Laboratory Manual

Digital Electronics Laboratory
EEE/ECE 302 (V1, V2)
EEE/ECE 283L (V3)

Department of Electrical & Electronic Engineering (EEE)
School of Engineering (SoE)
Brac University

Revision: Revision: August 2021

A. Course Objectives:

The objectives of this course are to:

- train students to construct, test and debug combinational, sequential, and arithmetic circuits using Breadboard
- train students to simulate combinational, sequential, and arithmetic circuits using Proteus software.

B. Course Outcomes, CO-PO-Taxonomy Domain & Level- Delivery-Assessment Tool:

Sl.	**CO Description**	**POs**	**Bloom's taxonomy domain/level**	**Delivery methods and activities**	**Assessment tools**
EEE_ECE 301 / EEE_ECE 283 Digital Electronics					
CO1	**Apply** the concept of digital logic design to solve the problem using gates to replicate logic functions	a	Cognitive/Apply	Lectures, notes	Assignment, Quiz, Exam
CO2	**Analyze** combinational and sequential logic circuits built with various logic gates, flip-flops, registers, counters etc. represented through schematic diagram or hardware description language.	a	Cognitive/ Analyze	Lectures, notes	Assignment, Quiz, Exam
CO3	**Design** combinational and sequential logic circuits using various logic gates, flip-flops, registers, counters as building blocks	c	Cognitive/ Create	Lectures, notes	Assignment, Project
EEE_ECE302 / EEE_ECE 283 L Digital Electronics Lab					
CO4	**Perform** effectively as an individual or in a team to design and build combinational and sequential logic circuits in the laboratory or project development	i	Affective/ Valuing	Lab Class	Observation, Peer-review
CO5	**Communicate** the findings of hardware and software experiments and projects through reports and presentations	j	Affective/ Valuing	Lab Class	Lab Reports, Project Reports and Presentation

C. Mark Distribution

Assessment Tools	Weightage
Attendance	10%
Class Performance	10%
Lab Report (Hardware, Software)	20%
Project	30%
Lab Exam (Hardware)	30%
Total	100%

D. References

Sl.	Title	Author(s)	Publication Year	Edition	Publisher	ISBN
1	Digital Systems: Principles and Applications	Ronald J Tocci and Neal S Widmer,	2011	11th Ed	Prentice Hall,	0135103827, 9780135103821
2	Digital Logic and Computer Design	M. Morris Mano and Michael D. Ciletti,	2004	4th Ed.	Pearson/Prentice Hall,	013140539X, 9780131405394
3	Fundamentals of Logic Design	Roth, HR,	2010	6th edn	Thomson-Brooks/Cole	0495471690, 9780495471691

Lab Safety and Security Issues

1. *Laboratory Safety Rules (General Guidelines):*

The Department of EEE maintains general safety rules for laboratories. The guideline is attached in front of the door in each of the laboratories. The written rules are as follows.

1. Closed shoes must be worn that will provide full coverage of the feet and appropriate personnel clothing must be worn.

2. Always check if the power switch is off before plugging in to the outlet. Also, turn the instrument or equipment OFF before unplugging from the outlet.
3. Before supplying power to the circuit, the connections and layouts must be checked by the teacher.
4. Voltage equal or above 50V are always dangerous. Therefore, extra precautions must be taken as voltage level is increased.
5. Extension cords should be used only when necessary and only on a temporary basis.
6. Once the lab exercise is done, all equipment must be powered down and all probes, cords and other instruments must be returned to their proper position.
7. In case of fire, disconnect the electrical mains power source if possible.
8. Students must be familiar with the locations and operations of safety and emergency equipment like Emergency power off, Fire alarm switch and so on.
9. Eating, drinking, chewing gum inside electrical laboratories are strictly prohibited.
10. Do not use damaged cords or cords that become too hot or cords with exposed wiring and if something like that is found, inform the teacher/LTO right away.
11. No laboratory equipment can be removed from their fixed places without the teacher/LTO’s authorization.
12. No lab work must be performed without the laboratory teacher/lab technical officer being present.

2. Electrical Safety

To prevent electrical hazards, there are symbols in front of the Electrical Distribution Board, High voltage three phase lines in the lab, Backup generator and substation. Symbols related to Arc Flash and Shock Hazard, Danger: High Voltage, Authorized personnel only, no smoking etc. are posted in required places. Only authorized personnel are allowed to open the distribution boxes.

3. Electrical Fire:

If an electrical fire occurs, try to disconnect the electrical power source, if possible. If the fire is small, you are not in immediate danger, use any type of fire extinguisher except water to extinguish the fire. When in doubt, push in the Emergency Power Off button.

4. IMPORTANT:

Do not use water on an electrical fire.

List of Experiments

Lab No.	Experiment Name	Page	Tentative schedule	Is this experiment used for any CO assessment?	
				Yes	No
	Introductory Class (CO-PO, Group formation, Grading details etc.)	6	1st week	√	
1	Familiarization of Fundamental Logic Gates	12	2nd week	√	
2	Study and Application of Logic Simplification Techniques	18	3rd week		
3	Study and Application of Combinational Circuit Design Blocks: Multiplexer and Demultiplexer	19	4th week	√	
1-3	Proteus Simulation of Experiment of 1,2, and 3	6-26	5th week	√	
	MID WEEK		6th week		
4	Study and Application of Combinational Circuit Design Blocks: Encoder, Decoder and Data Converter	27	7th week	√	
5	Design and Implementation of Binary Adders and Subtractors	35	8th week	√	
6	Implementation of Counters	43	9th week	√	
4-6	Proteus Simulation of Experiment of 4-6	27-47	10th week	√	
	Exam (Hardware/Software)		11th week	√	
7	Demonstration of Open Ended Project	48	12th week	√	

Updated by:

1. Dr. A.S.M. Mohsin (Lab 1-11 and Project)

BRAC UNIVERSITY

Department of Electrical and Electronic Engineering

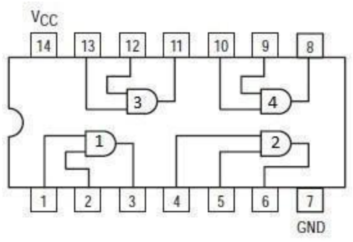
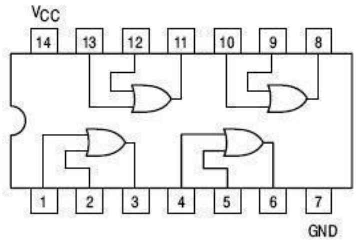
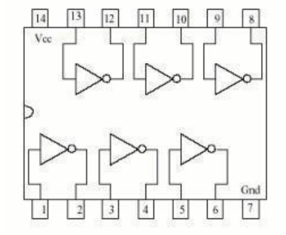
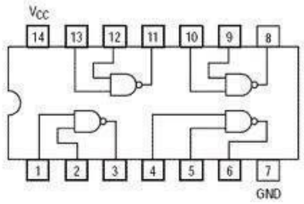
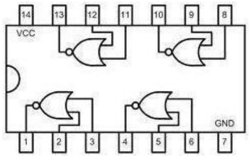
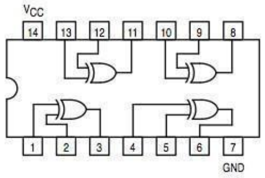
EEE/ECE 302/301LL: Digital Electronics/Digital Logic Design

Experiment No. 1

Familiarization with Logic Gates and Construction of Simple Logic Circuits using fundamental and universal gates

1. **Objective:**
   a. To get familiarized with logic gates AND (IC – 7408), OR (IC – 7432) and NOT (IC – 7404), XOR(IC-7486), NAND(IC-7400), NOR(IC-7402).
   b. To gain experience working with practical circuits
   c. To construct logic circuits using both fundamental and universal gates.

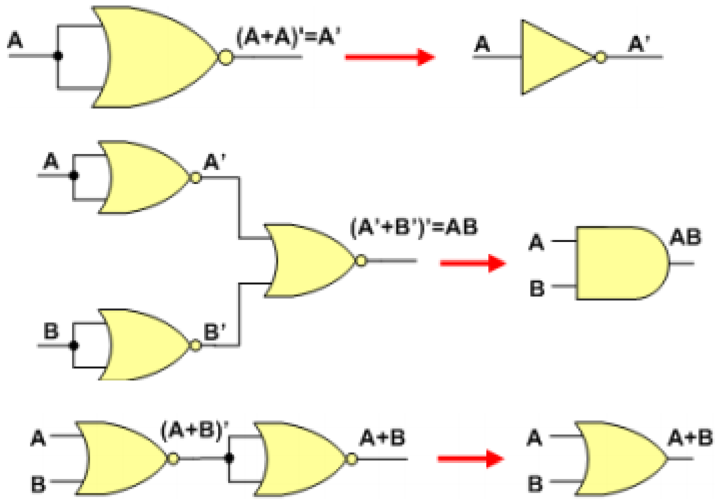
2. **Introduction/ Theoretical Background/ Problem Statement:**

 7408	 7432	 7404
 7400	 7402	 7486

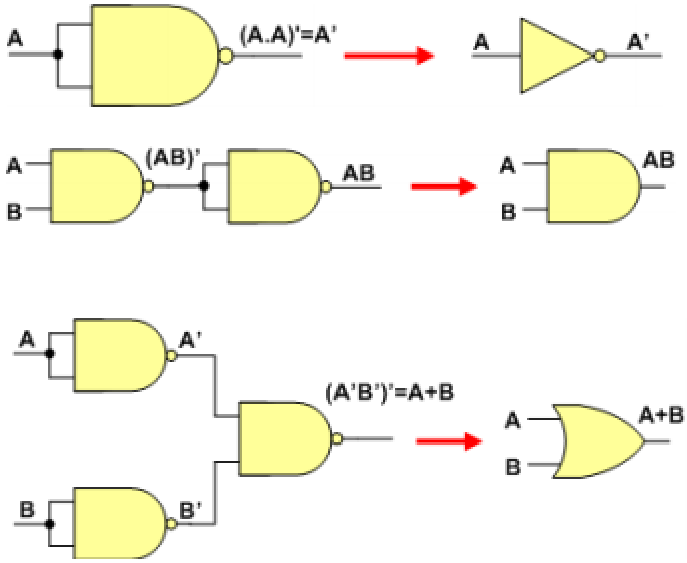
To find the pin configuration of other 74xx series ICs, visit
http://electronicsclub.info/74series.htm

Implementing fundamental gates using universal gates:

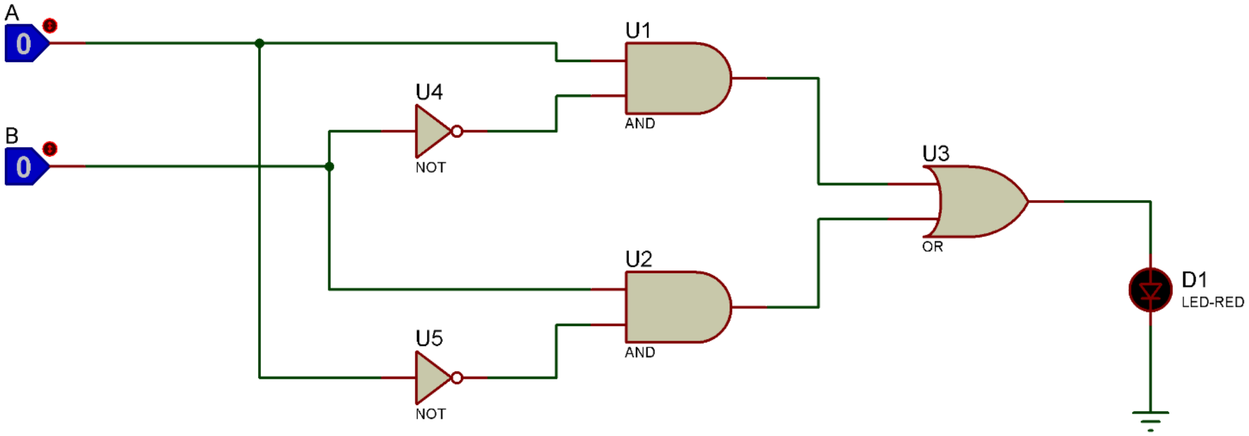
NOR Gate Implementation:



NAND Gate Implementation:



3. Circuit Diagram/ Experimental setup/Breadboard Implementation (*If applicable*):



Confirm the successful breadboard implementation

4. Components/ Equipment/ Tools/ Requirements/ Software requirements: (*If applicable*)

AT-700	1 Unit
DC Power Supply	1 Unit
Multimeter	1 Unit
IC: 7400	1 Piece
IC: 7402	1 Piece
IC: 7404	1 Piece
IC: 7408	1 Piece
IC: 7432	1 Piece
Connecting Wires	
Software : Proteus	

5. Procedure:

i. Draw and construct the circuit of the equation: $A\underline{B}+\underline{A}B$ on the breadboard of AT-700 using

   1. Fundamental logic gates only
   2. Multiple NOR Gates only

ii. Remember each IC"s pin 14 connected to "+5V" position of DC Power Supply of AT-700, and pin 7 connected to "GND" position.

iii. Connect the inputs to Data switches and outputs to any positions of LED Display.
iv. Find out the outputs for all possible combinations of input states.
v. Write down the input-output in tabular form.

