LDCO LAB ASSIGNMENT -1

**NAME :**

**ROLL NO-** SEIT

**SUBJECT-** Logic Design and Computer Organization

**BATCH:**

**1. Code Converter**

## Title: Code Converter

**Aim**: Design and implementation of 4-bit Code convertors. i.BCD to Excess – 3 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(23, 22, 21, 20). It is easy to convert between BCD code numbers and the familiar decimal numbers. It is the main advantage of this code. With four bits, sixteen numbers (0000 to 1111) can be represented, but in BCD code only 10 of these are used. The six code combinations (1010 to 1111) are not used and are invalid.

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

It is observed that more number of bits are required to code a decimal number using BCD code than using the straight binary code. However in spite of this disadvantage it is very convenient and useful code for input and output operations in digital systems.

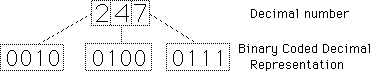


Fig. 3 BCD Coded Decimal Representation

## 1. 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](http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/number3.html#c1).The code for each decimal number is obtained by adding decimal 3 and then converting it to a 4-bit binary number. For e.g. decimal 2 is coded as 0010 + 0011 = 0101 in Excess-3 code.

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

## BCD To Excess – 3 Code Conversion:

Convert BCD 2 i. e. 0010 to Excess – 3 code

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

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

## Excess – 3 Code To BCD Conversion:

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

e.g. Convert 4-bit Excess-3 value 0101 to equivalent BCD code. 0101-0011= 0010- BCD for 2

## A. BCD To Excess-3 Code Conversion: Truth Table:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **INPUT (BCD CODE)** | | | | **OUTPUT (EXCESS-3 CODE)** | | | |
| **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 |

1. **K-Map For Reduced Boolean Expressions Of Each Output:**

**1. Code Converter**

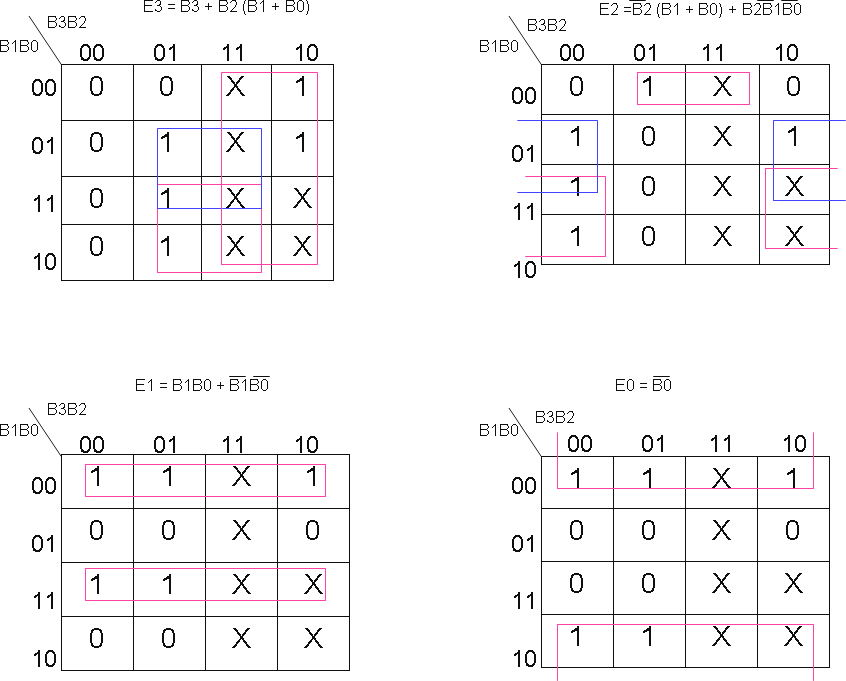


Fig. 8 K-Map For Reduced Boolean Expressions Of Each Output (Excess-3 Code)

