



**Ahsanullah University of Science and Technology**  
**Department of Electrical and Electronic Engineering**

LABORATORY MANUAL  
FOR  
ELECTRICAL AND ELECTRONIC SESSIONAL COURSES

Student Name :  
Student ID :

Course no : EEE 3104  
Course Title: Digital Electronics-I Laboratory

For the students of  
Department of Electrical and Electronic  
Engineering  
3<sup>rd</sup> Year, 1<sup>st</sup> Semester

## Experiment: 1


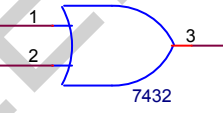
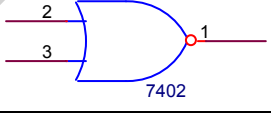
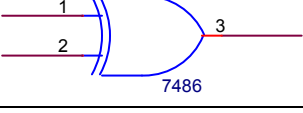
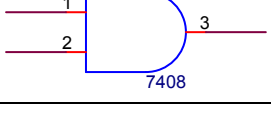
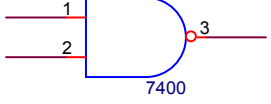
**Experiment name:** *Introduction to different digital ICs.*

---

### Introduction:

In this experiment you will be introduced to different digital ICs that will be used in this digital lab to perform different functions and also the function of each IC. You are asked to memorize the followings associated with each IC.

1. IC number
2. IC name
3. Total number of pins
4.  $V_{cc}$  pin number
5. Ground pin number

IC number	IC name	Schematic view
7404	NOT	
7432	OR	
7402	NOR	
7486	XOR	
7408	AND	
7400	NAND	

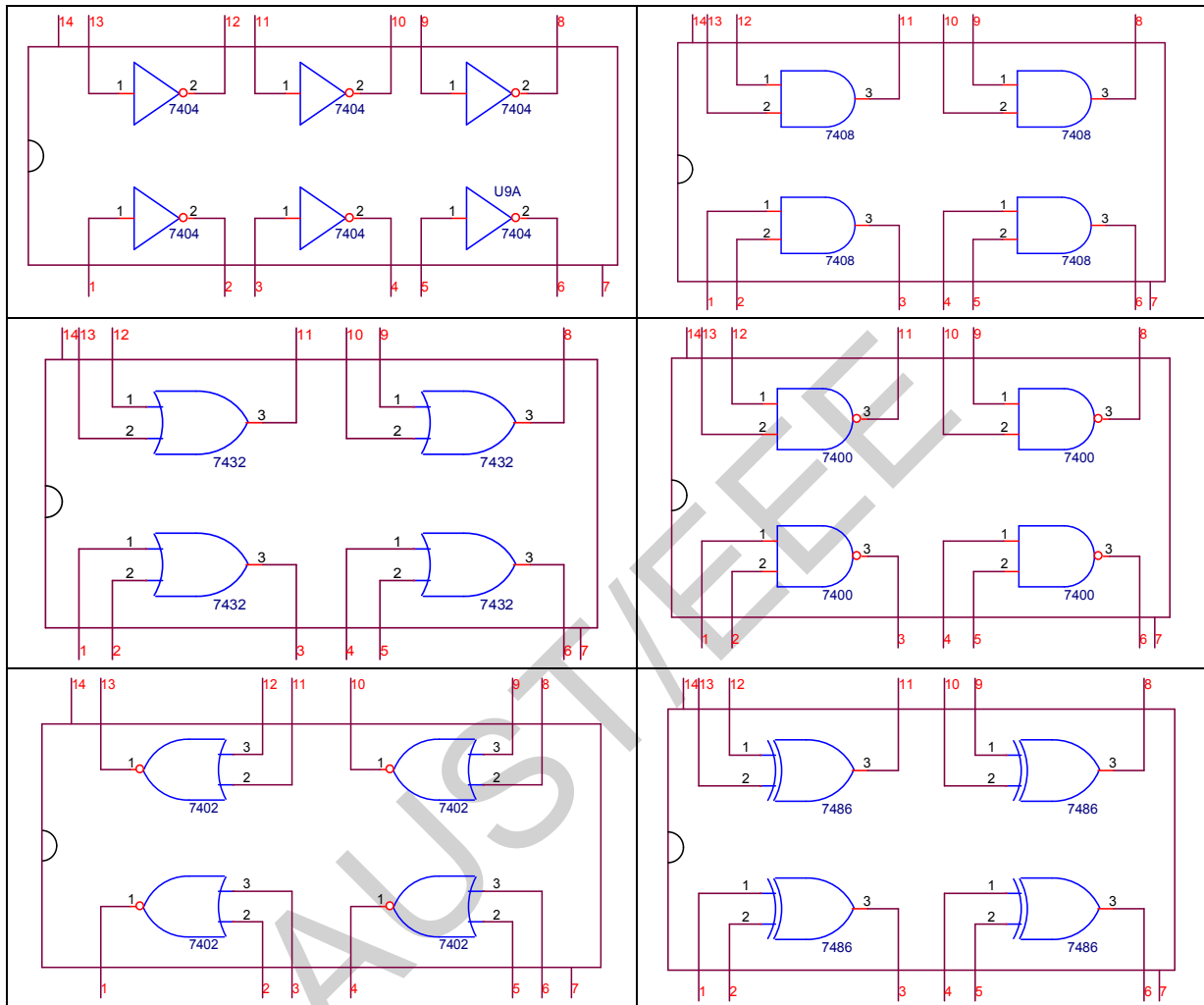
### Equipment:

1. Trainer Board
1. IC 7400,7402,7404,7408,7432,7486
2. Microprocessor Data handbook

### Procedure:

1. Take any of the ICs. From microprocessor data handbook find the name of the IC, total number of pins that it has,  $V_{cc}$  pin and ground pin.

