PREFACE

This document has been prepared to serve as a laboratory manual for EEE 4307 Digital Electronics course for electrical engineering students. The manual consists of a set of experiments designed to allow students to build, and verify digital circuits and systems. This set of experiments cover relevant topics prescribed in the syllabus and are designed to reinforce the theoretical concepts taught in the classroom with practical experience in the lab. By the end of the course, students are expected to have a good understanding of digital logic design and implementation with SSI and MSI devices.

I would like to acknowledge the contributions of Mr. Ishtiza Ibne Azad, Mr. Md. Moshiur Rahman Farazi, Mr. Md. Ghulam Saber and Mr. Md. Thesun Al-Amin. This manual could not have been written without the unique support of them.

Suggestions from readers for improving the accuracy and clarity of this manual will be warmly welcomed.

Contents

EXPERIMENT NO. 1: FAMILIARISATION AND USE OF TRUTH TABLE WITH BASIC LOGIC GATES.

OBJECTIVE

- To study the basic logic gates: AND, OR, INVERT, NAND and NOR.
- To study the representation of these functions by truth tables, logic diagrams and Boolean algebra.
- To observe the pulse response of logic gates.

APPARATUS:

- IC Type 7400 Quadruple 2-input NAND gates
- IC Type 7402 Quadruple 2-input NOR gates
- IC Type 7404 Hex Inverters
- IC Type 7408 Quadruple 2-input AND gates
- IC Type 7432 Quadruple 2-input OR gates
- IC Type 7486 Quadruple 2-input XOR gateS

THEORY:

AND A multi-input circuit in which the output is 1 only if all inputs are 1.

The symbolic representation of the AND gate is shown in Fig. 1a.

OR A multi-input circuit in which the output is 1 when any input is 1.

The symbolic representation of the OR gate is shown in Fig. 1b.

INVERT The output is 0 when the input is 1, and the output is 1 when the input is 0.

The symbolic representation of an inverter is shown in Fig. 1c.

NAND AND followed by INVERT.

The symbolic representation of the NAND gate is shown in Fig 1d.

NOR OR followed by INVERT as shown in Fig 1e.

EX-OR The output of the Exclusive –OR gate, is 0 when it's two inputs are the same and its output is 1 when its two inputs are different.

Truth Table Representation of the output logic levels of a logic circuit for every possible combination of levels of the inputs. This is best done by means of a systematic tabulation.

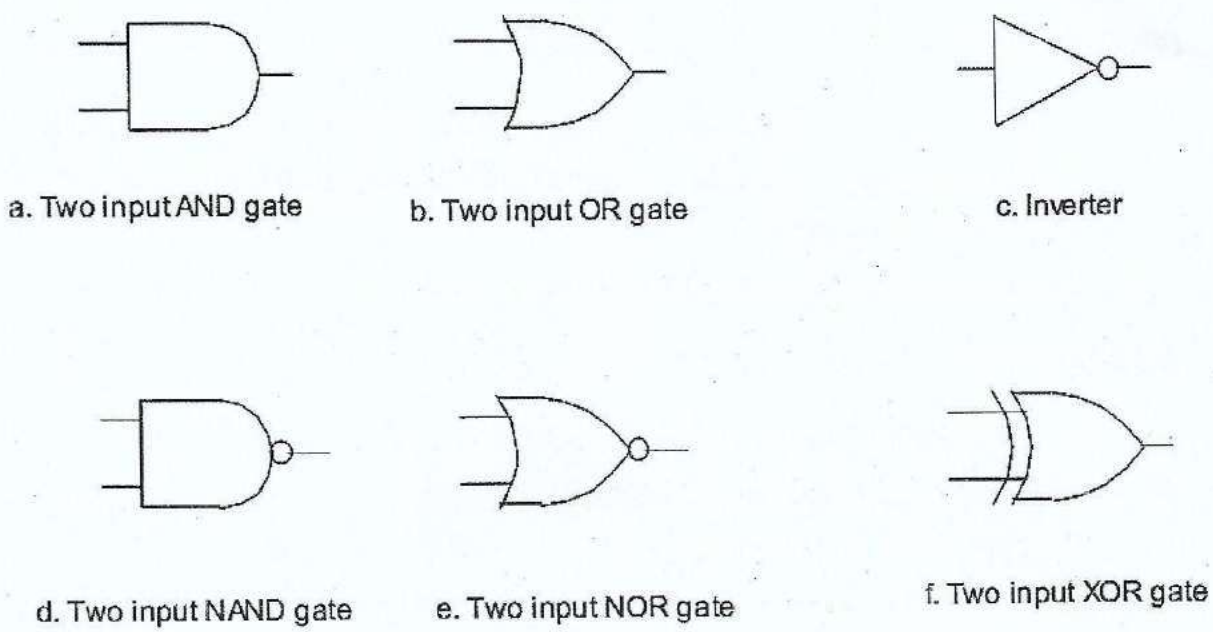


Fig.1 Symbols for digital logic gates

Part 1: Logic Functions

I. AND, OR, NAND, and NOR gates.

1. Use one gate for each IC 7400 (NAND), 7402 (NOR), 7408 (AND), 7432 (OR), 7486 (XOR). Each has input pins, 1 and 2, and output pin 3.

2. Connect pin 1 to switch S1-1, pin 2 to switch S1-2, and pin 3 to LED-1 for every gate as shown in Fig 2 as an example for the NAND gate.

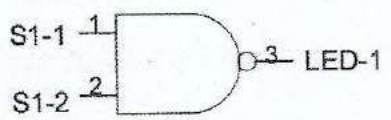


Fig.2 Two input NAND gate

3. Using logic switches S1-1 and S-2, apply the logic levels 0 and 1 to gate inputs (pin 1, pin 2), in the sequence shown in table 1. Record the output logic levels (see lamp LED-1) in table 1. Repeat the recordings for each gate.

Remember: Lamp ON = Logic 1, (High)
Lamp OFF = Logic 0 (Low)

IC 7408 (AND)

Pin 1	Pin 2	Pin 3
0	0	0
0	1	0
1	0	0
1	1	1

IC 7432 (OR)

Pin 1	Pin 2	Pin 3
0	0	0
0	1	1
1	0	1
1	1	1

IC 7400 (NAND)

Pin 1	Pin 2	Pin 3
0	0	1
0	1	1
1	0	1
1	1	0

IC 7402 (NOR)

Pin 1	Pin 2	Pin 3
0	0	1
0	1	0
1	0	0
1	1	0

IC 7486 (XOR)

Pin 1	Pin 2	Pin 3
0	0	0
0	1	1
1	0	1
1	1	0

4. Use an inverter gate from IC 7404 whose input pin is pin 1 and whose output pin is pin 2.

5. Using logic switches S1-1, apply the logic levels 0 and 1 in the sequence shown in table 2. Record the output logic levels in table 2

Table 2.

Pin 1	Pin 2
0	1
1	0

IC 7404 (Inverter)

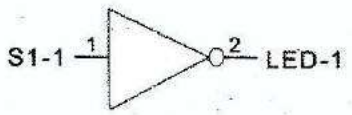


Fig.3 Inverter gate

Part-2: Response of Logic Gates:

Connect the circuits of figures 4 and 5 and write the corresponding truth tables 3 and 4, respectively.

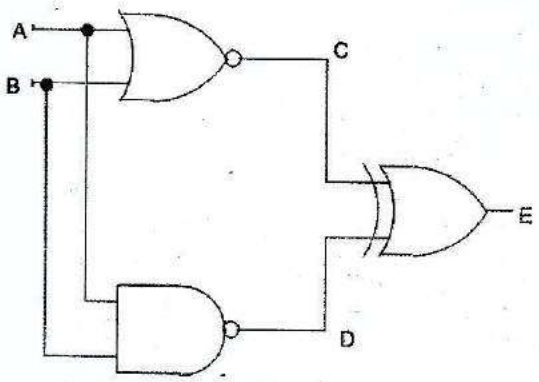


Fig. 4

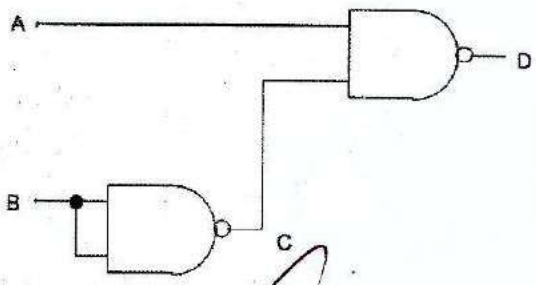


Fig. 5

Table 3.

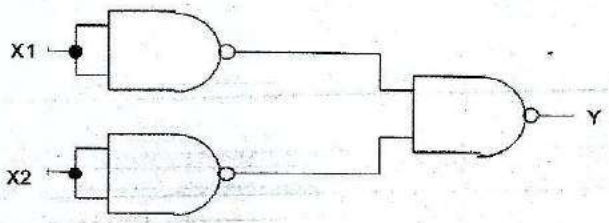
A	B	C	D	E
0	0	1	1	0
0	1	0	1	1
1	0	0	1	1
1	1	0	0	0

Table 4.

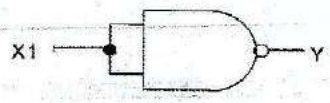
A	B	C	D
0	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1

Review Questions:

1. Write a truth table for each circuit. Derive Boolean expressions for all outputs.

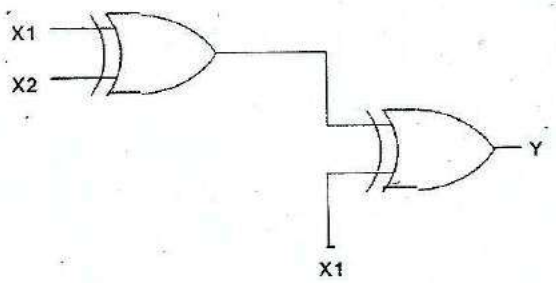


circuit 1.1

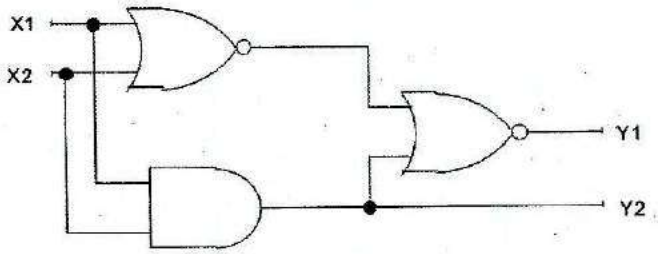


circuit 1.2

circuit 1.3



circuit 1.4



circuit 1.5

2. A burglar alarm for a car has a normally low switch on each of four doors. If any door is opened the output of that switch goes HIGH. The alarm is set off with an active-LOW output signal. What type of gate will provide this logic? Support your answer with an explanation.

Answer:

1. circuit 1.1

$$y = \overline{(\overline{x_1} \cdot x_1)(\overline{x_2} \cdot x_2)}$$

$$\Rightarrow y = \overline{\overline{x_1} \cdot \overline{x_2}}$$

$$\Rightarrow y = \overline{\overline{x_1}} + \overline{\overline{x_2}}$$

$$\therefore y = x_1 + x_2$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

circuit 1.2

$$y = \overline{x_1 \cdot x_1}$$

$$\therefore y = \overline{x_1}$$

x_1	y
0	1
1	0

circuit 1.3

$$y = \overline{\overline{x_1 \cdot x_2} \cdot \overline{x_1 \cdot x_2}}$$

$$\Rightarrow y = \overline{\overline{x_1 \cdot x_2}}$$

$$\therefore y = x_1 \cdot x_2$$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

circuit 1.4

$$y = (x_1 \oplus x_2) \oplus x_1$$

x_1	x_2	y
0	0	0
0	1	1
1	0	0
1	1	1

circuit 1.5

$$y_1 = \overline{(\overline{x_1 + x_2}) + (x_1 \cdot x_2)}$$

$$= x_1 \oplus x_2$$

$$y_2 = x_1 \cdot x_2$$

x_1	x_2	y_1	y_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2. An OR-gate will provide this function. The work of an OR-gate is to provide a high output if any of the inputs are high.

EXPERIMENT NO. 2: FAMILIARISATION WITH UNIVERSAL GATES AND STUDY OF DE MORGAN'S THEOREM.

OBJECTIVE

- To study the universal logic gates: NAND and NOR.
- To prove the De Morgan's Theorem.
- To observe the simplification using De Morgan's theorem.

APPARATUS:

- IC Type 7400 Quadruple 2-input NAND gates
- IC Type 7402 Quadruple 2-input NOR gates
- IC Type 7404 Hex Inverters
- IC Type 7408 Quadruple 2-input AND gates
- IC Type 7432 Quadruple 2-input OR gates

THEORY:

De Morgan's theorem state that the complement of a function is obtained by interchanging AND and OR operators and complementing each literal.

$$(A+B)' = A'B' \cdots (1)$$

$$(AB)' = A'+B' \cdots (2)$$

The complement of any function may be derived algebraically through De Morgan's Theorem. The theorem can be extended to three or more variables. So, according to the generalized form of De Morgan's theorem it can be shown that,

$$(A+B+C+D+ \ldots..+X+Y+Z)' = A'B'C' \ldots..X'Y'Z'$$

$$(ABCD \ldots XYZ)' = A'+B'+C'+D'+....+X'+Y'+Z'$$

This experiment will deal with the verification of De Morgan's Theorem using basic logic gates and Universal gates.

Universal Gates can be used to implement any Boolean function or digital system. The NAND gate is universal and so is a NOR gate. The very basic logical operations AND, OR and NOT can be implemented with NAND gates.