Faculty Signature and Date

6. Lab report directions/ Questions/ Discussions/ Assignments:

The report should cover the followings

a. Name of the Experiment
b. Objective
c. Apparatus
d. Experimental Setup (Show the connection diagram with IC pinouts)
e. Results (Truth Table)
f. Questions and answers
   i. Why NAND and NOR gates are termed as “universal gates”?
   ii. Design the circuit used in experiment using multiple NAND gates only.
   iii. Design the circuit of the expression$(\underline{A} + B).(A + B + C).$ $(\underline{C})$ using fundamental gates, NAND gate and NOR gate using the least number of gates possible.
g. Discussion

Software Simulation (*Experiment No. 1 Cont'd*)

1. Objective: This part is intended to illustrate the simulation procedures of implementing logic gates.

2. Problem statement: Use Proteus software for building and simulating using basic gates

3. Software and Device requirement:

Proteus

Minimum PC specifications:

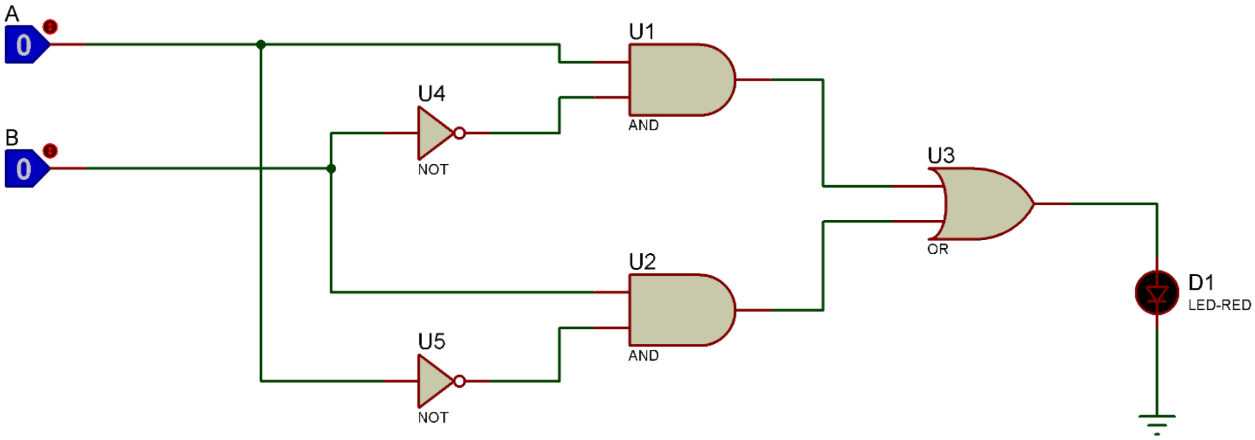
Windows/ Mac: Microsoft® Windows® 7 Professional, Enterprise, Ultimate or Home Premium (64-bit); Windows 8 (64-bit) (All Service Packs); Windows 10 (64-bit); Windows 2008 R2 Server; Windows 2012 *Server* (All Service Packs).

Ram: 2 GB

Processor: Intel® Pentium® 4 or AMD Athlon XP 2000 with multi-core CPU

Display resolutions: 1,024 x 768 display resolution with true color (16-bit color)

4. Circuit Diagram/ Experimental setup:



Confirm the successful Proteus implementation

5. Procedures:

- At first, select the ‘Component Mode’ and choose ‘Pick Devices’ and take the following components from the library: Required no. of AND gates, required no. of OR gates, required no. of NOT gates ...
- Design the circuits given in the circuit diagram
- Finally, go to the ‘Component Mode’ and pick devices named ‘LOGICTOGGLE’ and LED-RED to see the input/output bits.

Faculty Signature and Date

6. Lab report directions/ Questions/ Discussions/ Assignments:

The report should cover the followings

a. Name of the Experiment
b. Objective
c. Apparatus
d. Experimental Setup (Show the connection diagram with IC pinouts)
e. Results (Truth Table)
f. Questions and answers
    i. Why NAND and NOR gates are termed as “universal gates”?
    ii. Design the circuit used in experiment using multiple NAND gates only.
    iii. Design the circuit of the expression $(\underline{A}+B).(A+B+C).(\underline{C})$ using fundamental gates, NAND gate and NOR gate using the least number of gates possible.
g. Discussion

BRAC UNIVERSITY

Department of Electrical and Electronic Engineering

EEE/ECE 302/301LL: Digital Electronics/Digital Logic Design

Experiment No. 2
Study and Application of Logic Simplification Techniques

1. Objective:

- To investigate the rules of Boolean algebra
- To have insight into minimization principle of K-maps
- To simplify a complex function using Boolean algebra and K-map and observe the difference

2. Introduction/ Theoretical Background/ Problem Statement:

One of the primary requirements when dealing with digital circuits is to find ways to make them as simple as possible. This constantly requires that complex logical expressions be reduced to simpler expressions that nevertheless produce the same results under all possible conditions. The simpler expression can then be implemented with a smaller, simpler and in most cases faster circuit, which in turn saves the price of the unnecessary gates, reduces the number of gates needed, and reduces the power and the amount of space required by those gates. One tool to reduce logical expressions is the mathematics of logical expressions known as ***Boolean Algebra***. The rules of Boolean Algebra are simple and straightforward and can be applied to any logical expression.

AND Operations (·)		**OR Operations (+)**		**NOT Operations (')**
$0 \cdot 0 = 0$ $1 \cdot 0 = 0$ $0 \cdot 1 = 0$ $1 \cdot 1 = 1$	$A \cdot 0 = 0$ $A \cdot 1 = A$ $A \cdot A = A$ $A \cdot A' = 0$	$0+0 = 0$ $1+0 = 1$ $0+1 = 1$ $1+1 = 1$	$A+0 = A$ $A+1 = 1$ $A+A = A$ $A+A' = 1$	$0' = 1$ $1' = 0$ $(A')' = A$

Associative Law	**Distributive Law**	**De Morgan's Theorem**
(A·B)·C = A·(B·C) = A·B·C (A+B)+C = A+(B+C) = A+B+C	A·(B+C) = (A·B) + (A·C) A+(B·C) = (A+B) · (A+C)	(A·B)' = A' + B' (NAND) (A+B)' = A' · B' (NOR)

Principle of Duality: The *dual* of any true statement (axiom or theorem) in Boolean algebra is also a true statement. Given a logic expression, its *dual* is obtained by replacing all + operators with **·** operators, and vice versa, and by replacing all 0s with 1s, and vice versa.

Minterms: For a function of *n* variables, a product term in which each of the *n* variables appears once is called a *minterm*. For a given row of the truth table, the minterm is formed by including x_i if $x_i = 1$ and by including x_i' if $x_i = 0$. To identify the individual minterms easily, it is convenient to represent each minterm by an index that corresponds to the row numbers in the truth table.

Sum–of-Product Form: A function *f* can be represented by an expression that is a sum of minterms, where each minterm is ANDed with the value of *f* for the corresponding valuation of input variables. A logic expression consisting of product (AND) terms that are summed (ORed) is said to be of the sum-of-products (SOP) form. If each product term is a minterm, then the expression is called a canonical sum-of-products.

Cost of a Logic Circuit: A good indication of the *cost* of a logic circuit is the total number of gates plus the total number of inputs to all gates in the circuit. However primary inputs, namely, the input variables, are assumed to be available in both true and complemented forms at zero cost.

Row Number	**x1**	**x0**	**F**	**Minterm**
0	0	0	1	x0’ x1’
1	0	1	1	x0 x1’
2	1	0	0	x0’ x1
3	1	1	1	x0 x1

$$f = m_0 \cdot 1 + \ m_1 \cdot 1 + \ m_2 \cdot 0 + \ m_3 \cdot 1$$

$$f = m_0 + \ m_1 + \ m_3 = \sum (m_0, m_1, m_3) = \sum m(0,1,3)$$

$$f = \overline{x_0}\,\overline{x_1} + \ x_0\,\overline{x_1} + \ x_0\,x_1$$

This implementation requires 3 AND gates (2 input each) and 1 OR gate (3 input). Total cost will be 13. But this is not minimum cost implementation, applying Boolean theorems,

$$f = (\overline{x_0} + x_0)\,\overline{x_1} + x_0\,(\overline{x_1} + x_1)$$

$$f = \overline{x_1} + x_0$$

Cost of this implementation will be only 3.

Karnaugh Map: K map is a truth table arranged so that terms which differ by exactly one variable are adjacent to one another so that potential reductions can be readily observed. Like a truth table, K-map shows the output value for every combination of inputs. Each square represents a minterm. Only one variable changes between adjacent squares.

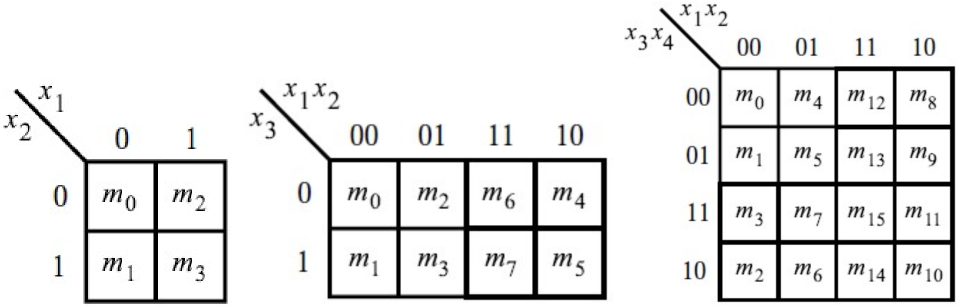
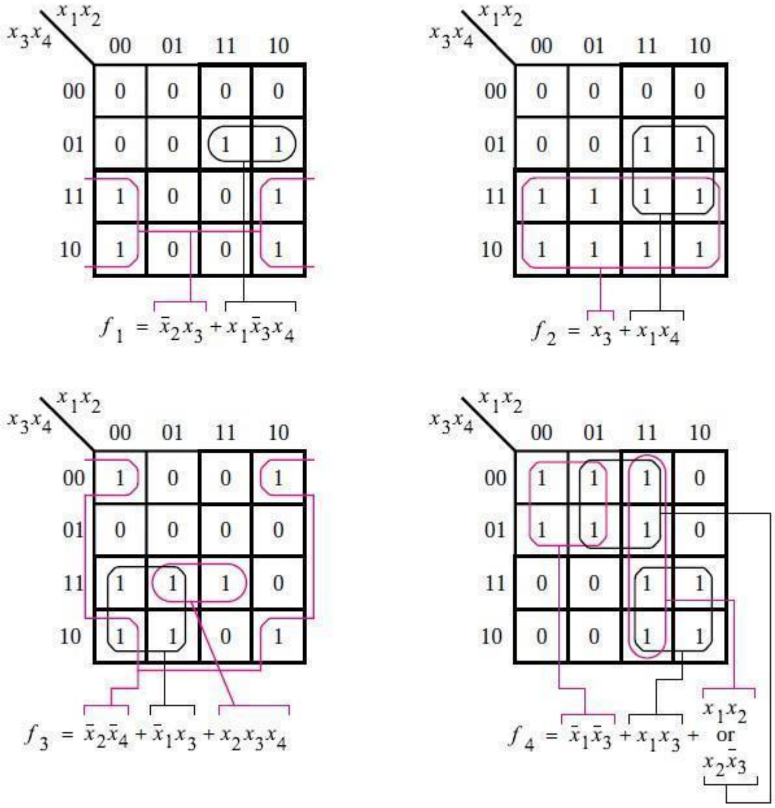


Fig: Karnaugh Map with different number of variables

Simplification Process using K-map:

1. Construct the K-map and place 1s and 0s in the squares according to the SOP expression or truth table
2. Find the largest grouping of 1s that are not already all in a group; if there is more than one possibility; choose a grouping that minimizes the total number of groups
3. Repeat step 2 until only isolated 1s remain
4. Form single groups of the remaining isolated 1s
5. Find the product term that corresponds to each group
6. OR together all the product terms



x_1x_2
x_3x_4
00 01 11 10
$f_1 = \bar{x}_2x_3 + x_1\bar{x}_3x_4$
$f_2 = x_3 + x_1x_4$
$f_3 = \bar{x}_2\bar{x}_4 + \bar{x}_1x_3 + x_2x_3x_4$
$f_4 = \bar{x}_1\bar{x}_3 + x_1x_3 +$ x_1x_2 or $x_2\bar{x}_3$

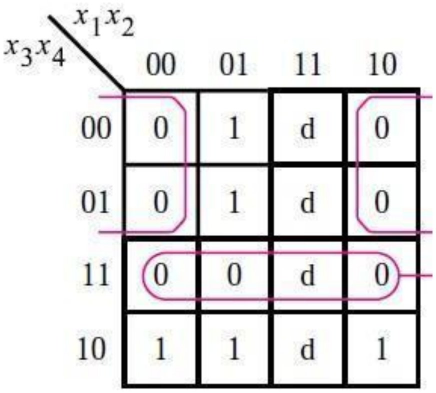
Incompletely Specified Conditions:

In digital systems it often happens that certain input conditions can never occur. That particular combination of input is considered as a *don't care condition* and system can be designed by ignoring this condition. A function that has *don't-care condition(s)* is said to be *incompletely specified*.

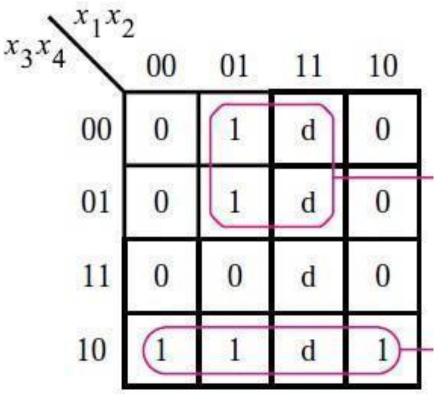
For example, x_1 and x_2 can be designed as two interlocked control switches such that both switches cannot be closed at the same time. Thus the input valuations (x_1, x_2) = 00, 01, and 10 are possible, but 11 is guaranteed not to occur.

