CHAPTER 1

DIGITAL ELECTRONICS

Syllabus: Digital electronics: logic functions, minimization, design and synthesis of combinational and sequential circuits, number representation and computer arithmetic (fixed and floating point).

1.1 INTRODUCTION

Digital electronics represents discrete signals instead of signals in a continuous range. It uses two binary levels 0's (corresponding to false) and 1's (corresponding to true). The main reason for advancement in digital electronics is integrated circuits (ICs).

1.2 NUMBER SYSTEM

Number system is an age old method to represent numerals (Fig. 1.1). Decimal number system is the most common number system and binary number system is used by computers.

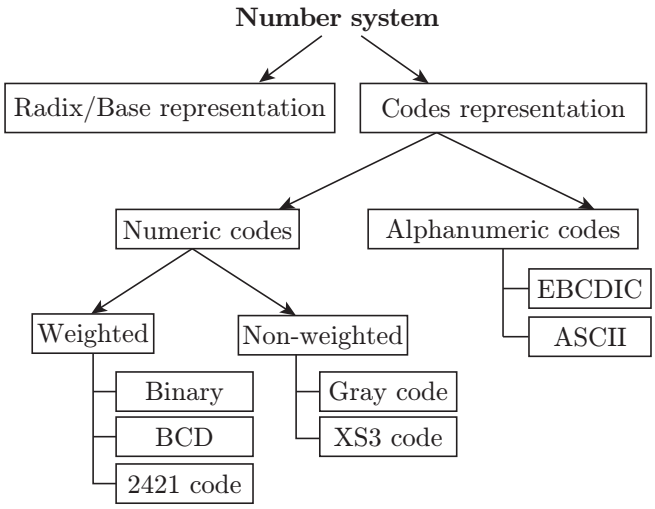


Figure 1.1 | Number system.

Table 1.1 Types of number systems

Base/Radix	Unique Numbers	Terminology	Example
10	0 to 9	Decimal number system	10.47
8	0 to 7	Octal number system	65.32
2	0 to 1	Binary number system	110.11
16	0 to 15	Hexadecimal number system	BAD.1A
	0 to 9 are numerals and 10 to 15 are represented with alphabets 10 = *A*, 11 = *B*, 12 = *C*, … 15 = *F*		

Radix or base is the number of unique digits, so the different base or radix can be expressed as represented in Table 1.1. For example, $(57)_{10}$ – here 10 is the base (or radix) and 57 is the number.

Example 1.1

Let us understand the logic of representation by considering $(57.1)_{10}$. We know 57 means 50 + 7 + (1/10). Now, representing the same by making use of radix (which is 10), we get

$$5\times(10)^1+7\times(10)^0+1\times(10)^{-1}$$

1.2.1 Conversions of Number System

A number of a particular number system can be converted into another number system, as follows:

1. **Decimal number system to any number system:** Consider the example of $(57.3)_{10}$. To find the binary equivalent, we have

2|57			
2|28	1		
2|14	0	$2\times0.3=0.6$	0
2|7	0	$2\times0.6=1.2$	1
2|3	1	$2\times0.2=0.4$	0
2|1	1	$2\times0.4=0.8$	0
$(111001)_2$		$(0.0100)_2$	

   Therefore, $(57.3)_{10}=(111001.0100)_2$

2. **Any number system to decimal number system:** Consider the example of $(111001.0100)_2$. To find the decimal equivalent we have

$$\begin{aligned}(111001.0100)_2 &= 1\times2^5+1\times2^4+1\times2^3+0\times2^2\\ &\quad+0\times2^1+1\times2^0+0\times2^{-1}\\ &\quad+1\times2^{-2}+0\times2^{-3}+0\times2^{-4}\\ &=32+16+8+0+0+1+0\\ &\quad+0.25+0+0=(57.25)_{10}\end{aligned}$$

   Another example is of $(101.11)_2=(5.75)_{10}=(5.6)_8=(5.C)_{16}$.

1.2.2 Complement of a Number

Any given number will have two complements:

1. **Radix minus one complement (R-1's complement):** It is obtained by subtracting the given number from the highest possible number. For example, if there is a two-digit decimal number, it will be subtracted from 99; whereas a 3-bit number in a binary system will be subtracted from 111.

   Note: The 1's complement of a binary number is obtained by interchanging the 1's and 0's. For example, 1's complement of $(1011)_2$ is $(0100)_2$.

2. **Radix complement (R's complement):** This is also known as true complement. It is obtained by adding one to R-1's complement. For example, $(57)_{10}$ – its 9's complement is 42 and its 10's complement is 43.

1.2.2.1 Subtraction Using R-1's Complement

For the following subtractions,

$$\begin{array}{r} A \\ -B \\ \hline \\ \hline \end{array}$$

the steps involved are as follows:

Step 1. Find R-1's complement of *B*.

Step 2. Add *A* and *B*.

Step 3. If there is carry, then add this carry to the answer.

If there is no carry, then find the R-1's complement of the answer and place a negative sign in front of the answer.

Problem 1.1: $(101)_2 - (011)_2$

Solution:

Step 1. 1's Complement of 011 is 100.

Step 2. The addition of these two numbers is (101 + 100 = 1 001).

Step 3. There is a carry of 1, so now we will add this carry to the answer (1 + 001), which is $(010)_2$.

Problem 1.2: $(011)_2 - (101)_2$

Solution:

Step 1. 1's Complement of 101 is 010.

Step 2. The addition of these two numbers is (011 + 010 = 0 101).

Step 3. There is a no carry, so now we will find 1's complement of the answer obtained (which is 010) and place a negative sign in front of it. So, the answer is $(-010)_2$.

1.2.2.2 Subtraction Using R's Complement

In this method, we find R's complement of the subtrahend and we do not add the carry, as in step 2. Rest of the steps are the same.

Problem 1.3: $(101)_2 - (011)_2$

Solution:

Step 1. 2's complement of 011 is 101.

Step 2. The addition of these two numbers is (101 + 101 = 1 010).

Step 3. There is a carry of 1, so now we will not add this carry to the answer (001), which is $(010)_2$.

Problem 1.4: $(011)_2 - (101)_2$

Solution:

Step 1. 2's complement of 101 is 011.

Step 2. The addition of these two numbers is (011 + 011 = 0 110).

Step 3. There is a no carry, so now we will find 2's complement of the answer obtained (which is 010) and place a negative sign in front of it. So, the answer is $(-010)_2$.

1.2.3 Representation of Negative Numbers

In$-(57)_{10}$ = (1111001), the most significant bit is 1, showing that the number is negative binary number.

1.2.4 The IEEE Standard for Floating Point Numbers

The IEEE (Institute of Electrical and Electronics Engineers) has provided a standard for floating point numbers and their arithmetic. This standard specifies how single precision (32-bit) and double precision (64-bit) floating point numbers are to be represented.

1. **Single Precision:** The IEEE single precision floating point standard representation requires a 32-bit word. The first bit to the left is the sign bit, the next eight bits are the exponent bits, and the final 23 bits form the fraction part (Fig. 1.2).

   The value of a normalized number $= (-1)^s \times 1.m \times 2^{e^{-127}}$

   where $s = 1$ for negative number, $s = 0$ for positive number, m = mantissa and e = exponent.
2. **Double Precision:** The IEEE double precision floating point standard representation requires a 64-bit word. The first bit is the sign bit, the next eleven bits are the exponent bits, and the final 52 bits form the fraction part (Fig. 1.3).

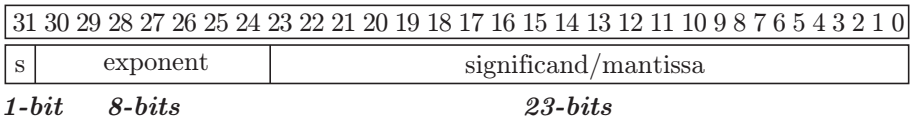


Figure 1.2 Single Precision (32 bits).

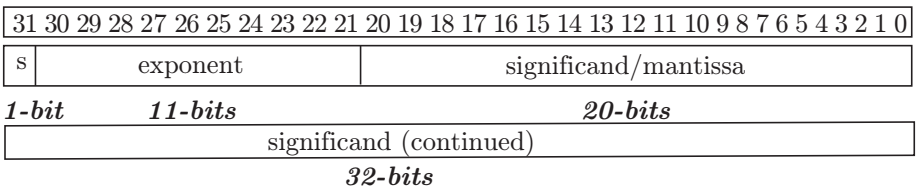


Figure 1.3 Double Precision (64 bits).

Problem 1.5: Consider the decimal number: +116.25. Represent it using single precision IEEE floating point standard.

Solution: The equivalent binary representation is +1110100.01

$$\text{Normalizing number} = +1.11010001 \times 2^{6}$$
$$= +1.11010001 \times 2^{133-127}$$
$$= +1.11010001 \times 2^{10000101-01111111}$$

In IEEE 754 format:

Sign	Exponent	Fraction
0	10000101	11010001000000000000000

1.3 BOOLEAN LOGIC

The concept of Boolean logic was proposed by George Boole, in which any information or data is represented in the form of two states, which are complement to each other (such as 0 and 1, hot and cold, black and white, north and south, etc.).

Logic gates form the basic building block of the digital system. A logic gate is an electronic device that takes logical decisions based on given input combinations and its output can be represented by Boolean expression. The symbols used to represent logic gates are shown in Fig. 1.4.

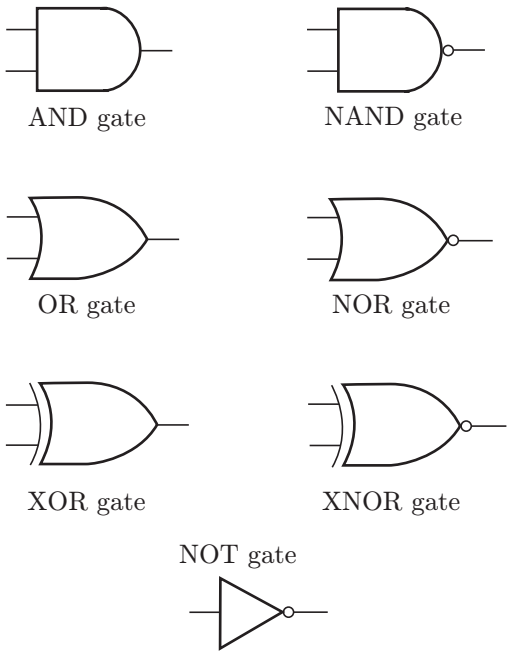


Figure 1.4 | Logic gates symbols.

Table 1.2 represents 15 basic operations that can be performed.

Table 1.2 | Basic operations in Boolean algebra

Inputs		**Operations**					
X	***Y***	**Null**	**Identity**	**Inhibit**			**Transfer**
				X* Inhibit *Y	***Y* Inhibit *X***		
0	0	0	1	0	0		0
0	1	0	1	0	1		1
1	0	0	1	1	0		0
1	1	0	1	0	0		1
Inputs		**Operations**					
X	***Y***	**AND**	**OR**	**NOT**		**XOR**	**XNOR**
				NOT *X*	**NOT *Y***		
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	0	1	1	0
1	1	1	1	0	0	0	1

(*Continued*)

Table 1.2 | Continued

Inputs		Operations			
X	Y	NAND	NOR	Implication	
				X Implication Y	Y Implication X
0	0	1	1	1	1
0	1	1	0	1	0
1	0	1	0	0	1
1	1	0	0	1	1

Table 1.2 is interpreted as follows:

1. Null − All the outputs are zero.
2. Identity − All the outputs are one.
3. X inhibit Y − Represented as $(x \cdot \sim y)$
4. Y inhibit X − Represented as $(y \cdot \sim x)$
5. Transfer − Same as y
6. AND − $x \cdot y$ (Similar to intersection in set theory)
7. OR − $x + y$ (Similar to union in set theory)
8. NOT X − $\sim x$ (Also known as complement of x)
9. NOT Y − $\sim y$ (Also known as complement of y)
10. XOR − $(\sim x \cdot y) + (x \cdot \sim y)$ (Also known as parity checker)
11. XNOR − $(x \cdot y) + (\sim x \cdot \sim y)$
12. NAND − $\sim(x \cdot y)$
13. NOR − $\sim(x + y)$
14. X implication Y $(\sim x + y)$
15. Y implication X $(\sim y + x)$

Problem 1.6: It is defined that $x * y = x'y' + xy$ and $z = x * y$. Among the following P, Q and R, which statement is correct?

P: $x = y * z$

Q: $y = x * z$

R: $x * y * z = 1$

Solution:

x	Y	z	P	Q	R
0	0	1	0	0	1
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1

1.3.1 Boolean Algebra Laws (Huntington's Postulates)

The laws of Boolean Algebra are given by Huntington's postulates:

1. $X + 0 = X$
2. $X + (\sim X) = 1$
3. $X + X = X$
4. $X + 1 = 1$
5. $X + Y = Y + X$
6. $X + (Y + Z) = (X + Y) + Z$
7. $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$
8. $\sim(X + Y) = \sim X \cdot \sim Y$ (Also called DeMorgan's law)
9. $X + (X \cdot Y) = X$
10. $\sim(\sim X) = X$

1.3.2 Duality Theorem

Every expression remains valid if we interchange the operator and identity elements. Following laws are the dual of the respective laws written above.

