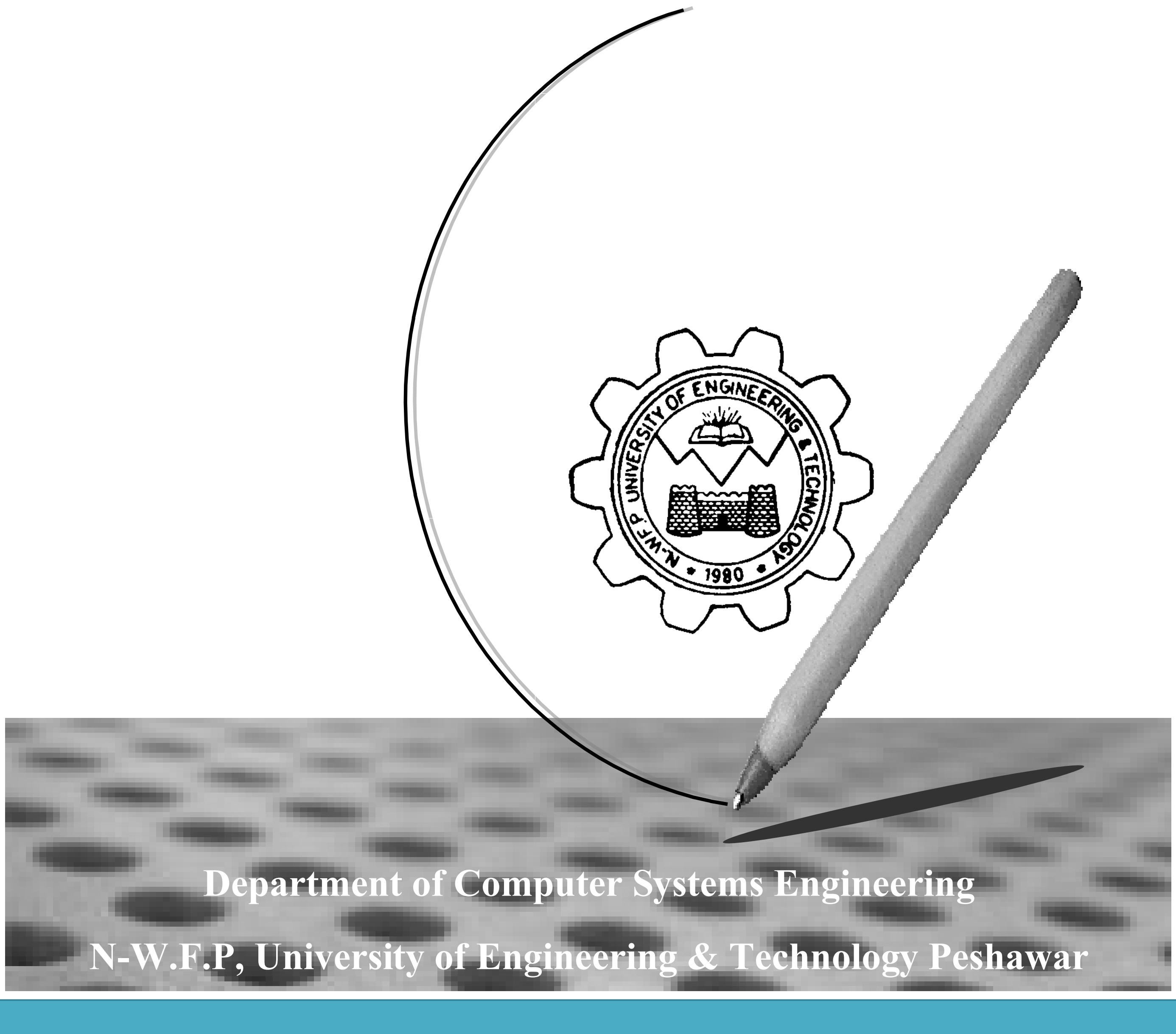
Lab Manual

Digital Logic Design



Department of Computer Systems Engineering

N-W.F.P, University of Engineering & Technology Peshawar

Table of contents

Lab 0
Introduction to the Students

Building a Digital Circuit:

A. Overview

This section describes the procedure for wiring logic circuits with any general-purpose white prototype board for your breadboard. One of these is contained in each lab kit.

1. Before wiring any circuit, generate a neat, complete logic.

2. Every time you add a wire or component to the physical circuit, mark off the corresponding part of the wiring diagram with a colored pencil or marker. This makes it easy to see what parts of the circuit have been built so far. If you make any circuit changes, draw these on your wiring diagram.

3. Insert IC packages into the appropriate breadboard area before inserting any wires. You will usually need to bend the IC leads (pins) slightly inward so that the spacing closely matches the spacing of sockets on the breadboard. Be careful to check that all IC leads actually make it into the correct sockets. Also make sure that pin 1 of the IC is in the correct position.

4. To remove an IC, use an extraction tool, screwdriver, pliers or tweezers to avoid bending or breaking IC leads, or personal injury.

6. Use only solid-conductor wire in the size range of AWG 20 to AWG 26. Wire with larger diameter may damage the socket spring clips of the breadboard. Wire strippers should be used to cut wires to appropriate lengths and to check wires that are suspected of having a larger diameter than permitted. Trim and re-strip the end of any jumper wire that
appears badly nicked or overly flexed.

7. It is possible to insert most wires by hand. In tight places, using the forceps from the tool kit can make the job much easier. In either case, wires are easier to insert if they have been cut at an angle of approximately 45 degrees with respect to the axis of the wire.

8. When removing wires, be sure to pull at a right angle to the socket to avoid damage.

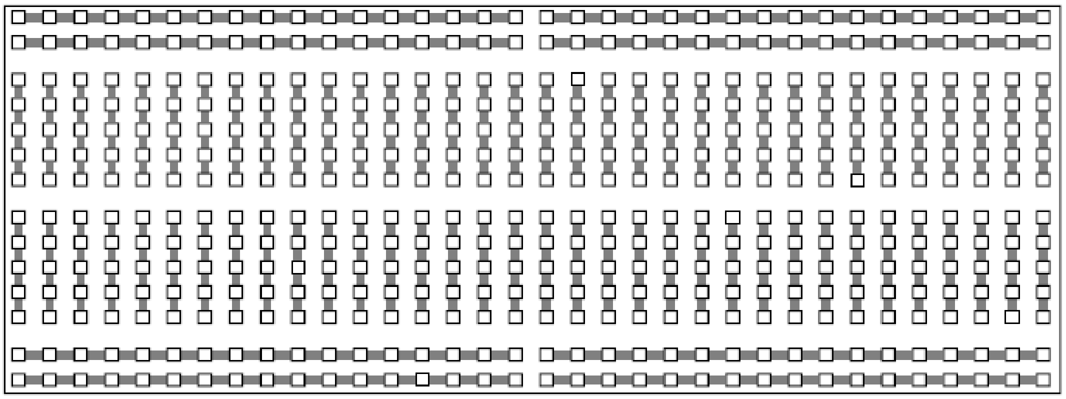
9. Route wires around IC packages, not over them. Occasionally an IC turns out to be defective. If wires have been placed over the IC, you will have to remove them so that the IC can be replaced.

10. It is best to wire a circuit in stages, beginning with power and ground connections. Add wires with the power switch OFF. Before turning power ON, remove all hand jewelry and make sure that no foreign metal objects are near the circuit. Check every IC to make sure it is not overheating. If any IC is too hot to touch immediately shut the power off and check all leads. (Be careful because shorted Ics can become very hot and leave a brand on your finger!) Also make sure that no IC has been inserted backwards.

11. IC devices can be damaged if the power level exceeds 5.5V. Damage may also occur if the supply voltage connection is removed from the IC pin while power is still being applied to the circuit.

12. To debug a circuit, use a Oscilloscope to check logic levels. Start at a position in the circuit where the logic level is known to be correct and work outward from there. If an IC does not appear to produce the correct signal, check that power and ground are correctly connected to the IC; also check all inputs to the component. Finally, check that the output of the IC is not incorrectly connected to some other signal.

13. If you cannot get your circuit to work, bring it and a current circuit diagram or schematic to Engineer for help.



Internal Connections of Bread-Board

B. Wiring guidelines

1. Use new wire. A box of new wire is available in the Lab.

a. Old wire can break inside the insulation, causing incorrect circuit behavior that is difficult to troubleshoot.

b. Old wire should be recycled; place old wires in the wire recycling box next to the new wire box in Lab.

2. Strip 4 breadboard squares worth of insulation off the ends of a wire when using it in the breadboard. This is approximately 5/16 inch or 8 mm.

a. If you strip too much, the wires in adjacent breadboard columns can touch, causing a short circuit and most likely incorrect behavior.

b. If you don’t strip enough, the insulation can prevent the spring clips in the breadboard holes from closing properly around the non-insulated part of the wire that is inserted into the hole.

3. Create power and ground busses at the top and bottom of your breadboard.

a. The connection pattern used in the breadboard is shown in Figure(to follow shortly) .

b. The top and bottom rows can be used to distribute +5VDC and ground to the ICs,

Note that the top and bottom "bus" rows have a break in the very middle! If you want a power or ground bus to run the length of the breadboard, you must insert a jumper in the middle of the row to join the two half rows together. This makes your wiring less crowded, and makes it easy to see power and ground connections.

4. Run all power signals in red wire and all ground signals in black wire.

a. Do not use red or black wire for any other signals. This makes it easy to tell which wires are power and ground wires, and which are actual signal wires.

b. Use a single power or ground wire from the bus to the chip. Do not daisy chain power or ground connections. Think parallel, not serial.

c. You may wire from the bus to the breadboard hole next to the chip. This makes it easy to see that the power and ground wires are connected to the correct pin.

d. You may wire from the bus to the breadboard column that connects to the chip. This allows more room for signal wires, without covering the power and ground wires.

5. Color code your wiring in some way. Here are some suggestions that are meant to make it easier to trace your wiring:

a. Use the same color for all the wires of a signal that runs to multiple gates.

b. Use different colors for different inputs of a gate.

c. If you have a bus, make all the wires of the bus the same color. However, if you have long runs of parallel wires that are the same color, it will be more difficult to trace individual bits of the bus.

Be creative.

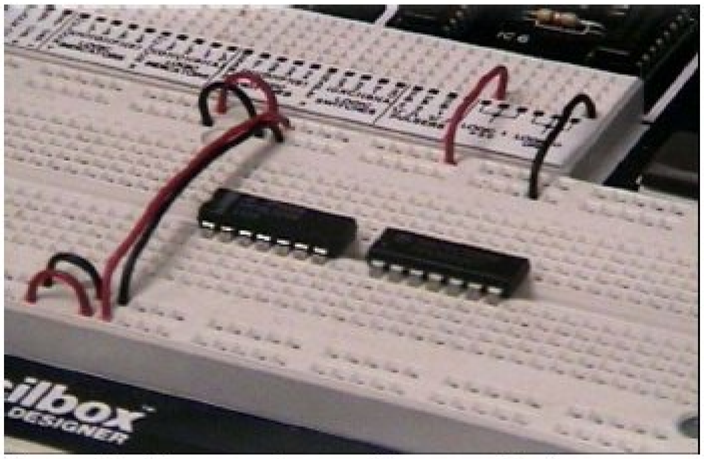
6. Wires should be routed no more than ½" (12 mm) above the breadboard.
a. If the wires are too high, it will be difficult to trace signals through your circuit.
b. If the wires are low, be sure the stripped wire ends are seated firmly in the breadboard. Careful routing is essential for efficient troubleshooting. Tight wiring can create sharp bends, which can cause trouble.

7. Avoid sharp bends in the wires. Sharp bends in the wire can cause the wire to break inside the insulation.
8. Run wires around or between chips rather than over them.
a. Your chips may be defective or be damaged while in use, and it is much easier to remove chips for testing/replacement if you do not have to remove your wiring in order to remove your chips.
b. When possible, leave 2 or 3 rows of the breadboard between chips, to allow signals to pass from one side of the IC to the other.

9. Make short wire lengths from source to destination.
a. Route wires point to point, rather than squaring corners.
b. Do not daisy chain power and ground wires. Think parallel, not serial.

c. Do not daisy chain signal lines from a switch input to several gates. Think parallel, not serial.

