

Digital Circuit Design/ Lab

2020-2021

الدراسات الاولى /المرحلة الاولى / الفصل الثاني

أساتذة المادة العملية:

م.د.بشرى عبدالله سلطان

ا.م.د. اسماء عبدالله فهد

م. قصواء خالد

م.م ياسمين علاء

رئيس مبرمجين ولاء سامي

Experiment No. 1

Logic Gates

Objectives:

- Understanding the meaning of Binary logic, Logic Gates and Boolean Functions.

Theory:

1. Definition of Binary logic:

Binary logic consists of binary variables and logic operation. The variables are designated by letters e.g. A, B, C, r, w, x, y with only two distinct values: 1 and 0. The basic logical operators are AND, OR and NOT.

AND similar to * in binary arithmetic

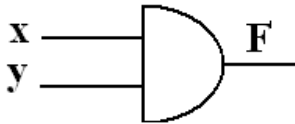
OR similar to + in binary arithmetic but $1+1=10$ (in binary arithmetic)



$$1+1=1 \quad (\text{in binary logic})$$

2. Logic Gates:

Digital Circuits, Switching Circuits, Logic Circuits and Logic Gates are the same. Gates are block of hardware that produces a logic-1 or logic-0 output signal if input logic requirement are satisfied. Figure (1) illustrates the basic logic gates.

Fig. 1: The basic logic gates

Name	Graphical Symbol	Algebraic Function	Truth Table		
AND		$F=x.y$	x	y	F
			0	0	0
			0	1	0
			1	0	0
			1	1	1

OR		$F=x+y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT, Inverter			<table><tr><th>x</th><th>x'</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	x'	0	1	1	0									
x	x'																	
0	1																	
1	0																	

Procedure:

1. Verify the truth table of AND gate by connecting the gate shown in Figure (1), using logic circuit designer (logisim).
2. Repeat step1 for all types of logic gates shown in Figure (1).
3. Verify the truth table of 3-inputs OR , AND, NOR & NAND gates

Experiment No. 2-1

Boolean Function

Objectives:

- Determinates the operator precedence for evaluating Boolean expression when obtaining their T.T and logic circuits.

Theory:

2. Boolean Functions:

Boolean functions are formed from binary variables, logic operators and equal sign. The function value can be either 1 or 0:

Ex: $F1 = xy\bar{z}$
 $F2 = x + \bar{y}z$

A Boolean function also represented in truth table. A Boolean function may be transformed from an algebraic expression into a logic diagram or circuit composed of AND, OR and NOT get.

NOTE: The operator precedence for evaluating Boolean expression is:

1. Parentheses 2. NOT 3. AND 4. OR

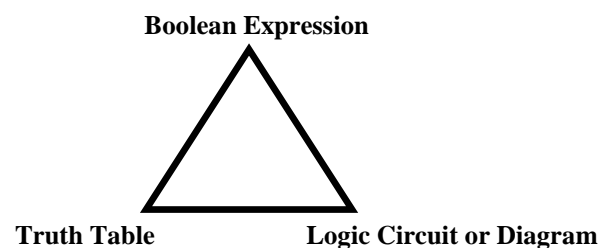
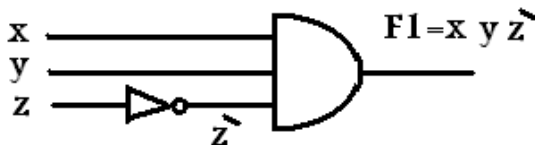
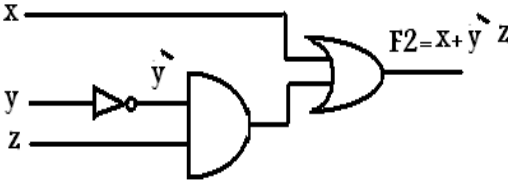


Table (1): The Boolean expressions, circuits and T.T of F1 and F2.