IC Number	IC name	Total number of pin	$V_{cc}$ pin no.	Ground pin - no.
7400	NAND	14	14	7
7402	NOR	14	14	7
7404	NOT	14	14	7
7408	AND	14	14	7
7432	OR	14	14	7
7486	XOR	14	14	7



- Note the number of gates each IC has from the handbook.
- Now fill up the following table.

Input A	Input B	7400 NOT $Y = \overline{A}$	7432 OR $Y = A + B$	7402 NOR $Y = \overline{A + B}$	7486 XOR $Y = A \oplus B$	7408 AND $Y = AB$	7400 NAND $Y = \overline{AB}$
0	0						
0	1						
1	0						
1	1						

- Now verify the observed output with the desired output for different combination of inputs.
- Repeat step 1 to 4 for different ICs.

**Assignment:**

- How can you make a three input AND/OR/XOR gate with a two input AND/OR/XOR gate?
- Is it possible to make a three input NAND/NOR gate with a two input NAND/OR gate? Justify your answer.

## Experiment: 2

Experiment name: *Introduction to Combinational logic.*

### Introduction:

Logic design basically means the construction of appropriate function, presented in Boolean algebraic form, then edification of the logic diagram, and finally choosing of available ICs and implementing the IC connection so that the logic intended is achieved. The efficiency in simplifying the algebra leads to less complicated logic diagram, which in the end leads to easier IC selection and easier circuit implementation.

### Caution:

1. Remember to properly identify the pin numbers so that no accidents occur.
2. Properly bias the ICs appropriate voltages to appropriate pins.

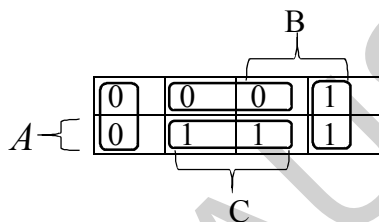
### Equipment:

1. Trainer Board
2. IC 7400,7402,7404,7408,7432,7486
3. Microprocessor Data handbook

### Job 1:

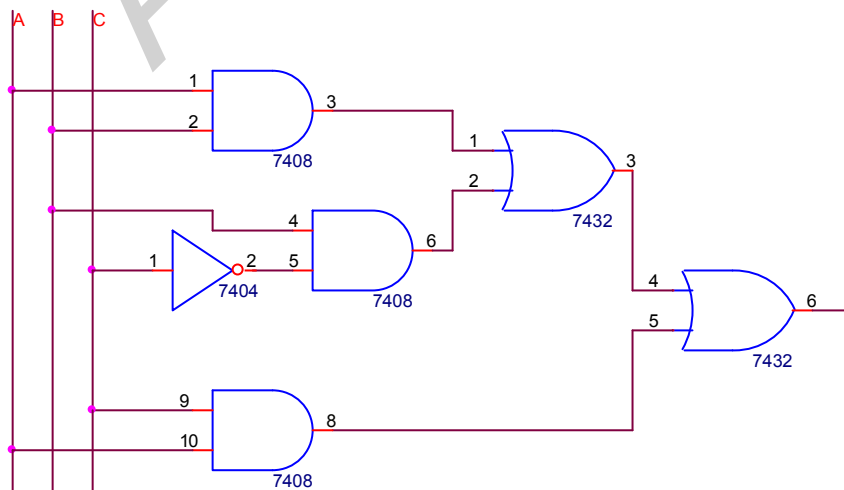
Implement of function  $f = AB + BC' + CA$

$$\begin{aligned} &= ABC + ABC' + ABC' + A'BC' + ABC + AB'C \\ &= ABC + ABC' + A'BC' + AB'C \\ &= m_7 + m_6 + m_2 + m_5 \end{aligned}$$



$$\begin{aligned} F &= AC + BC' \\ &= \text{SOP} \end{aligned}$$

$$\begin{aligned} F' &= B'C' + A'C \\ \Rightarrow F &= (B'C' + A'C)' \\ \Rightarrow F &= (B + C)(A + C') \end{aligned}$$



### Procedure:

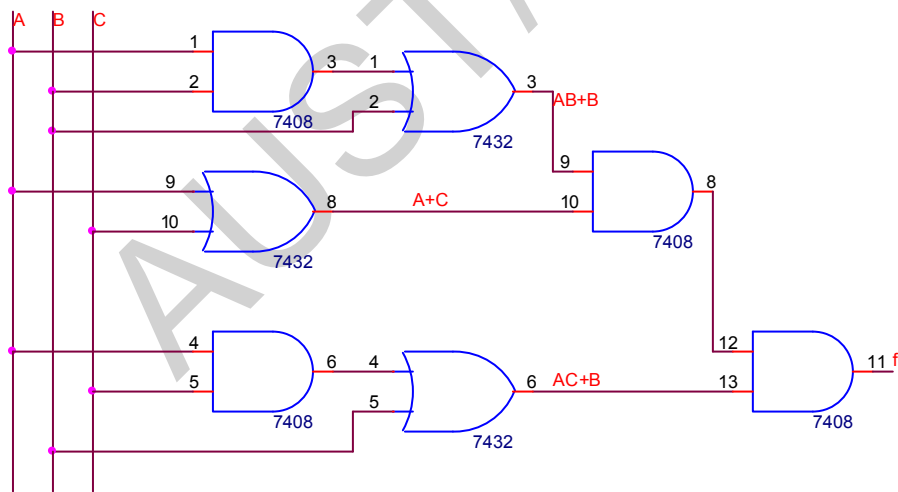
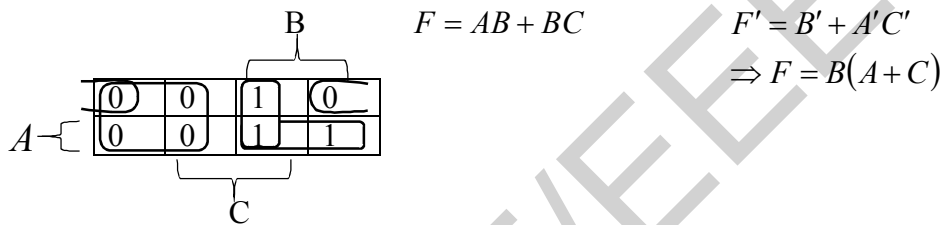
1. Draw logic diagram to implement the function.
2. Select ICs from the equipment list.

