MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L 01

LECTURE HANDOUTS

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : I- BASIC CONCEPTS OF DIGITAL SYSTEMS AND LOGIC FAMILIES

Date of Lecture:

Topic of Lecture: Review of Number systems, Number Representation

Introduction :

Number systems use different number bases. A number base indicates how many different digits are available when using a particular numbering system.

Prerequisite knowledge for Complete understanding and learning of Topic:

A basic idea regarding the initial concepts of Digital Electronics is enough to understand the topics covered.

Detailed content of the Lecture:

If base or radix of a number system is 'r', then the numbers present in that number system are ranging from zero to r-1. The total numbers present in that number system is 'r'. So, we will get various number systems, by choosing the values of radix as greater than or equal to two.

In this chapter, let us discuss about the popular number systems and how to represent a number in the respective number system. The following number systems are the most commonly used.

- Decimal Number system
- Binary Number system
- Octal Number system
- Hexadecimal Number system

Decimal Number System

The base or radix of Decimal number system is 10. So, the numbers ranging from 0 to 9 are used in this number system. The part of the number that lies to the left of the decimal point is known as integer part. Similarly, the part of the number that lies to the right of the decimal point is known as fractional part.

In this number system, the successive positions to the left of the decimal point having weights of 100, 101, 102, 103 and so on. Similarly, the successive positions to the right of the decimal point having weights of 10-1, 10-2, 10-3 and so on. That means, each position has specific weight, which is power of base 10

Example

Consider the decimal number 1358.246. Integer part of this number is 1358 and fractional part of this number is 0.246. The digits 8, 5, 3 and 1 have weights of 100, 101, 102 and 103 respectively. Similarly, the digits 2, 4 and 6 have weights of 10-1, 10-2 and 10-3 respectively.

Mathematically, we can write it as

$1358.246 = (1 \times 103) + (3 \times 102) + (5 \times 101) + (8 \times 100) + (2 \times 10\text{-}1) +$

$(4 \times 10\text{-}2) + (6 \times 10\text{-}3)$

After simplifying the right hand side terms, we will get the decimal number, which is on left hand side.

Binary Number System

All digital circuits and systems use this binary number system. The base or radix of this number system is 2. So, the numbers 0 and 1 are used in this number system.

The part of the number, which lies to the left of the binary point is known as integer part. Similarly, the part of the number, which lies to the right of the binary point is known as fractional part.

In this number system, the successive positions to the left of the binary point having weights of 20, 21, 22, 23 and so on. Similarly, the successive positions to the right of the binary point having weights of 2-1, 2-2, 2-3 and so on. That means, each position has specific weight, which is power of base 2.

Example

Consider the binary number 1101.011. Integer part of this number is 1101 and fractional part of this number is 0.011. The digits 1, 0, 1 and 1 of integer part have weights of 20, 21, 22, 23 respectively. Similarly, the digits 0, 1 and 1 of fractional part have weights of 2-1, 2-2, 2-3 respectively.

Mathematically, we can write it as

$1101.011 = (1 \times 23) + (1 \times 22) + (0 \times 21) + (1 \times 20) + (0 \times 2\text{-}1) +$

$(1 \times 2\text{-}2) + (1 \times 2\text{-}3)$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of binary number on left hand side.

Octal Number System

The base or radix of octal number system is 8. So, the numbers ranging from 0 to 7 are used in this number system. The part of the number that lies to the left of the octal point is known as integer part. Similarly, the part of the number that lies to the right of the octal point is known as fractional part.

In this number system, the successive positions to the left of the octal point having weights of 80, 81, 82, 83 and so on. Similarly, the successive positions to the right of the octal point having weights of 8-1, 8-2, 8-3 and so on. That means, each position has specific weight, which is power of base 8.

Example

Consider the octal number 1457.236. Integer part of this number is 1457 and fractional part of this number is 0.236. The digits 7, 5, 4 and 1 have weights of 80, 81, 82 and 83 respectively. Similarly, the digits 2, 3 and 6 have weights of 8-1, 8-2, 8-3 respectively.

Mathematically, we can write it as

$1457.236 = (1 \times 83) + (4 \times 82) + (5 \times 81) + (7 \times 80) + (2 \times 8\text{-}1) +$

$(3 \times 8\text{-}2) + (6 \times 8\text{-}3)$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of octal number on left hand side.

Hexadecimal Number System

The base or radix of Hexa-decimal number system is 16. So, the numbers ranging from 0 to 9 and the letters from A to F are used in this number system. The decimal equivalent of Hexa-decimal digits from A to F are 10 to 15.

The part of the number, which lies to the left of the hexadecimal point is known as integer part. Similarly, the part of the number, which lies to the right of the Hexa-decimal point is known as fractional part.

In this number system, the successive positions to the left of the Hexa-decimal point having weights of 160, 161, 162, 163 and so on. Similarly, the successive positions to the right of the Hexa-decimal point having weights of 16-1, 16-2, 16-3 and so on. That means, each position has specific weight, which is power of base 16.

Example

Consider the Hexa-decimal number 1A05.2C4. Integer part of this number is 1A05 and fractional part of this number is 0.2C4. The digits 5, 0, A and 1 have weights of 160, 161, 162 and 163 respectively. Similarly, the digits 2, C and 4 have weights of 16-1, 16-2 and 16-3 respectively.

Mathematically, we can write it as

$1A05.2C4 = (1 \times 163) + (10 \times 162) + (0 \times 161) + (5 \times 160) + (2 \times 16\text{-}1) +$

$(12 \times 16\text{-}2) + (4 \times 16\text{-}3)$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of Hexa-decimal number on left hand side.

Video Content / Details of website for further learning (if any):
https://www.khanacademy.org/math/algebra-home/alg-intro-to-algebra/algebra-alternate-number-bases/v/number-systems-introduction
https://www.youtube.com/watch?v=L2zsmYaI5ww

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (1)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 02

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : I- BASIC CONCEPTS OF DIGITAL SYSTEMS AND LOGIC FAMILIES

Date of Lecture:

Topic of Lecture: Boolean algebra, Boolean postulates and laws

Introduction : Boolean Algebra is an algebra, which deals with binary numbers & binary variables. Hence, it is also called as Binary Algebra or logical Algebra. A mathematician, named George Boole had developed this algebra in 1854. The variables used in this algebra are also called as Boolean variables.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Decimal Number system
- Binary Number system
- Octal Number system
- Hexadecimal Number system

Detailed content of the Lecture:

The range of voltages corresponding to Logic 'High' is represented with '1' and the range of voltages corresponding to logic 'Low' is represented with '0'.

Postulates and Basic Laws of Boolean Algebra

In this section, let us discuss about the Boolean postulates and basic laws that are used in Boolean algebra. These are useful in minimizing Boolean functions.

Boolean Postulates

Consider the binary numbers 0 and 1, Boolean variable xx and its complement x′x′. Either the Boolean variable or complement of it is known as literal. The four possible logical OR operations among these literals and binary numbers are shown below.

$$x + 0 = x \qquad x + 1 = 1 \qquad x + x = x \qquad x + x' = 1$$

Similarly, the four possible logical AND operations among those literals and binary numbers are shown below.

$$x.1 = x \qquad x.0 = 0 \qquad x.x = x \qquad x.x' = 0$$

These are the simple Boolean postulates. We can verify these postulates easily, by substituting the Boolean variable with '0' or '1'.

Note− The complement of complement of any Boolean variable is equal to the variable itself. i.e., x′x′′=x.

Basic Laws of Boolean Algebra

Following are the three basic laws of Boolean Algebra.

- Commutative law
- Associative law
- Distributive law

Commutative Law

If any logical operation of two Boolean variables give the same result irrespective of the order of those two variables, then that logical operation is said to be Commutative. The logical OR & logical AND operations of two Boolean variables x & y are shown below

$$x + y = y + x$$

$$x.y = y.x$$

The symbol '+' indicates logical OR operation. Similarly, the symbol '.' indicates logical AND operation and it is optional to represent. Commutative law obeys for logical OR & logical AND operations.

Associative Law

If a logical operation of any two Boolean variables is performed first and then the same operation is performed with the remaining variable gives the same result, then that logical operation is said to be Associative. The logical OR & logical AND operations of three Boolean variables x, y & z are shown below.

$$x + y+zy+z = x+yx+y + z$$
$$x.y.zy.z = x.yx.y.z$$

Associative law obeys for logical OR & logical AND operations.

Distributive Law

If any logical operation can be distributed to all the terms present in the Boolean function, then that logical operation is said to be Distributive. The distribution of logical OR & logical AND operations of three Boolean variables x, y & z are shown below.

$$x.y+zy+z = x.y + x.z$$
$$x + y.zy.z = x+yx+y.x+zx+z$$

Distributive law obeys for logical OR and logical AND operations.

These are the Basic laws of Boolean algebra. We can verify these laws easily, by substituting the Boolean variables with '0' or '1'.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=gj8QmRQtVao
https://www.youtube.com/watch?v=2U71nZYb990

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (38)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 03

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : I- BASIC CONCEPTS OF DIGITAL SYSTEMS AND LOGIC FAMILIES

Date of Lecture:

Topic of Lecture: De-Morgan's Theorem - Principle of Duality

Introduction :

One part may be obtained from the other if the binary operators and the identity elements are interchanged. This important property of Boolean algebra is called the duality principle.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Decimal Number system
- Binary Number system
- Octal Number system
- Hexadecimal Number system
- Boolean algebra
- Boolean postulates and laws

Detailed content of the Lecture:

Theorems of Boolean Algebra

The following two theorems are used in Boolean algebra.

- Duality theorem
- DeMorgan's theorem

Duality Theorem

This theorem states that the dual of the Boolean function is obtained by interchanging the logical AND operator with logical OR operator and zeros with ones. For every Boolean function, there will be a corresponding Dual function.

Let us make the Boolean equations relationsrelations that we discussed in the section of Boolean postulates and basic laws into two groups. The following table shows these two groups.

Group1	Group2

x + 0 = x	x.1 = x
x + 1 = 1	x.0 = 0
x + x = x	x.x = x
x + x’ = 1	x.x’ = 0
x + y = y + x	x.y = y.x
x + y+zy+z = x+yx+y + z	x.y.zy.z = x.yx.y.z
x.y+zy+z = x.y + x.z	x + y.zy.z = x+yx+y.x+zx+z

In each row, there are two Boolean equations and they are dual to each other. We can verify all these Boolean equations of Group1 and Group2 by using duality theorem.

DeMorgan’s Theorem

This theorem is useful in finding the complement of Boolean function. It states that the complement of logical OR of at least two Boolean variables is equal to the logical AND of each complemented variable.

DeMorgan’s theorem with 2 Boolean variables x and y can be represented as

$$x+yx+y' = x'.y'$$

The dual of the above Boolean function is

$$x.yx.y' = x' + y'$$

Therefore, the complement of logical AND of two Boolean variables is equal to the logical OR of each complemented variable. Similarly, we can apply DeMorgan’s theorem for more than 2 Boolean variables also.

Simplification of Boolean Functions

Till now, we discussed the postulates, basic laws and theorems of Boolean algebra. Now, let us simplify some Boolean functions.

Example 1

Let us simplify the Boolean function, f = p’qr + pq’r + pqr’ + pqr

We can simplify this function in two methods.

Method 1

Given Boolean function, f = p’qr + pq’r + pqr’ +pqr.

Step 1 − In first and second terms r is common and in third and fourth terms pq is common. So, take the common terms by using Distributive law.

$$\Rightarrow f = p'q+pq'p'q+pq'r + pqr'+rr'+r$$

Step 2 − The terms present in first parenthesis can be simplified to Ex-OR operation. The terms present in second parenthesis can be simplified to ‘1’ using Boolean postulate

Method 2

Given Boolean function, f = p’qr + pq’r + pqr’ + pqr.

Step 1 − Use the Boolean postulate, x + x = x. That means, the Logical OR operation with any Boolean variable 'n' times will be equal to the same variable. So, we can write the last term pqr two more times.

$$\Rightarrow f = p'qr + pq'r + pqr' + pqr + pqr + pqr$$

Step 2 − Use Distributive law for 1st and 4th terms, 2nd and 5th terms, 3rd and 6th terms.

$$\Rightarrow f = qrp'+pp'+p + prq'+qq'+q + pqr'+rr'+r$$

Step 3 − Use Boolean postulate, x + x' = 1 for simplifying the terms present in each parenthesis.

$$\Rightarrow f = qr11 + pr11 + pq11$$

Step 4 − Use Boolean postulate, x.1 = x for simplifying the above three terms.

$$\Rightarrow f = qr + pr + pq$$

$$\Rightarrow f = pq + qr + pr$$

Therefore, the simplified Boolean function is f = pq + qr + pr.

So, we got two different Boolean functions after simplifying the given Boolean function in each method. Functionally, those two Boolean functions are same. So, based on the requirement, we can choose one of those two Boolean functions.

Example 2

Let us find the complement of the Boolean function, f = p'q + pq'.

The complement of Boolean function is f' = p′q+pq′p′q+pq''.

Step 1 − Use DeMorgan's theorem, x+yx+y' = x'.y'.

$$\Rightarrow f' = p'qp'q'.pq'pq''$$

Step 2 − Use DeMorgan's theorem, x.yx.y' = x' + y'

$$\Rightarrow f' = \{p'p'' + q'\}.\{p' + q'q''\}$$

Step3 − Use the Boolean postulate, x′x''=x.

$$\Rightarrow f' = \{p + q'\}.\{p' + q\}$$

$$\Rightarrow f' = pp' + pq + p'q' + qq'$$

Step 4 − Use the Boolean postulate, xx'=0.

$$\Rightarrow f = 0 + pq + p'q' + 0$$

$$\Rightarrow f = pq + p'q'$$

Therefore, the complement of Boolean function, p'q + pq' is pq + p'q'.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=yi6oxF8kgXk

https://www.youtube.com/watch?v=20bEwa8W4sQ

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (46)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 04

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : I- BASIC CONCEPTS OF DIGITAL SYSTEMS AND LOGIC FAMILIES

Date of Lecture:

Topic of Lecture: Simplification using Boolean algebra

Introduction :

Boolean algebra is an algebraic structure defined by a set of elements, B, together with two binary operators, + and #, provided that the following (Huntington) postulates are satisfied.

Prerequisite knowledge for Complete understanding and learning of Topic:

1. (a) The structure is closed with respect to the operator +.
(b) The structure is closed with respect to the operator # .
2. (a) The element 0 is an identity element with respect to +; that is, x + 0 =0 + x = x.
(b) The element 1 is an identity element with respect to # ; that is, x # 1 = 1 # x = x.
3. (a) The structure is commutative with respect to +; that is, x + y = y + x.
(b) The structure is commutative with respect to # ; that is, x # y = y # x.
4. (a) The operator # is distributive over +; that is, x # (y + z) = (x # y) + (x # z).
(b) The operator + is distributive over # ; that is, x + (y # z) = (x + y) # (x + z).
5. For every element x H B, there exists an element x_ H B (called the complement of x) such that (a) x + x_ = 1 and (b) x # x_ = 0.
6. There exist at least two elements x, y H B such that x _ y.

Detailed content of the Lecture:

we discussed the postulates, basic laws and theorems of Boolean algebra. Now, let us simplify some Boolean functions.

Example 1

Let us simplify the Boolean function, f = p'qr + pq'r + pqr' + pqr

We can simplify this function in two methods.

Method 1

Given Boolean function, f = p'qr + pq'r + pqr' +pqr.

Step 1 − In first and second terms r is common and in third and fourth terms pq is common. So, take the common terms by using Distributive law.

⇒ f = p′q+pq′p′q+pq′r + pqr′+rr′+r

Step 2 − The terms present in first parenthesis can be simplified to Ex-OR operation. The terms present in second parenthesis can be simplified to '1' using Boolean postulate

⇒ f = p⊕qp⊕qr + pq11

Step 3 − The first term can't be simplified further. But, the second term can be simplified to pq using Boolean postulate.

⇒ f = p⊕qp⊕qr + pq

Therefore, the simplified Boolean function is f = p⊕qp⊕qr + pq

Method 2

Given Boolean function, f = p'qr + pq'r + pqr' + pqr.

Step 1 − Use the Boolean postulate, x + x = x. That means, the Logical OR operation with any Boolean variable 'n' times will be equal to the same variable. So, we can write the last term pqr two more times.

⇒ f = p’qr + pq’r + pqr’ + pqr + pqr + pqr
Step 2 − Use Distributive law for 1st and 4th terms, 2nd and 5th terms, 3rd and 6th terms.
⇒ f = qrp′+pp′+p + prq′+qq′+q + pqr′+rr′+r
Step 3 − Use Boolean postulate, x + x’ = 1 for simplifying the terms present in each parenthesis.
⇒ f = qr11 + pr11 + pq11
Step 4 − Use Boolean postulate, x.1 = x for simplifying the above three terms.
⇒ f = qr + pr + pq
⇒ f = pq + qr + pr
Therefore, the simplified Boolean function is f = pq + qr + pr.
So, we got two different Boolean functions after simplifying the given Boolean function in each method. Functionally, those two Boolean functions are same. So, based on the requirement, we can choose one of those two Boolean functions.

Example 2

Let us find the complement of the Boolean function, f = p’q + pq’.
The complement of Boolean function is f’ = p′q+pq′p′q+pq′’.
Step 1 − Use DeMorgan’s theorem, x+yx+y’ = x’.y’.
⇒ f’ = p′qp′q’.pq′pq′’
Step 2 − Use DeMorgan’s theorem, x.yx.y’ = x’ + y’
⇒ f’ = {p′p′’ + q’}.{p’ + q′q′’}
Step3 − Use the Boolean postulate, x′x′’=x.
⇒ f’ = {p + q’}.{p’ + q}
⇒ f’ = pp’ + pq + p’q’ + qq’
Step 4 − Use the Boolean postulate, xx’=0.
⇒ f = 0 + pq + p’q’ + 0
⇒ f = pq + p’q’
Therefore, the complement of Boolean function, p’q + pq’ is pq + p’q’.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=0as464WmfCo
https://www.youtube.com/watch?v=59BbncMjL8I

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti “Digital Design” , Pearson Education- V Edition,2013. Page No (43)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 05

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : I- BASIC CONCEPTS OF DIGITAL SYSTEMS AND LOGIC FAMILIES

Date of Lecture:

Topic of Lecture: Canonical forms Sum of product and Product of sum

Introduction :

A truth table consists of a set of inputs and outputs. If there are 'n' input variables, then there will be 2n possible combinations with zeros and ones. So the value of each output variable depends on the combination of input variables. So, each output variable will have '1' for some combination of input variables and '0' for some other combination of input variables.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws

Detailed content of the Lecture:

- Canonical SoP form
- Canonical PoS form

Canonical SoP form

Canonical SoP form means Canonical Sum of Products form. In this form, each product term contains all literals. So, these product terms are nothing but the min terms. Hence, canonical SoP form is also called as sum of min terms form.

First, identify the min terms for which, the output variable is one and then do the logical OR of those min terms in order to get the Boolean expression functionfunction corresponding to that output variable. This Boolean function will be in the form of sum of min terms.

Follow the same procedure for other output variables also, if there is more than one output variable.

Example:Here, the output ff is '1' for four combinations of inputs. The corresponding min terms are p'qr, pq'r, pqr', pqr. By doing logical OR of these four min terms, we will get the Boolean function of output ff.

Therefore, the Boolean function of output is, f = p'qr + pq'r + pqr' + pqr. This is the canonical SoP form of output, f. We can also represent this function in following two notations.

$$f=m3+m5+m6+m7f=m3+m5+m6+m7$$

$$f=\sum m(3,5,6,7)f=\sum m(3,5,6,7)$$

Consider the following truth table.

Inputs		Output	
p	q	r	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

In one equation, we represented the function as sum of respective min terms. In other equation, we used the symbol for summation of those min terms.

Canonical PoS form

Canonical PoS form means Canonical Product of Sums form. In this form, each sum term contains all literals. So, these sum terms are nothing but the Max terms. Hence, canonical PoS form is also called as product of Max terms form.

First, identify the Max terms for which, the output variable is zero and then do the logical AND of those Max terms in order to get the Boolean expression function corresponding to that output variable. This Boolean function will be in the form of product of Max terms.

Follow the same procedure for other output variables also, if there is more than one output variable.

Example: Consider the same truth table of previous example. Here, the output ff is '0' for four combinations of inputs. The corresponding Max terms are p + q + r, p + q + r', p + q' + r, p' + q + r. By doing logical AND of these four Max terms, we will get the Boolean function of output ff.

Therefore, the Boolean function of output is, f = p+q+rp+q+r.p+q+r′p+q+r′.p+q′+rp+q′+r.p′+q+rp′+q+r. This is the canonical PoS form of output, f. We can also represent this function in following two notations.

f=M0.M1.M2.M4f=M0.M1.M2.M4

f=∏M(0,1,2,4)f=∏M(0,1,2,4)

In one equation, we represented the function as product of respective Max terms. In other equation, we used the symbol for multiplication of those Max terms.

The Boolean function, f = p+q+rp+q+r.p+q+r′p+q+r′.p+q′+rp+q′+r.p′+q+rp′+q+r is the dual of the Boolean function, f = p'qr + pq'r + pqr' + pqr.

Therefore, both canonical SoP and canonical PoS forms are Dual to each other. Functionally, these two forms are same. Based on the requirement, we can use one of these two forms.

Standard SoP and PoS forms

We discussed two canonical forms of representing the Boolean outputss. Similarly, there are two standard forms of representing the Boolean outputss. These are the simplified version of canonical forms.

- Standard SoP form
- Standard PoS form

We will discuss about Logic gates in later chapters. The main advantage of standard forms is that the number of inputs applied to logic gates can be minimized. Sometimes, there will be reduction in the total number of logic gates required.

Standard SoP form

Standard SoP form means Standard Sum of Products form. In this form, each product term need not contain all literals. So, the product terms may or may not be the min terms. Therefore, the Standard SoP form is the simplified form of canonical SoP form.

We will get Standard SoP form of output variable in two steps.

- Get the canonical SoP form of output variable
- Simplify the above Boolean function, which is in canonical SoP form.

Follow the same procedure for other output variables also, if there is more than one output variable. Sometimes, it may not possible to simplify the canonical SoP form. In that case, both canonical and standard SoP forms are same.

Example:Convert the following Boolean function into Standard SoP form.

$$f = p'qr + pq'r + pqr' + pqr$$

The given Boolean function is in canonical SoP form. Now, we have to simplify this Boolean function in order to get standard SoP form.

Step 1 − Use the Boolean postulate, x + x = x. That means, the Logical OR operation with any Boolean variable 'n' times will be equal to the same variable. So, we can write the last term pqr two more times.

$$\Rightarrow f = p'qr + pq'r + pqr' + pqr + pqr + pqr$$

Step 2 − Use Distributive law for 1st and 4th terms, 2nd and 5th terms, 3rd and 6th terms.

$$\Rightarrow f = qrp'+pp'+p + prq'+qq'+q + pqr'+rr'+r$$

Step 3 − Use Boolean postulate, x + x' = 1 for simplifying the terms present in each parenthesis.

$$\Rightarrow f = qr11 + pr11 + pq11$$

Step 4 − Use Boolean postulate, x.1 = x for simplifying above three terms.

$$\Rightarrow f = qr + pr + pq$$

$$\Rightarrow f = pq + qr + pr$$

This is the simplified Boolean function. Therefore, the standard SoP form corresponding to given canonical SoP form is f = pq + qr + pr

Standard PoS form

Standard PoS form means Standard Product of Sums form. In this form, each sum term need not contain all literals. So, the sum terms may or may not be the Max terms. Therefore, the Standard PoS form is the simplified form of canonical PoS form.

We will get Standard PoS form of output variable in two steps.

- Get the canonical PoS form of output variable
- Simplify the above Boolean function, which is in canonical PoS form.

Follow the same procedure for other output variables also, if there is more than one output variable. Sometimes, it may not possible to simplify the canonical PoS form. In that case, both canonical and standard PoS forms are same.

Example: Convert the following Boolean function into Standard PoS form.

$$f = p+q+rp+q+r.p+q+r'p+q+r'.p+q'+rp+q'+r.p'+q+rp'+q+r$$

The given Boolean function is in canonical PoS form. Now, we have to simplify this Boolean function in order to get standard PoS form.

Step 1 − Use the Boolean postulate, x.x = x. That means, the Logical AND operation with any Boolean variable 'n' times will be equal to the same variable. So, we can write the first term p+q+r two more times.

$$\Rightarrow f = p+q+rp+q+r.p+q+rp+q+r.p+q+rp+q+r.p+q+r'p+q+r'.p+q'+rp+q'+r.p'+q+rp'+q+r$$

Step 2 − Use Distributive law, x + y.zy.z = x+yx+y.x+zx+z for 1st and 4th parenthesis, 2nd and 5th parenthesis, 3rd and 6th parenthesis.

$$\Rightarrow f = p+q+rr'p+q+rr'.p+r+qq'p+r+qq'.q+r+pp'q+r+pp'$$

Step 3 − Use Boolean postulate, x.x'=0 for simplifying the terms present in each parenthesis.

$$\Rightarrow f = p+q+0p+q+0.p+r+0p+r+0.q+r+0q+r+0$$

Step 4 − Use Boolean postulate, x + 0 = x for simplifying the terms present in each parenthesis

$$\Rightarrow f = p+qp+q.p+rp+r.q+rq+r$$
$$\Rightarrow f = p+qp+q.q+rq+r.p+rp+r$$

This is the simplified Boolean function. Therefore, the standard PoS form corresponding to given canonical PoS form is f = p+qp+q.q+rq+r.p+rp+r. This is the dual of the Boolean function, f = pq + qr + pr.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=iiIBvuv1QU4
https://www.youtube.com/watch?v=OaLUco7KL1Q
https://www.youtube.com/watch?v=Dv0KZ6E3tc4

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (51)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 06

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : I- BASIC CONCEPTS OF DIGITAL SYSTEMS AND LOGIC FAMILIES

Date of Lecture:

Topic of Lecture: Minimization using Karnaugh map

Introduction :

This procedure of minimization is awkward because it lacks specific rules to predict each succeeding step in the manipulative process. The map method presented here provides a simple, straightforward procedure for minimizing Boolean functions. This method may be regarded as a pictorial al form of a truth table. The map method is also known as the Karnaugh map or K-map .

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum

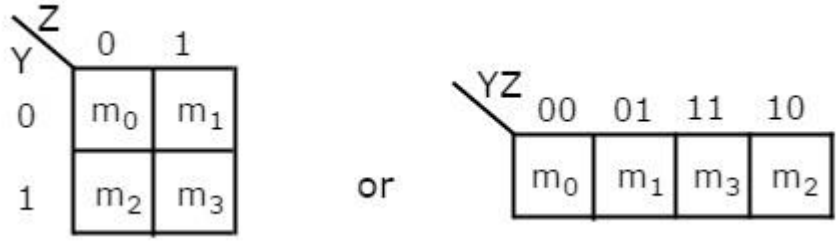
Detailed content of the Lecture:

K-Maps for 2 to 5 Variables

K-Map method is most suitable for minimizing Boolean functions of 2 variables to 5 variables. Now, let us discuss about the K-Maps for 2 to 5 variables one by one.

2 Variable K-Map

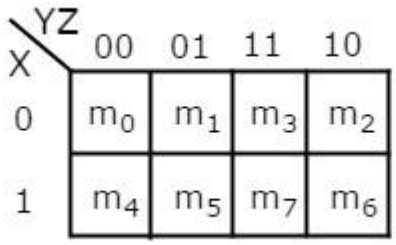
The number of cells in 2 variable K-map is four, since the number of variables is two. The following figure shows 2 variable K-Map.



- There is only one possibility of grouping 4 adjacent min terms.
- The possible combinations of grouping 2 adjacent min terms are {(m0, m1), (m2, m3), (m0, m2) and (m1, m3)}.

3 Variable K-Map

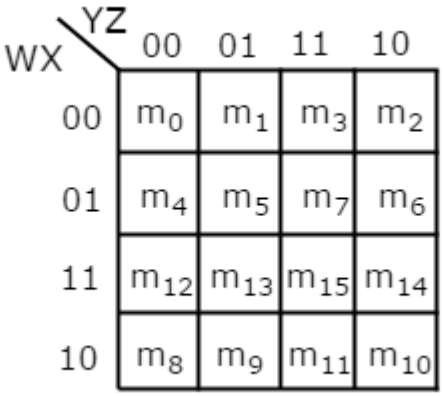
The number of cells in 3 variable K-map is eight, since the number of variables is three. The following figure shows 3 variable K-Map.



- There is only one possibility of grouping 8 adjacent min terms.
- The possible combinations of grouping 4 adjacent min terms are {(m0, m1, m3, m2), (m4, m5, m7, m6), (m0, m1, m4, m5), (m1, m3, m5, m7), (m3, m2, m7, m6) and (m2, m0, m6, m4)}.
- The possible combinations of grouping 2 adjacent min terms are {(m0, m1), (m1, m3), (m3, m2), (m2, m0), (m4, m5), (m5, m7), (m7, m6), (m6, m4), (m0, m4), (m1, m5), (m3, m7) and (m2, m6)}.
- If x=0, then 3 variable K-map becomes 2 variable K-map.

4 Variable K-Map

The number of cells in 4 variable K-map is sixteen, since the number of variables is four. The following figure shows 4 variable K-Map.



- There is only one possibility of grouping 16 adjacent min terms.
- Let R1, R2, R3 and R4 represents the min terms of first row, second row, third row and fourth row respectively. Similarly, C1, C2, C3 and C4 represents the min terms of first column, second column, third column and fourth column respectively. The possible combinations of grouping 8 adjacent min terms are {(R1, R2), (R2, R3), (R3, R4), (R4, R1), (C1, C2), (C2, C3), (C3, C4), (C4, C1)}.
- If w=0, then 4 variable K-map becomes 3 variable K-map.

5 Variable K-Map

The number of cells in 5 variable K-map is thirty-two, since the number of variables is 5. The following figure shows 5 variable K-Map.

V=0

WX \ YZ	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

V=1

WX \ YZ	00	01	11	10
00	m_{16}	m_{17}	m_{19}	m_{18}
01	m_{20}	m_{21}	m_{23}	m_{22}
11	m_{28}	m_{29}	m_{31}	m_{30}
10	m_{24}	m_{25}	m_{27}	m_{26}

- There is only one possibility of grouping 32 adjacent min terms.
- There are two possibilities of grouping 16 adjacent min terms. i.e., grouping of min terms from m0 to m15 and m16 to m31.
- If v=0, then 5 variable K-map becomes 4 variable K-map.

