SND COLLEGE OF ENGINEERING & RC,YEOLA

LAB MANUAL

Subject: LOGIC DESIGN AND COMPUTER ORGANIZATION [214446]

Department of Information Technology

VISION

The information Technology Department is committed to provide high academic goals to the students and make them the world leaders, both in Educational and Research, through effective Teaching and Learning!

MISSION

- To produce the best quality professionals by imparting quality training, hands on experience and values education.
- To pursue new technologies in cross discipline in order to serve the needs of industry, society, and the scientific community.
- Establish Industry Institute Interaction program to enhance the entrepreneurship skills.
- Promote research based projects/activities in the emerging areas of technology convergence.

PROGRAM SPECIFIC OUTCOMES (PSOs)

At the end of program, Information Technology students will be able to:

PSO01: Apply the fundamentals of mathematics, science and engineering knowledge to understand, analyze and develop computer programs in the areas related to Algorithms, Multimedia, Big Data Analytics, Machine Learning, Artificial Intelligence and Networking for efficient design of IT systems of varying complexity.

PSO02: Apply appropriate techniques and modern engineering hardware and software tools for the design and integration of computer system and related technologies, to engage in lifelong learning for the advancement of technology and its adaptation in multi-disciplinary environments.

PSO03: Implementation of professional engineering solutions for the betterment of society keeping the environmental context in mind, be aware of professional ethics and be able to communicate effectively.

PROGRAM OUTCOMES (POs)

Engineering Learners are expected to know and be able to–

PO-1 Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-2 Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-3 Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO-4 Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO-5 Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO-6 The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-7 Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-8 Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO-9 Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10 Communication: Communicate effectively on complex engineering activities with the 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.

PO-11 Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12 Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

TABLE OF CONTENTS

List of Laboratory Assignments

Group A

Combinational Logic Design– CO1

1. Design and implement 4-bit BCD to Excess-3 code
2. Design and implement 1 digit BCD adder usingIC7483
3. Design and implement following using multiplexer IC 74153 1) full adder 2) Any three variable function (cascade method)
4. Design and implement full subtractor using decoder IC 74138

Group B

Sequential Logic Design– CO 2

1. Design and implement 3 bit Up and 3 bit Down Asynchronous Counters using master slave JK flip-flop IC 7476
2. Design and implement 3 bit Up and 3 bit Down Synchronous Counters using master slave JK flip-flop IC 7476
3. Design and implement Modulo 'N' counter using IC7490. (N= 100 max)

Group C

Computer organization– CO 3

Any **two** of following, using virtual lab simulator

1. Design& simulate single bit RAM cell **OR** 4 address*2bit memory using 8 single bit RAM cells.
2. Design& simulate single bit ALU with four functions (AND, OR, XOR, ADD).
3. Design& simulation of single instruction CPU.

Note - Instructor should take care that datasheets of all the required ICs are available in the laboratory& students will be able to verify the functionality of ICs being used.

Reference Books:

1.R.P. Jain, "Modern Digital Electronics", 3rdISBN:0-07-049492-4.

2. Virtual Lab simulator Link **http://vlabs.iitkgp.ac.in/coa/**

CO-PO Mapping:

CO	Program outcomes											
	PO1	PO2	PO3	PO4	PO5	PO 6	PO 7	PO 8	PO 9	PO1 0	PO1 1	PO12
C206. 1	**3**	**3**	3	**2**	**2**	--	--	--	--	--	--	--
C206. 2	**3**	**3**	**3**	**2**	**2**	--	**2**	--	**2**	--	--	**2**
C206. 3	**3**	**3**	**3**	**2**	--	--	--	--	--	--	--	--
C206	**3**	**3**	**3**	**2**	**2**		**2**		**2**			**2**

CO-PSO Mapping:

CO	PSO1	PSO2	PSO3
C206.1	3	--	--
C206.2	3	2	--
C206.3	3	--	--
C206	3	2	

Assignment No: 01

Title: Design (truth table, K-map) and implementation of 4-bit BCD to Excess-3 Code converter

Assignment No: 01

Aim: Design and implementation of 4-bit Code convertors.

i) BCD to Excess – 3 Code

Objectives: To Understand and Design 4-bit Code Convertors.

Software Used (if Applicable) / Programming Languages Used / Hardware Used: IC's:7432 (OR-gate), 7408 (AND-gate), 7486 (Ex-or gate)

Theory:

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. 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.

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. It is easy to convert BCD code numbers and the familiar decimal numbers.With four bits, sixteen numbers (0000 to 1111) Binary Number can be represented, but in BCDcode only 10 of these are used. The six code combinations (1010 to 1111) are not used in BCD and are invalid.

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. Thecode 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.

BCD to Excess – 3 Code Conversions:

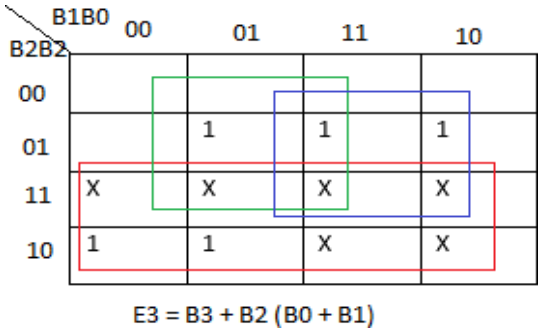
For converting 4 bit BCD code to Excess – 3, add 0011 i. e. decimal 3 to the respective code using rules of binary addition.

BCD to Excess-3 Code Conversion: Truth Table:

BCD INPUT				EXCESS-3 OUPUT			
B3	B2	B1	B0	E3	E2	E1	E0
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

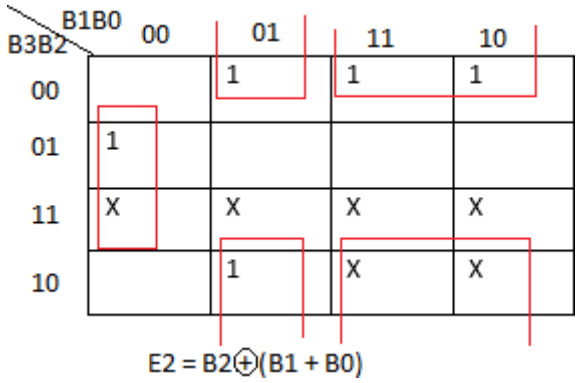
K-Map for Reduced Boolean Expressions of Each Output for BCD to Excess-3 Code Conversion:

K-Map for E3:-

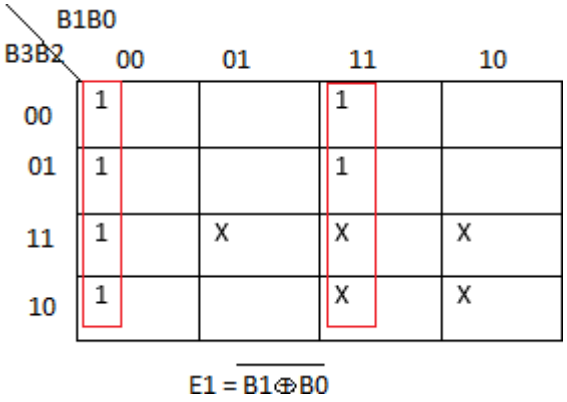


E3 = B3 + B2 (B0 + B1)

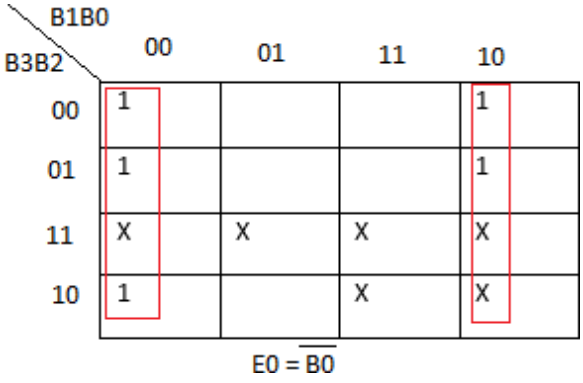
K-Map for E2:-



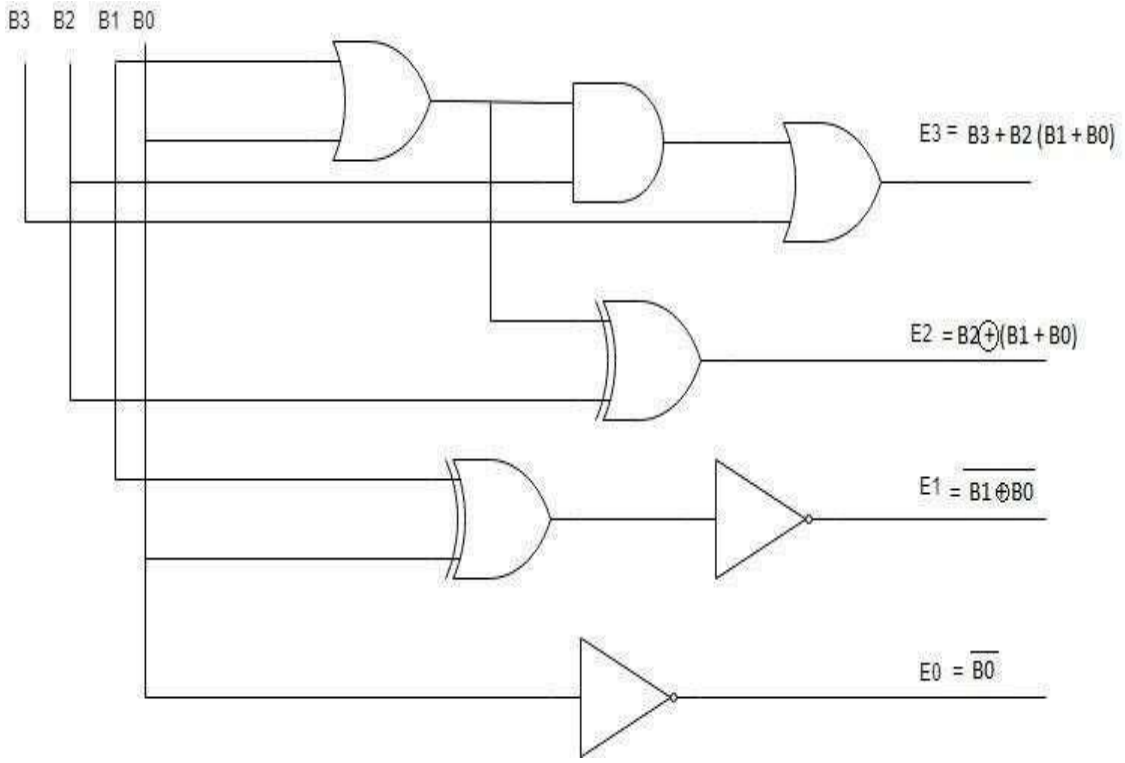
K-Map for E1:-



K-Map for E0:-



Circuit Diagram: BCD TO EXCESS-3 CONVERTER



Conclusion: Thus, we studied BCD & Excess-3 Code and also studied their conversions including applications. The truth tables have been verified using IC 7486, 7432, 7408, and 7404.

FAQ:

1) What is the need of code converters?

2) What are weighted codes and non-weighted codes?

3) Why is Excess-3 code called as self-complementing code?

