EEE - 2286

Digital Electronics

AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT
OF
ELECTRICAL AND ELECTRONIC ENGINEERING

EEE - 2286

Digital Electronics-1 Lab

Edition 2021

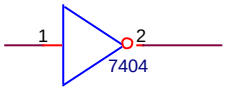
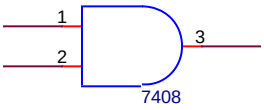
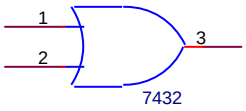
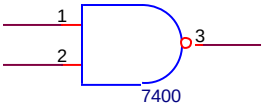
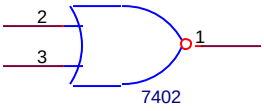
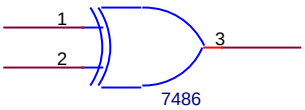
Table of Contents

Experiment: 1
Experiment name: Introduction to different digital ICs.

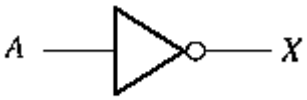
Introduction:

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/INVERTER	
7408	AND	
7432	OR	
7400	NAND	
7402	NOR	
7486	XOR	

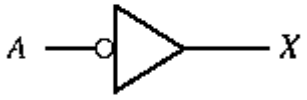
The INVERTER/NOT Gate



A	X
0	1
1	0

$X = \overline{A}$

Boolean expression



Distinctive shape symbols

Truth table

0 = LOW
1 = HIGH

The output of an inverter is always the complement (opposite) of the input.

The AND Gate

Distinctive shape symbol

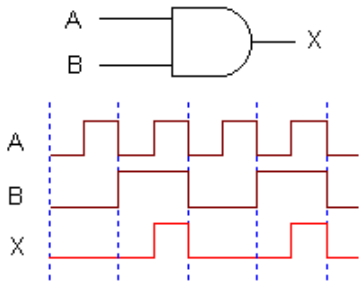
$X = AB$

Boolean expression

B	A	X
0	0	0
0	1	0
1	0	0
1	1	1

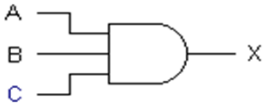
Truth table

0 = LOW
1 = HIGH



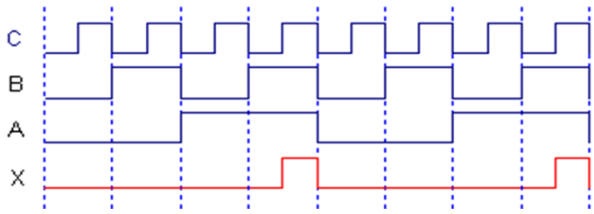
Pulsed Waveforms

The output of an AND gate is HIGH only when all inputs are HIGH.



X = ABC

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



3 Input AND Gate

The OR Gate

Distinctive shape symbol

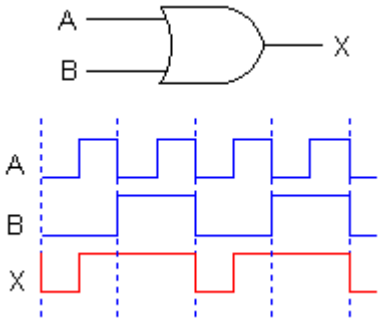
$X = A + B$

Boolean expression

B	A	X
0	0	0
0	1	1
1	0	1
1	1	1

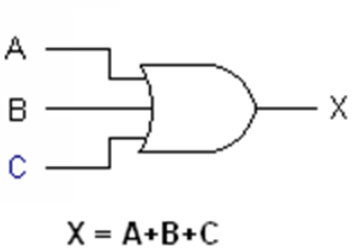
Truth table

0 = LOW
1 = HIGH



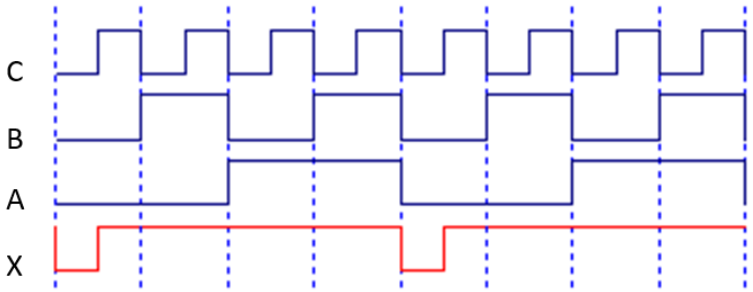
Pulsed Waveforms

The output of an OR gate is HIGH whenever one or more inputs are HIGH.