In the above all K-maps, we used exclusively the min terms notation. Similarly, you can use exclusively the Max terms notation.

Minimization of Boolean Functions using K-Maps

If we consider the combination of inputs for which the Boolean function is '1', then we will get the Boolean function, which is in standard sum of products form after simplifying the K-map.Similarly, if we consider the combination of inputs for which the Boolean function is '0', then we will get the Boolean function, which is in standard product of sums form after simplifying the K-map.

Follow these rules for simplifying K-maps in order to get standard sum of products form.

- Select the respective K-map based on the number of variables present in the Boolean function.
- If the Boolean function is given as sum of min terms form, then place the ones at respective min term cells in the K-map. If the Boolean function is given as sum of products form, then place the ones in all possible cells of K-map for which the given product terms are valid.
- Check for the possibilities of grouping maximum number of adjacent ones. It should be powers of two. Start from highest power of two and upto least power of two. Highest power is equal to the number of variables considered in K-map and least power is zero.
- Each grouping will give either a literal or one product term. It is known as prime implicant. The prime implicant is said to be essential prime implicant, if atleast single '1' is not covered with any other groupings but only that grouping covers.
- Note down all the prime implicants and essential prime implicants. The simplified Boolean function contains all essential prime implicants and only the required prime implicants.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=wjM2RDG5yTI
https://www.youtube.com/watch?v=CpsJoAwreqo

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (73)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 07

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : I- BASIC CONCEPTS OF DIGITAL SYSTEMS AND LOGIC FAMILIES

Date of Lecture:

Topic of Lecture: Minimization using Tabulation method

Introduction :

Quine-McClukey tabular method is a tabular method based on the concept of prime implicants. We know that prime implicant is a product orsumorsum term, which can't be further reduced by combining with any other product orsumorsum terms of the given Boolean function.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Example

Let us simplify the following Boolean function, f(W,X,Y,Z)=∑m(2,6,8,9,10,11,14,15)f(W,X,Y,Z)=∑m(2,6,8,9,10,11,14,15) using Quine-McClukey tabular method.

The given Boolean function is in sum of min terms form. It is having 4 variables W, X, Y & Z. The given min terms are 2, 6, 8, 9, 10, 11, 14 and 15. The ascending order of these min terms based on the number of ones present in their binary equivalent is 2, 8, 6, 9, 10, 11, 14 and 15. The following table shows these min terms and their equivalent binary representations.

Group Name	Min terms	W	X	Y	Z
GA1	2	0	0	1	0
	8	1	0	0	0
GA2	6	0	1	1	0

	9	1	0	0	1
	10	1	0	1	0
GA3	11	1	0	1	1
	14	1	1	1	0
GA4	15	1	1	1	1

The given min terms are arranged into 4 groups based on the number of ones present in their binary equivalents. The following table shows the possible merging of min terms from adjacent groups.

Group Name	Min terms	W	X	Y	Z
GB1	2,6	0	-	1	0
	2,10	-	0	1	0
	8,9	1	0	0	-
	8,10	1	0	-	0
GB2	6,14	-	1	1	0
	9,11	1	0	-	1
	10,11	1	0	1	-
	10,14	1	-	1	0
GB3	11,15	1	-	1	1
	14,15	1	1	1	-

The min terms, which are differed in only one-bit position from adjacent groups are merged. That differed bit is represented with this symbol, '-'. In this case, there are three groups and each group contains combinations of two min terms. The following table shows the possible merging of min term pairs from adjacent groups.

Group Name	Min terms	W	X	Y	Z

GB1	2,6,10,14	-	-	1	0
	2,10,6,14	-	-	1	0
	8,9,10,11	1	0	-	-
	8,10,9,11	1	0	-	-
GB2	10,11,14,15	1	-	1	-
	10,14,11,15	1	-	1	-

The successive groups of min term pairs, which are differed in only one-bit position are merged. That differed bit is represented with this symbol, '-'. In this case, there are two groups and each group contains combinations of four min terms. Here, these combinations of 4 min terms are available in two rows. So, we can remove the repeated rows. The reduced table after removing the redundant rows is shown below.

Group Name	Min terms	W	X	Y	Z
GC1	2,6,10,14	-	-	1	0
	8,9,10,11	1	0	-	-
GC2	10,11,14,15	1	-	1	-

Further merging of the combinations of min terms from adjacent groups is not possible, since they are differed in more than one-bit position. There are three rows in the above table. So, each row will give one prime implicant. Therefore, the prime implicants are YZ', WX' & WY.

The prime implicant table is shown below.

Min terms / Prime Implicants	2	6	8	9	10	11	14	15
YZ'	1	1			1		1	
WX'			1	1	1	1		
WY					1	1	1	1

The prime implicants are placed in row wise and min terms are placed in column wise. 1s are placed in the common cells of prime implicant rows and the corresponding min term columns.

The min terms 2 and 6 are covered only by one prime implicant YZ'. So, it is an essential prime implicant. This will be part of simplified Boolean function. Now, remove this prime implicant row and the corresponding min

term columns. The reduced prime implicant table is shown below.

Min terms / Prime Implicants	8	9	11	15
WX’	1	1	1	
WY			1	1

The min terms 8 and 9 are covered only by one prime implicant WX’. So, it is an essential prime implicant. This will be part of simplified Boolean function. Now, remove this prime implicant row and the corresponding min term columns. The reduced prime implicant table is shown below.

Min terms / Prime Implicants	15
WY	1

The min term 15 is covered only by one prime implicant WY. So, it is an essential prime implicant. This will be part of simplified Boolean function.

In this example problem, we got three prime implicants and all the three are essential. Therefore, the simplified Boolean function is

$$fW,X,Y,ZW,X,Y,Z = YZ' + WX' + WY.$$

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=l1jgq0R5EwQ
https://www.youtube.com/watch?v=P0ULm0cbIhE

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti “Digital Design” , Pearson Education- V Edition,2013. Page No (103)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 08

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : I- BASIC CONCEPTS OF DIGITAL SYSTEMS AND LOGIC FAMILIES

Date of Lecture:

Topic of Lecture: Digital Logic Families - TTL

Introduction :

The electronic components used in the construction of the basic circuit are usually used as the name of the technology.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Digital Logic Families

The electronic components used in the construction of the basic circuit are usually used as the name of the technology. The following are the most popular:

–RTL resistor-transistor logic

–DTL diode-transistor logic

–TTL transistor-transistor logic (widespread, standard)

–ECL emitter-coupled logic (high speed)

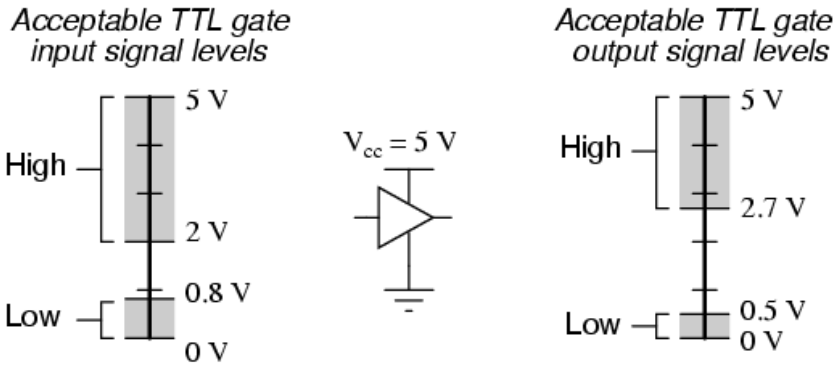
–MOS, PMOS, NMOS metal-oxide semiconductor (high component density)

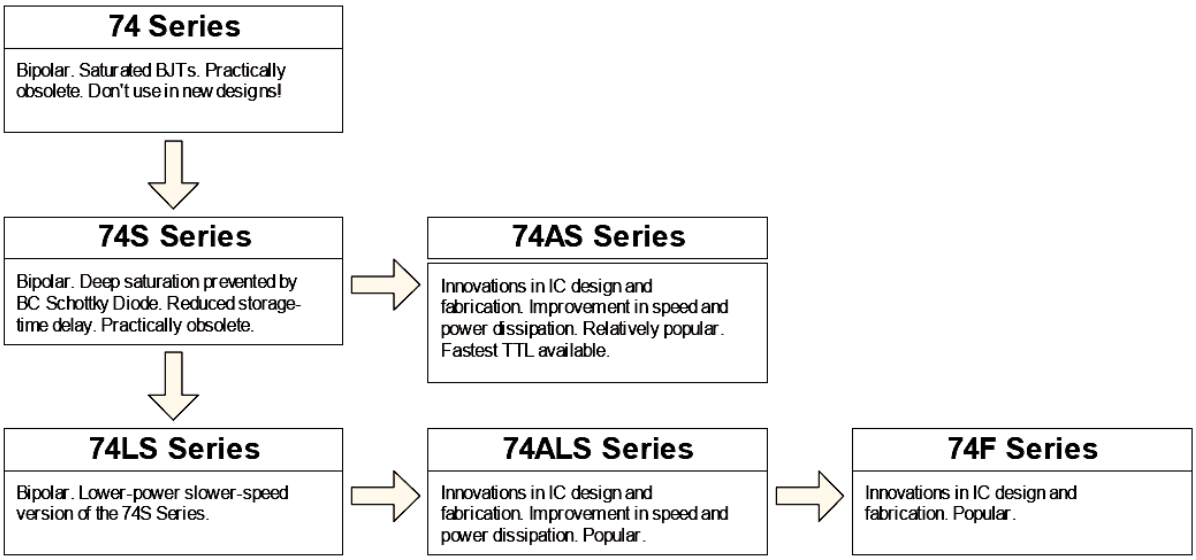
–CMOS complementary metal-oxide semiconductor (low power consumption)

Transistor-transistor logic (TTL)

- based on bipolar transistors
- one of the most widely used families for small- and medium-scale devices – rarely used for VLSI
- typically operated from 5V supply
- typical noise immunity about 1 – 1.6 V
- many forms, some optimised for speed, power, etc.

- high speed versions comparable to CMOS (~ 1.5 ns)
- low-power versions down to about 1 mW/gate





Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=sW1FEm2yNnA

https://www.youtube.com/watch?v=aU0nLAicRKQ

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (97)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 09

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : I- BASIC CONCEPTS OF DIGITAL SYSTEMS AND LOGIC FAMILIES

Date of Lecture:

Topic of Lecture: Digital Logic Families – ECL, CMOS

Introduction :

The electronic components used in the construction of the basic circuit are usually used as the name of the technology.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum

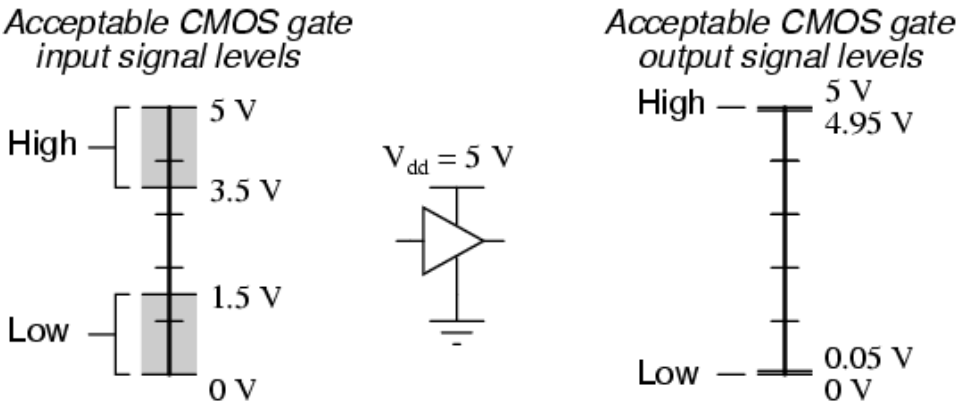
Detailed content of the Lecture:

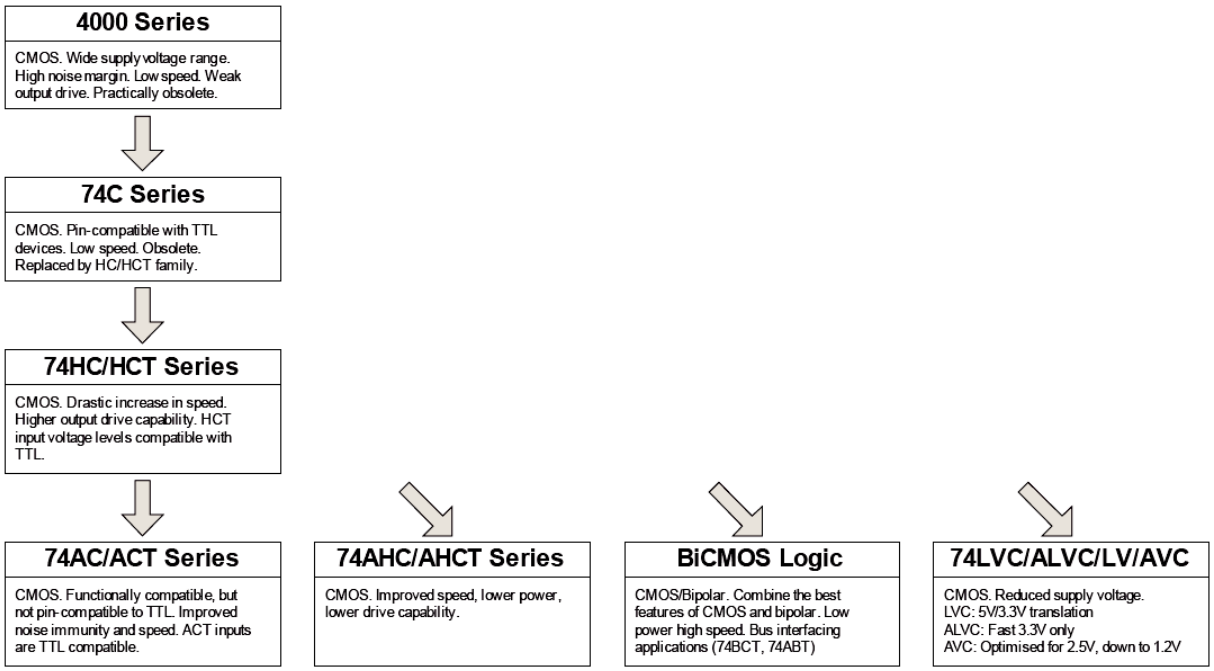
Parameter	CMOS	TTL	ECL
Basic gate	NAND/NOR	NAND	OR/NOR
Fan-out	>50	10	25
Power per gate (mW)	1 @ 1 MHz	1 - 22	4 - 55
Noise immunity	Excellent	Very good	Good
t_{PD} (ns)	1 - 200	1.5 – 33	1 - 4

CMOS Logic Families

- Complementary metal oxide semiconductor (CMOS)
- most widely used family for large-scale devices

- combines high speed with low power consumption
- usually operates from a single supply of 5 – 15 V
- excellent noise immunity of about 30% of supply voltage
- can be connected to a large number of gates (about 50)
- many forms – some with tPD down to 1 ns
- power consumption depends on speed (perhaps 1 mW)





Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=bj4oiFUoYAs

https://www.mepits.com/tutorial/29/basic-electronics/logic-families-ttl-cmos-ecl

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (507)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS | **L 10**

ECE | **II / III**

Course Name with Code :19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan,AP/ECE

Unit : II- COMBINATIONAL CIRCUITS **Date of Lecture:**

Topic of Lecture:
Realization of combinational logic using gates

Introduction : Digital circuits are frequently constructed with NAND or NOR gates rather than with AND and OR gates. NAND and NOR gates are easier to fabricate with electroniccomponents and are the basic gates used in all IC digital logic families. Because of theprominence of NAND and NOR gates in the design of digital circuits, rules and procedureshave been developed for the conversion from Boolean functions given in termsof AND, OR, and NOT into equivalent NAND and NOR logic diagrams.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Implement the following Boolean function with NAND gates:

$$F(x, y, z) = (1, 2, 3, 4, 5, 7)$$

The first step is to simplify the function into sum-of-products form. This is done bymeans of the map of Fig. 3.19 (a), from which the simplified function is obtained:

$$F = xy\_ + x\_y + z$$

The two-level NAND implementation is shown in Fig. 3.19 (b) in mixed notation.
Notethat input z must have a one-input NAND gate (an inverter) to compensate for thebubble in the second-level gate. An alternative way of drawing the logic diagram is givenin Fig. 3.19 (c). Here, all the NAND gates are drawn with the same graphic symbol. Theinverter with input z has been removed, but the input variable is complemented anddenoted by z_.

The procedure described in the previous example indicates that a Boolean function can be implemented with two levels of NAND gates. The procedure for obtaining the logic diagram from a Boolean function is as follows:

1. Simplify the function and express it in sum-of-products form.
2. Draw a NAND gate for each product term of the expression that has at least twoliterals. The inputs to each NAND gate are the literals of the term. This procedureproduces a group of first-level gates.
3. Draw a single gate using the AND-invert or the invert-OR graphic symbol in thesecond level, with inputs coming from outputs of first-level gates.
4. A term with a single literal requires an inverter in the first level. However, if thesingle literal is complemented, it can be connected directly to an input of the secondlevelNAND gate.

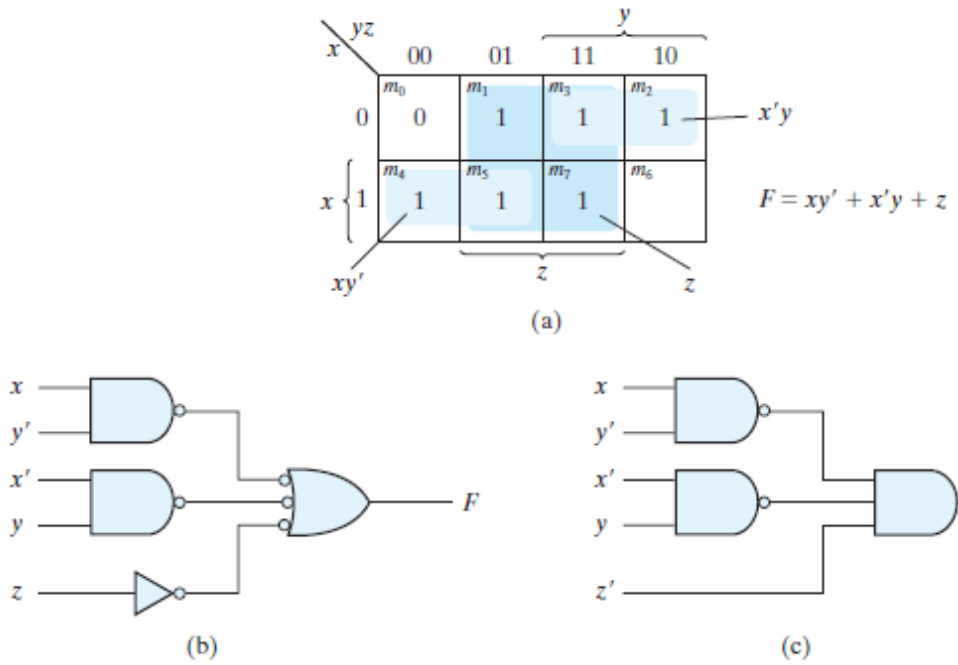


FIGURE 3.19
Solution to Example 3.9

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=wAGoIaF5qWM&vl=hi
https://www.youtube.com/watch?v=ykfGh4Ye_Lc

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (92)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 11

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : II- COMBINATIONAL CIRCUITS

Date of Lecture:

Topic of Lecture:

Design of combinational circuits, Adder , Subtractor

Introduction :

The basic arithmetic circuits like Binary adder and Binary subtractor. These circuits can be operated with binary values 0 and 1.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Binary Adder

The most basic arithmetic operation is addition. The circuit, which performs the addition of two binary numbers is known as Binary adder. First, let us implement an adder, which performs the addition of two bits.

Half Adder

Half adder is a combinational circuit, which performs the addition of two binary numbers A and B are of single bit. It produces two outputs sum, S & carry, C.

The Truth table of Half adder is shown below.

Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1

1	1	1	0

When we do the addition of two bits, the resultant sum can have the values ranging from 0 to 2 in decimal. We can represent the decimal digits 0 and 1 with single bit in binary. But, we can't represent decimal digit 2 with single bit in binary. So, we require two bits for representing it in binary.

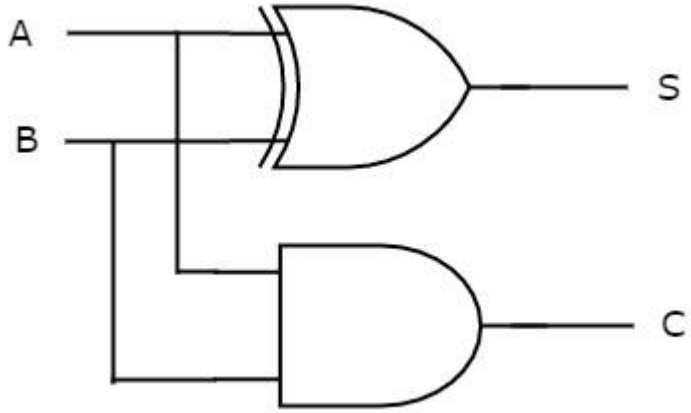
Let, sum, S is the Least significant bit and carry, C is the Most significant bit of the resultant sum. For first three combinations of inputs, carry, C is zero and the value of S will be either zero or one based on the number of ones present at the inputs. But, for last combination of inputs, carry, C is one and sum, S is zero, since the resultant sum is two.

From Truth table, we can directly write the Boolean functions for each output as

$$S=A \oplus BS=A \oplus B$$

$C=ABC=AB$

We can implement the above functions with 2-input Ex-OR gate & 2-input AND gate. The circuit diagram of Half adder is shown in the following figure.



In the above circuit, a two input Ex-OR gate & two input AND gate produces sum, S & carry, C respectively. Therefore, Half-adder performs the addition of two bits.

Binary Subtractor

The circuit, which performs the subtraction of two binary numbers is known as Binary subtractor. We can implement Binary subtractor in following two methods.

- Cascade Full subtractors
- 2's complement method

In first method, we will get an n-bit binary subtractor by cascading 'n' Full subtractors. So, first you can implement Half subtractor and Full subtractor, similar to Half adder & Full adder. Then, you can implement an n-bit binary subtractor, by cascading 'n' Full subtractors. So, we will be having two separate circuits for binary addition and subtraction of two binary numbers.

In second method, we can use same binary adder for subtracting two binary numbers just by doing some modifications in the second input. So, internally binary addition operation takes place but, the output is resultant subtraction.

We know that the subtraction of two binary numbers A & B can be written as,

$$A-B=A+(2'scomplimentofB)A-B=A+(2'scomplimentofB)$$

$\Rightarrow A-B=A+(1'scomplimentofB)+1$

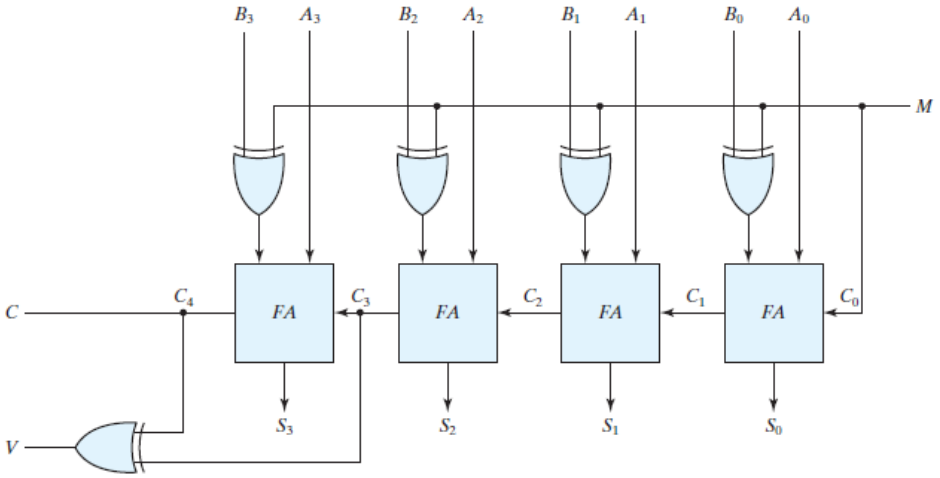


FIGURE 4.13
Four-bit adder–subtractor (with overflow detection)

The circuit for subtracting A - B consists of an adder with inverters placed between each data input B and the corresponding input of the full adder. The input carry C0 must be equal to 1 when subtraction is performed. The operation thus performed becomes A , plus the 1's complement of B , plus 1. This is equal to A plus the 2's complement of B . For unsigned numbers, that gives A - B if A Ú B or the 2's complement of 1B - A2 if A 6 B. For signed numbers, the result is A - B, provided that there is no overflow.

The addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive-OR gate with each full adder. A four-bit adder–subtractor circuit is shown in Fig. 4.13 . The mode input M controls the operation. When M = 0, the circuit is an adder, and when M = 1, the circuit becomes a subtractor. Each exclusive-OR gate receives input M and one of the inputs of B . When M = 0, we have B { 0 = B. The full adders receive the value of B , the input carry is 0, and the circuit performs A plus B . When M = 1, we have B { 1 = B_ and C0 = 1. The B inputs are all complemented and a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B . (The exclusive-OR with output V is for detecting an overflow.)

It is worth noting that binary numbers in the signed-complement system are added and subtracted by the same basic addition and subtraction rules as are unsigned numbers. Therefore, computers need only one common hardware circuit to handle both types of arithmetic. The user or programmer must interpret the results of such addition or subtraction differently, depending on whether it is assumed that the numbers are signed or unsigned.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=clCQ8J-yk7I

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (133)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 12

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : II- COMBINATIONAL CIRCUITS

Date of Lecture:

Topic of Lecture:

Parallel adder Subtractor, Carry look ahead adder

Introduction :

Addition of n-bit numbers requires a chain of n full adders or a chain of one-half adder and n 9 1 full adders. In the former case, the input carry to the least significant position is fixed at 0. Figure 4.9 shows the interconnection of four full-adder (FA) circuits to provide a four-bit binary ripple carry adder

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

4-bit Binary Adder

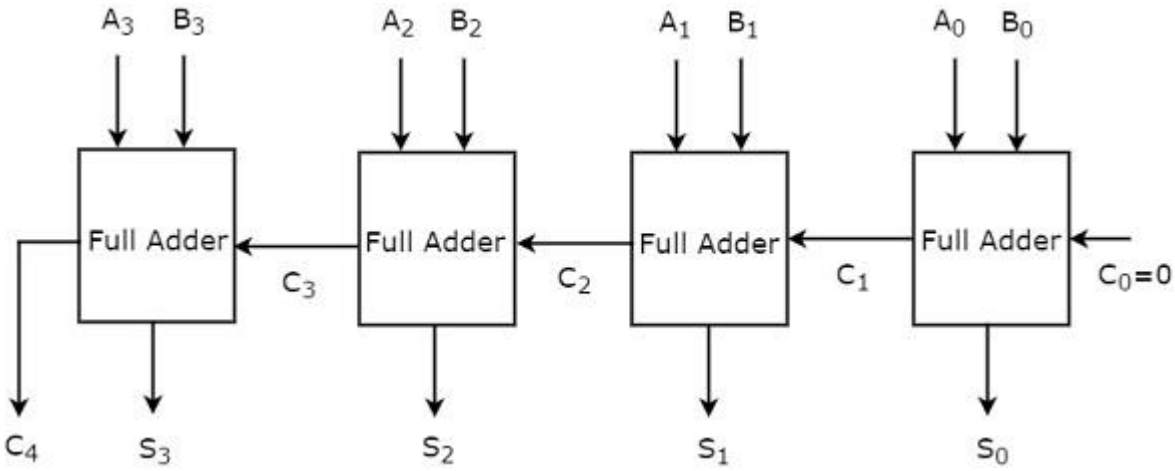
The 4-bit binary adder performs the addition of two 4-bit numbers. Let the 4-bit binary numbers, A=A3A2A1A0A=A3A2A1A0 and B=B3B2B1B0B=B3B2B1B0. We can implement 4-bit binary adder in one of the two following ways.

- Use one Half adder for doing the addition of two Least significant bits and three Full adders for doing the addition of three higher significant bits.
- Use four Full adders for uniformity. Since, initial carry Cin is zero, the Full adder which is used for adding the least significant bits becomes Half adder.

For the time being, we considered second approach. The block diagram of 4-bit binary adder is shown in the following figure.

Here, the 4 Full adders are cascaded. Each Full adder is getting the respective bits of two parallel inputs A & B. The carry output of one Full adder will be the carry input of subsequent higher order Full adder. This 4-bit binary adder produces the resultant sum having at most 5 bits. So, carry out of last stage Full adder will be the MSB.

In this way, we can implement any higher order binary adder just by cascading the required number of Full adders. This binary adder is also called as ripple carry binarybinary adder because the carry propagates ripplesripples from one stage to the next stage.

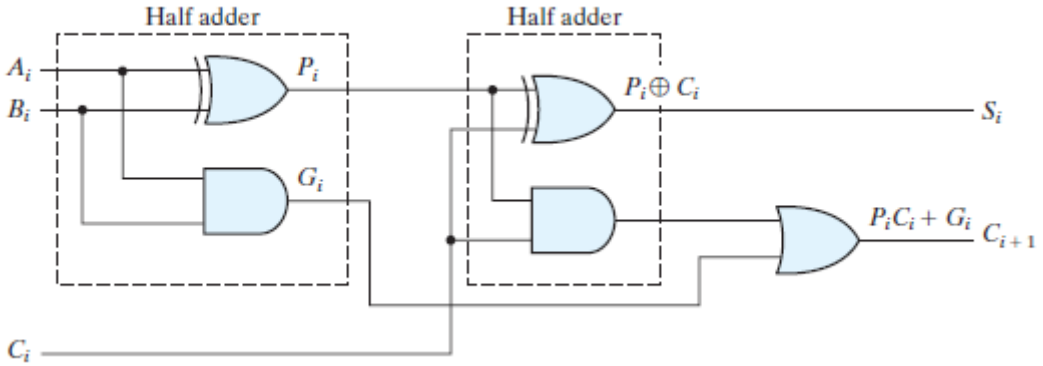


Carry Propagation

The addition of two binary numbers in parallel implies that all the bits of the augendand addend are available for computation at the same time. As in any combinational circuit, the signal must propagate through the gates before the correct output sum is available in the output terminals. The total propagation time is equal to the propagation delay of a typical gate, times the number of gate levels in the circuit. The longest propagation delay time in an adder is the time it takes the carry to propagate through the full adders. Since each bit of the sum output depends on the value of the input carry, the
value of Si at any given stage in the adder will be in its steady-state final value only afterthe input carry to that stage has been propagated. In this regard, consider output S3 in Inputs A3 and B3 are available as soon as input signals are applied to the adder. However, input carry C3 does not settle to its final value until C2 is available from the previous stage. Similarly, C2 has to wait for C1 and so on down to C0. Thus, only after the carry propagates and ripples through all stages will the last output S3 and carry C4 settle to their final correct value.

