Accelerated Learning Series

(www.ALearnOnline.com – A site dedicated to education)

Modules in Computer Science & Engineering

Fundamentals in Digital Design

A set of notes detailing fundamental concepts in digital electronic design.

Dedicated To

Mum & Dad

FOR THEIR LOVE
AND
THEIR STRUGGLES IN LIFE

Author: SKG.
October 2005.

Revision History

Version 1.0 (Nov 2005)

- First version created

Version 1.1 (Jan 2006) – Revision based on student feedback

- Added eight rules on Karnaugh map grouping to section 2.4
- Typographical corrections

Table of Contents.

Prerequisites

- Designed for Grade 10 science students and above.

Preface

A hundred years from now when history judges our era, it is very likely that Software Engineering will take the credit for the innovations that define our time. But Software Engineers will in turn have to credit the Hardware Engineers for the General Purpose Microprocessor, without which the pace of innovation would never have come close to what we have achieved today. The General Purpose Microprocessor in turn must credit the technologies that allowed the Large Scale Integration of Solid State Electronic components. Solid State Electronics must in turn give credit to the Vacuum Tubes for defining the possibilities of Electronics. And so the list goes on.

Our quest for knowledge, through the apparatus of science, is a continuum of contributions and innovations. While the final goal of science is, arguably, the understanding of the absolute laws that govern matter and spirit, the journey of science provides several revelations along the way. Some of these revelations may contradict other revelations, but that does not take away its value to mankind. Niels Bohr (Nobel prize winner for Physics in 1922) once said that “the opposite of a correct statement is a false statement. But the opposite of a profound truth may well be another profound truth”. The scientific process of knowledge gathering inherently allows for the continual evolution in our understanding of matter and spirit.

In this module on the Fundamentals on Digital Design, we will get a broad overview of the many areas of science that come together in the technologies used to design modern digital electronic circuitry. At the completion of this module the student would be expected to have an adequate comprehension of the relevant basic scientific laws of matter and the derived technologies that exploit these laws in the manufacture of modern digital circuitry.

I will defer exercises in Application Specific Integrated Circuits and General Purpose Microprocessor designs to future modules. However the fundamentals required for both of these tasks will be adequately covered in this module. Hence this module will serve as a foundation and a prerequisite for all future modules in hardware design in the Accelerated Learning Series.

Acknowledgements

I am sincerely grateful to A. Walker for taking the time to review this document and for providing very valuable and constructive feedback.

1.0 – Fundamental Mathematics

If you can count, you probably have enough background to start this course on digital design. However you are probably used to counting elements in sets of 10. Digital electronics refers to a field of study where there are only 2 unique elements. So counting in sets of 2 (base 2) is a necessary prerequisite. So in this section on Fundamental Mathematics we will study number bases and in particular we will study base 2 and base 16 and how they relate to the base 10 numerals that we are normally accustomed to.

1.1 – Arithmetic and Number Bases

In the diagram below how many “X’s” are there?

X X X X X X X X X X X X X X X

X X X

If you count with Arabic numerals, you will probably start by counting 1, 2, 3 and so on and come up with an answer of 18. You used a series of unique symbols while counting from 1 to 9, but when you got to the X after the 9^{th} X, you decided it was “10”. You didn’t come up with a unique symbol for the 10^{th} X, but instead decided to call it 1 full set with a remainder of 0. Then when you counted the next X after the 10^{th}, you call it 11 or 1 full set with a remainder of 1. Finally when you finish counting all of them, you say it is one full set with a remainder of 8.

Let us try to understand the operation of counting a little better.

Every time we run out of a unique symbol we add 1 to the digit on the left. So what happens when we run out of unique symbols in the digit to the left of the first digit? We will add a 3^{rd} digit to the left of the 2^{nd} digit and so on.

The first digit (or the right most digit) also referred to as the “least significant digit” has a weight factor of “1”. In other words, the number in the least significant digit must be multiplied by “1” to get the total number of elements represented by that digit.

The second digit from the left has a weight factor of 10. In other words, the 2^{nd} digit multiplied by “10” gives us the total number of elements represented by that digit.

Similarly the total number of elements represented by the third digits can be calculated by multiplying its value with “100”.

The weight associated with any particular digit is 10 to the power of the position of the digit (also written as 10^N, where N is the position). The sign “18” implies $(1 \times 10^1) + (8 \times 10^0)$. The least significant digit has a position of “0” and the next digit to the left of it has a position of “1” and so on. This way of defining a full set as ten elements is referred to as the base 10 arithmetic.

What if we had only 8 unique symbols? These symbols will be 0,1,2,3,4,5,6,7. Then if we could count the X’s again, we will say we have 2 full sets with a remainder of 2 or 22. This is defined as base 8 arithmetic. In base 8, the sign “22” implies $(2 \times 8^1) + (2 \times 8^0)$. So the weight associated with any particular digit is 8 to the power of the position of the digit. Note that the position of the digit is always counted (starting at zero) from right to left.

What if we had only 4 unique symbols? These symbols will be 0,1,2,3. Then if we could count the X’s again, we will say we have 100 full sets with a remainder of 2 or 102. This is defined as base 4 arithmetic. In base 4, the sign “102” implies $(1 \times 4^2) + (0 \times 4^1) + (2 \times 4^0)$. So the weight associated with any particular digit is 4 to the power of the position of the digit.

What if we had only 2 unique symbols? These symbols will be 0,1. Then if we could count the X’s again, we will say we have 1,001 full sets, with a remainder of 0 or 10010. This is defined as base 2 arithmetic. In base 2, the sign “10010” implies $(1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$. So the weight associated with any particular digit is 2 to the power of the position of the digit.

What if we had 16 unique symbols? These symbols will be 0,12,3,4,5,6,7,8,9,A,B,C,D,E,F. Then if we could count the X’s again, we will say we have 1 full set, with a remainder of 2 or 12. This is defined as base 16 arithmetic. In base 16, the sign “12” implies $(1 \times 16^1) + (2 \times 16^0)$. So the weight associated with any particular digit is 16 to the power of the position of the digit.

1.2 - Converting from and to Base 10

Since most of us are used to base 10, it is often our base of reference. A value represented in base 10 will make inherent sense to us. So in this section we will discuss a mathematical method to convert from and to base 10.

In the last section we first counted the number of Xs in base 10. Then we counted in base 8, 4, 2 and 16 respectively. Each time we counted in a particular base, we regrouped the Xs into the number of unique digits available in that base. This regrouping can also be achieved mathematically by successive division.

For example to convert "18" in base 10 to base 2, we do the following successive division;

2)18
  9 remainder **0**
2)9
  4 remainder **1**
2)4
  2 remainder **0**
2)2
  1 remainder **0**
2)1
  0 remainder **1**

In other words, 18 in base 10 is equivalent 10010 in base 2. Note that the result is built up from right to left. This means the result of the first division is the right-most digit.

Converting the binary number back to decimal is equivalent to summing the weighted values of each digit in the binary number. The weights being determined by the position of the digit as discussed in the previous section.

So for example to covert 10010 in base 2 to base 10 we do the following sum;

$(1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 16 + 0 + 0 + 2 + 0 = 18$

Let us repeat the above example for base 16.

16)18
  1 remainder **2**
16)1
  0 remainder **1**

In other words, 18 in base 10 is equivalent to 12 in base 16

And to covert 12 in base 16 to base 10, we do the following sum;

$(1 \times 16^1) + (2 \times 16^0) = 16 + 2 = 18$

1.3 – Significance of Base 2, 4, 8 and 16

Digital computers only have 2 unique symbols. These symbols correspond to electrical energy potential measurements in Volts. But for simplicity we will say that the unique symbols are 0 and 1. In general a “0” will correspond to 0 Volts and “1” will correspond to +5 Volts. But the exact Voltage values are dependent on the type of electronic technologies used.

So if a digital computer were to count the Xs in section 1.1, it would come up with an answer of 10010, which is the same result we got when counting in base 2. Digital computers perform all their operation in binary (or base 2) arithmetic. Hence becoming proficient with binary arithmetic is very useful in understanding and trouble shooting digital computer behavior.

In the previous section we discussed the successive division method and the weighted multiplication method for converting numbers from and to base 10. However to covert numbers between bases 2, 4, 8 or 16 there is an easier technique. Learning this technique will prove very useful when dealing with digital computers.

Any number represented in binary can be converted to base 4 by dealing with 2 digits at a time starting from the right. For example to convert 10010 in base 2 to base 4, we can first convert the least significant 2 digits, which are “10”. This is a 2 in binary and 2 is a unique digit available in base 4. So the least significant 2 digits can be written as “2” in base 4. The next 2 digits are “00”. This is a ‘0” in base 2 and “0” is a unique digit available in base 4. So these two digits can be written as “0” in base 4. The next two digits are “01” (note adding a zero to the left has no value) and that is “1” in base 2 and 4. So the number 10010 in base 2 can be translated visually to 102 in base 4

01 00 10 (base 2) = 1 0 2 (base 4)

Similarly taking 3 binary digits at a time, we can visually convert binary numbers into octal (base 8) numbers.

010 010 (base 2) = 2 2 (base 8)

And taking 4 binary digits at a time, we can visually convert binary numbers into hexadecimal (base 16) numbers.

0001 0010 (base 2) = 1 2 (base 16)

One of the problems with binary numbers is that it can be very cumbersome to deal with, since even a relatively small number like 18 in base 10 will require 5 digits to represent it in binary. So using a higher base can prove very efficient in presentation. But there isn’t an easy visual way to convert from binary to base 10. Hence most computer professionals prefer to use base 16 or hexadecimal representation when presenting numbers.

The visual method of conversion can also be used to go from base 4, 8 or 16 to binary. The operation is exactly the inverse of the method used above to go from binary to a higher base.

So for example to convert a hexadecimal value of “12F” into binary, we take each digit and represent it by 4 binary digits. An “F” in base 16 is the same as “1111” in binary. A “2” in base 16 is the same as “0010” in binary. And a “1” in base 16 is the same as “0001” in binary.

1 2 F (base 16) = 0001 0010 1111 (base 2)

Learning the binary equivalent for any hexadecimal digit will prove very handy and so I have provided the conversions below. Also note that a base 16 representation is often prefixed with a “0x”. So the number “12F” in base 16 will be written as 0x12F. A binary number is often denoted by a terminating “b”. So 10010 in base 2 will be written as 10010b.

Hexadecimal	Binary	Hexadecimal	Binary	Hexadecimal	Binary
0x0	0000b	0x6	0110b	0xC	1100b
0x1	0001b	0x7	0111b	0xD	1101b
0x2	0010b	0x8	1000b	0xE	1110b
0x3	0011b	0x9	1001b	0xF	1111b
0x4	0100b	0xA	1010b		
0x5	0101b	0xB	1011b		

2.0 – Fundamental Logic

It was the Greek philosopher Aristotle who first proposed a system for reasoning and deriving truths. Aristotle defined a system where one starts with a known truth that is indisputable and always true. This starting point is referred to as a “premise”. From this premise he then defined a way to derive other truths by using a method of argumentation that classified a derived statement as either true or false. If a derived statement was classified as true, then it can effectively be a premise for a subsequent derivation. The uniqueness of this scheme is that there are only two possibilities for a derived argument. It is either true or it is false. It cannot be both at the same time. And it cannot be neither true nor false. This system for deriving truths was defined as “logos” or “logic” and was part of Aristotle’s greater dissertation on rhetoric which included “ethos” and “pathos”.

It was the British mathematician, George Boole who converted Aristotle logical system of reasoning into a mathematical form with well defined mathematical rules for deriving relationships between mathematical variables that conformed to the limitation that they represented only two possible values – true or false. This system of mathematics is referred to as Boolean Algebra.

It was almost a hundred years later, in the early 1900s, that Claude Shannon discovered that Boolean Algebra had an invaluable application in Digital Electronics, where the state of an electronic system was always defined as a “on” or “off”. At the time George Boole worked on Boolean Algebra, he would have never imagined the practical value of his efforts. And yet today, Boolean Algebra is indispensable in the design of digital circuits.