10. Wire from a complete schematic diagram. The chip's pin numbers should match the pin numbers in the diagram.



Ground and Power Buss

Lab 1
Basic logic gates

OBJECTIVES

- To correctly perform digital experiments
- To study about logic gates and verify their truth tables.

COMPONENTS REQUIRED

- 7404 hex inverter
- 7432 quad two input OR gate
- 7430 quad two input AND gate

THEORY

Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output.

Digital circuits have two discrete voltage levels to represent the binary digits *(bits)* 1 and 0. All digital circuits are switching circuits. Instead of mechanical switches, they use high-speed transistors to represent either an ON condition or an OFF condition. Various types of logic, representing different technologies, are available to the logic designer. The choice of a particular family is determined by factors such as speed, cost, availability, noise immunity, and so forth. The key requirement within each family is compatibility; that is, there must be consistency within the logic levels and power supplies of various integrated circuits made by different manufacturers. The experiments in this lab book use primarily transistor-transistor logic, or TTL. The detailed performance characteristics of TTL depend on the particular subfamily. However, all TTL is designed to operate from a 5 V power supply, and the logic levels are the same for all TTL integrated circuits.

AND GATE

The AND gate performs a logical multiplication commonly known as AND function. The output is high when both the inputs are high. The output is low level when any one of the inputs is low.

OR GATE

The OR gate performs a logical addition commonly known as OR function. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.

NOT GATE

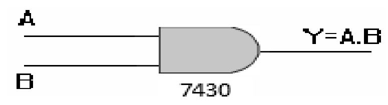
The NOT gate is called an inverter. The output is high when the input is low. The output is low when the input is high.

PROCEDURE

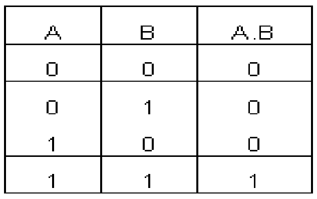
- Connections are given as per circuit diagram.
- Logical inputs are given as per circuit diagram.
- Observe the output and verify the truth table.

AND GATE

SYMBOL:

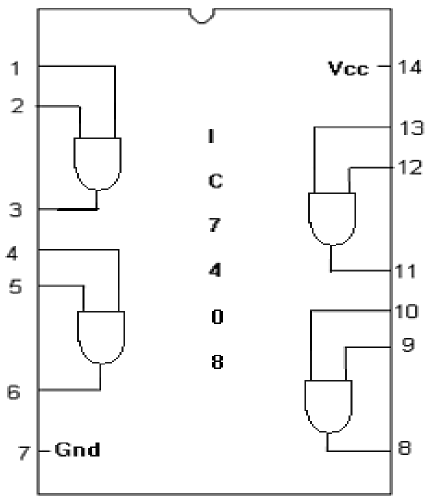


TRUTH TABLE



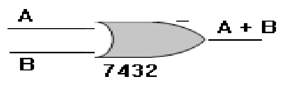
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

PIN DIAGRAM:



OR GATE

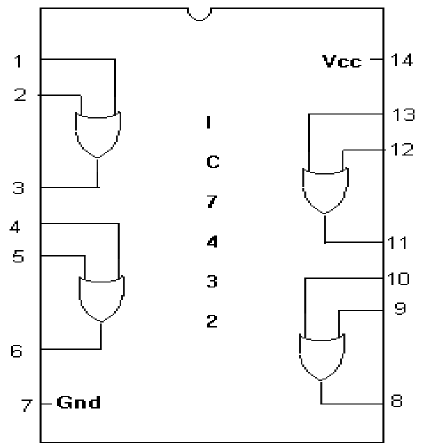
SYMBOL :



TRUTH TABLE

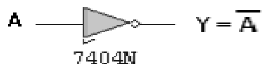
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

PIN DIAGRAM :



NOT GATE

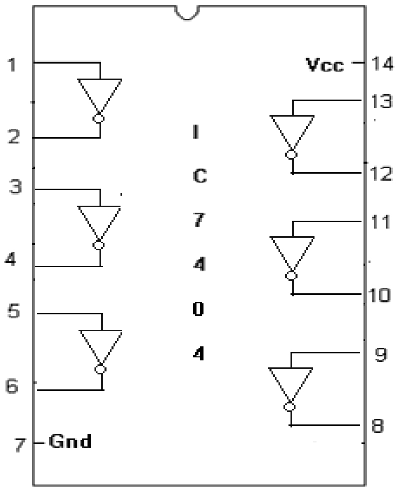
SYMBOL:



TRUTH TABLE :

A	$\overline{A}$
0	1
1	0

PIN DIAGRAM:



REVIEW QUESTIONS

- A burglar alarm for a car has a normally LOW (grounded) 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 . What type of gate will provide this logic?
- If more than two input AND & OR gates are available, how will you connect its inputs so that they work as 2 input gates? Perform it for 3 and 4 input AND & OR gates.

Lab 2

De-Morgan's Theorem

OBJECTIVE

After completing this experiment, you will be able to:

- Experimentally verify the De-Morgan's theorems using two input variables

COMPONENTS REQUIRED

- 7432 quad 2-input OR gate
- 7404 hex inverter
- LED
- 7430 quad 2-input AND gate
- DIP switch
- Three 1 kΩ resistors

DEMORGAN'S THEOREM

- (X + Y)' = X' . Y' .... (a)
- (X . Y)' = X' + Y' .... (b)

PROCEDURE

- Build the circuit for left part of equation(a) as shown in figure 2.1 and monitor the behavior of LED for different test inputs
- Then complete the circuit of figure 2.2 for the right part of equation(a) and complete the truth table 2.1 by testing each combination of inputs of appropriate switches
- Compare both the column results and check whether equation (a) is verified or not
- Repeat the above process by building the circuits of figure 2.3 and 2.4 and comparing its results for De-Morgans theorem verification of equation(b) .

LOGIC CIRCUIT DIAGRAMS

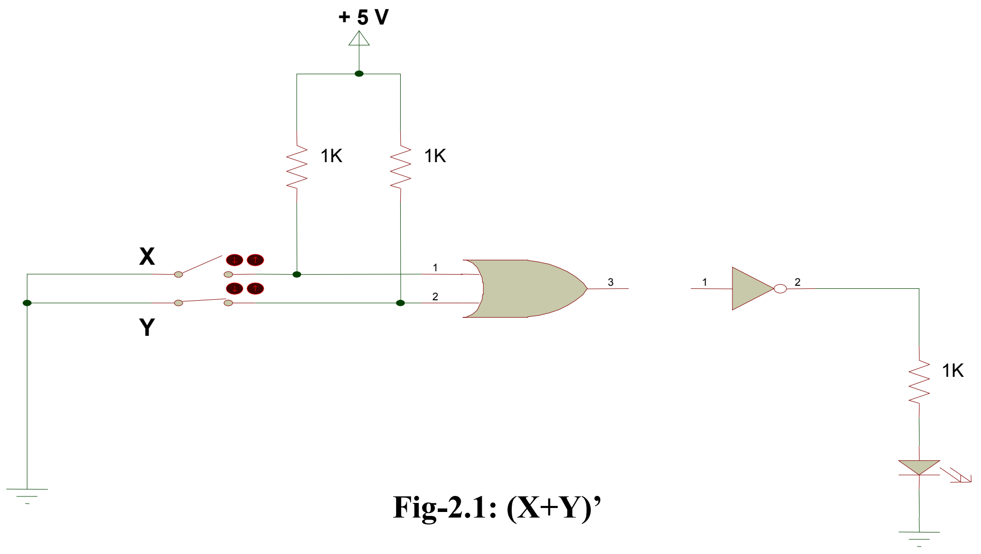


Fig-2.1: (X+Y)’

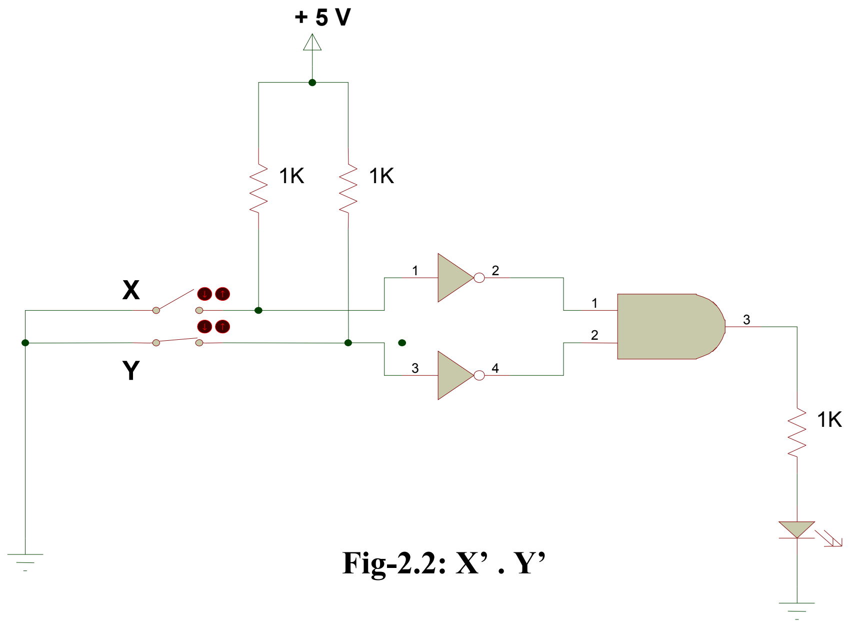


Fig-2.2: X’ . Y’

Truth table-2.1

X	Y	(X + Y)’	(X’ . Y’)
0	0		
0	1		
1	0		

1	**1**		

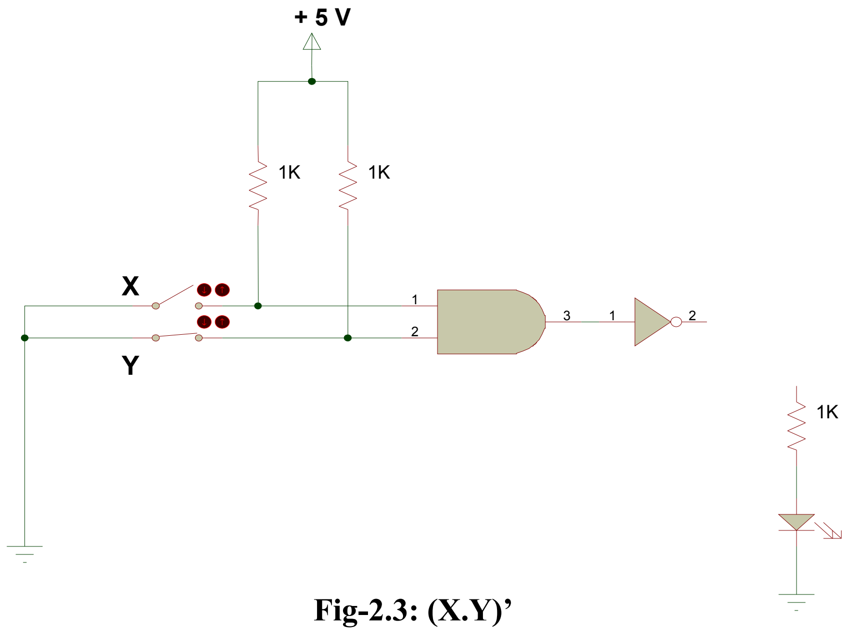


Fig-2.3: (X.Y)’