The number of gate levels for the carry propagation can be found from the circuit of the full adder. The circuit is redrawn with different labels in Fig. for convenience. The input and output variables use the subscript i to denote a typical stage of the adder.

The signals at Pi and Gi settle to their steady-state values after they propagate through their respective gates. These two signals are common to all half adders and depend on only the input augend and addend bits. The signal from the input carry Ci to the output carry Ci+1 propagates through an AND gate and an OR gate, which constitute two gate levels. If there are four full adders in the adder, the output carry C4 would have 2 * 4 = 8 gate levels from C0 to C4. For an n -bit adder, there are 2 n gate levels for the carry to propagate from input to output.



The carry propagation time is an important attribute of the adder because it limits the speed with which two numbers are added. Although the adder—or, for that matter,any combinational circuit—will always have some value at its output terminals, theoutputs will not be correct unless the signals are given enough time to propagate throughthe gates connected from the inputs to the outputs. Since all other arithmetic operations are implemented by successive additions, the time consumed during the addition process is critical. An obvious solution for reducing the carry propagation delay time is to
employ faster gates with reduced delays. However, physical circuits have a limit to their capability. Another solution is to increase the complexity of the equipment in such away that the carry delay time

is reduced. There are several techniques for reducing the carry propagation time in a parallel adder. The most widely used technique employs the principle of carry lookahead logic .Consider the circuit of the full adder shown in Fig. 4.10 . If we define two new binary variables

$$P_i = A_i \{ B_i$$

$$G_i = A_iB_i$$

the output sum and carry can respectively be expressed as

$$S_i = P_i \{ C_i$$

$$C_{i+1} = G_i + P_iC_i$$

Gi is called a carry generate , and it produces a carry of 1 when both Ai and Bi are 1, regardless of the input carry Ci. Pi is called a carry propagate , because it determines whether a carry into stage i will propagate into stage i + 1 (i.e., whether an assertion of Ci will propagate to an assertion of Ci+1).

We now write the Boolean functions for the carry outputs of each stage and substitute the value of each Ci from the previous equations:

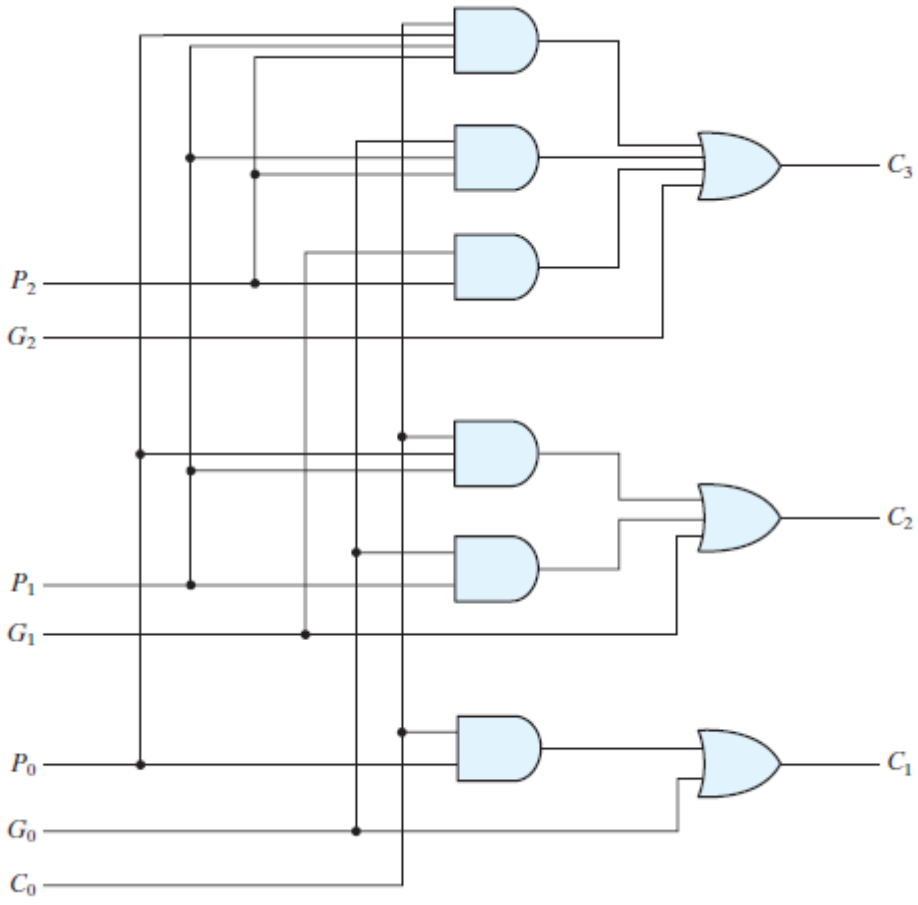
$$C0 = \text{input carry}$$

$$C1 = G0 + P0C0$$

$$C2 = G1 + P1C1 = G1 + P11G0 + P0C02 = G1 + P1G0 + P1P0C0$$

$$C3 = G2 + P2C2 = G2 + P2G1 + P2P1G0 = P2P1P0C0$$

Since the Boolean function for each output carry is expressed in sum-of-products form, each function can be implemented with one level of AND gates followed by an OR gate (or by a two-level NAND). The three Boolean functions for C1, C2, and C3 are implemented in the carry lookahead generator shown in Fig. 4.11 . Note that this circuit can add in less time because C3 does not have to wait for C2 and C1 to propagate; in fact, C3is propagated at the same time as C1 and C2. This gain in speed of operation is achieved at the expense of additional complexity (hardware). The construction of a four-bit adder with a carry lookahead scheme is shown in Fig. 4.12 . Each sum output requires two exclusive-OR gates. The output of the first exclusive-ORgate generates the Pi variable, and the AND gate generates the Gi variable. The carriesare propagated through the carry lookahead generator (similar to that in Fig. 4.11) andapplied as inputs to the second exclusive-OR gate. All output carries are generated after



Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=XcPwcpFEiy4
https://www.youtube.com/watch?v=_bJ53XErKY8

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (138)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 13

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : II- COMBINATIONAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Magnitude Comparator

Introduction :
The comparison of two numbers is an operation that determines whether one number is greater than, less than, or equal to the other number. A magnitude comparator is a combinational circuit that compares two numbers A and B and determines their relative magnitudes

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Magnitude Comparator

The comparison of two numbers is an operation that determines whether one number is greater than, less than, or equal to the other number. A magnitude comparator is a combinational circuit that compares two numbers A and B and determines their relative magnitudes. The outcome of the comparison is specified by three binary variables that indicate whether A 7 B, A = B, or A 6 B. On the one hand, the circuit for comparing two n -bit numbers has 22n entries in thetruth table and becomes too cumbersome, even with n = 3. On the other hand, as onemay suspect, a comparator circuit possesses a certain amount of regularity. Digital functionsthat possess an inherent well-defined regularity can usually be designed by means of an algorithm—a procedure which specifies a finite set of steps that, if followed, give the solution to a problem. We illustrate this method here by deriving an algorithm for the design of a four-bit magnitude comparator.

The algorithm is a direct application of the procedure a person uses to compare the relative magnitudes of two numbers. Consider two numbers, A and B , with four digits each. Write the coefficients of the numbers in descending order of significance:

A = A3 A2 A1 A0

B = B3 B2 B1 B0

Each subscripted letter represents one of the digits in the number. The two numbers are equal if all pairs of significant digits are equal: A3 = B3, A2 = B2, A1 = B1, and A0 = B0. When the numbers are binary, the digits are either 1 or 0, and the equality of each pair of bits can be expressed logically with an exclusive-NOR function as

$$xi = AiBi + A\_i\ B\_i \text{ for } i = 0, 1, 2, 3$$

where xi = 1 only if the pair of bits in position i are equal (i.e., if both are 1 or both are 0).

The equality of the two numbers A and B is displayed in a combinational circuit by an output binary variable that we designate by the symbol 1A = B2 . This binary variableis equal to 1 if the input numbers, A and B , are equal, and is equal to 0 otherwise.For equality to exist, all xi variables must be equal to 1, a condition that dictates an AND operation of all variables:

$$1A = B2 = x3x2x1x0$$

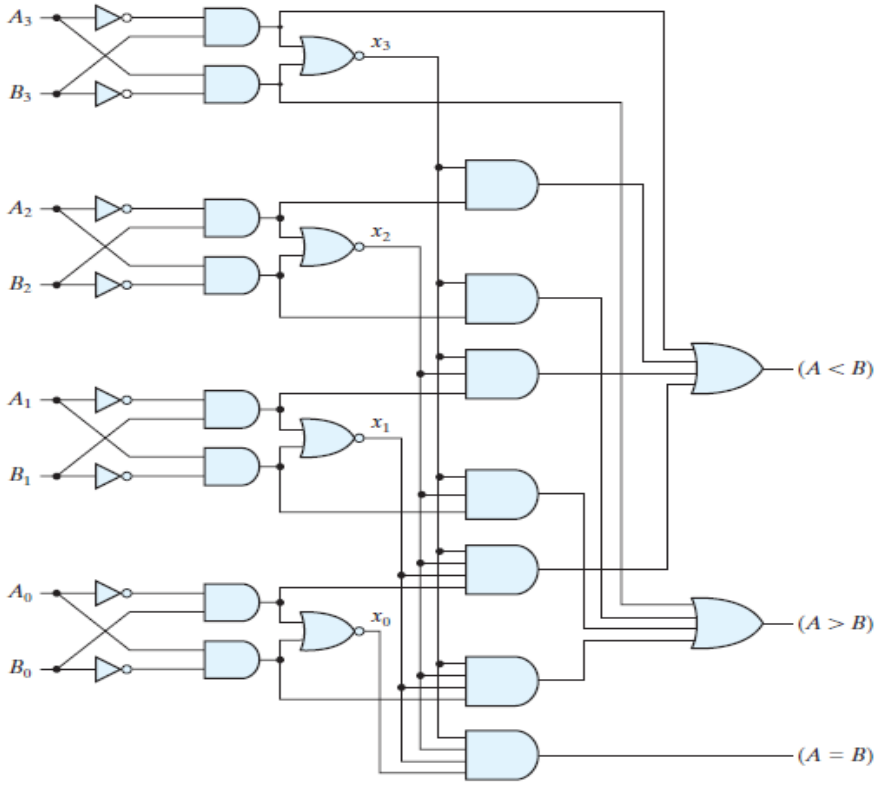
The binary variable 1A = B2 is equal to 1 only if all pairs of digits of the two numbers are equal.

To determine whether A is greater or less than B , we inspect the relative magnitudes of pairs of significant digits, starting from the most significant position. If the two digits of a pair are equal, we compare the next lower significant pair of digits. The comparison continues until a pair of unequal digits is reached. If the corresponding digit of A is 1 and that of B is 0, we conclude that A 7 B. If the corresponding digit of A is 0 and that
of B is 1, we have A 6 B. The sequential comparison can be expressed logically by the two Boolean functions

$$1A\ 7\ B2 = A3B\_3 + x3\ A\ 2B\_2 + x3x2A1B\_1 + x3x2x1A0B\_0$$

$$1A\ 6\ B2 = A\_3B3 + x3A\_2B2 + x3x2A\_1B\_1 + x3x2x1A\_n0B\_0$$

The symbols 1A 7 B2 and 1A 6 B2 are binary output variables that are equal to 1
when A 7 B and A 6 B, respectively.



Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=BhUUmbz76P0
https://www.youtube.com/watch?v=3BiSCqXXGo0

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (148)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 14

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : II- COMBINATIONAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Parity generator and checker

Introduction :
The parity generating technique is one of the most widely used error detection techniques for the data transmission. In digital systems, when binary data is transmitted and processed , data may be subjected to noise so that such noise can alter 0s (of data bits) to 1s and 1s to 0s.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

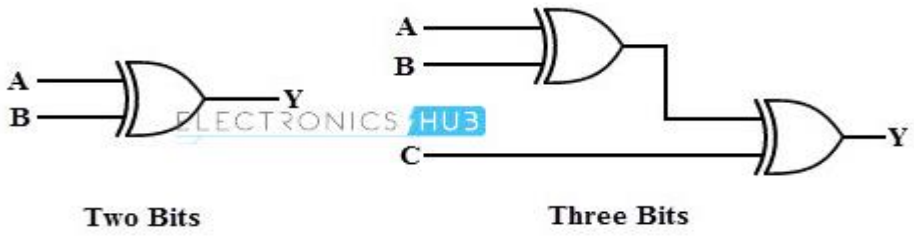
Detailed content of the Lecture:
Parity generator and checker

A parity generator is a combinational logic circuit that generates the parity bit in the transmitter. On the other hand, a circuit that checks the parity in the receiver is called parity checker. A combined circuit or devices of parity generators and parity checkers are commonly used in digital systems to detect the single bit errors in the transmitted data word.

The sum of the data bits and parity bits can be even or odd . In even parity, the added parity bit will make the total number of 1s an even amount whereas in odd parity the added parity bit will make the total number of 1s odd amount.

The basic principle involved in the implementation of parity circuits is that sum of odd number of 1s is always 1 and sum of even number of 1s is always zero. Such error detecting and correction can be implemented by using Ex-OR gates (since Ex-OR gate produce zero output when there are even number of inputs).

To produce two bits sum, one Ex-OR gate is sufficient whereas for adding three bits two Ex-OR gates are required as shown in below figure.



It is combinational circuit that accepts an n-1 bit stream data and generates the additional bit that is to be transmitted with the bit stream. This additional or extra bit is termed as a parity bit.

In **even parity** bit scheme, the parity bit is '**0**' if there are **even number of 1s** in the data stream and the parity bit is '**1**' if there are **odd number of 1s** in the data stream.

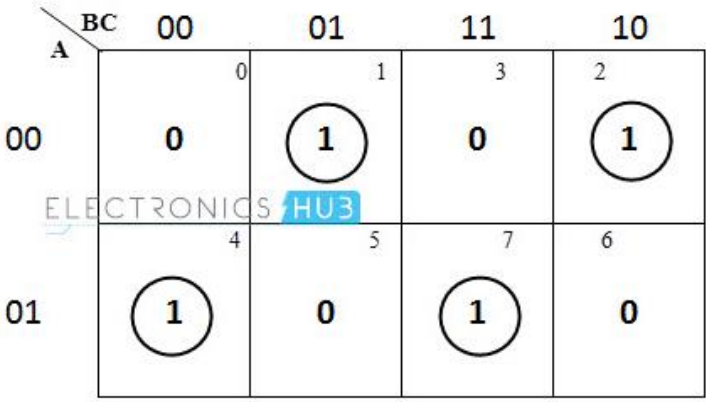
In **odd parity** bit scheme, the parity bit is '**1**' if there are **even number of 1**s in the data stream and the parity bit is '**0**' if there are **odd number of 1s** in the data stream. Let us discuss both even and odd parity generators.

Even Parity Generator

Let us assume that a 3-bit message is to be transmitted with an even parity bit. Let the three inputs A, B and C are applied to the circuits and output bit is the parity bit P. The total number of 1s must be even, to generate the even parity bit P.

The figure below shows the truth table of even parity generator in which 1 is placed as parity bit in order to make all 1s as even when the number of 1s in the truth table is odd.

3-bit message			Even parity bit generator (P)
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



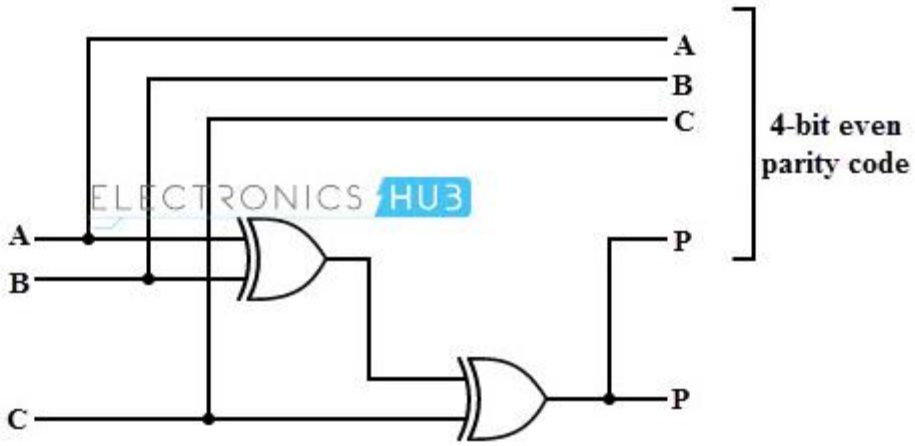
From the above truth table, the simplified expression of the parity bit can be written as

$$P = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C$$

$$= \bar{A}\,(\bar{B}\,C + B\,\bar{C}) + A\,(\bar{B}\,\bar{C} + B\,C)$$

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

$$P = A \oplus B \oplus C$$



Parity Check

It is a logic circuit that checks for possible errors in the transmission. This circuit can be an even parity checker or odd parity checker depending on the type of parity generated at the transmission end. When this circuit is used as even parity checker, the number of input bits must always be even.

When a parity error occurs, the ‘sum even’ output goes low and ‘sum odd’ output goes high. If this logic circuit is used as an odd parity checker, the number of input bits should be odd, but if an error occurs the ‘sum odd’ output goes low and ‘sum even’ output goes high.

Odd Parity Checker

Consider that a three bit message along with odd parity bit is transmitted at the transmitting end. Odd parity checker circuit receives these 4 bits and checks whether any error are present in the data.

If the total number of 1s in the data is odd, then it indicates no error, whereas if the total number of 1s is even then it indicates the error since the data is transmitted with odd parity at transmitting end.

The below figure shows the truth table for odd parity generator where **PEC =1** if the 4-bit message received consists of **even number of 1s** (hence the error occurred) and **PEC= 0** if the message contains **odd number of 1s** (that means no error).

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

PEC = (A Ex-NOR B) Ex-NOR (C Ex-NOR D)

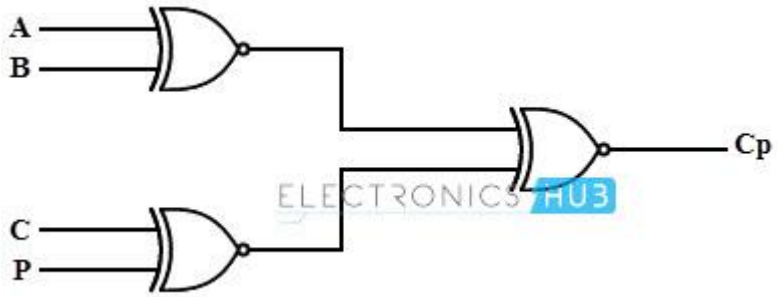
The expression for the odd parity checker can be designed by using three Ex-NOR gates as shown below.

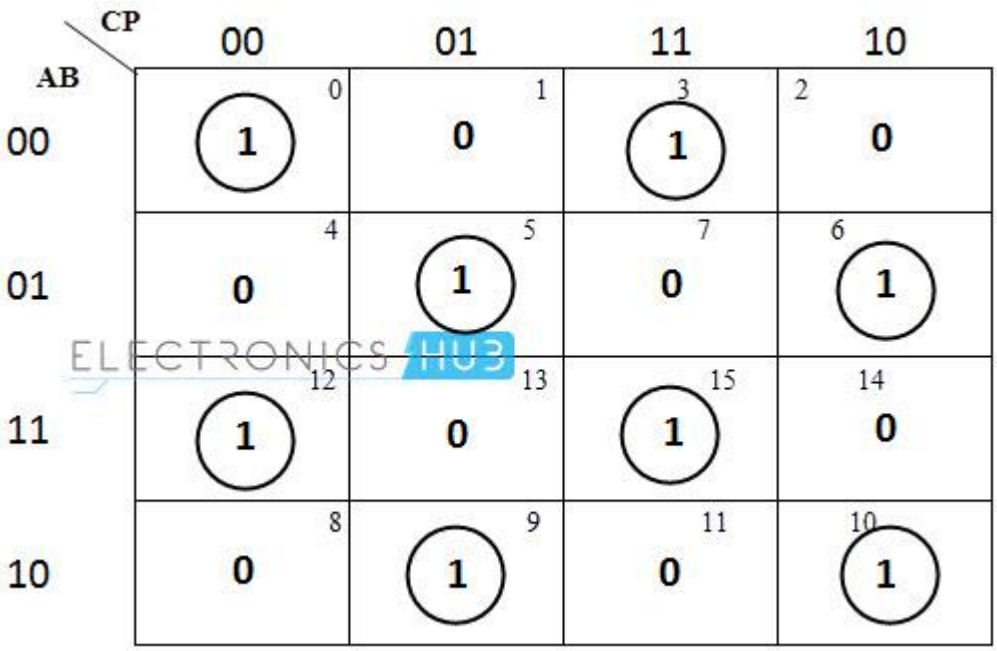
PEC = (A Ex-NOR B) Ex-NOR (C Ex-NOR D)

The expression for the odd parity checker can be designed by using three Ex-NOR gates as shown below.

PEC = (A Ex-NOR B) Ex-NOR (C Ex-NOR D)

The expression for the odd parity checker can be designed by using three Ex-NOR gates as shown below.





Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=ke-kpusTSvs
https://www.youtube.com/watch?v=789NN7c1RdI

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (156)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 15

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : II- COMBINATIONAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Encoder, Decoder

Introduction :
Encoder is a combinational circuit that has '2n' input lines and maximum of n output lines.
Decoder is a combinational circuit that has 'n' input lines and maximum of 2n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. The outputs of the decoder are nothing but the min terms of 'n' input variables lineslines, when it is enabled

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

ENCODER

An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2n (or fewer) input lines and n output lines. The output lines, as an aggregate,generate the binary code corresponding to the input value. An example of an encoderis the octal-to-binary encoder whose truth table is given in Table 4.7 . It has eight inputs(one for each of the octal digits) and three outputs that generate the corresponding binary number. It is assumed that only one input has a value of 1 at any given time.

The encoder can be implemented with OR gates whose inputs are determined directly from the truth table. Output z is equal to 1 when the input octal digit is 1, 3, 5, or 7. Output y is 1 for octal digits 2, 3, 6, or 7, and output x is 1 for digits 4, 5, 6, or 7. Theseconditions can be expressed by the following Boolean output functions:

$$z = D1 + D3 + D5 + D7$$
$$y = D2 + D3 + D6 + D7$$
$$x = D4 + D5 + D6 + D7$$

The encoder can be implemented with three OR gates.

Truth Table of an Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

The encoder defined in Table 4.7 has the limitation that only one input can be activeat any given time. If two inputs are active simultaneously, the output produces an undefined combination. For example, if D3 and D6 are 1 simultaneously, the output of the encoder will be 111 because all three outputs are equal to 1. The output 111 does notrepresent either binary 3 or binary 6. To resolve this ambiguity, encoder circuits must establish an input priority to ensure that only one input is encoded. If we establish a higher priority for inputs with higher subscript numbers, and if both D3 and D6 are 1 at
the same time, the output will be 110 because D6 has higher priority than D3.
Another ambiguity in the octal-to-binary encoder is that an output with all 0's is generated when all the inputs are 0; but this output is the same as when D0 is equal to 1. The discrepancy can be resolved by providing one more output to indicate whether atleast one input is equal to 1.

Decoder

Decoder is a combinational circuit that has 'n' input lines and maximum of 2n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. The outputs of the decoder are nothing but the min terms of 'n' input variables lineslines, when it is enabled.

2 to 4 Decoder

Let 2 to 4 Decoder has two inputs A1 & A0 and four outputs Y3, Y2, Y1 & Y0. The block diagram of 2 to 4 decoder is shown in the following figure.

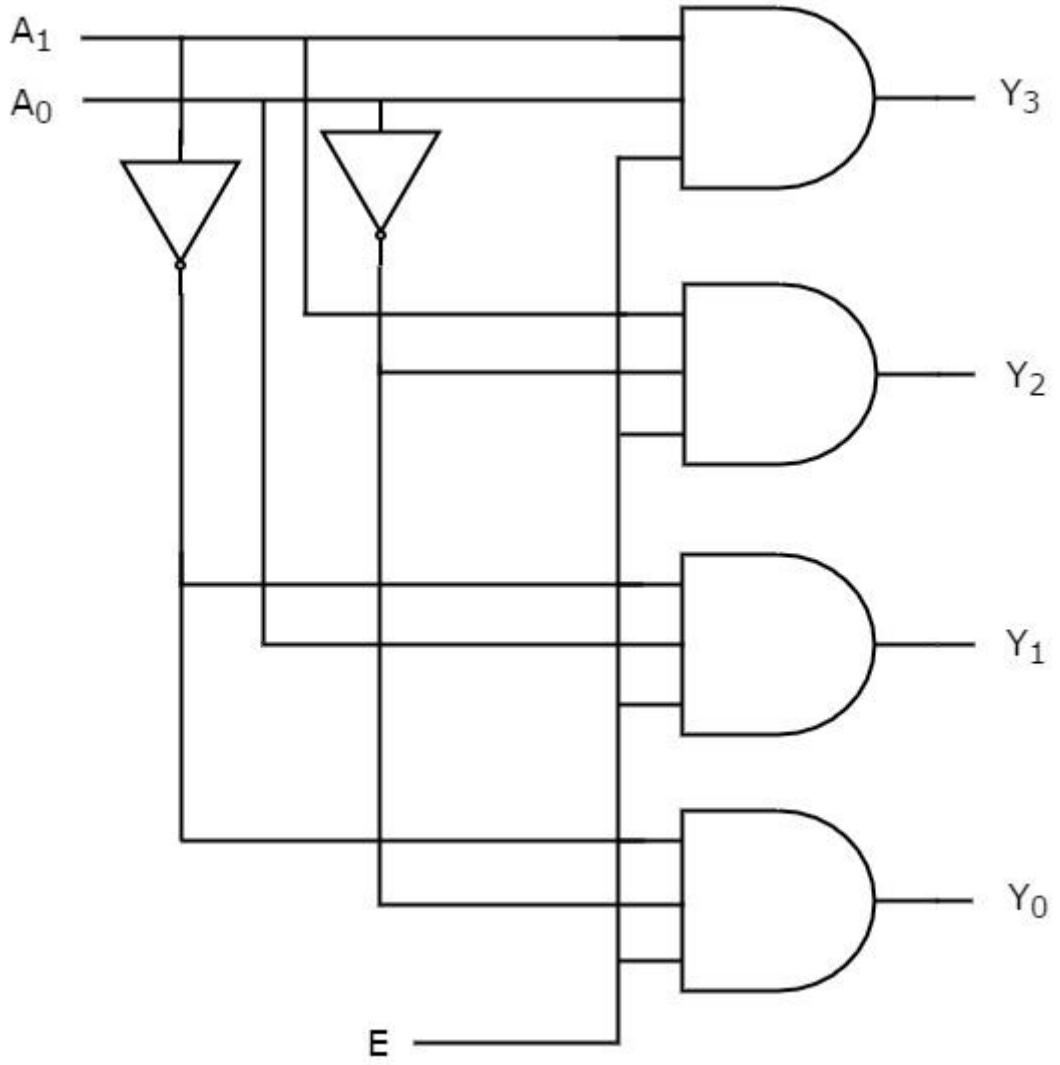
ne of these four outputs will be '1' for each combination of inputs when enable, E is '1'. The Truth table of 2 to 4 decoder is shown below.

Enable	Inputs		Outputs			
E	A1	A0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

From Truth table, we can write the Boolean functions for each output as

$$Y3=E.A1.A0Y3=E.A1.A0$$

$$Y2=E.A1.A0'$$



Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=t3Ed13z9uz8
https://www.youtube.com/watch?v=4kgPMT9k3bg

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (150)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS | **L 16**

ECE | **II / III**

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : II- COMBINATIONAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Multiplexer/ Demultiplexer

Introduction :
A multiplexer also known as a data selector, is a device that selects between several analog or digital input signals and forwards it to a single output line.
A demultiplexer (or demux) is a device that takes a single input line and routes it to one of several digital output lines. A demultiplexer of 2n outputs has n select lines, which are used to select which output line to send the input.

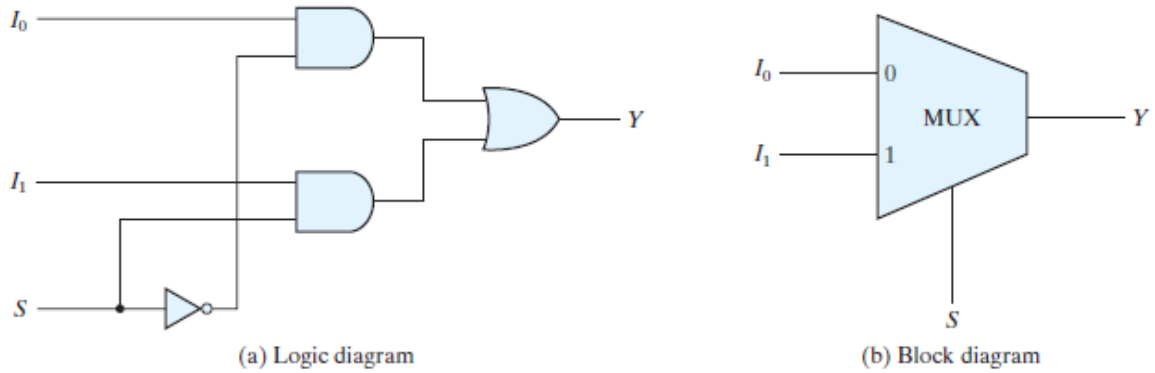
Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:
A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally, there are 2n input lines and n selection lines whose bit combinations determine which input is selected.