1. $X \cdot 1 = X$
2. $(X \cdot \sim X) = X \cdot X' = 0$
3. $X \cdot X = X$
4. $X \cdot 0 = 0$
5. $X \cdot Y = Y \cdot X$
6. $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
7. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
8. $(X \cdot Y)' = X' + Y'$
9. $X \cdot (X + Y) = X$

1.3.3 Consensus Theorem

It is applicable only if a Boolean function has

1. three variables
2. each variable is used two times
3. only one variable in complemented or non-complemented form

Then, the term related to the complemented and non-complemented variable is the answer.

Example 1.2

$$AB + A'C + BC = AB + A'C$$

1.3.4 Positive Logic and Negative Logic

Usually, the two (binary) levels are represented in two different ways:

1 (signifying high) and 0 (signifying low) – Known as positive logic

or

1 (signifying low) and 0 (signifying high) – Known as negative logic

1.3.4.1 Canonical and Standard Forms

The minterms and the maxterms are given in Table 1.3.

Product-of-sums (POS) form:

$$\begin{aligned} F &= (x'y'z') + (xy'z') + (xyz) \\ &= m_0 + m_4 + m_7 \\ &= \sum m(0,4,7) \end{aligned}$$

Sum-of-products (SOP) form:

$$\begin{aligned} F &= (X+Y+Z)\cdot(X'+Y+Z)\cdot(X+Y+Z) \\ &= M_0 \cdot M_4 \cdot M_7 \\ &= \prod M(0,4,7) \end{aligned}$$

In order to convert minterm to maxterm, put all those numbers in maxterm which are not present in minterm, and vice versa.

$$\sum m(0,\,1,\,5) = \prod M(2,\,3,\,4,\,6,\,7)$$

1.3.4.2 Minimization of Expression

A Boolean expression which cannot be further reduced is said to be in minimal form. Each term in minimal Boolean expression is called prime implicant. The following are the two methods for minimization:

1. **Boolean Theorems (postulates):** Using the Boolean postulates, we can simplify an expression using algebraic method. This method is relatively difficult as we need to remember all the postulates.

Problem 1.7: Simplify the Boolean function $X = (A+B)(A+C)$.

Solution:

$$\begin{aligned} X &= AA + AC + AB + BC \\ &= A + AC + AB + BC \\ &= A(1+C) + AB + BC \\ &= A(1) + AB + BC \\ &= A + AB + BC \\ &= A(1+B) + BC \\ &= A(1) + BC \\ &= A + BC \end{aligned}$$

Problem 1.8: Simplify the Boolean function $X = A + A'B$.

Solution:

$$\begin{aligned} X &= A(1+B) + A'B \\ &= A\cdot B + A + A'B = A\cdot B + A'B + A \\ &= (A+A')\cdot B + A = 1\cdot B + A = B + A \end{aligned}$$

Table 1.3 | Minterms and Maxterms

		Minterms (SOP)		Maxterms (POS)	
Decimal	**Binary**	**Term**	**Designation**	**Term**	**Designation**
0	000	$x'y'z'$	m_0	$X + Y + Z$	M_0
1	001	$x'y'z$	m_1	$X + Y + Z'$	M_1
2	010	$x'yz'$	m_2	$X + Y' + Z$	M_2
3	011	$x'yz$	m_3	$X + Y' + Z'$	M_3
4	100	$xy'z'$	m_4	$X' + Y + Z$	M_4
5	101	$xy'z$	m_5	$X' + Y + Z'$	M_5
6	110	xyz'	m_6	$X' + Y' + Z$	M_6
7	111	xyz	m_7	$X' + Y' + Z'$	M_7

2. **Karnaugh-Map Method (K-Map method):** Karnaugh-map (K-map) involves graphical representation for simplifying Boolean expressions. It consists of group of adjacent cells in which each cell corresponds to one of the combinations of maxterms or minterms of n variables and is represented by group of literals. For n-variable K-map, there are 2^n cells required (Figs. 1.5−1.7). From one cell to the neighbouring cell there must be only one literal change. While solving the K-map from higher-order group to lower-order group only, the simplification must be done. In K-map, leftmost column and rightmost column of the same row are considered to be adjacent.
   If $K = 1$, it represents that all the cells are 1 so there is no need of circuit.

1.3.4.3 Grouping of Cells for Simplification

The grouping of cells for simplification is done in the following ways:

1. **Pairs:** If two adjacent cells either in row or in column contain 1's, they are said to be in pairs. Grouping of two adjacent cells removes one variable and output will contain common variables in two terms.
2. **Quads:** We can group four adjacent cells in K-map forming a quad. Grouping a quad eliminates the two variables that appear in both normal and complemented form. Resultant terms contain $(n - 2)$ variables, where n is the number of variables.
3. **Octet:** A group of eight adjacent cells is called octet. Grouping of 1's eliminates three variables. Resultant terms contain $(n - 3)$ variables, where n is the number of variables.

1.3.4.4 SOP Simplification Using K-Map

A higher priority will be given to more possible covering, that is, octet then to quad and then to pairs. The following procedure is followed:

1. Check the SOP expression and convert to standard and canonical form if required.
2. Draw the K-map based on the number of variables.
3. On the basis of SOP canonical form, 1's are entered to corresponding cells. Group the adjacent cells and take the common variable from each grouping.
4. Combine each group variable to obtain simplified expression.

1.3.4.5 POS Simplification Using K-Map

1. Check the POS expression and convert it to standard and canonical form if required.
2. Draw the K-map based on the number of variables.
3. Enter 0's in corresponding cells and group the adjacent cells.
4. Take the common variable from each grouping and write terms in OR form. The uncomplemented variable is assigned 0 and complemented is assigned 1.
5. Combine each OR term from the different groups to obtain simplified expression.

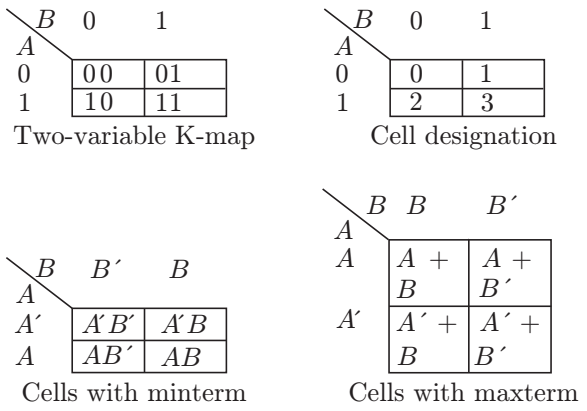


Figure 1.5 | Two-variable K-map.

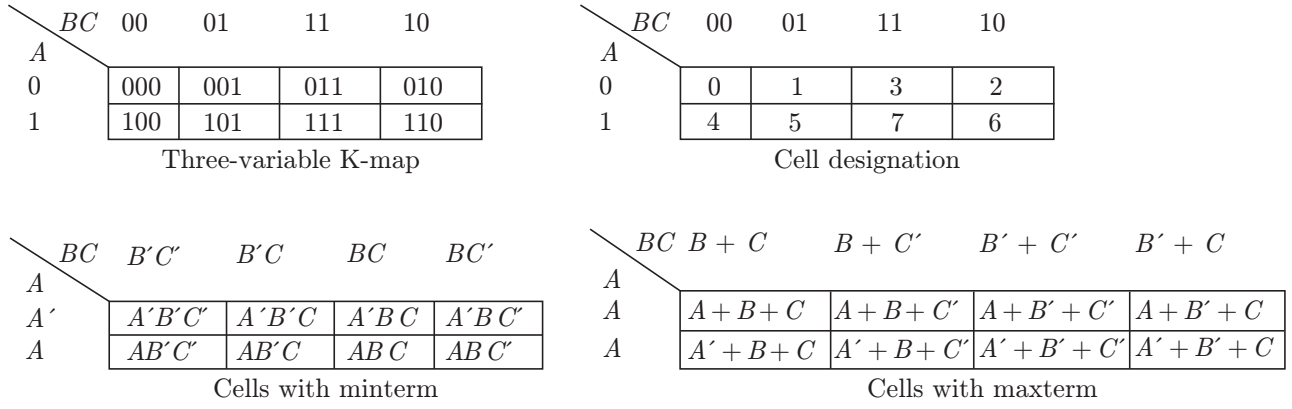


Figure 1.6 | Three-variable K-map.

AB \ CD	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

Four-variable K-map

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Cell designation

AB \ CD	$C'D'$	$C'D$	CD	CD'
$A'B'$	$A'B'C'D'$	$A'B'C'D$	$A'B'CD$	$A'B'CD'$
$A'B$	$A'BC'D'$	$A'BC'D$	$A'BCD$	$A'BCD'$
AB	$ABC'D'$	$ABC'D$	$ABCD$	$ABCD'$
AB'	$AB'C'D'$	$AB'C'D$	$AB'CD$	$AB'CD'$

Cells with minterm

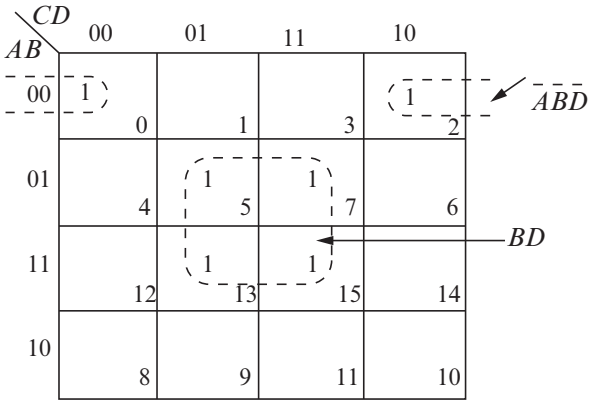
AB \ CD	$C+D$	$C+D'$	$C'+D'$	$C'+D$
$A+B$	$A+B+C+D$	$A+B+C+D'$	$A+B+C'+D'$	$A+B+C'+D$
$A+B'$	$A+B'+C+D$	$A+B'+C+D'$	$A+B'+C'+D'$	$A+B'+C'+D$
$A'+B'$	$A'+B'+C+D$	$A'+B'+C+D'$	$A'+B'+C'+D'$	$A'+B'+C'+D$
$A'+B$	$A'+B+C+D$	$A'+B+C+D$	$A'+B+C'+D'$	$A'+B+C'+D$

Cells with maxterm

Figure 1.7 Four-variable K-map.

Problem 1.9: Simplify $F(A,B,C,D) = \sum m(0, 2, 5, 7, 13, 15)$.

Solution:



The simplified Boolean expression is $F = A'B'D' + BD$.

Problem 1.10: Simplify $F(A,B,C,D) = ACD + A'B + D'$

Solution: Converting to standard SOP form. The first term ACD has one missing variable B.

$$ACD = ACD(B + B') = ABCD + AB'CD$$

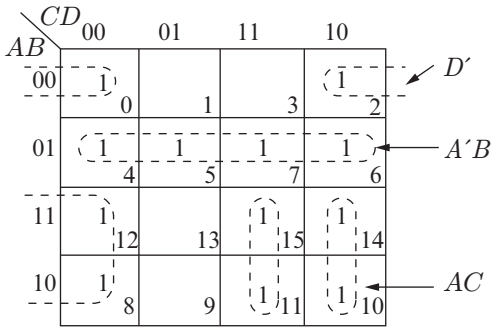
Similarly,

$$A'B = A'B(C + C')(D + D')$$
$$= A'BCD + A'BCD' + A'BC'D + A'BC'D'$$

$$D' = D'(B + B')(C + C') + (A + A')$$

Combining all terms, we get

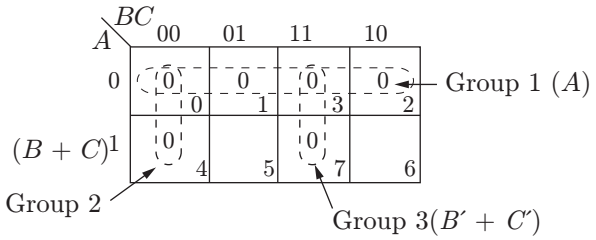
$$F = ABCD + AB'CD + A'BCD + A'BCD'$$
$$+ A'BC'D + A'BC'D' + ABC'D'$$
$$+ AB'CD' + AB'C'D' + A'B'CD' + A'B'C'D'$$



The simplified Boolean expression is $F = D' + A'B + AC$.

Problem 1.11: Simplify $S = \Pi M(0,1,2,3,4,7)$

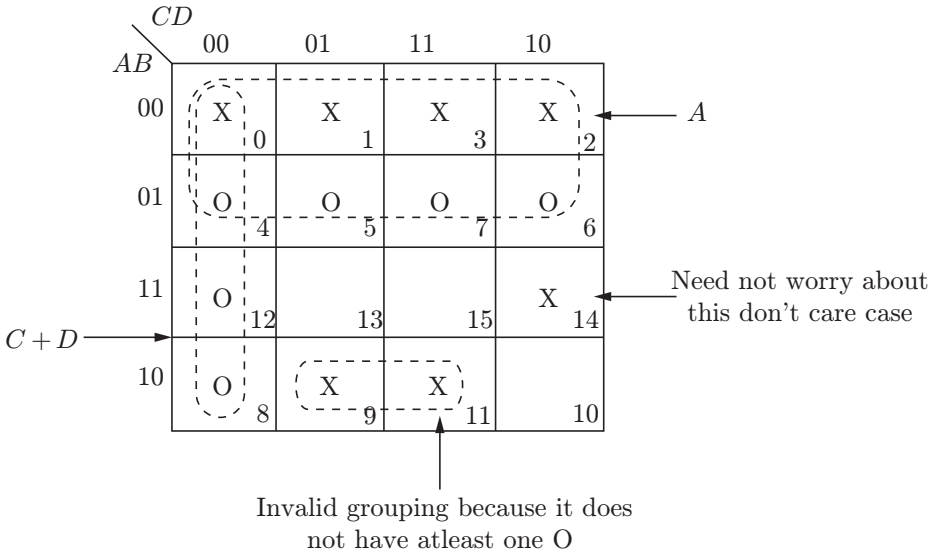
Solution:



The simplified Boolean expression is $S = A \cdot (B + C) \cdot (B' + C')$

Problem 1.12: Simplify $F(A, B, C, D) = \Pi M(4, 5, 6, 7, 8, 12) \cdot d(0, 1, 2, 3, 9, 11, 14)$

Solution: A logic circuit output may be either 0 or 1 but some circuit has output that will never occur for certain combinations of inputs. These conditions are don't care conditions and represented by X in K-map.



