Introduction

Digital and Analog Signals

Signals carry information and are defined as any physical quantity that varies with time, space, or any other independent variable. For example, a sine wave whose amplitude varies with respect to time or the motion of a particle with respect to space can be considered as signals. A system can be defined as a physical device that performs an operation on a signal. For example, an amplifier is used to amplify the input signal amplitude. In this case, the amplifier performs some operation(s) on the signal, which has the effect of increasing the amplitude of the desired information-bearing signal.

Signals can be categorized in various ways; for example discrete and continuous time domains. Discrete-time signals are defined only on a discrete set of times. Continuous-time signals are often referred to as continuous signals even when the signal functions are not continuous; an example is a square-wave signal.

Figure 1a: Analog Signal

Figure 1b : Digital Signal

Another category of signals is discrete-valued and continuous-valued or otherwise known as digital and analog signals. Digital signals are discrete-valued and analog signals are continuous electrical signals that vary in time as shown in Figure 1 (a) and (b). Analog devices and systems process signals whose voltages or other quantities vary in a continuous manner. They can take on any value across a continuous range of voltage, current, or other metric. The analog signals can have an infinite number of values. Analog systems can be called wave systems. They have a value that changes steadily over time and can have any one of an infinite set of values in a range. Analog signals represent some physical quantity and they can be a model of the real quantity. Most of the time, the variations corresponds to that of the non-electric (original) signal. For example, the telephone transmitter converts the sounds into an electrical voltage signal. The intensity of the voice causes electric current variations. Therefore, the two are analogous hence the name analog. At the receiving end, the signal is reproduced in the same proportion. Hence the electric current is a model and is an electrical representation of one's voice.

Not all analog signals vary as smoothly as the waveform shown in Fig 1(a). Digital signals are non-continuous, they change in individual steps. They consist of pulses or digits with discrete levels or values. The value of each pulse is constant, but there is an abrupt change from one digit to the next. Digital signals have two amplitude levels. The value of which are specified as one of two possibilities such as 1 or 0, HIGH or LOW , TRUE or FALSE and so on. In reality, the values are anywhere within specific ranges and we define values within a given range

A digital system is the one that handles only discrete values or signals. Any set that is restricted to a finite number of elements contains discrete information. The word digital describes any system based on discontinuous data or events. Digital is the method of storing, processing and transmitting information through the use of distinct electronic pulses that represent the binary digits 0 and 1. Examples of discrete sets are the 10 decimal digits, the 26 letters of the alphabet etc. A digital system would be to flick the light switch on and off. There's no 'in between' values.

Advantages of digital signals

The usual advantages of digital circuits when compared to analog circuits are:

Noise Margin (resistance to noise/robustness) : Digital circuits are less affected by noise. If the noise is below a certain level (the noise margin), a digital circuit behaves as if there was no noise at all. The stream

of bits can be reconstructed into a perfect replica of the original source. However, if the noise exceeds this level, the digital circuit cannot give correct results.

Error Correction and Detection : Digital signals can be regenerated to achieve lossless data transmission, within certain limits. Analog signal transmission and processing, by contrast, always introduces noise

Easily Programmable : Digital systems interface well with computers and are easy to control with software. It is often possible to add new features to a digital system without changing hardware, and to do this remotely, just by uploading new software. Design errors or bugs can be worked-around with a software upgrade, after the product is in customer hands. A digital system is often preferred because of (re-)programmability and ease of upgrading without requiring hardware changes.

Cheap Electronic Circuits : More digital circuitry can be fabricated per square millimeter of integrated-circuit material. Information storage can be much easier in digital systems than in analog ones. In particular, the great noise-immunity of digital systems makes it possible to store data and retrieve it later without degradation. In an analog system, aging and wear and tear will degrade the information in storage, but in a digital system, as long as the wear and tear is below a certain level, the information can be recovered perfectly. Theoretically, there is no data-loss when copying digital data. This is a great advantage over analog systems, which faithfully reproduce every bit of noise that makes its way into the signal.

Disadvantages The world in which we live is analog, and signals from this world such as light, temperature, sound, electrical conductivity, electric and magnetic fields, and phenomena such as the flow of time, are for most practical purposes continuous and thus analog quantities rather than discrete digital ones. For a digital system to do useful things in the real world, translation from the continuous realm to the discrete digital realm must occur, resulting in quantization errors. This problem can usually be mitigated by designing the system to store enough digital data to represent the signal to the desired degree of fidelity. The Nyquist-Shannon sampling theorem provides an important guideline as to how much digital data is needed to accurately portray a given analog signal.

Digital systems can be fragile, in that if a single piece of digital data is lost or misinterpreted, the meaning of large blocks of related data can completely change. This problem can be diminished by designing the digital system for robustness. For example, a parity bit or other error-detecting or error-correcting code can be inserted into the signal path so that minor data corruptions can be detected and possibly corrected.

Digital circuits use more energy than analog circuits to accomplish the same calculations and signal processing tasks, thus producing more heat as well. In portable or battery-powered systems this can be a major limiting factor.

Digital circuits are made from analog components, and care has to be taken to all noise and timing margins, to parasitic inductances and capacitances, to proper filtering of power and ground connections, to electromagnetic coupling amongst data lines. Inattention to these can cause problems such as "glitches", pulses do not reach valid switching (threshold) voltages, or unexpected ("undecoded") combinations of logic states.

A corollary of the fact that digital circuits are made from analog components is the fact that digital circuits are slower to perform calculations than analog circuits that occupy a similar amount of physical space and consume the same amount of power. However, the digital circuit will perform the calculation with much better repeatability, due to the high noise immunity of digital circuitry.

Number Systems

Introduction

Number systems provide the basis for all operations in information processing systems. In a number system the information is divided into a group of symbols; for example, 26 English letters, 10 decimal digits etc. In conventional arithmetic, a number system based upon ten units (0 to 9) is used. However,

arithmetic and logic circuits used in computers and other digital systems operate with only 0's and 1's because it is very difficult to design circuits that require ten distinct states. The number system with the basic symbols 0 and 1 is called binary. ie. A binary system uses just two discrete values. The binary digit (either 0 or 1) is called a bit.
A group of bits which is used to represent the discrete elements of information is a symbol. The mapping of symbols to a binary value is known a binary code. This mapping must be unique. For example, the decimal digits 0 through 9 are represented in a digital system with a code of four bits. Thus a digital system is a system that manipulates discrete elements of information that is represented internally in binary form.