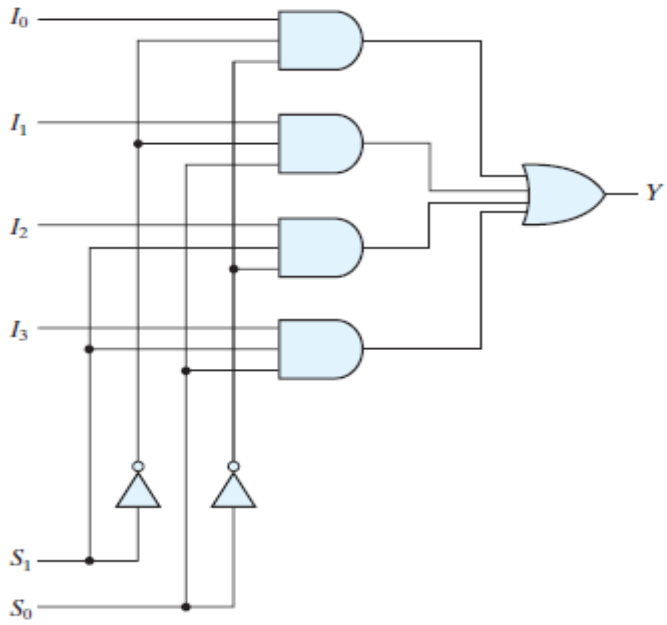
A two-to-one-line multiplexer connects one of two 1-bit sources to a common destination,as shown in Fig. 4.24 . The circuit has two data input lines, one output line, and one selection line S . When S = 0, the upper AND gate is enabled and I0 has a path to the output. When S = 1, the lower AND gate is enabled and I1 has a path to the output.
The multiplexer acts like an electronic switch that selects one of two sources. The block diagram of a multiplexer is sometimes depicted by a wedge-shaped symbol, as shown in Fig. 4.24(b) . It suggests visually how a selected one of multiple data sources is directed into a single destination. The multiplexer is often labeled “MUX” in block diagrams. A four-to-one-line multiplexer is shown in Fig. 4.25 . Each of the four inputs, I0through I3, is applied to one input of an AND gate. Selection lines S1 and S0 are decoded to select a particular AND gate. The outputs of the AND gates are applied to a single OR gate that provides the one-line output. The function table lists the input that is passed to the output for each combination of the binary selection values. To demonstrate the operation of the circuit, consider the case when S1S0 = 10. The AND gate associated with input I2 has two of its inputs equal to 1 and the third input connected to I2. The other three AND gates have at least one input equal to 0, which makes their outputsequal to 0. The output of the OR gate is now equal to the value of I2, providing a pathfrom the selected input to the output. A multiplexer is also called a data selector , since

it selects one of many inputs and steers the binary information to the output line. The size of a multiplexer is specified by



(a) Logic diagram (b) Block diagram

FIGURE 4.24
Two-to-one-line multiplexer



(a) Logic diagram

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

FIGURE 4.25
Four-to-one-line multiplexer

the number 2n of its data input lines and the single output line. The n selection lines are implied from the 2n data lines. As in decoders, multiplexers may have an enable input tocontrol the operation of the unit. When the enable input is in the inactive state, the outputs are disabled, and when it is in the active state, the circuit functions as a normal multiplexer.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=HU31Wh-4_K8

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (158)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 17

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : II- COMBINATIONAL CIRCUITS

Date of Lecture:

Topic of Lecture:

Function realization using Multiplexer, Decoder

Introduction :

A multiplexer also known as a data selector, is a device that selects between several analog or digital input signals and forwards it to a single output line.

A Decoder is a combinational circuit that converts binary information from input lines to unique output lines. Apart from the Input lines, a decoder may also have an Enable input line.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Function realization using Multiplexer

consider the Boolean function

$$F(x, y, z) = \_(1, 2, 6, 7)$$

This function of three variables can be implemented with a four-to-one-line multiplexeras shown in Fig. 4.27 . The two variables x and y are applied to the selection lines in that order; x is connected to the S1 input and y to the S0 input. The values for the data input lines are determined from the truth table of the function. When xy = 00, output F is equal to z because F = 0 when z = 0 and F = 1 when z = 1. This requires that variable z be applied to data input 0. The operation of the multiplexer is such that when xy = 00, data input 0 has a path to the output, and that makes F equal to z . In a similar fashion,we can determine the required input to data lines 1, 2, and 3 from the value of F when xy = 01, 10, and 11, respectively. This particular example shows all four possibilities that can be obtained for the data inputs. The general procedure for implementing any Boolean function of n variables with a multiplexer with n - 1 selection inputs and 2n-1 data inputs follows from the previous example. To begin with, Boolean function is listed in a truth table. Then first n - 1 variables in the table are applied to the selection inputs of the multiplexer. For each combination of the selection variables, we evaluate the output as a function of the last variable. This function can be 0, 1, the variable, or the complement of the variable. These values are then applied to the data inputs in the proper order.

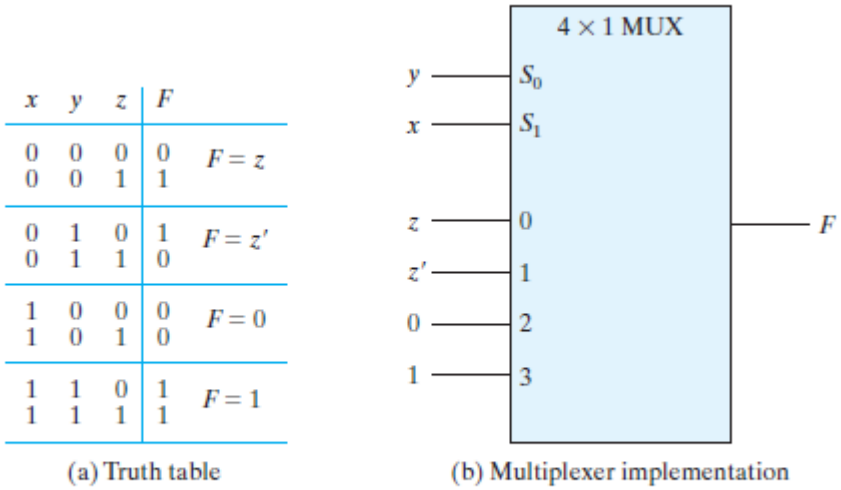


FIGURE 4.27
Implementing a Boolean function with a multiplexer

Function realization using Decoder

A decoder provides the 2n minterms of n input variables. Each asserted output of the decoder is associated with a unique pattern of input bits. Since any Boolean function can be expressed in sum-of-minterms form, a decoder that generates the minterms of the function, together with an external OR gate that forms their logical sum, provides a hardware implementation of the function. In this way, any combinational circuit with n inputs and m outputs can be implemented with an n -to-2 n -line decoder and m OR gates.

The procedure for implementing a combinational circuit by means of a decoder and OR gates requires that the Boolean function for the circuit be expressed as a sum of minterms. A decoder is then chosen that generates all the minterms of the input variables.

The inputs to each OR gate are selected from the decoder outputs according to the list of minterms of each function. This procedure will be illustrated by an example that implements a full-adder circuit.

From the truth table of the full adder (see Table 4.4), we obtain the functions for the combinational circuit in sum-of-minterms form:

$$S(x, y, z) = \_(1, 2, 4, 7)$$
$$C(x, y, z) = \_(3, 5, 6, 7)$$

Since there are three inputs and a total of eight minterms, we need a three-to-eight-line decoder. The implementation is shown in Fig. 4.21 . The decoder generates the eightminterms for x , y , and z . The OR gate for output S forms the logical sum of minterms 1,2, 4, and 7. The OR gate for output C forms the logical sum of minterms 3, 5, 6, and 7.

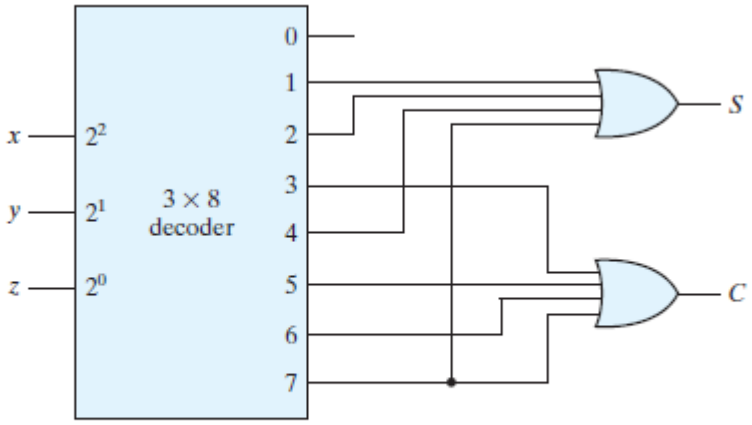


FIGURE 4.21
Implementation of a full adder with a decoder

A function with a long list of minterms requires an OR gate with a large number of inputs. A function having a list of k minterms can be expressed in its complemented form F_ with 2n - k minterms. If the number of minterms in the function is greater than 2n>2,then F_ can be expressed with fewer minterms. In such a case, it is advantageous to use a NOR gate to sum the minterms of F_. The output of the NOR gate complements this sum and generates the normal output F . If NAND gates are used for the decoder, as in Fig. 4.19 , then the external gates must be NAND gates instead of OR gates. This is because a two-level NAND gate circuit implements a sum-of-minterms function and is equivalent to a two-level AND–OR circuit.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=4Nmv9_n3WsI

https://www.youtube.com/watch?v=HU31Wh-4_K8

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (154)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 18

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : II- COMBINATIONAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Code converters

Introduction :
To convert from binary code A to binary code B, the input lines must supply the bit combination of elements as specified by code A and the output lines must generate the corresponding bit combination of code B. A combinational circuit performsthis transformation by means of logic gates. The design procedure will be illustrated by an example that converts binary coded decimal (BCD) to the excess-3 code for the decimal digits.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Boolean algebra
- Boolean postulates and laws
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:
The bit combinations assigned to the BCD and excess-3 codes are listed in Table 1.5.Since each code uses four bits to represent a decimal digit, there must be four input variables and four output variables. We designate the four input binary variables by the symbols A, B, C, and D, and the four output variables by w, x, y , and z . The truth table relating the input and output variables is shown in Table 4.2 . The bit combinations for the inputs and their corresponding outputs are obtained directly from Section 1.7. Note that four binary variables may have 16 bit combinations, but only 10 are listed in the truth table. The six bit combinations not listed for the input variables are don't-care combinations. These values have no meaning in BCD and we assume that they will never occur in actual operation of the circuit. Therefore, we are at liberty to assign to the output variables either a 1 or a 0, whichever gives a simpler circuit.
The maps in Fig. 4.3 are plotted to obtain simplified Boolean functions for the outputs. Each one of the four maps represents one of the four outputs of the circuit as a function of the four input variables. The 1's marked inside the squares are obtained from the minterms that make the output equal to 1. The 1's are obtained from the truth table by going over the output columns one at a time. For example, the column under output z has five 1's; therefore, the map for z has five 1's, each being in a square corresponding to the minterm that makes z equal to 1. The sixdon't-care minterms 10 through 15 are marked with an X . One possible way to simplify the functions into sum-of-products form is listed under the map of each variable.
A two-level logic diagram for each output may be obtained directly from the Boolean expressions derived from the maps. There are various other possibilities for a logic diagram that implements this circuit. The expressions obtained in Fig. 4.3 may be manipulated algebraically for the purpose of using common

gates for two or more outputs. This manipulation, shown next, illustrates the flexibility obtained with multiple-output systems when

Table 4.2
Truth Table for Code Conversion Example

Input BCD				Output Excess-3 Code			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

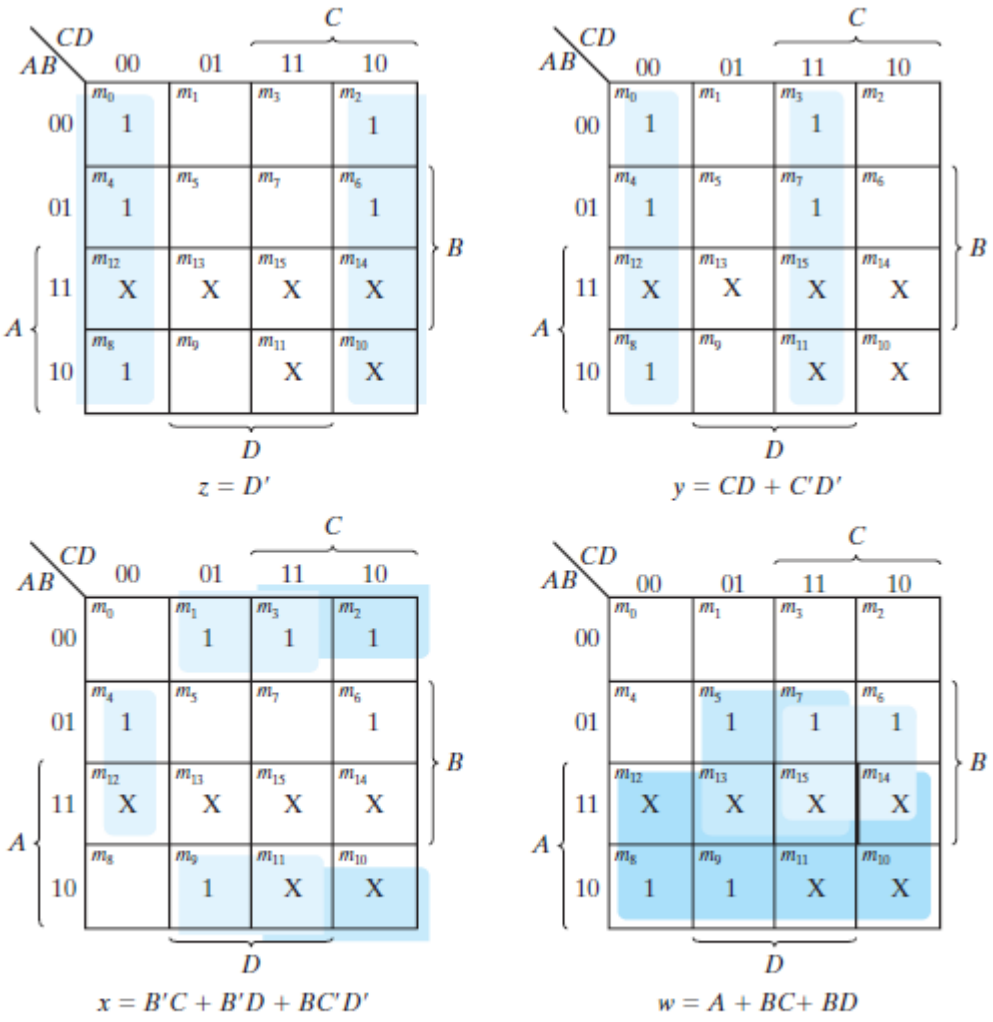


FIGURE 4.3
Maps for BCD-to-excess-3 code converter

implemented with three or more levels of gates:
z = D_
y = CD + C_D_ = CD + 1C + D2_
x = B_C + B_D + BC_D_ = B_1C + D2 + BC_D_
= B_1C + D2 + B1C + D2_
w = A + BC + BD = A + B1C + D2

The logic diagram that implements these expressions is shown in Fig. 4.4 . Note that the OR gate whose output is C + D has been used to implement partially each of three outputs. Not counting input inverters, the implementation in sum-of-products form requires seven AND gates and three OR gates. The implementation of Fig. 4.4 requires four AND gates, four OR gates, and one inverter. If only the normal inputs are available, the firstimplementation will require inverters for variables B, C , and D , and the second implementation will require inverters for variables B and D . Thus, the three-level logic

circuit requires fewer gates, all of which in turn require no more than two inputs.

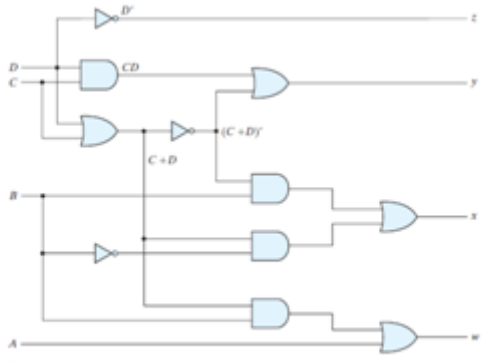


FIGURE 4.4
Logic diagram for BCD-to-excess-3 code converter

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=CZT7dgX6CTo
https://www.youtube.com/watch?v=kgL5UaSVuro

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (130)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 19

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : III- SEQUENTIAL CIRCUITS **Date of Lecture:**

Topic of Lecture: Flip-flops - SR, JK

Introduction: Storage elements that operate with signal levels (rather than signal transitions) are referred to as latches; those controlled by a clock transition are flip-flops. Latches are said to be level sensitive devices; flip-flops are edge-sensitive devices.

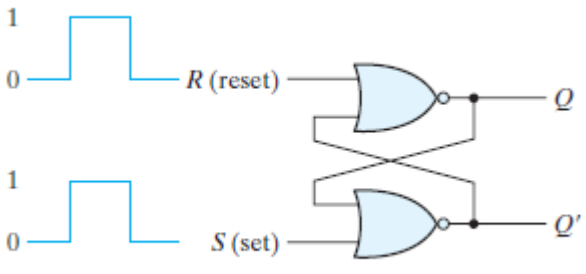
Prerequisite knowledge for Complete understanding and learning of Topic:

- Latches
- Logic Gates
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

SR Latch

The SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates, and two inputs labeled S for set and R for reset. The SR latch constructed with two cross-coupled NOR gates is shown in Figure. The latch has two useful states. When output Q = 1 and Q' = 0, the latch is said to be in the set state. When Q = 0 and Q' = 1, it is in the reset state. Outputs Q and Q' are normally the complement of each other. However, when both inputs are equal to 1 at the same time, a condition in which both outputs are equal to 0 (rather than be mutually complementary) occurs. If both inputs are then switched to 0 simultaneously, the device will enter an unpredictable or undefined state or a metastable state. Consequently, in practical applications, setting both inputs to 1 is forbidden.



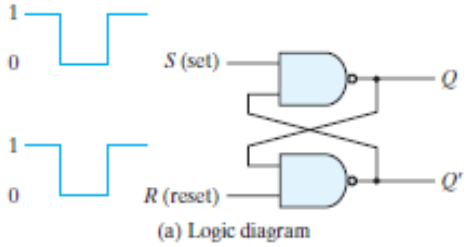
(a) Logic diagram

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

(b) Function table

Under normal conditions, both inputs of the latch remain at 0 unless the state has to be changed. The application of a momentary 1 to the S input causes the latch to go to the set state. The S input must go back to 0 before any other changes take place, in order to avoid the occurrence of an undefined next

state that results from the forbidden input condition. As shown in the function table of Figure, two input conditions cause the circuit to be in



(a) Logic diagram

S	R	Q	Q′	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

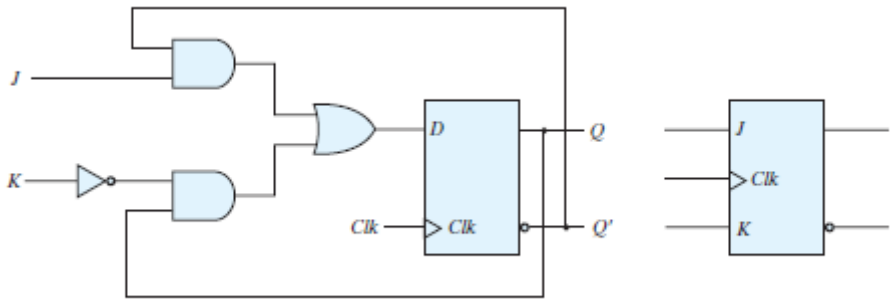
(b) Function table

the set state. The first condition (S = 1,R = 0) is the action that must be taken by input S to bring the circuit to the set state. Removing the active input from S leaves the circuit in the same state. After both inputs return to 0, it is then possible to shift to the reset state by momentary applying a 1 to the R input. The 1 can then be removed from R, whereupon the circuit remains in the reset state. Thus, when both inputs S and R are equal to 0, the latch can be in either the set or the reset state, depending on which input was most recently a 1.

SR Latch

Synchronized by a clock signal, the JK flip-flop has two inputs and performs all three operations. The circuit diagram of a JK flip-flop constructed with a D flip-flop and gates is shown in Fig. 5.12 (a). The J input sets the flip-flop to 1, the K input resets it to 0, and when both inputs are enabled, the output is complemented. This can be verified by investigating the circuit applied to the D input:

D = JQ' + K'Q



(a) Circuit diagram (b) Graphic symbol

When J = 1 and K = 0, D = Q' + Q = 1, so the next clock edge sets the output to 1. When J = 0 and K = 1, D = 0, so the next clock edge resets the output to 0. When both J = K = 1 and D = Q', the next clock edge complements the output. When both J = K = 0 and D = Q, the clock edge leaves the output unchanged. The graphic symbol for the JK flip-flop is shown in Fig. 5.12 (b). It is similar to the graphic symbol of the D flip-flop, except that now the inputs are marked J and K .

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=tSti91b6qec
https://www.youtube.com/watch?v=V_JJ_YrPU6Q

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (193, 200)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 20

ECE

II / III

Course Name with Code :19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : III-SEQUENTIAL CIRCUITS Date of Lecture:

Topic of Lecture:
Flip-flops - D and T- Master-Slave

Introduction : Storage elements that operate with signal levels (rather than signal transitions) are referred to as latches; those controlled by a clock transition are flip-flops. Latches are said to be level sensitive devices; flip-flops are edge-sensitive devices.

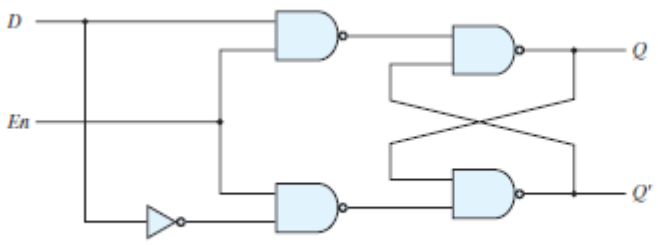
Prerequisite knowledge for Complete understanding and learning of Topic:

- Latches
- Logic Gates
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

D Latch

One way to eliminate the undesirable condition of the indeterminate state in the SRlatch is to ensure that inputs S and R are never equal to 1 at the same time. This isdone in the D latch, shown in Fig. 5.6 . This latch has only two inputs: D (data) and



(a) Logic diagram

En	*D*	Next state of *Q*
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

(b) Function table

En (enable). The D input goes directly to the S input, and its complement is appliedto the R input. As long as the enable input is at 0, the cross-coupled SR latch has bothinputs at the 1 level and the circuit cannot change state regardless of the value of D .The D input is sampled when En = 1. If D = 1, the Q output goes to 1, placing thecircuit in the set state. If D = 0, output Q goes to 0, placing the circuit in the resetstate.

T- Master-Slave

The T (toggle) flip-flop is a complementing flip-flop and can be obtained from a JKflip-flop when inputs J and K are tied together. This is shown in Fig. 5.13 (a). WhenT = 0 (J = K = 0), a clock edge does not change the output. When T = 1 (J = K = 1),a clock edge complements the output. The

complementing flip-flop is useful for designingbinary counters.

The T flip-flop can be constructed with a D flip-flop and an exclusive-OR gate asshown in Fig. 5.13 (b). The expression for the D input is

$$D = T \oplus Q = TQ' + T'Q$$

When T = 0, D = Q and there is no change in the output. When T = 1, D = Q' andthe output complements. The graphic symbol for this flip-flop has a T symbol in the input.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=kQ9WICIFWnU
https://www.youtube.com/watch?v=PyWQGZDx-aY

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (195, 201)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 21

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : III- SEQUENTIAL CIRCUITS

Date of Lecture:

Topic of Lecture:

Triggering - Characteristic table and equation – Application table

Introduction :

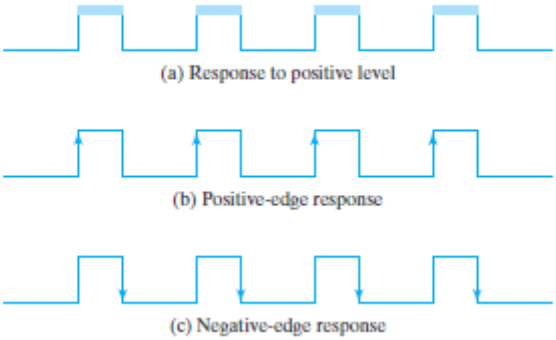
The state of a latch or flip-flop is switched by a change in the control input. This momentary change is called a trigger, and the transition it causes is said to trigger the flip-flop.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Latches
- Logic Gates
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Flip-flop circuits are constructed in such a way as to make them operate properly when they are part of a sequential circuit that employs a common clock. The problem with the latch is that it responds to a change in the level of a clock pulse. As shown in Fig. 5.8 (a), a positive level response in the enable input allows changes in the output when the D input changes while the clock pulse stays at logic 1. The key to the proper operation of a flip-flop is to trigger it only during a signal transition.



(a) Response to positive level

(b) Positive-edge response

(c) Negative-edge response

Edge-Triggered D Flip-Flop

The construction of a D flip-flop with two D latches and an inverter is shown in Fig. 5.9. The first latch is called the master and the second the slave. The circuit samples the D input and changes its

output Q only at the negative edge of the synchronizing or controlling clock (designated as Clk). When the clock is 0, the output of the inverter is 1. The slave latch is enabled, and its output Q is equal to the master output Y . The master latch is disabled because Clk = 0. When the input pulse changes to the logic-1 level, the data from the external D input are transferred to the master. The slave, however, is disabled
as long as the clock remains at the 1 level, because its enable input is equal to 0. Any change in the input changes the master output at Y, but cannot affect the slave output. When the clock pulse returns to 0, the master is disabled and is isolated from the D input. At the same time, the slave is enabled and the value of Y is transferred to the output of the flip-flop at Q . Thus, a change in the output of the flip-flop can be triggered only by and during the transition of the clock from 1 to 0.

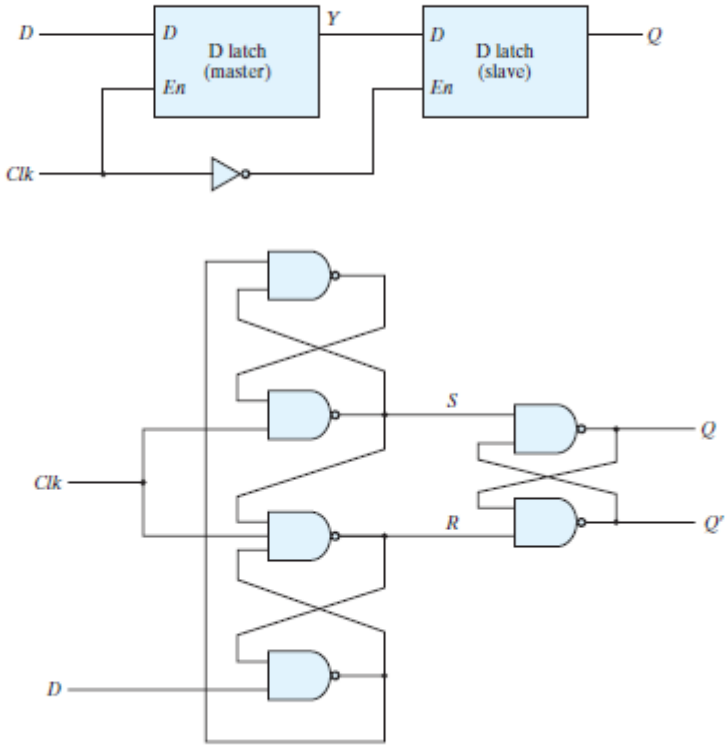


FIGURE 5.10
D-type positive-edge-triggered flip-flop

In sum, when the input clock in the positive-edge-triggered flip-flop makes a positive transition, the value of D is transferred to Q . A negative transition of the clock (i.e., from 1 to 0) does not affect the output, nor is the output affected by changes in D when Clk is in the steady logic-1 level or the logic-0 level. Hence, this type of flip-flop responds to the transition from 0 to 1 and nothing else.

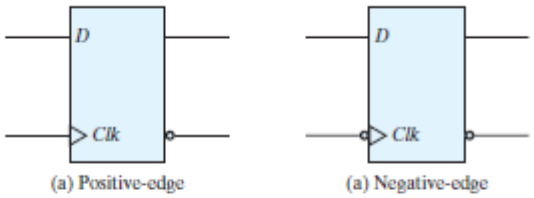


FIGURE 5.11
Graphic symbol for edge-triggered *D* flip-flop

Characteristic Tables

A characteristic table defines the logical properties of a flip-flop by describing its operation in tabular form. The characteristic tables of three types of flip-flops are presented in Table 5.1 . They define the next state (i.e., the state that results from a clock transition) as a function of the inputs and the present state. Q (t) refers to the present state (i.e., the state present prior to the application of a clock edge).

Q(t + 1) is the next state one clock period later. Note that the clock edge input is not included in the

characteristic table, but is implied to occur between times t and t + 1. Thus, Q(t) denotes the state of the flip-flop immediately before the clock edge, and Q(t + 1) denotes the state that results from the clock transition.

Table 5.1
Flip-Flop Characteristic Tables

JK Flip-Flop

J	K	Q(t + 1)	
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

D Flip-Flop

D	Q(t + 1)	
0	0	Reset
1	1	Set

T Flip-Flop

T	Q(t + 1)	
0	Q(t)	No change
1	Q'(t)	Complement

The characteristic table for the JK flip-flop shows that the next state is equal to the present state when inputs J and K are both equal to 0. This condition can be expressed as Q(t + 1) = Q(t), indicating that the clock produces no change of state. When K = 1 and J = 0, the clock resets the flip-flop and Q(t + 1) = 0. With J = 1 and K = 0, the flip-flop sets and Q(t + 1) = 1. When both J and K are equal to 1, the next state changes to the complement of the present state, a transition that can be expressed as

$$Q(t + 1) = Q'(t).$$

Characteristic Equations

The logical properties of a flip-flop, as described in the characteristic table, can be expressed algebraically with a characteristic equation. For the D flip-flop, we have the characteristic equation

$$Q(t + 1) = D$$

which states that the next state of the output will be equal to the value of input D in the present state. The characteristic equation for the JK flip-flop can be derived from the characteristic table or from the circuit of Fig. 5.12 . We obtain

$$Q(t + 1) = JQ' + K'Q$$

where Q is the value of the flip-flop output prior to the application of a clock edge. The characteristic equation for the T flip-flop is obtained from the circuit of Fig. 5.13 :

$$Q(t + 1) = T \oplus Q = TQ' + T'Q$$

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=xQgLFK0Jv2g
https://www.youtube.com/watch?v=WE2wbOAVzdo

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (196)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 22

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : III- SEQUENTIAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Asynchronous counters

Introduction :
Counter is a sequential circuit. A digital circuit which is used for a counting pulses is known counter. Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Latches
- Logic Gates
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

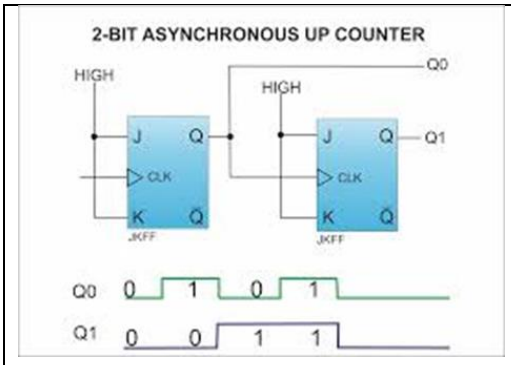
Detailed content of the Lecture:

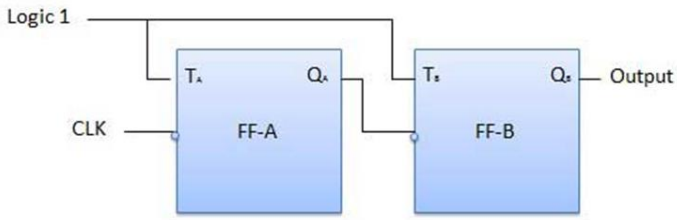
Asynchronous counters

If the flip-flops do not receive the same clock signal, then that counter is called as Asynchronous counter. The output of system clock is applied as clock signal only to first flip-flop. The remaining flip-flops receive the clock signal from output of its previous stage flip-flop. Hence, the outputs of all flip-flops do not change affect at the same time. An 'N' bit Asynchronous binary up counter consists of 'N' T flip-flops. It counts from 0 to $2N - 1$.