Don't-care conditions can be used to advantage in the design of logic circuits. Since these input valuations will never occur, the designer may assume that the function value for these valuations is either 1 or 0, whichever is more useful in trying to find a minimum-cost implementation.

$$f = \sum m(2,4,5,6,10) + D(12,13,14,15)$$



x_3x_4 \ x_1x_2	00	01	11	10
00	0	1	d	0
01	0	1	d	0
11	0	0	d	0
10	1	1	d	1



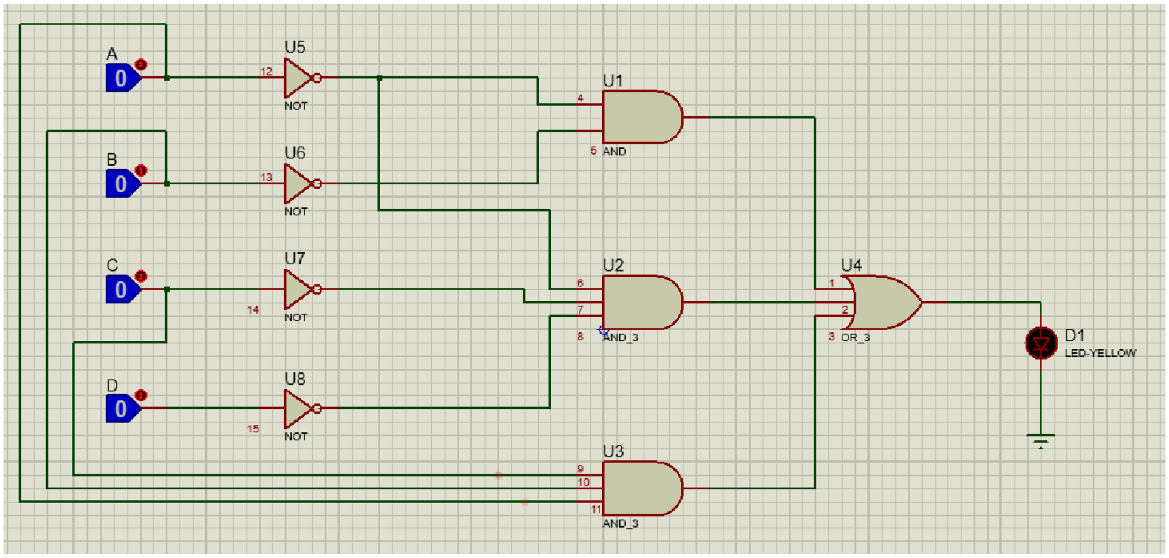
x_3x_4 \ x_1x_2	00	01	11	10
00	0	1	d	0
01	0	1	d	0
11	0	0	d	0
10	1	1	d	1

POS implementation treating d=0 **SOP implementation treating d=1**

2. **Components/ Equipment/ Tools/ Requirements/ Software requirements:** (*If applicable*)

- AT-700 Trainer Board
- Basic Logic Gates (AND 7408, OR 7432, NOT 7404)
- Connecting Wire

3. **Circuit Diagram/ Experimental setup/Breadboard Implementation** (*If applicable*)**:**



Confirm the successful breadboard implementation

Faculty Signature and Date

4. **Lab Work/Procedure:**

- Simplify the following logic function using K map (SOP Implementation), find out the truth table, write down the logic expression.

  F(A,B,C,D)=Σm(1,2,4,15)+ Σd(0,3,14)
- Implement your simplified circuit in the breadboard and verify the functionality of the circuit.

5. **Data Table/K-Map/Truth Table:** (*If applicable, should also include provision for calculation*)

Show related K-Map/Truth Table

6. **Lab report directions/ Questions/ Discussions/ Assignments:**

The report should cover the followings

1. Name of the Experiment
2. Objective
3. Apparatus
4. Experimental Setup (Show the connection diagram with IC pinouts)
5. Results (Truth Table)
6. Questions and answers

a What is the cost of the circuit built in the lab-task?

b Simplify the following logic function using K map (POS Implementation), find out the truth table, write down the logic expression and draw the simplified circuit. F(A,B,C,D)=Σm(1,2,4,15)+ Σd(0,3,14)

c Determine the cost of the circuit in (b).

d Compare the cost of the same logic circuit (a) and (b) when simplified by SOP and POS separately.

7. Discussion

7. **Acknowledgements:** (*If applicable*)

Software Simulation (*Experiment No. 2 Cont'd***)**

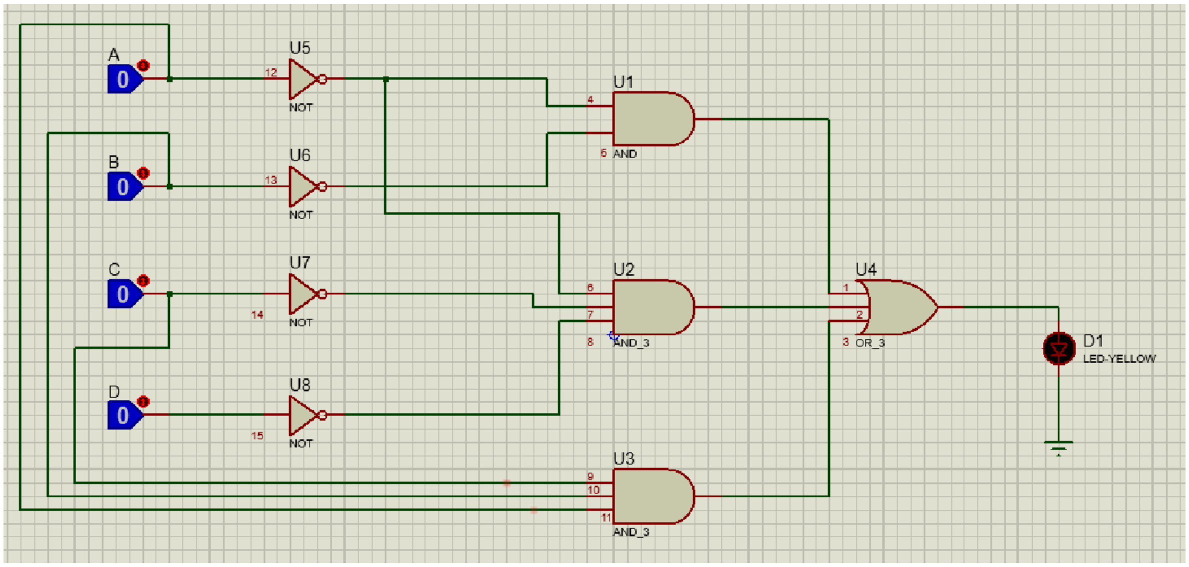
1. Objective: This part is intended to illustrate the simulation procedures of a simplified SOP/POS circuit.

2. Problem statement: Use Proteus software for building the given logic expression (Simplified SOP/POS circuit).

3. Software and Device requirement:

Proteus
<u>*Minimum PC specifications*</u>:
Windows/ Mac: Microsoft® Windows® 7 Professional, Enterprise, Ultimate or Home Premium (64-bit); Windows 8 (64-bit) (All Service Packs); Windows 10 (64-bit); Windows 2008 R2 Server; Windows 2012 *Server* (All Service Packs).
Ram: 2 GB
Processor: Intel® Pentium® 4 or AMD Athlon XP 2000 with multi-core CPU
Display resolutions: 1,024 x 768 display resolution with true color (16-bit color)

4. Circuit Diagram/ Experimental setup/Breadboard Implementation:



Confirm the successful Proteus implementation

Faculty Signature and Date

5. Procedure:

i) Open Proteus and Bring the components required for this experiment from library

ii) Connect all the components with wire

iii) Bring logic states, logic probe and logic toggle buttons to give input, LED as output and connect them with the circuit

iv) Verify the working of a simplified SOP/POS

BRAC UNIVERSITY

Department of Electrical and Electronic Engineering
EEE/ECE 302/301LL: Digital Electronics/Digital Logic Design

Experiment No. 3

Study and Application of Combinational Circuit Design Blocks: Multiplexer and Demultiplexer

1. Objective:

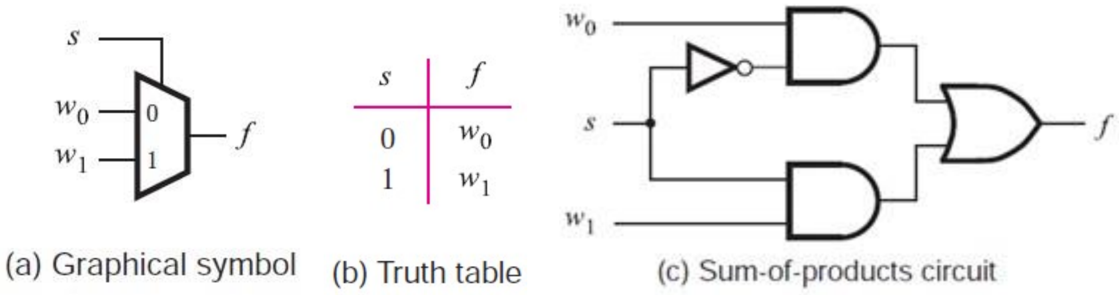
- **To design and implement Multiplexer and Demultiplexer using logic gates**
- **To verify the various functions of IC 74153(MUX) and IC 74139(DEMUX).**

2. Introduction/ Theoretical Background/ Problem Statement:

Theory:

In a combinational logic circuit, the output is dependent at all times on the combination of its inputs. If one of its input conditions changes state so does the output as combinational circuits have no *memory*. Multiplexer, Demultiplexer, Encoder, Decoder, Code Converter etc. are the most common blocks used to build larger combinational circuits. All these blocks can be constructed using basic logic gates.

Multiplexer or Data Selector: A multiplexer is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination. The basic multiplexer has several data input lines and a single output line. It also has data selector inputs, which permit digital data on any one of the inputs to be switched to the output line. Multiplexers are also known as data selector.



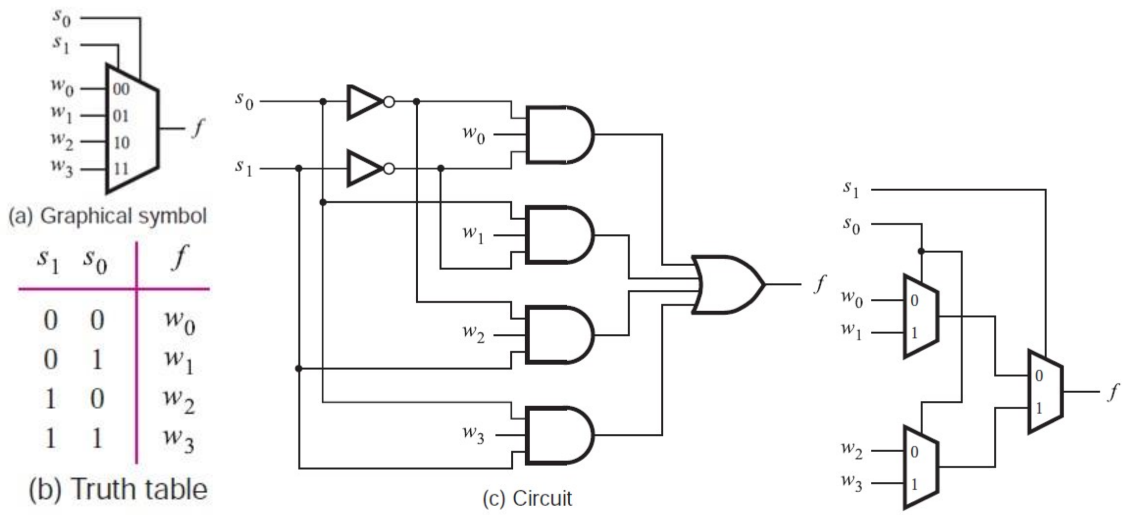
(a) Graphical symbol

s	f
0	w_0
1	w_1

(b) Truth table

(c) Sum-of-products circuit

Fig: 2-to-1 multiplexer



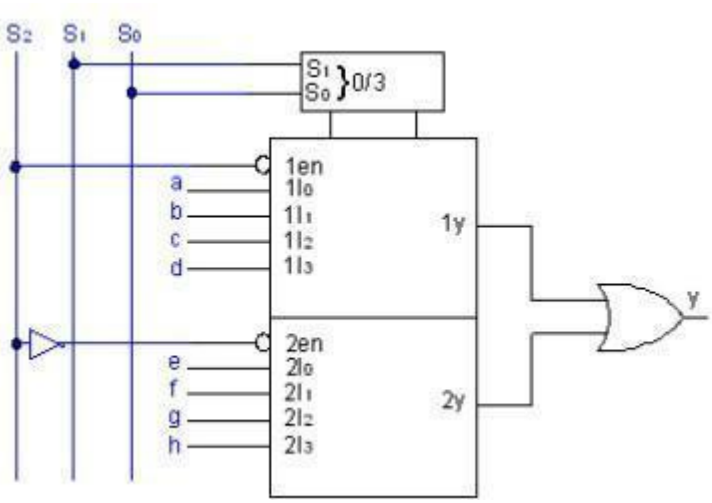
(a) Graphical symbol

s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

(b) Truth table

(c) Circuit

Fig: 4-to-1 multiplexer



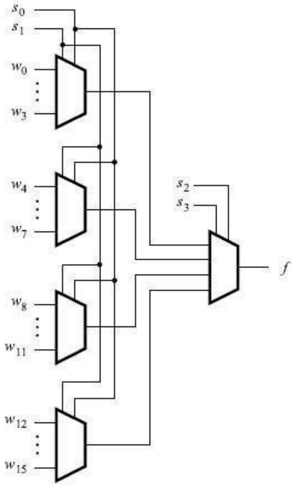
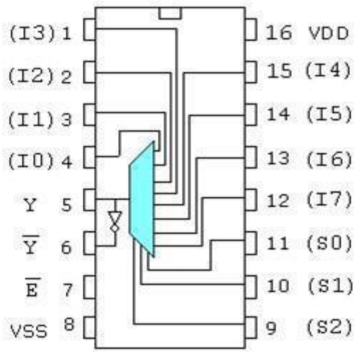
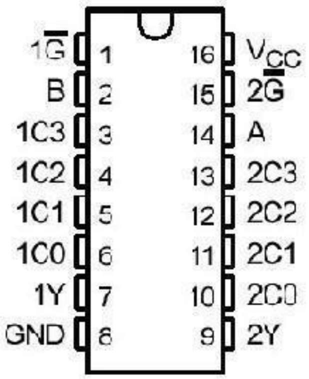


Fig: 8-to-1 multiplexer Fig: 16-to-1 multiplexer

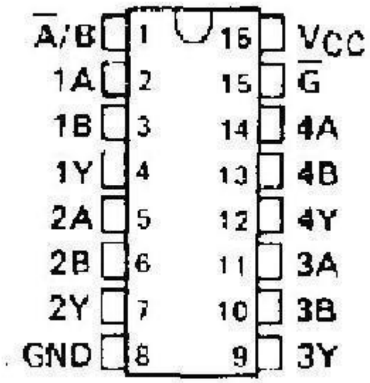
Common Multiplexer ICs



74151
8 to 1 line multiplexer



74153
Dual 4 to 1 line multiplexer



74157
Quad 2 to 1 line multiplexer

Demultiplexer:

A circuit that places the value of a single data input onto multiple data outputs is called a *demultiplexer*. It is used when a circuit wishes to send a signal to one of many devices. The demux contains one data input line, n data output lines and log$_2$n selector inputs. The valuation of selector inputs determine which output line is to be set to the data input line.