3. Note the output logic for all combination of inputs.
4. Now fill up the truth table for that function.
5. Simplify the function in POS and in SOP form using K-map.
6. Repeat step-1, 2 and 3.

**Job 2:**

Implement of function  $f = (AB + B)(C + A)(AC + B)$

$$\begin{aligned}
 &= B(A + B)(A + C)(A + B)(B + C) \\
 &= B(A + B)(A + C)(B + C) \\
 &= B(AA')(A + B)(A + C)(B + C) \\
 &= (A + B)(A' + B)(A + C)(B + C) \\
 &= (A + B + C)(A + B + C')(A' + B + C)(A' + B + C')(A + B' + C)(A + B + C)(A' + B + C) \\
 &= (A + B + C)(A + B + C')(A' + B + C)(A' + B + C')(A + B' + C) \\
 &= M_0 M_1 M_4 M_2 M_5
 \end{aligned}$$



**Procedure:**

1. Simplify the function in POS form and in SOP form by using Boolean algebra.
2. Draw logic diagram to implement the function.
3. Select ICs from the equipment list.
4. Note the output logic for all combination of inputs.

**Experiment: 3**

**Experiment name:** *Construction of adders and sub tractors using basic logic gates.*

---

**Introduction:**

Adders and sub tractors are the basic operational units of simple digital arithmetic operations. In this experiment, the students will construct the basic adder and sub tractor circuit with common logic gates and test their operability. Then in the last job, they will cascade adder ICs to get higher bit adders.

**Caution:**

1. Remember to properly identify the pin numbers so that no accidents occur.
2. Properly bias the ICs with appropriate pins.

**Equipment:**

1. Trainer Board
2. IC 7400, 7402, 7404, 7408, 7432, 7486
3. Microprocessor Data handbook

**Job 1:**

*Implementation of a half adder and a half sub tractor.*

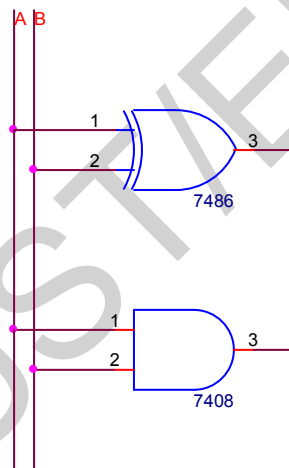


Fig: Half Adder

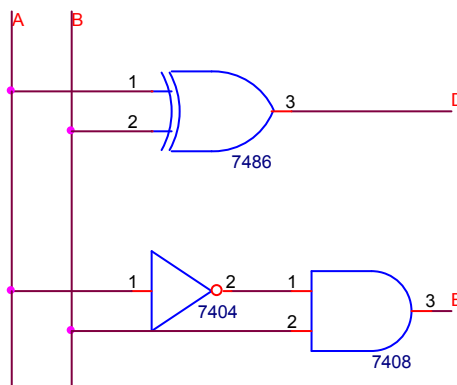


Fig: Half sub tractor

**Procedure:**

1. Fill up the truth table for a half adder.

A	B	C	S

$$S = A \oplus B$$

$$C = AB$$

- Determine the Boolean function for a half adder.
- Construct the logic diagram from the Boolean functions.
- Select the ICs from the equipment list.
- Implement the output logic and compare with step-1.
- Repeat the whole procedure for half a sub tractor.