The simplified Boolean expression is $F = A(C + D)$.

1.4 DIGITAL CIRCUITS

The classification of digital circuitsis shown in Fig. 1.8.

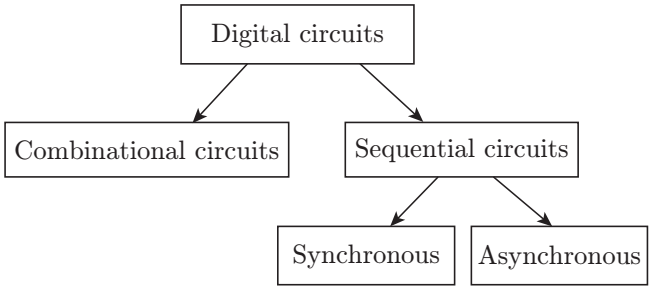


Figure 1.8 | Types of digital circuits.

1.4.1 Combinational Circuits

A logic circuit whose output at any instant of time depends only on the present input is called combinational circuit. It contains no memory element.

1.4.1.1 Half Adder

A half adder is a combinational circuit for the addition of two 1-bit numbers. X and Y are inputs and outputs are sum (S) and carry (C) (Fig. 1.9 and Table 1.4).

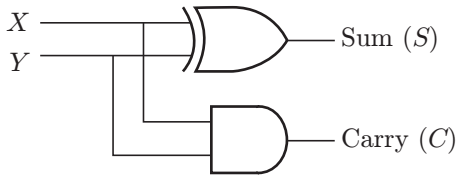


Figure 1.9 | Logic diagram of a half adder.

Table 1.4 | Truth table

Inputs		Outputs	
X	***Y***	**Carry (*C*)**	**Sum (*S*)**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The sum and carry are as follows:

$$S = X'Y + XY'$$
$$C = XY$$

1.4.1.2 Full Adder

A full adder is a combinational circuit that performs the addition of 3 bits (2 significant bits and previous carry). It consists of three inputs and two outputs, two input bits that are to be added, the third input is the carry from the previous position (Fig. 1.10 and Table 1.5).

Table 1.5 | Truth table

Inputs			Outputs	
X	***Y***	C_{in}	**Sum (*S*)**	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The sum and carry are as follows:

$$S = X'Y'C_{in} + X'YC'_{in} + XYC_{in}$$
$$C_{out} = AC_{in} + XY + YC_{in}$$

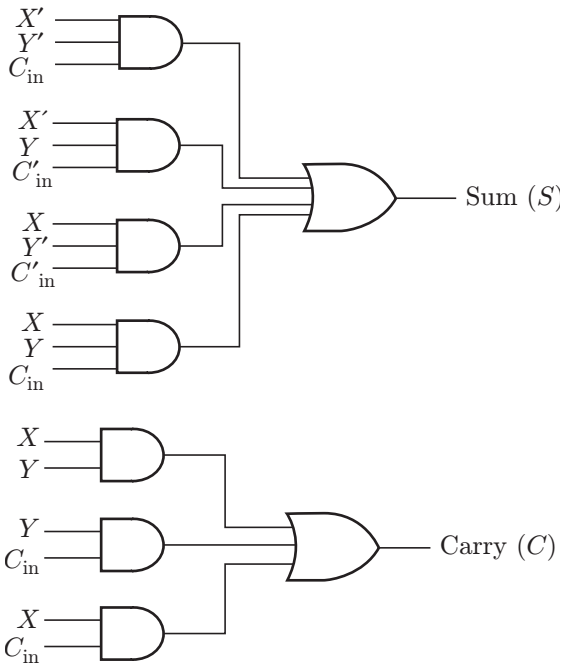


Figure 1.10 | Logic diagram of a full adder.

1.4.1.3 Half Subtractor

A half subtractor is a combinational circuit that subtracts 2 bits (minuend and subtrahend) and produces their differences and borrow as output (Fig. 1.11 and Table 1.6).

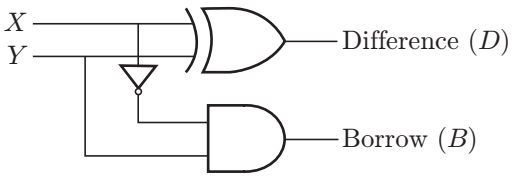


Figure 1.11 | Logic diagram of a half subtractor.

Table 1.6 | Truth table

Inputs		Outputs	
X	***Y***	**Difference (*D*)**	**Borrow (*B*)**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

The difference and borrow are as follows:

$$D = XY' + X'Y$$
$$B = XY$$

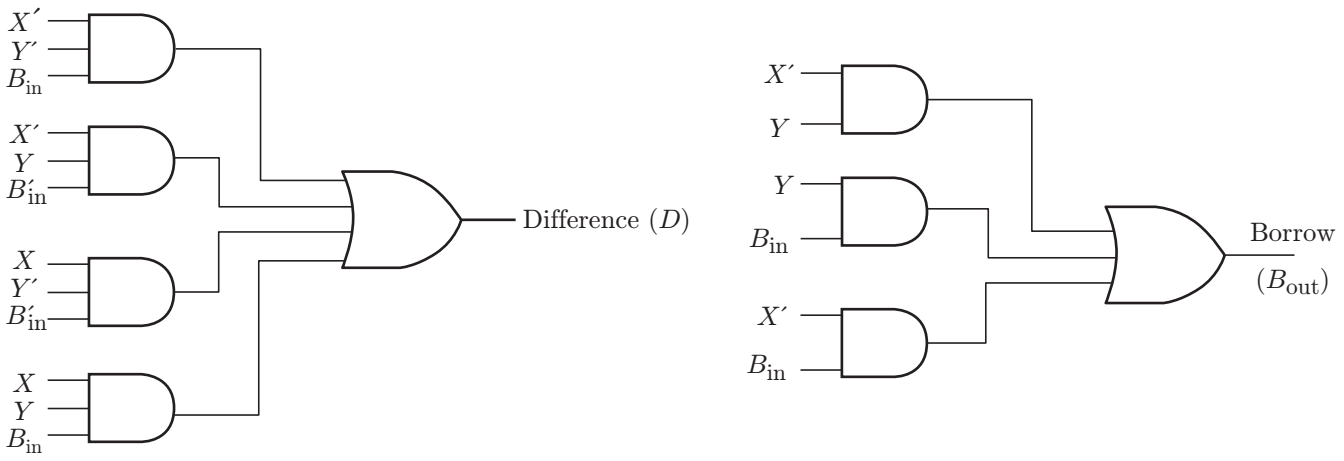


Figure 1.12 | Logic diagram of a full subtractor.

1.4.1.4 Full Subtractor

A full subtractor is a combinational circuit that performs subtraction involving 3 bits (minuend bit, subtrahend bit and borrow from previous stage) (Table 1.7 and Fig. 1.12).

Table 1.7 | Truth table

Inputs			Outputs	
X	***Y***	B_{in}	**Difference (*D*)**	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The difference and borrow are as follows:

$$D = X'Y'B_{in} + X'YB'_{in} + XY'B'_{in} + XYB_{in}$$
$$B = X'Y + X'YB_{in} + YB_{in}$$

1.4.1.5 Parallel Adder (Ripple Carry Adder)

A parallel adder consists of full adders connected in cascade, that is, carry output of each adder is connected to the carry input of the next higher-order adder (Fig. 1.13). An n-bit parallel adder is constructed using $(n-1)$ full adders and 1 half adder. As full adder consists of 2 half adders and 1 OR gate, so n-bit parallel adder can be made from $(2n-1)$ half adders and $(n-1)$ OR gates.

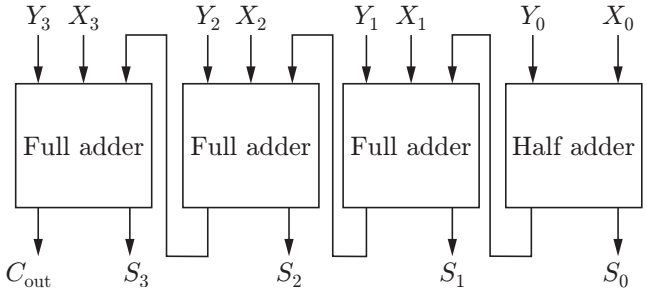


Figure 1.13 | Block diagram of a parallel adder.

1.4.1.6 Look-Ahead Carry Header

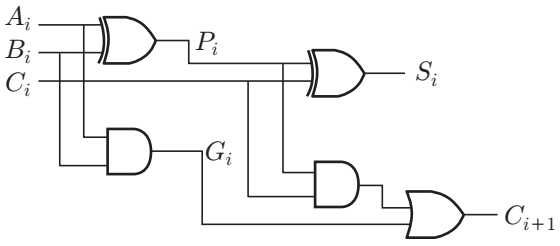
In a parallel adder, carry propagation delay is present. To overcome this difficulty, look-ahead carry adder is used. It uses logic gates to look at lower-order bits of augend and addend to check if higher-order carry is generated or not. It uses two functions, carry generate and carry propagate, for the same (Fig. 1.14).

Problem 1.13: How many AND and OR gates are required to design 7-bit look-ahead carry adder.

Solution: To design a 7-bit look-ahead carry adder, we need

$$\text{AND gate} = \frac{n(n+1)}{2} = \frac{7(7+1)}{2} = 28$$
$$\text{OR gate} = n = 7$$

where n is the number of bits.



Full Adder circuit with P_i and G_i

Carry propagation $P_i = A_i \oplus B_i$

Carry generation $G_i = A_i B_i$

Output sum $S_i = P_i \oplus C_i$

Output carry $C_{i+1} = G_i + P_i C_i$

Figure 1.14 | Logic diagram of a look-ahead adder.

1.4.1.7 Decoder

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines (Fig. 1.15). The size of a decoder is n to m where $m \leq 2^n$. It is used to route input data to a specified output line for memory addressing (Fig. 1.16 and Table 1.8).

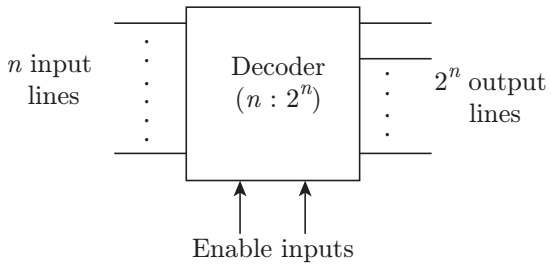


Figure 1.15 | Block diagram of a decoder.

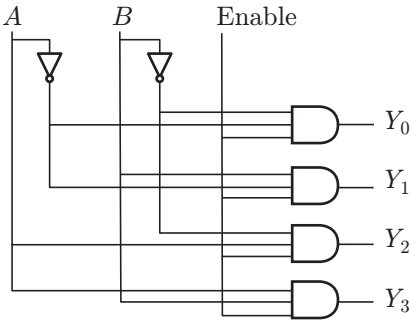


Figure 1.16 | Logic diagram of a 2X4 decoder.

Table 1.8 | Truth table of 2X4 decoder

Inputs		Outputs			
A	B	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

1.4.1.8 Encoder

An encoder is a combinational circuit that has 2^n (or fewer) inputs and n output lines (Fig. 1.17). It accepts an active level on one of its inputs representing a decimal or octal digit and converts it to a coded output.

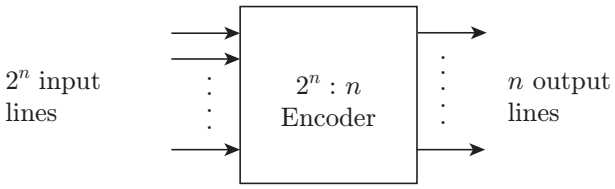


Figure 1.17 | Block diagram of an encoder.

1.4.1.9 Multiplexer

A multiplexer is a combinational circuit that provides inputs one by one at single output; and which input will be out at any instant depends upon the combination at select lines (Fig. 1.18). It converts parallel data to serial data (Fig. 1.19 and Table 1.9).

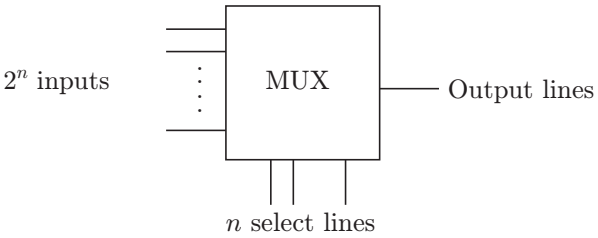


Figure 1.18 | Block diagram of a multiplexer.

