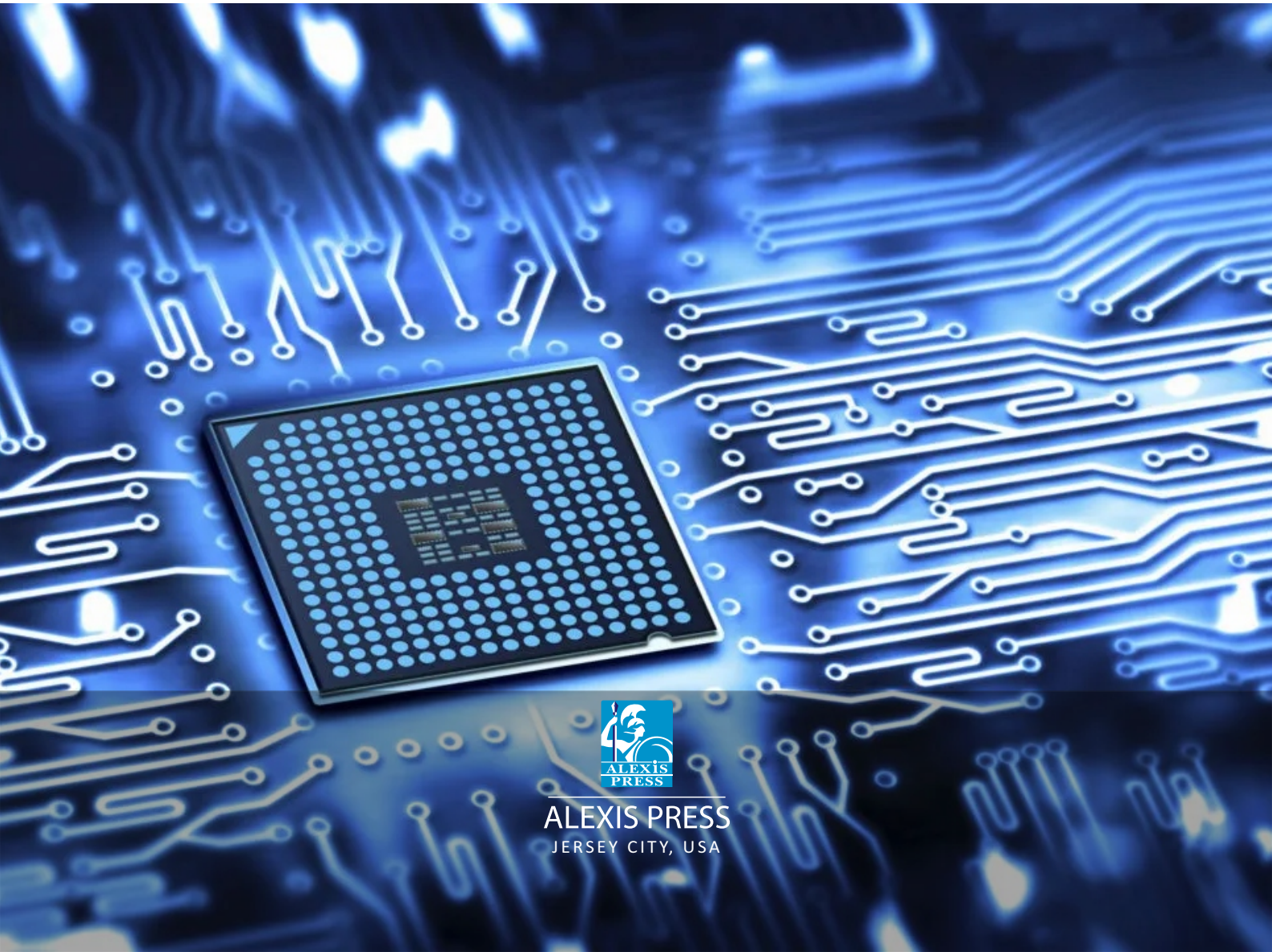
Harsh Shrivastava
Manaswini R

DIGITALEL ECTRONICS



ALEXIS PRESS
JERSEY CITY, USA

DIGITAL ELECTRONICS

DIGITAL ELECTRONICS

Harsh Shrivastava

Manaswini R

ALEXIS PRESS

Published by: Alexis Press, LLC, Jersey City, USA
www.alexispress.us

First Published 2022

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication Data

Includes bibliographical references and index.

Digital Electronics by *Harsh Shrivastava, Manaswini R*

ISBN 978-1-64532-427-0

CONTENTS

CHAPTER 1

NUMBER SYSTEMS

Mr. Harsh Shrivastava, Assistant Professor
Department of Electrical Engineering, Jaipur National University, Jaipur, India
Email Id-ershrivastava@jnujaipur.ac.in

The number system is defined as a system used to represent data collection in mathematical form.A number is a mathematical value that helps to measure or count objects and it also helps in solving mathematical calculations. The study of the number system is important from the point of view to learn about how data are represented before the data is processed in the digital system including the digital computer. In this, we will discuss how different number represents a data collection of a measured quantity. There are some types of number systems in digital computers such as decimal number systems, binary number systems, octal number systems, and hexadecimal number systems. In this article, we are going to learn about the number system in maths, different types, and conversion procedures with both number systems.A number system is a system that expresses different values in numbers. It is the mathematical notation for representing numbers of a given set by consistently using digits or symbols. It provides a unique representation of every number that represents the arithmetic and algebraic structures of the figures. It also permits the operation of arithmetic operations such as addition subtraction, multiplication, and division[1]–[3].

The mathematical expression of any value of any digit in a number can be determined as given below:

1. The digit
2. Its position in the number
3. The base of the number system.

Analogue vs Digital: there are two basic forms of representing the numerical values or data of the Physical quantity of any object in mathematical form. Referred to an analog system, is to represent the numerical value or data of the physical quantity as a continuous range of values between the two expected values. For example voltage across a certain component in an electrical system or circuit may be expressed as 6.5v or 6.49v or 6.4879v. the underlying concept in this mode represents the variation in the numerical data of the physical quantity that is continuous and could have any of the infinite possible values between two points.

On the other hand, referred to as digital, expressed the numerical data of the quantity in steps of discrete values. The numerical values are mostly expressed using binary codes (0 or 1) for example, the output voltage of a digital voltmeter is 10v, 13v, or 20v, and so on. It means digital numbers represent only discrete output data or values of a given measured physical quantity fractional part not included in digital numbers. Analog systems consist of devices that process or operated on various physical quantities represented in analog form. Digital systems consist of devices that process or operated the physical quantities expressed in digital form.

Digital techniques and systems can be relatively much easier to design and have higher accuracy, noise immunity, easy storage of data, programmability, and easy fabrication in the integrated circuit form leading to the availability of more difficult functions in a smaller size. Most physical

quantities- velocity, acceleration, force, position, flowrate, pressure, and temperature are analogs in nature. In a complex system with analog inputs and outputs, analog variables have digitalized with the help of analog to digital converter or reconverted back to analog form at the output using digital to analog converter block.

Number: A number is a mathematical value used for counting, expressing the quantity, ormeasuring or labeling objects. Numbers are used to operating arithmetic calculations. Examples of numbers are natural numbers, whole numbers, rational and irrational numbers, etc. 0 is also a number that represents a null value.

Base: a number base in a number system is defined as the unique or many different symbols and notations to represent a value. For example, the number base 2 represents that there are only two unique notations 0 and 1. The common number base is decimal, whose base is 10.

Types of number systems:
The most common four types of number systems are given below:

1. Decimal number system (base-10)
2. Binary number system(base-2)
3. Octal number system (base-8)
4. Hexadecimal number system(base-16)

The different types of number systems:

1. Decimal number system:

The decimal number system has base 10 it uses ten digits from 0 to 9 to represent a value. In the decimal system, the positions successive to the left of the decimal point represent units, tens, hundreds, thousands, and so on. This system represents a decimal number. The positions show the power of the base (10). The method of writing higher-ordered numbers after 9 represents the writing second digit first (i.e. 1), followed by the other digits, one by one, to obtain the next ten numbers from 10 to 19. The next ten numbers from 20 to 29 are obtained by writing the third digit (i.e, 2) first, followed by one by one from the digit 0 to 9. This method of writing higher ordered numbers continues until we get all possible two-digit numbers combinations and reached 99. Then we begin to write three-digit combination numbers. The first three-digit number uses the lowest two-digit numbers followed by 0 (i.e.100) and the process goes on endlessly.

The positions of different digits mixed in a decimal number, starting from the decimal point, are10^0,10^1,10^2 and so on for the integer parts and the fractional part 10^-1,10^-2,10^-3 and so on.

For example: (1*10^3)+(5*10^2)+(4*10^1)+(2*10^0)=1000+500+40+2

=1542(for integer part)

= 2*10^-1+5*10^-2+6*10^-3=0.2+0.06+0.005=0.265 (for the fractional part)

Decimal number = 1542.265

2. Binary number system:

The number with base 2 is known as the binary number system. The binary number system uses only two notations i.e. 0 and 1 the usual base-2 is a radix of 2. The binary number is used in the logical operation of computers. the procedure for writing high-ordered terms/ numbers after 1 is the same process as the one explained in the case of the decimal number system ., the first 16 binary numbers in the binary system would be 0,1,10, 11,100, 101, 110, 111, 1000, 1010, 1011, 1100, 1110 and 1111. The next binary number after this is 10000, which is the lower binary number with 5 digits.

The backbone of any digital computer system is logic operations. Although computer systems involve arithmetic operations too for solving complex problems. The introduction of logic operation given by George Boole laid the foundation for the modern digital computer decreased the mathematics of logic to binary notations such as 0 and 1.The advantage of the binary number system was that all types of data could be efficiently represented in terms of 0s and 1s basic electronics used a binary number system for the implementation of hardware for could be efficiently or conveniently operated on two distinctly different modes. A bipolar transistor may be operated either cut–off or in saturation very efficiently.

For example, 110101 is a binary number.

3. Octal number system:

The octal number system has a base is 8. It uses number notation from 0 to 7 to represent the number. Octal numbers are used in computer applications. The octal number has a radix of 8 and it used eight distinct digits. All higher-ordered numbers are represented as a combination of these as the same pattern followed in the case of decimal and binary number systems. The independent digits of the octal number system are 0, 1,2,3,4,5,6,7. The next number follows 7 i.e. would 10, 11, 12,13,14,15,16,17,20, and 21. The position of values for different digits in the octal number system is 8^0,8^1,8^2,8^3 and so on for the integer parts and the fractional parts 8^-1,8^-2,8^-3 and so on.

For example (215) with base 8.

2*8^2+1*8^1+5*8^0

=2*64+1*8+5*1

= 128+8+5

=141 with base 10.

4. Hexadecimal number system:

It has the base 16. it uses the notations from 0 to 9 and the numbers are represented using the alphabet from A, B, C, D, E, and F. the hexadecimal number system has radix-16 and the independent's radix of this system are 0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, and F[4]–[6].

The hexadecimal number system includes a convenient way of expressing large binary numbers stored and operated inside the computer. One such example is representing an address of different memory locations the decimal number is not used in digital computers and the binary code notations here appear too cumbersome and inconvenient to handle.

For example, 412F is a hexadecimal number with a base of 16.

Number system conversion:

Binary-to-Decimal conversion: the decimal equivalent to the binary number $(1001.01021)_2$ is determined as follows:

Integer part = 1001

The decimal equivalent = 1*2^3+0*2^2+0*2^1+1*2^0 = 8+0+0+1 = 9

For fractional part = 0*2^-1+1*2^-2+0*2^-3+1*2^-4 = 0+0.25+0+0.0625 = 0.3125
The decimal equivalent of a binary number $(1001.01010)_2 = (9.3125)_{10}$
Octal to decimal conversion: the decimal equivalent of the octal number $(137.21)_8$is determined as follows:
The integer part = 137
The decimal equivalent = 1*8^2+3*8^1+7*8^0 = 64+24+7 = 9
Fractional part = .21 = 2*8^-1+1*8^-2 = 0.265
$(137.21)_8 = (95.265)_{10}$
Hexadecimal to decimal conversion: $(1E0.2A)_{16} = (\quad)_{10}$
Integer part = 1E0 = 1*16^2+14*16^1+ 0*16^0 = 256+224+0 = 480
Fractional part = 2*16^-1+ 10*16^-2 = 0.164
$(1E0.2A)_{16} = (480.164)_{10}$

Decimal to Binary conversion: Convert the decimal number 112 into a binary number (Table 1.1).

Table 1.1:Decimal to binary conversion

Division	**The Remainder (R)**
112 / 2 = 56	0
56 / 2 = 28	0
28 / 2 = 14	0
14 / 2 = 7	0
7 / 2 = 3	1
3 / 2 = 1	1
1 / 2 = 0	1

Now, write the remainder from Downward to up (in reverse order), that is 1110000 which is an equivalent binary number of decimal integers 112.

$(112)_{10} = (1110000)_2$

For the fractional part: Convert the decimal fractional number 0.8125 into a binary number.write these resultant integer parts from upward to downward, this will be 0.11010 which is the equivalent binary fractional number of decimal fractional 0.8125.

$(0.81215)_{10} = (0.11010)_2$

DECIMAL TO OCTAL CONVERSION (TABLE 1.2):

Table 1.2: Conversion $(7562)_{10}$ to octal:

Division by 8	Quotient (integer)	Remainder (decimal)	Remainder (octal)	Digit #
7562/8	945	2	2	0
945/8	118	1	1	1
118/8	14	6	6	2
14/8	1	6	6	3
1/8	0	1	1	4

It counts the remainder in octal from downward to upward

$(7562)_{10} = (16612)_8$

For the fractional part: Convert the decimal fractional number 0.140869140625 into an octal number (Table 1.3).

Table1.3: Conversion of decimal fraction into octal

Multiplication	**Resultant integer part**
0.140869140625 x 8=0.12695313	1
0.12695313 x 8=0.01562504	1
0.01562504 x 8=0.12500032	0
0.12500032 x 8=0.00000256	1
0.00000256 x 8=0.000020544	0

This will be approximately 0.11010 which is an equivalent octal fractional number of decimal fractional 0.140869140625.

DECIMAL TO HEXADECIMAL:

For the integer part: Convert 7562_{10} to hexadecimal (Table 1.4 and Table 1.5)

Table1.4: Decimal to hexadecimal conversion

Division by 16	Quotient (integer)	Remainder (decimal)	Remainder (hex)	Digit
7562/16	472	10	A	0

472/16	29	8	8	1
29/16	1	13	D	2
1/16	0	1	1	3

So counting the hexadecimal remainder from bottom to top

$(7562)_{10} = (1D8A)_{16}$

Table 1.5: Conversion of decimal fraction into Hexa decimal

	Fractional part	**Real part**
0.65625 x 16	0.5	10 = A
0.5 x 16	0.0	8

The fractional part becomes 0 here, we completed the conversion. If we count the real part from top to bottom, we will obtain the number in hexadecimal.

$(0.65625)_{10} = (0.A8)_{16}$

Floating point numbers: floating point notations are used efficiently to express both large as well as small fractional numbers. This makes the procedure of arithmetic operations on these values relatively much easy. Floating point representation increases the range of the numbers, from smaller to higher, that could be expressed using given values of digits. Floating points numbers can be represented in the form

$N = m \times b^e$

Where m is the fractional part, known as significant or mantissa, e is the integer part called exponent, and b is the base of the number.

Decimal number system

$N = m \times 10^e$

Binary number system

$N = m \times 2^e$

Octal number system

$N = m \times 8^e$

Hexa decimal number system

$N = m \times 16^e$

For example:

Decimal numbers 0.0003574 and 3754 will represent floating point notations as 3.754×10^{-4} and 3.754×10^3.

Hexadecimal number 257.ABF representas $2.57ABF \times 16^2$.

Range of numbers and precision: the range of numbers that could be expressed in any machine depends on the number of bits in the exponent, while the fractional accuracy and precision are determined by the number of bits in the mantissa. The number higher in the bits in the exponent, the larger the range of the number that could be expressed.

The four Axioms: the conversion of a given number in one number system to it is equivalent in other systems have been discussed at length in the preceding sections. The methodology has been illustrated to solve examples. The complete methodology can be summarized as four axioms or principles, which, if understood properly, could make it possible to solve any complex problem.

Binary addition: binary addition is much similar to the addition of decimal numbers including the addition of 10 alternative values, it carries two values only.

For example, as we add two decimal numbers 9 +5 the answer is 14, so we can write the result by two digits 1 and 4. Similar binary addition is done (Table 1.6).

Table1.6: Binary addition table

A	B	A+B	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Binary subtraction: In subtraction, this is the primary method. In this method, ensure that the subtracting number must be from a larger number to a smaller one (Table 1.7).

Table 1.7: Binary subtraction table

A	B	A-B	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

1's and 2's complement of Binary number (Table 1.8):

Table 1.8: 1's and 2's complement of the binary number.

Parameters	1's Complement Representation	2's Complement Representation
Process of Generation	To generate a 1's complement for any given binary number, you only need to invert that number.	To generate a 2's complement for any given binary number, you need to invert it. Then you need to add 1 to the LSB (Least Significant Bit) of the generated result.
Example	For a binary number like 110010, the 1's complement would be 001101.	For a binary number like 110010, the 2's complement would be 001110.
Logic Gates Used	It is a very simple type of implementation that makes use of the NOT gate for every bit of input.	It makes use of the BOT gate for every bit of input, along with a full adder.
Number Representation	You can use the 1s in case of a signed binary representation. Still, you cannot use it as an ambiguous representation in the case of the number 0.	You can use the 2's in case of a signed binary representation. It is also the most suitable in the form of an unambiguous representation of all the available numbers.
K-bits Register	In the case of a k-bits register, it can store the lowest negative number as -(2(k-1)-1) and the largest positive number as (2(k-1)-1).	In the case of a k-bits register, it can store the lowest negative number as -(2(k-1)) and the largest positive number as (2(k-1)-1).
Representation of 0	The number 0 has two major representations- one is +0 (for instance, 0 0000 in a five-bit register), and the second one is -0 (for instance, 1 1111 in a five-bit register).	The number -0 has a single representation for both +0 and -0 (for instance, 0 0000 in a five-bit register). Here, we always consider the zero to be positive.
Sign Extension	We need to use the sign extension for the conversion of any signed integer from a given sign to another one.	In this case, too, we need to use the sign extension. It converts any signed integer from a given sign to another one.
End-Around-Carry-Bit	In the case of a 1's complement arithmetic operation, there occurs an addition of the end-around-carry-bit. We add it to the result's LSB.	In the case of a 2's complement arithmetic operation, no addition of end-around-carry-bit occurs. It ignores this addition.
Ease of Operation	The complement arithmetic operations of 1's type are not easier than the 2's type because they always need to add end-around-carry-bit.	The complement arithmetic operations of the 2's type are much easier to operate than the 1's type because they lack the addition of an end-around carry-bit.

Signed binary number: signed binary number is much the same as the binary number system which includes the negative number. It requires an arithmetic sign. The MSB (Most Significant Bit) represents the sign bit.

It the sign bit of the binary number is zero, the signed number is positive, otherwise the signed number is negative (Table 1.9)[7]–[9].

Table 1.9: Signed binary number:

Binary	Decimal	Signed binary
NA	-3	100
NA	-2	101
NA	-1	110
0	0	000
001	1	001
010	2	010
011	3	011

Unsigned binary number: unsigned binary numbers include positive numbers. It does not require any arithmetic sign. an m-bit unsigned binary number expressed all numbers in the range 0 to 2_m-1.

A number system is a method of representing the value or quantity of measured physical objects in terms of numbers. Also, we study different types of the number system we used in the logical operation of a digital computer system such as – binary, decimal, octal, and hexadecimal. We study about conversion of one form of the number system to another number system and the rules, and laws of the number system[10].

REFERENCES

[1] D. Odic and A. Starr, "An Introduction to the Approximate Number System," *Child Dev. Perspect.*, 2018, doi: 10.1111/cdep.12288.

[2] M. V. Valueva, N. N. Nagornov, P. A. Lyakhov, G. V. Valuev, and N. I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Math. Comput. Simul.*, 2020, doi: 10.1016/j.matcom.2020.04.031.

[3] M. Inglis and C. Gilmore, "Indexing the approximate number system," *Acta Psychol. (Amst).*, 2014, doi: 10.1016/j.actpsy.2013.11.009.

[4] A. Cochrane, L. Cui, E. M. Hubbard, and C. S. Green, "'Approximate number system' training: A perceptual learning approach," *Attention, Perception, Psychophys.*, 2019, doi: 10.3758/s13414-018-01636-w.

[5] H. Mei, Z. Gao, Z. Guo, M. Zhao, and J. Yang, "Storage mechanism optimization in blockchain system based on residual number system," *IEEE Access*, 2019, doi: 10.1109/ACCESS.2019.2934092.

[6] M. Corliss, T. Brown, T. A. Hurly, S. D. Healy, and M. C. Tello-Ramos, "Estimating on the fly: The approximate number system in rufous hummingbirds (Selasphorus rufus)," *Learn. Behav.*, 2021, doi: 10.3758/s13420-020-00448-z.

[7] O. Borysenko, S. Matsenko, and V. Bobrovs, "Binomial number system," *Appl. Sci.*, 2021, doi: 10.3390/app112311110.

[8] N. K. DeWind, G. K. Adams, M. L. Platt, and E. M. Brannon, "Modeling the approximate number system to quantify the contribution of visual stimulus features," *Cognition*, 2015, doi: 10.1016/j.cognition.2015.05.016.

[9] D. Sasanguie, E. Defever, B. Maertens, and B. Reynvoet, "The approximate number system is not predictive for symbolic number processing in kindergarteners," *Q. J. Exp. Psychol.*, 2014, doi: 10.1080/17470218.2013.803581.

[10] S. A. Alam, J. Garland, and D. Gregg, "Low-precision Logarithmic Number Systems: Beyond Base-2," *ACM Trans. Archit. Code Optim.*, 2021, doi: 10.1145/3461699.

CHAPTER 2

BOOLEAN ALGEBRA

Mr. M.Sashilal, Associate Professor
Department of Electrical Engineering, Jaipur National University, Jaipur, India
Email Id-msashilal@jnujaipur.ac.in

Boolean algebra is the type of algebra in mathematical logic in which the variable's values are the truth values, true and false**,** and denoted by binary expression code 1 for high and 0 for low respective. It is used to analyze and solve digital circuits or digital gates. It is also called Binary Algebra, and logical Algebra. It has been fundamental in the development of digital electronics and is provided for in all modern programming languages. It is also used in statistics and digital electronics flip-flops. The most important mathematical logical operations performed in Boolean algebra are conjunction (∧), disjunction (∨), and negation (¬). Hence, this algebra is very different from elementary algebra in maths where the values of variables are numerical and arithmetic operations like addition, and subtraction is been performed for solving the given problem[1]–[3].

Boolean algebra was first introduced by George Boole in his," The Mathematical Analysis of Logic (1847)" and set forth more fully in his An Investigation of the Laws of Thought (1854). According to Huntington, the term "Boolean algebra" was first suggested by Sheffer in 1913, although Charles Sanders Peirce gave the title "A Boolean Algebra with One Constant" to the first chapter of his "The Simplest Mathematics" in 1880. Boolean algebra has been fundamental in the development of digital electronics and is provided for in all modern programming languages. It is also used in set theory and statistics.

Laws of Boolean algebra: Boolean algebra has 6 types of laws.

1. Commutative law: (A.B = B.A), (A + B = B + A)
2. Associative law: (A.B).C = A.(B.C), A+(B+C) = (A+B) + C
3. Distributive law: A.(B+C) = (A.B)+(A.C), (A+B).C = (A.C)+(B.C)
4. AND Law: A.0 = 0, A.1 = A, A.A = A
5. OR Law: A+0 = A, A+1 = A, A+A =A

***Boolean algebra Terminology**:*

Let's study the important terminologies used in Boolean algebra.

Boolean algebra: Boolean algebra is the branch of algebra that deals with logical operations and binary code variables such as statements TRUE or FALSE, and uses 1 for high or 0 for low.

Boolean Variables: A Boolean variable is defined as a variable or a symbol, generally an alphabet that is used to express the logical operation quantities such as 0 or 1.

Boolean Function: A Boolean function that is used binary code variables, logical operators, constants such as 0 and 1, equal to the operator, and the parenthesis symbols is known as a Boolean function.

Literal: A literal may be defined as a variable or a complement of a variable is called Literal.

Complement: The complement is defined as the inverse of a variable, which is expressed by a bar over the variable.

Truth Table: The truth table is a table that includes all the possible values of logical operation variables and the combination of the different logical operation variables. It converts the Boolean algebra operational equations into a truth table. The number of rows in a truth table should be equal to 2^n, where "n" is the number of variables in the equation. For example, if a Boolean equation has 4 variables, then the number of rows in the truth table is 16. (i.e.,) $2^4 = 16$, as shown in Table 2.1.

Table 2.1: Boolean algebra operation

A	B	A ∧ B	A ∨ B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Boolean algebra rules: the most important rules used in Boolean algebra are expressed below;

1. The variables used in Boolean algebra can have only two values. Binary code used 1 for high and 0 for low.
2. The complement of a variable in Boolean algebra is expressed by an overbar.
3. OR operation of the variables is expressed by plus (+) sign between the variables.

For example, or operation of the variables A, B, and C is expressed as A+B+C.

4. The logical AND operation of two or more variables is represented by a dot (.) between the variables. Such as A.B.C

Boolean algebra operations: following are the basic operation performed by Boolean algebra (Table 2.2);

1. Conjunction or AND operation
2. Disconjunction or OR operation
3. Negation or NOT operation

Table 2.2: Boolean algebra operation

Operator	**Symbol**	**Precedence**
NOT	‘ (or) ¬	Highest
AND	. (or) ∧	Middle
OR	+ (or) ∨	Lowest

1. A conjunction B or A AND B, performed A∧B =True if A = B = True or else A∧B = False.
2. A disjunction B or A OR B, performed A∨B = False if A = B = False, else A∨B = True.
3. Negation A or ¬A satisfies¬A = False if A = True and ¬A = True if A = False.

Boolean expression: A Logical operation or statement is expressed as the result of a Boolean value, either true or false, which is known as a Boolean expression. Sometimes, we used to express statements such as 'yes' for 'True' and 'No' for 'False'. Also, we expressed that 1 and 0 are used by digital circuits for True and False respectively. Boolean expressions are the results or statements of the logical operators, which are used in the logical operation. I.e. AND, OR, XOR, and NOT. The statement of operation X AND Y =True, is a Boolean expression.

Boolean algebra theorems: The two most important Boolean algebra theorems are De Morgan's First law and De Morgan's second law. These theorems are used to modify the Boolean expression of logical operation. These theorems are used to help to modify the given Boolean expression in the simplified form. These two De Morgan's laws are used to modify the expression from one form to another form. Now, let us discuss the theorems in detail.

De Morgan's First Law: states that "the complement of the product of the two variables A and B is equal to the sum of their complements of variables", as shown in Table 2.3[4]–[6].

i.e. (A.B)' = A'+B'.

Table 2.3: Morgan's first law

A	B	A'	B'	(A.B)'	A'+B'
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Hence, (A.B)' = A' + B' proved.

De morgan's second law: States, "that the complement of the sum of variables o is equal to the product of their complements of a variable" (Table 2.4).

i.e. (A + B)' = A'.B'

Table 2.4: de morgan's second law

A	B	A'	B'	(A+B)'	A'.B'
0	0	1	1	1	1
0	1	1	0	0	0

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

Hence, (A +B)' = A'.B'.

Logic gates: logic gates are the most basic building block of any digital computing device. Logic gates are electronic circuits that can be used to implement basic logic operations. Laws of Boolean algebra could be used to implement binary code variables and solve logic expressions. These are implemented in a digital computer system with the help of electronic circuits known as logic gates.

OR gate: an OR gate is a logic circuit that uses two or more inputs and gives one output.An OR gate is performing the OR operation on two or more two logic variables. If two logic variables. A and B perform the OR operation, Then the output of the operation represented by Y is written as Y = A+B. It is read Y is equal to A OR B, not A plus B. when both inputs of the gate are low (i.e. 0), then the output of the OR gate is low (Table 2.5 and Table 2.6). If anyone of the inputs is high (i.e. 1), then the output of the gate is high (Figure 2.1).

Y= A+B

Figure 2.1: OR gate

Table 2.5: Truth table of OR GATE

A	B	Y= A+B
0	0	0
0	1	1
1	0	1
1	1	1

If we use three variables A, B, and C, then the output Y= A+B+C

Table 2.6: Truth of OR GATE

A		B	C	Y= A+B+C
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

AND gate: an AND gate is a logic circuit using two or more variables as input and giving one output (Table 2.7). The output of an AND gate operation is high when both inputs are high. If anyone of the inputs is low then the output of the operation is low (Figure 2.2).

Y = A.B

Figure 2.2: AND gate

Table 2.7: Truth Table of AND Gate

A	B	Y= A.B
0	0	0
0	1	0
1	0	0
1	1	1

NOT gate: NOT gate is a single input and single output logical circuit (Table 2.8). The output of the gate is the complement of the input. If the input is low the output is high and if the input is high then the output of the operation is low i.e. Y = complement of A = A'.

Table 2.8: the truth of NOT gate

A	Y = A'
0	1
1	0

XOR Gate: Exclusive – OR gate is two input and one output logical circuit. XOR is a digital logic gate that gives a true (1 or HIGH) output when the number of true inputs is odd an XOR represents the inequality function, i.e., the output is true if the inputs are not alike otherwise the output is false (Table 2.9).

i.e. $Y = A \oplus B$
$Y = (A' B + AB')$

Table 2.9: XOR gate

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

NAND Gate: NAND is the combination of NOT and AND Gate an AND gate followed by a NOT logic circuit (Table 2.10). The output of the AND gate is the product of variable A.B and it is the input of the NOT gate. The output of the NOT gate is the complement of the product (A.B).

Y = Complement of the product (A.B) = (A.B)'

Table 2.10: Truth Table of NAND gate

Input		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Nor gate: NOR gate is also a logical circuit. It is the combination of NOT – OR gate. When both inputs of the gate are low, the output of this logic is high, otherwise, the output is low (Table 2.11).

Y= (A+B)'

Table 2.11: Truth NOR gate

A	B	Y = (A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

XNOR Gate: XNOR is a logical circuit that is the complement of the output of the XOR gate (Figure 2.3). It is the combination of XOR and NOT gate (Table 2.12). $Y = (A \oplus B)'$

$Y = (A' B + AB')'$

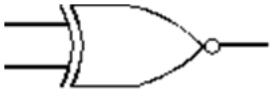


Figure 2.3: XNOR gate

Table 2.12: Truth Table of XNOR gate

A	B	Y= (A ⊕ B)'
0	0	1
0	1	0
1	0	0
1	1	1

MINTERM AND MAXTERM: THERE ARE TWO METHODS IN WHICH WE COULD PLACE THE BOOLEAN FUNCTION. THE METHODS ARE THE MINTERM CANONICAL FORM AND MAXTERM CANONICAL FORM.

Literal: A Literal mark the Boolean variables with their complements. For as X is a boolean variable and the complements of X are ~X or X', That is also called the literals.

Minterm: The product of all literals, either with complement or without complement, is known as minterm.

Karnaugh map method (K-map method): The K-map is a systematic method of solving Boolean expressions. With the help of the K-map method, we can find the easier POS and SOP expression, which is known as the minimum expression. The K-map provides a cookbook for simplification.

Just like the truth table, a K-map contains all the possible values of input variables and their corresponding output values. However, in K-map, the values are stored in cells of the array. In each cell, a binary value of each input variable is stored[7]–[10].

The K-map method is used for representing containing two or more variables. For a higher number of variables, there is another method used for simplification called the Quine-McClusky method. In the K-map, the number of cells is the same as the total number of variable input combinations.

For example, if the number of variables is two, the number of cells is 2^2=4, and if the number of variables is three, the number of cells is 2^3. The K-map takes the SOP and POS forms. The K-map grid is filled using 0's and 1's. The K-map is solved by making groups.

In this chapter, we study Boolean algebra, the laws, and the operations used in Boolean algebra. We also learn about different types of logic gates and how these are working in logical operations. At last, we study the introduction Karnaugh map (K-map method).

REFERENCES

[1] T. Flaminio, L. Godo, and H. Hosni, "Boolean algebras of conditionals, probability and logic," *Artif. Intell.*, 2020, doi: 10.1016/j.artint.2020.103347.

[2] S. Garti and S. Shelah, "Depth+ and length+ of boolean algebras," *Houst. J. Math.*, 2019.

[3] E. M. Jiménez-Hernández, H. Oktaba, F. Díaz-Barriga, and M. Piattini, "Using web-based gamified software to learn Boolean algebra simplification in a blended learning setting," *Comput. Appl. Eng. Educ.*, 2020, doi: 10.1002/cae.22335.

[4] A. Roslanowski and S. Shelah, "More on cardinal invariants of Boolean algebras," *Ann. Pure Appl. Log.*, 2000, doi: 10.1016/S0168-0072(98)00066-9.

[5] S. Geschke and S. Shelah, "Some notes concerning the homogeneity of Boolean algebras and Boolean spaces," *Topol. Appl.*, 2003, doi: 10.1016/S0166-8641(03)00103-2.