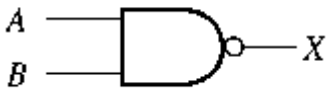
$X = A+B+C$

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

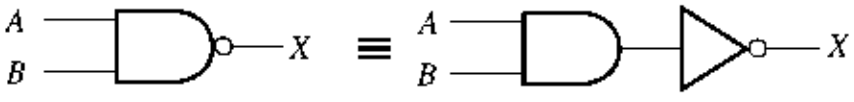


3 Input OR Gate

The NAND Gate



Distinctive shape symbol



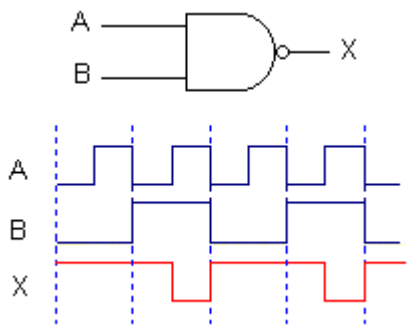
B	A	X
0	0	1
0	1	1
1	0	1
1	1	0

Truth table

0 = LOW
1 = HIGH

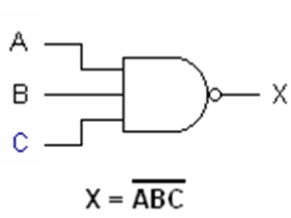
$X = \overline{AB}$

Boolean expression



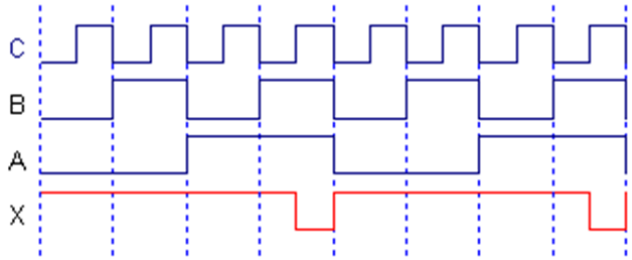
Pulsed Waveforms

The output of a NAND gate is HIGH whenever one or more inputs are LOW.



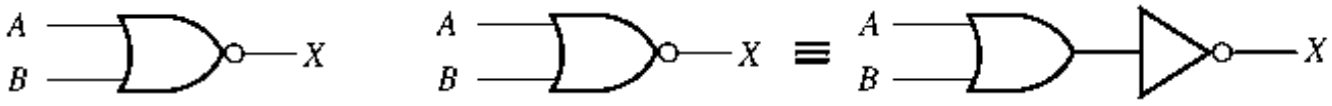
$X = \overline{ABC}$

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



3 Input NAND Gate

The NOR Gate



Distinctive shape symbol

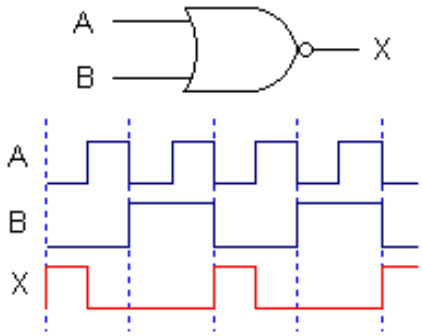
B	A	X
0	0	1
0	1	0
1	0	0
1	1	0

Truth table

0 = LOW
1 = HIGH

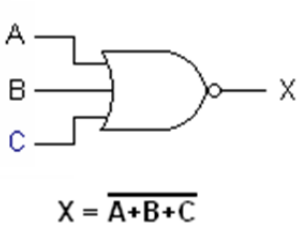
$X = \overline{A + B}$

Boolean expression



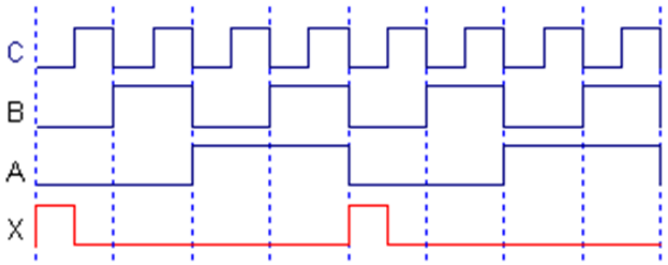
Pulsed Waveforms

The output of a NOR gate is LOW whenever one or more inputs are HIGH.



