Ldcol Lab Manual

Information technology (Savitribai Phule Pune University)

GROUP A
(COMBINATIONAL LOGIC DESIGN)

Assignment 1: Code Converter

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

i) BCD to Excess – 3 Code

ii) Excess-3 to BCD Code

IC's Used:

IC 7404(Hex INV), 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. 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 is 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.

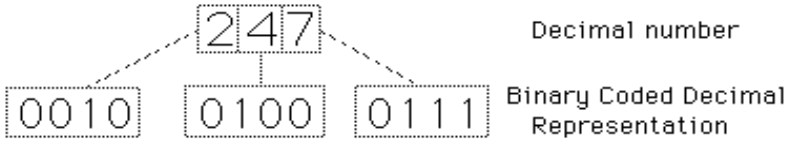


Fig. 3 BCD Coded Decimal Representation

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 Conversions:

Convert BCD 2 i. e. 0010 to Excess – 3 codes
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 Codes 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:

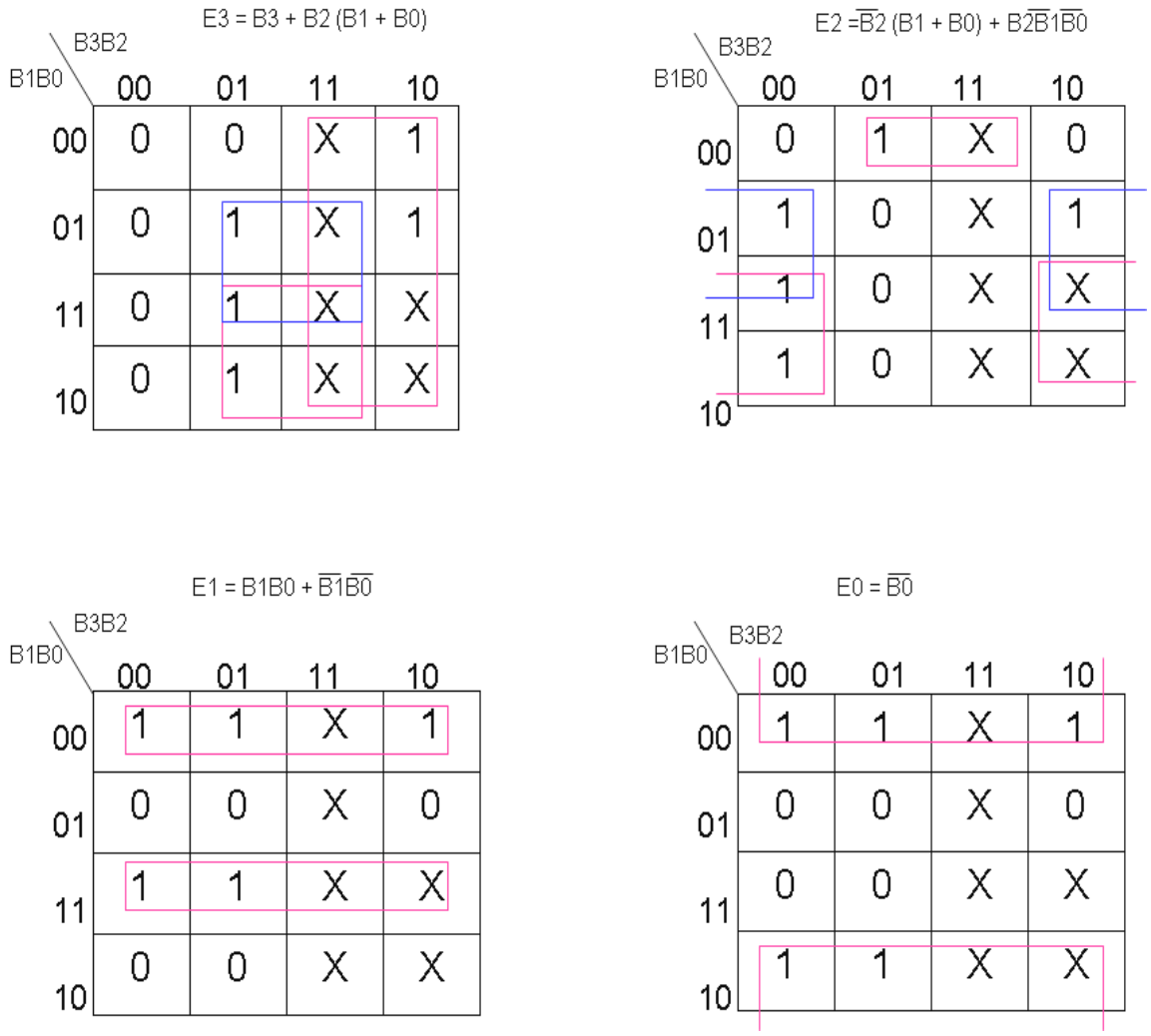


Fig. 8 K-Map for Reduced Boolean Expressions of Each Output (Excess-3 Code)

3) Circuit Diagram:

BCD TO EXCESS-3 CONVERTER

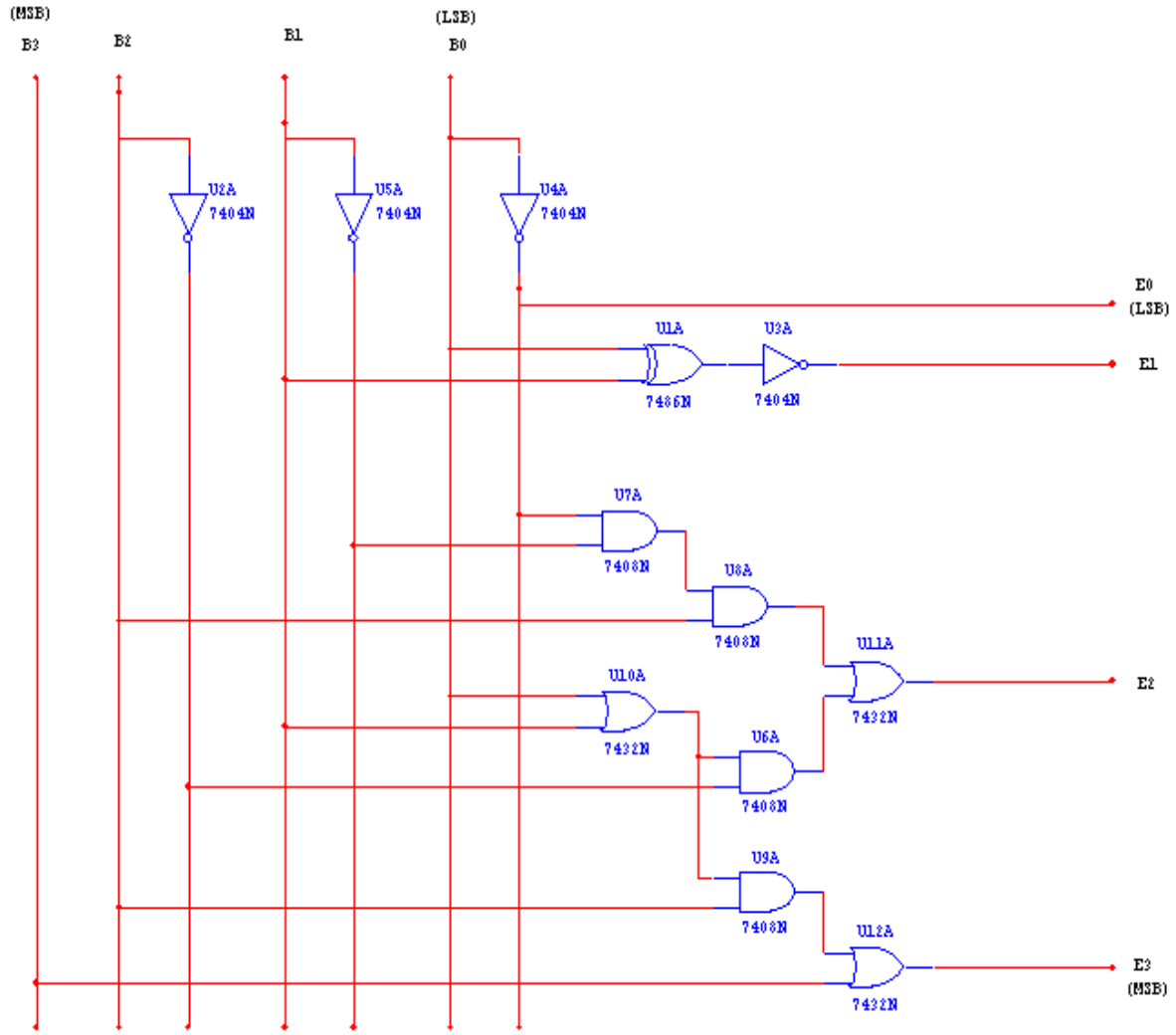


Fig.9 Logic Diagram for BCD to Excess-3 Code Conversion

4) Hardware Requirements Table:

GATE	Quantity	IC	Quantity
XOR	1	7486	1
NOT	4	7404	1
AND	4	7408	1
OR	3	7432	1

Table 5 Hardware Requirement Table

B. Excess-3 to BCD Conversion: Truth Table:

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

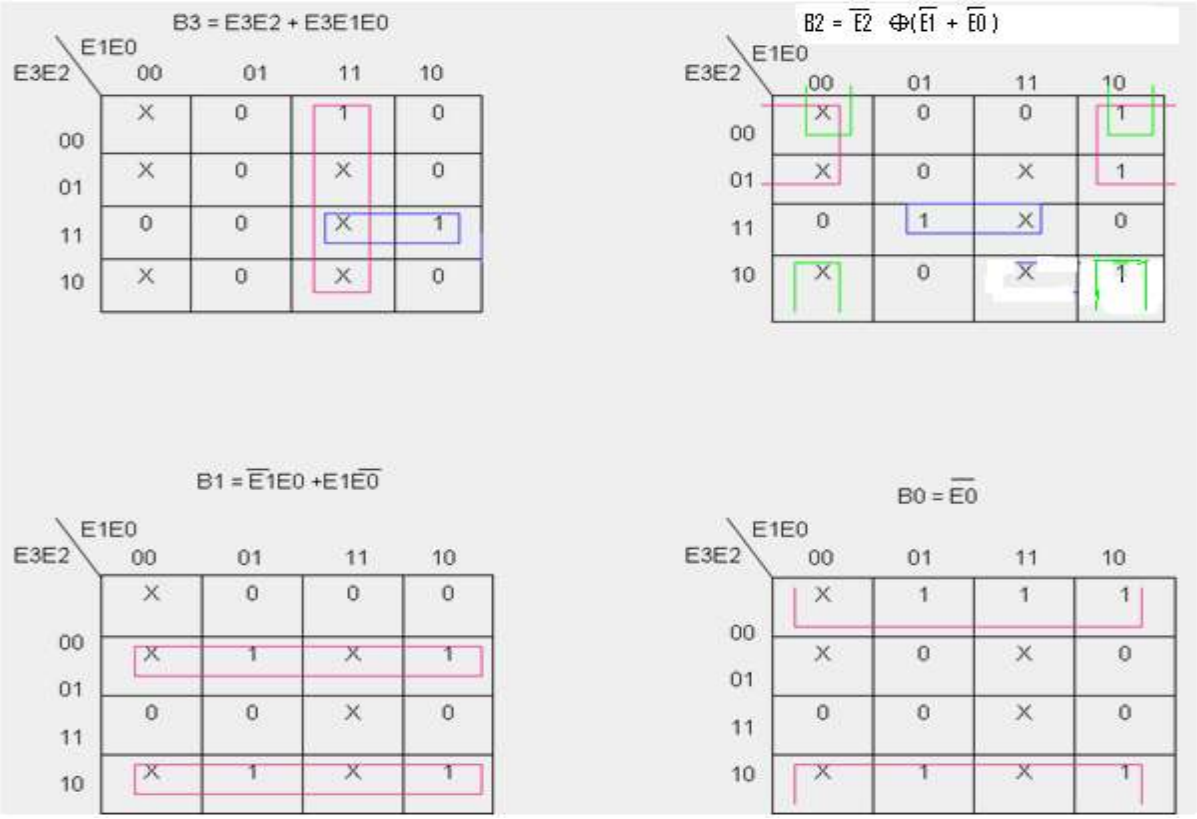


Fig 10 K-Map For Reduced Boolean Expressions Of Each Output (BCD Code)

3) Circuit Diagram: EXCESS-3 TO BCD CONVERTER

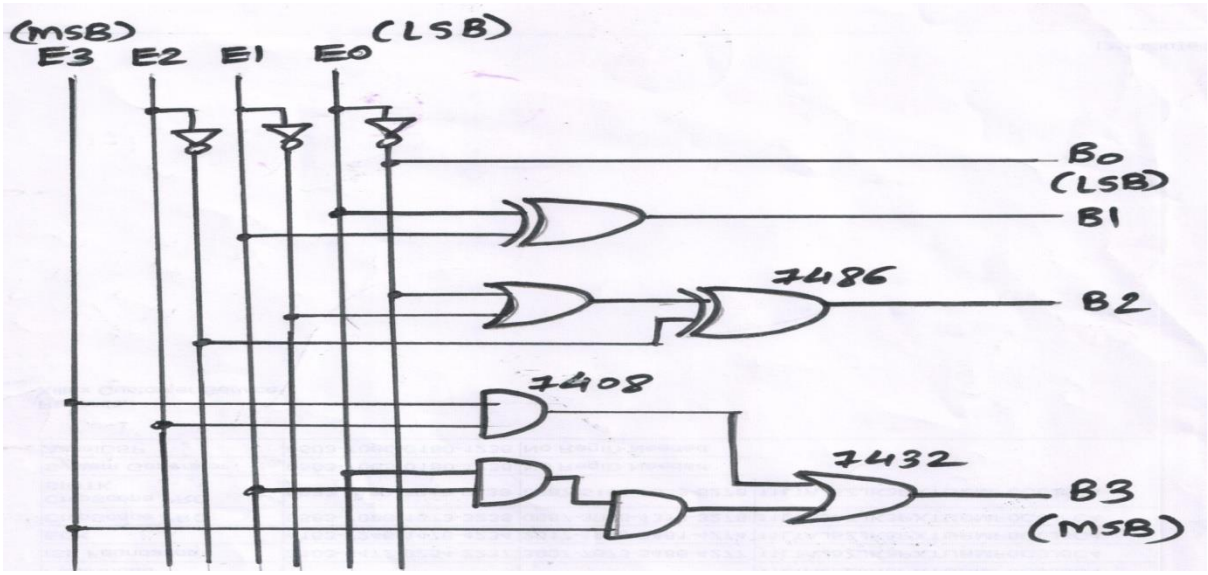


Fig.11 Logic Diagram for Excess-3 to BCD Conversion

4) Hardware Requirements Table:

GATE	Quantity	IC	Quantity
XOR	2	7486	1
NOT	3	7404	1
AND	3	7408	1
OR	2	7432	1

Table 5 Hardware Requirement Table

Test the circuit for all possible combinations of input and output codes.

Conclusion:

Thus, we studied different codes and their conversions including applications.

The truth tables have been verified using IC 7486, 7432, 7408, and 7404.

Assignment No. 2: BCD Adder

Aim: Design and implement BCD adder using IC 7483

Objective: 1. Study the BCD arithmetic rules.

2. Comparison between binary and BCD codes.

Ic's used:

IC 7483 (4 bit Binary adder), 7432 (OR-gate), 7408 (AND-gate)

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

Add BCD digits 3 & 4

```
  0 0 1 1
+ 0 1 0 0
 ----------
  0 1 1 1
```

Answer is valid BCD number = **(7)** $_{BCD}$ & so 0110 is not added

5. **CASE II: Sum > 9 & carry = 0.**

Add BCD digits 6 & 5

```
  0 1 1 0
+ 0 1 0 1
 ----------
  1 0 1 1
```

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

```
  1 0 1 1
+ 0 1 1 0
 ----------
1 0 0 0 1
```

(1 1)$_{BCD}$

Valid BCD result = **(11)** $_{BCD}$

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

Add BCD digits 9 & 9

```
 1 0 0 1
+ 1 0 0 1
-----------
1 0 0 1 0
```

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

```
  0 0 1 0
+ 0 1 1 0
-----------
1 1 0 0 0
```

(1 8)

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, and S0) & only 1 output Y.

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

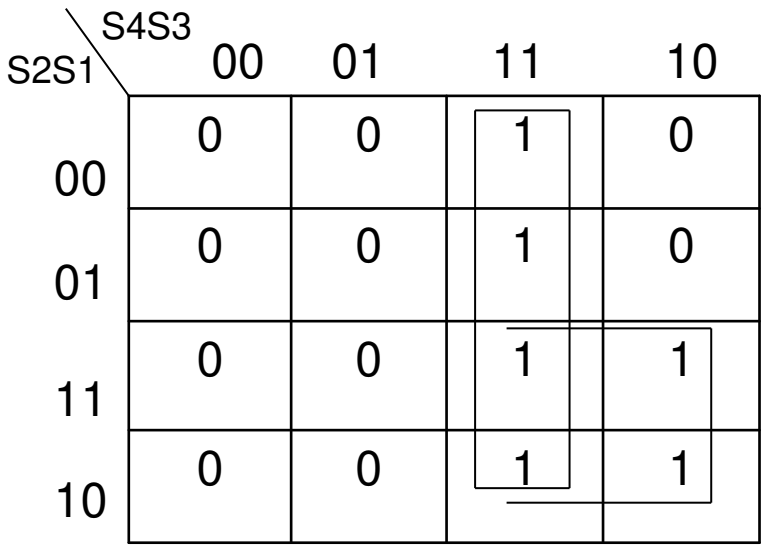
b) If BCD result is invalid i e. sum output of first 7483 we have to add (6)10 i.e. (0110)2 that means we need one more binary adder IC 7483.

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 carries & carry output of second adder IC is ignored.

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

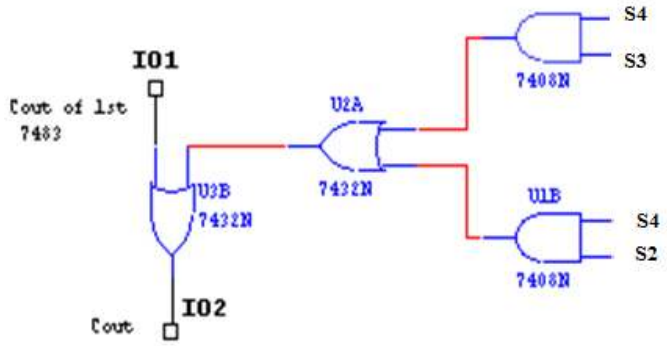
INPUT				OUTPUT
S4	**S3**	**S2**	**S1**	**Y**
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

ii) K-map for reduced Boolean expressions of output:

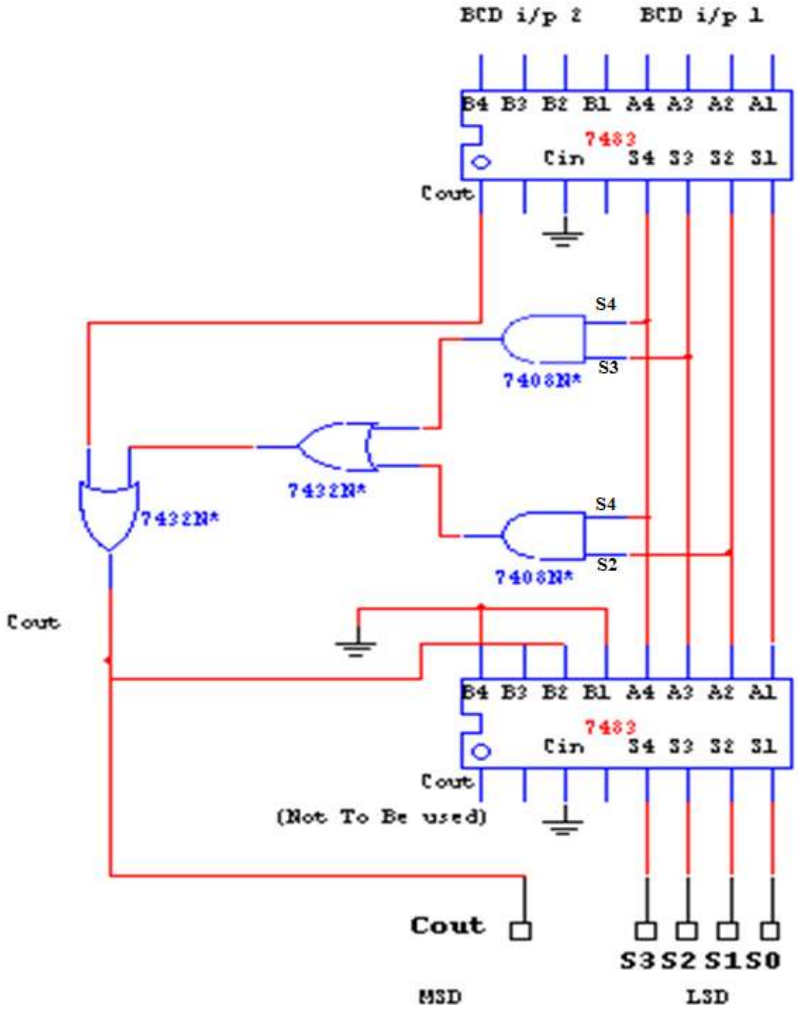


Y = S4S3 + S4S2

iii) Circuit diagram:



iv) Circuit diagram for BCD adder:



v) Hardware Requirements:

GATE	Quantity	IC	Quantity
Binary adder	2	7483	2
AND	2	7408	1
OR	2	7432	1

BCD adder:

INPUT								OUTPUT				
1st Operand				2nd Operand				MSD	LSD			
A3 (MSB)	A2	A1	A0 (LSB)	B3 (MSB)	B2	B1	B0 (LSB)	Cout	S3 (MSB)	S2	S1	S0 (LSB)

Conclusion:

BCD adder is designed & tested for all possible combinations.

Assignment 3: Multiplexer

AIM: MUX IC 74153

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

OBJECTIVE:

1. To study the difference between multiplexer, de-multiplexer and decoder.
2. To study the applications of multiplexer.

IC's USED:

IC 74153, 7404, 7432

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 multi position 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, De-multiplexer& 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 cannot 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.
3) To select between multiple sources & signal destination.
4) Inter register transfer.

Advantages:

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

Disadvantage:

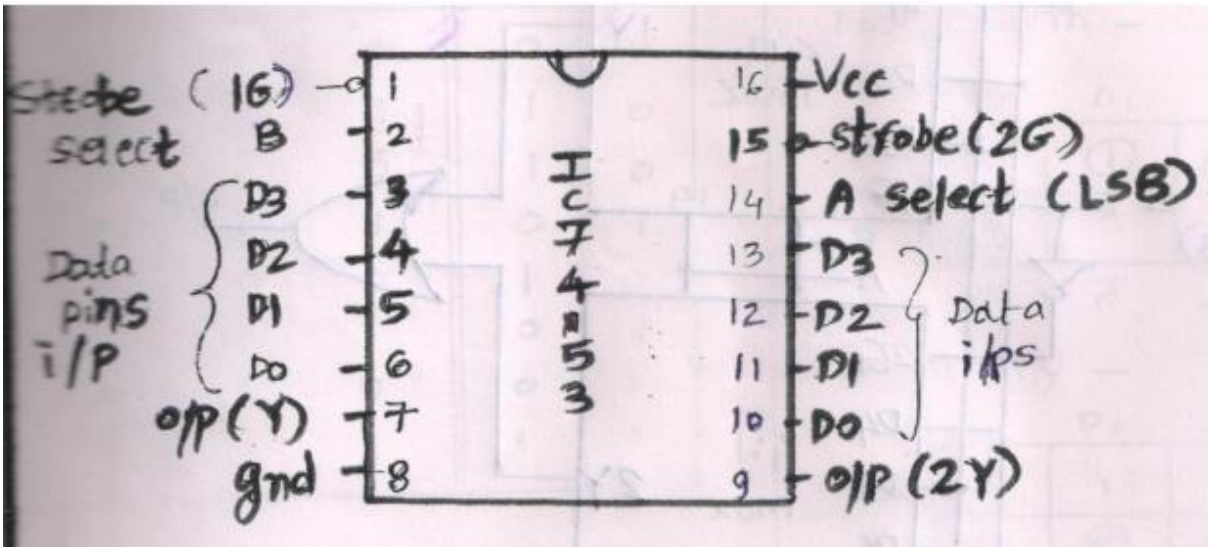
Only one function can be implemented using one MUX. Hence they can't be used in combinational logic circuit which contains many function.

Part-A (MUX IC 74153)

1. VERIFICATION OF IC 74153:

IC 74153 is a dual layer 4:1 MUX. It has four input lines for (I0D-I3D) for second MUX & active high output. 'Y_a', 'Y_b' (1Y or 2Y). It has select lines S_1 S_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.

Pin out of IC 74153:



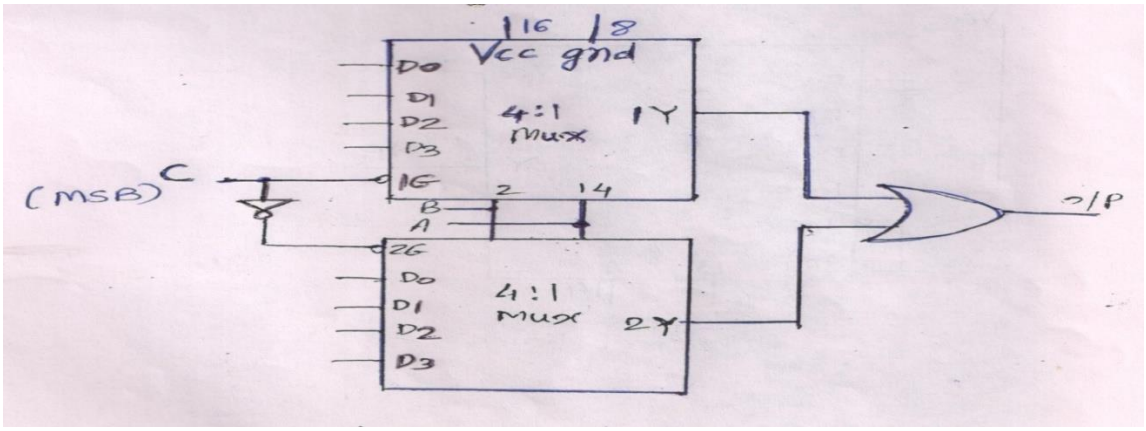
Function table of IC 74153: (X= Don't Care Condition)

			Inputs (I or II)				Output
S1	**S0**	$\overline{\textbf{E}}$ **(I or II)**	**D0**	**D1**	**D2**	**D3**	**Y**
X	X	1	X	X	X	X	0
0	0	0	0	X	X	X	0
0	0	1	1	X	X	X	1
0	1	0	X	0	X	X	0
0	1	1	X	1	X	X	1
1	0	0	X	X	0	X	0
1	0	1	X	X	1	X	1
1	1	0	X	X	X	0	0
1	1	1	X	X	X	1	1

2. Cascading of IC 74153:

Cascading is done to expand two or more MUX IC's to a digital multiplexer with larger no. of inputs i.e. multiplexer stocks or tress is designed. The enable input is used for cascading. In case of IC 74153 we have only two select lines. But for certain application 3 select lines are required then it can be obtained by cascading using enable. Now with 3 select lines we have 8 combinations. Out of this combination the MSB is O. MSB is 1 for last four combination so we can use these MSB to select any 1 MUX out of two by connecting it to E pin of first 4:1 MUX .

