

MIT WORLD PEACE UNIVERSITY

Digital Electronics and Computer Architecture
Second Year B. Tech, Semester 3

4 BIT CODE CONVERSION BETWEEN BINARY AND
GRAY CODE USING BASIC LOGIC GATES

PRACTICAL REPORT

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 34

September 11, 2022

Contents

1 Problem Statement	1
2 ICs Used	1
3 Platform Used	1
4 Theory	1
4.1 Involved Truth Tables	1
4.1.1 Binary to Gray Code	1
4.1.2 Gray to Binary Code	2
4.1.3 XOR Gate	2
4.2 Reduced Boolean Expressions from Respective Karnaugh Maps	3
4.2.1 Karnaugh Maps for Binary to Gray	3
4.2.2 Karnaugh Maps for Gray to Binary	5
5 Design and Implementation	7
5.1 Binary to Gray Conversion Logic Gate Design	7
5.2 Gray to Binary Conversion Logic Gate Design	8
6 Circuit Diagrams	8
6.1 Pin Diagram of IC 7486	8
6.2 Circuit Diagrams for Conversions	8
7 Procedure	11
8 Conclusion	11
9 FAQs	11

1 Problem Statement

Design and Implementation of 4 Bit code convertors using Basic Logic Gates.

1. 4 Bit Binary to Gray Code
2. 4 Bit Gray to Binary Code

2 ICs Used

74LS86 (Quad 2-Input Exclusive - OR Gates)

3 Platform Used

Digital Trainer Kit

4 Theory

4.1 Involved Truth Tables

4.1.1 Binary to Gray Code

Binary Code				Gray Code			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

4.1.2 Gray to Binary Code

Gray Code				Binary Code			
G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1

4.1.3 XOR Gate

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

4.2 Reduced Boolean Expressions from Respective Karnaugh Maps

4.2.1 Karnaugh Maps for Binary to Gray

1. For Variable G_3

		B_1B_0			
		00	01	11	10
B_3B_2	00				
	01				
	11	1	1	1	1
	10	1	1	1	1

Simplification of Karnaugh Map Expression from the above K - Map :

$$G_3 = B_3 \quad (1)$$

1. For Variable G_2

		B_1B_0			
		00	01	11	10
B_3B_2	00				
	01	1	1	1	1
	11				
	10	1	1	1	1

Simplification of Karnaugh Map Expression from the above K - Map :

$$G_2 = \overline{B_3}B_2 + \overline{B_2}B_3 \quad (2)$$

$$G_2 = B_3 \oplus B_2 \quad (3)$$

1. For Variable G_1

		B_1B_0			
		00	01	11	10
B_3B_2	00			1	1
	01	1	1		
	11	1	1		
	10			1	1

Simplification of Karnaugh Map Expression from the above K - Map :

$$G_1 = \overline{B_1}B_2 + \overline{B_2}B_1 \quad (4)$$

$$G_1 = B_2 \oplus B_1 \quad (5)$$

1. For Variable G_0

		B_1B_0			
		00	01	11	10
B_3B_2	00		1		1
	01		1		1
	11		1		1
	10		1		1

Simplification of Karnaugh Map Expression from the above K - Map :

$$G_0 = \overline{B_0}B_1 + \overline{B_1}B_0 \quad (6)$$

$$G_0 = B_1 \oplus B_0 \quad (7)$$

Final Expressions for Binary to Gray

$$G_0 = B_3 \quad (8)$$

$$G_2 = B_3 \oplus B_2 \quad (9)$$

$$G_1 = B_2 \oplus B_1 \quad (10)$$

$$G_0 = B_1 \oplus B_0 \quad (11)$$

4.2.2 Karnaugh Maps for Gray to Binary

There is difference when we plot minterms for K maps here, as K Maps are inherently written in Gray code, when we write the min terms, we have to fill them in the respective Gray code > Decimal value, instead of the usual binary > Decimal value.

1. For Variable B_3

		G_1G_0			
		00	01	11	10
G_3G_2	00				
	01				
	11	1	1	1	1
	10	1	1	1	1

Simplification of Karnaugh Map Expression from the above K - Map :

$$B_3 = G_3 \quad (12)$$

1. For Variable B_2

		G_1G_0			
		00	01	11	10
G_3G_2	00				
	01	1	1	1	1
	11				
	10	1	1	1	1

Simplification of Karnaugh Map Expression from the above K - Map :

$$B_2 = \overline{G_2}G_3 + \overline{G_2}G_2B_2 = G_3 \oplus G_2 \quad (13)$$

1. For Variable B_1

		G_1G_0			
		00	01	11	10
G_3G_2	00			1	1
	01	1	1		
	11			1	1
	10	1	1		

Simplification of Karnaugh Map Expression from the above K - Map :

$$\begin{aligned}
 B_1 &= \tilde{G}_3G_2\tilde{G}_1 + G_3\tilde{G}_2\tilde{G}_1 + \tilde{G}_3\tilde{G}_2G_1 + G_3G_2G_1 \\
 &= G_3(\tilde{G}_2\tilde{G}_1 + G_2G_1) + \tilde{G}_3(G_2\tilde{G}_1 + \tilde{G}_2G_1) \\
 &= G_3(\overline{G_2\tilde{G}_1} + \tilde{G}_2G_1) + \tilde{G}_3(G_2\tilde{G}_1 + \tilde{G}_2G_1) \\
 &= G_3(\overline{G_2 \oplus G_1}) + \tilde{G}_3(\overline{G_2 \oplus G_1}) = G_3 \oplus G_2 \oplus G_1
 \end{aligned} \quad (14)$$

$$B_1 = G_3 \oplus G_2 \oplus G_1 \quad (15)$$

1. For Variable B_0

		G_1G_0			
		00	01	11	10
G_3G_2	00		1		1
	01	1		1	
	11		1		1
	10	1		1	

Simplification of Karnaugh Map Expression from the above K - Map :

$$\begin{aligned}
 B_0 &= \tilde{G}_3\tilde{G}_2\tilde{G}_0G_0 + \tilde{G}_3\tilde{G}_2G_1\tilde{G}_0 + \tilde{G}_3G_2\tilde{G}_1\tilde{G}_0 + \tilde{G}_3G_2G_1G_0 + G_3\tilde{G}_2\tilde{G}_1G_0 + G_3\tilde{G}_2G_1\tilde{G}_0 + G_3\tilde{G}_2\tilde{G}_1\tilde{G}_0 \\
 &\quad + G_3\tilde{G}_2G_1G_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0
 \end{aligned} \quad (16)$$

$$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus +G_0 \quad (17)$$

Final Expressions for Gray to Binary

$$B_3 = G_3 \quad (18)$$

$$B_2 = G_3 \oplus G_2 \quad (19)$$

$$B_1 = G_3 \oplus G_2 \oplus G_1 \quad (20)$$

$$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus +G_0 \quad (21)$$

5 Design and Implementation

1. We only need XOR gates for this so, we can just start with drawing the Inputs for Binary to Gray B0, B1, B2, B3 and Gray to Binary G0, G1, G2, G3 respectively.
2. Refer to the final Expressions obtained from the K-Maps for each conversion.
3. Join the inputs to the respective Gate Inputs
4. To make the Circuit Diagram, Replace the Gates with their respective ICs, in this case IC7486 for XOR gates, and join inputs of the IC to the Inputs respectively.
5. Write the Pin numbers of the IC after marking V_{cc} and GND .

5.1 Binary to Gray Conversion Logic Gate Design

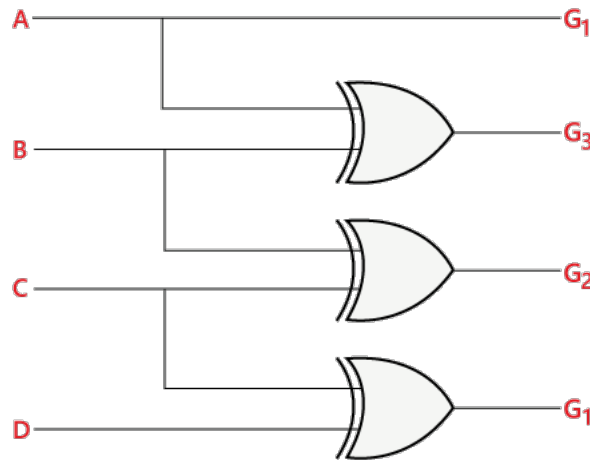


Figure 1: Binary to Gray Code conversion using Logic Gates

5.2 Gray to Binary Conversion Logic Gate Design

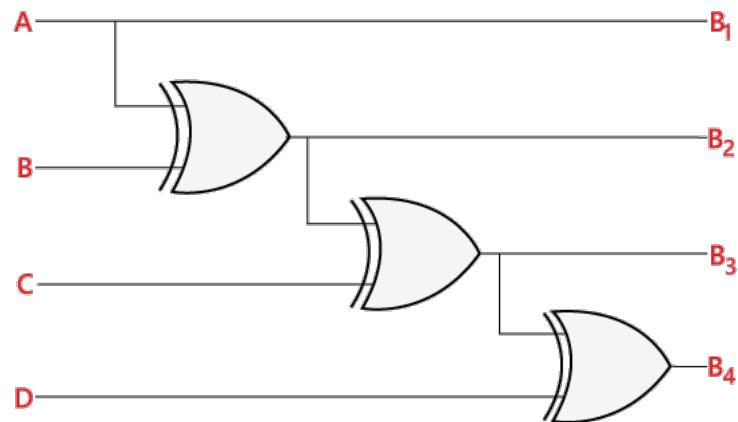
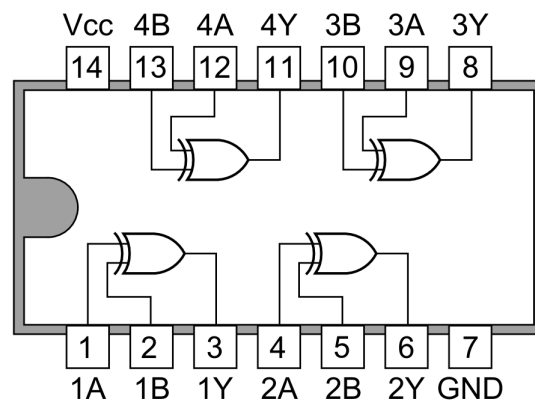