Table 1.9 | Truth table of 4 to 1 MUX

Select Lines		Outputs
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

The Boolean function is

$$Y = I_0S_0'S_1' + I_1S_0S_1' + I_0S_0'S_1 + I_0S_0S_1$$

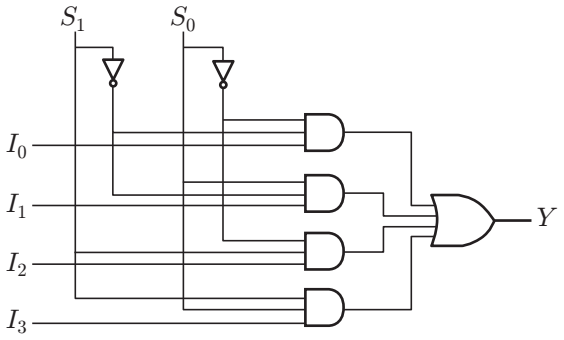


Figure 1.19 | Logic diagram of 4 to 1 MUX.

1.4.1.10 Boolean Function Implementation Using Multiplexers

1. Take one variable for input lines and others for selection lines.
2. Write the minterms with the variable in complemented form in the first row and with variable selected in uncomplemented form in the second row.
3. Then encircle the minterms which are present in the function.
   - If there is circled variable in the column, then we put 0 on the corresponding line.
   - If there are circled variables, then we put 1 on the line.
   - If bottom variable is circled and top is not circled, A is used in input line.
   - If bottom variable is not circled and top is circled, A' is used as input to that line.

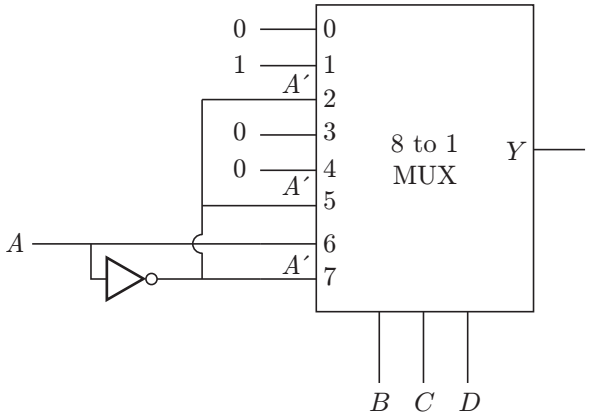
Problem 1.14: Implement the function $F(A, B, C, D) = \Sigma m(1, 2, 5, 7, 9, 14)$ using MUX.

Solution: Select variable A for input as B, C and D for selection lines.

As there are four variables so size of MUX is $2^{N-1} = 2^3 = 8$ to 1 MUX.

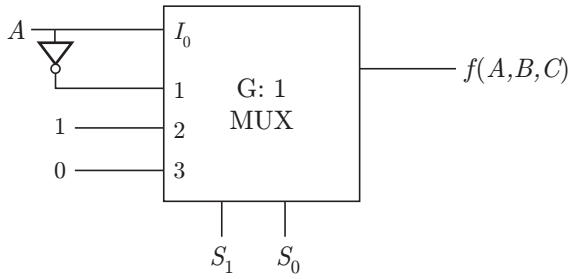
1. Minterms with A' are $0 - 7$.
2. Minterms with A are $8 - 15$.

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	(1)	(2)	3	4	(5)	6	(7)
A	8	(9)	10	11	12	13	(14)	25
	0	1	A'	0	0	A'	A	A'



Logic diagram

Problem 1.15: The combinatorial circuit below is equivalent to _____.



Solution:

	I_0	I_1	I_2	I_3
A'	0	①	②	3
A	④	5	⑥	7
	A	A'	1	0

It is equivalent to $\sum m(1, 2, 4, 6)$.

1.4.1.11 Demultiplexer

A demultiplexer is a combinational circuit that receives information on a single line and transmits this on one of 2^n of possible output line (Fig. 1.20). The selection of possible output line is controlled by bit values of n select lines.

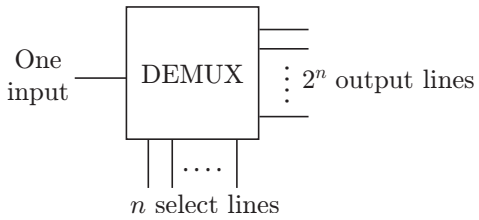


Figure 1.20 Block diagram of a demultiplexer.

Implementation of higher order decoders and multiplexers by using lower order. Whenever odd number implementation is there, one NOT gate is required.

Example 1.3

Design 4 ×16 decoders using 2 × 4 decoders

16/4 = 4

4/4 = 1

Total = 5

Example 1.4

Design 8 × 256 decoders using 4 × 64 decoders

256/64 = 4

4/64 = 0

Total = 4

Example 1.5

Design 6 × 64 decoders using 2 × 4 decoders

64/4 = 16

16/4 = 4

4/4 = 1

Total = 21

1.4.2 Sequential Circuits

A logic circuit whose output at any instant of time depends on the present input as well as past output is known as sequential circuit. It contains memory element.

1.4.2.1 Flip–Flops

A flip−flop or bistable multivibrator is a synchronous bistable circuit made up of logic gates that can exist in either of two stable states in presence of a falling or raising edge of a clock signal. Flip−flops can change its state by means of some external signal. It can "store" binary information. There are different types of flip−flops based on the manner the inputs affect their output state:

1. *RS* flip−flop
2. *D* flip−flop
3. *JK* flip−flop
4. *T* flip−flop

RS Flip−Flop (Direct Coupled)

This sequential logic circuit has two stable states ($Q = 1$ set state and $Q = 0$ reset state) achieved by giving proper inputs to R and S inputs (Figs. 1.21 and 1.22 and Table 1.10). If the circuit is in one particular state it continues to remain in that state and can store 1-bit data.

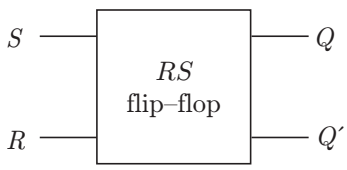


Figure 1.21 Block diagram of an *RS* flip−flop.

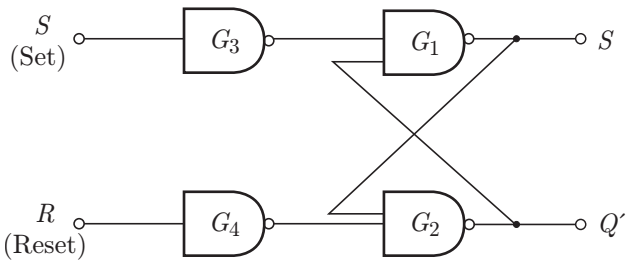


Figure 1.22 | Logic diagram of an *RS* flip−flop.

Table 1.10 | Truth table of *RS* flip−flop

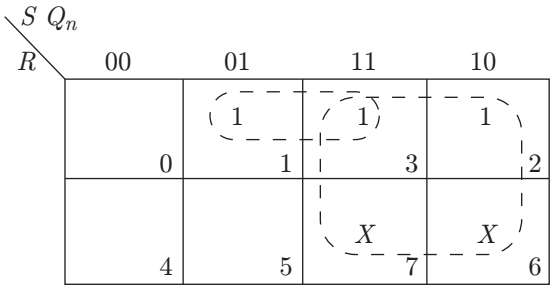
Inputs		Outputs
S_n	R_n	Q_{n+1}
0	0	Q_n (No change)
0	1	0 (Reset)
1	0	1 (Set)
1	1	Prohibited

The characteristic equation which gives the algebraic description of the next state of a flip−flop obtained by K-map simplification is as follows (see Table 1.11):

$$Q_{n+1} = S + R'Q_n$$

Table 1.11 | Truth table

Flip−Flop Inputs		Present State	Next State
R_n	S_n	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	X



$Q_{n+1} = S + R'Q_n$

Clocked *RS* Flip−Flop

This circuit sets or resets the memory cell in synchronization with the clock pulse (Figs. 1.23 and 1.24 and Table 1.12). If clock pulse is not present, the gates G3 and G4 are inhibited; their outputs are 1.

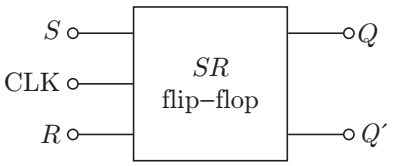


Figure 1.23 | Block diagram of a clocked *SR* flip−flop.

Table 1.12 | Truth table of *SR* flip−flop

Clock	Flip−Flop Inputs		Outputs
CLK	S_n	R_n	Q_{n+1}
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Prohibited
0	X	X	Q_n

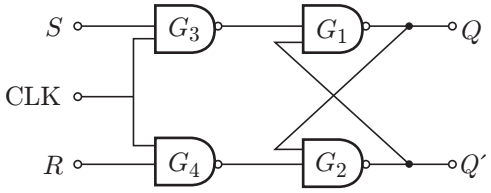
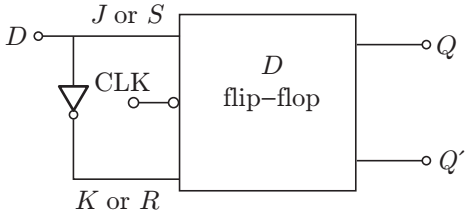


Figure 1.24 | Logic diagram of a clocked *SR* flip−flop.

D Flip−Flop

This circuit has only one input terminal, and output is followed by input. It has delay of exactly one clock cycle (Figs. 1.25 and 1.26 and Table 1.13). The input to *R* is through an inverter from *S*.



JK or *SR* Flip−Flop converted to *D*

Figure 1.25 | Block diagram of a *D* flip−flop.

Table 1.13 | Truth table of D flip−flop

Flip−Flop Inputs	Present State	Next State
D_n	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

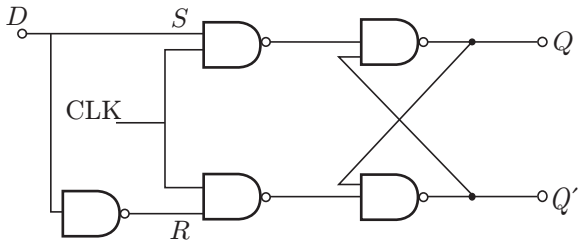


Figure 1.26 | Logic diagram of a D flip−flop.

The characteristic equation is $Q_{n+1} = D$ (Fig. 1.27)

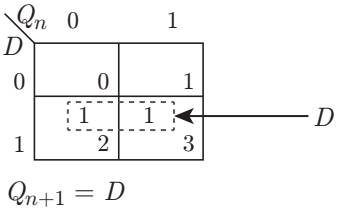


Figure 1.27 | K-Map

JK Flip−Flop

The uncertainty in SR flip−flop when $S_n = R_n = 1$ can be avoided by using JK flip−flop (Figs. 1.28 and 1.29 and Table 1.14). The output of JK flip−flop is connected back to the inputs of gate. The inputs J and K are ANDed with Q and Q' to obtain S and R inputs:

$$S = JQ'$$
$$R = J \cdot Q$$

The flip−flop sets when input J is high and resets when input K is high. If $J = K = 1$, the flip−flop will toggle on the next positive clock.

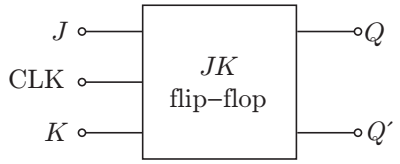


Figure 1.28 | Block diagram of a JK flip−flop.

Table 1.14 | Truth table of JK flip−flop

Flip−Flop Inputs		Present State	Next State
J_n	K_n	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

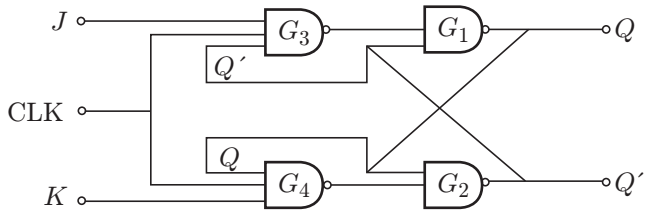


Figure 1.29 | Logic diagram of a JK flip−flop.

The characteristic equation is $Q_{n+1} = JQ'_n + KQ_n$ (Fig. 1.30).

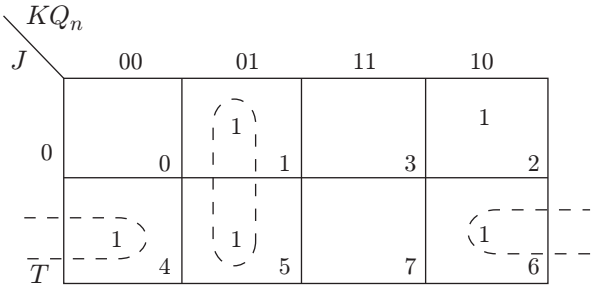


Figure 1.30 | K-Map

Repetition of toggle for a single-clock pulse is known as **race-around problem**. The output of JK flip−flop will oscillate between 0 and 1 during a single-clock pulse so that at the end of clock pulse the value of Q is uncertain. This problem can be avoided by decreasing the pulse width, by edge triggering or increasing propagation delay.