A	B	B	D

$$D = A \oplus B$$

$$B = A'B$$

**Job 2:**

*Implement of a full adder and a full sub tractor.*

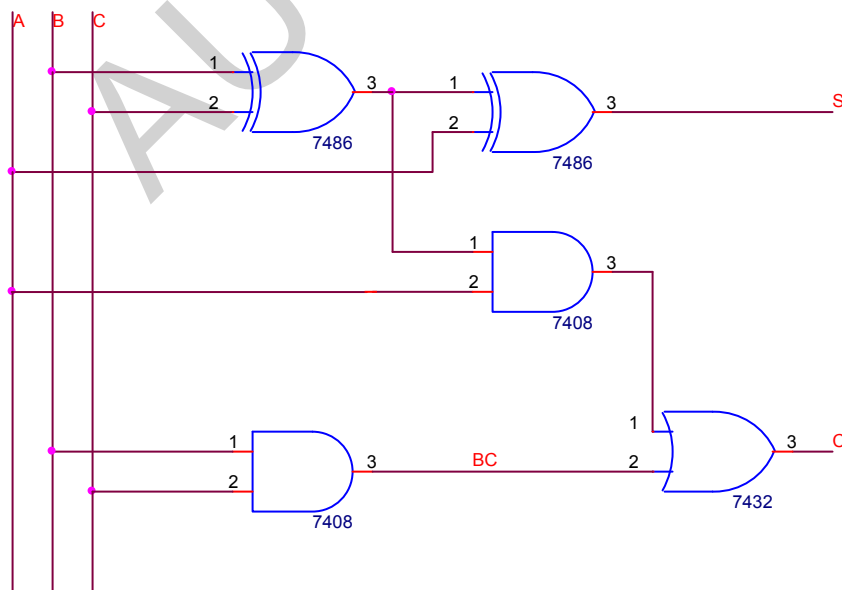


Fig: full adder

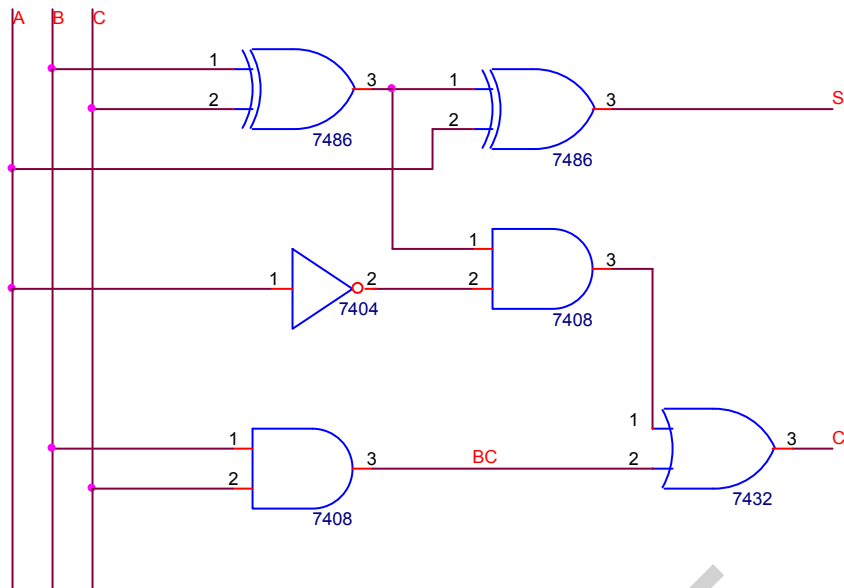


Fig: full subtractor

**Procedure:**

1. Fill up the truth table for a full adder.

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

$$S = A'B'C + A'BC' + AB'C' + ABC$$

$$\Rightarrow S = A \oplus B \oplus C$$

$$C = A'BC + AB'C + ABC' + ABC$$

$$\Rightarrow C = BC + A(B \oplus C)$$

2. Determine the Boolean function for a full adder.
3. Construct the logic diagram from the Boolean functions.
4. Select the ICs from the equipment list.
5. Implement the output logic and compare with step-1.
6. Repeat the whole procedure for a full sub tractor.
7. Now draw a full adder using two half adder block and basic gates.
8. Repeat step-7 for a full sub tractor.

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

$$S = A'B'C + A'BC' + AB'C' + ABC$$

$$\Rightarrow S = A \oplus B \oplus C = D$$

$$C = A'BC + AB'C + ABC' + ABC$$

$$\Rightarrow C = BC + A'(B \oplus C)$$



**Experiment: 4**

**Experiment name:** Design a Combinational circuit that will act as an Adder if control bit is '0' and as a sub tractor if control bit is '1'.

**Introduction:**

Addition of two 4-bit binary numbers can be easily done using a 4-bit binary adder IC (7483/74283). Taking the 2's complement of the subtrahend and then adding that with the minuend can do subtraction of two 4-bit binary numbers.

**Caution:**

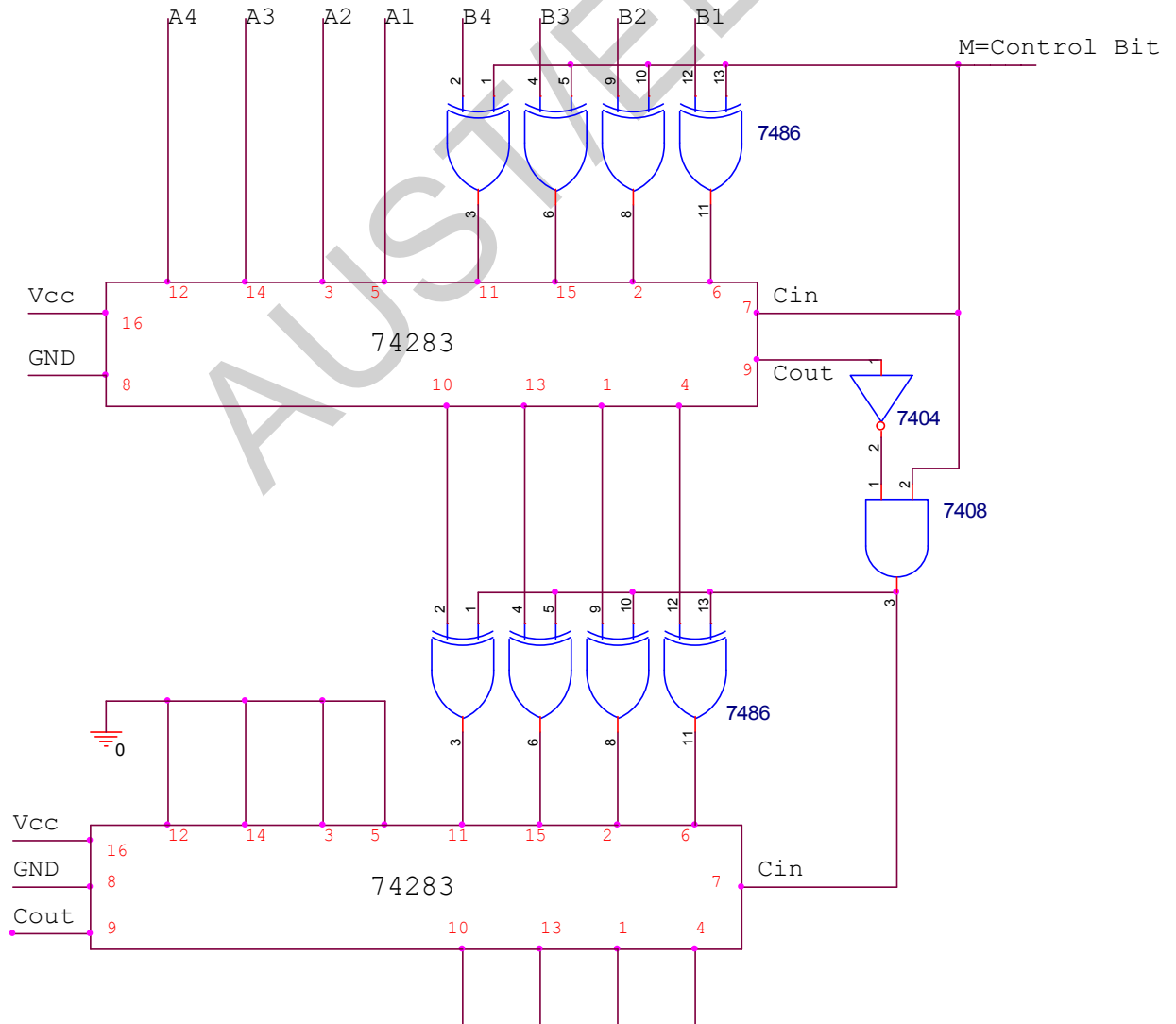
- 1. Remember to properly identify the pin numbers so that no accidents occur.
- 2. Properly bias the ICs with appropriate voltages to appropriate pins.