[6] S. Shelah, "Special subsets ofcf(μ) μ Boolean algebras and Maharam measure algebras," *Topol. Appl.*, 1999, doi: 10.1016/s0166-8641(99)00138-8.

[7] S. Shelah, "Cellularity of free products of Boolean algebras (or topologies)," *Fundam. Math.*, 2001, doi: 10.4064/fm-166-1-2-153-208.

[8] S. Bonzio and A. Loi, "Probability over Płonka sums of Boolean algebras: States, metrics and topology," *Int. J. Approx. Reason.*, 2021, doi: 10.1016/j.ijar.2021.05.003.

[9] S. Garti and S. Shela, "Depth of boolean algebras," *Notre Dame J. Form. Log.*, 2011, doi: 10.1215/00294527-1435474.

[10] S. Shelah, "The depth of ultraproducts of Boolean algebras," *Algebr. Universalis*, 2005, doi: 10.1007/s00012-005-1925-1.

CHAPTER 3

COMBINATIONAL FUNCTION MINIMIZATION – K MAP, BOOLEAN IDENTITIES

Mr. Vivek Jain, Associate Professor
Department of Electrical Engineering, Jaipur National University, Jaipur, India
Email Id-vivekkumar@jnujaipur.ac.in

In this chapter, we will study combinational function minimization procedures. The laws and theorems of Boolean algebra studied in the previous chapter for simplifying or more accurately with precision minimizing complex Boolean logical operation expressions. The prime objective of the minimization process is to acquire an expression that has the least number of terms. Gaining an expression with the least possible number of literals is generally the minor objective. If there is more than one possible solution with a similar number of terms, the one having the least number of literals is special.

The methods to be discussed comprise (a) the Quine–McCluskey tabular method; (b) the Karnaugh map method.

Before we move on to argue this minimization method in detail, it would be applicable briefly to describe sum-of-products and product-of-sums Boolean expressions. The given Boolean look will be in also the two forms, and the objective will be to discover a minimized appearance in the similar or the former form.

Sum-of-Products Boolean Expressions: A sum-of-products expression holds the sum of dissimilar terms, with individual terms being each a single literal or a product of more than one literal. It can be found from the truth table straight by allowing for those input arrangements that produce a logic '1' at the output. Each such input arrangement produces a term. Different terms are given by the product of the consistent literals. The sum of all terms contributes to the expression[1]–[3].

$$Y= A'.B'.C' + A'.B.C+A.B'.C + A.B.C'$$

Considering the initial term, the output is '1' when A = 0, B = 0, and C = 0. This is likely only when A, B, and C are ended. Also, for the second term, the output is '1' only when B, C, and An area AND. Additional terms can be explained similarly. A sum-of-products expression is also known as a minterm expression (Table 3.1).

Table 3.1: Sum-of-Products Boolean Expressions

A	B	C	Y
0	0	0	1
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	0

Product-of-Sums Expressions: A product-of-sums expression comprises the product of dissimilar terms, with each term being a single literal or a sum of additional than one literal. It can be found from the truth table by seeing those input combinations that yield a logic '0' at the output. Each such input combination provides a term, and the product of all such terms provides the expression. Unlike terms are gotten by taking the sum of the resultant literals. Here, '0' and '1' correspondingly mean the uncomplemented and complemented variables, unlike sum-of-products expressions where '0' and '1' respectively mean complemented and uncomplemented variables.

To illuminate this additional, consider once again the truth table given above. As respectively term in the case of the product-of-sums expression is going to be the sum of literals, this involves that it is working to be executed using an OR operation. Now, an OR gate produces a logic '0' only after all its inputs are in the logic '0' state, which means that the first term resultant to the second row of the truth table will be A+B+C'. The product-of-sum Boolean expression for this truth table is given by (A+B+C'). (A+B'C). (A+B+C). (A'+B'+C').

Transforming the expected product-of-sums expression into an equal sum-of-products expression is a direct process. Multiplying out the particular expression and carrying out the understandable explanation delivers the equivalent sum-of-products expression:

(A+B +C').(A+B'+C).(A'+B+C).(A'+B'+C') =
(A.A+A.B'+A.C+B.A+B.B'+B.C+C'.A+C'.B'+C'.C).(A'.A'+A'.B'+A'C'+B.A'+B.B'+B.C'+C.A'+C.B'+C.C')

(A+B.C+B'.C').(A'+B.C'+C.B') = A.B.C'+A.B'.C+A'.B.C+A'.B'.C'

A given sum-of-products expression can be changed into an equivalent product-of-sums expression by (a) taking the double of the given expression, (b) multiplying available dissimilar terms to become the sum-of-products form, (c) eliminating redundancy and (d) taking a dual to get the corresponding product-of-sums expression. As an illustration, let us discover the corresponding product-of-sums expression of the sum-of-products expression:

A.B +A'.B'

The dual of the given expression = (A+B).(A'+B');

(A+B).(A'+B') = A.A'+A.B'+B.A'+B.B'= 0+A.B'+B.A'+0 =A.B'+A'.B

The dual of (A.B'+A'.B) = (A+B').(A'+B)

Theefore, A.B + A'.B' = (A+B').(A'+B)

Expanded Forms of Boolean Expressions: Extended sum-of-products and product-of-sums forms of Boolean expressions are valuable not only in considering these expressions but also in the bid of minimization techniques like the Quine–McCluskey tabular method and the Karnaugh

mapping technique for simplifying specified Boolean expressions. The extended form, sum-of-products or product-of-sums, is found by comprising all likely combinations of disappeared variables. As a design, deliberate the succeeding sum-of-products expression:

A.B' + B.C' +A.B.C' +A'.C

This one is a three-variable expression. Extended versions of dissimilar minterms can be written as follows:

i. A.B' = A.B'(C +C') = A.B'.C +A.B'.C'
ii. B.C' = B.C'(A+A') = B.C'.A+B.C'.A
iii. A.B.C' is a comprehensive term and has no lost variable.
iv. A'.C = A'.C.(B +B') = A'.C.B +A'.C.B'.

The delayed sum-of-products expression is therefore given by

A.B'.C+A.B'.C'+A.B.C'+A'.B.C'+A.B.C'+A'.B.C+A'.B'.C=
A.B'.C+A.B'.C'+A.B.C'+A'.B.C'+A'.B.C+A'.B'.C

As an additional illustration, study the product-of-sums expression

(A'+B).(A'+B+C'+D')

This is a four-variable expression with A, B, C, and D being the four variables. A+B in this case extends to (A'+B+C+D).(A'+B+C+D').(A'+B+C'+D).(A'+B+C'+D').

The extended product-of-sums expression is then assumed by

(A'+B+C+D).(A'+B+C+D').(A'+B+C'+D).(A'+B+C'+D').(A'+B+C'+D')

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

Canonical Form of Boolean Expressions: An extended form of a Boolean expression, where an individual term comprises all Boolean variables in their true or complemented form, is similarly known as the canonical form of the expression. As an illustration, f (A.B. C) = A'.B'.C' + A'.B'.C + A.B.C is a Boolean function of three variables stated in canonical form. This function's later explanation decreases to A'.B' +A.B.C and misses its canonical form.

Quine–McCluskey Tabular Method: The Quine–McCluskey tabular method of generalization is founded on the complementation theorem, which says that

$$X.Y + X.Y' = X$$

Where X signifies either a variable or a term or an expression and Y is variable. This theorem indicates that, if a Boolean expression comprises two terms that vary only in one variable, then they can be joint together and substituted with a term that is lesser by one literal. A similar process is functional for the other sets of terms somewhere such a decrease is possible. All these terms condensed by one literal are further studied to see if they can be condensed more. The process remains till the terms become complicated. The complex terms are called prime implicants. An optimal set of prime implicants that can justification for all the original terms then creates the minimized expression. The system can be functional equally well for minimizing sum-of-products and product-of-sum expressions and is mainly useful for Boolean functions needing more than six variables as it can be computerized and run on a computer. On the other

hand, the Karnaugh mapping method, to be discussed later, is a graphical method and develops very bulky when the number of variables exceeds six. The step-by-step process for the application of the tabular method for minimizing Boolean expressions, both sum-of-products, and product-of-sums, is defined as follows: 1. The Boolean expression to be basic is extended if it is not in extended form. 2. Dissimilar terms in the expression are separated into sets depending on the number of 1s they have. True and complemented variables in a sum-of-products expression mean '1' and '0' respectively[4]–[6].

Sum of Product (SOP): A canonical sum of products is a boolean expression that completely contains minterms. The Boolean function F is definite on two variables X and Y. The X and Y are the inputs of the boolean Expression F whose output is true when any one of the inputs is set to true. The truth table for Boolean expression F is as follows (Table 3.2 and Table 3.3):

Table 3.2: Sum of Product

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

Table 3.3: Sum of Product

X	Y	F	Minterm value
0	0	0	X'Y'
0	1	1	X'Y
1	0	1	XY'
1	1	1	XY

Convert SOP to shorthand notation: The procedure of exchanging SOP form for shorthand notation is similar to the procedure of finding shorthand notation aimed at minterms. Here are the succeeding stages to find the shorthand notation of the assumed SOP expression.

1. Write the specified SOP expression.
2. Find the shorthand notation of all the minterms.
3. Substitute the minterms with their shorthand notations in the specified expression.

Example: F = A'B + AB' + AB

We determine the shorthand notations of the minterms A'B, AB', and AB.

$A'B = (01)_2 = x_1$

$AB' = (10)_2 = x_2$

AB = (11)2 = x3

F = x1+ x2+ x3

F=∑(1 2 3)

F = x1+ x2+ x3

F = 01+10+11

F = x'y+xy'+ xy

Product of sum (POS):A canonical product of sum is a boolean expression that contains max terms. The Boolean function F is well-defined on two variables A and B. The A and B of the boolean function F whose output is true when only one of the inputs is set to true. The truth table is the inputs for Boolean expression F is given below (Table 3.4 and Table 3.5):

Table 3.4: Product of sum

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Table 3.5: Product of sum

A	B	F	MINTERM
0	0	0	A'+B'
0	1	1	A'+B
1	0	1	A+B'
1	1	1	A+B

F = (A'+B').(A+B)

A'+B' = $(00)_2 = M_0$

A+B = $(11)_2 = M_3$

F= $M_0. M_3$

Conversion between Canonical forms: we already study about SOP (sum of product) and POS (product of sum) expressions and calculated POS and SOP forms for dissimilar Boolean

functions. In this section, we will study how we can signify the POS form in the SOP form and the SOP form in the POS form.

For changing the canonical expressions, we have to modify the symbols ∏, ∑. These signs are improved when we list out the index numbers of the equations. From the unique form of the calculation, these directories numbers are accepted. The SOP and POS forms of the boolean function are dual to each other.

The subsequent steps are using which we can easily change the canonical forms of the equations:

1. Modification of the operational symbols used in the equation, such as ∑, ∏.
2. Practice Duality's De-Morgan's standard to write the indexes of the terms that are not accessible in the assumed form of an equation or the index numbers of the Boolean function.

Conversion of pos to sop form: For changing the SOP form from the POS form, we have to modify the symbol ∏ ∑. Later, we mark the numeric tables of lost variables of the supposed Boolean function.

Here are the following stages to change the POS function F = Π a, b, c (2, 3, 5) = AB'C' + AB’C + ABC' into SOP form:

1. In the first step, we convert the operational sign to Σ.
2. Next, we find the lost indexes of the terms, 000, 110, 001, 100, and 111.
3. Lastly, we mark the product form of the noted terms.

Conversion of sop to pos form: For changing the POS form of the assumed SOP form expression, we will modify the symbol ∏ ∑. Later, we will write the numeric indexes of the variables which are lost in the boolean function.

There are the ensuing steps used to exchange the SOP function F = ∑ A, B, C (0, 2, 3, 5, 7) = A' B' C' + A B' C' + A B' C + ABC' + ABC into POS:

1. In the initial step, we convert the operational sign to ∏.
2. We find the misplaced indexes of the terms, 001, 110, and 100.
3. We mark the sum form of the noted terms.

Conversion of sop to standard sop form: For receiving or converting the standard SOP form of the assumed non-standard SOP form, we will add all the variables in respective product terms that do not have all the variables. By consuming the Boolean algebraic law, (A + A' = 0) and by following the beneath steps we can simply change the normal SOP function into standard SOP form.

1. Multiply each non-standard product term by the sum of its misplaced variable and its complement.
2. Recap step 1, till all subsequent product terms, comprise all variables
3. For a separately misplaced variable in the function, the total of product terms doubles.

Conversion of pos form to standard pos form: For the attainment of the standard POS form of the assumed non-standard POS form, we will add all the variables in respective product terms

that do not have all the variables. By intense the Boolean algebraic law (x * x' = 0) and by succeeding in the below steps, we can simply convert the normal POS function into a standard POS form.

1. By adding each non-standard sum term to the product of its absent variable and its complement, which affects 2 sum terms
2. Relating Boolean algebraic law, a + b c = (a + b) * (a + c)
3. Repeating step 1, till all resulting sum terms comprise all variables.

Karnaugh Map Method: A Karnaugh map is a graphical representation of the logic system. This container is drawn directly from moreover minterm (sum-of-products) or maxterm (product-of-sums) Boolean expressions. Illustration of a Karnaugh map from the truth table comprises an extra step of writing the minterm or maxterm expression dependent upon whether it is anticipated to have a minimized sum-of-products or a minimized product-of-sums expression.

The **K-map** is a methodical technique for simplifying Boolean expressions. With the support of the K-map method, we can discover the modest POS and SOP expression, which is known as the minimum expression. The K-map offers a cookery book for generalization[7]–[10].

An objective like the truth table, a K-map comprises all the likely values of input variables and their consistent output values. However, in K-map, the values are kept in cells of the array. In each cell, a binary value of each input variable is put in storage.

The K-map method is used for expressions comprising 2, 3, 4, and 5 variables. For a complex number of variables, there is an additional method used for generalization called the Quine-McClusky method. In K-map, the amount of cells is like the total amount of variable input arrangements. For example, if the amount of variables is 4, the amount of cells is 2^4=16, and if the amount of variables is 6, the quantity of cells is 2^6. The K-map proceeds with the SOP and POS forms. The K-map grid is occupied using 0's and 1's. The K-map is explained by assembly groups. There are the following steps used to explain the expressions using K-map:

1. First, we determine the K-map as per the number of variables.
2. Determine the maxterm and minterm in the specified expression.
3. Fill cells of K-map for SOP with 1 different to the minterms.
4. Fill cells of the block for POS with 0 respective to the maxterm.
5. Next, we make rectangular groups that comprise total terms in the power of two like 2, 4, 8, and try to cover as many elements as we can in one collection.
6. With the help of these collections, we determine the product terms and sum them up for the SOP form.

2-Variable K-map:following is a 2-variable K-map. Here are two variables X and Y in the 2-variable K-map. The following table is showing the construction of the 2-variable K-map (Table 3.6):

Table 3.6: 2-variable k-map

X	Y	K-MAP
0	0	M_0
0	1	M_1

1	0	M_2
1	1	M_3

3- Variable K- map:There is a 3-variable K-map. In this case, we used A, B, and C for the variable. We can use any letter for the names of the variables (Table 3.7 and Table 3.8).

Table 3.7: 3-variable k map

A	B	C	K-MAP
0	0	0	A'B'C'
0	0	1	A'B'C
0	1	0	A'BC'
0	1	1	A'BC
1	1	0	ABC'
1	1	1	ABC
1	0	0	AB'C'
1	0	1	AB'C

Table 3.8: 4-Variable K-map:

AB	CD	K-MAP
00	00	A'B'C'D'
00	01	A'B'C'D
00	11	A'B'CD
00	10	A'B'CD'
01	00	A'BC'D'
01	01	A'BC'D
01	11	A'BCD
01	10	A'BCD'
11	00	ABC'D'
11	01	ABC'D
11	11	ABCD
11	10	ABCD'
10	00	AB'C'D'

10	01	AB'C'D
10	11	AB'CD
10	01	AB'CD'

Don't care condition: The "Don't care" condition speaks that we can use the blank cells of a K-map to create a collection of the variables. To create a collection of cells, we can use the "don't care" cells as either 0 or 1, and if essential, we can similarly overlook that cell. We mostly use the "don't care" cell to create a huge collection of cells.

The cross (×) symbol is used to signify the "don't care" cell in K-map. This cross symbol signifies an unacceptable combination. The "don't care" in excess-3 codes are 0000, 0001, 0010, 1101, 1110, and 1111 because they are unacceptable combinations. Separately from this, the 4-bit BCD to Excess-3 code, the "don't care" are 1010, 1011, 1100, 1101, 1110, and 1111. We can vary the standard SOP purpose into a POS expression by creating the "don't care" terms similar to what they are.

The absent minterms of the POS form are written as max terms of the POS form. In a similar technique, we can modify the standard POS function into an SOP expression by assembling the "don't care" terms the similar as they are. The missing max terms of the SOP form are written as minterm of the SOP form.

In this chapter, we will study the different techniques of simplification or minimization of combinational functions. We also learn about Quine–McCluskey tabular method and SOP, POS, and how to convert SOP to POS, From POS to SOP. We also study how to convert normal POS and SOP forms to standard POS and SOP forms. At last, we will study the Karnaugh map (K-map) method and their specification.

REFERENCES

[1] B. Madoš, N. Ádám, Z. Bilanová, and M. Chovanec, "Fpga hw accelerator of the first step of systematic two-level minimization of single-output boolean function," *Acta Polytech. Hungarica*, 2020, doi: 10.12700/APH.17.7.2020.7.2.

[2] H. Zhang, J. Qian, B. Zhang, J. Yang, C. Gong, and Y. Wei, "Low-Rank Matrix Recovery via Modified Schatten p Norm Minimization with Convergence Guarantees," *IEEE Trans. Image Process.*, 2020, doi: 10.1109/TIP.2019.2957925.

[3] T. K. Gogoi and D. Konwar, "Exergy analysis of a H2O-LiCl absorption refrigeration system with operating temperatures estimated through inverse analysis," *Energy Convers. Manag.*, 2016, doi: 10.1016/j.enconman.2015.12.037.

[4] K. Zhu, M. Liu, H. Chen, Z. Zhao, and D. Z. Pan, "Exploring logic optimizations with reinforcement learning and graph convolutional network," in *MLCAD 2020 - Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, 2020. doi: 10.1145/3380446.3430622.

[5] V. V. Sapozhnikov, V. V. Sapozhnikov, and D. V. Efanov, "Signal correction for combinational automation devices on the basis of Boolean complement with control of calculations by parity," *Informatics*, 2020, doi: 10.37661/1816-0301-2020-17-2-71-85.

[6] W. W. Hager and S. Park, "The gradient projection method with exact line search," *J. Glob. Optim.*, 2004, doi: 10.1023/B:JOGO.0000049118.13265.9b.

[7] C. Ren *et al.*, "Construction of all-in-one peptide nanomedicine with photoacoustic imaging guided mild hyperthermia for enhanced cancer chemotherapy," *Chem. Eng. J.*, 2021, doi: 10.1016/j.cej.2020.127008.

[8] D. Baneres and R. Clarisó, "Evaluation of a new self-study platform for introductory digital systems," *Int. J. Eng. Educ.*, 2018.

[9] A. V. Kabulov, I. H. Normatov, and A. O. Ashurov, "Computational methods of minimization of multiple functions," in *Journal of Physics: Conference Series*, 2019. doi: 10.1088/1742-6596/1260/10/102007.

[10] S. Roy and C. T. Bhunia, "Simplification of switching functions using hex-minterms," *Int. J. Appl. Eng. Res.*, 2015.

CHAPTER 4

LOGIC GATES FAMILIES

Mr. Sunil Dubey, Associate Professor
Department of Electrical Engineering, Jaipur National University, Jaipur, India
Email Id-sunildubey@jnujaipur.ac.in

Logic families are a collection of logic circuits which is based on the actual specification of elements (resistors and transistors etc.). Families are recognized by how the elements are associated and by the arrangement of the elements used. Logic circuits of a specific family can be interlocked without having to use extra circuits. On the other hand, the output of one logic circuit can be used as the input to another logic circuit. This structure is known as compatibility. All the circuits within a logic family will be compatible with the additional circuits inside that family.

Digital logic is troubled with the interconnection between digital apparatuses and segments. The best-known design of a digital system is the general-purpose digital computer. Most digital circuits are created on a single chip, which is referred to as an integrated circuit (IC).

Integrated circuits comprise a large quantity of interconnected arithmetical circuits within a solitary small package. Small-scale integration (SSI) and medium-scale integration (MSI) devices provide arithmetical functions and large-scale integration (LSI) devices offer comprehensive computer components. The compound digital functions can be realized by employing these elementary automated components in a variety of forms and each form is referred to as a logic family[1]–[3].

Logic circuits are typically artificial as integrated circuits (ICs) and bundled in dual inline packages (DIP). An electric circuit in a package is generally presented via standard logic symbols instead of specific modules such as transistors, diodes, and subsequently faraway. Several logic circuit groups or families have been introduced. They differ primarily in the methods for carrying out the logic and the coupling to the inverter stages. A logic family of arithmetical integrated devices is a group of microelectronic logical operation gates made employing one of the numerous dissimilar designs, frequently with compatible logic levels and power supply features inside a family. Several logic families were created as dissimilar elements, each comprising one or a rare connected basic logical function, which might be used as structure blocks to make methods or as so termed glue to interrelate more complex unified paths.

A logic family might similarly be a collection of techniques used to tool logic within the VLSI Integrated circuits such as fundamental processors, memories, or alternative composite drives. Approximately such logic families use static methods to decrease design complications.

Technologies: the gradient of packed assembly block logic families can be divided into categories, arranged here in roughly successive order of overview across with their typical reductions:

Digital integrated circuit (IC): A chip is a small piece of semiconductor material. Utilizing advanced photographic, miniature circuits can be created on the surface of the chip. The finished network is so minor that you need a microscope to see the connection. The components such as

resistors, diodes, transistors, etc. are an integral part of the chip. Therefore, such a circuit is called an integrated circuit.

Level of integration: digital ICs can be classified according to the number of gates on a chip, this is referred to as the level of integration (Table 4.1).

Table 4.1: IC Classifications

IC Classification	Number of gates on a chip
Small scale integration (SSI)	Less than 12
Medium scale integration (MSI)	12 to 99
Large scale integration (LSI)	100 to 999
Very-large-scale integration (VLSI)	1000 to 9999
Ultra large-scale integration (ULSI)	10000 or more

Types of semiconductor digitals ic's: there are two types of semiconductor digitals ic's, as given below:

1. Unipolar
2. Bipolar

Unipolar or bipolar ic's are further sub-divided:
Unipolar ic's: PMOS, NMOS, CMOS.
Bipolar ic's: Saturated (RTL, DTL, TTL, IIL, HTL, DCT
Unsaturated (ECL, Schottky TTL)

Unipolar logic family: in unipolar logic families, unipolar devices are the key elements. MOSFET (Metal oxide semiconductor field effect transistor) is a unipolar device, in which the current flows because only one type of charge carries either electrons or holes.

Bipolar logic family: transistors and diodes are bipolar devices, in which the current flow because both the charge carries electrons and holes. In bipolar logic families, transistors and diodes are used as the key element.

- **PMOS:** P- channel MOSFET family
- **NMOS:** N- channel MOSFET family
- **CMOS:** Complementary MOSFET family
- **RTL:** Resistor transistor logic
- **DCTL:** Directly coupled transistor logic
- **IIL:** Integrated injection logic
- **DTL:** Diode transistor logic
- **HTL:** High threshold logic
- **TTL:** Transistor transistor logic
- **EML:** Emitter-coupled logic

Characteristics of ICs: are given below:

Propagation delay (speed): the propagation delay is defined as "the time required to change the output from one logic to another logic state after input is applied". Manufactured specifies the speed of the digital family in terms of propagation delay. It is denoted by "tp" for example, the TTL logic family has t_p = 10 ns. It means it takes 10 nano sec time to change the output from one state to another state.

Practically propagation delay is calculated by measuring the two propagation delays:
T_{phl}: delay required to change from high to low state (turn OFF delay).
T_{plh}: delay required to change from low to high (turn ON delay).
The higher the propagation delay, the lower the speed. Suppose 100 gates are connected in series and each is producing t_p of 10ns then the total delay is 10 x 100 = 1000ns or 1ms. Effectively it lowers the speed of operation.

Power dissipation: it is indirectly related to propagation delay. Low power dissipation is desirable because it generates less heat and therefore it avoids cooling, power supply, and cost problems. Low power dissipation means it should operate with minimum voltage and with the minimum current. The required current can be adjusted by increasing the resistance but it increases the time because T = R *C. manufactures specifics power dissipation "P_D" in milliwatts (mW). For example, if a gate takes 2mA current and is operated with a +5v supply then its power dissipation per gate is calculated as

Power dissipation = 2mA x 5v = 10 mW

A figure of merit: this feature indicates the total performance or quality of a family. If the resistance of the circuit is increased to reduce the current or power dissipation then the propagation delay increases. Oppositely, if we reduce the propagation delay to make it faster by reducing the resistance then more current is drawn by the gate, which increases the power dissipation. Therefore both these characteristics, propagation delay, and power dissipation are expressed in terms of the figure of merit. it is defined as " the product of propagation delay and power dissipation" of the gate. A figure of merit is measured in terms of "PJ" (Picojoules)

A figure of merit = F.M = t_p x P_D = n-sec x mW = pJ

Fan in – Fan Out: the capacity of driving loads of agate is expressed in terms of fan out. A fan is the "number of inputs to a gate". Fan out is defined as "the maximum number of gates or loads from the same family, which are driven reliably". Fan out is also known as the loading factor. In the case of TTL fan in and fan out the 10 that is a maximum of 10 TTL gates can be connected to the output terminal of a TTL gate[4]–[6].

Noise immunity (noise margin): the maximum noise voltage a device can withstand without making a false change in output. The noise may be an external signal generated by vehicle ignition, a signal generator.

Operating temperature: manufacturer specifies the temperature range in which the devices operate properly. For industrial and consumer applications the temperature range is 0 to 70°C. For military applications, it is 55 to 125°C. In the case of TTL, it is for industrial and consumer applications.

Transistor as a switch: a circuit that can turn on/off current in an electrical circuit is stated to as a switching circuit and a transistor can be working as an electronic switch. Cut region off state both junctions are reverse biased, I,e, IC = 0 and V< O.7V. Saturation region on state IC = maximum and V> 0.7v (Figure 4.1).

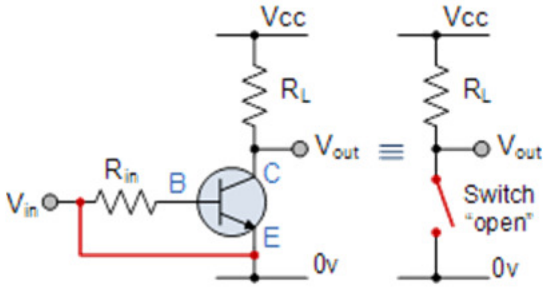


Figure 4.1: Transistor as switch

Resistor Transistor logic (RTL): the simpleRTL device is NOR gate. The inputs signify either logic level high (1) or low (0). The logic level low is the voltage that determines corresponding transistors in the cut-off region, while the logic level high drives it into the saturation region. If both the inputs are low the transistors are in the cut-off region i.e. they are turned off. Thus voltage Vcc seems at output i.e. high. If also transistor or both of them are functional high input, the voltage Vcc drops across Rc and output is low (Figure 4.2).

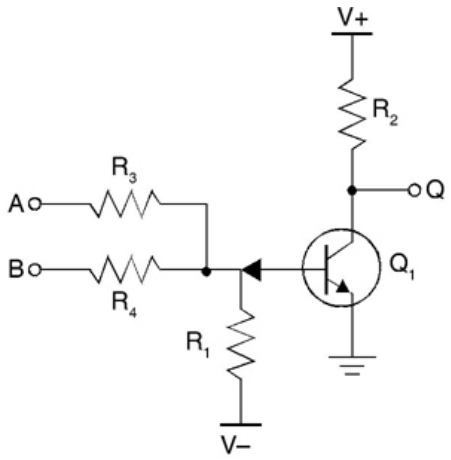


Figure 4.2: RTL using NOR gate

Advantage of RTL: the prime benefit of RTL knowledge was that it elaborates the least number of transistors which was an important deliberation of earlier integrated circuit technology as transistors were the most exclusive module to produce.

Disadvantage of RTL: The problems of RTL are the condition of humble noise margin, unfortunate fan-out capability, low speed, and great power dissipation. Owing to these unwanted characteristics, this family is currently outdated.

Limitations: the boundary of RTL is its high current dissipation after the transistor conducts to overdrivethe o/p biasing resistor. That involves more current being supplied to and heat removed from RTL circuits.

Diode transistor logic (DTL): The diode transistor logic also named DTL substituted the RTL family since of greater fan-out capability and, more noise margin. DTL circuits mainly contain diodes and transistor that comprises DTL strategies (Figure 4.3). The simple DTL device is a NAND gate. Two inputs to the gate are functional through diodes D1 and D2. The diode will conduct only when the corresponding input is low. If any of the diodes are conducting when at least one input is low, the voltage at output keeps the transistor cut off, and subsequently, the output of the transistor is high. If all inputs are high all diodes are non-conducting transistor is in saturation and its output is low.

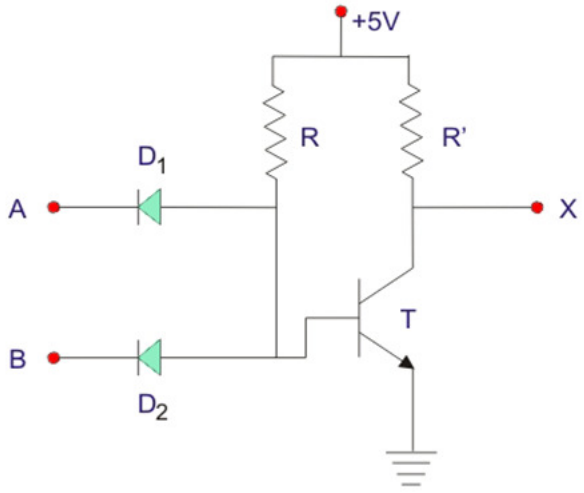


Figure 4.3: Transistor Transistor logic

Transistor Transistor logic (TTL): TTL is the alteration of DTL. It has derived into existence to overcome the speed boundaries of the DTL family. the basis of TTL is the NAND gate (Figure 4.4).

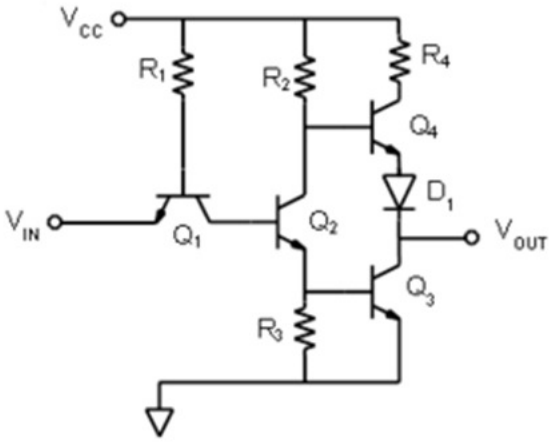


Figure 4.4: Transistor Transistor logic

Q_3 is limit acts similarly a high after outputting transistor Q_4 is saturated and Q3is saturated as soon as output Q_4 is high, consequently, one transistor is ON at one time. The grouping of Q3and

Q_4 is called totem pole arrangements. The output impedance is asymmetrical among the high and low states, making them unsuitable for driving transmission lines. This drawback is usually overcome by buffering the outputs with the special line driver devices where signals need to be sent through cables.

TTL disables the main problem related to DTL (Diode Transistor Logic), i.e., deficiency of speed. The involvement of a TTL circuit has continuously concluded the emitter(s) of the contribution transistor, which displays a low input resistance. The base of the input transistor, on the additional indicator, is linked to the VCC, which reasons the input transistor to permit a current of around 1.6 mA once the input voltage to the emitter(s) is logic '0'. Allowing a TTL input 'float' (left separate) will typically kind it goes to logic '1'. Though, such a state is weak to lost signals, which is why it is a respectable exercise to link TTL inputs to VCC with 1 k ohm pull-up resistors.

Emitter coupled logic (ECL): ECL logic family implements the gates in disparity amplifier arrangement in which transistors are not ever determined in the saturation region thus improving the speed of the circuit to an excessive opportunity. The ECL family is the fastest of all logic families. Based on BJT (Bipolar junction transistor), but removes problems of the delay time by preventing the transistor from saturation. The process of ECL is actual fast-operated propagation delays of 1ns or less. It must have low noise immunity of about 0.2 – 0.25V. The input impedance is high and the output impedance is low, as an effect the transistors change states fast, the gate delay is low and the fanout capability is great (Figure 4.5).

