ARMY INSTITUTE OF TECHNOLOGY

LAB MANUAL

Name of Subject	**Logic Design and Computer Organization Laboratory**
Department	**Information Technology**
Subject Code	**214446**
TW/OR/PR	**PR/TW**
Marks	**50/25**
Course (Pattern)	**2019 Revised**
Year	**SE**
Semester	**1**

List of Experiments

(As given by SPPU)

Additional List of Experiments

(Over and above SPPU Syllabus)

Sr. No	Title	Page No
1.	Implement 4 bit binary to gray code converter.	
2.	Implement 4 bit gray to binary code converter.	
3	Design and implement 4-bit BCD to Excess-3 code	
4	Design and implement 1 digit BCD adder using IC 7483	
5	Realization of Boolean Expressions using IC 74153 for a. $F(A,B,C) = \sum m(0,1,3,6,7)$ b. $F(A,B,C,D) = \sum m(0,2,5,,7,8,10,13,15)$	
6	Implement $F(A,B,C) = \sum m(0,2,4,5,7)$ using MUX tree method.	
7	Design Latch using NAND & NOR Gate	
8		

Pre-requisite for Lab

Prerequisite Courses/Lab	
Code	Subject
	Basic Electronics Engineering

Learning Objectives

Sr. No	Objective
1	To learn and understand basic digital design techniques.
2	To learn and understand design and construction of combinational and sequential circuits.
3	

Course Outcomes:

After the completion of this course students will be able to:-

Sr. No.	Outcomes
1	**CO1 :** Use logic function representation for simplification with K-Maps and design Combinational logic circuits using SSI & MSI chips.
2	**CO2:** Design Sequential Logic circuits: MOD counters using synchronous counters.
3	**CO3:** Understand the basics of simulator tool & to simulate basic blocks such as ALU & memory.

Precautions / Lab Safety Do's and Don'ts/ Lab Etiquettes

Sr. No	Details
1	The use of personal audio or video equipment is prohibited in the laboratory
2	Keep work area neat and free of any unnecessary objects.
3	Always perform the experiments as directed by your instructor.
4	Never work in the laboratory without the supervision of an instructor.

Evaluation Guidelines (Rubrics)

(Similar guidelines pertaining to each subject (lab) shall be used to evaluate **each experiment** performed in the lab)

Grade	Poor	Average	Good	Outstanding
Marks	0-3	4-5	6-8	9-10

Criteria	**Grade**		**Marks**
Set-up and Equipment Care	Poor (0-3)	Set-up of equipment is not accurate, help is required with several major details	
	Average (4-5)	Set-up of equipment is generally workable with several details that need refinement	
	Good (6-8)	Set-up of equipment is generally accurate with 1 or 2 small details that need refinement	
	Outstanding (9-10)	All equipments accurately placed	
Following Procedure	Poor (0-3)	Lacks appropriate knowledge of the lab procedures. Often requires help from the teacher to even complete basic procedures	
	Average (4-5)	Demonstrates general knowledge of lab procedures. Requires help from the teacher with some steps in procedures.	
	Good (6-8)	Demonstrates good knowledge of the lab procedures. Will discuss with peers to solve problems in procedures.	
	Outstanding (9-10)	Demonstrates very good knowledge of the lab procedures. Gladly helps other students to follow procedures.	
Data Collection	Poor (0-3)	Measurements are incomplete, inaccurate and imprecise. Observations are incomplete or not included. Symbols, units and significant figures are not included.	
	Average (4-5)	Measurements are somewhat inaccurate and imprecise. Observations are incomplete. There are 3 or more minor errors using symbols, units and significant digits or 2 major errors	
	Good (6-8)	Measurements are mostly accurate. Observations are generally complete. Work is organized. Only 2 or 3 minor errors using symbols, units and significant digits.	
	Outstanding (9-10)	Measurements are both accurate and precise. Observations are very thorough and may recognize possible errors in	

		data collection. Work is neat and organized. Includes appropriate symbols, units and significant digits.	
Analysing and Concluding	Poor (0-3)	Provides limited analysis of the data. Demonstrates limited ability to draw conclusions based on the data.	
	Average (4-5)	Provides some analysis of the data. Demonstrates some ability to draw conclusions based on the data.	
	Good (6-8)	Provides sufficient analysis of the data. Draws valid conclusions based on the data.	
	Outstanding (9-10)	Provides rich analysis of the data. Draws insightful conclusions based on the data.	
Safety	Poor (0-3)	Proper safety precautions are consistently missed. Needs to be reminded often during the lab.	
	Average (4-5)	Proper safety precautions are often missed. Needs to be reminded more than once during the lab.	
	Good (6-8)	Proper safety precautions are generally used. Uses general reminders of safe practices independently.	
	Outstanding (9-10)	Proper safety precautions are consistently used. Consistently thinks ahead to ensure safety. Will often help other students to conduct labs safely.	
Specifications (Computer Program)	Poor (0-3)	The program is producing incorrect results.	
	Average (4-5)	The program produces correct results but does not display them correctly.	
	Good (6-8)	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	
	Outstanding (9-10)	The program works and meets all of the specifications.	
Readability (Computer Program)	Poor (0-3)	The code is poorly organized and very difficult to understand.	
	Average (4-5)	The code is readable only by someone who knows what it is supposed to be doing	
	Good (6-8)	The code is fairly easy to understand.	
	Outstanding (9-10)	The code is exceptionally well organized and very easy to follow.	
Reusability (Computer Program)	Poor (0-3)	The code is not organized for reusability.	
	Average (4-5)	Some parts of the code could be reused in other programs.	
	Good (6-8)	Most of the code could be reused in other programs.	

	Outstanding (9-10)	The code could be reused as a whole, or each routine could be reused.	
Documentation (Computer Program)	Poor (0-3)	The documentation is simply comments embedded in the code and does not help the reader understand the code.	
	Average (4-5)	The documentation is simply comments embedded in the code with some simple header comments separating routines	
	Good (6-8)	The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code.	
	Outstanding (9-10)	The documentation is well written and clearly explains what the code is accomplishing and how.	
Timely submission (Computer Program)	Poor (0-3)	The code was more than 2 weeks overdue.	
	Average (4-5)	The code was within 2 weeks of the due date.	
	Good (6-8)	The program was delivered within a week of the due date.	
	Outstanding (9-10)	The program was delivered on time	
Efficiency (Computer Program)	Poor (0-3)	The code is huge and appears to be patched together.	
	Average (4-5)	The code is brute force and unnecessarily long	
	Good (6-8)	The code is fairly efficient without sacrificing readability and understanding.	
	Outstanding (9-10)	The code is extremely efficient without sacrificing readability and understanding	

Experiment No.1

Title: Implement 4 bit BCD to Excess-3 code converter & Vice-versa.

Aim: Design (truth table, K-map) and implementation of 4-bitBCD to Excess-3 Code converters.

Objectives:

- To understand the concept of number system.
- To study and implement the BCD, Excess-3, Gray Code & Kmap .

Hardware Required:

Digital Trainer Kit, Gates Required :- AND (7408),OR (7432), EXOR(7486), NOT(7404),Number of Patch Chords

Theory:

Explanation:

There is a wide variety of binary codes used in digital systems. Some of these codes are binary- coded -decimal (BCD), Excess-3, Gray, octal, hexadecimal, etc. Often it is required to convert from one code to another. For example the input to a digital system may be in natural BCD and output may be 7-segment LEDs. The digital system used may be capable of processing the data in straight binary format. Therefore, the data has to be converted from one type of code to another type for different purpose. The various code converters can be designed using gates.

1. **BCD Code:**

Binary Coded Decimal (BCD) is used to represent each of decimal digits (0 to 9) with a 4-bit binary code. For example $(23)_{10}$ is represented by 0010 0011 using BCD code rather than$(10111)_2$ This code is also known as 8-4-2-1 code as 8421 indicates the binary weights of four bits(2^3, 2^2, 2^1, 2^0). It is easy to convert between BCD code numbers and the familiar decimal numbers. It is the main advantage of this code. With four bits, sixteen numbers (0000 to 1111) can be represented, but in BCD code only 10 of these are used. The six code combinations (1010 to 1111) are not used and are invalid.

Applications: Some early computers processed BCD numbers. Arithmetic operations can be performed using this code. Input to a digital system may be in natural BCD and output may be 7-segment LEDs.

It is observed that more number of bits are required to code a decimal number using BCD code than using the straight binary code. However in spite of this

disadvantage it is very convenient and useful code for input and output operations in digital systems.

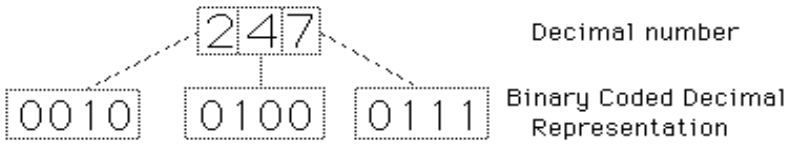


Fig. 3 BCD Coded Decimal Representation

2. EXCESS-3 Code:

Excess-3, also called XS3, is a non-weighted code used to express decimal numbers. It can be used for the representation of multi-digit decimal numbers as can BCD. The code for each decimal number is obtained by adding decimal 3 and then converting it to a 4-bit binary number. For e.g. decimal 2 is coded as 0010 + 0011 = 0101 in Excess-3 code.

This is self-complementing code which means 1's complement of the coded number yields 9's complement of the number itself. Self-complementing property of this helps considerably in performing subtraction operation in digital systems, so this code is used for certain arithmetic operations.

BCD To Excess – 3 Code Conversion:

Convert BCD 2 i. e. 0010 to Excess – 3 code
For converting 4 bit BCD code to Excess – 3, add 0011 i. e. decimal 3 to the respective code using rules of binary addition.

0010 + 0011 = 0101 – Excess – 3 code for BCD 2

Excess – 3 Code To BCD Conversion:

The 4 bit Excess-3 coded digit can be converted into BCD code by subtracting decimal value 3 i.e. 0011 from 4 bit Excess-3 digit.

e.g. Convert 4-bit Excess-3 value 0101 to equivalent BCD code.

0101-0011= 0010- BCD for 2

A. BCD To Excess-3 Code Conversion:

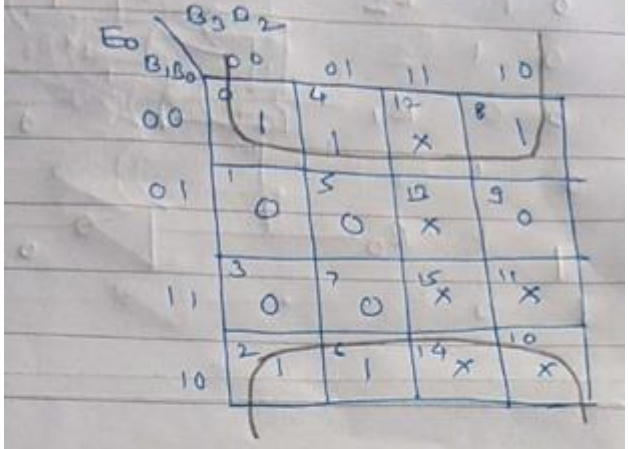
Truth Table:

INPUT (BCD CODE)				**OUTPUT (EXCESS-3 CODE)**			
B_3	**B_2**	**B_1**	**B_0**	**E_3**	**E_2**	**E_1**	**E_0**
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
1	0	1	0	x	x	X	x
1	0	1	1	x	x	X	x
1	1	0	0	x	x	X	x
1	1	0	1	x	x	X	x
1	1	1	0	x	x	X	x
1	1	1	1	x	x	X	x

2) K-Map For Reduced Boolean Expressions Of Each Output:

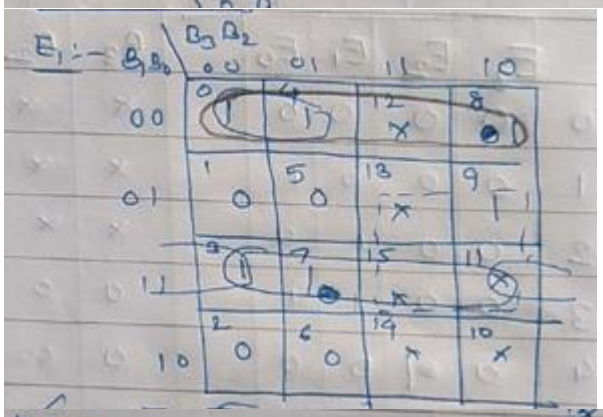
Kmaps

E_0



$E_0 = \overline{B_0}$

E_1 :- B_1B_0



Kmaps :- $E_1 = \bar{B}_1\bar{B}_0 + B_1B_0$

$= \overline{B_1 \oplus B_0} = B_1 \odot B_0$

$E_1 = B_1 \odot B_0$

E_2 :- B_1B_0 / B_3B_2

	00	01	11	10
00	0	4 X	12 X	8 0
01	1	5 0	13 X	9 1
11	3 1	7 0	15 X	11 X
10	2 1	6 0	14 X	10 X

$E_2 = B_2\bar{B}_1\bar{B}_0 + \bar{B}_2B_0 + \bar{B}_2B_1$

$= B_2\bar{B}_1\bar{B}_0 + \bar{B}_2(B_1 + B_0)$

$= B_2\overline{(B_1 + B_0)} + \bar{B}_2(B_1 + B_0)$

$\because B_1 \oplus B_0 \cdot \overline{B_1 + B_0} = \bar{B}_1 \cdot \bar{B}_0$

$E_2 = B_2 \oplus (B_1 + B_0)$

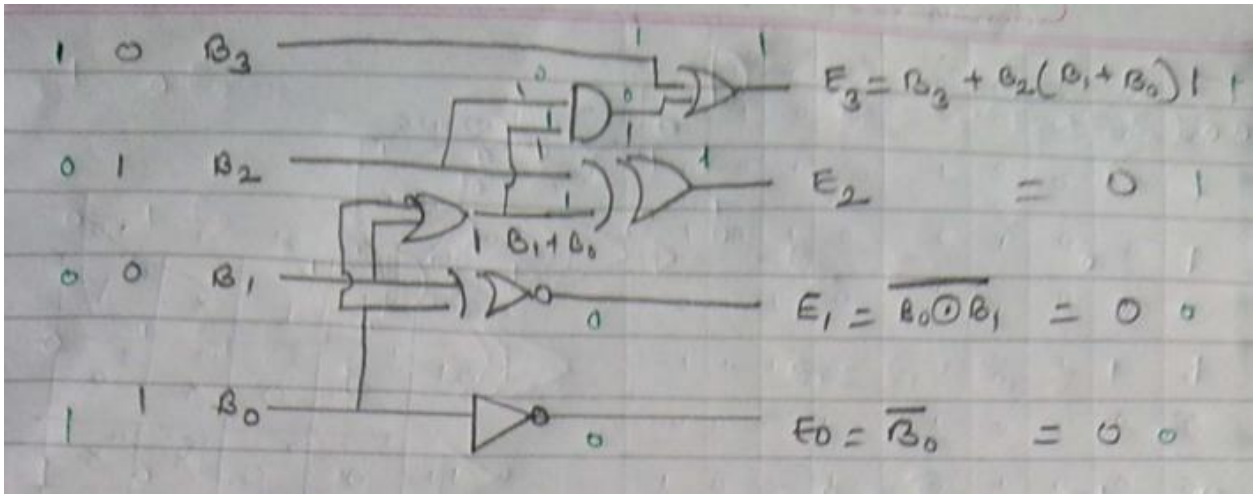
E_3 :- B_1B_0 / B_3B_2 (B_3)

	00	01	11	10
00	0 0	4 0	12 X	8 1
01	1 0	5 1	13 X	9 1
11	3 0	7 1	15 X	11 X
10	2 0	6 1	14 X	10 X

(B_0B_2 / B_1B_2)

$E_3 = B_3 + B_2B_1 + B_2B_0$

$E_3 = B_3 + B_2(B_1 + B_0)$

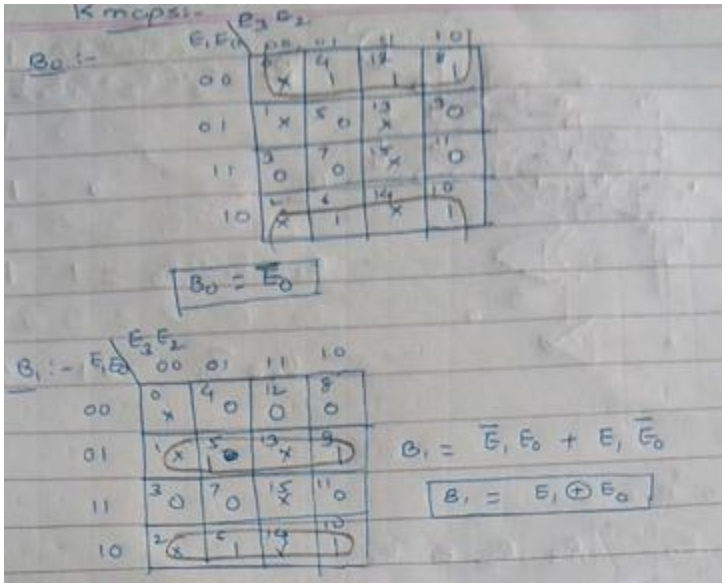


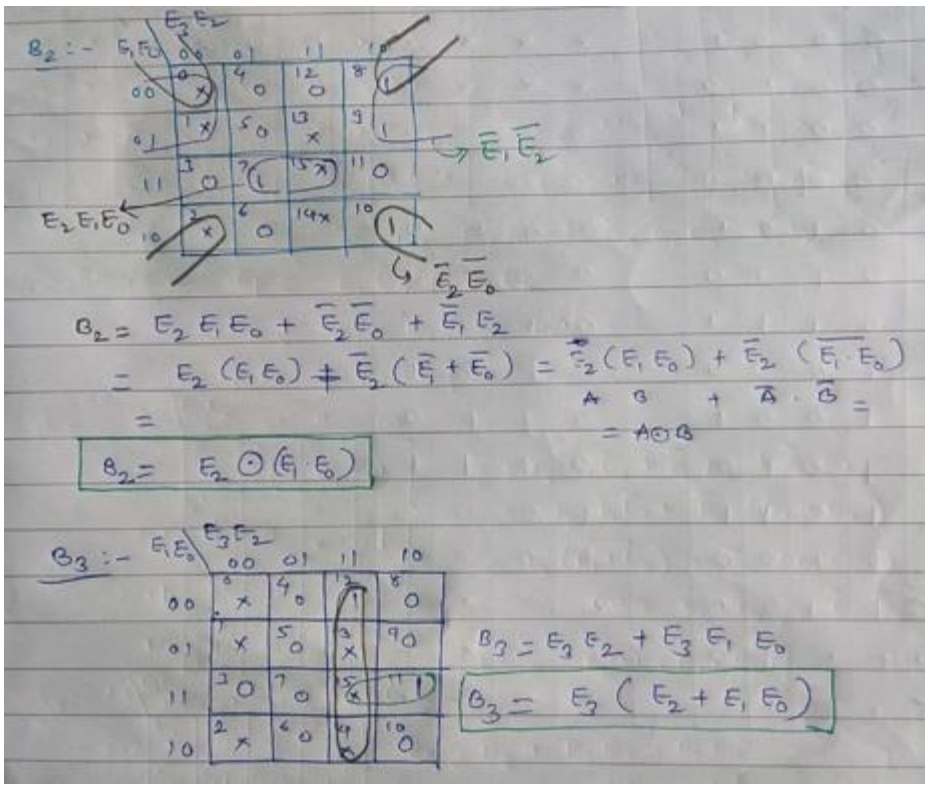
B. Excess-3 to BCD code converter

Excess-3 to BCD :-

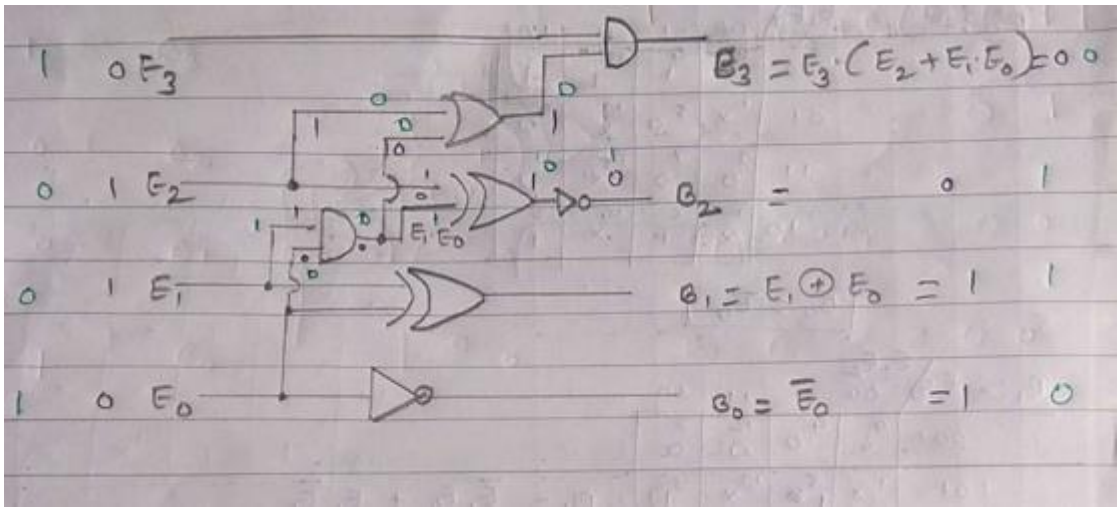
	E_3	E_2	E_1	E_0	B_3	B_2	B_1	B_0
0	0	0	0	0	x	x	x	x
1	0	0	0	1	x	x	x	x
2	0	0	1	0	x	x	x	x
3	0	0	1	1	0	0	0	0
4	0	0	0	0	0	0	0	1
5	0	1	0	1	0	0	1	0
6	0	1	1	0	0	0	1	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	0	1	0	1
9	1	0	0	1	0	1	1	0
10	1	0	1	0	0	1	1	1
11	1	0	1	1	1	0	0	0
12	1	1	0	0	1	0	0	1
13	1	1	0	1	x	x	x	x
14	1	1	1	0	x	x	x	x
15	1	1	1	1	x	x	x	x

Kmap reduction for Equations:-





Circuit Diagram for Excess-3 to BCD Code Converter:-



References:

- R.P. Jain, "Modern Digital Electronics", 3rd Edition,Tata McGraw-Hill, ISBN: 0-07-049492-4
- “Digital Design”, M Morris Mano, Prentice Hall, 3rd Edition, ISBN: 0130621218.