4) What is invalid BCD?

Assignment No: 02

Title: Design & implement of 1-digit BCD adder using IC 7483

Assignment No: 02

Aim: Design & implement of 1-digit BCD adder using IC 7483.

Objectives: To Understand and Design single digit BCD & Excess adder.

Software Used (if Applicable) / Programming Languages Used / Hardware Used: IC's:7432 (OR-gate), 7408 (AND-gate), 7486 (Ex-or gate), 7483 (4 bit Binary adder)

Theory:

BCDAdder:

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: If Sum <= 9 & carry = 0, then answer is valid BCD number & so 0110 is not added.

CASE II: IfSum > 9 & carry = 0, then answer is invalid BCD number so 0110 is to be added. Otherwise answer is valid no need to add 0110.

CASE III: IfSum < = 9 & carry = 1, then answer is invalid BCD so 0110 is to be added. Otherwise answer is valid no need to add 0110.

Design of BCD adder:

1. To execute first step i. e. binary addition of two 4 bit numbers we will use IC 7483 (with $C_{in} = 0$), 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 (S_3, S_2, S_1, S_0) & only 1 output Y.

1. Y output of this circuit will be ORed with carry output of first adder IC 7483.
2. 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.
3. 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 outputs carry & carry output of second adder IC is ignored.

Pin diagram of 7483:

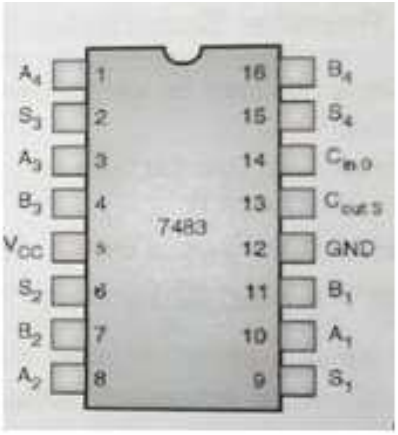


Fig1. Pin diagram of IC 7483

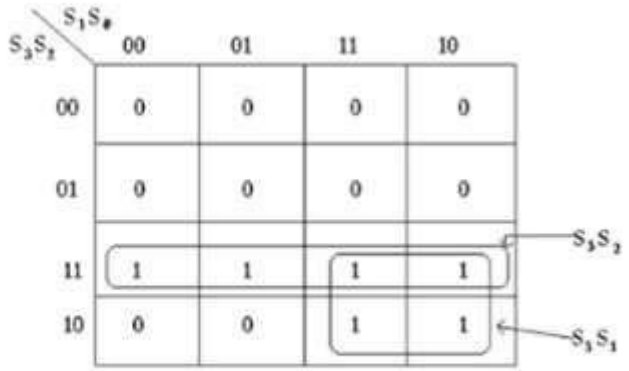
1) Truth Table for design of combinational circuit for BCD adder to check invalid BCD:

Inputs				Output
S_3	S_2	S_1	S_0	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	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Sum is invalid BCD number. Hence Y= 1

K-map for reduced Boolean expressions of output:

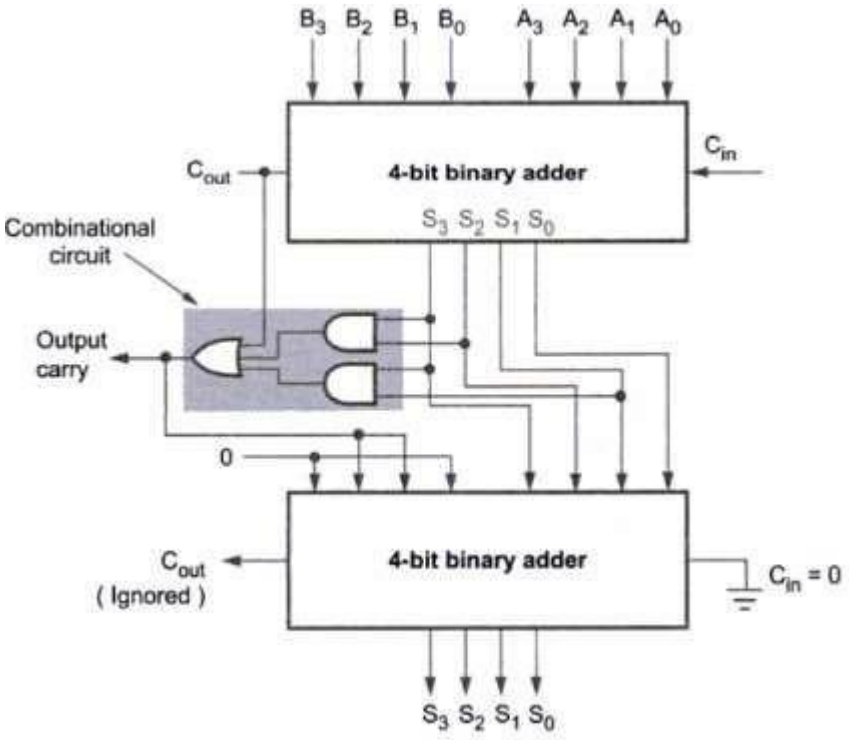
K-map:



The Boolean expression is

$Y = S_3S_2 + S_3S_1$

Circuit diagram for BCDAdder:



Conclusion:

BCD adder is designed & tested for all possible combinations.

FAQ:

Q1) Explain and Write the significance of BCD number system.

Q2) Write the applications of BCD code and explain the rules of BCD arithmetic.

Q4) what is the difference between BCD and binary codes?

Assignment No: 03

Title: Design and implement following using multiplexer IC 74153

1) Full adder
2) Any three variable function (cascade method)

Assignment No: 03

Aim:

Design and implement following using multiplexer IC 74153 1) Full adder 2) Any three variable function (cascade method)

Objectives: To Understand and Design Full adder using dual 4:1 IC 74153.

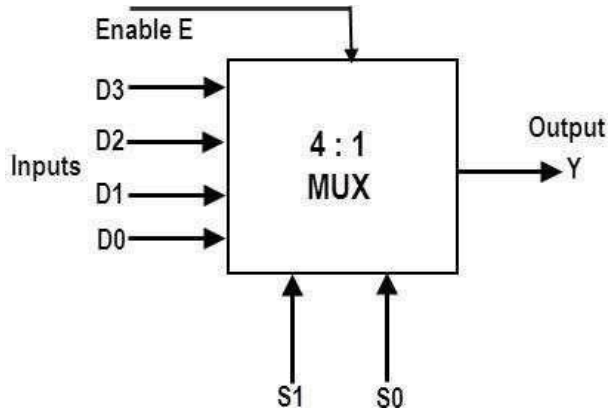
Software Used (if Applicable) / Programming Languages Used / Hardware Used: MUX(IC 74153), IC 7432(OR)

Theory:

BCDAdder:

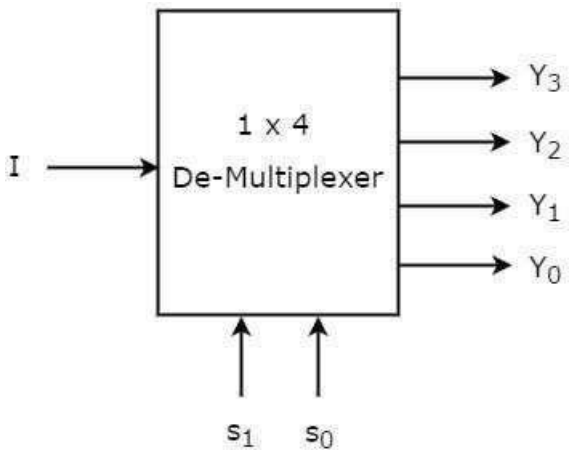
Digital Multiplexer:

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.



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 & 2m 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

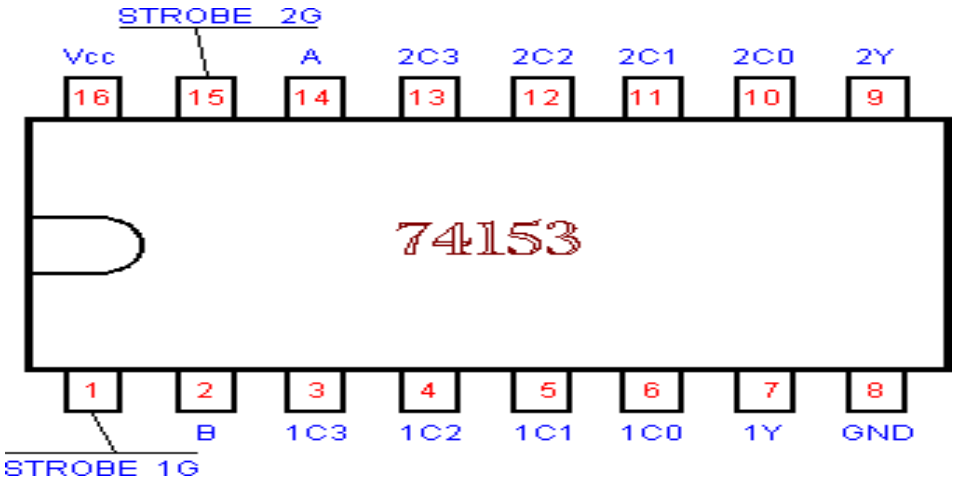
Function implementation:

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.

Pin Diagram of IC 74153(MUX):

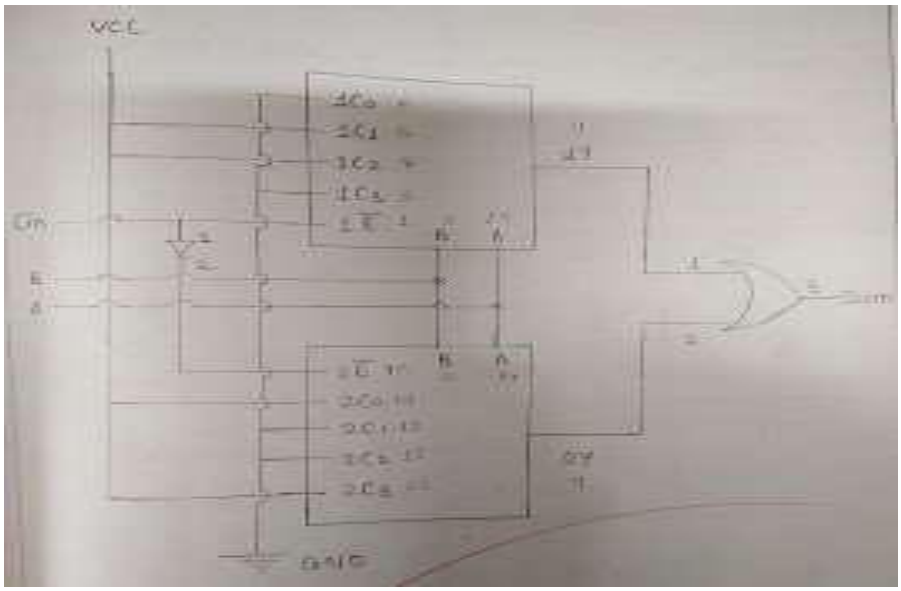


Truth table of Full Adder:

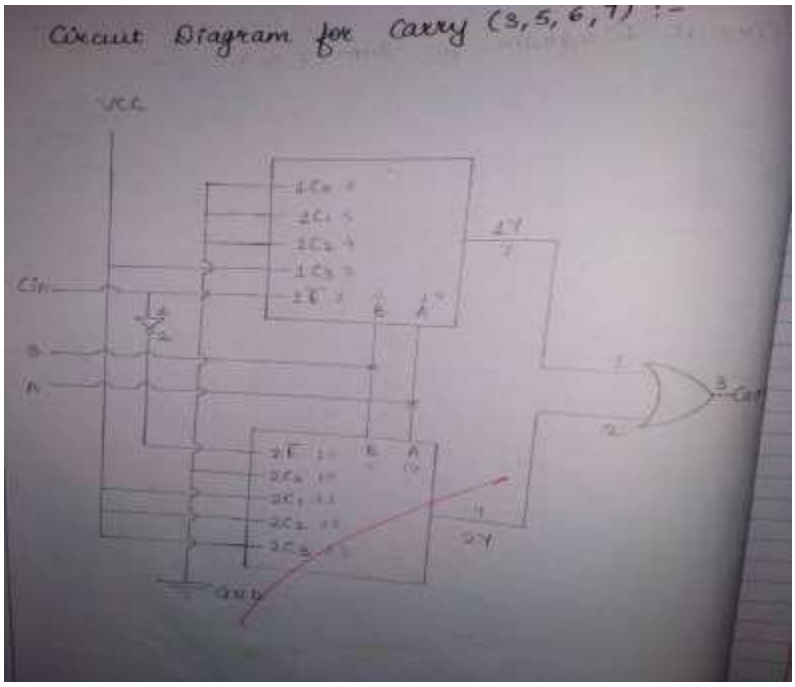
Full Adder Truth table

INPUTS			OUTPUTS	
A	B	C_{in}	SUM	CARRY $_{OUT}$
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

Circuit Diagram for Sum (1, 2, 4, 7)



Circuit Diagram for Carry (3, 5, 6, 7)



Conclusion: In this way multiplexer & its applications are studied, implemented &tested.

FAQ:

Q1) what is a Multiplexer and Demultiplexer?

Q2) what is the difference between Decoder & Demultiplexer?

Assignment No: 04

Title: Design and implement full subtractor using decoder IC 74138

Assignment No: 04

Aim: Part A – MUX IC 74153

i) Verification of IC.
ii) Implementation of 8:1 Mux by cascading 2, 4:1 Mux in IC 74153
iii) Boolean function implementation
iv) Full adder implementation using hardware reduction table.

Part B – Decoder IC 74138

i) Verification of IC.
ii) Boolean function implementation.
iii) Full subtractor implementation using hardware reduction table.

Objectives: To Understand and Design Full adder and Full Subtractor using dual 4:1 IC 74153.

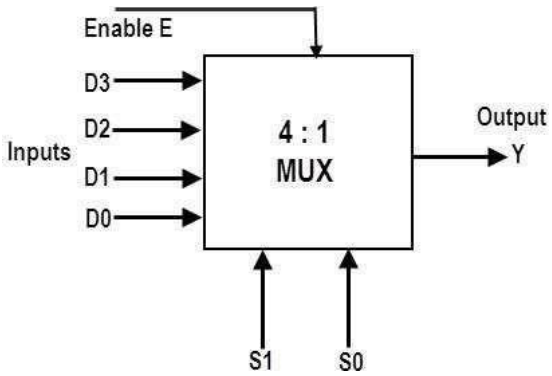
Software Used (if Applicable) / Programming Languages Used / Hardware Used: MUX(IC 74153), Decoder(IC 74138), IC 7432(OR)

Theory:

BCDAdder:

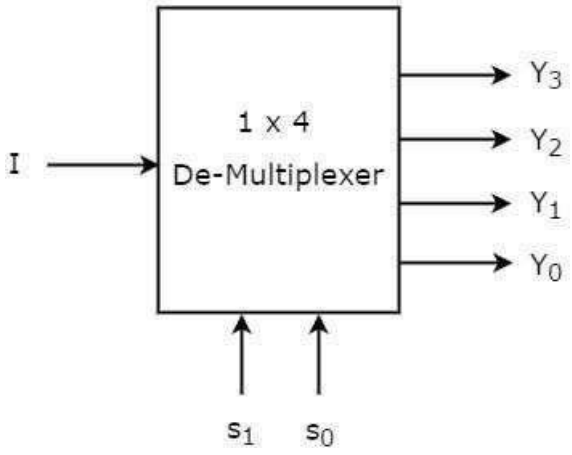
Digital Multiplexer:

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.



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 & 2m 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

Function implementation:

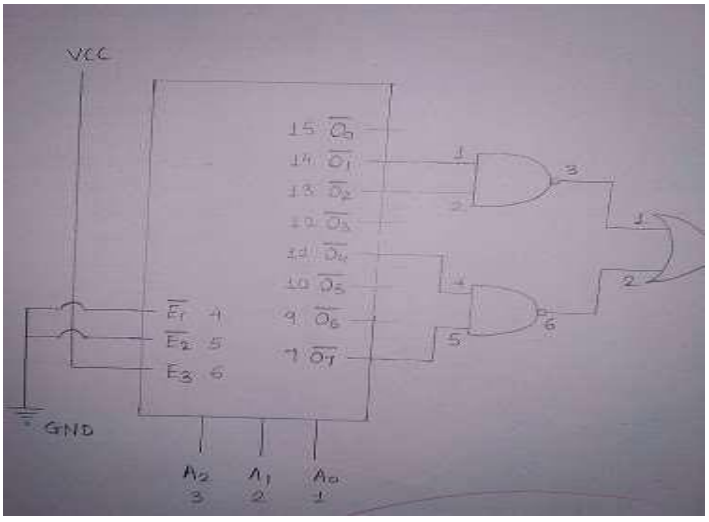
Implementation of Full Subtractor using IC 74138:

A full Subtractor is a combinational circuit that forms the arithmetic difference of two input bits with borrow. It consists of three inputs and two outputs. Two of these variables denoted byA and B represent the two significant bits to be subtracted. The third input represents the borrow from previous lower significant position.

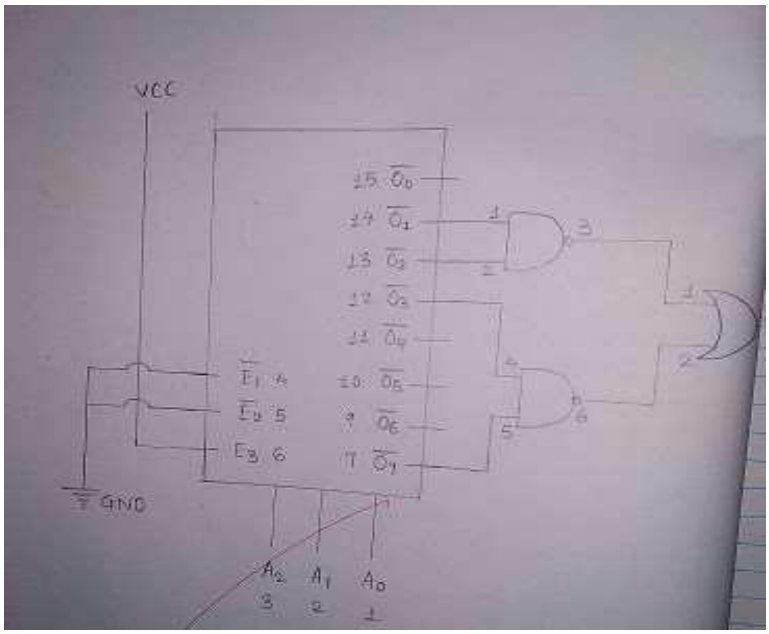
Truth Table for design of Full Subtractor:

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
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

Circuit Diagram for Difference (1, 2, 4, 7)



Circuit Diagram for Borrow (1,2,3,7)



Conclusion: In this way multiplexer & its applications are studied, implemented &tested.

FAQ:

Q1) what is a Multiplexer and Demultiplexer?

Q2) what is the difference between Decoder & Demultiplexer?

Assignment No: 05

Title: Design and implement 3 bit Up and 3 bit Down Asynchronous Counters using master slave JK flip-flop IC 7476

Assignment No: 05

Aim: Design (State diagram, state table & K map) and implementation of 3-bit Up and Down Asynchronous using master slave JK flip-flop IC 7476.

Objectives: To Understand, Design and Implement 3-bit Up and Down Asynchronous Counter.

Software Used (if Applicable) / Programming Languages Used / Hardware Used: IC 7476 (MS-JK Flip-flop), IC 7408, IC 7432 and IC 7404.

Theory:

Asynchronous counter:

The Asynchronous counter is also called as ripple counter. An Asynchronous counter uses T flip flop to perform a counting function.

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 the previous flip-flop. Therefore in an asynchronous counter the flip-flop are not clocked simultaneously.

1) 3-bit Asynchronous Up Counter:

Fig. shows 3-bit Asynchronous Up Counter. Here Flip-flop C act as a MSB Flip-flop and Flip-flop A can act as a LSB Flip-flop. Clock pulse is connected to the Clock of Flip-flop A. Output of Flip-flop A (Q_A) is connected to clock of next flip-flop (i.e. Flip-flop B) and so on. As soon as clock pulse changes, output is going to change (at the negative edge of clock pulse) as an Up count sequence. For 3 bit Up counter Truth table is as shown below.

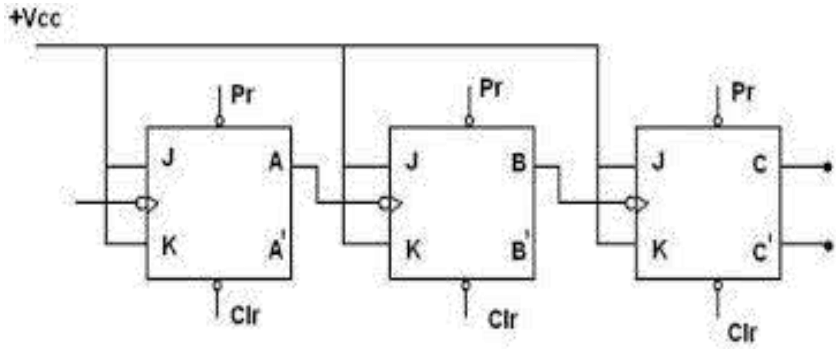


Fig: 3 bit Asynchronous up Counter

2) 3-bit Asynchronous Down Counter:

Fig. shows 3-bit Asynchronous down Counter. Here Flip-flop C act as a MSB Flip-flop and Flip-flop A can act as a LSB Flip-flop. Clock pulse is connected to the Clock

Of Flip-flop A. Output of Flip-flop A (QA') is connected to clock of next flip-flop (i.e. Flip- flop B) 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 Truth table is as shown below.