Decimal Numbers

The invention of decimal number system has been the most important factor in the development of science and technology. The decimal number system uses positional number representation, which means that the value of each digit is determined by its position in a number.

The base, also called the radix of a number system is the number of symbols that the system contains. The decimal system has ten symbols: 0,1,2,3,4,5,6,7,8,9. In other words, it has a base of 10. Each position in the decimal system is 10 times more significant than the previous position. The numeric value of a decimal number is determined by multiplying each digit of the number by the value of the position in which the digit appears and then adding the products. Thus the number 2734 is interpreted as

Here 4 is the least significant digit (LSD) and 2 is the most significant digit (MSD).
In general in a number system with a base or radix r, the digits used are from 0 to r-1 and the number can be represented as

Equation (1) is for all integers and for the fractions (numbers between 0 and 1), the following equation holds.

Thus for decimal fraction 0.7123

Binary Numbers

The binary number has a radix of 2. As r = 2, only two digits are needed, and these are 0 and 1. Like the decimal system, binary is a positional system, except that each bit position corresponds to a power of 2 instead of a power of 10. In digital systems, the binary number system and other number systems closely related to it are used almost exclusively. Hence, digital systems often provide conversion between decimal and binary numbers. The decimal value of a binary number can be formed by multiplying each power of 2 by either 1 or 0 followed by adding the values together.
Example : The decimal equivalent of the binary number 101010.

In binary r bits can represent symbols. e.g. 3 bits can represent up to 8 symbols, 4 bits for 16 symbols etc. For N symbols to be represented, the minimum number of bits required is the lowest integer 'r" that satisfies the relationship.

e.g. if N = 26, minimum r is 5 since .

Octal Numbers

Digital systems operate only on binary numbers. Since binary numbers are often very long, two shorthand notations, octal and hexadecimal, are used for representing large binary numbers. Octal systems use a base or radix of 8. Thus it has digits from 0 to 7 (r-1). As in the decimal and binary systems, the positional

valued of each digit in a sequence of numbers is fixed. Each position in an octal number is a power of 8, and each position is 8 times more significant than the previous position.
Example : The decimal equivalent of the octal number 15.2.

Hexadecimal Numbers
The hexadecimal numbering system has a base of 16. There are 16 symbols. The decimal digits 0 to 9 are used as the first ten digits as in the decimal system, followed by the letters A, B, C, D, E and F, which represent the values 10, 11,12,13,14 and 15 respectively. Table 1 shows the relationship between decimal, binary, octal and hexadecimal number systems.Decimal Binary Octal Hexadecimal

0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Hexadecimal numbers are often used in describing the data in computer memory. A computer memory stores a large number of words, each of which is a standard size collection of bits. An 8-bit word is known as a Byte. A hexadecimal digit may be considered as half of a byte. Two hexadecimal digits constitute one byte, the rightmost 4 bits corresponding to half a byte, and the leftmost 4 bits corresponding to the other half of the byte. Often a half-byte is called nibble.
If "word" size is n bits there are 2n possible bit patterns so only 2n possible distinct numbers can be represented. It implies that all possible numbers cannot be represent and some of these bit patterns (half?) to represent negative numbers. The negative numbers are generally represented with sign magnitude i.e. reserve one bit for the sign and the rest of bits are interpreted directly as the number. For example in a 4 bit system, 0000 to 0111 can be used to positive numbers from +0 to +2n-1 and represent 1000 to 1111 can be used for negative numbers from -0 to -2n-1. The two possible zero's redundant and also it can be seen that such representations are arithmetically costly.
Another way to represent negative numbers are by radix and radix-1 complement (also called r's and (r-1)'s). For example -k is represented as Rn -k. In the case of base 10 and corresponding 10's complement with n=2, 0 to 99 are the possible numbers. In such a system, 0 to 49 is reserved for positive numbers and 50 to 99 are for positive numbers.
Examples:
+3 = +3
-3 = 10 2 -3 = 97

2's complement is a special case of complement representation. The negative number -k is equal to 2 n -k. In 4 bits system, positive numbers 0 to 2n-1 is represented by 0000 to 0111 and negative numbers -2n-1 to -1 is represented by 1000 to 1111. Such a representation has only one zero and arithmetic is easier. To negate a number complement all bits and add 1

Example:

119 10 = 01110111 2

Complementing bits will result

10001000

+1 add 1

10001001

That is 10001001 2 = - 119 10

Properties of Two's Complement Numbers

X plus the complement of X equals 0.

There is one unique 0.

Positive numbers have 0 as their leading bit (MSB); while negatives have 1 as their MSB .

The range for an n-bit binary number in 2's complement representation is from -2 (n-1) to 2 (n-1) - 1

The complement of the complement of a number is the original number.

Subtraction is done by addition to the 2's complement of the number.

Value of Two's Complement Numbers

For an n-bit 2's complement number the weights of the bits is the same as for unsigned numbers except of the MSB . For the MSB or sign bit, the weight is -2 n-1. The value of the n-bit 2's complement number is given by:

A 2's-complement = (a n-1) x (-2 n-1) + (a n-2) x (2 n-1) + ... (a 1) x (2 1) + a 0

For example, the value of the 4-bit 2's complement number 1011 is given by:

= 1 x -2 3 + 0 x 2 2 + 1 x 2 1 + 1

= -8 + 0 + 2 + 1

= -5

An n-bit 2's complement number can converted to an m-bit number where m>n by appending m-n copies of the sign bit to the left of the number. This process is called sign extension. Example: To convert the 4-bit 2's complement number 1011 to an 8-bit representation, the sign bit (here = 1) must be extended by appending four 1's to left of the number:

1011 4-bit 2's-complement = 11111011 8-bit 2's-complement

To verify that the value of the 8-bit number is still -5; value of 8-bit number

= -27 + 26 + 25 + 24 + 23 +2 +1

= -128 + 64 + 32 + 16 +8 +2+1

= -128 + 123 = -5

Similar to decimal number addition, two binary numbers are added by adding each pair of bits together with carry propagation. An addition example is illustrated below:

X 190

Y 141

X + Y 331

Similar to addition, two binary numbers are subtracted by subtracting each pair of bits together with borrowing, where needed. For example:

X 229

Y	46
X - Y	183

Two' complement addition/subtraction example

Overflow occurs if signs (MSBs) of both operands are the same and the sign of the result is different. Overflow can also be detected if the carry in the sign position is different from the carry out of the sign position. Ignore carry out from MSB.