Figure 2: Gray to Binary conversion using Logic Gates

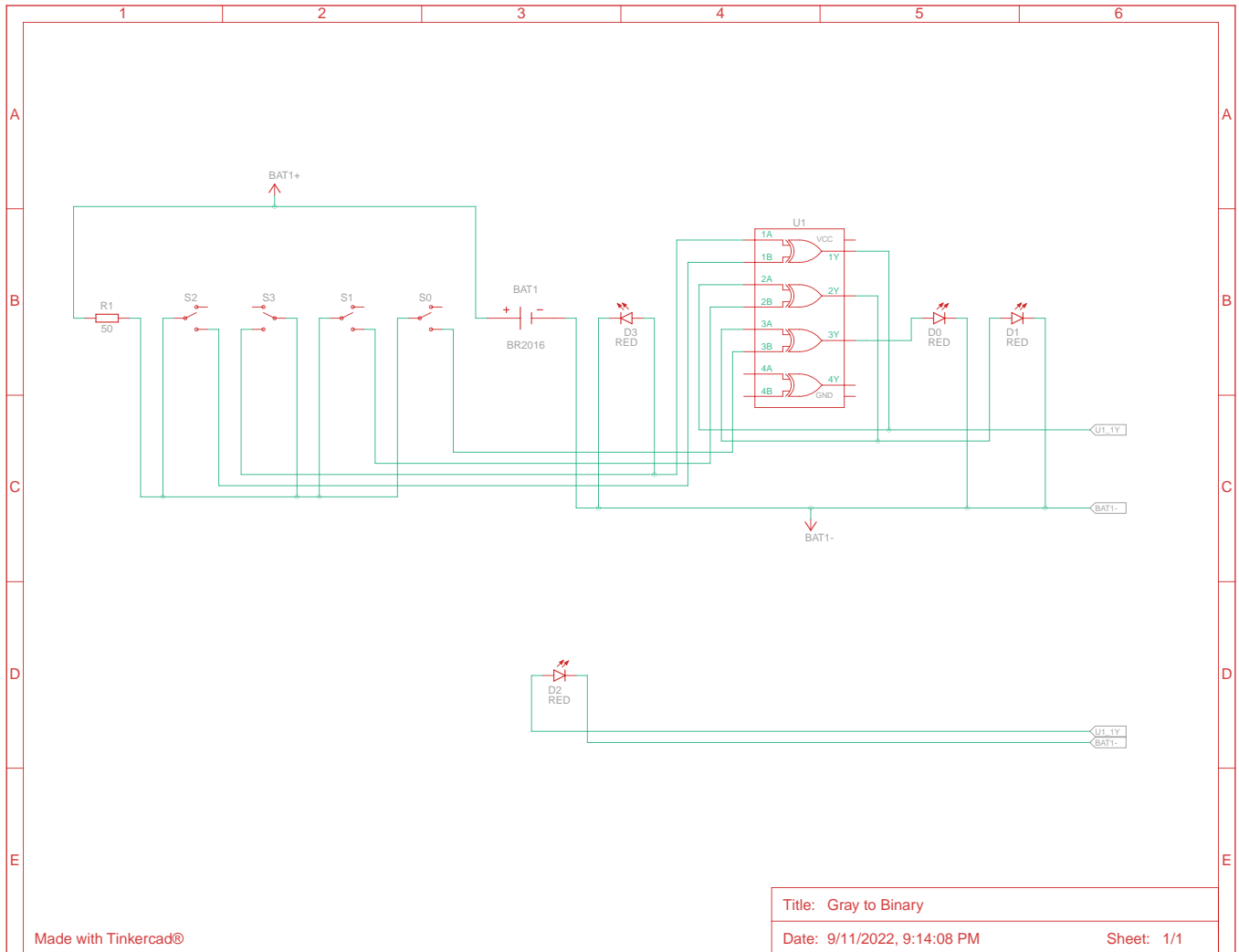
6 Circuit Diagrams

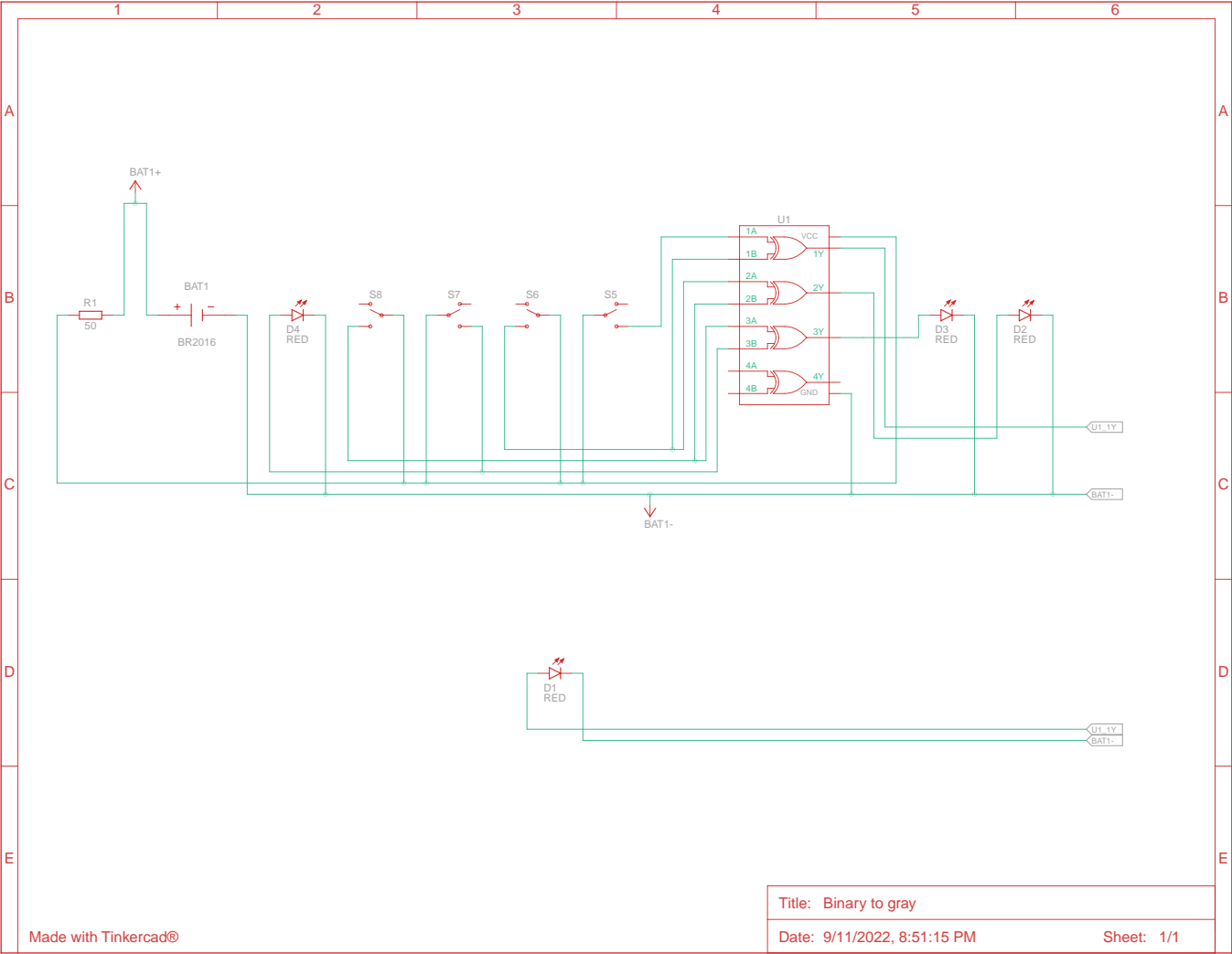
6.1 Pin Diagram of IC 7486

7486 Quad 2-input ExOR Gates



6.2 Circuit Diagrams for Conversions





7 Procedure

1. Design Combinational logic circuits as per given problem statement.
2. connect the IC 7486 and other basic logic gate ICs as per diagram.
3. Give V_{cc} supply and ground connection to each IC.
4. Give various combinations to select lines.
5. Observe the output and verify the truth table.
6. Switch off the power supply of the trainer kit.

8 Conclusion

We have learned the Implementation of Binary to Gray and Gray to Binary code converter using logic gates.

9 FAQs

1. Why is code conversion Necessary?

- (a) Code conversions are most commonly used in computers, digital electronics, and micro-processors, etc. There are numerous codes like binary, octal, hexadecimal, Binary Coded Decimal (BCD), Excess-3, Gray code, Error Correcting Codes (ECCs) and ASCII code . . .
- (b) Each Code just represents the basic numbers 1, 2, 3, 4, etc in a different Way. Each method is unique, and has some specific use.
- (c) In performing calculations and execution of certain algorithms, Certain types of codes may provide input that can be easily applied to the algorithm, as opposed to another type of code.
- (d) To Reduce the number of components used in a circuit, different types of such conversions from one code to another may be used, by applying different logic.
- (e) An Example is Gray code, which is used in shaft encoders because The code of successive numbers differs exactly by one bit from its preceeder.
- (f) Excess- 3 is extensively used for subtraction because every code in XS-3 has its complement. 1s complement of the code yields 9s complement of a number itself.
- (g) Alphanumeric codes by ASCII standards are widely used as a representation systems to the character set in computers.
- (h) BCD is used in 7 Segment Displays

2. Write any 2 Applications of Gray Code

- (a) The gray code is used in a few specific applications. The main applications include being used in analog to digital converters, as well as being used for error correction in digital communication. Gray code is used to minimize errors in converting analog signals to digital signals.
- (b) Boolean circuit minimization

- (c) Communication between clock domains
- (d) Error correction
- (e) Genetic algorithms
- (f) Mathematical puzzles
- (g) Position encoders

3. Explain Weighted and Non Weighted Codes with Examples

- (a) **Weighted Code:** The weighted codes are those that obey the position weighting principle, which states that the position of each number represents a specific weight.
- (b) Example of a Weight Code would be Binary, BCD, hex, Octal, etc where each position has a certain weight.
- (c) **Non Weighted Codes:** It is a type of code that does not have any positional weights associated with it.
- (d) Example would be Decimal, Binary, ASCII, Gray Code, etc.

MIT WORLD PEACE UNIVERSITY

Digital Electronics and Computer Architecture
Second Year B. Tech, Semester 3

DESIGN AND IMPLEMENTATION OF
COMBINATIONAL LOGIC DESIGN USING
MUX/DECODER ICs.

PRACTICAL REPORT

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 34

September 12, 2022

Contents

1 Problem Statement	1
2 ICs Used	1
3 Platform Used	1
4 Theory	1
4.1 What is a Multiplexer	1
4.1.1 Types	2
4.2 Advantages of a Multiplexer	2
4.3 Applications of a Multiplexer	2
4.4 Involved Truth Tables	3
4.4.1 OR Gate	3
4.4.2 NOT Gate	3
4.4.3 Truth Table of Half Adder	3
4.4.4 Truth Table of a Full Adder	3
4.4.5 Function Table of the IC74LS153	4
4.5 Symbols in the Pin Diagram of IC74LS153	4
4.5.1 Given SOP Function for 4 : 1 Implementation	5
4.5.2 Given POS Function for 4 : 1 Implementation	5
4.6 Given SOP Function for 8 : 1 Implementation	5
5 Pin Diagrams of ICs Used	6
5.1 Pin Diagram of IC74LS153	6
5.2 Pin Diagram of IC7432	6
5.3 Pin Diagram of IC7404	7
6 Design and Implementation	7
6.1 Logical Design for Verification of Truth Table of IC74LS153	7
6.2 Logical Design of SOP on 4: 1 Multiplexer using IC74LS153	8
6.3 Logical Design of POS on 4: 1 Multiplexer using IC74LS153	8
6.4 Logical Design of Half Adder using IC74LS153	9
6.5 Logical Design of Full Adder using IC74LS153	10
6.6 Logical Design for Implementing 8 : 1 Multiplexer using 2 4: 1 Multiplexers in IC74LS153 for Given SOP Function	11
6.7 Logical Design for Implementing 8 : 1 Multiplexer using only 1 4: 1 Multiplexers in IC74LS153 for Given SOP Function using Reduction Method	12
7 Procedure	13
8 Conclusion	13
9 FAQs	14

1 Problem Statement

Design and Implement Combinational Logic Design using MUX/Decoder ICs.

1. To Verify the Truth table of multiplexers using IC74153
2. Design and Implement two variable Functions (SOP and POS) and half adder using IC74LS153
3. Verify its Truth Table.
4. Design and implement an 8:1 Multiplexer using two 4:1 Multiplexers for a given function using IC 74LS153.

$$Y = f(A,B,C) = \sum m(0,2,4,7)$$

5. Design and implement 8:1 Multiplexer using One 4:1 Multiplexer, applying reduction method for given function using IC 74LS153

$$Y = f(A,B,C) = \sum m(0,2,4,7)$$

2 ICs Used

1. IC7404 (NOR Gate)
2. IC7432 (OR Gate)
3. IC74LS153 (Dual 4: 1 Multiplexer IC)

3 Platform Used

Digital Trainer Kit

4 Theory

4.1 What is a Multiplexer

- Multiplexer (MUX) is a network device that allows one or more analog or digital input signals to travel together over the same communications transmission link. The purpose of multiplexing is to combine and transmit signals over a single shared medium in order to optimize efficiency and decrease the total cost of communication.
- Essentially, a MUX functions as a multiple-input, single-output switch that allows multiple analog and digital input signals and to be routed through a single output line. At the receiving end, another device called a demultiplexer recovers the original individual signals.
- Multiplexers are capable of handling both analog and digital applications. In analog applications, multiplexers are made up of relays and transistor switches, whereas in digital applications, the multiplexers are built from standard logic gates. When the multiplexer is used for digital applications, it is called a digital multiplexer

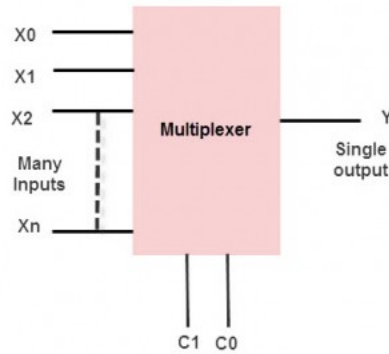


Figure 1: Multiplexer Representation

4.1.1 Types

1. 2-1 multiplexer (1 select line)
2. 4-1 multiplexer (2 select lines)
3. 8-1 multiplexer (3 select lines)
4. 16-1 multiplexer (4 select lines)

4.2 Advantages of a Multiplexer

1. In multiplexer, the usage of a number of wires can be decreased
2. It reduces the cost as well as the complexity of the circuit
3. The implementation of a number of combination circuits can be possible by using a multiplexer
4. Mux doesn't require K-maps and simplification
5. The multiplexer can make the transmission circuit less complex and economical
6. The multiplexer ability can be extended to switch audio signals, video signals, etc.
7. The digital system reliability can be improved using a MUX as it decreases the number of exterior wired connections.
8. MUX is used to implement several combinational circuits
9. The logic design can be simplified through MUX

4.3 Applications of a Multiplexer

Multiplexers are used in various applications wherein multiple-data need to be transmitted by using a single line.

1. Communication Systems

A communication system has both a communication network and a transmission system. By using a multiplexer, the efficiency of the communication system can be increased by allowing the transmission of data, such as audio and video data from different channels through single lines or cables.

2. Computer Memory

Multiplexers are used in computer memory to maintain a huge amount of memory in the computers, and also to reduce the number of copper lines required to connect the memory to other parts of the computer.

3. Telephone networks:

In telephone networks, multiple audio signals are integrated on a single line of transmission with the help of a multiplexer.

4. Transmission From the computer system of a Sattelite

The multiplexer is used to transmit the data signals from the computer system of a spacecraft or a satellite to the ground system by using a GSM satellite.