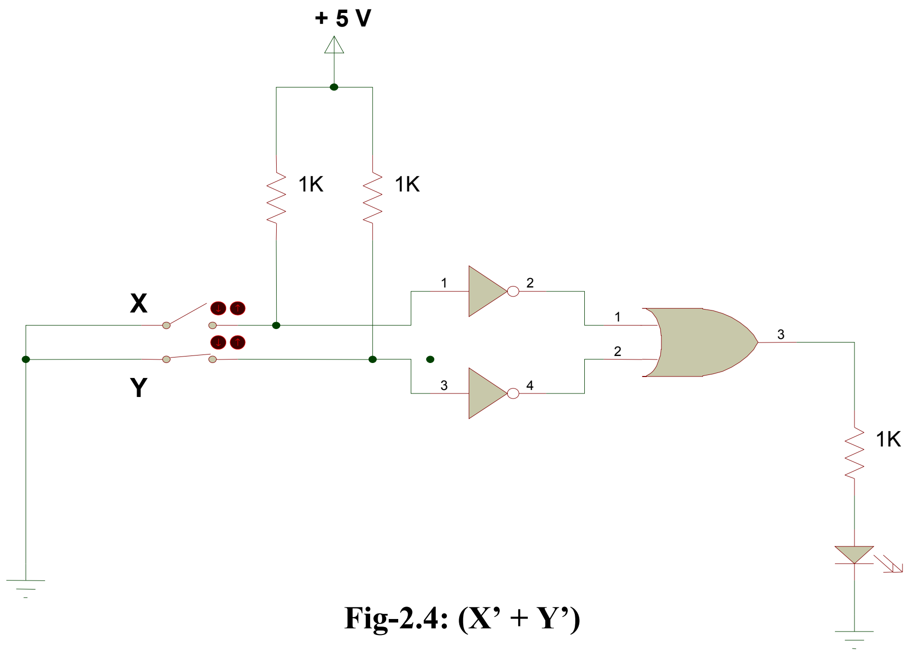


Fig-2.4: (X’ + Y’)

Truth table- 2.2

X	Y	(X . Y)’	(X’ + Y’)
0	**0**		
0	**1**		

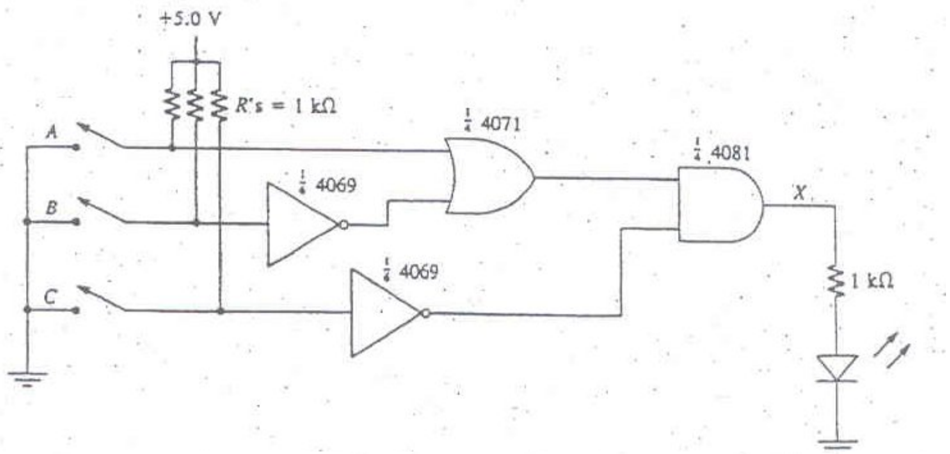
1	**0**		
1	**1**		

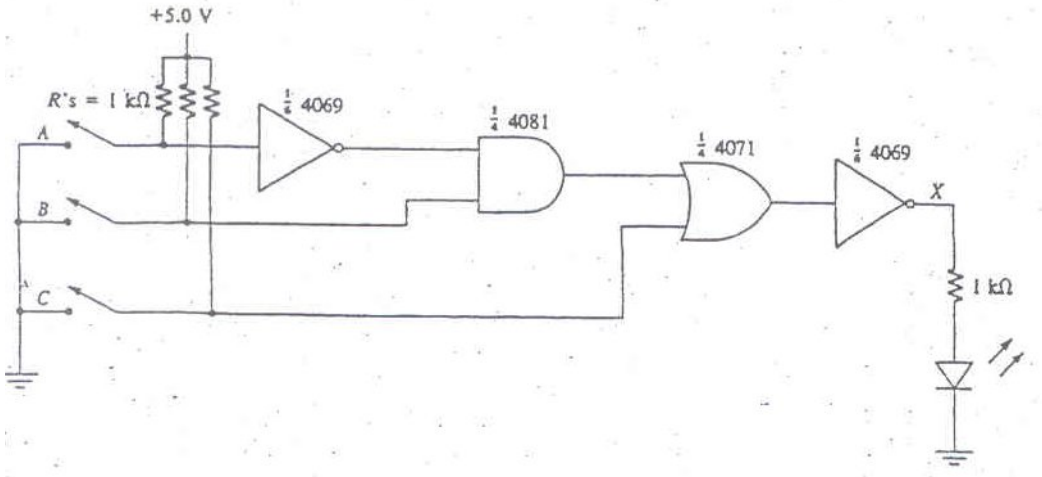
REVIEW QUESTIONS

- Simplify the expression using De-Morgans theorems and verify it simplified expression experimentally.

$$F = ((A \cdot B)' + A)'$$

- Determine whether the given circuits are equivalent. Then use DeMorgan's theorem to prove your answer algebraically.





Lab 3

Universal gates

OBJECTIVES

After completing this experiment, you will be able to:

- Determine the truth tables for universal gates (NAND and NOR).
- Use universal gates (NAND and NOR) to formulate other basic logic gates and demonstrate the equivalence between them.

COMPONENTS REQUIRED

- 7400 quad 2-input NAND gate
- 7402 quad 2-input NOR gate

THEORY

NAND and NOR are called the universal gates because the can be used in combination to perform the function of other basic logic gates and can implement any Boolean equation.

NAND GATE:

The NAND gate is a contraction of AND-NOT. The output is high when both inputs are low and any one of the input is low .The output is low level when both inputs are high.

NOR GATE:

The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.

PROCEDURE

Part-1

- Look up the connection diagram for the 7400 quad 2-input NAND gate and the 7402 quad 2-input NOR gate. Note that there are four gates on each of these ICs. Apply Vcc and ground to the appropriate pins.
- Then test one of the NAND gates by connecting all possible combination of inputs as listed in Table 3-1. Apply a logic 1 through a series 1 kΩ resistor and a logic 0 by connecting directly to ground. Show the logic output as well as the measured output voltage for each case, and include both in Table 3-1.

Table 3-1 (NAND gate)

IN1	IN2	OUT = (X . Y)’
0	0	
0	1	
1	0	
1	1	

- Repeat above steps for one of the NOR gate and tabulate your results in table 3-2

Table 3-2 (NOR gate)

IN1	IN2	OUT = (X + Y)’
0	0	
0	1	
1	0	
1	1	

Part-2

- Short circuit both the inputs of NAND and NOR gate as shown in Figure 3.1, to verify its functionality as inverter and complete the truth table 3-3.

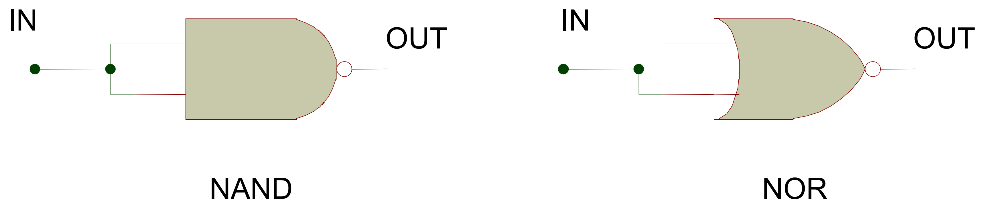


Figure 3.1: Inverter

Table 3-3

IN	OUT (NAND)	OUT (NOR)
0		
1		

- Build the circuit of the figure 3.2 to verify working as AND gate and complete the truth table 3-4

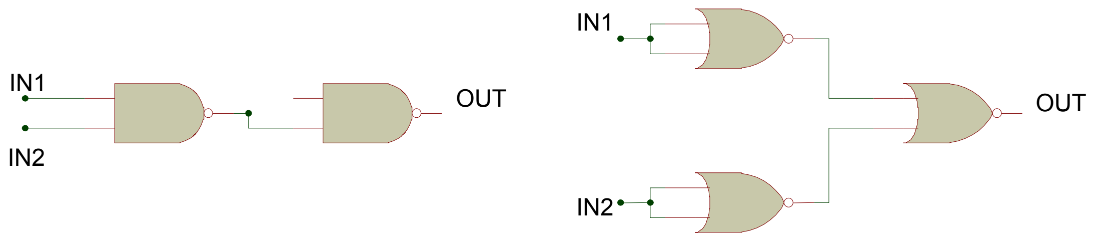


Figure 3.2 AND gate

Table 3-4

IN1	IN2	OUT (NAND)	OUT (NOR)
0	0		
0	1		
1	0		
1	1		

- Repeat the previous step for functionality of universal gates as OR gate and complete the truth table 3-5.

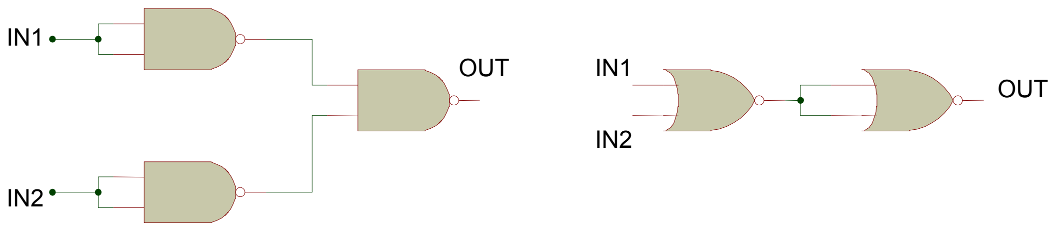


Figure 3.3 OR gate

Table 3-5

IN1	IN2	OUT (NAND)	OUT (NOR)
0	0		
0	1		
1	0		
1	1		

REVIEW QUESTIONS

- What do you mean by Universal gates? How will you implement any logic circuit using universal gates?
- Implement XNOR using NAND gates and repeat its implementation using NOR gates.
- Implement NAND using NOR gates.

LAB 4

ADDER AND SUBTRACTOR

OBJECTIVES

After completing this experiment, you will be able to:

- Design and construct half adder, full adder, half subtractor and full subtractor circuits
- Verify their truth tables using logic gates.

COMPONENTS REQUIRED

- 7432 or 7408 quad two input AND gates
- 7430 quad two input OR gates
- 7404 hex inverter
- 7486 quad two input XOR gates

THEORY

HALF ADDER:

A half adder has two inputs for the two bits to be added and two outputs one from the sum ‘ S’ and other from the carry ‘ c’ into the higher adder position. In half adder circuit, carry signal from the addition of the less significant bits sum from the XOR Gate the carry out from the AND gate.

FULL ADDER:

A full adder is a combinational circuit that forms the arithmetic sum of input; it consists of three inputs and two outputs. A full adder is useful to add three bits at a time but a half adder cannot do so. In full adder sum output will be taken from XOR Gate, carry output will be taken from OR Gate.

HALF SUBTRACTOR:

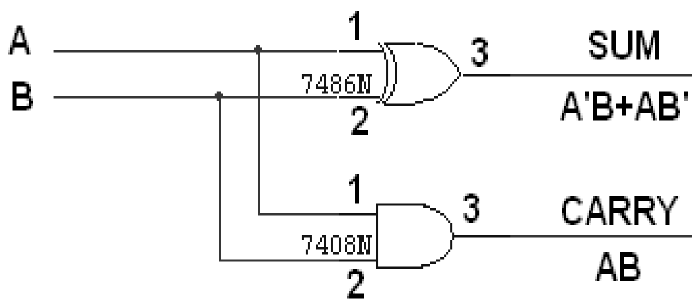
The half subtractor is constructed using XOR and AND Gate. The half subtractor has two input and two outputs. The outputs are difference and borrow. The difference can be applied using XOR Gate, borrow output can be implemented using an AND Gate and an inverter.

FULL SUBTRACTOR:

The full subtractor is a combination of XOR, AND, OR, NOT Gates. In a full subtractor the logic circuit should have three inputs and two outputs. The two half subtractor put together gives a full subtractor .The first half subtractor will be C and A B. The output will be difference output of full subtractor. The expression AB assembles the borrow output of the half subtractor and the second term is the inverted difference output of first XOR.