Boolean	Logic Circuit	Truth Table (T.T.)
---------	---------------	--------------------

Expression																																																														
$F1 = xy\bar{z}$		<table><tr><th colspan="5">Truth Table of F1</th></tr><tr><th>x</th><th>y</th><th>z</th><th>z'</th><th>F1=xyz'</th></tr><tr><td>0</td><td>0</td><td>0</td><td></td><td></td></tr><tr><td>0</td><td>0</td><td>1</td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>0</td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>1</td><td></td><td></td></tr><tr><td>1</td><td>0</td><td>0</td><td></td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td></td><td></td></tr><tr><td>1</td><td>1</td><td>0</td><td></td><td></td></tr><tr><td>1</td><td>1</td><td>1</td><td></td><td></td></tr></table>	Truth Table of F1					x	y	z	z'	F1=xyz'	0	0	0			0	0	1			0	1	0			0	1	1			1	0	0			1	0	1			1	1	0			1	1	1												
Truth Table of F1																																																														
x	y	z	z'	F1=xyz'																																																										
0	0	0																																																												
0	0	1																																																												
0	1	0																																																												
0	1	1																																																												
1	0	0																																																												
1	0	1																																																												
1	1	0																																																												
1	1	1																																																												
$F2 = x + \bar{y}z$		<table><tr><th colspan="6">Truth Table of F2</th></tr><tr><th>x</th><th>y</th><th>z</th><th>y'</th><th>y'z</th><th>x+y'z</th></tr><tr><td>0</td><td>0</td><td>0</td><td></td><td></td><td></td></tr><tr><td>0</td><td>0</td><td>1</td><td></td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>0</td><td></td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>1</td><td></td><td></td><td></td></tr><tr><td>1</td><td>0</td><td>0</td><td></td><td></td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td></td><td></td><td></td></tr><tr><td>1</td><td>1</td><td>0</td><td></td><td></td><td></td></tr><tr><td>1</td><td>1</td><td>1</td><td></td><td></td><td></td></tr></table>	Truth Table of F2						x	y	z	y'	y'z	x+y'z	0	0	0				0	0	1				0	1	0				0	1	1				1	0	0				1	0	1				1	1	0				1	1	1			
Truth Table of F2																																																														
x	y	z	y'	y'z	x+y'z																																																									
0	0	0																																																												
0	0	1																																																												
0	1	0																																																												
0	1	1																																																												
1	0	0																																																												
1	0	1																																																												
1	1	0																																																												
1	1	1																																																												

Procedure:

1. Verify the truth table of AND gate by connecting the gate shown in Figure (1), using logic circuit designer (logisim).
2. Repeat step1 for all types of logic gates shown in Figure (1).
3. Verify the truth table of 3-inputs OR , AND, NOR & NAND gates
4. Fill the T.T of F1 and F2 that shown in table (1)
5. Verify the truth table of F1 and F2 by connecting them, using logic circuit designer (logisim).

Report:

1. Write the (T.T) of the digital circuit shown in Figure (2).

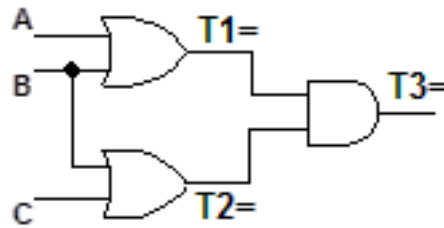


Fig. 2: Selected gates

2. Describe the output from the gates shown in Figure (2), or obtain the logical expression of T1, T2 and T3 in term of the input variables, for the following logic circuits.
3. Obtain the T.T and draw the logic diagram of the following Boolean function **without simplifying them**.
 - a) $F1 = B\bar{C} + AD$
 - b) $F2 = B(\bar{C} + A)$
 - c) $F3 = \overline{(X + Y)}(\bar{X} + Y)$

Experiment No. 3

Canonical

Objectives:

- How to obtain the Boolean expression for any logical function from its truth table

Theory:

3.1 Canonical forms:

A **Canonical form**: canonical forms (Sum of Minterms or Product of Maxterms) are used to obtain the function from the given truth table

x	y	z	Minterms	Notation	Maxterms	Notation
0	0	0	$x'y'z'$	m_0	$(x+y+z)$	M_0
0	0	1	$x'y'z$	m_1	$(x+y+z')$	M_1
0	1	0	$x'yz'$	m_2	$(x+y'+z)$	M_2
0	1	1	$x'yz$	m_3	$(x+y'+z')$	M_3
1	0	0	$xy'z'$	m_4	$(x'+y+z)$	M_4
1	0	1	$xy'z$	m_5	$(x'+y+z')$	M_5
1	1	0	xyz'	m_6	$(x'+y'+z)$	M_6
1	1	1	xyz	m_7	$(x'+y'+z')$	M_7
			Variable Primed if =0 Imprimed=1		Variable Primed if =1 Imprimed=0	

3.1 Sum of Minterms:

A Boolean function may be expressed algebraically as a sum of minterms from a given truth table by:

Step1: forming a minterm for each combination of the variables which produce 1 in the function.

