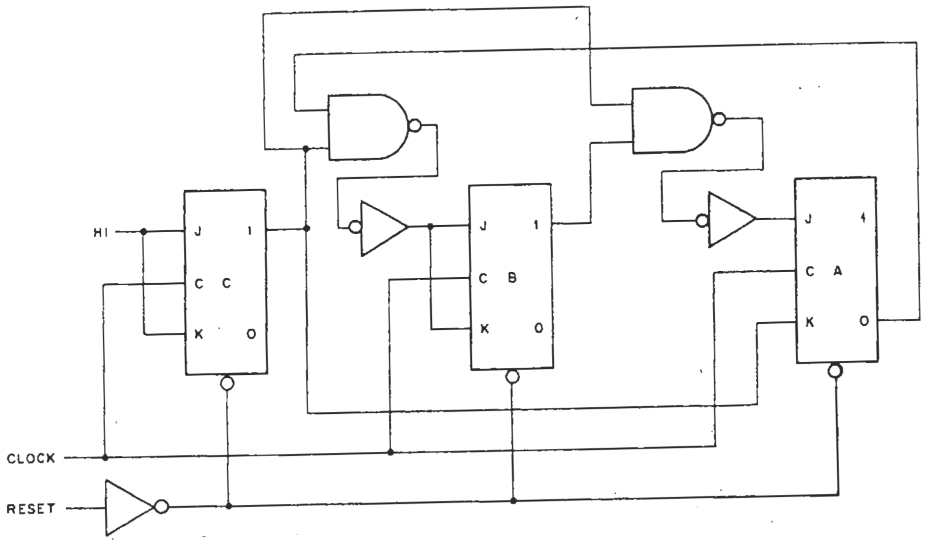
A. An indeterminate condition will result.

B. All flip-flops which have all less significant bits (flip-flops) the same will complement.

C. All flip-flops will switch and remain in the zero state.

D. All flip-flops will switch and remain in the one state.

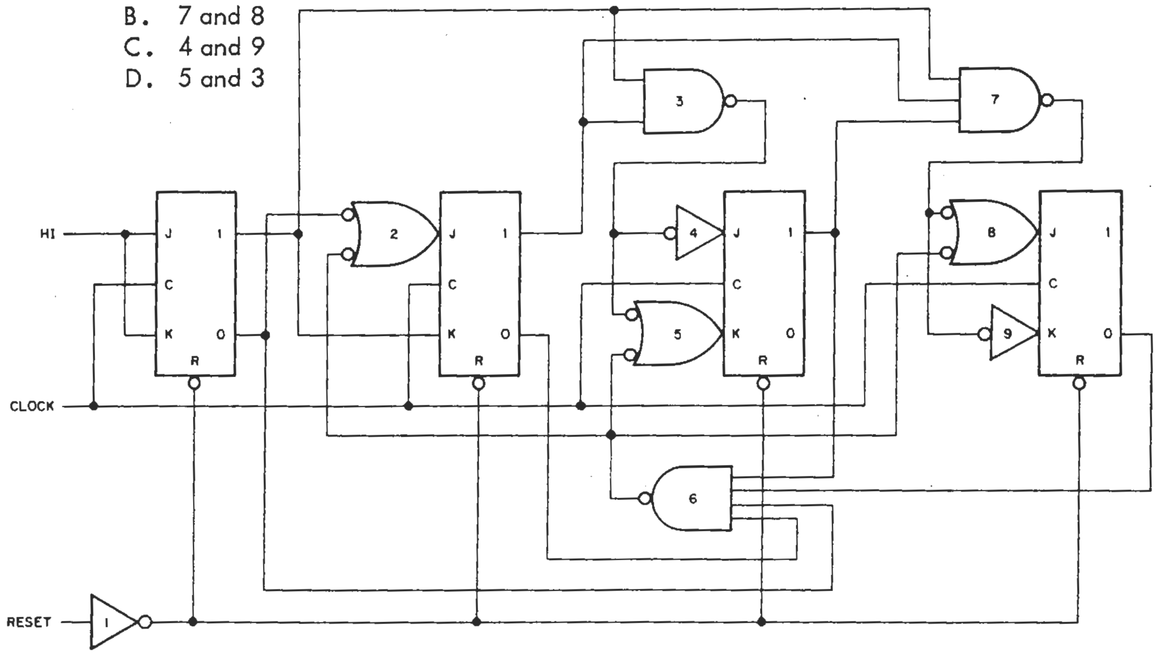
________12. What is the maximum binary count that the following counter reaches before recycling?