2 bit Asynchronous or ripple up counters

The logic diagram of a 2-bit ripple up counter is shown in figure. The toggle (T) flip flop are being used. But we can use the JK flip-flop also with J and K connected permanently to logic 1. External clock is applied to the clock input of flip-flop A and QA output is applied to the clock input of the next flip-flop i.e. FF-B





Operation

Initially let both the FFs be in the reset state

Q_BQ_A = 00 initially

After 1st negative clock edge
As soon as the first negative clock edge is applied, FF-A will toggle and QA will be equal to 1.QA is connected to clock input of FF-B. Since Q_A has changed from 0 to 1, it is treated as the positive clock edge by FF-B. There is no change in Q_B because FF-B is a negative edge triggered FF.

Q_BQ_A = 01 after the first clock pulse.

After 2nd negative clock edge
On the arrival of second negative clock edge, FF-A toggles again and Q_A = 0. The change in Q_A acts as a negative clock edge for FF-B. So it will also toggle, and QB will be 1.Q_BQ_A = 10 after the second clock pulse.

After 3rd negative clock edge
On the arrival of 3rd negative clock edge, FF-A toggles again and Q_A become 1 from 0. Since this is a positive going change, FF-B does not respond to it and remains inactive. So QB does not change and continues to be equal to 1.Q_BQ_A = 11 after the third clock pulse.

After 4th negative clock edge
On the arrival of 4th negative clock edge, FF-A toggles again and Q_A becomes 1 from 0. This negative change in Q_A acts as clock pulse for FF-B. Hence it toggles to change QB from 1 to 0.Q_BQ_A = 00 after the fourth clock pulse.

Clock	Counter output		State number	Deciimal Counter output
	Q_B	Q_A		
Initially	0	0	—	0
1st	0	1	1	1
2nd	1	0	2	2
3rd	1	1	3	3
4th	0	0	4	0

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=iHT-Jcgq2X0
https://www.youtube.com/watch?v=eEeBh8jfDjg

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (266)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 23

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : III- SEQUENTIAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Synchronous counters

Introduction :
Counter is a sequential circuit. A digital circuit which is used for a counting pulses is known counter. Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Latches
- Logic Gates
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Synchronous counters

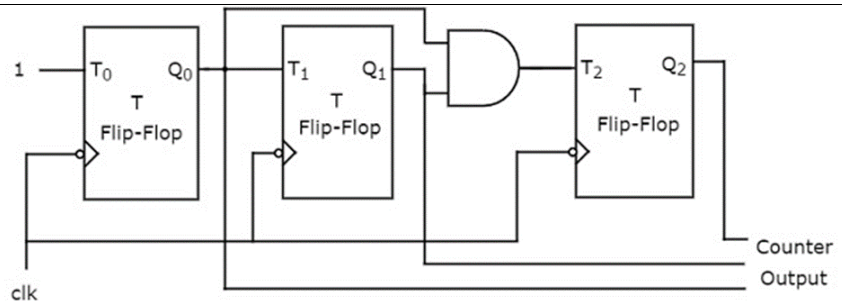
If all the flip-flops receive the same clock signal, then that counter is called as Synchronous counter. Hence, the outputs of all flip-flops change affect at the same time. Now, let us discuss the following two counters one by one.

Synchronous Binary up counter

Synchronous Binary down counter

3 Bit Synchronous Binary Up Counter

An 'N' bit Synchronous binary up counter consists of 'N' T flip-flops. It counts from 0 to $2N - 1$. The block diagram of 3-bit Synchronous binary up counter is shown in the following figure.



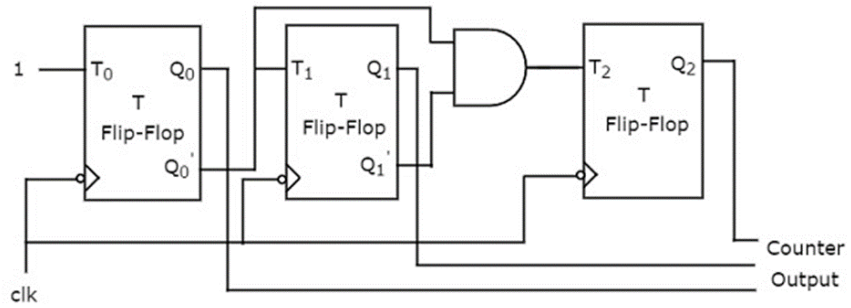
The 3-bit Synchronous binary up counter contains three T flip-flops & one 2-input AND gate. All these flip-flops are negative edge triggered and the outputs of flip-flops change affect synchronously. The T inputs of first, second and third flip-flops are 1, Q0 & Q1Q0 respectively.

The output of first T flip-flop toggles for every negative edge of clock signal. The output of second T flip-flop toggles for every negative edge of clock signal if Q0 is 1. The output of third T flip-flop toggles for every negative edge of clock signal if both Q0 & Q1 are 1

Counter State	Q_2	Q_1	Q_0
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
7	1	1	1

3-bit Synchronous binary down counter

An 'N' bit Synchronous binary down counter consists of 'N' T flip-flops. It counts from $2N - 1$ to 0. The block diagram of 3-bit Synchronous binary down counter is shown in the following figure.



The 3-bit Synchronous binary down counter contains three T flip-flops & one 2-input AND gate. All these flip-flops are negative edge triggered and the outputs of flip-flops change affect synchronously. The T inputs of first, second and third flip-flops are 1, Q0′ Q1′ & Q2′ respectively.

The output of first T flip-flop toggles for every negative edge of clock signal. The output of second T flip-flop toggles for every negative edge of clock signal if Q0′ is 1. The output of third T flip-flop toggles for every negative edge of clock signal if both Q1′ & Q0′ are 1.

No of negative edge of Clock	Q_0 LSB	Q_1	Q_2 MSB
0	0	0	0
1	1	1	1
2	0	1	1
3	1	0	1
4	0	0	1
5	1	1	0
6	0	1	0
7	1	0	0

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=6e8oV2blkGs
https://www.youtube.com/watch?v=svFUEJkoeVY

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (270)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 24

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : III- SEQUENTIAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Shift registers - Types – Universal shift registers

Introduction :
A register is a group of flip-flops, each one of which shares a common clock and is capable of storing one bit of information. An n -bit register consists of a group of n flip-flops capable of storing n bits of binary information.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Latches
- Logic Gates
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Shift registers

A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction, is called a shift register. The logical configuration of a shift register consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive common clock pulses, which activate the shift of data from one stage to the next.

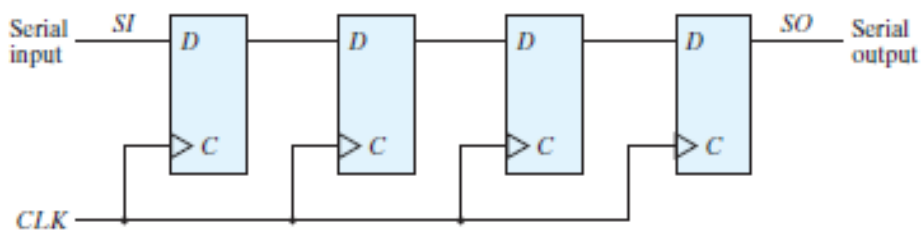


FIGURE 6.3
Four-bit shift register

The simplest possible shift register is one that uses only flip-flops, as shown in Fig. 6.3 . The output of a given flip-flop is connected to the D input of the flip-flop at its right. This shift register is unidirectional (left-to-right). Each clock pulse shifts the contents of the register one bit position to the right. The configuration does not support a left shift. The serial input determines what goes into the leftmost flip-flop during the shift. The serial output is taken from the output of the rightmost flip-flop. Sometimes it is necessary to control the shift so that it occurs only with certain pulses, but not with others. As with the

data register discussed in the previous section, the clock's signal can be suppressed by gating the clock signal to prevent the register from shifting. A preferred alternative in highspeed circuits is to suppress the clock action, rather than gate the clock signal, by leaving the clock path unchanged, but recirculating the output of each register cell back through a two-channel mux whose output is connected to the input of the cell. When the clock action is not suppressed, the other channel of the mux provides a datapath to the cell.

Universal Shift Register

If the flip-flop outputs of a shift register are accessible, then information entered serially by shifting can be taken out in parallel from the outputs of the flip-flops. If a parallel load capability is added to a shift register, then data entered in parallel can be taken out in serial fashion by shifting the data stored in the register.

Some shift registers provide the necessary input and output terminals for parallel transfer. They may also have both shift-right and shift-left capabilities. The most general shift register has the following capabilities:

1. A clear control to clear the register to 0.
2. A clock input to synchronize the operations.
3. A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift right.
4. A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift left.
5. A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
6. n parallel output lines.
7. A control state that leaves the information in the register unchanged in response to the clock. Other shift registers may have only some of the preceding functions, with at least one shift operation.

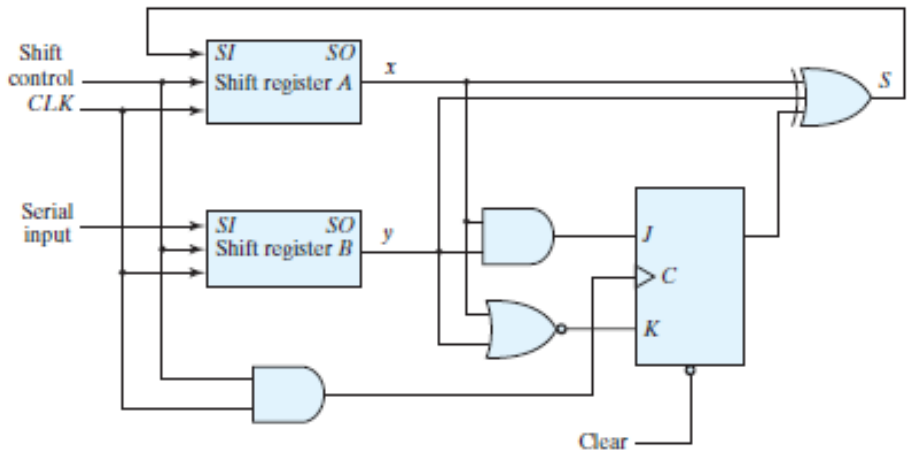


FIGURE 6.6
Second form of serial adder

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=unorn9n-UpE
https://www.youtube.com/watch?v=AEGzpMlOsvc

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (258)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS | **L 25**

ECE | **II / III**

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : III- SEQUENTIAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Ring counter

Introduction :
Counters can be designed to generate any desired sequence of states. A divide-by- N counter (also known as a modulo- N counter) is a counter that goes through a repeated sequence of N states. The sequence may follow the binary count or may be any other arbitrary sequence. Counters are used to generate timing signals to control the sequence of operations in a digital system. Counters can also be constructed by means of shift registers.

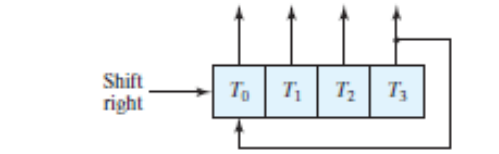
Prerequisite knowledge for Complete understanding and learning of Topic:

- Latches
- Logic Gates
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

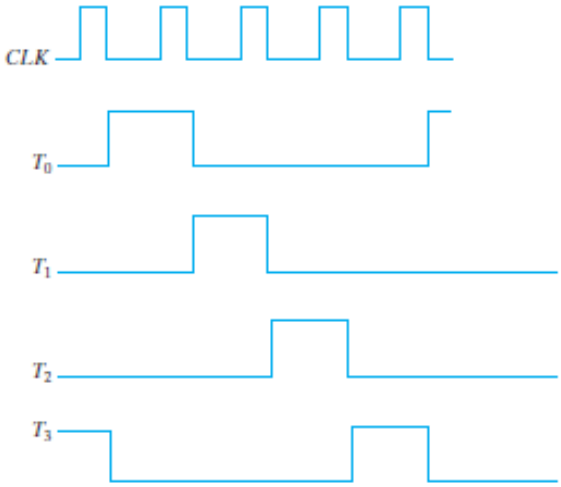
Detailed content of the Lecture:

Ring Counter

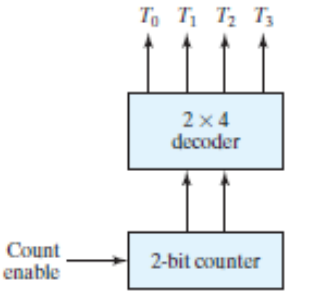
Timing signals that control the sequence of operations in a digital system can be generated by a shift register or by a counter with a decoder. A ring counter is a circular shift register with only one flip-flop being set at any particular time; all others are cleared. The single bit is shifted from one flip-flop to the next to produce the sequence of timing signals. Figure 6.17 (a) shows a four-bit shift register connected as a ring counter. The initial value of the register is 1000 and requires Preset/Clear flip-flops. The single bit is shifted right with every clock pulse and circulates back from T3 to T0. Each flip-flop is in the 1 state once every four clock cycles and produces one of the four timing signals shown in Fig. 6.17 (b). Each output becomes a 1 after the negative-edge transition of a clock pulse and remains 1 during the next clock cycle.



(a) Ring-counter (initial value = 1000)



(b) Sequence of four timing signals



(c) Counter and decoder

FIGURE 6.17
Generation of timing signals

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=yOW-JsJL1Ks

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (280)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 26

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : III- SEQUENTIAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Johnson Counter

Introduction :
Counters can be designed to generate any desired sequence of states. A divide-by- N counter (also known as a modulo- N counter) is a counter that goes through a repeated sequence of N states. The sequence may follow the binary count or may be any other arbitrary sequence. Counters are used to generate timing signals to control the sequence of operations in a digital system. Counters can also be constructed by means of shift registers.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Latches
- Logic Gates
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Johnson Counter

A k -bit ring counter circulates a single bit among the flip-flops to provide k distinguishable states. The number of states can be doubled if the shift register is connected as a switch-tail ring counter. A switch-tail ring counter is a circular shift register with the complemented output of the last flip-flop connected to the input of the first flip-flop. Figure 6.18 (a) shows such a shift register. The circular connection is made from the complemented output of the rightmost flip-flop to the input of the leftmost flip-flop.

The register shifts its contents once to the right with every clock pulse, and at the same time, the complemented value of the E flip-flop is transferred into the A flip-flop. Starting from a cleared state, the switch-tail ring counter goes through a sequence of eight states, as listed in Fig. 6.18 (b). In general, a k -bit switch-tail ring counter will go through a sequence of 2 k states. Starting from all 0's, each shift operation inserts 1's from the left until the register is filled with all 1's. In the next sequences, 0's are inserted from the left until the register is again filled with all 0's.

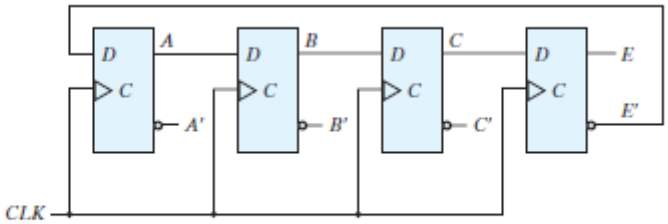
A Johnson counter is a k -bit switch-tail ring counter with 2 k decoding gates to provide outputs for 2 k timing signals. The decoding gates are not shown in Fig. 6.18 , but are specified in the last column of the table. The eight AND gates listed in the table, when connected to the circuit, will complete the construction of the Johnson counter. Since each gate is enabled during one particular state sequence, the outputs of the gates generate eight timing signals in succession.

The decoding of a k -bit switch-tail ring counter to obtain 2 k timing signals follows a regular pattern. The all-0's state is decoded by taking the complement of the two extreme flip-flop outputs. The all-1's state is decoded by taking the normal outputs of the two extreme flip-flops. All other states are decoded from an adjacent 1, 0 or 0, 1 pattern in the sequence. For example, sequence 7 has an adjacent 0, 1 pattern in flip-flops B and C . The decoded output is then obtained by taking the complement of B and the normal output of C, or B'C.

One disadvantage of the circuit in Fig. 6.18 (a) is that if it finds itself in an unused state, it will persist in moving from one invalid state to another and never find its way to a valid state. The difficulty can be corrected by modifying the circuit to avoid this undesirable condition. One correcting procedure is to disconnect the output from flip-flop B that goes to the D input of flip-flop C and instead enable the input of flip-flop C by the function

$$DC = (A + C)B$$

where DC is the flip-flop input equation for the D input of flip-flop C.



(a) Four-stage switch-tail ring counter

Sequence number	Flip-flop outputs A	B	C	E	AND gate required for output
1	0	0	0	0	$A'E'$
2	1	0	0	0	AB'
3	1	1	0	0	BC'
4	1	1	1	0	CE'
5	1	1	1	1	AE
6	0	1	1	1	$A'B$
7	0	0	1	1	$B'C$
8	0	0	0	1	$C'E$

(b) Count sequence and required decoding

FIGURE 6.18
Construction of a Johnson counter

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=X4mx7J1ckyU
https://www.youtube.com/watch?v=SXbXnfgY6jk

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (282)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 27

ECE

II / III

Course Name with Code : 19ECC04/Digital System Design

Course Teacher : Mr.S.Bhoopalan, AP/ECE

Unit : III- SEQUENTIAL CIRCUITS

Date of Lecture:

Topic of Lecture:
Serial adder / subtractor

Introduction :
To convert from binary code A to binary code B, the input lines must supply the bit combination of elements as specified by code A and the output lines must generate the corresponding bit combination of code B. A combinational circuit performsthis transformation by means of logic gates. The design procedure will be illustrated by an example that converts binary coded decimal (BCD) to the excess-3 code for the decimal digits.

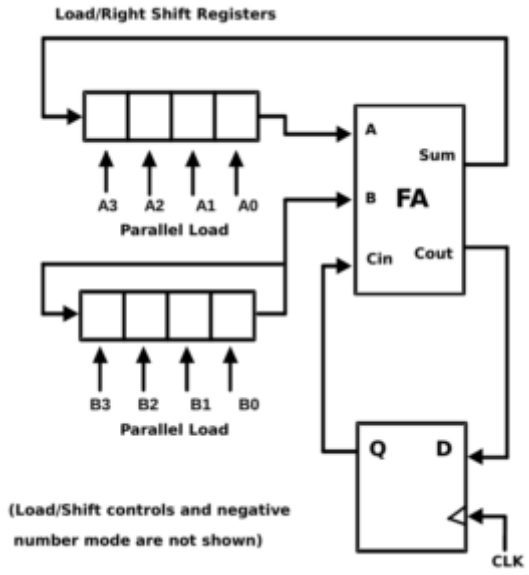
Prerequisite knowledge for Complete understanding and learning of Topic:

- Latches
- Logic Gates
- Canonical forms Sum of product and Product of sum
- Minimization using Karnaugh map

Detailed content of the Lecture:

Serial adder / subtractor

A 4-bit serial adder circuit consists of two 4-bit shift registers with parallel load, a full adder, and a D-type flip-flop for storing carry-out. A simplified schematics of the circuit is shown below:



In order to load registers A_REG and B_REG with numbers, shift capability of the registers should be disabled and loading mode should be enabled. Loading of numbers from inputs A, B to registers A_REG, B_REG occurs in one clock cycle. After loading registers with numbers, shifting mode should be enabled to perform the arithmetic operation. The addition of numbers stored in A_REG and B_REG requires 4 cycles. Starting with the least significant bit, at each cycle one bit of number A and one bit of number B are being added. The sum is stored at the most significant bit of register A_REG. Carry-out output produced after each cycle is fed back to the full adder as a carry-in of the next significant bit. For this purpose one D-type flip-flop is used as a temporary storage element. The least significant bit of B_REG is fed to the input of the most significant bit of B_REG. Hence the circuit performs rotation operation for register B_REG.

Video Content / Details of website for further learning (if any):
https://www.isabekov.pro/four-bit-serial-adder-subtractor/
https://unacademy.com/lesson/serial-adder-serial-subtractor-and-serial-adder-subtractor/ZXQER68L

Important Books/Journals for further learning including the page nos.:
Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (261)

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-28

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **IV- SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS**

Date of Lecture:

Topic of Lecture: Mealy and Moore models

Introduction :

- Mealy and Moore models are the basic models of state machines. A state machine which uses only Entry Actions, so that its output depends on the state, is called a Moore model. A state machine which uses only Input Actions, so that the output depends on the state and also on inputs, is called a Mealy model.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

Mealy and Moore models:

Finite automata may have outputs corresponding to each transition. There are two types of finite state machines that generate output

- Mealy Machine
- Moore machine

Mealy Machine

A Mealy Machine is an FSM whose output depends on the present state as well as the present input.

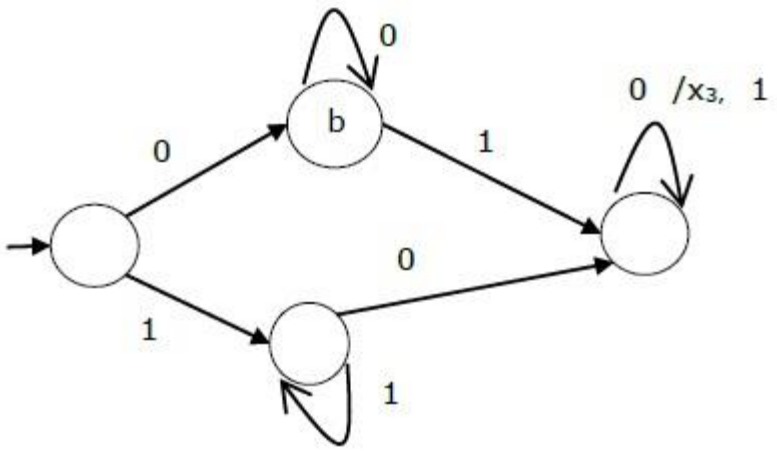
It can be described by a 6 tuple (Q, ∑, O, δ, X, q0) where −

- Q is a finite set of states.
- ∑ is a finite set of symbols called the input alphabet.
- O is a finite set of symbols called the output alphabet.
- δ is the input transition function where δ: Q × ∑ → Q
- X is the output transition function where X: Q × ∑ → O
- q0 is the initial state from where any input is processed (q0 ∈ Q).

The state table of a Mealy Machine is shown below −

Present state	Next state			
	input = 0		input = 1	
	State	Output	State	Output
→ a	b	x1	c	x1
b	b	x2	d	x3
c	d	x3	c	x1
d	d	x3	d	x2

The state diagram of the above Mealy Machine is −



Moore Machine

Moore machine is an FSM whose outputs depend on only the present state.

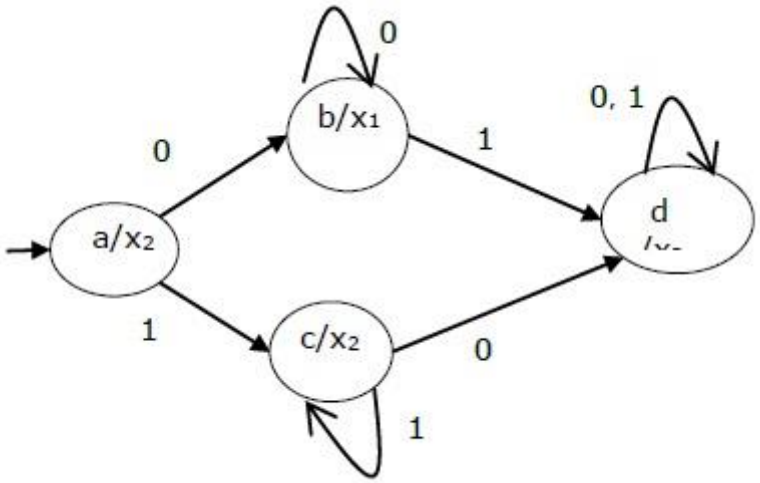
A Moore machine can be described by a 6 tuple (Q, ∑, O, δ, X, q0) where −

- Q is a finite set of states.
- ∑ is a finite set of symbols called the input alphabet.
- O is a finite set of symbols called the output alphabet.
- δ is the input transition function where δ: Q × ∑ → Q
- X is the output transition function where X: Q → O
- q0 is the initial state from where any input is processed (q0 ∈ Q).

The state table of a Moore Machine is shown below −

Present state	Next State		Output
	Input = 0	Input = 1	
→ a	b	c	x2
b	b	d	x1
c	c	d	x2
d	d	d	x3

The state diagram of the above Moore Machine is −



Mealy Machine vs. Moore Machine

The following table highlights the points that differentiate a Mealy Machine from a Moore Machine.

Mealy Machine	Moore Machine
Output depends both upon the present state and the present input	Output depends only upon the present state.
Generally, it has fewer states than Moore Machine.	Generally, it has more states than Mealy Machine.
The value of the output function is a	The value of the output function is a

	function of the transitions and the changes, when the input logic on the present state is done.	function of the current state and the changes at the clock edges, whenever state changes occur.	
	Mealy machines react faster to inputs. They generally react in the same clock cycle.	In Moore machines, more logic is required to decode the outputs resulting in more circuit delays. They generally react one clock cycle later.	

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=lH_qfCTh9Co
https://www.tutorialspoint.com/automata_theory/moore_and_mealy_machines.htm

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (190)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L-29

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **IV- SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS**

Date of Lecture:

Topic of Lecture: State Diagram, State Table

Introduction :

- A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.
- State Table. The state table representation of a sequential circuit consists of three sections labeled present state, next state and output. The present state designates the state of flip-flops before the occurrence of a clock pulse

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuits

State diagram, State Table :

In this model the effect of all previous inputs on the outputs is represented by a state of the circuit. Thus, the output of the circuit at any time depends upon its current state and the input. These also determine the next state of the circuit. The relationship that exists among the inputs, outputs, present states and next states can be specified by either the state table or the state diagram.

State Table

The state table representation of a sequential circuit consists of three sections labeled present state, next state and output. The present state designates the state of flip-flops before the occurrence of a clock pulse. The next state shows the states of flip-flops after the clock pulse, and the output section lists the value of the output variables during the present state.

State Diagram

In addition to graphical symbols, tables or equations, flip-flops can also be represented graphically by a state diagram. In this diagram, a state is represented by a circle, and the transition between states is indicated by directed lines (or arcs) connecting the circles. An

example of a state diagram is shown in Figure 3 below.

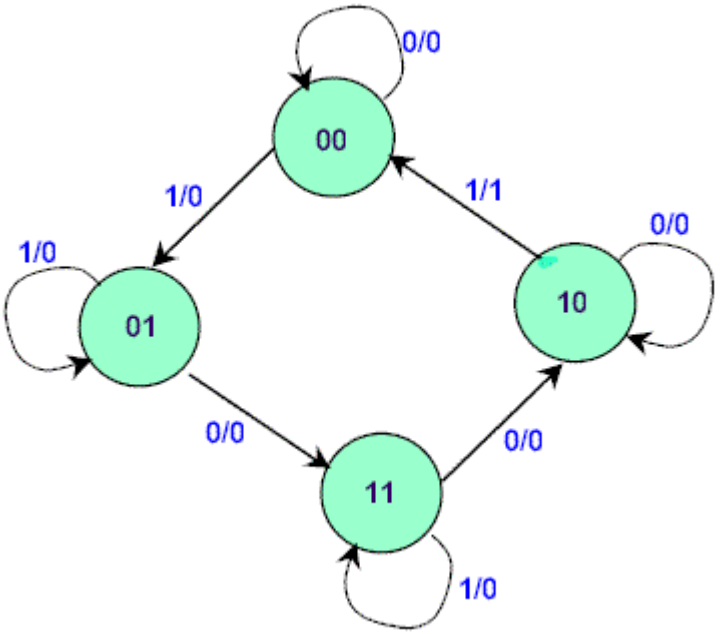


Figure 3. State Diagram

The binary number inside each circle identifies the state the circle represents. The directed lines are labeled with two binary numbers separated by a slash (/). The input value that causes the state transition is labeled first. The number after the slash symbol / gives the value of the output. For example, the directed line from state 00 to 01 is labeled 1/0, meaning that, if the sequential circuit is in a present state and the input is 1, then the next state is 01 and the output is 0. If it is in a present state 00 and the input is 0, it will remain in that state. A directed line connecting a circle with itself indicates that no change of state occurs. The state diagram provides exactly the same information as the state table and is obtained directly from the state table.

Example: This example is taken from P. K. Lala, *Practical Digital Logic Design and Testing*, Prentice Hall, 1996, p.155.

Consider a sequential circuit shown in Figure 4. It has one input x, one output Z and two state variables Q1Q2 (thus having four possible present states 00, 01, 10, 11).

