BCA I SEMESTER

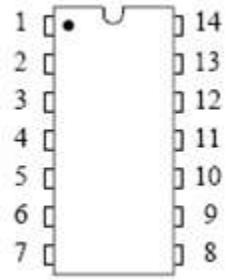
Subject Code: BCA-104P ELECTRONICS

LABORATORY MANUAL

Digital circuits are hardware components that are implemented using transistors and interconnections in complex semiconductor devices called integrated-circuits. Digital circuits work in binary logic domain which uses two discrete values, TRUE (High) and FALSE (Low). We can also refer to these values as 1(High) and 0 (Low). Logic-Gates are basic building blocks of digital circuits. Using these building blocks, complex functions or larger digital circuits can be built.

Integrated Chip (IC):

Each of the is available in 14 pin Dual-In-Line packages or DIPs. In a popular logic family called TTL (Transistor-Transistor Logic), the low logic level is assigned to 0V and the high logic level is assigned to 5V. Each IC or chip has an ID number. The pins are numbered as shown in figure below. Pin 1 is usually identified as the pin to the left of an indentation or cutout in one end of the chip that is visible when the chip is viewed from the top. Occasionally, it is also identified by a printed or indented dot placed just next to it.



1. Study of Logic Gates - AND, OR, NOT, NOR, NAND, XOR
2. Realisation of AND, OR and NOT Gates using Universal Gates
3. Design and realization of Half Adder/Subtractor using NAND gates.
4. Design and realization of Full Adder using Logic gates.
5. Design and realization of 4 bit adder/subtractor using IC 7483
6. Design and realization of BCD adder using IC 7483
7. Realisation of R-S Flip flop
8. Realisation of J-K Flip flop using IC 7400 and IC 7410
9. Realisation of T and D Flip flop using IC 7476
10. Implementation of SISO Shift Registers using Flip-flops. (IC 7476)
11. Implementation of SIPO Shift Registers using Flip-flops. (IC 7476)
12. Implementation of PISO Shift Registers using Flip-flops. (IC 7476)
13. Implementation of PIPO Shift Registers using Flip-flops. (IC 7476)
14. Design and implementation of odd and even parity checker generator using IC 74180

Laboratory Exercise 1: Study of Logic Gates
Objectives:

To investigate AND, OR, NOT, NAND and NOT gate operation. To study some fundamental laws of Boolean Algebra.

Introduction

This is the introductory laboratory session, to allow you to become familiar with very basic digital circuits and the equipment that you will use for the remainder of the experiments.

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. OR, AND and NOT are basic gates. NAND and NOR are known as universal gates. Basic gates form these gates.

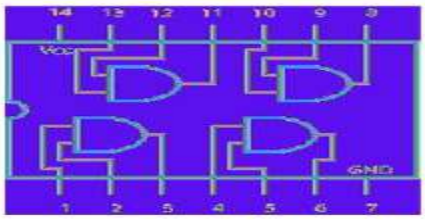
AND GATE: The AND gate performs a logical multiplication commonly known as AND function. The output is high when both the inputs are high. The output is low level when any one of the inputs is low.

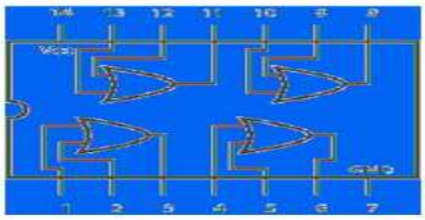
OR GATE: The OR gate performs a logical addition commonly known as OR function. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.

Equipment

The equipment you require is as follows: IC 7408 , IC 7432

Useful Chip Diagrams:





7408(AND) 7432(OR)

Section 1. AND Gate Implementation

(a) Connect one of the 2-input AND gates (7408) as shown in Figure 1. Remember to power the Vcc and GND terminals of the chip. Remember also to leave the turned off, until you are sure that your circuit is wired correctly.

Figure 1. The 2-input AND gate.

(b) Vary the inputs A and B (i.e. 0 and +5V) to obtain all the possible combinations and complete the truth table (as below) for the AND gate. Check the output F at the LED output.

A	B	Y
0	0	
0	1	
1	0	
1	1	

Section 2. Associative Laws

(a) Connect AND gates as shown in Figure 3 to implement the Boolean equations F = A(BC) and G=(AB)C.

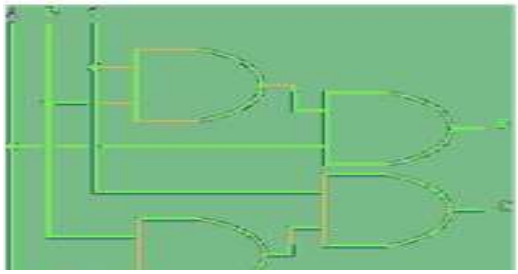


Figure 3. Proving The Associative Law.

(b) Vary inputs A, B, and C to obtain all of the possible combinations and to check the associative law: A(BC) = (AB)C. Fill in the truth table below:

A	B	C	F	G
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Section 3. Laws of Boolean Algebra

(a) To demonstrate the Distributive law, connect AND, OR gates as shown in Figure below. Vary the inputs A, B and C to obtain all possible combinations and check that the outputs F and G are identical.