## Circuit Diagram:

**BCD TO EXCESS-3 CONVERTER**

**1. Code Converter**

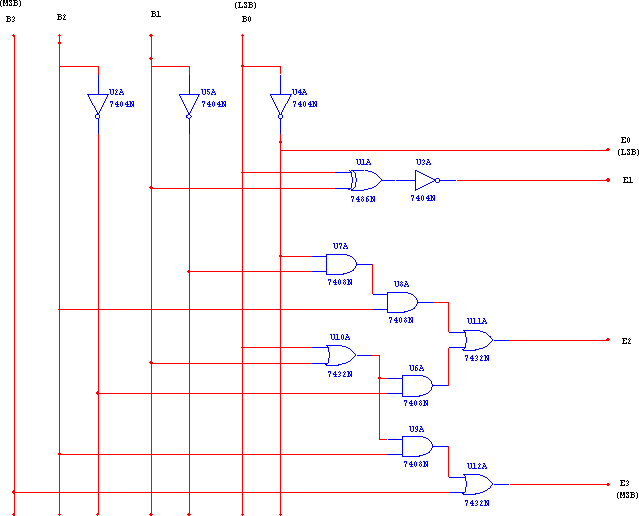


Fig.9 Logic Diagram for BCD to Excess-3 Code Conversion

## Hardware Requirements Table:

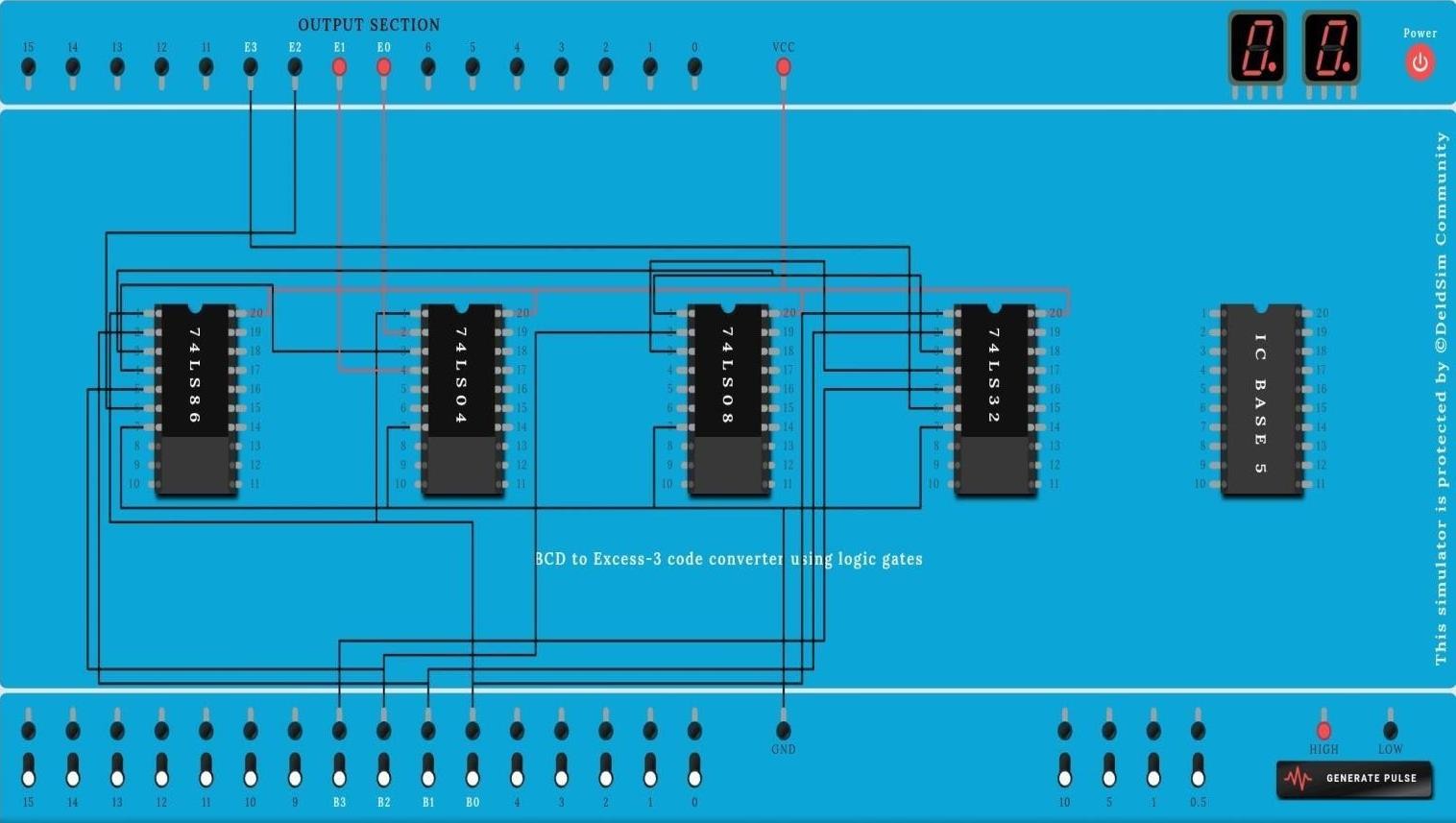
**1. Code Converter**

|  |  |  |  |
| --- | --- | --- | --- |
| **GAT E** | **Quantity** | **IC** | **Quantity** |

|  |  |  |  |
| --- | --- | --- | --- |
| XOR | 1 | 7486 | 1 |
| NOT | 4 | 7404 | 1 |
| AND | 4 | 7408 | 1 |
| OR | 3 | 7432 | 1 |

Table 5 Hardware Requirement Table

1.Code Converter :



## Conclusion:

Thus, we studied different codes and their conversions including applications. The truth tables have been verified using IC 7486, 7432, 7408, and 7404.

## FAQ’s with answers:

* 1. What is the need of code converters?

There is a wide variety of binary codes used in digital systems. 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.

* 1. What is Gray code?

**1. Code Converter**

It is a modified binary code in which a decimal number is represented in binary form in such a way that each Gray- Code number differs from the preceding and the succeeding number by a single bit.

(e.g. for decimal number 5 the equivalent Gray code is 0111 and for 6 it is 0101. These two codes differ by only one bit position i. e. third from the left.) It is non weighted code.

* 1. What is the significance of Gray code?

Important feature of Gray code is it exhibits only a single bit change from one code word to the next in sequence. Whereas by using binary code there is a possibility of change of all bits if we move from one number to other in sequence (e.g. binary code for 7 is 0111 and for 8 it is 1000). Therefore it is more useful to use Gray code in some applications than binary code.

* 1. What are applications of Gray code?

1. Important feature of Gray code is it exhibits only a single bit change from one code word to the next in sequence. This property is important in many applications such as Shaft encoders where error susceptibility increases with number of bit changes between adjacent numbers in sequence.
2. It is sometimes convenient to use the Gray code to represent the digital data converted from the analog data (Outputs of ADC).
3. Gray codes are used in angle-measuring devices in preference to straight forward binary encoding.
4. Gray codes are widely used in K-map
   1. What are weighted codes and non-weighted codes?

In weighted codes each digit position of number represents a specific weight. The codes 8421, 2421, and 5211 are weighted codes. Non weighted codes are not assigned with any

* + 1. **Code Converter**

weight to each digit position i.e. each digit position within the number is not assigned a fixed value. Gray code, Excess-3 code are non-weighted code.

* 1. Why is Excess-3 code called as self-complementing code?

Excess-3 code is called self-complementing code because 9’s complement of a coded number can be obtained by just complementing each bit.

* 1. What is invalid BCD?

With four bits, sixteen numbers (0000 to 1111) can be represented, but in BCD code only 10 of these are used as decimal numbers have only 10 digits fro 0 to 9. The six code combinations (1010 to 1111) are not used and are invalid.

LDCO LAB ASSIGNMENT -2

**NAME :**

**ROLL NO-**

**SUBJECT-** Logic Design and Computer Organization

**BATCH:**

**2. BCD & Excess 3 Adder AIM :** To design & implement of single digit BCD using IC 7483. **OBJECTIVE:** 1. To study the BCD arithmetic rules.

2. Comparison between binary and BCD codes.

## IC’s USED:

IC 7483 (4 bit Binary adder), IC 7404(Hex INV), 7432 (OR-gate), 7408 (AND-gate), 7486 (EX- OR 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.

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

Add BCD digits 3 & 4

1. 0 0 1 1

+ 0 1 0 0

0 1 1 1

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

## CASE II : Sum > 9 & carry = 0.

Add BCD digits 6 & 5

1. 0 1 1 0

+ 0 1 0 1

10 1

Invalid BCD (since sum > 9) so 0110 is to be added 2. 1

0 1 1

+ 0 1 1 0

**1** 0 0 0 1

## (1 1)BCD

Valid BCD result = **(11) BCD CASE III : Sum < = 9 & carry = 1.**

Add BCD digits 9 & 9

1. 1 0 0 1

+ 1 0 0 1

**1** 0 0 1 0

Invalid BCD ( since Carry = 1 ) so 0110 is to be added 2. 1 0 0 1 0

+ 0 1 1 0

1 1 0 0 0

## (1 8)BCD

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

**Design of BCD adder :**

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

will be zero.

**Circuit to check invalid BCD :**

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

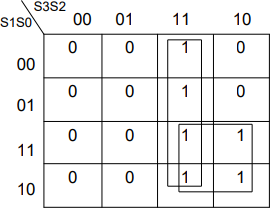
For digital circuit which we are going to design we will have 4 inputs ( S3, S2, S1, S0) & only 1 output Y.

* + 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 output carry & carry output of second adder IC is ignored.

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

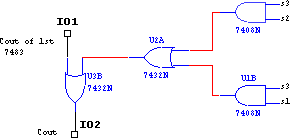
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INPUT** | | | | **OUTPUT** |
| **S3** | **S2** | **S1** | **S0** | **Y** |
| **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **1** | **0** |
| **0** | **0** | **1** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** |
| **0** | **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** |