Number Base Conversions

This section describes the conversion of numbers from one number system to another. Radix Divide and Multiply Method is generally used for conversion. There is a general procedure for the operation of converting a decimal number to a number in base r. If the number includes a radix point, it is necessary to separate the number into an integer part and a fraction part, since each part must be converted differently. The conversion of a decimal integer to a number in base r is done by dividing the number and all successive quotients by r and accumulating the remainders. The conversion of a decimal fraction is done by repeated multiplication by r and the integers are accumulated instead of remainders.

Integer part - repeated divisions by r yield LSD to MSD

Fractional part - repeated multiplications by r yield MSD to LSD

Example: Conversion of decimal 23 to binary is by divide decimal value by 2 (the base) until the value is 0

The answer is 23 10 = 10111 2

Divide number by 2; keep track of remainder; repeat with dividend equal to quotient until zero; first remainder is binary LSB and last is MSB.

The conversion from decimal integers to any base-r system is similar to this above example, except that division is done by r instead of 2

The conversion of decimal numbers with both integer and fraction parts is done by converting the integer and the fraction separately and then combining the two answers.

Thus (23.7854) 10 = (10111. 1100) 2

For converting a binary number to octal, the following two step procedure can be used.

Group the number of bits into 3's starting at least significant symbol. If the number of bits is not evenly divisible by 3, then add 0's at the most significant end.

Write the corresponding 1 octal digit for each group

Examples:

Similarly for converting a binary number to hex, the following two step procedure can be used.

Group the number of bits into 4's starting at least significant symbol. If the number of bits is not evenly divisible by 4, then add 0's at the most significant end.

Write the corresponding 1 hex digit for each group

Examples:

The hex to binary conversion is very simple; just write down the 4 bit binary code for each hexadecimal digit

Example:

Similarly for octal to binary conversion, write down the 8 bit binary code for each octal digit.
The hex to octal conversion can be carried out in 2 steps; first the hex to binary followed by the binary to octal. Similarly, decimal to hex conversion is completed in 2 steps; first the decimal to binary and from binary to hex as described above.

Boolean Algebra and Basic Operators

Due to historical reasons, digital circuits are called switching circuits, digital circuit functions are called switching functions and the algebra is called switching algebra. The algebraic system known as Boolean algebra named after the mathematician George Boole. George Boole Invented multi-valued discrete algebra (1854) and E. V. Huntington developed its postulates and theorems (1904). Historically, the theory of switching networks (or systems) is credited to Claude Shannon, who applied mathematical logic to describe relay circuits (1938). Relays are controlled electromechanical switches and they have been replaced by electronic controlled switches called logic gates. A special case of Boolean Algebra known as Switching Algebra is a useful mathematical model for describing the combinational circuits. In this section we will briefly discus how the Boolean algebra is applied to the design of digital systems.
Examples of Huntington 's postulates are given below:

Closure

If X and Y are in set (0, 1) then operations are also in set (0, 1)

Identity

Distributive

Complement

Note that for each property, one form is the dual of the other; (zeros to ones, ones to zeros, '.' operations to '+' operations, '+' operations to '.' operations).
From the above postulates the following theorems could be derived.

Associative

Idempotence

Absorption

Simplification

Consensus

Adjacency

Demorgans

In general form

Very useful for complementing function expressions; for example

Switching Algebra Operations
A set is a collection of objects (or elements) and for example a set Z {0, 1} means that Z is a set containing two elements distinguished by the symbols 0 and 1. There are three primary operations AND , OR and NOT.
NOT
It is anary complement or inversion operation. Usually shown as over bar ( ), other forms are and

AND
Also known as the conjunction operation; output is true (1) only if all inputs are true. Algebraic operators are '.', '&', ' '

OR
Also known as the disjunction operation; output is true (1) if any input is true. Algebraic operators are '+', '|', "

AND and OR are called binary operations because they are defined on two operands X and Y. Not is called a unary operation because it is defined on a single operand X. All of these operations are closed. That means if one applies the operation to two elements in a set Z {0, 1}, the result will be always an element in the set B and not something else.
Like standard algebra, switching algebra operators have a precedence of evaluation. The following rules are useful in this regard.
NOT operations have the highest precedence
AND operations are next
OR operations are lowest
Parentheses explicitly define the order of operator evaluation and it is a good practice to use parentheses especially for situations which can cases doubt.
Note that in Boolean algebra the operators AND and OR are not linear group operations; so one cannot solve equations by "adding to" of "multiplying" on both sides of the equal sign as is done with real, complex numbers in standard algebra.
Additional Logic Operations
For two inputs, there are 16 ways we can assign output values. Besides AND and OR, there are five other operations which are useful.
BUFFER
The unary Buffer operation is useful in the real world

NAND
NAND (NOT - AND) is the complement of the AND operation

NOR
NOR (NOT - OR) is the complement of the OR operation

XOR
Exclusive OR is similar to the inclusive OR except output is 0 for 1. It is stated in other words as the output is 1 when modulo 2 input sum is equal to 1.

XNOR
Exclusive NOR is the complement of the XOR operation. Alternatively the output is 1 when modulo 2 input sum is not equal to 1.