The NOT operation can be implemented using a 1-input NAND gate (Figure 2.1). The operation is actually obtained by shorting all inputs of the NAND gate. In this case, with both inputs shorted, only the first and last input combinations are possible and hence NOT operation is obtained. Alternately, NOT operation may also be obtained by grounding one of the two inputs and using the other for eternal input.

Figure: 2.1

The AND operation can be implemented using two NAND gates (Figure 2.2). While a first NAND gate would produce inverted AND output, the second NAND gate would act as an inverter to produce the desired normal output.

Figure: 2.2

The OR operation can be implemented using a NAND gate with inverters in each input (Figure 2.3).

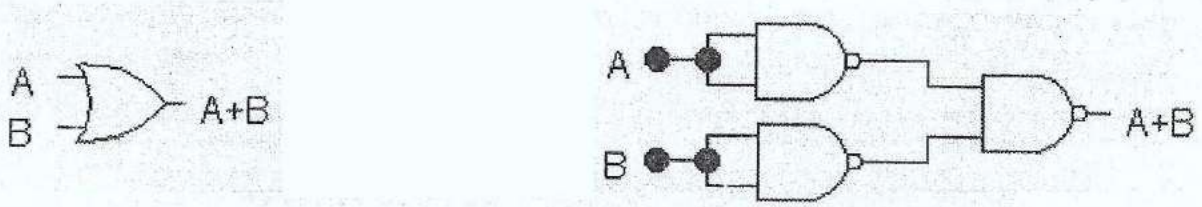
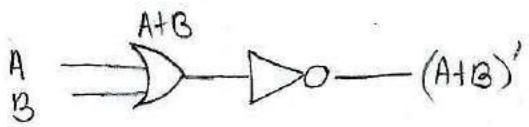


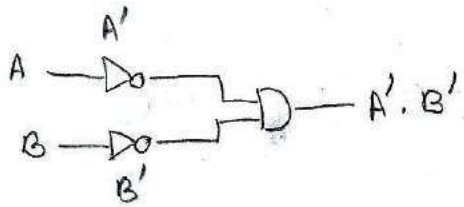
Figure: 2.3

Tasks:

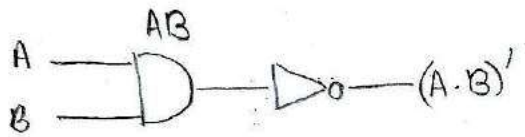
1. Use Hex Inverter IC 7404, Quad 2-input AND Gate IC 7408 and Quad 2-input OR Gate IC 7432 to implement and verify De Morgan's Theorem shown at equations (1) and (2) respectively.
   Draw the circuit diagram and truth tables.



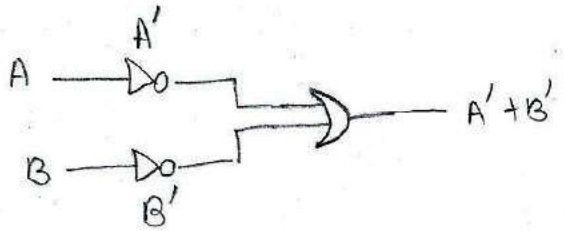
A	B	(A+B)'
Pin 1	Pin 2	Pin 3
0	0	1
0	1	0
1	0	0
1	1	0



A	B	A'.B'
Pin 1	Pin 2	Pin 3
0	0	1
0	1	0
1	0	0
1	1	0

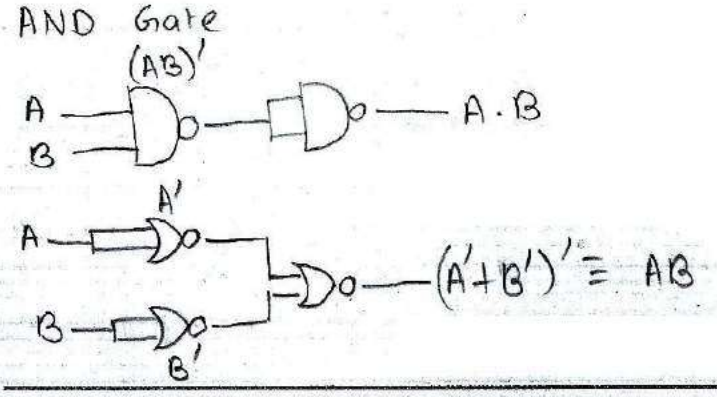


A	B	(A.B)'
Pin 1	Pin 2	Pin 3
0	0	1
0	1	1
1	0	1
1	1	0



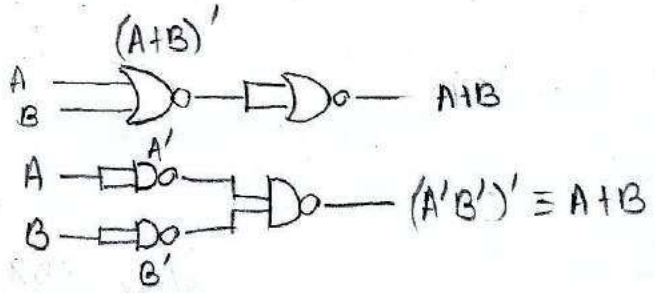
A	B	A'+B'
Pin 1	Pin 2	Pin 3
0	0	1
0	1	1
1	0	1
1	1	0

2. Implement AND gate, OR gate and NOT gate respectively using either Quad 2-input NAND gate IC 7400 or Quad 2-input NOR gate IC 7402. Verify your results using truth tables.
   Draw the circuit diagram here:



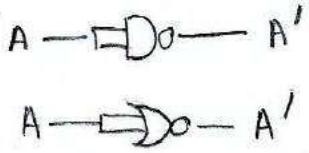
A	B	A.B
Pin 1	Pin 2	Pin 3
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate

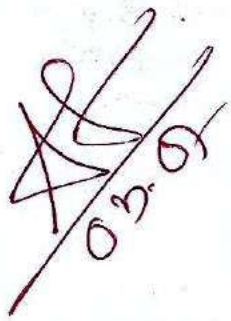


A	B	A+B
Pin 1	Pin 2	Pin 3
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gate

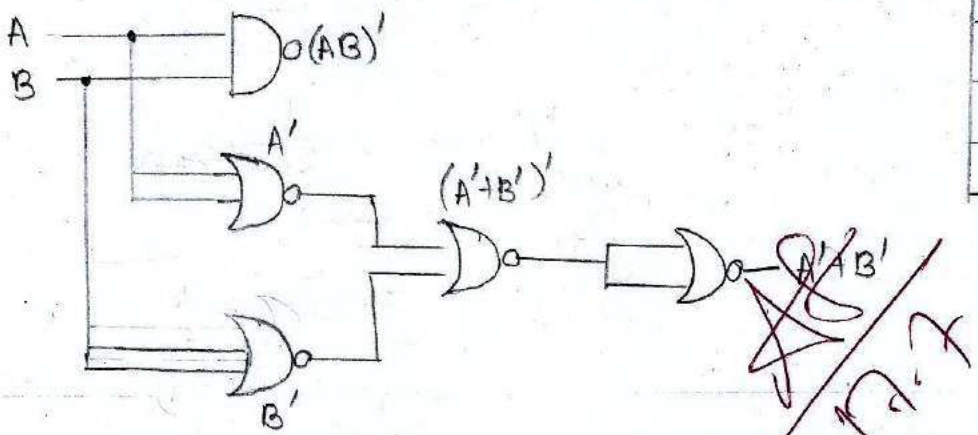


A	A	A'
Pin 1	Pin 2	Pin 3
0	0	1
1	1	0



3. Replace AND gates, OR gates and NOT gates with Universal gates where applicable. Verify De Morgan's Theorem with circuits with Universal gates only. You may use either Quad 2-input NAND Gate IC 7400 or Quad 2-input NOR gate IC 7402.
   Draw the circuit diagram here:

A	B	(AB)'	A'+B'
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0



$(AB)' \equiv A'+B'$

Tasks for Room:

Four chairs A, B, C, and D are placed in a row. Each chair may be occupied ("1") or empty ("0"). A Boolean function F is "1" if and only if there are two or more adjacent chairs that are empty.

1. Give the truth table defining the Boolean function F.
2. Express F as a minterm expansion (standard sum of product).
3. Express F as a maxterm expansion (standard product of sum).

Answer:

1.

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

K' Map:

AB \ CD	00	01	11	10
00	1	1	1	1
01	1	0	0	0
11	1	0	0	0
10	1	1	0	0

2. Standard SOP form:

$F = A'B' + C'D' + AB'C'$

3. Standard POS form:

$F = (B'+D')(B'+C')(A'+C')$

EXPERIMENT NO. 3: COMBINATIONAL CIRCUIT DESIGN-I.

OBJECTIVE

- To implement combinational circuits with minimum gates.
- To observe the simplification using K-Map.

APPARATUS:

- IC Type 7400 Quadruple 2-input NAND gates
- IC Type 7402 Quadruple 2-input NOR gates
- IC Type 7404 Hex Inverters
- IC Type 7408 Quadruple 2-input AND gates
- IC Type 7432 Quadruple 2-input OR gates

THEORY:

In combinational circuit design, you have to reduce a particular expression to its simplest form or change its form to a more convenient one to implement the expression more efficiently. Use the basic laws, rules and theorems of Boolean algebra to manipulate and simplify an expression. Thorough knowledge of Boolean algebra, considerable practice in its application, ingenuity and cleverness will help you in design.

Task 1: Design a 4-bit majority gate circuit whose output is 1 if the majority of the inputs are 1's and output is 0 otherwise. Simplify the circuit with the help of Boolean algebra if necessary.

Steps:

1. Create truth table.
2. Get the Boolean Expression.
3. Simplify using K-Map.
4. Draw the circuit diagram for minimum gates.
5. ~~Give pin numbers.~~
6. Verify the result with truth table.

Truth Table:

A	B	C	D	O
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	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Boolean Expression:

$BCD + ACD + ABD + ABC$

$= AB(C+D)' + CD(A+B)$

K-Map Simplification:

AB / CD	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	0

Task 2: Simplify with K' Map providing truth table and circuit diagram:

i) $y(wz' + wz) + xy$

Truth Table:

w	x	y	z	O_1
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	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

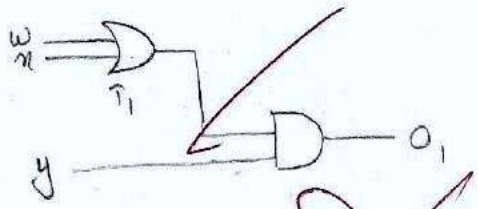
K' Map:

wx \ yz	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Simplified Expression:

$xy + wy = y(x + w)$

Circuit Diagram:



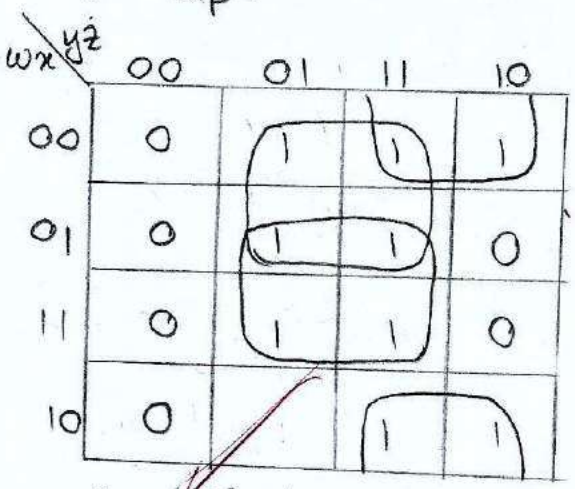
$T_1 = w + x$ $O_1 = y(w + x)$

(iii) $w'z + x'y + xy'z + yz$

Truth Table:

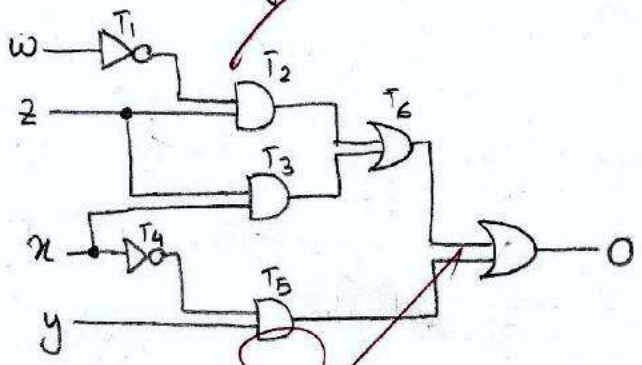
w	x	y	z	O
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

K' Map:



Simplified Expression:
$F = w'z + x'y + xz$

Circuit Diagram:

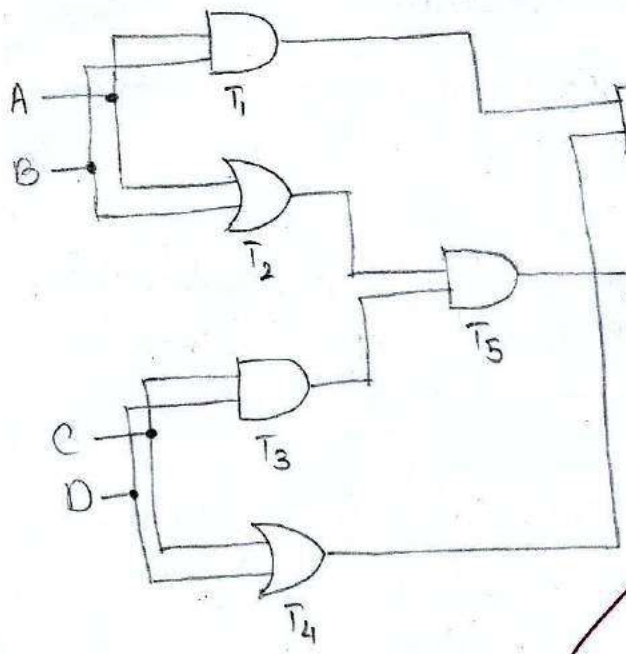


$T_1 = w'$ $T_2 = w'z$ $T_3 = xz$

$T_4 = x'$ $T_5 = x'y$ $T_6 = w'z + xz$

$O = w'z + x'y + xz$

Circuit Diagram ~~with Pin number:~~



$O_1 = AB(C+D) + CD(A+B)$

$T_1 = AB$ $T_2 = A+B$

$T_3 = CD$ $T_4 = C+D$

$T_5 = CD(A+B)$

$T_6 = AB(C+D)$

Task for Room:

Design a circuit which receives two 2-bit binary numbers A=a_1a_0 and B=b_1b_0. Design a logic circuit to produce an output 1 whenever A>B. Use Boolean Algebra if necessary. Provide design with truth table and simplification.