LOGIC DIAGRAM

HALF ADDER

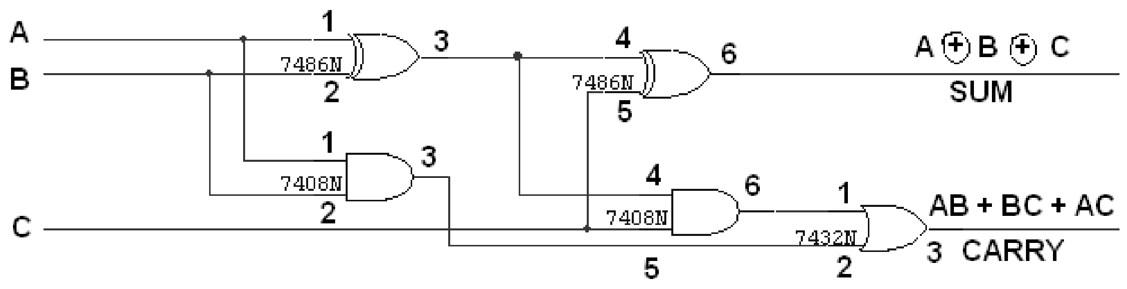


TRUTH TABLE

A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

LOGIC DIAGRAM
FULL ADDER

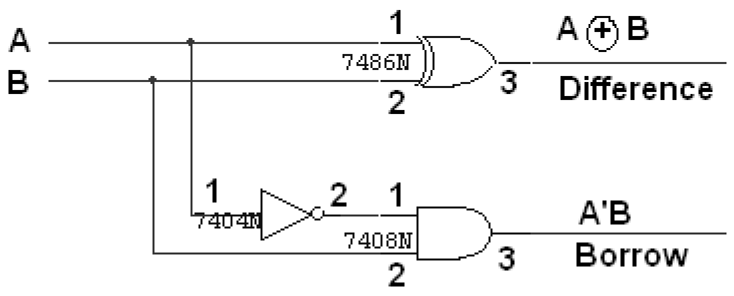
FULL ADDER USING TWO HALF ADDER



TRUTH TABLE

A	B	C	CARRY	SUM
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

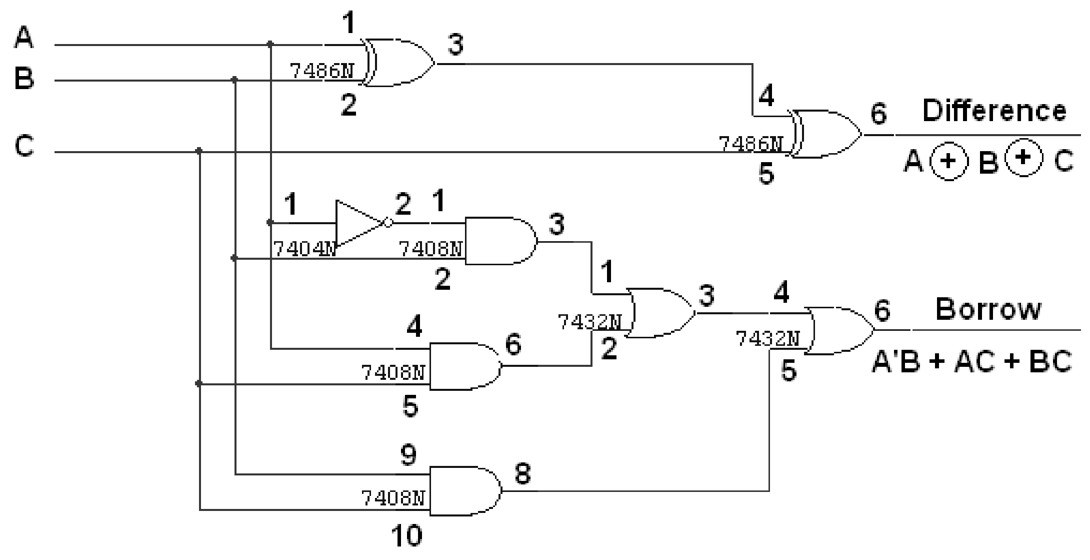
LOGIC DIAGRAM
HALF SUBTRACTOR



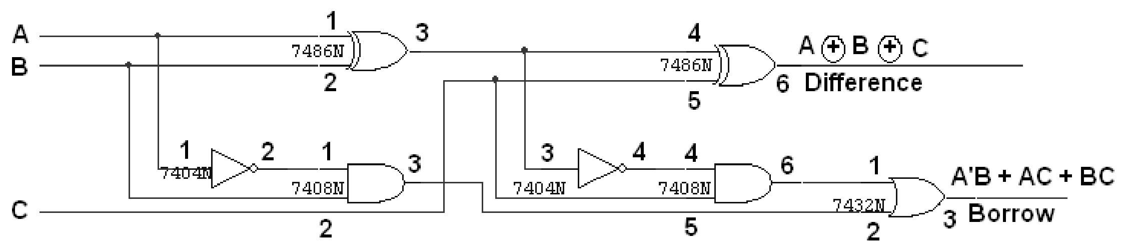
TRUTH TABLE

A	B	BORROW	DIFFERENCE
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

LOGIC DIAGRAM
FULL SUBTRACTOR



FULL SUBTRACTOR USING TWO HALF SUBTRACTOR



TRUTH TABLE

A	B	C	BORROW	DIFFERENCE
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

PROCEEDURE

- Connections are given as per circuit diagram.
- Logical inputs are given as per circuit diagram.
- Observe the output and verify the truth table.

Lab 5
ENCODER AND DECODER

OBJECTIVES

After completing this experiment you will be able to:

- Design and construct encoder and decoder
- Verify their truth tables using logic gates

COMPONENTS REQUIRED

- Two 7410, 3 I/P NAND gate
- Three 7432, 2 I/P OR gate
- 7404 hex inverter

THEORY

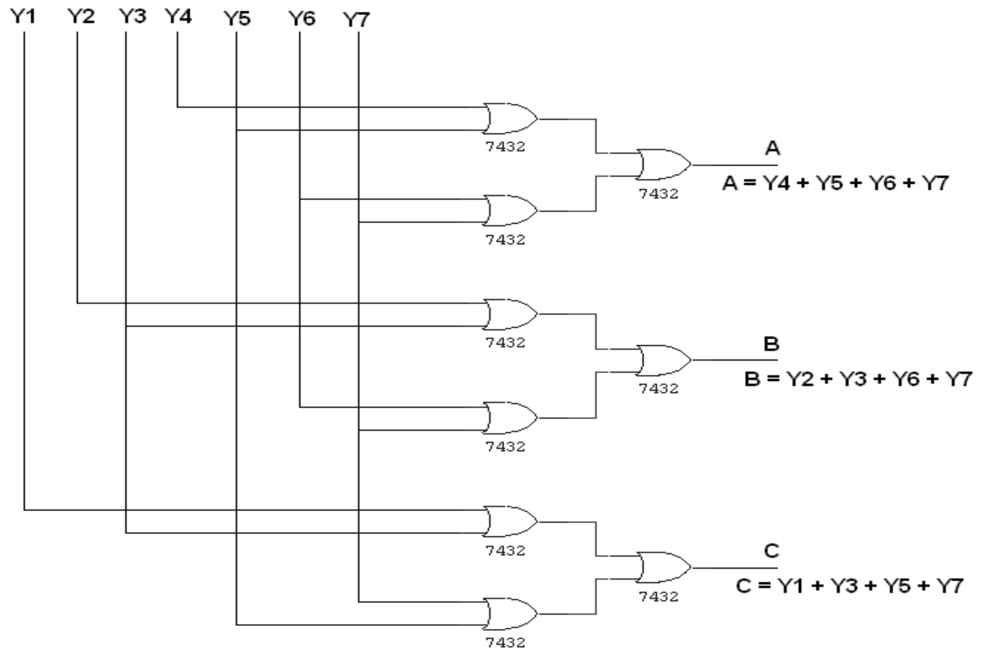
ENCODER:

An encoder is a digital circuit that perform inverse operation of a decoder. An encoder has 2^n input lines and n output lines. In encoder the output lines generates the binary code corresponding to the input value. In octal to binary encoder it has eight inputs, one for each octal digit and three output that generate the corresponding binary code. In encoder it is assumed that only one input has a value of one at any given time otherwise the circuit is meaningless. It has an ambiguila that when all inputs are zero the outputs are zero. The zero outputs can also be generated when D0 = 1.

DECODER:

A decoder is a multiple input multiple output logic circuit which converts coded input into coded output where input and output codes are different. The input code generally has fewer bits than the output code. Each input code word produces a different output code word i.e there is one to one mapping can be expressed in truth table. In the block diagram of decoder circuit the encoded information is present as n input producing 2^n possible outputs. 2^n output values are from 0 through out $2^n - 1$.

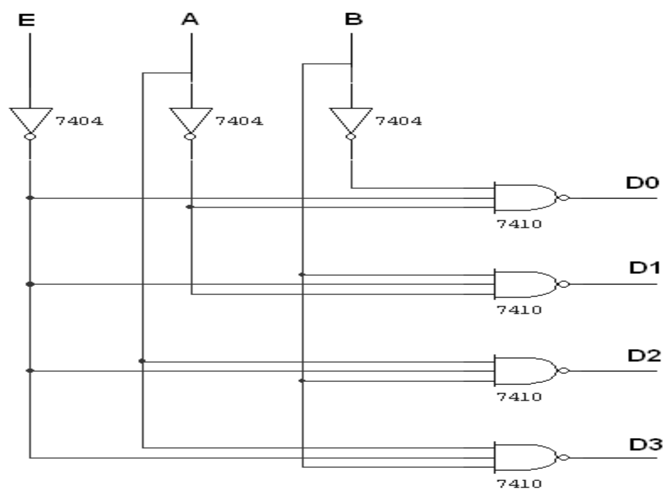
LOGIC DIAGRAM FOR ENCODER



TRUTH TABLE

INPUT							OUTPUT		
Y1	Y2	Y3	Y4	Y5	Y6	Y7	A	B	C
1	0	0	0	0	0	0			
0	1	0	0	0	0	0			
0	0	1	0	0	0	0			
0	0	0	1	0	0	0			
0	0	0	0	1	0	0			
0	0	0	0	0	1	0			
0	0	0	0	0	0	1			

LOGIC DIAGRAM FOR DECODER



TRUTH TABLE

INPUT			OUTPUT			
E	A	B	D0	D1	D2	D3
1	0	0				
0	0	0				
0	0	1				
0	1	0				
0	1	1				

PROCEDURE

- Connections are given as per circuit diagram.
- Logical inputs are given as per circuit diagram.
- Observe the output and verify the truth table.

REVIEW QUESTIONS

- How do you think that above encoder circuit work as an Encoder?
- Design an encoder using NOR gates only.
- What will be the output of the decoder circuit if NAND gates are replaced by AND gates?
- What decoder actually do?
- What is the purpose of enable input in decoder?
- Design a 2x4 decoder using two 1x2 decoders.

LAB 6

MULTIPLEXER AND DEMULTIPLEXER

OBJECTIVES

After completing this experiment, you will be able to:

- Design and construct Multiplexer and DeMultiplexer
- Verify their truth tables using basic logic gates

COMPONENTS REQUIRED

- Two 7411, 3 I/P AND gates
- 7432, OR gate
- 7404, hex inverter

THEORY

MULTIPLEXER:

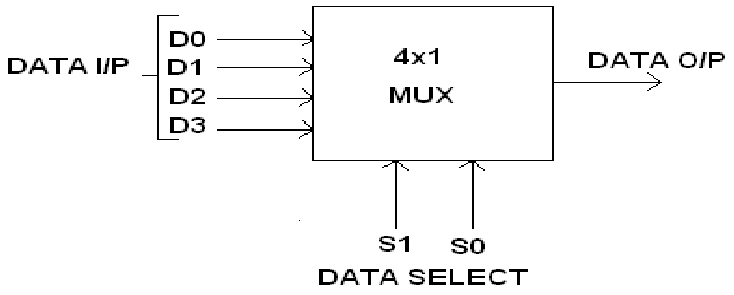
Multiplexer means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally there are 2^n input line and n selection lines whose bit combination determine which input is selected.

DEMULTIPLEXER:

The function of Demultiplexer is in contrast to multiplexer function. It takes information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. Decoder can also be used as demultiplexer.

In the 1: 4 demultiplexer circuit, the data input line goes to all of the AND gates. The data select lines enable only one gate at a time and the data on the data input line will pass through the selected gate to the associated data output line.