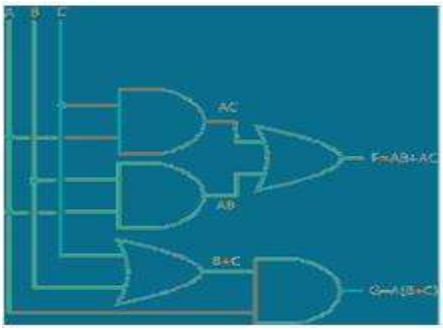


Figure 4. The Distributive Law.

Give the outputs in a truth table as shown below:

A	B	C	F	G
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

The NOT, NAND,NOR and XOR Gates:

Objectives:

To investigate NOT, NAND, NOR and XOR gate operation.

To study some the laws associated with the NOT, NAND and NOR operations. To investigate De Morgan's Theorem

Components Required : IC 7408 (AND), IC 7432 (OR), IC 7404 (NOT),

IC 7400(NAND), IC 7402(NOR), IC 7486 (XOR) Digital Trainer Kit

Theory

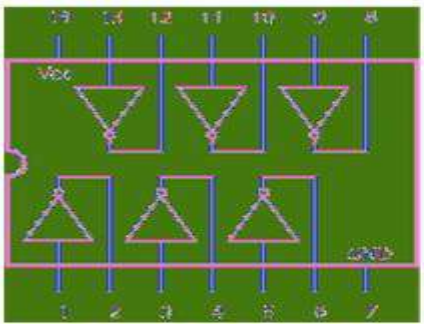
NOT GATE: The NOT gate is called an inverter. The output is high when the input is low. The output is low when the input is high.

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.

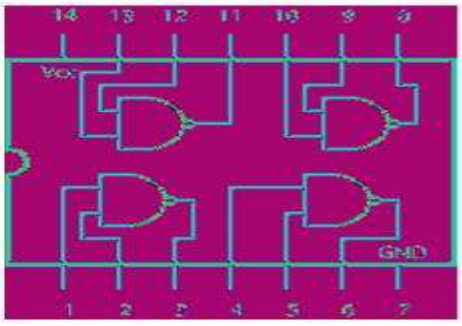
NOR GATE: The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.

XOR GATE: An Exclusive-OR (XOR) gate is gate with two or three or more inputs and one output. The output of a two-input XOR gate assumes a HIGH state if one and only one input assumes a HIGH state. This is equivalent to saying that the output is HIGH if either input X or input Y is HIGH exclusively, and LOW when both are 1 or 0 simultaneously.

Useful Chip Diagrams:



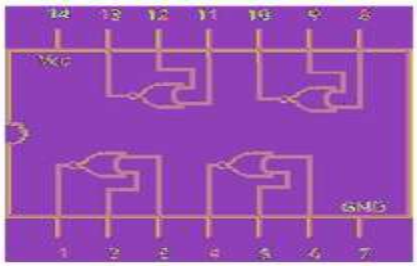
7404(NOT)

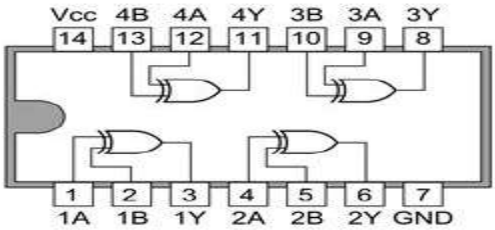


7400(NOR)

The Laboratory:

Section 1. NOT Gate Implementation





7402(NOR) 7486(X-OR)

(a) Connect one of the NOT gates (74LS04) as shown in Figure below. Remember to power the Vcc and GND terminals of the chip. Remember also to leave the Digital Trainer Kit turned off, until you are sure that your circuit is wired correctly.

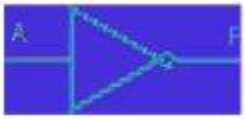


Figure The NOT gate

(b) Vary the input A to obtain all the possible combinations and complete the Truth table for the NOT gate. Check the output LED.

Section 2. De Morgan's Theorem

(a) Connect inputs A and B and their complements ~A and ~B to AND, OR gates as shown in Figure below. You will have to use NOT gates to obtain these states.

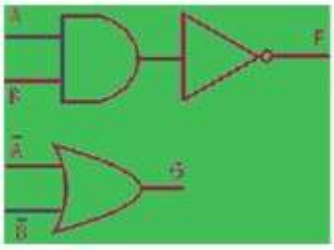


Figure. Proving De Morgan's Theorem.

(b) Vary inputs A and B and enter the output values into a truth table as show below.

A	B	F
0	0	
0	1	
1	0	
1	1	

Section 3. The implementation of the NAND gate.

(a) Connect one of the 2-input NAND gates as shown in Figure below.

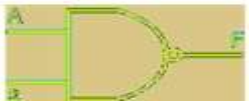
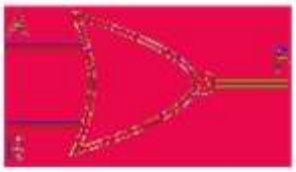


Figure. The NAND gate.