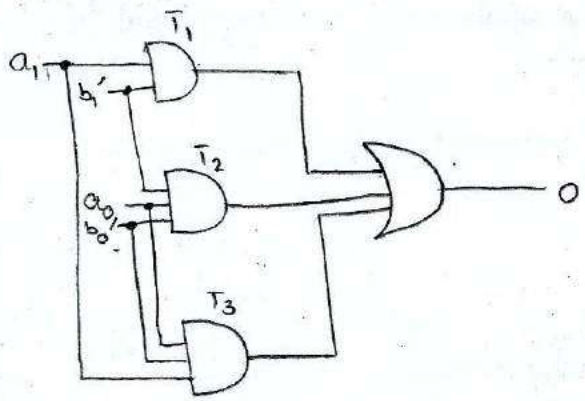
Truth Table:

a_1	a_0	b_1	b_0	O
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

K' Map:

a_1a_0 \ b_1b_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

$F = a_1b_1' + a_0b_1'b_0' + a_1a_0b_0'$



$T_1 = a_1b_1'$ $T_3 = a_1a_0b_0'$

$T_2 = a_0b_1'b_0'$ $O = a_1b_1' + a_0b_1'b_0' + a_1a_0b_0'$

EXPERIMENT NO. 4: BINARY ADDER OPERATION.

OBJECTIVE

- To understand and implement binary Half Adder and Full Adder circuits.
- To design and implement a 4-bit binary adder.

APPARATUS:

- IC Type 7408 Quadruple 2-input AND gates
- IC Type 7432 Quadruple 2-input OR gates
- IC Type 7486 Quadruple 2-input XOR gates
- IC Type 74283 4 bit binary Full Adder

THEORY:

Adders play an important role in computer and digital system in which numerical data are processed. Addition operations in such cases are carried out by using binary numbers. The addition of two binary numbers is performed in exactly same manner as addition of decimal numbers. The least significant digits are added first. A "Carry" of 1, if needed, is passed onto the next stage. The normal addition of two binary numbers is termed as "Half Adder" operation and addition involving two bits with a previous carry is called "Full Adder" operation. In other words, the Full Adder operation yields "Sum" and "Carry" whereas "Half" adders return no "Carry".

Task 1: Design a 1-bit Half Adder circuit using only basic logic gates.

Steps:

1. Create truth table.
2. Get the Boolean Expression for Carry **C** and Sum **S.**

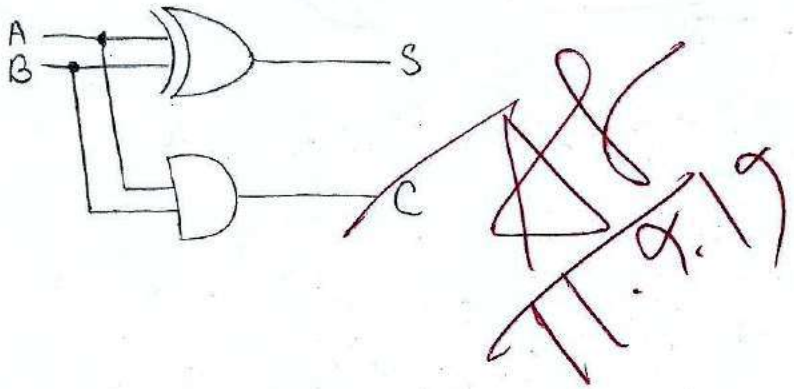
C = AB

S = $A \oplus B$

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

3. Draw the circuit diagram for minimum gates and pin numbers.

Circuit Diagram with Pin numbers:



4. Verify the result with truth table.

Task 2: Design a 1-bit Full Adder circuit using only basic logic gates.

1. Create the truth table pertaining to the input carry.
2. Get the Boolean Expression for Carry out C_{OUT} and Sum S with the help of K-Map simplification.

A \ BCin	00	01	11	10
0	0	0	1	0
1	0	1	1	1

← For Cout

$C_{OUT} = BC_{in} + AC_{in} + AB$

$= C_{in}(A \oplus B) + AB$

$S = A \oplus B \oplus C_{in}$

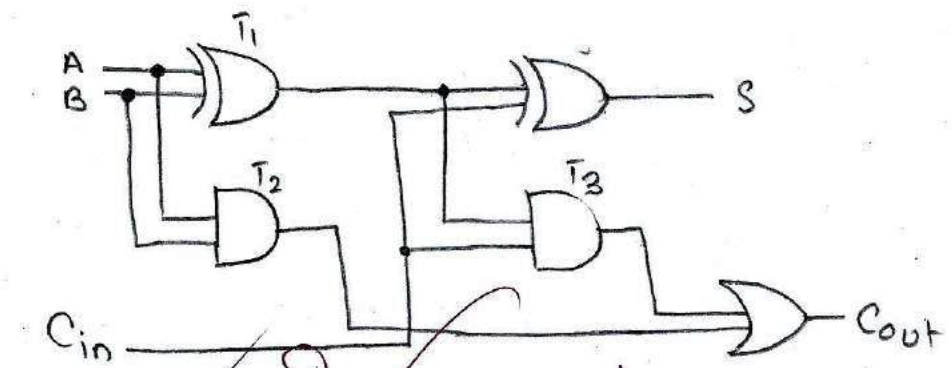
A \ BCin	00	01	11	10
0	0	1	0	1
1	1	0	1	0

← For S

A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

3. Draw the circuit diagram with minimum gates and pin numbers.

Circuit diagram with pin numbers:



$T_1 = A \oplus B$

$S = A \oplus B \oplus C_{in}$

$T_2 = AB$

$T_3 = C_{in}(A \oplus B)$

$C_{out} = C_{in}(A \oplus B) + AB$

4. Verify the result with truth table.

Task 3: Design a 4-bit binary adder circuit which will add two 4-bit binary numbers and generate a 4-bit sum and a 1-bit carry as output. Use 4-bit binary adder IC 74283.

- A and B represent the 2 4-bit binary numbers.
- Pin 7 and 9 are Input and Output Carry respectively.
- SUM1 through SUM4 are the addition results.

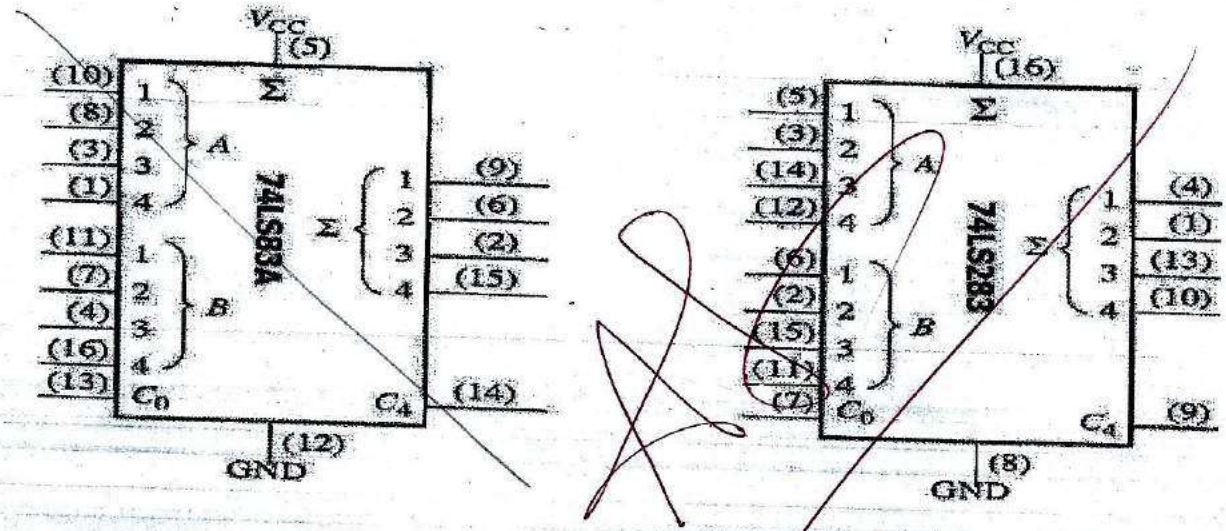
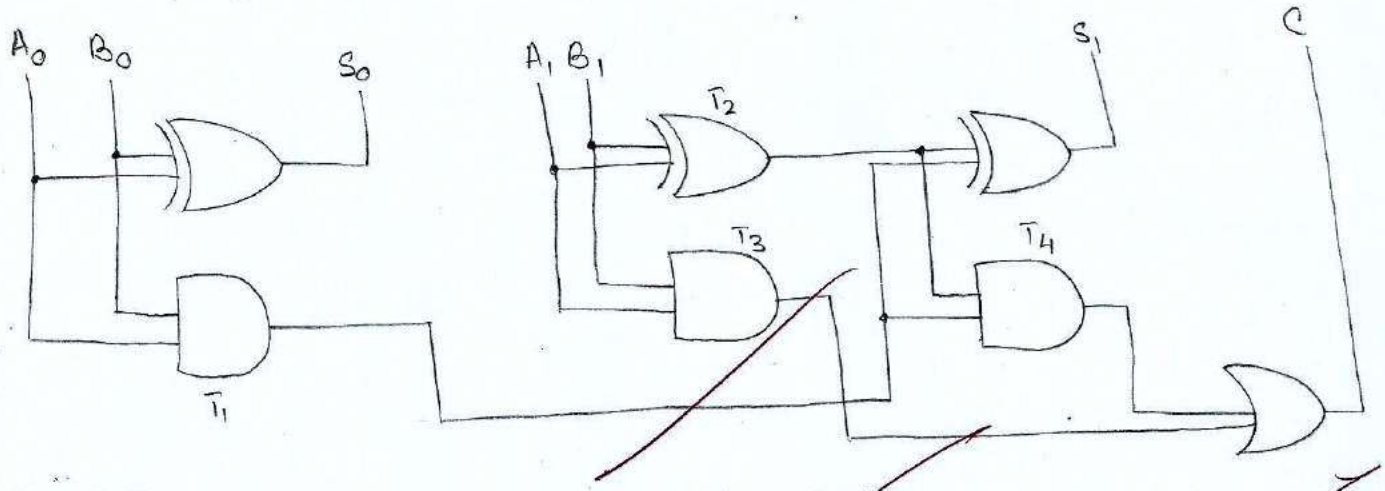


Figure 1: Pin Configuration of IC 7483 and 74283

Task 4: Design a 2-bit binary adder using basic logic gates.

Circuit Diagram:



$T_1 = A_0B_0$ $\quad T_3 = A_1B_1$ $\quad S_0 = A_0 \oplus B_0$ $\quad C = (A_1 \oplus B_1)(A_0B_0)$

$T_2 = A_1 \oplus B_1$ $\quad T_4 = (A_1 \oplus B_1)(A_0B_0)$ $\quad S_1 = A_1 \oplus B_1 \oplus (A_0B_0)$ $\quad + A_1B_1$

A_1	A_0	B_1	B_0	C	S_1	S_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Beyond the Lab:

Design a circuit which receives 2(two) 16-bit binary numbers $A=a_{15}$ .....a_0, $B= b_{15}$ .....b_0 and adds them. You cannot use any adder ICs (e.g. 4, 8, 16 bit adders), only can use basic gate ICs (e.g. AND, OR, XOR, NOR, NAND, NOT). You design should contain