Master−Slave *JK* Flip−flop

It is a cascade of two RS flip−flops (master and slave) with feedback from outputs of slave to input of master to avoid race-around problem. Positive clock pulses are applied to first flip−flop and pulses are inverted before applying to slave flip−flop (Fig. 1.31).

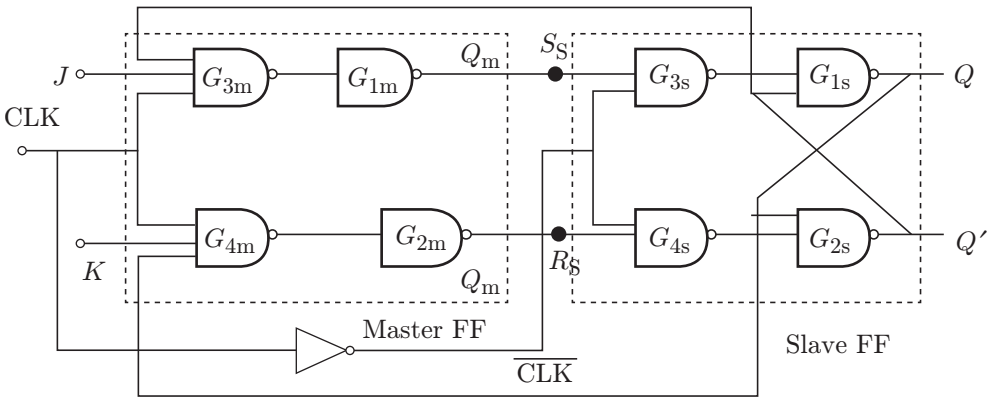


Figure 1.31 | Block diagram of a master−slave *JK* flip−flop.

T Flip−flop

It is also known as toggle flip−flop, which is obtained by connecting both inputs of *JK* flip−flop (Fig. 1.32 and Table 1.15).

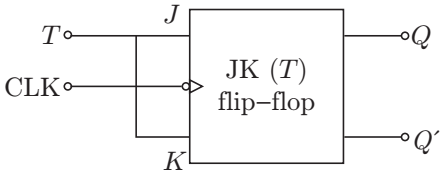
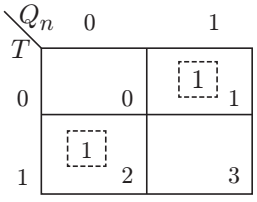


Figure 1.32 | Block diagram of a *T* flip−flop.

Table 1.15 | Truth table of *T* flip−flop

Flip−Flop Inputs	Present State	Next State
T_n	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0



$$Q_{n+1} = T'Q_n + TQ'_n$$

Figure 1.33 | K-Map

Excitation Table of Flip−Flop

The excitation table of flip−flop states is given in Table 1.16.

Conversion Equation

The flip−flop conversion equations are given in Table 1.17.

Table 1.16 | Excitation Table of Flip−flop States

Flip−Flop States		RS Flip−Flop Inputs		JK Flip−Flop Inputs		D Flip−Flop Inputs	T Flip−Flop Inputs
Present State (Q_n)	Next State (Q_{n+1})	*R*	*S*	*J*	*K*		
0	0	X	0	0	X	0	0
0	1	0	1	1	X	1	1
1	0	1	0	X	1	0	1
1	1	0	X	X	0	1	0

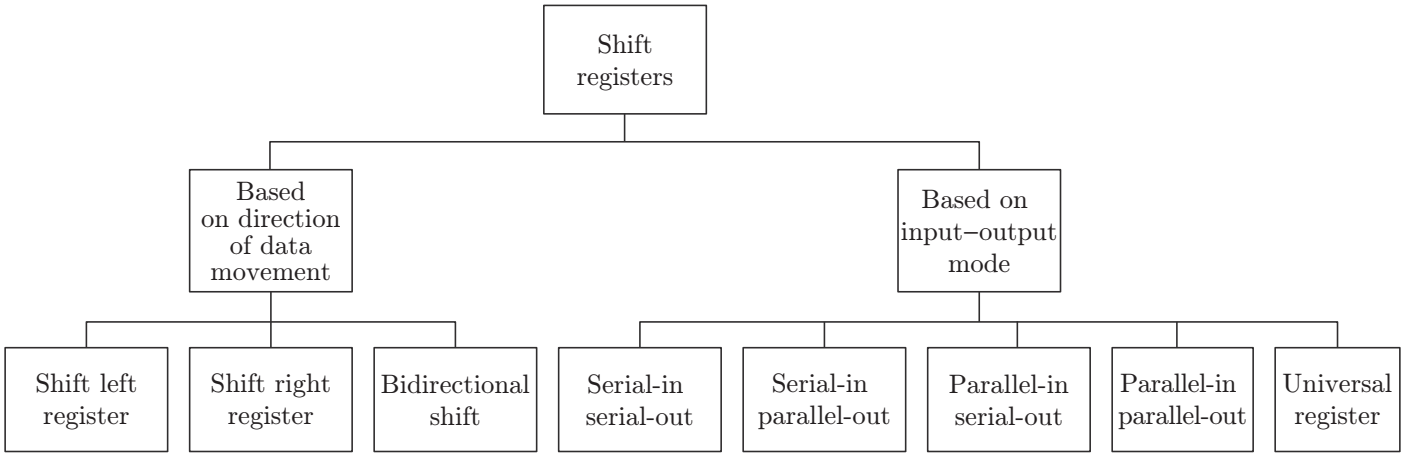


Figure 1.34 | Types of registers

Table 1.17 | Flip–flop conversion equations

1	SR to JK	$S = JQ'$ $R = K \cdot Q$
2	SR to D	$S = D$ $R = D'$
3	SR to T	$S = TQ'$ $R = TQ$
4	JK to SR	$J = S$ $K = R$
5	JK to D	$J = D$ $K = D'$
6	JK to T	$J = T$ $K = T$
7	D to JK	$D = TQ' + K'Q$
8	D to SR	$D = S + R'Q$
9	D to T	$D = T \oplus Q$
10	T to SR	$T = SQ' + RQ$
11	T to D	$T = D \oplus Q$
12	T to JK	$T = JQ' + KQ$

Applications of Flip–Flops

Registers

Registers are collection of flip–flops, and each flip–flop is capable of storing single bit of information. They make stored information available to the logic elements for the computation. There are different types of registers as shown in Fig. 1.34.

1. **Serial-In/Serial-Out Shift Register:** It accepts data serially on single input line and produces output in serial form. Data can be shifted right or left.
   - **Shift right register:** In shift right register, serial data is entered from the left side of a register and leaves from the right side. For example, a 4-bit number 1101 ($Q_D Q_C Q_B Q_A$) is entered serially into first flip–flop D from the left side (Fig. 1.35).

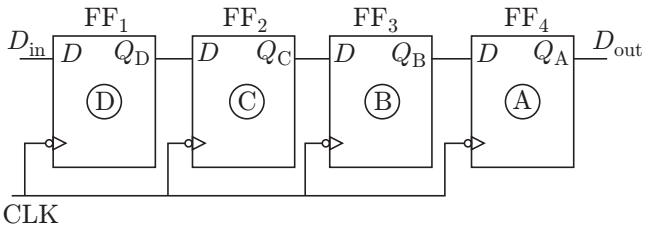


Figure 1.35 | 4-Bit shift right register.

The content of register after each clock pulse is shown in Table 1.18.

Table 1.18 | Content of register after each clock pulse

Clock Pulse	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	0	1	0
4	1	1	0	1

- **Shift left register:** In shift left register, serial data is entered from the right side of the register and leaves from the left side. For example, a 4-bit number 1101 ($Q_DQ_CQ_BQ_A$) is entered serially into input of shift left register (Fig. 1.36).

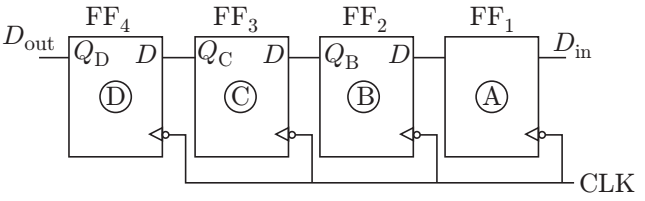


Figure 1.36 4-Bit shift left register.

The content of register after each clock pulse is given in Table 1.19.

Table 1.19 Content of register after each clock pulse

Clock Pulse	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	0
4	1	1	0	1

Note: A universal register can perform all the operations of shift register.

2. **Serial-In to Parallel-Out (SIPO):** The register is loaded with data serially, one bit at a time, with the stored data being available at the output in parallel form (Fig. 1.37).
The content of register after each clock pulse is given in Table 1.20.

Table 1.20 Content of register after each clock pulse

Clock Pulse	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
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

3. **Parallel-In to Serial-Out (PISO):** The parallel data is loaded simultaneously into the register and is shifted out of the register serially, one bit at a time (Fig. 1.38).

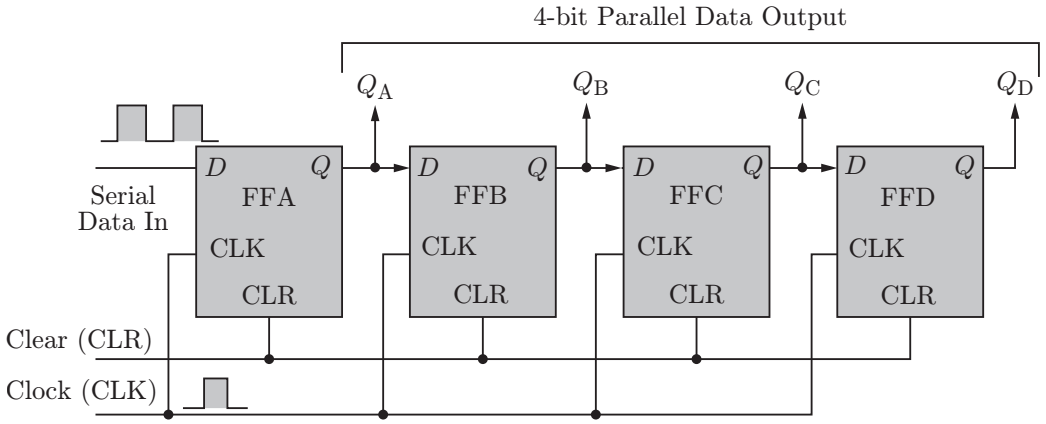


Figure 1.37 4-Bit SIPO register.

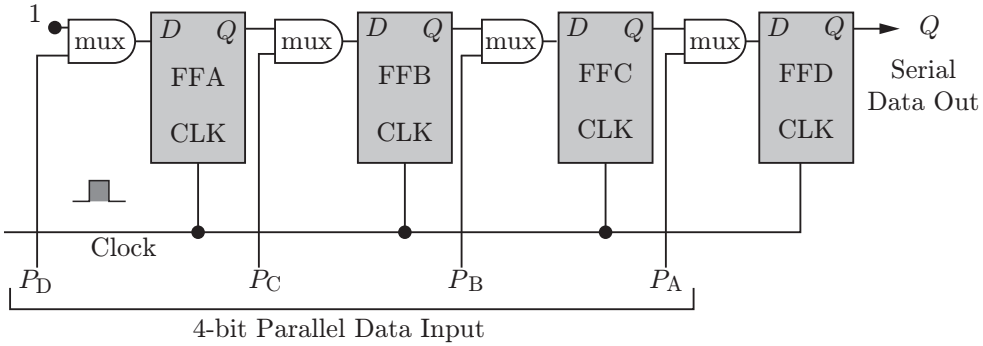


Figure 1.38 4-Bit PISO register.

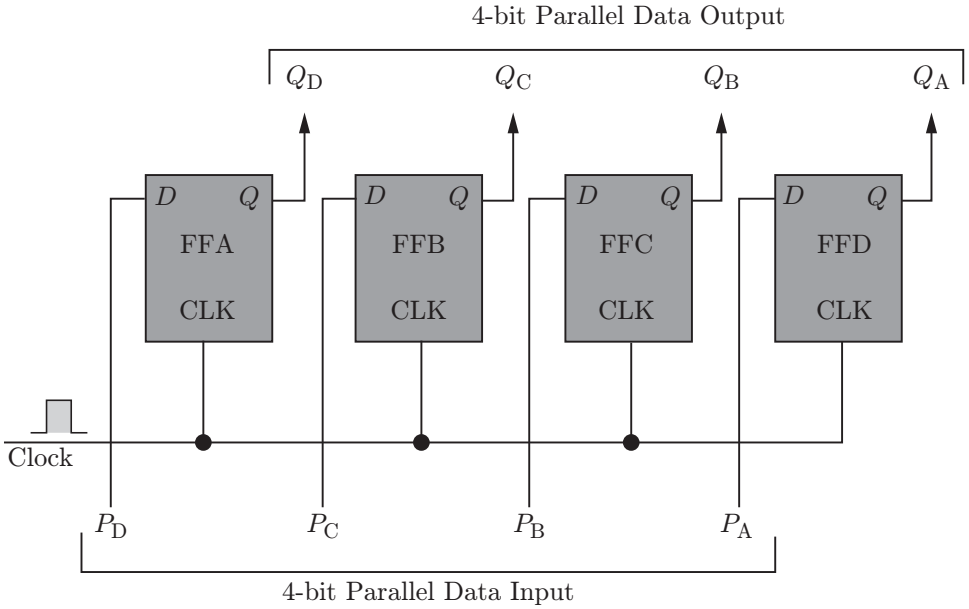


Figure 1.39 | 4-Bit PIPO register.