**Equipment:**

- 1. Trainer Board
- 2. IC 74283,7408,7432,7486.
- 3. Microprocessor Data handbook

**Procedure:**

- 1. Draw the logic diagram to implement the task.



2. Select the required ICs.
3. Note the output logic for different inputs.

<i>A</i>	<i>B</i>	<i>S</i>	Output
0001	0010	0	
0001	0010	1	
1001	0011	0	
1011	0111	1	
1011	0111	0	
1100	1001	1	
1111	1111	0	
1111	1111	1	

AUST/E/E/E

## Experiment: 5

**Experiment name:** Design a BCD adder that will add two BCD numbers and the sum will be also in BCD.

### Introduction:

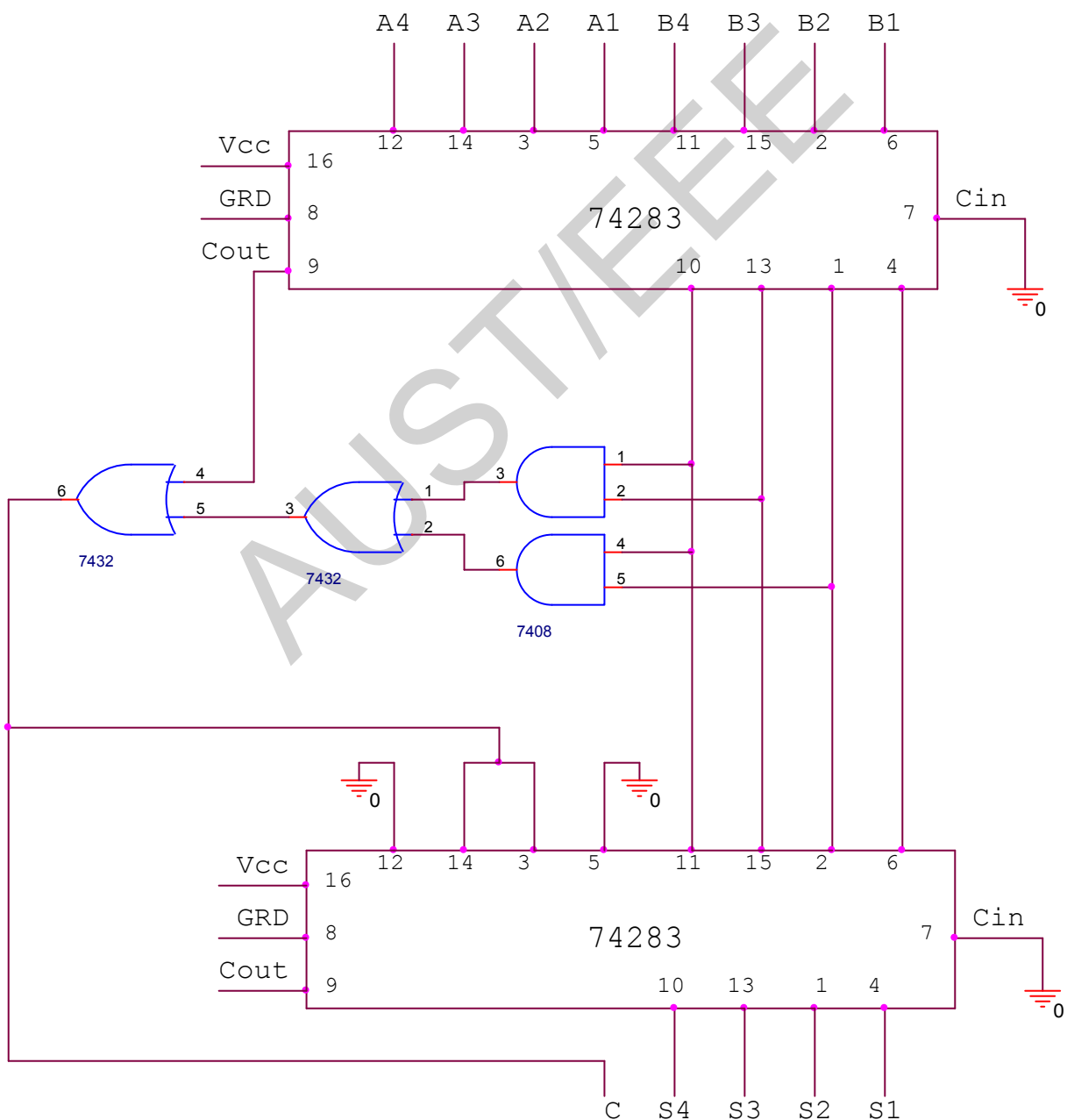
A BCD adder can be designed by considering the arithmetic addition of two decimal digits in BCD, together with a possible carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than  $9+9+1=19$ . This can be designed with a 4-bit binary adder together with a correction logic circuit.