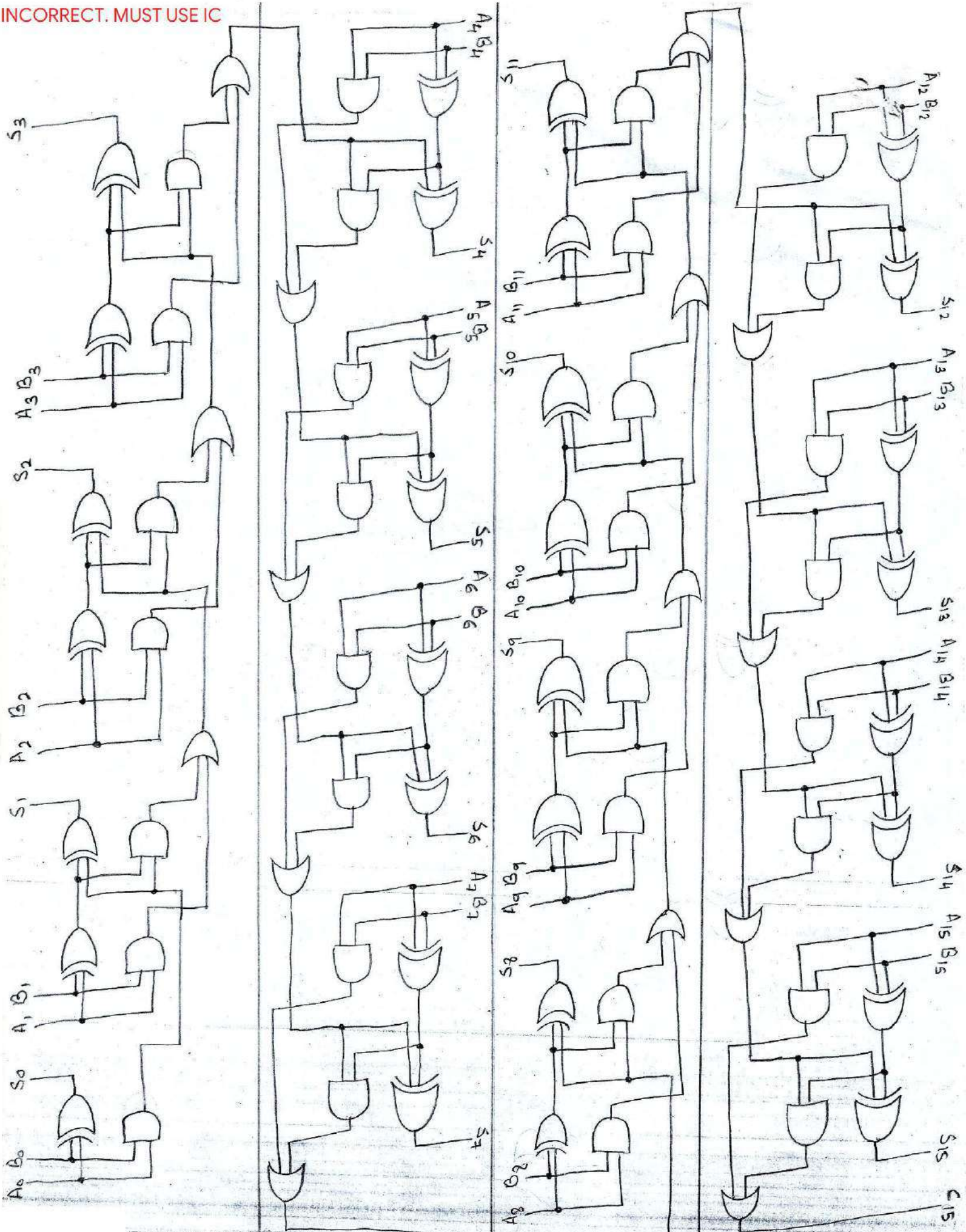
- IC diagram with pin numbers showing the interconnection between different ICs.
- A succinct description giving a coherent justification of the different parts of the design.

Hint: A 4 bit adder circuit can be created cascading 4 1-bit full adder circuits constructed only using basic gates (refer to Task 2).

A 16-bit binary adder is simply 1 Half-Adder and 15 Full-Adders. The first addition, $A_0 + B_0$, does not have a previous carry, and thus does not require a full adder. Each successive full adder after that takes three inputs, A_n, B_n and C_{n-1}, the carry from the previous addition, and give two outputs S_n and C_n, which goes to the next adder.

A full adder is two half adders, which in turn is an XOR gate and an OR gate. The inputs for the first half to both gates is A and B, and the inputs for the second half to both gates is C_{n-1} and the XOR output from the first half. The AND outputs from the first and second halves go to an OR gate to form C_n. The final result is $S = C_{15} S_{15} S_{14} \dots S_0$

INCORRECT. MUST USE IC



EXPERIMENT NO. 5: COMBINATIONAL CIRCUIT DESIGN-II (Arithmetic Circuit Design)

OBJECTIVE

- To implement combinational circuits.
- To design and implement a 4-bit binary adder/subtractor circuit with a selection variable to select the mode of operation.

APPARATUS:

- IC Type 7408 Quadruple 2-input AND gates
- IC Type 7432 Quadruple 2-input OR gates
- IC Type 7486 Quadruple 2-input XOR gates
- IC Type 74283 4 bit binary Full Adder

THEORY:

Digital computers perform variety of information tasks. Among the functions encountered are the various arithmetic operations. The most basic arithmetic operation is the addition of two binary digits. This simple addition consists of four possible elementary operations. $0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, $1 + 1 = 10$. The first
three operations produce sum of one digit, but when the both augends and addend bits are equal 1, the binary sum consists of two digits. The higher significant bit of the result is called carry.

A binary adder-subtractor is a combinational circuit that performs the arithmetic operations of addition and subtraction with binary numbers. We will develop this circuit by means of a hierarchical design. The half adder design is carried out first, from which we develop the adder. Connecting 'n' full adders in cascade produces a binary adder for two n-bit numbers. The subtraction circuit is included by providing a complementing circuit.

Task: Design a 4-bit Adder/ Subtractor circuit with one selection variable S and 2 4-bit inputs A and B. Use IC 74283 for your design.

PROCEDURE:

We can use XOR gates to perform a 1 's complement on command. The following figure 1 shows that if the control input is high we can achieve the complemented output.

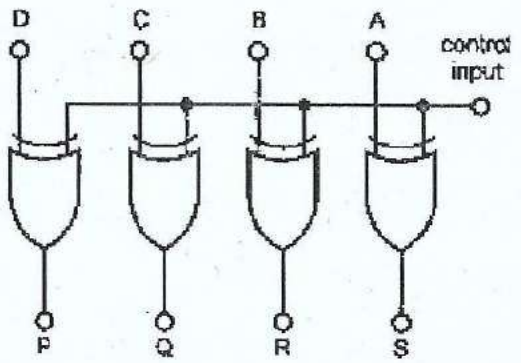


Figure 1

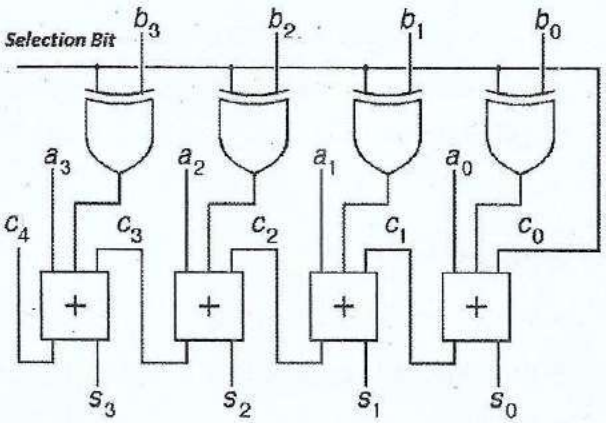
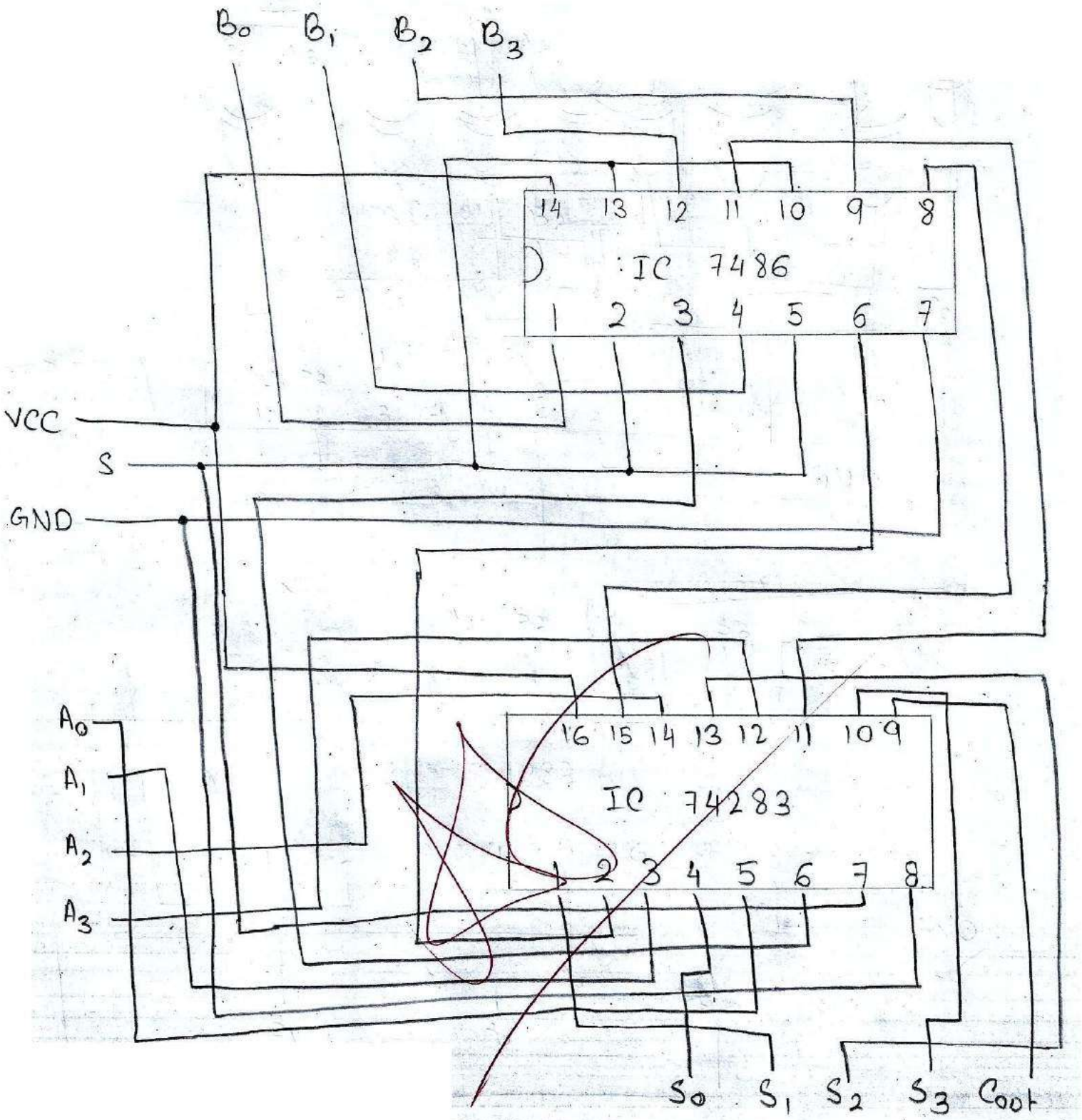


Figure 2

Now we can do the subtraction operation by the complemented output of the XOR gates. The addition and subtraction operations can be combined into one circuit with one common binary adder. This is done by including an XOR gate with each full adder. A 4-bit adder-subtractor circuit is shown in figure 2. The Selection Bit (SB) controls the operation. When SB = 0, the circuit is an adder, and when SB = 1, the circuit becomes a subtractor. Each XOR gate receives input SB and one of the inputs of B. when SB = 0, we have B (XOR) 0 = B. the full adder receive the value of B, the input carry is 0, and the circuit performs A plus B. when SB = 1, we

have B (XOR) 1= and C0 = 1. The B inputs are complemented and a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B.

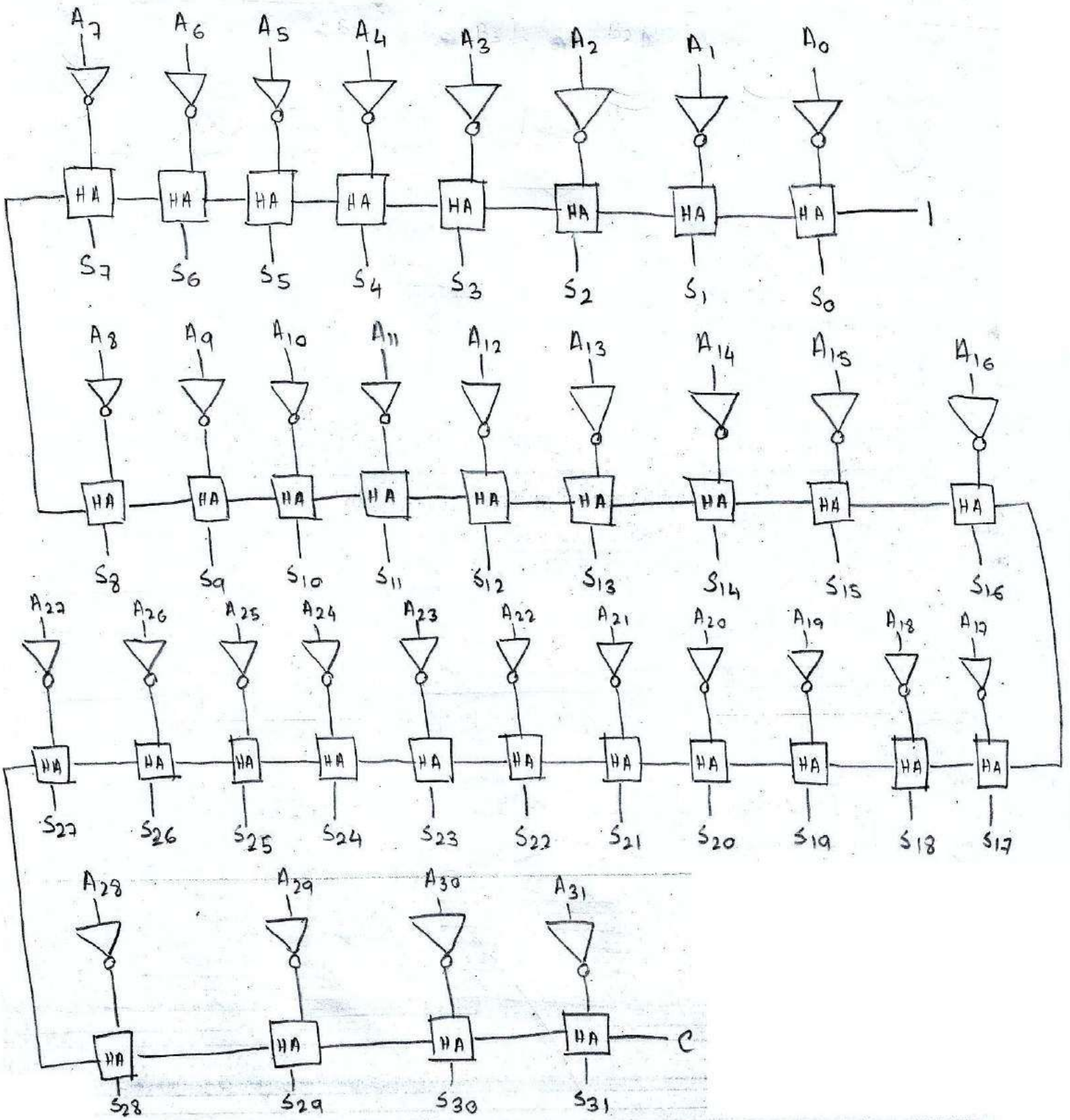
1. Draw the IC diagram showing the interconnection between the ICs. Give pin
   number and description if necessary.
2. Implement the circuit. Verify the results.