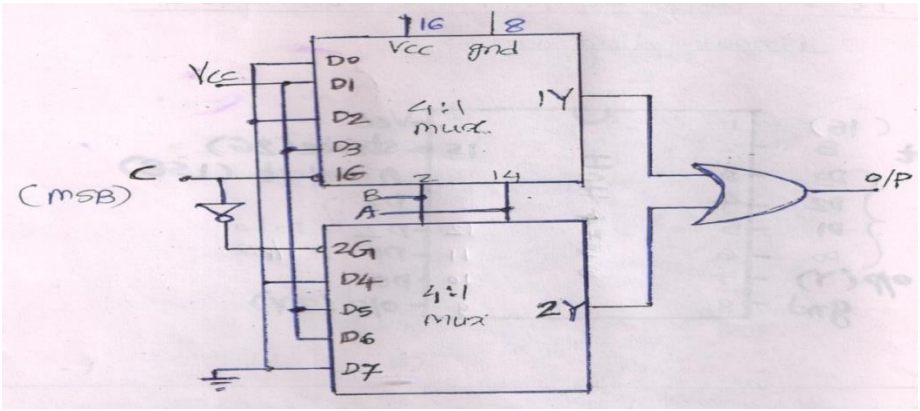
Logic Diagram:



Function table of IC 74153 as 8: 1 Mux by cascading 2 4:1 Mux:

Select Input			*Output*
C (1G / 2G)	*B (S_1)*	*A (S_0)*	*Y*
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

Logic diagram:



3. FUNCTION IMPLEMENTATION:

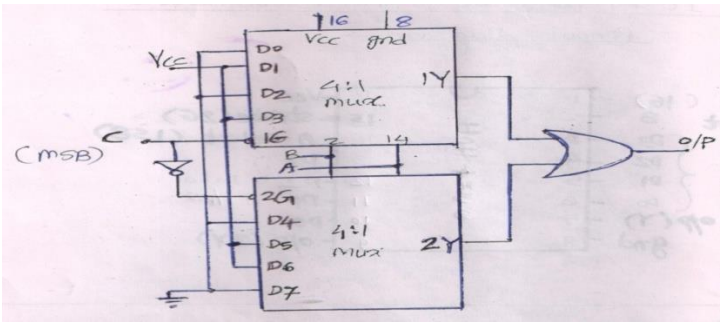
Y= ∑ m (1, 3, 5, 6)

This expression is in Standard SOP form and it is three variable functions. 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

Logic Diagram:



Hardware Requirements:

GATE	Quantity	IC	Quantity
Mux.	1	74153	1
NOT	1	7404	1
OR	1	7432	1

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)

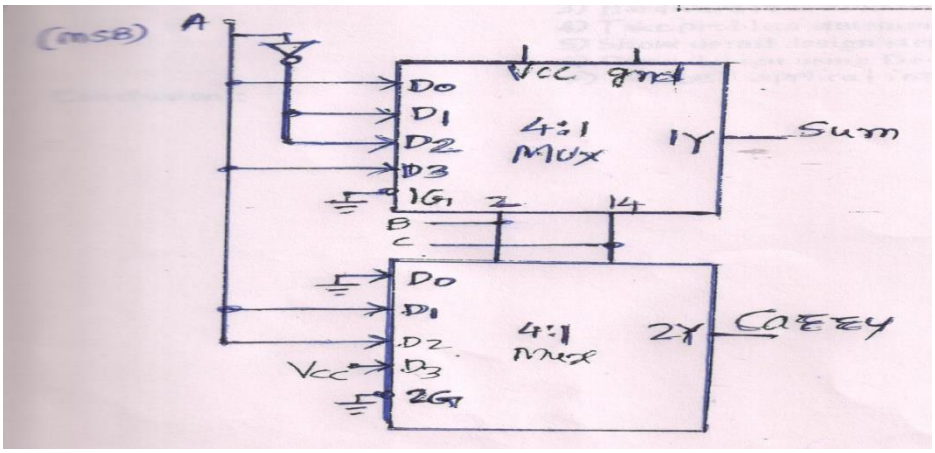
Hardware reduction table for Sum:

	D_0	D_1	D_2	D_3
$\underline{\bar{A}}$	0	1	2	3
A	4	5	6	7
i/p to MUX	A	$\underline{\bar{A}}$	$\underline{\bar{A}}$	A

Hardware reduction table for Carry:

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
i/p to MUX	0	A	A	1

Logic Diagram of Full Adder using IC 74153:



Hardware Requirements:

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

Conclusion:

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

Assignment 4: Decoder

AIM:

Decoder IC 74138

1. Verification of IC.
2. Full Subtractor

IC's USED:

IC 74138, 7404, 7432

THEORY:

Discrete quantities of information are requested in digital system with binary codes. A binary code of n bits is capable of representing into 2n 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=2n 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:

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.

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/p s, 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/p s 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

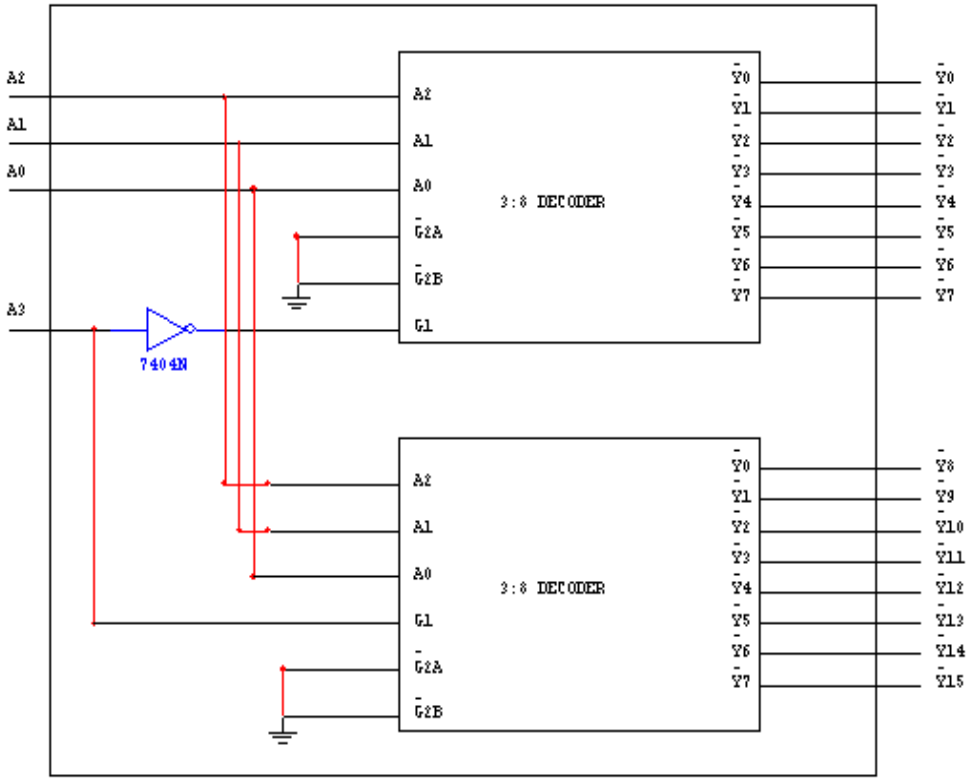
B) Cascading of IC 74138:

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



C) Implementation of Full Subtractor:

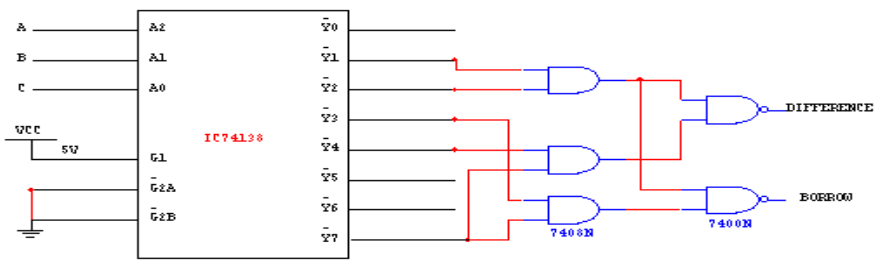
Same case will happen in this case. Again first of all we need to decide on which type of decoder the above Boolean function can be implemented. The highest minterm is 7 and minimum no. of bits required to represent it in binary form are 3. So we have 3 select lines in 3:8 decoders so we can use IC 74138.

To implement the function we require AND and NAND gate (7408 & 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

DIFFERENCE=∑m (1, 2, 4, 7), BORROW=∑m (1, 2, 3, 7)



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

GROUP B

(COMBINATIONAL LOGIC DESIGN)

Assignment No. 4: Asynchronous Counters

AIM:

Design and implement 3 bit up and 3 bit down Asynchronous Counters using master slave JK flip-flop IC 7476.

OBJECTIVE:

To understand design procedure of asynchronous counter

ICs USED:

IC 7476 (MS-JK Flip-flop), IC 7408(Quad 2 i/p AND Gate),
IC 7432 (Quad 2 i/p OR Gate) and IC 7404 (Hex Inverter)

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 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.

1) Asynchronous Up Counter:

Fig. 1 shows 3bit Asynchronous Up Counter. Here Flip-flop 2 acts 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 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:

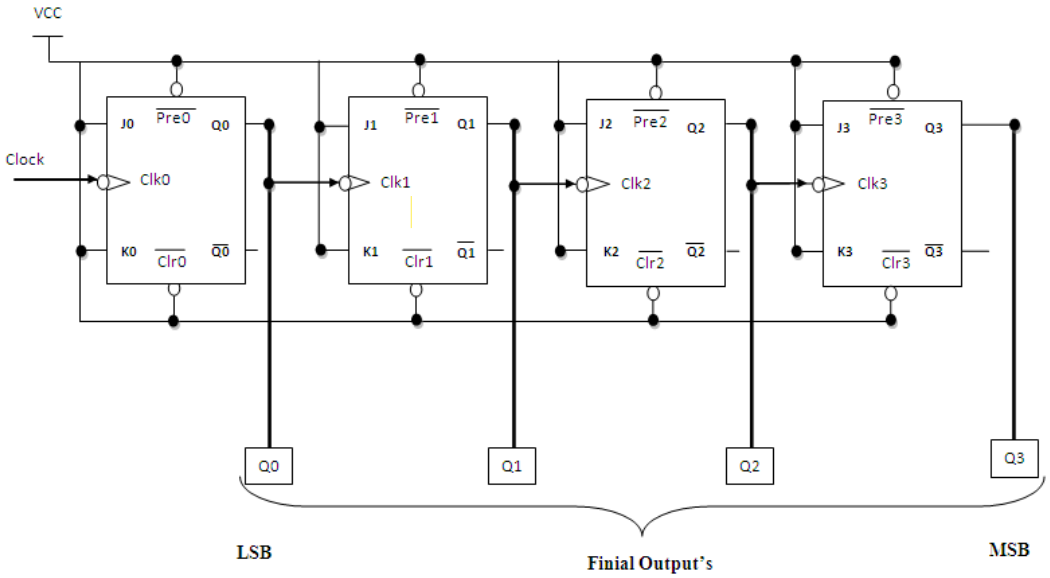
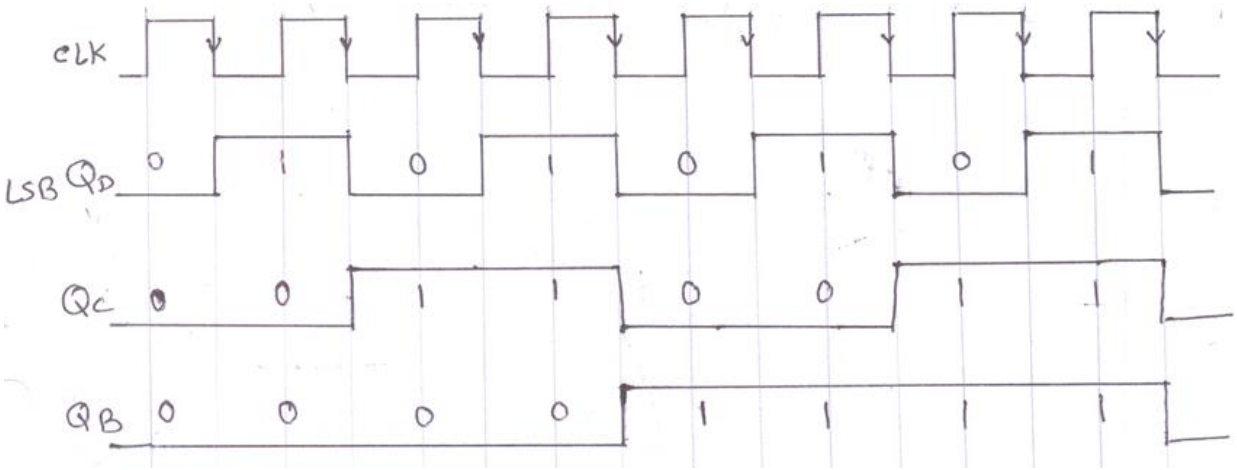