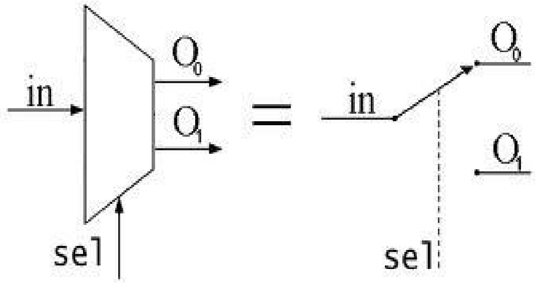


Fig: Principle of 1-to-2line Demux

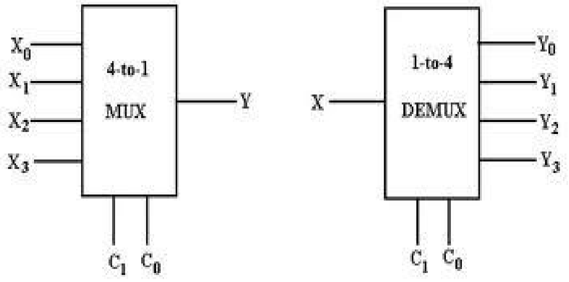
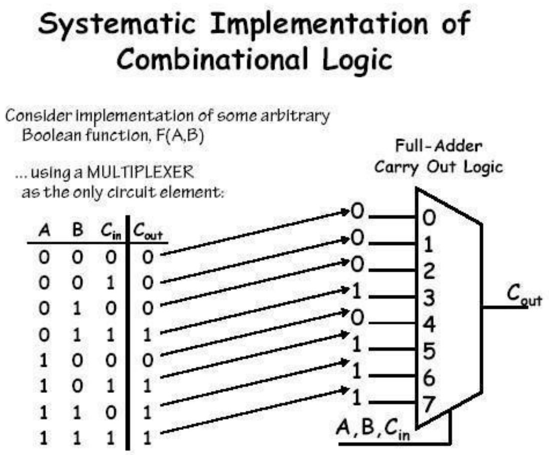


Fig: Comparison of Mux and Demux

Synthesis of Logic Functions Using Multiplexers:

A useful application of the data selector is in the generation of logic functions in sum of products form. When used in this way, the device can replace discrete gates, can often greatly reduce the number of ICs and can make the design changes much easier.



Systematic Implementation of Combinational Logic

Consider implementation of some arbitrary Boolean function, F(A,B)

... using a MULTIPLEXER as the only circuit element:

A	B	C_{in}	C_{out}
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

Fig: Implementation of Logic Function using Multiplexer

3. Components/ Equipment/ Tools/ Requirements/ Software requirements: (*If applicable*)

- One DC power supply
- IC 7404(NOT), IC 7408(AND), IC 7432(OR), IC 74151(3 – 8 MUX).
- Connecting wires and Breadboard.
- The Datasheets of the IC's.

4. Procedure:

i.Design a 2 to 1 MUX using fundamental gates: IC 7404(NOT), IC 7408(AND), IC 7432(OR)

ii.Design a 1 to 2 DEMUX using fundamental gates: IC 7404(NOT), IC 7408(AND), IC 7432(OR),

iii.Don't forget to connect the Vcc and GND for each IC used throughout your experiment.

iv.Simplify the following function using K-map and implement using 8 to 1 MUX IC (74151)

F(A,B,C)=Σm(1,3,6,7)

5. Lab report directions/ Questions/ Discussions/ Assignments:

- Name of the Experiment
- Objective
- Apparatus

- Experimental Setup (Show the connection diagram with IC pinouts)
- Results (Truth Table)
- Questions and answers
    - Design a 4 to 1 MUX using fundamental gates.
    - Design a 1 to 4 DEMUX using fundamental gates.
    - Simplify the following expression using K-map and design the circuit using two 8 to 1 MUX: F(A,B,C,D)=Σm(1,2,4,15)+ Σd(0,3,14)
- Discussion

Software Simulation (*Experiment No. 3 Cont'd*)

1. Objective: This part is intended to illustrate the simulation procedures of implementing Multiplexer and Demultiplexer using logic gates and illustrate it application using given logic expression

2. Problem statement: Use Proteus software for building Mux and Demux (using fundamental gates) and also building the given logic expression using 8x1 Mux fromProteus library.

3. Software and Device requirement:

Proteus

Minimum PC specifications:

Windows/ Mac: Microsoft® Windows® 7 Professional, Enterprise, Ultimate or Home Premium (64-bit); Windows 8 (64-bit) (All Service Packs); Windows 10 (64-bit); Windows 2008 R2 Server; Windows 2012 *Server* (All Service Packs).

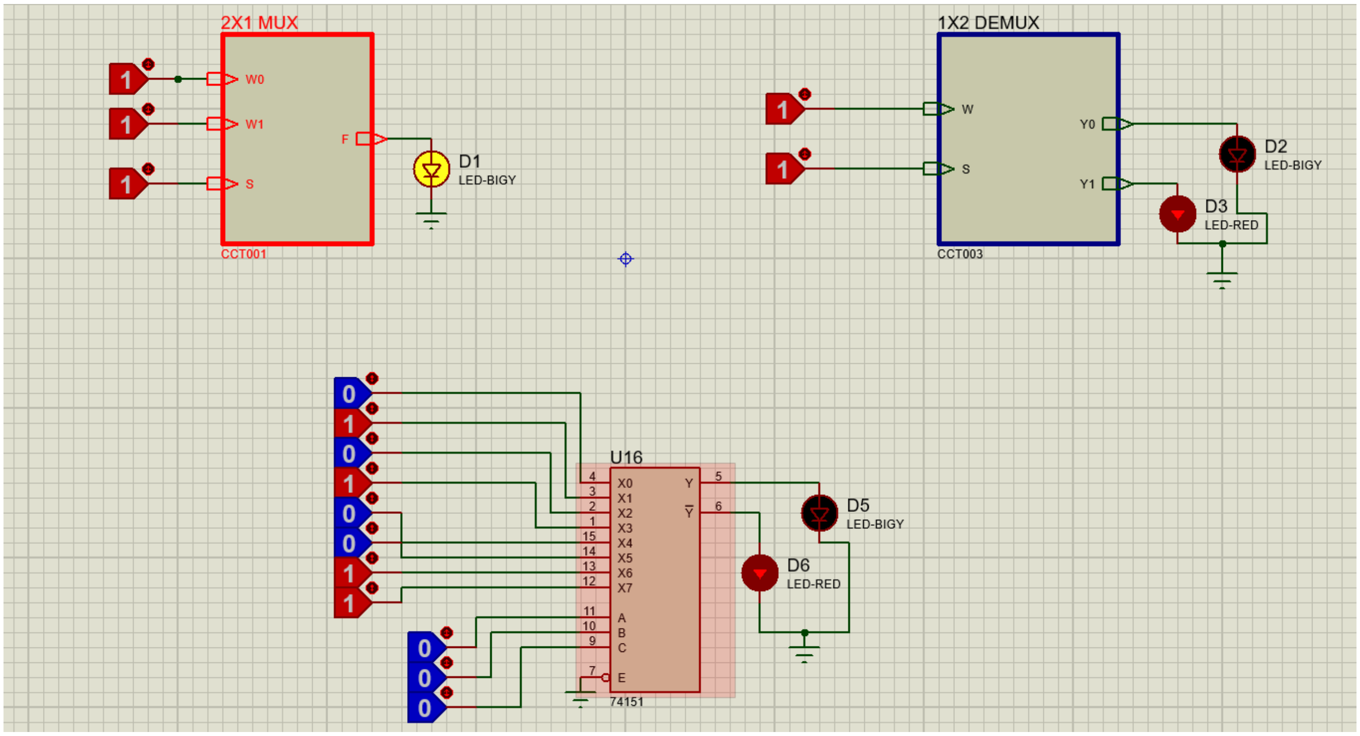
Ram: 2 GB

Processor: Intel® Pentium® 4 or AMD Athlon XP 2000 with multi-core CPU

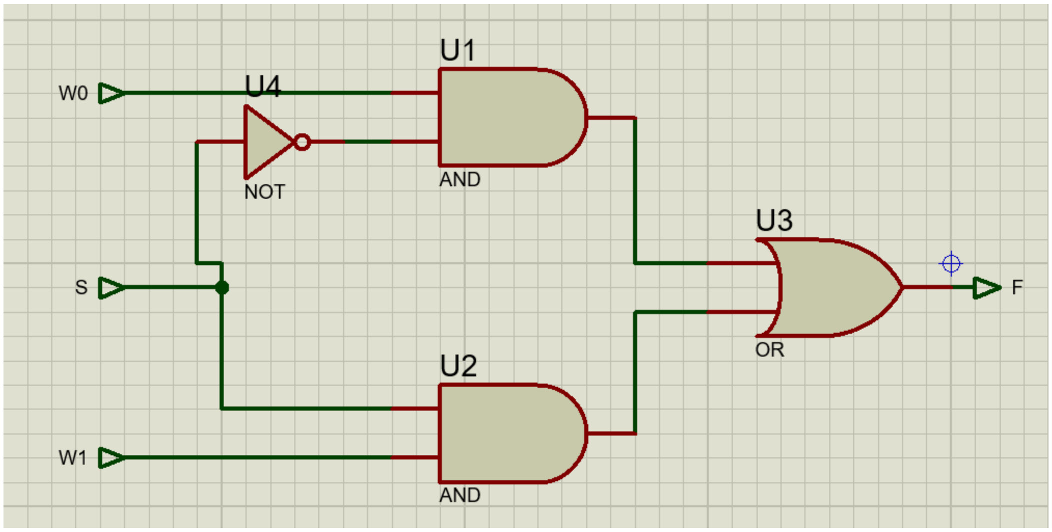
Display resolutions: 1,024 x 768 display resolution with true color (16-bit color)

4. Circuit Diagram/ Experimental setup:

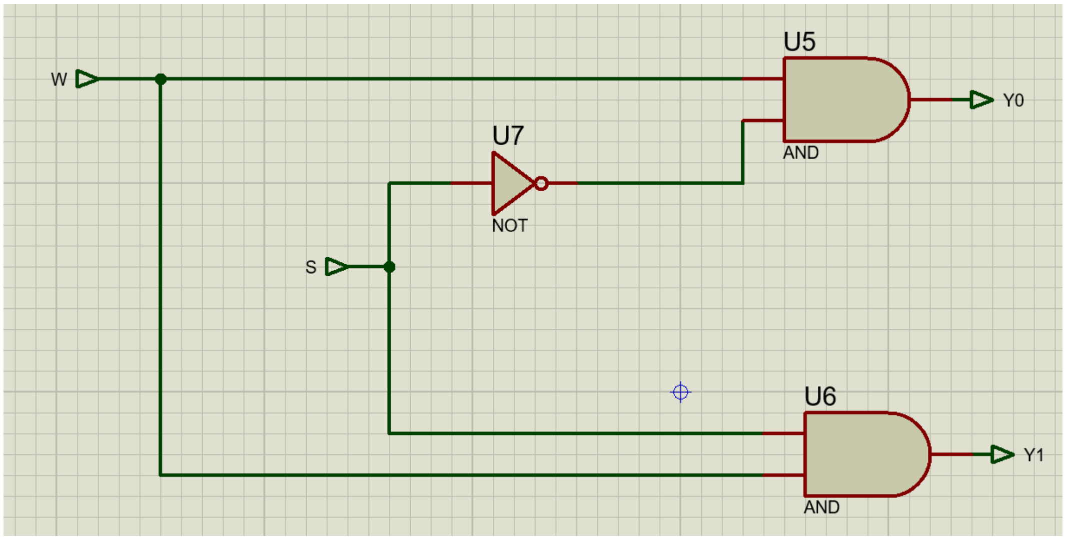
Parent Sheet:



Child Sheet for 2x1 MUX



Child Sheet for 2x1 DEMUX



5. Procedures:

- At first, select the ‘Component Mode’ and choose ‘Pick Devices’ and take the following components from the library: Required no. of AND gates, required no. of OR gates, required no. of NOT gates..
- Design the circuits given in the parent sheet diagram and also corresponding child sheets.
- To provide the input/output terminals, go to the ‘Terminals Mode’ and choose the ‘INPUT’ and ‘OUTPUT’ and give the names to the terminals.
- Now, go to the ‘Subcircuit Mode’ and create a Subcircuit, then cut the previously built circuit and paste it into the child sheet of the Sub circuit.
- Give the input/output terminals from the Subcircuit Mode and give the similar name of the child sheet input/output terminals.
- Finally, go to the ‘Component Mode’ and pick devices named ‘LOGICPROBE’ and ‘LOGICTOGGLE’ to see the input/output bits.