BLOCK DIAGRAM FOR 4:1 MULTIPLEXER:

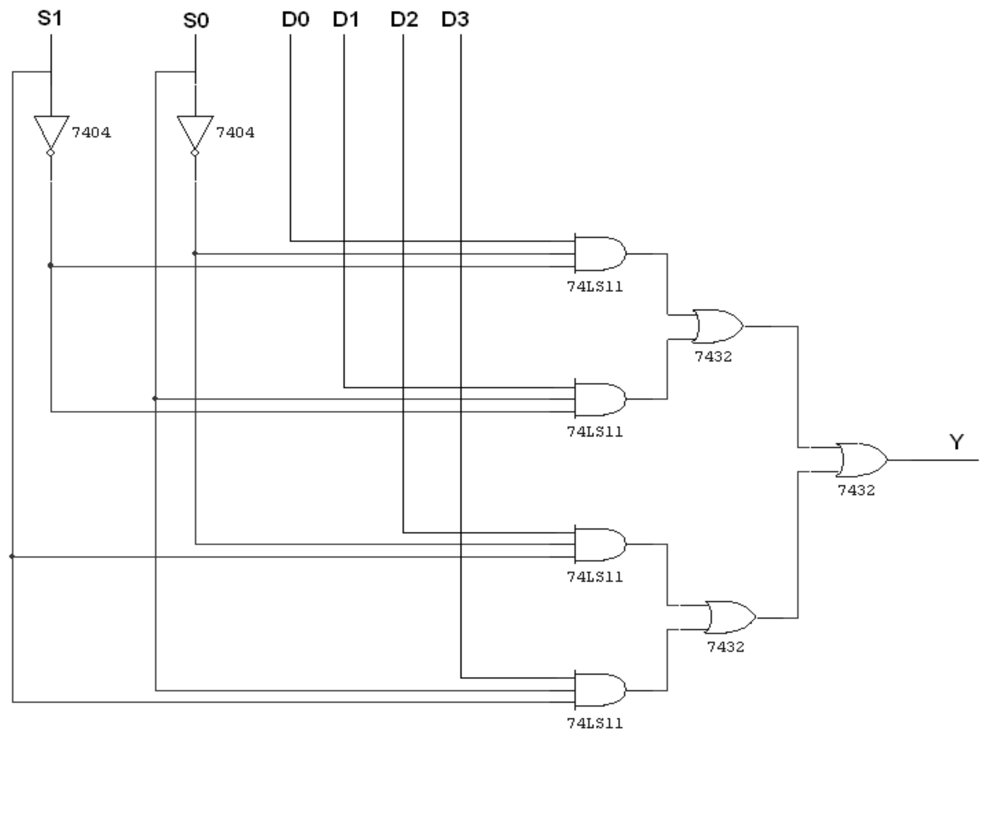


FUNCTION TABLE:

S1	S0	INPUTS Y
0	0	D0 → D0 S1' S0'
0	1	D1 → D1 S1' S0
1	0	D2 → D2 S1 S0'
1	1	D3 → D3 S1 S0

Y = D0 S1' S0' + D1 S1' S0 + D2 S1 S0' + D3 S1 S0

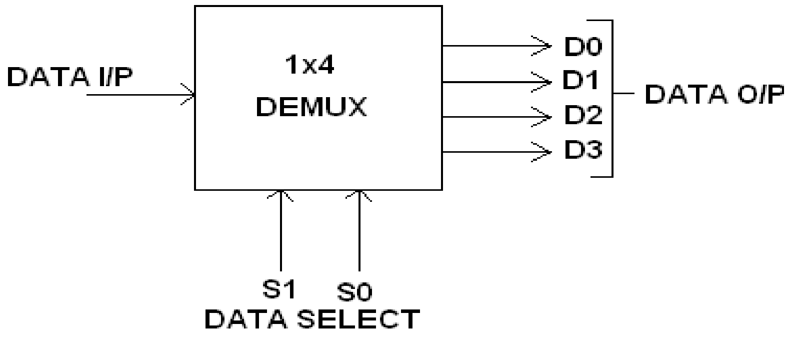
CIRCUIT DIAGRAM FOR MULTIPLEXER:



TRUTH TABLE:

S1	S0	Y = OUTPUT
0	0	D0
0	1	D1
1	0	D2
1	1	D3

BLOCK DIAGRAM FOR 1:4 DEMULTIPLEXER:

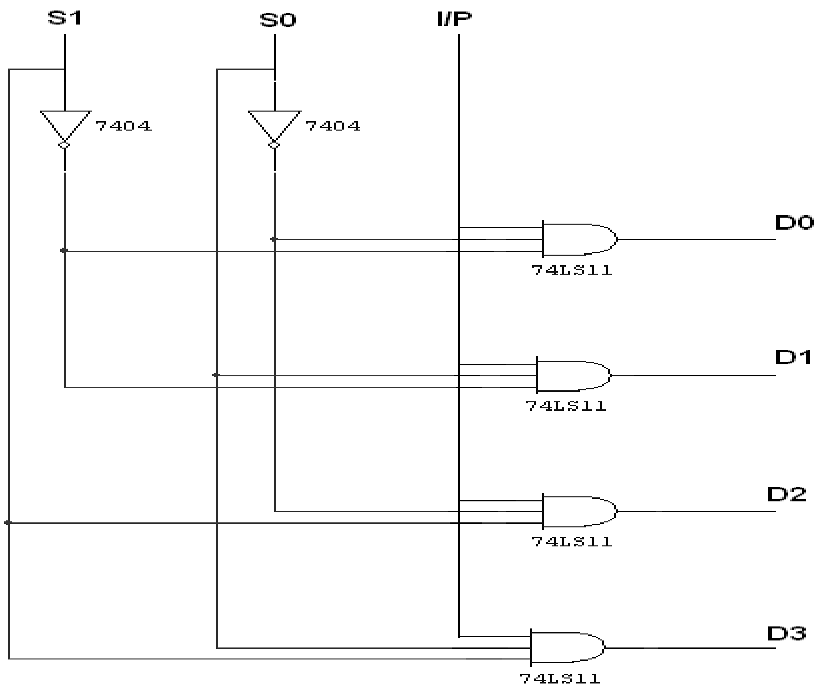


FUNCTION TABLE:

S1	S0	INPUT
0	0	X → D0 = X S1’ S0’
0	1	X → D1 = X S1’ S0
1	0	X → D2 = X S1 S0’
1	1	X → D3 = X S1 S0

Y = X S1’ S0’ + X S1’ S0 + X S1 S0’ + X S1 S0

LOGIC DIAGRAM FOR DEMULTIPLEXER:



TRUTH TABLE:

INPUT			OUTPUT			
S1	**S0**	**I/P**	**D0**	**D1**	**D2**	**D3**
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

PROCEDURE

- Connections are given as per circuit diagram.
- Logical inputs are given as per circuit diagram.
- Observe the output and verify the truth table.

REVIEW QUESTIONS

- What is the difference between Multiplexer and De-Multiplexer?
- Design the given function using Multiplexer.

  OUT (X,Y,Z) = X'.Y.Z + X.Y.Z' + X'.Y'.Z'
- Design a 4x1 MUX using two 2x1 MUXes.

LAB 7

MAGNITUDE COMPARATOR

OBJECTIVES

After completing experiment, you will be able to:

- Design and construct magnitude comparator
- Verify truth table for 2 – bit magnitude comparator using basic gates.
- Design and verify 8 – bit magnitude comparator using IC 7485.

COMPONENTS REQUIRED

- Two 7408 or 7430, 2 I/P AND gate
- 7432, 2 I/P OR gate
- 7404, hex inverter
- 7486, XOR gate
- 7485, 4-bit magnitude comparator

THEORY

The comparison of two numbers is an operator that determine one number is greater than, less than (or) equal to the other number. A magnitude comparator is a combinational circuit that compares two numbers A and B and determine their relative magnitude. The outcome of the comparator is specified by three binary variables that indicate whether A>B, A=B (or) A<B.

$A = A_3\ A_2\ A_1\ A_0$
$B = B_3\ B_2\ B_1\ B_0$

The equality of the two numbers and B is displayed in a combinational circuit designated by the symbol (A=B).

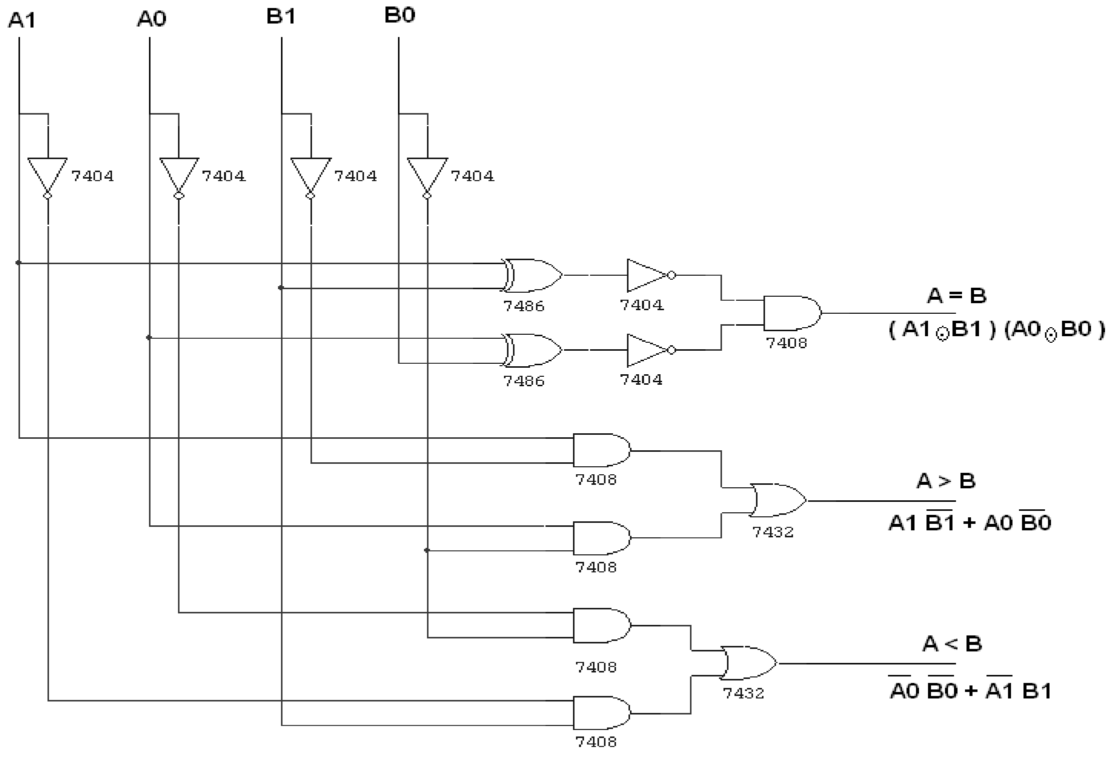
This indicates A greater than B, then inspect the relative magnitude of pairs of significant digits starting from most significant position. A is 0 and that of B is 0.