Fig 1: 3 Bit Asynchronous Up Counter

Hardware requirements:

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

Waveforms:



2) Down Counter:

Fig. 2 shows 2 bit Asynchronous Down Counter. Here Flip-flop 2 acts 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:

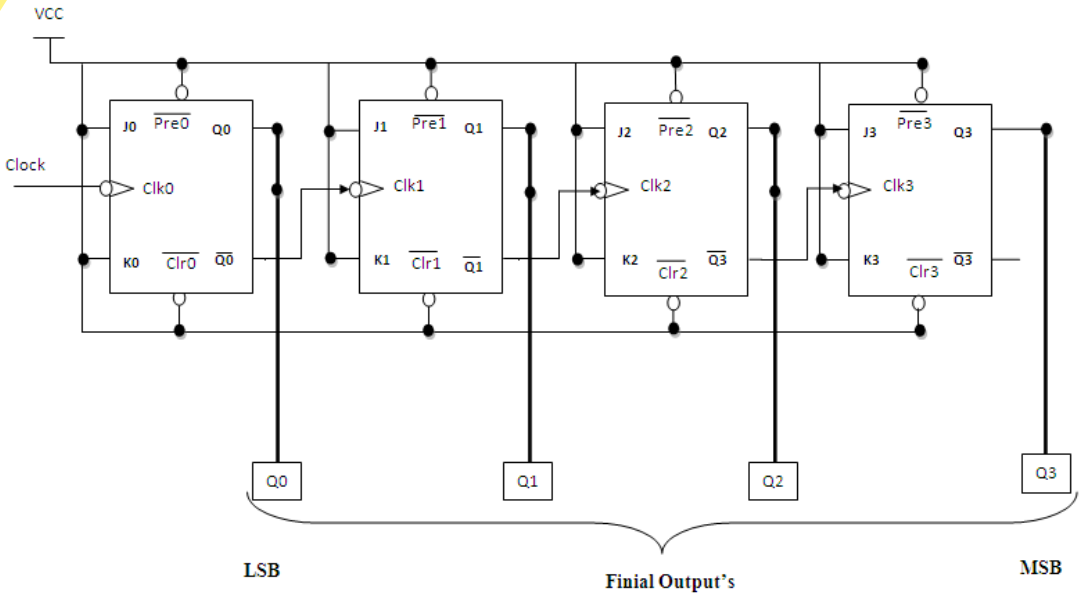
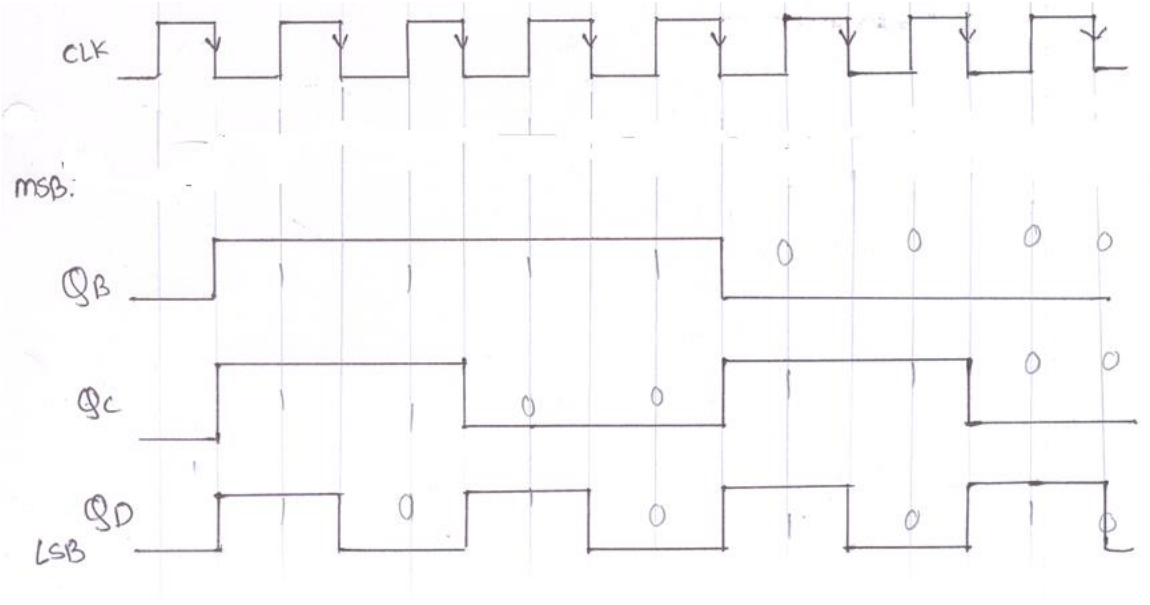


Fig 2: 3 Bit Asynchronous Down Counter

Hardware requirements:

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

Waveforms:



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.

Conclusion:

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

Assignment 5: Synchronous Counter

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

OBJECTIVE:
To understand design procedure of synchronous counter

ICs USED:
IC 7476 (MS-JK Flip-flop), IC 7408(Quad 2 i/p AND Gate),
IC 7432 (Quad 2 i/p OR Gate) and IC 7404 (Hex Inverter)

THEORY:
Types of Counters:
Counter are of two types:

1) Asynchronous counter.
2) Synchronous counter.

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.

Types of synchronous counter:

1) Up counter.
2) Down 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 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= Q1Q0

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

K_2= Q1Q0

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

J_1= Q0

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

K_1= Q0

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:

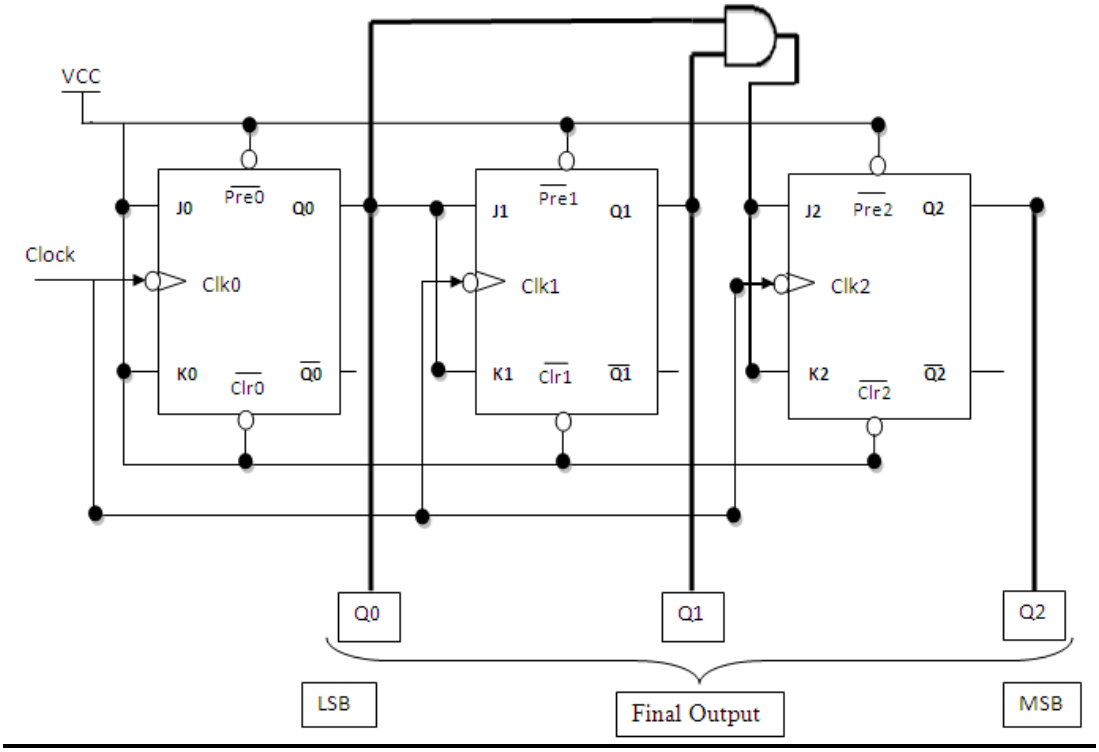


Fig 1: 3 bit Synchronous up counter

2. 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

$J_2 = \underline{Q1Q0}$

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

$K_2 = \underline{Q1Q0}$

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

$J_1 = \underline{Q0}$

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

$K_1 = \underline{Q0}$

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:

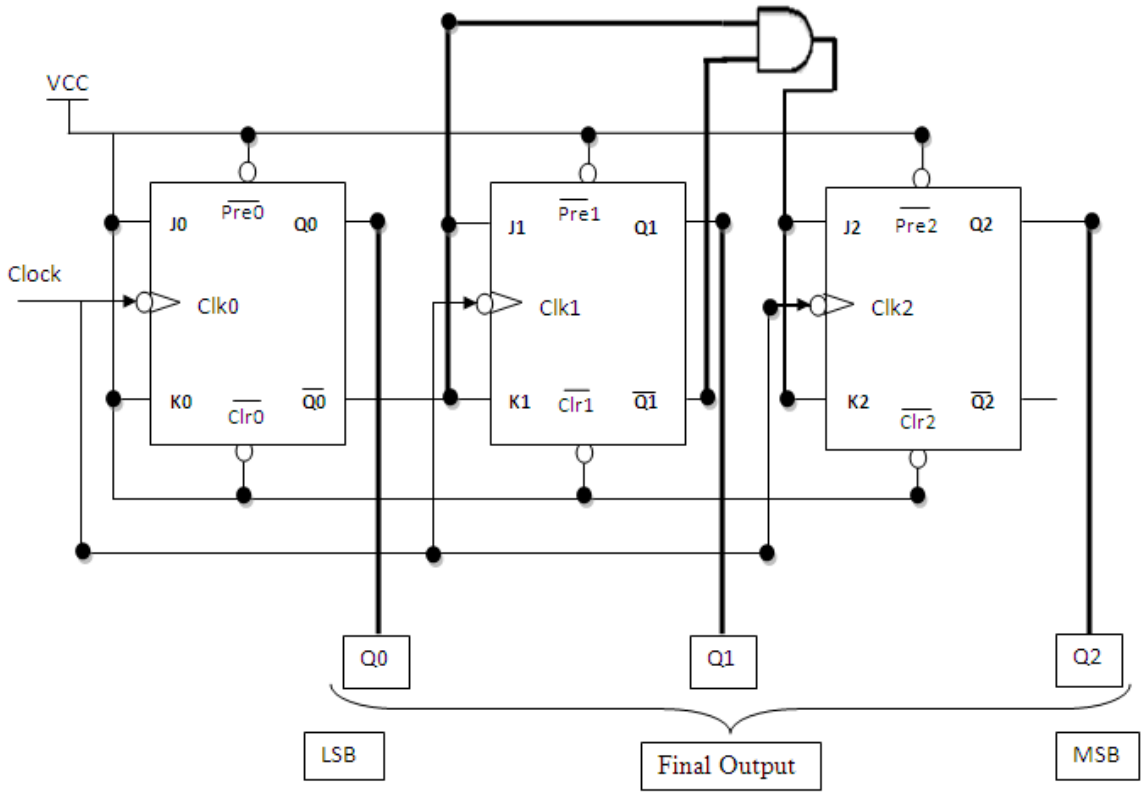


Fig 2: 3 bit Synchronous down counter

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.

Assignment 6: Modulo N Counter

AIM:

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

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

IC's USED: IC 7490, basic gates

THEORY:

IC 7490:

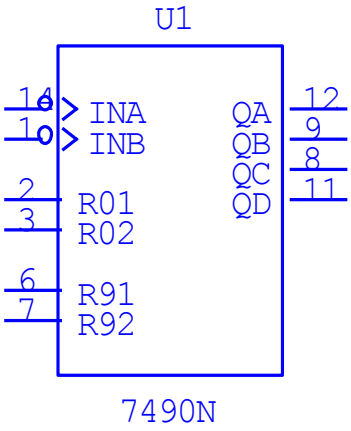
IC 7490 is a TTL MSI (medium scale integration) decade counter. It contains 4 master slave flip flops internally connected to provide MOD-2 i.e. divide by 2 and MOD-5 i.e. divide by 5 counters. MOD-2 and Mod-5 counters can be used independently or in cascading.

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.

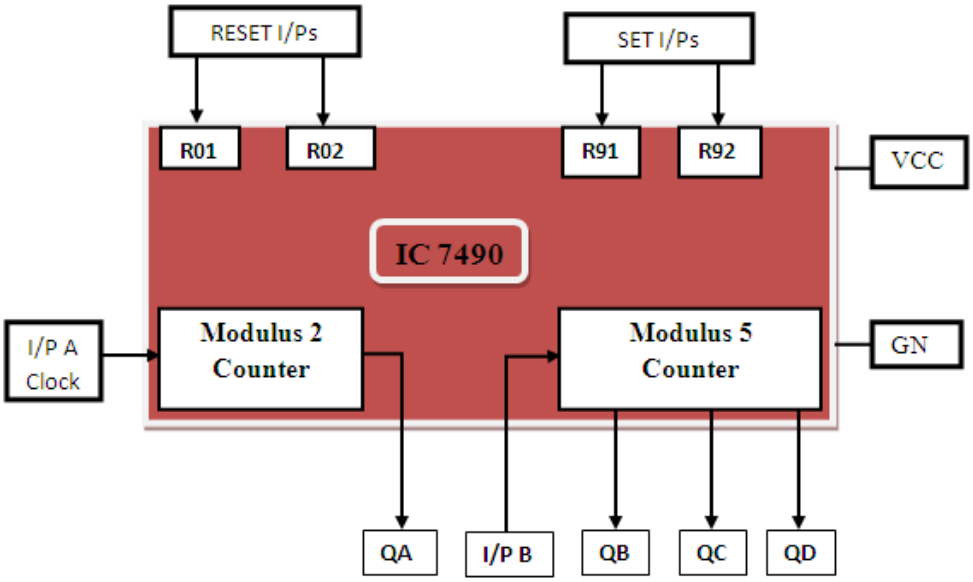
Since the o/p from the divide by 2 section is not internally connected to the succeeding stages. The device may be operated in various counting modes. In a BCD counter the CP_1 input must be externally connected to Q_A o/p. The CP_0i/p receives the incoming count producing a BCD count sequence. It is 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:

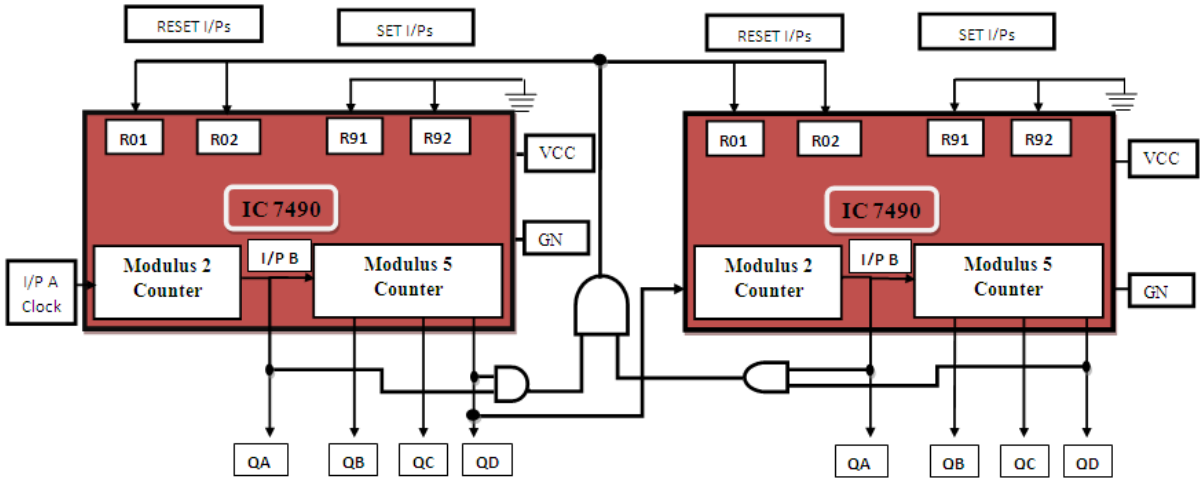


Function Table of MOD-2 counter:

Input A clock	Output	Count
↓	0	0
↓	1	1

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.



Conclusion:

Mod – N counters are successfully implemented, the counters are studied & o/p is checked. The state table is verified.

GROUP C
(COMPUTER ORGANIZATION)

Assignment 7

AIM:
To design and simulate single bit ALU with four function (AND, ADD, OR, XOR)

OBJECTIVE: Objective of 4 bit ALU (with AND, OR, XOR, ADD operation)

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:

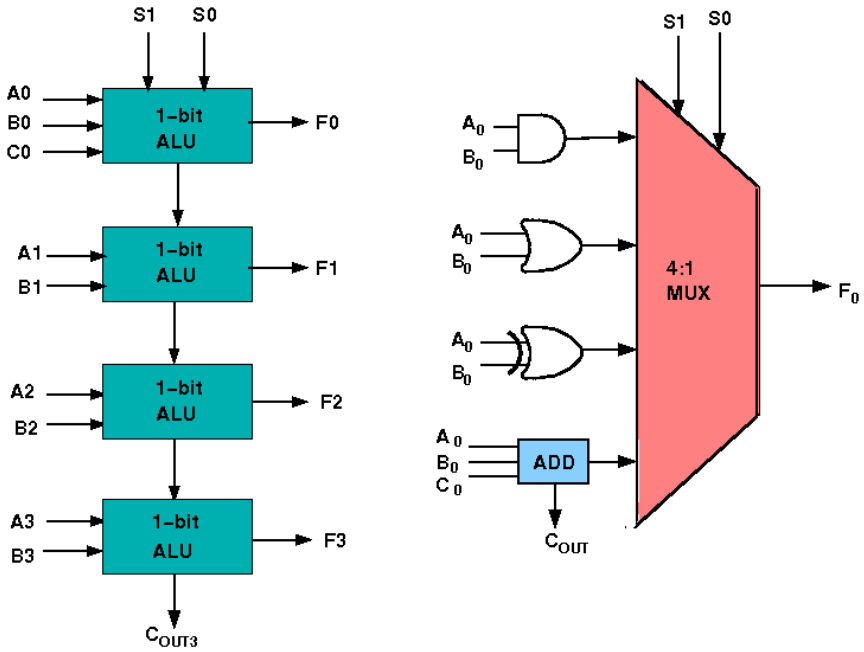


Fig. The 4-bit ALU block is combined using 4 1-bit ALU block

Design Issues:

The circuit functionality of a 1 bit ALU is shown here, depending upon the control signal S1 and S0 the circuit operates as follows:
For Control signal S1 = 0 , S0 = 0, the output is A And B,
For Control signal S1 = 0 , S0 = 1, the output is A Or B,
For Control signal S1 = 1 , S0 = 0, the output is A Xor B,
For Control signal S1 = 1 , S0 = 1, the output is A Add B.