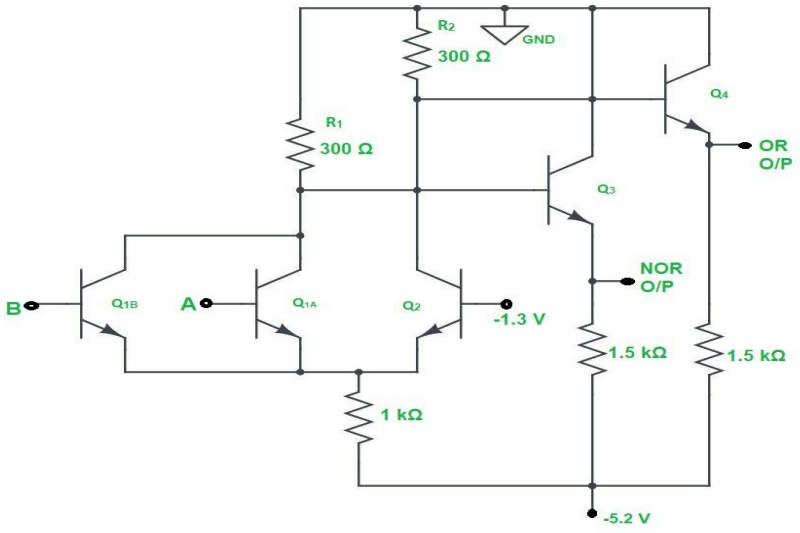


Figure 4.5: Emitter coupled logic

Advantage of ECL: High-speed operation is probable and so is the wildest logic family.Since transistors are not permissible to arrive at saturation, which decreases the storage delay. Fan-outcapability is great.

- Amongst all logic families, ECL gates have the fastest speed subsequently their BJTs work in the active region.

- ECL must have an important advantage over CMOS substituting in that its input stage's current-steering conduct (Q1 and Q2) organizes not similarly disturb the system.
- While it expenses additional static electricity to achieve this noise presentation.
- The outputs created by ECL gates are complimentary (OR-NOR).
- The wired-OR purpose can be applied by coupling outputs.
- Here are no current exchanging points in power supply lines and the properties of temperature on parameters are negligible.
- A single chip might implement a great number of functions and its typical supply voltage is -5.2 V.

The disadvantage of ECL: The main disadvantages of ECL are given below:

- ECL gates are further exclusive than TTL gates.
- It must have a particularly low noise margin (200 mV).
- The highest power dissipation of any logic gates.
- Its damaging supply voltage and logic stages are not feasible with other logic families (manufacture it exciting to use ECL associated with TTL and CMOS circuits).
- To boundary with dissimilar logic families, level shifters are essential.
- The loading limits of capacitors fan out.
- The VLSI project is complex because ECL gates require the manufacture of resistors.

Complimentary MOS (CMOS): It consumes considerably lower energy consumption compared to TTL and ECL. It has made portable electronics possible. It is the most widely used family because of large-scale devices. It combines high speed with less power consumption. It usually operates on a supply of 5-15V. CMOS has great noise immunity of about 30% of the operated voltage. It connected to a large number of gates about 50 gates. CMOS or Complementary Metal Oxide Semiconductor is a combination of NMOS and PMOS transistors that functions below the functional electrical field. The assembly of CMOS was originally industrialized for high-density and low-power logic gates. The NMOS and PMOS are the categories of Metal Oxide Semiconductor Field Outcome Transistors (MOSFET). The CMOS transistors are hand-me-down in several applications, such as amplifiers, switching circuits, logic circuits, Integrated circuit chips, microprocessors, etc. The position of CMOS in semiconductor knowledge is its low power dissipation and low working currents. Its industrial needs fewer steps as associated with the Bipolar Junction transistors[7]–[10].

Features of CMOS Logic Gates: The structures of CMOS Logic Gates are enumerated as follows:

1. Concentrated cost as it needs quite a single power supply.
2. Excessive logic swing.

3. Huge fan-out capability.
4. Actual great noise margin.
5. Minor propagation delay
6. Great speed is associated with NMOS transistors.
7. Minor power dissipation.
8. Superb temperature stability.
9. Lesser packaging density.

Directly coupled transistor logic (DCTL): Transistorized digital circuits achieve the three logical purposes of AND (or NAND) gating, OR (or NOR) gating, and signal reversal (NOT gate). A further function typically achieved, however not rational than yet an applied need is the signal extension. Further logical blocks, such as NOR, NAND, and flip-flops, are simply found consuming these three essential useful blocks. Several dissimilar circuit shapes can be cast off for these useful blocks. Usually, these circuits are secret conferring to the elements cast-off for interstage coupling or coupling among gates and inverters, and amplifiers. The most usually used coupling elements are diodes, resistors, resistor-capacitor groupings, and transistors themselves. It is also likely to project and use circuits short of any of these coupling elements. Such circuits are referred to as direct-coupled transistor-logic circuits or, additionally usually, DCTL. There are numerous advantages and disadvantages of DCTL, and we reflect on these after deliberating how this type of formation works.

DCTL Series Gating: The figure below demonstrates three transistors linked in series to form a NAND gate for positive input signals A, B, and C. If slightly of the three transistors are off, the output power at D will be the supply power (V_{CC}) in the dropped form. Below troubled circumstances, the voltage at D will depend on the resistor R_L and the $V_{BE(ON)}$ of the next-stage transistor. When all three transistors are on, the possible at D will be closer to the ground than in the previous case and will be the sum of the $V_{CE(SAT)}$ of Q_1, Q_2, and Q_3 in series. Consequently, the principal trouble of this configuration is the requirement to protect that the succeeding step transistor will be off once all three transistors are on. The sum of the three $V_{CE(SAT)}$ in series necessity is not as much as the $V_{BE(ON)}$ of the next stage transistor. Single means of accomplishing this is to supply more base drives to Q_1, Q_2, and Q_3, thereby illustrating them additional into saturation and dropping the saturation resistance (Figure 4.6).

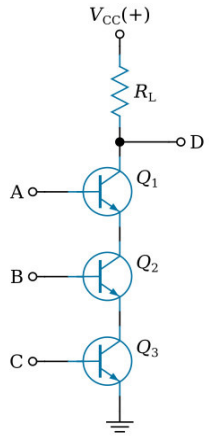


Figure 4.6: DCTL series gate

DCTL Parallel Gating: The figure displays a parallel DCTL gate, which is the before careful "inverter" phase with three transistors needing separate inputs in place of a single transistor. Perceptibly, this arrangement is a NOR circuit. If slightly input is high on somewhat of the three transistors, its collector determines the attraction current through the load resistor, producing the output to go low (Figure 4.7).

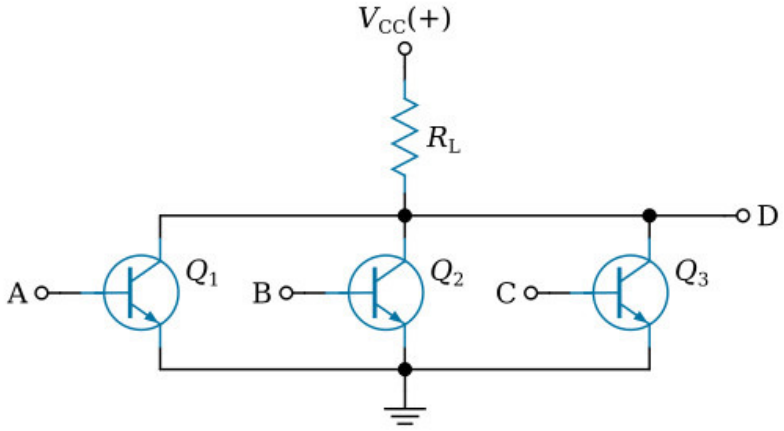


Figure 4.7: DCTL parallel gating

In this chapter, we study the logic gate families such as TTL, DCTL, ECL, RTL, and digital ICs or their characteristics. It also studies the features, types, advantages, or disadvantages of these logic family circuits.

REFERENCES

[1] K. Degawa, T. Aoki, T. Higuchi, H. Inokawa, and Y. Takahashi, "A single-electron-transistor logic gate family for binary, multiple-valued and mixed-mode logic," *IEICE Trans. Electron.*, 2004.

[2] C. K. Lee, S. J. Kim, S. J. Shin, J. B. Choi, and Y. Takahashi, "Single-electron-based flexible multivalued logic gates," *Appl. Phys. Lett.*, 2008, doi: 10.1063/1.2888164.

[3] F. Wei, X. Cui, and X. Cui, "An improved iMemComp or gate and its applications in logic circuits," *IEEE J. Electron Devices Soc.*, 2020, doi: 10.1109/JEDS.2019.2962822.

[4] K. M. Kim and R. Stanley Williams, "A Family of Stateful Memristor Gates for Complete Cascading Logic," *IEEE Trans. Circuits Syst. I Regul. Pap.*, 2019, doi: 10.1109/TCSI.2019.2926811.

[5] V. Jamshidi and M. Fazeli, "Pure Magnetic Logic Circuits: A Reliability Analysis," *IEEE Trans. Magn.*, 2018, doi: 10.1109/TMAG.2018.2846623.

[6] X. Cui, Y. Ma, F. Wei, and X. Cui, "The Synthesis Method of Logic Circuits Based on the NMOS-Like RRAM Gates," *IEEE Access*, 2021, doi: 10.1109/ACCESS.2020.2967080.

[7] Z. Zheng *et al.*, "Gallium nitride-based complementary logic integrated circuits," *Nat. Electron.*, 2021, doi: 10.1038/s41928-021-00611-y.

[8] O. C. Yee and L. S. King, "Nano-MOSFETS implementation of different logic families of two inputs NAND gate transistor level circuits: A simulation study," *J. Teknol.*, 2017, doi: 10.11113/jt.v79.9947.

[9] G. V. Resta *et al.*, "Doping-Free Complementary Logic Gates Enabled by Two-Dimensional Polarity-Controllable Transistors," *ACS Nano*, 2018, doi: 10.1021/acsnano.8b02739.

[10] R. Uma, J. Ponnian, and P. Dhavachelvan, "New low power adders in Self Resetting Logic with Gate Diffusion Input Technique," *J. King Saud Univ. - Eng. Sci.*, 2017, doi: 10.1016/j.jksues.2014.03.006.

CHAPTER 5

COMBINATIONAL CIRCUITS

Mr. Vivek Jain, Associate Professor
Department of Electrical Engineering, Jaipur National University, Jaipur, India
Email Id-vivekkumar@jnujaipur.ac.in

Combinational circuits perform a specific information processing operation fully specified logically by a set of Boolean functions. Combinational circuits consist of input variables, logic gates, and output variables. The output of combinational circuits depends on its present inputs only.A combinational circuit is one wherever the output at a slight time depends simply on the present combination of inputs now by total contempt for the past state-run of the inputs. The logic gate is the maximum elementary building block of combinational logic. The logical function achieved by a combinational circuit is entirely defined by a set of Boolean expressions. The additional group of logic circuits, called sequential logic circuits, includes both logic gates and memory elements such as flip-flops. Owed to the existence of memory elements, the output in a sequential circuit is contingent upon not only the present but also the past state of inputs (Figure 5.1).

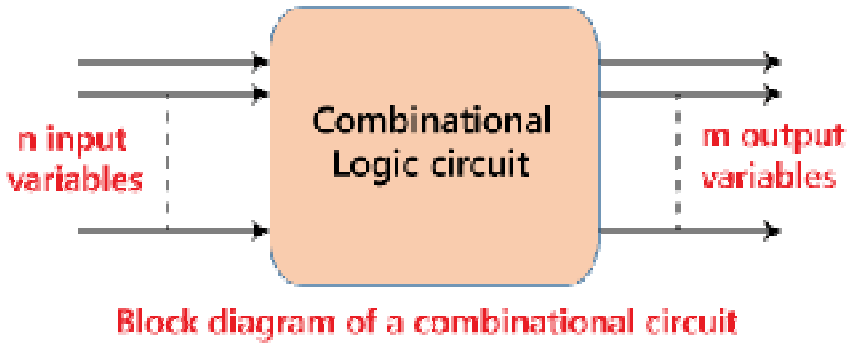


Figure 5.1: Combinational circuits

Design procedure:

1. The number of available input variables and required output voltages is determined.
2. The input and output variables are assigned letter symbols.
3. The truth table is derived that defines the required relationship between inputs and outputs variables.
4. A simplified Boolean function for each output is obtained.
5. Then, the logic diagram is drawn with the help of logic gates.

Combinational logic circuits are circuits that comprise dissimilar kinds of logic gates. Just, a circuit in which dissimilar types of logic gates are joined is identified as a **combinational logic circuit**. The output of the combinational circuit is expressed from the existing arrangement of inputs, irrespective of the previous input. The input variables, logic gates, and output variables are the elementary modules of the combinational logic circuit. There are dissimilar kinds of combinational logic circuits, such as Adder, Subtractor, Decoder, Encoder, Multiplexer, and De-multiplexer[1]–[3].

Half Adder: The half adder is a simple building block requiring two inputs and two outputs. It adds the two inputs and produces the sum S and the carry C. it perform the arithmetic operation of the addition of two single bit. The adder is cast-off to perform OR operation and the addition of two single-bit binary numbers. The **carry** and **sum** are two output conditions of the half adder.

Full Adder: The half adder is used for the addition of simply two numbers. Half adder is a simple circuit but not effective for adding more than two digits or numbers. To overcome this difficulty, the full adder was established. When three bits need to add, we use a full adder circuit (the third bit is the previous carry bit). The full adder is used to add three 1-bit binary numbers A, B, and carry C. The full adder has three input states and two output conditions i.e., sum and carry.

Half Subtractor: The half subtractor is correspondingly a building block of subtracting two binary numbers. It has two inputs and two outputs. This circuit is cast-off to subtract two single-bit binary numbers A and B. The **'diff'** and **'borrow'** are the two output states of the half adder.

Full Subtractor: The Half Subtractor is cast-off to deduct only two numbers. To stun this difficulty, a full subtractor was implemented. The full subtractor is cast-off to deduct three 1-bit numbers A, B, and C, which are minuend, subtrahend**,** and borrow**,** correspondingly. The full subtractor has three input conditions and two output conditions i.e., diff and borrow.

Multiplexers: The multiplexer is a combinational circuit that requires n-data inputs and a single output. It is similarly well-known as the data selector which picks one input from the inputs and ways it to the output. With the help of the designated inputs, one input line from the n-input lines is nominated. The enable input is represented by E, which is cast off in cascade.

De-multiplexers: A De-multiplexer executes the opposite process of a multiplexer. The de-multiplexer has a single input, which is dispersed over numerous outputs. One output line is designated at a time by picking lines. The input is communicated to the designated output line.

Encoder: The encoder is cast-off to execute the reverse operation of the decoder. An encoder requiring an n integer of inputs and an m integer of outputs is castoff to produce m-bit binary code which is associated with the digital input digit. The encoder proceeds with the digital world and changes it into an alternative digital word.

Decoder: A decoder is a combinational circuit ensuring n inputs and a thoroughgoing of m = 2n outputs. The decoder is similar to the de-multiplexer. The only modification between a de-multiplexer and a decoder is that in the decoder, there is no data input. The decoder executes an operation that is the opposite of an encoder.

Half–Adder: The half-adder is a simple building block requiring two inputs and two outputs. It adds the two inputs and produces the sum S and the carry C. it perform the arithmetic operation of the addition of two single bit. The adder is cast-off to perform OR operation and the addition of two single-bit binary numbers. The carry and sum are two output conditions of the half adder.

The Half adder circuit has two inputs that express the two bits to be added and two outputs which is producing the SUM output and another output is showing the CARRY bit (Figure 5.2). The Boolean function for the SUM and CARRY outputs are given as follows (Table 5.1):

SUM S = A.B’ + A’.B

CARRY C = A.B

Truth table 5.1: Half Adder

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

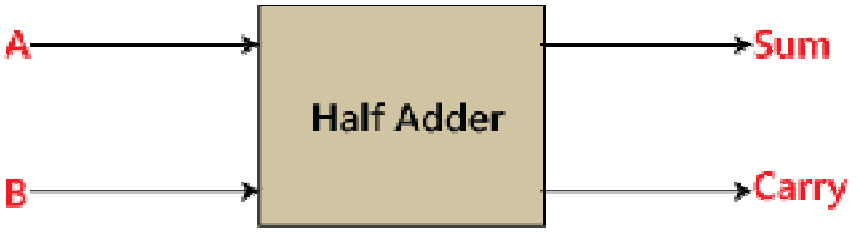


Figure 5.2: Half Adder

Construction of Half–Adder circuit:

1. The SUM bit is produced with the help of 2- an input Exclusive-OR gate (EX-OR).
2. The CARRY (C) bit is generated wi the help of 2- input AND gate (Figure 5.3).

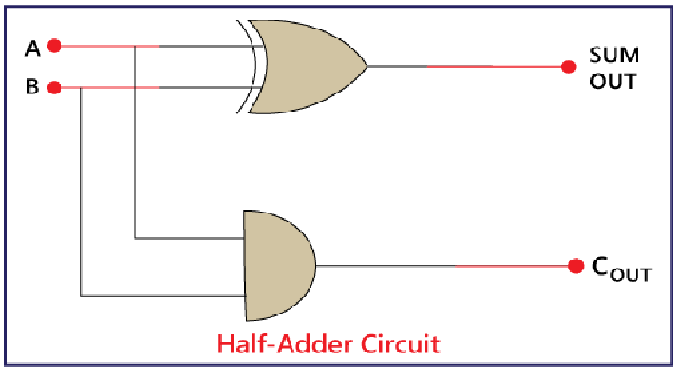


Figure-5.3: Half Adder

SUM S = A.B' + A'.B

CARRY C = A.B

Full Adder: The half adder is used for the addition of simply two numbers. Half adder is a simple circuit but not effective for adding more than two digits or numbers. To overcome this difficulty, the full adder was established. When three bits need to add, we use a full adder circuit

(third bit is the previous carry bit). The full adder is used to add three 1-bit binary numbers A, B, and carry C. The full adder has three input states and two output conditions i.e., sum and carry.

A full adder is a circuit an arithmetic block that the container is cast-off to add three bits to generate a SUM and a CARRY output. The full adder circuit disables the boundaries of the half-adder. Which can be cast-off to add two bits only, letting the recollection of the process for adding the largest binary number. We start with the addition of LSBs of the two numbers. We note the sum below the LSB column and proceed with the carry. If any, onward to the succeeding higher column bits. As a result, after we added the next higher column bits, we desired to be essential to add three bits if there were a carry from the preceding addition (Table 5.3). We have the same condition for the other higher column bits also till we influence MSB. A full adder is important for the hardware operation of an adder circuit accomplished by adding the largest binary number. A half-adder can also be cast-off for the addition of LSBs only (Figure 5.4).

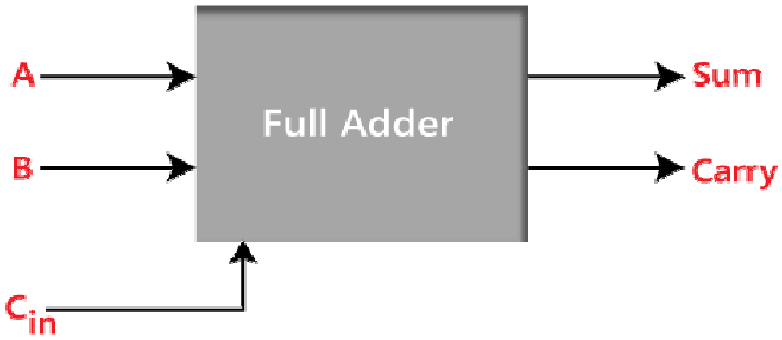


Figure 5.4: Full Adder

SUM = A'B'C + A'BC + AB'C + ABC

CARRY = AB + AC + BC

Table 5.3: Truth of Full Adder

A	B	C	Sum	Carry
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

Construction of FULL adder circuit (Figure 5.5.):

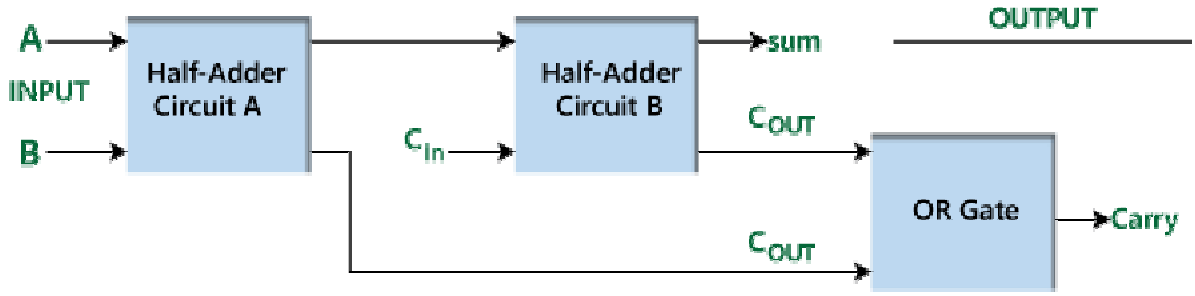


Figure 5.5: Half Adder circuit

In the overhead circuit, two half-adder circuits are mutually consuming the OR gate. The first half adder has two single-bit binary inputs A and B. As per what we recognize, the half adder generates two outputs, i.e., Sum and Carry. The 'Sum' output of the first adder will be the first input of the second half adder, and the 'Carry' output of the first adder will be the second input of the second half adder. The second half adder will over-provide 'Sum' and 'Carry'. The conclusion of the Full adder circuit is the 'Sum' bit. To bargain the last output of the 'Carry', we offer the 'Carry' output of the first and the second adder into the OR gate. The conclusion of the OR gate will be the final carry-out of the full adder circuit. The MSB is signified by the final 'Carry' bit (Figure 5.6).

The full adder logic circuit could be assembled utilizing the 'AND' and the 'XOR' gate through an OR gate**.**

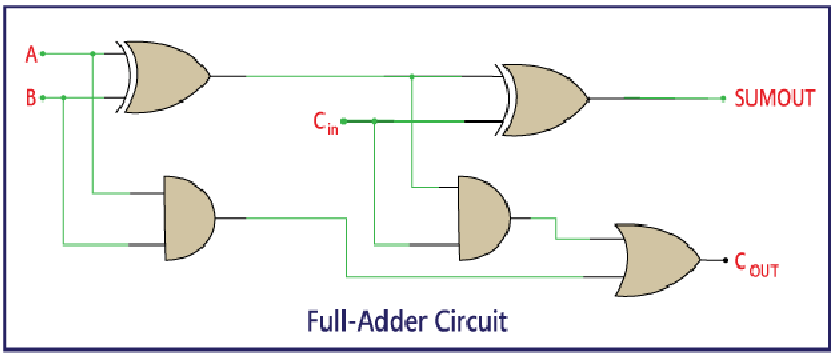


Figure 5.6: Full adder circuit

1. **SUM:** Implement the XOR operation of inputs A and B.
2. Execute the XOR operation of the conclusion with carry. So, the sum is (A XOR B) XOR C_{in} Which is represented by: $(A \oplus B) \oplus C_{in}$

CARRY: Execute the 'AND' operation of inputs A and B.

1. Execute the 'XOR' operation of inputs A and B.
2. Execute the 'OR' operations of mutually the outputs that outcome from the preceding two phases. So the 'Carry' can be expressed employing: $A.B + (A \oplus B)$.

Half Subtractor: The half subtractor is correspondingly a building block of subtracting two binary numbers. It has two inputs and two outputs (Table 5.4). This circuit is cast-off to subtract two single-bit binary numbers A and B. The 'diff' and 'borrow' are the two output states of the half adder (Figure 5.7)[4]–[6].

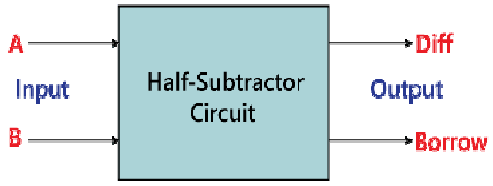


Figure 5.7: Half Subtractor

The SOP (SUM Of Product) form of the **Diff** and **Borrow** is as follows:

$$DIFF = X'Y + XY'$$

$$BORROW = X'Y$$

Table 5.4: Truth Table of Half Subtractor

X	Y	DIFF	BORROW
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Construction of half subtractor:

1. To generate the DIFF bit it is used 2- an input EX-OR gate is.
2. Input to produce the BORROW bit.
3. The NOT gate is cast-off to become the opposite output. We can combine the 'AND' besides 'NOT' gates to become the combinational gate 'NAND'. By reversing the input 'A' using the 'NOT' gate and formerly using the output of the 'NOT' gate as the input of the 'AND' gate, we can acquire the 'Borrow' bit (Figure 5.8).

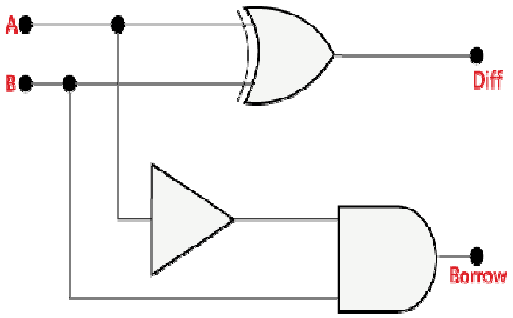


Figure 5.8: Half Subtractor

Diff= A XOR B (A⊕B)

Borrow= NOT-A AND B (A'.B)

Full Subtractor: The Half Subtractor is cast-off to deduct only two numbers. To stun this difficulty, a full subtractor was implemented. The full subtractor is cast-off to deduct three 1-bit numbers A, B, and C, which are minuend, subtrahend**,** and borrow**,** correspondingly (Figure 5.8). The full subtractor has three input conditions and two output conditions i.e., diff and borrow (Table 5.5).

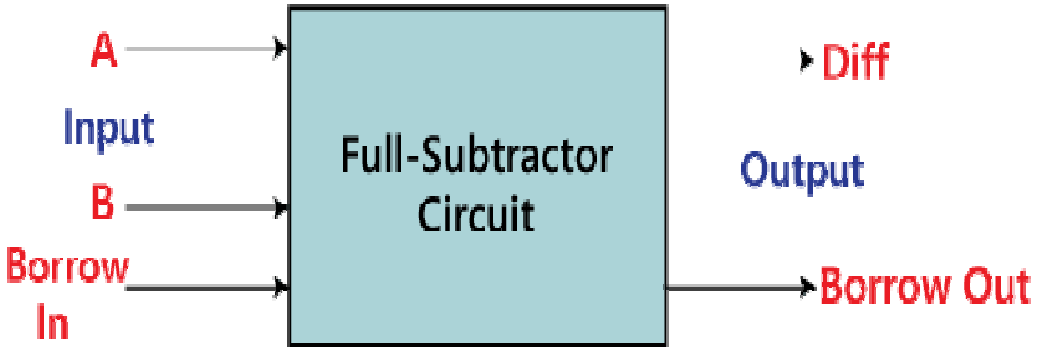


Figure-5.9: Full Subtractor

Diff=AB'C'+A'B'C+ABC+A’BC'

Borrow=A’C+A'B+BC

Table 5.5: Truth Table of Full Subtractor

A	B	C(Borrow$_{in}$)	Diff	Borrow
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

Construction of Full adder (Figure 5.10):

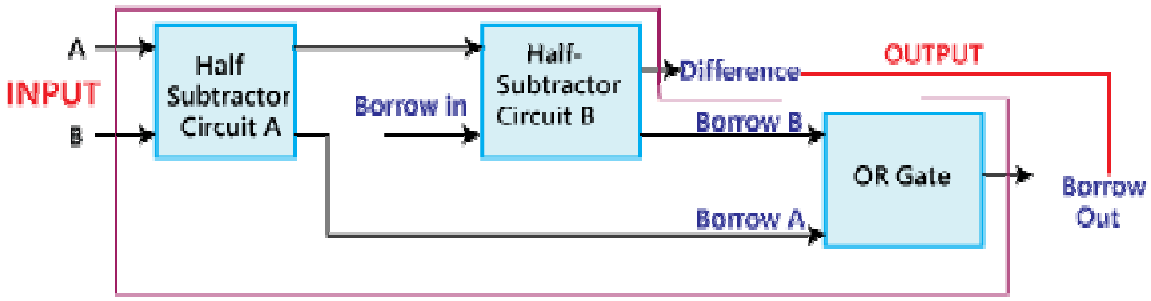


Figure 5.10: Constructions of full adder

In the overhead circuit, two half-adder circuits are joined employing the OR gate. The first half subtractor requires two single-bit binary inputs A and B. As we recognize, the half subtractor generates two outputs, i.e., 'Diff' and 'Borrow'. The 'Diff-output of the first subtractor will be the first input of the second half subtractor, and the 'Borrow' output of the first subtractor will be the second input of the second half subtractor. The second half subtractor will once more deliver 'Diff' and 'Borrow'. The conclusion of the Full subtractor circuit is the 'Diff' bit. To generate the final output of the 'Borrow', we provide the 'Borrow' of the first and the second subtractor into the OR gate. The conclusion of the OR gate will be the concluding carry 'Borrow' of the full subtractor circuit. The MSB is signified by the last 'Borrow' bit. The full subtractor logic circuit can be created utilizing the 'AND', 'XOR', and NOT gate with an OR gate (Figure 5.11).

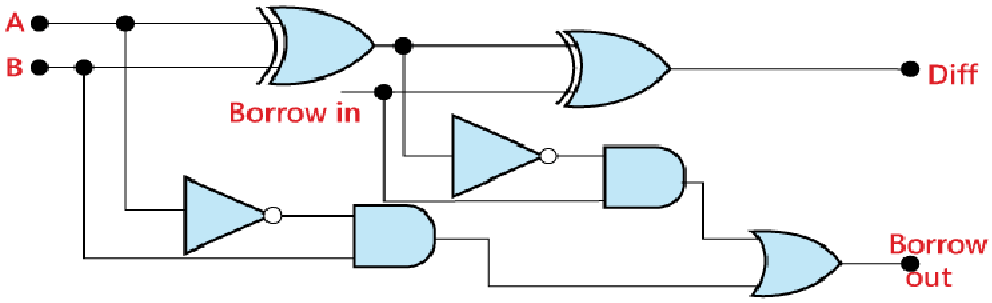


Figure-5.11: Construction of full subtractor

DIFF: Achieve the XOR operation of input A and B. Implement the XOR operation of the conclusion with 'Borrow'. So, the variation is (A XOR B) XOR 'Borrow$_{in}$' which is also signified as (A $\oplus$ B) $\oplus$ 'Borrow$_{in}$'

BORROW: Execute the 'AND' operation of the reversed inputs A and B. Perform the 'XOR' operation of inputs A and B.

Binary adder: The registers perform a significant part in the execution of the micro-operations. The registers hold the digital module and the data which executes the arithmetic operation. The Binary Adder is a rational circuit that is cast-off to execute the addition operation of two binary numbers of any length. The Binary Adder is designed with the assistance of the Full-Adder circuit. The Full-Adders are associated in series, and the output carry of the first Adder will be preserved as the input carry of the subsequent Full-Adder[7]–[10].

Binary adder subtractor: A Binary Adder-Subtractor is a superior kind of circuit that is cast off to execute both operations, i.e., Addition and Subtraction. The operation which is working to be used is determined by the values enclosed by the resistor signal. In Arithmetic Logical Unit, it is one of the supreme significant components. To effort through Binary Adder-Subtractor, we must identify the XOR gate, Full-Adder, Binary Addition, and subtraction (Figure 5.12).

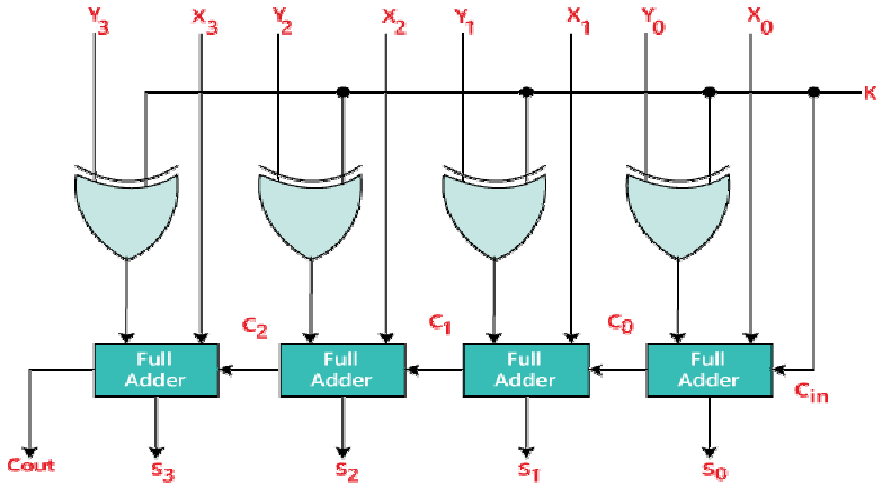


Figure-5.12: Binary adder subtractor

Decimal or BCD Adder: The BCD-Adder is cast-off in computers and calculators that execute arithmetic operations straightly in the decimal number system. The BCD-Adder takes the binary-coded form of decimal numbers. The Decimal Adder needs a least nine inputs and five outputs.

REFERENCES

[1] G. Liu, S. Shen, P. Jin, G. Wang, and Y. Liang, "Design of Memristor-Based Combinational Logic Circuits," *Circuits, Syst. Signal Process.*, 2021, doi: 10.1007/s00034-021-01770-1.

[2] S. Shiraz, O. Hasan, and S. Member, "A library for combinational circuit verification using the HOL theorem prover," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2018, doi: 10.1109/TCAD.2017.2705049.

[3] R. Bhat, M. R. Ansari, and R. Khanam, "Effect of integrated power and clock networks on combinational circuits," *Int. J. Reconfigurable Embed. Syst.*, 2020, doi: 10.11591/ijres.v9.i3.pp242-248.