Minimal Logic Operator Sets
AND , OR, NOT are all that's needed to express any combinational logic function as switching algebra expression. However two other minimal logic operator sets are also possible with NAND gates or NOR gates. The following is a demonstration of how just NANDs or NORs can do AND , OR, NOT operations
NAND as a Minimal Set

NOR as a Minimal Set

Three State Outputs
Standard logic gate outputs only have two states; high and low. Outputs are effectively either connected to +V or ground, that means there is always a low impedance path to the supply rails. In certain applications require a logic output that we can "turn off" or disable. It means that the output is disconnected (high impedance state). This is the three-state output and can be implemented by a stand-alone unit (a buffer) or part of another function output. This circuit is so-called tri-state because it has three output states: high (1), low (0), and high impedance (Z).
In the logic circuit of Fig. 15(a), there is an additional switch to a digital buffer, which is called as enabled input denoted by E . When E is low, the output is disconnected from the input circuit. When Eis high, the switch is connected and the circuit behaves like a digital buffer. All these states are listed in Truth Table 15(b). Figure 8.14(c) depicts the symbol of a Tri-state Buffer.

Fig. 15 : (a) Switch Configuration, (b) Truth Table, and (c) Symbol of a Tri-state Buffer

Analyses and Synthesis of Combinational Logic Circuits
The important terms we are discussing in this section are
Logic Expression - a mathematical formula consisting of logical operators and variables.
Logic Operator - a function that gives a well defined output according to switching algebra.
Logic Variable - a symbol representing the two possible switching algebra values of 0 and 1.
Logic Literal - the values 0 and 1 or a logic variable or it's complement.
The analysis means a digital circuit is given and we are asked to determine its input-output relationship (its purpose, operation, what it does). One studies the circuit and then states the input-output relationship of the circuit in text or on a truth table or on an operation table or on an operation diagram. The synthesis means an input-output relationship is given and we are asked to design the digital circuit. The input-output relationship is a very crucial component of digital circuit study. Complex digital circuits are blocks are analyzed/designed individually and finally the whole circuit is analyzed/designed.
Combinational circuit analysis starts with a schematic and answers the following questions:
What is the truth table(s) for the circuit output function(s)
What is the logic expression(s) for the circuit output function(s)
Two types of analyses are possible literal as well as symbolic analysis. Literal analysis is process of manually assigning a set of values to the inputs, tracing the results, and recording the output values. For 'n" inputs there are 2n possible input combinations. From input values, gate outputs are evaluated to form

next set of gate inputs and evaluation continues until gate outputs are circuit outputs. The literal analysis only gives us the truth table.

Symbolic analysis also starts with the circuit diagram like literal analysis. But instead of assigning values, gate output expressions are determined. Intermediate expressions are combined in following gates to form complex expressions. Symbolic analysis is more work but gives us complete information of both the truth table and logic expression.

Now we will consider an example for the analysis of combinational logic circuit shown in Fig. 3.

Figure 3

Analyzing this circuit, it can be seen that

Output of Gate G1 = AB

Output of Gate G2 = CD

Output of Gate G3 = AB + CD

From this we could then construct a truth table (Table 3) to calculate the output of the circuit. The truth table is constructed by considering the output of each gate in turn and then building up towards the complete output

Table 3. Truth Table

AB	C	D	AB	CD	AB+CD
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1

Alternatively, the output of the circuit can be evaluated by substituting values directly into the logic equation.

For example, when A = 1, B = 1, C = 1, D = 0

then Y = AB + CD = 1 . 1 + 1. 0 = 1 + 0 = 1

This can then be repeated for all other input combinations.

The analysis is followed by synthesis i.e. we will consider how to design and implement a logic circuit to enable it to perform the desired specified operation. In this instance, we start with the equation and determine circuit to implement. For example consider the logic function

X = AB + CDE

This is composed of two terms, AB and CDE . The first term is formed by ANDing A and B and the second term is formed by ANDing together C , D and E . These two terms are then ORed together. This can then be implemented using the AND and OR gates, as shown in Fig. 4. Generally, as the number of

levels are increased, the overall delay is increased due to the contribution of propagation delays at each gate.

Figure 4: Implementation of X=AB+ CDE

Analysis also can be categorized into the functional analysis (determine what is computed) and the timing analysis (determine how long it takes to compute it). The logic expression is manipulated using Boolean (or switching) algebra and optimized to minimize the number of gates needed, or to use specific type of gates

Canonical and Standard forms

A binary variable may be either in its true form or its complement . For n variables, the maximum number of input variable combinations is given by N = 2 n. Then considering the AND gate, each of the N logic expressions formed is called a standard product or minterm. As indicated in Table 1-13, binary digits '1' and '0' are taken to represent a given variable for example or its complement respectively. Also from Table 1-13 note that each minterm is assigned a symbol (P j) each where j is the decimal equivalent to the binary number of the minterm designated.

Similarly, if we consider an OR gate, each of the N logic expressions formed is called a standard sum ormaxterm. In this case binary digits '1' and '0' are taken to represent a given complemented variable and its true form respectively. As shown in Table 1-13, a symbol (S j) is assigned to each maxterm where j is the decimal equivalent to the binary number of the maxterm designated. Also observe that each maxterm is the complement of its corresponding minterm, and vice versa.

Input			Minterms		Maxterms	
A	B	C	Terms	Designation	Terms	Designation
0	0	0		P 0		S 0
0	0	1		P 1		S 1
0	1	0		P 2		S 2
0	1	1		P 3		S 3
1	0	0		P 4		S 4
1	0	1		P 5		S 5
1	1	0		P 6		S 6
1	1	1		P 7		S 7

The minterms and maxterms may be used to define the two standard forms for logic expressions, namely the sum of products (SOP), or sum of minterms, and the product of sums (POS), or product of maxterms. These standard forms of expression aid the logic circuit designer by simplifying the derivation of the function to be implemented. Boolean functions expressed as a sum of products or a product of sums are said to be in canonical form. Note the POS is not the complement of the SOP expression.

SUM OF PRODUCTS (OR of AND terms)

The SOP expression is the equation of the logic function as read off the truth table to specify the input combinations when the output is a logical 1. To illustrate, let us consider Table 6.