Step2: OR all of the minterms in step1.

Example: From the given truth table express F as a sum of minterms

Given				Solution	
Inputs			Output	Step1	Step2
x	y	z	F	minterms	Sum of minterms
0	0	0	0		$F = x'y'z + x'yz + xyz$
0	0	1	1	$x'y'z$	$F = m_1 + m_3 + m_7$
0	1	0	0		$F(x,y,z) = \Sigma(1,3,7)$
0	1	1	1	$x'yz$	
1	0	0	0		
1	0	1	0		
1	1	0	0		
1	1	1	1	xyz	

3.2 Product of Maxterms:

A Boolean function may be expressed algebraically as a product of maxterms from a given truth table by:

Step1: forming a maxterms for each combination of the variables which produce 0 in the function.

Step2: form the AND of all the maxterms in step1.

Example: From the given truth table express F as a product of maxterms

Given				Solution	
Inputs			Output	Step1	Step2
x	y	z	F	maxterms	Product of maxterms
0	0	0	0	$(x+y+z)$	$F = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')(x'+y'+z)$
0	0	1	1		$F = M_0.M_2.M_4.M_5.M_6$
0	1	0	0	$(x+y'+z)$	$F(x,y,z) = \Pi(0,2,4,5,6)$
0	1	1	1		
1	0	0	0	$(x'+y+z)$	
1	0	1	0	$(x'+y+z')$	
1	1	0	0	$(x'+y'+z)$	
1	1	1	1		

Procedure:

For the given T.T do the following:

Given			
Inputs			Inputs
x	y	z	F
0	0	0	1

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

1. Express F as sum of minterms.
2. Verify the truth table of F that expressed in (1) by connecting it, using logic circuit designer (logisim).
3. Express F as product of maxterms.
4. Verify the truth table of F that expressed in (3) by connecting it, using logic circuit designer (logisim).

Assignment No. 4

Standard Forms

Objectives:

- Extract the relation between the canonical and standard forms.

Theory:

4.1 Standard Forms:

Sum terms: single variable or logical sum of several variables such as (A, B, (x+y), (A+C')).

Product terms: single variable or logical product of several variables such as (x, y, AB', CD')

Note: the expression $x+y'z$ (not sum term nor product terms)

4.2 Sum of Products (SOP): is a Boolean expression containing AND terms, called product terms, of one or more literals each. The sum denotes the ORing of these terms. An example of a function expressed as a sum of product is

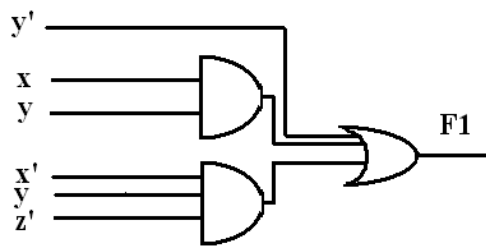
$$F1 = y' + xy + x'yz'$$

The expression contains three product terms (y' one literal, xy two literals and x'yz' three literals). Their sum is in effect an OR operation. The logic diagram if a sum-of-product expression consist of a group of AND gates followed by a single OR gate.

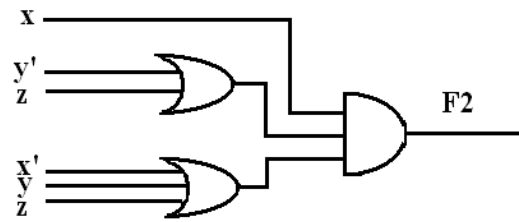
4.3 Product of Sums (POS): is a Boolean expression containing OR terms, called sum terms, of one or more literals each. The product denotes the ANDing of these terms. An example of a function expressed as a product of sum is

$$F2 = x(y' + z)(x' + y + z)$$

The expression contains three sum terms (x one literal, y'+z two literals and x'+y+z three literals). The product is an AND operation. The logic diagram if a product-of-sum expression consist of a group of OR gates followed by a single AND gate.



a. Sum of Product



b. Product of Sum

Two-level implementation

Example: From the given truth table express F as a sum of minterms **then** simplify as a sum of product