[4] A. Singh, "Design and Analysis of Memristor-based Combinational Circuits," *IETE J. Res.*, 2020, doi: 10.1080/03772063.2018.1486741.

[5] "A Review on Reversible Computing and it's applications on combinational circuits," *Int. J. Emerg. Trends Eng. Res.*, 2021, doi: 10.30534/ijeter/2021/28962021.

[6] R. A. Jaber, J. M. Aljaam, B. Owaidat, S. Al-Maadeed, A. Kassem, and A. M. Haidar, "Ultra-Low Energy CNFET-Based Ternary Combinational Circuits Designs," *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3105577.

[7] N. Kehl and W. Rosenstiel, "An efficient ser estimation method for combinational circuits," *IEEE Trans. Reliab.*, 2011, doi: 10.1109/TR.2011.2161700.

[8] A. H. El-Maleh and K. A. K. Daud, "Simulation-Based Method for Synthesizing Soft Error Tolerant Combinational Circuits," *IEEE Trans. Reliab.*, 2015, doi: 10.1109/TR.2015.2440234.

[9] H. Li and Y. Wang, "Boolean derivative calculation with application to fault detection of combinational circuits via the semi-tensor product method," *Automatica*, 2012, doi: 10.1016/j.automatica.2012.01.021.

[10] L. A. M. de Souza, J. E. H. da Silva, L. J. Chaves, and H. S. Bernardino, "A benchmark suite for designing combinational logic circuits via metaheuristics," *Appl. Soft Comput. J.*, 2020, doi: 10.1016/j.asoc.2020.106246.

CHAPTER 6

ENCODER AND MULTIPLEXER

Mr. Sunil Dubey, Associate Professor
Department of Electrical Engineering, Jaipur National University, Jaipur
India, Email Id-sunildubey@jnujaipur.ac.in

The encoder is cast-off to execute the reverse operation of the decoder. An encoder requiring an n integer of inputs and an m integer of outputs is castoff to produce m-bit binary code which is associated with the digital input digit. The encoder proceeds with the digital world and changes it into an alternative digital word.

The combinational circuits that modify the binary data into N output lines are known as Encoders**.** The binary data is accepted in the form of 2^N input lines. The output lines describe the N-bit code for the binary data. In modest arguments, the Encoder executes the opposite operation of the Decoder**.** At a stage, only one input line is triggered for effortlessness. The created N-bit output code is corresponding to the binary information (Figure 6.1)

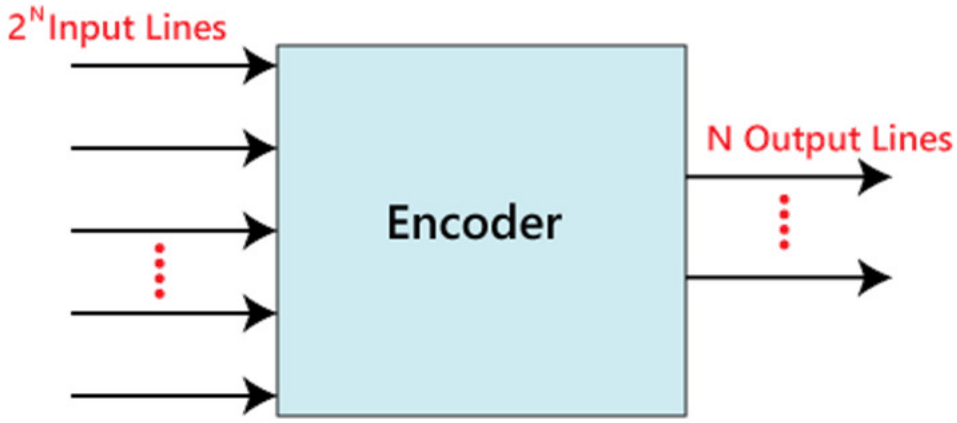


Figure 6.1: Represent the Encoder

4 to 2 line Encoder:In a 4 to 2 line encoder, the encoder is an overall of four inputs, Y_0, Y_1, Y_2, and Y_3, and two outputs, A_0 and A_1. In four input lines, one input line is fixed to true at a time to become the corresponding binary code on the output on the side (Figure 6.1). Beneath are the block diagram and the truth table of the 4 to a 2-line encoder (Figure 6.2 and Figure 6.3)[1]–[3].

Table 6.1: Truth Table of Encoder

Y_0	Y_1	Y_2	Y_3	A_0	A_1
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
1	0	0	0	1	1

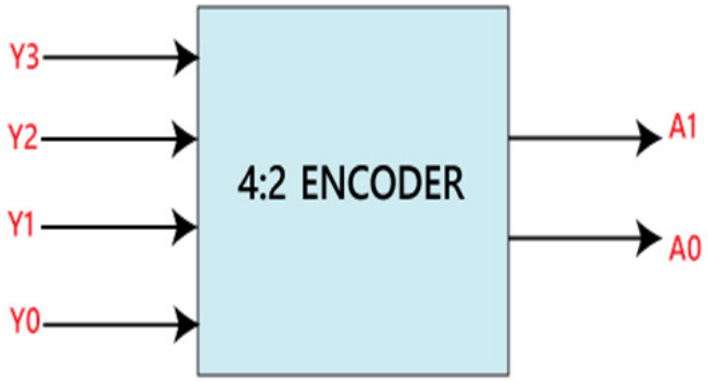


Figure 6.2: 4 to 2-line Encoder

$$A_0 = Y_3 + Y_1$$

$$A_1 = Y_3 + Y_2$$

The logic circuit of the 4 to 2- line encoder is specified as follows:

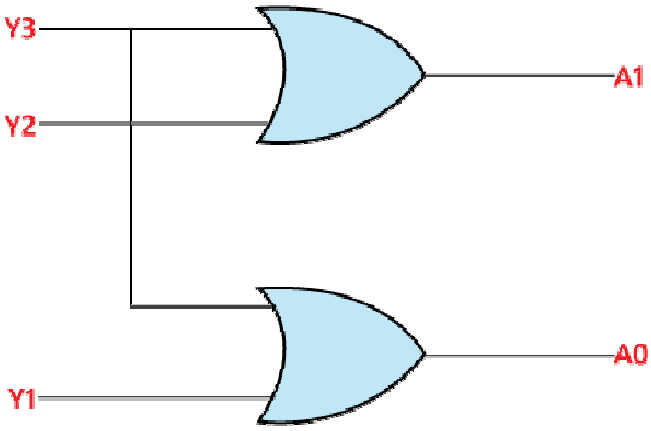


Figure 6.3: Encoder circuit diagram

8 to 3-line Encoder: The 8 to 3 line Encoder is correspondingly identified as an Octal to Binary Encoder. In the 8 to 3 line encoder, here is a whole of eight inputs, Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7, and three outputs, A_0, A1, and A_2. In 8 input lines, one input line is fixed to true at a time to become the particular binary code on the output side (Figure 6.4). Underneath are the block diagram and the truth table of the 8 to 3-line encoder as follows (Table 6.2).

Table 6.2: Truth Table of Encoder

Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	A_0	A_1	A_2
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0

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

$A_0=Y_7+Y_5+Y_3+Y_1$

$A_1=Y_2+Y_3+Y_6+Y_7$

$A_2=Y_4+Y_5+Y_6+Y_7$

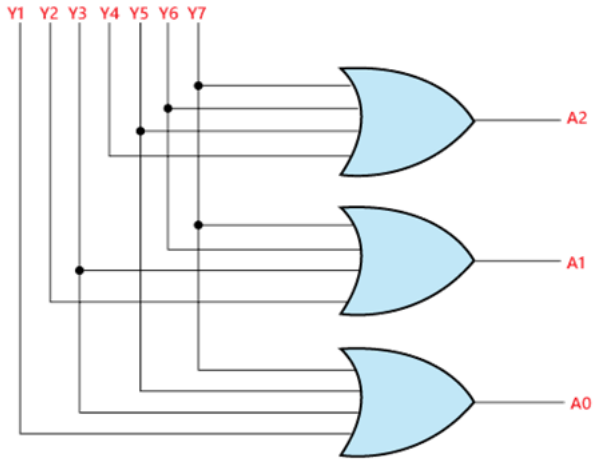


Figure 6.4: Illustrate the Encoder

Decoder: The combinational circuit that variations the binary data into 2^N output lines is known as Decoders**.** The binary data is accepted in the form of N input lines. The output lines describe the 2^N-bit code aimed at the binary data. In modest arguments, the Decoder accomplishes the opposite operation of the Encoder**.** At a time, only one input line is triggered for easiness. The formed 2^N-bit output code is corresponding to the binary data (Figure 6.5).

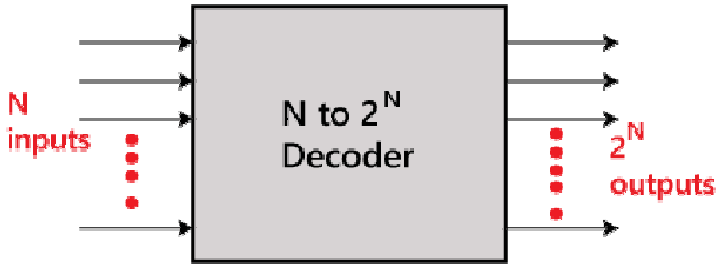


Figure 6.5: Decoder

2 to 4- line Decoder: In the 2 to 4 line decoder, here is a whole of three inputs, A_0, A_1, and Enable, and four outputs, Y_0, Y_1, Y_2, and Y_3. On behalf of the respective arrangement of inputs, when the enable 'E' is set to 1, one of these four outputs will be 1 (Table 6.3). The block diagram and the truth table of the 2 to 4-line decoder are specified as follows (Figure 6.6):

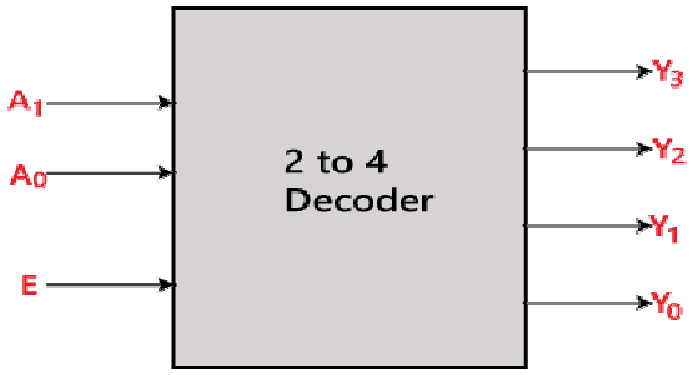


Figure 6.6: Represent the 2:4 Decoder

Table 6.3: Truth Table of 2:4 Decoder

E	A_0	A_1	Y_0	Y_1	Y_2	Y_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	1

$Y0=E.A_1'.A_0'$

$Y_1=E.A_1'.A_0$

$Y_2=E.A_1.A_0'$

$Y_3=E.A_1.A_0$

Multiplexer: The multiplexer is a combinational circuit that needs n-data inputs and a single output. It is similarly well-known as the data selector which picks one input from the inputs and procedures it to the output. With the help of the designated inputs, one input line from the n-input lines is nominated. The enable input is represented by E, which is cast off in cascade.

A multiplexer is a combinational circuit that has 2^ninput lines and a single output line. A multiplexer is an electronic switch that connects 1 out of n inputs to an output. It is functionally complete i.e. all Boolean functions can be realized using one multiplexer without any other gates.

For different encoders and decoders, there are n selection lines and 2^n input lines. Therefore, there is a total of 2^N possible groupings of inputs. A multiplexer is also treated as Mux**.**

There are various types of the multiplexer which are specified below as:

2 x 1 Mux: In a 2×1 multiplexer, there are just 2 inputs, A_0 and A_1, one selection line S_0, and single outputs Y. Based on the arrangement of inputs that are existing at the selection line S_0, one of these 2 inputs will be coupled to the output (Figure 6.7). The block diagram of the 2×1 multiplexer are specified as follows (Table 6.4):$Y=S_0'.A_0+S_0.A_1$[4]–[6]

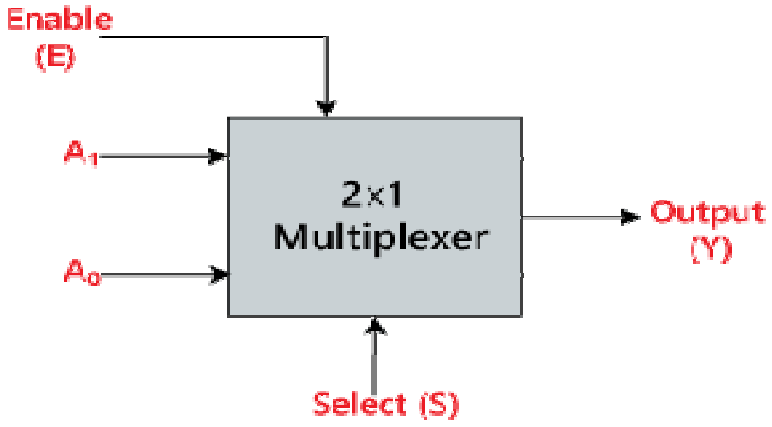


Figure 6.7: 2x1 MUX

Table 6.4: Truth Table of 2x1 MUX

S_O	Y
0	A_0
1	A_1

4 x 1 MUX: In the 4×1 multiplexer, there is an entire of 4 inputs A_0, A_1, A_2, and A_3, 2 selection lines, S_0 and S_1, and a single output, Y. Based on the arrangement of inputs that are existing at the selection lines S^0 and S_1, one of these four inputs are associated to the output. The block diagram (Figure 6.8) of the 4×1 multiplexer is specified as follows**:** $Y=S_1'\ S_0'\ A_0+S_1'\ S_0\ A_1+S_1\ S_0'\ A_2+S_1\ S_0\ A_3$ (Table 6.5).

Table 6.5: Truth Table of 4x1 MUX

S_0	S_1	Y
0	0	A_0
1	0	A_1
0	1	A_2
1	1	A_3

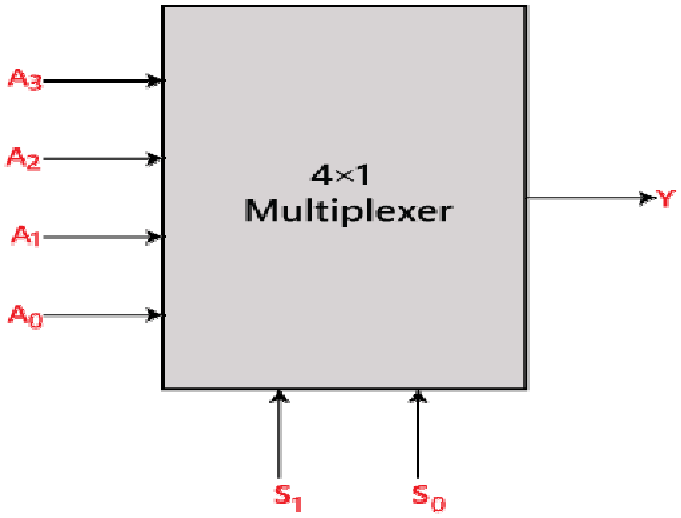


Figure 6.8: 4x1 MUX

8x1 MUX: In the 8 to 1 multiplexer, there are an overall of eight inputs, A_0, A_1, A_2, A_3, A_4, A_5, A_6, and A_7, three selection lines, S_0, and S_2, and a single output, Y (Table 6.6). Based on the grouping of inputs that are existing at the selection lines S^0, $S^{1,}$ and S_2, one of these 8 inputs are coupled to the output. The block diagram of the 8×1 multiplexer is specified beneath (Figure 6.9):

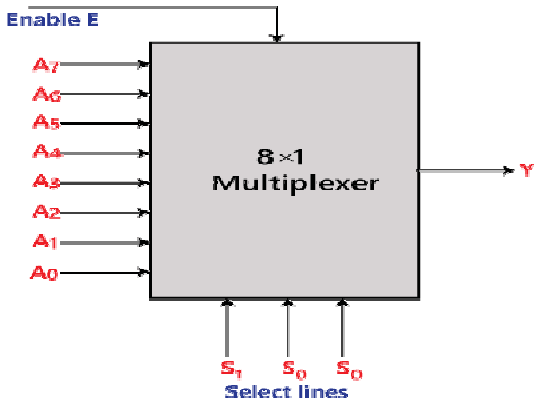


Figure 6.9: Represent the 8x1 MUX

Table 6.6: Truth Table of 8x1 MUX

S_0	S_1	S_2	Y
0	0	0	A_0
1	0	0	A_1
0	1	0	A_2
1	1	0	A_3
0	0	1	A_4

1	0	1	A_5
0	1	1	A_6
1	1	1	A_7

the logical circuit diagram of 8x1 MUX is specified below:

Y= $S_0'.S_1'.S_2'.A_0+S_0.S_1'.S_2'.A_1+S_0'.S_1.S_2'.A_2+S_0.S_1.S_2'.A_3+S_0'.S_1'.S_2\ A_4+S_0.S_1'.S_2\ A_5+S_0'.S_1.S_2\ .A_6+S_0.S_1.S_3.\quad A_7$

8x1 MUX using 4x1 MUX and 2x1 MUX: We can make the 8×1 multiplexer utilizing a lower-order multiplexer. To make the 8×1 multiplexer, we require two 4×1 multiplexers and one 2×1 multiplexer. The 4×1 multiplexer has 2 selection lines, 4 inputs, and 1 output. The 2×1 multiplexer has only 1 selection line. Used for receiving 8 data inputs, we require two 4×1 multiplexers. The 4×1 multiplexer expressed one output. Therefore, to acquire the last output, we elementary a 2×1 multiplexer. The block diagram of the 8×1 multiplexer utilizing 4×1 and 2×1 multiplexer is specified further down (Figure 6.10).

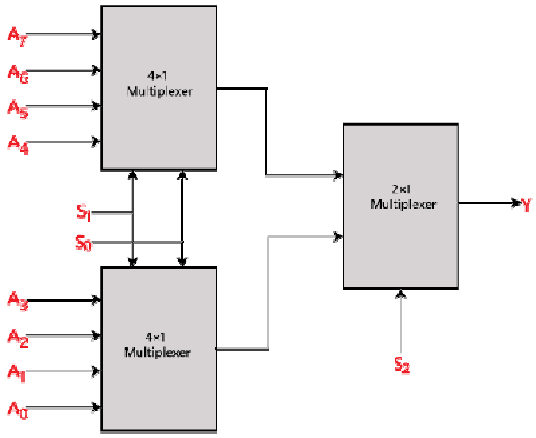


Figure 6.10: Represent 8*1 MUX

16x1 Multiplexer: In the 16 to 1 multiplexer, there is an overall of sixteen inputs, A_0, A_1, …, A_{16}, 4 selection lines S_0, S_1, S_2, and S_3, and a single output Y (Figure 6.11). Based on the arrangement of inputs that are current at the selection lines S_0, S_1, and S_2, one of these sixteen inputs will be coupled to the output. The block diagram of the 16×1 is specified as follows (Table 6.7):

Table 6.7: Truth Table of 16x1 MUX.

S_0	S_1	S_2	S_3	Y
0	0	0	0	A_0
0	0	0	1	A_1

0	0	1	0	A_2
0	0	1	1	A_3
0	1	0	0	A_4
0	1	0	1	A_5
0	1	1	0	A_6
0	1	1	1	A_7
1	0	0	0	A_8
1	0	0	1	A_9
1	0	1	0	A_{10}
1	0	1	1	A_{11}
1	1	0	0	A_{12}
1	1	0	1	A_{13}
1	1	1	0	A_{14}
1	1	1	1	A_{15}

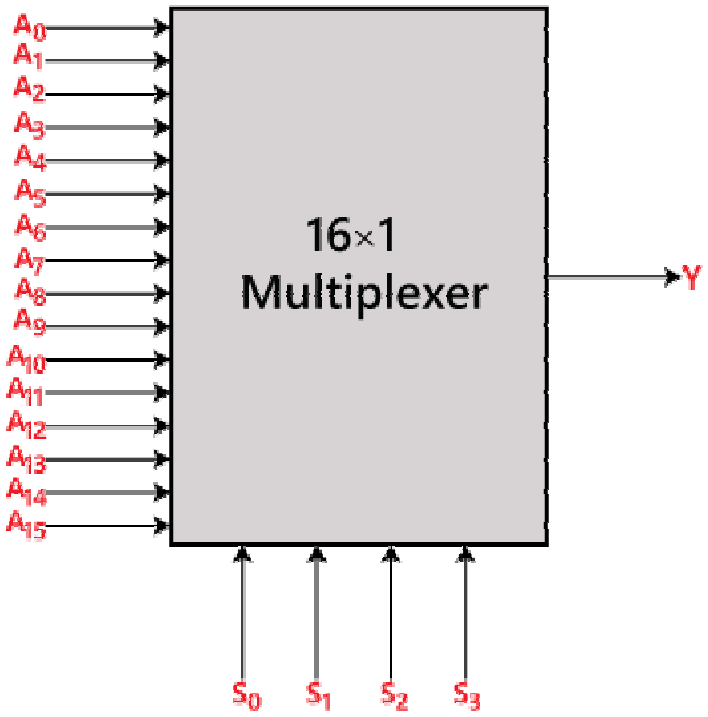


Figure 6.11: 16x1 MUX

16x1 mux using 8x1 mux and 2x1 mux: We can make the 16×1 multiplexer utilizing a lower order multiplexer. To make the 8×1 multiplexer, we require two 8×1 multiplexers and one 2×1 multiplexer. The 8×1 multiplexer takes three selection lines, four inputs, and one output. The 2×1 multiplexer has single selection input (Figure 6.12).

Designed for the accomplishment of sixteen data inputs, we require two 8 ×1 multiplexers. The 8×1 multiplexer generates one output. Therefore, in the direction to develop the final output, we require a 2×1 multiplexer. The block diagram of the 16×1 multiplexer using 8×1 and 2×1 multiplexer is specified as follows:

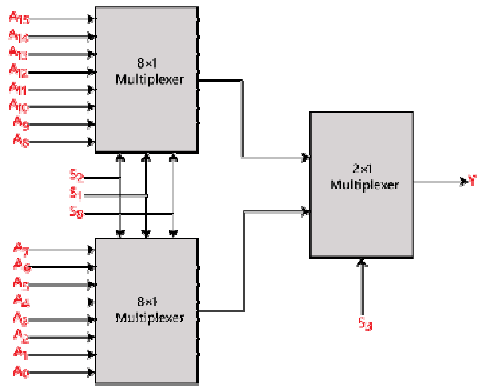


Figure 6.12: Represent the 16*1 MUX using 2*1 MUX.

Demultiplexer:A De-multiplexer executes the opposite process of a multiplexer. The de-multiplexer has a 1 input, which is dispersed over numerous outputs. One output line is designated at a time by picking lines. The input is communicated to the designated output line.

A De-multiplexer is a combinational circuit that proceeds only with a single input line and 2^N output lines. Just, the multiplexer is a single-input and multi-output combinational circuit. The information is conventional from the single input lines and concentrated on the output line. At the beginning of the inputs of the selection lines, the input will be associated with one of these outputs. De-multiplexer is the reverse of the multiplexer. Dissimilar encoder and decoder, here are n selection lines and 2^n outputs. So, there is a total of 2^n conceivable arrangements of inputs. De-multiplexer is correspondingly dried as **De-mux**[7]–[10].

The multiple types of demultiplexers are discussed below:

1x2 Demultiplexer: In the 1 to 2 De-multiplexer, there are simply 2 outputs Y_0, and Y_1, Single selection lines, S_0, and single input A. Based on the selected value, the input will be associated with one of the outputs. The block diagram of the 1×2 multiplexer is specified as follows (Table 6.8):

Table 6.8: Truth Table of 1x2 DEMUX

S_0	Y_0	Y_1
0	A	0
1	0	A

The Boolean expression for the output Y :

$$Y_0=S_0'.A$$
$$Y_1=S_0.A$$

1x4 DEMUX: In 1 to 4 De-multiplexer, here are the overall of 4 outputs, Y_0, Y_1, Y_2, and Y_3, two selection lines S_0 and S_1, and single input A. Based on the arrangement of inputs which are existing at the selection lines S_0 and S_1, the input is joined to one of the outputs (Figure 6.13). The block diagram of the 1×4 multiplexer is specified as follows (Table 6.9):

Table 6.9: Truth Table of 1x4 DEMUX

S_0	S_1	Y_0	Y_1	Y_2	Y_3
0	0	A	0	0	0
1	0	0	A	0	0
0	1	0	0	A	0
1	1	0	0	0	A

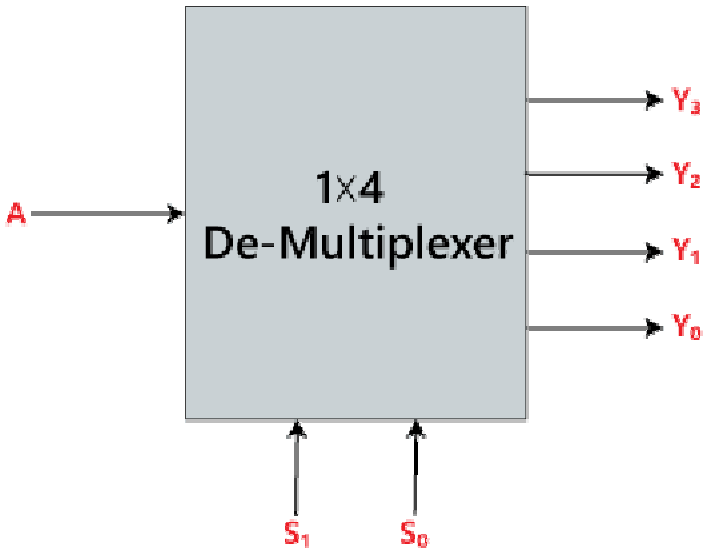


Figure 6.13: 1x4DEMUX

The Boolean expression for the 1x4 demux is expressed below:

$Y_0=S_1' S_0' A$
$Y_1=S_1' S_0 A$
$Y_2=S_1 S_0' A$
$Y_3=S_1 S_0 A$

1x8 Demux: In 1x 8 De-multiplexer, there are overall of 8 outputs, Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7, three selection lines, S_0, $S_1$and S_2 and one input A. Based on the arrangement of inputs that are existing at the selection lines S_0, S_1, and S_2, the input will be associated with one of these outputs (Table 6.10).

The Boolean expression for 1x8 demux:

$Y_0=S_0'.S_1'.S_2'.A$
$Y_1=S_0.S_1'.S_2'.A$
$Y_2=S_0'.S_1.S_2'.A$
$Y_3=S_0.S_1.S_2'.A$
$Y_4=S_0'.S_1'.S_2\ A$
$Y_5=S_0.S_1'.S_2\ A$
$Y_6=S_0'.S_1.S_2\ A$
$Y_7=S_0.S_1.S_3.A$

Table 6.10: Truth Table of 1x8 DEMUX

S_0	S_1	S_2	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	A	0	0	0	0	0	0	0
1	0	0	0	A	0	0	0	0	0	0
0	1	0	0	0	A	0	0	0	0	0
1	1	0	0	0	0	A	0	0	0	0
0	0	1	0	0	0	0	A	0	0	0
1	0	1	0	0	0	0	0	A	0	0
0	1	1	0	0	0	0	0	0	A	0
1	1	1	0	0	0	0	0	0	0	A

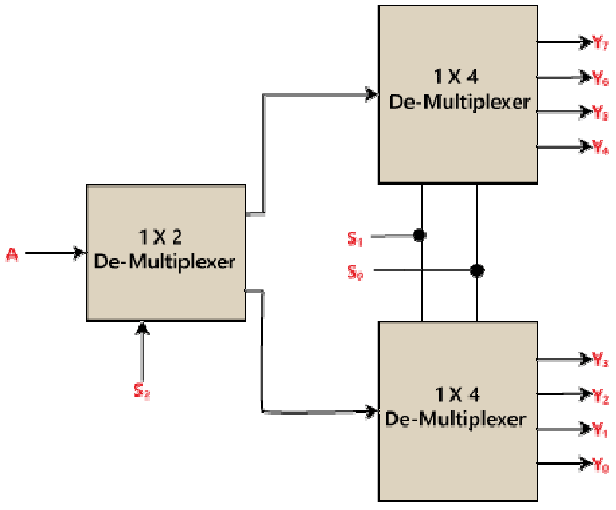


Figure 6.14: 1x8 Demux using 1x4 Demux and 1x2 Demux

1x8 Demux using 1x4 Demux and 1x2 Demux: We could make the 1×8 de-multiplexer using a lower-order de-multiplexer. To make the 1×8 de-multiplexer, we require 2, 1×4 de-multiplexer, and 1, 1×2 de-multiplexer. The 1×4 multiplexer possesses two selection lines, four outputs, and one input. The 1×2 de-multiplexer ensures individual one selection line. Designed for the

accomplishment of eight data outputs, we require a 2, 1×4 de-multiplexer. The 1×2 de-multiplexer generates 2 outputs. So, to become the final output, we must permit the outputs of the 1×2 de-multiplexer as an input of both the 1×4 de-multiplexer. The block diagram of the 1×8 de-multiplexer using the 1×4 and 1×2 de-multiplexer is specified as follows (Figure 6.14):

16x1 Demultiplexer: In 1×16 de-multiplexer, there are entire of sixteen outputs, Y_0, Y_1, …, Y_{16}, four selection lines, S_0, S_1, S_2, and S_3 and only one input A. Based on the arrangement of inputs which are existing at the selection lines S_0, S_1, and S_2, the input will be coupled to one of these outputs. The block diagram of the 1×16 de-multiplexer is specified as follows (Table 6.11):

Table 6.11: Truth Table of 1x16 DEMUX

S_0	S_1	S_2	S_3	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}
0	0	0	0	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	A	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	A	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	A	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	A	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	A	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	A	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	A	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	A	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	A	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	A	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	A	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	A	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A

1x16 Demux using 1x8 Demux and 1x2 Demux: We could make the 1×16 de-multiplexer by implementing a lower-order de-multiplexer. To make the 1×16 de-multiplexer, we required 2, 1×8 de-multiplexer, and a single 1×2 de-multiplexer. The 1×8 multiplexer has three selection lines, single input, and eight outputs. The 1×2 de-multiplexer has only one selection line. For the accomplishment of sixteen data outputs, we required a 2, 1×8 de-multiplexer. The 1×8 de-multiplexer generates 8 outputs. Therefore, to acquire the last output, we required a single 1×2

de-multiplexer to generate 2 outputs from one input. Formerly we permit these outputs into mutually the de-multiplexer as an input.

In this chapter, we studied the encoder and decoder. We also learn about multiplexer, demultiplexer and their types, block diagram, truth tables, and their Boolean expression for outputs, and also study the logical operation diagram. We also learn about how to make large mux and demux using a combination of small mux and demux.

REFERENCES

[1] R. Kaler and R. S. Kaler, "Implentation of optical encoder and multiplexer using Mach-Zehnder inferometer," *Optik (Stuttg).*, 2011, doi: 10.1016/j.ijleo.2010.09.030.

[2] A. Tandon *et al.*, "Demonstration of Arithmetic Calculations by DNA Tile-Based Algorithmic Self-Assembly," *ACS Nano*, 2020, doi: 10.1021/acsnano.0c01387.

[3] S. V. Saravanan, R. S. Ganesh, and S. Sasipriya, "Design and performance analysis of low power 4 bit flash analog to digital converter," *J. Adv. Res. Dyn. Control Syst.*, 2017.

[4] M. Damghanian and S. J. Azhari, "A low-power 6-bit MOS CML flash ADC with a novel multi-segment encoder for UWB applications," *Integr. VLSI J.*, 2017, doi: 10.1016/j.vlsi.2017.01.006.

[5] M. A. Nabulsi and M. A. F. Al-Husainy, "Using combinational circuits for control purposes," *J. Comput. Sci.*, 2009, doi: 10.3844/jcssp.2009.507.510.

[6] M. Ignacio, M. Ortega, J. Wysowski, and A. A. Borodzhieva, "Designing an Interactive Multimedia Bilingual Application for the Course 'Pulse and Digital Devices,'" *Proc. Univ. Ruse*, 2019.

[7] S. Z. M. Muji, M. H. A. Wahab, R. Ambar, and W. K. Loo, "Design and implementation of electronic chess set," in *2016 International Conference on Advances in Electrical, Electronic and Systems Engineering, ICAEES 2016*, 2017. doi: 10.1109/ICAEES.2016.7888087.

[8] J. Andréasson and U. Pischel, "Smart molecules at work—mimicking advanced logic operations," *Chem. Soc. Rev.*, 2010, doi: 10.1039/b820280j.

[9] H. Li, Y. Liu, S. Dong, and E. Wang, "DNA-based advanced logic circuits for nonarithmetic information processing," *NPG Asia Mater.*, 2015, doi: 10.1038/am.2015.16.

[10] M. D. Gupta and R. K. Chauhan, "Design of an efficient parallel comparator architecture for low power delay product," *Adv. Electr. Electron. Eng.*, 2021, doi: 10.15598/aeee.v19i2.4101.