Row Number	Input			Output
	A	B	C	F
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	

6	1	1	0	1
7	1	1	1	0

Table 6

Observe that the output is high for the rows labelled 3, 5 and 6. The SOP expression for this circuit is thus given any of the following:

F = P 3 + P 5 + P 6

Each product (AND) term is a Minterm. ANDed product of literals in which each variable appears exactly once, in true or complemented form (but not both). Each minterm has exactly one '1' in the truth table. When minterms are ORed together each minterm contributes a '1' to the final function. Note that all product terms are not minterms.

PRODUCT OF SUMS (AND of OR terms)

The POS expression is the equation of the logic function as read off the truth table to specify the input combinations when the output is a logical 0. To illustrate, let us again consider Table 1-14. Observe that the output is low for the rows labeled 0, 1, 2, 4 and 7. The POS expression for this circuit is thus given by any of the following:

F = S 0 S 1 S 2 S 4 S 7

Each OR (sum) term is a Maxterm. ORed product of literals in which each variable appears exactly once, in true or complemented form (but not both). Each maxterm has exactly one '0' in the truth table. When maxterms are ANDed together each maxterm contributes a '0' to the final function. Please note that not all sum terms are maxterms

Logic Expression Minimization

The goal of logic expression minimization is to find an equivalent of an original logic expression that has fewer variables per term, has fewer terms and needs less logic to implement. There are three main manual methods used for logic expression minimization; algebraic minimization, Karnaugh Map minimization and Quine-McCluskey (tabular) minimization

Algebraic minimization

The algebraic minimization process is the application of the switching algebra postulates, laws, and theorems to transform the original expression. It is hard to recognize when a particular law can be applied and difficult to know if resulting expression is truly minimal. The incorrect implementation or dropped variables etc can easy lead to a mistake.

The following are two examples of the algebraic minimization process by exploiting the adjacency theorem. Look for two terms that are identical except for one variable in the following expression

Application removes one term and one variable from the remaining term

In the following example one can look for the adjacency

The first and third term differ only and
The third and fourth term differ only and
The second and third term differ only and
Duplicate 3rd. term and rearrange

Apply adjacency on term pairs

Karnaugh Map (or K-map) minimization
The Karnaugh map provides a systematic method for simplifying a Boolean expression or a truth table function. The K map can produce the simplest SOP or POS expression possible. K-map procedure is actually an application of adjacency and guarantees a minimal expression. It is easy to use, visual, fast and familiarity with Boolean laws is not required.
The K map is a table consisting of N =2 n cells, where n is the number of input variables. Assuming the input variable are A and B then the K map illustrating the four possible variable combinations is shown

Figure 5: Two variable K map
Similarly three variable and four variable K-maps can be constructed as shown below

Figure 5: Three variable and four variable K maps
For a SOP expression each cell represents one particular combination of the variables in product form. The table format is such that there is a single variable change between any adjacent cells. This is the characteristic the will determine adjacency. This method is typically applicable to limited number of variables (4 ~ 8) and for n > 5 the K map technique becomes impractical unless implemented on computer. Manual errors are possible in translation from Truth Table to K-map, or when grouping of cells not done correctly.
Basic K-map is a 2-D rectangular array of cells, each K-map represents one bit column of output and each cell contains one bit of output function. The arrangement of cells in array facilitates recognition of adjacent terms and adjacent terms differ in one variable value; equivalent to difference of one bit of input row values, e.g. m6 (110) and m7 (111). The standard Truth Table ordering does not show adjacency. One uses gray code for row order however, it is still hard to see all possible adjacencies. For any cell in 2-D array, there are four direct neighbors (top, bottom, left, right). The 2-D array can therefore show adjacencies of up to four variables. One should not forget that cells are adjacent top to bottom and side to side. The number of TT rows must match number of K-map cells. Watch out for ordering of 10 and 11 rows and columns.
To simplify a SOP for of a Boolean expression using a K map, first identify all the input combinations that produce an output of logic level 1 and place them in their appropriate K map cell. Consequently, all other cells must contain zero (0). Second, group the adjacent cells that contain 1 in a manner that maximizes the size of the groups but also minimizes the total number of groups. All 1's in the output must be included in a group even if the group is only one cell. Third, as each SOP term represents an AND expression, each (AND) grouping is written with only the input variables that are common to the group. Finally, the simplified expression is formed by ORing each of the (AND) groups.

When the input combinations are irrelevant or cannot occur, the output states are in the Truth table and the K map are filled with an X and are referred to as don't care states . The don't cares can work to our advantage during minimization; we can assign either 0 or 1 as needed. When simplifying K maps with

don't care states, the contents of the undefined cells (1 or 0) are chosen according to preference. The aim is to enlarge group sizes thereby eliminating as many input variables from the simplified expression as possible. Only those X's that assist in simplifying the function should be included in the groupings. No additional X's should be added that would result in additional terms in the expression. To illustrate let us consider the function specified by Table 8 and its corresponding K map shown in Fig. 8. Note that the two groupings determine that the simplified expression is expressed as

Table 8: Truth Table of the Function Input Ouput

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

If two cells have the same value and are next to each other, the terms are adjacent. This adjacency is shown by enclosing them. Groups can have common cells. Group size is a power of 2 and groups are rectangular. You can group 0s or 1s. If '1's are grouped, the expression will be a product term and '0's are grouped the expression will be a sum term. It is important to note when a variable values change as you go cell to cell. This determines how the term expression is formed by the following table. The bottom most row right side cell is having a value "1". It can be represented by the expression . Similarly, expressions for individual as well as grouped "1"s are shown in the figure

How the minimum expression of a function is determined using a Karnaugh Map? The concept of prime implicants can be used to determine the minimum solution. Single cells or groups that could be part of a larger group are known as implicants and a group that is as large as possible is a prime implicant. Single cells can be prime implicants is they cannot be grouped with any other cell. In the following Karnaugh map, the implicants and prime implicants are marked separately. The term is not a prime implicant because it can be combined with or

The minimum SOP expression for a function consists of some (but not necessarily all) of the prime implicants of a function. In other words, a SOP expression containing a term, which is not a prime implicant, cannot be the minimum. This is true because if a nonprime term were present, the expression could be simplified by combining the nonprime term with additional minterms. Any set of implicants that encloses (covers) all values is "sufficient"; i.e. the associated logical expression represents the desired function. For example, all minterms or maxterms are sufficient. However, the smallest set of prime implicants that covers all values forms a minimal expression for the desired function. There may be more than one minimal set.