$X = \overline{A+B+C}$

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



3 Input NOR Gate

Exclusive-OR Gate

Distinctive shape symbol

B	A	X
0	0	0
0	1	1
1	0	1
1	1	0

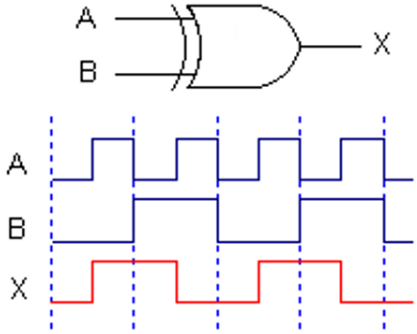
Truth table

0 = LOW
1 = HIGH

$X = A \oplus B$

Boolean expression

The output of an XOR gate is HIGH whenever the two inputs are different.



Pulsed Waveforms

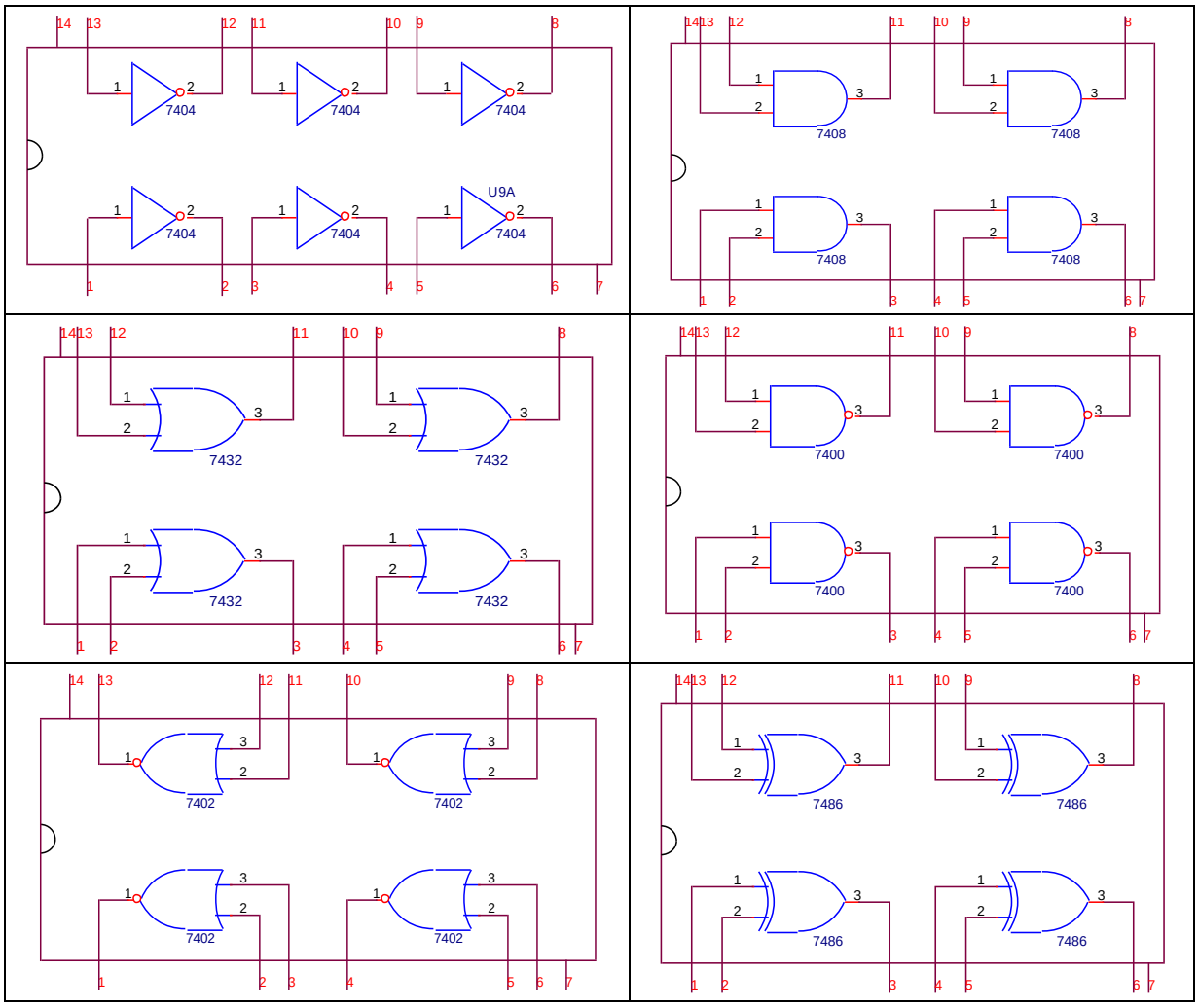
Equipment:

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

Procedure:

1. Take any of the following 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



2. Note the number of gates each IC has from the handbook.
3. 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						

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

Report:

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

Experiment: 2
Experiment name: Introduction to Combinational logic and K map minimization.