Enter the output values into a truth table as show below. Vary the inputs and record the output of the gate F and the actual voltage V. Examine the output of the NAND gate when one of the inputs is left floating (not connected). How can a NAND gate be used as an inverter?

A	B	F
0	0	
0	1	
1	0	
1	1	

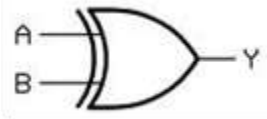
Section 4. The implementation of the NOR gate.



Enter the output values into a truth table for different values for A & B as show below. Vary the inputs and record the output of the gate F. were F is given by F = (A + B)'

A	B	F
0	0	
0	1	
1	0	
1	1	

Section 5: Implementation of X-OR Gate:



Enter the output values into a truth table for different values for A & B as show below. Vary the inputs and record the output of the gate F. were F is given by F = (A + B)'

A	B	F
0	0	
0	1	
1	0	
1	1	

Laboratory Exercise 2. Realization of Basic Gates using Universal Gates

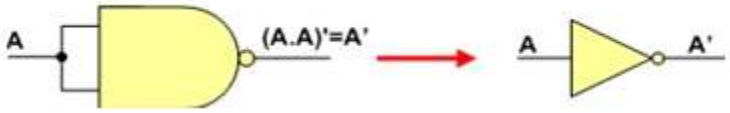
Universal gates are the ones which can be used for implementing any gate like AND, OR and NOT, or any combination of these basic gates; NAND and NOR gates are universal gates.

Theory

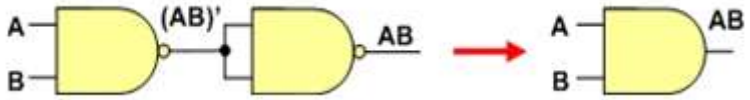
Any logic function can be implemented using NAND gates. To achieve this, first the logic function has to be written in Sum of Product (SOP) form. Once logic function is converted to SOP, then is very easy to implement using NAND gate. In other words any logic circuit with AND gates in first level and OR gates in second level can be converted into a NAND-NAND gate circuit.

Any logic function can be implemented using NOR gates. To achieve this, first the logic function has to be written in Product of Sum (POS) form. Once it is converted to POS, then it's very easy to implement using NOR gate. In other words any logic circuit with OR gates in first level and AND gates in second level can be converted into a NOR-NOR gate circuit.

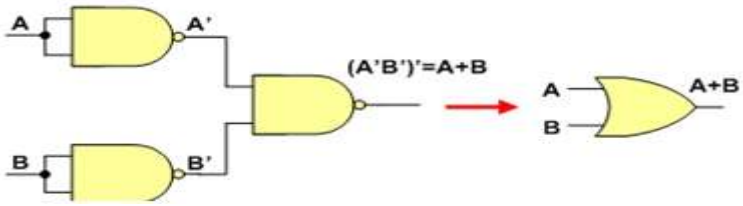
NOT - Using NAND



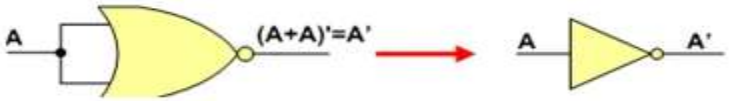
AND - Using NAND



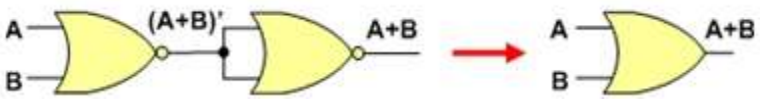
OR - Using NAND



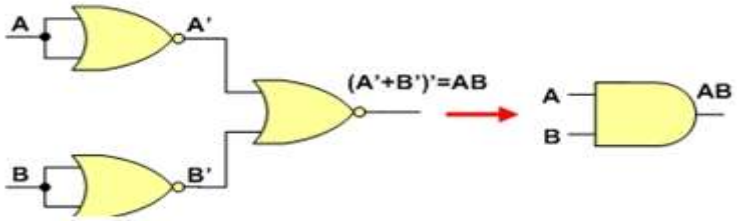
NOT - Using NOR



OR - Using NOR



AND - Using NOR



Truth Table to realize AND and OR using Universal Gates

A	B	F
0	0	
0	1	
1	0	
1	1	

Truth Table to realize NOT using Universal Gates

A	S
0	
1	

Laboratory Exercise 3. Half Adder / Subtractor 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. Above circuit is called as a carry signal from the addition of the less significant bits sum from the X-OR 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 X-OR Gate, carry output will be taken from OR Gate.

HALF SUBTRACTOR:

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

Objectives:

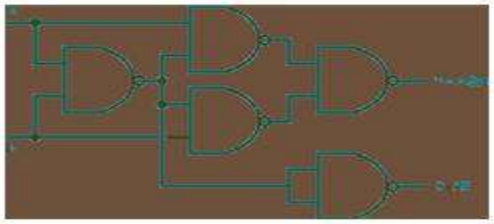
To implement a half adder using basic gates
To implement a half subtractor using Universal gate (NAND)

Equipment

The equipment you require is as follows: IC 7404, IC 7400, IC 7483

A half adder is a logical circuit that performs an addition operation on two one-bit binary numbers often written as A and B. The half adder output is a sum of the two inputs usually represented with the signals Cout and Sum

Section 1. The Half Adder



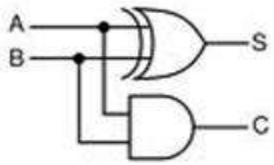


Figure: The Half adder circuit (a)Using NAND (b) Using X-OR

(b) Vary the inputs A and B to obtain all the possible combinations and complete a Truth table for the sum output S and the carry output C.