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	What is Gray Code?	1	1
2	Explain the Weighted & Non-Weighted Number System.	3	1
3	Which code can be used in Kmap?	1	1
4	What do you mean by Reflective & Self Complement Code?	1	1
5	Explain the applications of Gray Code?	3	1
6	What is Universal Gate?	1	1
7	Design Ex-OR gate from universal gate.	2	1
8	List down all the ICs used in Digital Laboratory.	3	1
9	Describe Do Not care combination.	2	1
10	What is Boolean algebra?	1	1
11	Why do we use Boolean algebra?	1	1
12	What is invalid BCD?	1	1

Guidelines for Students

- The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

- Every experiment must be included in the file in following format.

  a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

  b. **Theory**: Write brief theory related to practical.

  c. **Algorithm**: Write Algorithm for given task.

  d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

  e. **Output**: describe the results in few lines

  f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

  g. **Source Code**: Submit in the form of soft copies.

- Marking criteria.
  a. Experiment completion (Timely)
  b. Lab file (neatness and regularity)
  c. Viva (from time to time)
  d. Mock Practical Exam
  e. Exam (end term): Practical +Viva

Assessment Methodology

- Timely completion of assignment- 2marks
- Program demonstration- 4 marks
- Viva-voce -2 marks
- Timely submission of journal- 2 marks

Experiment No. 2

Title: Study of IC-74LS83 as a BCD adder.

Problem Definition & Aim of Experiment:

1. Design(truth table, K-map) and Implement 1 digit BCD adder using IC-74LS83

Objective of Experiment:

To understand how both arithmetic operations, addition in BCD can be performed by using IC 7483.

Lab Equipment's and IC's Used:

Digital Trainers Kit74LS83, 74LS08, 74LS32, 74LS04.

Theory:

BCD Adder:

BCD adder is a circuit that adds two BCD digits & produces a sum of digits also in BCD.

Rules for BCD addition:

1. Add two numbers using rules of Binary addition.
2. If the 4 bit sum is greater than 9 or if carry is generated then the sum is invalid. To correct the sum add 0110 i. e. (6)10 to sum. If carry is generated from this addition add it to next higher order BCD digit.
3. If the 4 bit sum is less than 9 or equal to 9 then sum is in proper form.

CASE I : Sum <= 9 & carry = 0.

Add BCD digits 3 & 4

1. 0 0 1 1
   \+ 0 1 0 0
   ---
   0 1 1 1

Answer is valid BCD number = **(7)BCD**& so 0110 is not added.

CASE II : Sum > 9 & carry = 0.

Add BCD digits 6 & 5

1. 0 1 1 0
   \+ 0 1 0 1
   ---
   10 1 1

Invalid BCD (since sum > 9) so 0110 is to be added

2. 1 0 1 1
   \+ 0 1 1 0
   ---
   1 0 0 0 1

(1 1)BCD

Valid BCD result = **(11) BCD**

CASE III : Sum < = 9 & carry = 1.

Add BCD digits 9 & 9

1. 1 0 0 1

\+ 1 0 0 1

---

1 0 0 1 0

Invalid BCD (since Carry = 1) so 0110 is to be added

2. 1 0 0 1 0
\+ 0 1 1 0

---

1 1 0 0 0

(1 8)BCD

Valid BCD result = **(18) BCD**

Design of BCD adder :

1. To execute first step i. e. binary addition of two 4 bit numbers we will use IC 7483
   (with Cin = 0), which is 4 bit binary adder.
2. We need to design a digital circuit which will sense sum & carry of IC 7483 & if sum exceeds 9 or carry = 1, this digital circuit will produce high output otherwise its output will be zero.

Circuit to check invalid BCD :

First we will design circuit to check sum & then we will logically OR output of this circuit to carry output of IC 7483

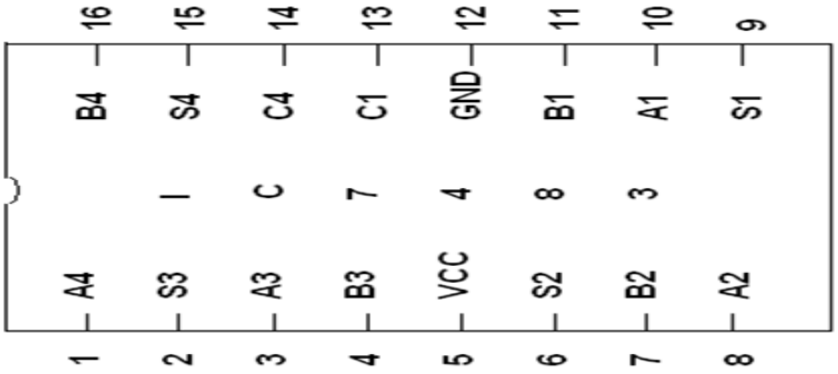
For digital circuit which we are going to design we will have 4 inputs(S3, S2, S1, S0) & only 1 output Y.

a) Y output of this circuit. Will be ORed with carry output of first adder IC 7483.

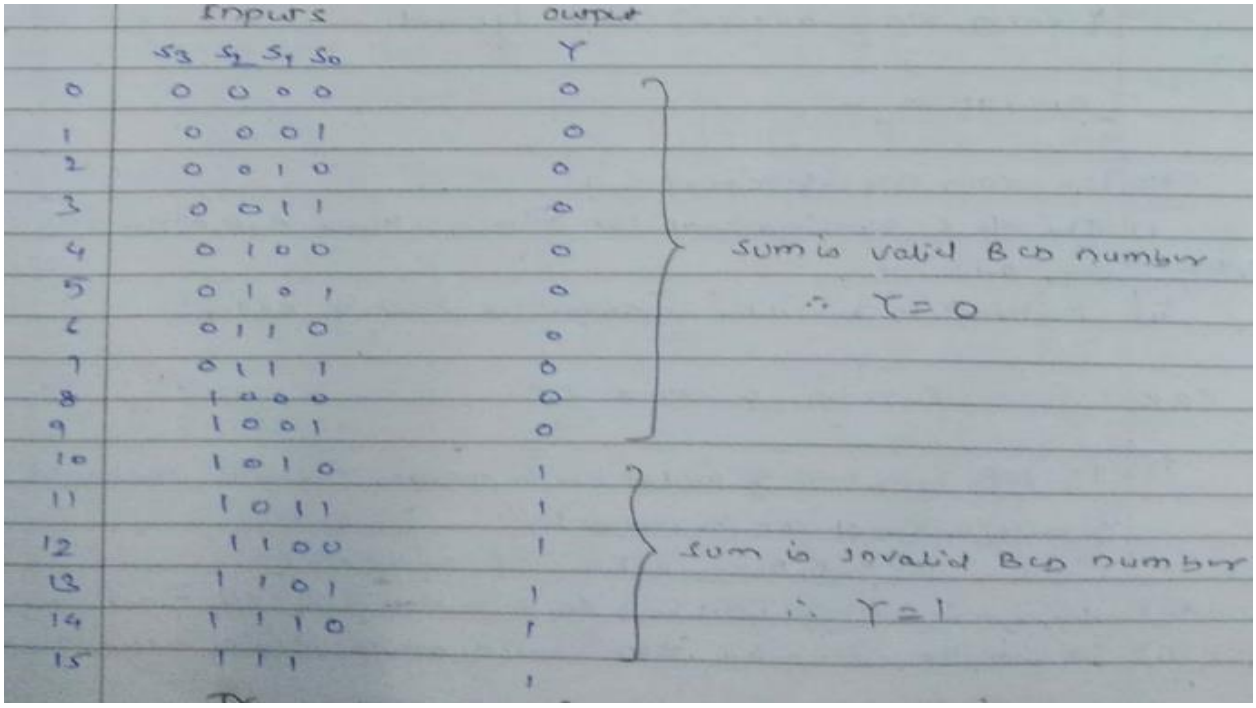
b) If BCD result is invalid i. e. sum output of first 7483 we have to add $(6)_{10}$ i.e. $(0110)_2$ that means we need one more binary adder IC 7483.

c) If BCD result is valid (i.e. final output of the circuit to check validity is 0) we will make an arrangement that second adder IC 7483 adds $(0)_{10}$ i. e. (0000 $)_2$to the sum of the first adder IC 7483. The output of the combinational circuit is used as final output carry & carry output of second adder IC is ignored.

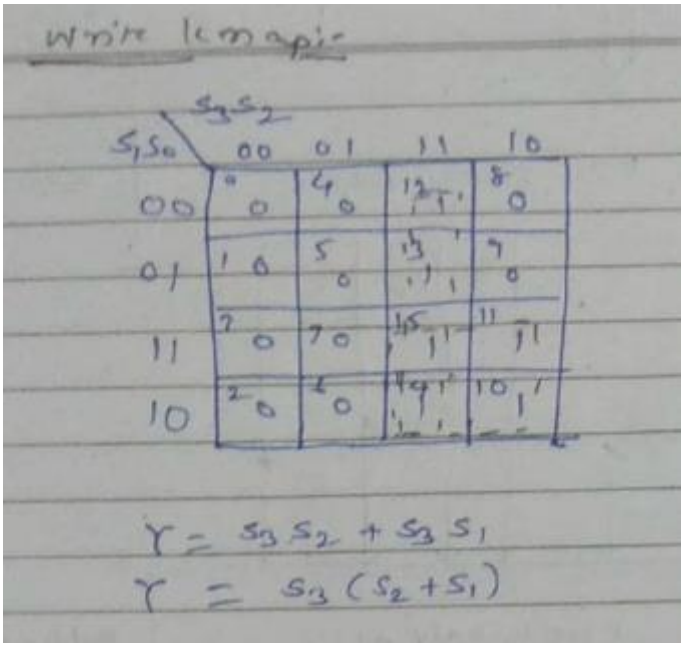
Pin Diagram of IC7483:



TruthTable for invalid BCD numbers:



K Map for Output Y:



i) The o/p of combinational ckt should be 1 if Cout of adder-1 is adder-1 is high.
∴ Y is ORed with Cout of adder 2 as shown in fig.

ii) The o/p of combinational ckt is connected to B_1 B_2 i/p's of adder-2 & $B_3 = B_0 = 0$ as they are connected to ground permanently.
This makes $B_3 B_2 B_1 B_0 = 0110$ if $Y' = 1$

iii) The sum o/p's of adder-1 are applied to $A_3 A_2 A_1 A_0$ of adder-2.

BCD ADDER DIAGRAM:

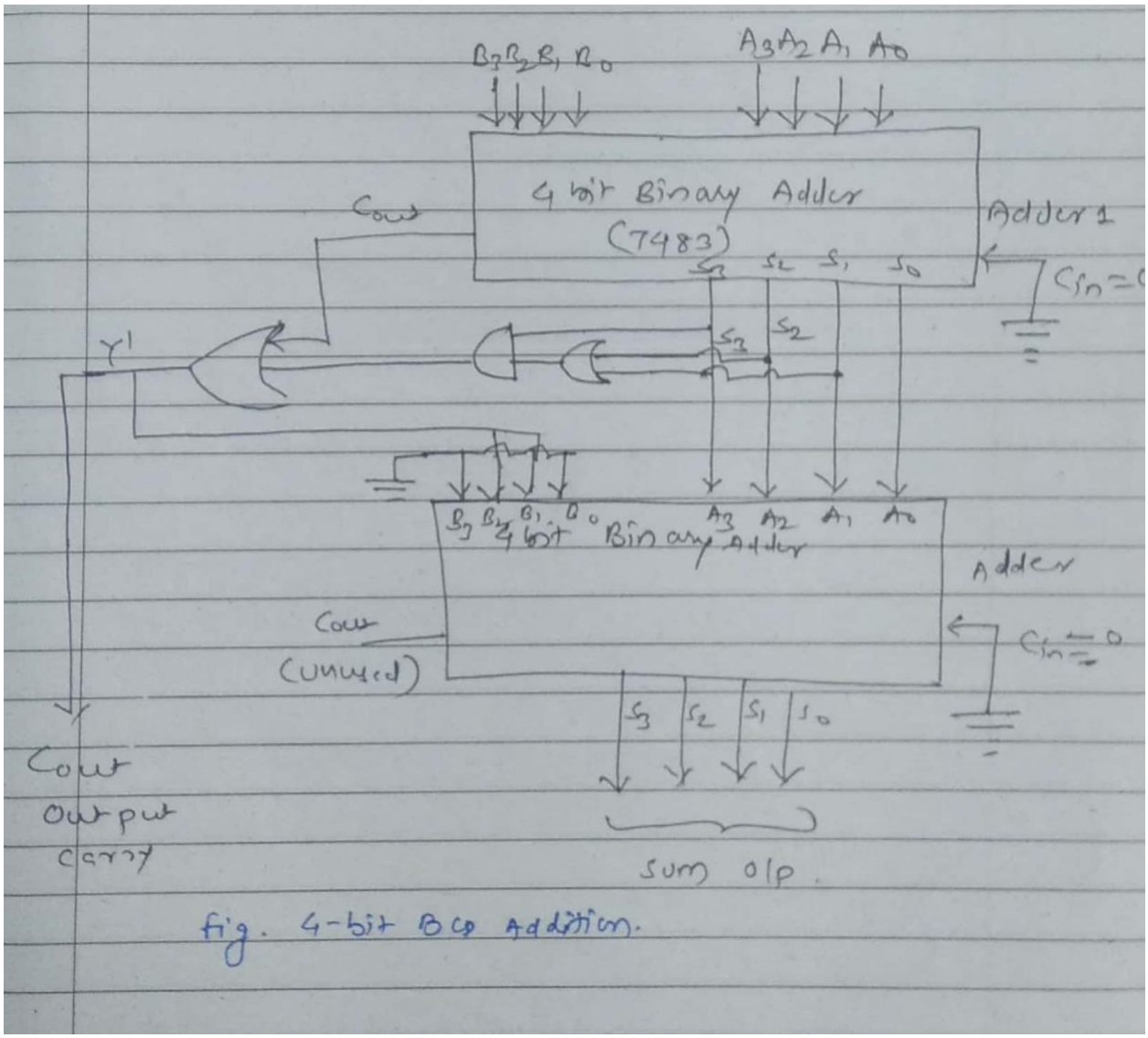


fig. 4-bit BCD Addition.

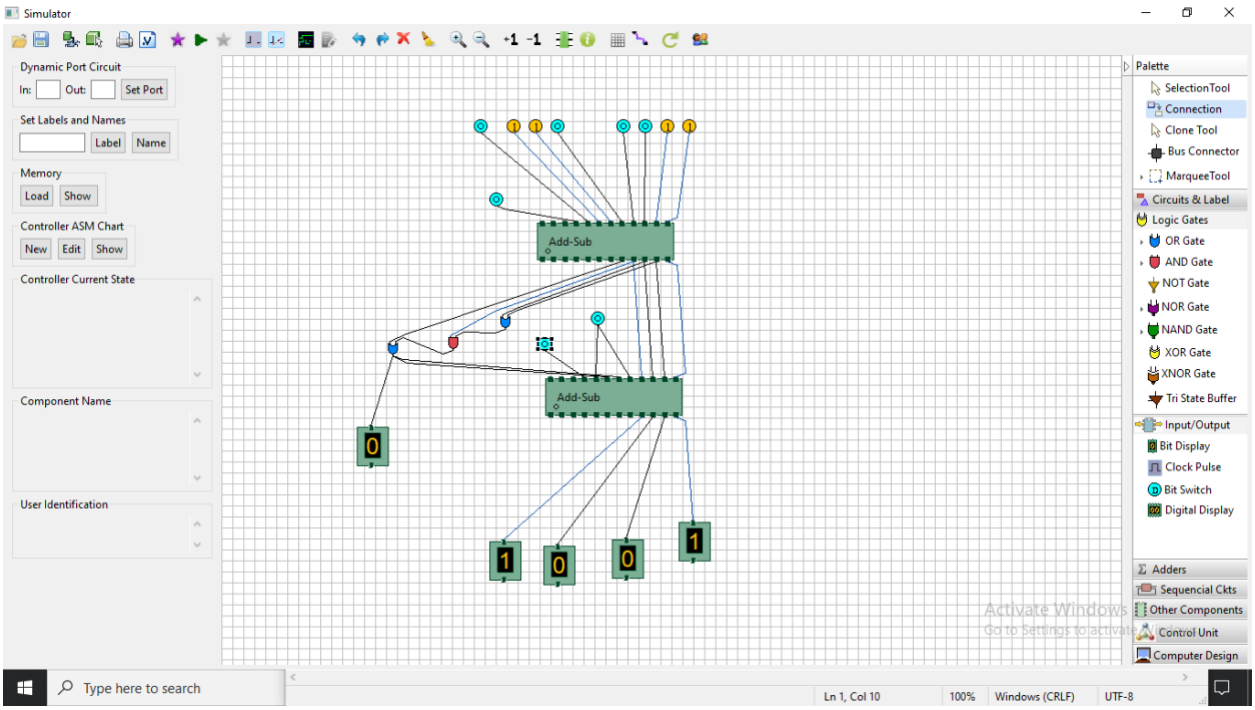
HARDWARE REQUIREMENTS for BCD Adder:

Sl No.	IC	Description	Quantity
1	7408	AND Gate	02 (gates)
2	7483	4 bit Binary Adder	02
3	7408	Quad 2 input and 1 output AND gate	01