We have A<B, the sequential comparison can be expanded as
$A>B = A3B_3^1 + X_3A_2B_2^1 + X_3X_2A_1B_1^1 + X_3X_2X_1A_0B_0^1$
$A<B = A_3^1B_3 + X_3A_2^1B_2 + X_3X2A_1^1B_1 + X_3X_2X_1A_0^1B_0$

The same circuit can be used to compare the relative magnitude of two BCD digits.
Where, A = B is expanded as,

$$A = B = \underbrace{(A_3 + B_3)}_{x_3}\ \underbrace{(A_2 + B_2)}_{x_2}\ \underbrace{(A_1 + B_1)}_{x_1}\ \underbrace{(A_0 + B_0)}_{x_0}$$

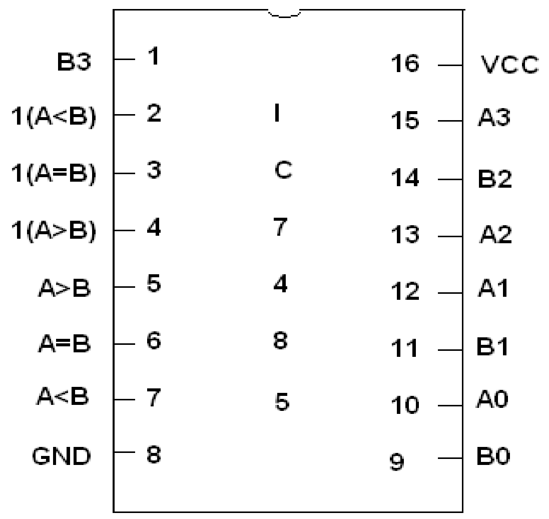
LOGIC DIAGRAM
2 BIT MAGNITUDE COMPARATOR


A1
A0
B1
B0
7404
7486
7408
7432
A = B
(A1 ⊙ B1) (A0 ⊙ B0)
A > B
A1 $\overline{B1}$ + A0 $\overline{B0}$
A < B
$\overline{A0}$ $\overline{B0}$ + $\overline{A1}$ B1

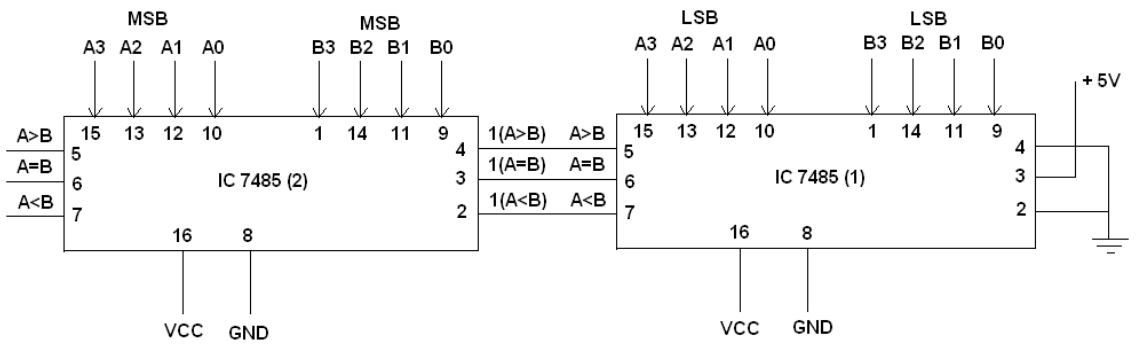
TRUTH TABLE

A1	A0	B1	B0	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

PIN DIAGRAM FOR IC 7485



LOGIC DIAGRAM
8 BIT MAGNITUDE COMPARATOR



TRUTH TABLE

A	B	A>B	A=B	A<B
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0	1	0
0 0 0 1 0 0 0 1	0 0 0 0 0 0 0 0	1	0	0
0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 1	0	0	1

PROCEDURE

- Connections are given as per circuit diagram.
- Logical inputs are given as per circuit diagram.
- Observe the output and verify the truth table.

Lab 8

The D latch and D flip flop

OBJECTIVES

After completing this experiment, you will be able to:

- Construct and test a D latch from four NAND gates and an inverter.
- Test a D flip-flop and investigate several application circuits for both the latch and the flip-flop.

COMPONENTS REQUIRED

- LEDs
- 7400 quad NAND gate
- 7404 hex inverter
- Resistors

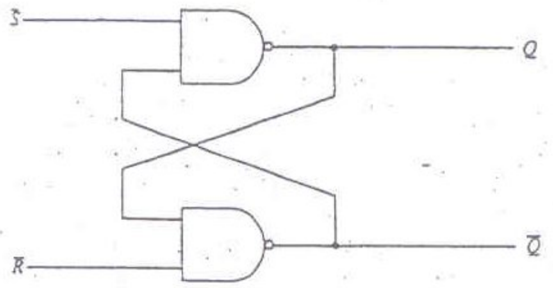
THEORY

As we have seen, combinational logic circuits are circuits in which the outputs are determined fully by the current inputs. Sequential logic circuits contain information about previous conditions—the dial telephone, for example. The difference between combinational and sequential circuits is that sequential circuits contain memory and combinational circuits do not.

The most basic memory unit is the latch, which uses feedback to lock onto and hold data. It can be constructed from either NAND or NOR gates. The ability to remember previous conditions is easy to demonstrate with Boolean algebra. For example, Figure. 7-1 shows an R-S latch made from NAND gates. This circuit is widely used for switch denouncing and is available as an integrated circuit containing four latches (the 74LS279).

A simple modification of the basic latch is the addition of steering gates and an inverter as shown in Figure 7-2. This circuit is called a D (for Data) latch. An enable input (sometimes referred to as a clock input) allows data present on the D input to be transferred to the output when enabled. When the enable input is not asserted, the last level of Q is latched. This circuit is available in integrated circuit form as the 7475A quad D-latch. Although there are four latches in this IC, there are only two shared enable signals.

Design problems are often simplified by having all transitions in a system occur synchronously (at the same time) by using a common source of pulses to cause the change. This common pulse is called a clock. The output changes occur only on the leading or the trailing edge of the clock pulse. Some ICs have inputs that directly set or reset the output any time they are asserted. These inputs are labeled, asynchronous inputs because no clock pulse is required. The D-type flip-flop with positive edge-triggering and asynchronous inputs is the 7474.



Equation for top NAND gate:

$$Q = \overline{\overline{S} \cdot \overline{Q}}$$

Applying DeMorgan's theorem:

$$Q = S + Q$$

Thus, Q appears on both sides of the equation.
If $\overline{S} = 1$, then $S = 0$ and $Q = 0 + Q$ (Q is previous state) output is latched.

FIGURE 7-1

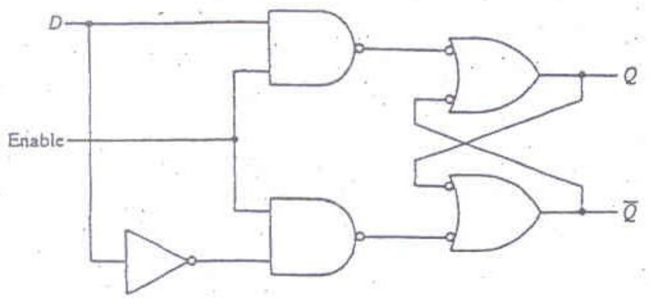


FIGURE 7-2

PROCEDURE

Applications of the D Latch

1. Build the R-S latch shown in figure 7-3. We will first test this circuit and then use it as part of a new circuit. Use a wire to simulate the single-pole, double-throw (SPDT) switch. The LEDs will be used in this section as logic monitors. Note that because TTL logic is much better at sinking current than at sourcing current, the LEDs are arranged to be ON when the output is LOW. To make the LEDs read the HIGH output when they are ON, we can connect them to the opposite output! This simple trick sometimes avoids the use of an inverter in circuits.
2. Leave the wire on the A terminal and note the logic shown on the LEDs. Now

simulate a bouncing switch by removing the A end of the wire. Do NOT touch B yet! Instead, reconnect the wire to A several times. What happens to the logic?

3. Remove the A end of the wire and touch B. Simulate the switch bouncing several times by removing and reconnecting B, (Switches never bounce back to the opposite terminal, so you should not touch A.) What happens?

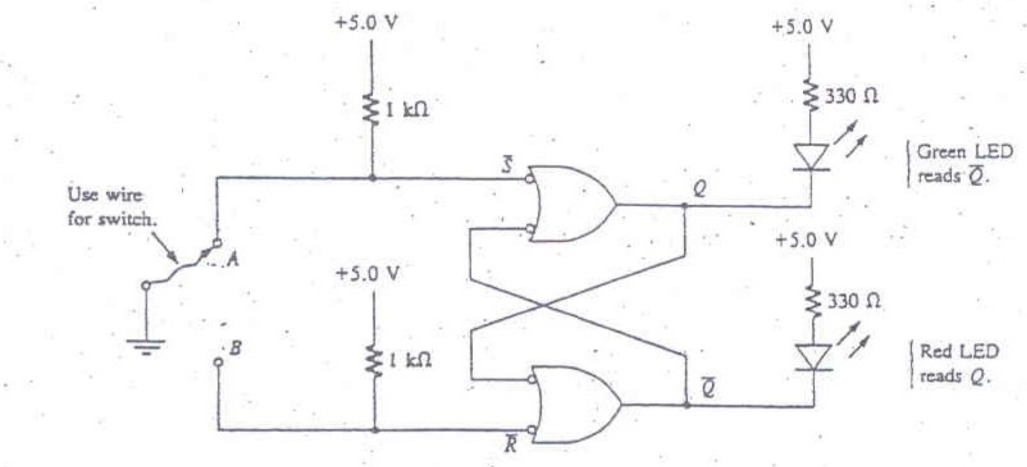
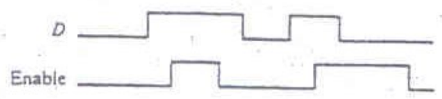


FIGURE 7-3

Review Questions

1. Write the truth table for the D latch shown in Figure 7-2.

2. Sketch the output of a D latch for the timing diagram of Figure 7-6.



Output:

FIGURE 7-6

LAB 9

The J-K Flip-Flop

OBJECTIVES

After completing this experiment, you will be able to:

- Test various configurations for a J-K flip-flop, including the asynchronous and synchronous inputs.
- Compare edge-triggering with pulse-triggering.

COMPONENTS REQUIRED

- 7411, 3 I/P AND gate
- 7402, 2 I/P OR gate

THEORY

The edge-triggered D flip-flop avoids the problem of the R-S invalid output states by not allowing an invalid state. Instead of four possible input states, the D flip-flop has only two possible input states: HIGH or LOW. However, unlike the R-S flip-flop, the D flip-flop cannot be latched unless the clock pulses are removed—a condition that limits a number of applications for this device. A solution to these problems is the J-K flip-flop, which is basically a clocked R-S flip-flop with additional logic to replace the R-S invalid output states with a new mode called toggle causes the flip-flop to change to the state opposite to its present state. It is similar to the operation of an automatic garage door opener. If the button is pressed when the door is open, the door will close; if it is closed, it will open.

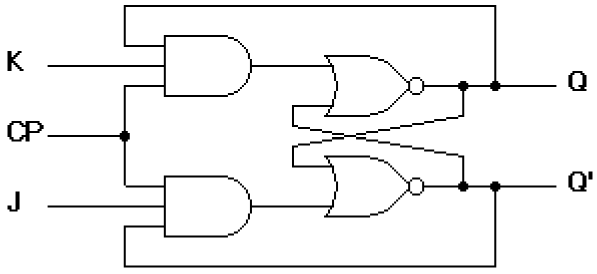
The J-K flip-flop is the most versatile of the basic flip-flops. All applications for flip-flops can be accomplished with either the D or the J-K flip-flop. The clocked R-S flip-flop is seldom used; it is used mostly as an internal component of integrated circuits. The inputs are labeled J (the set mode) and K (the reset mode) to avoid confusion with the R-S flip-flop.

The need to assure that input data do not affect the output until they are at the correct level led to the concept of edge-triggering, investigated in the last experiment. Edge-triggering is not the only method for assuring synchronous, transitions, although it is preferred. The other method is pulse-triggered or master-slave flip-flops. In these flip-flops, the data are clocked into the master on the leading edge of the clock and into the slave on the trailing edge of the clock. It is imperative that the input data not change during the time the clock pulse is HIGH or the data in the master will be changed. In

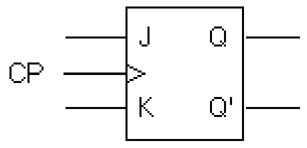
principle, any flip-flop could use pulse-triggering. However, in practice D flip-flops are available only as edge triggered devices. J-K flip-flops are available as either edge-, or pulse-triggered devices. A modification called the data-lockout flip-flop transfers data to the master on the leading edge of the clock and into the slave on the trailing edge of the clock, eliminating the requirement to hold the data at a constant level during the clock pulse.
A certain amount of time is required for the input of a logic gate to affect the output. This time, called the propagation delay time, depends on the logic family.

PROCEDURE

- Construct the circuit of the figure a.
- Use LEDs as logic monitors to monitor the output by giving different combination of inputs.
- Verify the truth table and repeat the whole process for NAND version of JK flip flop. Check, what will happen to the output?



(a) Logic diagram



(b) Graphical symbol

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

(c) Transition table

Clocked JK flip-flop

Review Questions

1. What is the difference between an asynchronous and a synchronous input?
2. Describe how you would set a J-K flip-flop asynchronously?
3. What is the difference between edge-triggering and pulse-triggering?