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

Section 2. The Half Subtracror (NAND Gates)

The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow).

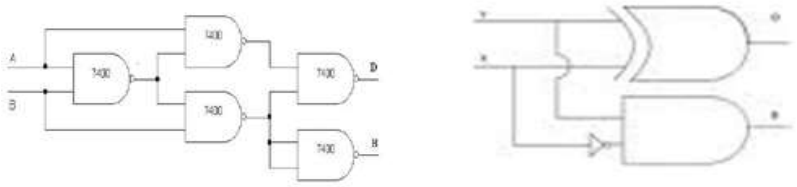


Figure: The Half Subtractor circuit (a)Using NAND (b) Using X-OR

Vary the inputs A and B to obtain all the possible combinations and complete a Truth table for the sum output S and the carry output C.

A	**B**	**D**	**B**
0	0		
0	1		
1	0		
1	1		

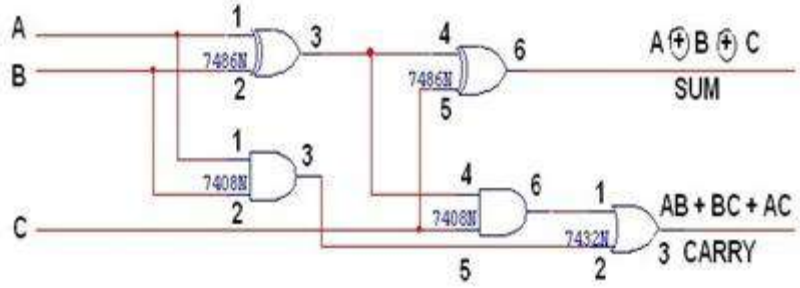
Laboratory 4. The Full Adder

Components: IC 7486, IC 7408, IC 7432 Theory:

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 X-OR Gate, carry output will be taken from OR Gate.

a) Using Logic Gates:



(b) Using Universal Gate (NAND)

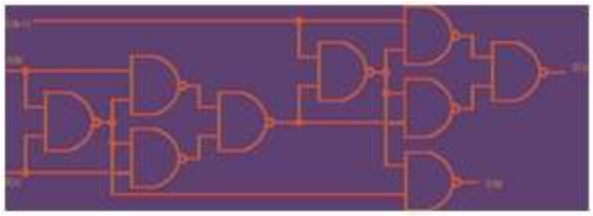


Figure. The Full Adder Circuit

(b) Vary inputs Ak Bk and Ck-1 and complete a Truth table for the circuit.

A	B	C0	S1	C1
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Laboratory 5 : 4-bit Adder / Subtractor Components : IC 7483, IC 7486

Theory

4 BIT BINARY ADDERS:

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of next full adder in chain. The augends bits of 'A' and the addend bits of 'B' are designated by subscript numbers from right to left, with subscript 0 denoting the least significant bits. The carries are connected in chain through the full adder. The input carry to the adder is C0 and it ripples through the full adder to the output carry C4.

4 BIT BINARY SUBTRACTOR:

The circuit for subtracting A-B consists of an adder with inverters, placed between each data input 'B' and the corresponding input of full adder. The input carry C0 must be equal to 1 when performing subtraction.

The 74LS83 is a high speed 4-bit Full Adder. It accepts two 4-bit words (A1 to A4) and (B1 to B4) and a carry input (C0). It generates the binary sum outputs (S1 to S4) and the carry output (C4) from the most significant bit. Connect the two binary numbers shown below to the inputs of the 4-bit parallel adder. Connect the carry input C0 to 0.

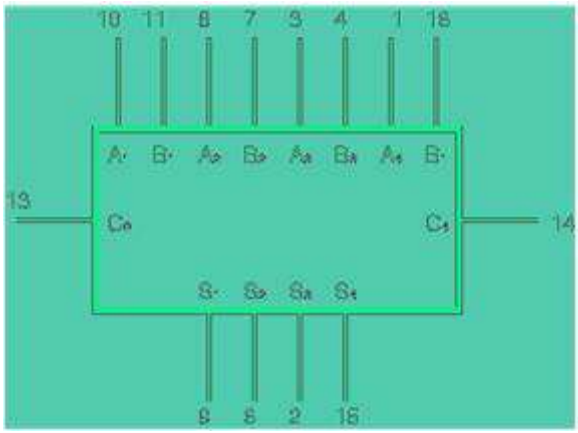
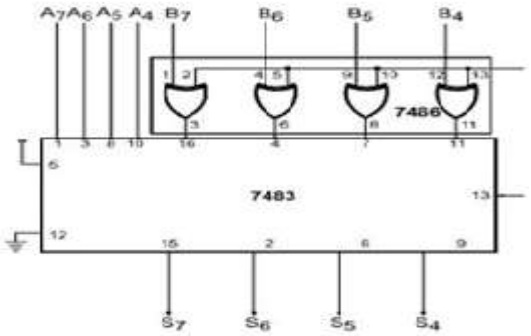


Figure. The Four Bit Adder

The addition and subtraction operation can be combined into one circuit with one common binary adder. The mode input M controls the operation. When M=0, the circuit is adder circuit. When M=1, it becomes subtractor.