Implementation of BCD Adder using Simulator software

4 Bit BCD Adder for Single Digit Sum

1) If Sum <= 9 and Carry = 0 ; 6+3 Sum = 9 and Carry = 0



2) If Sum > 9 and Carry = 0 ; 6+6 Sum = 9 and Carry = 0

0110

\+ 0110

Sum = 1100

Invalid Sum so add 6 i.e., 0110

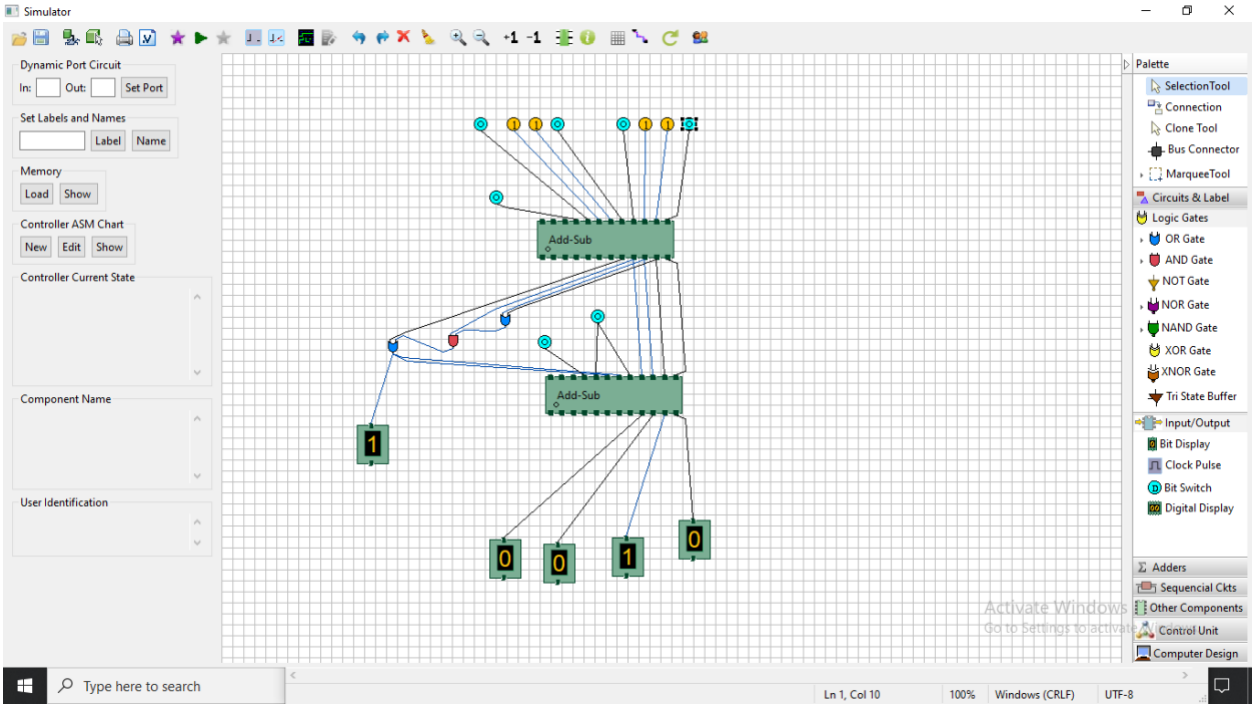
1100

0110

---

1 0010

Carry Sum



3) If Sum < 9 and Carry =1 ; 8+8 Sum = 6 and Carry = 1

8 = 1000

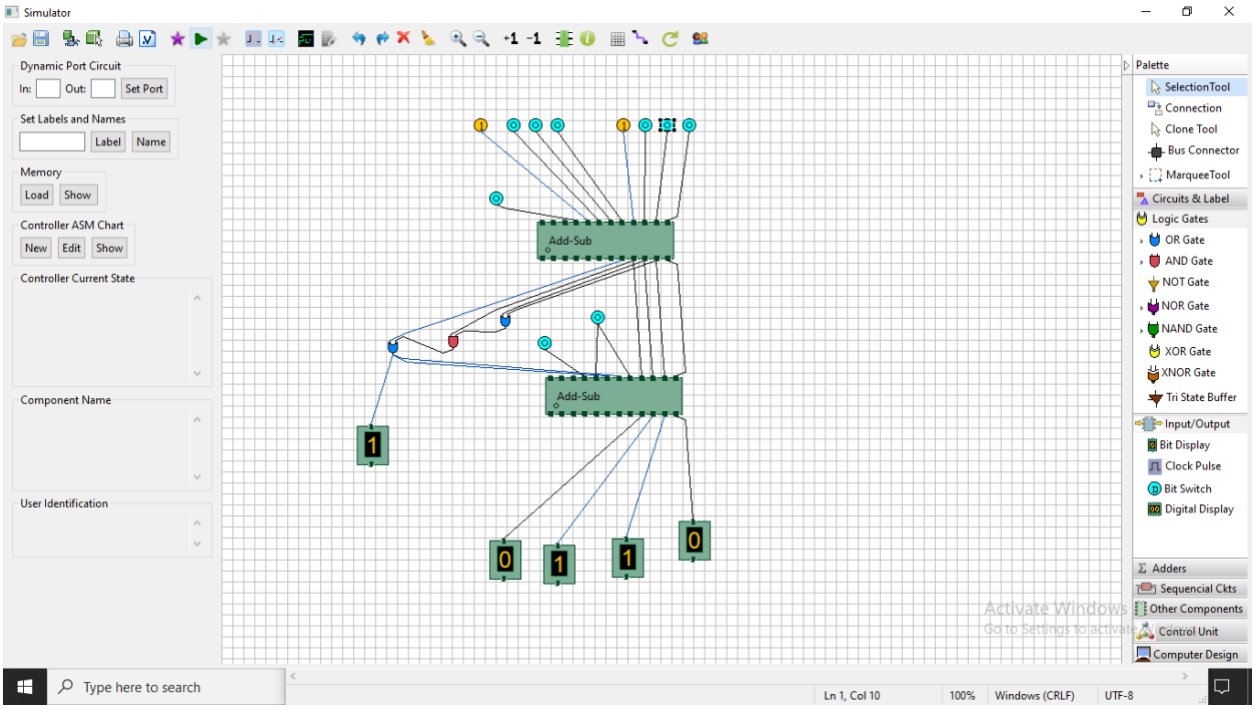
+8 = 1000

------- --------

16 = 1 0000 wrong result so add 6 in sum

0110

0110



References:

1. R.P. Jain , “Modern digital electronics” , 3rd edition , 12th reprint Tata McGraw Hill Publication, 2007.
2. Morris Mano, “Digital Logic and Computer Design” 4th edition, Prentice Hall of India, 2013.

Questions:

S. No	QUESTION	BT Level	CO
1	Write different BCD addition rules.	1	1
2	Discuss and design circuit to detect invalid BCD number and implement using NAND gate only.	1	1
3	A combinational circuit has four inputs (A,B,C,D), which represent a binary coded-decimal digit. The circuit has two groups of four outputs -S,T,U,V(MSB digit) and W,X,Y,Z.(LSB digit). Each group represents a BCD digit. The output digits represent a decimal number which is five times the input number. Illustrate the minimum expression for all the outputs?	3	1
4	Draw and describe the working principle of a two-digit BCD adder.	2	1
5	Why do we add 6 i.e. $(0110)_2$ for BCD correction?	2	2
6	Write a short note on five-bit BCD codes?	2	2
7	**Solve** arithmetic operation indicated below. Follow signed bit notation: i. 001110 + 110010 ii. 101011 - 100110. b) **Explain** the importance of gray code?	3	1
8	**Solve** (3250 - 72532)10 using 10's complement?	3	1
9	In a 32 bit computer, what are the maximum and minimum possible binary numbers? **Convert** these into maximum and minimum possible positive decimal numbers?	2	1
10	**Convert** the octal numbers into binary, decimal, BCD and Hexadecimal numbers(3600)octal,(1200)octal,(0200)octal,(0777)octal.	2	1
11	**Convert** the decimal numbers into binary, BCD and Hexadecimal numbers (3600)d, (1200)d, (0200)d, (0777)d.	2	1
12	Suppose you have a cheque for RS.10000/-.what is the number system used? **Define** base system used and what are the weights of the digits 1,0,0,0,0 and 0 now?	3	1

Guidelines for students:

- The students should verify the truth table.
- Fair diagram should be redrawn with the pencil.
- The conclusion should be written based on the results generated after performing the experiment.

Assessment methodology:

- Each experiment will be accessed out of 10.
- Attending the lab session will carry 01 mark.
- Performing and showing result of the experiment carry 02 marks.
- Submitting the file in the scheduled time will carry 07 marks.

Experiment No. 3

Title: Study of IC-74LS153 Multiplexer & IC 74138 as Decoder.

Problem Definition & Aim of Experiment:

1. Design(truth table) and Implement 8:1 mux using IC 74153.
2. Design (truth table) and Implement Boolean function (Any function can be given in exam.)
3. Design (truth table) Full adder implementation using hardware reduction table.
4. Design (truth table) Full Subtractor implementation using hardware reduction table.
5. Design (truth table) and Full Adder & Subtractor using IC 74138.

Objective of Experiment:

To understand how multiplexer can be used to reduce the hardware required.

Lab Equipment's and IC's Used:

Digital Trainers Kit74LS153,74LS138, 74LS32, 74LS04.

AIM: Part A – MUX IC 74153

1) Verification of IC.
2) Implementation of 8:1 Mux by cascading 2, 4:1 mux in IC 74153

Part B – Decoder IC 74138

1) Verification of IC.
2) Boolean function implementation.

THEORY :

Digital Multiplexer:

Multiplexer are combinational digital circuits equating as controlled switches with several data inputs (I_0, I_1, I_2 …) & one single data output ("out").At any time one of the I/p is transmitted to output. According to binary signals applied on control pairs to circuit. Usually the number of data inputs is a power of two. Multiplexing is the process of transmitting a large no. of information units over a small no. of channel / digital multiplexer is a combinational large circuit which performs the operation of multiplexing .It selects the operation of multiplexing. It selects the operation of binary information from one of the many input lines & transfer to a single o/p line. Multiplexer is called a data selector or multiposition switch because it selects one of the many input. Selection of a particular line is controlled by a set of a selection lines or selects inputs. The number of select lines depends upon no. of input lines.

Generally there is 'n' selects line for 'm' input lines. By applying a particular code on select lines is transmitted on the output lines.Block diagram of MUX is shown. at contains '2^m' input lines 'm' select & one unable input which is used to activate or dedicate MUX . Depending upon the no. of I/P & O/P lines various types of multiplexers are available. We have 2:1, 4:1, 8:1, 16:1 MUX. Here the first no. indicates the no. of input lines & second no. indicates the no. of output lines.

Demultiplexer :

Demultiplexer is a logic used to perform exactly reverse function performed by multiplexer. It accepts a single input and distributes among several outputs. The selection of a particular output line is controlled by a set of selection line. There are n input lines & 2^m is the number of selection line whose bit combinations determine which output to be selected.

Difference between Multiplexer, Demultiplexer & Decoder

Point	Multiplexer	Demultiplexer	Decoder
Input	Many input lines	Single input line	Many input line also Acts as select line
Output	Single output line	Many output lines	Many output line, Active low output
Select line	$2^m = n$	$n = 2^m$	Enable inputs used

Encoder & Decoder :

1. Encoders are used to encode given digital number into different numbering format like decimal to BCD Encoder, Octal to Binary.
2. Decoders are used to decode a coded binary word like BCD to seven segment decoder.
3. Thus encoder and decoder are application specific logic develop, we can not use any type of input for any encoder and decoder.
4. Need to select input according to encoder and decoder being selected for a particular application as mention in examples above.

Uses of Mux. :

1) Use for Boolean function implementation.
2) Construct a common bus system.
    1) To select between multiple sources & signal destination.
    2) Inter register transfer.

Advantages :

1) Simplification of logic expression not required.
2) Logic design is simplified.

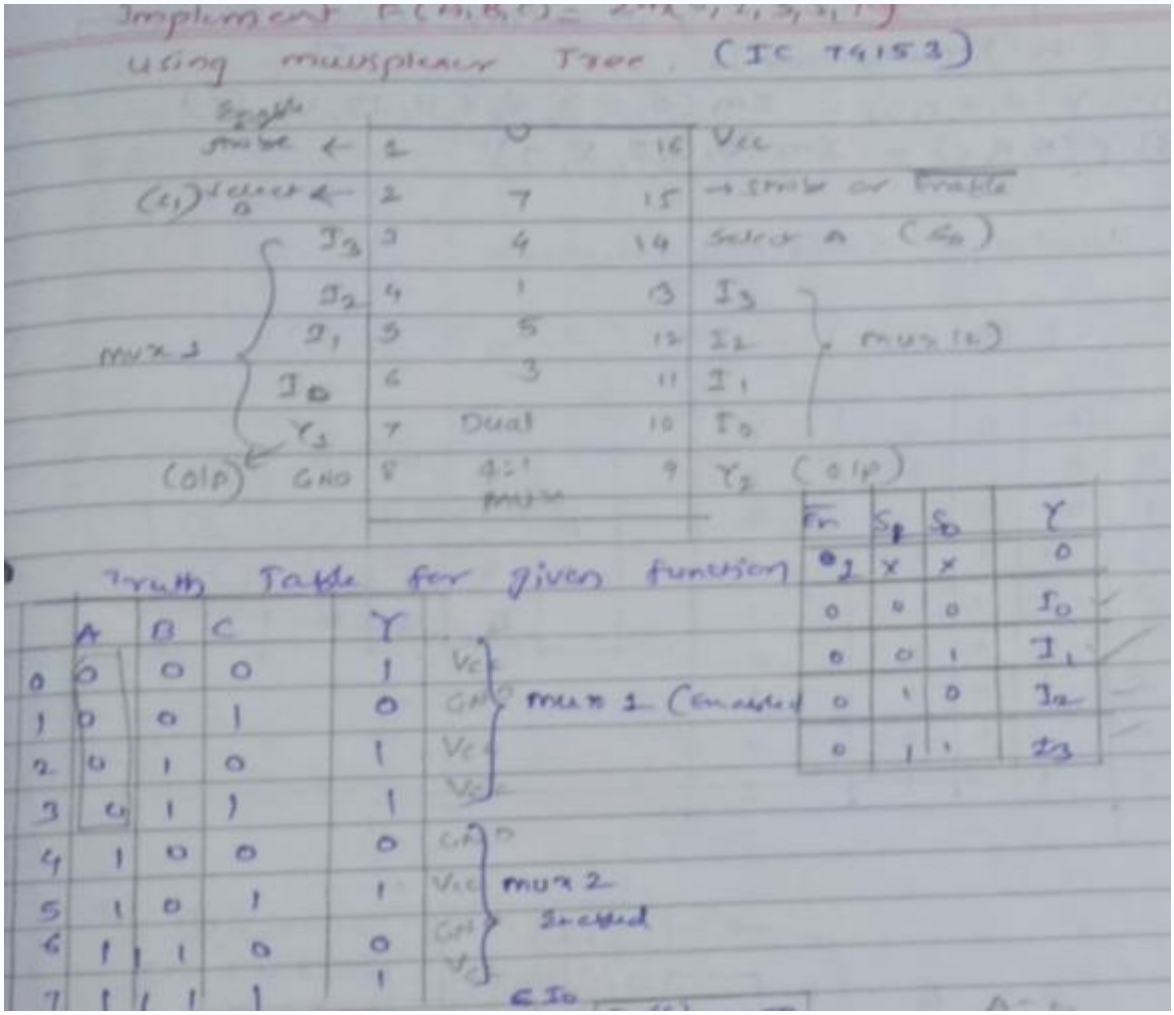
Disadvantage :

Only one function can be implemented using one MUX. Hence they can't be used in combinational logic circuit which contains many function.

Part-A (IC 74153)

1. VERIFICATION OF IC 74153 :

IC 74153 is a dual layer 4:1 MUX. It has four input lines for (I_0D-I_3D) for second MUX & active high output. 'Y_a', 'Y_b' (1Y or 2Y). It has select lines S_1S_0 common to both MUX. The Enable inputs are active low, E_a& E_b (1G and 2G). The MUX is activated when they are at logic o.



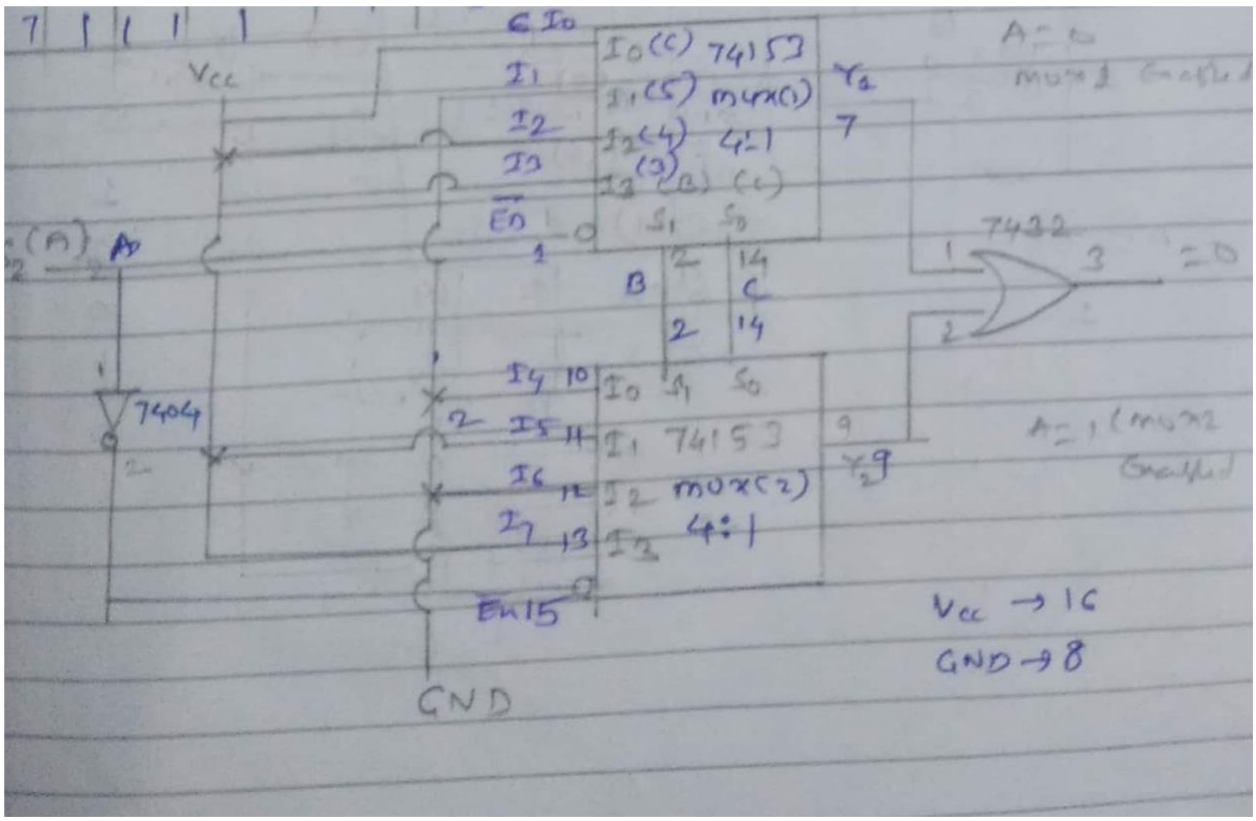


Fig. Multiplexer Tree Method

2. FUNCTION IMPLEMENTATION:

Y= ∑ m (1, 3, 5, 6)

This expression is in Standard SOP form and it is three variable function. So, we need to use mux with three select inputs i.e. 8:1 Mux. Already we have implemented 8:1 Mux using IC 74153. For Boolean function in Standard SOP form we connect data inputs corresponding to the minterms present in the given function to Vcc and remaining data inputs to ground.

Truth table :

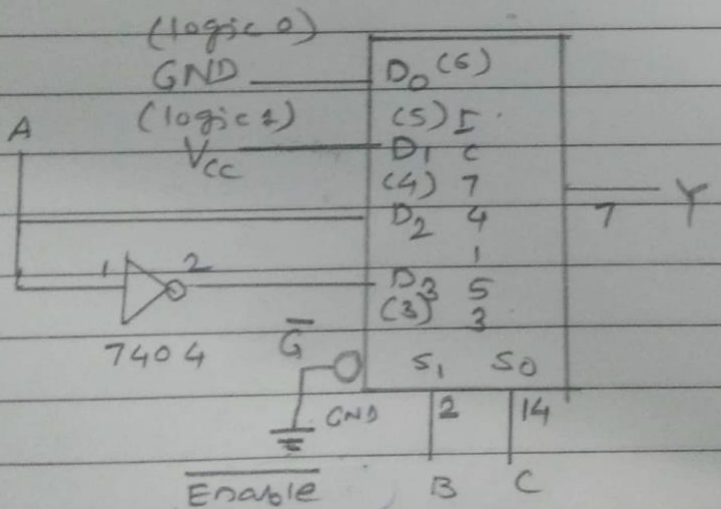
Inputs			***Output***
C	***B***	***A***	***Y***
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$Y = \Sigma m(1, 3, 5, 6)$

Using Hardware Reduction Table (MSA) Approach

	D_0	D_1	D_2	D_3
$\bar{A}$	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	$\bar{A}$

Fig. Hardware Reduction Table



4. IMPLEMENTATION OF FULL ADDER USING IC 74153:

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 these variables denoted by A and B represent the two significant bits to be added. The third input represents the carry from previous lower significant position.