2.1 – Boolean Arithmetic

Let us start our study of Boolean algebra by defining the possible operations one can use in Boolean arithmetic. First of all the fundamental rule in Boolean algebra is that a variable can have one of two values – “0” or “1”. It cannot have any other value. Boolean arithmetic allows for addition and multiplication of Boolean variables. Note that subtraction and division are disallowed. To allow subtraction, we will need the use of negative numbers. But remember that Boolean variables can only be a “0” or a “1”. There are no negative numbers. And division is a compounded form of subtraction and hence that too is disallowed.

The rules for addition are as follow;

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1
1 + 0 + 1 + 1 + 1 = 1

Note that the first three operations should seem normal for someone used regular arithmetic. However the last two operations are odd. We have already stated that the only values a variable can have in Boolean Algebra are a “1” or a “0”. If you add two 1s, the sum cannot be expected to be a “0”. And the only other option available is “1” and so “1 + 1 = 1” in Boolean algebra. Further it does not matter how many variables you add, as long as any one of them is a “1” the answer is “1” as shown in the last operation above.

The rules for multiplication are as follow;

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1
1 x 0 x 1 x 1 x 1 = 0

When it comes to multiplication, the rules are identical to regular algebra.

Like regular algebra, Boolean variables can also be represented by names. A Boolean variable can be referred to as “A” or “B” or “C” and so on. And any variable can be defined to have a value of “1” or “0”. If the variable “A” has a value of “1” then the complement of “A” (also referred to as A-NOT and denoted as A’) will have its opposite value, which in this case would be a “0”.

In regular arithmetic there are some operations that always have a predefined answer like the addition of “0” to any variable will not change the value of that variable. These are referred to as identities or always true. Now let us discuss the identities in Boolean arithmetic. We will use “A”, “B” and “C” as Boolean variable in the illustrations below.

A + 0 = A
A + 1 = 1
A + A = A
A + A’ = 1

A x 0 = 0
A x 1 = A
A x A = A
A x A’ = 0

Complementing A an even number of times will always result in A.

A + B = B + A (Commutative property for addition)
A x B = B x A (Commutative property for multiplication)

(A + B) + C = A + (B + C) (Associative property for addition)
A x (B x C) = (A x B) x C (Associative property for multiplication)

A x (B + C) = (A x B) + (A x C) (Distributive property)
Addition in Boolean algebra is identical to an “OR” operation in digital design. An “OR” circuit will output a “1” if any of its input is defined as a “1”.

Multiplication in Boolean algebra is identical to an “AND” operation in digital design. An “AND” circuit will inspect all the voltage values at its input and define an output of “1” if all the inputs are “1”. If any input is “0”, the AND logic will output a “0”.

There is one operation in digital design that we have not discussed in the Boolean arithmetic operations above. This is the exclusive-OR operation. An “exclusive-OR” is the equivalent of (A x B’) + (A’ x B). Another way of looking at this is to think of this as an “OR” operation with a minor twist that if both A and B are “1”, then the output is “0”.

2.2 – Boolean Rules for Simplification

The most practical use for Boolean algebra in digital design is in the simplification of digital circuits. Much like regular algebra where we can eliminate the need for several variables based on our knowledge of identities, the same is true in Boolean algebra. And since digital electronics have variables that conform to the Boolean constraint of 2 possible values for any variable, the rules of simplifying Boolean algebra are also applicable to digital design. We will list some of these simplifications in this section. But how they relate to simplification in digital circuits will be become more evident in section 4.

Rule 1:

A + (A x B) = A

Proof: We know that "A x 0 = 0". We also know that "A x 1 = A". So depending on the value of "B", the possibilities in the above equation are "A + 0" or "A + A". And we know the answer to both is always "A".

Rule 2:

A + (A' x B) = A + B

Proof: From the last rule, we can rewrite A as "A + (A x B)". Then the rule above becomes "A + (A x B) + (A' x B)". Using the distributive property we can rewrite this as "A + (B x (A + A'))". But (A + A') is always 1. So this reduces to "A + (B x 1)". Which is the same as "A + B".

Rule 3:

(A + B) x (A + C) = A + (B x C)

Proof: "(A + B) x (A + C)" is the same as "(A x A) + (A x C) + (B x A) + (B x C)", which is the same as "A + (A x C) + (B x A) + (B x C)". But we know that "A + (A x B)" is A. So the equation becomes "A + (A x C) + (B x C)". Similarly "A + (A x C)" is also A. So the equation becomes "A + (B x C)".

Rule 4: (De Morgan's theorem)

(A x B)' = A'+ B'

Proof: The proof for De Morgan's theorem is long, but can be easily shown empirically for the 2 variable case.

Rule 5: (De Morgan's theorem)

(A + B)' = A' x B'

Proof: The proof for De Morgan's theorem is long, but can be easily shown empirically for the 2 variable case.

2.3 – Truth Tables

In the previous two sections we learnt the rules for Boolean arithmetic and Boolean logic simplification. The reason for this study was because the primary constraint in Boolean logic is that a Boolean variable can only assume one of two possible values which are “0” and “1” and this constraint is also true for variables in digital design. Hence the rules we learn in Boolean logic, can be applied directly to digital electronics as we will see in section 4.

In this section we will discuss a tabular methodology used in Boolean algebra for representing the results of Boolean operations on Boolean variables that has a direct application in digital design. This methodology is referred to as a Truth Table.

Let us assume that we have 2 Boolean variables “A” and “B”. We want to perform an addition operation using these two variables and assign the answer to a new variable “C”. Our possible results are as follows;

A	B	A + B = C
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 2.3.1

The above table is referred to as a **Truth Table**.

An operation we perform with Boolean variables is referred to as a “**connective**”. In the above operation our connective was an addition. But irrespective of how complicated a connective we use, we can see that the Truth Table allows us to tabulate all possible results for all possible operand values. We can even extend this truth table to cases where we have more than 2 input variables and more than 1 output variable.

As an example, let us use a Truth table to present the result for the following connectives. Here we have 3 input variables (A, B and C) and 2 output variables (D and E).

(A + B) x C = D
D’ = E

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

Fig. 2.3.2

Isn’t this great! As simple as “0” and “1”! Nothing too complicated!

2.4 – Karnaugh Maps

An alternative way to draw the table in Fig 2.3.1 in the previous section is as follows;

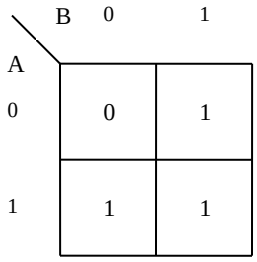


Fig 2.4.1

In this method, representing a Boolean expression consists of a box for every line in the truth table. In Figure 2.4.1 above, the binary values to the left side of the boxes represent the values for the variable “A”. The binary values above the boxes are the values for the variable “B”. And each box represents the value for the variable “C”, for a given combination of “A” and “B”. Such a pictorial representation of a Boolean function is called a **Karnaugh map**.

So why do we need another way to represent information in a truth table?

As it turns out, the Karnaugh map is an invaluable tool in simplifying Boolean expressions. But there is one very important rule to follow when constructing a Karnaugh map. Failure to follow this rule will make it useless in its role as a logic simplification tool.

Unlike a truth table where input variable values are typically listed as an increasing binary sequence (such as 00, 01, 10, 11), the input variable values in a Karnaugh map must be ordered such that the values for adjacent columns vary only by a single digit. So a sequence for 2 digits in a Karnaugh map would be 00,01,11, 10. Note how we went from “01” to “11” and not to “10”. This is because going from “01” to “10” requires two bits to change. This form of ordering where only a single bit changes at a time, is called a **gray code**.

Let us illustrate this by translating the truth table in Figure 2.3.2 in the previous section into a Karnaugh map for output D.

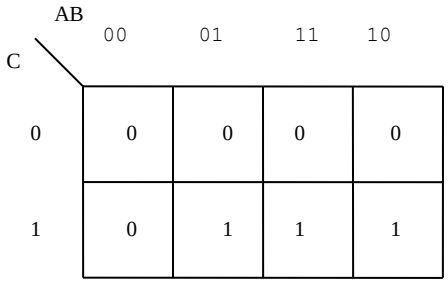


Fig 2.4.2.

Now to understand the benefit of Karnaugh map in logic simplification, let us try to reverse engineer the logic equation that represents the output D based on Fig 2.4.2 above. To help us do that, we will group all the “1”s in such a way that we can eliminate the relevance of certain input variables.

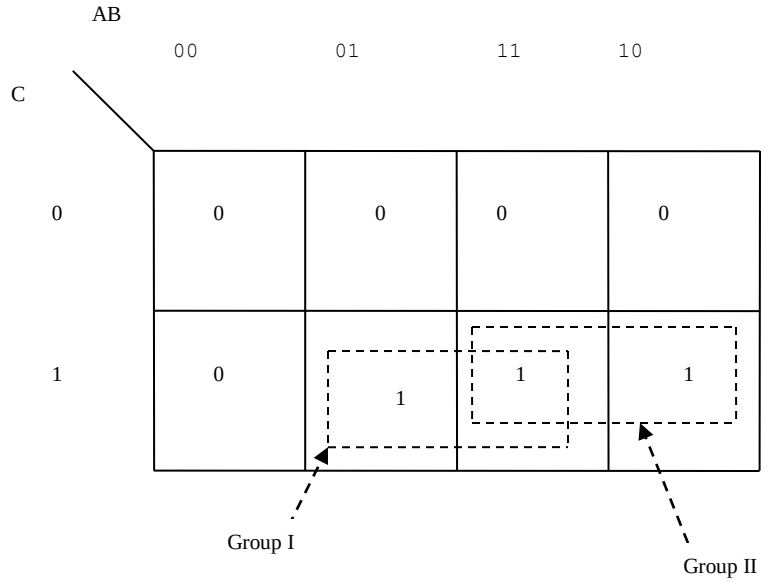


Fig 2.4.3 – Logic simplification with Karnaugh maps

In Fig 2.4.3, we have made 2 groups of 1s.

In Group I, the input variable A is both 0 and 1 and hence it is not relevant in determining the value of D in this group. So the value of D in Group I must have B **AND** C turned on. Or in other words D = B X C in group I.

Similarly in Group II the variable B is both 0 and 1 and hence it is not relevant in determining the value of D in this group. So the value of D in Group II must have A **AND** C turned on. Or in other words D = A X C in group 2.

But we know that D is a “1” in Groups I and II. So as long as the conditions for Group I **OR** the conditions for Group II are satisfied, D will be a “1”.

Hence the logic equation that represents D can be written as follow;

D = (A X C) + (B X C)

Or in other words…

D = (A + B) X C.

This matches the equation we started with in the truth table in Fig. 2.3.2.

Note that when deciding on groups in a karnaugh map the following eight rules must be observed.

1. Only cells with "1" must be included.
2. A group cannot be made of diagonals.
3. The number of cells in a group must be a power of 2.
4. Each group should be as large as possible.
5. Every "1" must be in at least one group.
6. Overlapping groups is allowed.
7. Cells in the edges wrap around.
8. Ensure that the total number of groups is the minimum possible.

3.0 – Fundamental Physics

Some of the earliest recorded study of matter dates back to 500 BC. Leucippus and Democritus, the early Greek philosophers, were among the first to suggest that the substance that formed the universe was made up of indivisible particles called “atoms” (Greek for indivisible) and empty spaces called “voids”. This theory was to counter an earlier theory by another popular Greek philosopher, Zeno, who believed that the universe was formed by an all encompassing and all permeating motionless mass with no empty spaces. Democritus argued that since our senses detected motion, empty spaces would be a prerequisite in the composition of the universe.

The views of Leucippus and Democritus would not however win the approval of the more prominent philosophers of the time. Aristotle for example, was unable to accept the idea of a “void” or the idea that atoms could not move on their own accord. And since the views of Aristotle would eventually make their way into early theology, that atomic view point was perceived as heretical.

By the 1500s the atomic theory made a great revival with support from influential philosophers like Pierre Gassendi, Robert Boyle and Isaac Newton. It was English chemist, John Dalton (1808) who first postulated on the modern atomic theory. John Dalton made the following five postulates;

1) All of matter consists of tiny indivisible particles called atoms.
2) All atoms of a particular element are exactly alike, but atoms of different elements are different.
3) All atoms are unchangeable.
4) Atoms of elements combine to form molecules of compounds.
5) In chemical reactions, atoms are neither created nor destroyed, but are only rearranged.