PIN DIAGRAM (IC 7483):



Sub

Cin

Check the operation of the Four Bit adder by adding the numbers:

(a) 0110 + 0101 (b) 1011 + 0011 (c) 1110 + 0100 (d) 1111 + 1111

Laboratory 6: 4-BIT BCD ADDER:

Components: IC 7483, IC 7408, IC 7432

Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than 19, the 1 in the sum being an input carry. The output of two decimal digits must be represented in BCD and should appear in the form listed in the columns.

A BCD adder that adds 2 BCD digits and produce a sum digit in BCD. The 2 decimal digits, together with the input carry, are first added in the top 4 bit adder to produce the binary sum.

Objective: To design and implement BCD adder using 4 bit binary adder IC 7483.

Apparatus Required: Two IC 7483

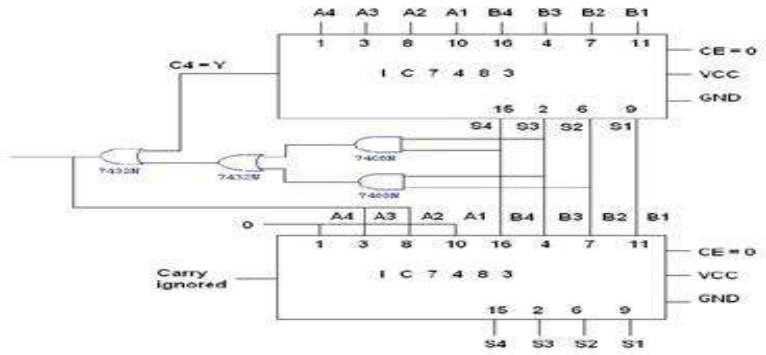
Theory: BCD Addition

Binary Coded Decimal is a method of using binary digits to represent the decimal digits 0 through 9. The valid BCD numbers are (0000 to 1001)BCD. Each digit of the decimal number will be represented by its four bit binary equivalent.

Ex: (127)10 - BCD equivalent (0001 0010 0111)2. In BCD addition the following three cases are observed:

1. The resulting BCD number equal to less than (1001)BCD.
2. The resulting BCD number greater than (1001)BCD.
3. Carry is generated in the BCD addition.

For case 2 and 3, the result is added with correction factor (0110) BCD so that the result is in valid BCD number.



The two BCD inputs to be added are applied at inputs A and B of the first binary adder IC 7483. The sum output of the first binary adder is given to the B input of the second binary adder. The A input of the binary adder is given (0110)BCD when a carry is generated from the first adder or when sum from the first binary adder is greater than (0110)BCD, else A input is (0000)BCD. The following Boolean expression is used to find whether (0110)BCD or (0000)BCD needs to

be applied to the A input, Cout = Cout1 + S4 (S3 + S2). Where S4, S3, S2, S1 are the sum of the BCD from the first binary adder with S4 as the MSB and S1 as the LSB. Cout1 is the carry output from the first binary adder.

Procedure:

1. Verify the gates.

2. Make the connections as per the circuit diagram.

3. Apply and verify the various combination of input according to the truth table for BCD adder.

Laboratory Exercise 7: RS Flip Flop

Objectives: To study the fundamentals of basic memory units. To become familiar with various types of flip-flops. To implement a data register.

Components: IC 7400

Introduction:

In this laboratory we will build on concepts that we examined in the previous laboratory sessions. We are now examining flip-flops. It is likely that we have not examined flip-flops in lectures at this point, but hopefully the laboratory exercise will help explain the concepts as we go along.

A flip-flop maintains a binary state indefinitely (as long as the circuit is powered) until it receives an input signal to switch states. There are various types of flip flops that differ in the number of inputs they possess and how the inputs affect the binary state.

THEORY

RS flip-flop is also called Synchronous flip-flop. That means that this flip-flop is concerned with time. Digital circuits can have a concept of time using a clock signal. The clock signal simply goes from low-to-high and high-to-low in a short period of time.

Apparatus Required: IC 7400(NAND), 7402 (NOR)

Section 1. The Asynchronous RS Flip-Flop

(a) Connect the two NAND gates as shown in Figure.

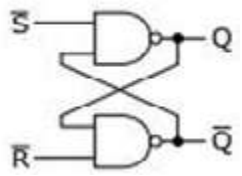


Figure. An RS Flip-Flop created using NAND gates.

(b) Vary the inputs R and S to obtain all the possible combinations for Q and /Q.