1. **K-map for reduced Boolean expressions of output :**

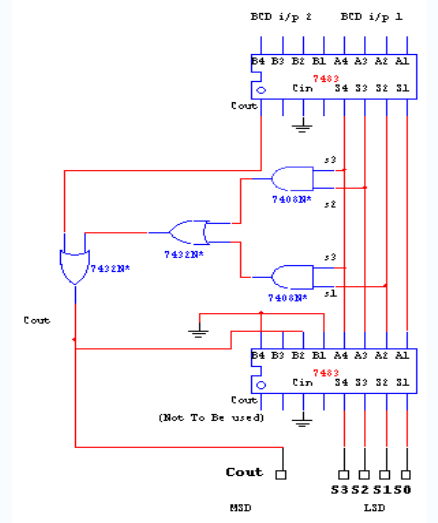


**Y = S3S2 + S3S1**

1. **Circuit diagram:**



1. **Circuit diagram for BCD adder :**



## Hardware Requirements :

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

**OBSERVATION :**

**BCD adder :**

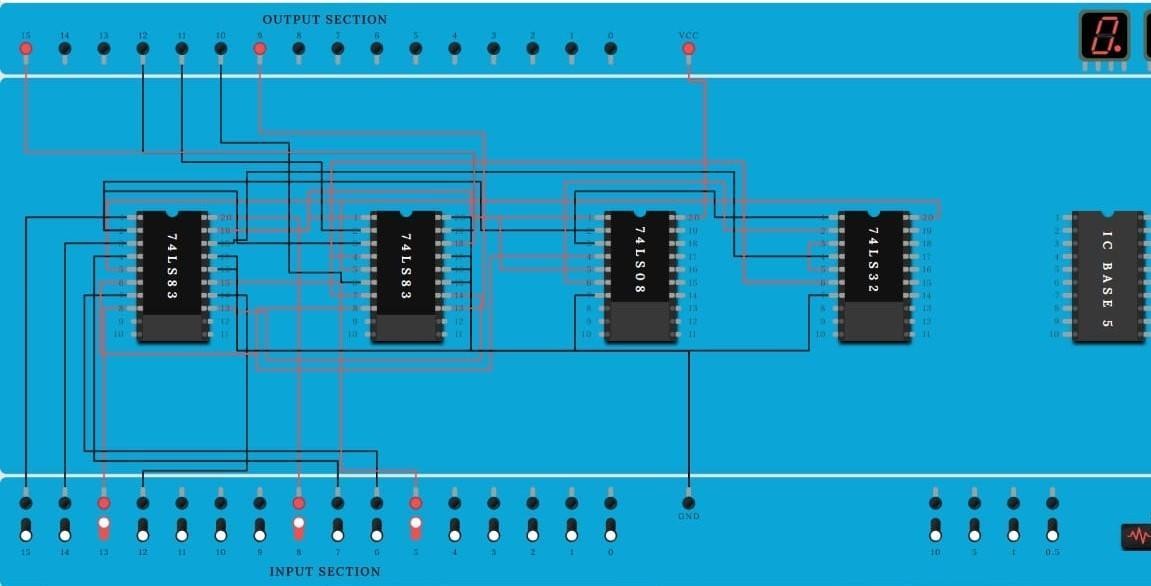
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **INPUT** | | | | | | | | **OUTPUT** | | | | |
| **1st Operand** | | | | **2nd Operand** | | | | **MSD** | **LSD** | | | |
| **A3 (MSB)** | **A2** | **A1** | **A0 (LSB**  **)** | **B3 (MS**  **B)** | **B2** | **B1** | **B0 (LSB**  **)** | **Cout** | **S3 (MS**  **B)** | **S 2** | **S 1** | **S0 (LSB**  **)** |
| **0** | **1** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | **1** |
| **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **0** |

1. **BCD & Excess 3 Adder**

**CONCLUSION :**

BCD adder is designed using IC 7483 & tested for all possible combinations**.**

**OUTPUT:**



LDCO LAB ASSIGNMENT - 3

**NAME-**

**ROLL NO-**

**SUBJECT-** Logic Design and Computer Organization

**BATCH**

1. **Study of Multiplexer & Decoder**

## AIM: Part A – MUX IC 74153

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

## OBJECTIVE:

1. To study the multiplexer, Types of Multiplexer.
2. To study the applications of multiplexer.

## IC’s USED :

IC 74153, 7404.

## THEORY :

**Multiplexer:**

Multiplexer are combinational digital circuits equating as controlled switches with several data inputs (I0, I1, I2 …) & one single data output (“out”).At any time one of the I/p is transmitted to output. According to binary signals applied on control pairs to circuit. Usually the number of data inputs is a power of two. Multiplexing is the process of transmitting a large no. of information units over a small no. of channel / digital multiplexer is a combinational large

circuit which performs the operation of multiplexing. It selects the operation of multiplexing. It selects the operation of binary information from one of the many input lines & transfer to a single o/p line. Multiplexer is called a data selector or multiposition switch because it selects one of the many input. Selection of a particular line is controlled by a set of a selection lines or selects inputs. The number of select lines depends upon no. of input lines. Generally, there is ‘n’ selects line for ‘m’ input lines. By applying a particular code on select lines is transmitted on the output lines. Block diagram of MUX is shown. at contains ‘2m’ input lines ‘m’ select & one unable input which is used to activate.

**Study of Multiplexer**

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.

## Uses of Mux. :

Use for Boolean function implementation. Construct a common bus system.

To select between multiple sources & signal destination. Inter register transfer.

## Advantages :

Simplification of logic expression not required. Logic design is simplified.

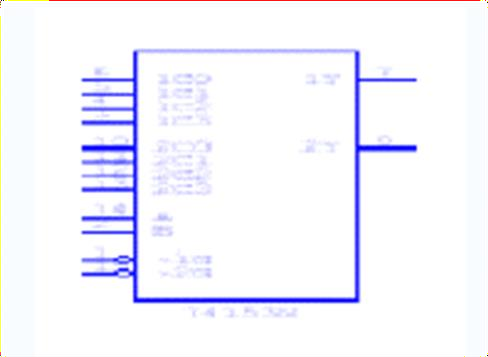
## Disadvantage :

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

***Part-A (IC 74153*)**

## 1. VERIFICATION OF IC 74153 :

IC 74153 is a dual layer 4:1 MUX. It has four input lines for (I0D-I3D) for second MUX & active high output. ‘Ya’, ‘Yb’ (1Y or 2Y). It has select lines S1S0 common to both MUX. The Enable inputs are active low, Ea & Eb (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*** | ***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 | 0 | 1 | X | X | X | 1 |
| 0 | 1 | 0 | X | 0 | X | X | 0 |
| 0 | 1 | 0 | X | 1 | X | X | 1 |
| 1 | 0 | 0 | X | X | 0 | X | 0 |
| 1 | 0 | 0 | X | X | 1 | X | 1 |
| 1 | 1 | 0 | X | X | X | 0 | 0 |
| 1 | 1 | 0 | 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** :

S

el

Y

|  |  |  |  |
| --- | --- | --- | --- |
| ***Select Input*** | | | ***Output*** |
| ***C (1G / 2G )*** | ***B (S1)*** | ***A (S0)*** | ***Y*** |
| 0 | 0 | 0 | D0 |
| 0 | 0 | 1 | D1 |
| 0 | 1 | 0 | D2 |
| 0 | 1 | 1 | D3 |
| 1 | 0 | 0 | D4 |
| 1 | 0 | 1 | D5 |
| 1 | 1 | 0 | D6 |
| 1 | 1 | 1 | D7 |

**3. FUNCTION IMPLEMENTATION:**

**Y = ∑ m (1, 3, 5, 6)**

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

**Truth table :**

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

**LOGIC DIAGRAM :**

Y

In puts

## 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= ∑m (1, 2, 4, 7), Carry=∑m

(3, 5, 6, 7)

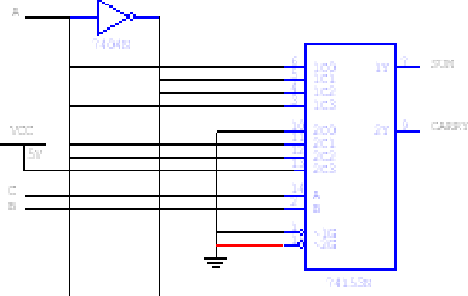
## Hardware reduction table for Sum:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **D0** | **D1** | **D2** | **D3** |
| **A** | 0 | 1 | 2 | 3 |
| **A** | 4 | 5 | 6 | 7 |
| **i/p to MUX** | A | A | A | A |

**Hardware reduction table for Carry:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **D0** | **D1** | **D2** | **D3** |
| **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:



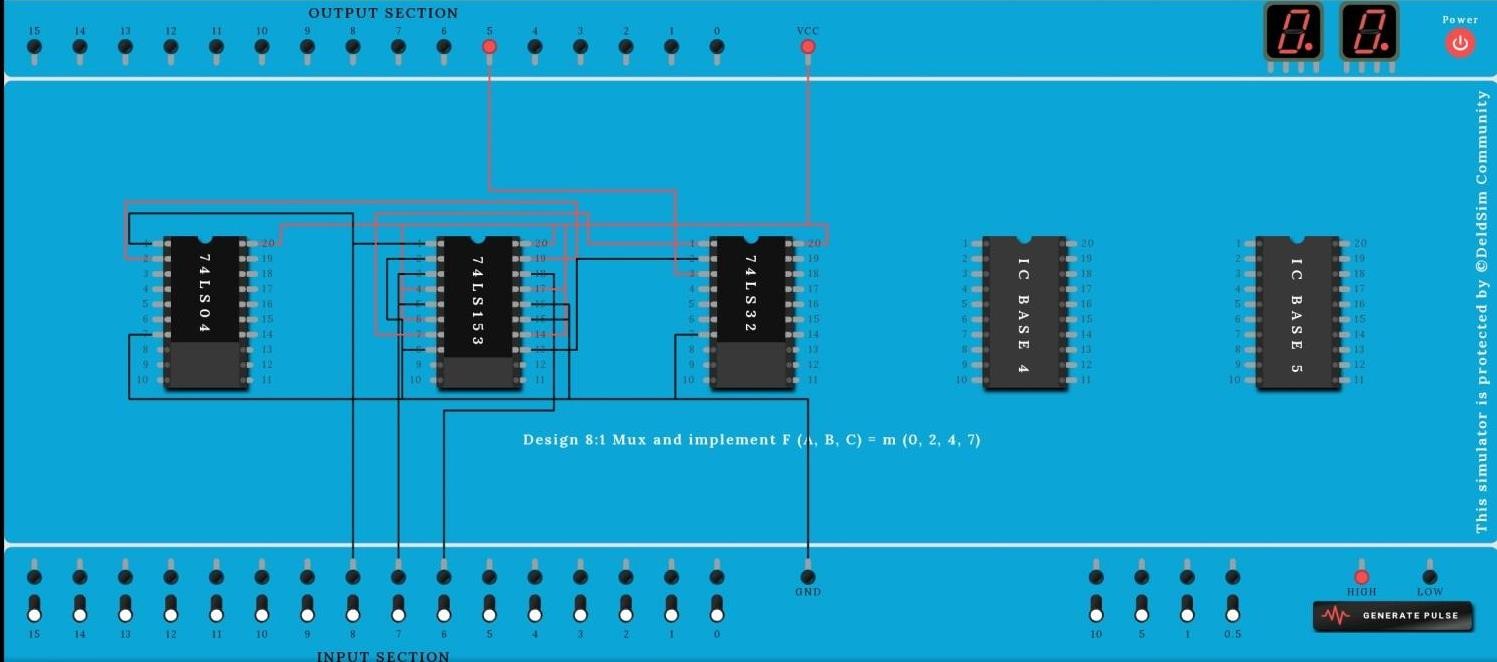
1. **Study of Multiplexer**

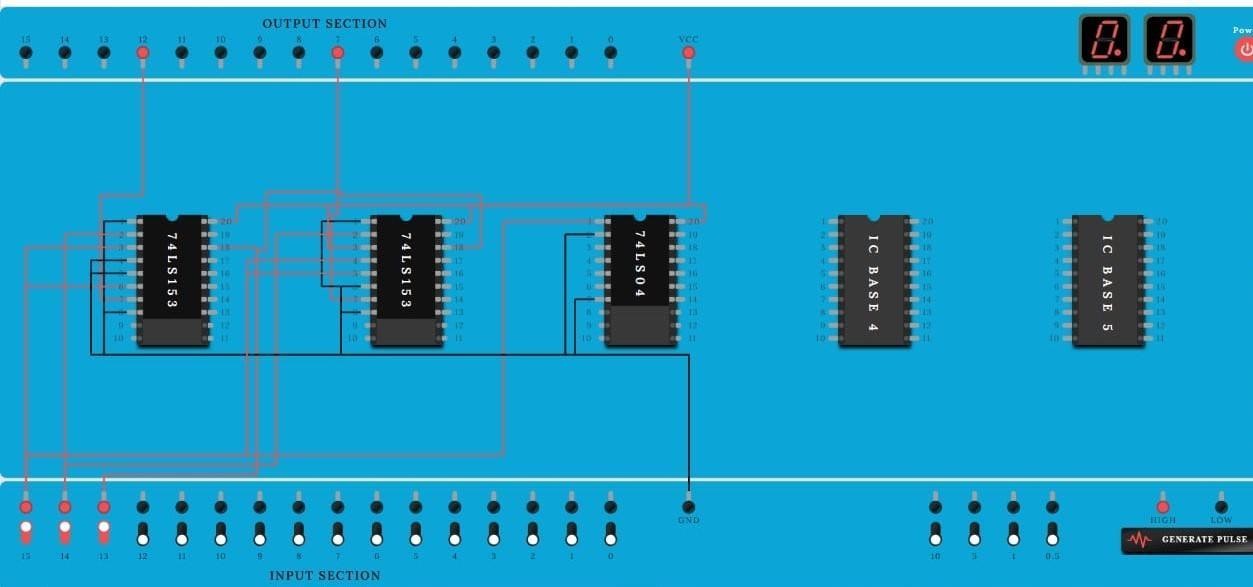
## Hardware Requirements :

|  |  |  |  |
| --- | --- | --- | --- |
| **GATE** | **Quantity** | **IC** | **Quantity** |
| Mux. | 1 | 74153 | 1 |
| NOT | 1 | 7404 | 1 |

**Conclusion :** In this way multiplexer function and full adder is Designed, implemented & tested. & its applications are studied, implemented & tested.

**OUTPUT:**





LDCO LAB ASSIGNMENT -4

**NAME :**

**ROLL NO-**

**SUBJECT-** Logic Design and Computer Organization

**BATCH:**

1. **Study of Decoder**

## AIM: Decoder IC 74138

1. Verification of IC.
2. Implementation of Full Subtractor using IC 74138.

## OBJECTIVE:

1. To study the difference between multiplexer, demultiplexer and decoder.
2. To study the applications of Demultiplexer.

## IC’s USED :

IC 74138,7410.

## THEORY :

**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 | 2m = n | n = 2m | 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.

1. Decoders are used to decode a coded binary word like BCD to seven segment decoder.
2. Thus encoder and decoder are application specific logic develop, we can not use any type of input for any encoder and decoder.
3. Need to select input according to encoder and decoder being selected for a particular application as mention in examples above.

## 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/ps. Two active low (G2A, G2B) and one active high (G1). Every output will be high unless G2A, G2B are low and G1 is high. It has demultiplexing capability and multiple enable i/ps for easy expansion.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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).

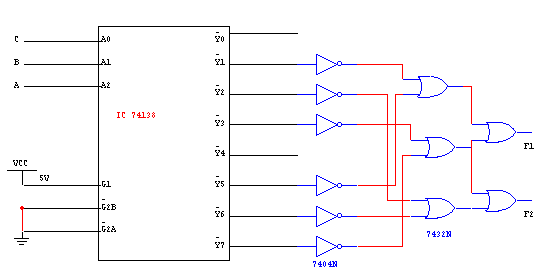
The i/p line A3 is used to enable /disable the 2 IC 74138 decoders. When A3=0, first IC is enabled and second is disabled. Thus the first decoder will generate minterms from 0000 to 0111 as o/p and the second decoder will generate nothing. When A3=1, the enable conditions are reversed and thus second 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 Boolean function:**  The procedure for implementation of combinational circuit by means of a decoder and ‘OR’ gates requires that the Boolean function fir the circuit be expressed in Sum of Minterms.  These  forms can be obtained by expanding the function. A decoder is then chosen which generates all | | | | | | | | | | | | | | | | | | | | |  |

the minterms of n i/p variables. The i/p to each OR gate are selected from the decoder outputs according to the minterms list in each function.



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

**D) 1) Implementation of Full Adder:**

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 SUM and CARRY when respective minterms are selected.