Truth Table for Design of full adder:

Input			***Output***	
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
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

Sum= $\sum$m (1, 2, 4, 7),

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

Equations for Full Adder

Sum & Carry

Sum = $\Sigma m(1, 2, 4, 7)$ = `

(A, B, C)

Carry = $F(A, B, C) = \Sigma m(3, 5, 6, 7)$

Sum:-

	D_0	D_1	D_2	D_3
$\bar{A}$	0	(1)	(2)	3
A	(4)	5	6	(7)
	A	$\bar{A}$	$\bar{A}$	A

Carry:-

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	(3)
A	4	(5)	(6)	(7)
	0	A	A	1

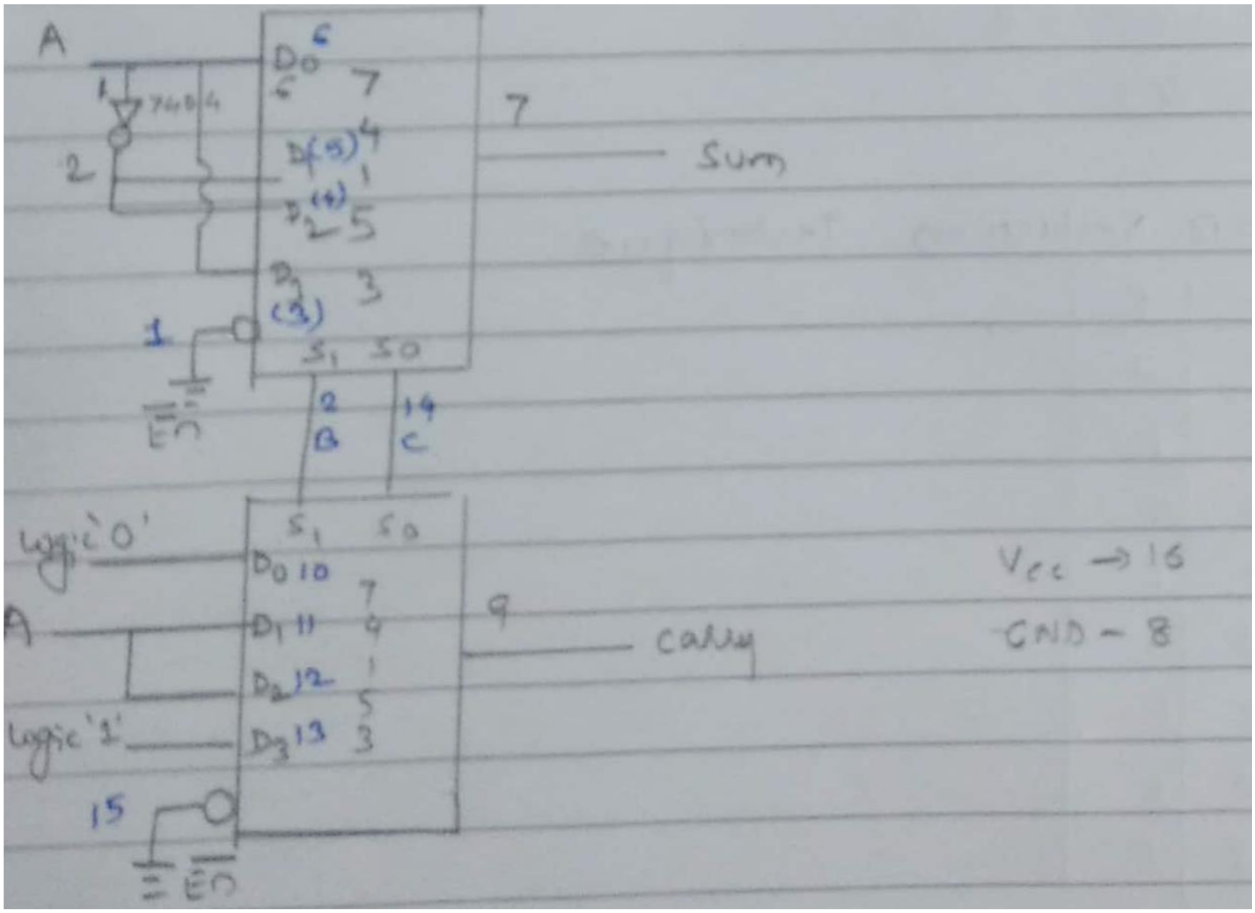
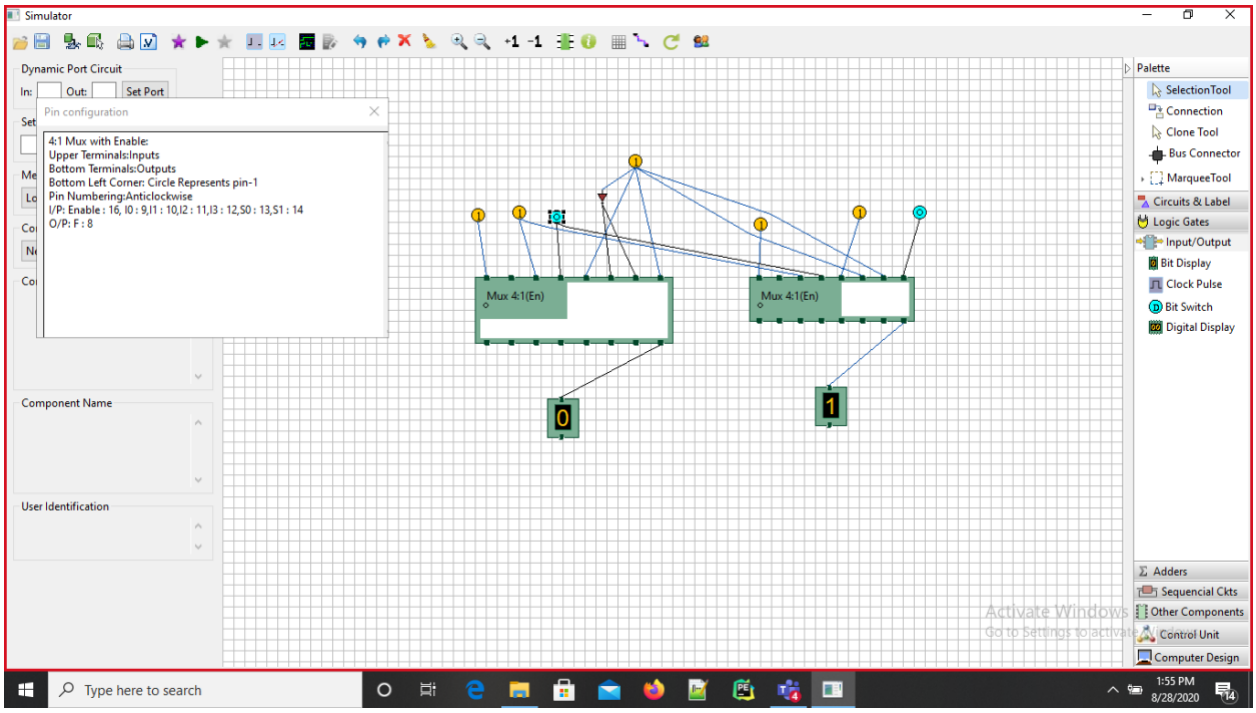


Fig. Circuit Diagram for Full Adder

Hardware Requirements :

GATE	Quantity	IC	Quantity
Mux.	1	74153	1
NOT	1	7404	1

Implement Full Adder using Multiplexer 4:1 with Enable Input



Part-B Decoder (IC 74138)

THEORY:

Discrete quantities of information are requested in digital system with binary codes. A binary code of n bits is capable of representing into 2^n distinct elements of the coded information.

Decoder converts coded input to coded outputs accepts one of the code.

There are different types of decoders such as 3:8 decoder, 4:16 line decoders etc. These are in general called as n: m line decoder where $m=2^n$ and n= no. of input lines and m=no. of output lines.

Demux also takes one input data line source and selectively distributes it to one of n output channels. The only difference between demux and decoder is that demux has Din (data i/p) line whereas decoder does not have.

ADVANTAGES:

1) The decoder provides best implementation whenever there are many outputs of the combinational circuit and each o/p of the function (or its complement) is required to be expressed with a small no. of minterms.
2) The decoder can function as demux. If the Enable i/p line is taken as Din (data i/p) .

DISADVANTAGES:

Since decoder method requires an OR gate for each o/p function, so there is new hardware used. And it is always advisable to use minimum hardware as we come across problems like propagation delay of gates.

APPLICATIONS:

Decoder is worthily used for decoding binary information and memory interfacing. It is used for the implementation of Boolean function.

A) Verification of IC 74138:

We use IC 74138 which accepts 3 binary weighted inputs (A0, A1, A2) and when enabled provides mutually exclusive active low outputs (y0-y7). It features 3 Enable i/ps. Two active low (G2A, G2B) and one active high (G1). Every output will be high unless G2A, G2B are low and G1 is high. It has demultiplexing capability and multiple enable i/ps for easy expansion.

Function Table of 3:8 decoder:

Input						Output							
Enable			Data										
G2A	**G2B**	**G1**	**A2**	**A1**	**A0**	**Y0**	**Y1**	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**
0	0	0	X	X	X	1	1	1	1	1	1	1	1
0	1	1	X	X	X	1	1	1	1	1	1	1	1
1	0	1	X	X	X	1	1	1	1	1	1	1	1
1	1	1	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

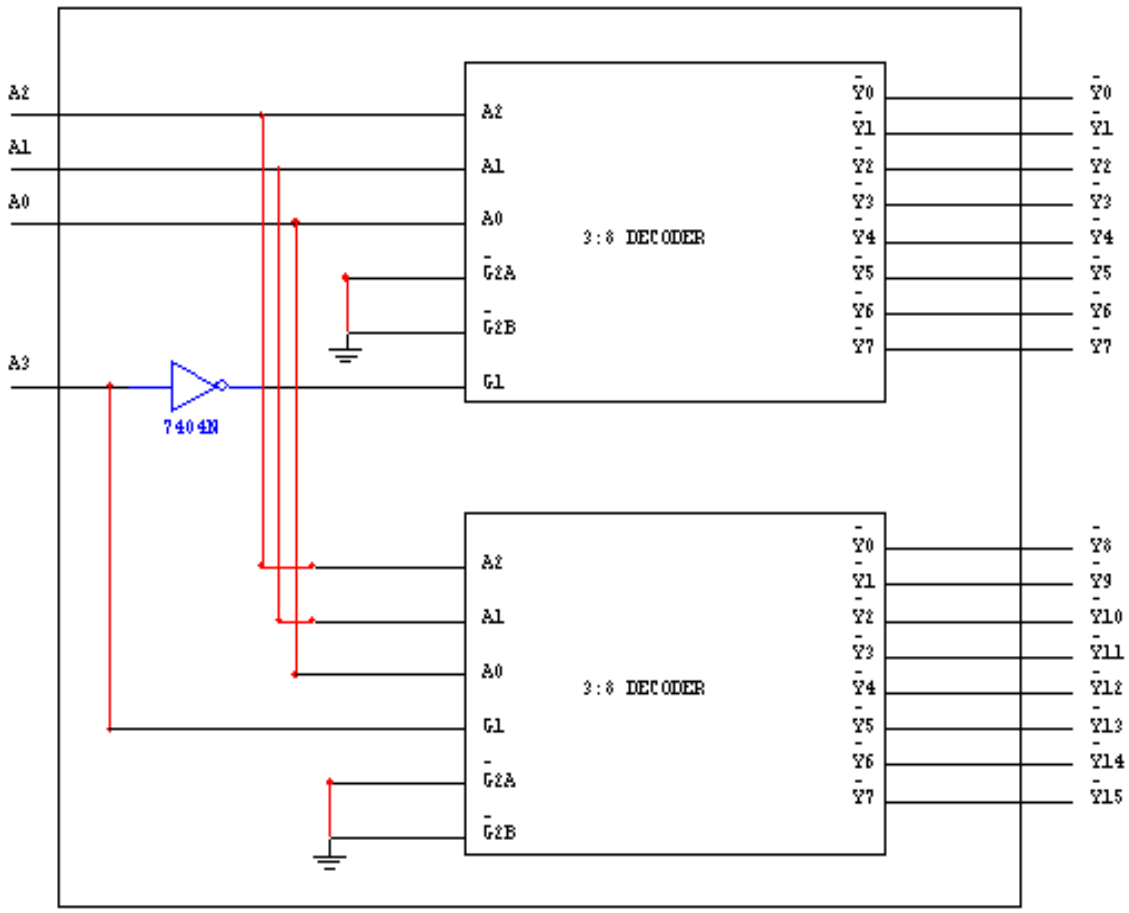
B) Cascading of IC 74138:

The enable i/p G1 active high of IC 74138 is used for cascading. for cascading 2 IC's ,the enable i/p G1 of first IC is connected to G1 enable i/p of second IC through a NOT gate. This enable i/p is used as MSB select i/p line A3. the other three select input lines of both IC's (A0,A1,A2) are also shorted to select input lines of second IC to get single i/p select lines (A0,A1,A2).

The i/p line A3 is used to enable /disable the 2 IC 74138 decoders. When A3=0, first IC is enabled and second is disabled. Thus the first decoder will generate minterms from 0000 to 0111 as o/p and the second decoder will generate nothing. When A3=1, the enable conditions are reversed and thus second decoder IC will generate minterms 1000 to 1111.

Function Table of 4:16 decoder using IC 74138 (3:8 decoder):

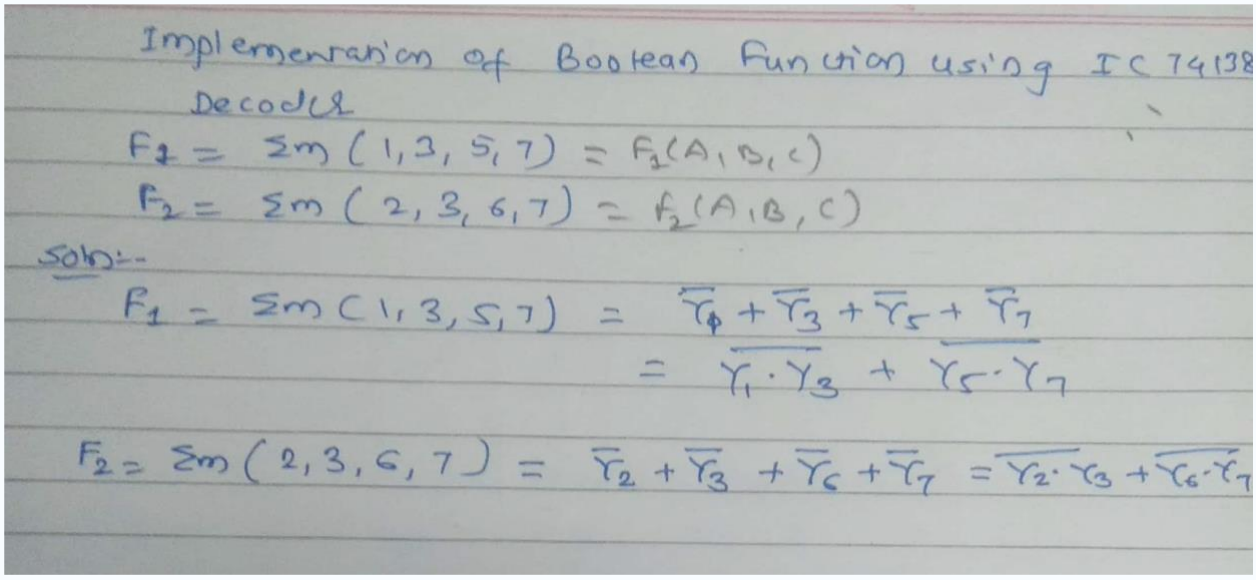
Input						**Output**															
Enable		**Data**																			
G2A	**G2B**	**A3**	**A2**	**A1**	**A0**	**Y0**	**Y1**	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**	**Y8**	**Y9**	**Y10**	**Y11**	**Y12**	**Y13**	**Y14**	**Y15**
0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0


A2
A1
A0
A3
7404N
A2
A1
A0
G2A
G2B
G1
3:8 DECODER
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
A2
A1
A0
G1
G2A
G2B
3:8 DECODER
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
Y8
Y9
Y10
Y11
Y12
Y13
Y14
Y15

C) Implementation of Boolean function:

The procedure for implementation of combinational circuit by means of a decoder and 'OR' gates requires that the Boolean function fir the circuit be expressed in Sum of Minterms. These forms can be obtained by expanding the function. A decoder is then chosen which generates all the minterms of n i/p variables. The i/p to each OR gate are selected from the decoder outputs according to the minterms list in each function.

For example, **F1=∑m (1, 3, 5, 7) and F2=∑m (2, 3, 6, 7**)