CHAPTER 7

SEQUENTIAL CIRCUITS

Mrs. Manaswini R, Assistant Professor,
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India,
Email Id-manaswini.r@presidencyuniversity.in

In the previous chapter, we study the different types of combinational circuit and their working. The combinational circuit has a set of outputs dependent ON the present input only.in this chapter, we will study sequential circuits and their working. The sequential circuit is defined as "the circuits which have a series of inputs and outputs. The output of these circuits depends ON the present input as well as the previous output. So sequential circuit is made from the arrangement of the combinational circuit with the memory element (Figure 7.1). The memory element is used to acquire the previous output of the circuit".

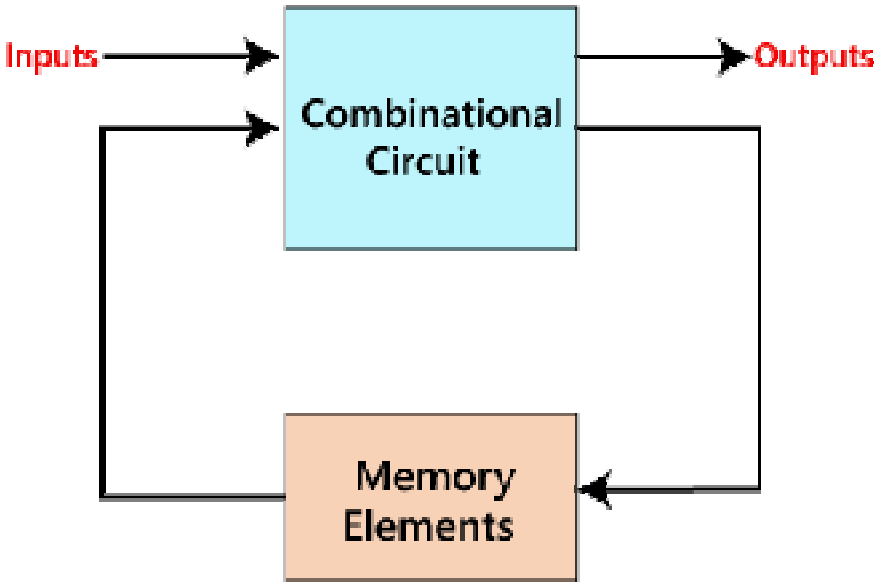


Figure 7.1: Sequential Circuit

Difference between Combinational and Sequential circuits (Table 7.1):

Table 7.1: Difference between Combinational and Sequential circuits

Combinational circuit	**Sequential circuit**
The output of these circuits depends on the present inputs only.	The output of the sequential circuit depends on the present input as well as the previous output.
In these circuits, no memory element is provided.	The memory element is provided in the sequential circuit to acquire the previous output.

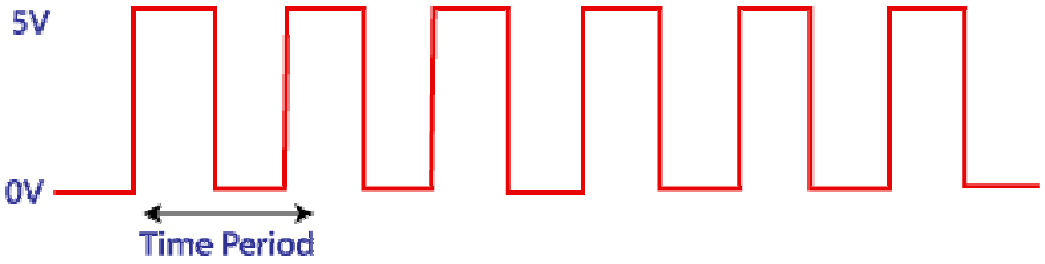
The feedback system is not provided in these circuits.	These circuits used the feedback paths for operating the circuits.
The clock signal is not required in these circuits.	The clock signal is used to operate the sequential circuits.
These circuits are quite simple to design.	It is difficult to design sequential circuits.

Different types of sequential circuits: There are mostly two types of sequential circuits given below:

Asynchronous sequential circuits: In these circuits, the clock signals are not utilized. These circuits operate through the pulses. So, the variation in the inputs could be changed by the state-run of the circuit. These circuit doesn't employ the clock pulses. The internal state is varied when the input variable is varied. The un-clocked flip-flops and time delay are the memory elements of these circuits. These circuits are similar to the combinational circuit along with the feedback system.

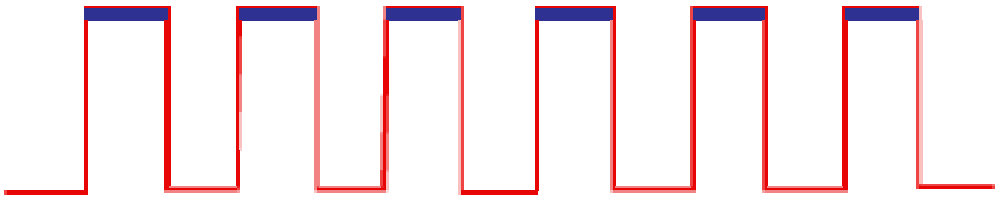
Synchronous sequential circuits: in these circuits, synchronization of the memory elements phase is obtained by the clock signal. The output is deposited in the memory devices. The synchronization of the output is obtained by only the negative edges of the clock signal or only the positive edges[1]–[3].

Clock signal: A clock signal is an episodic signal in which ONN time and OF time requirements were not similar. When the ONN time and OF time of the clock signal are similar, a rectangular wave is cast-off to signify the clock signal. Underneath is a figure which signifies the clock signal. "A clock signal is measured by way of the rectangular wave. Occasionally, the signal breaks at logic, moreover high 5V or low 0V, to an equal volume of time. It recaps through a certain time, which will remain equivalent to double the 'ONN' time and 'OF' time".

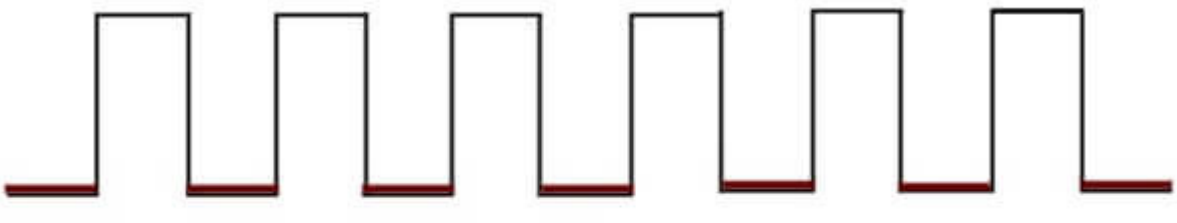


Level triggers: The logic High and Low are the two levels in the clock signal. In level triggering, once the clock pulse is ONN at a certain level, simply formerly the circuit is triggered. Different types of level triggered are specified as follows:

Positive level triggering: positive level triggering is the signal with Logic High occurs. in positive-level triggering, the circuit is worked through a different type of clock signal. Lower the figure shows the positive level triggering:



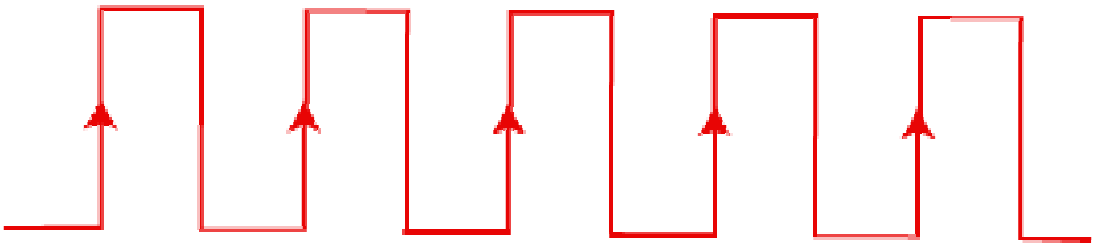
Negative level triggering: negative level triggering is the signal through Logic Low happens. in negative-level triggering, the circuit functiONed through a different type of clock signal. Underneath is the figure shows the Negative level triggering:



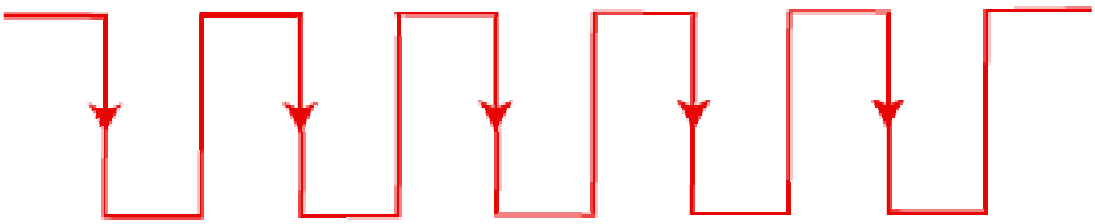
Edge triggering: In edge triggering of the clock signal the two kinds of transitions take place. one transitiON for Logic Low to Logic High and the second transitiON for Logic High to Logic Low.

Due to the transitiON of triggering, edge triggering is subdivided into two which are specified beneath:

Positive edge triggering: The positive edge triggering is the transitiON from Logic Low to Logic High that happens in the clock signal. in this type of triggering the circuit is worked through such kind of clock signal. The figure of positive edge triggering is specified under.



Negative edge triggering: The negative edge triggering is the transitiON from Logic High to Logic low that happens in the clock signal. in this triggering, the circuit is worked through the category of the clock signal. The figure for negative edge triggering is specified underneath.



Flip-flops: flip-flop is a circuit that involves two steady conditions and is preserved as a **flip-flop**. These steady cONditiONs are cast-off to collect binary data that can be reformed by spreading over changing inputs. flip flops are the important building blocks of the digital system. Flip flops and latches are samples of pieces of information storing components. the sequential logic circuit is used the flip flop as the elementary storing component. The latches and flip flops are elementary storing compONents but dissimilar in operatiON. There are the succeeding kinds of flip-flops are given as follows[4]–[6]:

S-R flip flop: The S-R flip flop is the basic type of flip flop that is employed in the digital system. The operation of SR flip flop is described as " when the set input "S" is true, the output Y will be high, and Y' will be low" (Table 7.2). It is essential that the cabling of the circuit is preserved ONce the outputs are recognized. We uphold the cabling till set or reset input drives high, or power is closed (Figure 7.2).

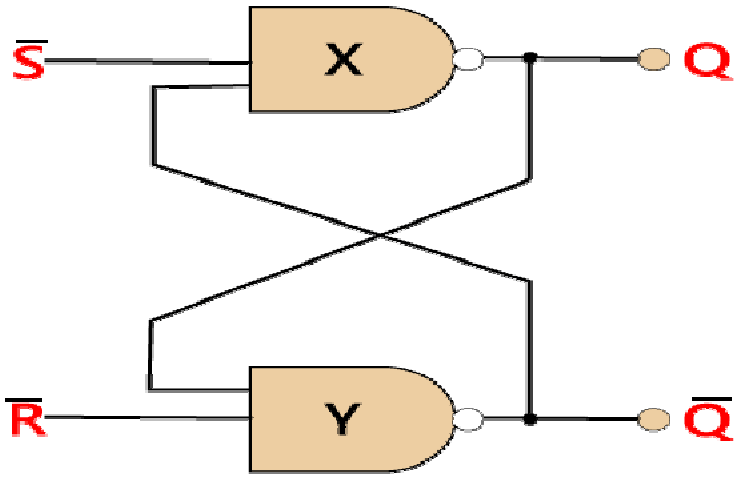


Figure 7.2: S-R flip flop

Table 7.2: Truth Table of S-R flip flop

S	R	Y	Y'
0	0	0	1
0	1	0	1
1	0	1	0
1	1	∞	∞

J-K flip-flops: The JK flip flop is defined as the flip flops, which is employed to eliminate the problem of the S-R flip flop such as indeterminate conditions (Table 7.3). The JK flip flop is designed through achievement alteration in the SR flip flop. The S-R flip flop is enhanced by the concept of the J-K flip flop. The SR flip flop provides an imprecise result Once S and R input is established to be true. Then in the situation of JK flip flop, it stretches the correct output. If in J-K flip-flop inputs are dissimilar, then the value of J at the succeeding clock edge is occupied by the output Y. If the j-k flip-flop input is low, then no variation happens, and if the input is high

ON the clock edge, then from one state to the additional, the output will be toggled. This flip-flop is utilize a Set or Reset Flip flop in the digital system (Figure 7.3).

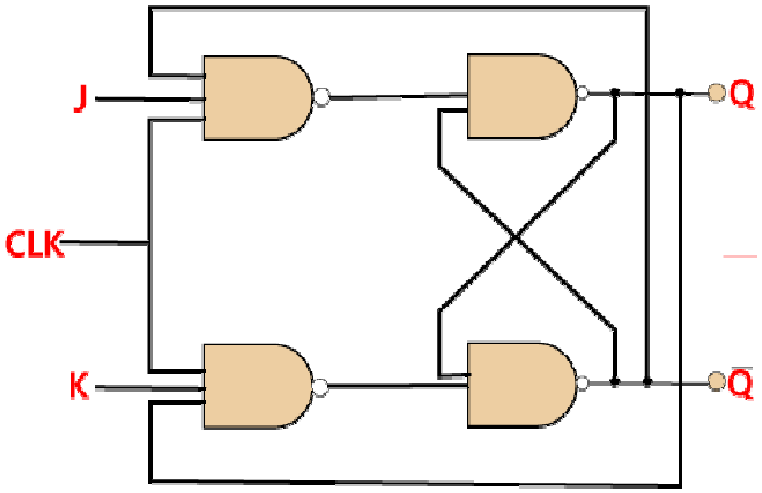


Figure 7.3: JK FLIP FLOP

Table 7.3: Truth table of J -K flip flop

J	K	Y	Y'
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	1
0	1	1	0
1	0	1	1
1	1	1	0

D flip flop: A D flip flop is defined as "A digital electronic circuit cast-off to delay the variation of the state of its output signal till the succeeding growing edge of a clock timing input signal happens" (Figure 7.4).D flip flop is frequently utilized for counters, input synchronization, and shift registers (Table 7.4).

Table 7.4: Truth Table of D flip flop

Clock	D	Y	Y'
↓ » 0	0	0	1
↑ » 1	0	0	1
↓ » 0	1	0	1
↑ » 1	1	1	0

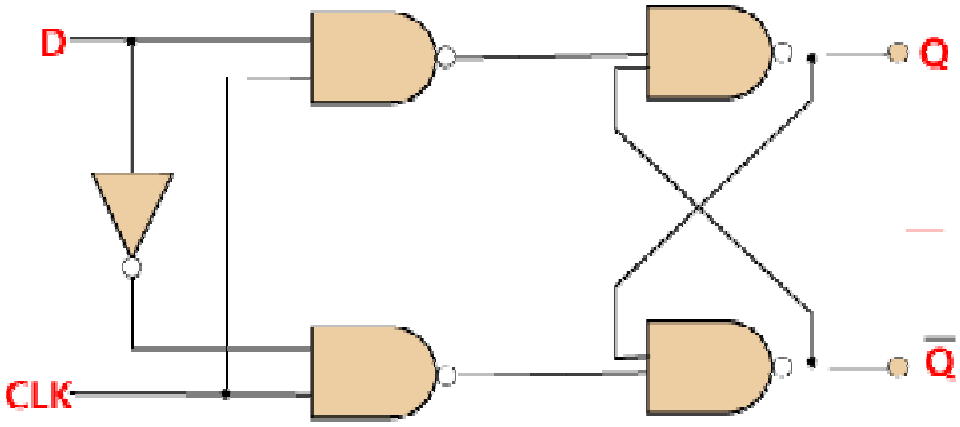


Figure 7.4: D flip flop

T flip flop: T flip flop is a modification of the JK flip-flop. The T flip-flop is established by connecting mutual inputs of a JK flip-flop. The T flip-flop is expected by connecting the inputs 'J' and 'K'. ONce T = zero, mutually AND gates are restricted (Figure 7.5).T flip flop is individual single input through the clock input. The T flip flop is erected by connecting the inputs of the JK flip flop composed as a single input. T flip flop is also called toggle flip flop (Table 7.5).

Table 7.5: **Truth Table of T-flip flop**

T	Y	Y (t+1)
0	0	0
1	0	1
0	1	1
1	1	0

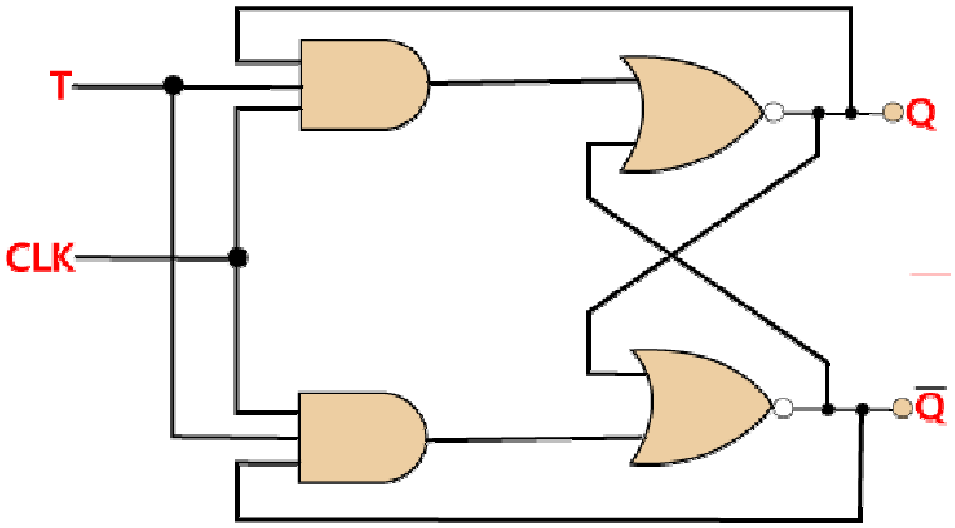


Figure 7.5: T flip flop

Master Slave flip flop: The Master-Slave Flip-Flop is defined as "A fundamentally an arrangement of two JK flip-flops associated composed in a series formation. One performance as the "master" and the other as a "slave"**.** The output of master flip flop is associated to the 2 inputs of the slave flip flop whose output is served back to inputs of the master flip flop". In the calculation of these two flip-flops, the circuit also comprises an inverter**.** The inverter is associated through the clock pulse in such a technique that the inverted clock pulse is assumed to be the slave flip-flop. In other words, if CP= zero for a master flip-flop, then CP= one for a slave flip-flop, and if CP= one for a master flip-flop at that time it converts zero for a slave flip-flop (Figure 7.6).

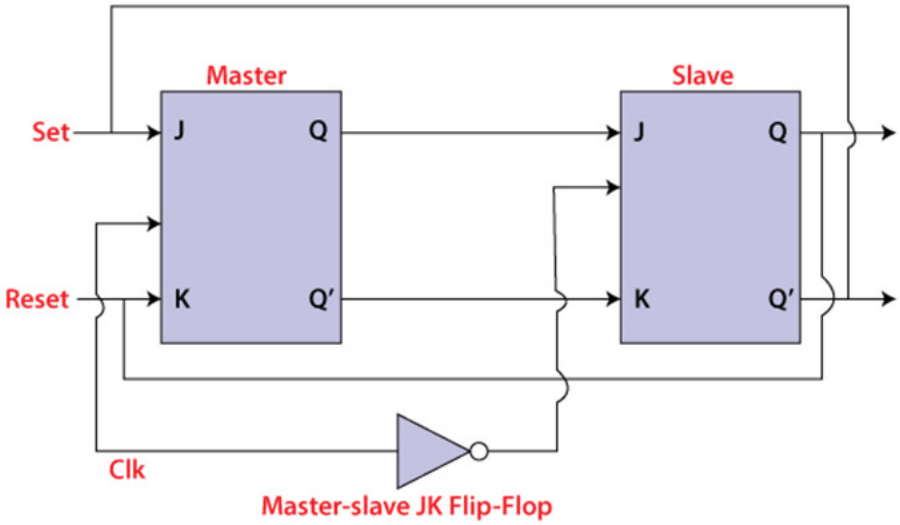


Figure 7.6: Master slave flip flop

Working of master-slave flip flops:

1. Once the clock pulse is true, the slave flip flop will stay in the inaccessible state, and the system's state might be pretentious by the J and K inputs. The "slave" ruins are inaccessible till the CP is 1. After the CP is set to 0, the master flip-flop permits the data to the slave flip-flop to get the output.
2. The master flip-flop reacts firstly by the slave since the master flip-flop is the positive level trigger, and the slave flip-flop is in the negative level trigger.
3. The output Q'=one of the master flip flop is accepted to the slave flip flop by way of an input K ONce the input J is set to 0 and K set to 1. The clock empowers the slave flip-flop to work as rearranged, and formerly the slave duplicates the master flip-flop[7]–[10].
4. When J= one, and K= zero, the output Q= one is approved to the J input of the slave. The clock's negative transitiON arrangements the slave and replicas the master.
5. The master flip-flop toggles arranged the clock's positive transitiON after the inputs J and K are fixed to 1. After, the slave flip-flop closures arranged the clock's negative transitiON.
6. The flip-flop will remain restricted, and Q rests unaffected ONce separately the inputs of the JK flip-flop are fixed to 0.

Timing diagram:

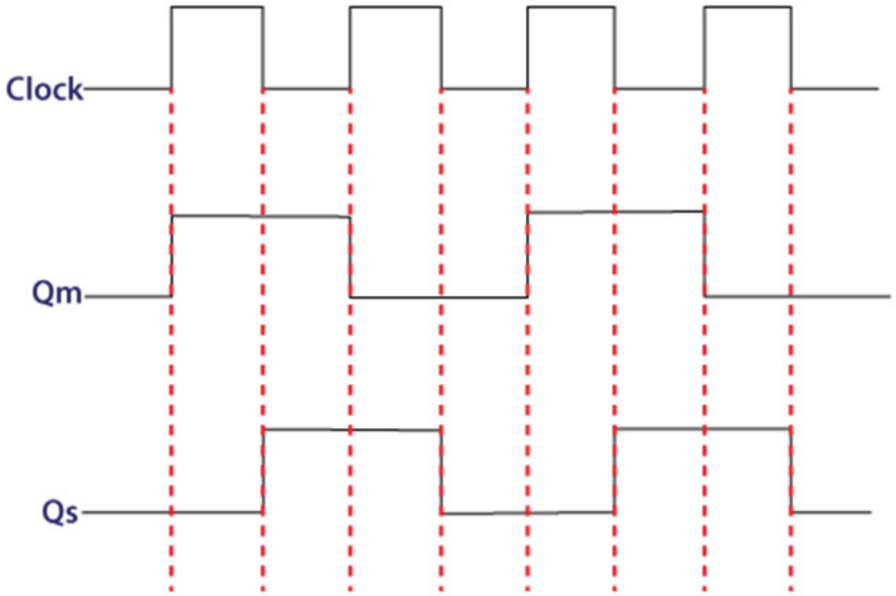


Figure 7.7: Shows the timing diagram flip flop

- After the clock pulse is fixed to one, the output of the master flip-flop will stay at 1 till the clock input rests at zero.
- Once the clock pulse comes to be 1 over, the master's output is at zero, which will remain fixed to one ONce the clock develops 1 yet again.
- The flip-flop is working once the clock pulse is at ONe. The slave's output rests at zero till the clock is not fixed to zero since the slave flip-flop is not working.
- The flip-flop is working after the clock pulse is at 0. The output of the master rests at 1 till the clock is not fixed to zero over again.
- Toggling arises throughout the whole procedure since the output deviations when in the cycle.

In this chapter, we study the sequential circuit and their working, their different types. We study the different types of flip flops, their truth table, and their block diagram. At last in this chapter, we learn the master-slave flips flops and their working or clock signal timing diagram.

REFERENCES

[1] C. A. Kumar, B. K. Madhavi, and K. L. Kishore, "Enhanced Clock Gating Technique for Power Optimization in SRAM and Sequential Circuit," *J. Autom. Mob. Robot. Intell. Syst.*, 2021, doi: 10.14313/JAMRIS/2-2021/11.

[2] J. L. Massey and M. K. Sain, "Inverses of Linear Sequential Circuits," *IEEE Trans. Comput.*, 1968, doi: 10.1109/TC.1968.229392.

[3] V. Keerthy Rai and R. Sakthivel, "Designs protracted to combinational and sequential circuits by using hybrid MOS transistor with memristor," *Int. J. Adv. Technol. Eng. Explor.*, 2021, doi: 10.19101/IJATEE.2021.874470.

[4] G. H. Mealy, "A Method for Synthesizing Sequential Circuits," *Bell Syst. Tech. J.*, 1955, doi: 10.1002/j.1538-7305.1955.tb03788.x.

[5] R. Saito, C. L. Ayala, O. Chen, T. Tanaka, T. Tamura, and N. Yoshikawa, "Logic synthesis of sequential logic circuits for adiabatic quantum-flux-parametron logic," *IEEE Trans. Appl. Supercond.*, 2021, doi: 10.1109/TASC.2021.3061636.

[6] A. L. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2001, doi: 10.1109/43.945306.

[7] Y. Yuan *et al.*, "Thin Film Sequential Circuits: Flip-Flops and a Counter Based on p-SnO and n-InGaZnO," *IEEE Electron Device Lett.*, 2021, doi: 10.1109/LED.2020.3038223.

[8] L. E. Yu, C. Shin, S. Paik, J. J. Liou, and Y. Shin, "Sampling correlation sources for timing yield analysis of sequential circuits with clock networks," *J. Circuits, Syst. Comput.*, 2011, doi: 10.1142/S0218126611008043.

[9] L. M. Zhang, Z. W. Yang, Y. K. Pang, T. Zhou, C. Zhang, and Z. L. Wang, "Tribotronic triggers and sequential logic circuits," *Nano Res.*, 2017, doi: 10.1007/s12274-017-1564-9.

[10] C. Lou *et al.*, "Synthesizing a novel genetic sequential logic circuit: A push-on push-off switch," *Mol. Syst. Biol.*, 2010, doi: 10.1038/msb.2010.2.

CHAPTER 8

REGISTERS AND COUNTERS

Mrs. Aruna Dore, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id-aruna.dore@presidencyuniversity.in

In the previous chapter, we study the sequential circuits such as flip flops (SR flip flop, JK flip flop, D flip flop, and T flip flop) and their workings. In this chapter, we will study the register and counters which are implemented by utilizing the combination of flip flops.

Counter: counter is a device that stores and occasionally shows the number of times a specific occasion or technique has happened. Frequently in connection to a clock. A counter circuit is typically built of several flip-flops associated with a cascade. The counter is a superior kind of sequential circuit that is utilized to tally the pulse is called counter. An assembly of several flip-flops which is operated on the clock signal is known as counters. The counter is an extensive application of flipflops on the base of the clock pulse, the output of a counter comprises a predefined state. The value of pulses can be counted by employing the output of the counter (Table 8.1)[1]–[3].

Table 8.1: Counter response

Clock	Counter output		State number	Decimal counter output
	Q_B	Q_A		
Initially	0	0	-	0
1st	0	1	1	1
2nd	1	0	2	2
3rd	1	1	3	3
4th	0	0	4	0

The counters are subdivided into two parts:

1. Asynchronous counter
2. Synchronous counter

Asynchronous counter: This counter is also called the Ripple counter. A two-bit asynchronous counter is implemented by utilizing two T- FF (Flip-Flops). In addition, we also implement a 2-bit asynchronous counter by using the combination of two jk flip flops. The input of both asynchronous counters is fixed to logic 1 (Figure 8.1).

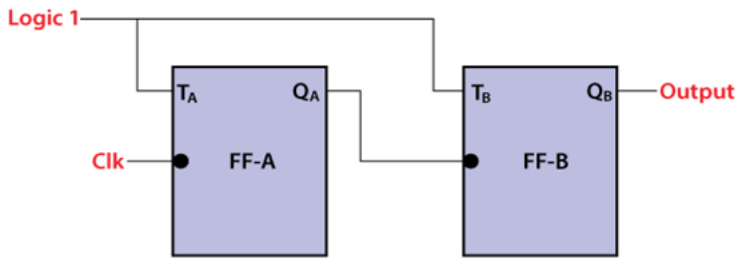


Figure 8.1: Asynchronous counter

The clock signal diagram of the asynchronous counter is specified below (Figure 8.2):

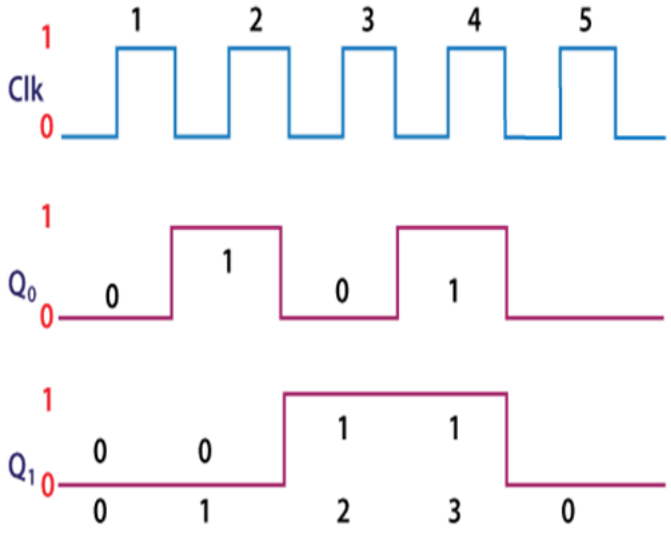


Figure 8.2: Signal Diagram of the Asynchronous Counter

Synchronous counter:the counters are associated with similar a sequence in the asynchronous counter because the current counter's output permits the input of the succeeding counter. It creates a counting delay, and the propagation delay also occurs during the counting stage is the major drawback of the asynchronous counter. For dealing with this disadvantage a synchronous counter is designed. The same clock pulse is accepted to the clock input of all the flip flops in the synchronous counter. All the flip-flops shaped by the clock signals are similar to respectively other. A 2-bit synchronous counter is specified below in which the inputs of the first flip flop(FF-A) are fixed to one. The FF-A (first flip-flop) will operate as a toggle flip-flop. The FF-A output is agreed to mutually the inputs of the next JK flip flop.

The clock signal diagram is given below in Figure 8.3:

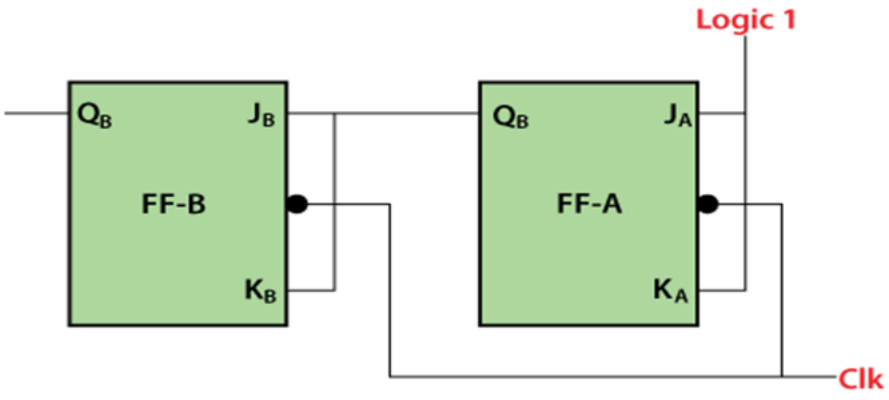


Figure 8.3: Synchronous counter

Ripple counter: Asynchronous counter is also known as a ripple counter in which the clock pulse ripples over the circuit. By joining n number of flip-flops the n-MOD ripple counter

implements. The limitation of the n-MOD ripple counter, it can calculate 2n states, and at that time the counter changes to its primary value. The different features of the ripple counter are given below:

1. Dissimilar types of ff(flip-flops) along with unlike clock pulses are utilized.
2. A ripple counter is a special type of asynchronous counter.
3. The flip flops are cast off in toggle mode.
4. The peripheral clock pulse is functional to a single flip-flop. The flip-flop output is preserved through a clock pulse designed for the succeeding flip-flop.
5. The flip flop in which the external clock pulse is accepted performs as lsb(least significant bit) in the counting sequence.
6. Up counter, down counter, and up-down counter is basic types of a ripple counter[4]–[6].

Binary ripple counter: A 2-Mod counter which calculates up to 2-bit state values, (22 = 4 values) is called a binary ripple counter. The binary ripple counter flip flops having a comparable state of affairs for toggling corresponding T and JK is cast-off to create the Ripple counter. The circuit diagram of a binary ripple counter is specified below. The binary ripple counter is considered by employing two JK flip flops. The inputs of mutual flip-flops are approved for the high-voltage signal. The input of a high voltage signalupholds the flip flops at a state first (Figure 8.4). The negative generated clock pulse utilized by the jkflip flops. The least significant bit (LSB) and most significant bit(MSB) bits are the outputs Q_0 and Q_1, respectively. Understanding the functioning of the counter of jk flip flop is done through the truth Table 8.2.