4. **Parallel-In to Parallel-Out (PIPO):** The parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse (Fig. 1.39).

Counters

A counter is a sequential logic circuit that is capable of counting the number of clock pulses arriving at its input in a cyclic sequence. It is just a register that shifts through a predetermined sequence of states upon the application of input clock pulses. There are two different types of counters − synchronous counter (parallel counter) and asynchronous counter (serial or ripple counter) (Table 1.21).

Table 1.21 | Differences between asynchronous and synchronous counter.

Asynchronous Counter	Synchronous Counter
The output of first flip−flop drives the clock for next flip−flop. So all flip−flops are not clocked simultaneously.	There is no connection between output of first flip−flop and clock input of next flip−flop. All the flip−flops are clocked simultaneously.
Speed is low.	Speed is high.
Simple logic circuit as number of states increases.	Complex design as number of states increases.
Example: Ring counter, Johnson counter.	Example: Binary ripple counter, Up-down counter.

The counter with n flip−flops has a maximum modulus 2^n so it can count from 0 to 2^{n-1}. It can also produce MOD numbers less than 2^n by skipping the states.

1. **4-Bit Asynchronous Up Counter:** Figures 1.40−1.42 show a 4-bit asynchronous counter constructed using JK flip−flops. The clock input of first flip−flop (the one with the Q_0 output) is connected to the external clock. The clock of second flip−flop drives from output Q_0 of the first flip−flop and so on. Each flip−flop input ($J = K = 1$) is connected to HIGH voltage so it toggles when negative edge is arrived at clock input and counts from 0000 to 1111.

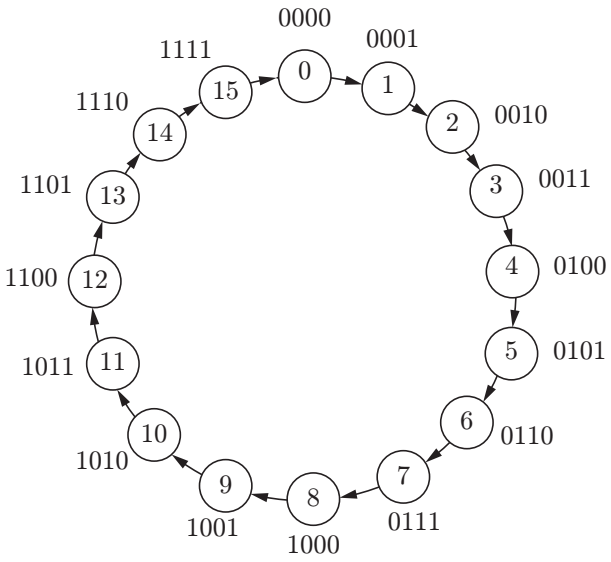


Figure 1.40 | State diagram of a 4-bit asynchronous counter.

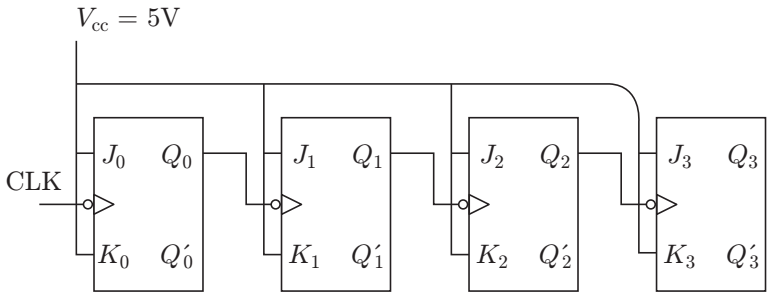


Figure 1.41 | Logic diagram for a 4-bit asynchronous counter.

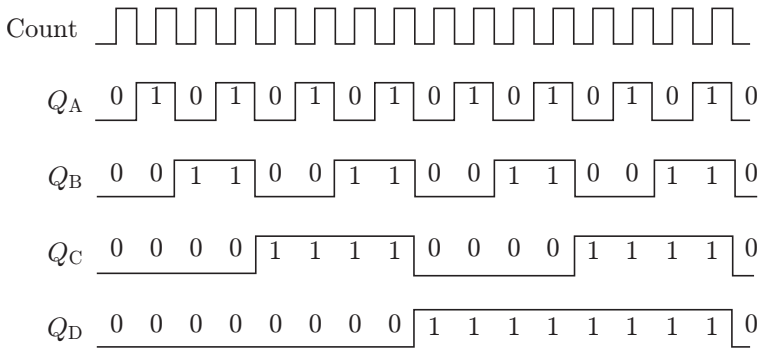


Figure 1.42 | Timing diagram for a 4-bit asynchronous counter.

2. MOD-10 Asynchronous Counter: A MOD-10 asynchronous counter has 10 states and number of flip−flops required are $2^4 \geq 10$ (Figs. 1.43 and 1.44). It counts the sequence from 0000 to 1001 and after that it returns to the initial state.

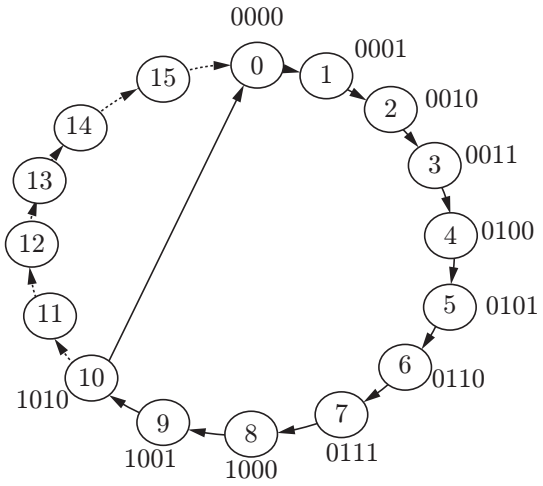


Figure 1.43 | State diagram of a 4-bit MOD-10 counter.

3. 4-Bit Synchronous Counter: In a 4-bit synchronous counter, the least significant bit of flip−flop is connected at high level and other inputs of *JK* flip−flop are driven by some combination of flip−flop outputs. Figure 1.45 shows the logic diagram of a 4-bit synchronous counter. The output of flip−flop0 always toggle, and that of flip−flop1 toggles when Q_0 is 1, otherwise it maintains the previous state. The flip−flop 2 toggles when both Q_1 and Q_0 are 1. The flip−flop 3 toggles when Q_2, Q_1 and Q_0 are 1 (Table 1.22).

Table 1.22 | Truth table for 4-bit synchronous counter

Count	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	0	0
15	1	1	0	1

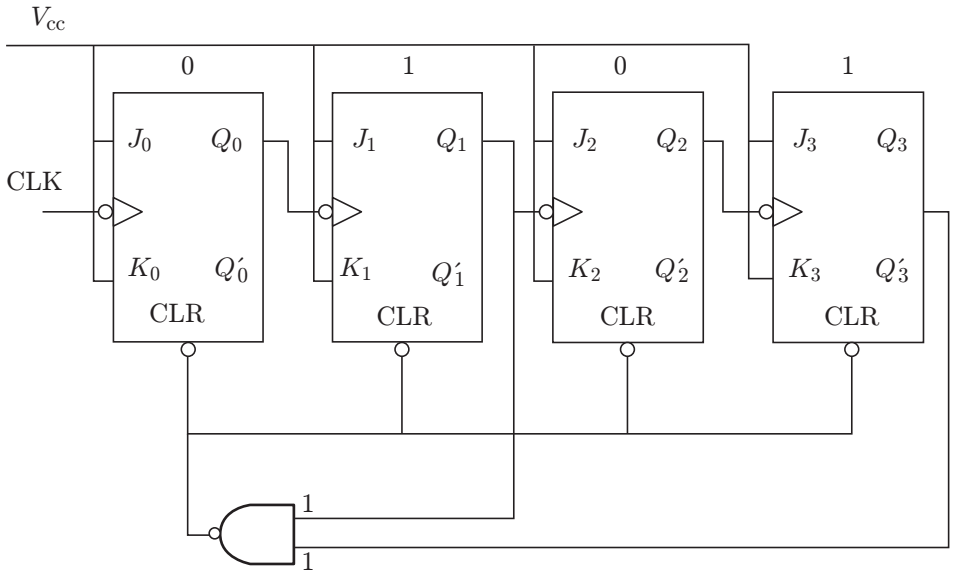


Figure 1.44 | Logic diagram for a MOD-10 asynchronous counter.

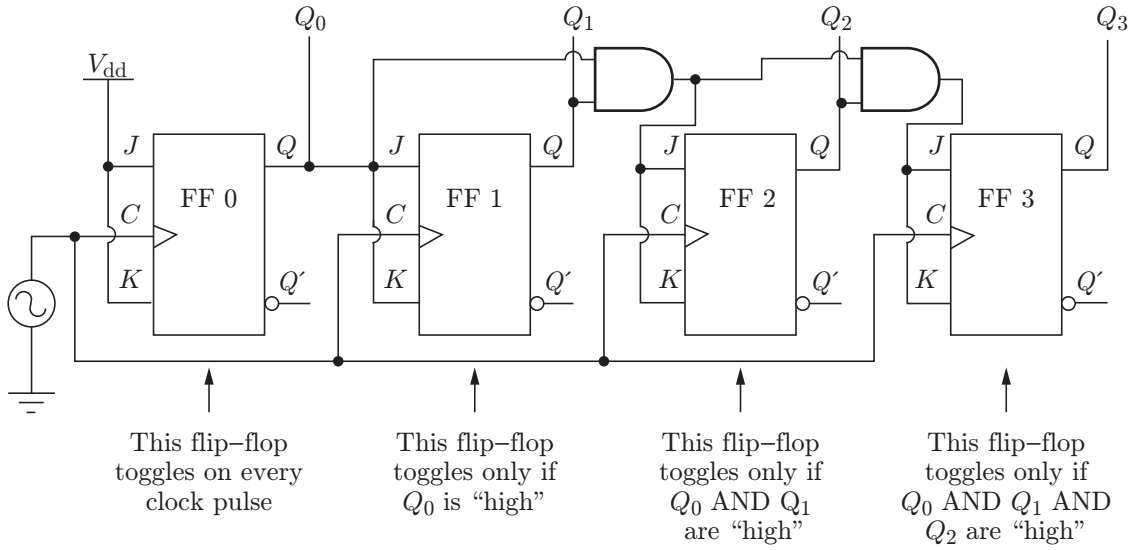


Figure 1.45 | Logic diagram of a 4-bit synchronous counter.

4. MOD-5 Synchronous Counter:

Step 1: Number of states is five, so counting sequence 0 to 4.

Flip−flops required are $2^n \geq 5$, so $n = 3$.

Let us label these flip−flops as A, B and C.

Step 2: Obtain the excitation table for *JK* flip−flop (Table 1.23).

Step 3: Develop state table using the excitation table (Table 1.24).

Step 4: K-map simplification for finding expressions corresponding to each input (Fig. 1.46).

Table 1.23 | Excitation table for *JK* flip−flop

Flip−Flop States		**JK Flip−Flop Inputs**	
Present State (Q_n)	**Next State (Q_{n+1})**	***J***	***K***
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table 1.24 | State table

Present State			Next State			Flip–Flop Inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

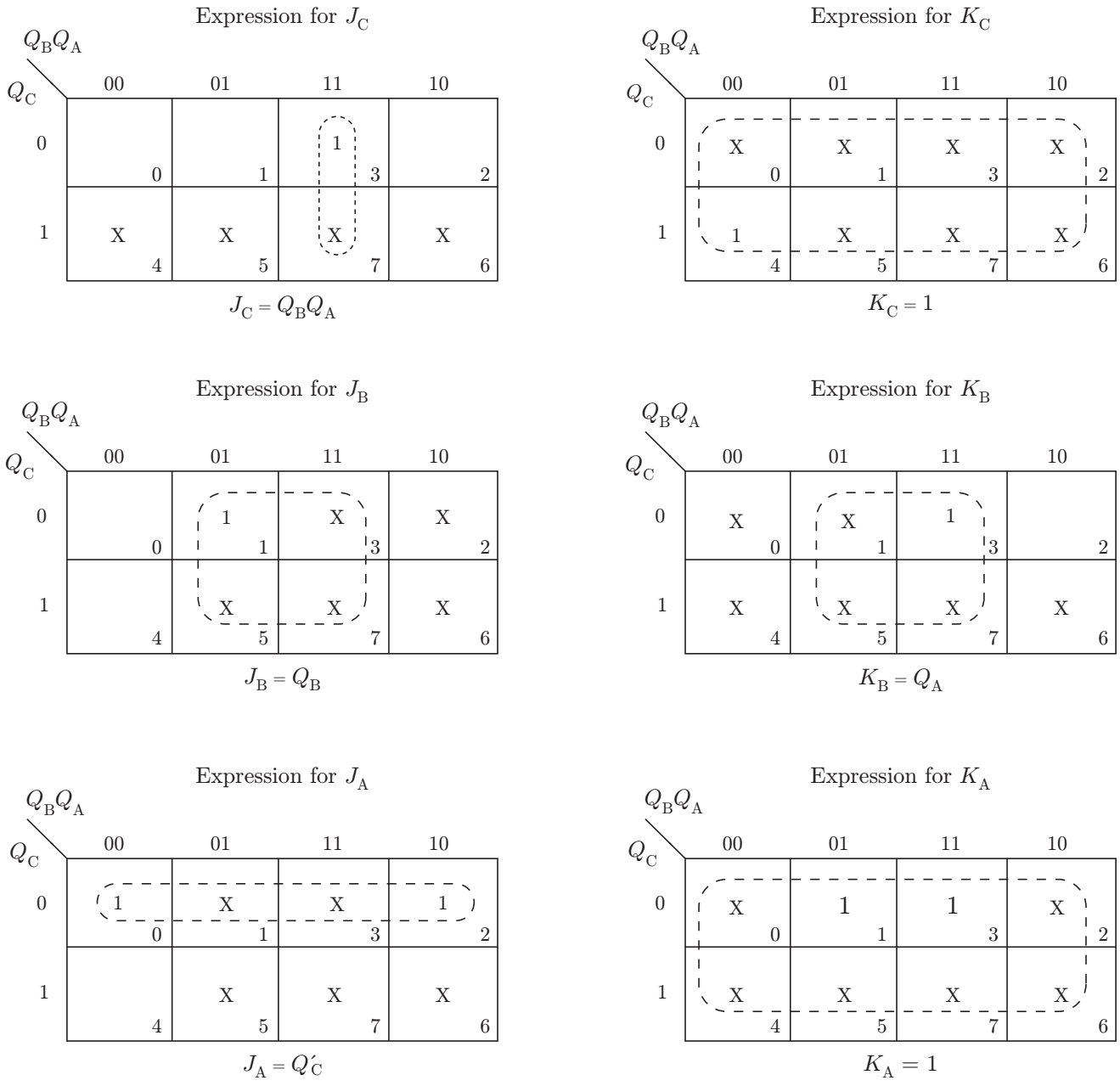


Figure 1.46 | K-map simplification.