Lab 10

Ripple counters

OBJECTIVES

After completing this experiment, you will be able to:

- Design and verify 4 bit ripple counter
- Design and verify mod 10/ mod 12 ripple counter
- Explain how the counter is a frequency divider

COMPONENTS REQUIRED

- Two 7476 JK flip flop ICs
- One 7400, 2 I/P NAND gate

THEORY

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived. A specified sequence of states appears as counter output. This is the main difference between a register and a counter. There are two types of counter, synchronous and asynchronous. In synchronous common clock is given to all flip flop and in asynchronous first flip flop is clocked by external pulse and then each successive flip flop is clocked by Q or Q output of previous stage. A soon the clock of second stage is triggered by output of first stage. Because of inherent propagation delay time all flip flops are not activated at same time which results in asynchronous operation.

PIN DIAGRAM FOR IC 7476

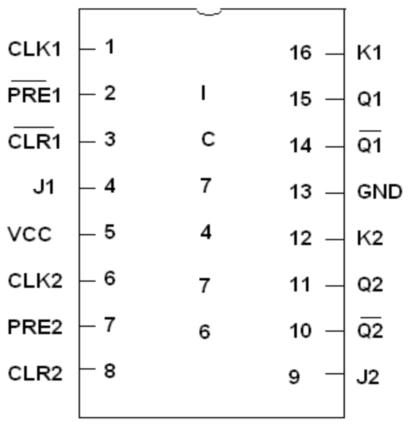


Fig: 10-1

LOGIC DIAGRAM FOR 4 BIT RIPPLE COUNTER

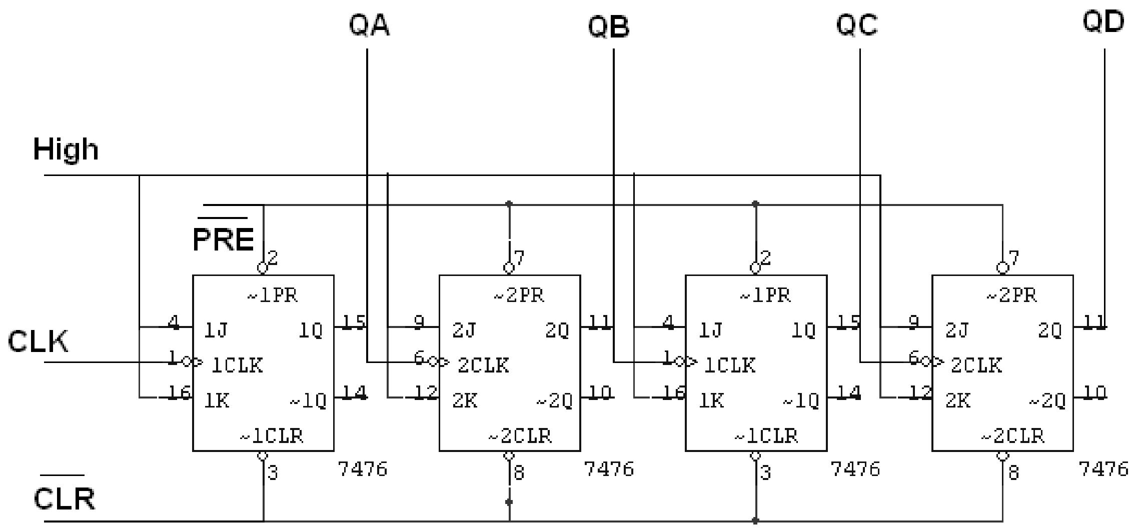


Fig:10-2

TRUTH TABLE

CLK	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1
13	1	0	1	1
14	0	1	1	1
15	1	1	1	1

LOGIC DIAGRAM FOR MOD - 10 RIPPLE COUNTER:

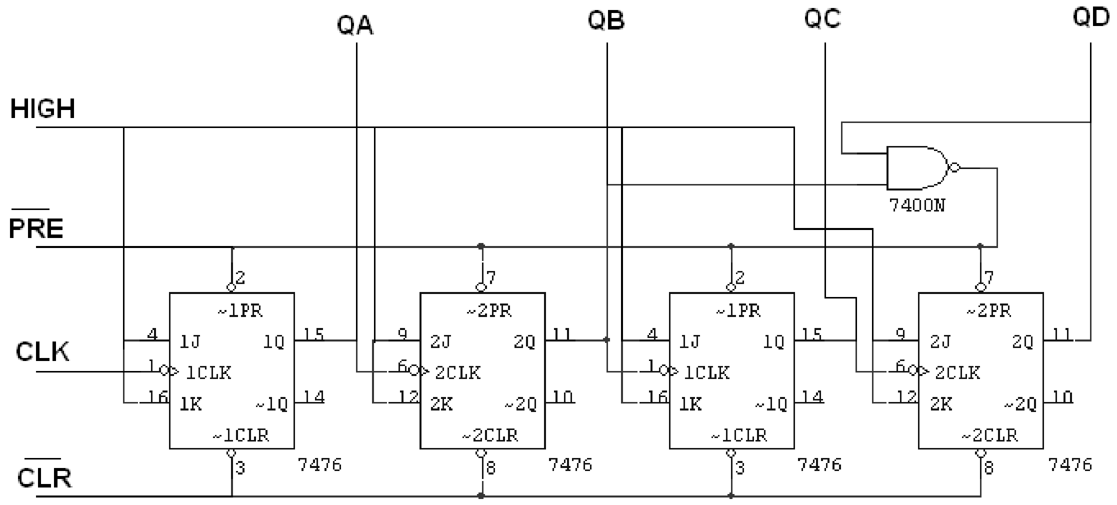


Fig:9-3

TRUTH TABLE

CLK	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	0	0	0

LOGIC DIAGRAM FOR MOD - 12 RIPPLE COUNTER

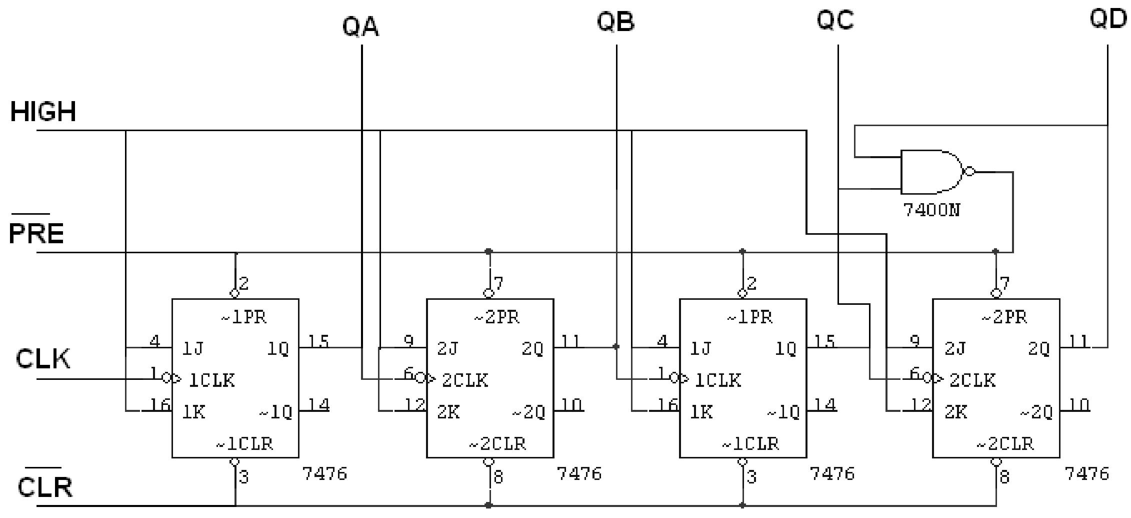


Fig:9-4

TRUTH TABLE

CLK	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	0	0

PROCEDURE

- Connections are given as per circuit diagram.
- Logical inputs are given as per circuit diagram.
- Observe the output and verify the truth table.
-

REVIEW QUESTIONS

1. Counters can be used as frequency dividers. When the clock frequency in Fig:9-2 was 1 kHz, what was the output frequency of flip-flop A and flip-flop B?
   f_A= ________________________
   f_B= ________________________

2. Would inverters on the clock inputs change the count direction of a ripple counter?

3. How many flip-flops are needed to build a moduIus-5 counter?

LAB 11
Shift Registers

OBJECTIVES

After completing this experiment, you will be able to design and implement shift registers with

(i) Serial in serial out
(ii) Serial in parallel out
(iii) Parallel in serial out
(iv) Parallel in parallel out

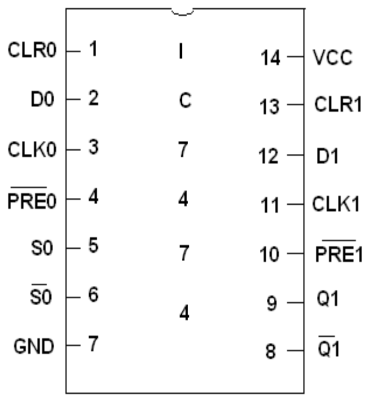
COMPONENTS REQUIRED

- Two 7474 D- Flip Flop
- One 7432 OR gate

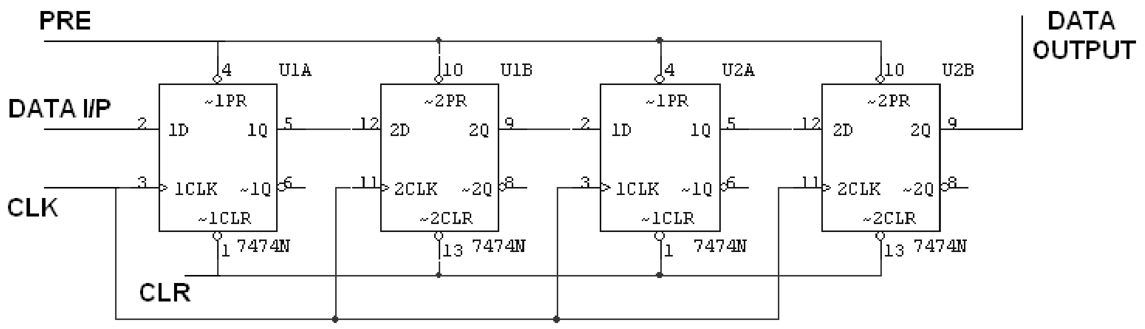
THEORY

A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration of shift register consist of a D-Flip flop cascaded with output of one flip flop connected to input of next flip flop. All flip flops receive common clock pulses which causes the shift in the output of the flip flop. The simplest possible shift register is one that uses only flip flop. The output of a given flip flop is connected to the input of next flip flop of the register. Each clock pulse shifts the content of register one bit position to right.