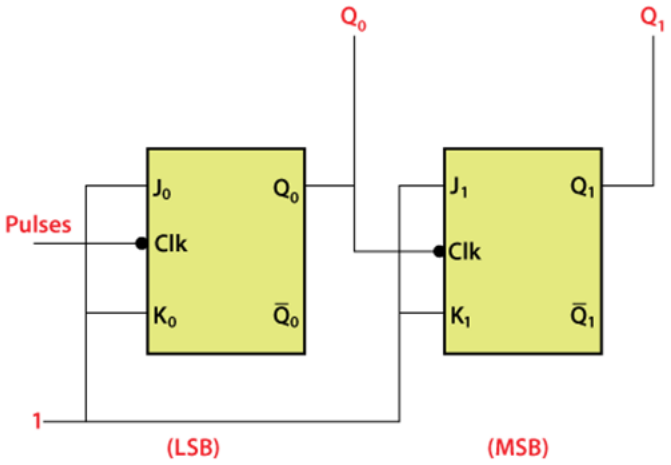


Figure 8.4: Binary ripple counter

Table 8.2: Binary ripple counter

J_n	K_n	Qn+1
0	0	Q_n
1	0	1
0	1	0
1	1	Q_n

The inputs of the flip flops are functional to the high voltage, the JK flip flop occurs in the fourth condition. Once we apply high voltage to the input of the flip-flop, the flip-flops will stay on state first. The negative going end of the clock pulse transferred the states of the flip flops permissions. It means that the flip flop toggled once the clock pulse transition takings place between one to zero after the negative clock edge permits the flip flop, the condition of the output Q_0 variation. The whole of the flip-flops is fixed to zero initially. Once the approved clock varied from 1 to 0, These flip-flops varied their conditions once the inputs of the flip flops are 1, the JK flip flop toggles, and formerly the flip flop variations its conditions from zero to one. The procedure rests similarly for all the clock pulsates.

The second flip flop as a clock pulsation permits the output of the first flip-flop. The position of the second flip-flop is different once the output Q_0 drives the transition from one to zero, from the timing diagram below. LSB and MSB represent the outputs Q_0 and Q_1. The values 00, 01, 10, and 11 are calculated by the counter. The counter rearranges itself, once counting these values, and jumps counting yet once more from 00, 01, 10, and 1. The clock pulsates are approved to J_0K_0 flip flop, till the counter calculates values[7]–[10].

Ring counter: A ring counter is worked on the principle of the SISO (**Serial IN Serial OUT)** Shift register. The ring counter is practically similar to the shift counter. The individual modification between the shift register and the ring counter is that the latest flip-flop output is taken as the output in the case of the shift register, but in the case of the ring counter, this output is connected to the first flip-flop as an input. Except this, all of the outstanding things in the ring counter are the equivalent of the shift register. The Number of states in the ring counter is equal to the number of flip-flops is employing (Figure 8.5).

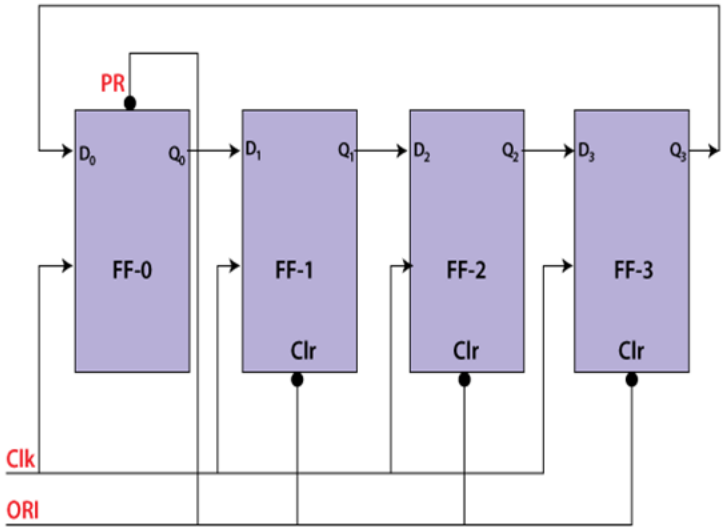


Figure 8.5: Ring counter

Once the preset is fixed to zero, the outcome is one. Once the clear is fixed to zero, the outcome is zero. Both PR and CLR constantly exert in value zero since they are lively low signals.

PR = 0, Q = 1

CLR = 0, Q = 0

Johnson counter: The Johnson counter is like the Ring counter. The individual alteration between the Johnson counter and the ring counter is that the output of the latter flip-flop is approved to the initial flip-flop through response. Nevertheless, the reversed output Q' of the latest flip-flop is accepted as an input in Johnson's counter. The outstanding effort of the Johnson counter is similar by way of a ring counter. The Johnson counter is correspondingly stated to as the Creeping counter (Figure 8.6).

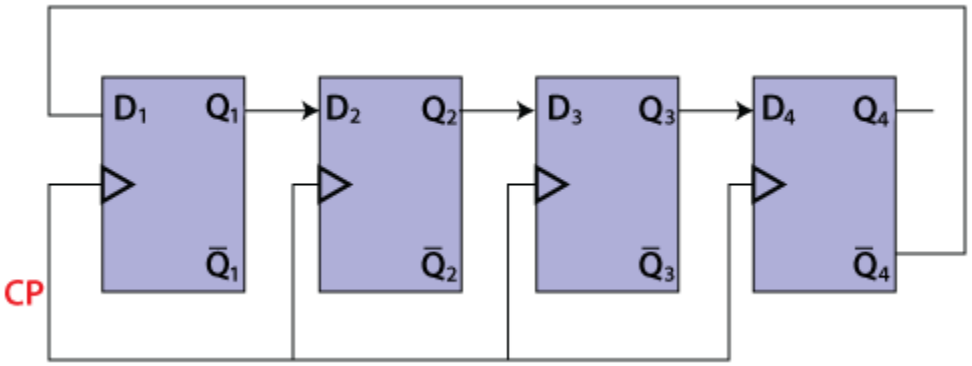


Figure 8.6: Johnson counter

Number of states in Johnson counter = Number of flip-flop castoff

No. of castoff states =2n

No.of unemployed states =2n - 2*n

The clock signal diagram of the Johnson counter is specified below:

Advantage of Johnson counter:

1. The NO. Of flip flops in the Johnson counter is equivalent to the NO. Of flip flops in the ring counter, and the Johnson counter counts double the Number of conditions the ring counter can tally.
2. The Johnson counter could similarly be considered through D or JK flip flop.
3. The pieces of information are tallied in an endless loop in the Johnson ring counter.
4. The route of the Johnson counter is auto-decoding.

A disadvantage of Johnson counter:

1. The Johnson counter is not used in the BCD number system because it doesn't count the conditions/states in the binary arrangement.
2. The number of unused states of a Johnson counter is greater than the number of states/ conditions used by the Johnson counter.
3. The NO. Of flip flops is equal to one-half of the NO. Of CLOCK signals.
4. The Johnson counter for different No. of a clock, a signal sequence is likely to design.

Registers: A Register is an assembly of flip-flops. To store the single-bit digital data, a flip flop is utilized. The storage ability is improved through a combination additional than one flip flop for storage of a bulky number of bits. We have to usage an n-bit register comprising n number of flip flops to collect an n-bit word. Dissimilar types of operations are executed by the registers,

and the CPU usages these registers for operating the operations. The registers will store the faded inputs to the system. The register will store the outcome returned by the system.

Fetch: To take the instructions given by the users and to fetch the instruction deposited in the key memory.

Decode: To interpret the instructions which are given by the users, the decode operation is executed. The CPU is identified as the operation executed on the instructions. In another way we can say that to decode the instructions from the users, the decode operation is used.

Execute: to store the result generates by the CPU in the memory, the execute operation is used. The stored result is displayed on the user's screen (Figure 8.7).

Different types of registers:

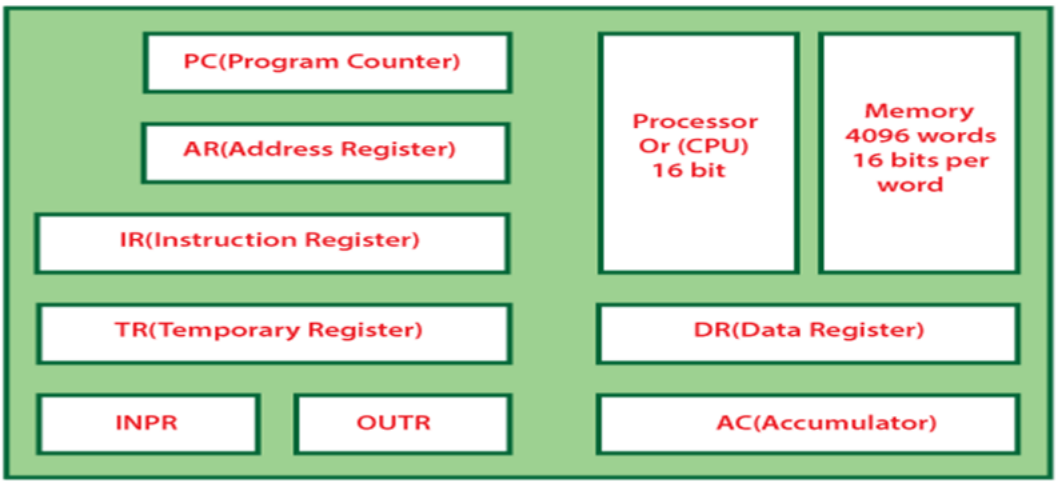


Figure 8.7: Types of Register

Memory address register (MAR): To comprise the memory address of the data and instruction, the memory register is used. To access instructions and data from memory in the execution phase, the MAR is used as the main task of the register. The data is to be read or to be stored by the CPU, and the address of that memory location is used in the MAR.

A program counter (PC): The PC is correspondingly known as an Instruction Pointer or Instruction Address register. The PC contained the succeeding memory address of the instruction, which is executed subsequently implementing the execution of existing instruction. The memory address of the location of the next instruction comprises or is carried out by the program counter.

Accumulator register: An Accumulator register is employed by the CPU mostly. The accumulator register is cast-off to collect the system result. Once the CPU generates some results for later processing, all the results will be stored in the accumulator register.

Data register: The data register is cast-off to briefly store the data. This data transmits to or from a peripheral device.

Memory buffer register: it is also called the MBR register. The MBR comprises the Metadata of the pieces of information and instructions inscribed in or recite since memory. In other words, it adds is cast-off to collect the coming data/instruction from the memory and go to memory.

Index register: The **Index Register** is the hardware component that stores the number. The number complements the computer instruction's address to generate an operative address. To modify the operand address during the running program the index register as a processor register is used in the CPU.

In this chapter, we study the register and counter or their different types. We also study the synchronous and asynchronous counter. We learn how to operate the ripple counter, binary ripple counter, Johnson counter, and their clock signal diagram and circuit diagram. At last, we discuss the program counter, data register, memory buffer register, and index register. In the next chapter, we will discuss the shift register.

REFERENCES

[1] P. T. Elektro *et al.*, "Pengembangan Media Pembelajaran Trainer Shift Register dan Counter Pada Mata Pelajaran Penerapan Rangkaian Elektronika di SMK Negeri 3 Surabaya," *J. Pendidik. Tek. Elektro*, 2017.

[2] P. Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo- Random Sequence Generators," *Xilinx*, 1996.

[3] R. Fitrianto, "Trainer Digital Register Dan Counter Sebagai Media Pembelajaran Untuk Mahasiswa Elektronika Komunikasi Di Jurusan Teknik Elektro Universitas Negeri Surabaya," *J. Pendidik. Tek. Elektro*, 2014.

[4] I. V. Savitskiy, "Computations on register machines with counters," *Discret. Math. Appl.*, 2018, doi: 10.1515/dma-2018-0010.

[5] D. Morrison, D. Delic, M. R. Yuce, and J. M. Redoute, "Multistage Linear Feedback Shift Register Counters with Reduced Decoding Logic in 130-nm CMOS for Large-Scale Array Applications," *IEEE Trans. Very Large Scale Integr. Syst.*, 2019, doi: 10.1109/TVLSI.2018.2872021.

[6] I. Mubaroq, W. Isna, I. W. Septiani, and A. C. Fauzan, "Pengalamatan Mikroprosesor 8086/8088 Menggunakan Operasi Aritmatika," *Briliant J. Ris. dan Konseptual*, 2019, doi: 10.28926/briliant.v4i3.342.

[7] J. Sundberg and C. Högset, "Voice source differences between falsetto and modal registers in counter tenors, tenors and baritones," *Logop. Phoniatr. Vocology*, 2001, doi: 10.1080/140154301300109107.

[8] O. Bishop and O. Bishop, "Counters and registers," in *Electronics - Circuits and Systems*, 2020. doi: 10.4324/9780080942346-28.

[9] H. Oberdiek, "The colonequals package," *Equals*, 2006.

[10] H. Oberdiek, "The aliascnt package," *Syntax*, 2006.

CHAPTER 9

SHIFT REGISTERS

Mrs. Sowmya C S, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id-sowmya.cs@presidencyuniversity.in

The collection or combination of flip flops that are cast off to accumulate several bits of data and the data is stimulated after one flip flop to other is called Shift Register after the clock pulse is applied inside or outside the registers, the bits stored in registers are shifted. We must associate n number of flip flops to form an n-bit shift register. The number of bits of the binary code is straightly related to the No. of FF (flip-flops). The FF flip-flops are associated in such a technique that the first FF flip-flops outcome turns into the input of the other FF flip-flop.

An SR (Shift Register) could change the bits on both sides to the left or the right. The register which is changing the bit to the left is called the left shift register. The register which is changing the bit to the right is called the right shift register.

Different types of the shift register are given below:

1. SISO (Serial In Serial Out)
2. SIPO (Serial In Parallel Out)
3. PISO (Parallel In Serial Out)
4. PIPO (Parallel In Parallel Out)
5. Bi-directional Shift Register
6. Universal Shift Register

Serial In Serial Out: The data is shifted to "IN" or "OUT" serially in SISO. A single bit is moved at a time in the right or left direction below clock control in the serial in serial out shift register. all the FF(flip-flops) are fixed in the "reset" state ($Y_3 = Y_2 = Y_1 = Y_0 = 0$) initially. The block diagram of SISO is specified below Figure 9.1[1]–[3].

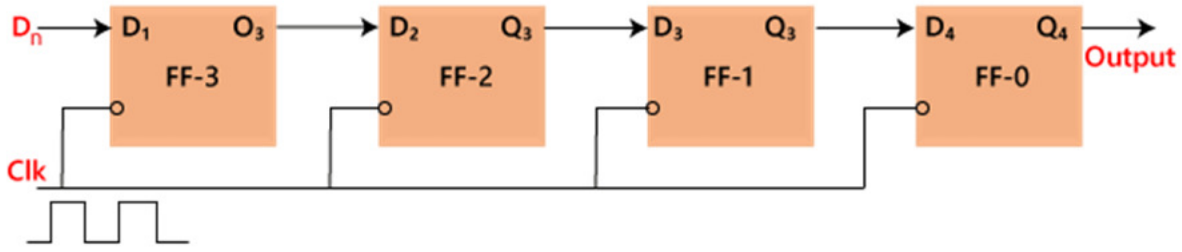


Figure 9.1: Serial in Serial Out

Serial in Parallel Out: The data is agreed serially to the flip flop, and outcomes are got in a parallel way in the serial in a parallel-out shift register. In the SIPO "The data is accepted bit by bit in the register and the outcome rests deactivated till the data is not acceptable to the data input". The outcomes are permitted once the data is approved to register, and the flip-flops comprise their yield value. The 4-bit serial in the parallel-out shift register is specified below. To rearrange these four flip flops, the circuit having 4 D flip flops comprises a clear and timing signal to reset these 4 flip flops. The input of the second flip-flop is the outcome of the first flip-

flop and so on in serial in a parallel-out shift register. To synchronize respective flip flops, the same clock signal is applied to respectively flip flops. The parallel outcomes are cast-off for communication (Figure 9.2).

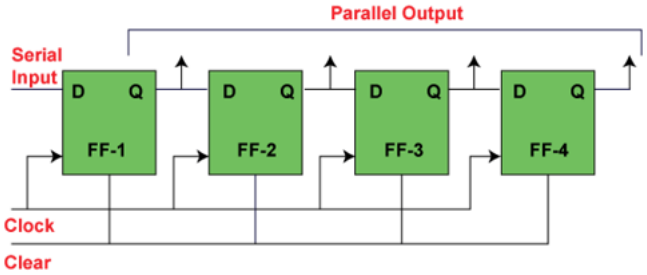


Figure 9.2: Serial in Parallel Out

Parallel in Serial out (PISO): The data is entered in a parallel way, and the output comes serially in the parallel in serial out shift register. A PISO shift register of 4 bits is specified under. The outcome of the previous Flip Flop is the input of the flip flop. "The input and outcome are linked utilizing the combinational circuit". The binary input B_0, B_1, B_2, and B_3 are agreed upon through this combinational circuit. The "PISO" circuit works in 2 modes first is shift mode and the second one is load mode (Figure 9.3).

Load mode: After the 2nd, 4th, and 6th AND gates are operating the bits B_0, B_1, B_2, and B_3 are passed to the equivalent flip flops. Once the shift or load bar line is fixed to zero, these gates are operating. Once the edge of the clock is o, the binary inputs B_0, B_1, B_2, and B_3 will be overloaded into the respective flip-flop to take place the parallel loading.

Shift mode: after the load and shift line is fixed to zero, the 2nd, 4th, and 6th gates are quiet. We are not capable of containing data in a parallel way. When the 1st, 3rd, and 5th gates will be operated, and the ever-changing data will be from the left to the right bit occurs the PISO operation.

The block diagram of the PISO operation is given below:

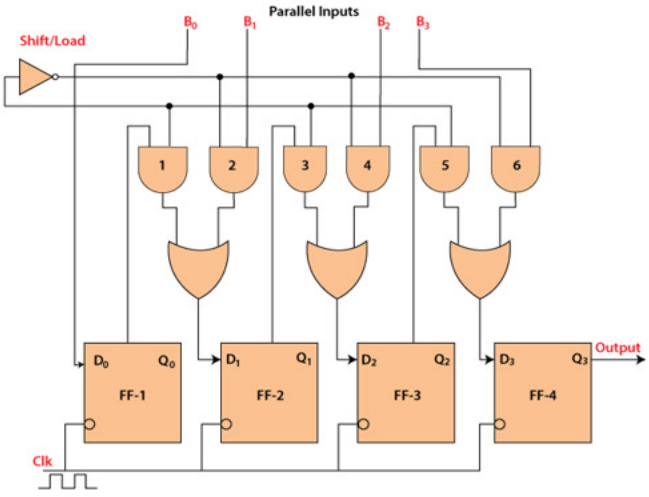


Figure 9.3: Parallel in Serial out

Parallel in Parallel out (PIPO), as shown in Figure 9.4:

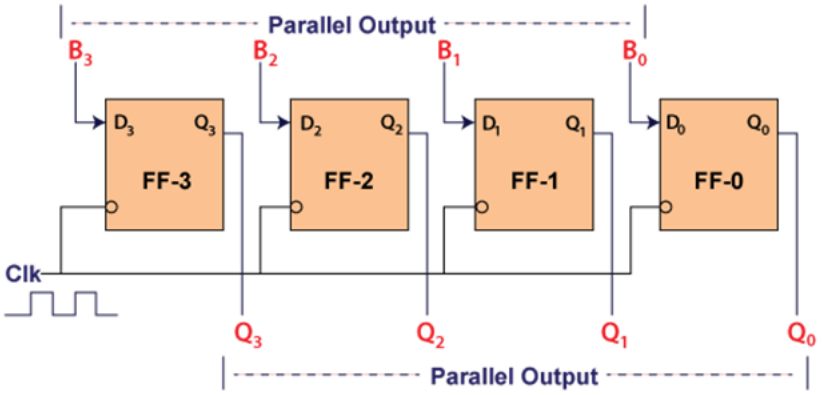


Figure 9.4: Parallel in Parallel out

The inputs and the outcomes will come in a parallel way in the parallel-out shift register (PIPO). Data inputs D_0, D_1, D_2, and D_3 of the individual flip flop are straight accepted by the inputs A_0, A_1, A_2, and A_3. After the negative clock edge is operated, the bits of the binary input is burdened to the flip flops. Aimed at loading all the bits the clock signal pulses are compulsory. The loaded bits appeared on the output side.

Bidirectional shift register: The bidirectional shift register executes the operation of "the binary number afterward shifting each bit of the number to the left by one location will be equal to the number generated through product the original number by two and In the similar technique, the binary number afterward shifting each bit of the number to the right by one location will be corresponding to the number generates through separating the original number by two". it is required that the data should be moved in both directions, left or right in the register for performing the multiplication and division operation using the shift register. The following registers are known as the bidirectional shift register[4]–[6].

The 4-bit bidirectional shift register is given below Figure 9.5:

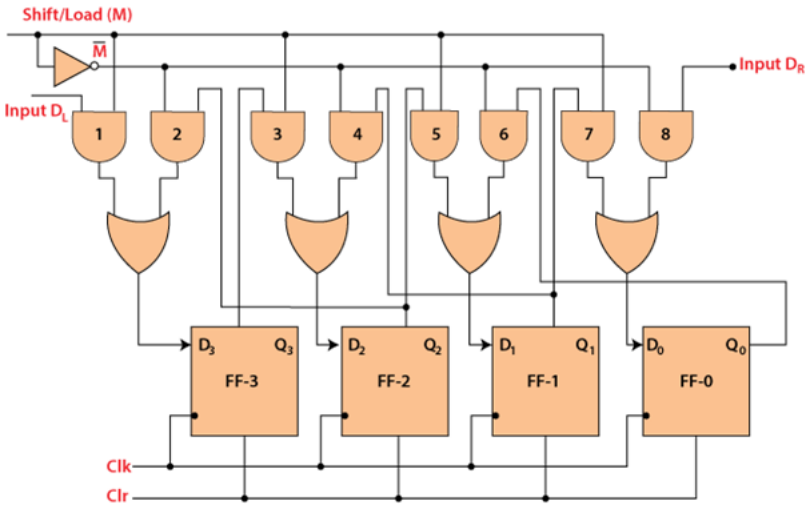


Figure 9.5: Bidirectional shift register

Universal shift register: the shift register in which "data is shifted in one direction is called the unidirectional universal shift register, The bidirectional shift register in which the data is shifted in both the direction is known as bidirectional shift universal register, a shift register which can load the data in a parallel way and shift the data in both directions, right and left is called the universal shift register".

The input M, the method switch input, is fixed to one to execute the parallel loading operation. The serial shifting operation is executed, if this input is fixed to zero. The circuit will effort as a "bi-directional" register if we attach the mode control input with the ground. The universal shift register block diagram is specified as follows. The universal shift register executes the "serial left" operation once the input is approved to the serial input. The register accomplishes the serial right operation, once the input is passed the input D (Figure 9.6).

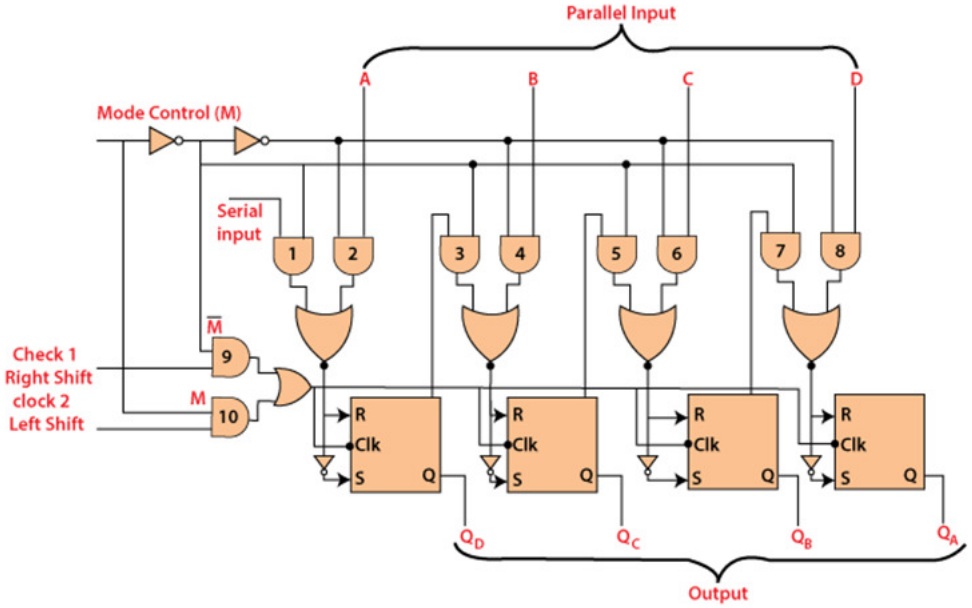


Figure 9.6: Unidirectional shift register

Data converter: data converter is utilized to convert the different kinds of data one form of data into another form of data as digital data is converted into analog data through a digital-to-analog circuit converter and analog data is converted into digital data through the analog-to-digital circuit converter.

ADC Converter: it is known as an Analog-to-Digital circuit converter.ADC Circuits convert the analog signal to digital form known as an ADC converter. ADC is an Integrated circuit (IC) that changes the data straight from continuous data to discrete data. ADC circuit converter can be represented by A/D, ADC, A to D. ADC circuit sign and symbol are specified as follows. The procedure of changing an analog signal to a digital one can be done utilizing some methods. There are dissimilar kinds of ADC chips accessible from dissimilar producers such as the ADC08xx series. An ADC circuit could be deliberate through the assistance of discrete apparatuses.

The different features of the ADC circuit are represented below:

1. The model rate of an ADC is not anything then exactly how dissolute an ADC can change the indicator from analog to digital i.e convert continuous data to discrete data.
2. Bit determination is not other than exactly how much accuracy can an analog-to-digital converter can convert the signal from analog to digital i.e. the accuracy of conversion of continuous data to discrete data.

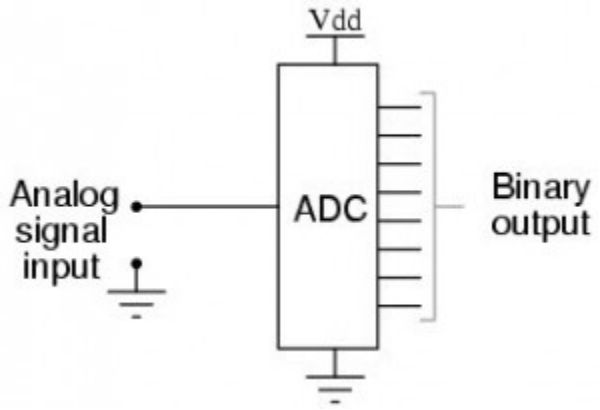


Figure 9.7: ADC Converter

The applications of the A/D Converter or ADCs are extant and controller arrangements, industrialized equipment, communiqué arrangements, and entirely additional sensory-based arrangements. Grouping of ADCs founded on features similar performance, bit rates, power, cost, etc.

Types of A/D converter:

Dual slope A/D converter: A resistor, capacitor, and operational amplifier grouping are formed in an integrated circuit by using comparison voltage produced is called a dual slope A/D converter. This integrator generates a saw tooth waveform on its output from zero to the value Vref Through the fixed value of Vref. the counter starts counting from 0 to 2^n-1 where n is the number of bits of A/D Converter Once the integrator waveform is taking place consistently (Figure 9.8).

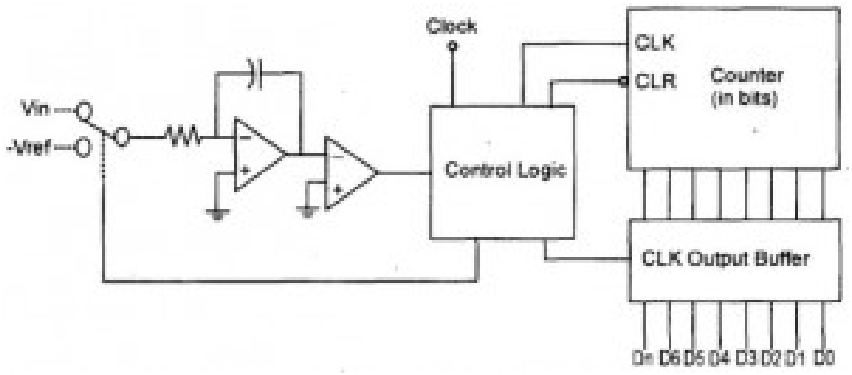


Figure 9.8: Types of A/D circuit

"Once the input voltage Vin is equivalent to the voltage of the waveform, formerly the controller circuit internments the counter value which is the digital value of the consistent analog input value. This Dual slope ADC is a comparatively average rate and measured speediness method".

Flash A/D converter: The most widely used efficient ADC in terms of its speed is the flash ADC converter. It is also called the parallel ADC, which comprises a series of comparators where each one associates the input signal through a distinctive mention voltage Vref. When the analog input voltage exceeds the reference voltage the outcome would be in a high state. For generating binary code based on higher-order input activity by ignoring other active inputs, the outcome is given to the priority encoder. The flash A/D converter has very high-speed devices and the cost of this type of converter is relatively high (Figure 9.9).

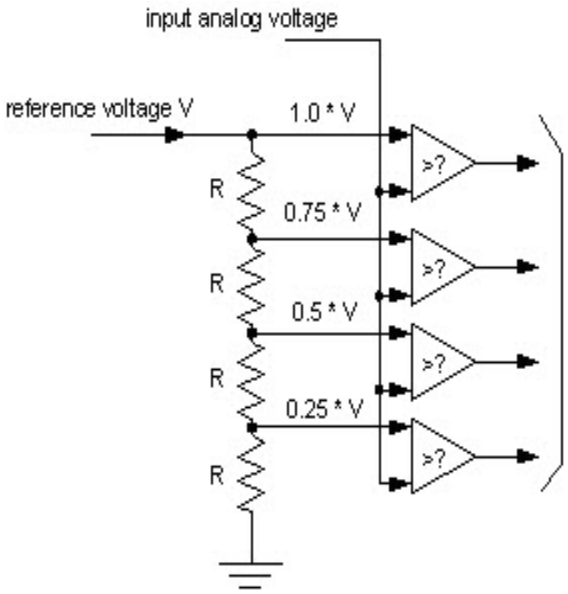


Figure 9.9: flash A/D Converter

Semi-flash ADC: This type of ADC converts mostly effort almost their restriction size through two distinct flash converters, where each converter determination is part of the bits for the semi-flush method. The capability of a single flash converter is, it switches the MSB bit where the additional holders are the LSB bit.

ADC0808: it is a type of ADC converter which converts analog data into digital data.ADC0808 has eight analog inputs and eight digital outcomes. ADC0808 permits us to display up to eight dissimilar transducers through simply a solo chip. This removes the essential for outside 0 and full-scale alterations.

Features of ADC0808:

1. Relaxed interface to entire microchips
2. No outside 0 or full-scale alteration essential
3. 8-channel multiplexer through discourse logic
4. 0V to 5V input choice through single 5V power supply
5. Outcomes encounter TTL voltage level provisions
6. Carrier chip bundle through 28-pin

DAC (digital-to-analog converter): It is known as a Digital-to-Analog circuit converter.DAC Circuits convert the digital signal to analog form known as a DAC converter. DAC is an Integrated circuit (IC) that changes the data straight from discrete data to continuous data. DAC circuit converter can be represented by D/A, DAC, D to A. DAC circuit sign and symbol are specified as follows. The procedure of changing a Digital signal to an analog one can be done utilizing some methods. There are dissimilar kinds of DAC chips accessible from dissimilar Converters. A D/A circuit (digital to analog converter) changes arithmetical discrete input data into equivalent continuous outcome data. The discrete type digital data is expressed through a binary code, which is an arrangement of bits low (0) or high (1) which is applied to Digital to Analog Converters for data conversion.D/A circuit comprises numerous binary inputs and one outcome. The No. of binary input data of a DAC will be equal to the 2^n[7]–[10].

Weighted resistor DAC: A weighted resistor is a type of DAC converter that generates an analog outcome that is equivalent to the digital (binary) input by utilizing binary-weighted resistors in the reversing computer track, this binary weighted resistor D/A is known as weighted resistor D/A converter.

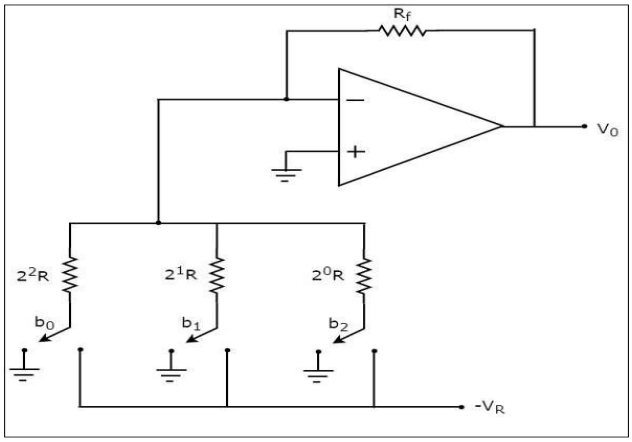


Figure 9.10: DAC converter weighted register

R-2R Ladder circuit: it is a type of DAC circuit that overpowers to deal with the drawbacks of a binary weighted resistor D/A circuit converter. R-2R Ladder D/A generates an analog continuous outcome, that is virtually equivalent to the digital (binary) input utilizing an R-2R ladder circuit in the overturning computer track (Figure 9.11).

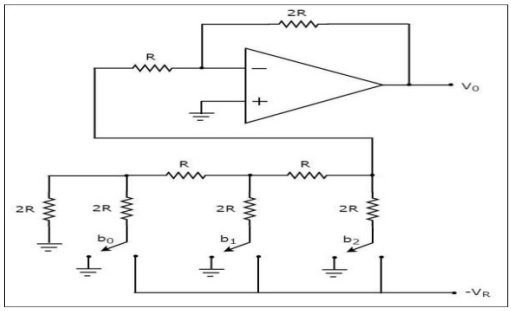


Figure 9.11: R2R ladder circuit

REFERENCES

[1] H. Fujiwara, K. Fujiwara, and T. Hosokawa, "Universal testing for linear feed-forward/feedback shift registers," *IEICE Trans. Inf. Syst.*, 2020, doi: 10.1587/transinf.2019EDP7205.