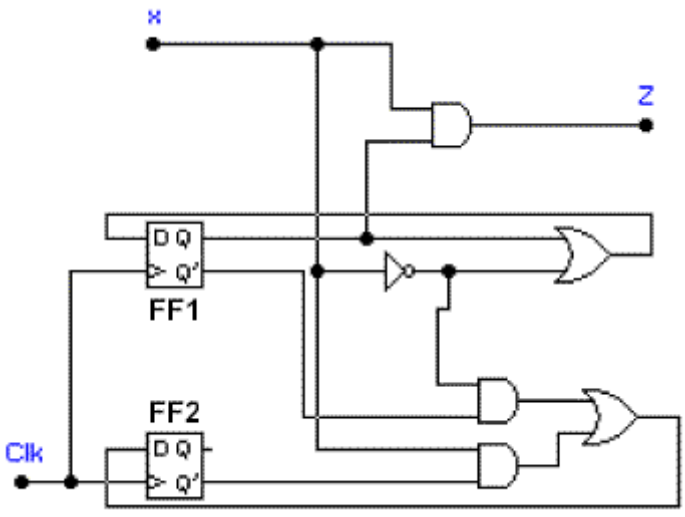


Figure 4. A Sequential Circuit

The behavior of the circuit is determined by the following Boolean expressions:

Z = x * Q1

D1 = x' + Q1

D2 = x * Q2' + x' * Q1'

These equations can be used to form the state table. Suppose the present state (i.e. Q1Q2) = 00 and input x = 0. Under these conditions, we get Z = 0, D1 = 1, and D2 = 1. Thus the next state of the circuit D1D2 = 11, and this will be the present state after the clock pulse has been applied. The output of the circuit corresponding to the present state Q1Q2 = 00 and x = 1 is Z = 0. This data is entered into the state table as shown in Table 2.

Present State	Next State		Output	
Q1Q2	x = 0	x = 1	x = 0	x = 1
0 0	1 1	0 1	0	0
0 1	1 1	0 0	0	0
1 0	1 0	1 1	0	1
1 1	1 0	1 0	0	1

Video Content / Details of website for further learning (if any):

http://osp.mans.edu.eg/cs212/Seq_circuitst_FF_states.htm
https://www.youtube.com/watch?v=NNOSWnTHakY

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (206 - 208)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L-30

ECE

II/III

Course Name with Code : **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **IV- SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS**

Date of Lecture:

Topic of Lecture: State minimization, State Assignment

Introduction :

- The number of states in a sequential circuit is closely related to the complexity of the resulting circuit. It is therefore desirable to know when two or more states are equivalent in all aspects. The process of eliminating the equivalent or redundant states from a state table/diagram is known as state reduction.
- When implementing a given state table, we often desire a "state assignment" that will minimize the amount of logic required. Having 1's next to each other in a K-map will generally result in simpler (lower cost) logic equations.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuits

State minimization

Any design process must consider the problem of minimizing the cost of the final circuit. The two most obvious cost reductions are reductions in the number of flip-flops and the number of gates.

The number of states in a sequential circuit is closely related to the complexity of the resulting circuit. It is therefore desirable to know when two or more states are equivalent in all aspects. The process of eliminating the equivalent or redundant states from a state table/diagram is known as state reduction.

State Reduction (State Minimization)

By reducing or minimising the total number of states, the number of flip-flops required for a design is also reduced. For example if a finite state machine drops from 8 states to 4 states, only two flip-flops are required rather than three. The total number of states is reduced by eliminating the equivalent states.

Two states are equivalent if they have the same output for all inputs, and if they transition to equivalent states on all inputs.

Present State	Next State		Present Output
	X=0	X=1	
a	d	c	0
b	d	c	0
c	d	a	0
d	d	c	1

- Comparing states a and b [a,b], we can see that the outputs are the same 0->0 and the next states when X=0 is d->d (d=d), when X=1 is c->c (c=c). Therefore the states a and b are equivalent and one is redundant and can be eliminated.
- Comparing states a and c [a,c], we can see that the outputs are the same 0->0 and the next states when X=0 is d->d (d=d), when X=1 is c->a. For c->a, as the states are referring to each other (we are comparing states a and c), we can ignore this. Therefore the states a and c are equivalent and one can be eliminated.
- Comparing state a and d [a,d], we can see that the outputs are different 0->1. Therefore we can conclude that the states a and d are NOT equivalent. No further checks are required.
- In the end, the 4 states a,b,c,d can be reduced to 2 a,d and the number of flip flops required for this design is reduced from 2 to 1.

State Reduction using the Implication Table

One method to eliminate the redundant states is to use an implication table. Using the implication table involves the following steps:

Present State	Next State		Present Output
	X=0	X=1	
a	g	c	0
b	f	h	0
c	e	d	1
d	a	c	0
e	c	a	1
f	f	b	1
g	a	c	0
h	c	g	1

State Assignment:

When implementing a given state table, we often desire a “state assignment” that will minimize the amount of logic required. Having 1’s next to each other in a K-map will generally result in simpler (lower cost) logic equations. Our objective, therefore, is to somehow make an assignment that results in groups of 1’s being next to each other. One solution is to simply try all possible assignments and then pick the one those results in the least amount of logic. However, this is not practical when there are more than a handful of states. A more practical approach is to follow a set of guidelines that tend to result in a lot of 1’s next to each other in the required K-maps. (Recall that adjacent cells in a Kmap differ by only one variable.) We will use the following guidelines: 1) (Highest priority) States which have the same next state for each possible input combination should be given

adjacent assignments. Identical next states in only some "columns" (i.e., for some input combinations) may be included in this list but with lesser priority. 2) (Medium priority) States which are next states of the same state should be given adjacent assignments. 3) (Lowest priority) States which have the same output for each input should be given adjacent assignments. States having the same output for only some columns (input combinations) may still be included but with lesser priority You might want a particular state to be a "Reset" state, so assign that one to the state having all bits as zero. (E.g., state "A" below.) Clearing all flip-flops (assuming a clear line is attached to each one) will therefore put the state machine in this state. We won't necessarily be able to satisfy all guidelines, but in general the better job we do in satisfying the higher priority guidelines, the simpler the logic tends to be. We can either lay out the states on a Boolean "assignment" cube or in a K-map style grid. A sixstate example assignment (requiring three state variables; i.e., three flip-flops).

Video Content / Details of website for further learning (if any):

https://cseweb.ucsd.edu/classes/sp10/cse140/lectures/wk6.pdf
https://www.youtube.com/watch?v=IXORIs1dHs0

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (231 - 236)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-31

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **IV- SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS**

Date of Lecture:

Topic of Lecture: Excitation table

Introduction :

- An excitation table shows the minimum inputs that are necessary to generate a particular next state (in other words, to "excite" it to the next state) when the current state is known.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuits

Excitation table

In electronics design, an excitation table shows the minimum inputs that are necessary to generate a particular next state (in other words, to "excite" it to the next state) when the current state is known. They are similar to truth tables and state tables, but rearrange the data so that the current state and next state are next to each other on the left-hand side of the table, and the inputs needed to make that state change happen are shown on the right side of the table.

States		Input
Present	Next	T
0	0	0
0	1	1
1	0	1

1	1	0

Flip-flop excitation tables

In order to obtain the excitation table of a flip-flop, one needs to draw the Q(t) and Q(t + 1) for all possible cases (e.g., 00, 01, 10, and 11), and then make the value of flip-flop such that on giving this value, one shall receive the input as Q(t + 1) as desired.

T flip-flop

The characteristic equation of a T flip-flop is {\displaystyle Q({\text{next}})=TQ'+T'Q=T\oplus Q}

SR flip-flop

("X" is "don't care")

States		Inputs	
Present	Next	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

The characteristic equation of a SR flip-flop is {\displaystyle Q({\text{next}})=S+QR'}

States		Inputs	
Present	Next	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

States		Input
Present	Next	D
0	0	0
0	1	1
1	0	0
1	1	1

JK flip-flop

("X" is "don't care")

The characteristic equation of a JK flip-flop is {\displaystyle Q({\text{next}})=JQ'+K'Q}

D flip-flop

The characteristic equation of a D flip-flop is {\displaystyle Q({\text{next}})=D}

Video Content / Details of website for further learning (if any):

https://en.wikipedia.org/wiki/Excitation_table#:~:text=In%20electronics%20design%2C%20an%20excitation,the%20current%20state%20is%20known.
https://www.youtube.com/watch?v=uiKKRPZbuXA

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (231-236)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L-32

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **IV- SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS**

Date of Lecture:

Topic of Lecture: Design of Synchronous sequential circuits: Counters

Introduction :

- The flip-flops receive the same clock signal, then that counter is called as Synchronous counter. Hence, the outputs of all flip-flops change affect at the same time.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuits

Synchronous Counters

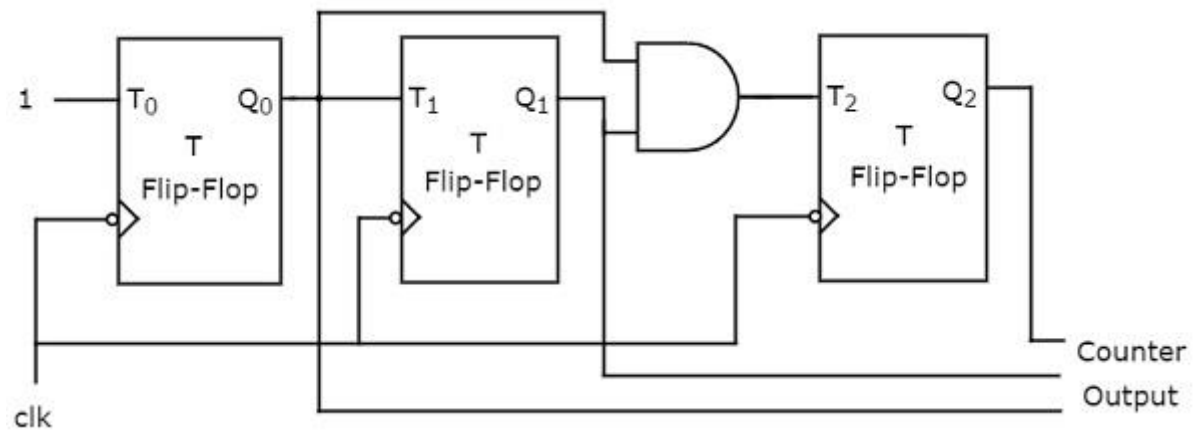
If all the flip-flops receive the same clock signal, then that counter is called as Synchronous counter. Hence, the outputs of all flip-flops change affect at the same time.

Now, let us discuss the following two counters one by one.

- Synchronous Binary up counter
- Synchronous Binary down counter

Synchronous Binary Up Counter

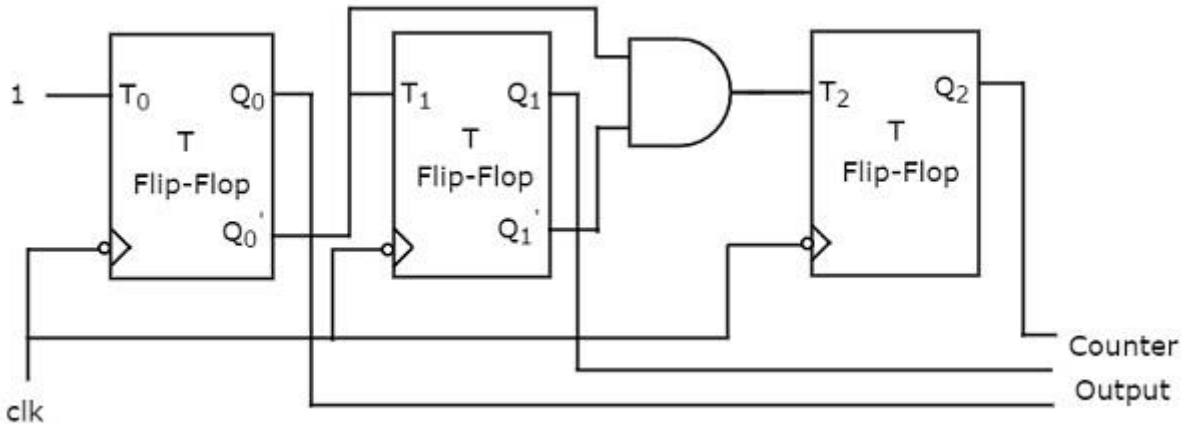
An 'N' bit Synchronous binary up counter consists of 'N' T flip-flops. It counts from 0 to 2. The block diagram of 3-bit Synchronous binary up counter is shown in the following figure.



The 3-bit Synchronous binary up counter contains three T flip-flops & one 2-input AND gate. All these flip-flops are negative edge triggered and the outputs of flip-flops change affectaffect synchronously. The T inputs of first, second and third flip-flops are 1, Q0Q0 & Q1Q0Q1Q0 respectively.
The output of first T flip-flop toggles for every negative edge of clock signal. The output of second T flip-flop toggles for every negative edge of clock signal if Q0Q0 is 1. The output of third T flip-flop toggles for every negative edge of clock signal if both Q0Q0 & Q1Q1 are 1.

Synchronous Binary Down Counter

An 'N' bit Synchronous binary down counter consists of 'N' T flip-flops. It counts from 2□ − 1 to 0. The block diagram of 3-bit Synchronous binary down counter is shown in the following figure.



The 3-bit Synchronous binary down counter contains three T flip-flops & one 2-input AND gate. All these flip-flops are negative edge triggered and the outputs of flip-flops change affectaffect synchronously. The T inputs of first, second and third flip-flops are 1, Q0′Q0′ &' Q1′Q1′Q0′Q0′ respectively.
The output of first T flip-flop toggles for every negative edge of clock signal. The output of second T flip-flop toggles for every negative edge of clock signal if Q0′Q0′ is 1. The output of third T flip-flop toggles for every negative edge of clock signal if both Q1′Q1′ & Q0′Q0′ are 1.

Video Content / Details of website for further learning (if any):

https://www.electronics-tutorials.ws/counter/count_3.html
https://www.youtube.com/watch?v=5vkWccb7uO4

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (271 - 278)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L-33

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **IV- SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS**

Date of Lecture:

Topic of Lecture: Design of Synchronous sequential circuits: Sequence generators

Introduction :

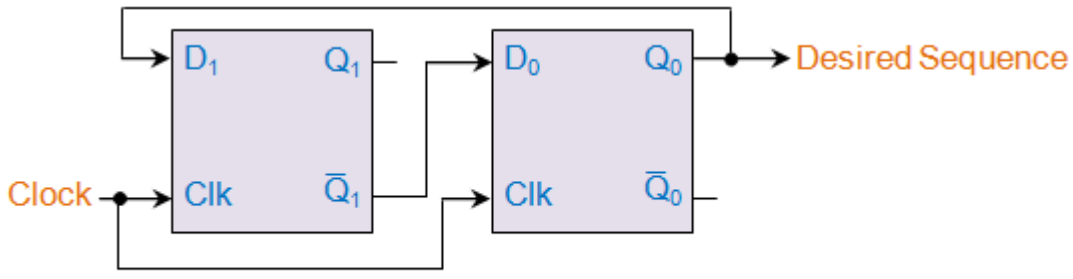
- A sequence generator is one kind of digital logic circuit. The main function of this is to generate a set of outputs. Every output is one of a number of binary or Q-ary logic levels or symbols.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuits

Sequence generators

We all know that there are counters which pass through a definite number of states in a pre-determined order. For example, a 3-bit up-counter counts from 0 to 7 while the same order is reversed in the case of 3-bit down counter. These circuits when suitably manipulated can be made to count till an intermediate level also. This means that instead of counting till 7, we can terminate the process by resetting the counter just at, say, 5. Such counters are then known as mod-N counters. However, even this case, the order in which they count will not alter. But, what-if if we need to through a specific pattern which does not adhere to this standard way of counting? The solution would be to design a *sequence generator*.



Sequence Generator Designed Using D Flip-Flops

This is because, the sequence generators are nothing but a set of digital circuits which are designed to result in a specific bit sequence at their output. There are several ways

in which these circuits can be designed including those which are based on multiplexers and flip-flops. Here in this article we deal with the designing of sequence generator using D flip-flops (please note that even JK flip-flops can be made use of). As an example, let us consider that we intend to design a circuit which moves through the states 0-1-3-2 before repeating the same pattern. The steps involved during this process are as follows.

Step1

At first, we need to determine the number of flip-flops which would be required to achieve our objective. In our example, there are 4 states which are identical to the states of a 2-bit counter except the order in which they transit. From this, we can guess the requirement of flip-flops to be 2 in order to achieve our objective.

Step2

Having this in mind, let us now write the state transition table for our **sequence generator**. This shown by the first four columns of Table I in which the first two columns indicate the present states while the next two columns indicate the corresponding next states. For instance, first state in our example is 0 = "00" which leads to the next state 1 = "01" (as shown by the gray shaded row in Table I).

Step 3

Now this state transition table is to be extended so as to include the excitation table of the flip-flop with which we desire to design our circuit. In our case, it is nothing but D flip-flop due to which we have the fifth and the sixth columns of the table representing the excitation table of D flip-flop.
For example, look at the orange shaded row in Table I in which the present and the next states 1 and 0 (respectively) result in D1 to be 0. The same row also shows the case wherein $Q_0 = {Q_0}^+ = 0 \; leads \; to \; D_0 = 0$

Video Content / Details of website for further learning (if any):

https://www.pitt.edu/~kmram/0132/lectures/sequential-circuit-design.pdf
https://www.youtube.com/watch?v=NbON135lf60

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (190 - 193)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-34

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **IV- SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS**

Date of Lecture:

Topic of Lecture: Asynchronous sequential circuits

Introduction :

In designing asynchronous sequential circuits, care must be taken to conform to certain restrictions and precautions to ensure that the circuits operate properly. The circuit must be operated in fundamental mode with only one input changing at any time and must be free of critical races. In addition, there is one more phenomenon called a hazard that may cause the circuit to malfunction.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuits

Asynchronous sequential circuits: *f*

Do not use clock pulses. The change of internal state occurs when there is a change in the input variable. *f* Their memory elements are either unclocked flip-flops or time-delay elements. *f* They often resemble combinational circuits with feedback. *f* Their synthesis is much more difficult than the synthesis of clocked synchronous sequential circuits. *f* They are used when speed of operation is important. The communication of two units, with each unit having its own independent clock, must be done with asynchronous circuits. The general structure of an asynchronous sequential circuit is as follows:

Asynchronous sequential circuits:

· Do not use pulses. The change of internal state occurs when there is a change in the input variable.

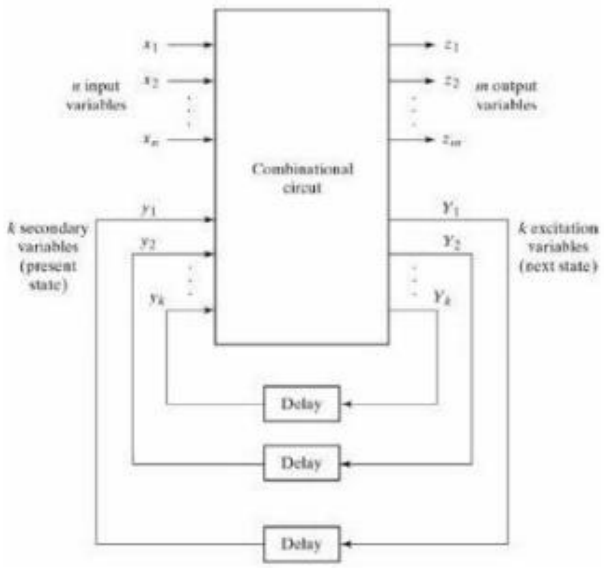
· Their memory elements are either unclocked flip flops or time delay elements.

· They often resemble combinational circuit with feedback.

· Their synthesis is much difficult than the synthesis of clocked synchronous sequential circuit.

- They are used when speed of operation is important.

The communication of two units, with each unit having its own independent clock, must be done with asynchronous circuits.



There are n input variables, m output variables, and k internal states.

The present state variables (y1 and y2) are called secondary variables. The next state variables (y1 and y2) are called excitation variables.

Fundamental mode operation assumes that the input signals change one at a time and only when the circuit is in a condition.

ANALYSIS PROCEDURE

The analysis of asynchronous sequential circuits proceeds in much the same way as that of clocked synchronous sequential circuits. From a logic diagram. Boolean expressions are written and then transferred into tabular form.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/digital_circuits/digital_circuits_sequential_circuits.htm
https://www.youtube.com/watch?v=QfIoAPio8oE

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (191)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L-35

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **IV- SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS**

Date of Lecture:

Topic of Lecture: Hazards and Races

Introduction :

- A hazard in a system is an undesirable effect caused by either a deficiency in the system or external influences.
- A race condition or race hazard is the condition of an electronics, software, or other system where the system's substantive behavior is dependent on the sequence or timing of other uncontrollable events. It becomes a bug when one or more of the possible behaviors is undesirable.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuits

Hazards

In digital logic, a hazard in a system is an undesirable effect caused by either a deficiency in the system or external influences. Logic hazards are manifestations of a problem in which changes in the input variables do not change the output correctly due to some form of delay caused by logic elements (NOT, AND, OR gates, etc.) This results in the logic not performing its function properly. The three different most common kinds of hazards are usually referred to as static, dynamic and function hazards.

Hazards are a temporary problem, as the logic circuit will eventually settle to the desired function. Therefore, in synchronous designs, it is standard practice to register the output of a circuit before it is being used in a different clock domain or routed out of the system, so that hazards do not cause any problems. If that is not the case, however, it is imperative that hazards be eliminated as they can have an effect on other connected systems.

Static hazards

A static hazard is the situation where, when one input variable changes, the output changes momentarily before stabilizing to the correct value. There are two types of static hazards:

Static-1 Hazard: the output is currently 1 and after the inputs change, the output momentarily changes to 0,1 before settling on 1

Static-0 Hazard: the output is currently 0 and after the inputs change, the output momentarily changes to 1,0 before settling on 0

In properly formed two-level AND-OR logic based on a Sum Of Products expression, there will be no static-0 hazards. Conversely, there will be no static-1 hazards in an OR-AND implementation of a Product Of Sums expression.

The most commonly used method to eliminate static hazards is to add redundant logic (consensus terms in the logic expression).

Dynamic hazards

A dynamic hazard is the possibility of an output changing more than once as a result of a single input change. Dynamic hazards often occur in larger logic circuits where there are different routes to the output (from the input). If each route has a different delay, then it quickly becomes clear that there is the potential for changing output values that differ from the required / expected output.

E.g. A logic circuit is meant to change output state from 1 to 0, but instead changes from 1 to 0 then 1 and finally rests at the correct value 0. This is a dynamic hazard.

As a rule, dynamic hazards are more complex to resolve, but note that if all static hazards have been eliminated from a circuit, then dynamic hazards cannot occur.

Functional hazards

In contrast to static and dynamic hazards, functional hazards are ones caused by a change applied to more than one input. There is no specific logical solution to eliminate them. One really reliable method is preventing inputs from changing simultaneously, which is not applicable in some cases. So, circuits should be carefully designed to have equal delays in each path.

Races:

A typical example of a race condition may occur when a logic gate combines signals that have traveled along different paths from the same source. The inputs to the gate can change at slightly different times in response to a change in the source signal. The output may, for a brief period, change to an unwanted state before settling back to the designed state. Certain systems can tolerate such glitches but if this output functions as a clock signal for further systems that contain memory, for example, the system can rapidly depart from its designed behaviour (in effect, the temporary glitch becomes a permanent glitch).

Consider, for example, a two-input AND gate fed with a logic signal A on one input and its negation, NOT A, on another input. In theory the output (A AND NOT A) should never be true. If, however, changes in the value of A take longer to propagate to the second input than the first when A changes from false to true then a brief period will ensue during which both inputs are true, and so the gate's output will also be true.[2]

Design techniques such as Karnaugh maps encourage designers to recognize and eliminate race conditions before they cause problems. Often logic redundancy can be added to eliminate some kinds of races.

As well as these problems, some logic elements can enter metastable states, which create further problems for circuit designers.

Critical and non-critical forms

A critical race condition occurs when the order in which internal variables are changed determines the eventual state that the state machine will end up in.

A non-critical race condition occurs when the order in which internal variables are changed does not determine the eventual state that the state machine will end up in.

Static, dynamic, and essential forms

A static race condition occurs when a signal and its complement are combined together.

A dynamic race condition occurs when it results in multiple transitions when only one is intended. They are due to interaction between gates. It can be eliminated by using no more than two levels of gating.

An essential race condition occurs when an input has two transitions in less than the total feedback propagation time. Sometimes they are cured using inductive delay line elements to effectively increase the time duration of an input signal.

Video Content / Details of website for further learning (if any):

https://www.geeksforgeeks.org/static-hazards-in-digital-logic/
https://www.youtube.com/watch?v=Jr8o8b5HkXI

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (422 - 425)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L-36

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **IV- SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS**

Date of Lecture:

Topic of Lecture: Hazard free combinational circuits.

Introduction :

In designing asynchronous sequential circuits, care must be taken to conform to certain restrictions and precautions to ensure that the circuits operate properly. The circuit must be operated in fundamental mode with only one input changing at any time and must be free of critical races. In addition, there is one more phenomenon called a hazard that may cause the circuit to malfunction.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuits

Hazard free combinational circuits.

Hazards are unwanted switching transients that may appear at the output of a circuit because different paths exhibit different propagation delays. Hazards occur in combinational circuits, where they may cause a temporary false output value. When they occur in asynchronous sequential circuits hazards may result in a transition to a wrong stable state.

Hazards In Combinational Circuits

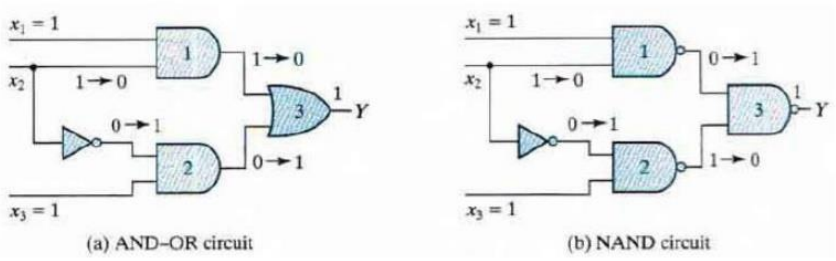


Fig: Circuits with Hazards

A hazard is a condition in which a change in a single variable produces a momentary change

in output when no change in output should occur.

Assume that all three inputs are initially equal to 1. This causes the output of gate 1 10 be 1, that of gate 2 to be 0 and that of the circuit to be 1. Now consider a change in x 2 from 1 to 0. Then the output of gate 1 changes to 0 and that of gate 2 changes to 1, leaving the output at 1. However, the output may momentarily go to 0 if the propagation delay through the inverter is taken into consideration. The delay in the inverter may cause the output of gate 1 to change to 0 before the output of gale 2 changes to 1.

The two circuits shown in Fig implement the Boolean function in sum-of-products form:

This type of implementation may cause the output to go to 0 when it should remain a 1. If however, the circuit is implemented instead in product-of-sums form namely,then the output may momentarily go to 1 when it should remain 0. The first case is referred to as static 1-hazard and the second case as static 0-hazard.
A third type of hazard, known as dynamic hazard, causes the output to change three or more times when it should change from 1 to 0 or from 0 to 1.

Fig: Types of hazards

The change in x2 from 1 to 0 moves the circuit from minterm 111 to minterm 101. The hazard exists because the change in input results in a different product term covering the two minterm.

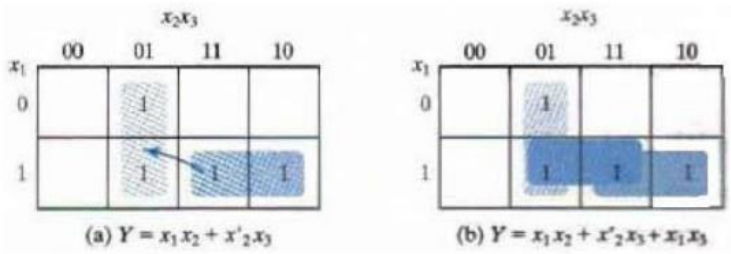


Fig: Illustrates hazard and its removal

Minterm 111 is covered by the product term implemented in gate 1 and minterm 101 is covered by the product term implemented in gate 2. The remedy for eliminating a hazard is to enclose the two minterms with another product term that overlaps both groupings. The hazard-free circuit obtained by such a configuration is shown in figure below. The extra gate in the circuit generates the product term x1x3. In general, hazard s in combinational circuits can be removed by cove ring any two minterms that may produce a hazard with a product term common to both. The removal of hazards requires the addition of redundant gates to the circuit.

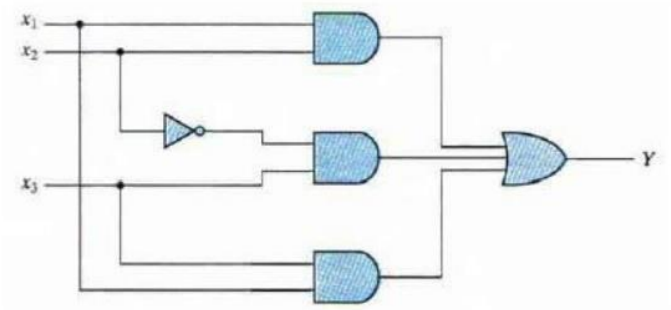


Fig: Hazard free circuit

Fig: Hazard free circuit

Video Content / Details of website for further learning (if any):

http://www.brainkart.com/article/Hazards-in-Combinational-Circuits-and-Sequential-Circuits_6769/
https://www.youtube.com/watch?v=XbRIJDeYzz4

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design", Pearson Education- V Edition,2013. Page No (422 - 425)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L-37

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **V - PROGRAMMABLE LOGIC DEVICES MEMORY AND VHDL**

Date of Lecture:

Topic of Lecture: Memories: ROM, PROM, EPROM

Introduction :

ROM stands for **Read-only Memory**. It is a type of memory that does not lose its contents when the power is turned off. For this reason, ROM is also called **non-volatile memory**.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

What is ROM?

ROM stands for **Read-only Memory**. It is a type of memory that does not lose its contents when the power is turned off. For this reason, ROM is also called **non-volatile memory**. Because ROMs are deployed in such a wide variety of applications, there are different types of ROMs suited to different applications across the industry.

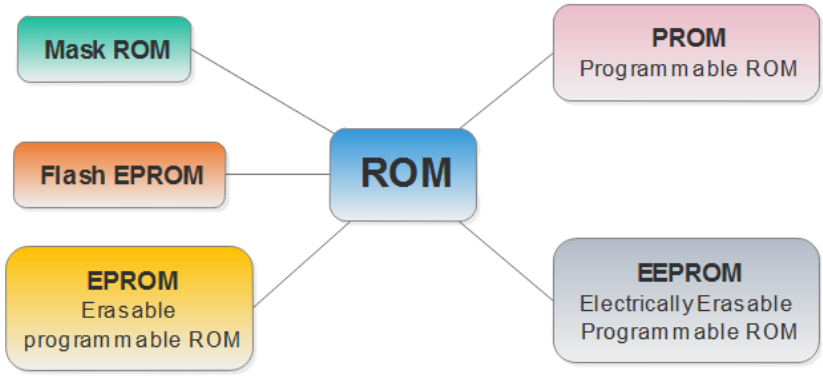
Different Types of ROM

Although all ROM basically serves the same purpose, there are a few different types commonly in use today.