R	S	Q	~Q
0	0		
0	1		
1	0		
1	1		

Section 2. The Synchronous Flip-Flop

(a) Synchronous means that this flip-flop is concerned with time. Digital circuits can have a concept of time using a clock signal. The clock signal simply goes from lowto-high and high-to-low in a short period of time.

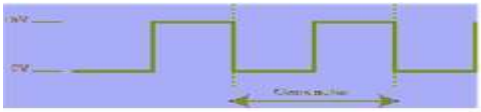


Figure. A typical clock signal.

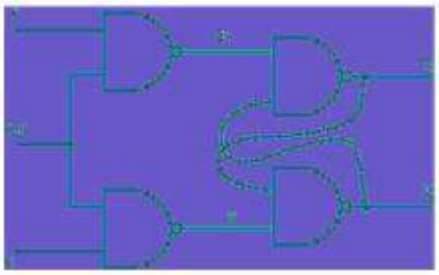


Figure. The Synchronous Flip Flop using NAND (IC 7400)

Implement the circuit in Figure . You can simulate a clock signal by moving the clock line from low to high and back again to low.

(b) Vary inputs R and S and apply the clock pulse. Write the output states into a table as below:

Qn	/Qn	R	S	Qn+1	/Qn+1
0	1	0	0		
1	0	0	0		
0	1	0	1		
1	0	0	1		
0	1	1	0		
1	0	1	0		
0	1	1	1		
1	0	1	1		

Laboratory Exercise 8: J-K Flip-Flop

The JK flip- flop is another modification of the RS flip-flop which has its own way of dealing with the RS flip-flop's undesirable input combination (when S= 1 and R= 1). The JK flip-flop has 2 inputs J and K, each ANDed with the clock pulse and its corresponding NOR output. The schematic for the JK flip- flop is shown below.

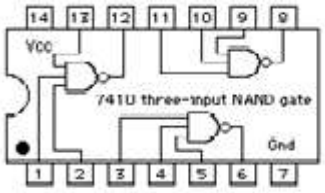
Components: IC 7400, IC 7410 (3 input NAND gate)

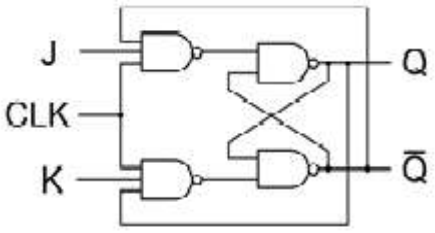
Theory:

What used to be the S and R inputs are now called the J and K inputs, respectively. The old two-input AND gates have been replaced with 3-input AND gates, and the third input of each gate receives feedback from the Q and not-Q outputs. What this does for us is permit the J input to have effect only when the circuit is reset, and permit the K input to have effect only when the circuit is set. In other words, the two inputs are interlocked, to use a relay logic term, so that they cannot both be activated simultaneously. If the circuit is "set," the J input is inhibited by the 0 status of not-Q through the lower AND gate; if the circuit is "reset," the K input is inhibited by the 0 status of Q through the upper AND gate.

When both J and K inputs are 1, however, something unique happens. Because of the selective inhibiting action of those 3-input AND gates, a "set" state inhibits input J so that the flip-flop acts as if J=0 while K=1 when in fact both are 1. On the next clock pulse, the outputs will switch ("toggle") from set (Q=1 and not-Q=0) to reset (Q=0 and not-Q=1). Conversely, a "reset" state inhibits input K so that the flip-flop acts as if J=1 and K=0 when in fact both are 1. The next clock pulse toggles the circuit again from reset to set.

The end result is that the S-R flip-flop's "invalid" state is eliminated (along with the race condition it engendered) and we get a useful feature as a bonus: the ability to toggle between the two (bistable) output states with every transition of the clock input signal.



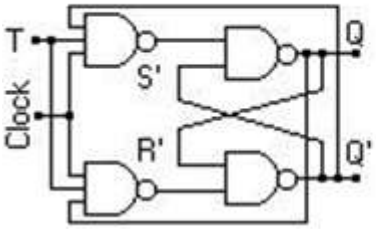


C	J	K	Q	$\overline{Q}$
⌐	0	0	latch	latch
⌐	0	1	0	1
⌐	1	0	1	0
⌐	1	1	toggle	toggle
x	0	0	latch	latch
x	0	1	latch	latch
x	1	0	latch	latch
x	1	1	latch	latch

Laboratory Exercise 9: T and D Flip-Flop T Flip-flop

The toggle, or T, flip-flop is a bi-stable device that changes state on command from a common input terminal.

a) From JK Flip flop:



T	Q	Q_{next}	Comment
1	0	1	toggle
1	1	0	toggle

JK type flip-flop connected as a toggle type. On each clock pulse positive going edge, Q will go to the state bar Q was before the clock pulse arrived. Remember that bar Q is the opposite level to Q. Therefore Q will toggle.

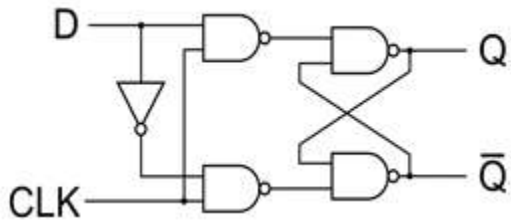
Exercise:

1. Verify the gates.
2. Make the connections as per the circuit diagram.
3. Apply and verify the various combination of input according to the truth table.