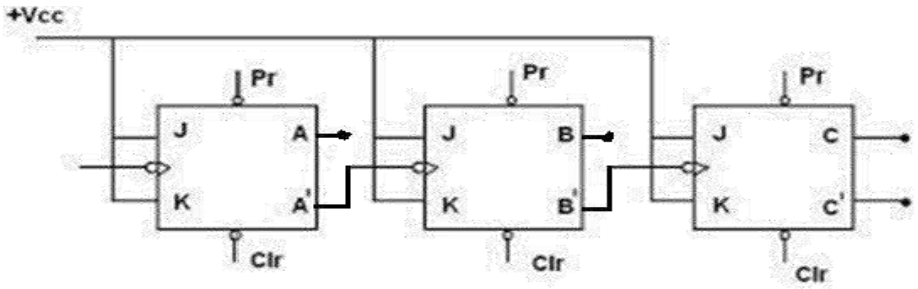


Fig: 3 bit Asynchronous down Counter

Conclusion:

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

FAQ:

Q1) what do you mean by Counter?

Q2) what are the types of Counters? Explain each.

Q3) what are the applications of synchronous counters?

Assignment No: 06

Title: Design and implement 3 bit Up and 3 bit Down Synchronous Counters using master slave JK flip-flop IC 7476

Assignment No: 06

Aim: Design (State diagram, state table & K map) and implementation of 3-bit Up and Down Synchronous using master slave JK flip-flop IC 7476.

Objectives: To Understand, Design and Implement 3-bit Up and Down Synchronous Counter.

Software Used (if Applicable) / Programming Languages Used / Hardware Used: IC 7476 (MS-JK Flip-flop), IC 7408, IC 7432 and IC 7404.

Theory:

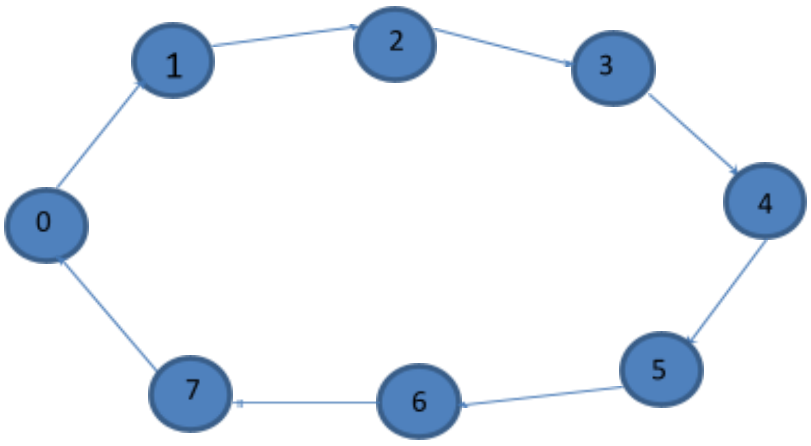
6.1 Synchronous Counter:

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.

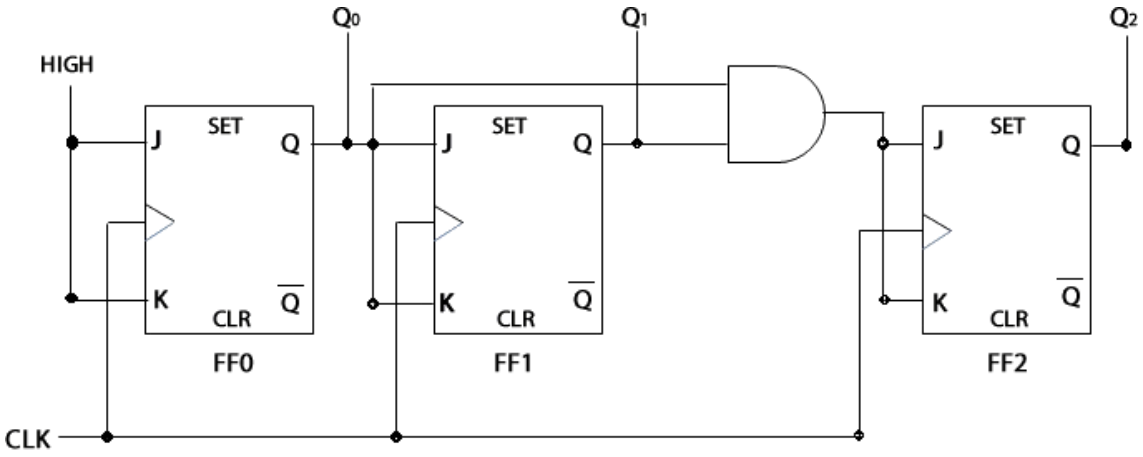
1) 3-bit Synchronous up counter:

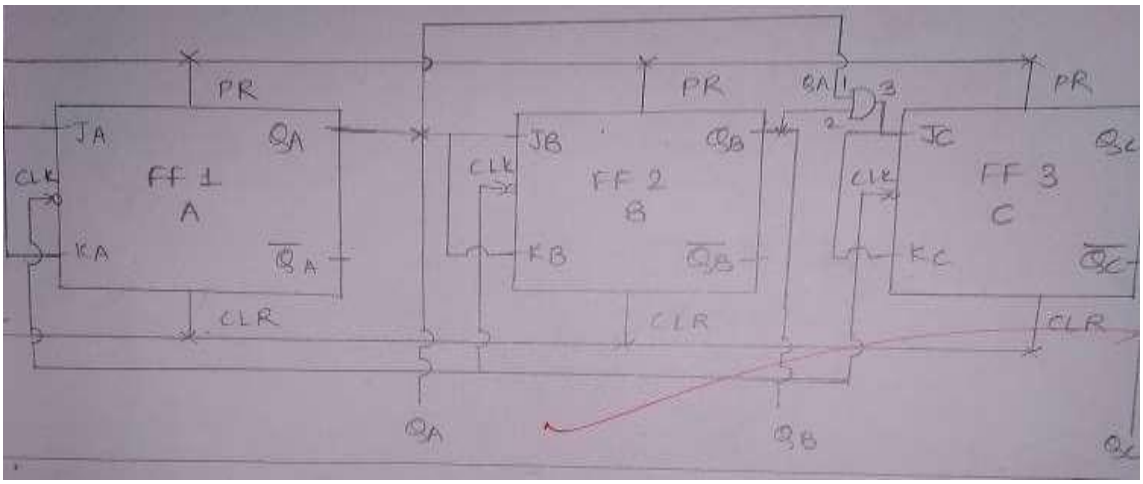
The up counter counts from 0 to7 i.e. (000 to 111).for this we are using MS JK flip flop. In IC 74LS76, 2 MS J-K flip flops are present. The clock pulse is given at pin 1 & 6 of the 1st IC & pin 1 of 2nd IC. Next state decoder logic is designed with the help of state table.

State table for 3 bit synchronous Up counter:



Logic Diagram for 3 bit synchronous up Counter:

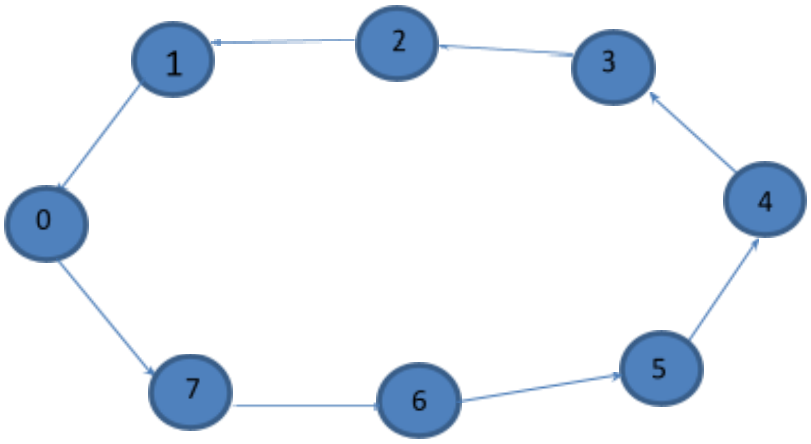




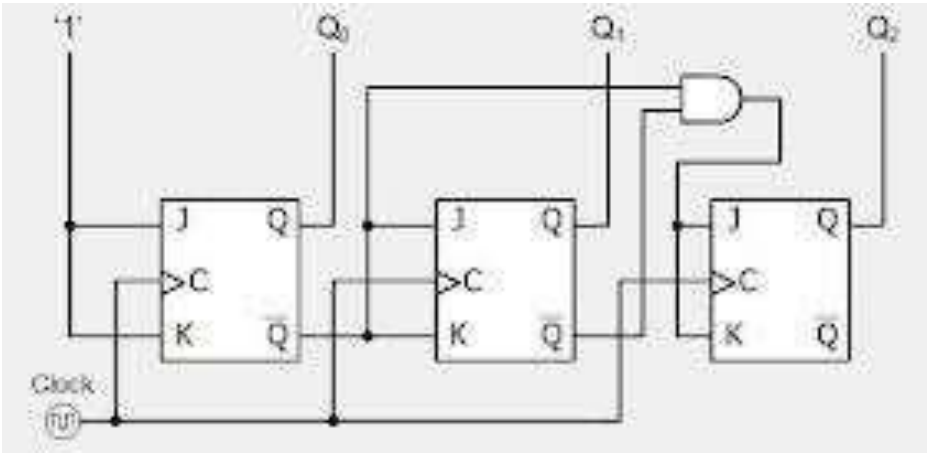
2) 3-bit Synchronous down counter:

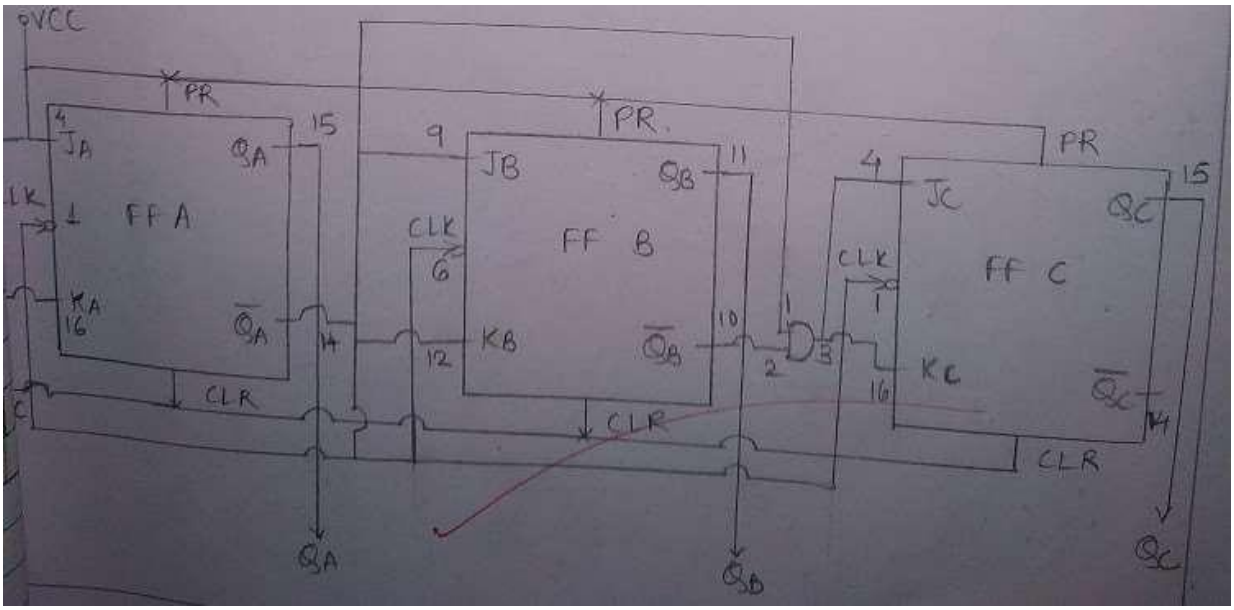
This is used to count from 7-0 i.e.(111-000).for this also 2 IC"s of 74LS76 are required & hence we use 3 MS JK flip flops. Here also clock is given to 1st& 6th pin of 1st IC & 1st pin of 2nd IC enabling to apply clock to all flip flop at a time. Next state decoder logic is designed with the help of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

State table for 3 bit synchronous down counter:



Logic Diagram for 3 bit synchronous down counter:





Conclusion:

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

Assignment No: 07

Title: Design and implement Modulo 'N' counter using IC7490. (N= 100 max)

Assignment No: 07

Aim: Design and implement Modulo 'N' counter using IC7490. (N= 100 max)

Objectives: To Understand, Design and Implement Module 'n' counter.