In 1897 English physicist J.J. Thomson proved Dalton wrong with the discovery of electrons, which could be extracted from atoms, thus making the atom divisible. In 1898 research by Pierre and Marie Curie led to the discovery of radioactivity which proved that some atoms could be changed (decayed) into other kinds of atoms. In 1911 the New Zealander physicist, Ernest Rutherford, contributed to our understanding of the nucleus of the atom. In 1922 the Danish physicist, Neils Bohr (student of Ernest Rutherford), provided an understanding of the patterns observed among elements of the periodic table.

Subsequent to the contribution by Neils Bohr, the study of the atomic structure became less a study in physical modeling and more a study in mathematical modeling. In 1923 French Physicist Louis De Brogile observed that electrons sometimes behaved as if they were particles and at other times behaved as if they were waves, leading to the wave-particle duality. In 1927 German Physicist Werner Heisenberg, proposed the principle of uncertainty which stated that you could not determine the velocity and position of a particle simultaneously. Heisenberg went on to publish “The Physical Principles of Quantum Theory”. In 1930 Austrian Physicist Erwin Schrodinger described electrons as continuous clouds, which led to “wave mechanics” as a mathematical model for the atom. Also in 1930, the English Engineer Paul Dirac expanded on Heisenburg’s publication by proposing “anti-particles”.

As you can see the knowledge of the atomic theory is a continually evolving process in our time. While the theories that we subscribe to today reconcile with empirical evidence thus far, it may not represent the absolute reality of nature. Nevertheless our current level of knowledge allows us the possibility of modifying atomic behaviors in a controlled manner with predictable results that have a wide variety of applications.

In this section, we will study the modern atomic theory and how it applies to the technologies in semiconductor physics and in the design of solid state electronic components. For this purpose, the physical models proposed by Rutherford and Bohr will prove adequate, and hence that will be the extent of our attempts at understanding the atomic structure.

3.1 – Bohr Atomic Structure

The Bohr's atomic structure is made up of subatomic particles called **Protons**, **Electrons** and **Neutrons**. These subatomic particles stay together with the help of an energy balance to form an atom. The energy balance is determined by the **weight** and **electric charge** associated with these subatomic particles. An electric charge is a fundamental property of a proton and an electron.

An **electron** represents a fundamental unit of negative charge (-1.6×10^{-19} Coulombs). The mass of an electron is 9.11×10^{-31} kg.

A **proton** represents a fundamental unit of positive charge ($+1.6 \times 10^{-19}$ Coulombs). The mass of a proton is 1.6726×10^{-27} kg. This represents a weight that is about 1,836 times the weight of an electron.

A **neutron** has no electric charge and a mass of 1.6749×10^{-27} kg, which is slightly higher than the weight of a proton.

An **atom** consists of a nucleus, made up of neutrons and protons, in its centre. Electrons surround the nucleus and rotate around the nucleus in electron "**shells**" (or energy levels) that vary in shape and distance from the nucleus. Each shell has **sub-shells**. And each sub-shell can have multiple electron **orbits**. Any electron orbital can accommodate 2 electrons.

The first shell is called the "**K**" shell. It has only 1 sub-shell called the "**s**" sub-shell and the "s" sub-shell only has 1 orbital. And hence the "K shell can accommodate a total of 2 electrons.

The second shell is called the "**L**" shell. It has 2 sub-shells – "**s**" and "**p**". The "p" sub-shell has 3 orbits and hence can accommodate a total of 6 electrons. So the total number of electrons in the 2nd shell is 8 (2 in "s" and 6 in "p").

The third shell is called the "**M**" shell. It has 3 sub-shells – "**s**", "**p**" and "**d**". The "d" sub-shell has 5 orbits and hence can have a total of 10 electrons. So the total number of electron in the 3rd shell is 18 (2 in "s", 6 in "p", 10 in "d").

The forth shell is called the "**N**" shell. It has 4 sub-shells – "**s**", "**p**", "**d**" and "**f**". The "f" sub-shell has 7 orbits and hence can have a total of 14 electrons. So the total number of electrons in the 4th shell is 32 (2 in "s", 6 in "p", 10 in "d", 14 in "f").

The fifth shell is called the "**O**" shell and it holds the same number of electrons as the fourth or "**N**" shell.

The sixth shell is called the "**P**" shell and it **does not have** the "**f**" sub-shell and hence holds a total of 18 electrons.

The seventh shell is call the "**Q**" shell and it **does not have** the "**d**" and "**f**" sub-shells and hence holds a total of 8 electrons.

The electron orbits represent energy levels. Electrons in the higher orbits represent a higher energy level. The order in which the electron orbits are filled are as follows;

Ks2, Ls2, Lp6, Ms2, Mp6, Ns2, Md10, Np6, Os2, Nd10, Op6, Ps2, Nf14, Od10, Pp6, Qs2, Of14, Pd10, Qp6.

Note that the "s" sub-shell in the "P" shell gets filled before the "f" sub-shell in the "N" shell.

The **atomic number** of an element defines the number of protons in the nucleus of an atom. An element is made up of identical atoms. So the number of protons or the atomic number is a way to identify an element.

The **atomic weight** of an element is the number of times an atom of that element, is heavier than an atom of hydrogen. The atomic weight of hydrogen is always 1 (Note: Hydrogen has 1 neutron, 1 proton and 1 electron).

The **Mass number** of an element is the sum of the protons and neutrons in the nucleus. This does not have to be a constant for a given element. An element can have varying number of neutrons in its nucleus. Eg.

Carbon 12 has 6 neutrons. Carbon 14 has 8 neutrons. But they both have 6 protons and hence they are both Carbon.

An **element** is a substance made up of atoms of just one kind. There are 82 natuarally-occuring elements and about 31 artificially created elements.

A **molecule** is the smallest particle in a substance that is formed by combining atoms of the same or different elements.

When atoms or molecules of different elements combine in a fixed proportion, a **compound** is formed. Eg. 2 atoms of Hydrogen combine with 1 atom of Oxygen to form water (H_2O).

An **ion** is any atom or group of atoms with a net positive or negative charge because it has either lost an electron (net positive charge) or gained an electron (net negative charge).

The **Valency** of an element is the number of hydrogen atoms that can combine with, or displace, one atom of the element to form a compound. Eg. 2 atoms of hydrogen combine with 1 atom or oxygen to form water. So the valency of Oxygen is 2. If an element does not combine with hydrogen, then its valency is determined by the combining power of the element with another element whose valency is known. Valency can also be defined as the number of electrons that an atom donates or accepts to complete the number of electrons required by a shell. Some elements can exhibit variable valencies.

Hydrogen and all metals have a **positive valency** – meaning they donate electrons to other atoms or molecules in a chemical reaction to form compounds. All non-metals have a **negative valency** – meaning they accept electrons from other atoms or molecules in a chemical reaction.

In general when 2 atoms or molecules combine, they attempt to end up with **2** or **8** electrons in the **outer most shell**. This is achieved by sharing of electrons from the donor and receiver.

Elements with increasing atomic numbers are arranged in a table called the **Periodic Table**. The elements of the periodic table are grouped according to their properties. Elements of group 1 have 1 valence electron. Elements in group 2 have 2 valence electrons and so on.

The **valence band** is the outer most electron energy level or band with electrons at the **absolute zero temperature** (for our purposes assume this means a very low energy state). The electrons that occupy the valence band are called valence electrons and they are the ones that are instrumental in making **bonds** with other atoms or molecules by donating or accepting electrons.

For conducting electricity (ie. for electrons to move away from the atom) however, the valence electrons have to jump to energy bands higher than the valence band by gaining energy (based on the application of an **electric field**) and thus making them free from the attraction of the protons in the nucleus. The energy band next to the valence band is called the **conduction band**.

In **conductors** the available energy bands for conduction range from the highest energy of the valence band upwards and so it is easy to move electrons from the valence band into the conduction band.

In **insulators** there is **band gap** that acts as a threshold preventing electrons from jumping easily from the valence band to the conduction band.

3.2 – The Semiconductor

A **semiconductor** is much like an insulator, in that it has a band gap. However, the band gap is much smaller than that in an insulator. The most common semiconductors are either from **group 4** of the periodic table (Silicon) or are compounds formed from elements on either side of group 4 such that the resultant compound also has a valency of 4.

The reduced band gap of a semiconductor can be exploited to cause the semi-conductor to behave in a variety of controlled ways by adding impurities to a semi-conductor. This process of adding impurities to a semiconductor is referred to as **doping** a semi-conductor.

If the impurity atoms have **more** valence electrons than the semiconductor atom, then an occupied band is created closer to the conduction band effectively reducing the band gap and allowing **electrons** to jump into the conduction band and carry electricity. This type of impurity addition is referred to as **n-type doping** (n for negative – electrons add a net negative charge).

If the impurity atoms have **fewer** valence electrons than the semiconductor atom, then an unoccupied band with energy just above the valence band is created. Electrons from the valence band can easily jump into this band, however once they are there, they are tightly bound to impurity atoms and can no longer jump into the conduction band. However the movement of the electron into this band leaves a net positive charge (referred to as a **hole**) which can also carry electricity in the presence of an electric field. This type of impurity addition is referred to as **p-type doping** (p for positive – holes add a net positive charge).

By sandwiching p-type and n-type semiconductors, many devices with very specific electronic properties can be manufactured. When people refer to **Solid State electronics**, they mean electronics based on components manufactured using such semiconductors. This is to distinguish similarly behaving devices that were previously made using vacuum tube technologies.

N-type doping generally involves **donors** from group 5 of the periodic table. These atoms have five valence electrons. Four of these can form a bond with four of the semiconductor's valence electrons leaving a weakly bound fifth electron. This electron needs very little energy to jump into the conduction band and become a carrier that is free to roam.

P-type doping involves **acceptors** from group 3 of the periodic table. These atoms have 3 valence electrons. Note that a pure semiconductor has 4 valence electrons and is bound to 4 adjacent atoms in a lattice crystal formation. When a P-type impurity is introduced, the impurity atom only has 3 valence electrons. But to fit into the existing lattice, it will create an empty electronic state in the band gap, near the top of the valence band. This will make it easy for an adjacent semiconductor atom's valence electron to jump into this energy level, leaving behind a net positive charge or a **hole**. Now the semiconductor atom representing the hole is short of an electron, much like what that impurity atom was previously. And this will cause another atom to lose an electron. The net effect is that the hole now moves freely around the crystal lattice.

3.3 – The Semiconductor Diode

A **semiconductor diode** is the simplest kind of semiconductor device which allows the flow of current in one-direction only. The device is made by putting a p-doped semiconductor next to an n-doped semiconductor as shown in Fig 3.3.1 below.

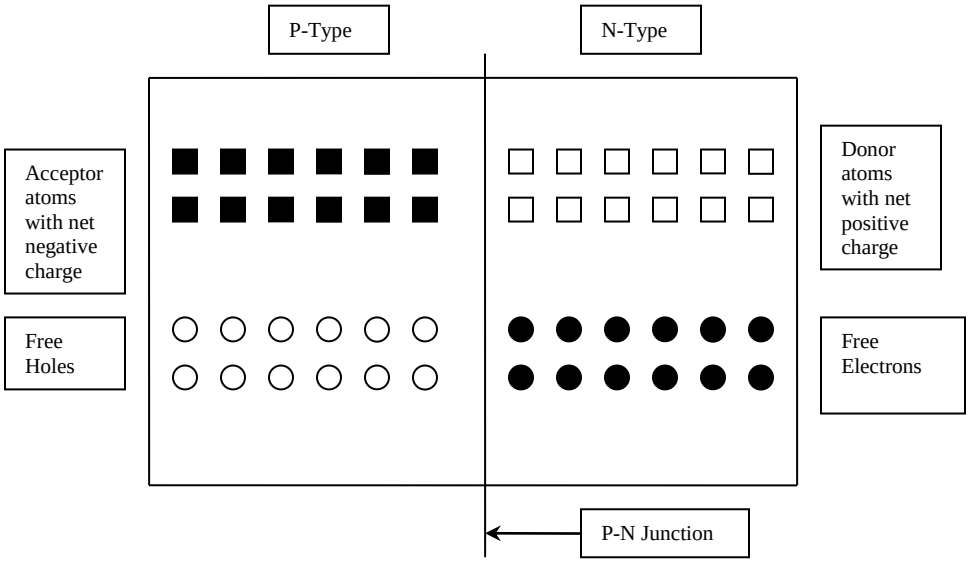


Fig 3.3.1 – P-N Junction

When such a P-N Junction is made, the free electrons close to the junction in the N-Type region will be attracted to the free holes in the P-Type region and will merge into them. This will lead to a loss of free charges close the P-N Junction, leading to what is commonly referred to as the **depletion region** as shown in Fig 3.3.2 below.