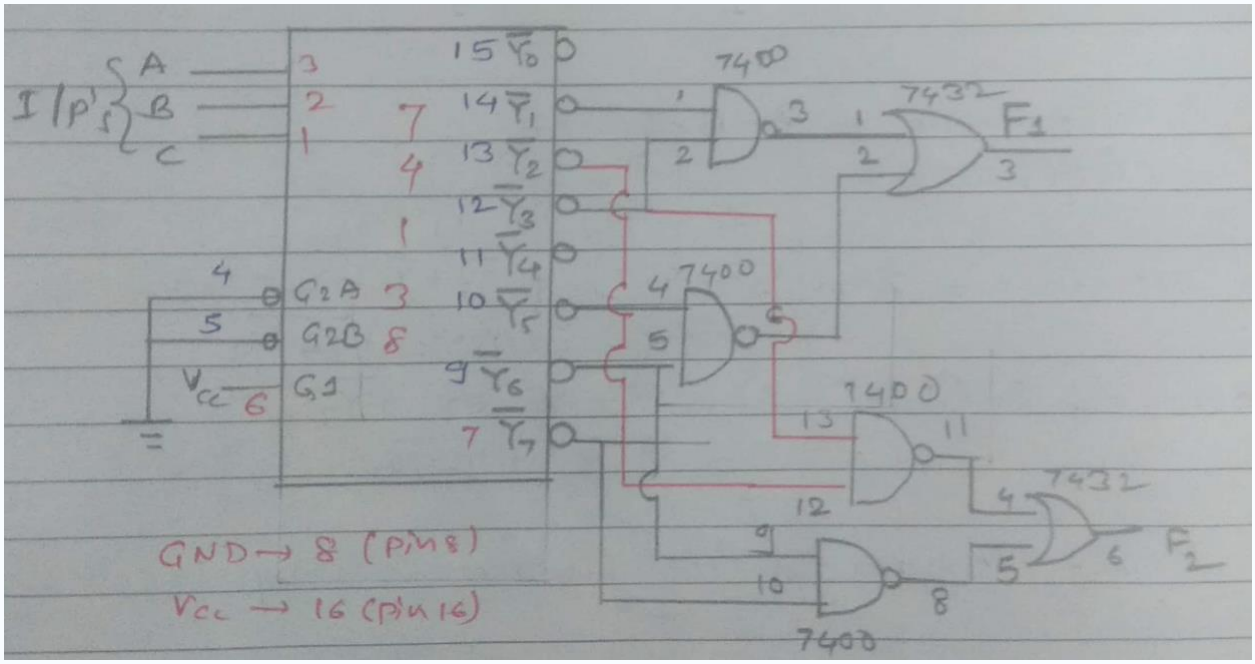
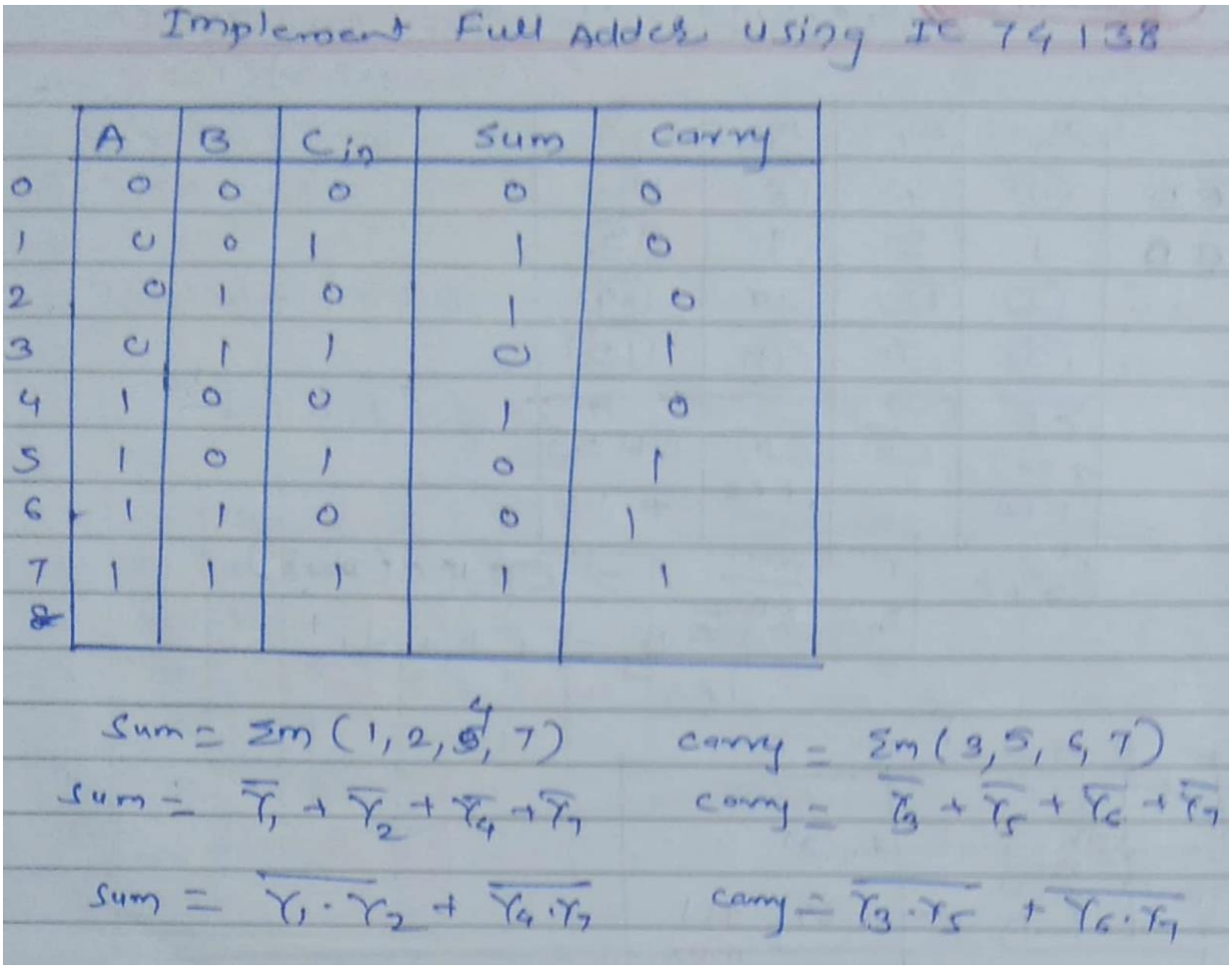


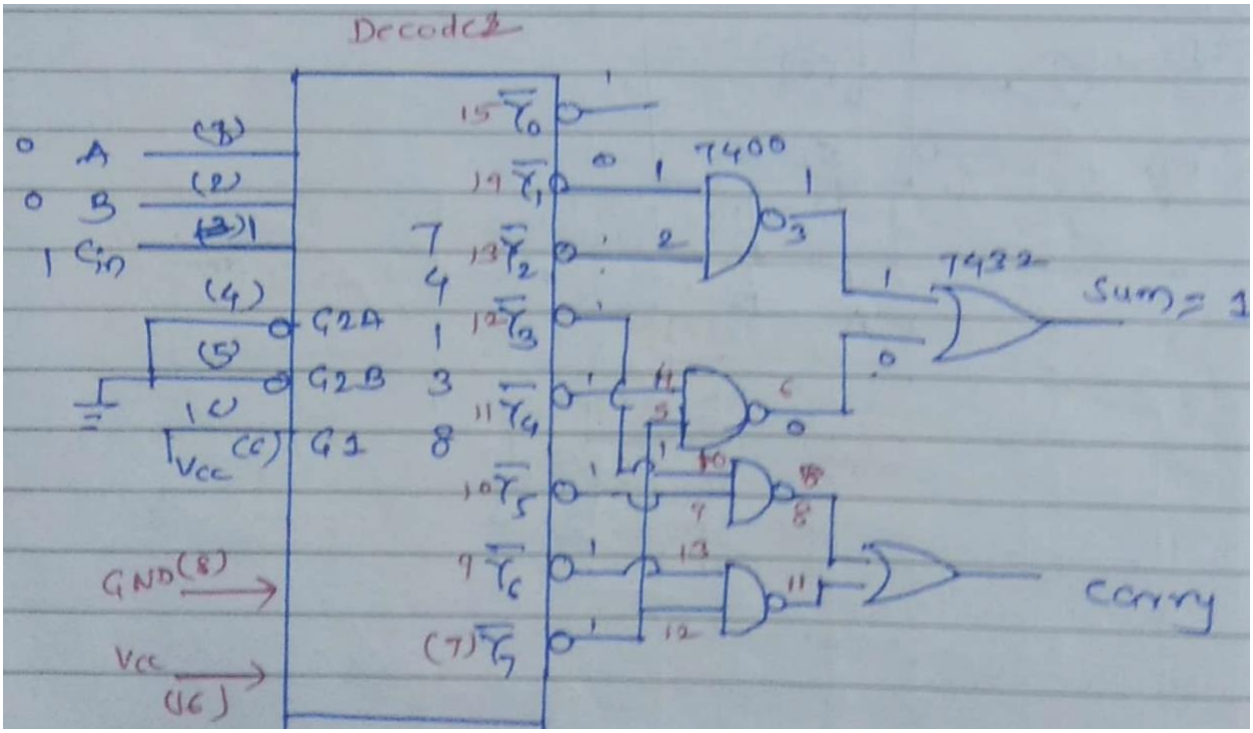
Fig. Logic Circuit diagram for Function Implementation using IC 74138 Decoder

a) Implementation of Full Adder using IC 74138:-

First of all we need to decide on which type of decoder the above Boolean function can be implemented. The highest minterm is 7 and minimum no. of bits required to represent it in binary form are 3. So we have 3 select lines in 3:8 decoders so we can use IC 74138.

To implement the function we require two OR gates and four NAND gates (7432& 7400). As the o/p of the decoder IC 74138 are active low and we need to get o/p active high at the o/p pin of the function SUM and CARRY when respective minterms are selected.





Hardware Required:-

Sr.No.	IC Number	Quantity	Name of IC
1.	74LS138	1	3:8 Decoder
2.	74LS00	1 (4 gates used)	Quad 2 input & 1 output NAND gate
3.	74LS32	1(2 gates used)	Quad 2 input & 1 output NAND gate
4.	Patch chords	Min 26 required	

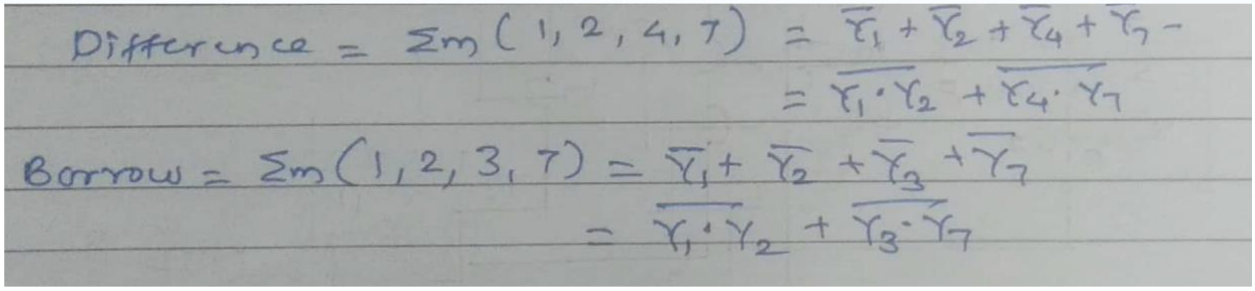
Implementation of Full Subtractor:

Same case will happen in this case. Again first of all we need to decide on which type of decoder the above Boolean function can be implemented. The highest minterm is 7 and minimum no. of bits required to represent it in binary form are 3. So we have 3 select lines in 3:8 decoders so we can use IC 74138.

To implement the function we require 2-OR gates and 3-NAND gates (7432 & 7400). As the o/p of the decoder IC 74138 are active low and we need to get o/p active high at the o/p pin of the function DIFFERENCE and BORROW when respective minterms are selected.

Truth Table for design of Full Subtractor:

INPUT			**OUTPUT**	
A2	**A1**	**A0**	**DIFFERENCE**	**BORROW**
A	**B**	**C**		
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



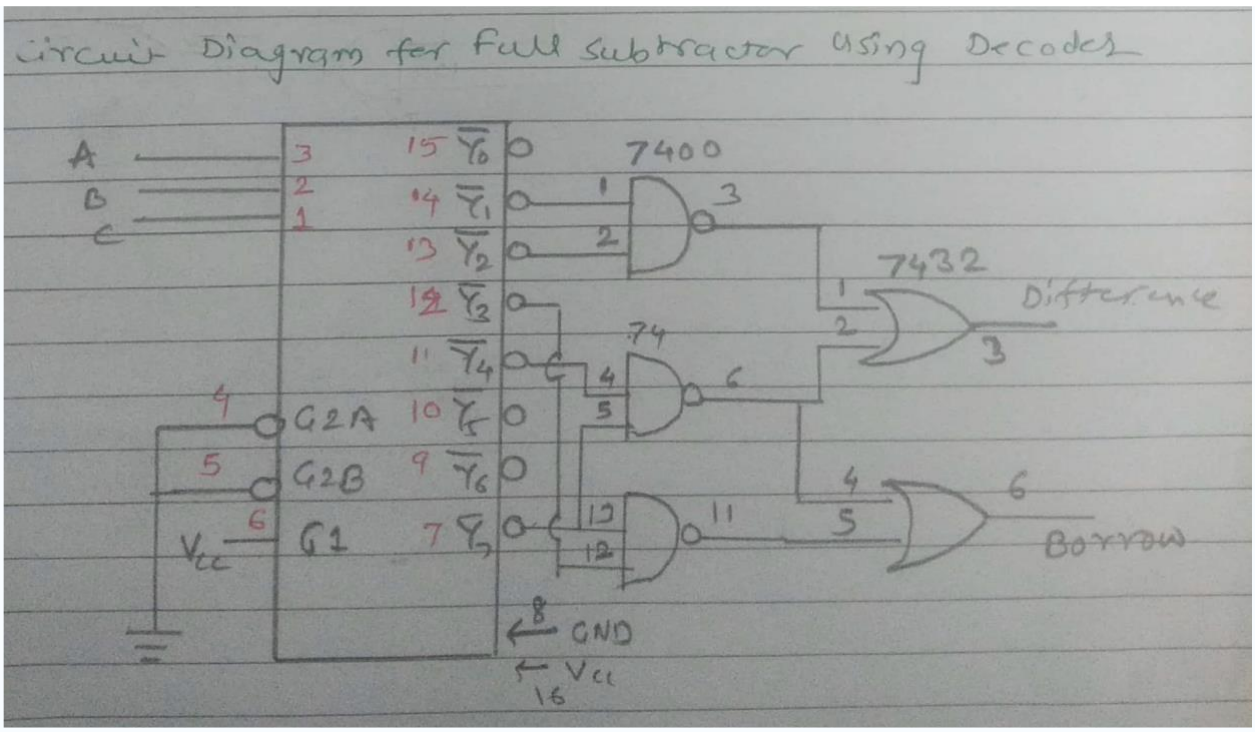


Fig. Circuit diagram for Full Subtractor Decoder using IC 74LS138

S. No	QUESTION	BT Level	C O
1	Define Multiplexer, Demultiplexer.	1	2
2	Define Encoder, Decoder	1	1
3	What is the role of select lines?	1	2
4	Explain the real life applications of multiplexer.	2	2
5	Develop down the relation between MUX inputs & select lines.	3	2
6	How many MUX's are present in IC 74LS138.	1	3
7	What is the name of IC 74LS138?	1	2
8	What is the relation between Input lines & output lines in Decoder?	1	2
9	What is the relation between Input lines & output lines in Encoder?	1	1
10	Distinguish between MUX & DEMUX.	3	2
11	Explain Strobe Input in MUX & DEMUX .	2	2

References:

- R.P. Jain, "Modern Digital Electronics", 3rd Edition, Tata McGraw-Hill, ISBN: 0-07-049492-4
- “Digital Design”, M Morris Mano, Prentice Hall, 3rd Edition, ISBN: 0130621218.

Assessment Methodology:

- Assessment of submission by student for every experiment will be a continuous assessment for the TW
- Assessment will be based on understanding of theory, attentiveness during practical, and understanding.
- How efficiently the student can do connections to get the results.
- Timely submission of journal.

Experiment No. 4

Title:- Design & Implement Asynchronous & Synchronous Counter

AIM: To design and implement 3 bit UP, Down Ripple & Synchronous Counter using MS-JK Flip-flop.

OBJECTIVE:

- To understand concept and role of clock in sequential circuits.
- To understand the design procedure of asynchronous & Synchronous counter.

ICs USED : IC 7476 (MS-JK Flip-flop), IC 7408(Quad 2 i/p AND Gate),

THEORY:

Counters : counters are logical device or registers capable of counting the no. of states or no. of clock pulses arriving at its clock input where clock is a timing parameter arriving at regular intervals of time, so counters can be also used to measure time & frequencies. They are made up of flip flops. Where the pulse are counted to be made of it goes up step by step & the o/p of counter in the flip flop is decoded to read the count to its starting step after counting n pulse in case of module counters.

Types of Counters:

Counter are of two types**:**

1) Asynchronous counter.

2) Synchronous counter.

Asynchronous counter:

A digital counter is a set of flip flop. The flip flop are connected such that their combined state at any time is binary equivalent of total no. of pulses that have occurred up to that time. Thus its name implies a counter is used to count pulse. A counter is used as frequency dividers. To obtain waveform with frequency that is specific fraction of clock frequency.

Counter may be Asynchronous or synchronous. The Asynchronous counter is also called as ripple counter .An Asynchronous counter uses T flip flop to perform a counting function. The actual hardware used is usually J-K flip flop with J & K connected to logic1. Even D flip flops may be used here.

In asynchronous counter commonly called ripple counter, the first flip-flop is clocked by the external clock pulse & then each successive flip-flop is clocked by the Q or Q' output of the previous flip-flop. Therefore in an asynchronous counter the flip-flop's are not clocked simultaneously. The input of MS-JK is connected to

VCC because when both inputs are one output is toggled. As MS-JK is negative edge triggered at each high to low transition the next flip-flop is triggered.

Synchronous Counter :

When counter is clocked such that each flip flop in the counter is triggered at the same time, the counter is called as synchronous counter. The gates propagation delay at reset time will not be present or we may say will not occur.

1) Asynchronous Up Counter:

Fig. 1 shows 3bit Asynchronous Up Counter. Here Flip-flop 2 act as a MSB Flip-flop and Flip-flop 0 act as a LSB Flip-flop. Clock pulse is connected to the Clock of Flip-flop 0. Output of Flip-flop 0(Q_0) is connected to clock of next flip-flop (i.e Flip-flop 1) and so on. As soon as clock pulse changes output is going to change (at the negative edge of clock pulse) as a Up count sequence. For 3 bit Up counter state table is as shown below.

State Table :

Counter States	Count		
	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
8	0	0	0

Logic diagram :

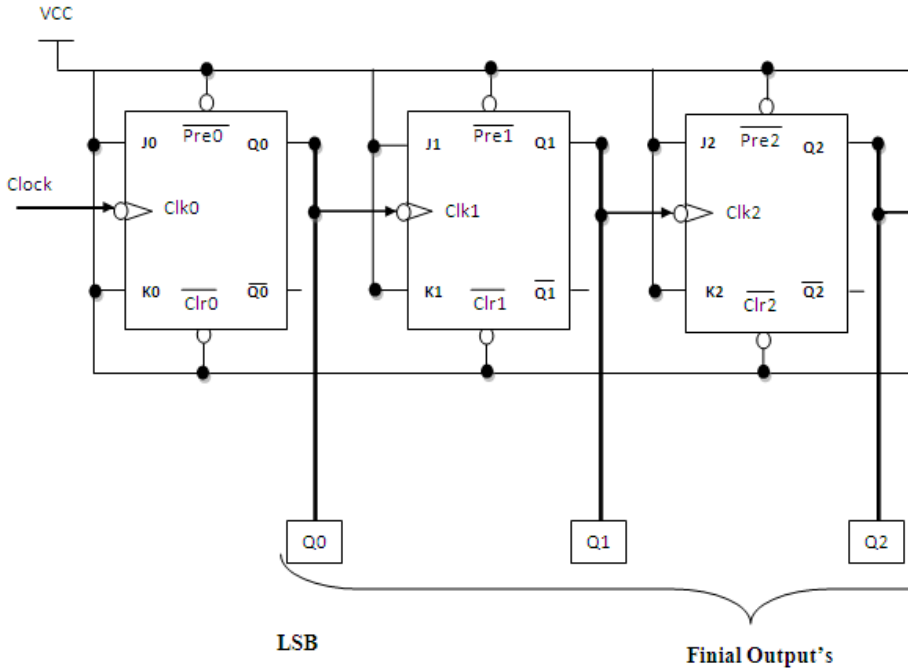


Fig 1: 3 Bit Asynchronous Up Counter

Hardware requirements :

Gate / Flip flop	Quantity	IC	Quantity
MS JK	3	7476	2

2) Down Counter:

Fig. 2 shows 2 bit Asynchronous Down Counter. Here Flip-flop 2 act as a MSB Flip-flop and Flip-flop 0 act as a LSB Flip-flop. Clock pulse is connected to the Clock of Flip-flop 0. Output of Flip-flop 0 (Q_0') is connected to clock of next flip-flop (i.e Flip-flop 1) and so on. As soon as clock pulse changes output is going to change (at the negative edge of clock pulse) as a down count sequence. For 3 bit down counter sate table is as shown below.

In both the counters Inputs J and K are connected to Vcc, hence J-K Flip flop work in toggle mode. Preset and Clear both are connected to logic 1.

State Table :

Counter States	Count		
	Q_2	Q_1	Q_0
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0
7	1	1	1

Logic diagram :

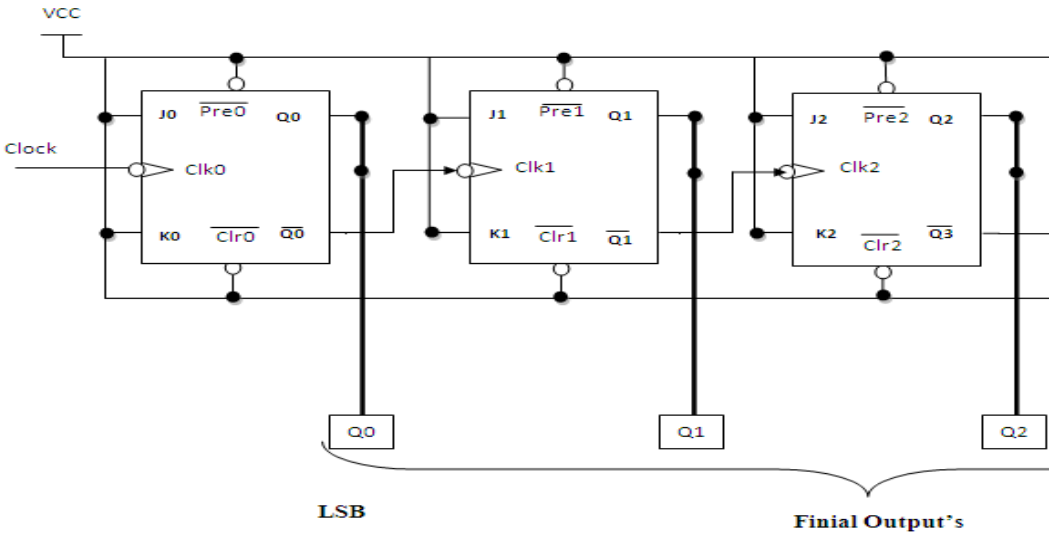


Fig 2: 3 Bit Asynchronous Down Counter

Hardware requirements :

Gate / Flip flop	Quantity	IC	Quantity
MS JK	3	7476	2

Applications :

The asynchronous counters are specially used as the counting devices.