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 `

<u>Procedure to perform the experiment: Design of 4 bit ALU</u>

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), and 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.

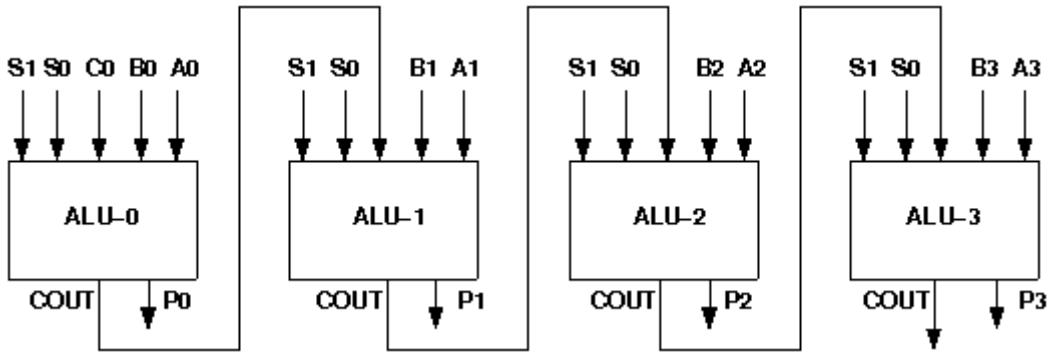
4. For 1-bit ALU input A0 is in pin-9,B0 is in pin-10, C0 is in pin-11 (this is input carry), for selection of operation, S0 is in pin-12, S1 is in pin-13, output F is in pin-8 and output carry is pin-7
5. Click on the 1-bit ALU component (in the Other Component drawer in the pallet) and then click on 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 palette.
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 combinations of different values and check the result. The operations are implemented using the truth table for 4 bit ALU given in the theory.

Components:

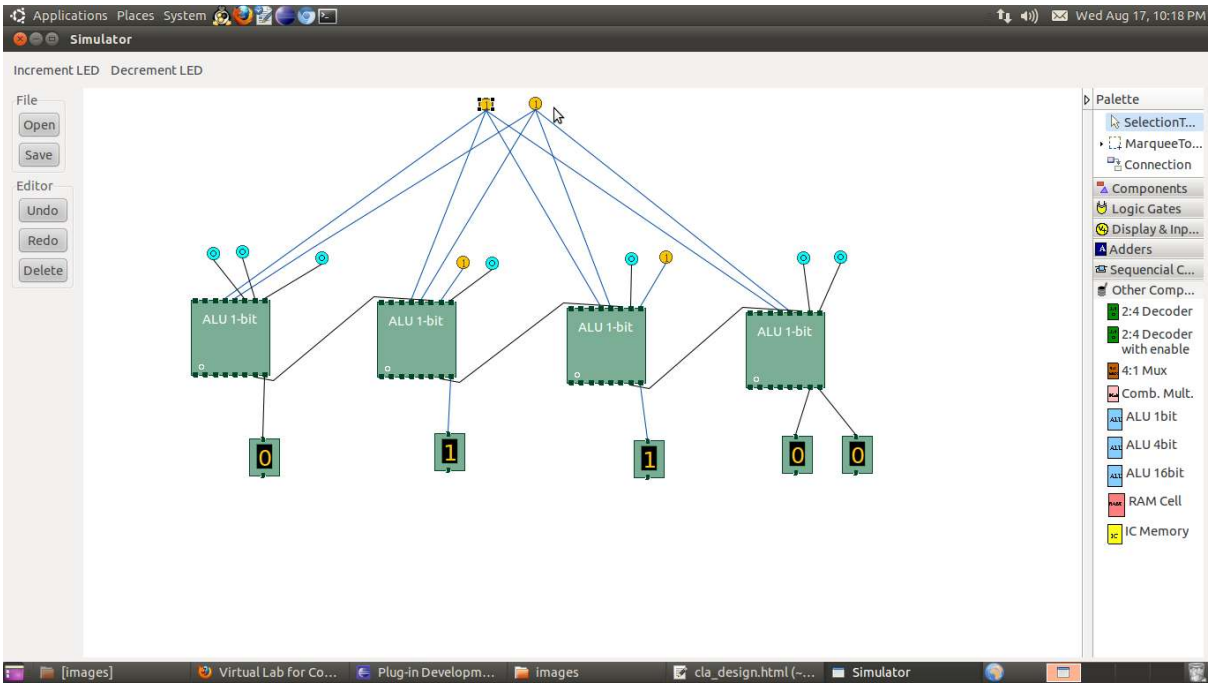
To build any 4-bit ALU, we need

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

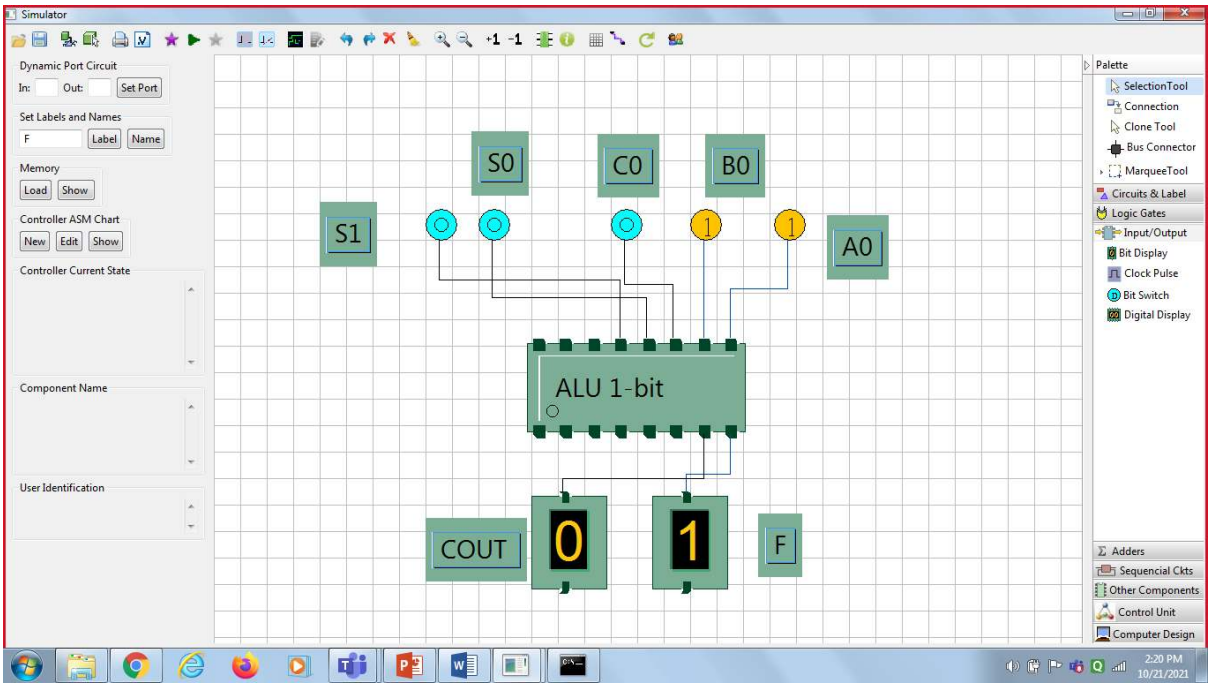
Circuit Diagram of4-bit ALU:

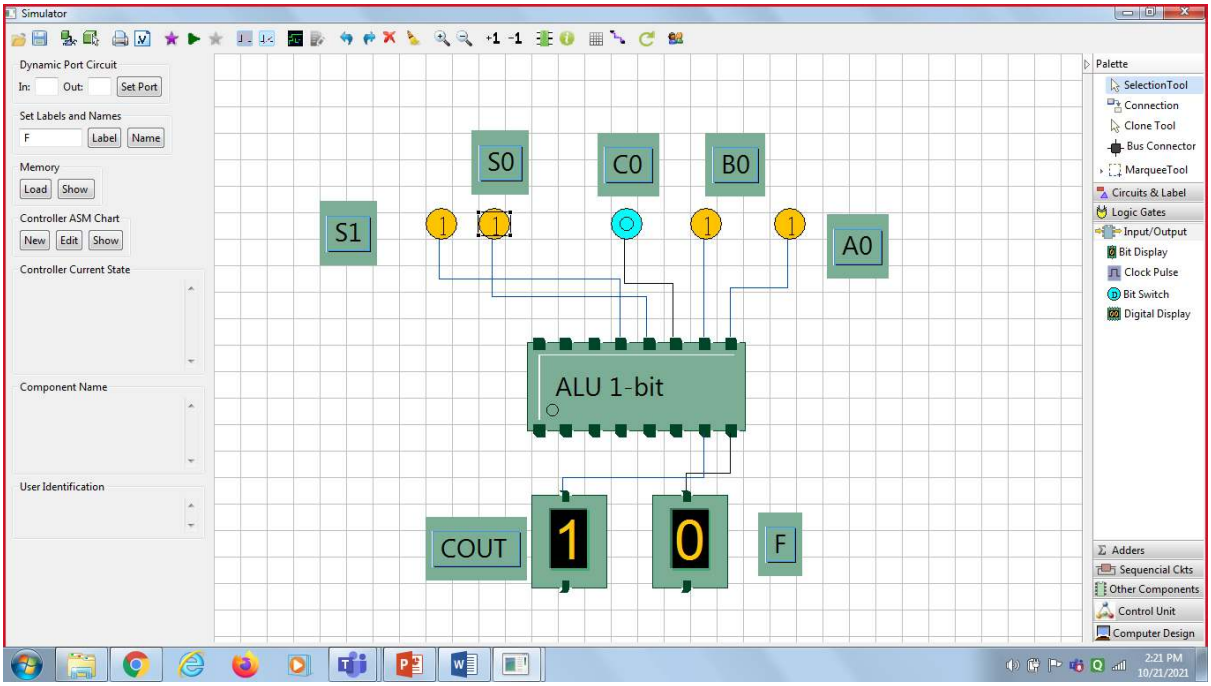


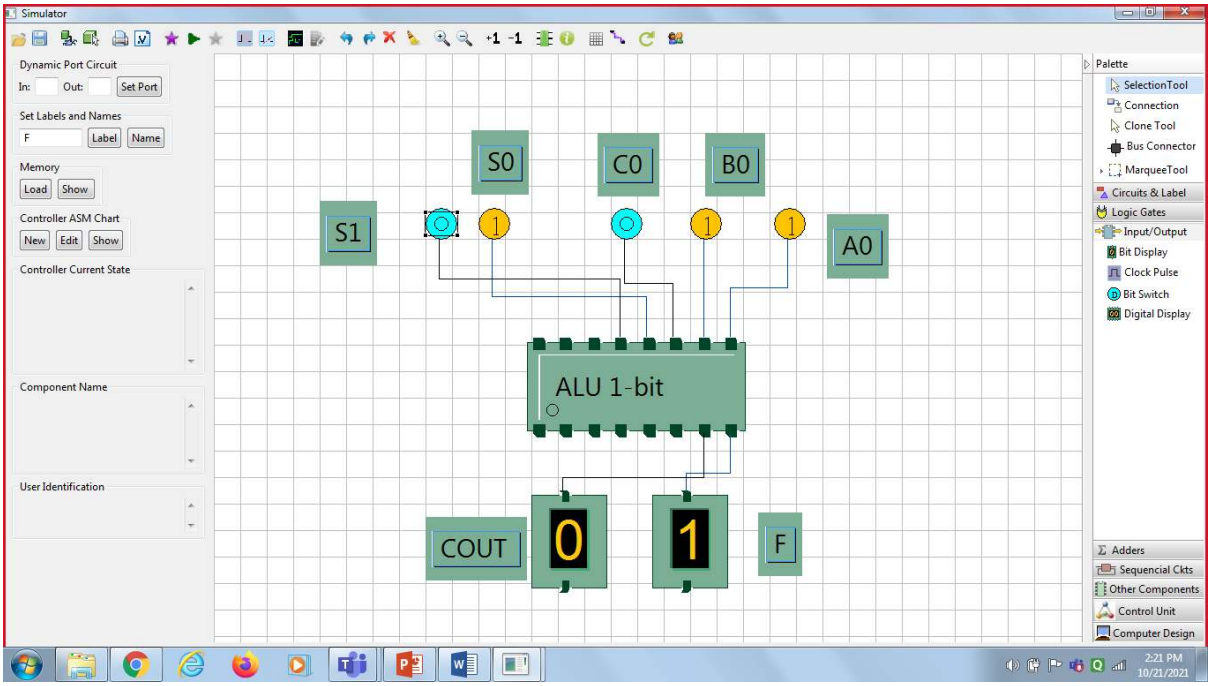
Screenshot of Design of 4-bit ALU:

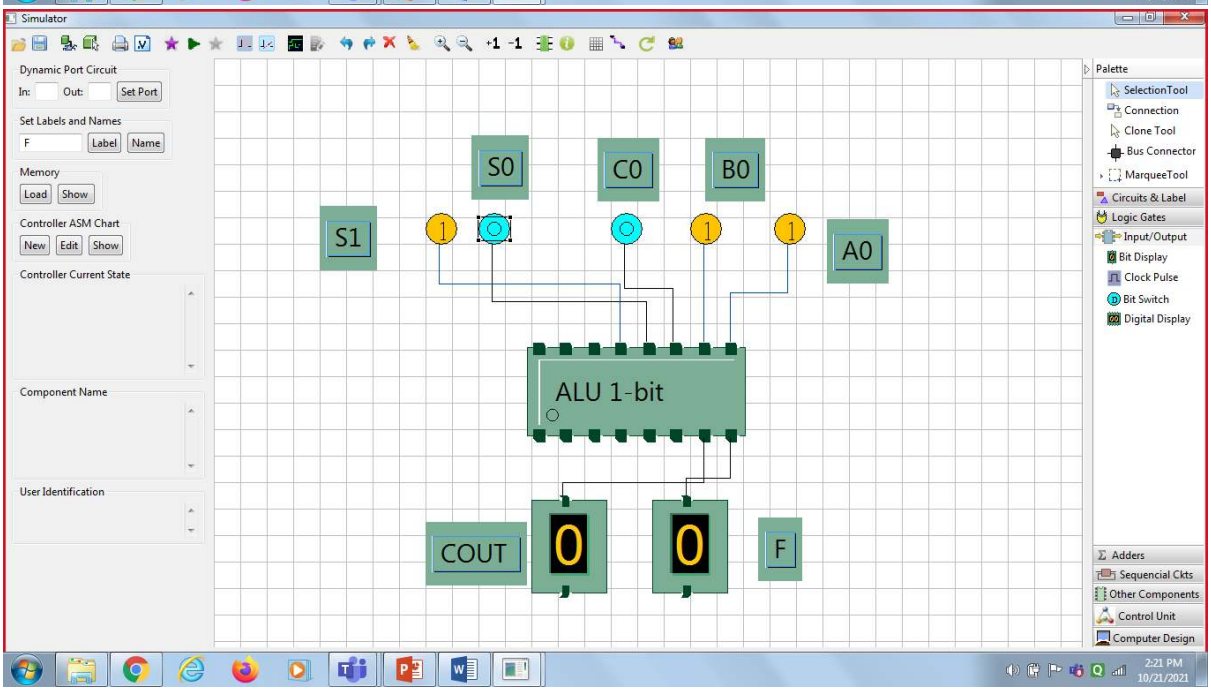


Output:









Conclusion:

Hence we have studied single bit ALU with different functionalities.

Assignment 9

AIM: To design and simulate single instruction CPU

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:

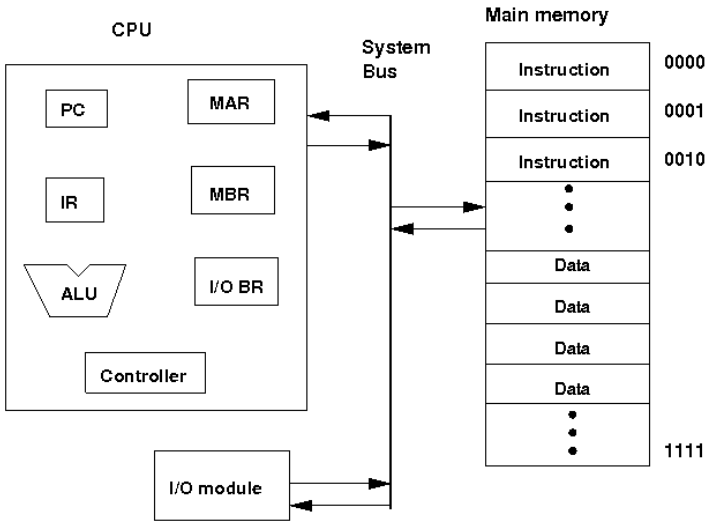
- Describing the external behaviour of each component i.e. the data and the control signals that it exchanges with other components
- Describing the interconnection structure and the controls required

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:

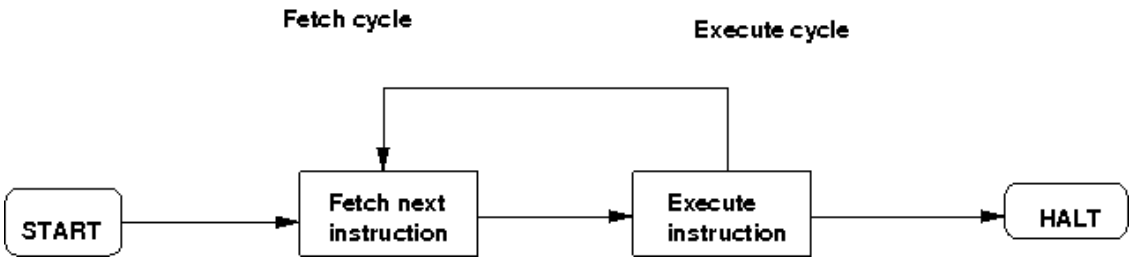


  - Memory address register (MAR) which specifies the address for the next read or write

- 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 vise 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 consist 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

In this experiment, no I/O module has been included for the purpose of simplicity. The working memory should contain the program and data in binary format. The CPU executes the program. For halting, it uses self loop, no other halt instruction is provided.

Conclusion:

Hence we have studied single instruction CPU.