

Digital

Electronics Laboratory Experiments



Departement Of Physics

The Islamia Unibersity Bahawalpur

DIGITAL
ELECTRONICS
LAB

**Let's
Begin**

**Digital
niarsu**

Supervised By:-

Dr. Sheikh Aftab Ahmed Sb.

Written & Composed By:-

Raza-ur Rehman Hashmi

(cell # 03447169879)

Experiment 1

Constructing A Logic Probe

Objectives

- ❑ Construct a simple logic probe using an IC 7404.
- ❑ Use this logic probe to test another circuit.
- ❑ Measure logic levels with the digital multimeter (DMM) and an oscilloscope, and compare these with valid input logic levels.

Components

7404 hex inverter
 Two LEDs
 Two signal diodes(1N4007)
 Resistors: three 330Ω , one $2.0k\Omega$
 $1k\Omega$ potentiometer

Summary

Digital circuits have two discrete voltage levels to represent the binary digits (bits) 1 and 0. All digital circuits are switch circuits that use high speed transistors to represent either an ON or an OFF condition. The input logic levels for TTL are Shown Below

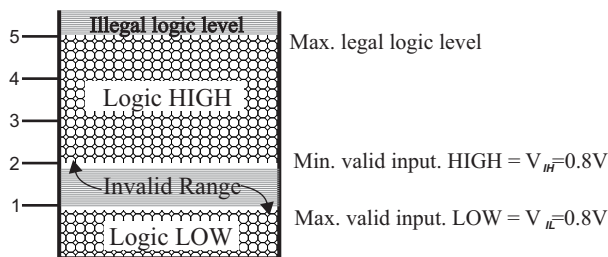


Fig 1: TTL logic levels.

For any IC to function properly, power and ground must be connected. But in practice the power and ground are frequently omitted from diagrams of the logic circuits

An IC 7404 Hex Inverter is shown below

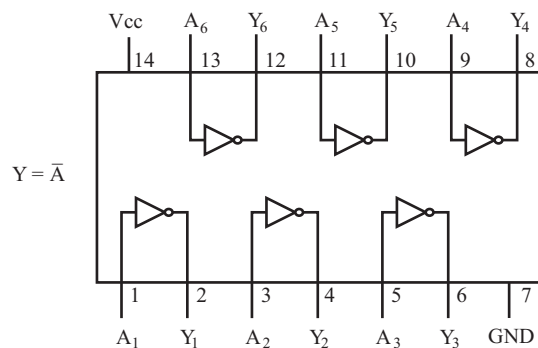


Fig 2 : Connection diagram

Pins are numbered counterclockwise from the top, starting with a circle at the top or next to pin 1 as shown in Fig. 3.

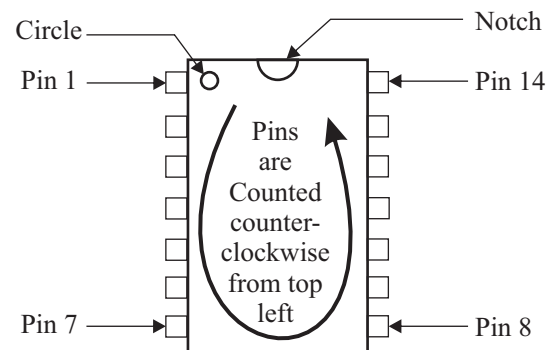


Fig 3 : Numbering of pins.

Logic probes are useful for detecting the presence of a HIGH or a LOW logic level in a circuit. The logic probe in this experiment is a simple one designed only to illustrate the use of this tool and the wiring of integrated circuits. The simple probe is shown in the Fig 4.