They are also used to count number of pulses applied.

It also works as frequency divider.

It helps in counting the number of product coming out of the machinery where product is coming out at equal interval of time.

Types of synchronous counter:

1) Up counter.

2) Down counter.

1. A 3 bit Synchronous up counter:

The up counter counts from 0 to7 i.e.(000 to 111).for this we are using MS JK flip flop. In IC 74LS76, 2 MS J-K flip flops are present. The clock pulse is given at pin 1 & 6 of the 1st IC & pin 1 of 2nd IC. Next state decoder logic is designed with the help of state table.

State table for synchronous up counter:

Present state			**Next state**			**Flip flop 3**		**Flip flop 2**		**flip flop 1**	
Q2	**Q1**	**Q0**	**Q2**	**Q2**	**Q0**	**J2**	**K2**	**J1**	**K1**	**J0**	**K0**
0	0	0	0	0	1	0	x	0	X	1	x
0	0	1	0	1	0	0	x	1	X	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	X	1	X
1	0	1	1	1	0	x	0	1	X	x	1
1	1	0	1	1	1	x	0	x	0	1	X
1	1	1	0	0	0	x	1	x	1	x	1

K-Map :

Q1Q0 / Q2	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J_2 = Q_1Q_0$

Q1Q0 / Q2	00	01	11	10
0	X	X	X	X
1	0	0	1	0

$K_2 = Q_1Q_0$

Q1Q0 / Q2	00	01	11	10
0	0	1	X	X
1	0	1	X	X

$J_1 = Q_0$

Q1Q0 / Q2	00	01	11	10
0	X	X	1	0
1	X	X	1	0

$K_1 = Q_0$

Q1Q0 / Q2	00	01	11	10
0	1	X	X	1
1	1	X	X	1

$J_0 = 1$

Q1Q0 / Q2	00	01	11	10
0	X	1	1	X
1	X	1	1	X

$K_0 = 1$

Logic Diagram:

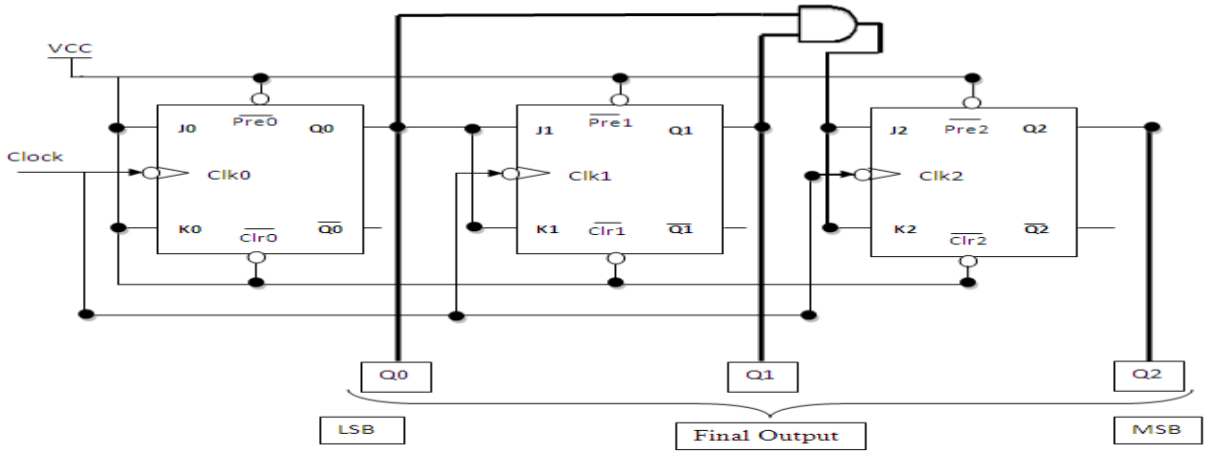


Fig 1: 3 bit Synchronous up counter

2. A 3 bit Synchronous down counter:

This is used to count from 7-0 i.e.(111-000).for this also 2 IC's of 74LS76 are required & hence we use 3 MS JK flip flops. Here also clock is given to 1st& 6th pin of 1st IC & 1st pin of 2nd IC enabling to apply clock to all flip flop at a time. Next state decoder logic is designed with the help of state table.

State table for synchronous down counter :

Present state			Next state			Flip flop 3		Flip flop 2		Flip flop 1	
Q2	Q1	Q0	Q2	Q1	Q0	J2	K2	J1	K1	J0	K0
1	1	1	1	1	0	X	0	X	0	X	1
1	1	0	1	0	1	X	0	X	1	1	X
1	0	1	1	0	0	X	0	0	X	X	1
1	0	0	0	1	1	X	1	1	X	1	X
0	1	1	0	1	0	0	X	X	0	X	1
0	1	0	0	0	1	0	X	X	1	1	X
0	0	1	0	0	0	0	X	0	X	X	1
0	0	0	1	1	1	1	X	1	X	1	X

K-Map :

Q1Q0 / Q2	00	01	11	10
0	1	0	0	0
1	X	X	X	X

$J_2 = Q'_1 Q'_0$

Q1Q0 / Q2	00	01	11	10
0	**X**	**X**	**X**	**X**
1	**1**	**0**	**0**	**0**

$K_2 = Q'_1 Q'_0$

Q1Q0 / Q2	00	01	11	10
0	1	0	X	X
1	1	0	X	X

$J_1 = Q_0'$

Q1Q0 / Q2	00	01	11	10
0	X	X	0	1
1	X	X	0	1

$K_1 = Q_0'$

Q1Q0 / Q2	00	01	11	10
0	1	X	X	1
1	1	X	X	1

$J_0 = 1$

Q1Q0 / Q2	00	01	11	10
0	X	1	1	X
1	X	1	1	X

$K_0 = 1$

Logic Diagram :

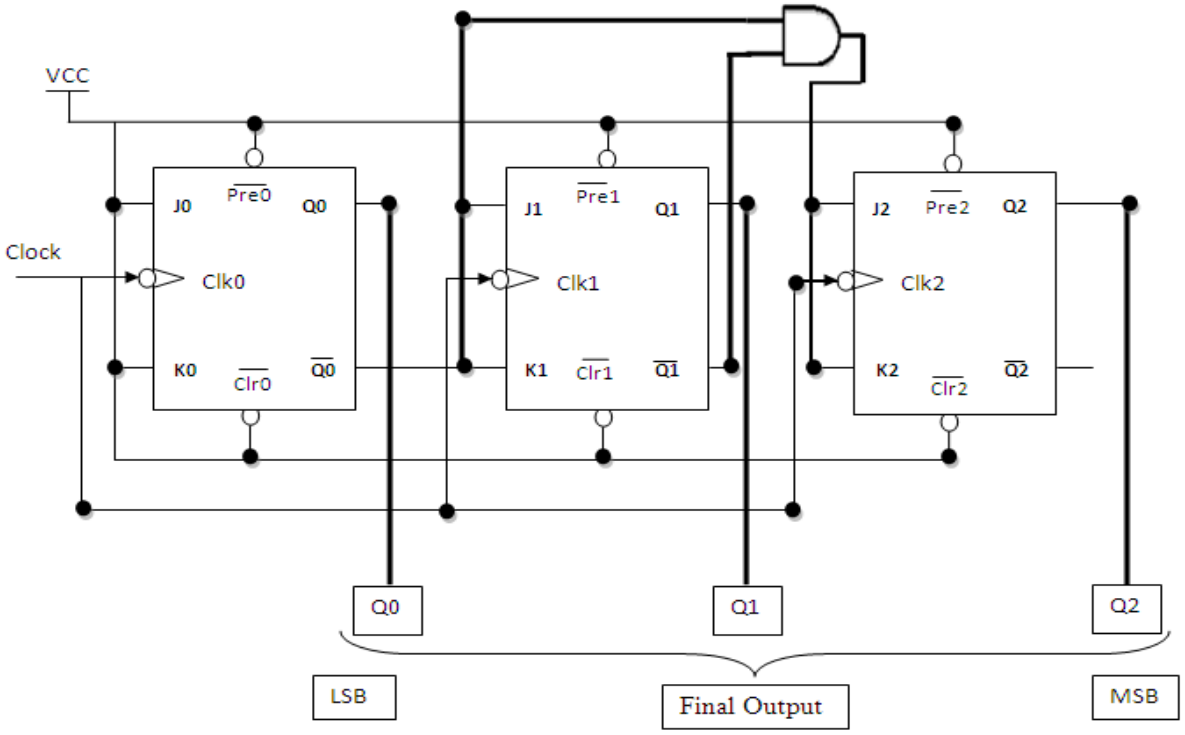


Fig : A 3 bit Synchronous down counter

Uses:

1. Specially used as the counting devices.
2. Used in frequency divider circuit.
3. Used in digital voltmeter.
4. Used in counter type A to D converter.
5. Used for time measurement..
6. It helps in counting the no of product coming out from machinery where product is coming out at equal interval of time.

Conclusion:

Up and down counters are successfully implemented, the counters are studied & o/p are checked. The state table is verified.

PRACTICE ASSIGNMENTS / EXERCISE / MODIFICATIONS:

1. Design & implement 2 bit controlled synchronous counter.
2. Design & implement 4 bit controlled synchronous counter.
3. Design & implement truncated synchronous up or down counter.

FAQ's with answers:

1. What do you mean by Counter?

   A Counter is a register capable of counting the no. of clock pulses arriving at its clock inputs. Count represents the no. of clock pulses arrived. A specified sequence of states appears as the counter output.

2. What are the types of Counters? Explain each.

   There are two types of counters as Asynchronous Counter and Synchronous Counter. Asynchronous Counter: In this counter, the first flip-flop is clocked by the external clock pulse and then each successive flip-flop is clocked by the Q or Q' o/p of the previous flip-flop. Hence in Asynchronous Counter flip-flops are not clocked simultaneously and hence called as Ripple Counter. Synchronous Counter: In this counter, the common clock input is connected to all the flip-flops simultaneously.

3. What do you mean by pre-settable counters?

   A counter in which starting state is not zero can be designed by making use of the preset inputs of the flip flops. This is referred to as loading the counter asynchronously. This is referred to as pre-settable counter.

4. What are the applications of synchronous counters?

   Digital clock
   Frequency divider circuits
   Frequency counters
   Used in analog to digital converters

5. What are the advantages of synchronous counters over asynchronous counters?

   Propagation delay time is reduced.
   Can operate at a much higher frequency than the asynchronous counters.

6. Ring counter is an example of synchronous counters or asynchronous counter?

   Synchronous counter. Since all the flip flops are clocked simultaneously.

7. Twisted Ring (Johnson's) counter is an example of synchronous counters or asynchronous counter?

   Synchronous counter. Since all the flip flops are clocked simultaneously.

8. What is the difference between ring counter and twisted ring counter?

   In ring counter pulses to be counted are applied to a counter , it goes from state to state and the output of the flip flop s in the counter is decoded to read the count. Here the uncomplimentary output (Q) of last flip flop is fed back as an input to first flip flop. Ring counters are referred as MOD 'N' counters.

   But in Twisted ring counter the complimentary output (Q bar) of last flip flop is fed back as an input to first flip flop. Twisted Ring counters are referred as MOD '2N' counters.

9. What are the applications of ring counters?

   Ring counter outputs are sequential non-overlapping pulses which are useful for control state counters, Used in stepper motor, this requires pulses to rotate it from one position to the next. Used as divide by 'N' ((MOD 'N') counters.

10. What are the applications of ring counter twisted ring counters?

    Used as divide by '2N' ((MOD '2N') counters.

    Used for control state counters.

    Used for generation of multiphase clock.

11. List the Synchronous Counter ICs.

    IC 74160 : Decade Up Counter

    IC 74161 : 4 bit binary Up Counter

    IC 74162 : Decade Up Counter

    IC 74163 : 4 bit binary Up Counter

    IC 74168 : Decade Up/Down Counter

    IC 74169 : 4 bit Binary Up/Down Counter

    IC 74190 : Decade Up/Down Counter

    IC 74191 : 4 bit Binary Up/Down Counter

    IC 74192 : Decade Up/Down Counter

    IC 74193 : 4 bit Binary Up/Down Counter

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Define Counter	1	2
2	Compare Combinational & Sequential circuits	4	2

3	Explain the role of clock in Sequential circuits	2	2
4	What do you mean by Preset & Clear?	1	2
5	What are the Asynchronous inputs in Sequential circuits	1	2
6	Define the truth table JK Flip-Flop.	1	2
7	Explain Latch	2	2
8	Explain Flip-Flop.	2	2
9	Differentiate between Flip-Flop & Latch.	3	2
10	Explain triggering mechanism in flip-flop.	2	2
11	Extend the 3 bit synchronous counter to 4 bit synchronous counter	2	2

References:

- R.P. Jain, "Modern Digital Electronics", 3rd Edition, Tata McGraw-Hill, ISBN: 0-07-049492-4
- “Digital Design”, M Morris Mano, Prentice Hall, 3rd Edition, ISBN: 0130621218.

Experiment No. 5

Title:- Modulus n Counter

AIM: To design and implement mod - 10, mod – 7, mod - 99 asynchronous BCD counter using IC 7490.

OBJECTIVE: To know difference between regular & truncated counter as well as binary & BCD Counter

IC's USED: IC 7490, Basic gates 7408,7432.

THEORY: IC 7490

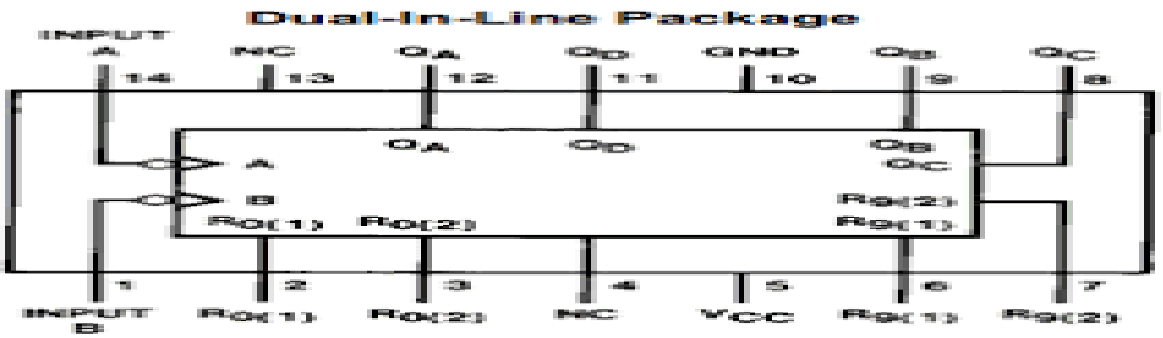
IC 7490 is a TTL MSI (medium scale integration) decade counter. It contains 4 master slave flip flops internally connected to provide MOD-2 i.e. divide by 2 and MOD-5 i.e. divide by 5 counters. MOD-2 and Mod-5 counters can be used independently or in cascading.

It is a 4-bit ripple type decade counter. The device consists of 4-master slave flip flops internally connected to provide a divide by two and divide by 5 sections. Each section has a separate clock i/p to initiate state changes of the counter on the high to low clock transition.

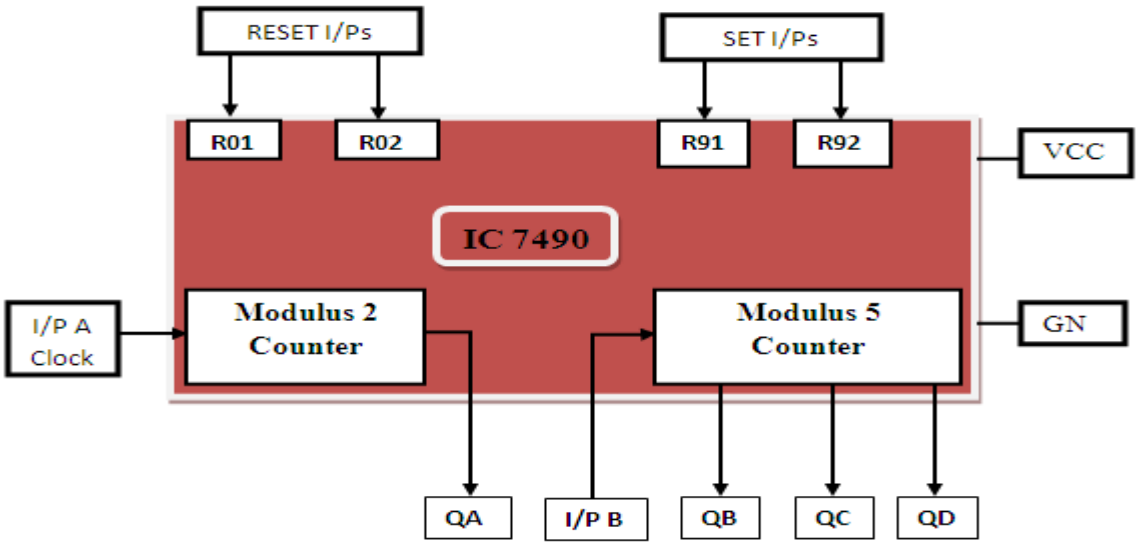
Since the o/p from the divide by 2 section is not internally connected to the succeeding stages. The device may be operated in various counting modes. In a BCD counter the CP_1 input must be externally connected to Q_A o/p. The CP_0 i/p receives the incoming count producing a BCD count sequence. It is also provided with additional gating to provide a divide by 2 counter and binary counter for which the count cycle length is divide by 5. The device may be operated in various counting modes.

There are 2 reset inputs $R_0(1)$ and $R_0(2)$ both of which need to be connected to the 'logic 1' for clearing all flip flops. Two set inputs $R_g(1)$ and $R_g(2)$ when connected to logic 1 are used for setting counter to 1001 (BCD 9).