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 $f = AB + BC' + CA$

$$= ABC + ABC' + ABC' + A'BC' + ABC + AB'C$$
$$= ABC + ABC' + A'BC' + AB'C$$
$$= m_7 + m_6 + m_2 + m_5$$
$$= \sum m(2,5,6,7)$$

Truth Table

Row no.	Input			Output
	A	B	C	*f*
0	0	0	0	0
1	0	0	1	0
2	0	1	0	**1**
3	0	1	1	0
4	1	0	0	0
5	1	0	1	**1**
6	1	1	0	**1**
7	1	1	1	**1**

K-Map

A \ BC	00	01	11	10
0	0	0	0	**1**
1	0	**1**	**1**	**1**

AC **BC'**

SOP: $F = AC + BC'$

POS: $F' = B'C' + A'C$

$\Rightarrow F = (B'C' + A'C)'$

$\Rightarrow F = (B + C)(A + C')$

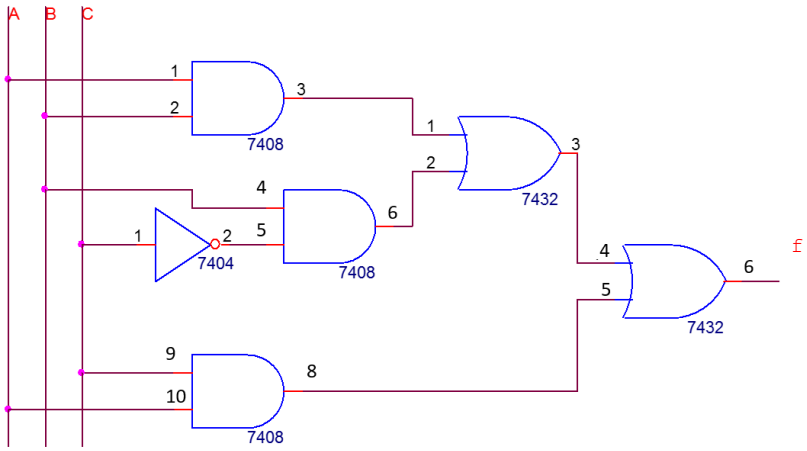


Figure 2.1: Logic Diagram of Job 1

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. Repeat step-1, 2 and 3 for SOP and POS function.

Job 2:

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

Now,

$$f = (AB + B)(C + A)(AC + B)$$
$$= B(A+1)(A + B)(A + C)(A + B)(B + C)$$
$$= B(A + B)(A + C)(B + C)$$
$$= (B + AA')(A + B)(A + C)(B + C)$$
$$= (A + B)(A' + B)(A + B)(A + C)(B + C)$$
$$= (A + B)(A' + B)(A + C)(B + C)$$
$$= (A + B + CC')(A' + B + CC')(A + C + BB')(B + C + AA')$$
$$= (A + B + C)(A + B + C')(A' + B + C)(A' + B + C')(A + B + C)(A + B' + C)(A + B + C)(A' + B + C)$$
$$= (A + B + C)(A + B + C')(A' + B + C)(A' + B + C')(A + B' + C)$$
$$= M_0 M_1 M_4 M_2 M_5$$
$$= \prod M\ (0,1,2,4,5)$$

Distributive Law

$x + yz = (x + y)(x + z)$

Truth Table

Row no.	Input			Output
	A	B	C	*f*
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

K-MAP

A \ BC	00	01	11	10
0	0	0	**1**	0
1	0	0	**1**	**1**

BC **AB**

SOP: $F = AB + BC$

POS: $F' = B' + A'C'$

$\Rightarrow F = B(A + C)$

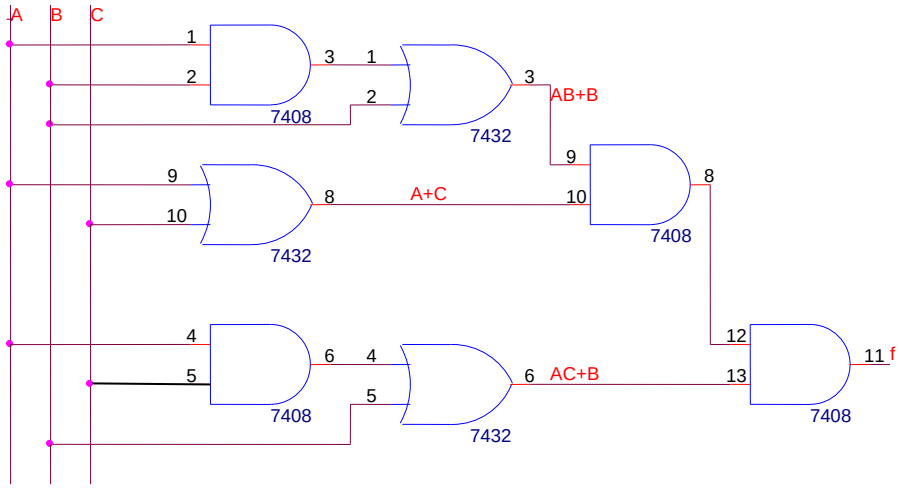


Figure 2.2: Logic Diagram of Job 2

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, 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.