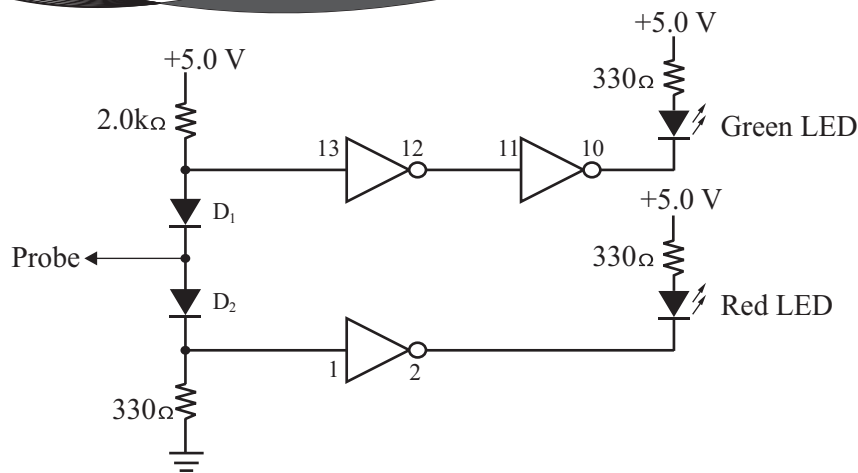


Fig 4 : Simple logic probe

This works as follows:

If the probe is not connected, the top inverter is pulled HIGH (through the 2.0k resistor) and the bottom inverter is pulled LOW. As a result, both outputs are HIGH and neither LED is on. (A Low is required to turn on either LED). If the probe input is connected to a voltage above approximately 2.0V, the voltage at the input of the lower inverter is interpreted as a logic HIGH through diode D. As a result, the output of the lower Red LED, representing a HIGH input, turns on. If the probe input is connected to a voltage below approximately 0.8V, the upper input inverter is pulled below the logic threshold for a LOW, and the output inverter is LOW. Then the upper green LED, which represents a logic LOW input turn on.

Procedure

1. Using the pin numbers shown, construct the simple logic probe circuit shown in Fig 4.

2. Test your circuit by connecting the probe to +5.0V and then to ground. One of the LEDs should light to indicate a HIGH, the other, a LOW. When the probe is not connected, neither LED should be on. The example of wiring of the logic probe as shown in Fig. 4.

Do You Know?

Signal diodes are marked on the cathode side with a line. The LEDs have a flat spot on the cathode side or longer element inside the diode.

Tip

If the circuit does not work double-check your wiring and the direction of the diode

3. You can test the HIGH and LOW threshold voltages of your logic probe with the circuit shown in fig 5(b). Connect the logic probe to a 0-5 V DC variable, as shown. Vary the voltage to find the HIGH and LOW thresholds. Use DMM to measure the threshold voltages. Record the thresholds V_{IL} & V_{IH} in the report .

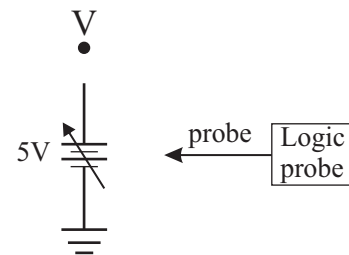


Fig 5(b) Test circuit to determine logic thresholds

4. In the last step, your should have observed that the thresholds for the logic probe are very close to the TTL specifications as given in the Fig 1, If so, you can now use the probe to test the logic for logic gates including various inverter circuits. Remember that there are six independent inverters in a 7404. (but they share a common power supply). Connect your logic probe to the output (pin 4) and observe the output logic when the input is LOW, OPEN, and is HIGH. Record your observations for these three cases in Table 1. An open input is not desirable because of potential noise problems.

5. Connect two inverters in series (cascade) as shown in figure 6(a). Move the logic probe to the output of the second inverter (pin 6). Check the OPEN, and HIGH as before. Record your observations for these three cases in Table 1.