A. 0111
B. 0101
C. 0110
D. 0011

________13. In the figure below what two gates can be removed in order to simplify the design.

A. 1 and 2
B. 7 and 8
C. 4 and 9
D. 5 and 3



________ 14. In doing BCD addition and adding two decimal numbers in 8421 code, if the sum of the two digits were greater than 9, a correction factor of _____ would be added to the decade.

A +6
B -3
C -6
D +9

________ 15. When adding the binary numbers 101 and 100 in a 3-bit serial adder, how many clock pulses are required before the answer appears in the AC?

A. 6
B. 3
C. 2
D. 1

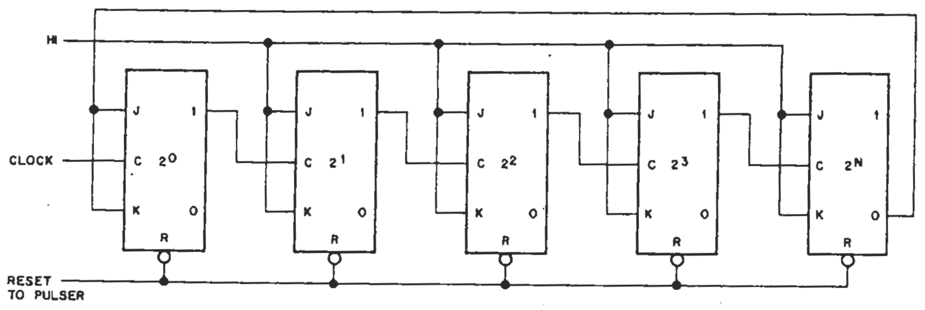
________ 16. In a full adder, a sum is generated when which <u>one</u> of the following conditions are true:

A. Carry in = 1; incident bit = 1; AC = 1
B. Carry in = 0; incident bit = 1; AC = 1
C. Carry in = 1; incident bit = 0; AC = 1
D. Carry in = 0; incident bit = 0; AC = 0

________ 17. The Gray Binary Code is a

A. parity error detection code
B. self complementing weighted code
C. octal divisor code
D. reflective code

________ 18. With the Reset Pulser depressed, the approximate maximum operating frequency range of the figure below is



A. 3.5 → 5.5 MHz
B. 9.5 → 10 MHz
C. 1.5 → 3.5 MHz
D. 2.3 → 3.5 MHz

———— 19. A 2-1/2D Memory

A. Has a larger core construction than 3D but faster access time.
B. Has fewer internal wires than a 3D Memory but slower access time.
C. Can only be accessed during inhibit time.
D. Is general a faster accessed memory than a 3D Memory.

———— 20. Batch processing refers to

A. Entering into the computor large amounts of preprocessed material.
B. Allowing a large number of terminals simultaneous access to the computer.
C. Allowing single terminal computer access.
D. Using two computer systems simultaneously.

———— 21. Which of the following peripherial equipment provides the fastest access time?

A. Magnetic tape
B. Magnetic drum
C. Punched cards
D. Paper tape

———— 22. Design a modulo 12 counter.

———— 23. Design a self-stopping synchronous 4-bit binary up-counter which will stop on a number selected by 4-bit rocker switch.

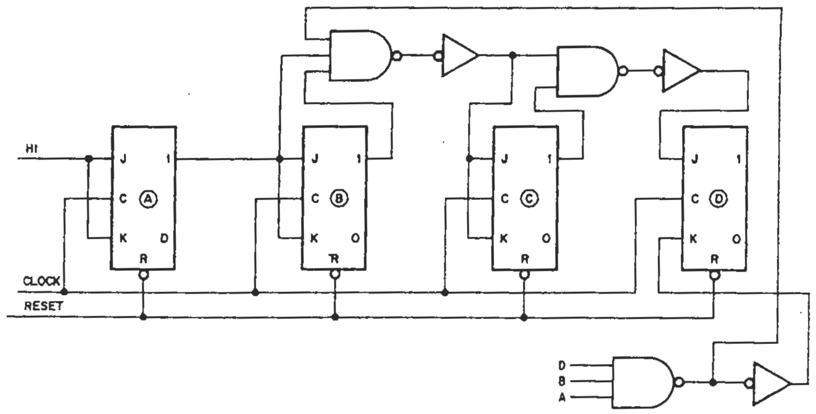
_______ 24. Design a 5421 synchronous up/down counter.

_______ 25. Write a program in the machine language you have been taught that will input data from the Teletype, store it sequentially in memory, and print it on the Teletype at the command comma EOT.