Software Used (if Applicable) / Programming Languages Used / Hardware Used: IC 7490, basic gates.

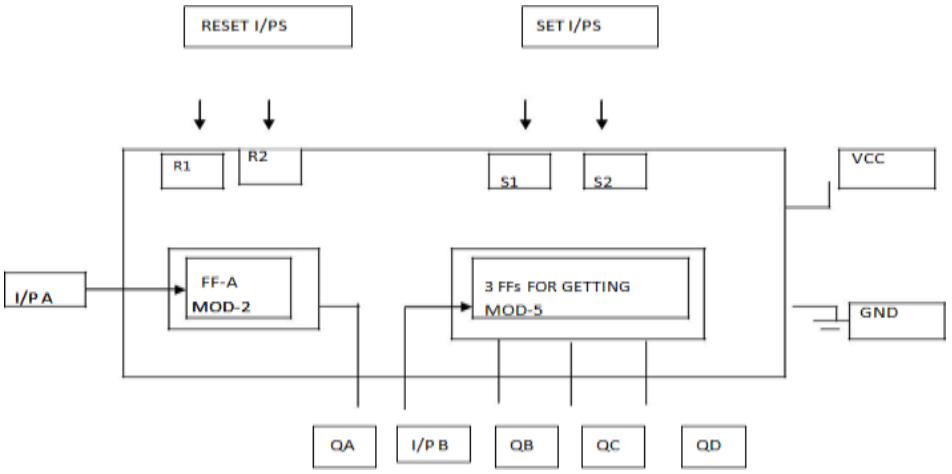
Theory:

BCDAdder:

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.

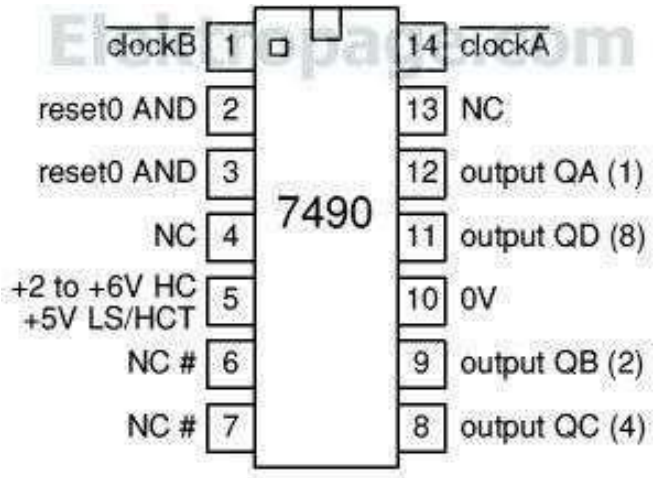
Basic internal Structure of IC 7490:



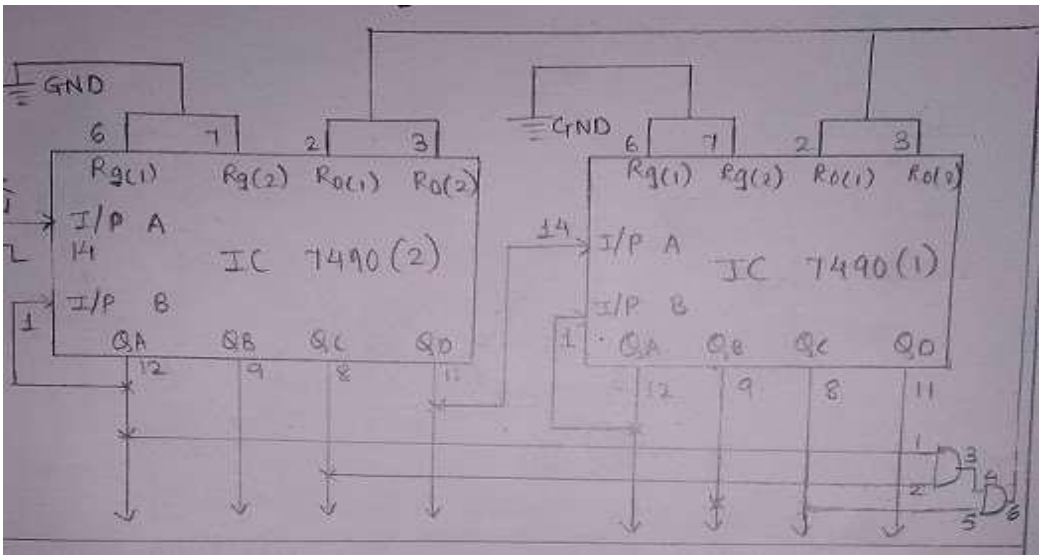
Design of Mod-7 Counter using IC 7490:

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.

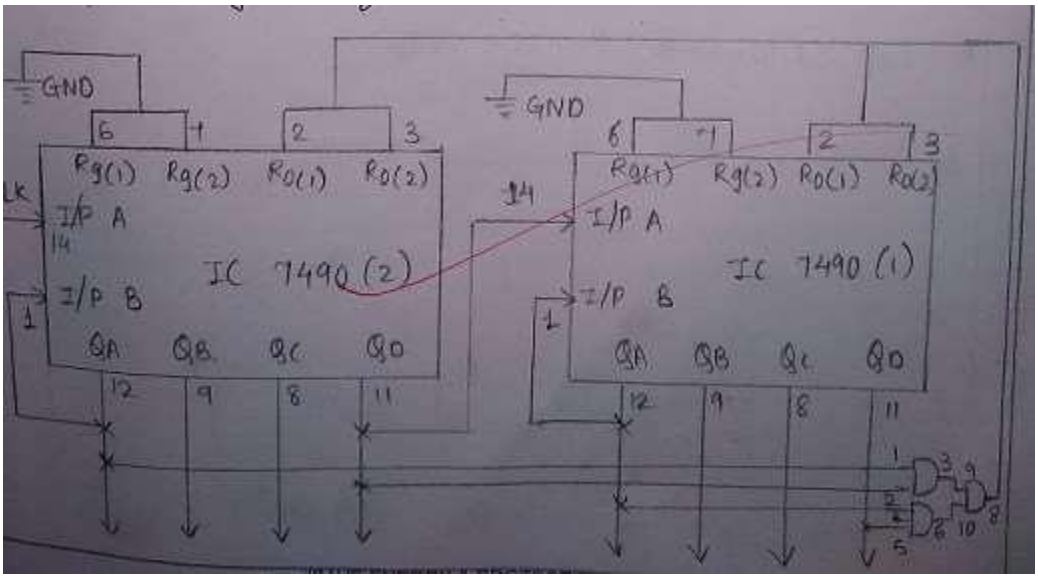
Pin Diagram of IC 7490:



MOD-45 Counter Diagram:



MOD-99 Counter Diagram:



Conclusion: Thus we studied the various MOD counters using IC 7490.

FAQ:

Q1) what do you mean by Counter?

Q2) what are the types of Counters? Explain each.

Q3) what are the applications of synchronous counters?

Assignment No: 08

Title: Design& simulate single bit ALU with four functions (AND, OR, XOR, ADD).

Assignment No: 08

Aim: Design& simulate single bit ALU with four functions (AND, OR, XOR, ADD).

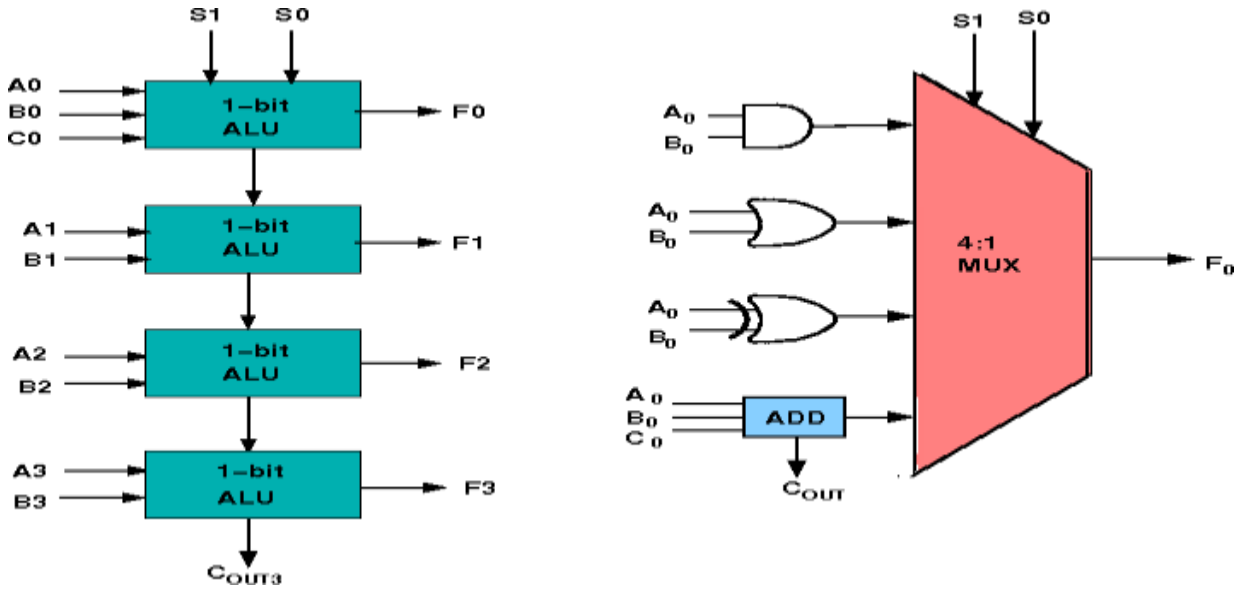
Objectives: To Understand, Design& simulate single bit ALU.

Software Used (if Applicable) / Programming Languages Used / Hardware Used:- Virtual Lab Simulator.

Theory

Design of ALU

ALU or Arithmetic Logical Unit is a digital circuit to do arithmetic operations like addition, subtraction, division, multiplication and logical operations 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 $S_1 = 0$, $S_0 = 0$, the output is A And B,

for Control signal $S_1 = 0$, $S_0 = 1$, the output is A Or B,

for Control signal $S_1 = 1$, $S_0 = 0$, the output is A Xor B,

for Control signal $S_1 = 1$, $S_0 = 1$, the output is A Add B.