5 Variable K Maps

A five variable K Map can be constructed using two 4 variable maps side-by-side. The groups spanning both maps occupy the same place in both maps

The Fig. is a map of
When checking for adjacencies, each term should be checked against the five possible adjacent squares

Transistor-Transistor Logic (TTL)

TTL is the short form of Transistor Transistor Logic. As the name suggests they refers to the digital integrated circuits that employ logic gates consisting primarily of bipolar transistors. The most basic TTL circuit is an inverter. It is a single transistor with its emitter grounded and its collector tied to VCCwith a pull-up resistor, and the output is taken from its collector. When the input is high (logic 1), the transistor is driven to saturation and the output voltage i.e. the drop across the collector and emitter is negligible and therefore output voltage is low (logic 0).

A two input NAND gate can be realized as shown in Fig.1a. When, at least any one of the input is at logic 0, the multiple emitter base junctions of transistor TA are forward biased whereas the base collector is reverse biased. The transistor TA would be ON forcing a current away from the base of TBand thereby TB is OFF. Almost all V CC would be dropping across an OFF transistor and the output voltage would be high (logic l). For a case when both Input1 and Input 2 are at VCC , both the base emitter junctions are reverse biased and the base collector junction of the transistor TA is forward biased therefore the transistor TB is on making the output at logic 0, or near ground .

However, most TTL circuits use a totem pole output circuit instead of the pull-up resistor as shown in Fig.1b. It has a VCC -side transistor (TC) sitting on top of the GND-side output transistor (TD). The emitter of the TC is connected to the collector of TD by a diode. The output is taken from the collector of transistor TD. TA is a multiple emitter transistor having only one collector and base but with multiple emitters. The multiple base emitter junction behaves just like an independent diodes. Applying a logic '1' input voltage to both emitter inputs of TA reverse-biases both base-emitter junctions, causing current to flow through R A into the base of TB, which is driven into saturation. When TB starts conducting, the stored base charge of TC dissipates through the TB collector, driving TC into cut-off. On the other hand, current flows into the base of TD , causing it to saturate and its collector emitter voltage is 0.2 V and the output is equal to 0.2 V, i.e. at logic 0. In addition, since TC is in cut-off, no current will flow from VCC to the output, keeping it at logic '0'. Since TD is in saturation, its input voltage is ~0.8 V. Therefore the output voltage at the collector of transistor T B is 0.8 V + VCESat (saturation voltage between conductor and emitter of a transistor is equal to ~0.2 V) = 1 V. T B always provides complementary inputs to the bases of TC and TD, such that TC and TD always operate in opposite regions, except during momentary transition between regions. The output impedance is low independent of of the logic state because one transistor (either TC or TD) is ON.

When at least one of inputs are at 0 V, the multiple emitter base junctions of transistor TA are forward biased whereas the base collector is reverse biased and transistor TB remains off and therefore the output voltage is equal to VCC . Since the base voltage for transistor TC is VCC , this transistor is on and the output is also VCC . And the input to transistor TD is 0 V, hence it remains off.

Figure 1: A 2-input TTL NAND Gate with a Totem Pole Output Stage

TTL overcomes the main problem associated with DTL (Diode Transistor Logic), i.e., lack of speed. The input to a TTL circuit is always through the emitter(s) of the input transistor, which exhibits a low input resistance. The base of the input transistor, on the other hand, is connected to the VCC , which causes the input transistor to pass a current of about 1.6 mA when the input voltage to the emitter(s) is logic '0'. Letting a TTL input 'float' (left unconnected) will usually make it go to logic '1'. However, such a state is

vulnerable to stray signals, which is why it is good practice to connect TTL inputs to VCC using 1 k pull-up resistors.

CMOS Logic

The term 'Complementary Metal-Oxide-Semiconductor' (CMOS), refers to the device technology for fabricating integrated circuits using both n- and p-channel MOSFET's. Today, CMOS is the major technology in manufacturing digital IC's and is widely used in microprocessors, memories, and other digital IC's. The input to a CMOS circuit is always to the gate of the input MOS transistor. The gate offers a very high resistance because it is isolated from the channel by an oxide layer. The current flowing through a CMOS input is virtually zero, and the device is operated mainly by the voltage applied to the gate, which controls the conductivity of the device channel. The low input currents required by a CMOS circuit results in lower power consumption, which is the major advantage of CMOS over TTL. In fact, power consumption in a CMOS circuit occurs only when it is switching between logic levels. Moreover, CMOS circuits are easy and cheap to fabricate resulting in high packing density than their bipolar counterparts. CMOS circuits are quite vulnerable to ESD damage, mainly by gate oxide punchthrough from high ESD (Electro static Discharge) voltages. Therefore, proper handling CMOS IC's is required to prevent ESD damage and generally, these devices are equipped with protection circuits.

Fig. 2(a) and Fig. 2(b) show the circuit symbols of an n-channel and a p-channel transistor respectively. An nMOS transistor is 'ON' if the gate voltage is at logic '1', or more precisely if the potential between the gate and the source terminals (VGS) is greater than the threshold voltage VT. An 'ON' transistor implies the existence of a continuous channel between the source and the drain terminals. On the other hand, an 'OFF' nMOS transistor indicates the absence of a connecting channel between the source and the drain. Similarly, a pMOS transistor is 'ON' if the potential VGS is lower (or more negative) than the threshold voltage VT..

Fig.2 : (a) symbol of an n-channel transistor (b) symbol of an a p-channel transistor (c) a CMOS inverter circuit (NOT gate)

Fig.2(c) shows a CMOS inverter circuit (NOT gate), where a p-channel and an n-channel MOS transistor is connected in series. A logic '1' input voltage would make T p (p-channel) turn off and T n(n-channel) turn on. Hence, the output will be low, pulling Output to logic '0'. A logic '0' Vin voltage, on the other hand, will make T p turn on and T n turn off, pulling Output to near V CC, or logic '1'. The p- and n-channel MOS transistors in the circuit are complementary and they are always in opposite states, i.e., when one of them is 'on' the other is 'off'.