Step 5: Logic diagram of MOD-5 counter (Fig. 1.47).

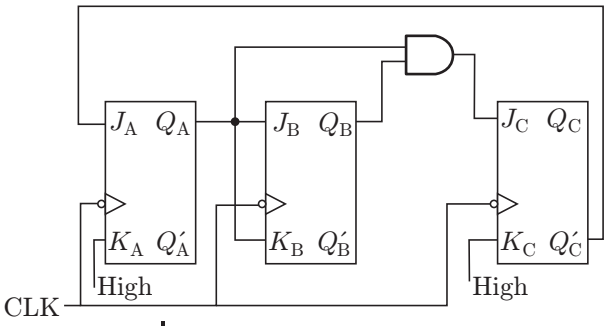


Figure 1.47 | Logic diagram of a MOD-5 counter.

5. **Shift register counter:** It is basically a shift register whose serial output is connected back to serial input in order to produce special sequences.
6. **Ring counter:** A ring counter is a circular shift register in which output of last flip−flop is connected to the input of the first in a ring (Fig. 1.48 and Table 1.25). In this, only one flip−flop is in state one while others are in their zero states. Each state repeats after every n clock cycles, where n is the number of flip−flops used. Initially the counter is reset by applying low signal to active low CLR input. All flip−flops reset except the first flip−flop.

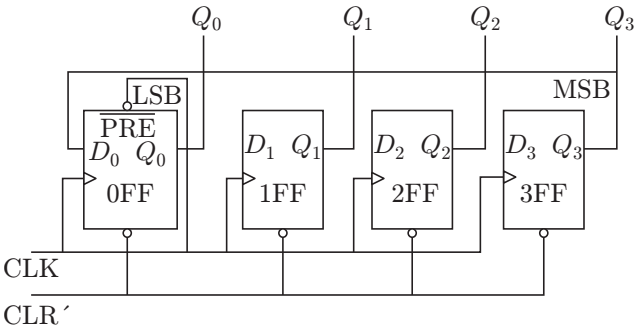


Figure 1.48 | Logic diagram of a 4-bit ring counter.

Table 1.25 | Truth table of 4-bit ring counter

CLK	CLR′	Q_3(MSB)	Q_2	Q_1	Q_0
0	0	0	0	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0

(*Continued*)

Table 1.25 | Continued

CLK	CLR′	Q_3(MSB)	Q_2	Q_1	Q_0
1	1	0	0	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0

7. **Johnson counter:** A Johnson counter (switch-tail ring counter, twisted-ring counter or walking-ring counter) is a variation of shift register counter, where the complement of the output of the last flip−flop is fed back as input to the first flip−flop (Fig. 1.49 and Table 1.26). The counter has counting sequence length of twice the length of the shift register, that is, number of flip−flops.

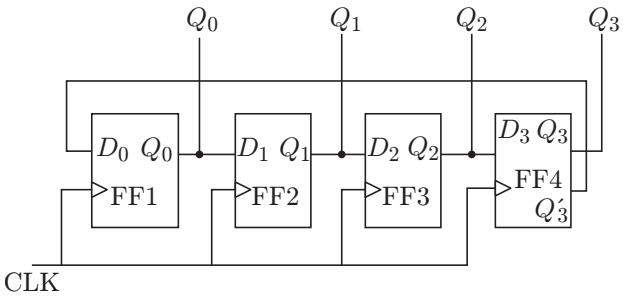


Figure 1.49 | 4-Bit Johnson counter.

Table 1.26 | Truth table of 4-bit Johnson counter

CLK	Q_0	Q_1	Q_2	Q_3	Q'_3
0	0	0	0	0	1
1	1	0	0	0	1
2	1	1	0	0	1
3	1	1	1	0	1
4	1	1	1	1	0
5	0	1	1	1	0
6	0	0	1	1	0
7	0	0	0	1	0
8	0	0	0	0	1

1.5 DIGITAL LOGIC FAMILIES

Refer Fig. 1.50 and Table 1.27 for classification and pros and cons of logic families.

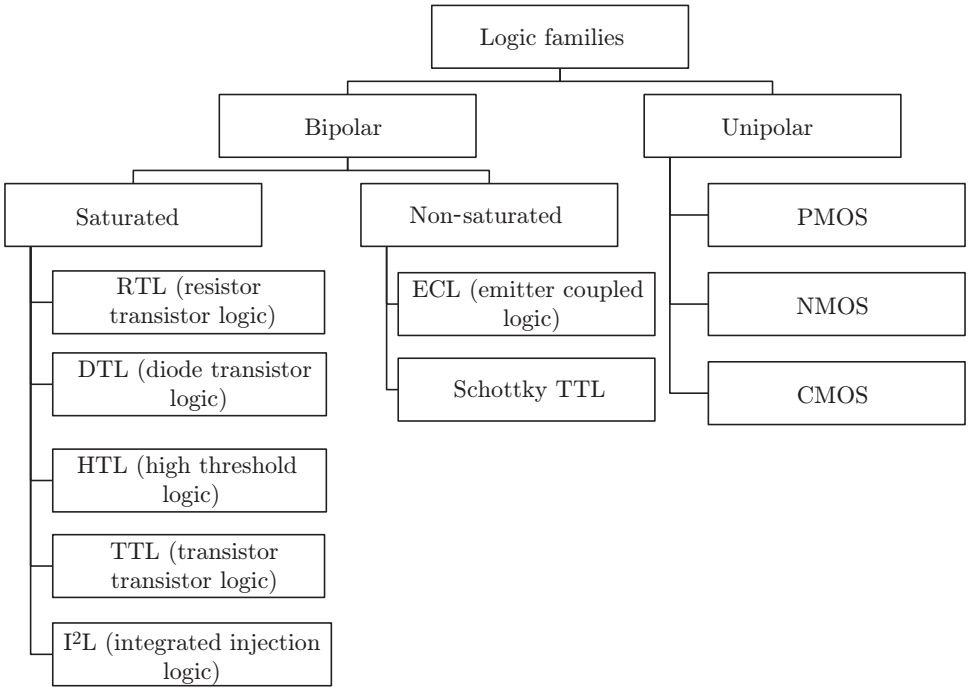


Figure 1.50 | Digital logic families.

Table 1.27 | Pros and cons of logic families

Logic Families	Pros	Cons
RTL	Economical and simple to design, can easily interface with other logic families	Low threshold and fan out, poor noise immunity, low speed and low power dissipation
DTL	High speed with 30-ns propagation delay, low power dissipation	Poor noise immunity and fan out
TTL	Fastest saturating logic family, high-speed Schottky, low-power Schottky, noise margin about 0.4 V, more fan out, can drive up to 10 gates	Power dissipation is more than MOS gates, noise immunity is not very high
ECL	Highest speed because transistors operate in active region, propagation delay of 2-ns, parameters resistant to temperature change	Very low noise margin, highest power dissipation, high cost
NMOS	Very low power dissipation, large fan out up to 20, very high noise margin	High power dissipation than CMOS, low speed of operation, large propagation delay per gate
I^2L	Composed of BJT so high speed of operation, low power dissipation, low power supply requirement	Lower packing density than NMOS, lower noise margin, external resistance required for proper functioning
CMOS	Large fan out (>50), lowest power dissipation, very high noise immunity, temperature resistant	Increased cost, less packing density than NMOS

IMPORTANT FORMULAS

1. Null − All the outputs are zero.
2. Identity − All the outputs are one.
3. X inhibit Y − Represented as $(x \cdot \sim y)$
4. Y inhibit X − Represented as $(y \cdot \sim x)$
5. Transfer − Same as y
6. AND − $x \cdot y$ (Similar to intersection in set theory)
7. OR − $x + y$ (Similar to union in set theory)
8. NOT X− $\sim x$ (Also known as complement of x)
9. NOT Y− $\sim y$ (Also known as complement of y)
10. XOR − $(\sim x \cdot y) + (x \cdot \sim y)$ (Also known as parity checker)
11. XNOR − $(x \cdot y) + (\sim x \cdot \sim y)$
12. NAND − $\sim(x \cdot y)$
13. NOR − $\sim(x + y)$
14. X implication Y $(\sim x + y)$
15. Y implication X $(\sim y + x)$
16. $X + 0 = X$
17. $X + (\sim X) = 1$
18. $X + X = X$
19. $X + 1 = 1$
20. $X + Y = Y + X$
21. $X + (Y + Z) = (X + Y) + Z$
22. $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$
23. $\sim(X + Y) = \sim X \cdot \sim Y$ (Also called **DeMorgan's law**)
24. $X + (X \cdot Y) = X$
25. $\sim(\sim X) = X$

Duality Theorem:

26. $X \cdot 1 = X$
27. $(X \cdot \sim X) = X \cdot X' = 0$
28. $X \cdot X = X$
29. $X \cdot 0 = 0$
30. $X \cdot Y = Y \cdot X$
31. $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
32. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
33. $(X \cdot Y)' = X' + Y'$
34. $X \cdot (X + Y) = X$

Half Adder:

35. $S = X'Y + XY'$
36. $C = XY$

Full Adder:

37. $S = X'Y'C_{\text{in}} + X'YC'_{\text{in}} + XY'C'_{\text{in}} + XYC_{\text{in}}$
38. $C_{\text{out}} = AC_{\text{in}} + XY + YC_{\text{in}}$

Half Subtractor:

39. $D = XY' + X'Y$
40. $B = X'Y$

Full Subtractor:

41. $D = X'Y'B_{\text{in}} + X'YB'_{\text{in}} + XY'B'_{\text{in}} + XYB_{\text{in}}$
42. $B = X'Y + X'YB_{\text{in}} + YB_{\text{in}}$

Characteristic Equations:

43. SR flip−flop: $Q_{n+1} = S + R'Q_n$
44. D flip−flop: $Q_{n+1} = D$
45. JK flip−flop: $Q_{n+1} = JQ'_n + KQ_n$
46. T flip−flop: $Q_{n+1} = TQ'_n + T'Q_n$

SOLVED EXAMPLES

1. A correct output is achieved from a master−slave JK flip−flop only if its inputs are stable while the

(a) Clock is LOW
(b) lave is ready to receive the input from master
(c) Master gives a reset signal to slave
(d) Clock is HIGH

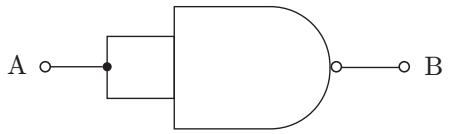
Solution: The clock is high, only the flip−flops are stable when the inputs are stable.

Ans. (d)

2. An inverter circuit can be realized with how many NAND gates?

(a) 1 (b) 2
(c) 3 (d) Not possible

Solution: An inverter or logic NOT gate can be made using standard NAND gate by connecting together **ALL** their inputs to a common input signal, for example,



Two-input NAND gate equivalent

Ans. (a)

3. A modulus-12 ring counter requires a minimum of _____.

(a) 10 flip−flops (b) 12 flip−flops
(c) 8 flip−flops (d) 6 flip−flops

Solution: In the ring counter, n flip-flops are used for the mod-n. 12 flip−flops because ring counter uses states n, so here modulus is 12.

Ans. (b)

4. How is a strobe signal used when serially loading a shift register?

(a) To turn the register set and reset
(b) To control the number of clocks
(c) To determine which output has a HIGH value
(d) To cascade it with other shift register

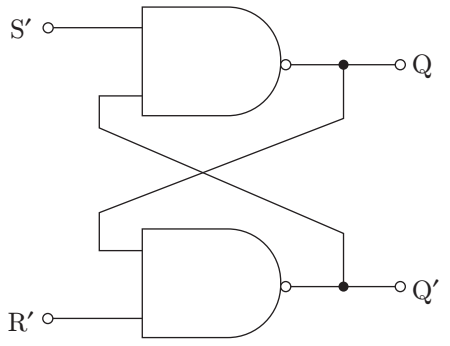
Solution: In digital circuits, a ***shift register*** is a cascade of flip−flops, sharing the same clock each time. This is called clocking or ***strobing*** to the register to set the clocks.