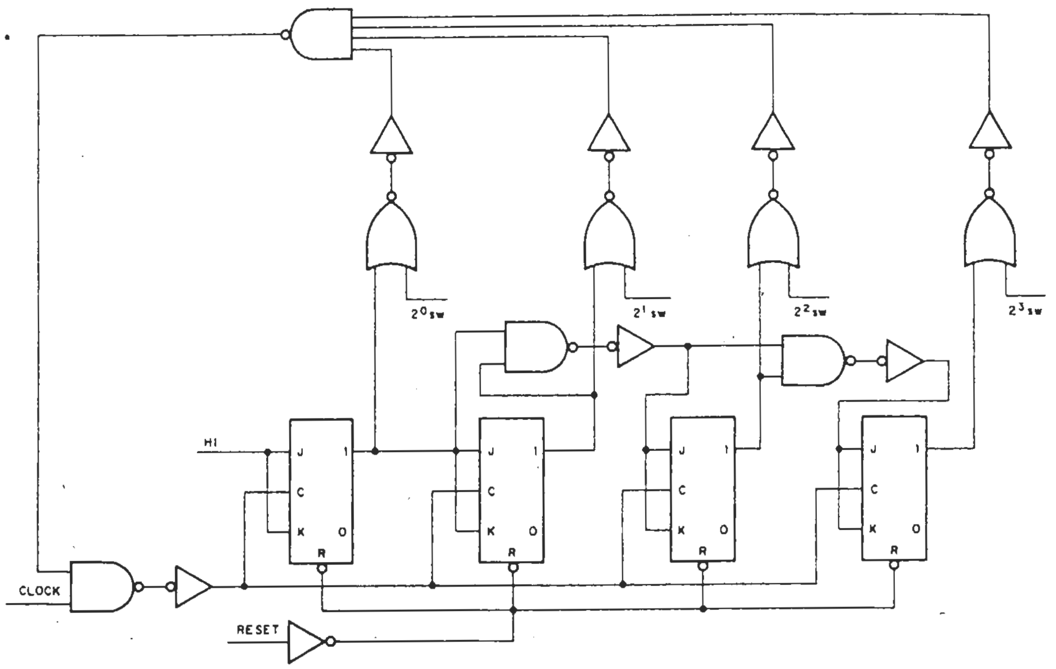
Computer Technician Program (Technical Institute)

Computer Technology, Final Exam
Answer Key

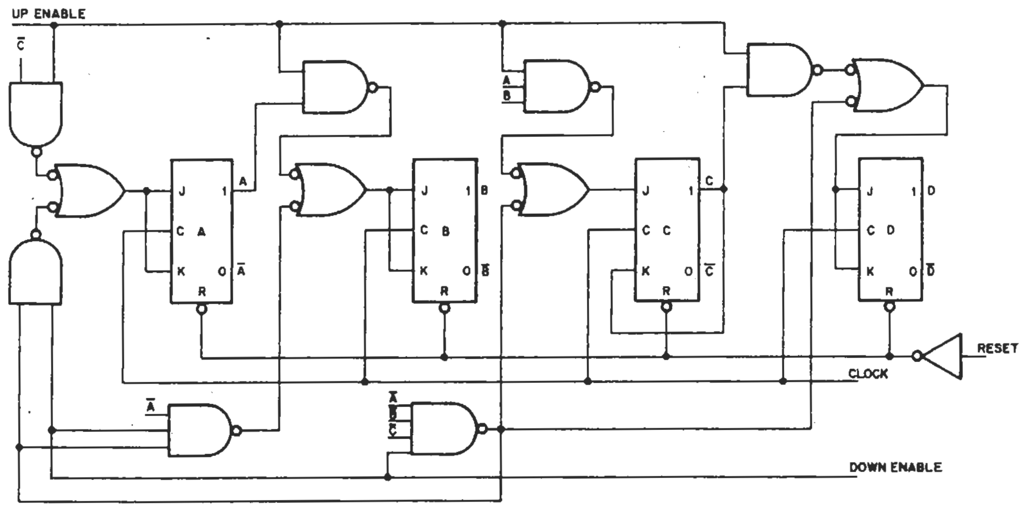
1. B
2. A
3. A
4. C
5. C
6. B
7. C
8. D
9. D
10. C
11. B
12. B
13. C
14. A
15. B
16. A
17. D
18. A
19. D
20. A
21. B
22.



23.



24.