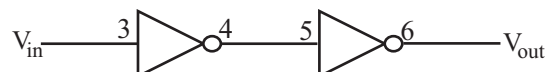


Fig 6(a) series inverters

6. Connect the two inverters as cross-coupled inverters as shown in Figure 6(b).

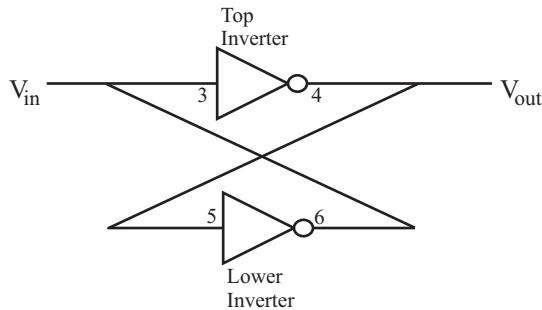


Fig 6(b) cross-coupled inverters

This is the basic latch circuit, the most basic form of memory. This arrangement is not the best way to implement latch but serves to illustrate the concept. This latch works as follows:

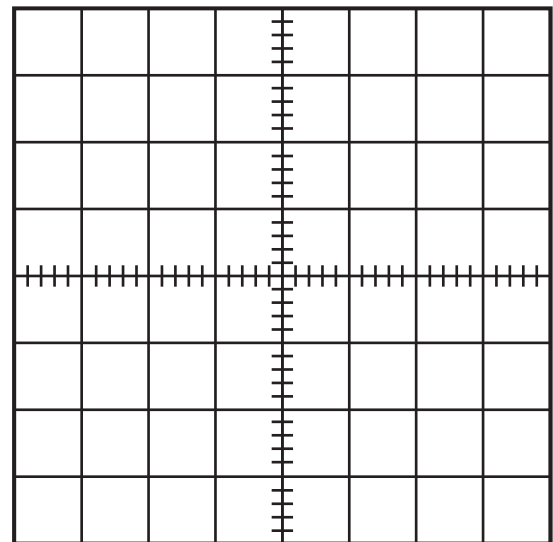
The input signal is first inverted by the top inverter. The original logic is inverted a second time by the lower inverter which restores the logic back to the original input logic level. This is “feed back” signal which forms the latch. If the input is now removed, the feedback signal keeps the input from changing and the circuit remains stable.

7. Connect the logic probe to pin 4 of the latch circuit. Then momentarily touch V_{in} (i.e pin 3) to ground. Observe the output logic and record it (HIGH, LOW, or INVALID) in Table 2.
8. Touch the input(pin 3) to +5.0V, test the output again, and record the logic in Table 2.
9. Place a fault in the circuit of Figure 6(b) by removing the wire that is connected to pin 5, the input of the lower inverter. Now momentarily touch the input pin 3, to ground. Test the logic levels at each point in the circuit and record them in Table 2.
10. An open circuit on the input of TTL logic has an invalid logic level. Even though it is invalid, it acts as a logic HIGH at the input to the gate. (However, open circuits should never be used as a means of connecting an input to a constant HIGH). In this step, repeat Step 9 but use a DMM to measure the actual voltages at each pin. Record the data in Table 2.

11. In order to practice with oscilloscope, repeat the measurements of Step 9 using the oscilloscope. You may want to review the procedure for making DC voltage measurements with the oscilloscope in the Oscilloscope Guide. Record the measured voltages in Table 2.

For Further Investigation

In this investigation, you will check the logic inverter with a pulse waveform. Set up the pulse generator with a 1kHz TTL compatible pulse. Set up the series inverter shown in the Fig 6(a). Then compare the wave forms on the input and on the output of the circuit. Sketch the wave forms in Plot1: be sure and label the voltage and time. Are the wave forms identical? If not, why not? Explain your observations in your Report.