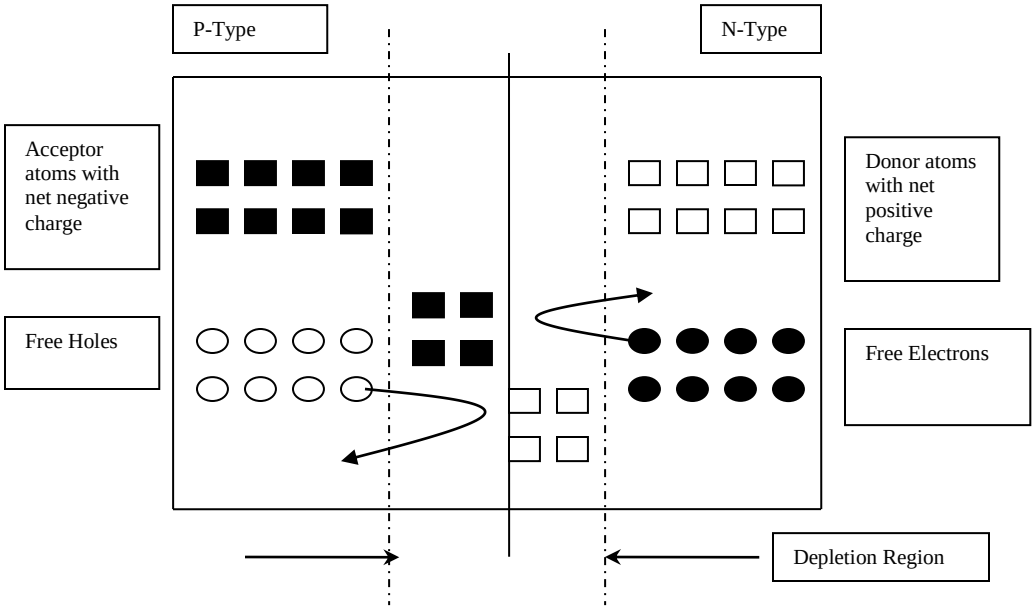


Fig. 3.3.2 – Depletion Region

The formation of the depletion region leads to a net negative charge in the P-Type region close to the junction because the holes (positive charges) have disappeared by merging with the electrons. Similarly there is net positive charge in N-Type region close to the junction, because the electrons in that region have disappeared.

Once the depletion region is formed, the free electrons in the N-Type region are further repelled from the depletion region because of the net negative charge in the P-Type region close to the junction. Similarly, the free holes in the P-Type region are further repelled from the depletion region because of the net positive charge in the N-Type region closer to the junction.

To move the holes and electrons across this junction now requires some form of extra energy. The junction thus acts as a barrier preventing the flow of charges across it. If an electric field is applied by placing a relative positive voltage at the N-Type region with respect to the P-Type region, the barrier effect will be further compounded because electrons will flow away from the barrier.

If on the other hand, a positive voltage was applied to the P-type region with respect to the N-type region, we will provide the energy required for the electrons and holes to cross the barrier formed by the depletion region. Hence this arrangement ensures that we only allow current to flow in one direction.

The application of a relative positive voltage on the P-type region is referred to as applying a **Forward bias**, while applying a relative positive voltage on the N-type region is referred to as applying a **Reverse bias**.

When a forward bias is applied, the depletion region can be visualized as effectively reducing in width in proportion to the value of the forward bias until such a point when a depletion region is non-existent. Any increase in forward bias at that point will no longer cause an increase in current flow.

When a reverse bias is applied, the depletion region can be visualized as effectively increasing in width in proportion to the value of the reverse bias.

Fig. 3.3.3 below shows a typical graph of the voltage applied, versus current flow in the forward and reverse directions.

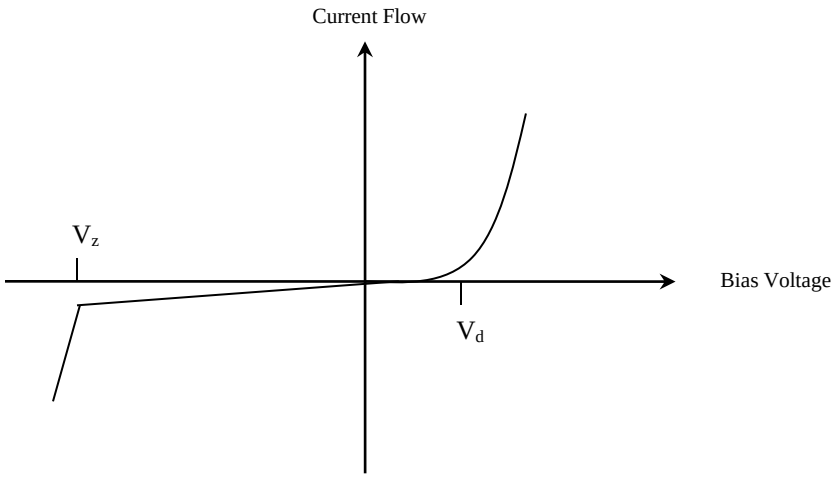


Fig. 3.3.3 – Bias Voltage vs. Current

Note how in the reverse direction, the diode “breaks down” and lets the flow of current after a certain point. This is referred to as the **Zener** effect and the voltage at which the breakdown happens is known as the Zener voltage. A diode that exploits this behavior is called a **Zener diode**. The reason for this breakdown or avalanche effect is that as the electric field increases, the speed of the electron flow increases. And when these high energy electrons collide with the atoms in the lattice, they can generate holes in the N-type region that can actually flow through the depletion region.

3.4 – The Semiconductor Transistor

The simplest way to view a Transistor is as a variable resistor. A transistor is a three terminal device and the resistance between two of the three terminals can be controlled either by the current or voltage at the third terminal. The variation of resistance, or the behavior of a transistor, can be categorized into three distinct states or regions. In the initial state, referred to as the **cut-off** state, the resistance between the two terminals of a transistor is very high and hence in this state the transistor behaves as a **switch that is turned off**. In the second state, referred to as the **linear** state, the decrease in resistance is linearly proportional to the current or voltage applied to the controlling terminal and hence this state is ideal for **controlled signal amplification**. And the third state, referred to as the **saturation** state, the amount of current is no longer impacted by the current or voltage in the controlling terminal. In this state the transistor can be viewed as a **switch that is turned on**.

A transistor can be used for mainly two purposes. First it can be used as an electrical **signal amplifier** and secondly it can be used as **switch**. In Digital Electronics the use of the transistor as a switch is the more common application.

There are two common types of transistors – **Bipolar Junction Transistors (BJTs)** and the **Field Effect Transistors (FETs)**.

Bipolar Junction Transistors (BJTs)

The BJTs can be viewed as two PN Junction diodes placed back to back to form a **NPN** or a **PNP** BJT. Fig 3.4.1 below shows an NPN BJT transistor.

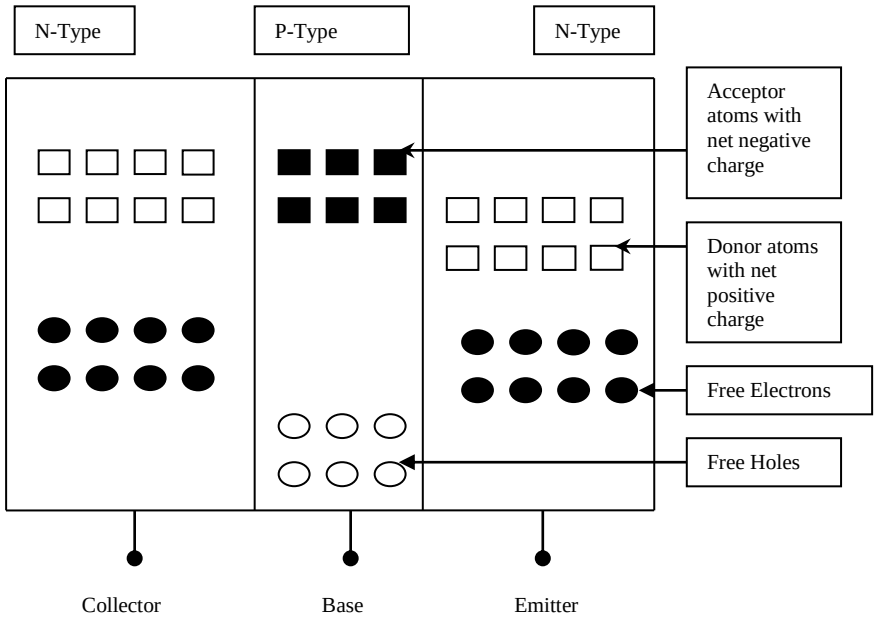


Fig. 3.4.1 – NPN BJT

The three terminals in a BJT are commonly referred to as the Base, Collector and Emitter. The reason for these names will become more evident as we understand the physics of its operation. When no electric field is applied to the terminals, there are two depletion regions in this device preventing the flow of an electric current through it.

However if the Base-Emitter junction is forward biased by applying a positive electric voltage at the Base relative to the Emitter, then electrons from the Emitter will flow into the Base and holes from the Base will flow into the Emitter. If the Base-Collector junction is reverse biased by applying a positive voltage to the collector relative to the Base, then the electrons that flow into the Base from the Emitter, can now flow into the collector. **This is the normal configuration of a BJT transistor when used for signal amplification.** A small current flow between the Base and the Emitter will can control a large current flow between the Emitter and the Collector.

Note that if there was no current flow between the Base and the Emitter, there will be no electrons available to cross the reverse biased junction between the Base and the Collector. So if the Base-Emitter is reverse biased, **the transistor will behave like a switch that is turned off.** This configuration is also called the **cut-off** region

If both the Base-Emitter and Base-Collector are forward biased, then the flow of current between the Emitter and the Collector is uninhibited and **the transistor will behave like a switch that is turned on**. The configuration is also called the **saturated** region.

The bulk of the electrons that flow from the Emitter to the Base will end up going into the Collector instead of merging with holes in the Base because the Base is often much thinner than the Emitter or the Collector. Note that in BJTs, it is the current flowing through the base that controls the current flowing through the Collector.

Fig 3.4.2 below shows a basic NPN BJT circuit along with the characteristic current curve through the Collector for varying values of voltages between the Collector and the Emitter. Note that until the Vce hits 0.7V, the Base-Collection junction is forward biased and hence the transistor behaves like a closed switch. Beyond that it functions as a linear amplifier, where the Collector current (I_C) is a constant multiplied by the Base current (I_B). If we keep increasing Vce, we eventually reach the Zener voltage and cause the transistor to break down.