### Caution:

1. Remember to properly identify the pin numbers so that no accidents occur.
2. Properly bias the ICs with appropriate voltages to appropriate pins.

### Procedure:

1. Draw the logic diagram to implement the task.



2. Select the required ICs.
3. Fill up the following truth table for 19 inputs.

Decimal	Binary Sum					BCD Sum				
	$K$	$Z_8$	$Z_4$	$Z_2$	$Z_1$	$C$	$S_8$	$S_4$	$S_2$	$S_1$
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										

## Experiment: 6

Experiment name: *Introduction to Multiplexers.*

### Introduction:

Multiplexers are the most important attributions of digital circuitry in communication hardware. These digital switches enable us to achieve the communication network we have today. In this experiment the students will have to construct MUX (as they call multiplexers) with simple logic gates and they will implement general logic using 8:1 MUX as the basic constructional unit.

### Caution:

1. Remember to properly identify the pin numbers so that no accidents occur.
2. Properly bias the ICs with appropriate voltages to appropriate pins.

### Equipment:

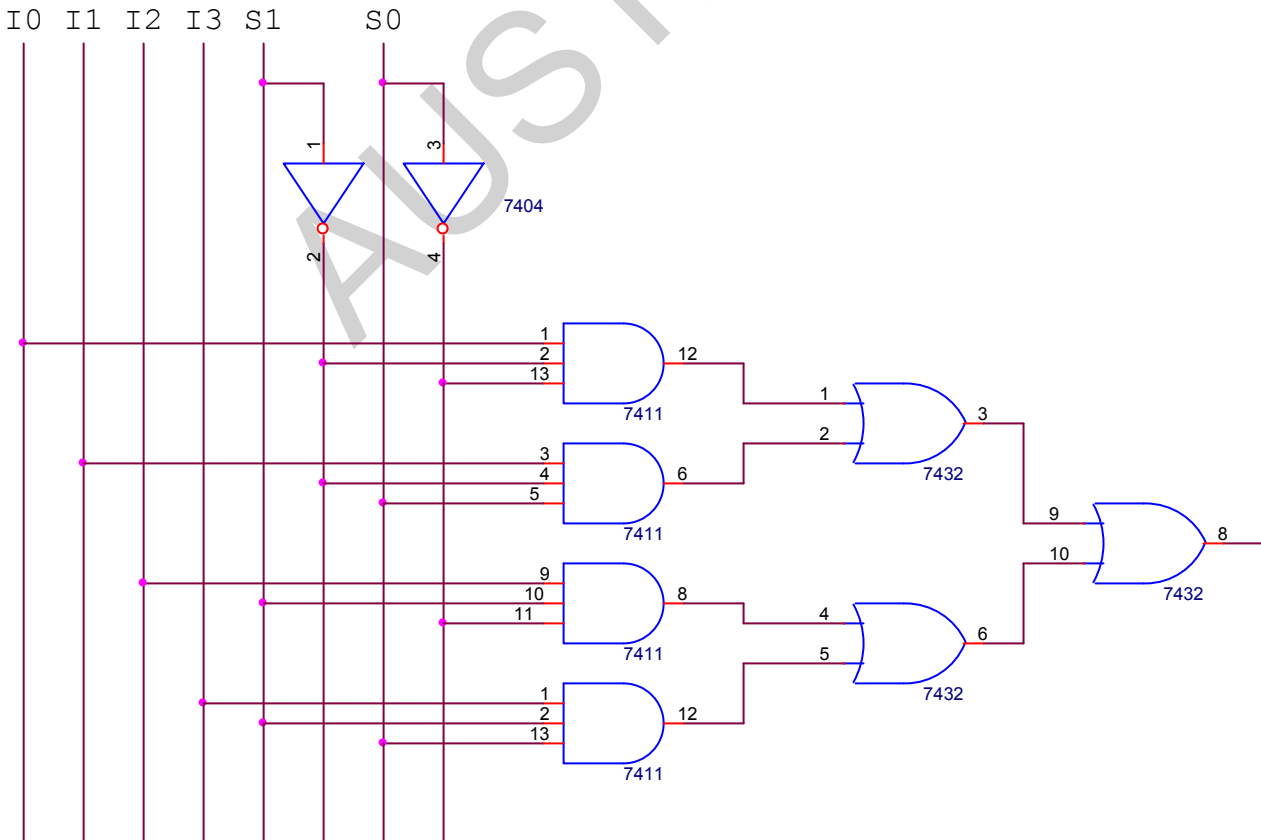
1. Trainer Board
2. IC 74151, 7432, 7408, 7404
3. Microprocessor Data handbook.

### Job 1:

*Implementation of a four to one way Multiplexer, (4:1 MUX) with basic gates.*

### Procedure:

1. Write the truth table for four to one way MUX.
2. Write the Boolean function for the output logic.
3. Draw the logic diagram to implement the Boolean function.



4. Select ICs from the equipment list.
5. Observe and note the output logic for all combination of inputs.

**Job 2:**

Implement the following function using an 8:1 MUX.

$$F(A, B, C, D) = \sum(0, 1, 3, 5, 8, 9, 14, 15)$$

**Procedure:**

1. Write the truth table for the above function.

	$I_0$	$I_1$	$I_2$	$I_3$	$I_4$	$I_5$	$I_6$	$I_7$
$A'$								
$A$								

2. Draw the logic diagram to implement the Boolean function.

$A$	$B$	$C$	$D$	$Y$

3. Select ICs from the equipment list.
4. Observe and note the output logic for all combination of inputs.

**Assignment:**

1. Implement a Full Adder using an 8:1 MUX.
2. Repeat 1 using two 4:1 MUX and basic gates.
3. How can you implement a 4:1 MUX using only three 2:1 MUX?