Faculty Signature and Date

6. Lab report directions/ Questions/ Discussions/ Assignments:

The report should cover the followings

1) Name of the Experiment
2) Objective
3) Apparatus
4) Experimental Setup (Show the connection diagram with IC pinouts)
5) Results (Truth Table)
6) Questions and answers
    - Design a 2 to 1 MUX using fundamental gates
    - Design a 1 to 2 DEMUX using fundamental gates:
    - Simplify the following function using K-map and implement using 8 to 1 MUX IC (74151); $F(A,B,C)=\Sigma m(1,3,6,7)$
7) Discussion

BRAC UNIVERSITY

Department of Electrical and Electronic Engineering

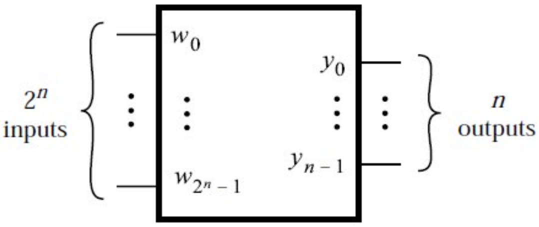
EEE/ECE 302/301LL: Digital Electronics/Digital Logic Design

Experiment No. 4

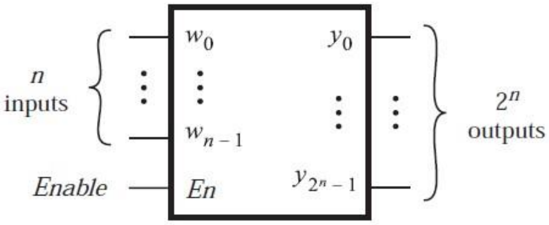
Study and Application of Combinational Circuit Design Blocks: Encoder, Decoder and Data Converter

1. **Objective:** This experiment is intended to study and apply combinational circuit design blocks like encoder, decoder and data converter.

2. **Introduction/ Theoretical Background/ Problem Statement:** The purpose of the decoder and encoder circuits is to convert from one type of input encoding to a different output encoding. For example, a 3-to-8 binary decoder converts from a binary number on the input to a one-hot encoding at the output. An 8-to-3 binary encoder performs the opposite conversion.



2^n-to-n binary encoder

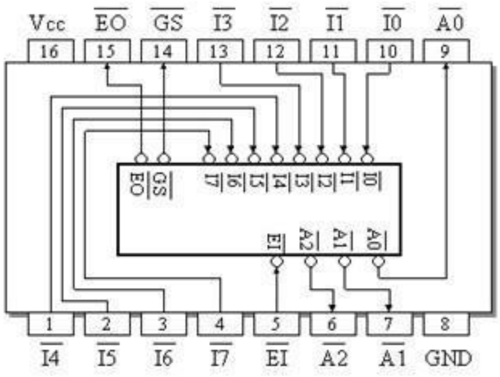


n-to-2^n binary decoder

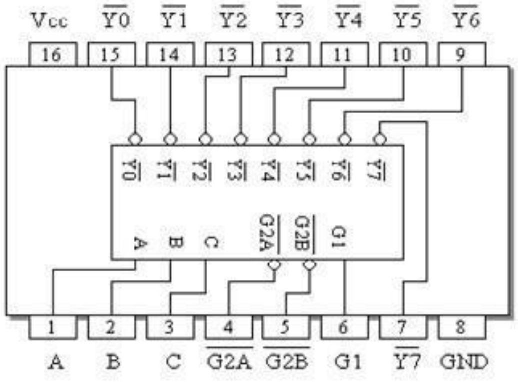
3. **Components/ Equipment/ Tools/ Requirements/ Software requirements:**

   - AT-700 Trainer Board
   - 74148 (8-to-3 priority encoder) – 1 pc
   - 74138 (3-to-8 binary decoder) – 1 pc
   - 7404 (Hex Inverter) – 1 pc

4. **Terminal Configuration:**



74148 (8-to-3 priority encoder)



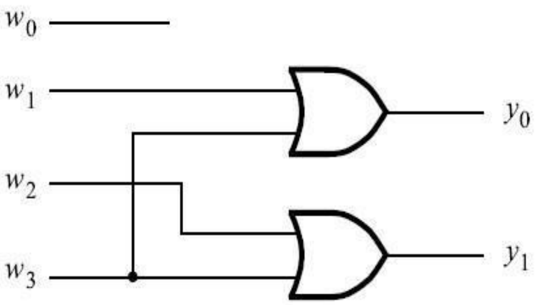
74138 (3-to-8 binary decoder)

5. Encoder:

A *binary encoder* encodes information from 2^n inputs into an *n*-bit code. Exactly one of the input signals should have a value of 1, and the outputs present the binary number that identifies which input is equal to 1. It encodes given information into a more compact form.

Encoders are used to reduce the number of bits needed to represent given information. A practical use of encoders is for transmitting information in a digital system. Encoding the information allows the transmission link to be built using fewer wires. Encoding is also useful if information is to be stored for later use because fewer bits need to be stored.

w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



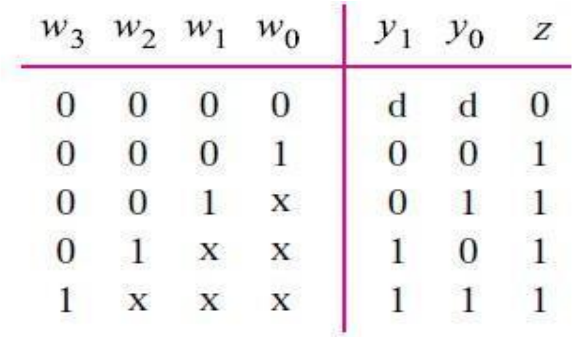
Truth table and logic circuit of a 4-to-2 binary encoder

6. Priority Encoder:

In a *priority encoder* each input has a priority level associated with it. The encoder outputs indicate the active input that has the highest priority. When an input with a high priority is asserted, the other inputs with lower priority are ignored.

The truth table for a 4-to-2 priority encoder is shown below. It assumes that w_0 has the lowest priority and w_3 the highest. The outputs y_1 and y_0 represent the binary number that identifies the highest priority input set to 1.

Since it is possible that none of the inputs is equal to 1, an output, z, is provided to indicate this condition. It is set to 1 when at least one of the inputs is equal to 1. It is set to 0 when all inputs are equal to 0. The outputs y_1 and y_0 are not meaningful in this case, and hence the first row of the truth table can be treated as a don't-care condition for y_1 and y_0.



w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

$$i_0 = w_3\, w_2\, w_1\, w_0$$

$$i_1 = w_3\, w_2\, w_1$$

$$i_2 = w_3\, w_2$$

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

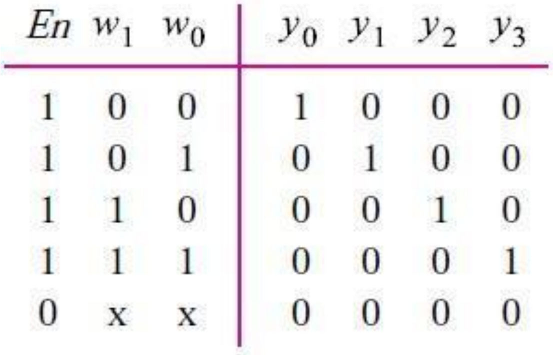
$$y_1 = i_2 + i_3$$

$$z = i_0 + i_1 + i_2 + i_3$$

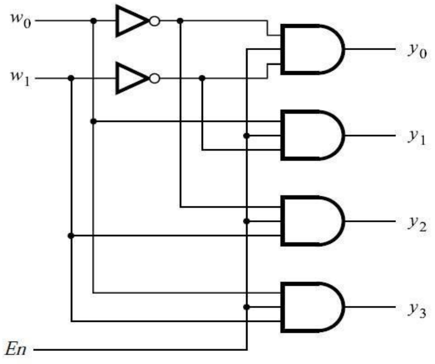
7. Decoder:

Decoder circuits are used to decode encoded information. A binary decoder is a logic circuit with n inputs and 2^n outputs. The outputs of a binary decoder are one-hot encoded. An n-bit binary code in which exactly one of the bits is set to 1 at a time is referred to as *one-hot encoded*, meaning that the single bit that is set to 1 is deemed to be "hot." Only one output is asserted at a time, and each output corresponds to one valuation of the inputs.

The decoder also has an enable input, *En*, that is used to disable the outputs; if *En* = 0, then none of the decoder outputs is asserted. If *En* = 1, the valuation of $w_{n-1} \cdot \cdot \cdot w_1 w_0$ determines which of the outputs is asserted. The output can be active LOW or active HIGH.



En	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0



Truth table and Logic circuit of a 2-to-4 binary decoder with active HIGH outputs

8. Synthesis of Logic Functions using Decoders:

Implement the function $f(w1, w2, w3) = \sum m(0, 1, 3, 4, 6, 7)$ by using a 3-to-8 binary decoder and an OR gate.

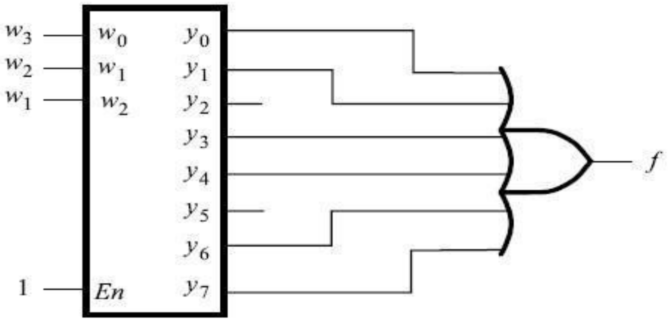


Fig: Implementation of a logic function (in SOP form) using decoders

9. Procedure:

i. Construct a 4 to 2 encoder with default priority.

ii. Construct a 2 to 4 decoder.

iii. Implement the following function using 3 to 8 decoder: F(C,B,A)=∑ $m(0,2,3,7) + d(1,4)$

Faculty Signature and Date

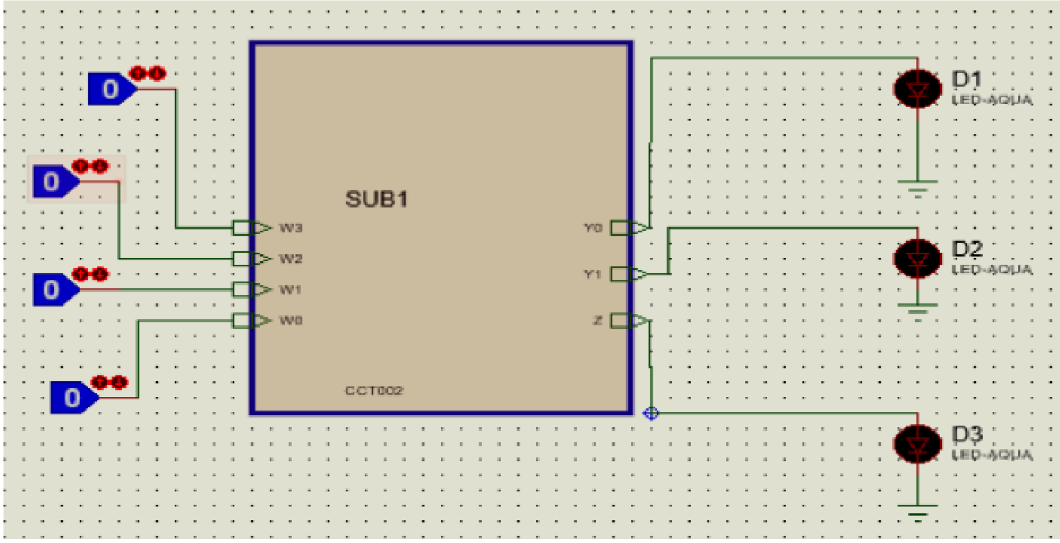
10. Lab report directions/ Questions/ Discussions/ Assignments:

1. Name of the Experiment
2. Objective
3. Apparatus
4. Experimental Setup (Show the connection diagram with IC pinouts)
5. Results (Truth Table)
6. Questions and answers
    a. Construct a priority encoder that implements the encoding of 0,3,1,2 with descending priority (i.e. 0 has the highest priority and 2 has the lowest).
    b. Draw a circuit diagram with encoder and decoder that will output the 1's complement of the 3-bit number.
7. Discussion