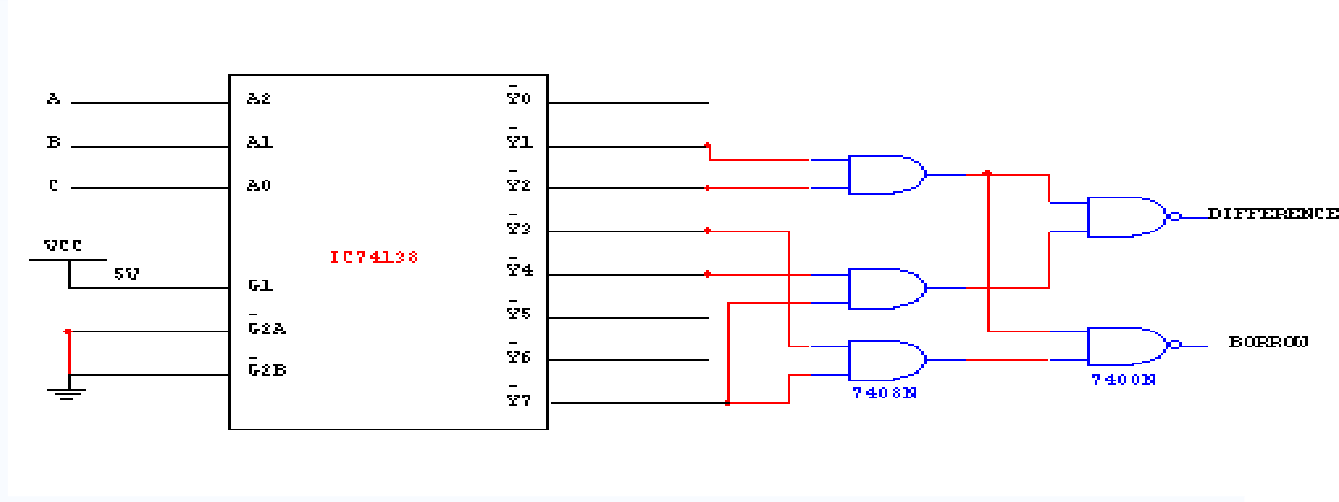
**Truth Table for design of Full Adder**:

**SUM=∑m (1, 2, 4, 7) CARRY=∑m (3, 5, 6, 7)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INPUT** | | | **OUTPUT** | |  |
| **A2** | **A1** | **A0** | **SUM** | **CARRY** |
| **A** | **B** | **C** |
| 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 |

**2) 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.



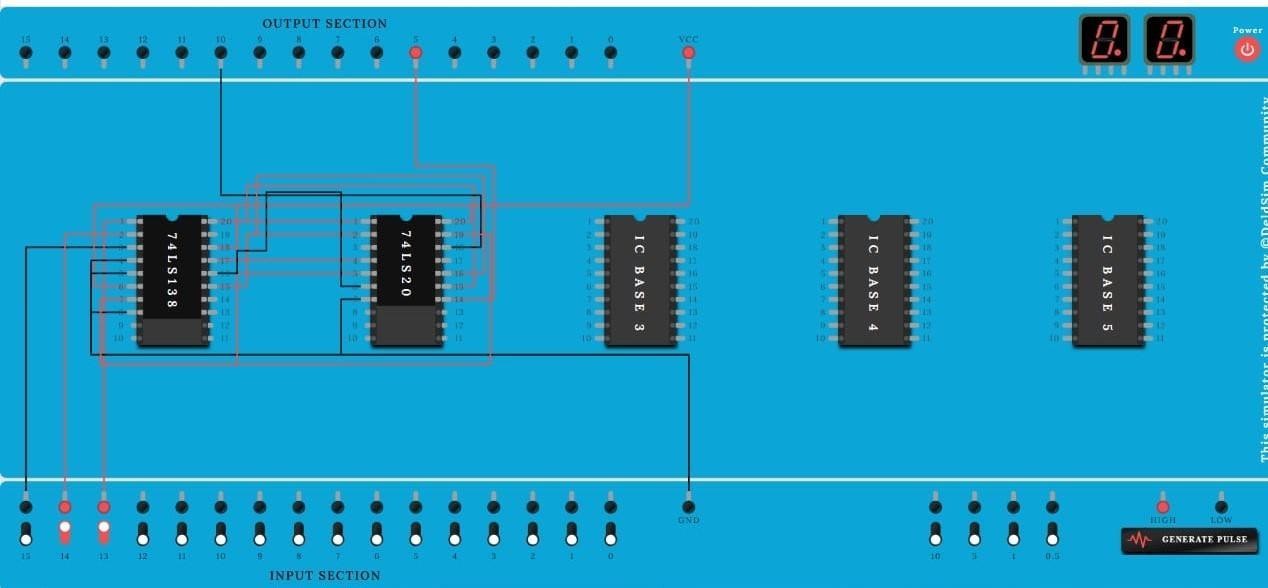
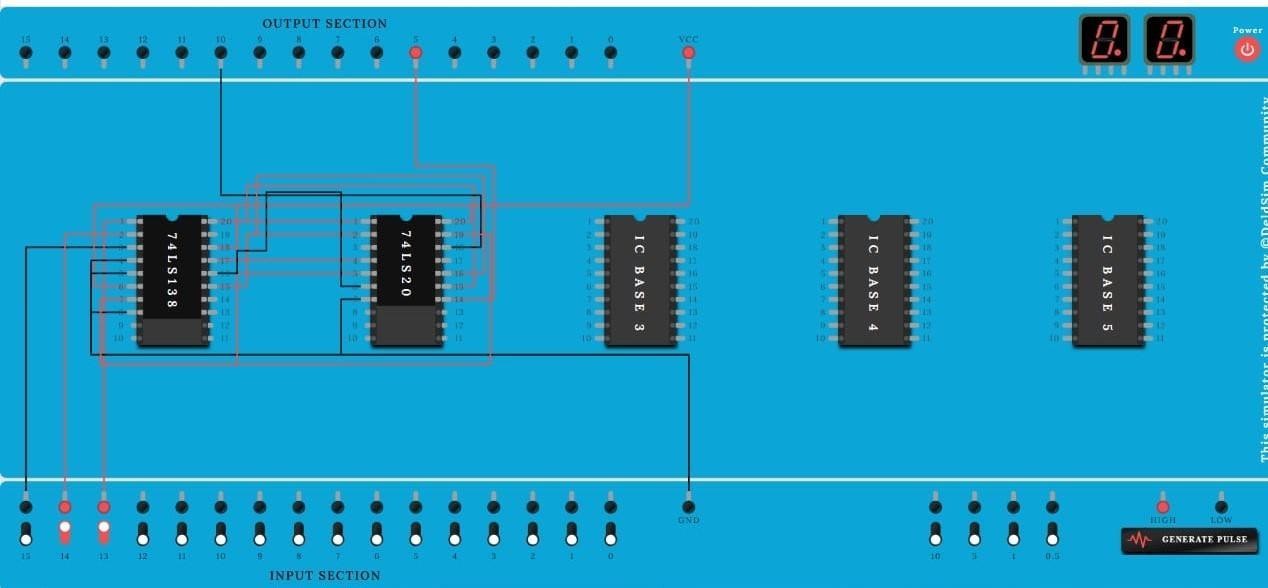
To implement the function we require AND and NAND gate (7408 & 7400). As the o/p of the decoder IC 74138 are active low and we need to get o/p active high at the o/p pin of the function DIFFERENCE and BORROW when respective minterms are selected.