Beyond the Lab:

Design a circuit which takes a 32 bit binary number as input and take the 2's complement of the input.

Hint: 2's complement can be obtained by adding 1 to the 1's complement.



EXPERIMENT NO. 6: OPERATION OF SEQUENTIAL LOGIC CIRCUITS: The JK Flip-Flop.

OBJECTIVE

- To understand and implement JK Flip-Flop using universal gates.
- To design and implement a Master-Slave JK Flip-Flop using JK Flip-Flop IC.

APPARATUS:

- IC Type 7400 Quadruple 2-input NAND gates
- IC Type 7410 Triple 3-input NAND gates
- IC Type 7473 Dual JK Flip-Flop with PRESET and CLEAR

THEORY:

Every digital system is likely to have combinational circuits, circuits in which the outputs at any instant of time are entirely dependent upon the inputs present at that time. However, most systems encountered in practice also include memory elements, which require that the system be described in terms of sequential logic.

A sequential logic circuit consists of a combinational circuit to which memory elements are connected to form a feedback path. The memory elements are devices capable of storing binary information within them. The binary information stored in the memory elements at any given time defines the state of the sequential circuit. The sequential circuit receives binary information from external inputs. The inputs, together with the present state of the memory elements, determine the binary value at the output terminals. They also determine the condition for changing the state in the memory elements.

There are two main types of sequential circuits: synchronous and asynchronous. Their classification depends on the timing of their signals. Synchronous sequential circuits that use clock pulses in the inputs of the memory elements are called clocked sequential circuits. The memory elements used in sequential circuits are called flip-flops. A flip-flop can maintain a binary state indefinitely until directed by an input signal to switch states. The major differences among various types of

flip-flops are in the number of inputs they possess and in the manner in which the inputs affect the binary state.

The JK (Jack Kilby, inventor of integrated circuit) flip-flop is a versatile and widely used flip-flop. The functioning of the JK flip-flop is identical to that of the SR flip-flop in the SET, RESET and no-change conditions of operation. The difference is that the JK flip-flop has no invalid state as does the SR flip-flop.

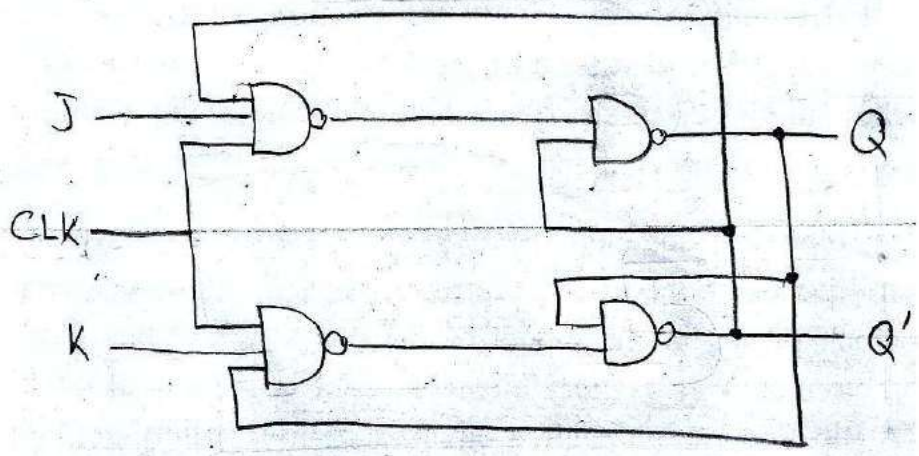
Task 1: Design and implement a JK Flip-Flop using both 2-input and 3-input NAND gates and derive the characteristic table.

Steps:

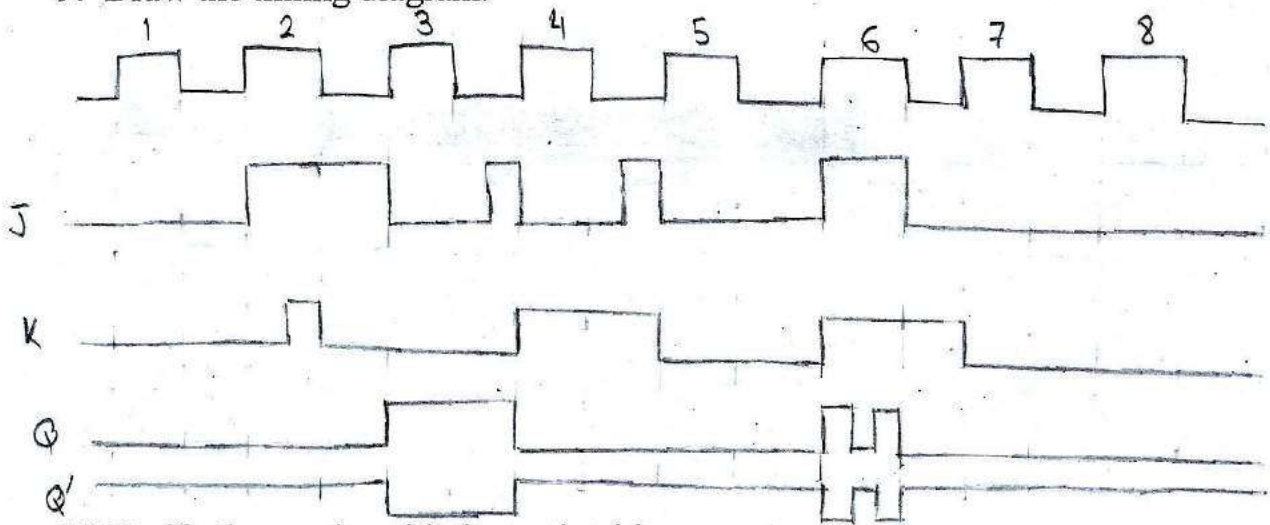
7. Create the truth table.

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	Q_t'

8. Draw the circuit diagram.



9. Draw the timing diagram.



10. Verify the results with the truth table.

Task 2: Use IC 7473 (Dual JK flip-flop IC with PRESET and CLEAR) to construct a Master-Slave JK flip-flop and observe the states of the inputs, the intermediate and final outputs with the variation of clock pulse. You are advised to use a low frequency clock pulse and if needed, a manual pulse.

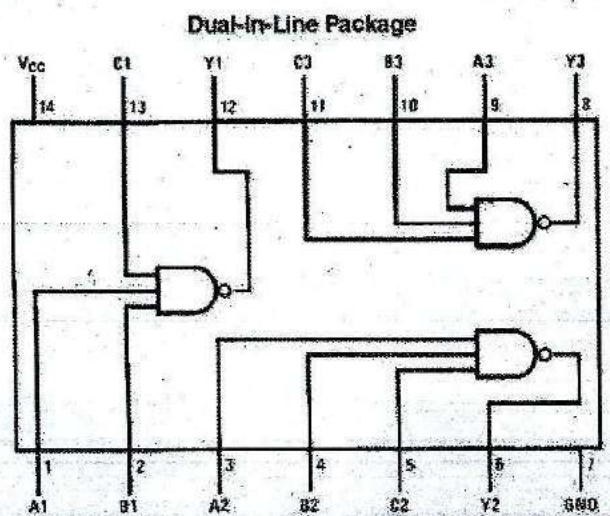


Fig. 1: IC 7410 (Triple 3-input NAND gate)

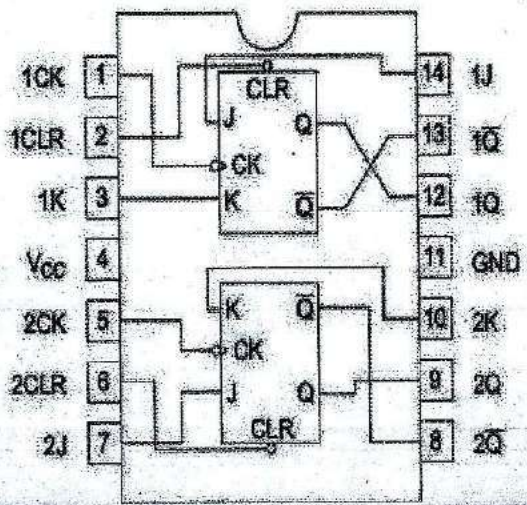


Fig. 2: IC 7473 (Dual JK flip-flop IC with PRESET and CLEAR)

Function Table

Inputs				Outputs	
Clear	Clock	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	↓	L	L	Q_0	$\overline{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	Toggle	
H	H	X	X	Q_0	$\overline{Q}_0$

H, high level, L, low level, X, irrelevant, ↓, transition from high to low level,

Q_0, level of Q before the indicated steady-state input conditions were established.

$\overline{Q}_0$, complement of $\overline{Q}_0$ or level of Q before the indicated steady-state input conditions were established.

Toggle, each output changes to the complement of its previous level on each active transition indicated by ↓.

Fig. 3: IC 7476 (Dual JK flip-flop IC with PRESET and CLEAR) Truth Table

Beyond the Lab:

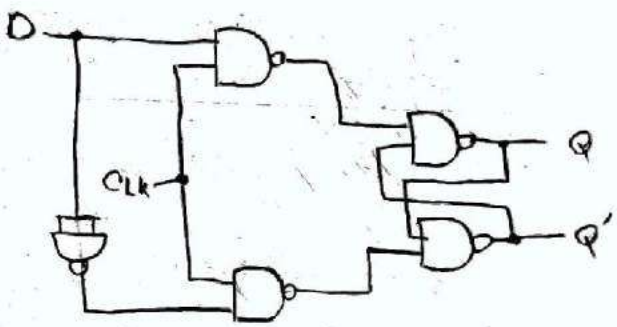
A *JN* flip-flop has two inputs, *J* and *N*. Input *J* behaves like the *J* input of a JK flip-flop and input *N* behaves like the complement of the *K* input of a JK flip-flop (that is, $N=K'$).

(a) Tabulate the characteristic table of the flip-flop.
(b) Show that by connecting two inputs together, one obtains a D flip-flop.

a)

Q	J	N	$Q_{(t+1)}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

b) For a D flip-flop, only 1-bit is stored, so whatever input is given goes directly to the output. From the truth table above, we can see that when the inputs are connected, i.e. J = N, the output, $Q_{(t+1)} = J = N$, thus acting as a D flip-flop. The circuit diagrams are the same.



D Flip Flop

EXPERIMENT NO. 7: SEQUNTIAL LOGIC CIRCUIT DESIGN: THE JOHNSON COUNTER AND RING COUNTER.

OBJECTIVE

- To understand about Synchronous counter
- To design and implement a Johnson Counter using IC 7474.
- To design and implement a Ring Counter using IC 7474

APPARATUS:

- IC **7474** Dual D-Type Positive Edge Triggered Flip-Flops with PRESET and CLEAR.

THEORY:

Shift registers consist of an arrangement of flip-flops and are important in applications involving the storage and transfer of data in a digital system. The basic difference between a register and a counter is that a register has no specified sequence of states, except in certain very specialized applications. A register in general is used solely for storing and shifting data entered into it from an external source and typically possesses no characteristic internal sequence of states. A shift register counter is basically a shift register with the serial output connected back to the serial input to produce special sequences. These devices are often classified as counters because they exhibit a specified sequence of states. Two of the most common types of shift register counters are the Johnson counter and the Ring counter. The **Johnson** digital counter is a synchronous shift register with feedback from the inverted output (Q') of the last flip-flop. Q' of the last flip flop is connected back to the input D of the first flip-flop. A **ring** counter is formed by feeding the output of a shift register to its own input.

Task 1: Design and implement a 4-bit Johnson counter and verify your result with the characteristic table.

Steps:

1. Draw the 4-bit Johnson counter circuit diagram.

2. Write down the truth table

3. Draw the timing diagram

4. Verify the results with the truth table.

Task 2: Design and implement a 4-bit Ring counter and verify your result with the characteristic table.

Steps:

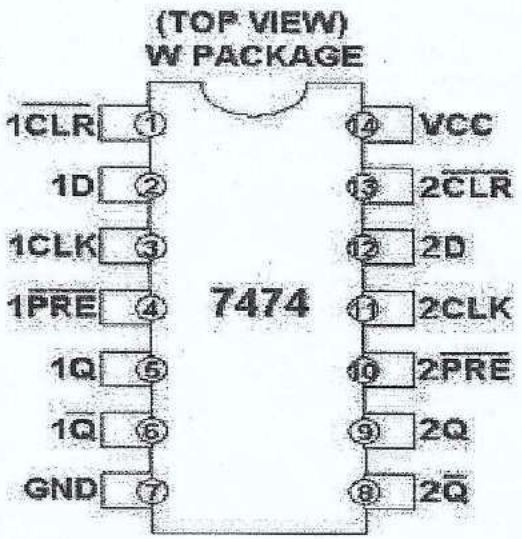
1. Draw the 4-bit Ring counter circuit diagram.