4.4 Involved Truth Tables

4.4.1 OR Gate

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

4.4.2 NOT Gate

A	Q
0	1
1	0

4.4.3 Truth Table of Half Adder

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

4.4.4 Truth Table of a Full Adder

A	B	Cin	Sum	Cout
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

4.4.5 Function Table of the IC74LS153

Select Inputs		Data Inputs				Strobe	Output
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

Figure 2: Function Table for IC 74LS153

4.5 Symbols in the Pin Diagram of IC74LS153

Pin	Symbol	Description
1	1E	enable input 1
2	S1	common data select input
3	1I3	data input from source 1
4	1I2	data input from source 1
5	1I1	data input from source 1
6	1I0	data input from source 1
7	1Y	multiplexer output from source 1
8	GND	ground
9	2Y	multiplexer output from source 2
10	2I0	data input from source 2
11	2I1	data input from source 2
12	2I2	data input from source 2
13	2I3	data input from source 2
14	S0	common data select input
15	2E	enable input 2
16	Vcc	supply voltage

4.5.1 Given SOP Function for 4 : 1 Implementation

$$Y = f(A,B,C) = \sum m(0,2)$$

Inputs	A	B	Y
I0	0	0	1
I1	0	1	0
I2	1	0	1
I3	1	1	0

4.5.2 Given POS Function for 4 : 1 Implementation

$$Y = f(A,B,C) = \prod M(1,2)$$

Inputs	B	A	Y
I0	0	0	1
I1	0	1	0
I2	1	0	0
I3	1	1	1

4.6 Given SOP Function for 8 : 1 Implementation

$$Y = f(A,B,C) = \sum m(0,2,4,7)$$

Decimal	C	B	A	Output
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

5 Pin Diagrams of ICs Used

5.1 Pin Diagram of IC74LS153

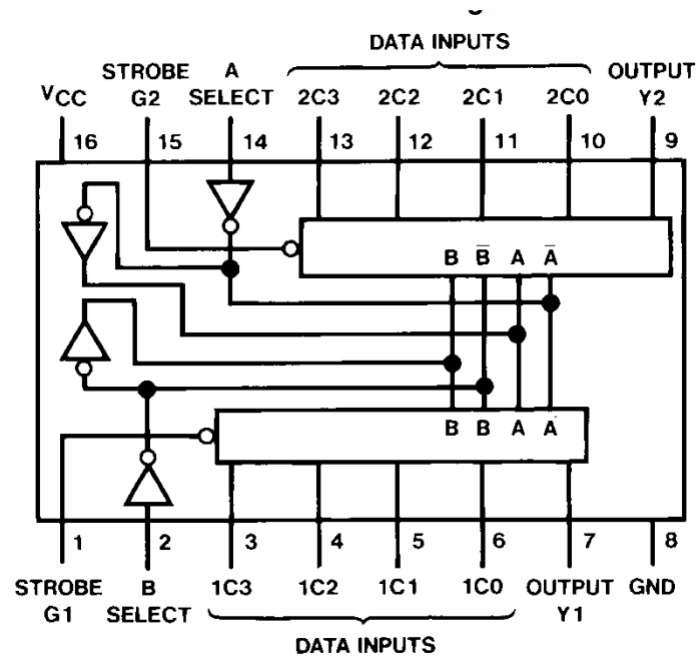


Figure 3: Pin Diagram for IC 74LS153

5.2 Pin Diagram of IC7432

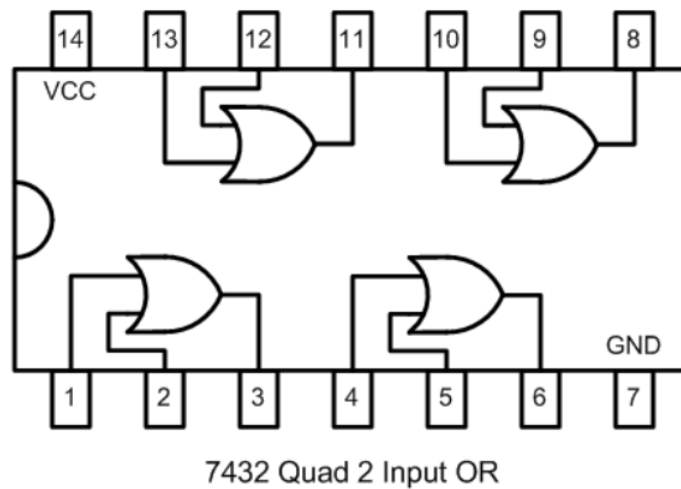


Figure 4: Pin Diagram for IC 7432

5.3 Pin Diagram of IC7404

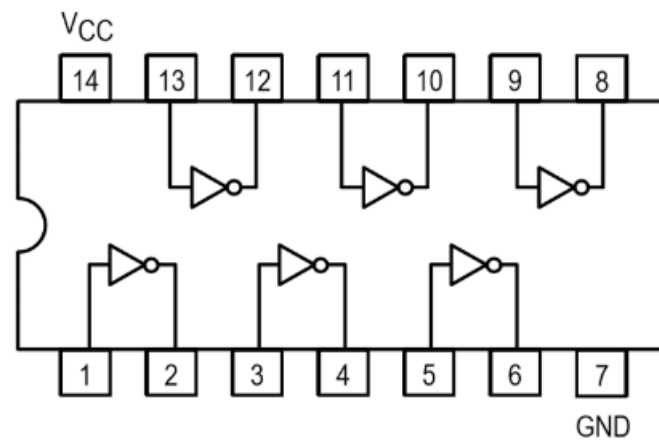


Figure 5: Pin Diagram for IC 7404

6 Design and Implementation

6.1 Logical Design for Verification of Truth Table of IC74LS153

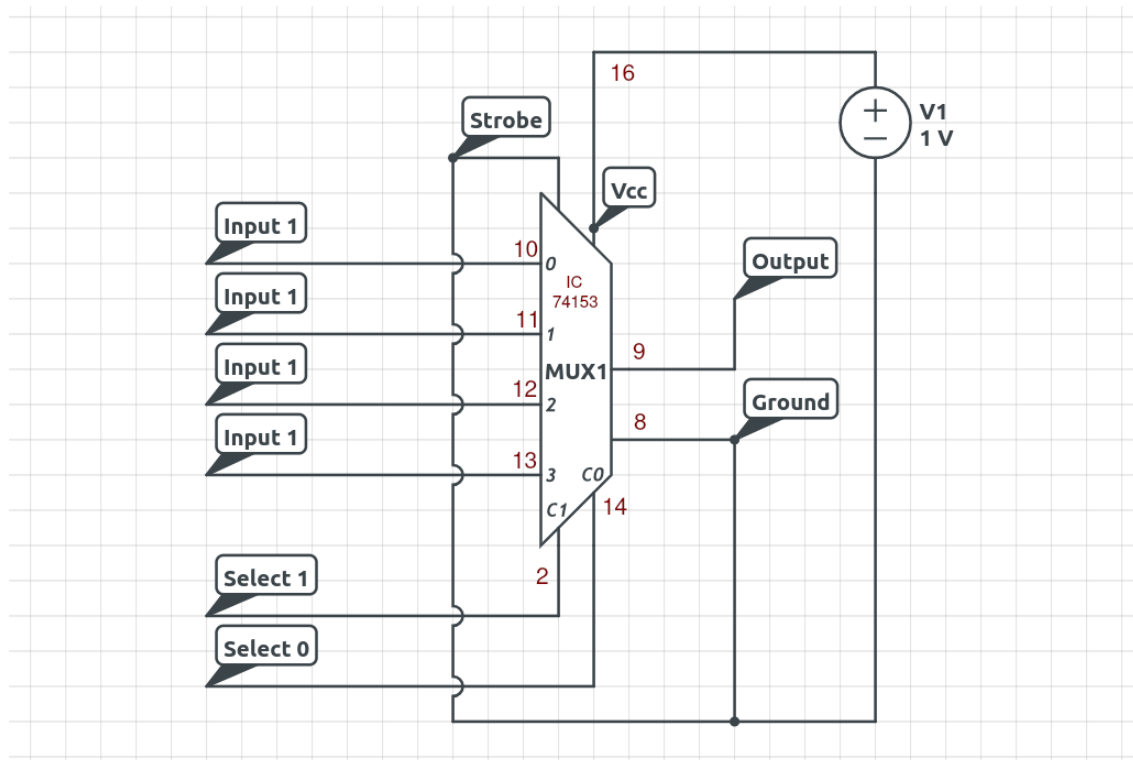


Figure 6: Truth Table for IC 74153

6.2 Logical Design of SOP on 4: 1 Multiplexer using IC74LS153

$$Y = f(A,B,C) = \sum m(0,2)$$

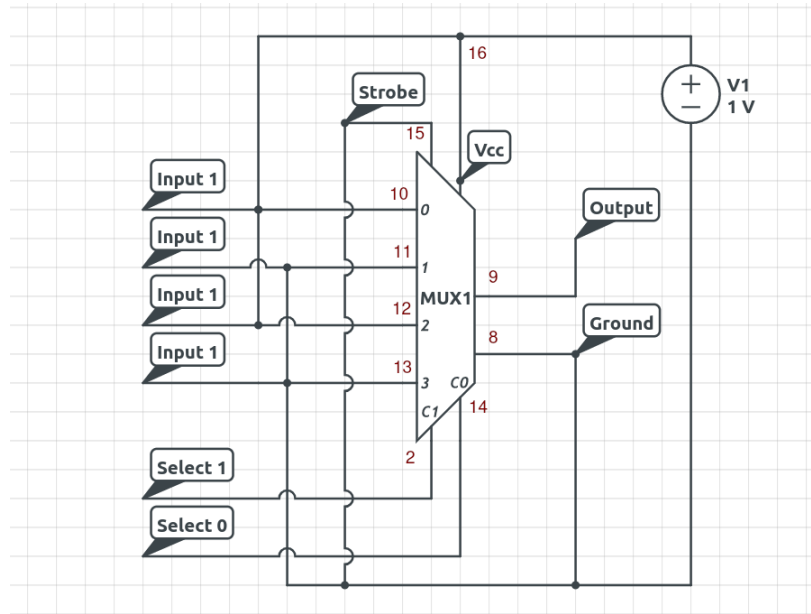


Figure 7: Diagram for Implementation of SOP

6.3 Logical Design of POS on 4: 1 Multiplexer using IC74LS153

$$Y = f(A,B,C) = \prod M(1,2)$$

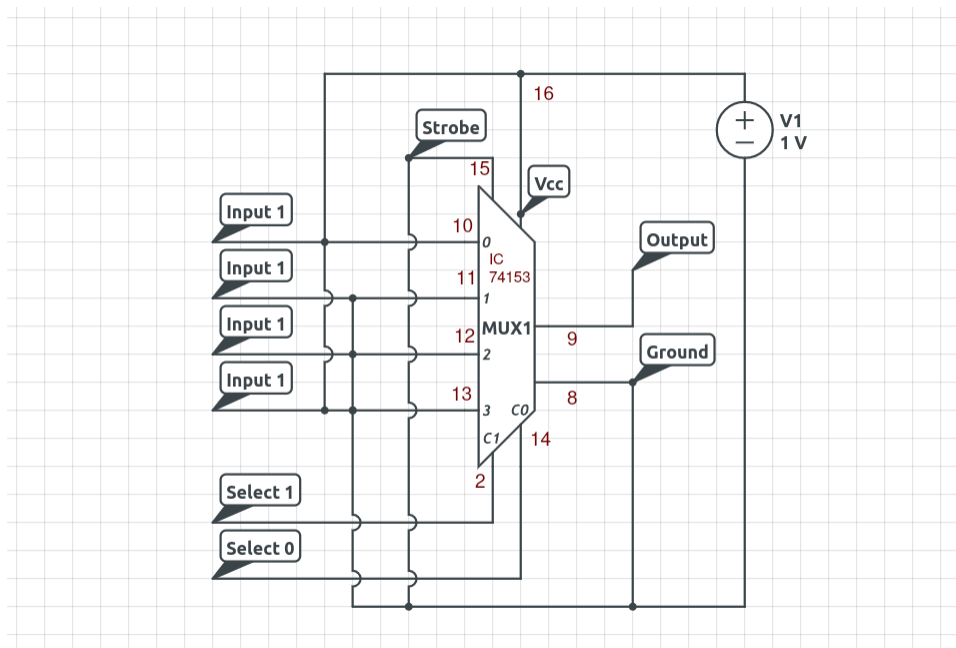


Figure 8: Diagram for Implementation of POS

6.4 Logical Design of Half Adder using IC74LS153

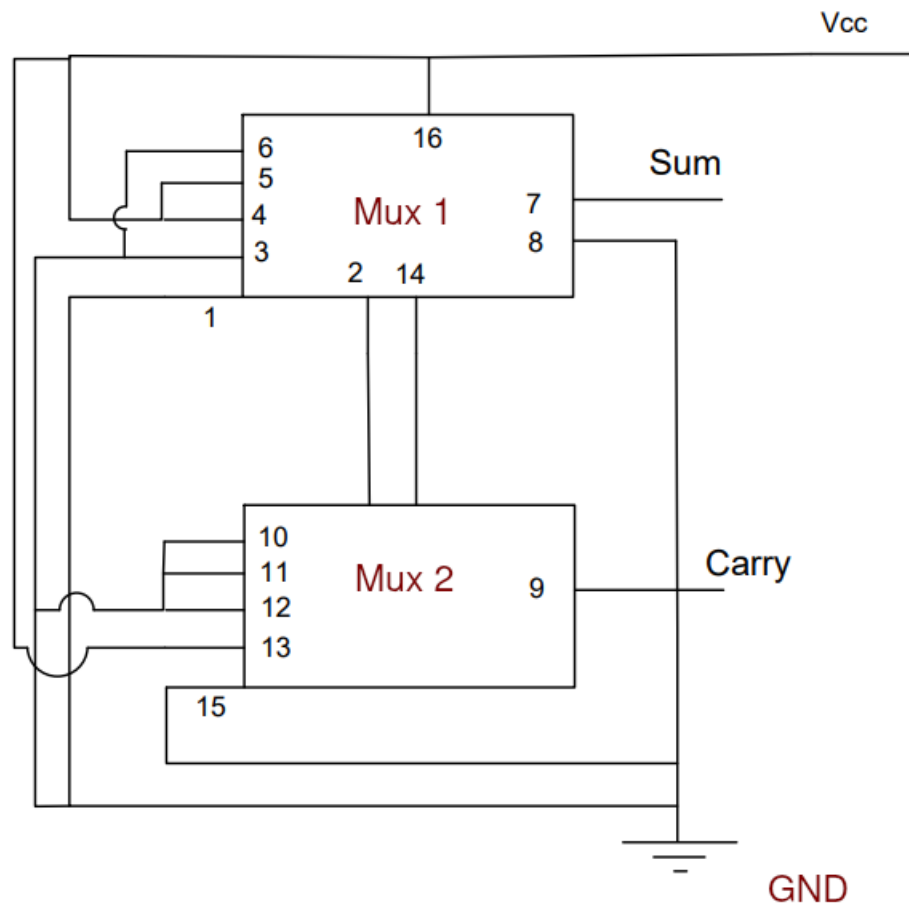


Figure 9: Half Adder Implementation using 2 4:1 Mux

6.5 Logical Design of Full Adder using IC74LS153

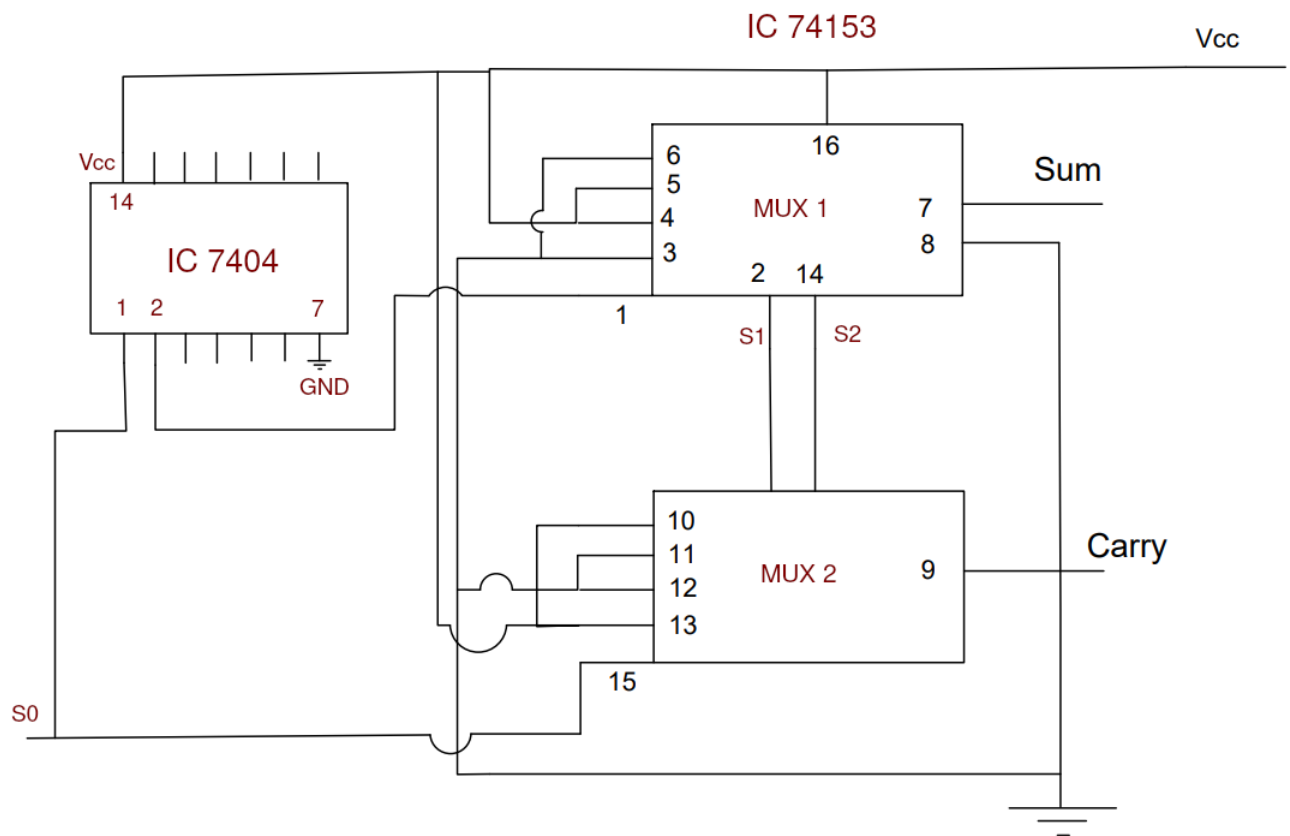


Figure 10: Full Adder Implementation using 2 4:1 Mux

6.6 Logical Design for Implementing 8 : 1 Multiplexer using 2 4: 1 Multiplexers in IC74LS153 for Given SOP Function

$$Y = f(A,B,C) = \sum m(0,2,4,7)$$

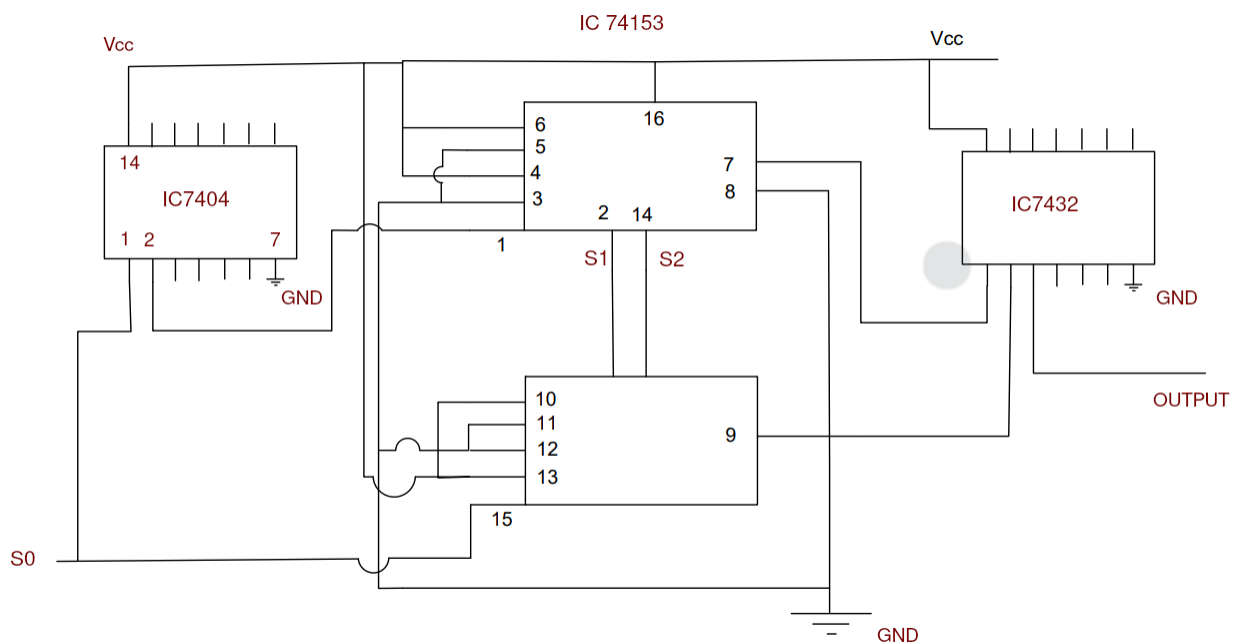


Figure 11: Given SOP Function Implementation 2 4:1 Mux

6.7 Logical Design for Implementing 8 : 1 Multiplexer using only 1 4: 1 Multiplexers in IC74LS153 for Given SOP Function using Reduction Method

$$Y = f(A,B,C) = \sum m(0,2,4,7)$$

Reduction Method

		$I_0 I_1 I_2 I_3$			
		00	01	11	10
A	0	1	0	1	0
	1	1	0	0	1

$$I_0 = V_{cc} \quad (1)$$

$$I_1 = GND \quad (2)$$

$$I_2 = \bar{A} \quad (3)$$

$$I_3 = A \quad (4)$$

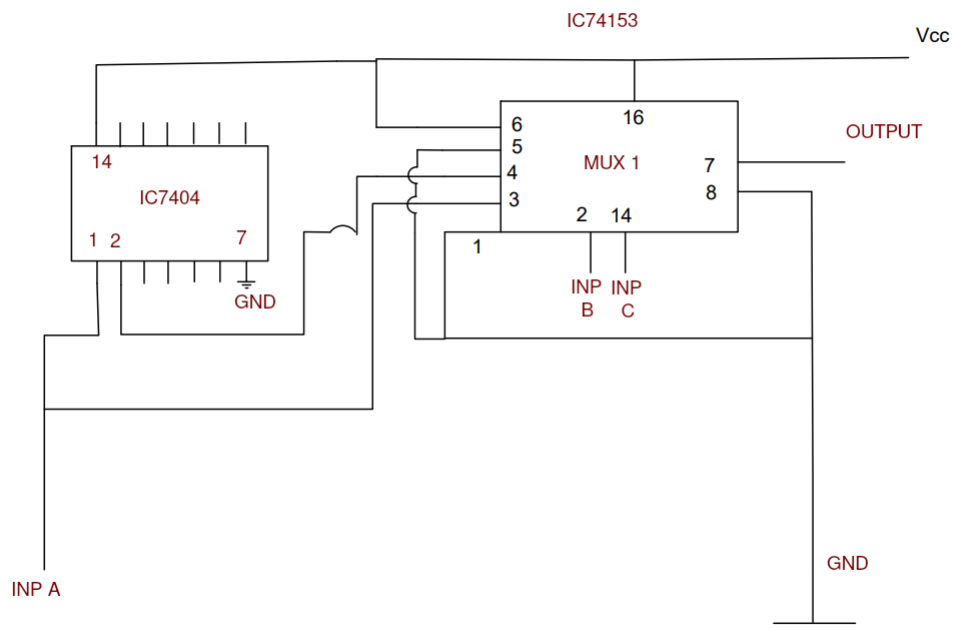


Figure 12: Given SOP Function Implementation 1 4:1 Mux

7 Procedure

1. Design Combinational logic circuit as per given problem statement.
2. Connect the IC 74153 and other basic logic gate ICs as per diagram.
3. Give Vcc supply and ground connection to each IC.
4. Give various combinations to select lines.
5. Observe the output and verify the truth table.
6. Switch off the power supply of the trainer kit.

8 Conclusion

We have learnt the application, usage and implementation of Multiplexers. SOP and POS Implementations on MUX were also understood. The Functionalities of Half Adder and Full adder were learnt in detail, and we also learnt how to apply basic logic gates in a different circuit according to its requirements.

9 FAQs

1. **What is the use of Strobe in a Multiplexer?** Strobe is the enable pin in MUX. It is the pin that enables the functioning of the MUX. Depending on it being Active Low or Active High, If we pass deactivating signal to strobe, none of the inputs are selected in the MUX. Each mux has a Strobe.