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

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 $\oplus$ 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	$(A \oplus B)'$	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

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

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

Procedure

Design of ALU:

To see the step by step process to perform the experiment in simulator follows the steps below.

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.

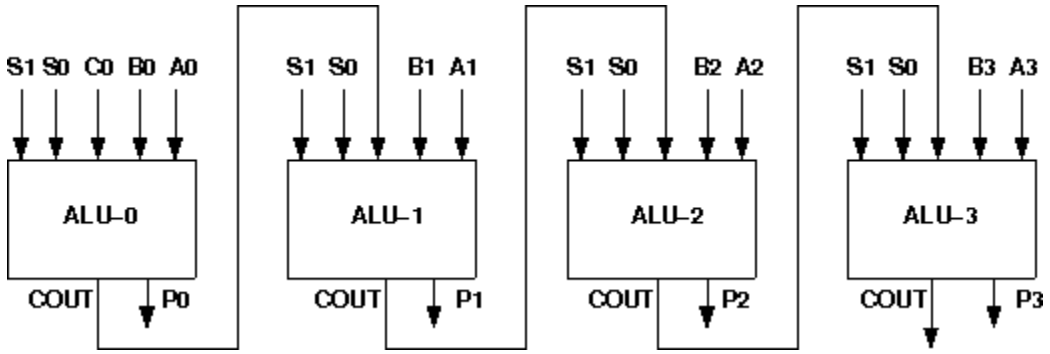
4. For 1-bit ALU input A0 is in pin-9,B0 is in pin-10, C0 is in pin-11 (this is input carry), for selection of operation, S0 is in pin-12, S1 is in pin-13, output F is in pin-8 and output carry is pin-7

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)

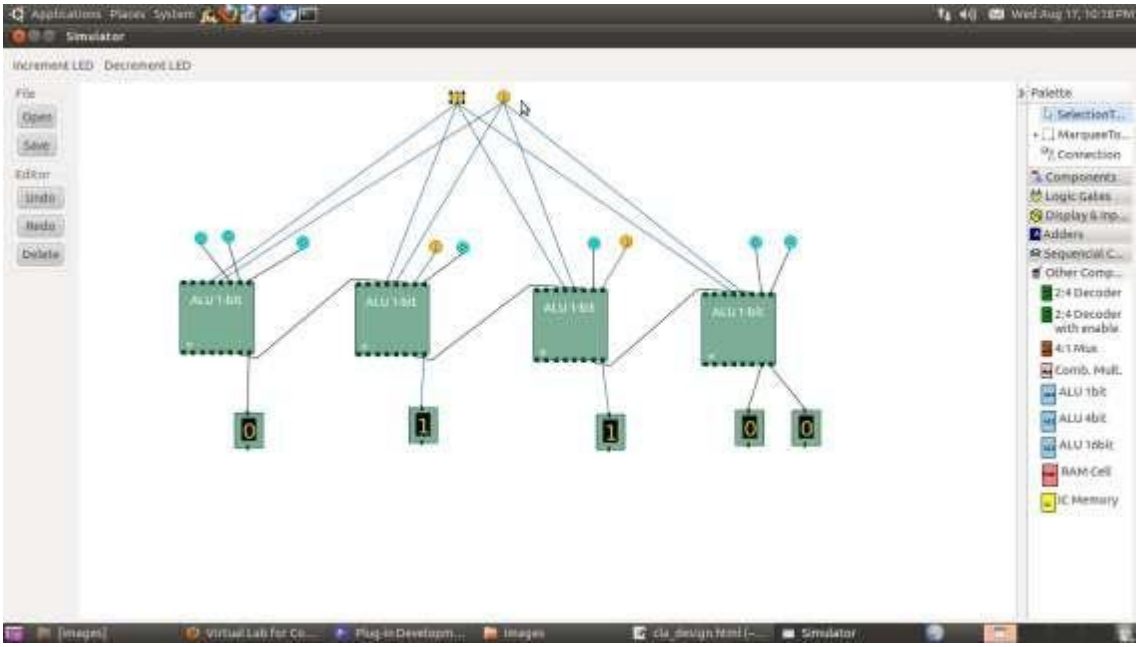
6. To connect any two components select the Connection menu of Palette, and then click on the Source terminal and click on the target terminal. According to the circuitdiagram connect all the components. Connect the Bit switches with the inputs and Bit displays component with the outputs. After the connection is over click the selection tool in the pallet.

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.

Conclusion: In this way 1-bit ALU is designed and implemented.

FAQ:

Q1) What does the selector inputs to an arithmetic logic unit (ALU) determine?

Q2) How to design 16-bit ALU using 4-bit ALU?

Assignment No: 09

Title: Design& simulation of single instruction CPU

Assignment No: 09

Aim: Design & simulation of single instruction CPU

Objectives: To understand working of Virtual Lab Simulator and Design& simulation of single instruction CPU

Software Used (if Applicable) / Programming Languages Used: Virtual Lab Simulator

Theory

CPU Design:

At the top level a computer consists of a CPU (central processing unit), memory, I/O components, with one or more modules of each type. These modules are interconnected in a specific manner to achieve the basic functionality of a computer i.e. executing programs. At the top level a computer system can be described as follows:

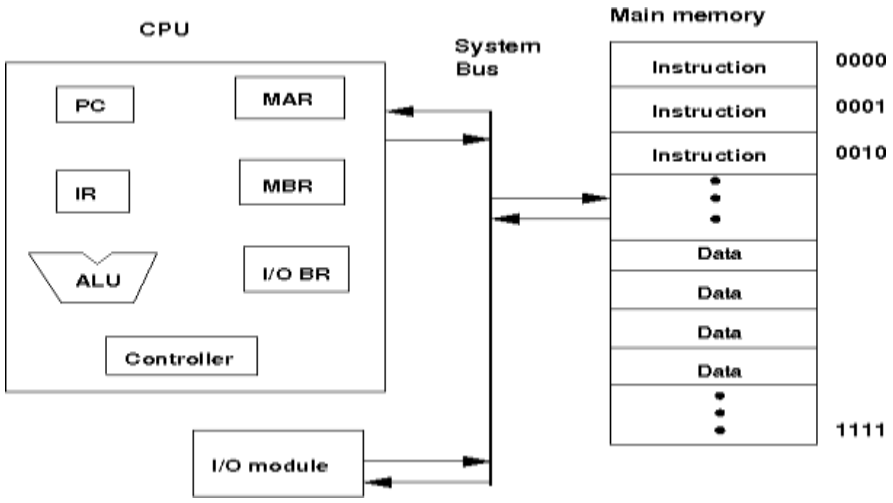
- describing the external behavior of each component i.e. the data and the control signals that ir exchanges with other components
- describing the interconnection structure and the controls required

We are considering the Von Neumann architecture. Some of the basic features of this architecture are as follows:

- data and instructions are stored in a single read-write memory
- the contents of the memory are addressable by location
- execution occurs in a sequential manner (unless explicitly specified) from one instruction to the next

Top level components and interactions among them:

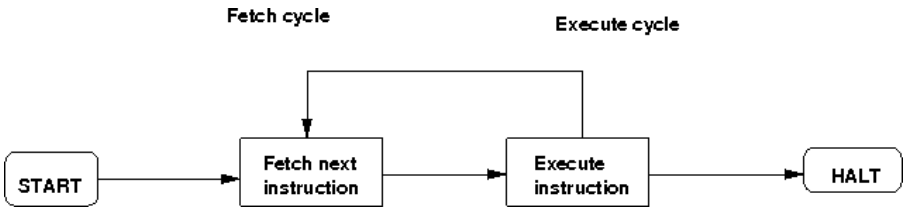
- CPU exchanges data with memory. For this CPU uses two internal resisters. Following is a block diagram of a basic computer system:



- Memory address register (MAR) which specifies the address for the next read or write
- Memory buffer register (MBR) which contains the data to be written into memory or receives the data read from memory
- I/O buffer (I/O BR) register is used for the exchange of data between an I/O module and the CPU

- A memory module consists of a set of a locations defined by sequentially numbered addresses. Each location contains a binary number that cam be interpreted as either an instruction or data.
- An I/O module transfers data from external devices to CPU and memory and vice versa
- The basic function of a computer is to execute a program which consists of a set of instructions stored in the memory. Processing required for a single instruction is called an instruction cycle which consists instruction fetch and instruction execute. A register called program counter (PC) holds the address of the next instruction. Unless told otherwise the processor always increments PC after each instruction fetch so that the next instruction is fetched in sequence. The fetched instruction is fetched into a register called instruction register (IR).

  The basic instruction cycle is shown in the following figure:



- After fetching an instruction, processor executes the instruction by doing some processing on the data which may involve arithmetic and logic unit (ALU), specified by the instruction, then processor writes back the result (if any) to the memory.

This experiment provides a single instruction CPU with built-in controller. A working memory has been provided to check the working principle of the CPU. The single instruction which this CPU supports is SBN (subtract and branch if negative). The format of this instruction is:

- SBN a,b,c Mem[a] = Mem[a] - Mem[b] if(Mem[a] < 0) goto c
- a, b, c are 4 bit addresses
- Mem[a] denotes the contents of memory address a

Procedure

CPU Design:

Procedure to perform the experiment: CPU Design

1. Start the simulator as directed. This simulator supports 5-valued logic.
2. To perform the experiment on the given modules, we need the CPU, the working memory with a program and data loaded, a clock input, Bit switch(to give input, which will toggle its value with a double click), Bit displays(for seeing output), wires.
3. Load memory: click on the *load memory* button in the left pane. you can either load the memory by filling the form or you can directly load the program form a text file. The memory provides 4-bit address space and 12 bit data word, thus providing 16 memory addresses starting from 0000 to 1111. For loading from file, the file should contain only binary values; it must contain 16 lines, each line containing the content to be stored in

the corresponding memory address. For example, content of first line will be loaded to the 0000 address of the memory, similarly, the second line will correspond to the 0001 address and so on, and finally the content of last line will be fed to the 1111 address. The program should use self-loop for halting, for example, the instruction stored at address 1010 will cause self-loop execution, if it content of a has -1 in binary format (in2's complement), the content of b has 0 and c is 1010, then once the execution reachesto this 1010 address, it will finally point to itself.

4. Instantiating the memory: after loading the memory, click on the memory component from the computer design drawer in the palette of the simulator then click on the position of the design editor where you want to put the component(no drag and drop, simple click will serve the purpose).

5. The pin configuration of the component is shown whenever the mouse is hovered on any canned component of the palette or pressing the *show pin configuration* button on the toolbar will show it constantly in the left pane. Pin numbering starts from 1 and from the bottom left corner (indicating with the circle) and increases anticlockwise.

6. Pin configuration of the memory module:

    - Input pins (upper terminals): memory enable : 30, R/W' : 29, address : 25-28, data : 13-24 (13 is LSB)

    - Output pins(lower terminals): data output : 1-12 (1 is MSB)

7. Instantiate the CPU from the computer design drawer in the palette of the simulator then click on the position of the design editor where you want to put the component.