Pin out of IC 7490:



Basic internal Structure of IC 7490:



Function Table of MOD-5 counter:

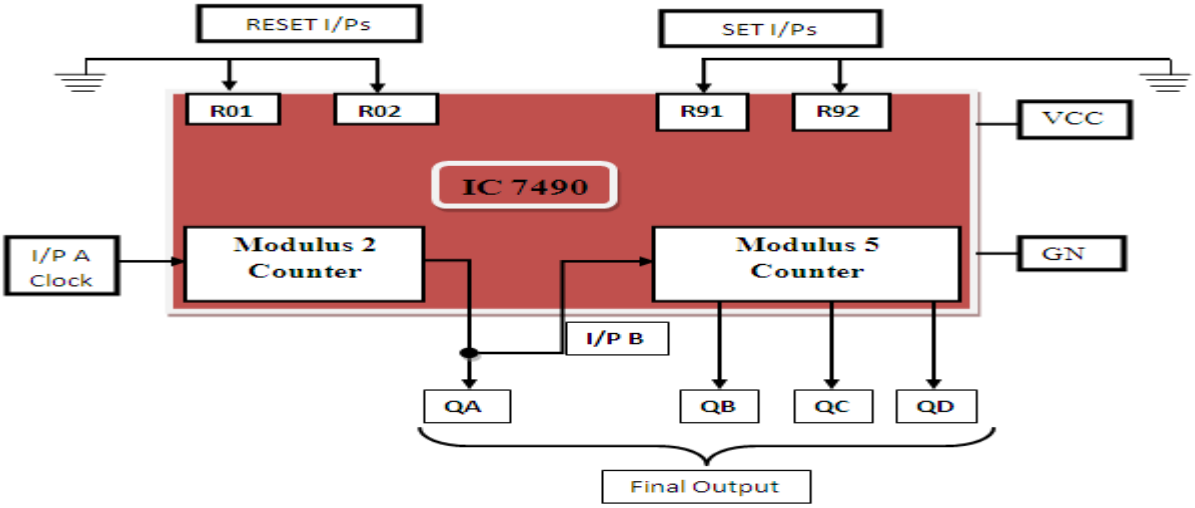
Input B clock	Output			Count
	QD	**QC**	**QB**	
	0	**0**	**0**	**0**
	0	**0**	**1**	**1**
	0	**1**	**0**	**2**
	0	**1**	**1**	**3**
	1	**0**	**0**	**4**

Design of MOD-10 counter using IC 7490:

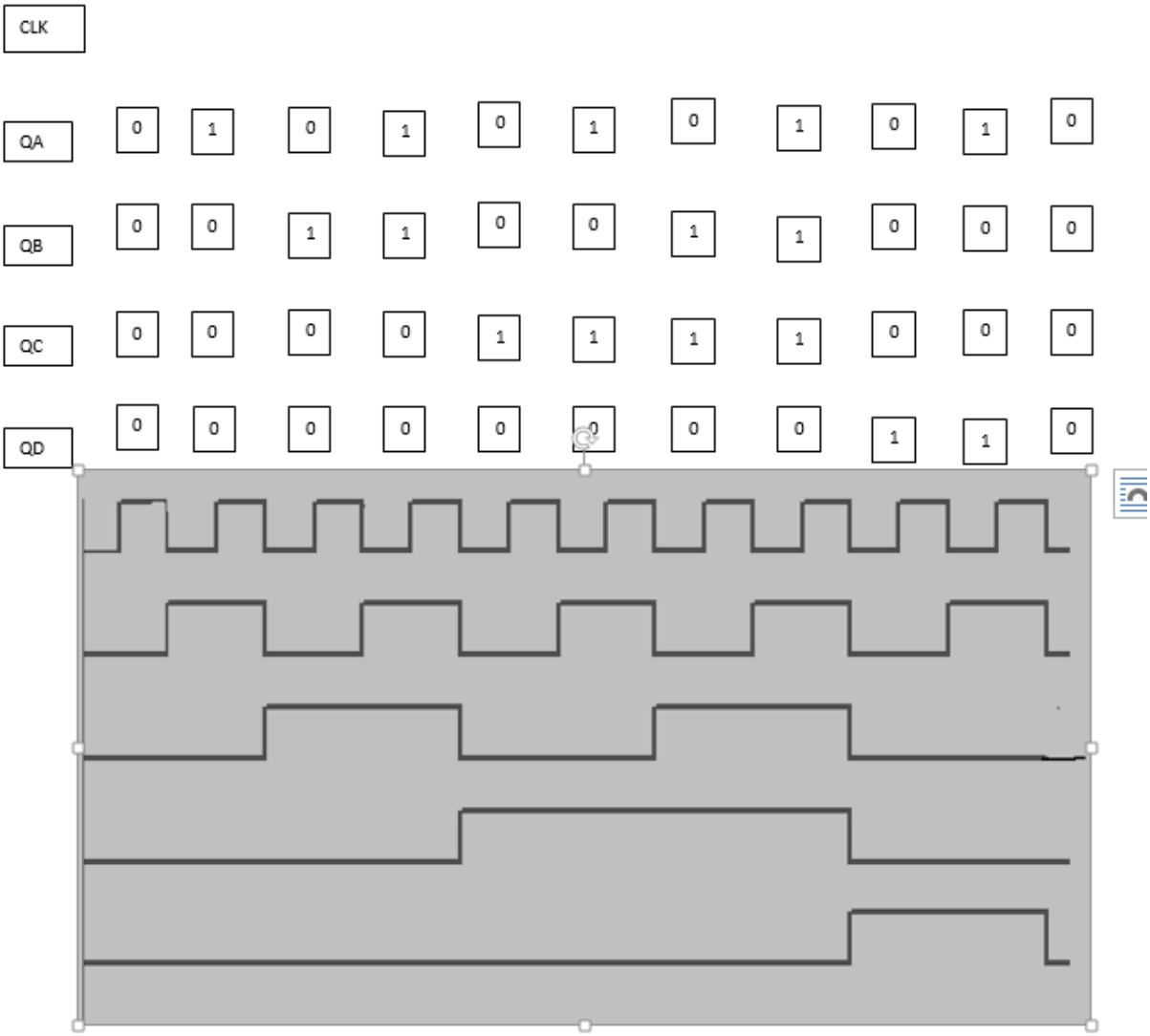
The QA o/p the first flip flop is connected to the input B which is clock i/p of internal MOD-5 ripple counter. Due to cascading of Mod-2 and Mod-5 counters, the overall configuration the decade counters count from 0000 to 1001. After 1001 mod-5 resets to 0000 and next count after 1001 is 0000.

When QA o/p is connected to B i/p, we have the Mod-2 counter followed by Mod-5 counter. The count sequence obtained is shown in the table. It may be noted that QA changes from 0 to 1 the state of Mod-5 counter doesn't change, whereas when QA changes from 1 to 0 the Mod-5 counter goes to the next state.

Logic Diagram MOD-10 counter using IC 7490:



Timing diagram of mod10:



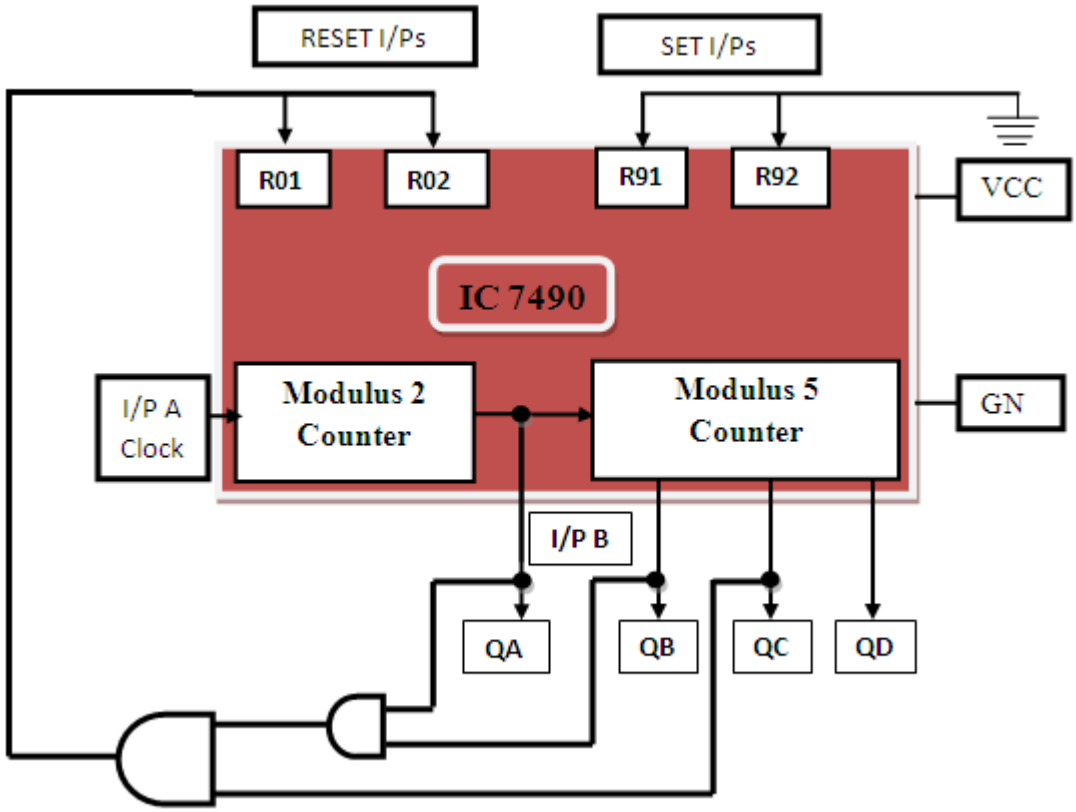
Design of Mod-7 Counter using IC 7490:

Mod-7 counter counts through seven states from 0 to 6 counters and it should reset as soon as the count becomes 7. The o/p of reset logic should be 1 corresponding to invalid states. The reset logic o/p should be applied to pin 2 and 3.

Truth Table of Reset Logic:

QD	QC	QB	QA	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1

Logic Diagram Mod 7 Counter using IC 7490 :

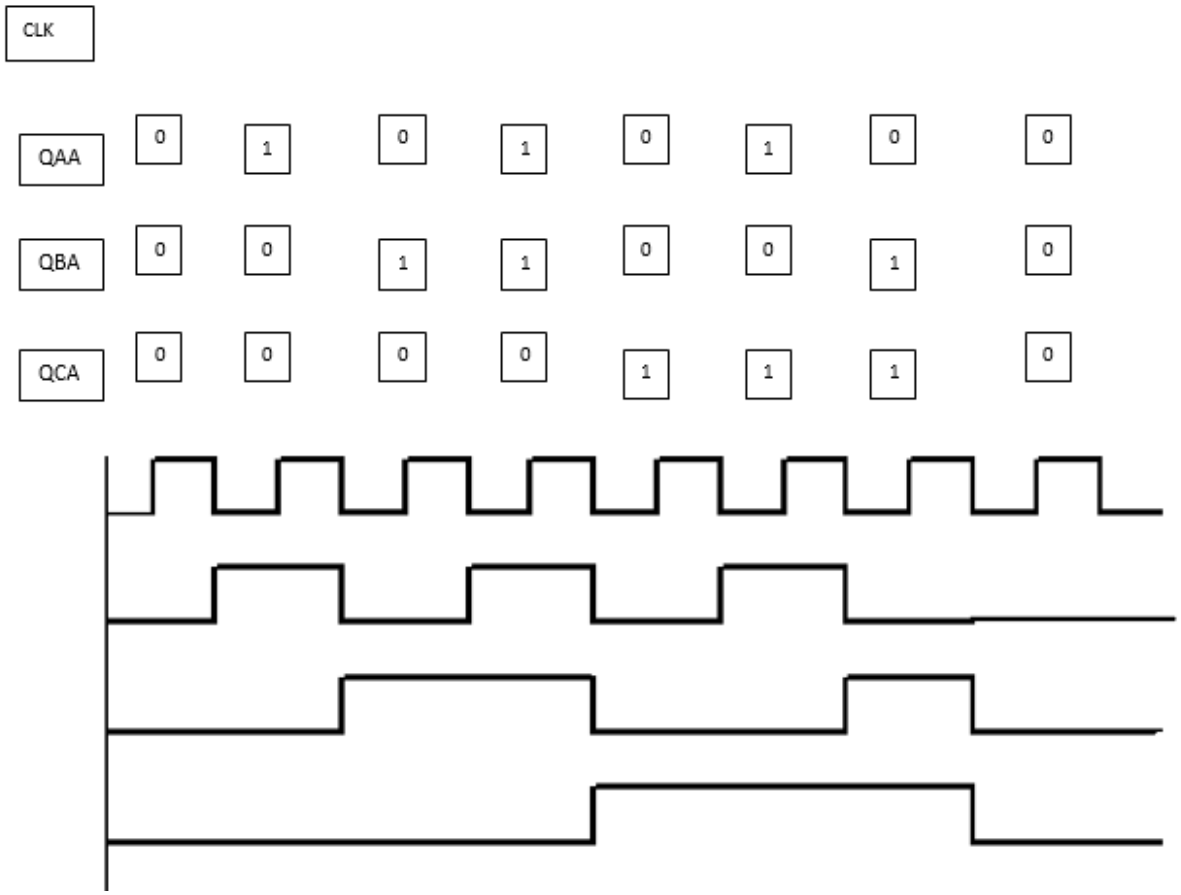


Function table:

	Output	

I/p clock	**QD**	**QC**	**QB**	**QA**	Count
	0	**0**	**0**	**0**	**0**
	0	**0**	**0**	**1**	**1**
	0	**0**	**1**	**0**	**2**
	0	**0**	**1**	**1**	**3**
	0	**1**	**0**	**0**	**4**
	0	**1**	**0**	**1**	**5**
	0	**1**	**1**	**0**	**6**

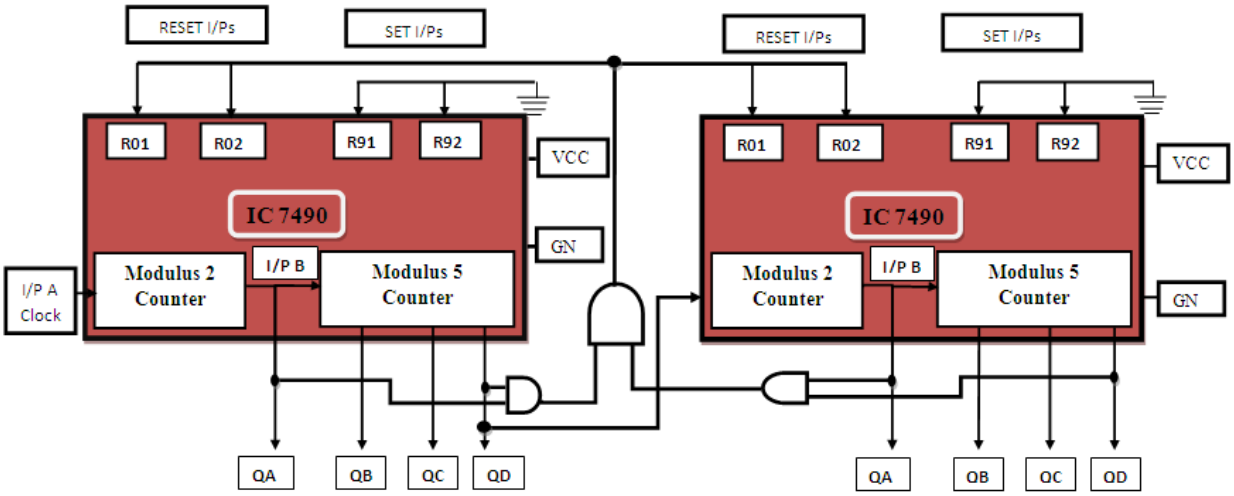
Timing diagram of mod7:



Design of Mod-99 using IC 7490:

For Mod-99 two IC 7490's will be required. Hence to implement a divide by 99 counter we have to use two decade counters IC's. A divide by 99 counter counts 99 states from 0 to 98 and the counter should reset as soon as the count becomes

99. So in order to reset the counter of 99 connect the Q o/p which are equal to 1 in the count of 99 to an 'And' gate & then connect and o/p to the reset i/p of both IC's.



FAQs:

1. What do you mean modulus counter?

It represents the number of possible states of counter.

2. How will you use the 7490 IC to design symmetrical divide by 10 frequency counter?

The divide by 5 circuit followed by divide by 2 circuit will give symmetrical output.

1. Where counters are used? Give real life example of counter.

Binary counter – An N stage counter that recycles after 2^N count. The count proceds in specified binary sequence.

5. Counter, Presetable- A counter which can be set to a desired value before the start of the counting/

6. UP/Down counter – A counter that can count in both up and down direction depending upon a control input.

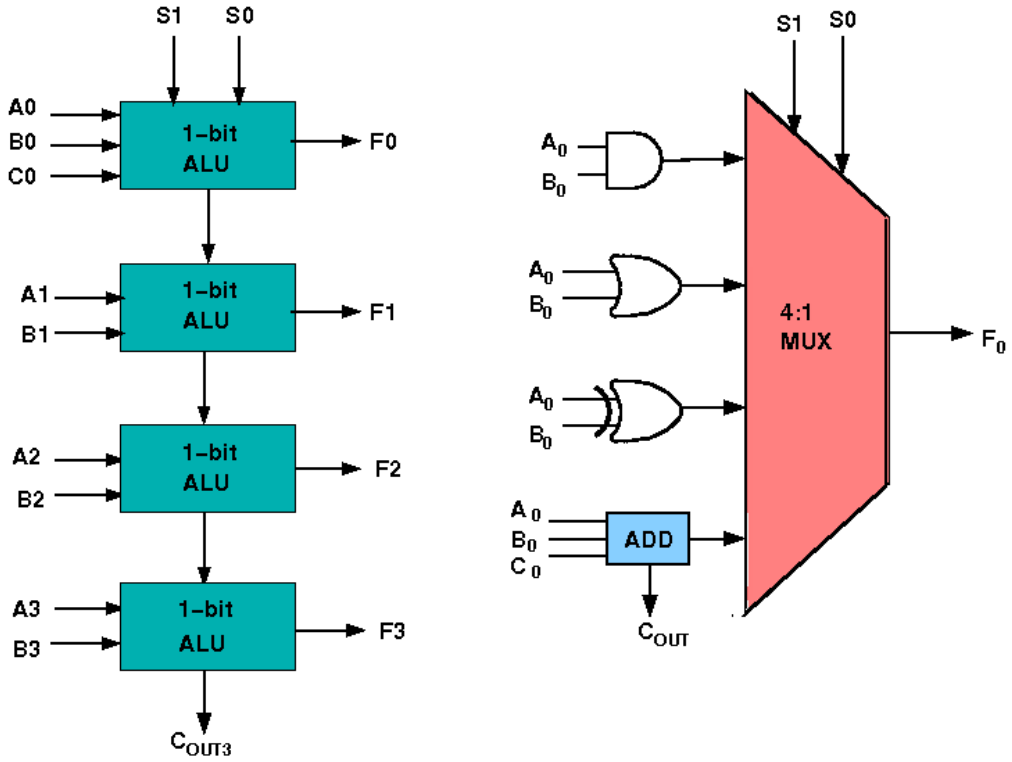
Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	What is the name of IC 7490 & 74191?	1	2
2	Identify Whether 7490 is Synchronous or Asynchronous?	2	3
3	How many flip-flops are present in IC 7490 & 74191.	1	2
4	What is Biquinary Counter ?	1	3
5	What is Truncated Counter ?	1	3
6	Identify the nature of circuit. (is Synchronous or Asynchronous)	3	2

Experiment No. 6:- Design of Arithmetic Logic Unit:

Theory for **Design of ALU**

ALU or Arithmetic Logical Unit is a digital circuit to do arithmetic operations like addition, subtraction,division, multiplication and logical oparations like and, or, xor, nand, nor etc. A simple block diagram of a 4 bit ALU for operations and,or,xor and Add is shown here :



The 4-bit ALU block is combined using 4 1-bit ALU block

Design Issues :

The circuit functionality of a 1 bit ALU is shown here, depending upon the control signal S_1 and S_0 the circuit operates as follows:

for Control signal $\mathbf{S_1 = 0}$, $S_0 = 0$, the output is **A And B**,

for Control signal $\mathbf{S_1 = 0}$, $S_0 = 1$, the output is **A Or B**,

for Control signal $\mathbf{S_1 = 1}$, $S_0 = 0$, the output is **A Xor B**,

for Control signal $\mathbf{S_1 = 1}$, $S_0 = 1$, the output is **A Add B**.

The truth table for 16-bit ALU with capabilities similar to 74181 is shown here:

Required functionality of ALU (inputs and outputs are active high)

Mode Select				**F_n for active HIGH operands**	
Inputs				**Logic**	**Arithmetic (note 2)**
S3	S2	S1	S0	(M = H)	(M = L) **(C_n=L)**
L	L	L	L	A'	A
L	L	L	H	A'+B'	A+B

L	L	H	L	A'B	A+B'
L	L	H	H	Logic 0	minus 1
L	H	L	L	(AB)'	A plus AB'
L	H	L	H	B'	(A + B) plus AB'
L	H	H	L	A ⊕ B	A minus B minus 1
L	H	H	H	AB'	AB minus 1
H	L	L	L	A'+B	A plus AB
H	L	L	H	(A ⊕ B)'	A plus B
H	L	H	L	B	(A + B') plus AB
H	L	H	H	AB	AB minus 1
H	H	L	L	Logic 1	A plus A (Note 1)
H	H	L	H	A+B'	(A + B) plus A
H	H	H	L	A+B	(A + B') plus A
H	H	H	H	A	A minus 1

The L denotes the logic low and H denotes logic high.