2. Write down the truth table

3. Draw the timing diagram:

4. Verify the results with the truth table.

Pin Configuration of IC 7474:



Function Table

Inputs				Outputs	
PR	CLR	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H (Note 1)	H (Note 1)
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q_0}$

H = HIGH Logic Level
X = Either LOW or HIGH Logic Level
L = LOW Logic Level
↑ = Positive-going transition of the clock.
Q_0 = The output logic level of Q before the indicated input conditions were established.
Note 1: This configuration is nonstable; that is, it will not persist when either the preset and/or clear inputs return to their inactive (HIGH) level.

Beyond the Lab:

Design and verify the truth table of a 4-bit up/down counter.

EXPERIMENT NO. 8: FLIP-FLOP APPLICATIONS: FREQUENCY DIVISION OPERATION AND SHIFT REGISTERS.

OBJECTIVE

- To perform frequency division operation using JK flip-flop.
- To design and implement a 4-bit Parallel Input Serial Output (PISO) shift register.

APPARATUS:

- IC Type 7400 Quadruple 2-input NAND gates
- IC Type 7474 Dual D-type Flip-Flop
- IC Type 7473 Dual JK Flip-Flop with PRESET and CLEAR

THEORY:

A flip-flop is a sequential device able to store one binary bit of information. When a pulse waveform is applied to the clock input of a JK flip-flop that is connected to toggle, the Q output is a square wave with half the frequency of the clock input. If more flip-flops are connected together, further division of the clock frequency can be achieved. Thus a single flip-flop can be applied as a divide-by-2 device. By connecting flip-flops in this way, a frequency division of 2^n is achieved, where **n** is the number of flip-flops. For example, four flip-flops divide the clock frequency by $2^4=16$.

Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop. Most of the registers possess no characteristic internal sequence of states. All the flip-flops are driven by a common clock, and all are set or reset simultaneously. Registers can be classified according to how their stored information is entered or removed. A serial register is one in which the data is entered or removed one bit at a time and a parallel register accepts or transfers all bits of data simultaneously. Serial Input- Parallel Output networks as well as the inverse are also available.

Task 1: Design and construct a frequency divider using IC 7476 that divides the clock frequency by 8.

1. Draw the circuit diagram.

2. Draw the timing diagram.

Task 2: Design a 4-bit Parallel Input Serial Output (PISO) shift register using IC 7474 and NAND gates.

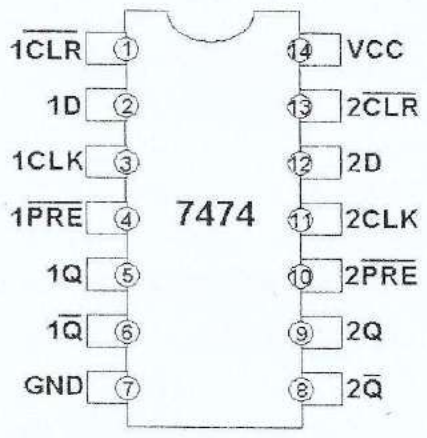


Figure: IC 7474 Dual D-type Flip-Flop

1. Draw the circuit diagram.

2. Draw the timing diagram.

Beyond the Lab:

Design the logic diagram of a 4-bit register with four *D* flip-flops and four 4x1 multiplexers with mode-selection inputs S_1 and S_0. The register operates according to the following function table:

S_1	S_0	Register Operation
0	0	No change
0	1	Complement the four outputs
1	0	Clear register to 0 (synchronous with the clock)
1	1	Load parallel data

EXPERIMENT NO. 9: REALIZATION OF BOOLEAN LOGIC CIRCUITS AND ITS SIMPLIFICATION.

OBJECTIVE

- To study the realization of Boolean logic circuits from given Boolean logic expression.
- To learn how to analyze a given digital circuit by finding its Boolean expression which represents that circuit.
- To convert any Boolean logic circuits to its equivalent universal gates' representation.

APPARATUS:

- IC Type 7400 Quadruple 2-input NAND gates
- IC Type 7402 Quadruple 2-input NOR gates
- IC Type 7404 Hex Inverters
- IC Type 7408 Quadruple 2-input AND gates
- IC Type 7432 Quadruple 2-input OR gates
- IC Type 7486 Quadruple 2-input XOR gates

THEORY:

In computer science, a **Boolean expression** is a logical statement that if we evaluate them, it will produce a Boolean value – one of the stable values from **true (1)** and **false (0)**. An exemplary Boolean expression may be compiled of a combination of the Boolean constants – **true** or **false**, Boolean variables, Boolean operators, and Boolean-valued functions.

Boolean algebra is the mathematics based on logic which we use to analyses digital gates and circuits. It is used to define and reduce Boolean expression using its set of rules and laws.

A universal logic gate is a logic gate that can be used to construct all other logic gates. The **NOR** gate and the **NAND** gates have such particular property that any

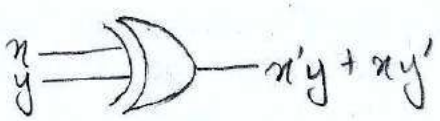
other basic logic gates as well as complex Boolean expression can be created if appropriately designed. In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families.

We can use the "Laws of Boolean" to both reduce and simplify a complex Boolean expression into an equivalent one in an attempt to reduce the number of logic gates required. If equivalent expression is achieved with fewer components, the result will be increased reliability and decreased cost of manufacture.

Task 1: Conversion of Boolean expression into logic circuits

1. Implement the following Boolean expressions into logic circuits using basic gates from 7408 (AND), 7432 (OR), 7486 (XOR), 7404 (Inverter).
   i. $x'y + xy'$
   ii. $xy + x'y' + y'z$
   iii. $y(wz' + wz) + xy$
2. Draw the circuit diagram for each of the Boolean expression mentioned above.
3. Create the corresponding truth tables for each of the Boolean expression using required number of variables.
4. Verify the result of the circuits with the truth tables

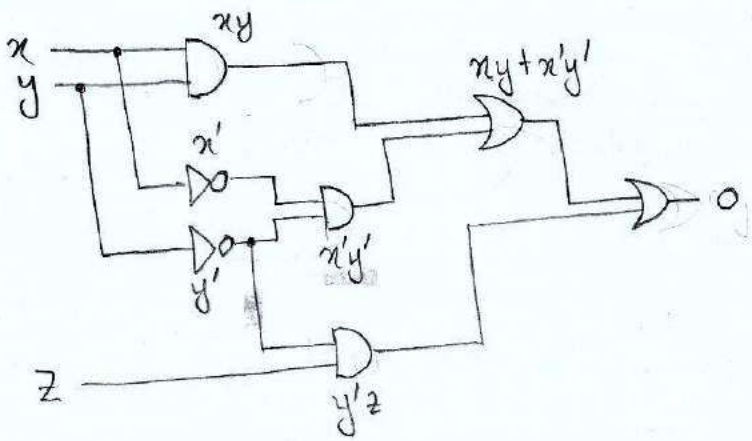
Circuit Diagram of (i)



Truth Table of (i)

x	y	$x'y + xy'$
0	0	0
0	1	1
1	0	1
1	1	0

Circuit Diagram of (ii)

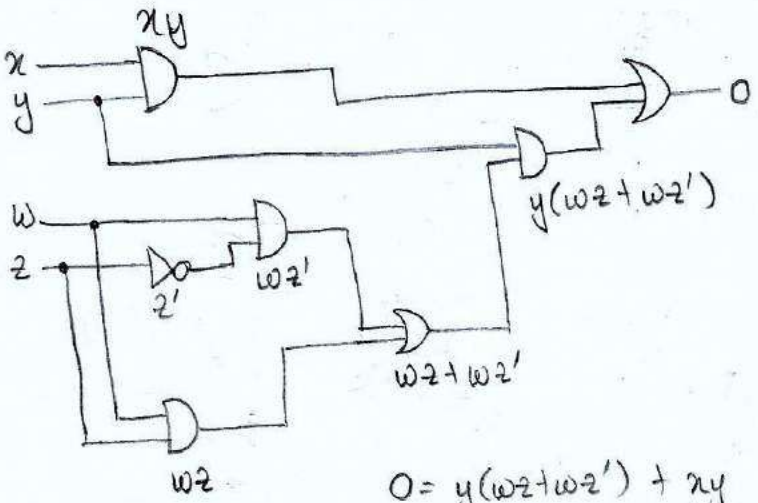


$O = xy + x'y' + y'z$

Truth Table of (ii)

x	y	z	O
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Circuit Diagram of (iii)



$O = y(wz + wz') + xy$

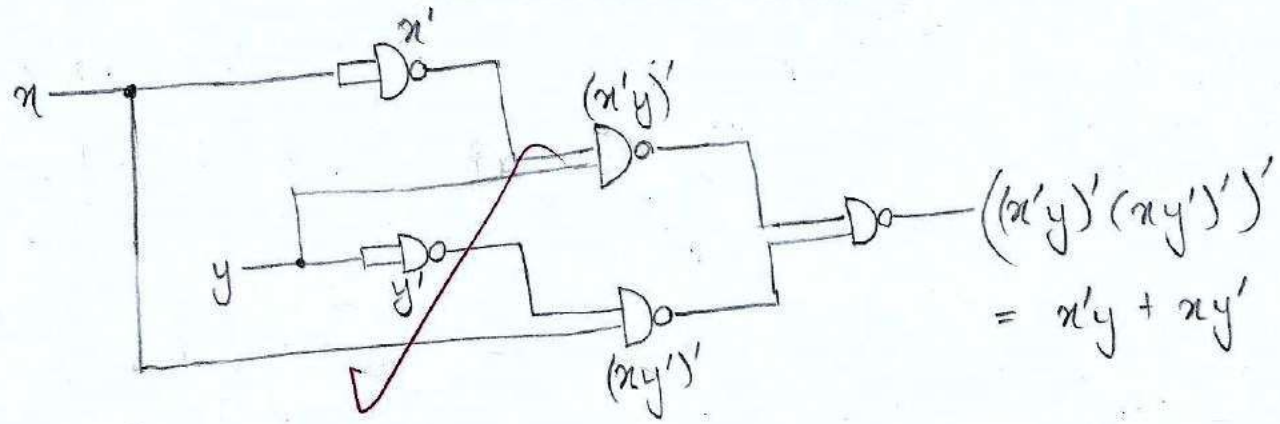
Truth Table of (iii)

w	x	y	z	O
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

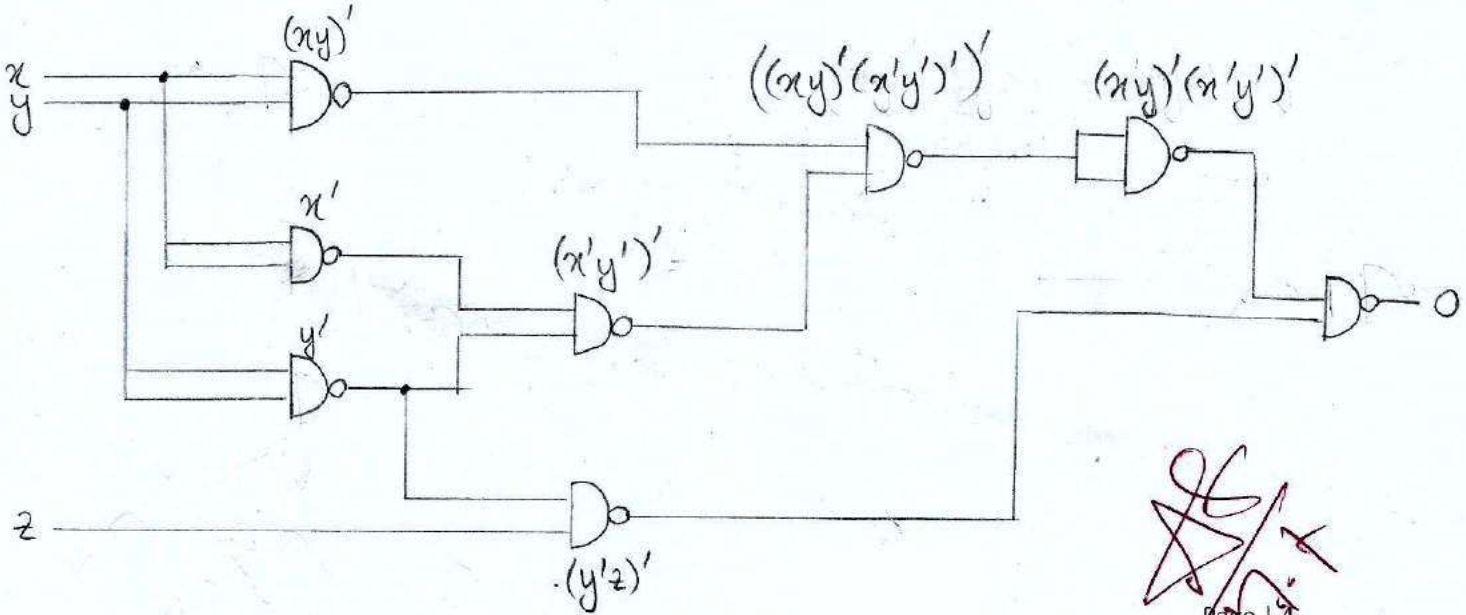
Task 2: Implementing logic circuits using Universal gates

1. From task 1, now implement all of the Boolean expressions listed using only the universal gates either IC 7400 (NAND) or 7402 (NOR)
2. Redraw the circuit diagrams using either NAND or NOR gates for each of Boolean expressions
3. Verify the result of the circuits with the truth tables created in task 1

Revised Circuit Diagram of (i)