**Experiment: 7****Experiment name:** *Implementation of Demultiplexers and Priority Encoders.*

---

**Introduction:**

A Demultiplexer does the opposite function of multiplexers. It has one input line and  $2^n$  output lines, where  $n$  is the number of selection bits. The output channel can be selected depending on the combination of selection bits. An encoder has  $2^n$  input lines and  $n$  output line. A priority encoder is designed to give output for lowest/highest input lines. For example, If D3, D2 and D1 lines have '1' as in their inputs, the output would be '11' as priority is given to highest input line.

**Caution:**

1. Remember to properly identify the pin numbers so that no accidents occur.
2. Properly bias the ICs with appropriate voltages to appropriate pins.

**Equipment:**

1. Trainer Board
2. IC 7432, 7408, 7404
3. Microprocessor Data handbook.

**Job 1:**

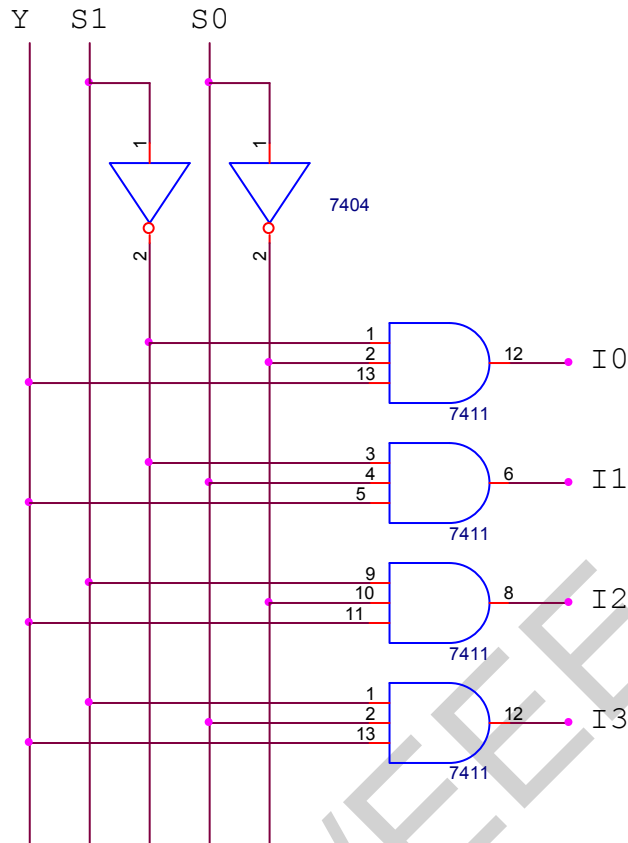
*Implementation of a one to four way Demultiplexer, (1:4 DEMUX) with basic gates.*

**Procedure:**

1. Write the truth table for one to four way DEMUX.

$S_1$	$S_0$	$I_0$	$I_1$	$I_2$	$I_3$

2. Write the Boolean function for the output logic.
3. Draw the logic diagram to implement the Boolean function.



4. Select ICs from the equipment list.
5. Observe and note the output logic for all combination of inputs.

**Job 2:**

*Implement a 4×2 priority encoder with basic gates.*

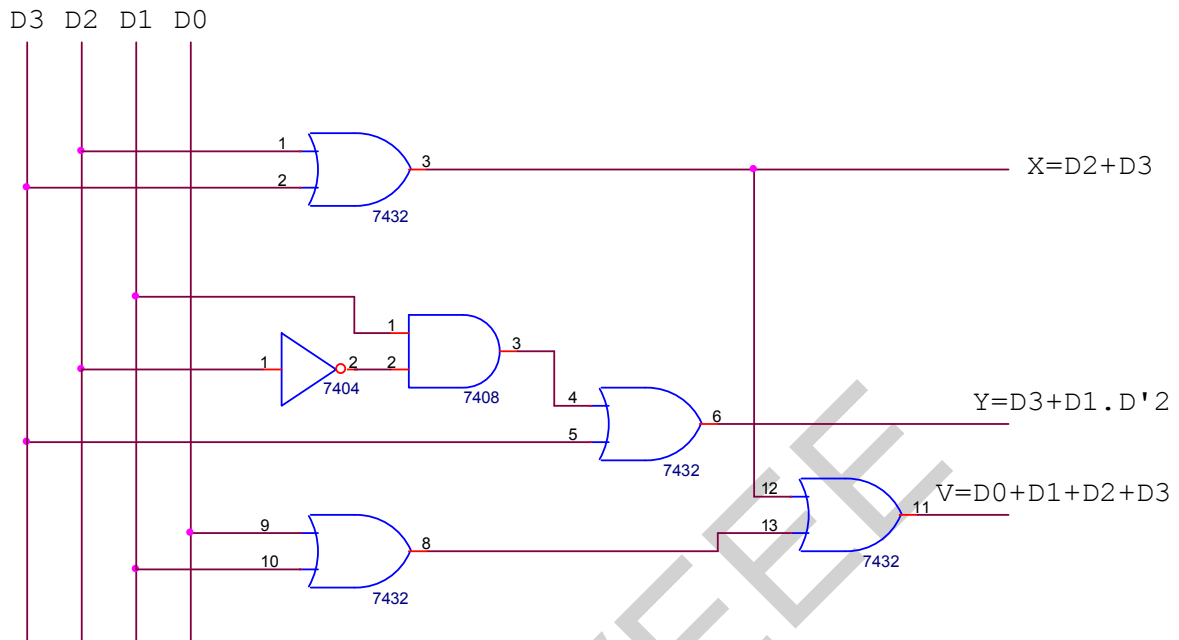
**Procedure:**

1. Write the truth table for 4×2 priority encoder.

$D_3$	$D_2$	$D_1$	$D_0$	$X$	$Y$	$V$



2. Write the Boolean function for the output logic.
3. Simplify the Boolean function using K-map.
4. Draw the logic diagram to implement the simplified Boolean function.