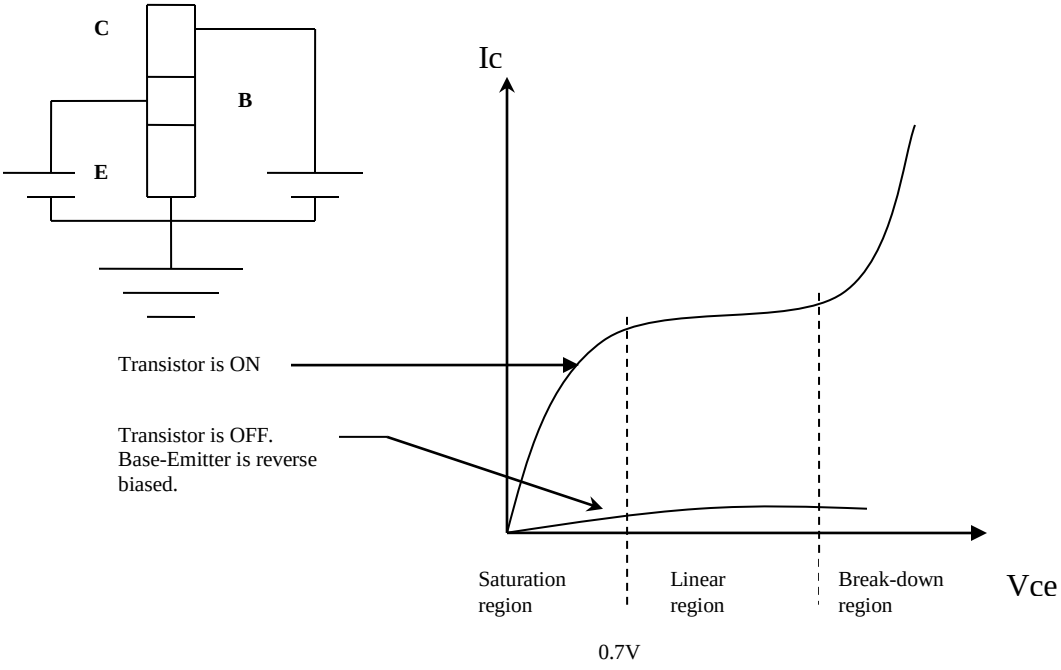


Fig 3.4.2 – BJT Current Characteristics

Note that in a BJT, the current is carried by both the holes and the electrons. In the case of a NPN BJT, the electrons are the **majority carriers** and holes are the **minority carriers**, since the Base (P) is very thin compared to the Emitter and Collector.

Field Effect Transistors (FETs)

Unlike a BJT, a Field Effect Transistor controls the current flow by the application of a voltage, which in turn creates an electric field, at the controlling terminal of the transistor. FETs come in different varieties. The Junction FET or **JFET** and the Metal Oxide Semiconductor FET or **MOSFET** are the most common types of FETs. In the following sections we will study the workings of each of these FET devices.

JFETs

Fig 3.4.3 below shows the layout of an N-channel JFET. Note that the terminals for a FET are named differently from that for a BJT. What used to be the Base is called the **Gate**, the Emitter is called the **Source** and the Collector is called the **Drain**. Also unlike a BJT, in the FET the current is carried only by the majority carriers. In an N-channel FET, the majority carriers will be electrons. In a P-channel FET, the majority carriers will be holes. The Gate only serves as a means to constrain the channel with the application of an electric field that will cause the depletion layer between the PN junction to be increased, thus reducing the channel width.

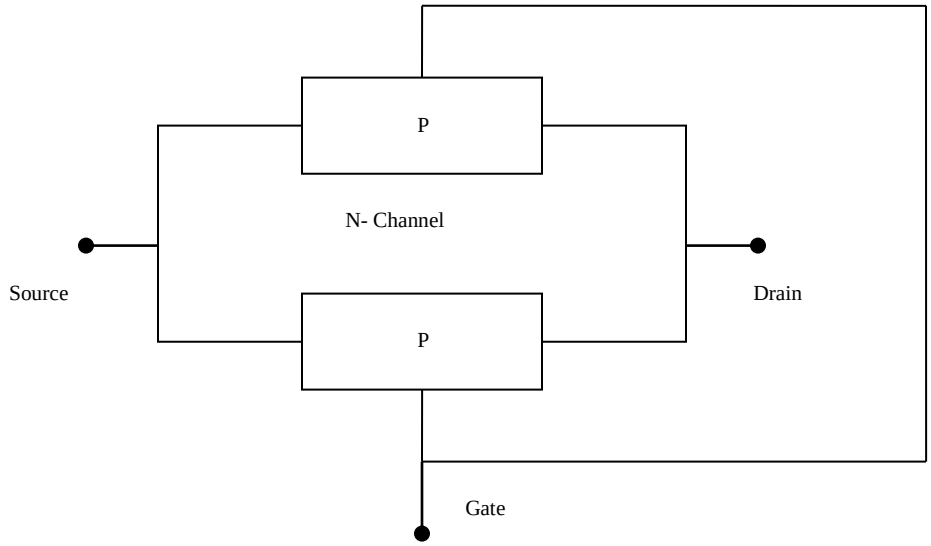


Fig 3.4.3 – N-Channel JFET

For an N-Channel FET a negative Gate to Source voltage will increase the depletion region and close the channel as this voltage is increased to a value called the **Pinch-off** voltage. For a P-channel FET a positive Gate to Source voltage will increase the depletion region and close the channel as this voltage is increased to the pinch-off value.

Fig 3.4.4 below show the current characteristics between the Source and the Drain for an N-Channel JFET, as a function of the voltage between the Source and the Drain for varying values of the voltage between the Gate and the Source.

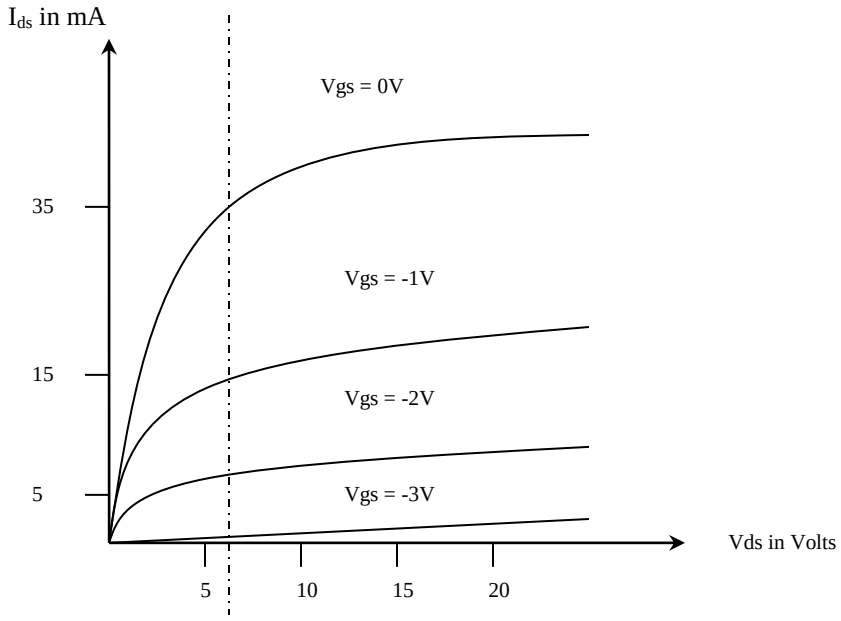


Fig 3.4.4 – N-Channel JFET Current Characteristics

For its application as a switch, we can turn off an N-channel JFET by applying a -3V to the Gate with respect to the Source. And we can turn it on by leaving the Gate and Source without a bias voltage.

Note that the behavior of a JFET to the left of the dotted line is much like any conductor that obeys ohm's law. But to the right of the dotted line the behavior of the JFET reaches a current saturation and no longer responds to increased Vds. In this region the JFET has applications as a voltage controlled current source.

MOSFETs

Unlike a JFET where the Gate is actually a P or N type semiconductor, the Gate of a MOSFET is a metal that is insulated from the semiconductor substrate with the help of an oxide and hence the name Metal-Oxide-Semiconductor. Fig 3.4.5 illustrates the construction of a MOSFET.

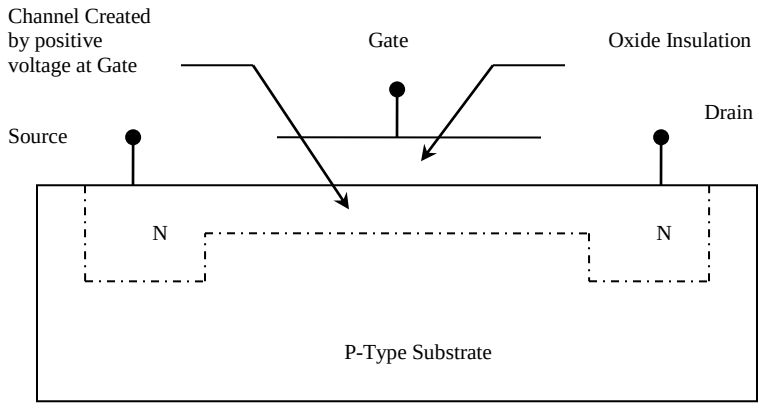


Fig 3.4.5 – N-MOSFET Transistor

Note that in a MOSFET the channel is created by applying a voltage at the Gate. Prior to the application of this electric field there is no channel in a MOSFET transistor. The Source and Drain are wells of N or P type

doped material and the rest of the substrate is of the opposite doping. In the case of an N-type Source and Drain wells (N-MOS), as shown in Fig 3.4.5, a positive voltage at the gate will cause the holes in the P-type substrate to be repelled and electrons from the Source and Drain well will be able to move freely in a thin layer close to the gate. This is how a channel in created in a MOSFET. In the case of a P-MOS, a negative voltage will have to be applied at the Gate to create a channel between the Source and the Drain.

Fig 3.4.6 below shows the current characteristics for an N-MOSFET transistor.

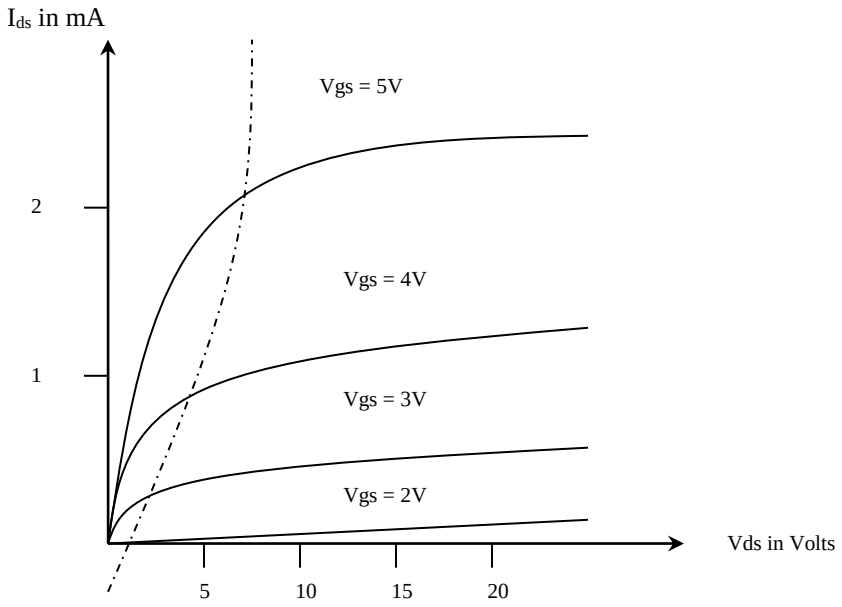


Fig 3.4.6 – N-MOSFET Current Characteristics

For its application as a switch, note that you need a positive voltage at the gate to turn a N-MOS transistor on. Similarly you need a corresponding negative voltage at the gate to turn a P-MOS transistor on. The actual voltage values required to turn these transistors on and off are functions of the silicon construction. But the fact that the inverse gate voltage requirement for a P-MOS versus an N-MOS is exploited in CMOS transistors discussed in the next chapter.

Fig 3.4.7 shows the commonly used electrical notations for N-MOS and P-MOS transistors respectively.