Given			
x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Solution: the function equal to 1 in $\{(2, x'yz'), (3, x'yz), (6, xyz')\}$

So $F = x'yz' + x'yz + xyz'$ (Sum of Minterms)

Simplification of F

$$F = \underline{x'yz'} + \underline{x'yz} + xyz'$$

$$= x'y(z' + z) + xyz' \quad (\text{distributive law})$$

$$= x'y.1 + xyz' \quad (\text{complement definition})$$

$$= x'y + xyz' \quad (\text{identity element})$$

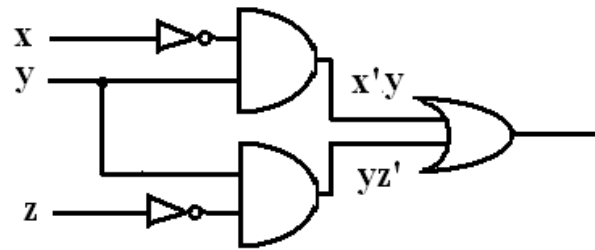
$$= y(x' + xz') \quad (\text{distributive law})$$

$$= y(x' + x)(x' + z') \quad (\text{distributive law})$$

$$= y.1.(x' + z') \quad (\text{complement definition})$$

$$= y(x' + z') \quad (\text{identity element})$$

$$=x'y+yz' \quad (\text{distributive law})$$



Logic Circuit of $F = x'y + yz'$

Example: From the given truth table express F as a product of maxterms **then** simplify as a product of sum

Given			
x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Solution: the function equal to 0 in $\{(0,x+y+z),(1,x+y+z'),(4,x'+y+z), (5,x'+y+z') ,(7,x'+y'+z')\}$

So $F = (x+y+z)(x+y+z')(x'+y+z)(x'+y+z')(x'+y'+z')$ product of maxterms

Simplification of F

$$F = (x+y+z)(x+y+z')(x'+y+z)(x'+y+z')(x'+y'+z')$$

$$= [(x+y)+zz'] [(x'+y)+zz'] (x'+y'+z') \quad (\text{distributive law})$$

$$= [(x+y)+0] [(x'+y)+0] (x'+y'+z') \quad (\text{complement definition})$$

$$= (x+y)(x'+y)(x'+y'+z') \quad (\text{identity element})$$

$$= (y+xx') (x'+y'+z') \quad (\text{distributive law})$$

$$= (y+0) (x'+y'+z') \quad (\text{complement definition})$$

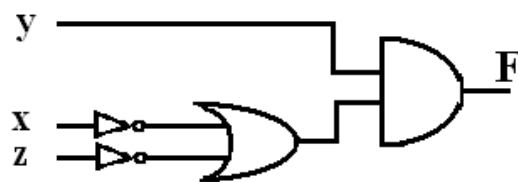
$$= y(x'+y'+z') \quad (\text{identity element})$$

$$= x'y+yy'+yz' \quad (\text{distributive law})$$

$$= x'y+0+yz' \quad (\text{complement definition})$$

$$= x'y+yz' \quad (\text{identity element})$$

$$= y(x'+z') \quad (\text{distributive law}) \quad \text{to convert the function from SOP to POS}$$



Logic Circuit of $F=y(x'+z')$

Report:

1. From the given truth

Given			
x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Do the following:

- Express F as a sum of minterms
- Simplify F expressed in (a) as a sum of product
- Verify the T.T for the simplified F that obtained from (b) by connecting it, using logic circuit designer (logisim).

2. From the given truth

Given			
x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Do the following:

- a. Express F as a product of maxterms
 - b. Simplify F expressed in (a) as a product of sums
 - c. Verify the T.T for the simplified F that obtained from (b) by connecting it, using logic circuit designer (logisim).
- 3.** What is the relation between the canonical and standard forms?

Experiment No. 5

Logic Function Forms

Objectives:

- Know how to converting the truth table to equivalent Boolean expression.
- Be able to use the map simplification to convert the function in canonical form (S.O. minterms) to the standard form (S.O.P.)

Theory:

5. Boolean Functions:

1. A Boolean function may be expressed algebraically as a sum of minterms from a given truth table by:

Step1: forming a minterm for each combination of the variables which produce 1 in the function.

Step2: OR all of the minterms in step1.