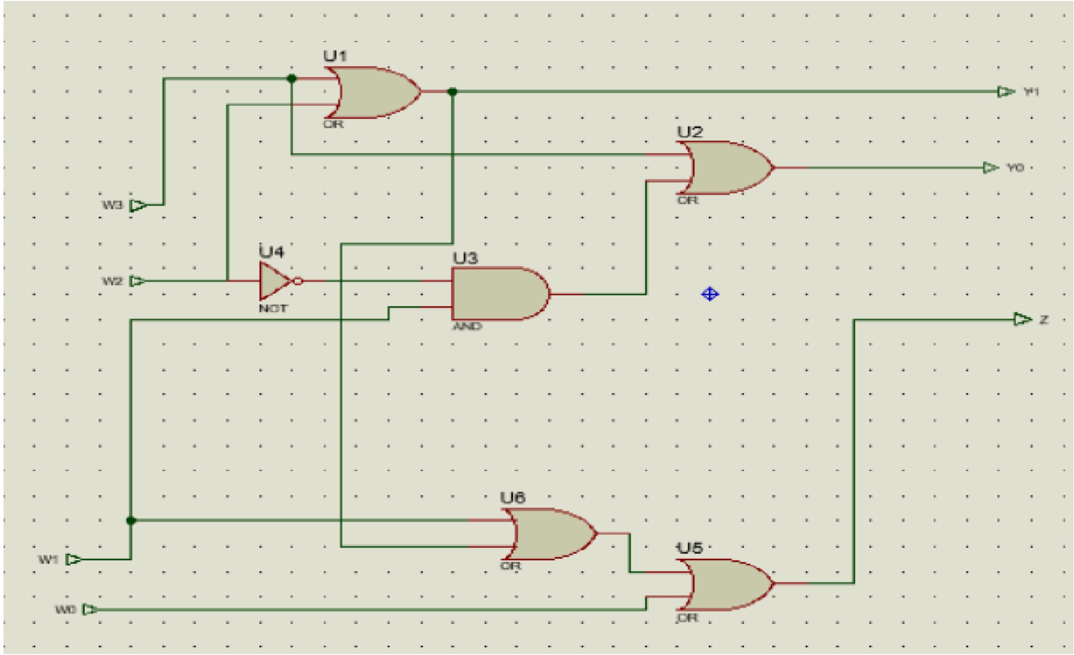
Simulation of Lab 4

Part 1: 4-to-2 binary encoder

Parent Sheet

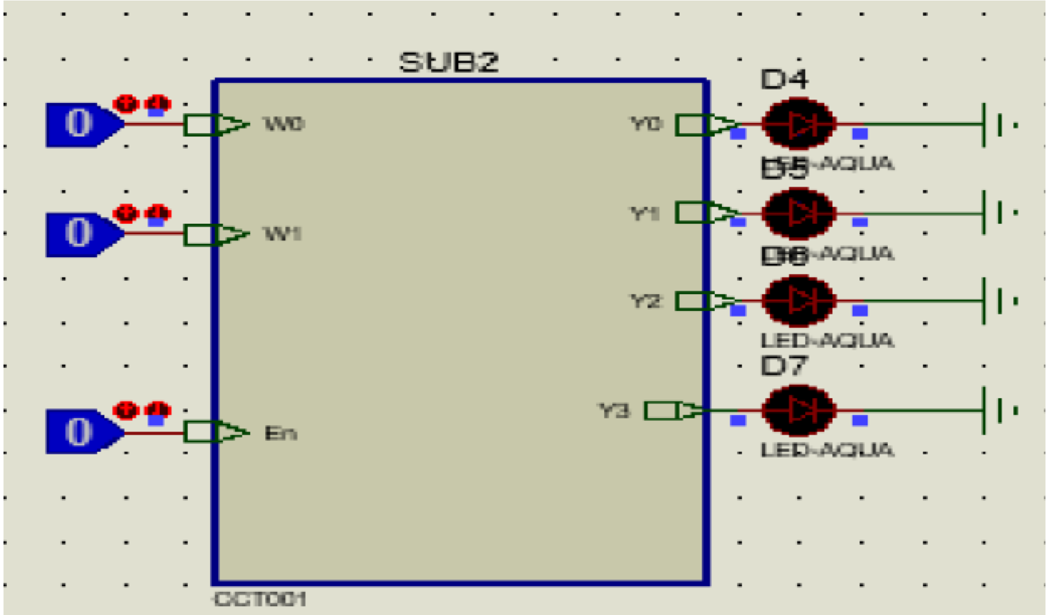


Child sheet

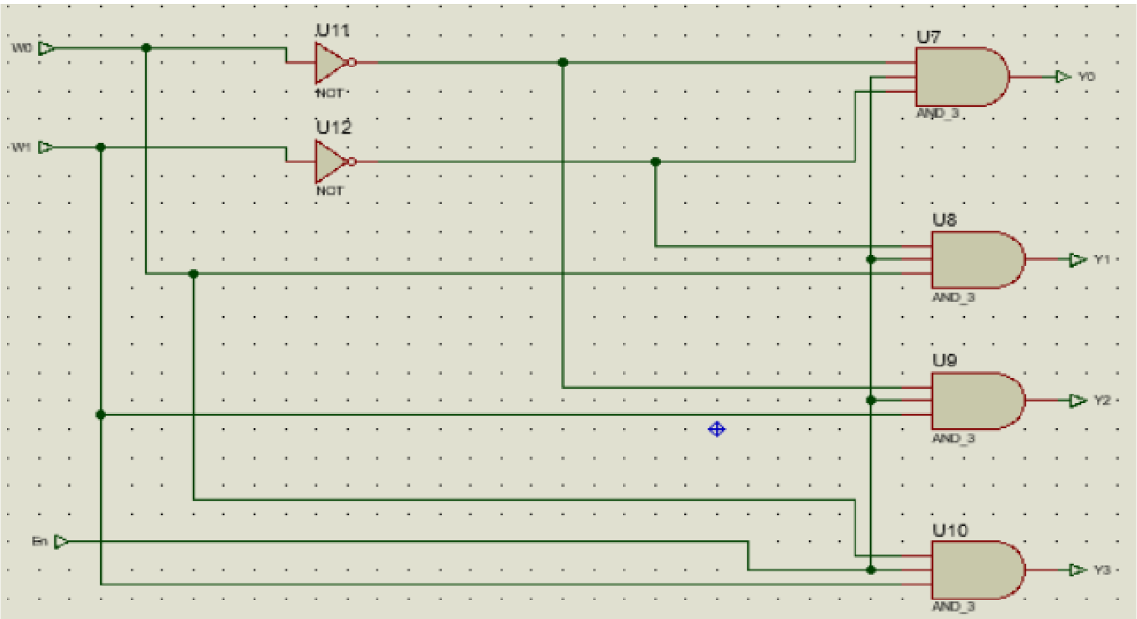


Part 2: 2-to-4 binary decoder

Parent sheet

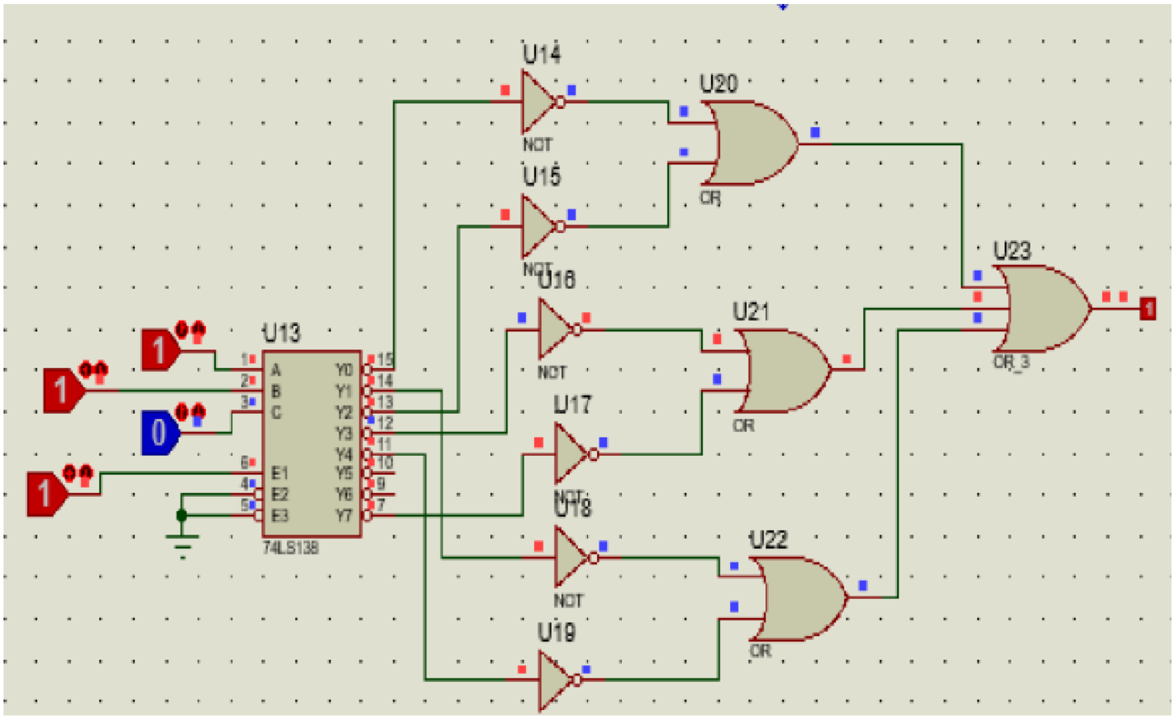


Child sheet



Part 3: Implement the function by using a 3-to-8 binary decoder and an OR gate.

$f\,(w1, w2, w3) = \sum m(0, 1, 3, 4, 6, 7)$



BRAC UNIVERSITY

Department of Electrical and Electronic Engineering

EEE/ECE 302/301LL: Digital Electronics/Digital Logic Design

Experiment No. 5

Design and Implementation of Binary Adders and Subtractors

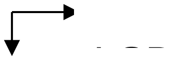
1. **Objective:** The objective of this experiment is to study and implementation of Binary Adders and Subtractors.

1. **Equipment Required:**
   - AT-700 Trainer Board
   - 7483 (4-bit Full Adder) – 1 pc
   - 7486 (Quad XOR Gate) – 1 pc
   - 7432 (Quad OR Gate) – 1 pc
   - 7408 (Quad AND Gate) – 1 pc

2. **Theoretical Background:**

The addition of two binary numbers is performed in exactly the same manner as the addition of decimal numbers.

Let us first review the decimal addition



```
3         7         6
4         6         1
-----------------------
8         3         7
```

The least significant digit position is operated on first, producing a sum of 7. The digits in the second position are then added to produce a sum of 13, which produces a **carry** of 1 into the third position. This produces a sum of 8 in the third position.

The same general steps are followed in binary addition. However only four cases can occur in adding the two binary digits (bits) in any position. They are

0+0=0
1+0=1
1+1=10=0+carry of 1 into the next position
1+1+1=11=1+carry of 1 into the next position

Here are several examples of the addition of two binary numbers:

```
  1001        1101
  1111        0110
  -------     --------
 11000        10011
```

- **Full adder:** A full adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by x and y represent the two significant bits to be added. The third input z represents the carry from the previous lower significant position. The two outputs are designed by the symbols S and C. The binary S gives the value of the least significant bit of the sum. The binary variable C gives the output carry.

  The truth table of the full adder is as follows:

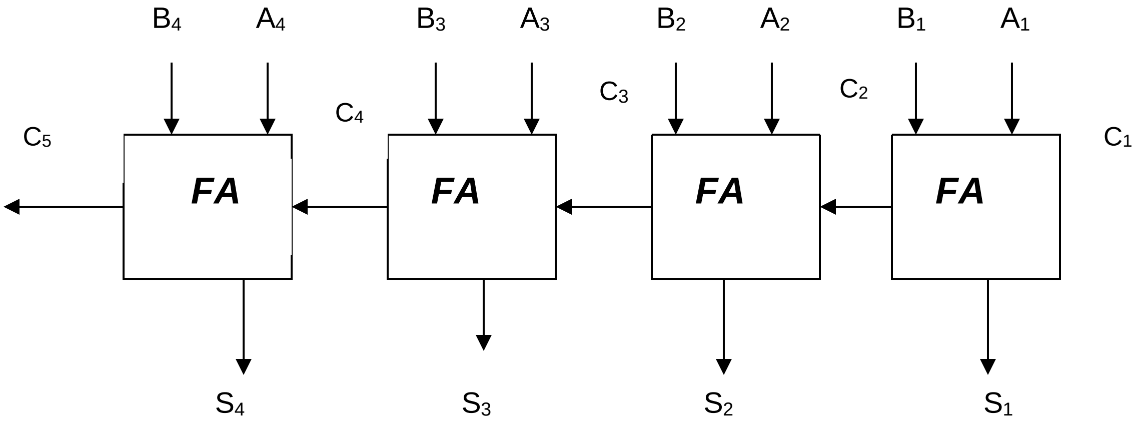
A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

From the truth table, we get:

$S = A \oplus B \oplus Cin$ and $Cout = A.B + B.Cin + A.Cin$

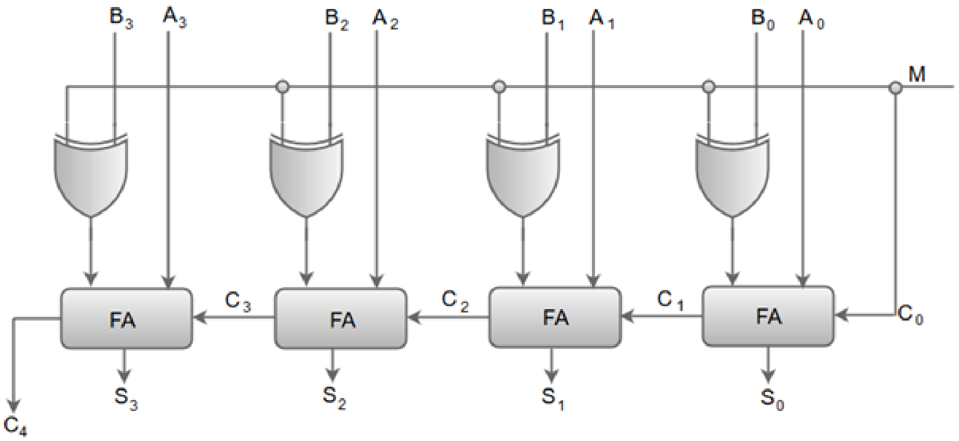
- **A 4-bit full adder:**

  A binary parallel adder is a digital function that produces the arithmetic sum of two binary numbers in parallel. It consists of full adders connected in cascade, with the output carry from one full adder connected to the input of the next full adder.



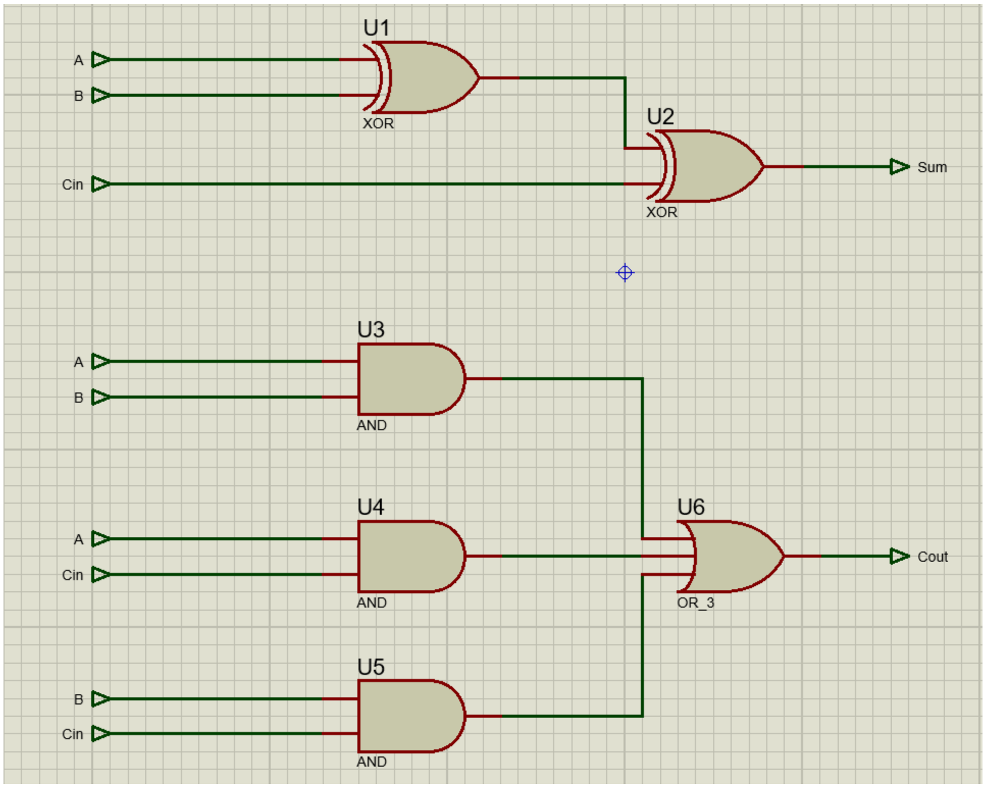
- **A 4-bit adder-subtractor:**

  An adder-subtractor is a logic circuit which adds two bits when an operation select pin (M) is low and performs subtraction when M is high. The circuit to implement a four bit adder-subtractor is:



M=0 gives the output S=A+B and M=1 gives the output S=A-B. If the result of subtraction is negative, the result is found in 2's complement form.