Objective

Objective of 4 bit arithmetic logic unit (with AND, OR, XOR, ADD operation):

1. Understanding behaviour of arithmetic logic unit from working module and the module designed by the student as part of the experiment
2. Designing an arithmetic logic unit for given parameter

Examining behaviour of arithmetic logic unit for the working module and module designed by the student as part of the experiment (refer to the circuit diagram):

Loading data in the arithmetic logic unit (refer to procedure tab for further detail and experiment manual for pin numbers):

- load the two input numbers as:
    - A(A3 A2 A1 A0): A3=1, A2=1, A1=0, A0=0
    - B(B3 B2 B1 B0): B3=1, B2=0, B1=0, B0=1
    - carry in(C0)=0

examining the AND behaviour:

- load data in select input as:
    - S1=0, S0=0 `
- check output:
    - F3=1, F2=0, F1=0, F0=0
    - cout=0 `

examining the OR behaviour:

- load data in select input as:
    - S1=0, S0=1 `
- check output:
    - F3=1, F2=1, F1=0, F0=1
    - cout=0 `

examining the XOR behaviour:

- load data in select input as:

    - S1=1, S0=0 `
- check output:
    - F3=0, F2=1, F1=0, F0=1
    - cout=0 `

examining the ADD behaviour:

- load data in select input as:
    - S1=1, S0=1 `
- check output:
    - F3=0, F2=1, F1=0, F0=1
    - cout=1 `

Recommended learning activities for the experiment: Leaning activities are designed in two stages, a basic stage and an advanced stage. Accomplishment of each stage can be self-evaluated through the given set of quiz questions consisting of multiple type and subjective type questions. In the basic stage, it is recommended to perform the experiment firstly, on the given encapsulated working module, secondly, on the module designed by the student, having gone through the theory, objective and procuder. By performing the experiment on the working module, students can only observe the input-output behavior. Where as, performing experiments on the designed module, students can do circuit analysis, error analysis in addition with the input-output behavior. It is recommended to perform the experiments following the given guideline to check behavior and test plans along with their own circuit analysis. Then students are recommended to move on to the advanced stage. The advanced stage includes the accomplishment of the given assignments which will provide deeper understanding of the topic with innovative circuit design experience. At any time, students can mature their knowledge base by further reading the references provided for the experiment.

- color configuration of wire for 5 valued logic supported by the simulator:
- if value is UNKNOWN, wire color= maroon
- if value is TRUE, wire color= blue
- if value is FALSE, wire color= black
- if value is HI IMPEDENCE, wire color= green
- if value is INVALID, wire color= orange

likewise the 16 bit arithmetic logic unit can be designed and tested

- by cascading 4 bit ALUs only the carry will propagate to the next level for ADD operation

Test plan :

1. Set inputs 0101 and 0011 and check output for all possible select input combinations.
2. Set any two 16-bit number and check output for all possible select input combinations.

Use Display units for checking output. Try to use minimum number of components to build. The pin configuration of the canned components are shown when mouse hovered over a component.

Assignment Statements :

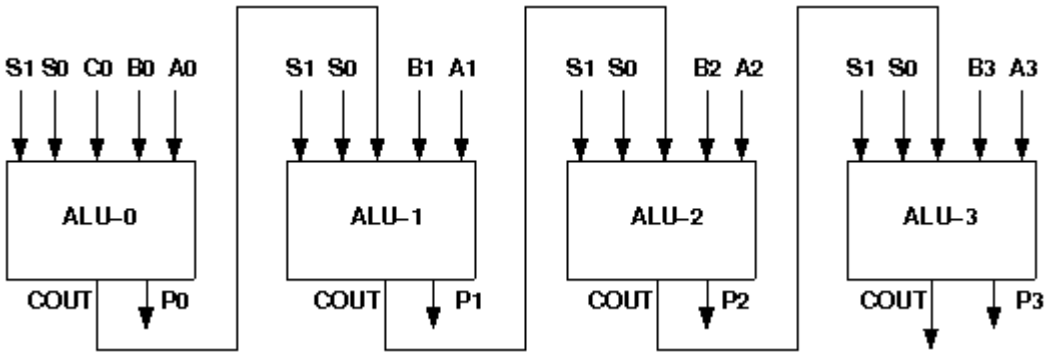
1. Design a 4 bit ALU comprising only the AND, OR, XOR and Add operations.
2. Design a 16-bit ALU with capabilities similar to 74181

Procedure
Design of ALU:

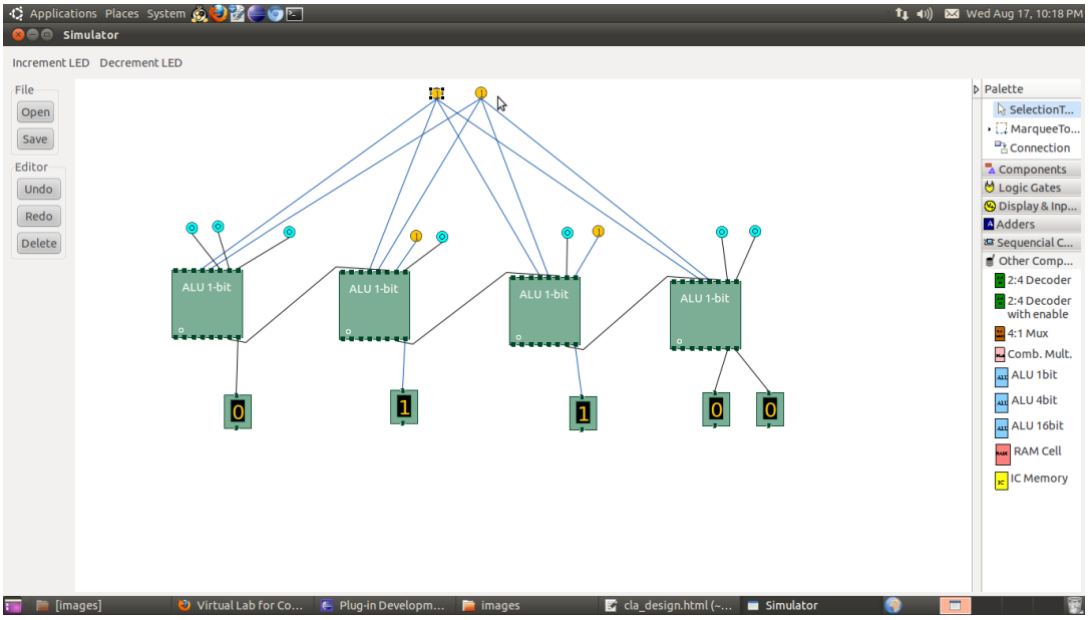
Procedure to perform the experiment: Design of 4 bit ALU

1. Start the simulator as directed. This simulator supports 5-valued logic.
2. To design the circuit we need 4 1-bit ALU, 11 Bit switch (to give input,which will toggle its value with a double click), 5 Bit displays (for seeing output), wires.
3. The pin configuration of a component is shown whenever the mouse is hovered on any canned component of the palette. Pin numbering starts from 1 and from the bottom left corner (indicating with the circle) and increases anticlockwise.
4. For 1-bit ALU input A0 is in pin-9,B0 is in pin-10, C0 is in pin-11 (this is input carry), for selection of operation, S0 is in pin-12, S1 is in pin-13, output F is in pin-8 and output carry is pin-7
5. Click on the 1-bit ALU component (in the Other Component drawer in the pallet) and then click on the position of the editor window where you want to add the component (no drag and drop, simple click will serve the purpose), likewise add 3 more 1-bit ALU (from the Other Component drawer in the pallet), 11 Bit switches and 5 Bit Displays (from Display and Input drawer of the pallet, if it is not seen scroll down in the drawer), 3 digital display and 1 bit Displays (from Display and Input drawer of the pallet, if it is not seen scroll down in the drawer)
6. To connect any two components select the Connection menu of Palette, and then click on the Source terminal and click on the target terminal. According to the circuit diagram connect all the components. Connect the Bit switches with the inputs and Bit displays component with the outputs. After the connection is over click the selection tool in the pallete.
7. See the output, in the screenshot diagram we have given the value of S1 S0=11 which will perform add operation and two number input as A0 A1 A2 A3=0010 and B0 B1 B2 B3=0100 so get output F0 F1 F2 F3=0110 as sum and 0 as carry which is indeed an add operation.you can also use many other combination of different values and check the result. The operations are implemented using the truth table for 4 bit ALU given in the theory.

Circuit diagram of 4 bit ALU:



Screenshot of Design of 4 bit ALU:



Components :

To build any 4 bit ALU, we need :

1. AND gate, OR gate, XOR gate
2. Full Adder,
3. 4-to-1 MUX
4. Wires to connect.

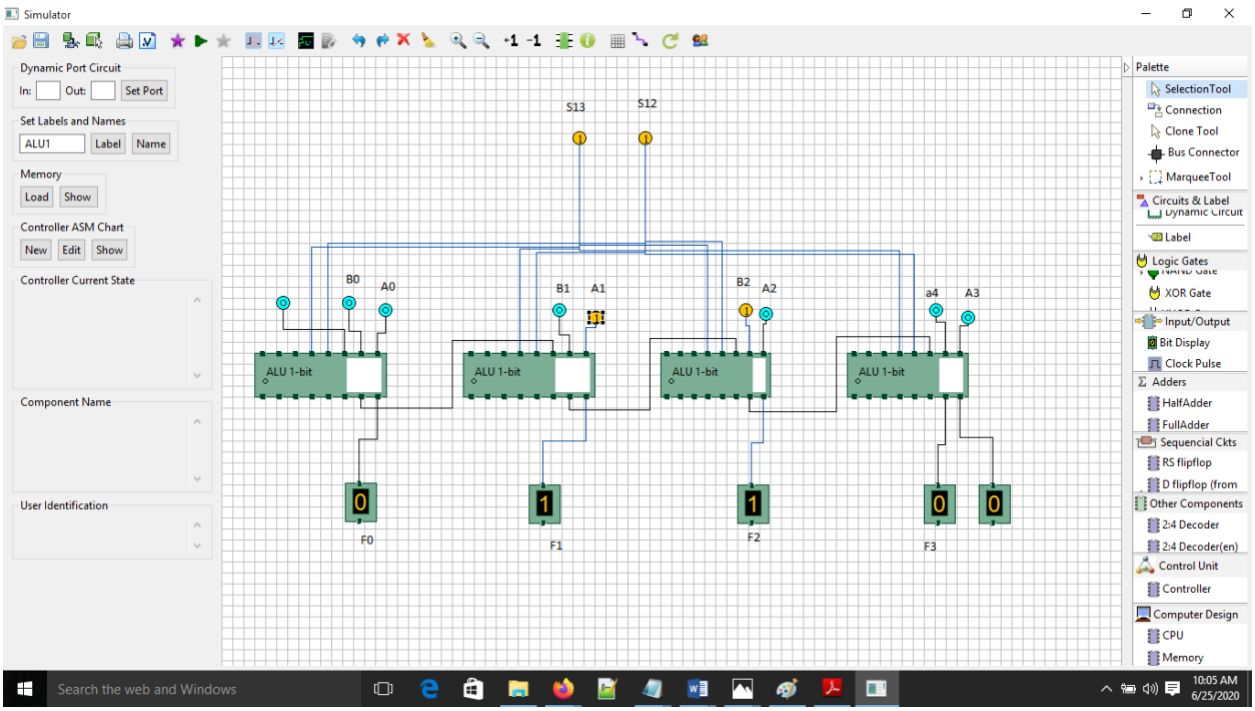
In case of counters the number of flip-flops depends on the number of different states in the counter.

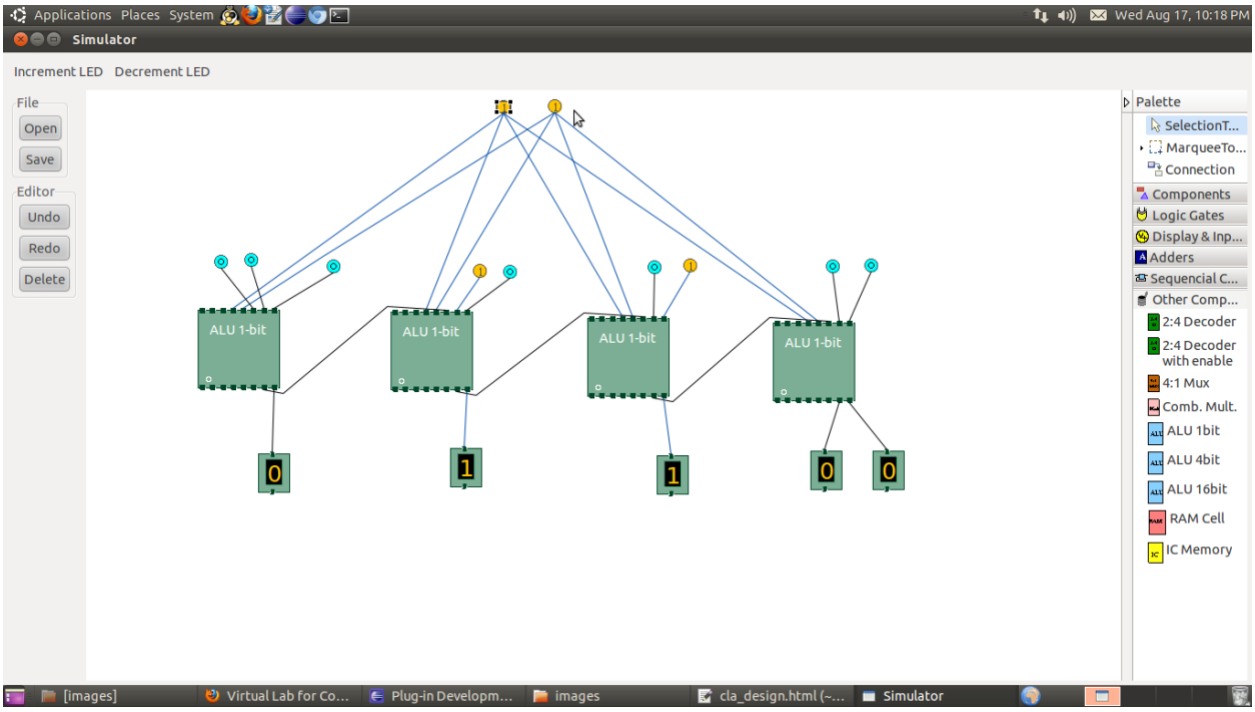
Experiment

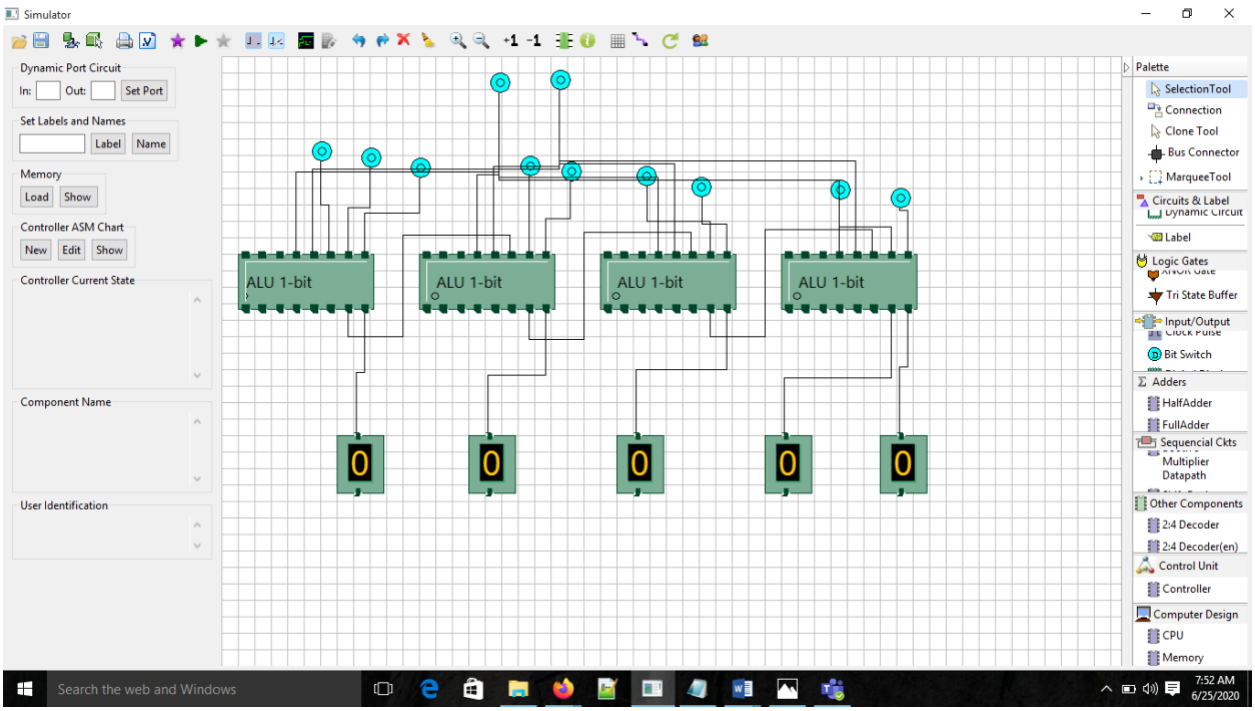
Design of ALU :

General guideline to use the simulator for performing the experiment:

- Start the simulator as directed. For more detail please refer to the manual for using the simulator
- The simulator supports 5-valued logic
- To add the logic components to the editor or canvas (where you build the circuit) select any component and click on the position of the canvas where you want to add the component
- The pin configuration is shown when you select the component and press the 'show pinconfig' button in the left toolbar or whenever the mouse is hovered on any canned component of palette
- To connect any two components select the connection tool of palette, and then click on the source terminal and then click on the the target terminal
- To move any component select the component using the selection tool and drag the component to the desired position
- To give a toggle input to the circuit, use 'Bit Switch' which will toggle its value with a double click
- Use 'Bit Display' component to see any single bit value. 'Digital Display' will show the output in digital format
- undo/redo, delete, zoom in/zoom out, and other functionalities have been given in the top toolbar for ease of circuit building
- Use start/stop clock pulse to start or stop the clock input of the circuit. Clock period can be set from the given 'set clock' button in the left toolbar
- Use 'plot graph' button to see input-output wave forms
- Users can save their circuits with .logic extension and reuse them
- **After building the circuit press the simulate button in the top toolbar to get the output**
- **If the circuit contains a clock pulse input, then the 'start clock' button will start the simulation of the whole circuit. Then there is no need to again press the 'simulate' button**


Simulator
Dynamic Port Circuit
In: Out: Set Port
Set Labels and Names
ALU1 Label Name
Memory
Load Show
Controller ASM Chart
New Edit Show
Controller Current State
Component Name
User Identification
S13
S12
B0
A0
B1
A1
B2
A2
a4
A3
ALU 1-bit
ALU 1-bit
ALU 1-bit
ALU 1-bit
0
1
1
0
0
F0
F1
F2
F3
Palette
SelectionTool
Connection
Clone Tool
Bus Connector
MarqueeTool
Circuits & Label
Label
Logic Gates
XOR Gate
Input/Output
Bit Display
Clock Pulse
Adders
HalfAdder
FullAdder
Sequencial Ckts
RS flipflop
D flipflop (from
Other Components
2:4 Decoder
2:4 Decoder(en)
Control Unit
Controller
Computer Design
CPU
Memory
Search the web and Windows
10:05 AM
6/25/2020


Applications Places System
Wed Aug 17, 10:18 PM
Simulator
Increment LED Decrement LED
File
Open
Save
Editor
Undo
Redo
Delete
ALU 1-bit
ALU 1-bit
ALU 1-bit
ALU 1-bit
0
1
1
0
0
Palette
SelectionT...
MarqueeTo...
Connection
Components
Logic Gates
Display & Inp...
Adders
Sequencial C...
Other Comp...
2:4 Decoder
2:4 Decoder with enable
4:1 Mux
Comb. Mult.
ALU 1bit
ALU 4bit
ALU 16bit
RAM Cell
IC Memory
[images]
Virtual Lab for Co...
Plug-in Developm...
images
cla_design.html (~...
Simulator


Simulator
Dynamic Port Circuit
In:
Out:
Set Port
Set Labels and Names
Label
Name
Memory
Load
Show
Controller ASM Chart
New
Edit
Show
Controller Current State
Component Name
User Identification
ALU 1-bit
ALU 1-bit
ALU 1-bit
ALU 1-bit
0
0
0
0
0
Palette
SelectionTool
Connection
Clone Tool
Bus Connector
MarqueeTool
Circuits & Label
Label
Logic Gates
Tri State Buffer
Input/Output
Bit Switch
Adders
HalfAdder
FullAdder
Sequencial Ckts
Multiplier
Datapath
Other Components
2:4 Decoder
2:4 Decoder(en)
Control Unit
Controller
Computer Design
CPU
Memory
Search the web and Windows
7:52 AM
6/25/2020