Binary Adder
Among the basic 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, namely, 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, and 1 + 1 = 10. The first three operations produce a sum whose length is one digit, but when both augend and addend bits are equal to 1, the binary sum consists of two digits. The higher significant bit of this result is called a *carry.* When the augend and addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher-order pair of significant bits. A combinational circuit that performs the addition of two bits is called a *half-adder.* One that performs the addition of three bits (two significant bits and a previous carry) is *full-adder.*

Half Adder
From the basic understanding of a half-adder, we find that the circuit needs two binary inputs and two binary outputs. The input variables designate the augend and addend bits; the output variables produce the sum and carry. It is necessary to specify two output variables because the result may consist of two binary digits. We arbitrarily assign symbols *x* and *y* to the two inputs and S (for sum) and C (for carry) to the outputs.
Now that we have established the number and names of the input and output variables, we are ready to formulate a truth table to identify exactly the function of the half-adder. This truth table is

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The carry output is 0 unless both inputs are 1. The S output represents the least significant bit of the sum.
The simplified Boolean functions for the two outputs can be obtained directly from the truth table. The simplified sum of products expressions are

$$S = x'y + xy' = x \oplus y$$

$$C = xy$$

The logic diagram for this implementation is shown below

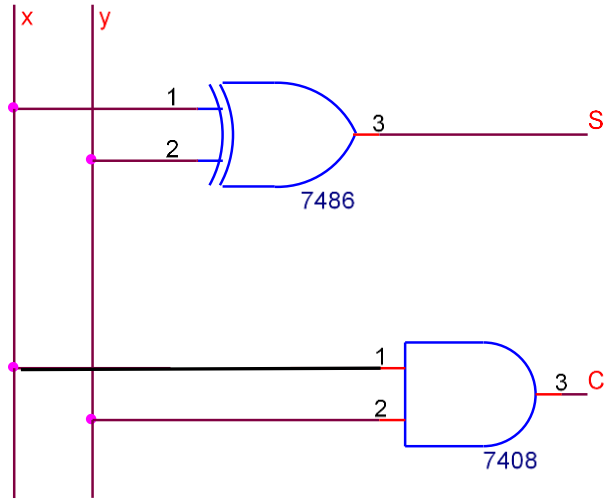


Fig 3.1. Half-adder

Full Adder

A full-adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by *x* and *y*, represent the two significant bits to be added. The third input, z, represents the carry from the previous lower significant position. Two outputs are necessary because the arithmetic sum of three binary digits ranges in value from 0 to 3, and binary 2 or 3 needs two digits. The two outputs are designated by the symbols S for sum and C for carry. The binary variable S gives the value of the least significant bit of the sum. The binary variable Cr gives the output carry. The truth table of the full-adder is

x	y	z	C	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

The eight rows under the input variables designate all possible combinations of 1's and 0's that these variables may have. The 1's and 0's for the output variables are determined from the arithmetic sum of the input bits. When all input bits are 0's, the output is 0. The S output is equal to 1 when only one input is equal to 1 or when all three inputs are equal to 1. The C output has a carry of 1 if two or three inputs are equal to 1. Physically, the binary signals of the input wires are considered binary digits added arithmetically to form a two-digit sum at the output wires. On the other hand, the same binary values are considered variables of Boolean functions when expressed in the truth table or when the circuit is implemented with logic gates. It is important to realize that two different interpretations are given to the values of the bits encountered in this circuit. The input-output logical relationship of the full-adder circuit may be expressed in two

Boolean functions, one for each output variable. This implementation uses the following Boolean expressions:

$$S = x'y'z + x'yz' + xy'z' + xyz = z'(x'y + xy') + z(x'y' + xy) = z'(x \oplus y) + z(x \oplus y)' = x \oplus y \oplus z$$
$$C = x'yz + xy'z + xyz' + xyz = x(y'z + yz') + yz(x + x') = x(y \oplus z) + yz$$

The logic diagram for the full-adder implemented in sum of products is shown in Fig. 2.2