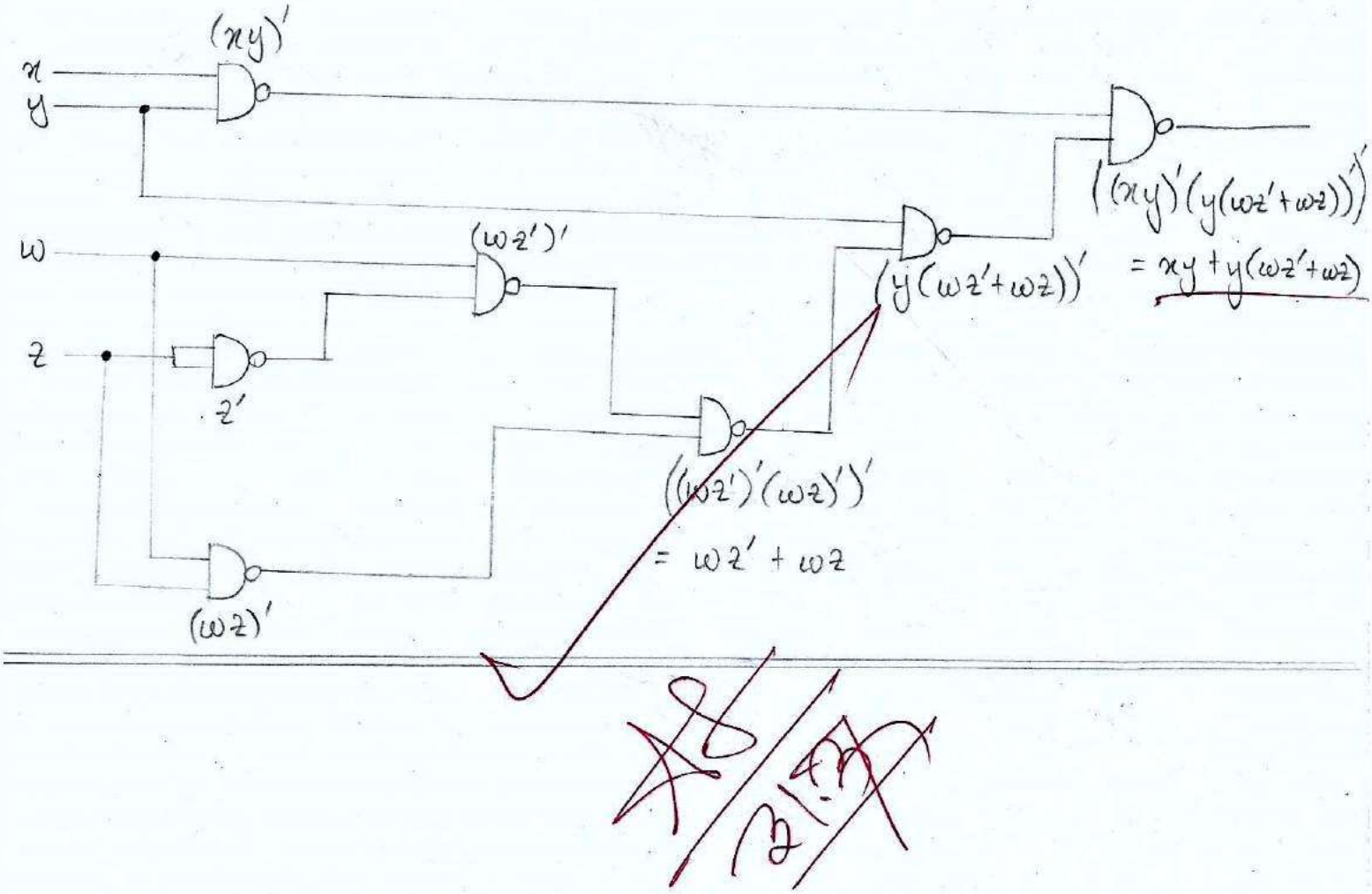


Revised Circuit Diagram of (ii)



$O = \left((xy)'(x'y')'(y'z)'\right)' = xy + x'y' + y'z$

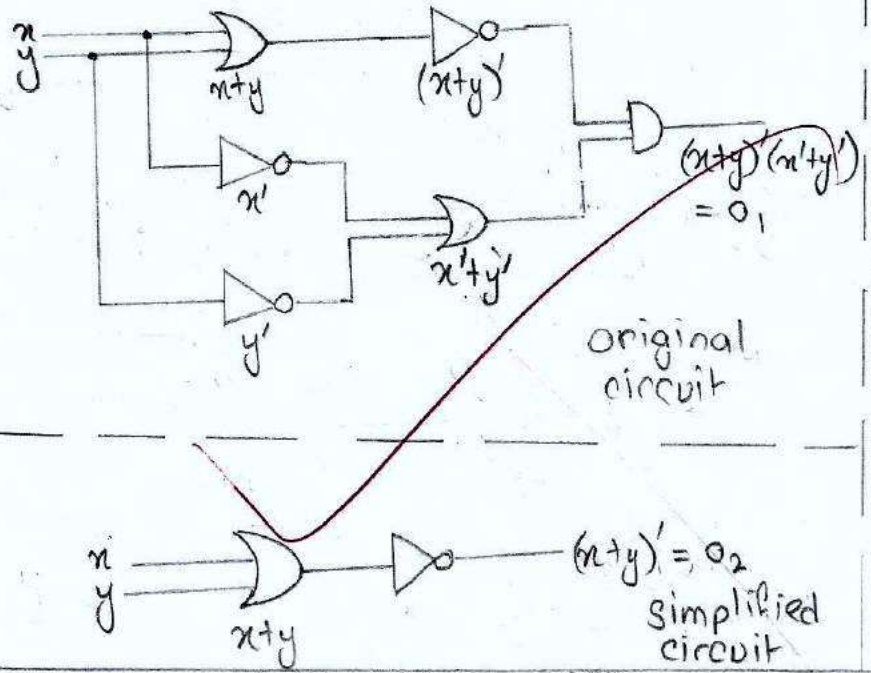
Revised Circuit Diagram of (iii)



Task 3: Simplification of Boolean expression and verification

1. Simplify the following Boolean expressions into minimum number of literals
   i. ~~$xy + x'(x + y)$~~ $(x+y)'(x'+y') = (x+y)'$
   ii. $x'z + x'y + xy'z + yz$ = $x'y + z$
2. Draw the circuit diagrams for each of the Boolean expression mentioned above in original and simplified form separately.
3. Create the corresponding truth tables for each of the Boolean expression using required number of variables both in original and simplified form.
4. Verify the result of the circuits for both forms with the their truth tables

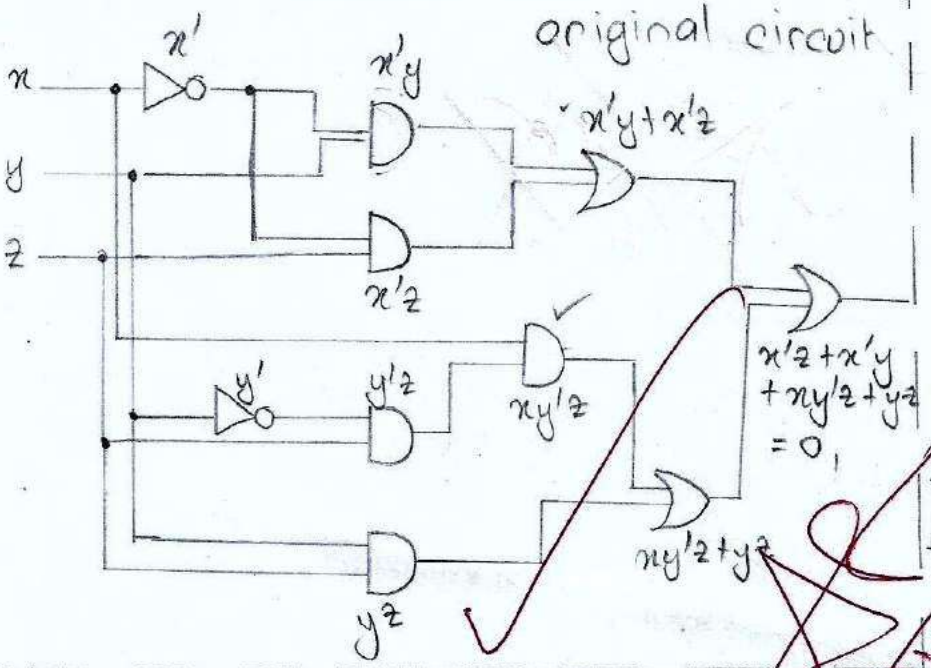
Circuit Diagrams of (i)



Truth Table of (i)

x	y	O_1	O_2
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Circuit Diagrams of (ii)

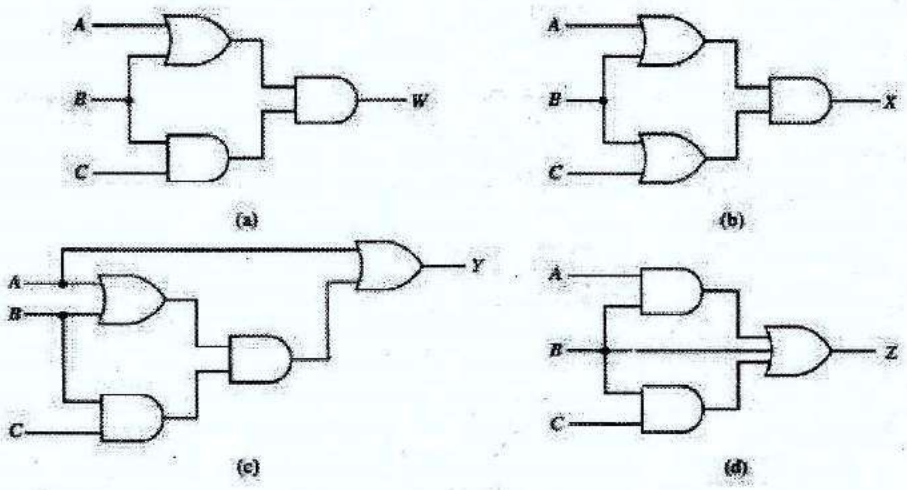


Truth Table of (ii)

x	y	z	O_1	O_2
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

Review Questions:

1. Find out the Boolean expressions for the logic circuits given bellow.



2. Simplify the Boolean expressions derived from question 1 into minimum number of literals.
3. Draw the logic circuits for the Boolean expressions derived from question 2 (in simplified form).
4. Redraw the equivalent circuit diagrams using universal logic gates (NAND or NOR) for both original and simplified forms.
5. Draw the truth tables to verify the original and simplified forms of above Boolean expressions.

Answer:

1. a) $(A+B)(BC)$ b) $(A+B)(B+C)$ c) $(A+B)(BC)+A$

d) $AB+B+BC$

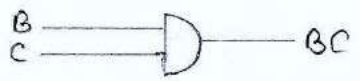
---

2. a) BC b) $AC+B$ c) $A+BC$

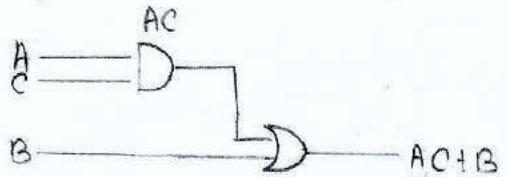
d) B

---

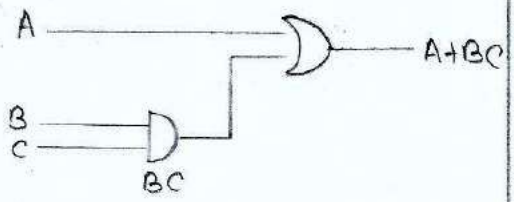
3. a)



b)

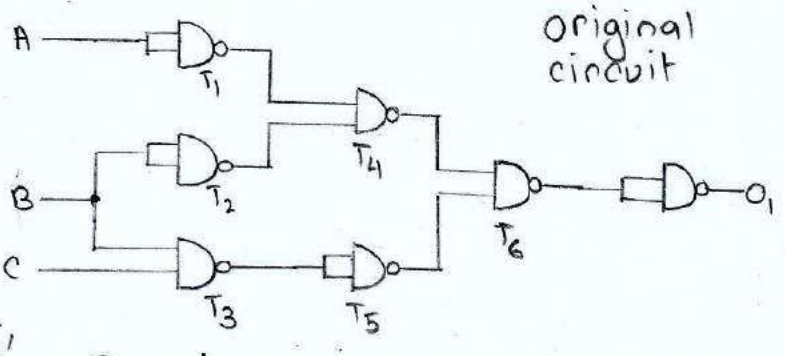


c)



d)

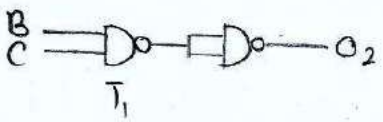
4. a) original circuit



$T_1 = A'$ $T_2 = B'$ $T_3 = BC'$ $T_4 = (A'B')' = (A+B)$

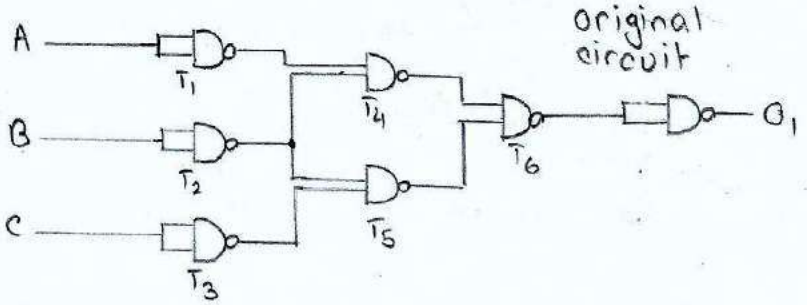
$T_5 = BC$ $T_6 = ((A+B)BC)'$ $O_1 = (A+B)(BC)$