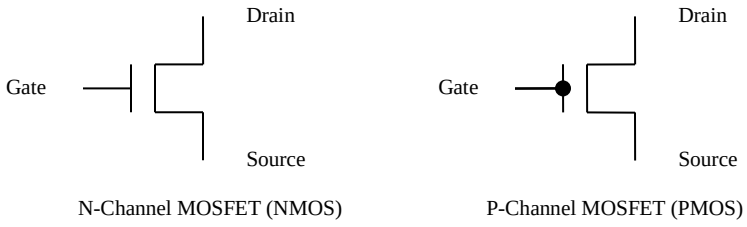


Fig 3.4.7 – Electrical symbols for NMOS & PMOS

3.5 – Gate Implementation using CMOS

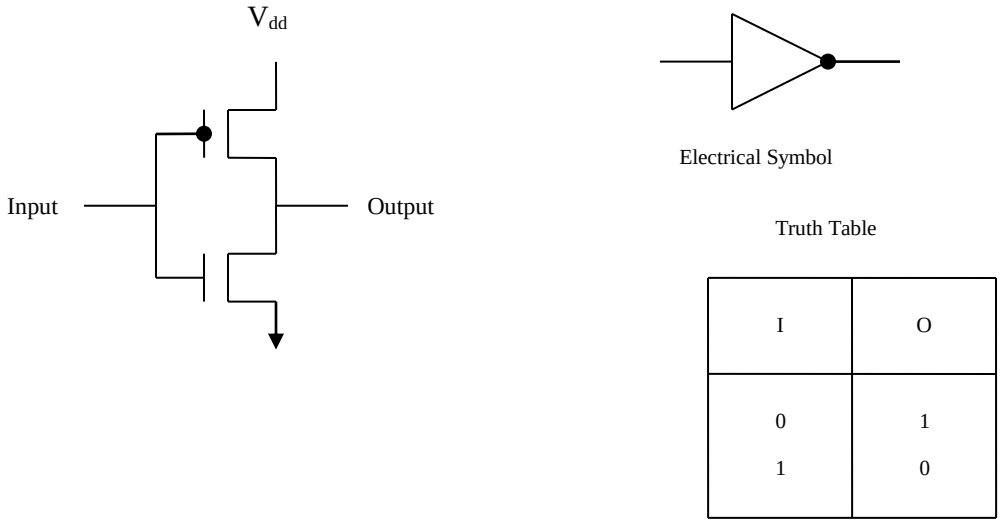
Complimentary MOS or **CMOS** is a configuration that uses NMOS and PMOS technologies in concert to realize common logic gates.

A logic gate is an electrical circuit that allows the fundamental operations in Boolean Algebra that we discussed in section 2. The logic gates that we will cover in this section are the NOT Gate, 2-Input NAND Gate, 2-Input NOR Gate, the Transmitter and the D-Flip-Flop. With these five fundamental gates, we have all the tools necessary to build very complicated Digital Circuits.

The primary merit of the CMOS technology relates to power consumption. When NMOS or PMOS are not used in concert, an open gate will allow current flow and thus lead to power consumption (Ohm's law - Power = I^2R). But if we use NMOS and PMOS in series such that when one gate is open the other is closed we can avoid the flow of current in steady state and hence conserve power. This will become clearer as we study each of the building block gates below.

NOT Gate

Fig 3.5.1 below shows how a PMOS and an NMOS can be used in series to implement a NOT gate (Inverter).



I	O
0	1
1	0

Fig 3.5.1 – CMOS NOT Gate implementation

Since we know that PMOS and NMOS are turned on by inverse signals, in the above layout if the "Input" value is high (usually 5V), then the NMOS transistor will be ON while the PMOS will be OFF. And hence the Output will be OFF (equal to ground since the Source of the NMOS is tied to ground and the NMOS is a closed circuit).

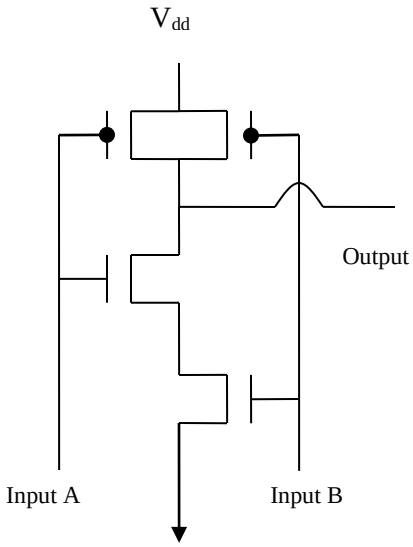
Similarly if the "Input" value is low (usually 0V), then the NMOS transistor will be OFF while the PMOS will be ON. And hence the "Output" will be ON (equal to V_{dd} because PMOS is a closed circuit and its Drain is tied high).

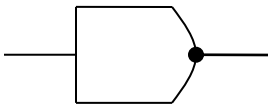
The beauty of this setup is that power is only consumed when the Input value changes state. No power is required to maintain a state. This is because there is no current flow while maintaining a state. This is not the case if NMOS or PMOS were used in a non-complimentary fashion.

Note that in this design a NOT gate consumes 2 transistors.

2-Input NAND Gate

Fig 3.5.2 below shows the implementation of a NAND gate (AND followed by a NOT).





Electrical Symbol

Truth Table

A	B	O
0	0	1
0	1	1
1	0	1
1	1	0

Fig 3.5.2 – CMOS NAND Gate implementation

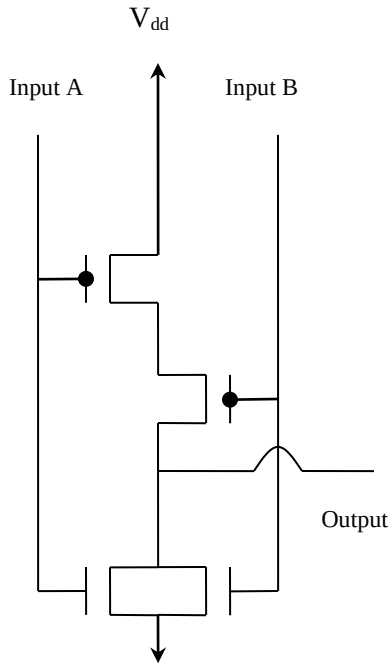
To confirm the truth table above let us study the circuit. For the "Output" to be low, both the NMOS gates need to be open. This can only happen when both Input A and Input B are high. If either Input A or Input B is low, then one of the PMOS transistors will be open and the output will be high.

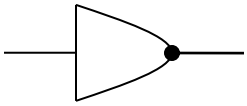
Again observe that there is no current flow while maintaining a state.

Note that in this design a NAND gate consumes 4 transistors. In general when design engineers refer to the gate count in an integrated circuit, they are talking about the number of NAND gate equivalents. Thus to get a measure of the number transistors for a given gate count, you can get a rough estimate by multiplying the gate count by 4.

2-Input NOR Gate

Fig 3.5.3 below shows the implementation of a NOR gate (OR followed by a NOT).





Electrical Symbol

Truth Table

A	B	O
0	0	1
0	1	0
1	0	0
1	1	0

Fig 3.5.3 – CMOS NOR Gate implementation

To confirm the truth table above let us study the circuit. For the “Output” to be high, both the PMOS gates need to be open. This can only happen when both Input A and Input B are low. If either Input A or Input B is high, then one of the NMOS transistors will be open and the output will be low.

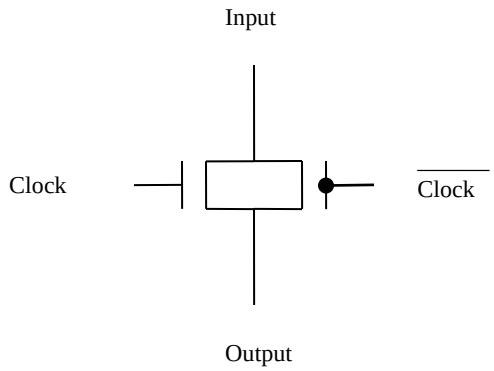
Again observe that there is no current flow while maintaining a state.

Transmission Gate

Fig 3.5.4 below shows the implementation of a Transmission gate (a gate that allows a signal to flow through on a rising clock edge). The application for this gate will become clear when we discuss the D-Flip-Flop. But for now assume that it does nothing more than transmit the input signal to the output when the clock is high.

The concept of a **clock** is what distinguishes **combinational logic** design from **sequential logic** design. This will be discussed in section 4. For now, it is sufficient to understand that a clock is a periodic rectangular pulse that is generated using a **crystal oscillator**.

Circuits that change output state only on a rising or a falling edge of a clock signal are referred to as synchronous circuits. Circuits that change output state as soon as the input signal changes, are referred to as Asynchronous circuits. The significance of **Synchronous** and **Asynchronous** designs will be covered in Section 4.



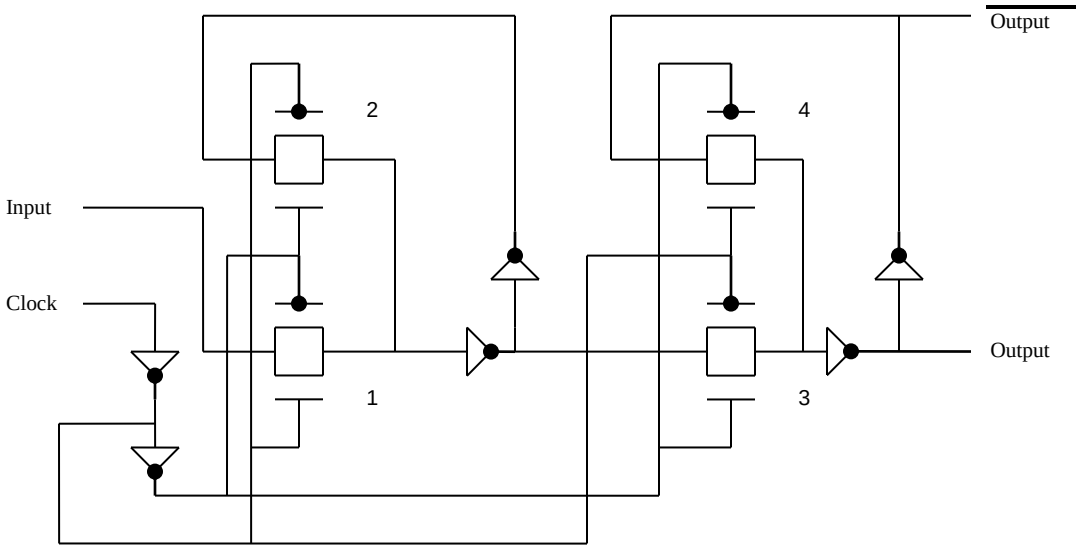
Truth Table

I	O
0	0
1	1

Fig 3.5.4 – CMOS Transmission Gate implementation

D-Flip-Flop

Fig 3.5.5 below shows the implementation of a D-Flip-Flop.



Truth Table

I	C	O
0	0	P
0	1	0
1	0	P
1	1	1

Electrical Symbol

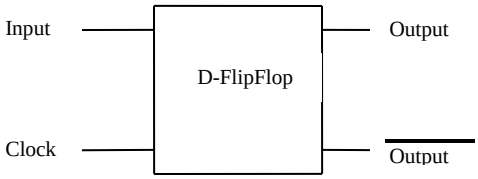


Fig 3.5.5 – CMOS D-Flip Flop implementation

A D-Flip-Flop is a very useful gate that allows you to "**lock**" or "**latch**" your signal for a period of time (generally the period of the clock). Thus even if your input signal changes during the clock period, the output remains unchanged until the clock goes from a low to a high. This is very useful in avoiding "**glitches**" that can lead to very unstable circuits especially when the outputs drive heavy loads. These concepts will become much clearer in Section 4.

To understand how the D-Flip-Flop allows for its memory capability, let use study the circuit in Fig. 3.5.5. When the clock is low, gate 1 will transmit and gate 2 and 3 will block signals. Hence the input appears at gates 2 and 3 but does not get transmitted by them. When the clock goes high, gate 1 will cease transmitting and gates 2 and 3 will begin transmitting. The input signal is "remembered" by the loop formed through gate 2. This memorized value passes through gate 3 and appears at the input of gate 4. When the clock goes low again, the output of the Flip-Flop is "remembered" by the loop through the now transmitting gate 4. Thus the signal appears at the output delayed.

In the truth table in Fig 3.5.5, I have indicated that the Output is "P" when the clock is low. This is to account for the Flip-Flop maintaining whatever was the previous value that was latched when the clock went from a low to a high.

4.0 – Concepts in Digital Design

In sections 1, 2, and 3, we discussed the prerequisite sciences that are employed in digital circuits. In this section, we will discuss the engineering concepts used in the realization of logic designs.

First we will discuss the most basic asynchronous combinational circuits and reveal their applications and limitations. This will lead us to synchronous designs. We will then discuss “state aware” or sequential circuits and the triggers that can be used to provide sequential behavior. In particular we will discuss the most common from of triggers - event and clock triggers.

To reinforce the concepts that we discuss, we will use real-world applications as examples for each type of design technique.

4.1 – Asynchronous Combinational Logic Design

Almost all user interfaces to digital circuits use some form of a 7-Segment Liquid Crystal Display (LCD) to display numbers and characters. The most common application for this is in digital watches. Here a Binary-Coded-Decimal (BCD) value is used to light appropriated LCD elements in the 7-Segment display. We will use the 7-Segment display as an example to discuss combinational logic circuits.