[2] K. C. Woo, H. J. Kang, and B. Do Yang, "Low-Area and Low-Power Latch-Based Thermometer-Code Shift-Register," *IEEE Trans. Circuits Syst. II Express Briefs*, 2020, doi: 10.1109/TCSII.2019.2943004.

[3] M. Jayasanthi and A. K. Kowsalyadevi, "Low power implementation of linear feedback shift registers," *Int. J. Recent Technol. Eng.*, 2019, doi: 10.35940/ijrte.A3379.078219.

[4] K. C. Woo and B. Do Yang, "5-T and 6-T thermometer-code latches for thermometer-code shift-register," *ETRI J.*, 2021, doi: 10.4218/etrij.2020-0323.

[5] M. N. Divshali and A. Rezai, "Novel circuits design for siso shift register in qca technology," *J. Circuits, Syst. Comput.*, 2021, doi: 10.1142/S0218126621502030.

[6] B. Kim *et al.*, "New depletion-mode IGZO TFT shift register," *IEEE Electron Device Lett.*, 2011, doi: 10.1109/LED.2010.2090939.

[7] B. Tsaban and U. Vishne, "Efficient linear feedback shift registers with maximal period," *Finite Fields their Appl.*, 2002, doi: 10.1006/ffta.2001.0339.

[8] P. Da□browski, G. Łabuzek, T. Rachwalik, and J. Szmidt, "Searching for nonlinear feedback shift registers with parallel computing," *Inf. Process. Lett.*, 2014, doi: 10.1016/j.ipl.2013.12.002.

[9] T. Purkayastha, D. De, and T. Chattopadhyay, "Universal shift register implementation using quantum dot cellular automata," *Ain Shams Eng. J.*, 2016, doi: 10.1016/j.asej.2016.01.011.

[10] H. T. Nguyen, G. N. Pham, A. N. Bui, B. A. Nguyen, N. T. Le, and H. T. Pham, "Linear feedback shift register and its applications in digital system design," *Int. J. Emerg. Technol. Adv. Eng.*, 2021, doi: 10.46338/IJETAE1121_24.

CHAPTER 10

SEMICONDUCTOR MEMORIES

Dr. Manikandan M, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id-manikandan.m@presidencyuniversity.in

Semiconductor memory is a digital electronic semiconductor device that is used for digital data stored in computer memory. Data is stored within the MOS memory cell on a silicon-integrated circuit memory chip. Memory is the most important element of a digital computer system. The computer could not perform or execute a simple task without it. The internal memories of a computer system are the semiconductor memories which is most important for the fast execution of a computer system. It is also known as the main memory or working memory of a computer system. When the computer executes a program or data, that program or data is used these memories for storage of data. The fastest memory devices are the bipolar and metal oxide semiconductor memories and these are available at an effective cost because of decreasing continuously as LSI technology improves.

A memorial is fair similar to a human brain, which is cast-off to collect data and commands. The data is to be managed and instructions essential for operation are stored in the storage space in computer memory. These semiconductor memories are separated into a huge number of small portions called cells, which are located or cell has a matchless address, which differs beginning from zero to memory size -1. Cache memory: To speed up the CPU, a very high-speed semiconductor memory is utilized called cache memory. Cache memory operates as a safeguard between the CPU and the key memory of the computer system. The cache memory is used to clamp individual parts of data and programs which are most commonly cast off by the CPU. The rations of data and programs are relocated from the disk to cache recollection by the functioning scheme, wherever the CPU container admits them[1]–[3].

Types of memory: memory is classified into two types which are given below:

RAM (Random Access Memory): it isalso called the read-write memory or the main memory or the primary memory. The programs and data that the CPU requires during the execution of a program are stored in this memory. It is also volatile memory as the data is lost when power is turned off. The internal memory of the CPU for storing data, program, and program results is called RAM. Each storage location inside the memory is as easy to reach as other locations and takes the same amount of time, so each storage location inside the memory is as easy to reach as other locations and takes the same amount of time. The data which is stored in RAM can access randomly and its cost is very high. The size of RAM is quite small but it can store huge amounts of data.

RAM (Random Access Memory) is a hardware device usually situated on the motherboard of a computer and performances as an inner memorial of the Central Processing Unit after you switch on the computer, it permits the CPU to store data, program, and package outcomes. The info can be written to it as well as read from this memory so it is called read and write memory of a computer. It does not store data or instructions permanently because RAM is a volatile memory. The data and instructions from the hard disk are stowed in the RAM, once you switch on the computer. The operating system and the program are loaded into RAM generally from an HDD

or SSD, once the computer is rebooted, and after you open a program and CPU uses this data to execute the compulsory odd jobs. The RAM loses the data when you shut down the computer. Once the computer is turned off, the data remnants in the RAM are extended as the computer is on and lost. , the data remains in the RAM as long as the computer is on and lost. The data reading from the RAM is much faster than reading from the hard drive is the benefit of loading the data into the RAM.

RAM is further divided into two types:

SRAM (Static RAM): The term static specifies that the memory recollects its substances by way of extended by way of power being provided. Though, data vanished once the power becomes cut off owing to an unstable environment. SRAM package usage a matrix of 6 transistors and no capacitors. Transistors do not need the power to avoid a leak, so SRAM essential not be rested happening an even foundation. RAM is made up of D-type flip-flops and it holds the content as long as the power is available. SRAM is faster than DRAM and it is more expensive.

It cannot store many bits per chip. It uses more power and generates more heat. SRAM is used as a cache memory. Static RAM has a superior prearrangement of transistors that varieties a flip-flop, a kind of memory cell, one memory cell supplies one bit of data. The maximum of the current SRAM memory cells are through six CMOS transistors, nevertheless, absence capacitors are not available. By using SRAM, the access time in SRAM chips dismiss stands by way of little as 10 nanoseconds While the access time in DRAM frequently leftovers overhead 50 nanoseconds.

The six-transistor arrangement of its circuit upholds the flow of current in one direction or the other (0 or 1), so SRAM operation is very fast. Lacking to come for the capacitor to seal up or ditch, the 0 or 1 state can be written and read suddenly. The current synchronous static RAM chips overlay read and write operations, as the initial asynchronous static RAM chips executed read and write operations successively.The disadvantage of SRAM is that its memory cells inhabit additional astronomical on a chip than the DRAM memory cells for a similar quantity of storing interplanetary (memory) as it has additional quantities than a DRAM. So, it proposes fewer memorials apiece bit[4]–[6].

DRAM (Dynamic RAM): It is similarly prepared awake of memory cells, and "It is an integrated circuit (IC) prepared of lots of transistors and capacitors which are very minor in size and respectively transistor is wrinkled up by a capacitor to generate an actual compressed memory cell consequently that lots of them can fit on a single memory chip. So, a memory cell of a DRAM has one transistor and one capacitor then individually cell signifies or provisions a solitary bit of data in its capacitor inside a combined route". The capacitor grips this bit of info or data, moreover as 0 or as 1. The electric circuit on the memory chip to read the capacitor and change its state allows by the transistor is also present in the memory cell and acts as a switch. The capacitor requires to remain rested afterward even during intermissions to uphold the custody of the capacitor. Due to this drawback, it is named dynamic RAM as it wants to be relaxed endlessly to uphold its information or it wanted to disremember whatever it is an allotment. This is achieved by placing the memory on a refresh circuit that rewrites the data several hundred times per second. To enhance or access the timing in Dynamic RAM is about 60 nanoseconds (Figure 10.1).

Difference between SRAM and DRAM:

Table 10.1: Difference between SRAM and DRAM

DRAM	SRAM
It is made up of capacitors that leak electricity.	It is made up of D-type flip-flops.
It requires refreshing every few variations to maintain its data.	It holds its content as long as power is available.
It is slower than SRAM.	It is faster than DRAM.
It is inexpensive.	It is expensive.
It can store many bits per chip.	It can store several bits per chip.
It consumes less power.	It consumes more power.
It generates less heat.	It produces more heat.
It is utilized as the main memory.	It is used as a cache memory

ROM (Random only Memory): it stands for Random Only Memory. Random Access Memory (ROM) is a memory method or storage intermediate in which data is stored eternally. It is likewise the main memory element of a processor laterally through the RAM. We could individually read the programs and data kept on it but cannot write on it, so it is called Random Only Memory. It is limited to interpretation arguments that are enduringly deposited inside the memory element. The producer of ROM seals the programs keen on the ROM at the time of built-up the ROM. Afterward, the content of the ROM dismiss be changed, which revenues you dismiss reprogram, rewrite, or remove its contented pieces of information. Though, there are several kinds of ROM wherever you can adjust the data and piece of pieces of information. ROM comprises superior interior electric waves of anger that can be automated for a specific interconnection arrangement (data). The binary data kept in the chip is stated by the producer and formerly entrenched in the component at the time of built-up to procedure the obligatory interconnection design (data).it halts inside the component smooth once the power is rotated off, after the design (data) is recognized. Consequently, it holds the data even when the power is rotated off, or you close down your processor, it is a non-volatile memory (Figure 10.1).

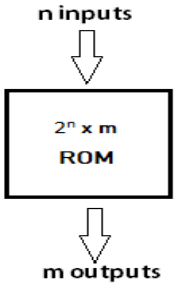


Figure 10.1: Represent the ROM

The internal structure of ROMs: The internal structure contains two elementary mechanisms: interpreter and OR gates. An interpreter is a path that interprets a programmed procedure (such as by way of binary implied decimal, BCD) to a number form. Consequently, the effort is in the binary procedure, and the outcome is its number (decimal) corresponding. Entirely the OR gates existing in the ROM will need the outcomes of the interpreter as the outcomes (Figure 10.2).

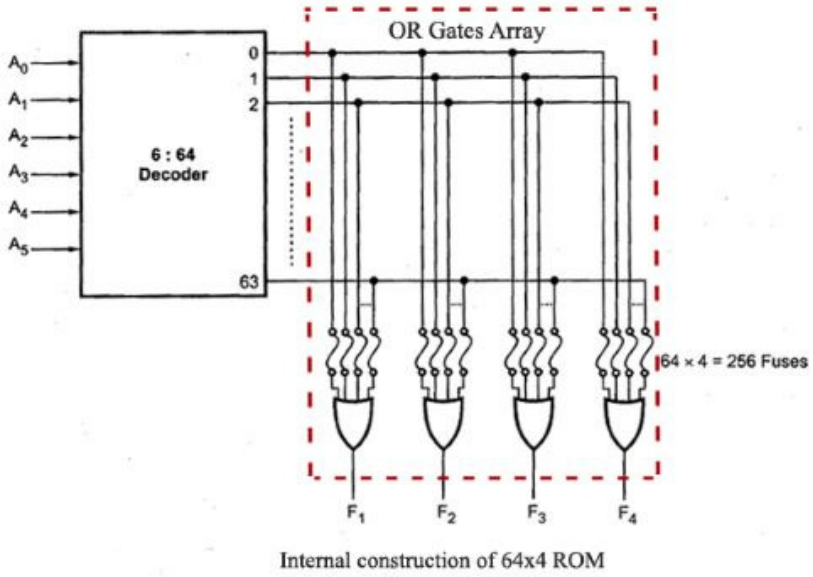


Figure 10.2: Internal structure of ROM

Above ROM comprises 64 arguments of 4 bits respectively. Consequently, nearby desired to be 4 outcomes outlines, and unique of the 64 arguments accessible on the amount produced lines is dogged after the 6 input lines as we require simply 6 inputs since in this ROM we require 26 = 64, consequently we container require 64 addresses or minterms. Intended for each address input, here is an exclusive designated term. Aimed at instance, uncertainty the input address is 000000, expression amount 0 determination be designated and practical to the outcomes lines. Uncertainty the input address is 111111, and expression number 63 is designated and functional to the outcome lines.

Types of ROM: the different types of ROM are specified below-

MROM (Masked Read Only Memory): It is the eldest kind of ROM. It is not cast off wherever in today's world because it has developed outdated. The programs and directives are kept at the time of built-up by the producer consequently it is named hardware memory devices. Consequently, it is automated throughout the built-up procedure, and termination is changed, and reprogrammed, before being removed earlier.The MROM imperfections are prepared of integrated circuits. Imperfections direct a current over a specific input-output passageway resolute by the position of tempers amongst the rackets and pillars on the mark. The existing must permit sideways a fuse-enabled track, consequently, it dismisses reoccurrence simply through the outcome the builder selects. This is the aim of the redrafting and any additional alteration is not unmanageable in this celebration[7]–[10].

PROM (Programmable Read Only Memory): PROM is an absolute system of ROM. It is artificial as absolute memory and automated afterward built-up. We can about that it is reserved absolute at the time of industrialization. You can achieve and formerly package it when utilizing a singular tool named a computer operator. Cutting-edge the chip, the current movements over wholly conceivable paths. The computer operator can pick one specific pathway aimed at the current by sweltering undesirable fits of anger by distributing a great power over them. The operator has the chance to plug in it or to improve data and directives by way of apiece his obligation. Owing to this motive, it is correspondingly identified as the user-programmed ROM as an operator could plug in it. To inscribe pieces of information against a PROM chip; a method called PROM computer operator or PROM burner is castoff. The procedure or program design of a PROM is recognized as scorching the PROM. When it is automated, the pieces of information dismiss be changed advanced, consequently, it is too named a one-time programmable expedient.

Uses: It is cast off in mobile phones, videocassette game tools, medicinal strategies, RFID labels, and additional.

Erasable and Programmable ROM: EPROM is a kind of ROM that could be reprogrammed or re-executed and removed numerous times periods. The technique to remove the information is dissimilar; it originates through a quartz window finish in which an exact incidence of infrared light is approved aimed at about forty minutes to remove the pieces of information. So, it recollects its content till it is unprotected from electromagnetic light. You essential a superior method named a PROM computer operator or PROM burner to reprogram and reoperate the EPROM.

REFERENCES

[1] Y. Fujisaki, “Current status of nonvolatile semiconductor memory technology,” *Japanese Journal of Applied Physics*. 2010. doi: 10.1143/JJAP.49.100001.

[2] L. A. Leventhal, “Semiconductor technologies and semiconductor memories,” *Simulation*, 1976, doi: 10.1177/003754977602700206.

[3] J. N. Barry and R. G. George, “Semiconductor memories,” *IEE Proc. E Comput. Digit. Tech.*, 1986, doi: 10.1049/ip-e.1986.0002.

[4] S. J. Wrazien, Y. Zhao, J. D. Krayer, and M. H. White, “Characterization of SONOS oxynitride nonvolatile semiconductor memory devices,” *Solid. State. Electron.*, 2003, doi: 10.1016/S0038-1101(02)00448-3.

[5] S. Hamsa, A. G. Ananth, and N. Thangadurai, “A study of semiconductor memory technology by comparing volatile and non-volatile memories,” *J. Adv. Res. Dyn. Control Syst.*, 2018.

[6] J. R. Creasy and J. E. Meschi, “SEMICONDUCTOR MEMORIES.,” *GTE Autom Electr Tech J*, 1975, doi: 10.3169/itej1954.29.680.

[7] J. J. Chang, “Nonvolatile Semiconductor Memory Devices,” *Proc. IEEE*, 1976, doi: 10.1109/PROC.1976.10272.

[8] D. Klein, “The History of Semiconductor Memory: From Magnetic Tape to NAND Flash Memory,” *IEEE Solid-State Circuits Mag.*, 2016, doi: 10.1109/MSSC.2016.2548422.

[9] C. L. Chen and M. Y. Hsiao, "ERROR-CORRECTING CODES FOR SEMICONDUCTOR MEMORY APPLICATIONS: A STATE-OF-THE-ART REVIEW.," *IBM J. Res. Dev.*, 1984, doi: 10.1147/rd.282.0124.

[10] M. She, "Semiconductor Flash Memory Scaling," *Thesis*, 2003.

CHAPTER 11

MICROPROCESSOR

Mr. Kiran Kale, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id-kirandhanaji.kale@presidencyuniversity.in

A microprocessor is a programmable microelectronic device package. It has calculating and decision-making abilities the same as to CPU (central processing unit) of a computer system network. Any microchip created on a system needing an imperfect number of properties is called a laptop computer. The microprocessor can be understood in just about totally kind of microelectronic device like mobile phones, laser photocopiers, laundry pieces of machinery, etc. Microprocessors are also cast off in progressive submissions like radars, satellite broadcasting, and air travel owing to the fast progression in microelectronic production and significant addition of strategies consequences in a major price decrease and amplified submissions of the microchip and their byproducts.

A microprocessor is a guiding component of a micro-computer, made up of a minor chip chain of execution ALU actions and collaborating through the others strategies associated with it. The microprocessor comprises an ALU register collection and a controller component. ALU executes arithmetic and rational procedures on the documents conventional from the recollection or an input method. Register array comprises of register recognized by letters such as B, C, D, E, H, L, and accumulator. The control component panels the movement of numbers and directives inside the computer. The microprocessor is a programmable, multipurpose, clock-driven, register-based microelectronic scheme that states binary directives as of a data collection device named memorial, receives binary information as input and procedures pieces of information conferring to individual's commands, and delivers outcomes (Figure 11.1)[1]–[3].

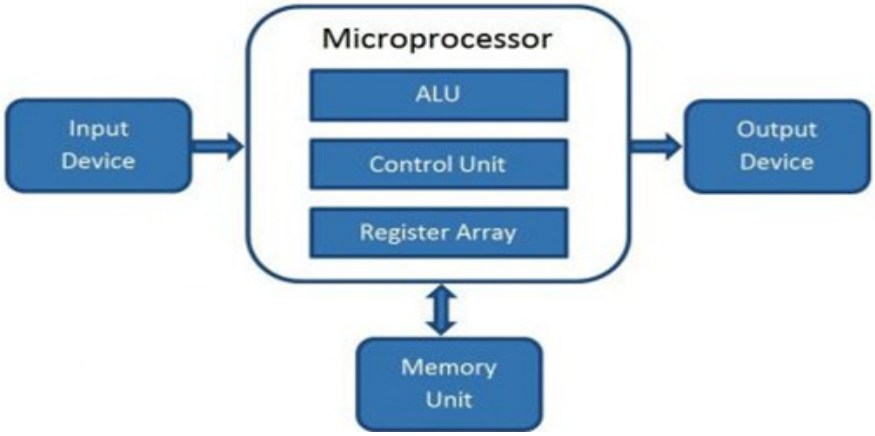


Figure 11.1: Microprocessor

A microprocessor involves an ALU, controller unit, and record collection. Wherever **ALU** executes mathematics and rational procedures on the information established after a contribution expedient or recollection. The controller unit controlled the directives and movement of information inside the processor. And, **record collection** comprises of record register recognized by letters such as B, C, D, E, H, L, and an accumulator.

Evolution of microprocessors: microprocessors could classify conferring to the generations or conferring to the opportunity of the microprocessorwhich is cast off in the laptop computer.

First generation: the first group of microchips was familiarized in the year 1971-1972 by Intel Corporation, called **Intel 4004** subsequently it was a 4-bit supercomputer. It was a PC on a solitary chip. It can execute artless mathematics and rational operations such as addition, subtraction, Boolean OR, and Boolean AND. I took a controller unit that accomplished execution controller purposes corresponding to fetching an instruction as of storing memory, interpreting it, and formerly producing controller pulsations to operate on instructions.

Second generation: in this type of generation microprocessors stayed presented in 1973 by Intel. It was the first 8-bit microprocessor that could execute mathematics and logic operations arranged 8-bit arguments. It was Intel 8008, and the additional enhanced variety was Intel 8088.

Third generation: The third-generation microprocessors, presented in 1978 remained signified through Intel's 8086, Zilog Z800, and 80286, which stayed 16-bit processors through a performance similar to mini-computers. It is a 16-bit processor.

Fourth generation: Numerous dissimilar corporations presented the 32-bit microprocessors, nevertheless the greatest general unique is the Intel 80386. It is 32- a bit microprocessor.

Fifth generation: the fifth-generation microprocessor is from 1995 to now. Afterward 80856, Intel originated available by a novel computer specifically the Pentium processor tracked through the Pentium Pro CPU, which permits numerous CPUs in a single organization to accomplish multitasking.

Features of microprocessor:

1. **Low Cost** – the microprocessor is accessible at a very low cost due to integrated circuit technology. Integrated circuits will decrease the price of a computer operating system.
2. **High Speed**: The microprocessor could work at actual high speed due to the technology elaborated in it. It can perform millions of commands per second.
3. **Small Size:** due to very large scale and ultra large-scale integration technology, a microprocessor is fabricated in a very less footprint. Due to this drawback, the size of the computer system is decreased.
4. **Versatile** - The similar chip could be cast-off and designed for numerous applications, consequently, microprocessors are versatile.
5. **Low Power Consumption** – it consumes less power, so Microprocessors are employing metal oxide semiconductor technology.
6. **Less Heat Generation** - Microprocessors use semiconductor equipment that will not produce far heat as associated with vacuum tube devices.
7. **Reliable** – it is very reliable and the failure rate is very less because it uses semiconductor technology.
8. **Portable** – Microprocessors are portable, due to their small size and low power consumption.

8085 Microprocessor: 8085 is an 8-bit microprocessor which is designed in 1977 by NMOS technology. We will study the features of 8085 as follows:

- 8085 microprocessor is a 40-pin Integrated circuit package that is fabricated on a single LSI chip.
- It operates on a single +5Vd.c. supply.
- It operates on a frequency between 3 -5 MHz.
- It is an 8-bit microprocessor.
- It has 16 address buses (16-bit) from AD_0, AD_1, to AD_7 and A_8, A_9,……A_{15} which can address up to 64KB.
- It has a 16-bit stack pointer register
- It has a 16-bit PC (Program Counter) register.
- It has 6 general purposes 8-bit register i.e. B, C, D, E, H, L, and it can operate as a 16-bit register as work in pair BC, DE, and HL.
- It has 5 interrupt signals.
- It has 8 data buses.
- It is a Nmos technology microchip.

The architecture of 8085: 8085 comprises various units and each unit executes its purposes. The numerous components of a microprocessor are enumerated lower (Figure 11.2):

1. Accumulator
2. Arithmetic and logic unit
3. General purpose register
4. Program counter
5. Stack pointer
6. Temporary register
7. Flag register
8. Instruction registers and decoder
9. Timings and control circuit
10. Interrupt control
11. Address buffer and address data buffer
12. Address bus and data bus

Accumulator: An accumulator is nobody nevertheless a register whose container accepts 8-bit data. Accumulator assistances in storing binary numbers. The pieces of information to be treated by mathematics and logic component is kept in the accumulator register. It correspondingly stores the outcome of the process approved and available by the Arithmetic and Logic components. The accumulator has correspondingly titled an 8-bit register, which can store the 8-bit binary data. The accumulator is associated with the Interior Data bus and Arithmetic and logic unit component devices. To send or receive data from the Internal Data bus the accumulator is cast off. ALU executes the arithmetic and logical operations: Addition, Subtraction, Logical AND, Logical OR, Logical EXCLUSIVE OR, Complement (Logical NOT), Increment, Decrement, Left shift, Rotate left, Rotate right, Clear, etc.

Register: Microprocessors are cast off for temporary storage and operation of data and commands in the register. They are sent to the I/O devices or memory Data remain in the registers till. Intel 8085 microprocessor has the succeeding registers is used for the storage of data and execution of commands: such as an 8-bit accumulator register (A), Six general purpose registers (B, C, D, E, H, and L), the stack pointer (SP), Program Counter (PC), Instruction register, Temporary register. In accumulation to the above-stated registers, the 8085

microprocessor comprises a set of 5 flip-flops that designate approximately circumstances that rises afterward the implementation of an arithmetic or logical instruction operation and commands.

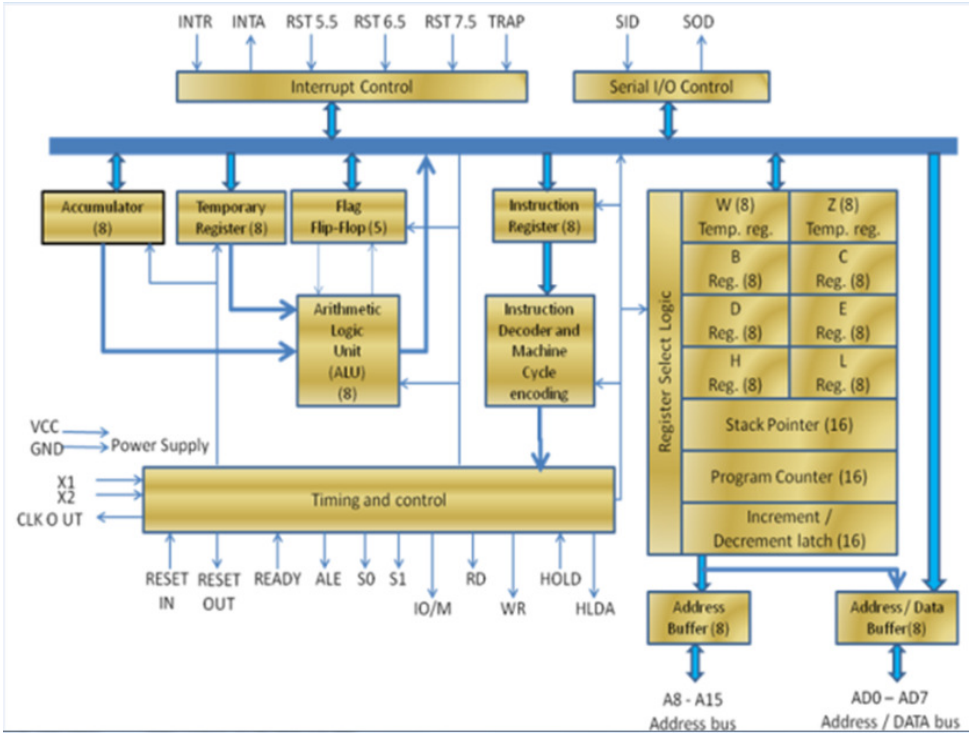


Figure 11.2: Architecture of 8085

ALU: Arithmetic and Logic Unit is continuously a requirement to execute arithmetic actions similar to add, subtraction, multiplication, and divide and to execute rational processes like AND, OR, NOT, etc. therefore, the current is a requirement aimed at manufacturing a distinct component that can execute such categories of actions. These actions are executed through the Arithmetic and Logic Unit (ALU).arithmetic logic unit executes these processes on 8-bit numbers. Except we must have an involvement (or) fact on which the anticipated process is to be executed, these operations cannot be executed. An accumulator is cast-off to deal with the purpose from where these inputs spread the ALU. ALU develops its Input afterward the accumulator and impermanent register. The consequence is kept back in the accumulator afterward dealing out the essential actions[4]–[6]**.**

General-Purpose Registers: it comprises six 8-bit general purpose registers which are: B, D, C, E, H, and L registers. The pairs of two 8-bit registers are utilized to store the 16–bit data. The grouping of two 8-bit registers is called register pair. The usable register pairs in the 8085 are BC, DE, and HL. HL register pair is also used to doing as a memory pointer.

1. **Program Counter (PC):** program counter is a 16-bit general purpose register in which the address of memory is of the next instruction to be executed upheld by the program counter (PC). While the instruction is being executed is tracked by the program counter. The end of the execution of an instruction it points to the next instruction's address in the program because the microprocessor increments the data of the next program during the execution of an instruction.

2. **Stack Pointer (SP):** stack pointer is a 16-bit general purpose function register castoff as memory pointer and a main portion of RAM. The data of individual registers are saved, which are needed in the later part of the program are stored in the stack pointer. The addressing of the stack is controlled by the stack pointer. The address of the top element of data stored in the stack comprises the stack pointer.
3. **Instruction Register:** This register fetches the opcode (operation code or instruction code) machine cycle of the instruction which is being decoded and executed and is stored in the instruction register.
4. **Temporary Register:** temporary register is an 8-bit general register connected using the ALU arithmetic logic unit element. The data during an arithmetic/logical operation is stored in a temporary register. The temporary register is not reachable to the programmer and it is utilized in the microprocessors.

Flags: The Intel 8085 microprocessor comprises five flip-flops to attend as position flags. During the execution of an arithmetic or logical operation, the flip flops are set or reset according to construction. The five status flags register is used in the 8085 microprocessor carry flag, parity flag, auxiliary carry flag, and zero flags, sign flag.

Interrupt control: The control interrupts a process that considers that a microprocessor is executing the main program. The microprocessor changes the controller from the key program to process the incoming request and afterward the completion of the request, the control drives back to the main program whenever the interrupt signal is enabled or requested. To acquire that the data is ready for input, an Input/output device may send an interrupt signal and the microprocessor provisionally breaks the accomplishment of the main program and transmissions governor to the I/O devices element. The control is transferred back to the main program after collecting the input data. There are 5 Interrupt signals which are existing in the 8085 microprocessor: • INTR • RST 7.5 • RST 6.5 • RST 5.5 •and TRAP INTR acknowledgment. Usually one of these instructions: One of the 8 RST instructions (RST0 - RST7), Once the interrupt occurs the processor fetches from the bus one instruction. Address of which is specified in the second and third bytes of the instruction the processor calls the subroutine**.**

Address and data bus: 8 bits of data can be transmitted in parallel from or to the microprocessor because 8085 is an 8-bit microprocessor whose data bus is 8-bit wide. As the memory addresses are 16-bits Intel 8085 requires an address bus of 16-bit wide. The address bus, A-bus (pins A_8 - A_{15}) transmits the 8 most significant bits of the address. The data/address bus, AD-bus (pins AD_0 - AD_7) are transmit the 8 least significant bits of the address (Figure 11.3).

Pin configuration:

Address Bus and Data Bus

A_8 - A_{15} (Output): These address bus lines are cast off for the MSB of the memory address or 8 bits of I/O address.

AD_0 - AD_7 (Input/Output): These multiplexed addresses/data buses serve dual purposes. They are used for the least significant 8 bits of the memory address or I/O address during the first cycle and again they are used for data during the 2nd and 3rd clock cycles.

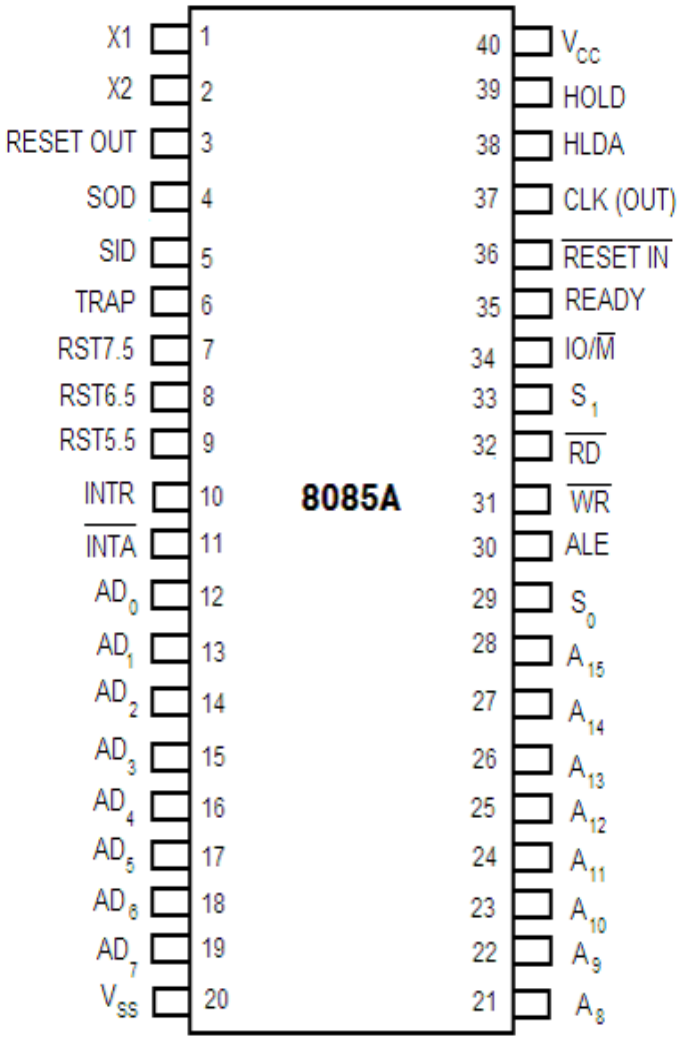


Figure 11.3: pin diagram 8085

Control and Status Signals

1. ALE (Output): during the first clock cycle of a machine cycle Address Latch Enable signal (ALE) goes high and enables the lower 8 bits of the address to be latched either into the memory or external latch.
2. IO/M (Output): the address is for memory or the I/O device is distinguished by status signal.
3. S_0, S_1 (Output): To distinguish the various types of operation these status signals are used sent by the microprocessors.
4. RD (Output): RD is a signal to control READ operation. When it goes low, the selected I/O device or memory is read.
5. WR (Output): WR is a signal to control the WRITE operation. When it goes low, the data bus' data is written into the selected memory or I/O location.
6. READY (Input): It is used by the microprocessor to sense whether a peripheral is ready to transfer data or not. If READY is high, the peripheral is ready. If it is low the microprocessor waits till it goes high.