25. As the program will vary depending on the language used, it is impractical to provide an answer to this problem.

Computer Lab Workbook - Textbook
Cross Reference Table

Chapter	Bartee, Thomas C. Digital Computer Fundamentals, McGraw-Hill Book Co., Inc., 1960, Pages	Flores, Ivan Computer Logic Prentice-Hall Inc., 1960, Pages	R.C. Baron A.T. Piccirilli Digital Logic And Computer Operations, McGraw-Hill Book Co., Inc., 1967, Pages	Chu, Yaohan Digital Computer Fundamentals McGraw-Hill Book Co., Inc., 1960 Pages
1	29 - 46 54 - 81	95 - 107	9 - 33	1 - 15
2	142 - 143, 156 280 - 282		219 - 224 225 - 230 222, 310 246,248,313	
3	81 - 95 149 - 154	146 162 - 165	69 - 102	185 - 196 368 - 378
4	113 - 145	123 - 144	33 - 68	89 - 122 182 - 183 122 - 132 378 - 383
5	95 - 99 139 - 141		78,105,309	1 - 15
6	160 - 162, 169 - 172, 176 - 177	150 - 158	119 - 132	363 - 371
7	158 - 160, 172 - 176	179 - 182	132 - 135	18 - 24 383 - 391 53 - 59
8	51 47 - 49, 154, 181-184, 186 - 187	110 - 122	103 - 118	53 - 57
9	285 - 286	193 - 196	106 - 109	78 - 87
10	290 - 333		209 - 219	209 - 213 392 - 393

WORLD-WIDE SALES AND SERVICE

MAIN OFFICE AND PLANT

DIGITAL EQUIPMENT CORPORATION
146 Main Street, Maynard, Massachusetts 01754
Telephone: From Metropolitan Boston: 646-8600
Elsewhere: (617)-897-5111
TWX: 710-347-0212 Cable: Digital Mayn. Telex: 94-8457

DOMESTIC

NORTHEAST

NORTHEAST OFFICE:
146 Main Street, Maynard, Massachusetts 01754
Telephone: (617)-646-8600 TWX: 710-347-0212

BOSTON OFFICE:
899 Main Street, Cambridge, Massachusetts 02139
Telephone: (617)-491-6130 TWX: 710-320-1167

ROCHESTER OFFICE:
480 Clinton Avenue So., Rochester, New York 14620
Telephone: (716)-454-2000 TWX: 510-253-3078

NEW HAVEN OFFICE:
129 College Street, New Haven, Connecticut 06510
Telephone: (203)-777-5797 TWX: 710-465-0692

MID-ATLANTIC

MID-ATLANTIC OFFICE:
Route 1, Princeton, New Jersey 08540
Telephone: (609) 452-9150

NEW YORK OFFICE:
Suite #1
71 Grand Avenue, Palisades Park, New Jersey 07650
Telephone: (201)-941-2016 or (212)-594-6955 TWX: 710-992-8974

PRINCETON OFFICE:
3 Ninianne Boulevard, Princeton, New Jersey 08540
Telephone: (609)-452-2940 TWX: 510-685-2337

LONG ISLAND
1919 Middle Country Road, Centereach, L.I., New York 11720
Telephone: (516)-585-5410 TWX: 510-228-6505

PHILADELPHIA OFFICE:
1100 West Valley Road, Wayne, Pennsylvania 19087
Telephone (215)-687-1405 TWX: 510-668-4461

WASHINGTON OFFICE:
Executive Building
7100 Baltimore Ave., College Park, Maryland 20740
Telephone: (301)-779-1100 TWX: 710-826-9662

ATLANTA OFFICE:
Suite 116, 1700 Commerce Drive N.W., Atlanta, Georgia 30318
Telephone: (404) 525-4353 TWX: 810-751-3251

SOUTHEAST

HUNTSVILLE OFFICE:
Suite 41 — Holiday Office Center
3322 Memorial Parkway S.W., Huntsville, Ala 35801
Telephone: (205)-881-7730 TWX: 810-726-2122

COCOA OFFICE:
412 High Point Drive, Cocoa, Florida 32922
Telephone: (305)-632-9283 TWX: 510-957-1448

CENTRAL

PITTSBURGH OFFICE:
400 Penn Center Boulevard, Pittsburgh, Pennsylvania 15235
Telephone: (412)-243-8500 TWX: 710-797-3657

CHICAGO OFFICE:
69 North Broadway, Des Plaines, Illinois 60016
Telephone: 312-299-0144 TWX: 910-233-0894

ANN ARBOR OFFICE:
3853 Research Park Drive, Ann Arbor, Michigan 48104
Telephone: (313)-761-1150 TWX: 610-223-6053

MINNEAPOLIS OFFICE:
Digital Equipment Corporation
15016 Minnetonka Industrial Road
Minnetonka, Minnesota 55343
Telephone: 612-935-1744 TWX: 910-576-2818