Fig. 3 : (a) 2 input CMOS NAND Gate (b) 2 input CMOS NOR Gate

Figure 3(a) shows a 2 input CMOS NAND Gate where the TN1 and TP1 have the same input A and TN2and TP2 has the same input B. When both the A and B are high, TN1 and TN2 are ON and TP1 and TP2 are OFF. Therefore, the output is at logic 0. On the other hand, when both input A and input B are logic 0, TN1 and TN2 is OFF and T P1 and TP2 are ON. Hence, the output is at V CC , logic 1. Similar situation arises when any one of the input is logic 0. In such a case, one of the bottom series transistors i.e. either TN1 or TN2 would be OFF forcing the output to logic 1.

Figure 3(b) shows a 2 input CMOS NOR Gate where the T N1 and T P1 have the same input A and TN2and TP2 has the same input B. When both the A and B are logic 0, TN1 and TN2 are OFF and TP1 and TP2 are ON. Therefore, the output is at logic 1. On the other hand, when at least one of the inputs A or B is logic 1, the corresponding nMOS transistor would be ON making the output logic 0. The output is

disconnected from VCC in such case because at least one of the TP1 or TP2 is OFF. Thus the CMOS circuit of Figure 3(b) acts as a NOR Gate.

Logic Threshold Voltage Levels

The Fig. 3 provides a comparison between the Input and Output [I / O] logic switching levels for CMOS, and TTL logic families. VOH and VOL represent the high and the low logic output levels of a gate respectively. The regions of acceptable high and low voltages are determined by the VIH and VILvoltage levels, respectively. Consider the TTL logic, the range for VIL is from 0 to 0.8 V as shown and the range for VOL is from 0 to 0.35 V. The region between 0.8 V to 2.0 V is called undefined region. The range for VIH is from 2.0 V to V CC and the range of VOH is from 2.0 V to VCC. Similarly, for a CMOS the values can be inferred from the diagram.

If one use a CMOS IC for reduced current consumption and a TTL IC feeds the CMOS chip, then you need to either provide a voltage translation or use one of the mixed CMOS/TTL devices. The mixed TTL/CMOS devices are CMOS devices, which just happen to have TTL input trigger levels, but they are CMOS ICs. Let us consider the TTL to CMOS interface briefly. TTL device need a supply voltage of 5 V, while CMOS devices can use any supply voltage from 3 to 10 V. One approach to TTL/CMOS interfacing is to use a 5V supply for both the TTL driver and the CMOS load. In this case, the worst case TTL output voltages are almost compatible with the worst case CMOS input voltages. There is no problem with the TTL low state window (0 to 0.35 V) because it fits inside the CMOS low state window (0 to 1.3 V). This means the CMOS load always interprets the TTL low state drive as low.

The problem is with TTL high state, which can be as low as 2.0 V. The CMOS device needs at least 3.7V for high state input. Typically what is done is to use a pull-up resistor between the TTL driver and the CMOS load. The other end of pull up resistor is connected to VCC. When the TTL output is low, this pull up resistor does not have any effect on this output voltage. Nevertheless, when the TTL output is high, the pull up resistor raises this output to approximately 5 V.

Fig. 4 : Logic Threshold Voltage Levels

Similarly for a CMOS to TTL interface, one need to make sure that the CMOS low-state output is less than 0.8 V and the high-state out is greater than 2 V. Additionally, TTL operates with a low sate input current of ~1.6 mA which is a far too much current for a CMOS device to sink without entering the TTL indeterminate region. Therefore the solution would be to use a CMOS/TTL buffer.

Integrated Injection Logic (IIL , I2L)

Integrated Injection Logic eliminates the need for any resistors, capacitors or transistor isolation. This enables an extremely compact logic circuit to be formed which has low power consumption while maintaining the normal speed of transistor-transistor logic. I2L is built with multiple collector bipolar junction transistors. Although the logic levels are very close (High: 0.7V, Low: 0.2V), I2L has high noise immunity because it operates by current instead of voltage. It is also known as merged-transistor logic.

Emitter Coupled Logic (ECL)

Emitter coupled logic (ECL) gates use differential amplifier configurations at the input stage. (ECL) is a non saturated logic, which means that transistors are prevented from going into deep saturation, thus eliminating storage delays. In other words, the transistor is switched on, but not completely on. This is the fastest logic family. ECL circuits operate using a negative 5.2 V supply and the voltage level for high is -0.9 V and for low is -1.7V; thus biggest problem with ECL is a poor noise margin.

MEMORIES

Semiconductor memories are classified in different ways. A distinction is made between read-only (ROM) and read-write (RWM) memories. The contents RWMs can be changed in a short time for a virtually unlimited number of times and contents of ROMs are mostly useful for frequent reading and occasional

writing. Since RWM memories use active circuitry (transistors) to store the information, they belong to the class of called volatile memories. This is because the data would be lost when the supply voltage is turned off. Read-only memories, on the other hand, encode information by the presence or absence of devices. Their data cannot be modified and they belong to the class of nonvolatile memories. That means the stored data is lost by the disconnection of supply voltage.

Table 1 : Classification Semiconductor MemoriesRWM NVRWM ROM

Random Access	Non Random Access
SRAM	
DRAM	FIFO
Shift Register	EPROM
E2PROM	
FLASH	Mask-programmed ROM
Programmable ROM	

Based on the access pattern, RWMs are classified as random access class and serial memories. FIFO (first-in-first-out) is an example for serial memories. Most memories belong to the random access class, which means memory locations can be read or written in random order. One would expect memories of this class to be called RAM (random access memory); nevertheless for historic reasons, RAM has been reserved for random access RWM memories. That means though most ROM units also provide random access, but the acronym RAM should not be used for them.