2. A Boolean function may be expressed algebraically as a product of maxterms from a given truth table by:

Step1: forming a maxterms for each combination of the variables which produce 0 in the function.

Step2: form the AND of all the maxterms in step1.

Example :

From the given truth table express F as a sum of minterms

Given				Solution	
Inputs			Output	Step1	Step2
X	Y	Z	F	minterms	Sum of minterms
0	0	0	0		$F = x'y'z + x'yz + xyz$
0	0	1	1	$x'y'z$	$F = m_1 + m_3 + m_7$
0	1	0	0		$F(x,y,z) = \Sigma(1,3,7)$
0	1	1	1	$x'yz$	
1	0	0	0		
1	0	1	0		
1	1	0	0		
1	1	1	1	xyz	

Practical Activities:

Practical 1:

For the function formed in the previous example do the following

- Draw a logic circuit and build it on the simulator logic circuit designer (logisim).
- Simplify using Map method.

Practical 2:

From the given truth table express F as a product of maxterms

Given			
Inputs			Output
x	Y	z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

For the function formed this practice in do the following

- Draw a logic circuit and build it on the simulator logic circuit designer (logisim).
- Simplify using Map method as S.O.P.

Practical 3:

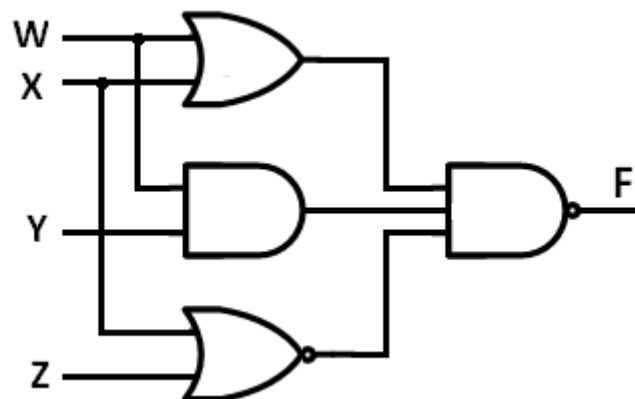


Figure (3.1)

For the logic circuit shown in figure (3.1) do the following:

- Write an expression for F.
- Build and test the logic circuit and obtain its T.T

Report:

1. For the following K_map suggest the perfect 1's collection and write the logical simplified S.O.P. Boolean expression.

CD		C			
		00	01	11	10
AB	00	1 ⁰	1 ¹	1 ³	1 ²
	01		1 ⁵	1 ⁷	
	11				
	10		1 ⁹	1 ¹¹	

Groupings: A (rows 11, 10), B (columns 01, 11), C (columns 11, 10), D (columns 01, 11)

2. For the following K_map suggest the perfect 1's collection and write the logical simplified S.O.P. Boolean expression.

CD		C			
		00	01	11	10
AB	00	1 ⁰			1 ²
	01	1 ⁴			
	11	1 ¹²	1 ¹³	1 ¹⁵	1 ¹⁴
	10	1 ⁸			1 ¹⁰

Groupings: A (rows 11, 10), B (columns 01, 11), C (columns 11, 10), D (columns 01, 11)

Assignment No. 6

The EXCLUSIVE-OR operation

Objectives:

- Understand the meaning of EXCLUSIVE-OR
- Know several methods of realizing it.

Theory:

6.Exclusive OR (XOR): odd function equal to 1 if the numbers the input variables have an odd number of 1's and 0 otherwise.

X	Y	F	Minterms	Maxterms
0	0	0		$(X + Y)$
0	1	1	$\bar{X}Y$ (S)	
1	0	1	$X\bar{Y}$ (S)	
1	1	0		$(\bar{X} + \bar{Y})$

$$F = X \oplus Y \quad \dots (1)$$

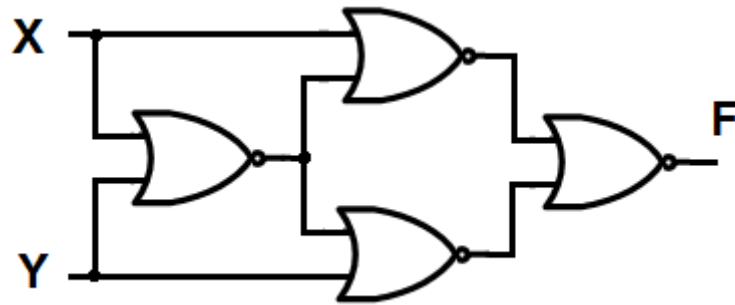
$$F = \bar{X}Y + X\bar{Y} \quad \dots (2) \quad \text{The EXCLUSIVE-OR expressed as a sum of minterms}$$

$$F = (X + Y).(\bar{X} + \bar{Y}) \quad \dots (3) \quad \text{The EXCLUSIVE-OR expressed as a product of maxterms}$$