Understanding the different types of ROMs will also help you understand how they're used for different applications, and which type may apply to your application.

The different types of ROM used in the industry are:

1. **PROM (Programmable ROM)**
2. **EPROM (Erasable Programmable ROM)**
3. **EEPROM (electrically erasable programmable ROM)**
4. **Flash EPROM**
5. **Mask ROM**



Types of ROM

1. PROM (programmable ROM) and OTP
PROM refers to the kind of ROM that the user can burn information into. In other words, PROM is a **user-programmable memory**.

For every bit of the PROM, there exists a fuse. PROM is programmed by blowing the fuses. If the information burned into PROM is wrong, that PROM must be discarded since its internal fuses are blown permanently. For this reason, PROM is also referred to as OTP (**One Time Programmable**).
Programming ROM also called burning ROM, requires special equipment called a ROM burner or ROM programmer.

2. EPROM (erasable programmable ROM) and UV-EPROM
EPROM was invented to **allow making changes** in the contents of PROM after it is burned. In EPROM, one can program the memory chip and erase it thousands of times. This is especially necessary during the development of the prototype of a **microprocessor-based project**.

A widely used EPROM is called **UV-EPROM**, where UV stands for ultraviolet. The only problem with UV-EPROM is that erasing its contents can take up to 20 minutes.
All UV-EPROM chips have a window through which the programmer can shine ultraviolet (UV) radiation to erase the chip's contents. For this reason, EPROM is also referred to as **UV-erasable EPROM** or simply UV-EPROM.

Programming a UV-EPROM

To program a UV-EPROM chip, the following steps must be taken:

1. Its contents must be erased. To erase a chip, remove it from its socket on the system board and place it in EPROM erasure equipment to expose it to UV radiation for 5-20 minutes.
2. Program the chip. To program a UV-EPROM chip, place it in the ROM burner (programmer). To burn code or data into EPROM, the ROM burner uses 12.5 volts or higher, depending on the EPROM type. This voltage is referred to as Vpp in the UV-EPROM datasheet.
3. Place the chip back into its socket on the system board.
   As can be seen from the above steps, not only is there an EPROM programmer (burner), but there is also separate EPROM erasure equipment.

The main problem, and indeed the major disadvantage of UV-EPROM, is that it cannot be erased and programmed while it is in the system board. To provide a solution to this problem,

EEPROM was invented.

3. EEPROM (electrically erasable programmable ROM)
EEPROM has several advantages over EPROM, such as the fact that its method of erasure is electrical and therefore instant as opposed to the 20-minute erasure time required for UV-EPROM.

In addition, in EEPROM one **can select which byte to be erased**, in contrast to UV-EPROM, in which the entire contents of ROM are erased.
However, the main advantage of EEPROM is that one can program and erase its contents while it is still in the system board. It does not require the physical removal of the memory chip from its socket. In other words, unlike UV-EPROM, EEPROM does not require an external erasure and programming device.
To utilize EEPROM fully, the designer must incorporate the circuitry to program the EEPROM into the system board. In general, the cost per bit for EEPROM is much higher than for UV-EPROM.

Video Content / Details of website for further learning (if any):

https://studyelectrical.com/2017/06/different-types-of-rom-prom-eprom.html
https://www.youtube.com/watch?v=U6i8Xmi0Y20

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (300 - 307)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L-38

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **V - PROGRAMMABLE LOGIC DEVICES MEMORY AND VHDL**

Date of Lecture:

Topic of Lecture: PLA, PLD

Introduction :

- Earlier, the designing of logic circuits can be done using SSI (small scale integration) components like logic gates, multiplexers, de-multiplexers, FFs, etc. But, now a PLD can replace all these SSI components. So this is the reason to decrease the SSI industry compared with PLD, and these are used in several applications.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

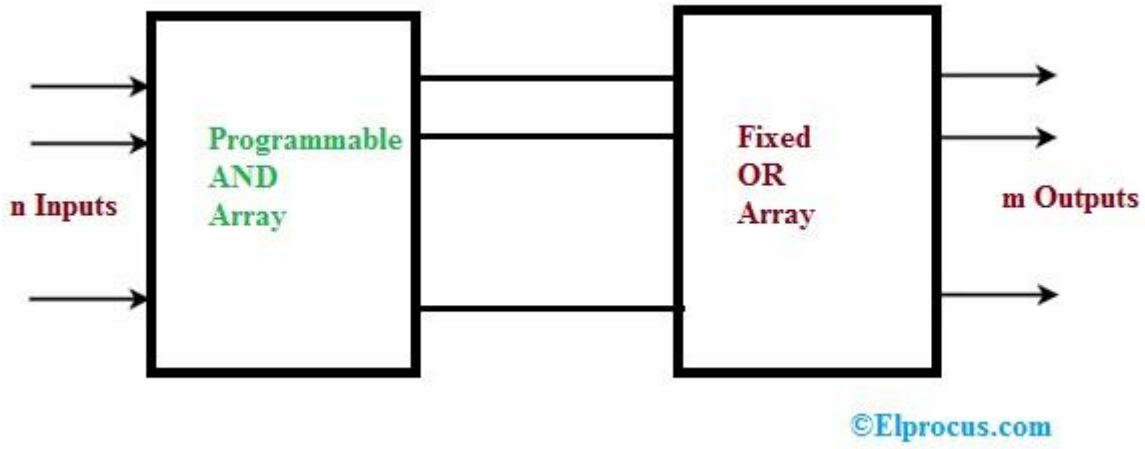
What are PAL and PLA, Design and Differences?

Earlier, the designing of **logic circuits** can be done using **SSI (small scale integration)** components like logic gates, multiplexers, de-multiplexers, FFs, etc. But, now a PLD can replace all these SSI components. So this is the reason to decrease the SSI industry compared with PLD, and these are used in several applications. The **programmable logic device or PLD** is one kind of chip used to implement the logic circuit. It includes a set of logic circuit elements that can be modified in several ways. A PLD is looked like a black box that consists of programmable switches as well as logic gates. The main function of the switches is to let the logic gates within the PLD to be associated mutually to execute logic circuits. PLDs are classified into different types such as SPLD-simple PLD (**PLA & PAL**), **CPLD-complex PLD**, **FPGAs-field programmable gate arrays**. This article discusses what is a PAL and PLA, design and their differences.

What are PAL and PLA?

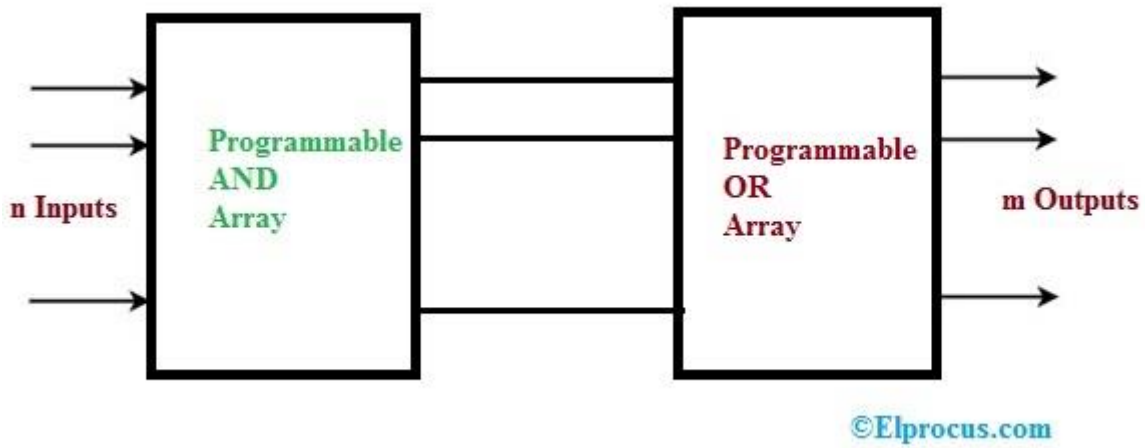
Both **Programmable Array Logic** and **Programmable Logic Array** are types of PLDs (programmable logic devices), and these are mainly used for designing combination logic mutually by sequential logic. The main difference among these two is that PAL can be designed with a collection of AND gates and fixed collection of OR gates whereas PLA can

be designed with a programmable array of AND although a fixed collection of OR gate. A programmable logic device offers a simple as well as flexible logic circuit designing.



Programmable Array Logic

Previous to programmable logic devices, the **combinational logic circuits** can be designed with multiplexers, and these circuits were rigid as well as compound, then PLDs are developed. The initial programmable logic device was ROM, but it was not successful due to the hardware wastage issues as well as exponential growth enhancement in the every hardware application. To overcome this issue, PAL and PLA were used. These two are programmable, and efficiently uses the hardware.



Programmable Logic Array

Design of Programmable Array Logic (PAL)

The **definition of term PAL or Programmable Array Logic** is one type of PLD which is known as Programmable Logic Device circuit, and working of this PAL is the same as the PLA. The designing of the programmable array logic can be done with fixed OR gates as well as programmable AND gates. By using this we can implement two easy functions wherever the associates AND gates with each OR gate denote the highest number of product conditions that can be produced in the form of **SOP (sum of product)** of an exact function.

As the logic gates like AND is connected continually toward the OR gates, and that indicates that the produced product term is not distributed with the output functions. The major notion behind PLD development is to fabricate a compound Boolean logic onto a single chip

by removing the defective wiring, avoiding the logic design, as well as decreasing the consumption of power.

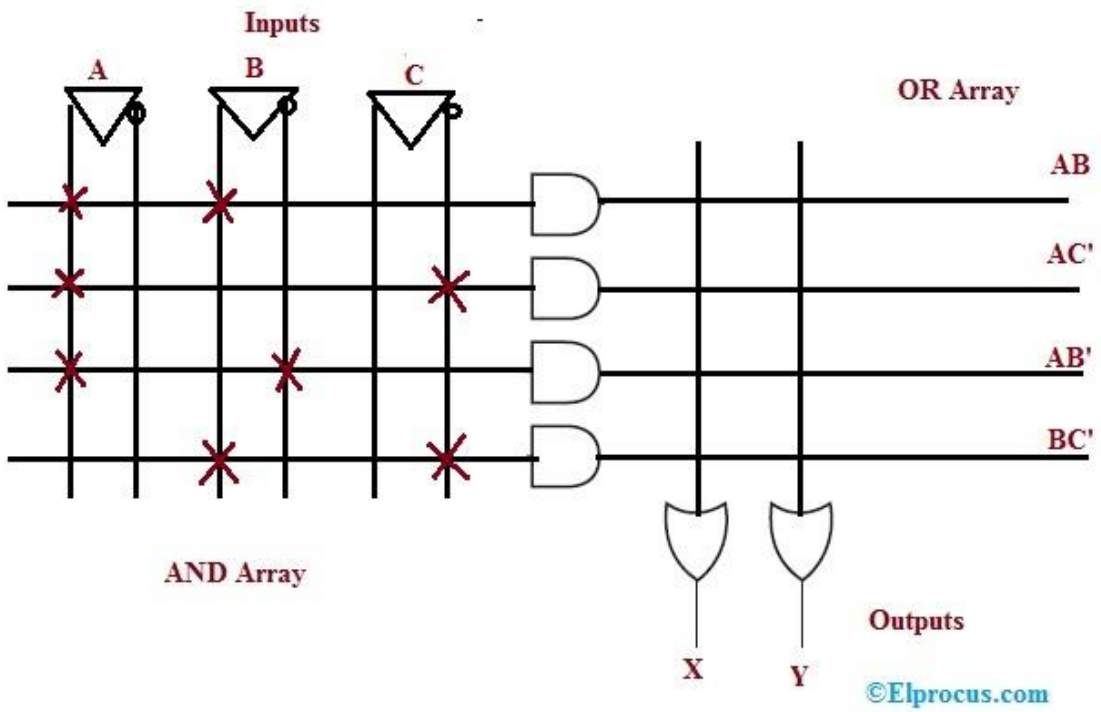
Example of PAL

Implement the following **Boolean expression** with the help of **programmable array logic (PAL)**

X =AB + AC′

Y= AB′ + BC′

The above given two Boolean functions are in the form of **SOP (sum of products)**. The product terms present in the Boolean expressions are X & Y, and one product term that is AC′ is common in every equation. So, the total required logic gates for generating the above two equations is AND gates-4 OR programmable gates-2. The equivalent PAL logic diagram is shown below.



PAL Logic Circuit

The AND gates which are programmable have the right of entry for normal as well as complemented variable inputs. In the above logic diagram, the available inputs for each AND gate are A, A′, B, B′, C, C′. So, in order to generate a single product term with every AND gate, the program is required.
All the product terms are obtainable at the inputs of an each OR gate. Here, the programmable connections on the logic gate can be denoted with the symbol 'X'.

Here, the OR gate inputs are fixed. Thus, the required product terms are associated with each OR gate inputs. As a result, these gates will generate particular Boolean equations. The **'.'** The symbol represents permanent connections.

Design of Programmable Logic Array (PLA)

The definition of term PLA presents the Boolean function in the form of a sum of product (SOP). The designing of this programmable logic array can be done using the logic gates like AND, OR, and NOT by fabricating on the chip, that makes every input as well as its compliment obtainable toward every AND gate.

An every AND gate's output is connected to the every OR gate. Finally, the output of the OR gate generates the output of the chip. Thus, this is how an appropriate association is finished to use the expressions of the sum of the product. In the programmable logic array, the connections of logic gates like AND & OR are programmable. PLA is expensive and difficult to compare with PAL. The PAL uses two dissimilar developed methods can be used for a programmable logic array for enhancing the effortlessness of programming. In this kind of method, every connection can be done using a fuse on each intersection point wherever the unnecessary connections can be detached by the fuse blowing. The final technique engages the making of connection while the process of the fabrication using the suitable cover offered for the precise interconnection model.

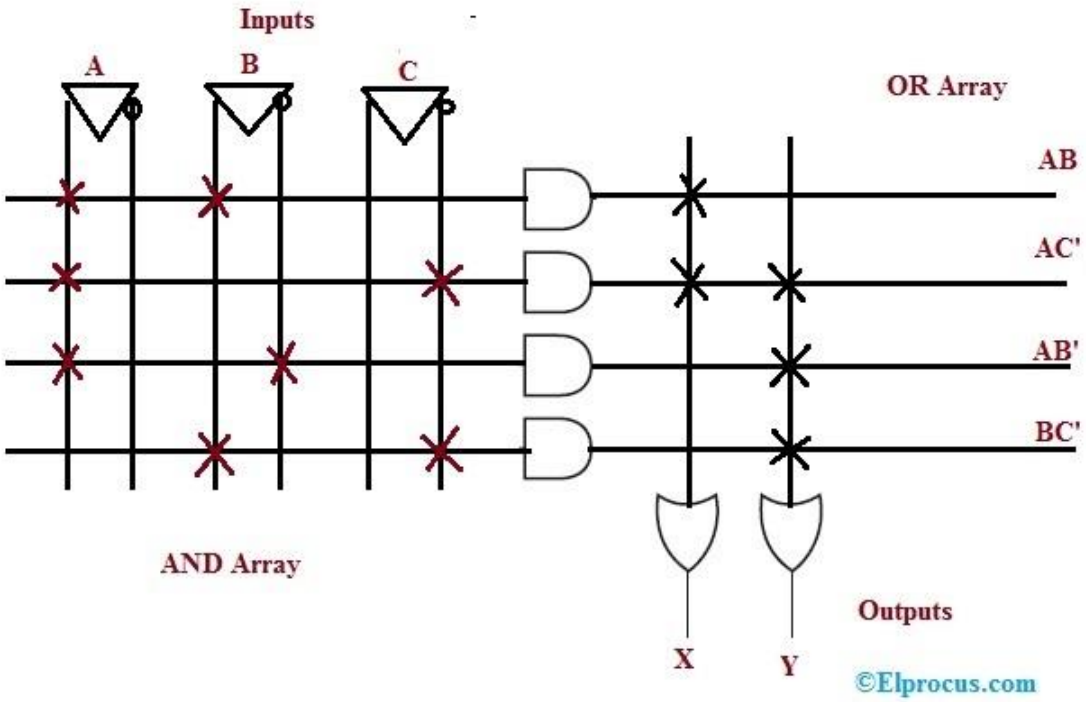
Example of PLA

Implement the following Boolean expression with the help of programmable logic array (PLA)

X = AB + AC'

Y = AB' +BC + AC'

The above given two Boolean functions are in the form of SOP (sum of products). The product terms present in the Boolean expressions are X & Y, and one product term that is AC' is common in every equation. So, the total required logic gates for generating the above two equations is AND gates-4, OR programmable OR gates-2. The equivalent PLA logic diagram is shown below.



PLA Logic Circuit

The AND gates which are programmable have the right of entry for normal as well as complemented variable inputs. In the above logic diagram, the available inputs for each AND gate are A, A', B, B', C, C'. So, in order to generate a single product term with every AND gate, the program is required.
All the product terms are obtainable at the inputs of each OR gate. Here, the programmable connections on the logic gate can be denoted with the symbol 'X'.

Video Content / Details of website for further learning (if any):

https://www.elprocus.com/what-are-pal-and-pla-design-and-differences/
https://www.youtube.com/watch?v=jrQ1YYgiOTo

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (321-325)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-39

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **V - PROGRAMMABLE LOGIC DEVICES MEMORY AND VHDL**

Date of Lecture:

Topic of Lecture: FPGA – VHDL Programming

Introduction :

- VHDL is a horrible acronym. It stands for VHSIC Hardware Description Language. An acronym inside an acronym, awesome! VHSIC stands for Very High Speed Integrated Circuit.
- A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term "field-programmable".

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

1. What is FPGA Programming?

As mentioned in the previous FPGA post, FPGAs are nothing, but reconfigurable logic blocks (logic gates, memory elements, DSP components, etc.) and interconnects. FPGA programming is actually (re)configuring FPGAs using Hardware Description Language (Verilog/VHDL) to connect these logic blocks and interconnects in a way that it can perform a specific functionality (adders, multipliers, processors, filters, dividers, etc.).

2. What is the FPGA programming language?

FPGA programming language is commonly called Hardware Description Language because it is actually used to describe or design hardware. The two major Hardware Description Languages are Verilog HDL and VHDL.

Verilog = "Verification" + "Logic", which originally created by P. Moorby, P. Goel, C.-L. Huang, and D. Warmke in 1984 to model gates and perform simulation in a logic simulator. Verilog was acquired by Cadence in 1990 and became IEEE Standard 1364 in 1995. Now, Verilog is commonly used for designing and verification of digital circuits and analog or mixed signal circuits as well.

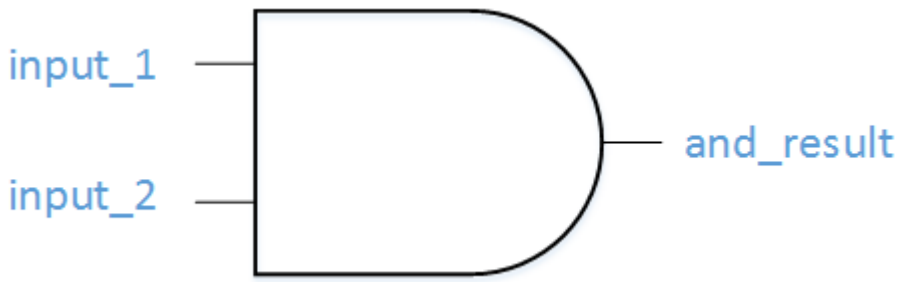
VHDL is VHSIC (Very High Speed Integrated Circuits) Hardware Description Language which was initially developed by US Defense Department to research and develop very high speed integrated circuits in the 1980s. VHDL became IEEE Standard 1076 in 1987. Now, VHDL is mostly used to model digital and analog mixed signal circuits such as FPGAs and ASICs.

Verilog HDL is similar to C programming language such as case-sensitive, keywords, operators, etc. VHDL is similar to Ada programming language since it was initially built based on Ada. The difference between Verilog and VHDL is presented here. Although there are similarities in syntax between HDLs (Verilog, VHDL) and software programming languages (C, Ada), FPGA programming is completely different from Software programming.

Introduction to VHDL

VHDL is a horrible acronym. It stands for VHSIC Hardware Description Language. An acronym inside an acronym, awesome! VHSIC stands for Very High Speed Integrated Circuit. Therefore, VHDL expanded is Very High Speed Integrated Circuit Hardware Description Language. PHEW that's a mouthful. VHDL is one of the two languages used by education and business to design FPGAs and ASICs. You might first benefit from an introduction to FPGAs and ASICs if you are unfamiliar with these fascinating pieces of circuitry. VHDL and Verilog are the two languages digital designers use to describe their circuits, and they are different by design than your traditional software languages such as C and Java.

For the example below, we will be creating a VHDL file that *describes* an And Gate. As a refresher, a simple And Gate has two inputs and one output. The output is equal to 1 only when both of the inputs are equal to 1. Below is a picture of the And Gate that we will be describing with VHDL.



An And Gate

Let's get to it! The fundamental unit of VHDL is called a signal. For now let's assume that a signal can be either a 0 or a 1 (there are actually other possibilities, but we will get to that). Here is some basic VHDL logic:

```
1    signal and_gate : std_logic;

2    and_gate <= input_1 and input_2;
```

The first line of code defines a signal of type std_logic and it is called and_gate. Std_logic is the type that is most commonly used to define signals, but there are others that you will learn about. This code will generate an AND gate with a single output (and_gate) and 2 inputs (input_1 and input_2). The keyword "and" is reserved in VHDL. The <= operator is known as the assignment operator. When you verbally parse the code above, you can say out loud, "The signal and_gate GETS input_1 and-ed with input_2."

Now you may be asking yourself where input_1 and input_2 come from. Well as their name implies they are inputs to this file, so you need to tell the tools about them. Inputs and outputs to a file are defined in an entity. An entity contains a port that defines all inputs and outputs to a file. Let's create a simple entity:

```
1   entity example_and is

2     port (

3       input_1    : in  std_logic;

4       input_2    : in  std_logic;

5       and_result : out std_logic

6     );

7   end example_and;
```

This is your basic entity. It defines an entity called example_and and 3 signals, 2 inputs and 1 output, all of which are of type std_logic. One other VHDL keyword is needed to make this complete and that is architecture. An architecture is used to describe the functionality of a particular entity. Think of it a thesis paper: the entity is the table of contents and the architecture is the content. Let's create an architecture for this entity:

```
1   architecture rtl of example_and is

2     signal and_gate : std_logic;

3   begin

4     and_gate <= input_1 and input_2;

5     and_result <= and_gate;

6   end rtl;
```

The above code defines an architecture called rtl of entity example_and. All signals that are used by the architecture must be defined between the "is" and the "begin" keywords. The actual architecture logic comes between the "begin" and the "end" keywords. You're almost done with this file. One last thing you need to tell the tools is which library to use. A library defines how certain keywords behave in your file. For now, just take it for granted that you need to have these 2 lines at the top of your file:

```
1   library ieee;

2   use ieee.std_logic_1164.all;
```

Congratulations! You have created your first VHDL file. You can see the completed file here:

```
1     library ieee;

2     use ieee.std_logic_1164.all;

3
```

```
4   entity example_and is
5     port (
6       input_1   : in  std_logic;
7       input_2   : in  std_logic;
8       and_result : out std_logic
9       );
10  end example_and;
11
12  architecture rtl of example_and is
13    signal and_gate : std_logic;
14  begin
15    and_gate   <= input_1 and input_2;
16    and_result <= and_gate;
17  end rtl;
```

Does it seem like you had to write a lot of code just to create a stupid and gate? First of all, and gates aren't stupid. Secondly, you are correct; VHDL is a very verbose language. Get used to the fact that doing something that was very easy in software will take you significantly longer in an HDL such as Verilog or VHDL.

Video Content / Details of website for further learning (if any):

https://www.nandland.com/vhdl/tutorials/tutorial-introduction-to-vhdl-for-beginners.html
https://www.youtube.com/watch?v=mwJ3uMWvJX0

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (381 - 390)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L-40

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **V - PROGRAMMABLE LOGIC DEVICES MEMORY AND VHDL**

Date of Lecture:

Topic of Lecture: RTL Design – Combinational Logic

Introduction :

- Logic is the process of decision making based on some conditions. Combinational Logic refers to the decision making combining one or many conditions. The entire decision making process when divided into fundamental decision making blocks, they become the basic logic gates which are And, Or, Not, Nor, Nand, Xor, X-Nor .

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

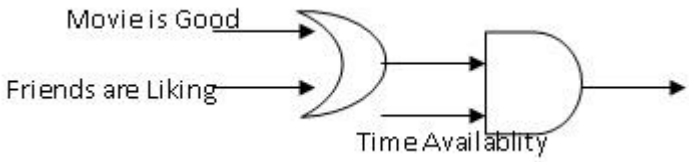
Combinational Logic

Logic is the process of decision making based on some conditions. Combinational Logic refers to the decision making combining one or many conditions. The entire decision making process when divided into fundamental decision making blocks, they become the basic logic gates which are And, Or, Not, Nor, Nand, Xor, X-Nor . All these decision are made based by associating Present/Absent or Yes / No or True/False to the individual conditions.

GATE	Decision
NOT	Inversion . (i.e. Condition Not present)
AND	All the conditions are present
OR	Any of the conditions are present
XNOR	Equivality of Conditions
XOR	The conditions are Mutually Exclusive
NOR	Not even a single condition exists
NAND	Not all condition exists

Essential concepts and detailed interview guide.

From this it is clear that any decision making , can be constructed from these basic decision making elements. For e.g. If you want to go for a movie, the decision will be made like follows. (Movie is Good or Friends with you want to go) AND (You are free time to spend for movie).



Here again the Movie is Good , is a decision collective of many factors, like The Star Cast is Good OR Screen Play of the movie is Good OR Story line of the movie is Good. Like this each condition can be further evaluated. This is a classical scenario of a decision making process. This is extensively used in the Digital circuit design as all the logical decision made in the Digital Domain are of the similar nature. E.g. If you want to create a logic to send some information to a receiver, it will be like The Information is available with you AND the Receiver is ready to accept the information. If this condition is met then you start sending the information.

Need for Sequential Logic

The above scenario of decision making, takes into account the present conditions existing only. This cannot be sufficient always to make a correct decision.
For e.g. If we consider the decision of a movie described above, You may want to go to a movie which you have not seen. So the decision will be made based on the present existing factors as described above and from your PAST, that is you have not watched the movie. The fact that you have not watched the movie can be considered as a state.
Which means you make a decision collectively based on the present prevailing conditions and the state you are existing.

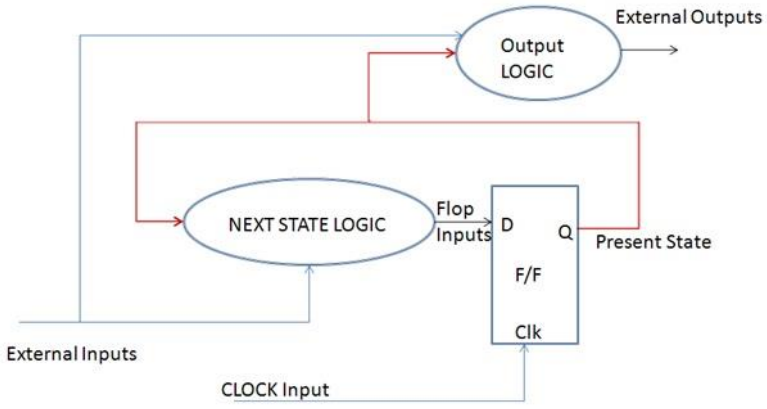
Hence there is a necessity to store these state information as it should be made available for the future decision making. Here comes Flip-Flops which are basic 1 bit storage elements for this purpose.

Finite State Machine

From the example above, It is clear that the present state will be a factor for deciding the next state along with the present inputs. The same can also be told as the present state was derived from the past state with the present inputs. Both are same… it is just a play of words!!
So…the question is what is a state and how am I going to represent it ??

The answer is simple. States are different stages in the decision making process flow. In digital system, any information is represented or stored as 0s and 1s. So the states are also represented in a binary code in the form of 0s and 1s.

These 0s and 1s are in form of a code where each bit of 0 or 1 is coming from a flip flop.
This entire information can be summarized in the below diagram.



General Block Diagram of a State Machine

The F/F acts as a storage element to store the state information.

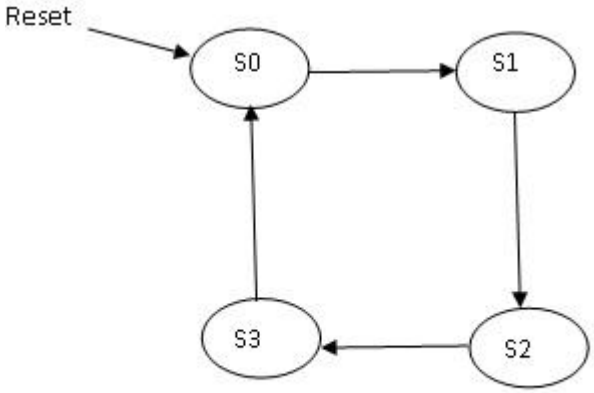
There is a decision making logic called the Next state Logic. It is to be noted that the inputs to the Next State Logic (NSL) are the present state (shown in brown) and the external input(shown in blue). Considering both, the decision for the next state is made. The decision made is latched by the flop on the clock edge.

So in our example, the "Movie not seen in the past" comes from the flop and the other inputs like the "Movie Reviews are good, etc ,etc" comes from the external inputs. Considering both the factors together the NSL decides whether to go to movie or not and this information is latched by the Flop on the clock edge. If the decision for watching the movie is taken the NSL generates '1'.. which means the flop input is now '1' and the state is updated as "Movie Watched"…. So this will be a driving factor in the Next decision making process.

State Machine : A state machine is a digital system working on a predefined finite states. So they are also referred as Finite State Machine or FSM in short.
Case Study : Let us understand the need for a state machine and what it is based on a case study.

Let us consider the scenario of a 2 bit Up counter.
This machine is a counter which can count in the ascending (Up). The machine is a 2 bit machine. So it will need 2 flops (1 Flop per bit).
[Remember a machine having n flops can have 2^n states (each flop can be 0 or 1 leading to 2^n permutations)].
On reset the machine is at "00". After reset the machine starts counting up on each clock edge. It will count as 00, 01, 10, 11 and after this again 00. So the states are finite and hence the name FSM. After this the states keep repeating in the same sequence. If I generalise the states as S0 , S1 , S2, S3 instead of 00, 01 ,10 ,11 … then the machine will be repeating the states S0, S1 , S2, S3 ,S0 ,S1,…… .