D Flip Flop

a) Using Logic gates

Since the enable input on a gated S-R latch provides a way to latch the Q and not-Q outputs without regard to the status of S or R, we can eliminate one of those inputs to create a multivibrator latch circuit with no "illegal" input states. Such a circuit is called a D latch.



E	D	Q	$\overline{Q}$
0	0	latch	latch
0	1	latch	latch
1	0	0	1
1	1	1	0

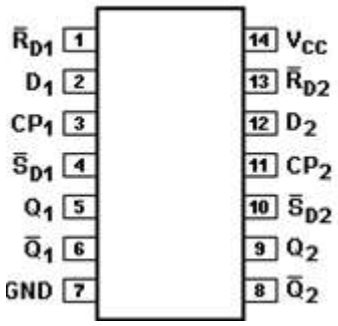
Figure: The D-type flip-flop (logic gates)

Note that the R input has been replaced with the complement (inversion) of the old S input, and the S input has been renamed to D. As with the gated S-R latch, the D latch will not respond to a signal input if the enable input is 0 -- it simply stays latched in its last state. When the enable input is 1, however, the Q output follows the D input.

Since the R input of the S-R circuitry has been done away with, this latch has no "invalid" or "illegal" state. Q and not-Q are always opposite of one another.

b) Using Dual D-flip flop (IC 7474)

OPERATING	INPUTS				OUTPUTS	
MODE	SD	RD	CP	D	Q	Q
Asyn. Set	L	H	X	X	H	L
Asyn. Reset (Clear)	H	L	X	X	L	H
Undetermined (a)	L	L	X	X	H	L
Load "1" (Set)	H	H	1	h	H	L
Load "0" (Reset)	H	H	1	l	L	H

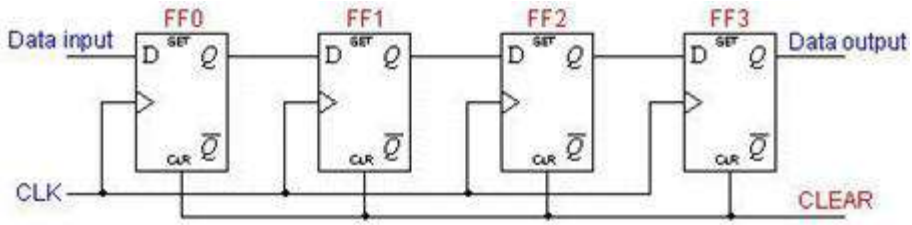


Laboratory Exercise 10 - 13 : Shift Registers

Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flipflop becomes the input of the next flip-flop. Most of the registers possess no characteristic internal sequence of states. All flip-flops are driven by a common clock, and all are set or reset simultaneously. The storage capacity of a register is the total number of bits (1 or 0) of digital data it can retain. Each stage (flip flop) in a shift register represents one bit of storage capacity. Therefore the number of stages in a register determines its storage capacity.

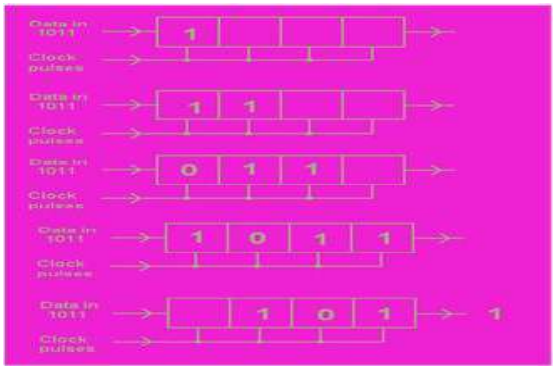
SISO:

Connect up the four bit data register as shown in Figure below. An n-bit binary word can be stored by n such flip-flops; called a n-bit register.

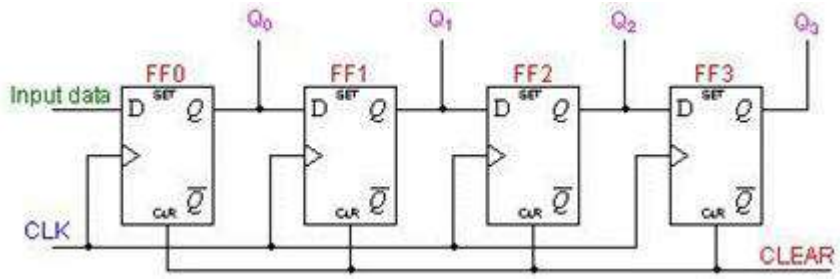


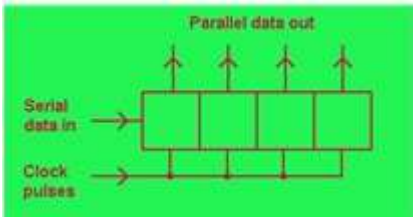
At the start, the contents of the register can be set to zero by means of the CLEAR line. If a 1 is applied to the input of the first flip-flop, then upon the arrival of the first clock pulse, this 1 is transferred to the output of flip-flop 1 (input of flip-flop 2).