Fig 4.1.1 below shows a 7-Segment LCD display. The 7 segments are labeled A through G.

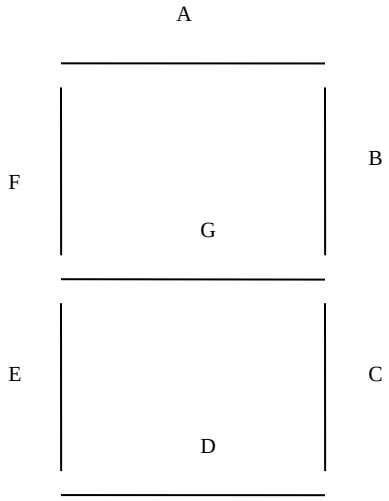


Fig 4.1.1 – 7– Segment LCD Display

A Binary Coded Decimal (BCD) is a 4 bit value that is used to represent decimal numbers. With 4 bits you can represent 2^4 or 16 unique values. But as we already know, a decimal value (base 10) can only have 10 unique values. And so 6 out of the 16 possible values are never used in the BCD representation.

Let us start our design of the BCD-To-7-Segment display by first creating a Truth Table. We will call our 4 input bits I3, I2, I1 and I0, where I0 is the least significant bit and I3 is the most significant bit. Our 7 outputs will be A,B,C,D,E,F and G.

When I3=0, I2=0, I1=0 and I0=0, we want to display a "0". To do this we need all segments to be lit up except G. This is shown in the first row of the truth table in Fig 4.1.2.

I3	**I2**	**I1**	**I0**	***A***	***B***	***C***	***D***	***E***	***F***	***G***
0	0	0	0	*1*	*1*	*1*	*1*	*1*	*1*	*0*
0	0	0	1	*0*	*1*	*1*	*0*	*0*	*0*	*0*
0	0	1	0	*1*	*1*	*0*	*1*	*1*	*0*	*1*
0	0	1	1	*1*	*1*	*1*	*1*	*0*	*0*	*1*
0	1	0	0	*0*	*1*	*1*	*0*	*0*	*1*	*1*
0	1	0	1	*1*	*0*	*1*	*1*	*0*	*1*	*1*
0	1	1	0	*1*	*0*	*1*	*1*	*1*	*1*	*1*
0	1	1	1	*1*	*1*	*1*	*0*	*0*	*0*	*0*
1	0	0	0	*1*	*1*	*1*	*1*	*1*	*1*	*1*
1	0	0	1	*1*	*1*	*1*	*1*	*0*	*1*	*1*

Fig 4.1.2 – BCD to 7-Segment Display Truth Table

Similarly when I3=0, I2=0, I1=0 and I0=1, we want to display a "1". To do this we need to light up segments B and C and turn off all other segments. This is shown in the second row of the truth table in Fig 4.1.2.

If you go through this exercise for the decimal numbers 1 through 9, you should come up with the table in Fig 4.1.2. Technically speaking, you should also account for the hex numbers A through F when using 4 bits for inputs. However for the illustrative purposes here, we will ignore those states. Ignoring those states will help reduce gate count in our design. But it can prove to be a nightmare when testing the final circuit.

Now let us use a Karnaugh map to design a circuit with 4 inputs and 1 output. The 4 inputs will correspond to our 4 BCD bits (I3, I2, I1, I0). And the 1 output will correspond to LCD segment A.

From the truth table in Fig 4.1.2, we know that A is “0” in the following cases;
I3, I2, I1, I0 = 0001 and 0100
In all other cases that we care about A is “1”.

What about the cases we don’t care about? Note that in the truth table we have not listed any input values greater than 1001. These include 1010, 1011, 1100, 1101, 1110 and 1111. For all these values, we will put an “X” to indicate that we don’t care if these input values correspond to a “0” or a “1”.

Fig 4.1.3 below is a Karnaugh map that shows the values for A for all possible values of I3, I2, I1 and I0.

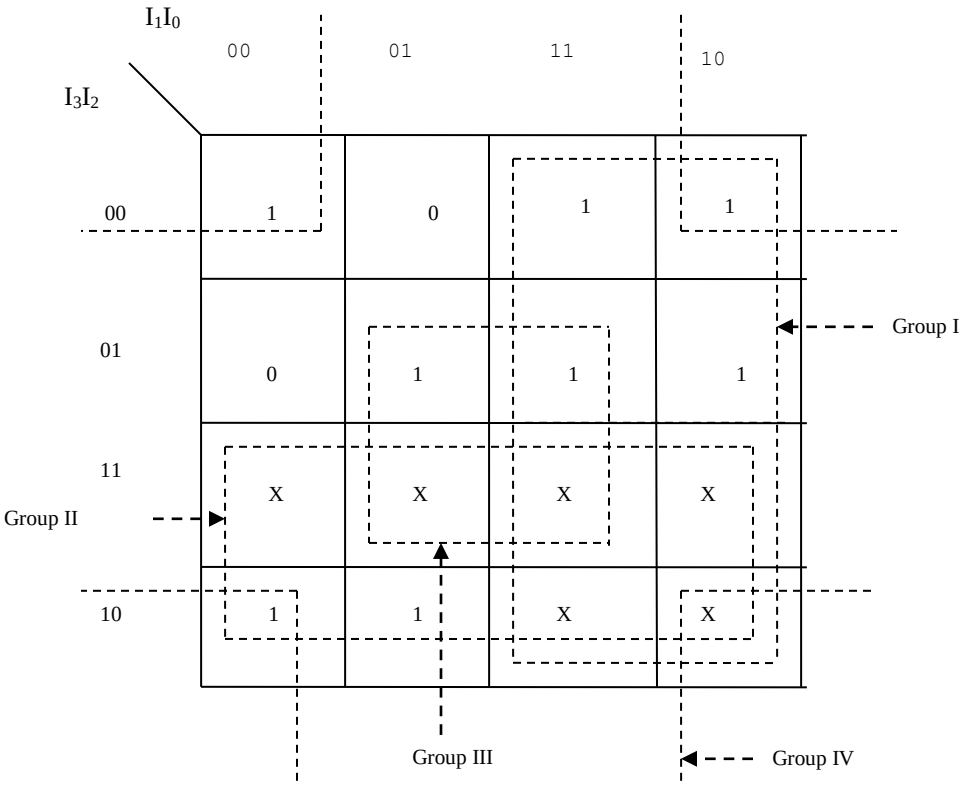


Fig 4.1.3 – Logic Design with Karnaugh maps

Notice that we have managed to get 4 groups of 1’s. To get the four groups, we have assumed that our “don’t care” (X) values can as well be a “1”. This is an example of using “don’t care” values to ease the complexity of a circuit and hence reduce the number of gates required to build a circuit.

In Group I, the only input value of relevance is I1 and it has to be a “1”
In Group 2, the only input value of relevance is I3 and it has to be a “1”
In Group 3, the only input values of relevance are I2 and I0 and both have to be a “1”
In Group 4, the only input values of relevance are I2 and I0 and both have to be a “0”

So we can summarize our logic as follows;

A = I1 + I3 + (I2 X I0) + (I2-NOT X I0-NOT)

Fig 4.1.4 shows a logic implementation for the above equation using NOT, AND and NOR gates.

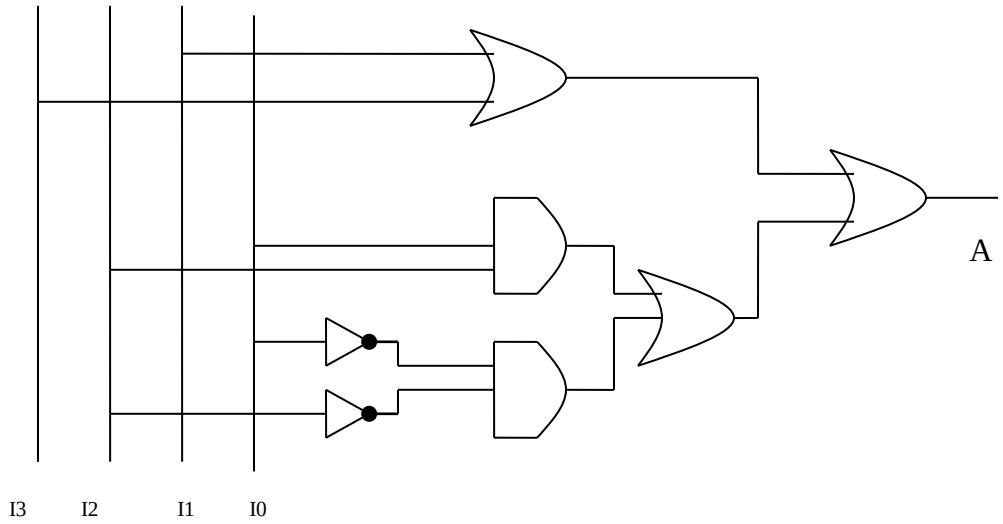


Fig 4.1.4 – Asynchronous Logic implementation

As an exercise you can now follow the same technique above and design circuits for the remaining 6 segments (B through G) in the 7-segment LCD display.

The circuit we have just designed is a **combination** of NOT, AND and OR gates. As we change the input values for I3, I2, I1 and I0, the value of the output A will change as quickly as the gates will allow the changes to occur. There is nothing in the circuit that attempts to synchronize the output with a notification that the changes being made to the input are complete. Such a circuit is referred to as an **asynchronous combinational logic circuit.**

In a combinational circuit, if the input lines don't change all at the same time or if the delay in the different paths from the input to the output are different, then we can expect some jitter and glitches in the value of A, until the input lines settle down.

To fully appreciate the problem with this jitter in the output line, let us consider an application for our 7-segment display in a sports stadium where the referee changes the input values based on the current game score. Even if our referee is very good with binary numbers, it is not humanly possible to flick the 4 switches all at once to the values that it ought to represent. What this means is that as the referee is changing the score, the 7-segment display will show invalid score values. This can be quiet alarming and unacceptable to the devoted fans at the stadium!

If output "A" was an input that drove a motor in a saw mill, the implications of the glitches in the value of "A" could have even more serious consequences. Imagine an electric saw going on when you are not expecting it to!

To account for these problems, most digital designs use some form of synchronization technique to ensure that the output values are only changed when it is safe to do so. We will explore one such technique in the next section.

4.2 – Synchronous Combinational Logic Design

Now that we recognize the limitations of the asynchronous circuit in Fig 4.1.4, let us try and enhance the circuit such that the output only changes when we want it to reflect the current input values, as opposed to changing the output as quickly as the input values change. To do this we will use the D-FlipFlop that we developed in section 3.5.

Fig 4.2.1 is a variation of the circuit in Fig 4.1.4 that latches the output A into a D-FlipFlop. Based on our study of the D-FlipFlop previously, we know that the output of the flip flop will only change on a rising edge on the clock line.

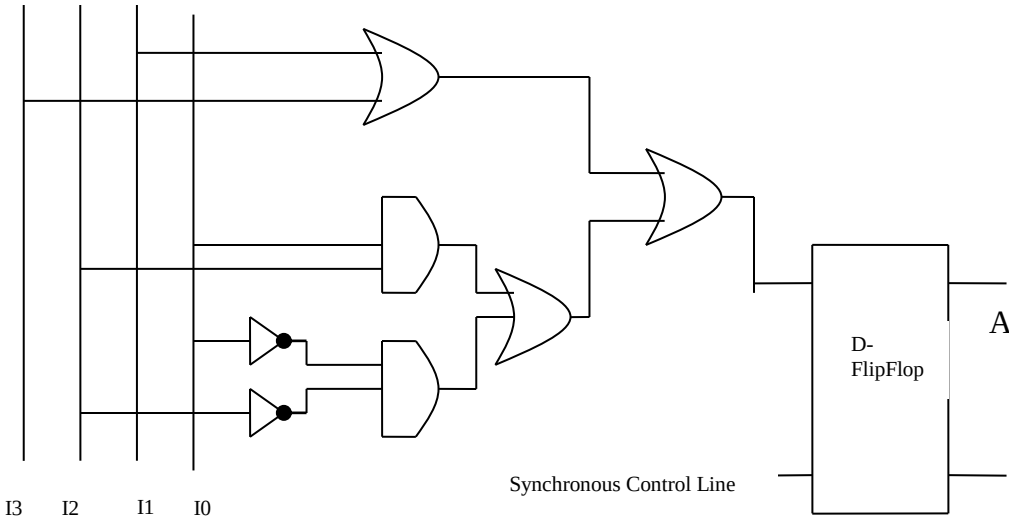


Fig 4.2.1 – Synchronous Logic implementation

In our new design, we will use our clock line as the synchronous control line. After we change the input values to our satisfaction, we will need to toggle the synchronous control line to ensure that our output reflects the changes we made to our input lines.