**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.

OUTPUT:



**LDCO LAB ASSIGNMENT**

**NAME :**

**ROLL NO :**

**SUBJEC :** Logic Design and Computer Organization

**BATCH :**

# Counter

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

**OBJECTIVE:** To understand design procedure of asynchronous & 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:

**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 incase of module counters.

## Types of Counters:

Counter are of two types**:**

## Asynchronous counter.

1. **Synchronous counter.**

## Asynchronous counter:

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

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

In asynchronous counter commonly called ripple counter, the first flip-flop is clocked by the external clock pulse & then each successive flip-flop is clocked by the Q or Q’ output of the previous flip-flop. Therefore in an asynchronous counter the flip-flop’s are not clocked

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

## Synchronous Counter :

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

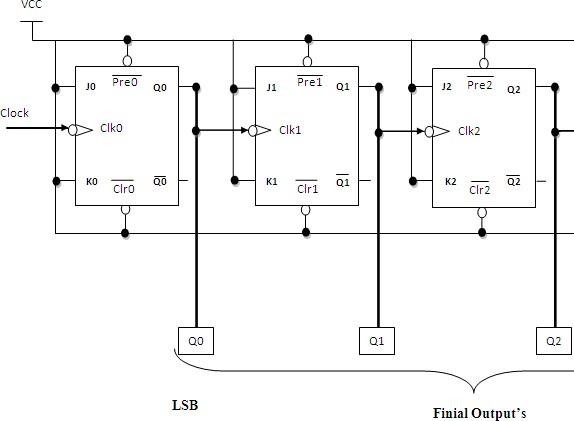
## Asynchronous Up Counter:

Fig. 1 shows 3bit Asynchronous Up Counter. Here Flip-flop 2 act as a MSB Flip-flop and Flip- flop 0 act as a LSB Flip-flop. Clock pulse is connected to the Clock of Flip-flop 0. Output of Flip-flop 0(Q0) is connected to clock of next flip-flop (i.e Flip-flop 1) and so on. As soon as clock pulse changes output is going to change (at the negative edge of clock

pulse) as a Up count sequence. For 3 bit Up counter state table is as shown below.

## State Table :

**Logic diagram :**



## Fig 1: 3 Bit Asynchronous Up Counter Hardware requirements :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Counter States | Count | | |  | | | |
| Q2 | Q1 | Q0 |
| 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 |
|  | | **Gate / Flip flop** | | | **Quantity** | **IC** | **Quantity** |
| MS JK | | | 3 | 7476 | 2 |

1. **Down Counter:**

Fig. 2 shows 2 bit Asynchronous Down Counter. Here Flip-flop 2 act as a MSB Flip-flop and Flip-flop 0 act as a LSB Flip-flop. Clock pulse is connected to the Clock of Flip-flop 0. Output of Flip-flop 0 (Q0’) 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

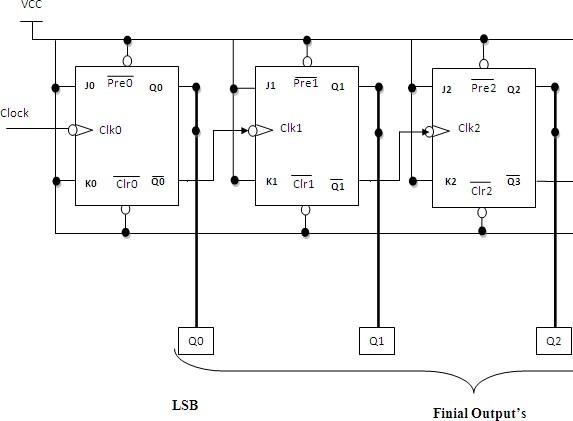
pulse) as a down count sequence. For 3 bit down counter sate table is as shown below.

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 | | |
| Q2 | Q1 | Q0 |
| 7 | 1 | 1 | 1 |
| 6 | 1 | 1 | 0 |
| 5 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 |
| 7 | 1 | 1 | 1 |

**Logic diagram :**



## Fig 2: 3 Bit Asynchronous Down Counter Hardware requirements :

|  |  |  |  |
| --- | --- | --- | --- |
| **Gate / Flip flop** | **Quantity** | **IC** | **Quantity** |

|  |  |  |  |
| --- | --- | --- | --- |
| MS JK | 3 | 7476 | 2 |

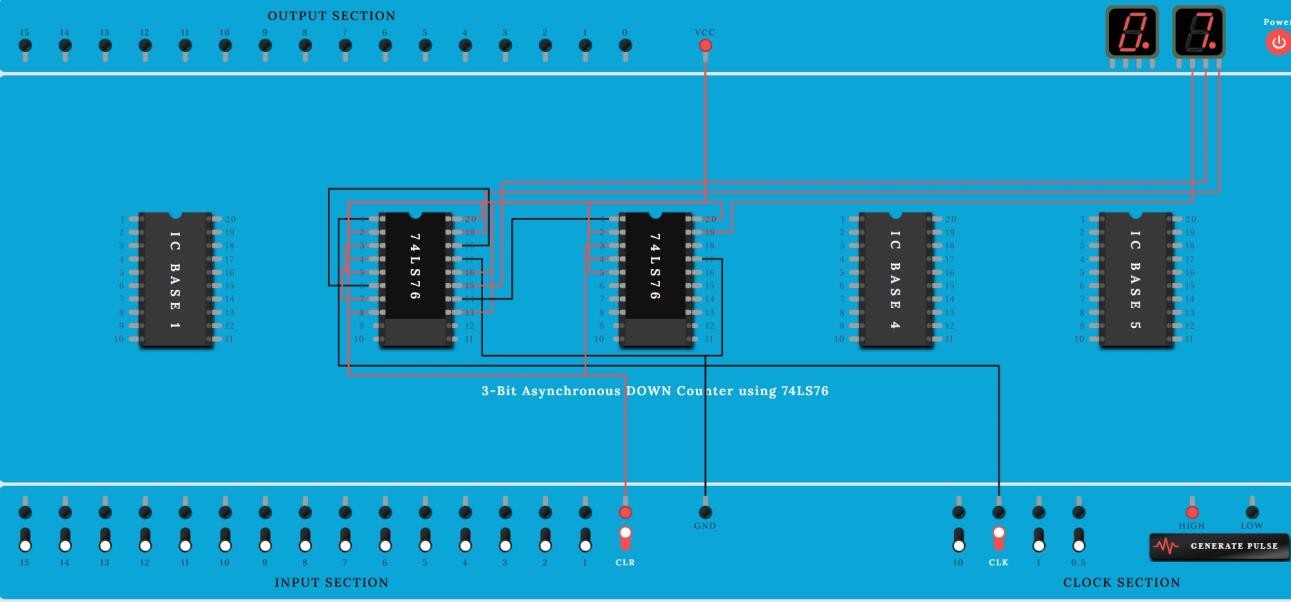
Applications **:**

The asynchronous counters are specially used as the counting devices. They are also used to count number of pulses applied.

It also works as frequency divider.

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

**Output:**



**LDCO LAB ASSIGNMENT**

**NAME :**

**ROLL NO :**

**SUBJEC :** Logic Design and Computer Organization

**BATCH :**

# Synchronous counter

## Up counter.

* 1. **Down counter.**

## 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 |

**J2 = Q1Q0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | X | X | X | X |
| **1** | 0 | 0 | 1 | 0 |

**K2 = Q1Q0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | 0 | 1 | X | X |
| **1** | 0 | 1 | X | X |

**J1 = Q0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | X | X | 1 | 0 |
| **1** | X | X | 1 | 0 |

**K1 = Q0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | 1 | X | X | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | 1 | X | X | 1 |

**J0 = 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | X | 1 | 1 | X |
| **1** | X | 1 | 1 | X |

**K0 = 1**

## Logic Diagram:

**Fig 1: 3 bit Synchronous up counter**

## 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 |

## J2 = Q1 Q0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | X | X | X | X |
| **1** | 1 | 0 | 0 | 0 |

**K2= Q1 Q0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | 1 | 0 | X | X |
| **1** | 1 | 0 | X | X |

## J1= Q0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | X | X | 0 | 1 |
| **1** | X | X | 0 | 1 |

**K1 = Q0**

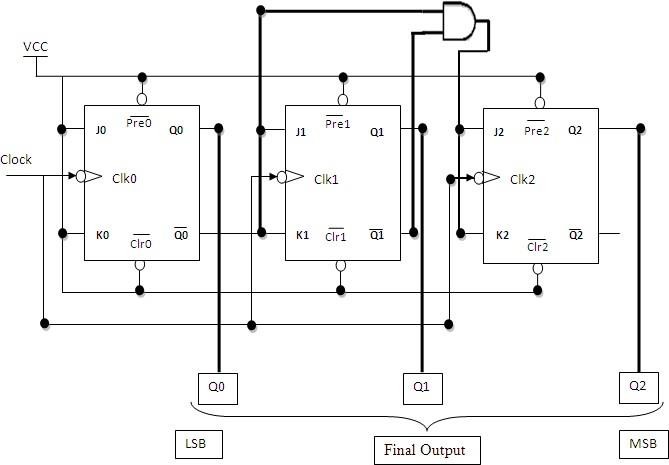
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | 1 | X | X | 1 |
| **1** | 1 | X | X | 1 |

**J0 = 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q1Q0**  **Q2** | **00** | **01** | **11** | **10** |
| **0** | X | 1 | 1 | X |
| **1** | X | 1 | 1 | X |

**K0 = 1**

## Logic Diagram :



**Fig 2: 3 bit Synchronous down counter**

**Uses:** 6. It helps in counting the no of product coming

1. Specially used as the counting devices. out from machinery where product is coming
2. Used in frequency divider circuit.out at equal interval of time.
3. Used in digital voltmeter.
4. Used in counter type A to D converter.
5. Used for time measurement..