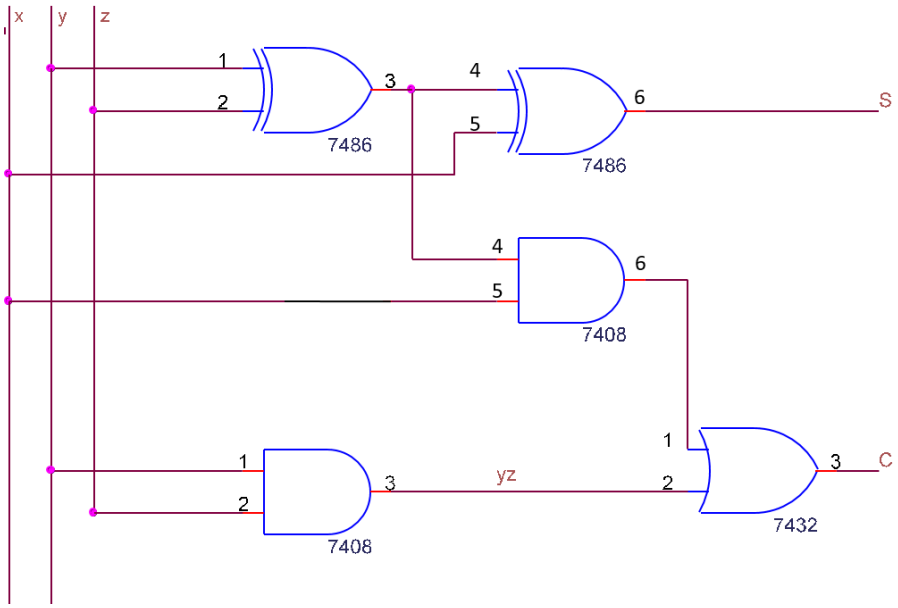


Fig 3.2. Full-adder

Half Subtractor

A half-subtractor is a combinational circuit that subtracts two bits and produces their difference. It also has an output to specify if a 1 has been borrowed. Designate the minuend bit by *X* and the subtrahend bit by *y*. To perform *x* - *y*, we have to check the relative magnitudes of *x* and *y*. If *x* '3 *y*, we have three possibilities: 0 - 0 = 0,
1 - 0 = 1, and 1 - 1 = O. The result is called the *difference bit.* If *x* < *y*, we have 0 - 1, and it is necessary to borrow a 1 from the next higher stage. The 1 borrowed from the next higher stage adds 2 to the minuend bit, just as in the decimal system a borrow adds 10 to a minuend digit. With the minuend equal to 2, the difference becomes
2 - 1 = 1. The half-subtractor needs two outputs. One output generates the difference and will be designated by the symbol *D.* The second output, designated *B* for borrow, generates the binary signal that informs the next stage that a 1 has been borrowed.
The truth table for the input-output relationships of a half-subtractor can now be derived as follows:

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

The Boolean functions for the two outputs of the half-subtractor are derived directly from the truth table:

$$D = x'y + xy' = x \oplus y$$

$$B = x'y$$

It is interesting to note that the logic for D is exactly the same as the logic for output S in the half-adder.

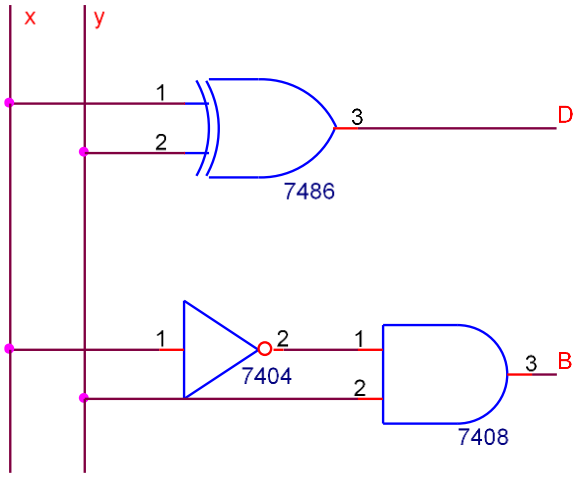


Fig 3.3. Half-subtractor

Full Subtractor

A full-subtractor is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage. This circuit has three inputs and two outputs. The three inputs, *x*, *y*, and z, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, *D* and *B*, represent the difference and output borrow, respectively. The truth table for the circuit is

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D = x'y'z + x'yz' + xy'z' + xyz = x'(y'z + yz') + x(y'z' + yz) = x'(y \oplus z) + x(y \oplus z)' = x \oplus y \oplus z$$

$$B = x'y'z + x'yz' + x'yz + xyz = x'(y'z + yz') + yz(x' + x) = x'(y \oplus z) + yz$$