Plot 1: Input and output waveforms for two series inverters

Report Of Experiment 1

Data and Observations:

Step 3: Logic thresholds: HIGH_____ V_{IH} LOW_____ V_{IL}

TABLE 1

Step		Input is LOW	Output Logic Level Input is OPEN	Input is HIGH
4	One Inverter			
5	Two series Inverters			

TABLE 2

Step		Input Logic Level(pin 3)	Output Logic Level(pin 4)	Logic Level (pin 5)	Logic Level (pin 6)
7	V momentarily on ground.				
8	V momentarily on +5.0V.				
9	Fault condition: open at pin 5.				
10	Voltages with fault(DMM):	V	V	V	V
11	Voltages with fault(scope):	V	V	V	V

Results and Conclusion:

Experiment 2

Number System

Objectives

Convert binary or binary coded decimal (BCD) numbers to decimal.

Construct a portion of a digital system that decodes a BCD number and displays it on a seven-segment display.

Components

Four LEDs
IC 7447A BCD/decimal decoder
seven-segment display
Four logic switches
Resistor: seven 330 Ω , one 1.0 k Ω

Summary

The number of symbols in a number system is called the base of the system. The decimal number system uses ten counting symbols, the digits 0 through 9, to represent quantities. Thus it is a base ten system. In this system, we represent quantities larger than 9 by using positional weighting of the digits. The column, that a digit occupies indicates the weight of that digit in determining the value of the number. The base 10 number system is weighted system. Because each column has a value associated with it.

Digital systems use two states to represent quantities and thus are binary in nature. The binary counting system has a base of two and uses only the number 0 or 1. It too is a weighted counting system, with each column value worth twice the value of the column to the immediate right. Because binary numbers have only two digits, large numbers expressed in binary require a long string of 0s and 1s. Other systems, which are related to binary in a simple way, are often used to simplify these numbers. These systems include octal, hexadecimal, and BCD.

The octal number system is a weighted number system using the digits 0 through 7. The column values in octal are worth 8 times that of the column to the immediate right. You convert from binary to octal by arranging the binary number in groups of 3 bits, starting at the binary point, and writing the octal symbol for each binary group. You can reverse the procedure to convert from octal to binary. Simply write an equivalent 3-bit binary number for each octal character.

The hexadecimal system is a weighted number system using 16 characters. The column values in hexadecimal are worth 16 times that of the column to the immediate right. The characters are the numbers 0 through 9 and the first six letters of alphabet, A through F. Letters were chosen because of their sequence. You convert binary numbers to hexadecimal numbers by arranging the binary numbers to hexadecimal numbers by arranging the binary number into 4-bit groups, starting at the binary point. Then write the next symbol for each group of 4-bits. You convert hex numbers to binary by reversing the procedure. That is, write an equivalent 4-bit binary number for each hexadecimal character,

The BCD system uses four binary bits to represent each decimal digit. It is a convenient code because it allows ready conversion from base ten to a code that a machine can understand; however, it is wasteful of bits.

Do You Know?

A 4-bit binary number could represent the number 0 to 15, but in BCD it represents the numbers 0 through 9.

“The binary representations of the numbers 10 through 15 are not used in BCD and are invalid.”

In this experiment we use the 7447 IC to for the decoding of BCD to 7-Segment, shown in figure 1.

7447

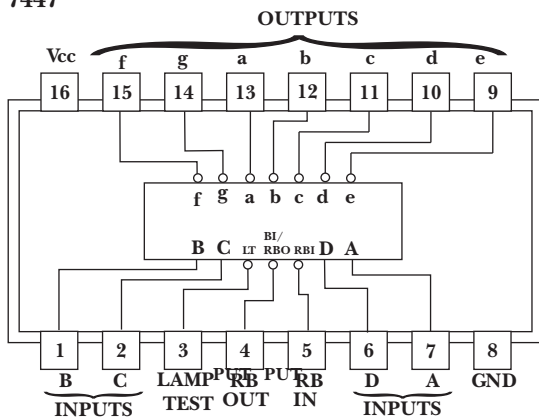


Fig 1: The 7447 DM74LS47 BCD to 7-Segment Decoder

The pin configuration for the Seven Segment Display is shown in Figure 2.