## Conclusion:

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

## PRACTICE ASSIGNMENTS / EXERCISE / MODIFICATIONS:

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

## FAQ’s with answers:

1. What do you mean by Counter?

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

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

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

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

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

1. What are the applications of synchronous counters? Digital clock Frequency divider circuits

Frequency counters

Used in analog to digital converters

1. What are the advantages of synchronous counters over asynchronous counters? Propagation delay time is reduced.

Can operate at a much higher frequency than the asynchronous counters.

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

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

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

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

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

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

flop. Ring counters are referred as MOD ‘N’ counters.

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

1. What are the applications of ring counters?

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

1. What are the applications of ring countertwistedring counters? Used as divide by ‘2N’ ((MOD ‘2N’) counters.

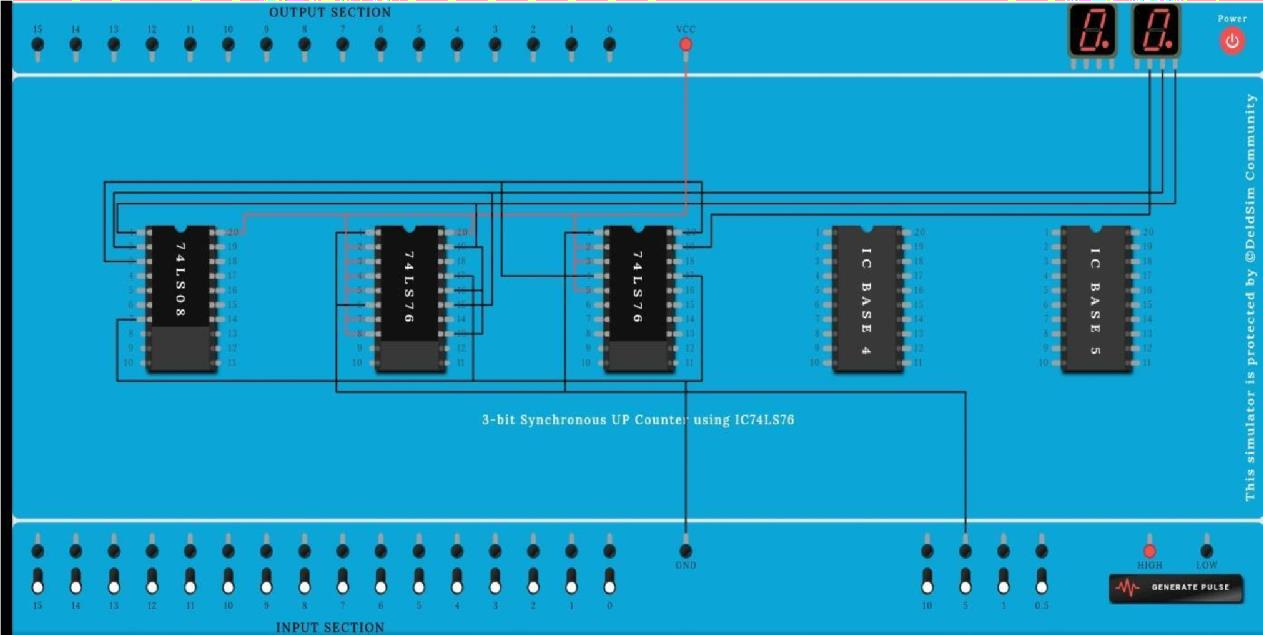
Used for control state counters.

Used for generation of multiphase clock.

1. List the Synchronous Counter ICs.

IC 74160 : Decade Up Counter

|  |  |
| --- | --- |
| IC 74161 | : 4 bit binary Up Counter |
| IC 74162 | : Decade Up Counter |
| IC 74163 | : 4 bit binary Up Counter |
| IC 74168 | : Decade Up/Down Counter |
| IC 74169 | : 4 bit Binary Up/Down Counter |
| IC 74190 | : Decade Up/Down Counter |
| IC 74191 | : 4 bit Binary Up/Down Counter |
| IC 74192 | : Decade Up/Down Counter |
| IC 74193 | : 4 bit Binary Up/Down Counter |



**LDCO LAB ASSIGNMENT**

**NAME :**

**ROLL NO :**

**SUBJEC :** Logic Design and Computer Organization

**BATCH :**

# 7. Modulus n Counter

**AIM :** To design and implement mod- 10, mod – 7, Mod-100 asynchronous BCD counter using IC 7490.

**OBJECTIVE:**To know Modules counter as well as binary &BCD Counter **IC’s USED:**IC 7490.

## THEORY: Part A – 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.

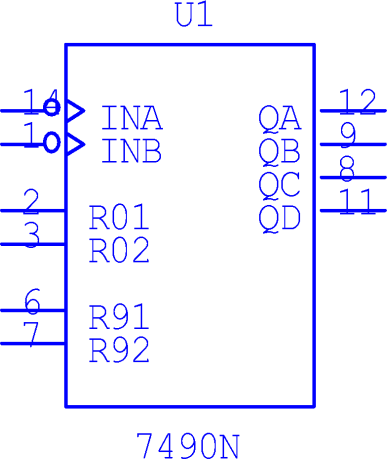
It is a 4-bit ripple type decade counter. The device consists of 4-master slave flip flops internally connected to provide a divide by two and divide by 5 sections. Each section has a separate clock i/p to initiate state changes of the counter on the high to low clock transition.

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 CP1 input must be externally connected to QA o/p. The CP0 i/p receives the incoming count producing a BCD count sequence. It is also provided with additional gating to provide a divide by 2 counter and binary counter for which the count cycle length is divide by 5. The device may be operated

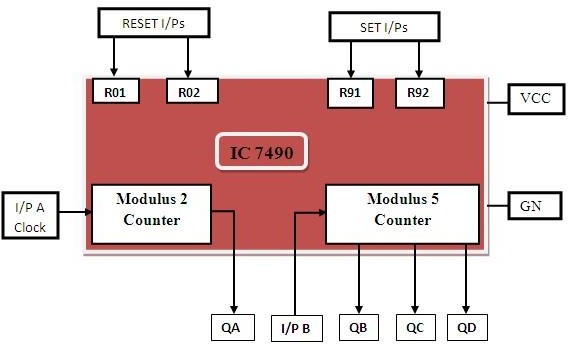
in various counting modes.