Practical Activities:

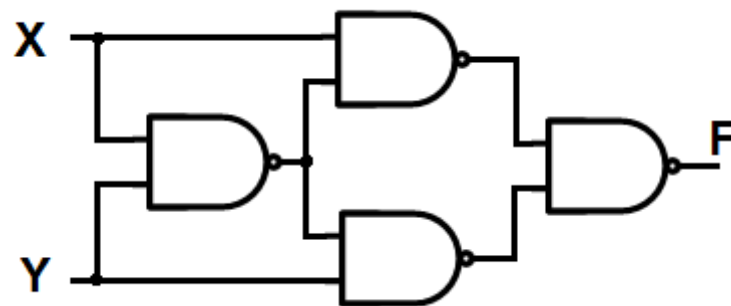
Practical 1:

- Draw a logic circuit in equation (1) and build it on the simulator logic circuit designer (logisim).
- Draw a logic circuit in equations (2 and 3) and build it on the simulator logic circuit designer (logisim).

Practical 2**Figure (4.1)**

For the logic circuit shown in figure (4.1) do the following:

- Write an expression for F.
- Build and test the logic circuit and obtain its T.T

Practical 3:**Figure (4.2)**

For the logic circuit shown in figure (4.2) do the following:

- Write an expression for F.
- Build and test the logic circuit and obtain its T.T

Report:

1. Compare the truth tables for figure (4.1) and (4.2), what do you conclude.
2. Derive the T.T of three variable XOR function, and express it as a sum of minterms

Assignment No. 7

Design Procedure

Objectives:

- How to design any combinational logic circuits

Theory:

7.Design Procedure:

The steps to design combinational circuits are as the following:

1. Understand the problem
2. Determine the number of input and output variables that are needed
3. Give symbols for the stated input and output
4. Construct a truth table that defines the relationship between the input and output
5. Obtain the Boolean function or the logical expression from the truth table in step 4, using K-map map or other known methods.
6. Draw a logic circuit based on the expression obtained from step 5 above.

Paractical Design a circuit that accept 3-bit binary number and produce 1 if the number of 0's greater than the number of 1's in the input combination and produce 0 otherwise.

Solution:

☒ Generate the truth table

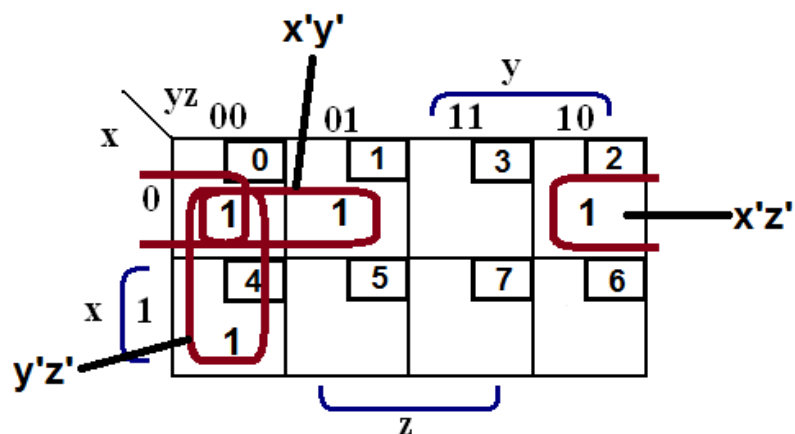
Inputs			N0. of 0's and 1's in the input combinations		Condition	Output
2^2	2^1	2^0	N0. of 0's	N0. of 1's	Is N0. Of 0's > N0. of 1's?	F
0	0	0	3	0	Is (3>0)=T	1
0	0	1	2	1	Is (2>1)=T	1
0	1	0	2	1	Is (2>1)=T	1
0	1	1	1	2	Is (1>2)=F	0
1	0	0	2	1	Is (2>1)=T	1
1	0	1	1	2	Is (1>2)=F	0
1	1	0	1	2	Is (1>2)=F	0
1	1	1	0	3	Is (0>3)=F	0