PIN DIAGRAM:



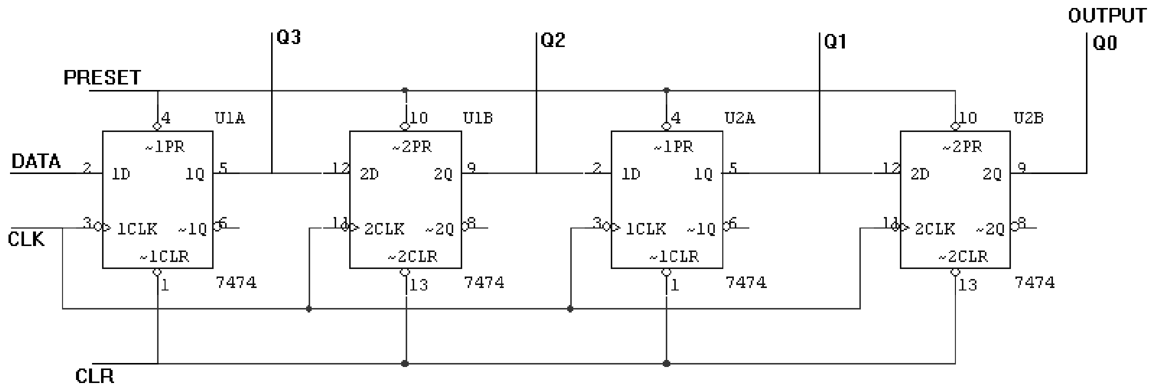
LOGIC DIAGRAM
SERIAL IN SERIAL OUT:



TRUTH TABLE:

CLK	Serial in	Serial out
1	1	0
2	0	0
3	0	0
4	1	1
5	X	0
6	X	0
7	X	1

LOGIC DIAGRAM
SERIAL IN PARALLEL OUT:

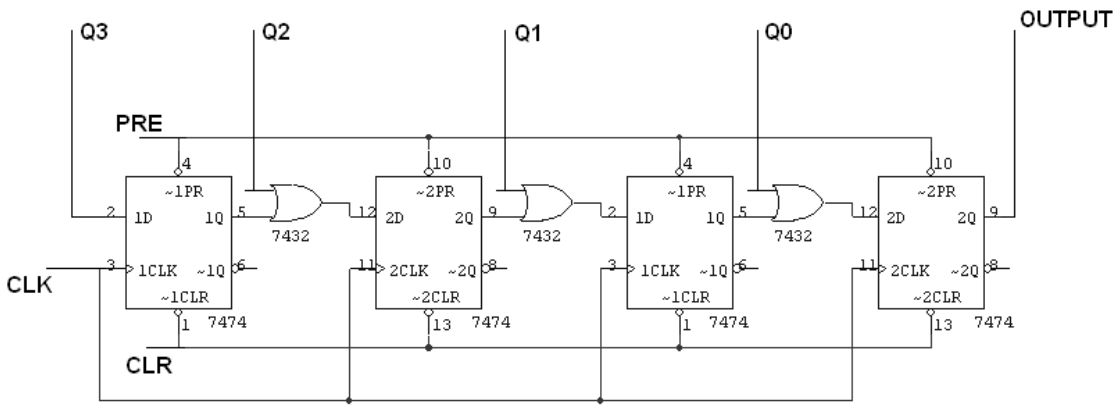


TRUTH TABLE:

CLK	DATA	OUTPUT			
		Q_A	Q_B	Q_C	Q_D
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	0	0	1

LOGIC DIAGRAM:

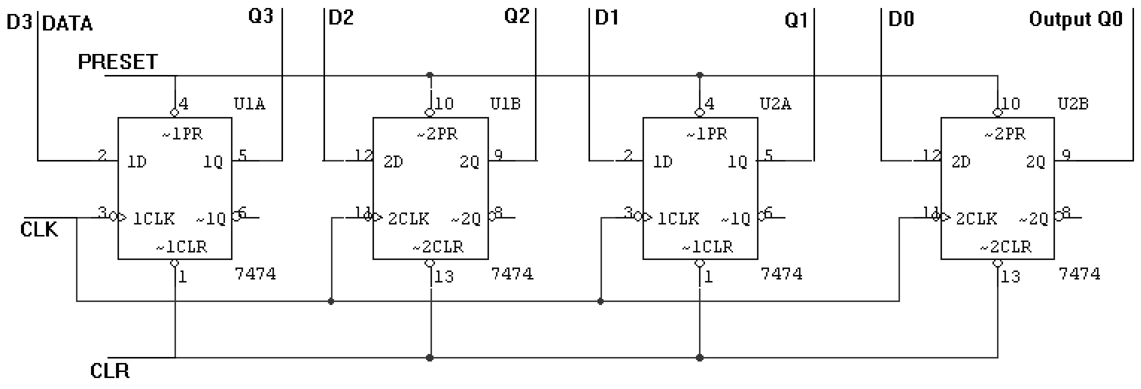
PARALLEL IN SERIAL OUT:



TRUTH TABLE:

CLK	Q3	Q2	Q1	Q0	O/P
0	1	0	0	1	1
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	1

LOGIC DIAGRAM:
PARALLEL IN PARALLEL OUT:



TRUTH TABLE:

CLK	DATA INPUT				OUTPUT			
	D_A	D_B	D_C	D_D	Q_A	Q_B	Q_C	Q_D
1	1	0	0	1	1	0	0	1
2	1	0	1	0	1	0	1	0

PROCEDURE

- Connections are given as per circuit diagram.
- Logical inputs are given as per circuit diagram.
- Observe the output and verify the truth table.

REVIEW QUESTIONS

1. In the transmitting shift register, you found that once the data are sent, they are no longer present in the sending shift register. How could you automatically reload the transmitted data?
2. Why is it necessary to use an edge-triggered device for a shift register?

Lab 12

Synchronous counters

OBJECTIVES

After completing this experiment, you will be able to:

- Analyze the count sequence of a synchronous counter.
- To design and implement 3 bit synchronous up/down counter.

COMPONENTS REQUIRED

- Two 7474, JK flip flop ICs
- One 7411, 3 I/P AND gate
- One 7432, 2 I/P OR gate
- One 7486, 2 I/P XOR gate
- One 7404, hex inverter

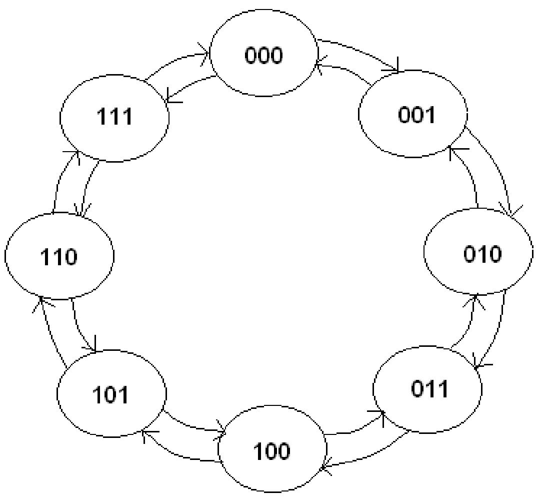
THEORY

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived.

Synchronous counters have all clock lines tied to a common clock, causing all flip-flops to change at the same time. For this reason, the time from the clock pulse until the next count transition is much faster than in a ripple counter. This greater speed reduces the problem of glitches (short, unwanted signals due to non-synchronous transitions) in the decoded outputs. However, glitches are not always eliminated, because stages with slightly different propagation delays can still have short intermediate states.

An up/down counter is one that is capable of progressing in increasing order or decreasing order through a certain sequence. An up/down counter is also called bidirectional counter. Usually up/down operation of the counter is controlled by up/down signal. When this signal is high counter goes through up sequence and when up/down signal is low counter follows reverse sequence.

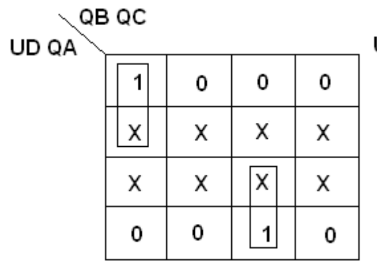
STATE DIAGRAM



CHARACTERISTICS TABLE

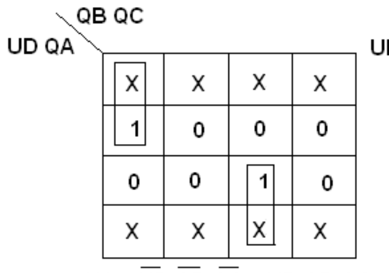
Q	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

K MAP



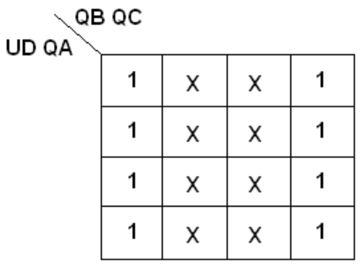
1	0	0	0
X	X	X	X
X	X	X	X
0	0	1	0

JA = $\overline{UD}\ \overline{QB}\ \overline{QC} + UD\ QB\ QC$



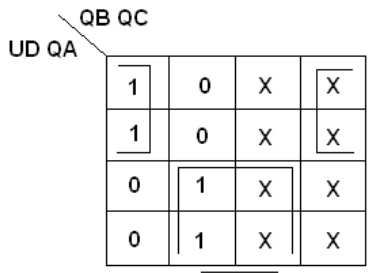
X	X	X	X
1	0	0	0
0	0	1	0
X	X	X	X

KA = $\overline{UD}\ \overline{QB}\ \overline{QC} + UD\ QB\ QC$



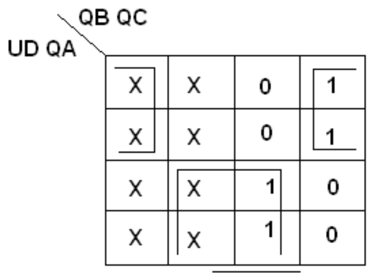
1	X	X	1
1	X	X	1
1	X	X	1
1	X	X	1

JC = 1



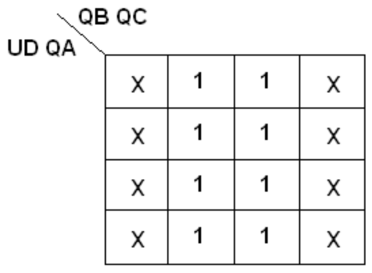
1	0	X	X
1	0	X	X
0	1	X	X
0	1	X	X

JB = $\overline{UD \oplus QC}$



X	X	0	1
X	X	0	1
X	X	1	0
X	X	1	0

KB = $(\overline{UD \oplus QC})$



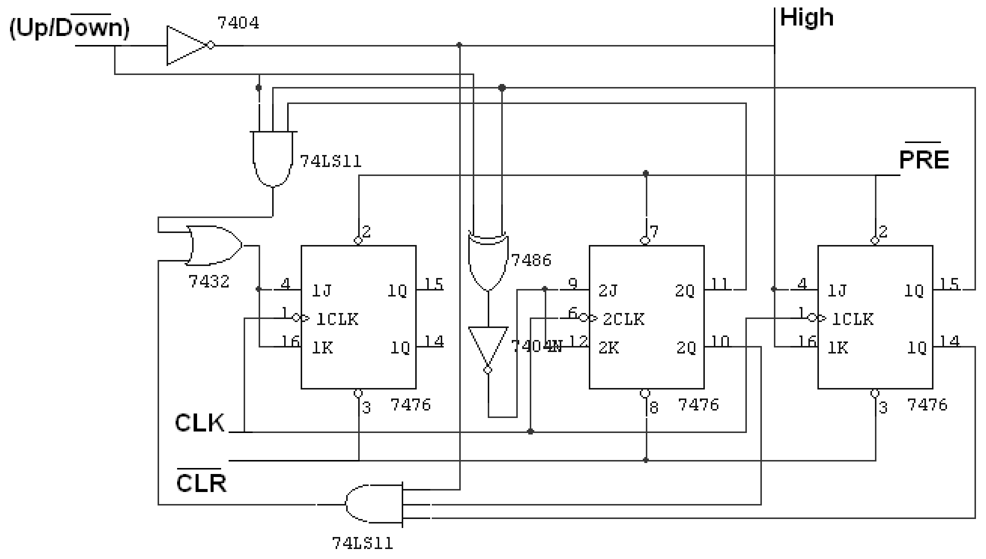
X	1	1	X
X	1	1	X
X	1	1	X
X	1	1	X

KC = 1

TRUTH TABLE

Input	Present State			Next State			A		B		C	
Up/Down	Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	1	1	1	1	X	1	X	1	X
0	1	1	1	1	1	0	X	0	X	0	X	1
0	1	1	0	1	0	1	X	0	X	1	1	X
0	1	0	1	1	0	0	X	0	0	X	X	1
0	1	0	0	0	1	1	X	1	1	X	1	X
0	0	1	1	0	1	0	0	X	X	0	X	1
0	0	1	0	0	0	1	0	X	X	1	1	X
0	0	0	1	0	0	0	0	X	0	X	X	1
1	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
1	0	1	0	0	1	1	0	X	X	0	1	X
1	0	1	1	1	0	0	1	X	X	1	X	1
1	1	0	0	1	0	1	X	0	0	X	1	X
1	1	0	1	1	1	0	X	0	1	X	X	1
1	1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	0	0	X	1	X	1	X	1

LOGIC DIAGRAM



PROCEDURE

- Connections are given as per circuit diagram.
- Logical inputs are given as per circuit diagram.
- Observe the output and verify the truth table.

REVIEW QUESTIONS

1) What is the difference between asynchronous and synchronous counters? Which counter will you prefer to use in your circuits and why? Also specify the main drawbacks of asynchronous counters.
2) Design a synchronous counter which count the sequence 0, -1, -2, -3 using D-flip flops. List all steps.
3) Draw the timing diagram for each flip flop output in part 2.

Lab 13

Introduction to Verilog HDL

OBJECTIVES

After doing this lab you should understand

- What is Verilog HDL
- How Verilog HDL Code is simulated on Modelsim