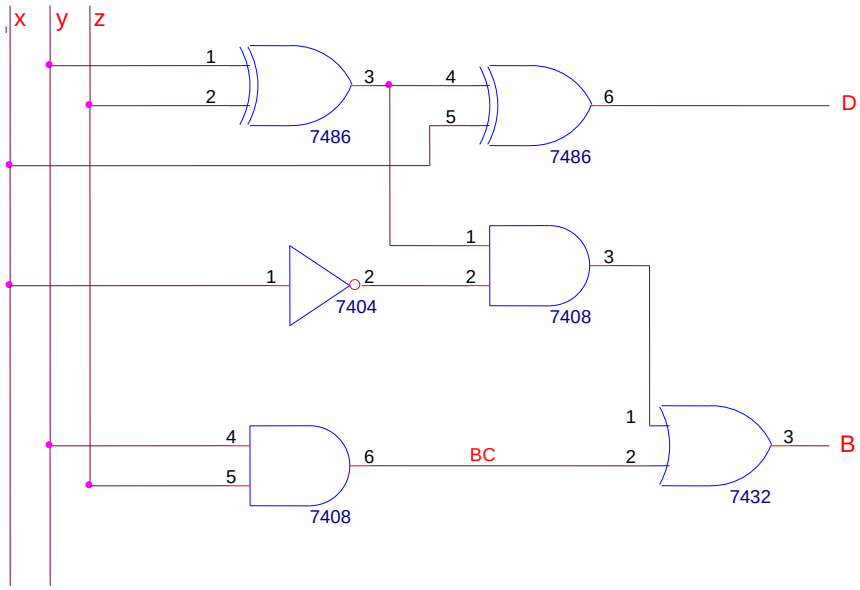


Fig 3.4. Full-subtractor

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

Procedure:

1. Fill up the truth table for a half adder
2. Verify the Boolean function for a half adder.
3. Construct the logic diagram from the Boolean functions.
4. Select the ICs from the equipment list.
5. Implement the output logic.
6. Repeat the whole procedure for half a subtractor.
7. Fill up the truth table for a full adder.
8. Verify the Boolean function for a full adder.
9. Construct the logic diagram from the Boolean functions.
10. Select the ICs from the equipment list.
11. Implement the output logic.
12. Repeat the whole procedure for a full sub tractor.

Report

1. Design a full adder using two half adder block and basic gates.

Experiment: 4
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.

Multiplexer
A modern home stereo system may have a switch that selects music from one of four sources: a cassette tape, a compact disc (CD), a radio tuner, or an auxiliary input such as audio from a VCR or DVD. The switch selects one of the electronic signals from one of these four sources and sends it to the power amplifier and speakers. In simple terms, this is what a **multiplexer (MUX)** does: it selects one of several input signals and passes it on to the output.

A digital multiplexer or data selector is a logic circuit that accepts several digital data inputs and selects one of them at any given time to pass on to the output. The routing of the desired data input to the output is controlled by SELECT inputs (often referred to as ADDRESS inputs). Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.
A 4 to l line multiplexer is shown in Figure. Each of the four input lines, I_0 to I_3 is applied to one input of an AND gate. Selection lines S_1 and S_0 are decoded to select a particular AND gate. The function table, Figure lists the input-to-output path for each possible bit combination of the selection lines. To demonstrate the circuit operation, consider the case when $S_1S_0 = 10$. The AND gate associated with input I_2 has two of its inputs equal to 1 and the third input connected to I_2. The other three AND gates have at least one input equal to 0, which makes their outputs equal to 0. The OR gate output is now equal to the value of I_2 thus providing a path from the selected input to the output.

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.