The change required to make an asynchronous circuit to become a synchronous circuit is fairly minimal, although it increases the gate count in a design. But the payoffs of a synchronous design are substantial. The instability issues associated with asynchronous designs are seldom worth the savings in gate count.

4.3 – Asynchronous Sequential Logic Design

The most common application for a logic circuit involves the implementation of a **state diagram**. A state diagram is simply a set of nodes, each defining a unique state. The lines joining the nodes indicate the possible paths and the required triggers to move from one state into another.

The design of a logic circuit is essentially an exercise in implementing the **states** and **state transition** rules defined by a state diagram with the help of logic gates. To maintain a state we need to implement some form of a memory circuit, much like the D-FlipFlop we discussed earlier. And to transition from one state into another, we need to monitor the triggers and our current state and use logic gates to change the outputs.

The **combinational circuits** discussed thus far can be viewed as a special case logic circuit where a state transition is only dependent on the input triggers but not on the current state. Logic circuits that are dependent on both the input triggers and on the current state are known as **sequential circuits**.

To study sequential circuits, we will implement a very fundamental building block in digital design known as a **counter**. Fig 4.3.1 shows a state diagram for a 3-bit counter. With 3 bits, we can have 2^3 or 8 unique states.

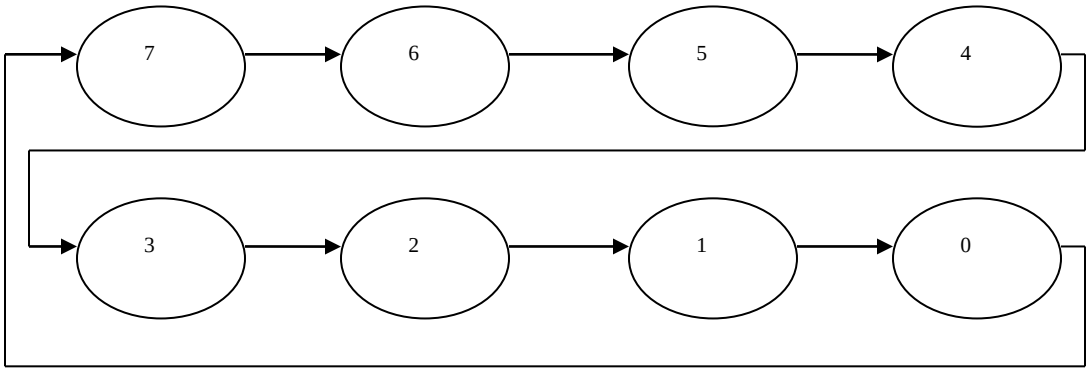


Fig 4.3.1 – State diagram for a 3-bit Counter

The arrows in the state diagram indicate a trigger. For the purpose of this illustration, we will assume that a clock (a rectangular pulse from a crystal oscillator) is our trigger. So when we get a rising edge on the clock and if our current state is “6” then we will move to state “5”. Similarly if we are currently in state “0” and we get a rising edge on the clock, then we roll back to state “7”.

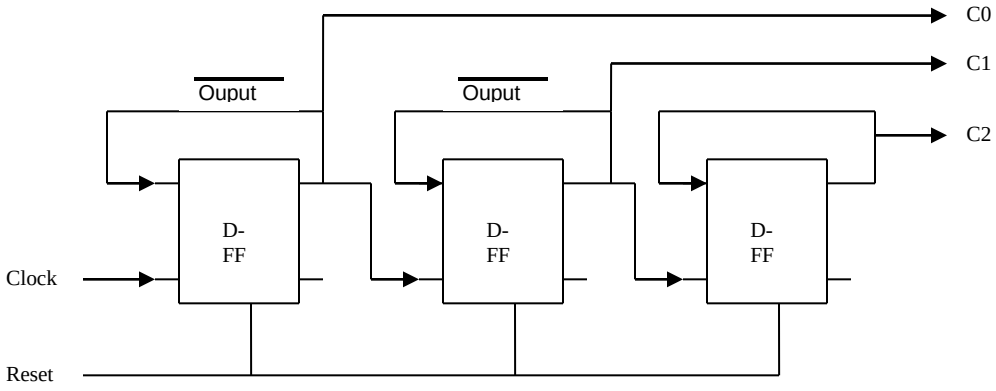


Fig 4.3.2 – 3 Bit Asynchronous Counter

Fig. 4.3.2 shows a very simple logic realization for the state diagram in Fig. 4.3.1 using the D-FlipFlops we studied earlier. To understand its operation, recall that the D-FlipFlop provided an output line as well as a line with the inverse of the output. In the above design we use the inverse output as the input to each of the three Flip-Flops. The least significant Flip-Flop gets its trigger from a clock signal. The other two Flip-Flops get their clocks from the inverse output. In addition, our D-Flip-Flop has a Reset line so that we can change the output of all three Flip-Flops to "0" by toggling the Reset line. Note that when the output is "0" after a Reset, the inverse of the output will be a "1".

Fig 4.3.3 shows a timing diagram with input clock and the values for C0, C1, and C2 subsequent to a Reset toggle. Note how C0, C1, and C2 start as a "1" after a Reset. Then C0 toggles with every rising edge of the Clock. C1 toggles with every rising edge of C0. C2 toggles with every rising edge of C1.

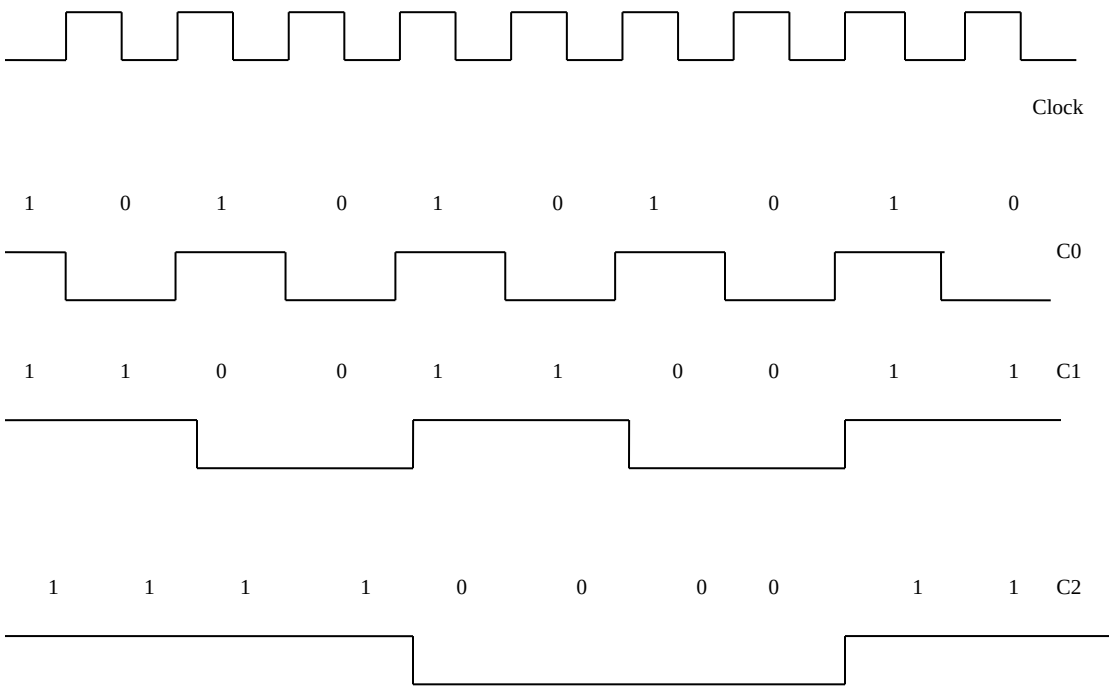


Fig. 4.3.3 – Timing Diagram for 3-bit Counter

Although the circuit in Fig 4.3.2 realizes the state diagram in Fig 4.3.1, it must be noted that the circuit is an asynchronous circuit in that the values of C0, C1 and C2 are attempting to change as quickly as possible. This will mean that there will be numerous glitches in these lines before settling into each state. Unless we latch the outputs at C0, C1 and C2 before passing it to other parts of the system, this design can generate unacceptable instability.

A common application for a counter like the one above is as a frequency divider. Notice that by tapping line C2 we effectively get a clock whose frequency is 8 times less than the original clock frequency.

4.4 – Synchronous Sequential Logic Design

In the previous section we acknowledged that our counter design was asynchronous. One of the main flaws of the design is that the “clock” used by each flip-flop is different. The first flip-flop uses the system clock as its trigger, the second flip-flop used the output of the first flip-flop as its clock and the third flip-flop uses the output of the second flip-flop as its clock. This scheme of using multiple clocks can create unanticipated glitches, due to the cumulative delays in receiving clock signals, and can make a system very unstable. One option we considered to improve upon this behavior was to latch the output of the counter before passing it to the rest of the system. This is not the best option however. Ideally a design engineer would like every related component to share the same clock. This helps ensure that state transition happens at the same time on all related “state aware” components.

Fig 4.4.1 shows an improved synchronous design that achieves this.

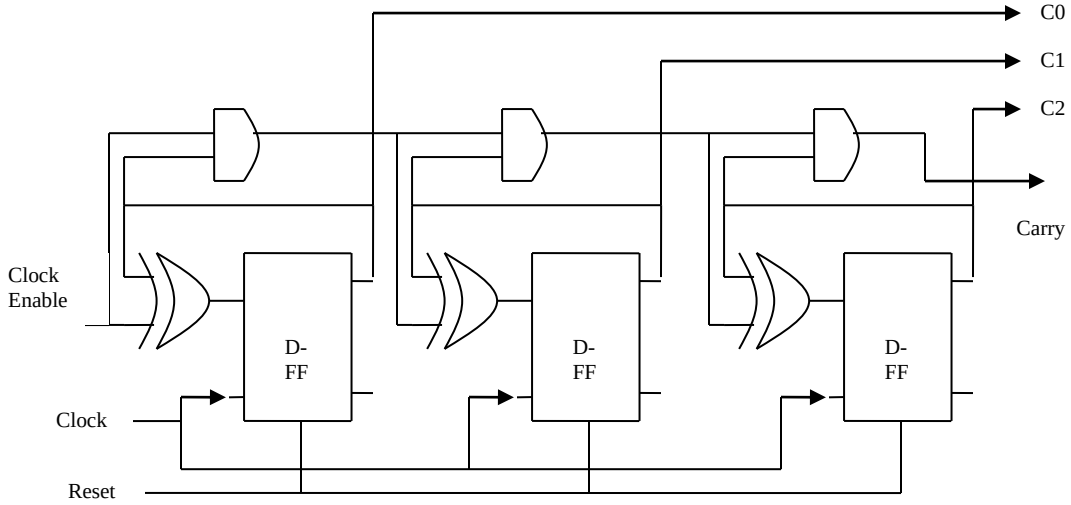


Fig 4.4.1 – 3-Bit Synchronous Counter

As an exercise, draw a timing diagram showing the state transitions for C0, C1 and C2. Assume that after a Reset, C0, C1 and C2 are set to “0”. Also assume that the “Clock Enable” line is set to “1”.

Note that if an “Enable” line is enabled when it is set to “1”, it is referred to as an “**Active High**” enable line. If it were in enabled when it is set to “0”, it is referred to as an “**Active Low**” enable line.

5.0 – Conclusion

It is hoped that the topics covered in this module has equipped the student with an adequate background in the fundamental sciences and digital design concepts. In the next module in this series we will build upon this knowledge by addressing the non-ideal behavior of semi-conductor devices in real-world applications and how the design considerations that were discussed in this module need to be adapted to physical realities and limitations of a semiconductor die.