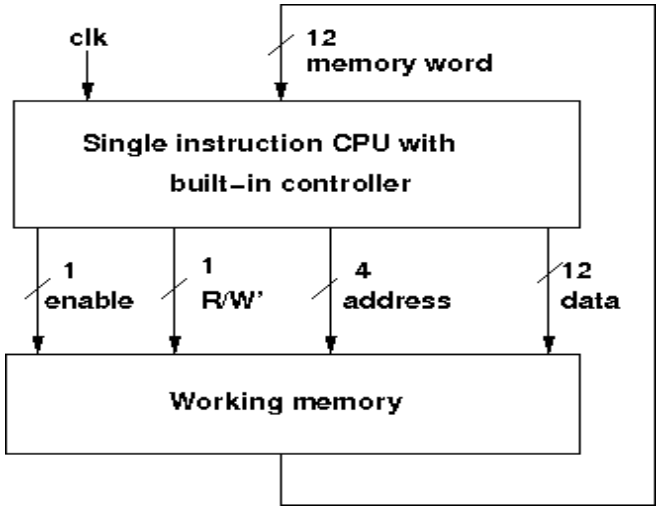
8. Pin configuration of the CPU:

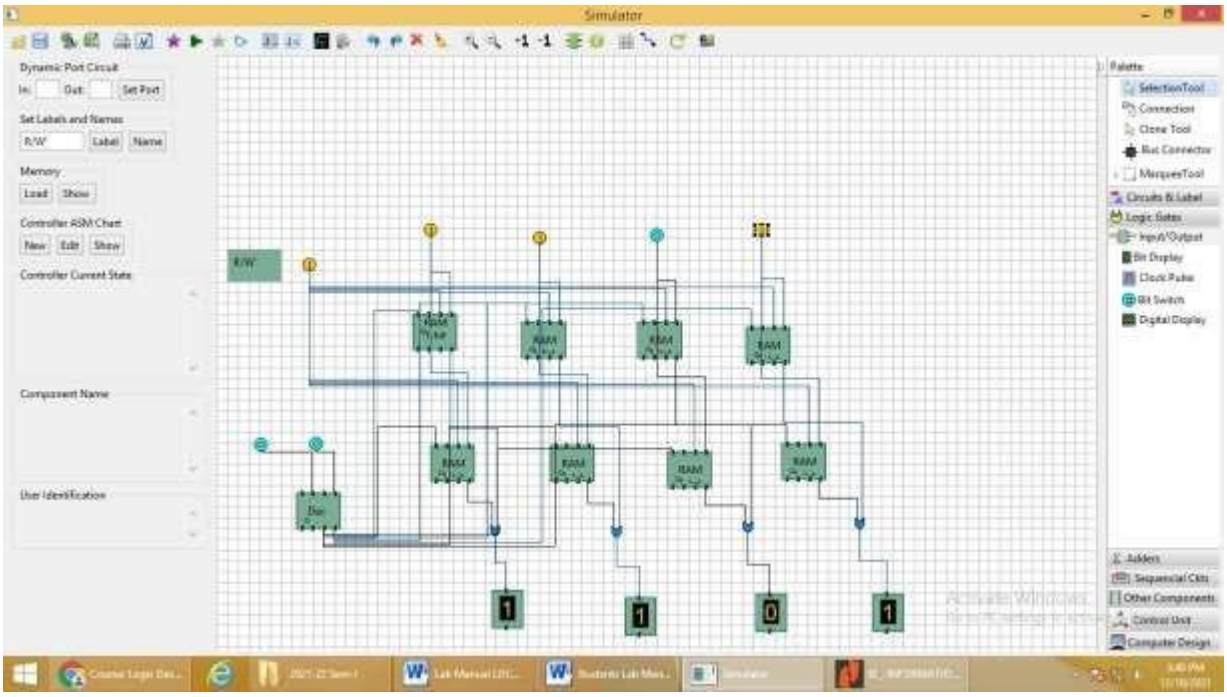
    - Input pins (upper terminals): data input : 20-31 (20 is MSB) ,clock input : 19

    - Output pins(lower terminals): memory enable : 1, R/W' : 2, address : 3-6(3 is MSB), data output : 7-18 (7 is MSB)

9. 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 following diagram connect all the components. Connect the memory outputs to the input terminals of the CPU, specified data path outputs to the inputs of the controller, the clock input, Bit switches with the inputs and Bit displays component with the outputs (from Display and Input drawer of the pallet, if it is not seen scroll down in the drawer).After the connection is over click the selection tool in the pallet.

10. Start clock and observe the behavior of the CPU. See the content of memory by clicking *show memory* button in the left pane. Observe how the program is executing sequentially and modify the data content as per the program.





Conclusion: Thus we studied the Virtual Lab Simulator & designed CPU

FAQ:

Q1) What is a stored program computer?

Q2) What general catagories of functions are specified by computer instructions?

Q3) What are the possible states that define the instruction execution?

Q4)What is wired AND operation?

Assignment No: 10

Title: Design (truth table, K-map) and implementation of 4-bit Gray to Binary Code converters

Assignment No: 10

Aim: Design and implementation of 4-bit Code convertors.

i) Gray to Binary Code

Objectives: To Understand and Design 4-bit Code Convertors.

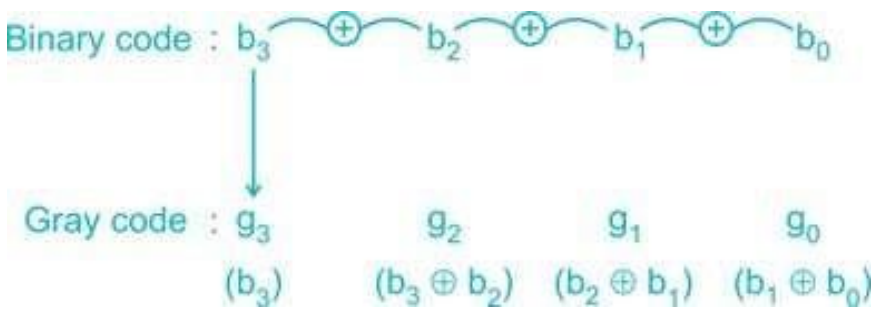
Software Used (if Applicable) / Programming Languages Used / Hardware Used: IC's:7432 (OR-gate), 7408 (AND-gate), 7486 (Ex-or gate)

Theory:

Binary Numbers is default way to store numbers, but in many applications binary numbers are difficult to use and a variation of binary numbers is needed. This is where Gray codes are very useful. Gray code has property that two successive numbers differ in only one bit because of this property gray code does the cycling through various states with minimal effort and used in K-maps, error correction, communication etc.

Binary to Gray Conversion:

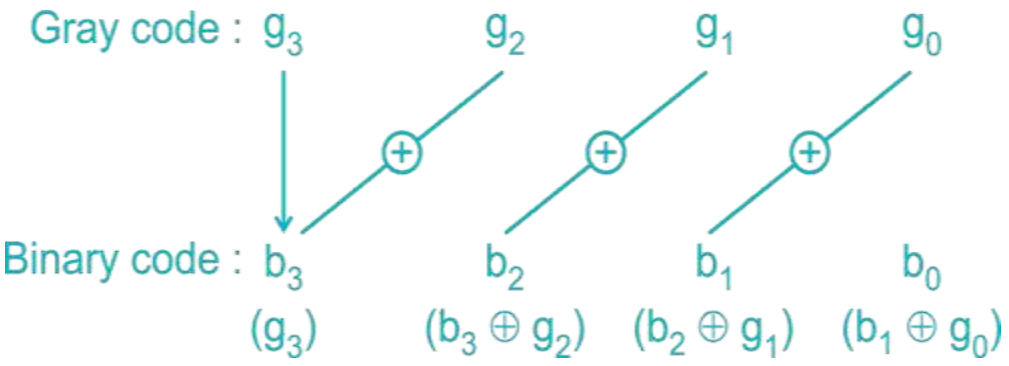
i) The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
ii) Other bits of the output gray code can be obtained by XORing binary code bit at that index and previous index.



Gray to binary Conversion:

i) The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given binary number.

ii) Other bits of the output binary code can be obtained by checking gray code bit at that index. If current gray code bit is 0, then copy previous binary code bit, else copy invert of previous binary code bit.

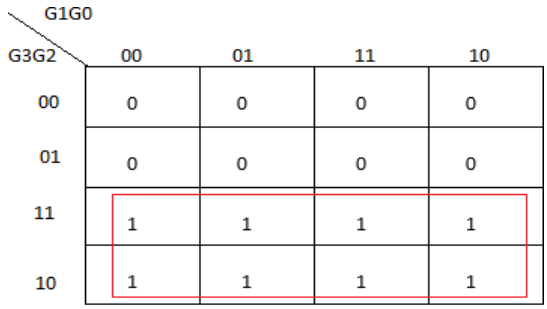


Gray to Binary Code Conversion: Truth Table:

GRAY CODE INPUT				BINARY OUTPUT			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

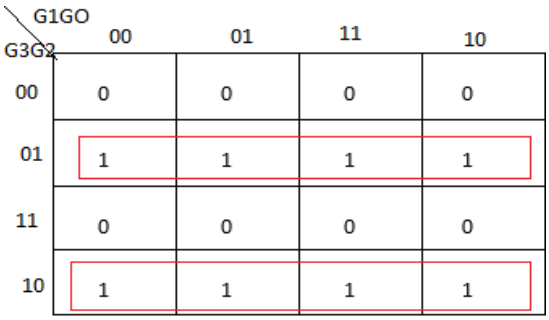
K-Map for Reduced Boolean Expressions of Each Output for Gray to Binary Code Conversion:

K-Map for B3:-



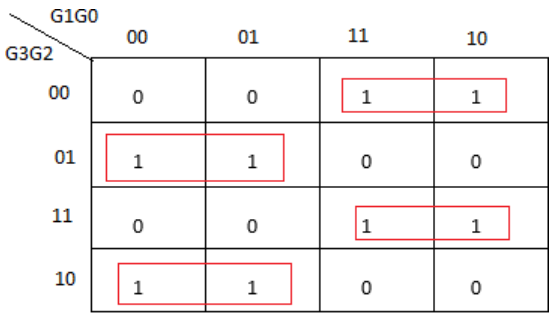
B3 = G3

K-Map for B2:-



B2 = G3⊕G2

K-Map for B1:-



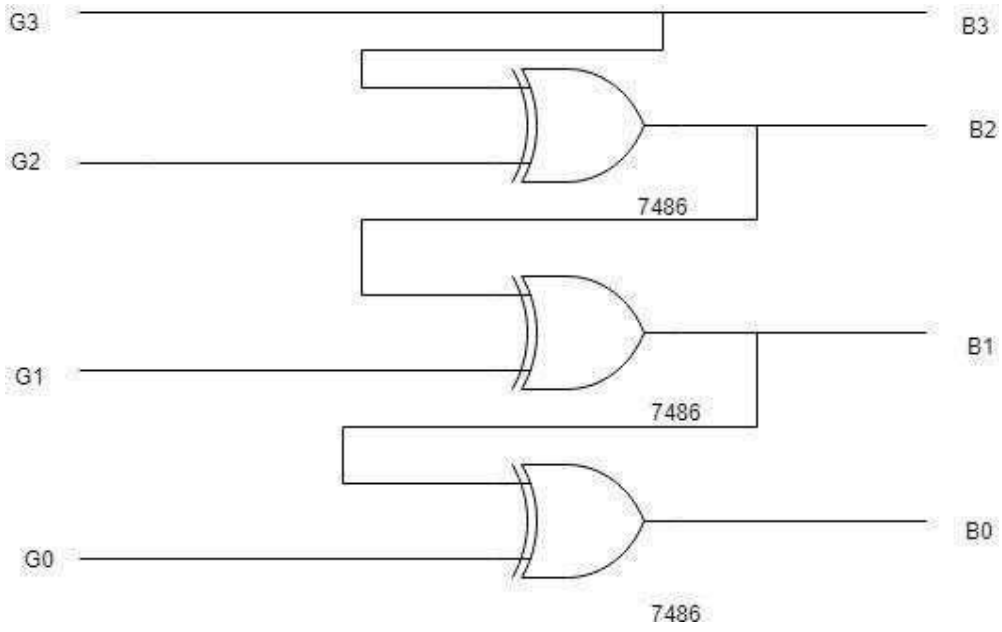
B1 = G3⊕ G2⊕ G1

K-Map for B0:-

G3G2 \ G1G0	00	01	11	10
00	0	(1)	0	(1)
01	(1)	0	(1)	0
11	0	(1)	0	(1)
10	(1)	0	(1)	0

BO = G3 ⊕ G2 ⊕ G1 ⊕ G0

Circuit Diagram: Gray to Binary CODE CONVERTER



Conclusion: Thus, we studied Gray code and the Binary code and also studied their conversions including applications. The truth tables have been verified using IC 7486, 7432,

7408, and 7404.