S_1	S_0	Y

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

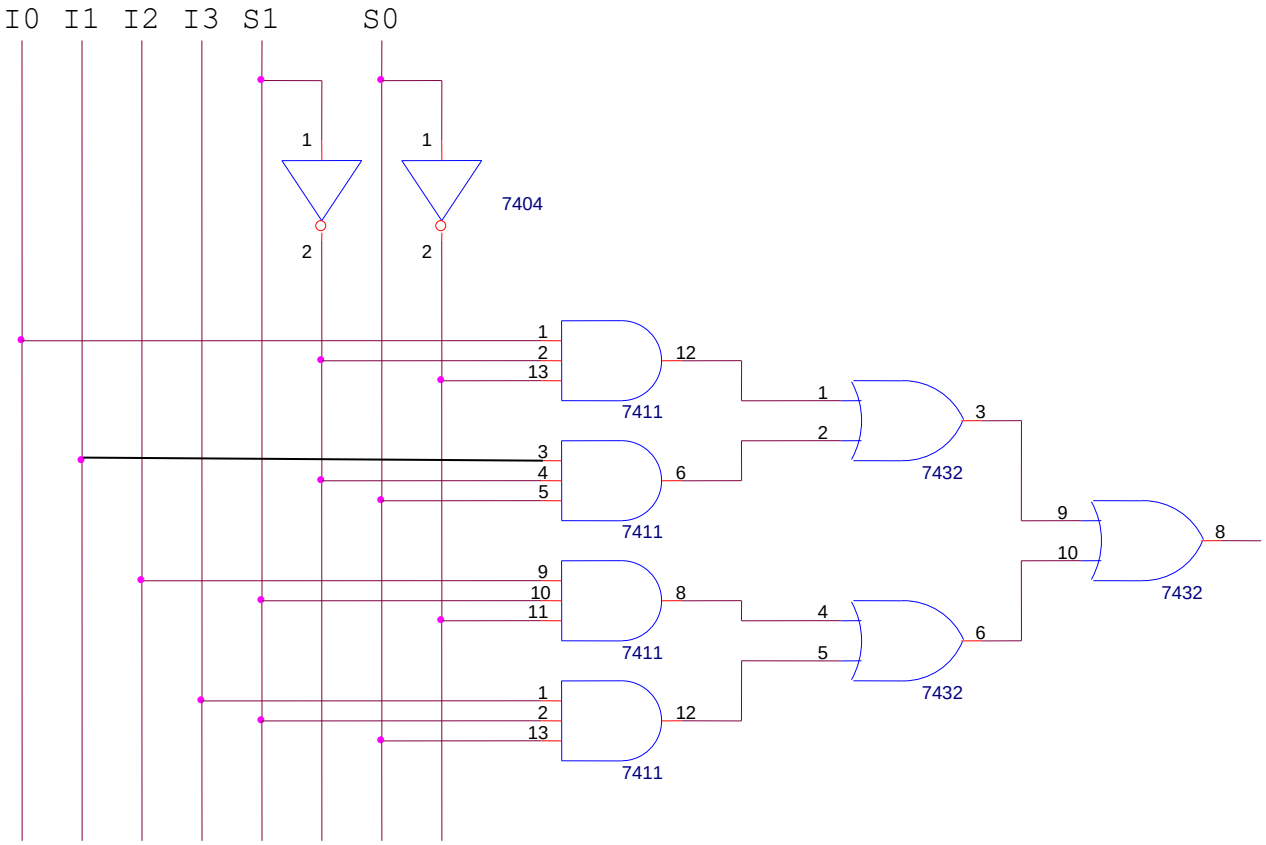


Figure 4.1: 4 to 1 Multiplexer

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.

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

If we have a Boolean function of $n + 1$ variables, we take n of these variables and connect them to the selection lines of a multiplexer. The remaining single variable of the function is used for the inputs of the multiplexer. If A is this single variable, the inputs of the multiplexer are chosen

to be either *A* or *A'* or 1 or 0. By judicious use of these four values for the inputs and by connecting the other variables to the selection lines, one can implement any Boolean function with a multiplexer. In this way, it is possible to generate any function of $n + 1$ variables with a 2^n to1 multiplexer.

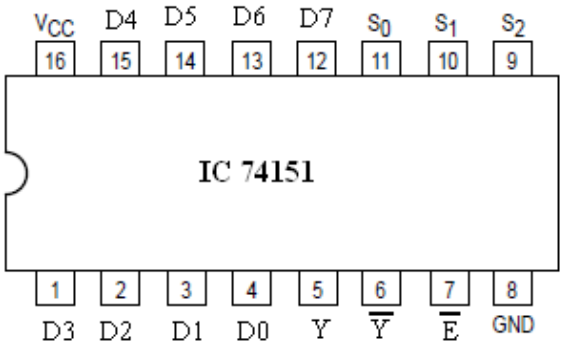


Fig 4.2. Pin diagram of IC 74151

Procedure:

1. Write the truth table for the above function.

A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
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	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Let, B, C, D of the 4 variables (A, B, C, D) are connected to the selection lines of a multiplexer and remaining single variable A of the function is used for the inputs of the multiplexer

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	1	1		1		1		
A	1	1					1	1
	1	**1**	**0**	A'	**0**	A'	A	A

2. Draw the logic diagram to implement the Boolean function.
3. Select ICs from the equipment list.
4. Observe and note the output logic for all combination of inputs.