Ans. (b)

5. An active-HIGH SR latch has a 0 on the S input and a 1 on the R input. Now when the R input goes to 0, the latch will be _____.

(a) SET (b) RESET
(c) CLR (d) STROBE

Solution:



The SR (Set-Reset) flip−flop is a simple sequential circuit which consists of two gates connected as shown above. The output of each gate is connected to one of the inputs of the other gate, thus giving positive feedback or cross-coupling. The circuit has two active low inputs NOT S and NOT R and two outputs Q and NOT Q. When there is 0 on the S input and a 1 on the R input, then NOT S is 0 and NOT R is 1, so Q is 0 and NOT Q is 1. Thus, Q is reset to 0 by 0 on NOT R.

Ans. (b)

6. The NAND or NOR gates are referred to as "universal" gates because

(a) they are easier to implement.
(b) they can be used to build all the other types of gates.
(c) they are available in different IC families.
(d) they are available everywhere in the world.

Solution: NAND and NOR gates are universal because all Boolean functions that can be implemented by AND, OR and NOT gates can be implemented by NAND and NOR gates.

Ans. (b)

7. Match the following:

IC Family	GATE
TTL	NOR
CMOS	NOT
ECL	NAND

(a) TTL:NAND; ECL:NOR; CMOS:NOT
(b) TTL:NAND; ECL:NOT; CMOS:NOR
(c) TTL:NOT; ECL:NAND; CMOS:NOR
(d) TTL:NOR; ECL:NOT; CMOS:NAND

Solution: The NAND function is actually the simplest, most natural mode of operation for this TTL design. The basic gates are OR/NOR gates for ECL Construction. Inverters can be constructed using two complementary transistors in a CMOS configuration.

Ans. (a)

8. Which of the following satisfies the commutative law but not the associative law?

(a) NOR (b) OR
(c) XOR (d) NAND

Solution
Let $\uparrow$ signify the NAND operation. Then there exist propositions p, q, r such that

$$p\uparrow (q\uparrow r) \neq (p\uparrow q) \uparrow r$$

This implies NAND is not associative.

$$p \uparrow q = q \uparrow p$$

This implies NAND is commutative.

Ans. (d)

9. Match the following:

IC Family	Characteristics
X. TTL	High fan-out
Y. ECL	Low propagation delay
Z. CMOS	High power dissipation

Code:

	X	Y	Z
(a)	3	2	1
(b)	3	1	2
(c)	2	1	3
(d)	1	2	3

Solution: The power dissipation is dependent on the power supply voltage, frequency, output load, and input rise time. Power dissipation for TTL gates is usually 10 mW per gate. ECL is the fastest logic gate and has least propagation delay of 2 ns. CMOS gates have high fan-outs.

Ans. (a)

10. Which of the following statements is false?

A. The 2's complement of 0 is 0.
B. In 2's complement, the leftmost bit cannot be used to express a quantity.
C. For an n-bit word (2's complement) which includes the sign bit, there are $2n - 1$ positive integers, $2n + 1$ negative integers and one 0 for a total of $2n$ unique states.
D. In 2's complement, the significant information is contained in the 1's of positive numbers and 0's of the negative numbers.

(a) A (b) B (c) C (d) D

Solution: For an n-bit word which includes the sign bit, there are $2^{n-1} - 1$ positive integers, 2^{n-1} negative integers and one 0, for a total of 2^n unique states.

Ans. (c)

11. The dual of any Boolean expression is obtained by

(a) Interchanging POS and SOP terms
(b) Interchanging 0's and 1's
(c) Interchanging Boolean sums and Boolean products and also interchanging 0's and 1's
(d) Interchanging Boolean sums and Boolean products

Solution: The dual of a Boolean expression is obtained by interchanging Boolean sums and products and interchanging 0's and 1's. For example, The dual of $x(y + 0) = x + (y \times 1)$.

Ans. (c)

12. A sequential circuit gives an output depending upon

(a) its present state inputs only.
(b) its present-1 state input only.
(c) both present- and past-state input.
(d) the size of the memory.

Solution: Sequential logic is a logic circuit in which the output depends not only on the present value of its input signals but on their past history. This is in contrast to *combinational logic*, where the output depends only on the present input.

Ans. (c)

13. Synchronization is achieved by a timing device called a _____.

(a) Strobe
(b) Reset
(c) CLK
(d) Master clock generator

Solution: Synchronization is achieved by a timing device called a clock generator that produces a periodic train of clock pulses.

Ans. (d)

14. A complete microcomputer system contains

(a) Microprocessor
(b) Memory
(c) I/O device
(d) All of the above

Solution: A microcomputer contains a microprocessor, ROM and RAM, I/O ports and bus or system of interconnecting wires, housed in a motherboard.

Ans. (d)

15. How many different Boolean functions of degree 4 are there?

(a) 2^4 (b) 2^8
(c) 2^{12} (d) 2^{16}

Solution: Different Boolean functions of degree n is given by 2^{2^n}.

Ans. (d)

16. A Boolean operator $\oplus$ is defined as follows:

$1 \oplus 1 = 1$, $1 \oplus 0 = 0$, $0 \oplus 1 = 0$ and $0 \oplus 0 = 1$
What will be the truth value of the expression $(x \oplus y) \oplus z = x \oplus (y \oplus z)$?

(a) Cannot be defined, as $\oplus$ is defined for two inputs.
(b) Always true.
(c) Always false.
(d) True when any one of the inputs is true.

Solution:

x	y	z	x⊕y	(x⊕y)⊕z	y⊕z	x⊕(y⊕z)
0	0	0	1	0	1	0
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	0	0	1	0
1	0	0	0	1	1	1
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Ans. (b)

17. Let * be defined as $x * y = \overline{x} + y$. Let $z = x*y$, then value of $z*x$ is

(a) $\overline{x} + y$ (b) x (c) 0 (d) 1

Solution: Substituting value of $z = x*y$ in the expression:

$$z * x = (x * y) * x = (\overline{x} + y) * x = (\overline{\overline{x} + y}) + x$$
$$= x \cdot \overline{y} + x$$

Now using absorption law: $a + ab = a$

$$x \cdot \overline{y} + x = x$$

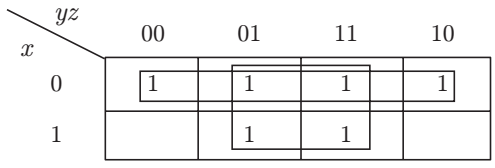
So, result is: $z*x = x$

Ans. (b)

18. Let $f(x,y,z) = \overline{x} + \overline{y}z + xz$ be a switching function. Which one of the following is valid?

(a) $\overline{y}x$ is a prime implicant of f.
(b) xz is a minterm of f.
(c) xz is an implicant of f.
(d) y is a prime implicant of f.

Solution:

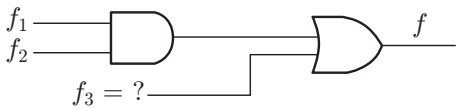


Number of possible joins are 2, so there are two prime implicants, that is, x' and z.

Both the prime implicants are essential. xz is an implicant of f.

Ans. (c)

19. Consider the logic circuit shown in the following figure. The functions f_1, f_2 and f (in canonical sum-of-products form in decimal notation) are



$f_1(w, x, y, z) = \sum (8, 9, 10)$
$f_2(w, x, y, z) = \sum (7, 8, 12, 13, 14, 15)$
$f(w, x, y, z) = \sum (8, 9)$

The function f_3 is

(a) $\sum$ 9, 10 (b) $\sum$ 9
(c) $\sum$ 1, 8, 9 (d) $\sum$ 8, 10, 18

Solution: By the logic circuit, function f is given as

$$f = f_1 f_2 + f_3$$
$$f_1 = \sum (8, 9, 10)$$
$$f_2 = \sum (7, 8, 12, 13, 14, 15)$$

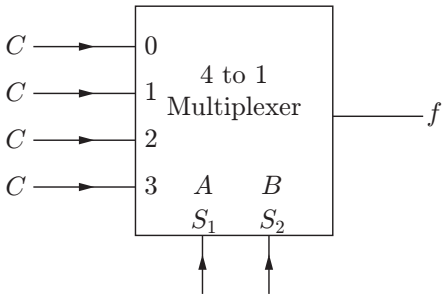
As according to logic, common minterm in f_1 and f_2 is 8, so

$$f_1 f_2 = \sum (8)$$

to get output as $f = \sum (8, 9)$, f_3 can be either $\sum (8, 9)$ or $f_3 = \sum 9$

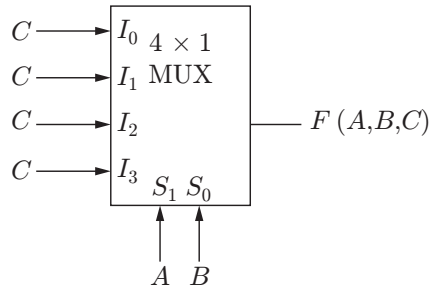
Ans. (b)

20. Consider the circuit shown in the following figure; f implements



(a) $\overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC$ (b) $A + B + C$
(c) $A \oplus B \oplus C$ (d) C

Solution: The circuit for function $f(A, B, C)$ is given as follows



The output of function will be

$$\overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + ABC = \overline{A}C(\overline{B} + B) + AC(\overline{B} + B)$$
$$= \overline{A}C + AC = C$$

Ans. (d)

21. Which of the following functions implements the Karnaugh map shown below?

AB \ CD	00	01	11	10
00	0	0	1	0
01	X	X	1	X
11	0	1	1	0
10	0	1	1	0

(a) $\bar{A}B + CD$ (b) $D(C + A)$

(c) $AD + \bar{A}B$ (d) $(C + D)(\bar{C} + D)(A + B)$

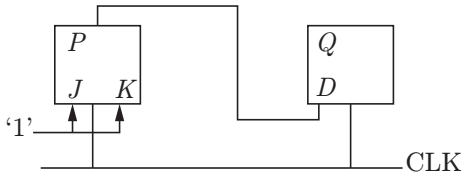
Solution: Karnaugh map of the function is given as follows:

AB \ CD	00	01	11	10
00	0	0	1	0
01	X	X	1	X
11	0	1	1	0
10	0	1	1	0

SOP form for the given function is $f(A, B, C, D) = CD + AD = D(C + A)$

Ans. (b)

22. Refer to the following arrangement of master−slave flip−flops.



It has the initial state of P, Q as 0, 1 (respectively). After the clock cycles, the output state P, Q is (respectively)

(a) 1, 0 (b) 1, 1 (c) 0, 0 (d) 0, 1

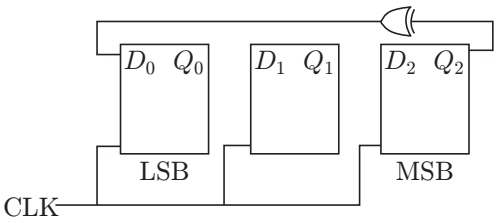
Solution:

	J	**K**	**D**		
	1	**1**	**(P)**	**P**	**Q**
Initially				0	1
	1	1	0	1	0
	1	1	1	0	1

For $J = 1$ and $K = 1$, the output is complement to previous state.

Ans. (a)

23. Consider the circuit given below with initial state $Q_0 = 1$, $Q_1 = Q_2 = 0$. The state of the circuit is given by the value $4Q_2 + 2Q_1 + Q_0$.



Which one of the following is the correct state sequence of the circuit?

(a) 1, 3, 4, 6, 7, 5, 2 (b) 1, 2, 5, 3, 7, 6, 4

(c) 1, 2, 7, 3, 5, 6, 4 (d) 1, 6, 5, 7, 2, 3, 4

Solution:

$D_2(Q_1)$	$D_1(Q_0)$	D_0 $(Q_1 \oplus Q_2)$	Q_2	Q_1	Q_0	$4Q_2 + 2Q_1 + Q_0$
Initially			**0**	**0**	**1**	**1**
0	1	0	0	1	0	2
1	0	1	1	0	1	5
0	1	1	0	1	1	3
1	1	1	1	1	1	7
1	1	0	1	1	0	6
1	0	0	1	0	0	4
0	0	1	0	0	1	1

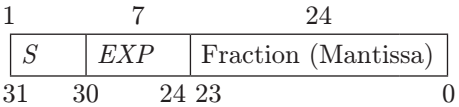
Therefore, the correct state sequence of the circuit is 1, 2, 5, 3, 7, 6, 4. Ans. (b)

24. A 32-bit floating-point number is represented by a 7-bit signed exponent, and a 24-bit fractional mantissa. The base of the scale factor is 16.

(a) The range of the exponent is _____.

(b) The range of the exponent is ____ if the scale factor is represented in excess −64 format.

Solution: The given floating point number format is



Bias + 64

The above floating-point architecture is IBM floating-point architecture.

The number is represented as the following formula

$$(-1)^S \times 0.\text{Mantissa} \times 10^{\text{EXP} - 64}$$

(a) The range of the exponent is the range of the 7-bit signed numbers, which is −64 to +63.

(b) The range of the exponent if the scale factor is represented in excess −64 format is 0 to 127.