5. Select ICs from the equipment list.
6. Observe and note the output logic for all combination of inputs.

**Experiment: 8**

**Experiment name:** *Design of Flip-flop using basic gates.*

---

**Caution:**

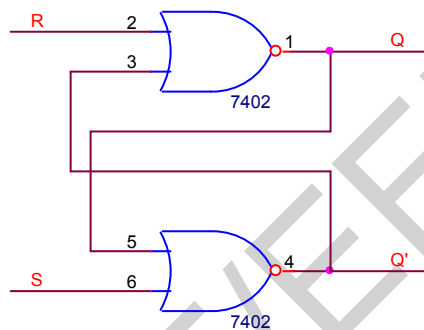
1. Remember to properly identify the pin numbers so that no accidents occur.
2. Properly bias the ICs with appropriate voltages to appropriate pins.

**Equipment:**

1. Trainer Board
2. IC 7400, 7402, 7432, 7408, 7404
3. Microprocessor Data handbook

**Job 1:**

*Design of an SR Flip-flop using NOR gates only.*



**Procedure:**

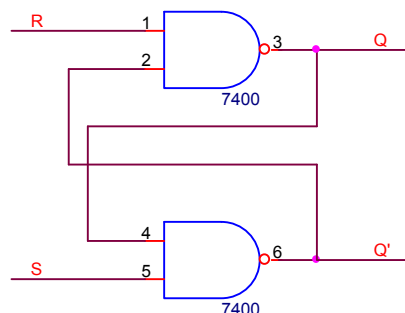
1. Draw the logic diagram to implement SR Flip-flop.
2. Fill up the table with different combination of inputs.

S	R	Q	Q'
1	0		
0	0		
0	1		
0	0		
1	1		

3. Observe the combination for which no change and invalid or race conditions arise.

**Job 2:**

*Design of an SR Flip-flop using NAND gates only.*



**Procedure:**

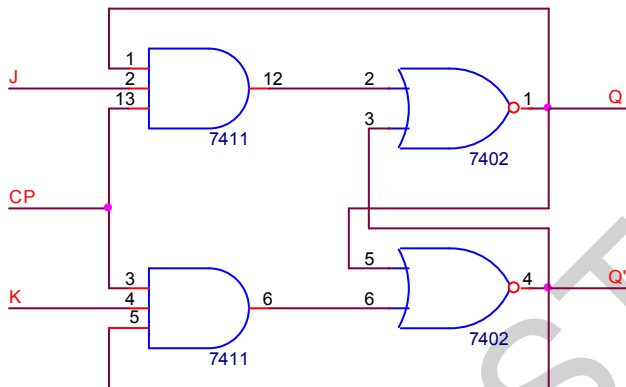
1. Draw the logic diagram to implement SR Flip-flop.
2. Fill up the table with different combination of inputs.

$S$	$R$	$Q$	$Q'$
1	0		
0	0		
0	1		
0	0		
1	1		

3. Observe the combination for which no change and invalid or race conditions arise.

**Job 3:**

*Design of a J-K Flip-flop using AND & NOR gate only.*



**Procedure:**

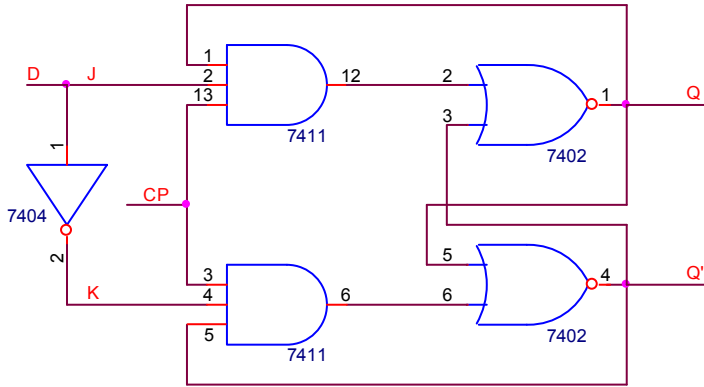
1. Draw the logic diagram to implement J-K Flip-flop.
2. Fill up the table with different combination of inputs.

$Q$	$J$	$K$	$Q(t+1)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

3. Observe the combination for which no change and invalid or race conditions arise.

**Job 4:**

*Design of a D Flip-flop from a J-K Flip-flop.*



**Procedure:**

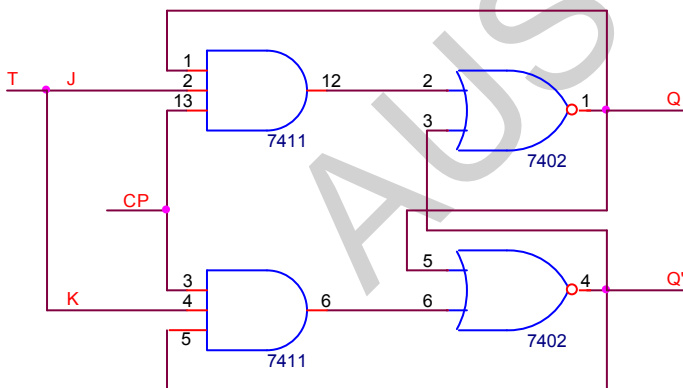
1. Draw the logic diagram to implement D Flip-flop.
2. Fill up the table with different combination of inputs.

$Q$	$D$	$Q(t+1)$
0	0	
0	1	
1	0	
1	1	

3. Observe the combination for which no change and invalid or race conditions arise.

**Job 5:**

*Design of a T Flip-flop from a J-K Flip-flop.*



**Procedure:**

1. Draw the logic diagram to implement T Flip-flop.
2. Fill up the table with different combination of inputs.

$Q$	$T$	$Q(t+1)$
0	0	
0	1	
1	0	
1	1	

3. Observe the output logic.