2. **Design and Implement a Full Adder using a Multiplexer** Implementation, Diagram and Truth Table are given above.

MIT WORLD PEACE UNIVERSITY

Digital Electronics and Computer Architecture
Second Year B. Tech, Semester 3

DESIGN AND IMPLEMENTATION OF
ASYNCHRONOUS COUNTERS USING
JK- FLIP FLOP.

PRACTICAL REPORT
ASSIGNMENT 3

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 34

December 2, 2022

Contents

1 Objectives	1
2 Problem Statement	1
3 ICs Used	1
4 Platform Used	1
5 Theory	1
5.1 Counters	1
5.2 Types of Counters	1
5.3 Asynchronous Counters	2
5.4 Advantages of Asynchronous Counters	2
5.5 Applications of Asynchronous Counters	2
5.6 Involved Truth Tables	3
5.6.1 NAND Gate	3
5.6.2 Truth Table of 3 Bit Asynchronous UP Counter	3
5.6.3 Truth Table of 3 Bit Asynchronous DOWN Counter	3
5.6.4 Function Table of the IC7476	4
5.7 JK Flip Flop Truth table	4
5.8 JK Flip Flop Excitation table	5
5.9 JK Flip Flop Characteristic table	5
6 Pin Diagrams of ICs Used	6
6.1 Pin Diagram of IC7476	6
6.2 Pin Diagram of IC7400	6
6.3 Timing diagram of a 3-bit asynchronous counter	7
7 Design and Implementation	7
7.1 Circuit diagram of a 3-bit asynchronous counter: (3 Bit Asynchronous - UP Counter / Modulus 8)	7
7.2 Circuit diagram of a 3-bit asynchronous counter: (3 Bit Asynchronous - DOWN Counter / Modulus 8)	8
7.3 Circuit Diagram MOD 5 (101) UP Asynchronous Counter - (Truncated Counter) using JK- Flip flop	9
8 Procedure	9
9 Conclusion	9
10 FAQs	10

1 Objectives

1. To understand the operation of Asynchronous counter
2. To design and implement MOD-N asynchronous counter using JK - Flip flop

2 Problem Statement

Design and Implement asynchronous counter using JK- Flip flop

3 ICs Used

1. IC7400 (NAND Gate)
2. IC7476 (Dual Master Slave JK Flip Flop)

4 Platform Used

Digital Trainer Kit

5 Theory

5.1 Counters

A Counter is a device which stores the number of times a particular event or process has occurred, often in relationship to a clock signal.

Counters are used in digital electronics for counting purpose, they can count specific event happening in the circuit.

For example, in UP counter a counter increases count for every rising edge of clock. Not only counting, a counter can follow the certain sequence based on our design like any random sequence 0,1,3,2 ...

They can also be designed with the help of flip flops. The main properties of a counter are timing, sequencing, and counting. Counter works in two modes

1. Up Counter
2. Down Counter

5.2 Types of Counters

Counters are broadly divided into two categories:

1. Asynchronous counter
2. Synchronous counter

5.3 Asynchronous Counters

In asynchronous counter we don't use universal clock, only first flip flop is driven by main clock and the clock input of rest of the following flip flop is driven by output of previous flip flops.

The Asynchronous counter is also known as the ripple counter. Below is a diagram of the 2-bit Asynchronous counter in which we used two T flip-flops. Apart from the T flip flop, we can also use the JK flip flop by setting both of the inputs to 1 permanently. The external clock pass to the clock input of the first flip flop, i.e., FF-A and its output, i.e., is passed to clock input of the next flip flop, i.e., FF-B.

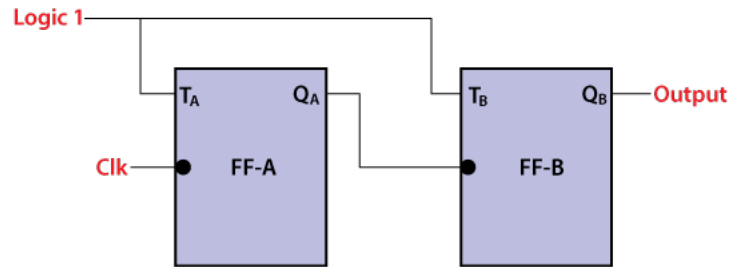


Figure 1: Asynchronous Counter

5.4 Advantages of Asynchronous Counters

1. Asynchronous counters can be easily designed by T flip flop or D flip flop.
2. These are also called as Ripple counters, and are used in low speed circuits.
3. They are used as Divide by- n counters, which divide the input by n, where n is an integer.
4. Asynchronous counters are also used as Truncated counters. These can be used to design any mod number counters, i.e. even Mod (ex: mod 4) or odd Mod (ex: mod3).

5.5 Applications of Asynchronous Counters

1. They are used as frequency dividers where the frequency of given pulse waveform is divided. Counters are sequential circuit that count the number of pulses can be either in binary code or BCD form.
2. Asynchronous counters are used as frequency dividers, as divide by N counters.
3. These are used for low power applications and low noise emission.
4. These are used in designing asynchronous decade counter.
5. Also used in Ring counter and Johnson counter.
6. Asynchronous counters are used in Mod N ripple counters. EX: Mod 3, Mod 4, Mod 8, Mod 14, Mod 10 etc.