This can be diagrammatically represented in a diagram called the State Diagram. In the state diagram each state is represented by a bubble and the transition from one state to other is represented by an arrow. (The direction of the arrow is important as it shows from which state to which state the transition is happening. This is important because the Transition from S0 to S1 is totally different from the transition from S1 to S0 state).



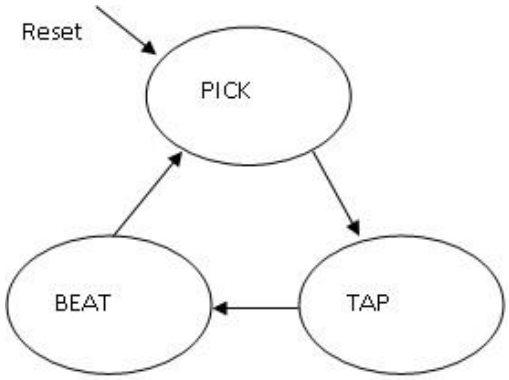
Please Note : Upon Reset the state machine is initialised at S0. This is very important. Upon reset any state machine has to be initialised to a Known Valid State.
Can there be an Unknown State ??
Ok Let us understand this clearly. There are 2 reasons why we have to define the reset state of the machine.

Reason1 :

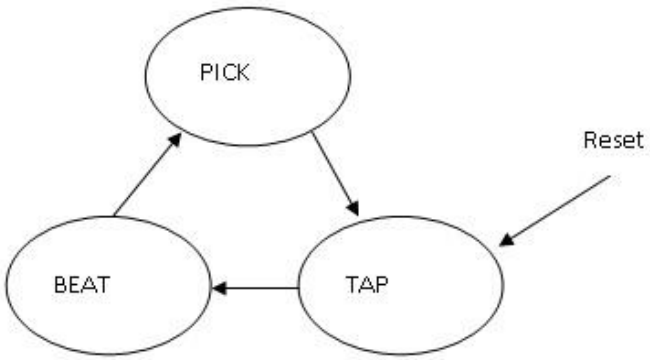
Let us take the example of a machine which will beat the EGGs and give us. Upon reset, the machine will pick up an egg, then tap it (break it) and take the yoke out of it in a bowl and then it will beat the yoke in the bowl. This is the sequence the machine is going to operate. (states are indicated in bold).

Now if upon reset I leave the machine in the tapping state instead of the Picking up the egg, Even without picking up an egg, it will try to tap it and break it. Two things can happen here, either the machine will wait for Egg in this state (it has not picked the egg, so it will wait for the egg) and it will hang here or it will miss the sequence and something that is not expected out of it will happen.

So the reset state of a state machine is very important and must be defined clearly to a known valid state.



This is the correct sequence and the reset state of the machine is defined properly.



This is the in-correct sequence and the reset state of the machine is defined WRONGLY. The machine will fail to perform the expected behaviour .

Reason 2:

Now let us take the example of a counter which needs to count 0 to 5, like 0,1,2,3,4,5,0,1,2,3,…. Upon reset the machine should be set to state 0(known state). What is a unknown state then ?? Ok …..From the implementation point of view …there are six states in the machine and we need 3 bits for implementation. But a machine which has n bits will have 2^n states which means here there are 8 states. But we are designing the machine to operate only in 6 states(0 to 5). Which means there are 2 unused states (6 and 7) or undefined states for the machine. The machine will not know what to do in this state as we have not designed it for these states. So it will hang or preform some totally unintended operation.

There is a state machine which decides when to fire the missile. Suppose the machine goes to undefined state and fires the missile in the own base itself!! …It is comedy to read it here but in real life it will be a tragedy!!!!!

So a good designer should always provide a means to initialize the machine in known state and if there are unused state and it is enetered because of some error conditions…there should be a way to reach the initial state. Else it will be disastrous!!.

State Transitions:

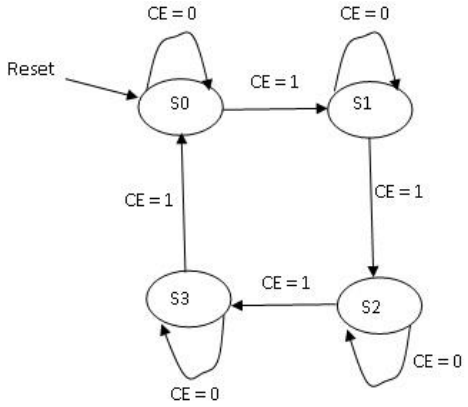
The change from any state to its corresponding next state is called a state transition. The

transition from one state to another state can be a conditional transition or an unconditional transition.

What does this mean…?
Let us understand it with an example. Let us take the scenario of the same 2bit counter, The intent is to count only when the Count_Enable is high, Else the counter should retain its present state.

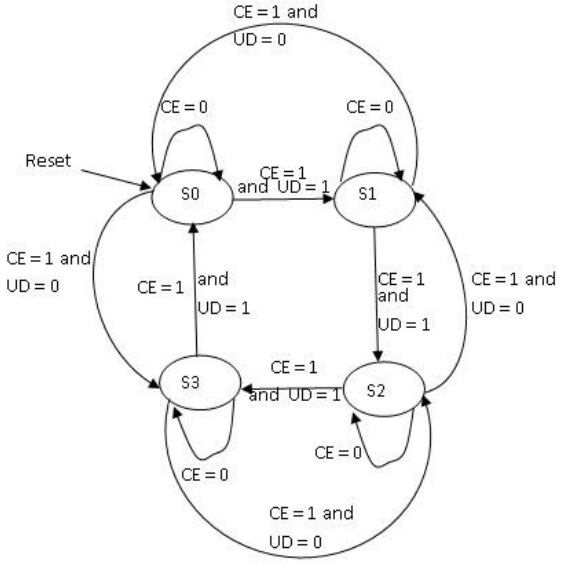
Now how to represent this scenario in the state Diagram…..Now from the given scenario it is very clear that the transition from the state is based on a condition [Count_Enable = 1] , these type of transitions are called Conditional Tranisitions in a State Machine. The Transitions which we saw earlier are Unconditional Transitions. The Unconditional Transition can occur on every clock edge, but conditional transition can happen on the edge of the clock where the condition is met.

Let us see , how this diagram looks. (the Count_Enable signale is shown as CE in the state diagram below)



Now let us complicate the situation and see. In the above scenario there is only one condition for every transition. But there can be more than a single condition to decide the transition from one state to another. The counter is a 2 bit Up-Down counter with Enable. When the counter is Enabled it should count Up (Ascending order) or Down (descending order) depending upon the UD signal. If UD = 1, it should count up and if UD = 0 it should Count Down.

So the state diagram has to accommodate these conditions also….



From the state diagram it is clear that when the Count Enable (CE) is low, there is no transition in state irrespective of the value of Up-Down (UD). Hence the value of UD is NOT

indicated in these cases.
Whereas the value of UD is considered to Count Up / Count Down when CE = 1. Hence for the transitions when happening CE = '1' is indicated with the value of UD.

Video Content / Details of website for further learning (if any):

https://www.vlsisystemdesign.com/combinational-logic/
https://www.youtube.com/watch?v=m5rEKAqHyKo

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (351 - 363)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-41

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **V - PROGRAMMABLE LOGIC DEVICES MEMORY AND VHDL**

Date of Lecture:

Topic of Lecture: Combinational Logic Types – Operators – Packages

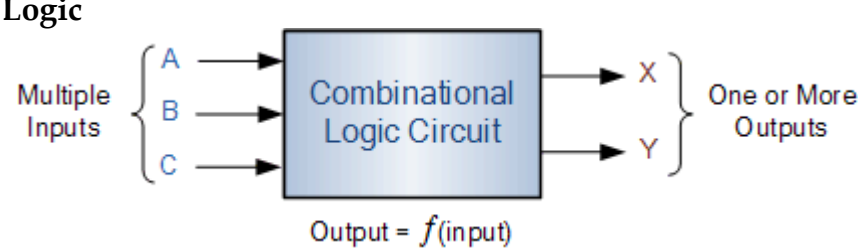
Introduction :

- Unlike Sequential Logic Circuits whose outputs are dependant on both their present inputs and their previous output state giving them some form of *Memory*. The outputs of **Combinational Logic Circuits** are only determined by the logical function of their current input state, logic "0" or logic "1", at any given instant in time.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

Combinational Logic



Combinational Logic Circuits are made up from basic logic NAND, NOR or NOT gates that are "combined" or connected together to produce more complicated switching circuits. These logic gates are the building blocks of combinational logic circuits. An example of a combinational circuit is a decoder, which converts the binary code data present at its input into a number of different output lines, one at a time producing an equivalent decimal code at its output.

Combinational logic circuits can be very simple or very complicated and any combinational circuit can be implemented with only NAND and NOR gates as these are classed as "universal" gates.

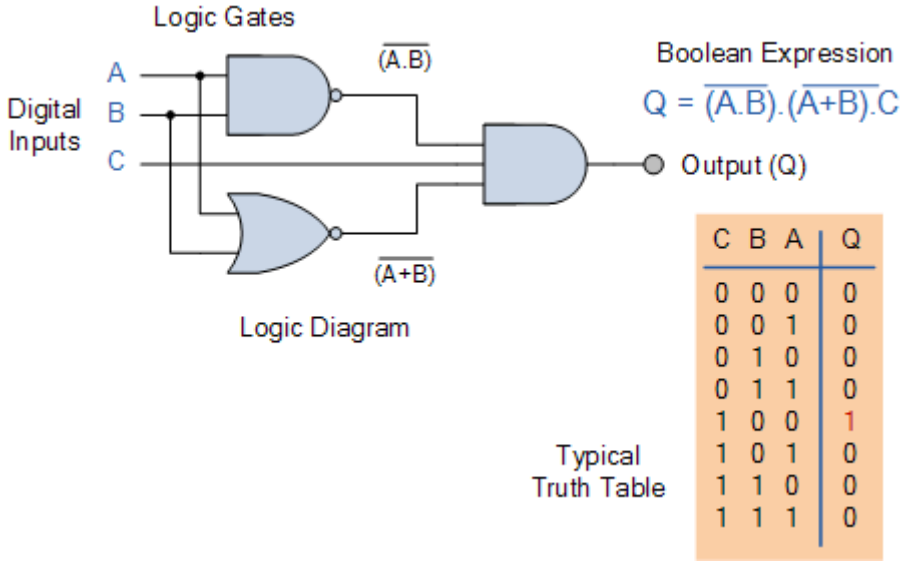
The three main ways of specifying the function of a combinational logic circuit are:

- 1. Boolean Algebra – This forms the algebraic expression showing the operation of the logic circuit for each input variable either True or False that results in a logic "1" output.
- 2. Truth Table – A truth table defines the function of a logic gate by providing a concise list that shows all the output states in tabular form for each possible

combination of input variable that the gate could encounter.

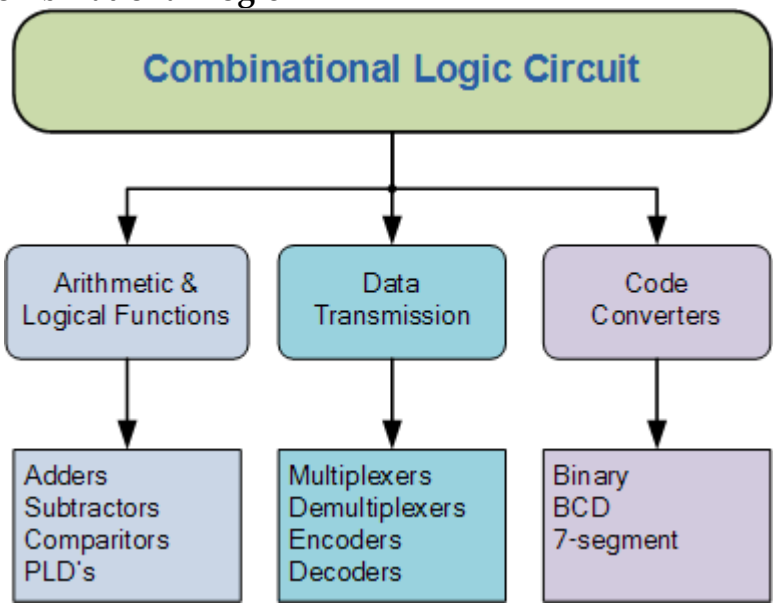
- 3. Logic Diagram – This is a graphical representation of a logic circuit that shows the wiring and connections of each individual logic gate, represented by a specific graphical symbol, that implements the logic circuit.

and all three of these logic circuit representations are shown below.



As combinational logic circuits are made up from individual logic gates only, they can also be considered as "decision making circuits" and combinational logic is about combining logic gates together to process two or more signals in order to produce at least one output signal according to the logical function of each logic gate. Common combinational circuits made up from individual logic gates that carry out a desired application include *Multiplexers*, *De-multiplexers*, *Encoders*, *Decoders*, *Full* and *Half Adders* etc.

Classification of Combinational Logic



One of the most common uses of combinational logic is in Multiplexer and De-multiplexer type circuits. Here, multiple inputs or outputs are connected to a common signal line and logic gates are used to decode an address to select a single data input or output switch.

A multiplexer consist of two separate components, a logic decoder and some solid state switches, but before we can discuss multiplexers, decoders and de-multiplexers in more detail we first need to understand how these devices use these "solid state switches" in their design.

Solid State Switches

Standard TTL logic devices made up from Transistors can only pass signal currents in one direction only making them "uni-directional" devices and poor imitations of conventional electro-mechanical switches or relays. However, some CMOS switching devices made up from FET's act as near perfect "bi-directional" switches making them ideal for use as solid state switches.

Solid state switches come in a variety of different types and ratings, and there are many different applications for using solid state switches. They can basically be sub-divided into 3 different main groups for switching applications and in this combinational logic section we will only look at the Analogue type of switch but the principal is the same for all types including digital.

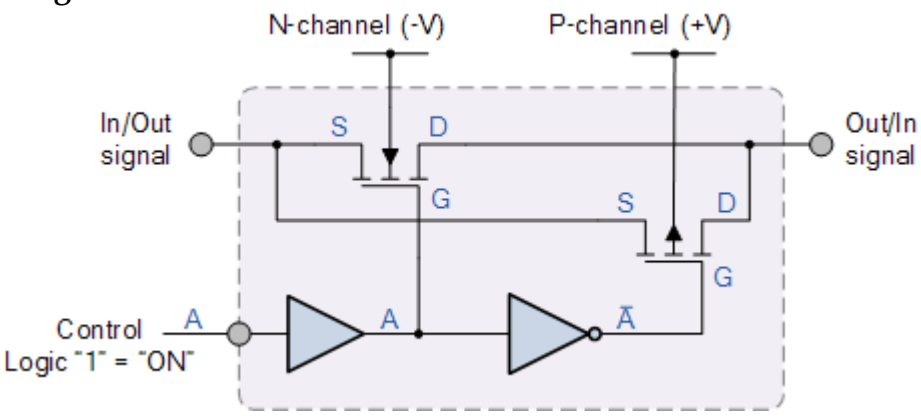
Solid State Switch Applications

- Analogue Switches – Used in Data Switching and Communications, Video and Audio Signal Switching, Instrumentation and Process Control Circuits …etc.
- Digital Switches – High Speed Data Transmission, Switching and Signal Routing, Ethernet, LAN's, USB and Serial Transmissions …etc.
- Power Switches – Power Supplies and General "Standby Power" Switching Applications, Switching of Larger Voltages and Currents …etc.

Analogue Bilateral Switches

Analogue or "Analog" switches are those types that are used to switch data or signal currents when they are in their "ON" state and block them when they are in their "OFF" state. The rapid switching between the "ON" and the "OFF" state is usually controlled by a digital signal applied to the control gate of the switch. An ideal analogue switch has zero resistance when "ON" (or closed), and infinite resistance when "OFF" (or open) and switches with R_{ON} values of less than 1Ω are commonly available.

Solid State Analogue Switch

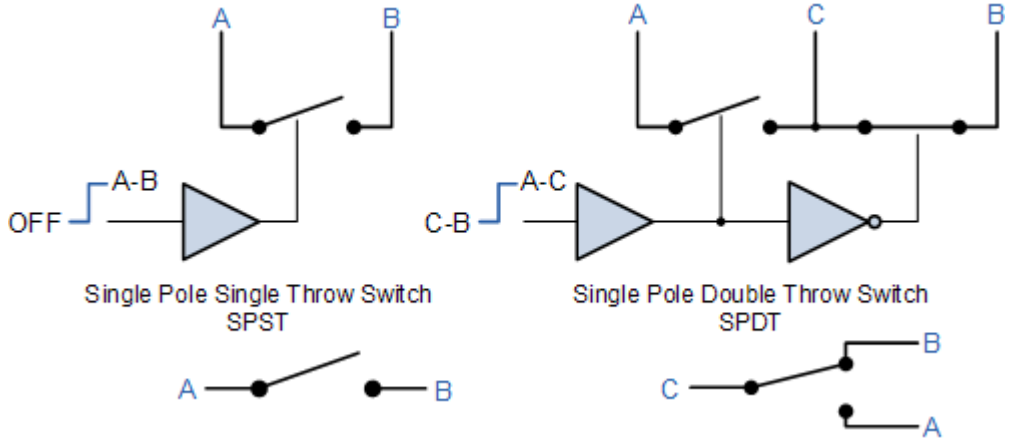


By connecting an N-channel MOSFET in parallel with a P-channel MOSFET allows signals to pass in either direction making it a "Bi-directional" switch and as to whether the N-channel or the P-channel device carries more signal current will depend upon the ratio between the input to the output voltage. The two MOSFET's are switched "ON" or "OFF" by two internal non-inverting and inverting amplifiers.

Contact Types

Just like mechanical switches, analogue switches come in a variety of forms or contact types, depending on the number of "poles" and "throws" they offer. Thus, terms such as "SPST" (single-pole single throw) and "SPDT" (single-pole double-throw) also apply to solid state analogue switches with "make-before-break" and "break-before-make" configurations available.

Analogue Switch Types



Individual analogue switches can be grouped together into standard IC packages to form devices with multiple switching configurations of SPST (single-pole single-throw) and SPDT (single-pole double-throw) as well as multi channel multiplexers.

The most common and simplest analogue switch in a single IC package is the 74HC4066 which has 4 independent bi-directional "ON/OFF" Switches within a single package but the most widely used variants of the CMOS analogue switch are those described as "Multi-way Bilateral Switches" otherwise known as the "Multiplexer" and "De-multiplexer" IC´s and these are discussed in the next tutorial.

Video Content / Details of website for further learning (if any):

https://www.electronics-tutorials.ws/combination/comb_1.html
https://www.youtube.com/watch?v=sUutDs7FFeA

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (351 - 363)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L-42

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **V - PROGRAMMABLE LOGIC DEVICES MEMORY AND VHDL**

Date of Lecture:

Topic of Lecture: Sequential Circuits – Sub Programs

Introduction :

- A **Sequential circuit** combinational logic **circuit** that consists of inputs variable (X), logic gates (Computational **circuit**), and output variable (Z). Combinational **circuit** produces an output based on input variable only, but **Sequential circuit** produces an output based on current input and previous input variables.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

Introduction of Sequential Circuits

A Sequential circuit combinational logic circuit that consists of inputs variable (X), logic gates (Computational circuit), and output variable (Z).

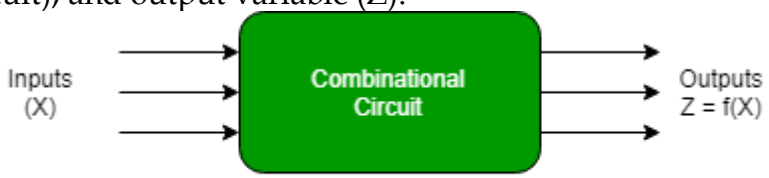


Figure: Combinational Circuits

Combinational circuit produces an output based on input variable only, but Sequential circuit produces an output based on current input and previous input variables. That means sequential circuits include memory elements which are capable of storing binary information. That binary information defines the state of the sequential circuit at that time. A latch capable of storing one bit of information.

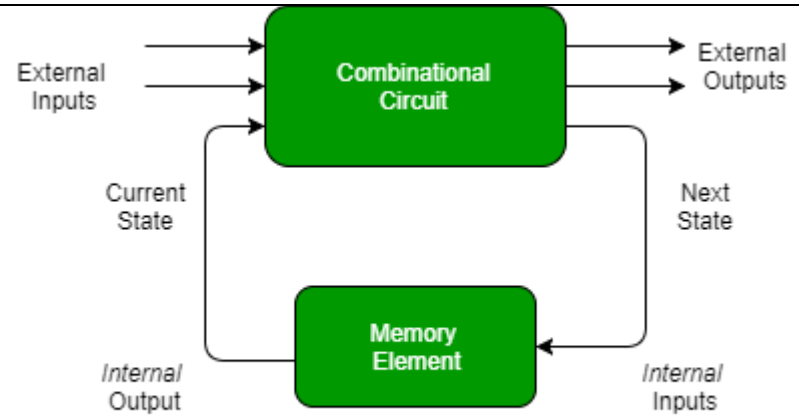


Figure: Sequential Circuit

As shown in figure there are two types of input to the combinational logic :

1. External inputs which not controlled by the circuit.
2. Internal inputs which are a function of a previous output states.

Secondary inputs are state variables produced by the storage elements, where as secondary outputs are excitations for the storage elements.

Types of Sequential Circuits – There are two types of sequential circuit :

1. **Asynchronous sequential circuit** – These circuit do not use a clock signal but uses the pulses of the inputs. These circuits are faster than synchronous sequential circuits because there is clock pulse and change their state immediately when there is a change in the input signal. We use asynchronous sequential circuits when speed of operation is important and independent of internal clock pulse.

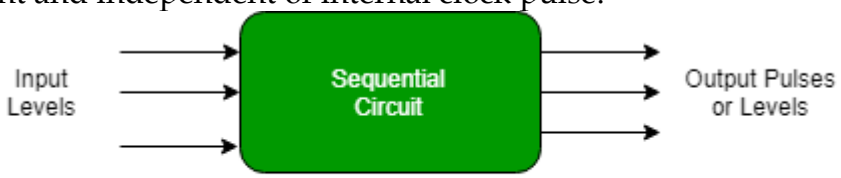


Figure: Asynchronous Sequential Circuit

But these circuits are more difficult to design and their output is uncertain.

2. **Synchronous sequential circuit** – These circuit uses clock signal and level inputs (or pulsed) (with restrictions on pulse width and circuit propagation). The output pulse is the same duration as the clock pulse for the clocked sequential circuits. Since they wait for the next clock pulse to arrive to perform the next operation, so these circuits are bit slower compared to asynchronous. Level output changes state at the start of an input pulse and remains in that until the next input or clock pulse.

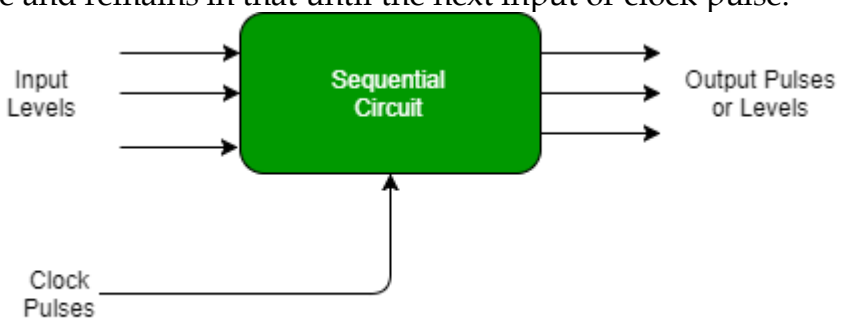


Figure: Synchronous Sequential Circuit

We use synchronous sequential circuit in synchronous counters, flip flops, and in the design of MOORE-MEALY state management machines.

Video Content / Details of website for further learning (if any):

https://www.geeksforgeeks.org/introduction-of-sequential-circuits/#:~:text=A%20Sequential%20circuit%20combinational%20logic,input%20and%20previous%20input%20variables.
https://www.youtube.com/watch?v=ibQBb5yEDlQ

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (329 - 346)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L-43

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **V - PROGRAMMABLE LOGIC DEVICES MEMORY AND VHDL**

Date of Lecture:

Topic of Lecture: Sequential Circuits - Testbenches

Introduction :

- Testbenches are used for simulation purpose only (not for synthesis), therefore full range of Verilog constructs can be used e.g. keywords 'for', 'display' and 'monitor' etc. can be used for writing testbenches.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

Testbench for combinational circuits

In this section, various testbenches for combinational circuits are shown, whereas testbenches for sequential circuits are discussed in next section. For simplicity of the codes and better understanding, a simple half adder circuit is tested using various simulation methods.

Half adder

Listing 1 shows the Verilog code for the half adder which is tested using different methods,

Listing 1 Half adder

```
1 // half_adder.v
2
3 module half_adder
4 (
5    input wire a, b,
6    output wire sum, carry
7 );
8
9 assign sum = a ^ b;
10 assign carry = a & b;
11
12 endmodule
```

Testbench with 'initial block'

Note that, testbenches are written in separate Verilog files as shown in Listing 2. Simplest way to write a testbench, is to invoke the 'design for testing' in the testbench and provide all the input values inside the 'initial block', as explained below,

Explanation Listing 2

In this listing, a testbench with name 'half_adder_tb' is defined at Line 5. Note that, ports of the testbench is always empty i.e. no inputs or outputs are defined in the definition (see Line 5). Then 4 signals are defined i.e. a, b, sum and carry (Lines 7-8); these signals are then connected to actual half adder design using structural modeling (see Line 13). Lastly, different values are assigned to input signals e.g. 'a' and 'b' at lines 18 and 19 respectively.

Note

'Initial block' is used at Line 15, which is executed only once, and terminated when the last line of the block executed i.e. Line 32. Hence, when we press run-all button in Fig. 1, then simulation terminated after 60 ns (i.e. does not run forever).

In Line 19, value of 'b' is 0, then it changes to '1' at Line 23, after a delay of 'period' defined at Line 20. The value of period is '20 (Line 11) * timescale (Line 3) = 20 ns'. In this listing all the combinations of inputs are defined manually i.e. 00, 01, 10 and 11; and the results are shown in Fig. 9.1, also corresponding outputs i.e. sum and carry are shown in the figure.

Note

To generate the waveform, first compile the 'half_adder.v and then 'half_adder_tb.v' (or compile both the file simultaneously.). Then simulate the half_adder_tb.v file. Finally, click on 'run all' button (which will run the simulation to maximum time i.e. 80 ns)and then click then 'zoom full' button (to fit the waveform on the screen), as shown in Fig. 1.

Listing 2 Simple testbench for half adder

```
1  // half_adder_tb.v
2
3  `timescale 1 ns/10 ps  // time-unit = 1 ns, precision = 10 ps
4
5  module half_adder_tb;
6
7     reg a, b;
8     wire sum, carry;
9
10    // duration for each bit = 20 * timescale = 20 * 1 ns  = 20ns
11    localparam period = 20;
12
13    half_adder UUT (.a(a), .b(b), .sum(sum), .carry(carry));
14
15    initial // initial block executes only once
16       begin
17          // values for a and b
18          a = 0;
19          b = 0;
20          #period; // wait for period
21
22          a = 0;
23          b = 1;
24          #period;
```

```
25
26          a = 1;
27          b = 0;
28          #period;
29
30          a = 1;
31          b = 1;
32          #period;
33        end
34 endmodule
```

Video Content / Details of website for further learning (if any):

https://verilogguide.readthedocs.io/en/latest/verilog/testbench.html
https://www.youtube.com/watch?v=QfIoAPio8oE

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (460 - 461)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L-44

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **V - PROGRAMMABLE LOGIC DEVICES MEMORY AND VHDL**

Date of Lecture:

Topic of Lecture: Examples: adders, counters, flip flops

Introduction :

- Testbenches are used for simulation purpose only (not for synthesis), therefore full range of Verilog constructs can be used e.g. keywords 'for', 'display' and 'monitor' etc. can be used for writing testbenches.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

Half Adder:

```
//Gate-level hierarchical description of 4-bit adder
//Description of half adder
module halfadder (S,C,x,y);
    input x,y;
    output S,C;
//Instantiate primitive gates
    xor (S,x,y);
   and (C,x,y);
endmodule
```

Full Adder:

```
//Description of full adder
module fulladder (S,C,x,y,z);
    input x,y,z;
    output S,C;
```

```
    wire S1, D1, D2;
//Instantiate primitive gates
     halfadder HA1 (S1, D1,x,y);
                    HA2 (S,D2,S1,z);
or g1(C,D2,D1);
endmodule
```

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=SWuricUYfKE
https://www.youtube.com/watch?v=k5qfPBbhXow

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (455 - 460)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L-45

ECE

II/III

Course Name with Code: **19ECC04- DIGITAL SYSTEM DESIGN**

Course Faculty : **Mr.S.BHOOPALAN, AP/ECE**

Unit : **V - PROGRAMMABLE LOGIC DEVICES MEMORY AND VHDL**

Date of Lecture:

Topic of Lecture: Examples: FSM, Multiplexers / De-Multiplexers

Introduction :

- Testbenches are used for simulation purpose only (not for synthesis), therefore full range of Verilog constructs can be used e.g. keywords 'for', 'display' and 'monitor' etc. can be used for writing testbenches.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Combinational Circuits
- Sequential Circuit

2x1 Multiplexer:

```
//Dataflow description of 2to1 line multiplexer
module mux2x1_df (A,B,select,OUT);
    input A,B,select;
    output OUT;
    assign OUT = select ? A : B;
endmodule
```

4x1 Multiplexer:

```
//Behavioral description of 4to1 line multiplexer
module mux4x1_bh (i0,i1,i2,i3,select,y);
    input i0,i1,i2,i3;
    input [1:0] select;
    output y;
    reg y;
   always @ (i0 or i1 or i2 or i3 or select)
```

```
        case (select)
          2'b00 : y = i0;
          2'b01 : y = i1;
          2'b10 : y = i2;
          2'b11 : y = i3;
        endcase
endmodule
```

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=dcN_P3P5FHY
https://www.youtube.com/watch?v=kfuX3ScWho4

Important Books/Journals for further learning including the page nos.:

Morris Mano M. and Michael D. Ciletti "Digital Design" , Pearson Education- V Edition,2013. Page No (453-454)

Course Faculty

Verified by HOD