Inputs			Output
X	Y	Z	F
0	0	0	1
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	0

- ☒ Express each output function as sum of minterms:

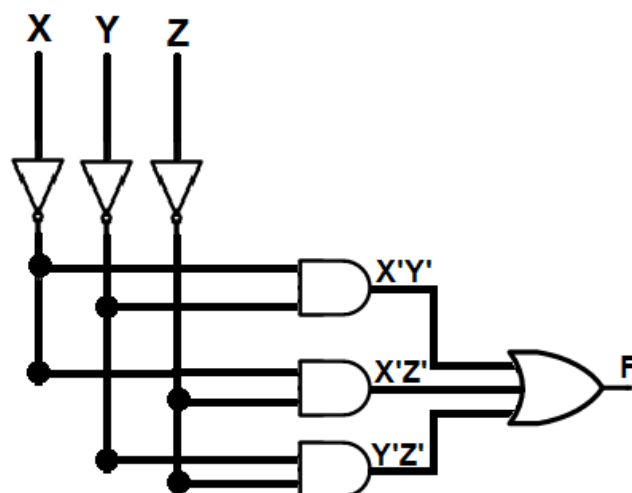
$$F(X,Y,Z) = \sum(0,1,2,4)$$

- ☒ Simplify each output function using(Boolean algebra or Map method)



$$F = Y'Z' + X'Z' + X'Y'$$

- ☒ Draw the logic circuit of the output function



Report:

1. Design a combinational circuit that accepts 3-bit binary number and produces the number of 1's in each input combination in binary.

Assignment No. 8

Code Conversions

Objectives:

1. Makes two systems compatible even though each uses a different binary code

Theory:

8. To convert a binary code A to binary code B, the input lines must supply the bit combination of element specified by code A and the output lines must generate the corresponding combination of code B.

Practical Design a combinational circuit that convert **decimal digit** represented in **2421 self-complemented** to **Excess-3** code.

Solution:

- The type of number system used is decimal (0-9)
- Excess-3 constructed from BCD code plus (0011).
- (2421) self-complemented is the input code while Excess-3 is the output code

Step1: write the input and output codes truth table

Decimal	Input Code				Decimal	Output Code			
	2	4	2	1		Excess-3			
	A	B	C	D		W	X	Y	Z
0	0	0	0	0	0	0	0	1	1
1	0	0	0	1	1	0	1	0	0
2	0	0	1	0	2	0	1	0	1
3	0	0	1	1	3	0	1	1	0
4	0	1	0	0	4	0	1	1	1
5	1	0	1	1	5	1	0	0	0
6	1	1	0	0	6	1	0	0	1
7	1	1	0	1	7	1	0	1	0
8	1	1	1	0	8	1	0	1	1
9	1	1	1	1	9	1	1	0	0
Table 1					Table 2				

Step2: for table 1 do the following

- Delete the decimal column and compute the binary values for each input combination

Binary	Input Code			
	8	4	2	1
	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
Table 3				

- From the binary column determine the don't care conditions (the numbers did not occurred in the binary column)=(5,6,7,8,9,10)
- The K-map used according to the number of input variables is 4-variable map

		C		
	0	1	3	2
	4	X	X	X
A	12	13	15	14
	X	X	11	X
		D		

- Delete the decimal column from table 2 and concatenate it with table 3

Binary	Input Code (2421)				Output Code (Excess-3)			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1

11	1	0	1	1	1	0	0	0
12	1	1	0	0	1	0	0	1
13	1	1	0	1	1	0	1	0
14	1	1	1	0	1	0	1	1
15	1	1	1	1	1	1	0	0
<u>Table 4</u>								

Practical Activities:

Practical 1:

From table 4 do the following:

- Express each output function as sum of minterms
- Find the Boolean expression of each output variable (i.e. w, x, y and z) using k-map for each one
- Determine the common terms among all output functions (if found)
- Draw the logic circuit

Report:

1. Design a combinational circuit that convert **decimal digit** represented in **84-2-1 code** to **2421 self-complemented** code.