5.6 Involved Truth Tables

5.6.1 NAND Gate

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

5.6.2 Truth Table of 3 Bit Asynchronous UP Counter





No of negative or Positive edge of Clock	Q0 LSB	Q1	Q2 MSB
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1

5.6.3 Truth Table of 3 Bit Asynchronous DOWN Counter

No of negative or Positive edge of Clock	Q0 LSB	Q1	Q2 MSB
0	0	0	0
1	1	1	1
2	0	1	1
3	1	0	1
4	0	0	1
5	1	1	0
6	0	1	0
7	1	0	0

5.6.4 Function Table of the IC7476

Function Tables:

Inputs					Outputs	
$\overline{\text{PRE}}$	$\overline{\text{CLR}}$	CLK	J	K	Q	$\overline{\text{Q}}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H [†]	H [†]
H	H		L	L	Q_0	\overline{Q}_0
H	H		H	L	H	L
H	H		L	H	L	H
H	H		H	H	Toggle	

† This configuration is nonstable; that is, it will not persist when wither preset or clear returns to its inactive (high) level.

Figure 2: Function Table for IC 7476

5.7 JK Flip Flop Truth table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	0
1	0	X	1
1	1	X	0

5.8 JK Flip Flop Excitation table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	0
1	0	X	1
1	1	X	0

5.9 JK Flip Flop Characteristic table

J	K	Q_n	Q_{n+1}	State
0	0	0	0	Q_n (Hold)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggel
1	1	1	0	

6 Pin Diagrams of ICs Used

6.1 Pin Diagram of IC7476

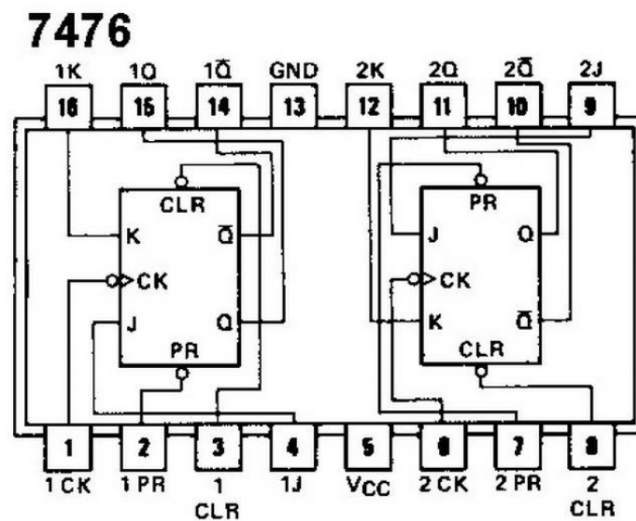


Figure 3: Pin Diagram for IC 7476

6.2 Pin Diagram of IC7400

7400 Quad 2-input NAND Gates

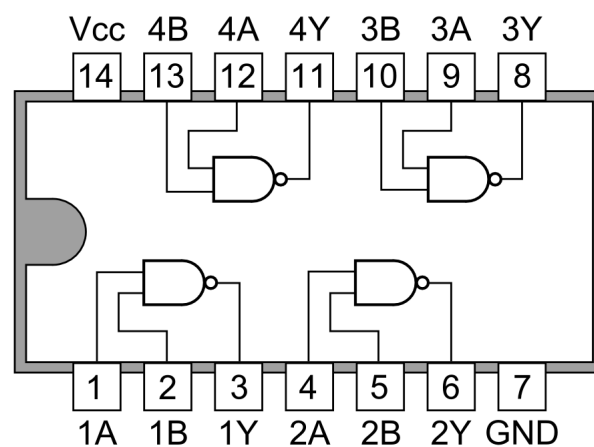


Figure 4: Pin Diagram for IC 7400

6.3 Timing diagram of a 3-bit asynchronous counter

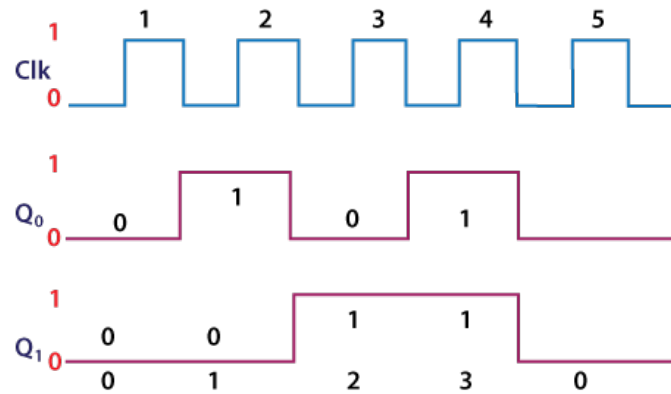


Figure 5: Timing diagram of a 3-bit asynchronous counter

7 Design and Implementation

7.1 Circuit diagram of a 3-bit asynchronous counter: (3 Bit Asynchronous - UP Counter / Modulus 8)

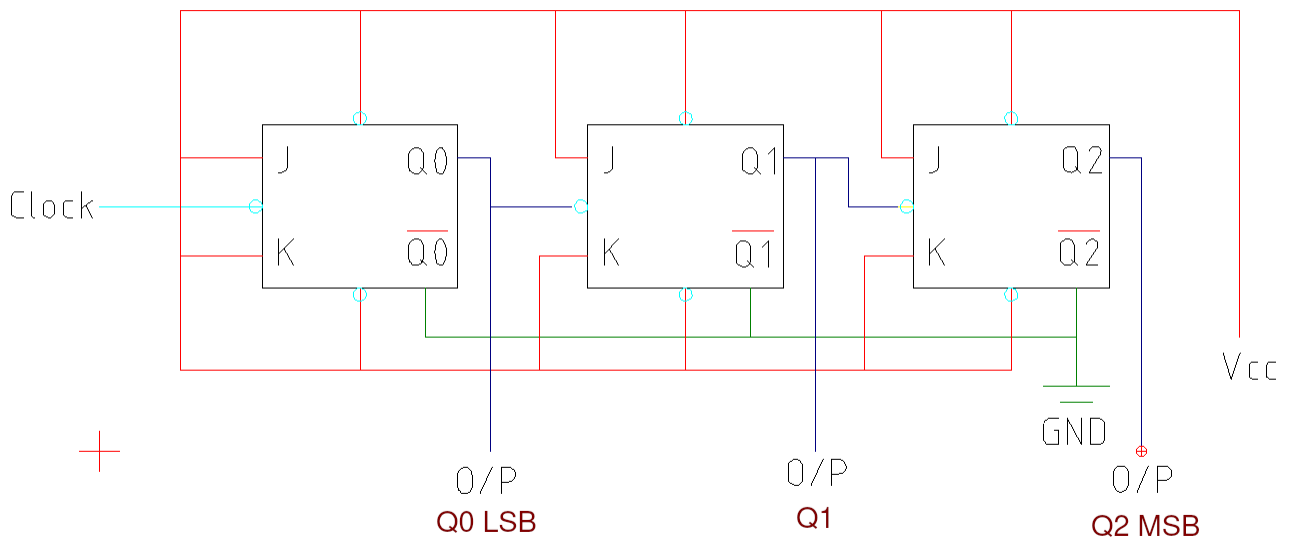


Figure 6: Circuit diagram of a 3-bit asynchronous counter - UP

7.2 Circuit diagram of a 3-bit asynchronous counter: (3 Bit Asynchronous - DOWN Counter / Modulus 8)

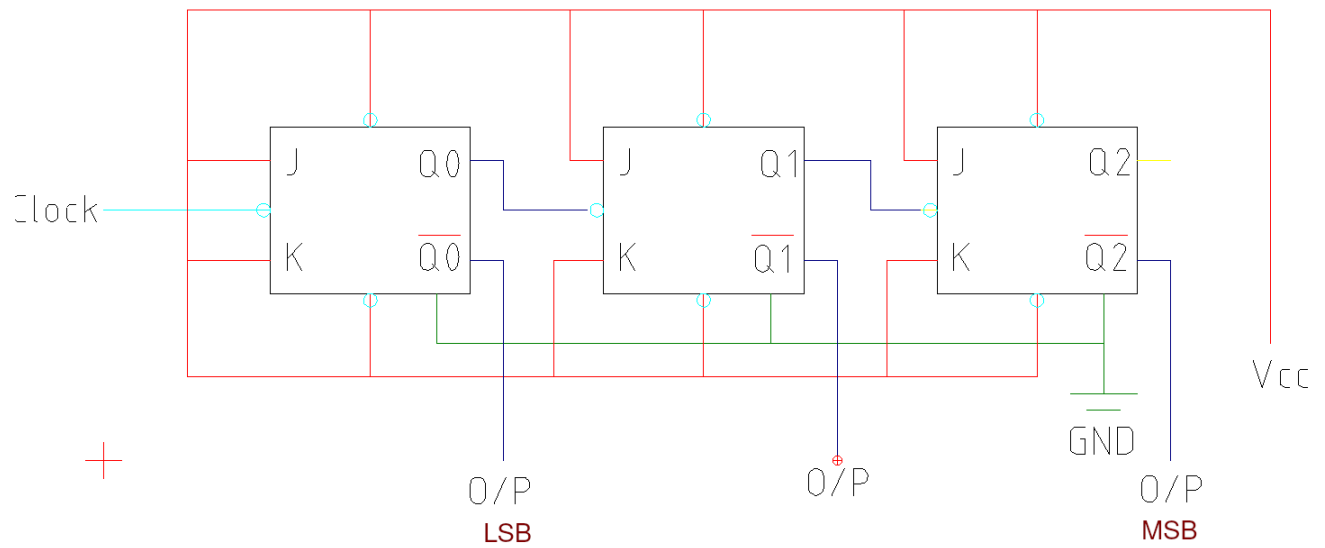


Figure 7: Circuit diagram of a 3-bit asynchronous counter - Down

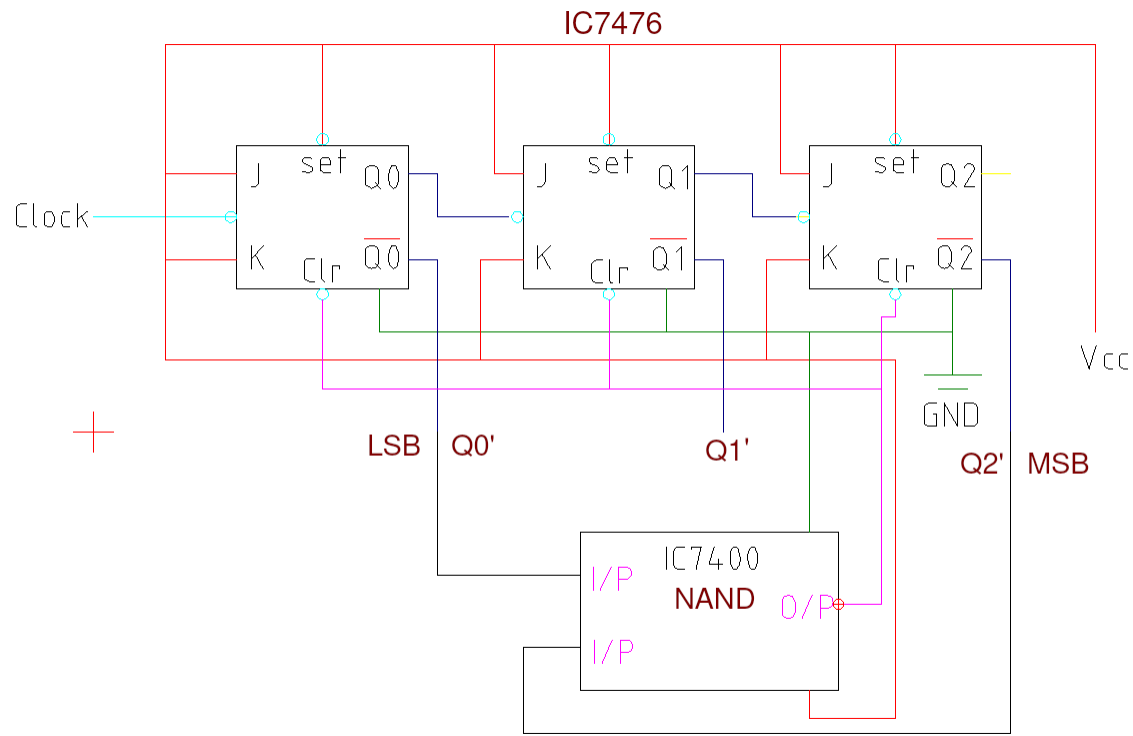


Figure 8: Circuit diagram of a 3-bit Mod Counter

8 Procedure

1. Design Sequential circuit logic circuit as per given problem statement.
2. Connect the IC 74LS76 and other basic logic gate ICs as per diagram.
3. Give VCC supply and ground connection to each IC.
4. Give clock to first JK FF.
5. Observe the output and verify the truth table.
6. Switch off the power supply of trainer kit.

9 Conclusion

Thus, we have learnt a fundamental application and working of the IC 7476, and verified the truth table of its dual Master Slave JK Flip Flops. The Logic of Flip Flops was understood in detail, and implemented on the Digital Trainer Kit. Asynchronous counters were also implemented with MOD 8 and MOD 5 as 2 Separate Circuits on the Trainer kit. Their results were observed, noted and understood.

10 FAQs

1. One of the major drawbacks to the use of asynchronous counters is?

While counting large number of bits, the propagation delay of asynchronous counters is very large. For high clock frequencies, counting errors may occur, due to propagation delay.

2. How many flip-flops are required to construct a decade counter?

In Binary we can count to 10 in just 4 bits, and so 4 Bit Asynchronous counter is to be implemented. That means we would require 4 Flip Flops, 1 for each bit.