There are 2 reset inputs R0(1) and R0(2) both of which need to be connected to the ‘logic 1’ for clearing all flip flops. Two set inputs Rg(1) and Rg(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 |

**Function Table of MOD-5 counter:**

|  |  |  |
| --- | --- | --- |
| InputB clock | Output | Coun |

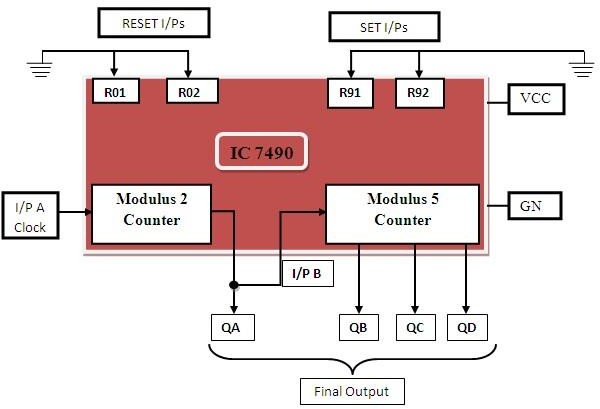
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Q D** | **Q C** | **Q B** | t |
|  | **0** | **0** | **0** | **0** |
|  | **0** | **0** | **1** | **1** |
|  | **0** | **1** | **0** | **2** |
|  | **0** | **1** | **1** | **3** |
|  | **1** | **0** | **0** | **4** |

## Design of MOD-10 counter using IC 7490:

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

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

## Logic DiagramMOD-10 counter using IC 7490:



**Function table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| I/p clock | Output | | | | Coun t |
| **QD** | **Q C** | **Q B** | **Q A** |
|  | **0** | **0** | **0** | **0** | **0** |
|  | **0** | **0** | **0** | **1** | **1** |
|  | **0** | **0** | **1** | **0** | **2** |
|  | **0** | **0** | **1** | **1** | **3** |
|  | **0** | **1** | **0** | **0** | **4** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **0** | **1** | **5** |
|  | **0** | **1** | **1** | **0** | **6** |
|  | **0** | **1** | **1** | **1** | **7** |
|  | **1** | **0** | **0** | **0** | **8** |
|  | **1** | **0** | **0** | **1** | **9** |

## Timing diagram of mod10:

**Design of Mod-7 Counter using IC 7490:**

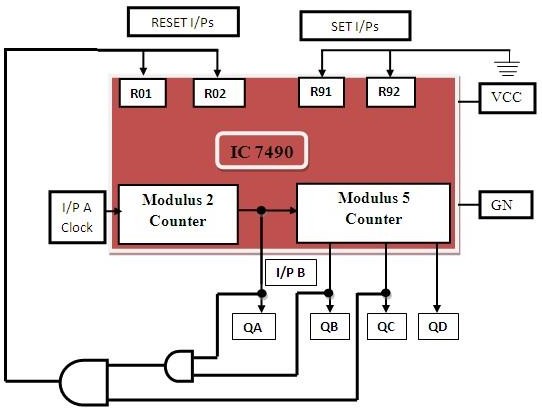
Mod-7 counter counts through seven states from 0 to 6 counters and it should reset as soon as the count becomes 7. The o/p of reset logic should be 1 corresponding to invalid states.

The reset logic o/p should be applied to pin 2 and 3.

## Truth Table of Reset Logic:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q D** | **Q C** | **Q B** | **Q A** | **Y** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |

**Logic Diagram Mod 7 Counter using IC 7490:**

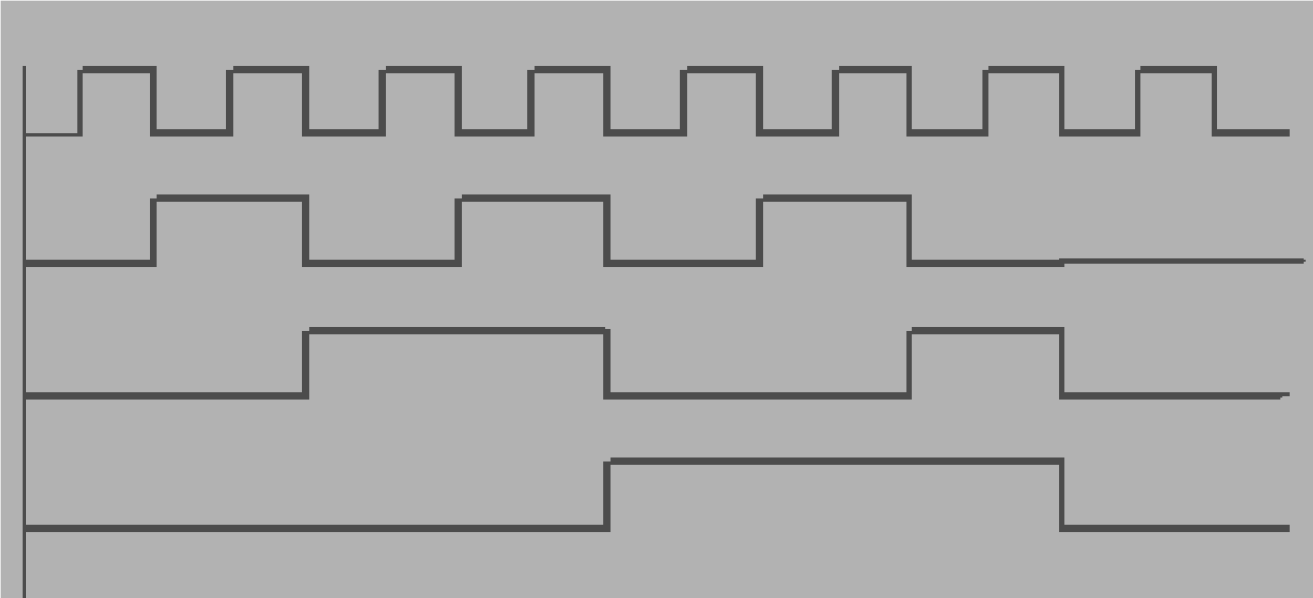


## Function table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| I/p clock | Output | | | | Coun t |
| **QD** | **Q C** | **Q B** | **Q A** |
|  | **0** | **0** | **0** | **0** | **0** |

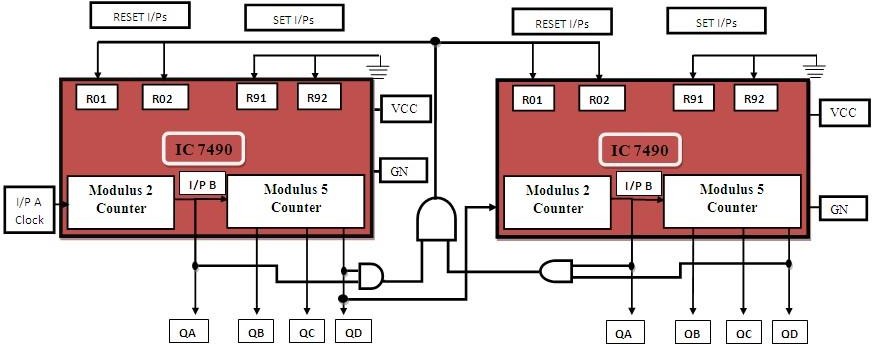
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **0** | **0** | **1** | **1** |
|  | **0** | **0** | **1** | **0** | **2** |
|  | **0** | **0** | **1** | **1** | **3** |
|  | **0** | **1** | **0** | **0** | **4** |
|  | **0** | **1** | **0** | **1** | **5** |
|  | **0** | **1** | **1** | **0** | **6** |

**Timing diagram of mod7:**



## Design of Mod-99 using IC 7490:

For Mod-99 two IC 7490’s will be required. Hence to implement a divide by 99 counter we have to use two decade counters IC’s. A divide by 99 counter counts 99 states from 0 to 98 and the counter should reset as soon as the count becomes 99. So in order to reset the counter of 99 connect the Q o/p which are equal to 1 in the count of 99 to an ‘And’ gate & then connect and o/p to the reset i/p of both IC’s.



## Conclusion:Thus we have Designed MOD -100 using IC 7490

**FAQs:**

1. What do you mean modulus counter?

It represents the number of possible states of counter.

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

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

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