VOLATILE MEMORIES

Static Random Access Memory (SRAM) and Dynamic Random Access Memory (DRAM) are volatile memories. SRAM is used as a cache memory in computers since it offers the fastest write/read (~8ns) speed among all memories. Hardware design of a single SRAM cell consists of 6 transistors. A DRAM cell consists of one transistor and one capacitor and it is based on the charge stored in a capacitor. It is superior to SRAM because of its low cost per bit storage; nevertheless it is slower (`50ns). In DRAM, the stored charge in the capacitor can be maintained only for few milli-seconds and therefore, an extra hardware circuit is needed to periodically refresh the data periodically.

Static Random Access Memory (SRAM)

A single SRAM memory cell is shown in Fig. 5. Two NMOS and two PMOS transistors (M1 to M4) forms the simple latch to store the data and two pass NMOS transistors (M5 and M6) are controlled by Word Line to pass Bit Line and into the cell.

A Write operation is performed by first charging the Bit Line and with values that are desired to be stored in the memory cell. Setting the Word Line high performs the actual write operation, and the new data is latched into the circuit.

A Read operation is initiated by pre-charging both Bit Line and to logic 1. Word Line is set high to close NMOS pass transistors to put the contents stored in the cell on the Bit Line and .

Transistors M1 to M4 constitute the latch and are constantly toggling back and forth. During these switching the power consumption in CMOS circuits takes place and therefore, the sizes of these transistors are kept as small as possible. NMOS transistors are basically switches opening and closing access to the SRAM cell. To minimize the propagation delay caused by these transistors their sizes are kept relatively larger.

Dynamic Random Access Memory (DRAM)

DRAM stores each bit in a storage cell consisting of a capacitor and a transistor. Capacitors tend to lose their charge rather quickly; thus, the need for recharging. The presence or absence of charge in the capacitor determines whether the cell contains a '1' or a '0'. The Read operation begins by precharging the bit line to an intermediate value, . The word line is raised to a high potential and the charge stored on capacitor is shared with that on the bit line. The change in the bit line voltage is given by the change on the bit line capacitor when the charge stored on capacitor C is shared with the bit line.

During the ' Write 1 ' operation, the word line is driven high, the bit line is tied to VDD corresponding to a 1, and voltage across C rises toward VDD - Vtn . To ' Write 0' in the cell, the bit line is grounded during the write operation and 0V is stored on capacitor C.

NON-VOLATILE MEMORIES

Based on the programmability of the devices non-volatile memories are categorized as follows. Writing data into ROMs is possible only at the time of manufacturing the devices and used only for reading the data stored. Even though these devices are less in cost the constraint that they are to be programmed at the time of manufacturing is an inconvenience. PROM devices are one time programmable ROM. At the time of device manufacturing every cell is stored with "1" and can be programmed by customer once. But, single write phase makes them unattractive. For instance, a single error in the programming process or application makes the device unusable.

EPROM is Erasable PROM. Multiple times programming feature is added in EPROM. In this case, first whole memory is to be erased by shining ultraviolet light. The erase process is slow and can take from seconds to several minutes, depending on the intensity of the UV source. Programming takes several (5-10) /word. EPROM cell is extremely simple and dense, making it possible to fabricate large memories at a low cost. EPROMs were therefore attractive in applications that not require frequent programming. Electrically-Erasable PROM (EEPROM) can be erased without removing from board, unlike in UV erasable where memory must be removed from the board. The voltage approximately applied for programming is 18V. In addition, it is a reverse process; means by applying high negative voltage at gate can erase the cell. Another advantage over EPROM is that EEPROM can be programmed for 105 cycles.

Flash Electrically Erasable PROM Technically the Flash EEPROM is a combination of the EPROM and EEPROM approaches. The main difference is that erasure can be performed for the complete chip, or for a sub-section of the memory. The control circuits on the memory chip can be regularly checked for the value of the threshold during erasure, and the erasure time can be adjusted dynamically. Flash technology has three basic weaknesses. First, its bulk erase nature prevents the use of normal byte-oriented update. Second, Based on the architecture used write and erase operations take different time and consume more power than read. Finally, each flash-memory block has a limitation on the erase cycle count.

Although transistors are used for realization of Read Only Memory, the functioning of Rom can be easily understood by diode matrix network depicted in Fig. 6. In this network, whichever switch is closed, those diodes will conduct and the output will be high (logic 1), sections where there is no diode connected there will be no current flowing and the output will be low (logic 0). For instance, when switch S5 is closed, the diodes D6 and D7 are on and therefore both output 1 and output 3 are at logic 1 and both output 2 and output 4 are at logic 0. Hence the corresponding binary number is 0101 and its decimal value is 5.

The disadvantage of a diode cell is that it does not isolate the bit line from the word line. For better isolation the diode can be replaced by gate-source connection of a NMOS transistor. Moreover, in order to achieve the programmability i.e. for multiple read write capability a modified transistor known as Floating Gate (FG) Transistor is employed. The structure is similar to a traditional MOS device, except that an

extra gate is inserted between gate and channel. The threshold voltage of the FG is programmable and corresponding to its different values the level 0 and level 1 can be identified.

Fig. 6 : A Diode ROM Matrix

Flash memory cells can be arranged in two popular architectures; the NOR and the NAND architectures as explained in the following sections.

NOR ARCHITECTURE

Here every cell is connected in NOR fashioned manner as shown in Fig. 4(a). Note that the transistors used are FG type and two gates can be seen in their symbols. Every source terminal of the transistor is connected to ground in NOR architecture. Metal lines are required between each individual cell to run the ground in NOR architectures and therefore they occupy more area. NOR-based flash has long erase and write times, but has a full address/data (memory) interface that allows random access to any location. This makes it suitable for storage of program code that needs to be infrequently updated, such as computers' BIOS.

NAND ARCHITECTURE

In the NAND structure, a series of floating gate transistors are connected between the bit line and ground line. This organization allows the elimination of all contacts to ground line and thus reducing the area by 40% compared to NOR architecture. It has faster erase and write times, higher density, and lower cost per bit than NOR flash. This can be obtained by arranging 8 to 16 floating gate transistors connected in series as shown in the Fig. 4(b). However, its I/O interface allows only sequential to data. This makes it suitable for mass-storage devices such as PC cards and various memory stick cards.