3. The terminal count of a typical modulus 10 binary counter is?

The Terminal Count of a Typical Mod 10 Binary counter is 9, which would be represented as 1001

MIT WORLD PEACE UNIVERSITY

Digital Electronics and Computer Architecture
Second Year B. Tech, Semester 3

DESIGN AND IMPLEMENTATION OF
ASYNCHRONOUS COUNTERS USING
JK- FLIP FLOP.

PRACTICAL REPORT
ASSIGNMENT 4

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

December 2, 2022

Contents

1 Objectives	1
2 Problem Statement	1
3 ICs Used	1
4 Platform Used	1
5 Theory	1
5.1 IC 7490	1
5.2 Ripple counter IC-7490 (decade counter)	1
5.3 Basic Structure of IC 7490	1
6 Involved Truth Tables	2
6.1 NAND Gate	2
6.2 Function table of IC7490	2
6.3 JK Flip Flop Truth table	2
6.4 JK Flip Flop Excitation table	3
6.5 JK Flip Flop Characteristic table	3
7 Pin Diagrams of ICs Used	4
7.1 Pin Diagram of IC7490	4
7.2 Pin Diagram of IC7400	4
8 Design and Implementation	5
8.1 Circuit diagram of a Mod 10 Counter	5
8.2 Circuit diagram of a Mod 6 Counter	5
8.3 Circuit diagram Mod 100 Counter	6
9 Conclusion	6
10 FAQs	7

1 Objectives

1. To understand working of Modulo N counter using 7490(N>10).

2 Problem Statement

Design and implement MOD 10, MOD 100 and MOD N counter using IC 7490.

3 ICs Used

1. IC7490
2. IC 7400

4 Platform Used

Digital Trainer Kit

5 Theory

5.1 IC 7490

The IC 7490 is a decade counter. It is a 16 pin DIP IC. It has 4 inputs and 4 outputs. It has a carry output and a carry input. It has a clock input and a reset input. It has a preset input. It has a master reset

5.2 Ripple counter IC-7490 (decade counter)

IC 7490 is a 4 bit ripple type decade counter. It consists of four master or slave flip flops, which are internally connected to form a divide by two section and a divide by five section. Each section has a separate clock input to change the output states of the counter on a high to low clock transition.

5.3 Basic Structure of IC 7490

IC 7490 is a 4-bit, ripple-type decade counter. It consists of four master/slave flip-flops, which are internally connected to form a divide-by-two section and a divide-by-five section.

Each section has a separate clock input to change the output states of the counter on a high-to-low clock transition. The output states do not change simultaneously due to the internal ripple delays. Therefore the decoded output signals are subject to decoding spikes and should not be used for clocks or strobes.

Since the output of the divide-by-two section is not internally connected to the succeeding stages, IC 7490 can be operated in various counting modes.

6 Involved Truth Tables

6.1 NAND Gate

A	B	Result
0	0	1
0	1	1
1	0	1
1	1	0

6.2 Function table of IC7490

Count	Qd	Qc	Qb	Qa
(Start) 0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
(New Cycle) 10	0	0	0	0

Figure 1: Function Table of IC 7490

6.3 JK Flip Flop Truth table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	0
1	0	X	1
1	1	X	0

6.4 JK Flip Flop Excitation table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	0
1	0	X	1
1	1	X	0

6.5 JK Flip Flop Characteristic table

J	K	Q_n	Q_{n+1}	State
0	0	0	0	Q_n (Hold)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggel
1	1	1	0	

7 Pin Diagrams of ICs Used

7.1 Pin Diagram of IC7490

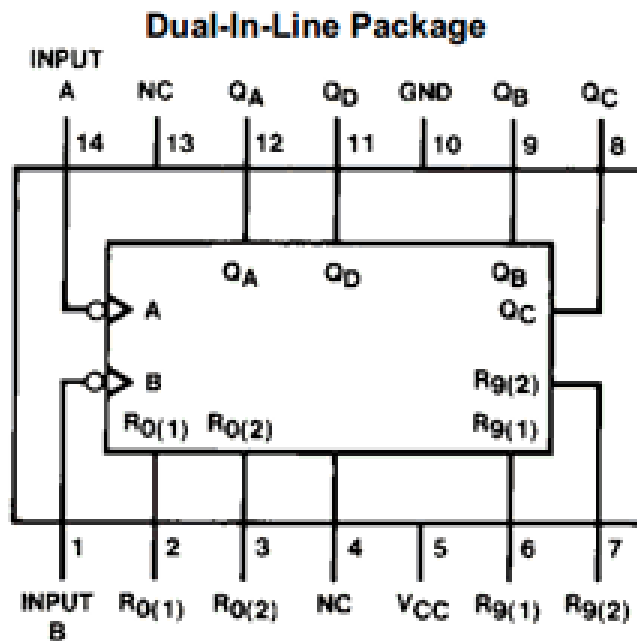


Figure 2: Pin Diagram for IC 7490

7.2 Pin Diagram of IC7400

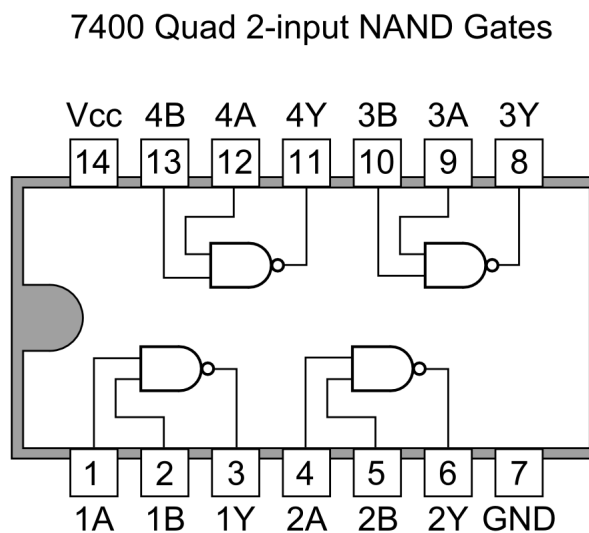


Figure 3: Pin Diagram for IC 7400

8 Design and Implementation

8.1 Circuit diagram of a Mod 10 Counter

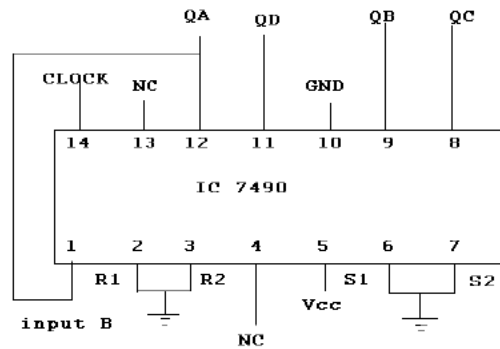


Figure 4: Circuit diagram of a Mod 10 Counter

8.2 Circuit diagram of a Mod 6 Counter

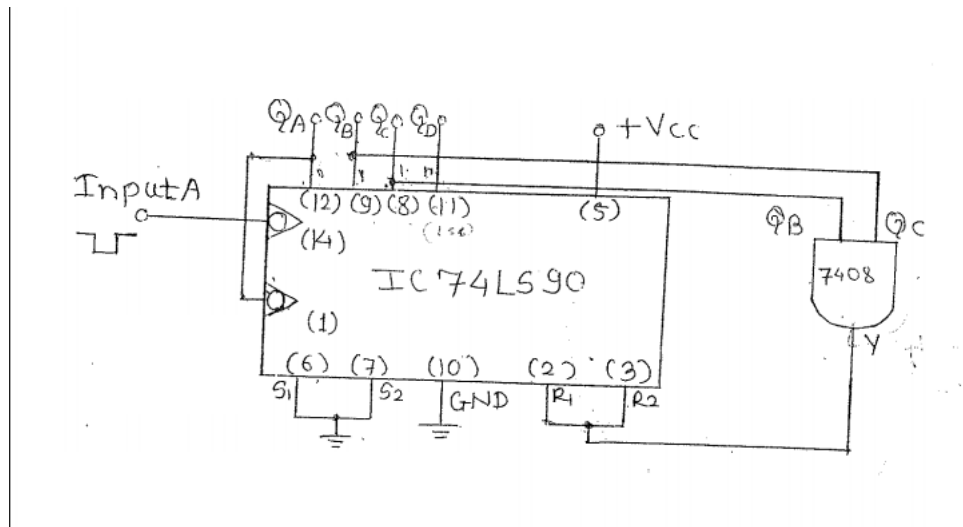


Figure 5: Circuit diagram of a Mod 6 Counter

8.3 Circuit diagram Mod 100 Counter

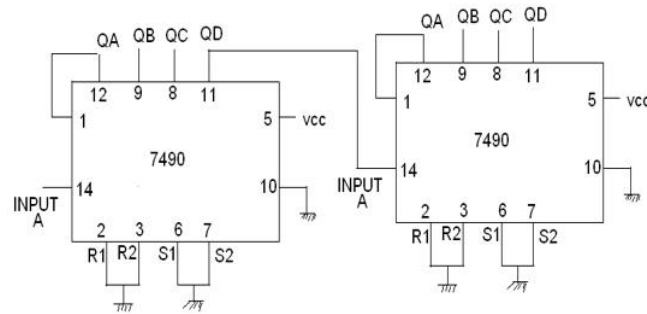


Figure 6: Circuit diagram of Mod 100 Counter

9 Conclusion

Thus, we have learnt how to implement mod 10 and Mod 100 counters. Mod N counter was also learnt and implemented. Working of IC7490 was understood in detail.

10 FAQs

1. What do you mean modulus Counter?

The modulus of a counter is the number of states it can have. For example, a 4-bit counter can have 16 states. The modulus of a 4-bit counter is 16. Counters are generally classified as either synchronous or asynchronous.

2. Application of IC 7490?

It is used in the design of asynchronous counters. It is also used in the design of asynchronous shift registers. It is also used in the design of asynchronous binary adders. It is also used in Automatic controller circuits.

MIT WORLD PEACE UNIVERSITY

Digital Electronics and Computer Architecture
Second Year B. Tech, Semester 3

3 BIT SYNCHRONOUS UP COUNTER USING JK-
FLIP FLOP

PRACTICAL REPORT
ASSIGNMENT 5

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 34

December 2, 2022

Contents

1 Objectives	1
2 Problem Statement	1
3 ICs Used	1
4 Platform Used	1
5 Theory	1
5.1 Definition of Synchronous counter	1
5.2 Advantages of Synchronous counter	1
5.3 Application of Synchronous counter	2
5.4 Involved Truth Tables	2
5.4.1 AND Gate	2
5.4.2 Truth table of 3 Bit synchronous down counter.	2
5.4.3 Function Table of the IC7476	2
5.5 JK Flip Flop Truth table	3
5.6 JK Flip Flop Excitation table	3
5.7 JK Flip Flop Characteristic table	3
5.8 State diagram of 3 Bit Synchronous Down Counter	4
6 Pin Diagrams of ICs Used	4
6.1 Pin Diagram of IC7476	4
6.2 Pin Diagram of IC7408	5
7 Design and Implementation	5
7.1 Circuit diagram of a 3-bit asynchronous counter: (3 Bit Asynchronous - DOWN Counter / Modulus 8)	5
8 Procedure	5
9 Conclusion	6
10 FAQs	7

1 Objectives

1. To understand the operation of Synchronous counter
2. To design and implement 3-Bit Synchronous UP counter using JK- Flip flop

2 Problem Statement

Design and Implement 3 bit Synchronous UP Counter using JK- Flip flop

3 ICs Used

1. IC7408 (AND Gate)
2. IC7476 (Dual Master Slave JK Flip Flop)

4 Platform Used

Digital Trainer Kit

5 Theory

5.1 Definition of Synchronous counter

Synchronous generally refers to something which is coordinated with others based on time. Synchronous signals occur at same clock rate and all the clocks follow the same reference clock.

In previous tutorial of Asynchronous Counter, we have seen that the output of that counter is directly connected to the input of next subsequent counter and making a chain system, and due to this chain system propagation delay appears during counting stage and create counting delays. In synchronous counter, the clock input across all the flip-flops use the same source and create the same clock signal at the same time. So, a counter which is using the same clock signal from the same source at the same time is called Synchronous counter.

5.2 Advantages of Synchronous counter

1. It's easier to design than the Asynchronous counter.
2. It acts simultaneously.
3. No propagation delay associated with it.
4. Count sequence is controlled using logic gates, error chances are lower.
5. Faster operation than the Asynchronous counter.

5.3 Application of Synchronous counter

1. Machine Motion control
2. Motor RPM counter
3. Rotary Shaft Encoders
4. Digital clock or pulse generators.
5. Digital Watch and Alarm systems.

5.4 Involved Truth Tables





5.4.1 AND Gate

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

5.4.2 Truth table of 3 Bit synchronous down counter.

5.4.3 Function Table of the IC7476

Function Tables:

Inputs					Outputs	
PRE	CLR	CLK	J	K	Q	\bar{Q}
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H†	H†
H	H		L	L	Q_0	\bar{Q}_0
H	H		H	L	H	L
H	H		L	H	L	H
H	H		H	H	Toggle	

† This configuration is nonstable; that is, it will not persist when wither preset or clear returns to its inactive (high) level.