FAQ:

1) what is the need of code converters?

2) what are weighted codes and non-weighted codes?

3) why is Gray code called as Unit distance code?

Assignment No:11

Title: Design and implementation of following circuits using 4-bit Parallel binary adder IC 7483:

i) 4-bit parallel subtractor.

ii) 4-bit parallel adder/subtractor.

Assignment No: 11

Aim: Design and implementation of following circuits using Parallel binary adder IC 7483:

i) 4-bit parallel subtractor.

ii) 4-bit parallel adder/subtractor.

Objectives: To Study & design4-bit Binary adder/subtractor circuit using IC 7483 which performs binary addition and subtraction.

Hardware used: IC 7483, IC 7486, Connecting Wires, Power Supply, Breadboard etc.

Software used: NOT APPLICABLE

Theory:

4-bit Parallel adder IC 7483:

IC type 7483 is a 4-bit binary adder with fast carry. The pin assignment is shown in Fig. l. The two 4-bit input binary numbers are Al through A4 and B1 through B4. The 4-bit sum is obtained from S1 through S4. CO is the input carry and C4 the out carry. This IC can be used as an adder-subtractor.

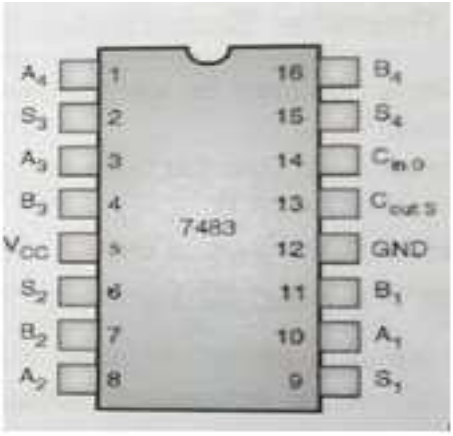


Fig1. Pin diagram of IC 7483

4-bit Parallel Subtractor:

The subtraction of two binary numbers can be done by taking the 2's complement of the subtrahend and adding it to the minuend. The 2's complement can be obtained by taking the 1's complement and adding 1. To perform A-B, we complement the four bits of B, add them to the four bits of A, and add 1 through the input carry. A 4-bit parallel subtractor using a 4-bit parallel adder is shown in fig. 2.

a. The number to be subtracted (B) is first passed through inverters to obtain its 1's compliment. One inverter per bit of word B is used so that all the bits of B get inverted.
b. Then 1 is added to 1's compliment of B, by making Cin=1. Thus we obtain the 2's compliment of B
c. The 4-bit adder then adds A and 2's compliment of B to produce the subtraction at its sum outputs S3 S2 S1 S0.
d. The word S3 S2 S1 S0 represents the result of binary subtraction (A-B) and carry output Cout represents the polarity of result.
e. If A>B the Cout = 0 and then the result is in true binary form but if A<B then Cout =1 and the result is negative and in the 2's compliment form.

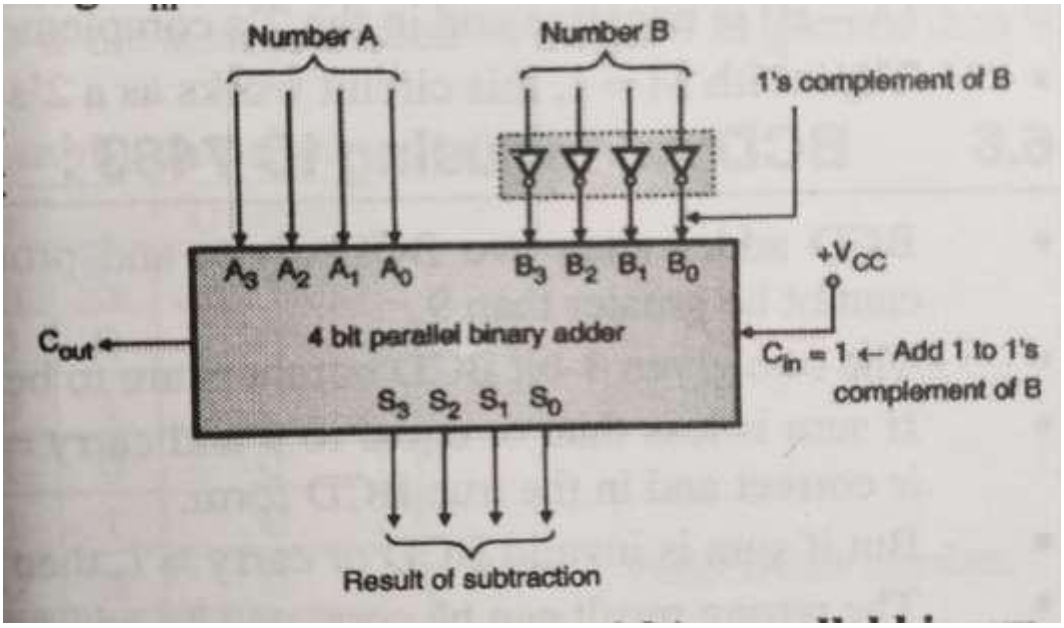


Fig.2:4-bit parallel binary adder using 2's compliment.

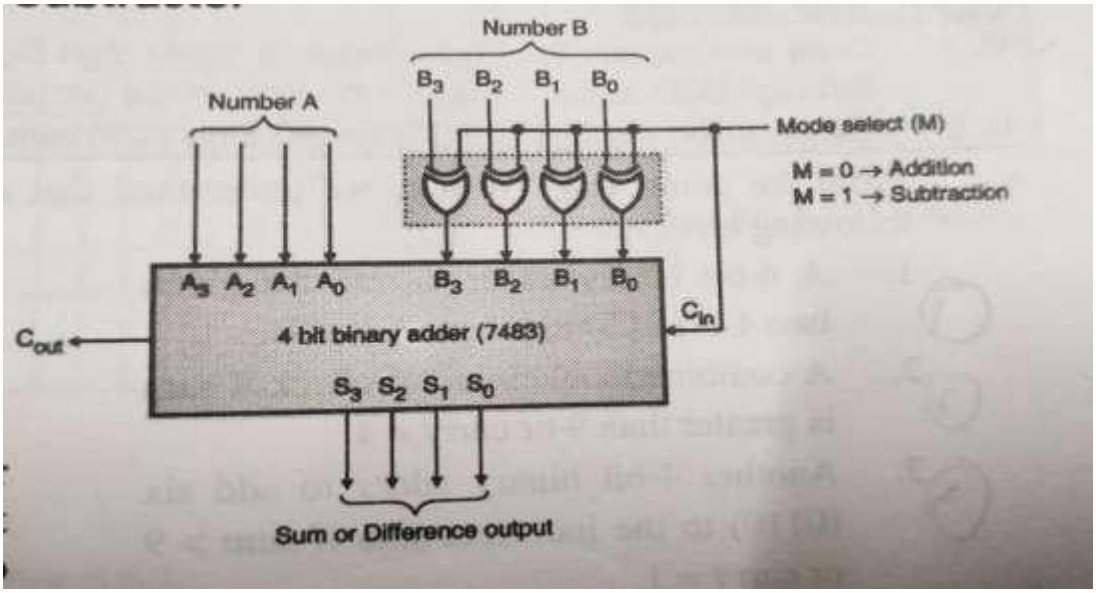
4-bit Binary Adder/Subtractor:

The addition or subtraction of any two 4-bit binary numbers can be obtained using the same circuit shown in the fig. below. Therefore it is called as an adder/subtractor.The operation performed by this circuit (Addition or Subtraction) depends upon the state of mode of select input i.e. M in the fig.3Number B is applied to the adder through four EX-OR gates. One input of each EX-OR gate is connected to the mode-select input (M).

1. **Operation as Adder(M=0):**
   a. For operation as an adder, the select mode (M) input is connected to ground. Therefore M=0.
   b. Since M=0, the output of the EX-OR gates will be same number B which was applied at their inputs. This is because $0 \oplus 0 = 0$ and $0 \oplus 1 = 1$. Hence B_3, B_2, B_1 and B_0 will passed unchanged through the EX-OR gates. The carry input C_{in} is connected to M Therefore $C_{in} = 0$
   c. Therefore the adder adds $A+B+C_{in} = A+B$ since $C_{in} = 0$. Thus with M=0 addition of A and B will take place.

1. **Operation as Subtractor(M=1):**
   a. For operation as subtractor mode select input is connected to V_{cc}. Therefore M=1.
   b. Since M = 1, one input of each EX-OR gate is now 1. Hence each EX-OR gate acts as inverter. This is because $1 \oplus 0 = 1$ and $1 \oplus 1 = 0$.
   c. Thus each bit of word B is inverted by the EX-OR inverters. Thus we get 1's compliment of number B at the output of EX-OR gates. The carry input terminal C_{in} is connected to M. Therefore $C_{in} = 0$
   d. The inverted number B adds with $C_{in} = 1$ to give the 2's compliment of B. Hencethe adder will add A with the 2's compliment of B and the result is actually subtraction A-B. If $C_{out} = 0$ then the subtraction (A-B) is positive and in true form. But if $C_{out} = 1$ then the subtraction (A-B) is negative and in 2's compliment form.
   e. Thus with M = 1, this circuit works as a 2's compliment subtractor.

Hence, Four XOR gates complement the bits of B when the mode select M=l and leave the bits of B unchanged when M = 0. Thus, when the mode select M is equal to 1, the input carry CO is equal to 1 and the sum output is A plus the 2's complement of B. When M is equal to 0, the input carry is equal to 0 and the sum generates A+B.



Conclusion: We have design and implemented binary adder & subtractor using IC7483.

FAQ:

1. Design Magnitude comparative using IC 7483.