Interrupts and Externally Initiated Signals

1. **HOLD (INPUT):** HOLD specifies that an alternative device is demanding the use of the address and data bus.
2. **HLDA (OUTPUT):** HLDA is a signal for **HOLD acknowledgment** which indicates that the HOLD request has been received. After the removal of this request, the HLDA goes low.
3. **INTR (Input):** INTR is an **Interrupt Request Signal**. Among interrupts, it has the lowest priority. The INTR is enabled or disabled by software.
4. **INTA (Output):** INTA is an **interrupt acknowledgment** sent by the microprocessor after INTR is received.
5. **RST 5.5, 6.5, 7.5, and TRAP (Input):** When any interrupt is recognized the next instruction is executed from a fixed location in the memory.

Reset Signals

RESET IN (Input): It resets the program counter (PC) to 0. It also resets interrupt enable and HLDA flip-flops. The CPU is held in reset condition till RESET is not applied.

RESET OUT (Output): RESET OUT indicates that the CPU is being reset.

Clock Signals

X_1, X_2 (Input): X1 and X2 are terminals to be connected to an external crystal oscillator that drives an internal circuitry of the microprocessor. It is used to produce a suitable clock for the operation of the microprocessor.

CLK (Output): CLK is a **clock output** for the user, which can be used for other digital ICs. Its frequency is the same at which the processor operates.

Serial I/O Signals

SID (Input): SID is the data line for **serial input**. The data on this line is loaded into the seventh bit of the accumulator when the RIM instruction is executed.

SOD (Output): SOD is a data line for **serial output**. The seventh bit of the accumulator is output on the SOD line when SIM instruction is executed.

Instruction word size:According to the word size, the Intel 8085 instructions are classified into the following three types:

1. One byte instruction
2. Two-byte instruction
3. Three-byte instruction

1. One-byte instruction:

MOV A, B - Move the data of register B to register A.

ADD B - Add the data of register B to the data of the accumulator.

2. Two-byte instruction: In two-byte tutoring, the first byte of the tutoring is its opcode fetch and the second byte is either data or address.

Example:

MVI B, 05; 05 moved to register B.

06, 05; MVI B, 05 is in the code form.

The first byte 06 is the opcode for MVI B and the second byte 05 is the data that is to be moved to register B.

3. *Three-byte instruction:* The first byte of the instruction is its opcode and the second and third bytes are either 16-bit data or 16-bit addresses.

LXI H, 2400H; Load H-L Pair with 2400H

21, 00, 24; LXI H, 2400H in the code form

The first byte 21 is the opcode for the instruction LXI H. The second 00 is 8 LSBs of the data (2400H), which is loaded into register L. The third byte 24 is 8 MSBs of the data (2400H), which is loaded into register H.

Addressing Modes: Every instruction of a program has to operate on data. The method of specifying the data to be operated by the instruction is called Addressing. The 8085 has the following 5 different types of addressing.

1. Immediate Addressing
2. Direct Addressing
3. Register Addressing
4. Register Indirect Addressing
5. Implied Addressing

Immediate Addressing: the data is identified in the instruction itself because The data will be a part of the program instruction In immediate addressing mode, EX. MVI B, 3EH - Move the data 3EH given in the instruction to B register; LXI SP, 2700H.

Direct Addressing: the address of the data is specified in the instruction. The data will be in memory. In this addressing mode, the program instructions and data can be stored in different memory In direct addressing mode, EX. LDA 1050H - Load the data available in memory location 1050H into accumulator; SHLD 3000H

Register Addressing: the instruction specifies the name of the register in which the data is available In register addressing mode, EX. MOV A, B - Move the data of B register to A register; SPHL; ADD C.

Register Indirect Addressing: the instruction specifies the name of the register in which the address of the data is available in register indirect addressing mode. Here the data will be in memory and the address will be in the register pair. EX. MOV A, M - The memory data addressed by the H L pair is moved to the A register. LDAX B.

Implied Addressing: the instruction itself specifies the data to be operated in implied addressing mode EX. CMA - Complement the data of the accumulator[7]–[10].

Timing Diagrams: The technique used to time diagram is the presentation of the beginning of read/write and transmission of data processes below the controller of 3-status signals IO/M', S1, and

S0. Respectively machine cycle is collected from numerous clock series. Subsequently, the data and instructions, together are deposited in the recollection, and the µP executes a fetch process to read the tutoring or statistics and formerly perform the tutoring. The 3-status indications: IO / M', S1, and S0 are produced at the start of respectively machine cycle. The characteristic combination of these 3-status signals classifies the read or write process and leftovers binding for the period of the cycle. Consequently, the time occupied by somewhat µP to perform one tutoring is designed in rapports of the alarm clock retro. The accomplishment of tutoring permanently needs to be delivered and writes operations to transfer data to or from the µP and memory or I/O devices. Respectively read/ write operation establishes one mechanism cycle. Respectively machine cycle comprises numerous clock periods/ cycles, named T-states. Each process is confidential the microprocessor is below the controller of the clock sequence. The clock signal controls the time occupied by the microprocessor to implement any training. The national is clear as the period interval among the 2-trailing or foremost limits of the timepiece. The mechanism sequence is the time compulsory to convey pieces of information to or after recollection or I/O devices. The 8085 microprocessors must have 5 rudimentary mechanism series. They are •the Opcode fetch cycle (4T) • Memory read cycle (3 T) • Memory writes cycle (3 T) • I/O read cycle (3 T) • I/O write cycle (3 T)

8085 microprocessor instructions:

Data transfer group: Commands that are cast off to convey the data from one register to an additional register afterward memory to register or register to memory derive underneath this collection (Table 11.1).

Table 11.1: Data transfer instructions

Opcode	Operand	Meaning	Explanation
MOVE	Rd, Sc M, Sc Dt, M	Print from the source (Sc) to the destination (Dt)	This order prints the data of the source register into the destination register lacking slight alteration.
MVI	Rd, data M, data	Move immediate 8-bit	The 8-bit data is deposited in the destination register or memory.
LDA	16-bit address	Load the accumulator	The data of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
LDAX	B/D Reg. pair	Load the accumulator indirect	The data of the designated register pair point to a memory location. This instruction prints the data of that memory location into the accumulator.
LXI	Reg. pair, 16-bit data	Load the register pair immediate	The instruction loads 16-bit data in the register pair designated in the register or the memory.
LHLD	16-bit address	Load H and L registers direct	The instruction prints the data of the memory location pointed out by the address into register L and prints the data of the next memory

			location into register H.
STA	16-bit address	16-bit address	The data of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.
STAX	16-bit address	Store the accumulator indirect	The data of the accumulator are copied into the memory location specified by the data of the operand
S H L D	16-bit address	Store H and L registers direct	The data of register L are stored in the memory location specified by the 16-bit address in the operand and the data of the H register are stored in the next memory location by incrementing the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.
XCHG	None	Exchange H and L with D and E	The data of register H are exchanged with the data of register D, and the data of register L are exchanged with the data of register E.
SPHL	None	Print H and L registers to the stack pointer	The instruction loads the data of the H and L registers into the stack pointer register. The data of the H register provide the high-order address and the data of the L register provide the low-order address.
XTHL	None	Exchange H and L with the top of the stack	The data of the L register are exchanged with the stack location pointed out by the data of the stack pointer register. The data of the H register are exchanged with the next stack location (SP+1).
PUSH	Reg. pair	Push the register pair onto the stack	The data of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the data of the high-order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the data of the low-order register (C, E, L, flags) are copied to that location.

POP	Reg. pair	Pop off the stack to the register pair	The data of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the data of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1.
OUT	8-bit port address	Output the data from the accumulator to a port with an 8bit address	The data of the accumulator are copied into the I/O port specified by the operand.
IN	8-bit port address	Input data to the accumulator from a port with an 8-bit address	The data of the input port designated in the operand are read and loaded into the accumulator.

Arithmetic group as shown in Table 11.2:

Table 11.2: Arithmetic operation instructions

Instruction Set	**Explanation**	**States**	**Flags**	**Addressing mode**	**Machine Cycles**
ADD r [A] ←[A]+[r]	Add register to the accumulator	4	All	Register	1
ADD M [A] ← [A] + [[H-L]]	Add memory to the accumulator	7	All	Register indirect	2
ACC r [A] ← [A] + [r] + [CS]	Add register with carry to accumulator	4	All	Register	1
ADC M [A] ← [A] + [[H-L]] [CS]	Add memory with carry to accumulator	7	All	Register indirect	2
ADI data [A] ← [A] + data	Add immediate data to the accumulator	7	All	Immediate	2

ACI data [A] ← [A] + data + [CS]	Add with carry immediate data to accumulator	7	All	Immediate	2
DAD rp [H-L] ←[H-L] + [rp]	Add register paid to H-L pair	10	CS	Register	3
SUB r [A] ←[A]-[r]	Subtract register from the accumulator	4	All	Register	1
SUB M [A] ← [A] - [[H-L]]	Subtract memory from the accumulator	7	ALL	Register indirect	2
SBB r [A] ←[A]-[H-L]] - [CS]	Subtract memory from accumulator with borrow	7	All	Register indirect	2
SUI data [A] ←[A]-data	Subtract immediate data from the accumulator	7	All	Immediate	2
SBI data [A] ←[A]-data-[CS]	Subtract immediate data from the accumulator with borrow	7	All	Immediate	2
INR r [r] ←[r]+1	Increment register content	4	All except carry flag	Register	1
INR M [[H-L]] ←[[H-L]]+1	Increment memory content	10	All except carry flag	Register indirect	3
DCR r [r] ←[r] -1	Decrement register content	4	All except carry flag	Register	1
DCR M [[H-L]] ← [[H-L]]-1	Decrement memory content	10	All except carry flag	Register indirect	3
INX rp [rp] ←[rp]+1	Increment memory content	6	None	Register	1
DCX rp [rp] ←[rp]-1	Decrement register pair	6	None	Register	1
DAA	Decimal adjust accumulator	4			1

Logical group as shown in Table 11.3:

Table 11.3: Logical group instructions

Instruction Set	Explanation	States	Flags	Addressing	Machine Cycles
ANA r [A] ←[A]∧[r]	AND register with the accumulator	4	All	Register	1
ANA M [A] ←[A]∧[[H-]]	AND memory with an accumulator	4	All	Register indirect	2
ANI data [A] ← [A] ∧ [data]	AND immediate data with accumulator	7	All	Immediate	2
ORA r [A] ←[A]∨[r]	OR-register with accumulator	4	All	Register	1
ORA M [A] ←[A]∨[[H-L]]	OR-memory with accumulator	7	All	Register indirect	2
ORI data [A] ← [A] ∨ [data]	OR -immediate data with accumulator	7	All	Immediate	2
XRA r [A] ← [A]∀[r]	XOR register with accumulator	4	All	Register	1
XRA M [A] ← [A] ∀ [[H-L]]	XOR memory with accumulator	7	All	Register indirect	2
XRI data [A] ←[A] ∀ [data]	XOR immediate data with accumulator	7	All	Immediate	2
CMA [A] ←[A]	Complement the accumulator	4	None	Implied	1
CMC [CS] ←[CS]	Complement the carry status	4	CS		1
STC [CS] ← 1	Set carry status	4	CS		1
CMP r [A]-[r]	Compare register with accumulator	4	All	Register	1
CMP M [A] - [[H-L]]	Compare memory with accumulator	7	All	Register indirect	2
CPI data [A] - data	Compare immediate data with accumulator	7	All	Immediate	2
RLC $[A_{n+1}] \leftarrow [A^n], [A^0]$	Rotate accumulator left	4	Cs	Implied	1

←[A^7], [CS] ←[A^7]					
RRC [A^7] ←[A^0], [CS] ←[A^0], [A^n] ←[A^{n+1}]	Rotate accumulator right		CS	Implied	1
RAL [A^{n+1}] ←[A^n], [CS] ←[A^7], [A^0] ←[CS]	Rotate accumulator left through carry		CS	Implied	1
RAR [A^n] ←[A^{n+1}], [CS] ←[A^0], [A^7] ←[CS]	Rotate the accumulator right through carry		CS	Implied	1

Conditional jump (Table 11.4):

Table 11.4: Conditional jump instructions

Jump addr (label) [PC] ← Label	Conditional jump: jump to the instruction specified by the address if the specified condition is fulfilled	10, if true and 7, if not true	3, if true and 2, if not true

Unconditional jump (Table 11.5):

Table 11.5: Unconditional jump

JMP addr (label) [PC] ← Label	Unconditional jump: jump to the instruction specified by the address	10	None	Immediate	3

PCHL (Table 11.6):

Table 11.6: PCHL Instructions

PCHL [PC] ← [H-L], [PCH] ←[H], [PCL] ←[L]	Jump address specified by H-L pair	6	None	Register	1

Stack, I/O, and Machine Control Group (Table 11.7):

Table 11.7: Stack I/O and machine control instructions

Instruction Set	**Explanation**	**States**	**Flags**	**Addressing**	**Machine Cycles**
IN port - address [A] ←	Input to accumulator from I/O port	10	None	Direct	3

[Port]					
OUT port-address [Port] ← [A]	Output from the accumulator to the I/O port	10	None	Direct	3
PUSH rp [[SP] - 1] ← [rh], [[SP] - 2] ← [rh], [SP] ← [SP] - 2	Push the content of the register pair to stack	12	None	Register(source)/register Indirect(destination)	3
PUSH PSW [SP]-1] ← [A], [[SP] -2] ← PSW, [SP] ← [SP] - 2	Push processor word	12	None	Register(source)/register Indirect(destination)	3
POP rp [rl] ← [[SP]], [rh] ← [[SP]+1], [SP] ← [SP] + 2	Pop the content of the register pair, which was saved, from the stack	10	None	Register(source)/register Indirect(destination)	3
HLT	Halt	5	None		1
XTHL [L] ↔ [[SP]], [H] ↔ [[SP] + 1]	Exchange top stack with H-L	16	None	Register indirect	5
SPHL [H-L] → [SP]	Moves the contents of the H-L pair to the stack pointer	6	None	Register	1
EI	Enable Interrupts	4	None		1
SIM	Set Interrupts Masks	4	None		1
RIM	Read Interrupts Masks	4	None		1

NOP	No Operation	4	None		1

REFERENCES

[1] E. Jonas and K. P. Kording, "Could a Neuroscientist Understand a Microprocessor?," *PLoS Comput. Biol.*, 2017, doi: 10.1371/journal.pcbi.1005268.

[2] J. Biggs *et al.*, "A natively flexible 32-bit Arm microprocessor," *Nature*, 2021, doi: 10.1038/s41586-021-03625-w.

[3] G. M. Rice, V. Shivashankar, E. J. Ma, J. L. Baryza, and R. Nutiu, "Functional Atlas of Primary miRNA Maturation by the Microprocessor," *Mol. Cell*, 2020, doi: 10.1016/j.molcel.2020.10.028.

[4] T. A. Nguyen *et al.*, "Functional anatomy of the human microprocessor," *Cell*, 2015, doi: 10.1016/j.cell.2015.05.010.

[5] K. R. Kaufman, K. A. Bernhardt, and K. Symms, "Functional assessment and satisfaction of transfemoral amputees with low mobility (FASTK2): A clinical trial of microprocessor-controlled vs. non-microprocessor-controlled knees," *Clin. Biomech.*, 2018, doi: 10.1016/j.clinbiomech.2018.07.012.

[6] J. Dolata, M. Taube, M. Bajczyk, A. Jarmolowski, Z. Szweykowska-Kulinska, and D. Bielewicz, "Regulation of plant microprocessor function in shaping microrna landscape," *Frontiers in Plant Science*. 2018. doi: 10.3389/fpls.2018.00753.

[7] S. A. Fuenzalida Squella, A. Kannenberg, and Â. Brandão Benetti, "Enhancement of a prosthetic knee with a microprocessor-controlled gait phase switch reduces falls and improves balance confidence and gait speed in community ambulators with unilateral transfemoral amputation," *Prosthet. Orthot. Int.*, 2018, doi: 10.1177/0309364617716207.

[8] R. Triboulet, H. M. Chang, R. J. Lapierre, and R. I. Gregory, "Post-transcriptional control of DGCR8 expression by the Microprocessor," *RNA*, 2009, doi: 10.1261/rna.1591709.

[9] S. Furber, "Microprocessors: The engines of the digital age," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2017. doi: 10.1098/rspa.2016.0893.

[10] F. H. Khan, M. A. Pasha, and S. Masud, "Advancements in microprocessor architecture for ubiquitous ai—an overview on history, evolution, and upcoming challenges in ai implementation," *Micromachines*, 2021, doi: 10.3390/mi12060665.

CHAPTER 12

8086 MICROPROCESSOR

Ms. Pallabi Kakati, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id-pallabi.kakati@presidencyuniversity.in

The 8086 Microchip is a superior form of the 8085Microprocessor that was deliberate through Intel in 1976. 8086 is a 16-bit Microchip obligating 20 address outlines and 16 data outlines that offer active to 1MB storing. It contains controlling task collection that offers processes similar to increase and separation effortlessly. It provisions 2 ways of process, Maximum mode, and Minimum mode. Minimum mode is apposite for an organization needing a single processor and Maximum mode is appropriate for an organization needing many processors end (Table 12.1)[1]–[3].

Features of 8086 microprocessor:

1. It operates on a single +5Vd.c. supply.
2. It operates on a frequency between 5-8 MHz.
3. It is a 16-bit microprocessor.
4. It has 20-bit address buses.
5. It has a 16-bit stack pointer register
6. It has a 16-bit PC (Program Counter) register.
7. It is a general-purpose register-based microprocessor.
8. It has 16-bit ALU.
9. It has 16-bit data buses.
10. It is the Hmos technology microchip invented in 1977.
11. It works in 2 modes such as minimum mode and maximum mode.
12. It has nine flags.
13. It supports memory segmentation.
14. The memory capacity is 1 Mb.
15. It works on 6-byte instruction.

Table 12.1: Difference between the 8085 microprocessor and 8086 microprocessor:

8085 Microprocessor	**8086 Microprocessor**
It is an 8-bit microchip.	It is a 16-bit microprocessor.
It does not support memory segmentation.	It supports memory segmentation.
It has 5 status flags.	It has 9 status flags.
It has a 16-bit address line	It has a 20-bit address line
It has an 8-bit data bus.	It has a 16-bit data bus.
It is invented by Nmos technology	It is invented by Hmos technology

The cost of 8085 is low	The cost of 8086 is high
It contains 6500 transistors	It doesn't support pipelining

The architecture of 8086 (Figure 12.1):

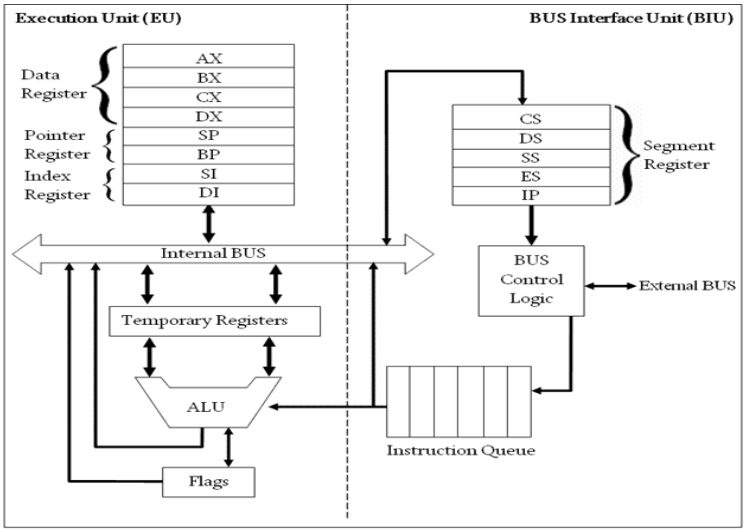


Figure 12.1: Architecture of 8086 microprocessor

EU (Execution Unit):it provides commands to the Buffer Interface Unit uttering after anywhere to raise the pieces of information and formerly decipher and perform individual commands. Its purpose is to control processes on pieces of information utilizing the tutoring interpreter & Arithmetic Logic Unit. The execution unit needs no straight linking through arrangement buses as exposed in the overhead figure, it achieves actions concluded figures over the buffer interface unit.

PIN DIAGRAM OF 8086 (Figure 12.2):

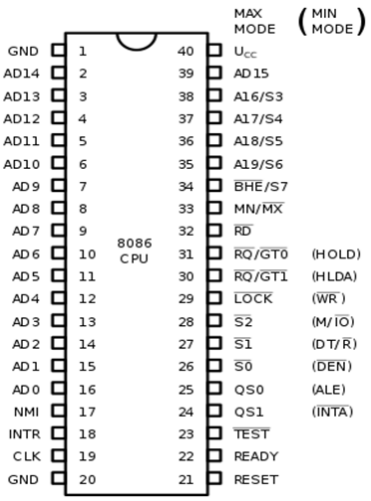


Figure 12.2: pin diagram of 8086

AD0-AD15 (Address Data Bus): Bidirectional address/data lines are low-order address buses AND multiplexed with data.

Once these lines are cast-off to convey memory address, the symbol A is used as an alternative to AD, for example, A0- A15.

A16 - A19 (Output): High-order address lines. These are multiplexed with status signals.

A16/S3, A17/S4: A16 and A17 are multiplexed with segment identifier signals S3 and S4.

A18/S5: A18 is multiplexed with interrupt status S5.

A19/S6: A19 is multiplexed with status signal S6.

BHE/S (Output): Bus High Enable/Status. During T1, it is low. It enables the data onto the most significant half of the data bus, D8-D15. An 8-bit device connected to the upper half of the data bus uses a BHE signal. It is multiplexed with status signal S7. The S7 signal is available during T3 and T4.

RD(Read): For read operation. It is an output signal. It is active when LOW.

Ready (Input): The addressed memory or I/O sends acknowledgment through this pin. When HIGH, it denotes that the peripheral is ready to transfer data.

RESET (Input): System reset. The signal is active HIGH.

CLK (input): Clock 5-10 MHz.

A16/S3, A17/S4: A16 and A17 are multiplexed with segment identifier signals S3 and S4.

A18/S5: A18 is multiplexed with interrupt status S5.

A19/S6: A19 is multiplexed with status signal S6.

BHE/S7 (Output): Bus High Enable/Status. During T1, it is low. It enables the data onto the most significant half of the data bus, D8-D15. An 8-bit device connected to the upper half of the data bus uses a BHE signal. It is multiplexed with status signal S7. The S7 signal is available during T3 and T4[4]–[6].

RD (Read): For read operation. It is an output signal. It is active when LOW.

Ready (Input): The addressed memory or I/O sends acknowledgment through this pin. When HIGH, it denotes that the peripheral is ready to transfer data.

RESET (Input): System reset. The signal is active HIGH.

CLK (input): Clock 5, 8, or 10 MHz.

PIN description for minimum mode:

INTA (Output): PIN 24 interrupts acknowledgment. On receiving the interrupt signal, the processor issues an interrupt acknowledgment signal. It is active LOW.

ALE (Output): Pin no. 25. Address latch enable. It goes HIGH during T1. The microprocessor 8086 sends this signal to latch the address into the Intel 8282/8283 latch.

DEN (Output): Pin no. 26. Data Enable. When Intel 8287/8286 octal bus transceiver is used this signal is. It is active LOW.

DT/R (output): Pin No. 27 data Transmit/Receives. When Intel 8287/8286 octal bus transceiver is used this signal controls the direction of data flows through the transceiver. When it is HIGH, data is sent out. When it is LOW, data is received.

M/IO (Output): Pin no. 28, Memory or I/O access. When this signal is HIGH, the CPU wants to access memory. When this signal is LOW, the CPU wants to access the I/O device.

WR (Output): Pin no. 29, Write. When this signal is LOW, the CPU performs memory or I/O write operation.

HLDA (Output): Pin no. 30, Hold Acknowledgment. It is sent by the processor when it receives a HOLD signal. It is an active HIGH signal. When HOLD is removed HLDA goes LOW.

HOLD (Input): Pin no. 31, Hold. When another device in the microcomputer system wants to use the address and data bus, it sends a HOLD request to the CPU through this pin. It is an active HIGH signal.

Bus Interface Unit (BIU)

The segment registers, instruction pointer, and 6-byte instruction queue are associated with the bus interface unit (BIU).

1. Handles transfer of data and addresses
2. Fetches instruction codes, stores fetched instruction codes in a first-in-first-out register set called a queue
3. Reads data from memory and I/O devices
4. Writes data to memory and I/O devices
5. It relocates addresses of operands since it gets un-relocated operand addresses from the EU. The EU tells the BIU where to fetch instructions or where to read data.

It has the following functional parts:

1. **Instruction Queue:** After the EU performs instructions, the BIU gets 6 bytes of the next instruction and stores them in the instruction queue this process is known as instruction pre-fetch. This process increases the speed of the processor.
2. **Segment Registers:** A segment register contains the addresses of instructions and data in memory which are used by the processor to access memory locations. It points to the starting address of a memory segment currently being used. There are 4 segment registers in 8086 as given below:

Code Segment Register (CS): The code segment of the memory holds the instruction codes of a program.

Data Segment Register (DS): The data, variables, and constants given in the program are held in the data segment of the memory.

Stack Segment Register (SS): The stack segment holds addresses and data of subroutines. It also holds the contents of registers and memory locations given in PUSH instruction.

Extra Segment Register (ES): Extra segment holds the destination addresses of some data of certain string instructions.

Instruction Pointer (IP): The instruction pointer in the 8086 microprocessor acts as a program counter. It indicates the address of the next instruction to be executed.

Execution Unit (EU)

1. The EU receives the opcode of an instruction from the queue, decodes it, and then executes it. While Execution, the unit decodes or executes an instruction, then the BIU fetches instruction codes from the memory and stores them in the queue.
2. The BIU and EU operate in parallel and independently. This makes processing faster.
3. General purpose registers, stack pointers, base pointer and index registers, ALU, flag registers (FLAGS), instruction decoder, and timing and control unit constitute the execution unit (EU). Let's discuss the:
4. General Purpose Registers: There are four 16-bit general purpose registers: AX (Accumulator Register), BX (Base Register), CX (Counter), and DX. Each of these 16-bit registers is further subdivided into 8-bit registers as shown below (Table 12.2):

Table 12.2: General purpose register

16-bit registers	8-bit high-order registers	8-bit low-order registers
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

Index Register: The following four registers are in the group of pointer and index registers:

1. Stack Pointer (SP)
2. Base Pointer (BP)
3. Source Index (SI)
4. Destination Index (DI)

ALU: It handles all arithmetic and logical operations. Such as addition, subtraction, multiplication, division, AND, OR, and NOT operations.

Flag Register: It is a 16-bit register that exactly behaves like a flip-flop, which means it changes states according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups i.e. conditional and control flags.

Conditional Flags: This flag represents the result of the last arithmetic or logical instruction executed. Conditional flags are:

1. Carry Flag
2. Auxiliary Flag
3. Parity Flag
4. Zero Flag
5. Sign Flag

6. Overflow Flag

Control Flags: It controls the operations of the execution unit. Control flags are:

1. Trap Flag
2. Interrupt Flag
3. Direction Flag

Interrupts

An interrupt is a process of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor. Microprocessor responds to these interrupts with an interrupt service routine (ISR), which is a short program or subroutine to instruct the microprocessor on how to handle the interrupt.

Addressing modes of 8086

How an operand is specified for an instruction in the accumulator, in a general-purpose register, or in the memory location, is called addressing mode. The 8086 microprocessors have 8 addressing modes. Two addressing modes have been provided for instructions that operate on register or immediate data[7]–[10].

These two addressing modes are:

Register Addressing: In register addressing, the operand is placed in one of the 16-bit or 8-bit general-purpose registers.

Example

- MOV AX, CX
- ADD AL, BL
- ADD CX, DX

Immediate Addressing: In immediate addressing, the operand is specified in the instruction itself.

Example

- MOV AL, 35H
- MOV BX, 0301H
- MOV [0401], 3598H
- ADD AX, 4836H

The remaining 6 addressing modes specify the location of an operand that is placed in memory.

These 6 addressing modes are:

Direct Addressing: In direct addressing mode, the operand's offset is given in the instruction as an 8-bit or 16-bit displacement element.

Example

ADD AL, [0301]

The instruction adds the content of the offset address 0301 to AL. the operand is placed at the given offset (0301) within the data segment DS.

Register Indirect Addressing: The operand's offset is placed in any one of the registers BX, BP, SI, or DI as specified in the instruction.

Example: MOV AX, [BX]

It moves the contents of memory locations addressed by the register BX to the register AX.

Based Addressing: The operand's offset is the sum of an 8-bit or 16-bit displacement and the contents of the base register BX or BP. BX is used as a base register for the data segment, and the BP is used as a base register for the stack segment.

Effective address (Offset) = [BX + 8-bit or 16-bit displacement].

Example

MOV AL, [BX+05]; an example of 8-bit displacement.

MOV AL, [BX + 1346H]; an example of 16-bit displacement.

Indexed Addressing: The offset of an operand is the sum of the content of an index register SI or DI and an 8-bit or 16-bit displacement.

Offset (Effective Address) = [SI or DI + 8-bit or 16-bit displacement]

Example

MOV AX, [SI + 05]; 8-bit displacement.

MOV AX, [SI + 1528H]; 16-bit displacement.

Based Indexed Addressing: The offset of the operand is the sum of the content of a base register BX or BP and an index register SI or DI.

Effective Address (Offset) = [BX or BP] + [SI or DI]

Here, BX is used as a base register for the data segment, and BP is used as a base register for the stack segment.

Example

ADD AX, [BX + SI]

MOV CX, [BX + SI]

Based Indexed with Displacement: In this mode of addressing, the operand's offset is given by:

Effective Address (Offset) = [BX or BP] + [SI or DI] + 8-bit or 16-bit displacement

Example

MOV AX, [BX + SI + 05]; 8-bit displacement

MOV AX, [BX + SI + 1235H]; 16-bit displacement

REFERENCES

[1] R. Rathod, V. Pawar, S. Mandal, and V. I. M. Road, "Microprocessor VS Microcontroller," *Int. J. Creat. Res. Thoughts*, 2020.

[2] S. P. Morse, "The Intel 8086 Chip and the Future of Microprocessor Design," *Computer*. 2017. doi: 10.1109/MC.2017.120.

[3] Berto Nadeak and Sony Bahagia Sinaga, "Perancangan Perangkat Lunak Pembelajaran Mikroprosesor Dengan Menggunakan Metode Computer Assisted Instruction," *JUKI J. Komput. dan Inform.*, 2021, doi: 10.53842/juki.v2i2.31.

[4] S. P. Morse, B. W. Ravenel, S. Mazor, and W. B. Pohlman, "Intel Microprocessors — 8008 to 8086," *Computer (Long. Beach. Calif).*, 1980, doi: 10.1109/MC.1980.1653375.

[5] S. Morse, W. B. Pohlman, and B. W. Ravenel, "The Intel 8086 Microprocessor: A16-bit Evolution of the 8080," *Computer (Long. Beach. Calif).*, 1978, doi: 10.1109/C-M.1978.218219.

[6] Kanika, S. Chakraverty, and P. Chakraborty, "A peer-assessment based approach for teaching microprogramming," *J. Eng. Educ. Transform.*, 2021, doi: 10.16920/jeet/2021/v34i4/146148.

[7] A. Kawamura, T. Haneyoshi, and R. G. Hoft, "Deadbeat Controlled PWM Inverter with Parameter Estimation Using Only Voltage Sensor," *IEEE Trans. Power Electron.*, 1988, doi: 10.1109/63.4341.

[8] K. P. Gokhale, A. Kawamura, and R. G. Hoft, "Dead beat microprocessor control of pwm inverter for sinusoidal output waveform synthesis," *IEEE Trans. Ind. Appl.*, 1987, doi: 10.1109/TIA.1987.4505001.

[9] E. A. Qaralleh and K. A. Darabkh, "A new method for teaching microprocessors course using emulation," *Comput. Appl. Eng. Educ.*, 2015, doi: 10.1002/cae.21616.

[10] F. A. Kittler and E. Butler, "16. There Is No Software," in *The Truth of the Technological World*, 2020. doi: 10.1515/9780804792622-016.

Questionnaires

1. What are the different types of number systems?
2. Discuss the Boolean algebra in detail.
3. Explain the Karnaugh mapping method in Boolean algebra.
4. What are logic gates? explain different types of logic gates.
5. Explain the arithmetic circuits.
6. What is the difference between an encoder and a decoder?
7. Explain the multiplexer in detail and also its field application.
8. What is the sequential circuit write down a short note on flip-flops?

9. Explain different types of data converters.
10. Write down a note on ROM and RAM.
11. Explain the pin diagram of the 8085 microprocessor.
12. Write down the features of the 8085 microprocessor.
13. Explain the architecture of the 8086 microprocessor.
14. Write a note on features and registers used in the 8086 microchips.

REFERENCE BOOKS:

1. Digital Electronics Book by R P Jain.
2. Digital Logic and Computer Design Book by Morris Mano.
3. Digital Electronics Book by Morris Mano.
4. Digital Circuits and Design Book by Salivahanan.
5. Fundamentals of Digital Circuits Book by A Anand Kumar.
6. Digital Electronics Book by Salivahanan.