Figure 1: Function Table for IC 7476

5.5 JK Flip Flop Truth table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	0
1	0	X	1
1	1	X	0

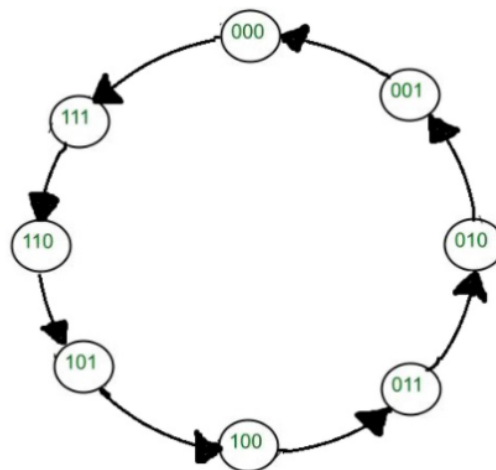
5.6 JK Flip Flop Excitation table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	0
1	0	X	1
1	1	X	0

5.7 JK Flip Flop Characteristic table

J	K	Q_n	Q_{n+1}	State
0	0	0	0	Q_n (Hold)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggel
1	1	1	0	

5.8 State diagram of 3 Bit Synchronous Down Counter



6 Pin Diagrams of ICs Used

6.1 Pin Diagram of IC7476

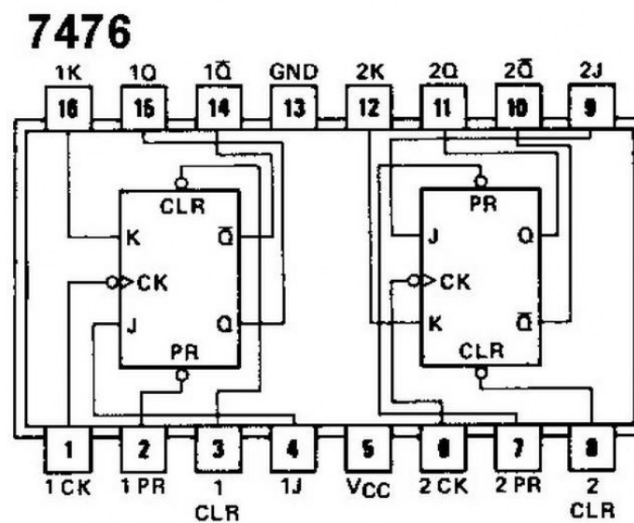


Figure 2: Pin Diagram for IC 7476

6.2 Pin Diagram of IC7408

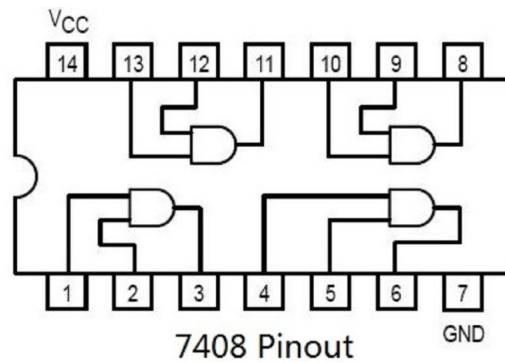


Figure 3: Pin Diagram for IC 7408

7 Design and Implementation

7.1 Circuit diagram of a 3-bit asynchronous counter: (3 Bit Asynchronous - DOWN Counter / Modulus 8)

Logic Diagram for 3-bit Down Counter:

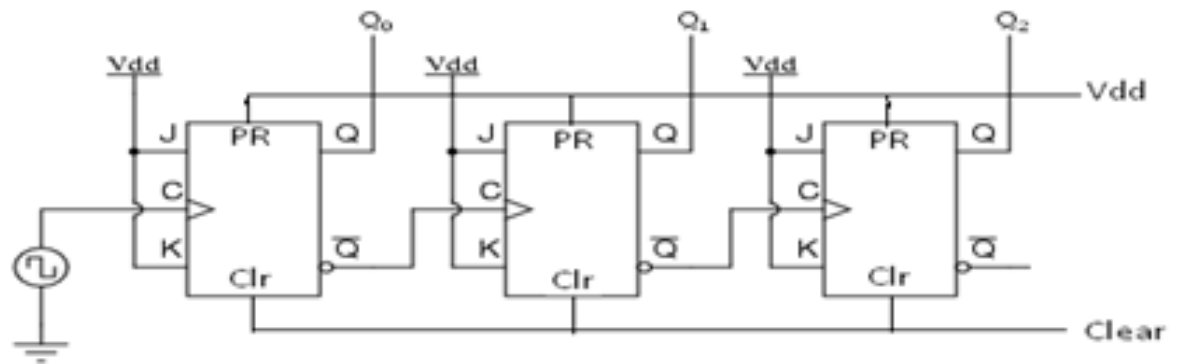


Figure 4: Circuit diagram of a 3-bit synchronous counter - Down

8 Procedure

1. Design Sequential circuit logic circuit as per given problem statement.
2. Connect the IC 74LS76 and other basic logic gate ICs as per diagram.
3. Give VCC supply and ground connection to each IC.
4. Give clock to first JK FF.

5. Observe the output and verify the truth table.
6. Switch off the power supply of trainer kit.

9 Conclusion

Thus, we have learnt a fundamental application and working of the IC 7476, and verified the truth table of its dual Master Slave JK Flip Flops. The Logic of Flip Flops was understood in detail, and implemented on the Digital Trainer Kit. Synchronous counters were also implemented with MOD 8. Their results were observed, noted and understood.

10 FAQs

1. Why a synchronous counter operate at a higher frequency than a ripple counter?

Synchronous counters can operate at a higher frequency than a ripple counter because the dual flip flops in the counter operate at the same time, therefore reducing the time by a huge margin, and subsequently increasing the frequency.

2. A 5-bit synchronous binary counter is made up of five flip-flops, each with a 12 ns propagation delay. The total propagation delay ($t_{p(\text{total})}$) is ?

Each bit has propagation delay = 12ns. So, 5 bits = $12\text{ns} * 5 = 60\text{ns}$. Since a counter is constructed using flip-flops, therefore, the propagation delay in the counter occurs only due to the flip-flops. Each bit has propagation delay = 15ns.

MIT WORLD PEACE UNIVERSITY

Digital Electronics and Computer Architecture
Second Year B. Tech, Semester 3

DESIGN A SEQUENCE DETECTOR TO DETECT THE
BIT SEQUENCE 110 USING MEALY MACHINE.

PRACTICAL REPORT
ASSIGNMENT 6

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

December 2, 2022

Contents

1 Objectives	1
2 Problem Statement	1
3 ICs Used	1
4 Platform Used	1
5 Theory	1
5.1 Application of state machine	1
5.2 Comparison Mealy Machine with Moore Machine	1
5.2.1 Mealy Machine	1
5.2.2 Moore Machine	2
5.2.3 Sequence detector circuit	2
5.3 Involved Truth Tables	2
5.3.1 AND Gate	2
5.3.2 Function Table of the IC7476	2
6 Pin Diagrams of ICs Used	3
6.1 Pin Diagram of IC7476	3
6.2 Pin Diagram of IC7408	3
7 Design and Implementation	4
7.1 State Diagram	4
7.2 State Transition Table	4
7.3 State Assignment	4
8 K-Maps	5
9 Circuit Diagram	6
10 Procedure	6
11 Conclusion	7
12 FAQs	8

1 Objectives

1. To learn Finite State Machine
2. To understand Moore and Mealy Machine
3. To learn and implement Sequence Detector

2 Problem Statement

Design a sequence detector to detect the bit sequence 110 using Mealy Machine.

3 ICs Used

1. IC7408 (AND Gate)
2. IC7476 (Dual Master Slave JK Flip Flop)

4 Platform Used

Digital Trainer Kit

5 Theory

5.1 Application of state machine

A state machine is any device storing the status of something at a given time. The status changes based on inputs, providing the resulting output for the implemented changes. A finite state machine has finite internal memory.

They are used in applications where distinguishable states exist. Each state can lead to one or multiple states and can also end the process flow. A state machine relies on user input and its original state calculation to determine to which state to go to next.

5.2 Comparison Mealy Machine with Moore Machine

5.2.1 Mealy Machine

1. Output depends on the present state as well as present input.
2. Mealy machine places its output on transition.
3. Less number of states are required.
4. Asynchronous output generation.

5.2.2 Moore Machine

1. Output depends only upon the present state.
2. Moore machine also places its outputs on transition.
3. Moore states are required.
4. Synchronous output and state generation.

5.2.3 Sequence detector circuit

A sequence detector is a sequential circuit that outputs certain data when a particular pattern of bits sequentially arrives at its data input.





5.3 Involved Truth Tables

5.3.1 AND Gate

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

5.3.2 Function Table of the IC7476

Function Tables:

Inputs					Outputs	
PRE	CLR	CLK	J	K	Q	\overline{Q}
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H†	H†
H	H		L	L	Q ₀	\overline{Q}_0
H	H		H	L	H	L
H	H		L	H	L	H
H	H		H	H	Toggle	

† This configuration is nonstable; that is, it will not persist when either preset or clear returns to its inactive (high) level.

Figure 1: Function Table for IC 7476

6 Pin Diagrams of ICs Used

6.1 Pin Diagram of IC7476

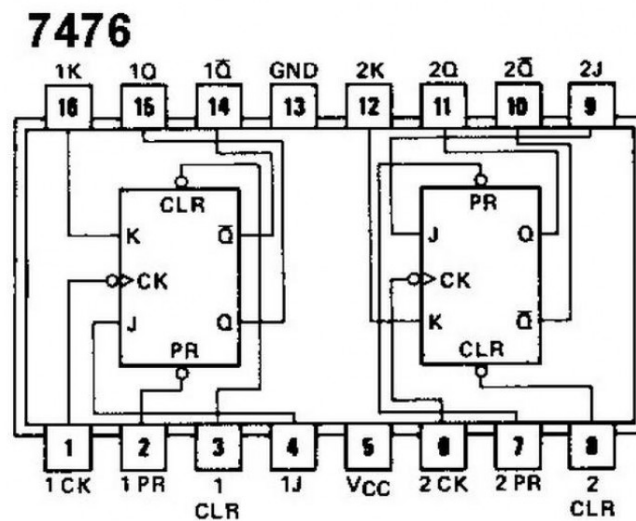


Figure 2: Pin Diagram for IC 7476

6.2 Pin Diagram of IC7408

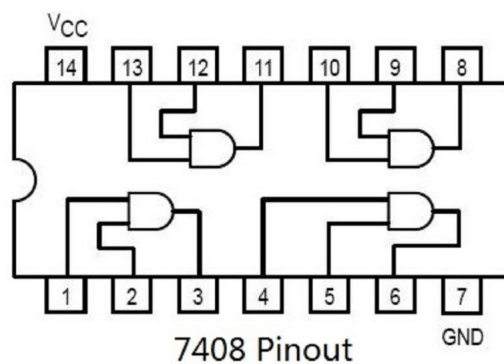


Figure 3: Pin Diagram for IC 7408

7 Design and Implementation

7.1 State Diagram

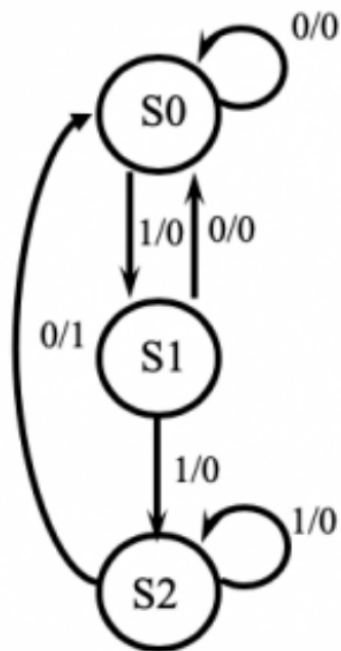


Figure 4: State Diagram

7.2 State Transition Table

Present State		Input Next State			Flip Flop Input				Output
Qa	Qb	X	Qa+1	Qb+ 1	Ja	Ka	Jb	Kb	Y
0	0	0	0	0	0	X	0	X	0
0	0	1	0	1	0	x	1	x	0
0	1	0	0	0	0	x	x	1	0
0	1	1	1	0	1	x	x	1	0
1	0	0	0	0	x	1	0	x	1
1	0	1	0	1	x	1	1	x	0
1	1	0	x	x	x	x	x	x	x
1	1	1	x	x	x	x	x	x	x

7.3 State Assignment

A = 00 B = 01 C = 10

8 K-Maps

		$Q_B Q_A$			
		00	01	11	10
1	0	0	0	1	0
	1	X	X	X	X

$$J_A = Q_{a+1}X$$

		$Q_B Q_A$			
		00	01	11	10
1	0	0	X	X	X
	1	X	X	X	X

$$K_A = 1$$

		$Q_B Q_A$			
		00	01	11	10
1	0	0	1	X	X
	1	0	1	X	X

$$J_B = X$$

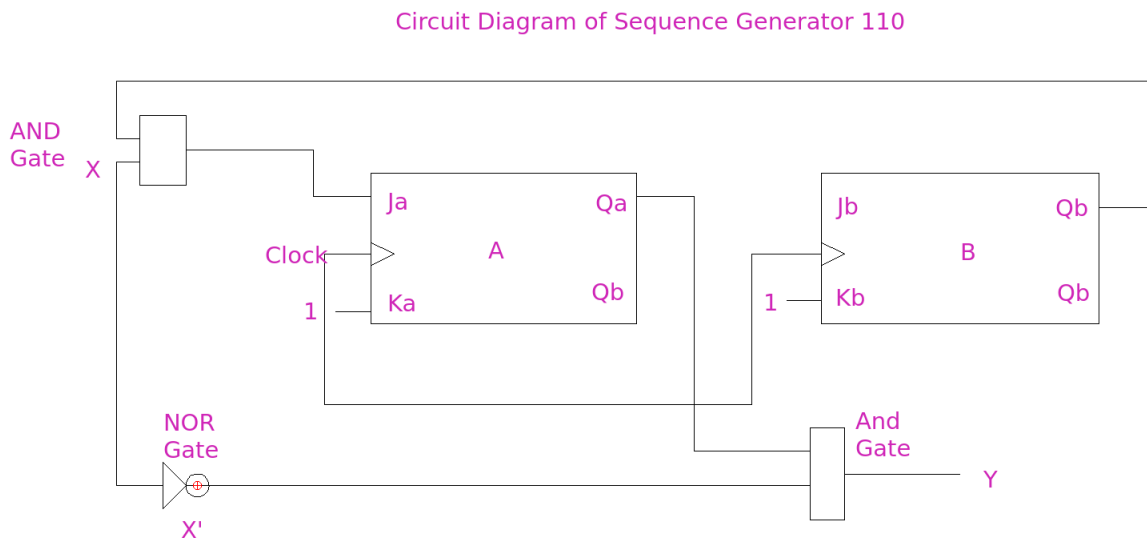
		$Q_B Q_A$			
		00	01	11	10
1	0	X	X	X	X
	1	X	X	X	X

$$K_B = 1$$

		$Q_B Q_A$			
		00	01	11	10
1	0	0	0	0	0
	1	1	0	X	X

$$Y = Q_a \bar{X}$$

9 Circuit Diagram



10 Procedure

1. Draw state diagram
2. Make a state table
3. Apply state reduction if required
4. Assign states
5. Re-write the state table using states assigned
6. Prepare state transition table
7. Use k-maps to find inputs to Flip Flop
8. Draw the circuit diagram.