After four clock pulses this 1 will be at the output of flip-flop 4. In this manner, a four bit number can be stored in the register. After four more clock pulses, this data will be shifted out of the register.



SIPO :

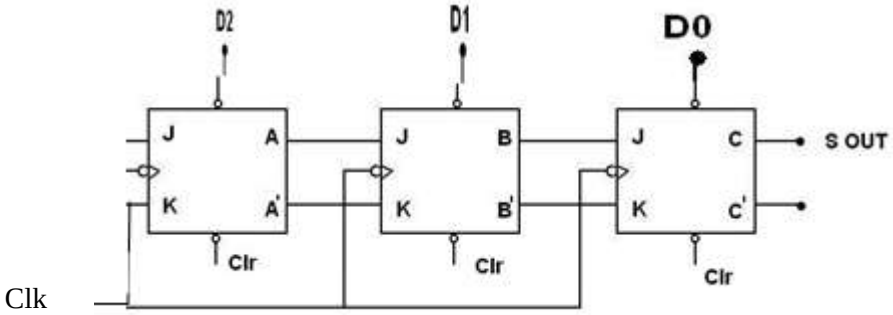


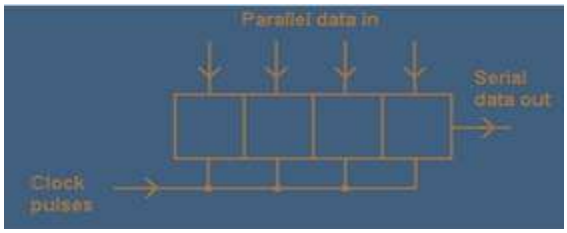


Data is fed into the SERIAL IN/PARALLEL OUT shift register bit by bit, in the same way as for the SISO shift register. However the four bits are all shifted out simultaneously, in parallel, as one word.

Clock Pulse No	FF0	FF1	FF2	FF3
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0

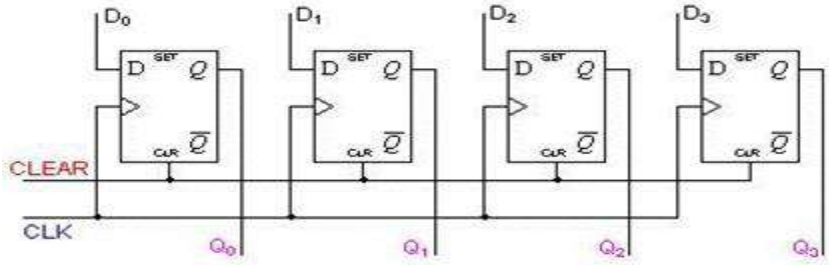
PISO

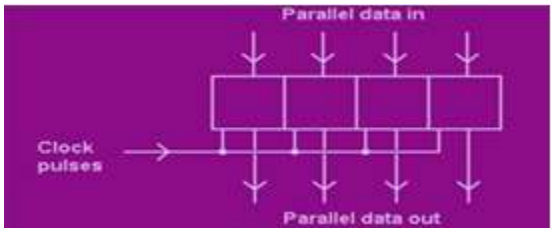




With the PARALLEL IN/SERIAL OUT shift register, four bits are shifted into the register simultaneously, in parallel. They are then clocked out, one after the other, in serial form.

PIPO

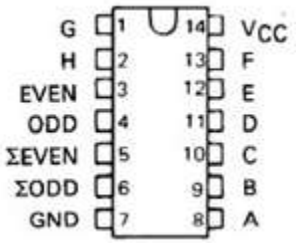




The PARALLEL IN/PARALLEL OUT shift register is loaded with four bits simultaneously, in parallel. They are also clocked out simultaneously, in parallel.

Laboratory Exercise 14: Odd and Even Parity Checker

Components: IC 74180 (8 bit parity checker/generator)



Theory:

When digital data is transmitted from one location to another, it is necessary to know at the receiving end, Whether data received is free from errors. To help make the transmission accurate, special error detection methods are used. To detect errors there must be constant check on the data being transmitted. To check accuracy of the data an extra bit can be generated and transmitted along with the data. This bit is called the parity bit. A parity bit is used for detecting errors during transmission of binary information.

Parity generators are circuits that accept an n-1 bit data stream and generate an extra bit that is transmitted with the bit stream. This extra bit is referred to as parity bit. In an even parity bit scheme, the parity bit is '1' if there are odd number of 1's in the data stream and the parity bit is '0' if there are even number of 1's in the data stream. In the case of odd parity bit scheme, the reverse happens, that is the parity bit is '0' for odd number of 1's and '1' for even number of 1's in the bit stream.

The 8 inputs for the Parity checker is given by A through H in the pin diagram and the same is referred as 0 through 7 in the truth table.

TRUTH TABLE

INPUTS			OUTPUTS	
Σ OF 1's AT 0 THRU 7	EVEN	ODD	Σ EVEN	Σ ODD
EVEN	H	L	H	L
ODD	H	L	L	H
EVEN	L	H	L	H
ODD	L	H	H	L
X	H	H	L	L
X	L	L	H	H