Assignment No. 9

Decoder

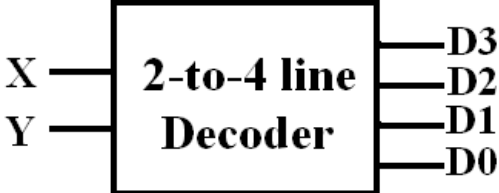
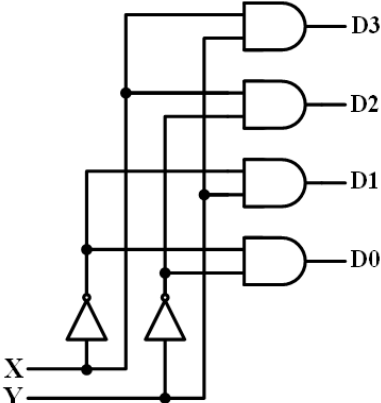
Objectives:

- Understanding the meaning of Decoder

Theory:

9.1 A binary code of n bits is capable of representing up to 2^n distinct binary code. A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n -bit decoded information has unused or don't-care combinations, the decoder output will have less than 2^n outputs.

- Practical 1 Design 2-to-4 line decoder

	<table><tr><th colspan="2">Inputs</th><th colspan="4">Outputs</th></tr><tr><th>x</th><th>y</th><th>D3</th><th>D2</th><th>D1</th><th>D0</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td></tr></table>	Inputs		Outputs				x	y	D3	D2	D1	D0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0
Inputs		Outputs																																			
x	y	D3	D2	D1	D0																																
0	0	0	0	0	1																																
0	1	0	0	1	0																																
1	0	0	1	0	0																																
1	1	1	0	0	0																																
$D3 = XY$ $D2 = X\bar{Y}$ $D1 = \bar{X}Y$ $D0 = \bar{X}\bar{Y}$																																					

Report

2. Design 4 to 8 line decoder

Assignment No. 10

Boolean Function Implementation Using Decoder

Objectives:

- How to implement the Boolean function using decoder

Theory:

10. A decoder provides the 2^n minterms of n input variables. Since, any Boolean function can be expressed in sum of minterms canonical form, one can use decoder to generate the minterms and external OR gate to form the sum. In this way, any combinational circuit with n inputs and m outputs can be implemented with n -to- 2^n line decoder and m OR gates.

- **Note:** if the number of minterms in the function is greater, than $2^n/2$ then F' can be expressed with fewer minterms than required for F . In such case, **it is advantages to neither use NOR gate** to sum minterms of F . the output of the NOR gate will generate the normal output F , as will be explained in the following example.

- **Paractical 1**

Implement $F = \bar{x}y + z$ using Decoder

$\because F = \bar{x}y + z$ is in S.O.P standard form so, it must be converted to S.O.minterms canonical form by adding the missing variables where the terms ($\bar{x}y$ missing z) and (z missing x and y)

$$F = \bar{x}y(z + \bar{z}) + z(x + \bar{x})$$

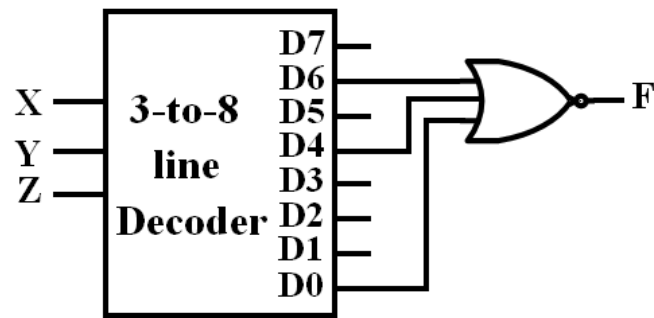
$$F = \bar{x}yz + \bar{x}y\bar{z} + xz + \bar{x}z$$

$$F = \bar{x}yz + \bar{x}y\bar{z} + xz(y + \bar{y}) + \bar{x}z(y + \bar{y})$$

$$F = \underbrace{\bar{x}yz}_3 + \underbrace{\bar{x}y\bar{z}}_2 + \underbrace{xzy}_7 + \underbrace{x\bar{y}z}_5 + \underbrace{\bar{x}yz}_3 + \underbrace{\bar{x}\bar{y}z}_1$$

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

\because The number of minterms in the F >, than $2^3/2$ then F' can be expressed as $\bar{F}(x, y, z) = \sum(0, 4, 6)$ and NOR gate used with 3-to- 2^3 line decoder to produce F .

**Reports:**

1. Implement H-A using decoder.