11 Conclusion

Thus, we have learnt a about Working of Mealy Machines. State diagrams, transition diagrams and conversion of K-maps to Boolean expressions were also studied. We have also learnt how to design a Mealy Machine using the above mentioned concepts.

12 FAQs

1. What is Sequence Detector?

A sequence detector is a sequential circuit that outputs 1 when a particular pattern of bits sequentially arrives at its data input. The figure below shows a block diagram of a sequence detector. It has two inputs and one output. The inputs are the clock used to synchronize the functionality of the circuit and the data input.

Sequence detector is of two types:

- Overlapping
- Non-Overlapping

In an overlapping sequence detector, the last bit of one sequence becomes the first bit of the next sequence. However, in a non-overlapping sequence detector, the last bit of one sequence does not become the first bit of the next sequence. In this post, we'll discuss the design procedure for non-overlapping 101 Mealy sequence detectors.

2. What are applications of Sequence Detector?

- Sequence detectors are used in the decoding equipment on the ground to provide “flags” which indicate the beginning (or end) of a data block (e.g., a TV frame).
- They can be used to detect if a bunch of states in a circuit occur in a particular desired pattern.

MIT WORLD PEACE UNIVERSITY

Digital Electronics and Computer Architecture
Second Year B. Tech, Semester 3

WRITE AN ASSEMBLY LANGUAGE PROGRAM TO
DISPLAY 2 DIGIT AND 4 DIGIT HEXADECIMAL
NUMBERS USING 64 BIT ASSEMBLY.

PRACTICAL REPORT
ASSIGNMENT 7

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

December 3, 2022

Contents

1 Problem Statement	2
2 Objective	2
3 Platform	2
4 Theory	2
4.1 Assembly Language Program Basic Structure	2
4.2 System calls to write and exit.	2
4.2.1 System Call to write/output call	2
4.2.2 System exit call	3
4.3 Instruction used in the program for implementation	3
4.4 Commands to execute the program	3
5 Algorithm	3
6 Input	4
7 Output	4
8 Code	4
9 Conclusion	5
10 FAQs	5

1 Problem Statement

Write an assembly language program (ALP) to display 2-digit and 4-digit hex numbers using 64-bit assembly language programming.

2 Objective

- To understand the structure of the assembly language program.
- To understand system call function for write and exit.

3 Platform

CPU - Core i7 Duo, 64 bit with 4 GHz clock frequency.

OS - Arch Linux, 64 bit

Editor - VS code

Assembler - NASM (Netwide Assembler)

Linker - LD, GNU linker.

4 Theory

4.1 Assembly Language Program Basic Structure

```
section.Data (declare data segment)
; initialized data declaration
section.bss (declare block started by segment sort)
; uninitialized data declaration.
section.text (declare code segment)
global-start (entry point for program)

-start:
;code.

(semicolon is used to give the comment)
```

4.2 System calls to write and exit.

4.2.1 System Call to write/output call

display variable -name contents of specified variable length on monitor.

```
mov rax,1 ; function number for writing/outputting the data.
mov rdi,0 ; file descriptor ID for standard input device (keyboard)
mov rsi,arr ; starting addresses of the variable used to store the data.
mov rdx,8 ; maximum bytes to be read.
syscall ; system call (in built function)
```

4.2.2 System exit call

function to exit or terminate program.

```
mov rax,60 ; function number for sys-exit
mov rdi,0  ; return code for zero error.
syscall    ; system call.
```

4.3 Instruction used in the program for implementation

ADD

add two numbers together

COMPARE

compare numbers

JUMP

jump to designated RAM address.

LOAD

Load information from RAM to the CPU.

4.4 Commands to execute the program

to assemble:

```
nasm -f elf64 hello.asm
```

to link:

```
ld -o hello hello.o
```

to execute:

```
./hello
```

where, hello is the filename.

5 Algorithm

1. start
2. Display message "Two-digit HEX Number"
3. Initialize hard coded two digit and four-digit number.
4. Write a procedure for unpacking BCD number (Display Outputs)
5. Display two digit and four digit numbers.
6. end.

6 Input

Two digit and four digit numbers.

7 Output

Two digit and four digit numbers.

The Two digit Hex number is:
2A

8 Code

```
1 ; display 2 digit hexx numbers
2 section .data
3     msg db "The Two digit Hex number is: ", 10
4     msglen equ $-msg
5     num1 db 2AH ; the number to be printed. h is for hex
6
7 section .bss
8 ; temp data assignment
9     sum resb 1
10    temp resb 1
11
12 section .text
13 global _start
14
15 _start:
16     ; printing the first message
17     mov rax, 1
18     mov rdi, 1
19     mov rsi, msg
20     mov rdx, msglen
21     syscall
22
23     ; assign one byte of num1 to al
24     mov al, byte[num1]
25     ;assign the value of al to sum
26     mov byte[sum], al
27     ;assign 2 to bp
28     mov bp, 2; bp = 2
29     ; shift all binary bits 4 times to right, this flips the nibbles
30     ; so rn its 0010 0011 after flipping it becomes 0011 0010
31
32 up:rol al, 4
33     ; assign al to bl
34     mov bl, al; al = 32H
35     ; and with 0FH, so 0000 1111 anded with 0000 0010 so youll end up with the
    0010
36     and al, 0FH ; al = 02H at this point
37     ; this would trigger some flag
38     cmp al, 09
39     ; goto down label if the above cmp statement gives less than or equal to
40     jbe down
41     add al, 07H
```

```
42
43 down: Add al, 30H; al = 32H
44     mov byte[temp], al
45     mov rax, 1
46     mov rdi, 1
47     mov rsi, temp
48     mov rdx, 1
49     syscall
50     mov al, bl ; bl = 23H
51     dec bp ; this is the loop register which we decrement if its 0 then we stop
the loop
52     jnz up; now go to up again, and this time you would use bl's value to al coz
you would rotate it again.
53
54 mov rax, 60
55 mov rdi, 0
56 syscall
```

9 Conclusion

Thus, implemented the program in assembly language to display two digit and four-digit hex numbers

10 FAQs

1. Explain assembler directives. List the assembler directives in your program.

Assembler Directives supply data to the program and control the assembly process. It enables to do the following:

- Assemble code and data into specified sections.
- Reserve space in memory for uninitialized variables.
- Control the appearance of listings.
- Initialize memory.
- Assemble conditional blocks.
- Define global variables.
- Specify libraries from which the assembler can obtain macros.
- Examine symbolic debugging information.

Assembler Directives in the program:

- .text switch to text segment.
- .data switch to initialized part of data segment.
- .bss switch to uninitialized part of data segment.

2. Illustrate the significance of the sections: .data, .bss, .text .bss segment stands for 'block starting symbol' is the memory space for uninitialized variable of your code . IT is the method of optimization to reduce the code size.

syntax: section.bss

var-name RES memory-Type memory size.

Eg. `section.bss`
`A resb 5D`
(Declare variable A allocate 50 bytes memory)

`.data` section holds the initialized value. It holds the data of the initialized variable (global or local)

syntax:
`section.data`
`var_name data_type variable_value.`

Eg:
`A DB 50`
(declared variable A of type byte with value 50)

`.text` segment is the code, vector table and constants. It is the section that holds the executable instructions.

syntax:
`section.text`
`global_start`
`_start:`
`;code`

3. What is the difference between RESB and DB?

DB stands for Declare/Define Byte. It is a directive that is used to allocate space for initialized data in the data section.

RESB Stands for Reserve Byte. It is a directive that is used to allocate space for uninitialized data in `.bss` section.

MIT WORLD PEACE UNIVERSITY

Digital Electronics and Computer Architecture
Second Year B. Tech, Semester 3

WRITE AN ASSEMBLY LANGUAGE PROGRAM (ALP)
TO IMPLEMENT ADDITION AND SUBTRACTION OF
8-BIT NUMBERS (USING USER INPUT, MACRO AND
PROCEDURE)

PRACTICAL REPORT
ASSIGNMENT 7

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

December 3, 2022

Contents

1 Objectives	1
2 Platform Used	1
3 Theory:	1
3.1 Assembly language program basic structure:	1
3.2 System calls to read, write and Exit.	1
3.3 Describe the instruction used (e.g MOV, ADD, SUB)	2
4 Code	2
5 Output	3
6 Conclusion	3
7 FAQs:	4
7.1 Explain assembler directives. List the assembler directives in your program. .	4
7.2 Explain why 30H 137H is added to convert the digit to ASCII?	4
7.3 Define Macro and Procedure:	4

1 Objectives

Write an assembly language program (ALP) to implement addition and subtraction of 8-bit numbers (Using user input, macro and procedure)

- To understand assembly language programming.
- To study instruction set of 8086.

2 Platform Used

Operating System : Arch Linux

Editor – Visual Studio Code

Assembler – NASM (Netwide Assembler)

LINKER – LD, a GNU linker

3 Theory:

3.1 Assembly language program basic structure:

A ALP is series of statements which are either assembly language instruction such as ADD and MOV or statements called directives. A program language instruction consists of following 4 fields.

[Label] mnemonic [operands] [;comment]

A square bracket([]) indicates that the field is optional

A Label field allows the program to refer to a line of code by name. The label Fields cannot exceed a certain no.of characters.

The mnemonics and operands fields together perform the real work of the program and accomplish the tasks statements like ADD A,C and MOV C,#68 where ADD and MOV are the mnemonics, which produce opcodes, "AC" and "C,#68" are operands. These two fields could contain directives. Directives do not generate machine code and are used only by the assembler, whereas instructions are translated into machine code for the CPU to execute.

The comment fields begins with a semicolon which is a comment indicator

Notice the label "HERE" in the program. Any label which refers to an instruction should be followed by a colour.

3.2 System calls to read, write and Exit.

Reading From a file:

- Put the system call sys_oread in EAX register
- Put file descriptors in the EBX register.
- Put the pointer to input buffer in ECX register.
- Put the pointer buffer size i.e. number of bytes to read in EDX register.

Writing to a file:

- Put system call `sys_write()` in EAX register
- Put file descriptive in EBX register
- Put buffer size i.e. the number of bytes to write in EDX.

Exit From File:

- Put system call `sys_exit` in EAX register.

3.3 Describe the instruction used (e.g MOV, ADD, SUB)

1. INC - used for incrementing and operand by one works on single operand that can be a memory or in a register
2. DEC - used for decrementing an operand by one work on a single operand.
3. ADD and SUB - used for performing simple addition subtraction of binary data in byte, word and double word size.
4. MUL/IMUL - used for multiplying data both affect the carry and overflow flag.

4 Code

```
1 ; display 2 digit hexx numbers
2 section .data
3     msg db "Addition of 2 numbers ", 10
4     msglen equ $-msg
5     num1 db 3AH ; the number to be printed. h is for hex
6     num2 db 22H ; the number to be printed. h is for hex
7
8 section .bss
9 ; temp data assignment
10    sum resb 1
11    temp resb 1
12
13 section .text
14 global _start
15
16 _start:
17     ; printing the first message
18     mov rax, 1
19     mov rdi, 1
20     mov rsi, msg
21     mov rdx, msglen
22     syscall
23
24     ; assign one byte of num1 to al
25     mov al, byte[num1]
26     add al, byte[num2]
27     ;assign the value of al to sum
28     ;mov byte[sum], al
29     ;assign 2 to bp
30     mov bp, 2; bp = 2
31     ; shift all binary bits 4 times to right, this flips the nibbles
32     ; so rn its 0010 0011 after flipping it becomes 0011 0010
```

```
33
34 up:rol al, 4
35     ; assign al to bl
36     mov bl, al; al = 32H
37     ; and with 0FH, so 0000 1111 anded with 0000 0010 so youll end up with the
    0010
38     and al, 0FH ; al = 02H at this point
39     ; this would trigger some flag
40     cmp al, 09
41     ; goto down label if the above cmp statement gives less than or equal to
42     jbe down
43     add al, 07H
44
45 down: Add al, 30H; al = 32H
46     mov byte[temp], al
47     mov rax, 1
48     mov rdi, 1
49     mov rsi, temp
50     mov rdx, 1
51     syscall
52     mov al, bl ; bl = 23H
53     dec bp ; this is the loop register which we decrement if its 0 then we stop
    the loop
54     jnz up; now go to up again, and this time you would use bls value to al coz
    you would rotate it again.
55
56 mov rax, 60
57 mov rdi, 0
58 syscall
```

5 Output

Addition of 2 numbers
5C

6 Conclusion

Thus learnt how to add numbers using Assembly Language and Display name.

7 FAQs:

7.1 Explain assembler directives. List the assembler directives in your program.

These are the statements that direct the assembles to do something. As the name says it directs the assembles to do a task. They are classified into the following categories based on the function performed by them.

- CODE - This assembler directive indicates the beginning of the code segment.
- Data - Indicates beginning of data segment
- Mode - Is used for selecting a standard memory model for the assembly program.
- Stack - The directive is used for displaying the stack.

7.2 Explain why 30H 137H is added to convert the digit to ASCII?

30H is the ASCII code for a digit '0' then 31H, 32H, 39H, corresponds 1, 2, 3, ..., 9. 41H is the ASCII code for a letter 'A'. As in hexadecimal a value of 10 would be represented by the 'A'. Basically that function segmented in two possible additions convert the internal value into a printable character that properly represents value.

7.3 Define Macro and Procedure:

- Macro - A macro in computer science is a set of rules or programmable patience which decrypts a specific sequence of output.
- Procedure - Procedures are used for a large set of rules. They help make a large program more readable. It indicates a set of instructions that executes a particular task.