Simplified circuit



$T_1 = (BC)'$ $O_2 = BC$

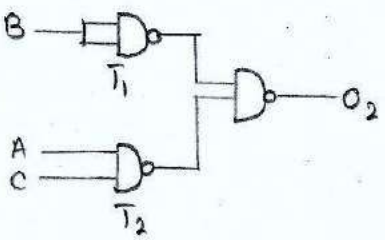
b) original circuit



$T_1 = A'$ $T_2 = B'$ $T_3 = C'$ $T_4 = (A'B')' = A+B$

$T_5 = (B'C')' = B+C$ $T_6 = ((A+B)(B+C))'$ $O_1 = (A+B)(B+C)$

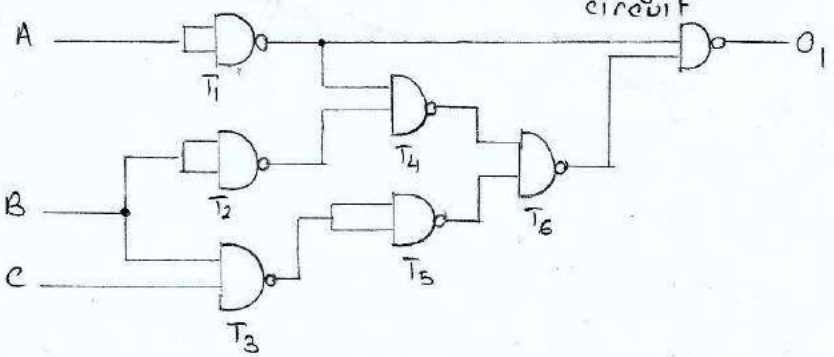
Simplified circuit



$T_1 = B'$ $T_2 = (AC)'$

$O_2 = (B'(AC)')' = B + AC$

c) original circuit

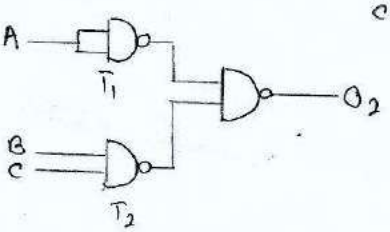


$T_1 = A'$ $T_2 = B'$ $T_3 = (BC)'$ $T_4 = (A'B')' = (A+B)$

$T_5 = BC$ $T_6 = ((A+B)BC)'$ $O_1 = (A'((A+B)BC)')'$

$= (A+B)(BC) + A$

Simplified circuit

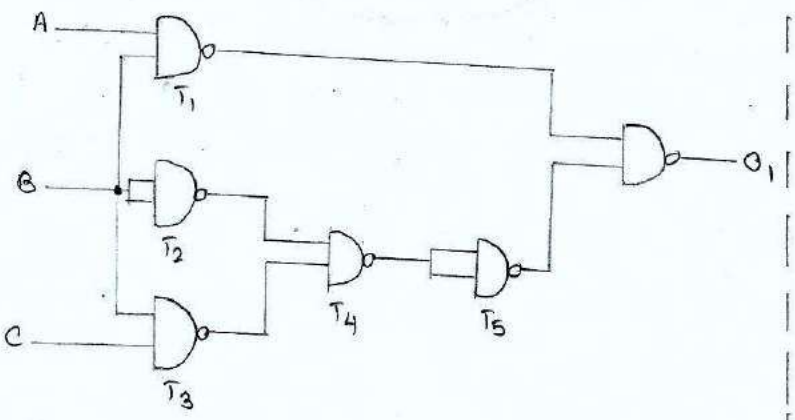


$T_1 = A'$ $T_2 = (BC)'$

$O_2 = (A'(BC)')' = A + BC$

d)

original circuit



Simplified circuit

$T_1 = (AB)'$ $T_2 = B'$ $T_3 = (BC)'$ $T_4 = (B'(BC)')'$

$T_5 = B'(BC)'$ $O_1 = ((AB)'B'(BC)')' = AB + B + BC$

$O_2 = B$

5. a)

A	B	C	O_1	O_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

b)

A	B	C	O_1	O_2
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

c)

A	B	C	O_1	O_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
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

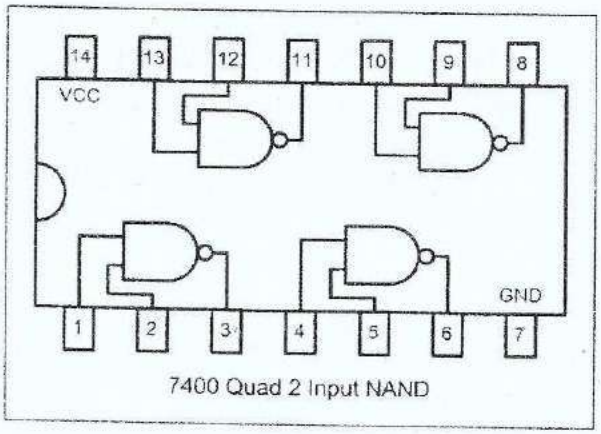
d)

A	B	C	O_1	O_2
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

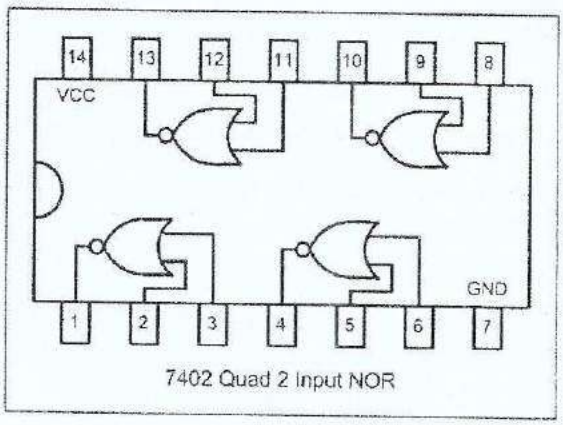
SEMESTER PROJECT

The project will be given based on the theoretical knowledge covered in EEE 4307. The project should be done in group. There will be different projects for different groups. The detail lecture on semester project will be delivered after the mid-semester examination.

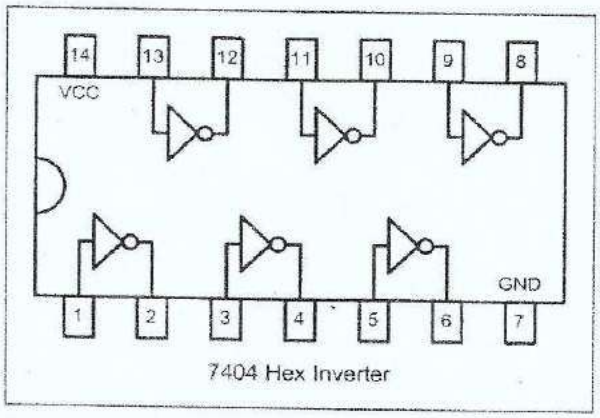
APPENDIX: Pin Configuration of Different ICs



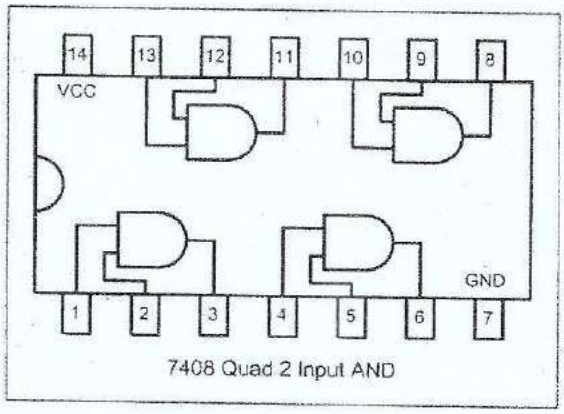
7400 Quad 2 Input NAND



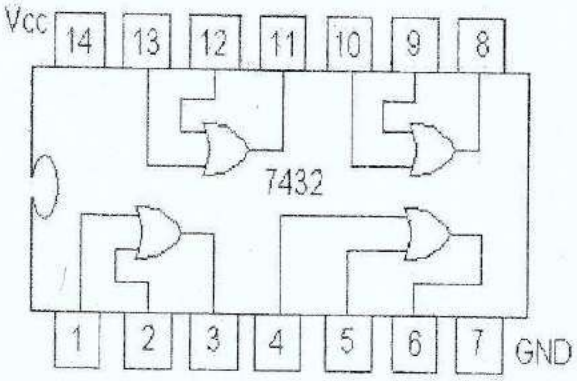
7402 Quad 2 Input NOR

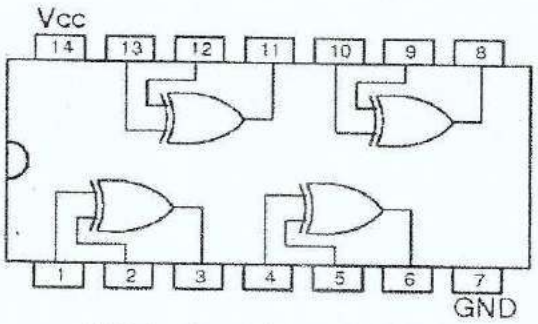


7404 Hex Inverter

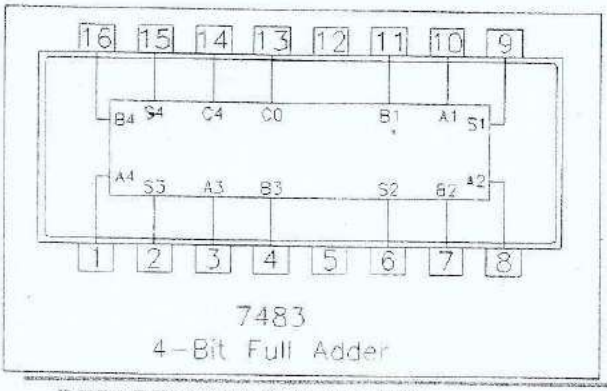


7408 Quad 2 Input AND

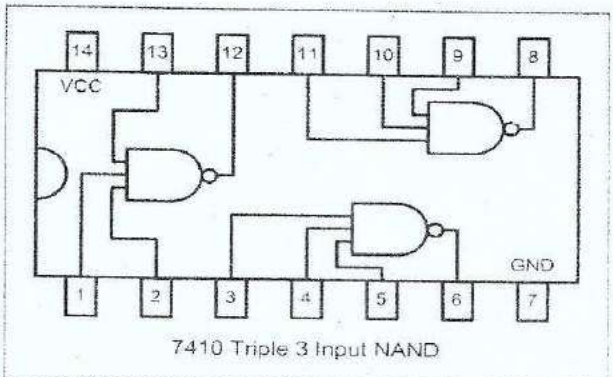




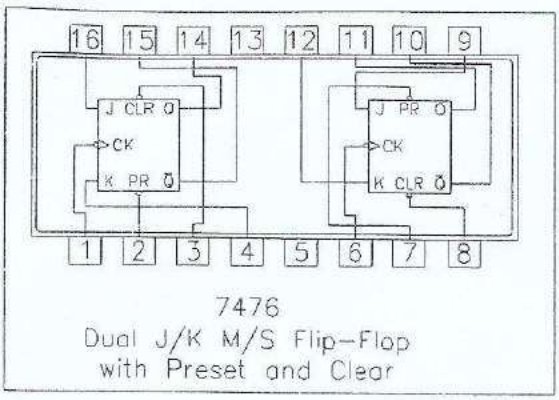
7486 Quad 2 Input XOR



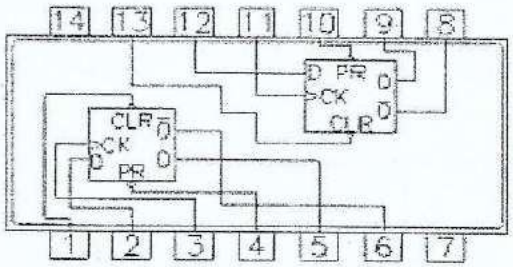
7483
4-Bit Full Adder



7410 Triple 3 Input NAND



7476
Dual J/K M/S Flip-Flop
with Preset and Clear



7474
Dual D Flip-Flop
with Preset and Clear

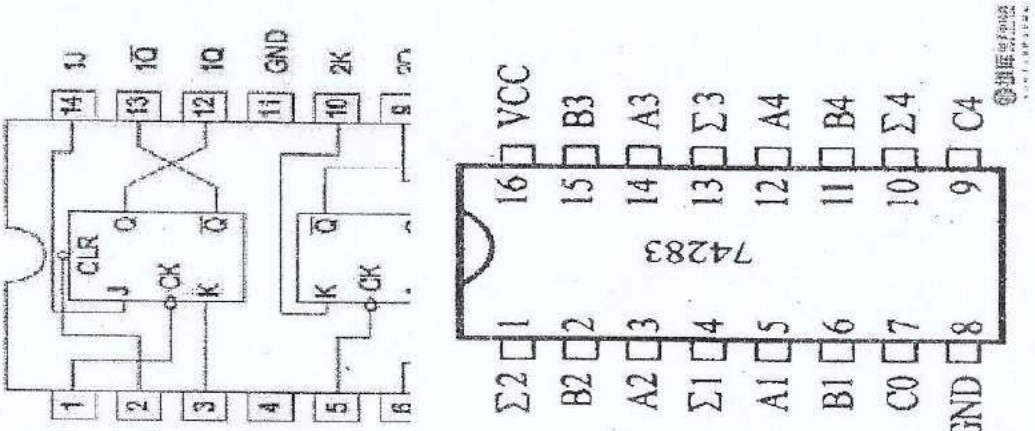


Figure: IC 7473 Dual JK Flip-Flop