3. Circuit Diagram/ Experimental setup:



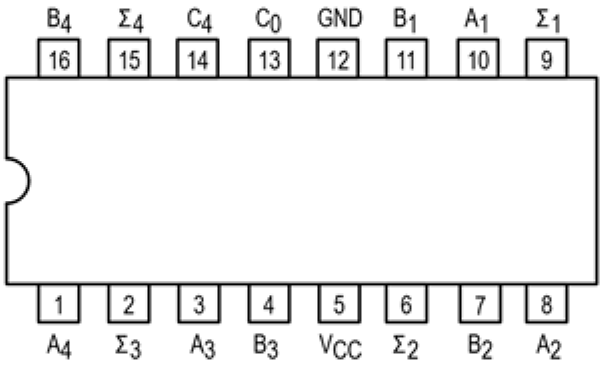
4. Data Table:

Verification of Full Adder:

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

5. Lab work:

74LS83 pinout



Vcc	5.5V max, 5V Typical
A_1-A_4	Operand A Inputs
B_1-B_4	Operand B Inputs
C_0	Carry Input
$\Sigma_1-\Sigma_4$	Sum Outputs (Note b)
C_4	Carry Output (Note b)

- Build a circuit for 4-bit Full Adder.
- Using 7483(4-bit full adder) and 7486(Quad XOR Gate), design a 4 bit adder subtractor.

Confirm the successful breadboard implementation

Faculty Signature and Date

Software Simulation (*Experiment No. 5 Cont'd*)

1. **Objective:** The objective of this experiment is to study and implementation of Binary Adders and Subtractors using Proteus software.

2. **Problem statement:** Design a full adder circuit using the fundamental and exclusive gates. Then build a 4-bit full adder circuit and adder subtractor circuits using the Proteus software.

3. **Software and Device requirement:**

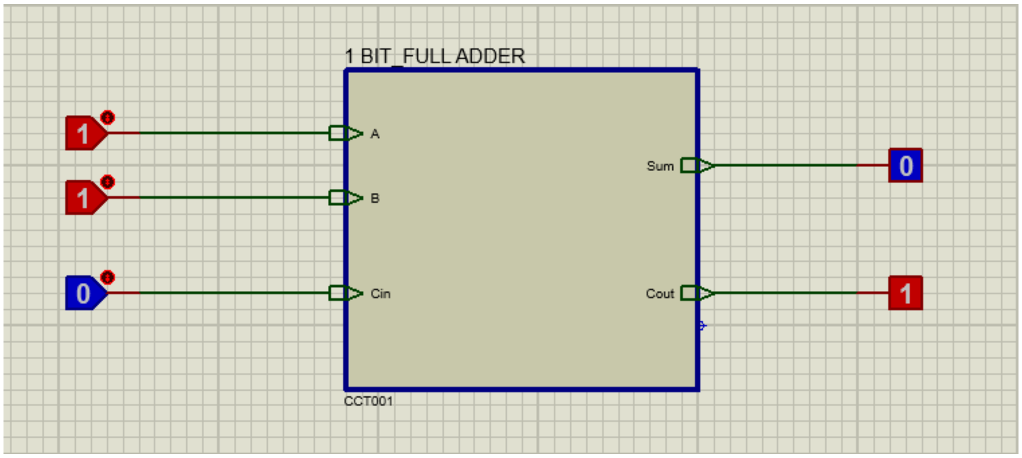
   Proteus 8 Professional

   Minimum PC specifications:
   Windows/ Mac: Microsoft® Windows® 7 Professional, Enterprise, Ultimate or Home Premium (64-bit); Windows 8 (64-bit) (All Service Packs); Windows 10 (64-bit); Windows 2008 R2 Server; Windows 2012 *Server* (All Service Packs).
   Ram: 2 GB
   Processor: Intel® Pentium® 4 or AMD Athlon XP 2000 with multi-core CPU
   Display resolutions: 1,024 x 768 display resolution with true color (16-bit color)

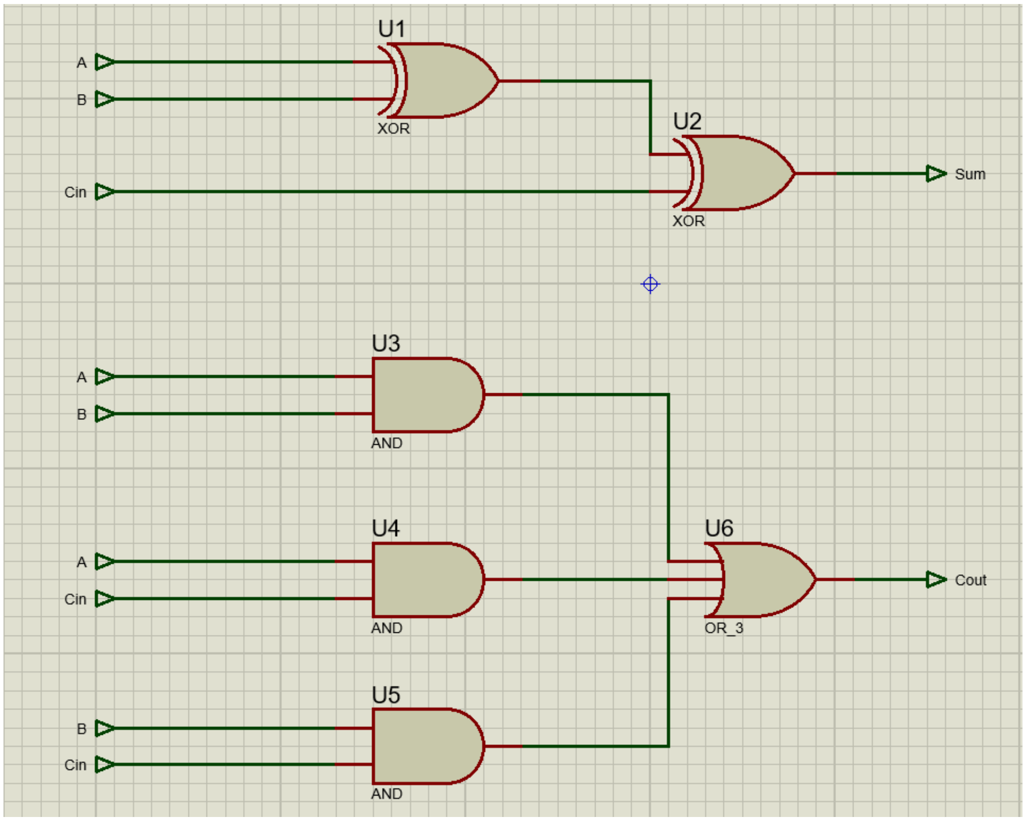
4. **Circuit Diagram/ Experimental setup:**

1-bit Full Adder

- **Parent Sheet Diagram**



- **Child Sheet Diagram**



5. Procedures:

- At first, select the 'Component Mode' and choose 'Pick Devices' and take the following components from the library: 3 AND gates, 2 X-OR gates, 1 OR gate (3-input terminals).
- Design the circuits following the child sheet.
- To provide the input/output terminals, go to the 'Terminals Mode' and choose the 'INPUT' and 'OUTPUT' and give the names to the terminals.
- Now, go to the 'Subcircuit Mode' and create a Subcircuit, then cut the previously built circuit and paste it into the child sheet of the Sub circuit.
- Give the input/output terminals from the Subcircuit Mode and give the similar name of the child sheet input/output terminals.
- Finally, go to the 'Component Mode' and pick devices named 'LOGICPROBE' and 'LOGICTOGGLE' to see the input/output bits.

6. Lab work:

- Build a circuit for 4-bit Full Adder using Proteus Software.
- Design a 4 bit Adder Subtractor using 4-bit Full Adder and X-OR gates in Proteus Software.

Confirm the successful circuit implementation

Faculty Signature and Date

7. Lab report directions/ Questions/ Discussions/ Assignments:

The report should cover the followings

8) Name of the Experiment
9) Objective
10) Apparatus
11) Experimental Setup (Show the connection diagram with IC pinouts)
12) Results (Truth Table)
13) Questions and answers
    - Design a 4-bit full adder using the fundamental and exclusive gates.
    - Using 4-bit Full Adder and XOR Gates, design a 4-bit adder subtractor.
14) Discussion

BRAC UNIVERSITY

Department of Electrical and Electronic Engineering

EEE/ECE 302/301LL: Digital Electronics/Digital Logic Design

Experiment No. 6

Implementation of Counters

7. **Objective:** This experiment is intended to build 4 bit synchronous counter and BCD counter with 74161 IC
8. **Introduction/ Theoretical Background/ Problem Statement:**

   Counter circuits are used in digital systems for many purposes. They may count the number of occurrences of certain events, generate timing intervals for control of various tasks in a system, keep track of time elapsed between specific events, and so on.

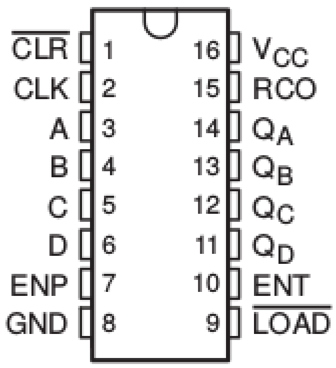
   Counters can be implemented using the adder/subtractor circuits. However, since we only need to change the contents of a counter by 1, it is not necessary to use such elaborate circuits. Instead, we can use much simpler circuits that have a significantly lower cost.

   Counters can be primarily classified into two categories:

   Asynchronous Counter: Asynchronous counters are those whose output is free from the clock signal. Because the flip flops in asynchronous counters are supplied with different clock signals, there may be delay in producing output. The required number of logic gates to design asynchronous counters is very less. So they are simple in design.

   Synchronous Counter: Synchronous generally refers to something which is coordinated with others based on time. Synchronous signals occur at same clock rate and all the clocks follow the same reference clock. When all the flip-flops are triggered by same clock pulse, then the counter is called synchronous counter.

   In this experiment, we use 74161 IC which is a 4 bit synchronous binary counter to observe the count with respect to clock pulses. We then design a BCD Counter using 74161.



Pinout of 74161

9. Components/ Equipment/ Tools/ Requirements/ Software requirements: (*If applicable*)

- AT-700 Trainer Board
- 74161 (4-bit Synchronous Counter) – 1 pc
- 7400 (Quad NAND Gate) – 1 pc

10. Circuit Diagram/ Experimental setup:

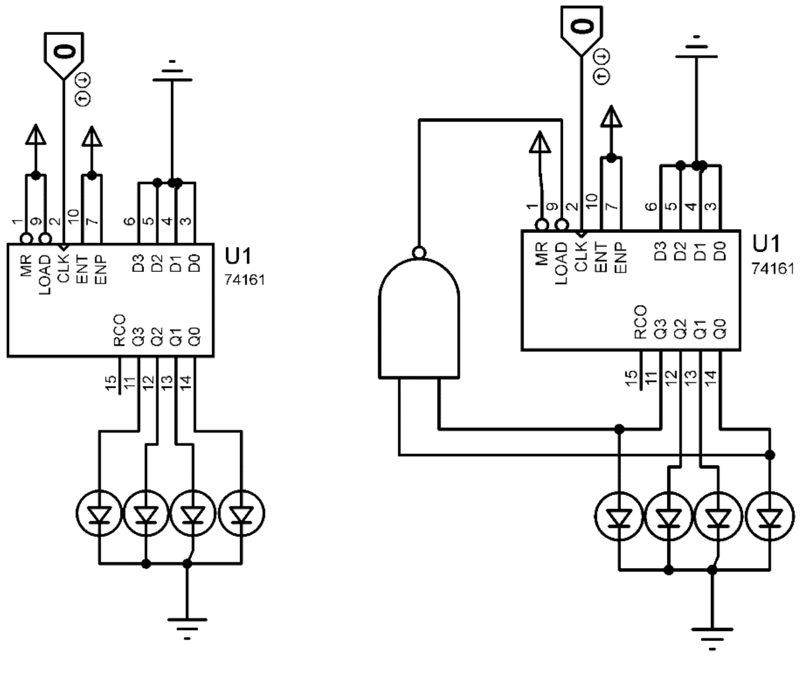


Figure: (a) 4 bit binary counter, (b) BCD Counter

11. Procedure:

Connect the above given circuits and verify their working as 4 bit binary counter and BCD counter. Show the truth table.