CLEVELAND OFFICE:
Park Hill Bldg., 35104 Euclid Ave., Willoughby, Ohio 44094
Telephone: (216)-946-8484

DAYTON OFFICE:
3110 South Kettering Blvd., Dayton, Ohio 45439
Telephone: (513)-299-7377 TWX: 810-459-1676

HOUSTON OFFICE:
3417 Milam Street, Suite A, Houston, Texas 77002
Telephone: (713)-523-2529 TWX: 910-881-1651

DALLAS OFFICE:
1625 W. Mockingbird Lane, Suite 309, Dallas, Texas 75207
Telephone: 214-638-4880

WEST

LOS ANGELES OFFICE:
801 E. Ball Road, Anaheim, California 92805
Telephone: (714)-776-6932 or (213)-625-7669 TWX: 910-591-1189

SAN FRANCISCO OFFICE:
560 San Antonio Road, Palo Alto, California 94306
Telephone: (415)-326-5640 TWX: 910-373-1266

SEATTLE OFFICE:
McAusland Building, 10210 N.E. 8th Street
Bellevue, Washington 98004
Telephone: (206)-454-4058 TWX: 910-443-2306

ALBUQUERQUE OFFICE:
6303 Indian School Road, N.E., Alburquerque, N.M. 87110
Telephone: (505)-296-5411 TWX: 910-989-0614

DENVER OFFICE:
2305 South Colorado Blvd., Suite #5, Denver, Colorado 80222
Telephone: 303-757-3332 TWX: 910-931-2650

SALT LAKE CITY OFFICE:
431 South 3rd East, Salt Lake City, Utah 84111
Telephone: 801-328-9838

INTERNATIONAL

CANADA

Digital Equipment of Canada, Ltd.
150 Rosamond Street, Carleton Place, Ontario, Canada
Telephone: (613)-257-2615 TWX: 610-561-1651

Digital Equipment of Canada, Ltd.
230 Lakeshore Road East, Port Credit, Ontario, Canada
Telephone: (416)-279-6111 TWX: 610-492-4306

Digital Equipment of Canada, Ltd.
640 Cathcart Street, Suite 205, Montreal, Quebec, Canada
Telephones: (514)-861-6394 TWX: 610-421-3960

Digital Equipment of Canada, Ltd.
5531-103 Street
Edmonton, Alberta, Canada
Telephone: (403)-434-9333 TWX: 610-831-2248

GERMANY

Digital Equipment GmbH
5 Koeln, Neue Weyerstr. 10, West Germany
Telephone: 23 55 01 Telex: 841-888-2269
Telegram: Flip Chip Koeln

Digital Equipment GmbH
8 Muenchen, 2 Theresienstr. 29, West Germany
Telephone: 28 30 53 Telex: 841-24226

ENGLAND

Digital Equipment Co. Ltd.
Arkwright Road, Reading, Berkshire, England
Telephone: Reading 85131 Telex: 851-84327

Digital Equipment Co. Ltd.
13/15 Upper Precinct
Bolton Road,Walkden,Worsley, Manchester, M285AZ England
Telephone: (061) 790 4591

FRANCE

Equipement Digital
233 Rue De Charenton, Paris 12ᵉ, France
Telephone: 344-76-07 Telex: 21339

BENELUX

Digital Equipment N.V.
Koninginnegracht 65, The Hague, Netherlands
Telephone: 635960 Telex: 32533

SWEDEN

Digital Equipment Aktiebolag
Vretenvagen 2, Solna 1, Stockholm, Sweden
Telephone: 98 13 90
Telex: Digital Stockholm 17050
Cable: Digital Stockholm

AUSTRALIA

Digital Equipment Australia Pty. Ltd.
75 Alexander Street, Crows Nest, N.S.W. 2065. Australia.
Telephone: 439-2566 Telex: AA20740
Cable: Digital, Sydney

Digital Equipment Australia Pty. Ltd.
60 Park Street, South Melbourne, Victoria, 3205
Telephone: 26 6542 Telex: AA30700

Digital Equipment Australia Pty. Ltd.
P.O. Box 18, St. Lucia. 4067, Australia.
Telephone: 7-5405 Telex: AA40616

JAPAN

Rikei Trading Co., Ltd. (sales only)
Kozato-Kaikan Bldg.
No. 18-14, Nishishimbashi 1-chome
Minto-ku, Tokyo, Japan
Telephone: 5915246 Telex: 7814208

Digital Equipment Corporation (service only)
Fukuyoshicho Building, No. 2-6, Roppongi 2-Chome,
Minato-Ky, Tokyo
Telephone No. 582-4952 Telex No. 0242-2650