Faculty Signature and Date

12. Lab report directions/ Questions/ Discussions/ Assignments:

1. Name of the Experiment
2. Objective
3. Apparatus
4. Experimental Setup (Show the connection diagram with IC pinouts)
5. Results (Truth Table)
6. Questions and Answers: Design the 4 bit mod-16 counter to perform the following counting operations.
    a. Operation as a Mod-8 Counter (0000 to 0111)
    b. Operation as an Odd Sequence Counter – 1, 3, 5, 7, … , F (0001, 0011, 0101, 0111, …. , 1111)
    c. Operation as an Even Sequence Counter – 0, 2, 4, 6, , … , E (0000, 0010, 0100, 0110, …. , 1110)
7. Discussion

Software Simulation (*Experiment No. 6 Cont'd*)

1. Objective: This part is intended to illustrate the simulation procedures for building 4 bit synchronous counter and BCD counter with 74161 IC

2. Problem statement: Use Proteus software for building 4 bit synchronous counter and BCD counter with 74161 IC and verify its working

3. Software and Device requirement:

Proteus
Minimum PC specifications:
Windows/ Mac: Microsoft® Windows® 7 Professional, Enterprise, Ultimate or Home Premium (64-bit); Windows 8 (64-bit) (All Service Packs); Windows 10 (64-bit); Windows 2008 R2 Server; Windows 2012 *Server* (All Service Packs).
Ram: 2 GB
Processor: Intel® Pentium® 4 or AMD Athlon XP 2000 with multi-core CPU
Display resolutions: 1,024 x 768 display resolution with true color (16-bit color)

4. Circuit Diagram/ Experimental setup:

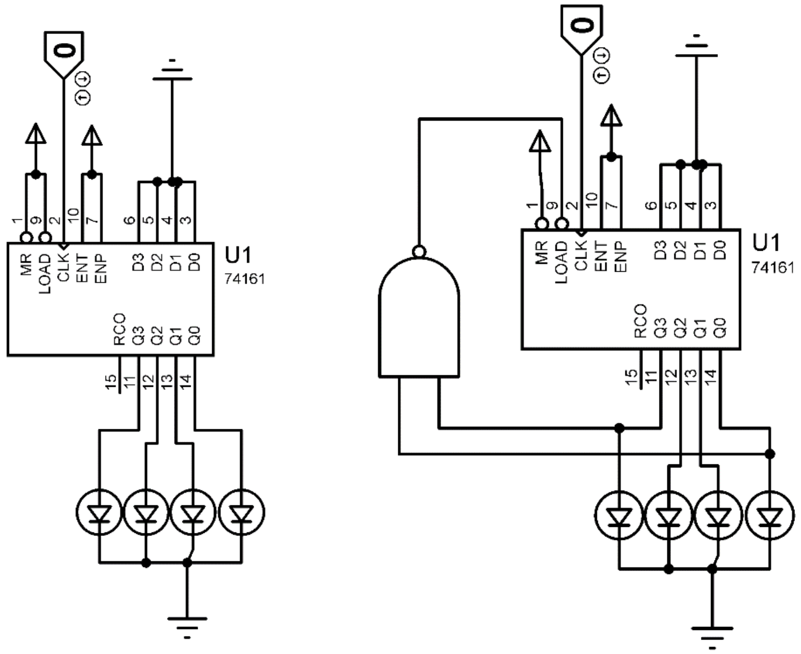


Figure: (a) 4 bit binary counter, (b) BCD Counter

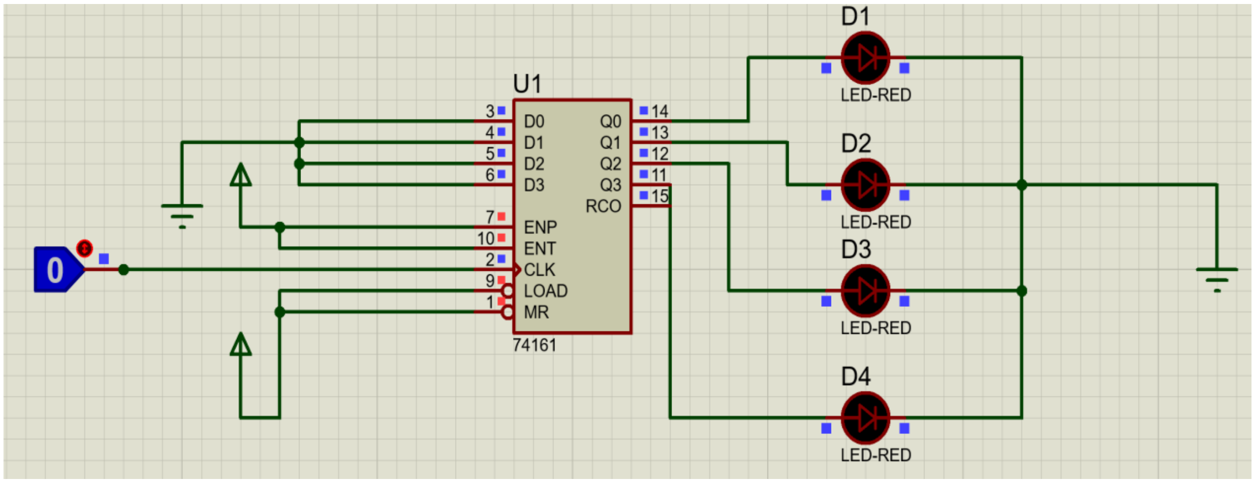


Fig: Simulation Setup of a 4 bit synchronous counter

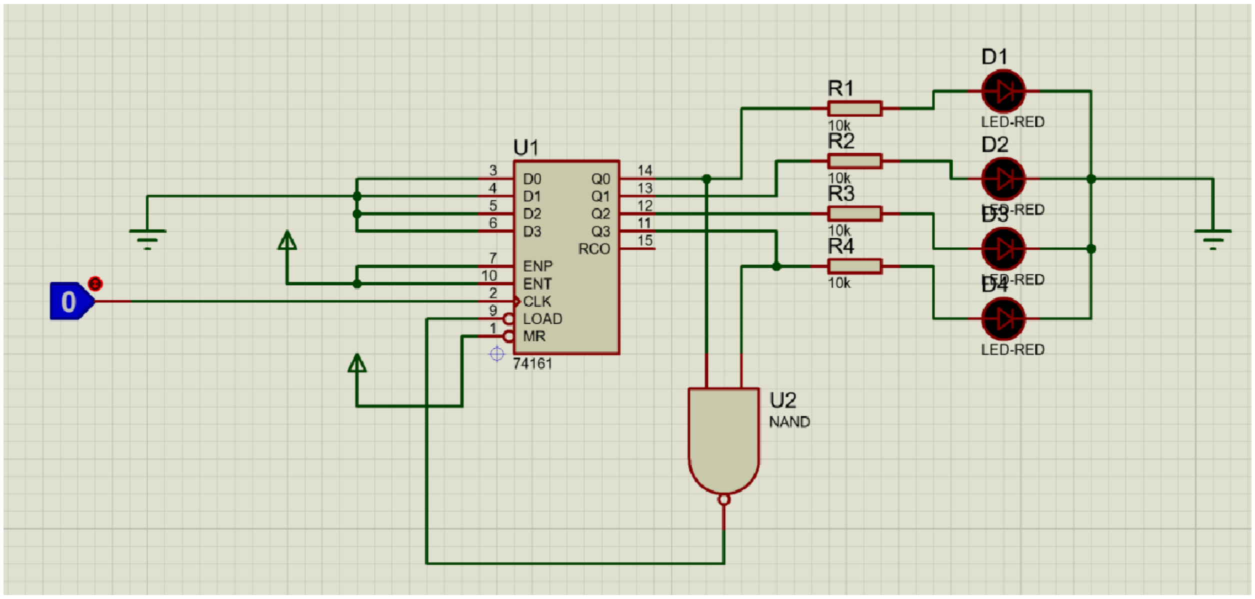


Fig: Simulation Setup of a BCD Counter that can count from 0 to 9

5. Procedure:

i) Open Proteus and Bring the components required for this experiment from library

ii) Connect all the components with wire

iii) Bring logic states, logic probe and logic toggle buttons to give input and connect them with the circuit

iv) Verify the working of designed counter

Faculty Signature and Date

BRAC UNIVERSITY

Department of Electrical and Electronic Engineering
EEE/ECE 302/301LL: Digital Electronics/Digital Logic Design
Open Ended Project/Assignment

Marks 200

Problem Statement: Suppose you are the administrator of BracU Optics and Photonics-OPTICA student chapter. You are advised to develop a majority voting system for electing the 2022 Executive body using logic gate, synchronous, combinational and arithmetic circuit etc. Consider you have 10 participant and 3 contestant. Three contestant is running for President, Vice President and Secretary Position. Develop a system so that majority vote wins and show the winner in the LCD display.

In addition to previous design you need to develop a timer circuit (stopwatch) with a resolution of 0.1 sec so that you can keep track of the election happening time. Your both of the subsystem should have at least three button such as start, stop and reset.

In your design/report you must address the following:

A) Specification & Requirement

B) Show the subsystem level block diagram.

C) Show working flowchart

D) Algorithm development procedure and related calculation, K-map, truth table, timing diagram.

E) Methodology

F) Result/Interpretation with test cases.

Instruction: You need to prepare a summary report addressing the mentioned criteria. Also you need to demonstrate your simulation (proteus) and hardware. Details of lab project demonstration schedule will be discussed in the class.

PO related KPI and rubrics:

PO(j)	KPI	Marks	Assessment Tools
Communication: Communicate effectively on complex activities with the (*electrical and electronic / electronic and communication)* engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.	Write effective technical reports, prepare assignments, design documentation etc. on complex engineering activities	40	Presentation/Report
	Prepare multi-media presentations, posters on engineering activities	30	
	Provide clear instructions to others and receive instructions from others on engineering activities	10	
	Deliver effective oral presentations, participate in technical discussions on complex engineering activities	20	

PO(i)	KPI	Marks	Assessment Tools
Individual work and teamwork: Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.	Fulfil duties as an individual, a team member or leader in a multidisciplinary environment	50	Peer Review
	Communicate effectively with other team members	25	
	Share responsibilities by performing research and information gathering activities, participating in report writing and team presentations etc.	25	

I. ***Individual work and teamwork:*** Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.

Performance Criteria	**Performance Measure**			
	Unsatisfactory (1)	**Developing (2)**	**Satisfactory (3)**	**Exemplary (4)**
1. Fulfil duties as an individual, a team member or leader in a multidisciplinary environment	Does not perform any duties of assigned team role	Performs very little duties	Performs nearly all duties	Performs all duties of assigned team role
0. Communicate effectively with other team members.	Rarely performs the required communication and always depends on others' directions	Performs insufficient amount of communication and depends on others' directions	Performs the required communication but depends on others' directions	Always performs the required communication and takes initiatives and proactive roles

0. Share responsibilities by performing research and information gathering activities, participating in report writing and team presentations etc.	Never take responsibilities or participate in activities related to research and information gathering, writing reports or presentations	Perform the assigned work in some cases– often needs reminding	In most cases, complete the assigned task, contribute in research and information gathering activities, report writings, presentations etc. Rarely needs any reminding	Always does the assigned responsibilities. Enthusiastically complete the assigned activities including research and information gathering, report writing, participate in preparing presentations and presenting it

J. Communication: Communicate effectively on complex activities with *the (electrical and electronic / electronic and communication)* engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

Performance Criteria	**Performance Measure**			
	Unsatisfactory (1)	**Developing (2)**	**Satisfactory (3)**	**Exemplary (4)**
1. Write effective technical reports, prepare assignments, design documentation etc. on complex engineering activities	Poor organization of technical reports, homework. Sections out of order, too much handwritten copy, sloppy formatting. Sometime there is no title page. Undeveloped or irrelevant introduction, unclear about main points and conclusion. Numerous mistakes. Sources not referenced at al or documented with ill-formatting. Consistent problems with word choices and sentence structure, leaving the reader unsure of the meaning	Generally good organization of technical reports. Sections are in order, contains the minimum allowable amount of handwritten copy, formatting is rough but readable. Some of the main points and conclusion are not clear or sufficiently stressed. References are accurately documented but some lack of proper formatting. Words and sentences are adequate in general but lack energy; reader has to struggle to keep reading to the end; multiple grammatical mistakes.	Satisfactory organization of technical reports, clear introduction. Main points are well stated and clear conclusion. Some transitions are somewhat sudden. All sources are accurately documented. All sections are in order, formatting generally well but could still be improved. Good writing style with proper choice of words. Smooth and even flow of sentences. Very Few grammatical issues	Superb organization of technical reports. Clear introduction, Main points well stated and argued, with each leading to the next point of discussion, clear summary and conclusion. Sections are in order, well-formatted, very readable. All sources are accurately documented and properly formatted. Compelling writing style; connects strongly with the reader and keeps him or her engaged right to the end. No Grammatical issues

0. Prepare multi-media presentations, posters on engineering activities	Presentation is very blended. The pace and sequence of topics are organized in a random way that ultimately fail to lead up to any clear conclusions;	Some of the ideas are presented well; others are lacking; offers plausible conclusion(s);	Ideas are well organized and help the reader move along; the key points are presented but does not demonstrate in-depth understanding; leads up to convincing conclusion(s);	The presentations, posters are very clear and focused. Relevant and quality details give the audience and reader important information and opportunity to get *insight* into the topic.
0. Provide clear instructions to others and receive instructions from others on engineering activities	Unable to give and receive instructions	Can provide instructions to others and understand instructions received from others in irregular basis	Can provide instructions to others and understand instructions received from others in most cases	Can provide clear instructions to others and understand instructions received from others always.
0. Deliver effective oral presentations, participate in technical discussions on complex engineering activities	Very poor delivery, hard to understand or inaudible. Reads most of the presentation from slides or notes with no eye contact. Not aware of audience reactions	Occasionally inaudible, low voice of delivery. Occasional eye contact with audience but mostly reads presentation. Some distracting filler words and gestures in the delivery Some awareness of audience reactions. Very brief responses to the questions	Good clear voice, generally effective delivery; minimal distracting gestures, etc., but somewhat monotone; generally aware of the reactions of the audience; maintains good eye contact with the audience while speaking and answering questions	Very clear, natural, confident delivery that conveys the message, excellent use of volume and pace. Keenly aware of the reactions of the audience. Always engaged the audiences throughout the presentation; modifies material as required based on the questions asked by the audience and their comments.