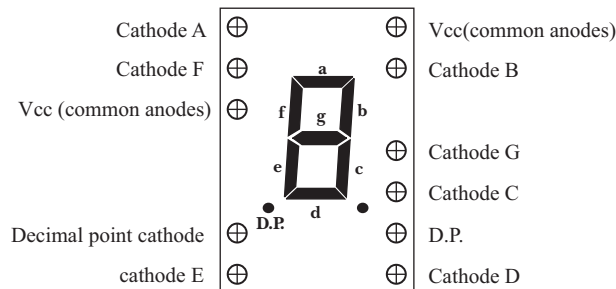
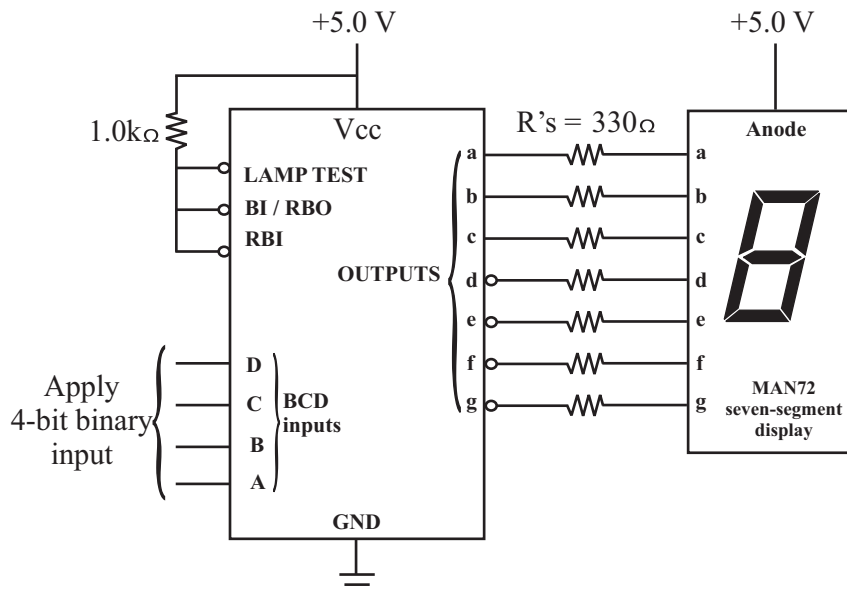


Fig 2: The Seven-Segment display

Procedure

1. Take a moment to review "Circuit Wiring" in the Introduction before constructing this experiment. It is a good idea to write the pin numbers directly on the schematic before you begin wiring.

2. Construct the circuit carefully as shown in the Figure 3.



3. Connect the Lamp test, BI/RBO, and RBI inputs through a 1kΩ resistor to +5.0 V. This is a pull-up resistor, used to assure a solid logic HIGH is present at these inputs. Connect terminals D, C, B, A to four logic switches for applying all possible input combinations of 0's and 1's (Table 1).

4. When you have completed the wiring, apply power, and test the circuit by setting each switch combination listed in Table 1. The last six codes are invalid BCD codes; however, you can set the switch combinations in binary and observe the display. It will show a unique display for each of the invalid codes. Complete the table by showing the appearance of the seven-segment display in the output column.

5. In this step, you will insert some simulated "troubles" in the circuit and observe the effect of these troubles on the output. The troubles are listed in Table 2 of the report. Insert the given trouble, and test its effect. Indicate what effect it has on the output. Assume each trouble is independent of others; that is, restore the circuit to its normal operating condition after each test.

6. The display you have built for this experiment is satisfactory only for showing a single decimal digit at a time. You could show more digits by simply replicating the circuit for as many digit as needed, although this isn't the most efficient way to make larger displays. With a larger number of digits, it is useful to blank (turn off) leading zeros in a number.

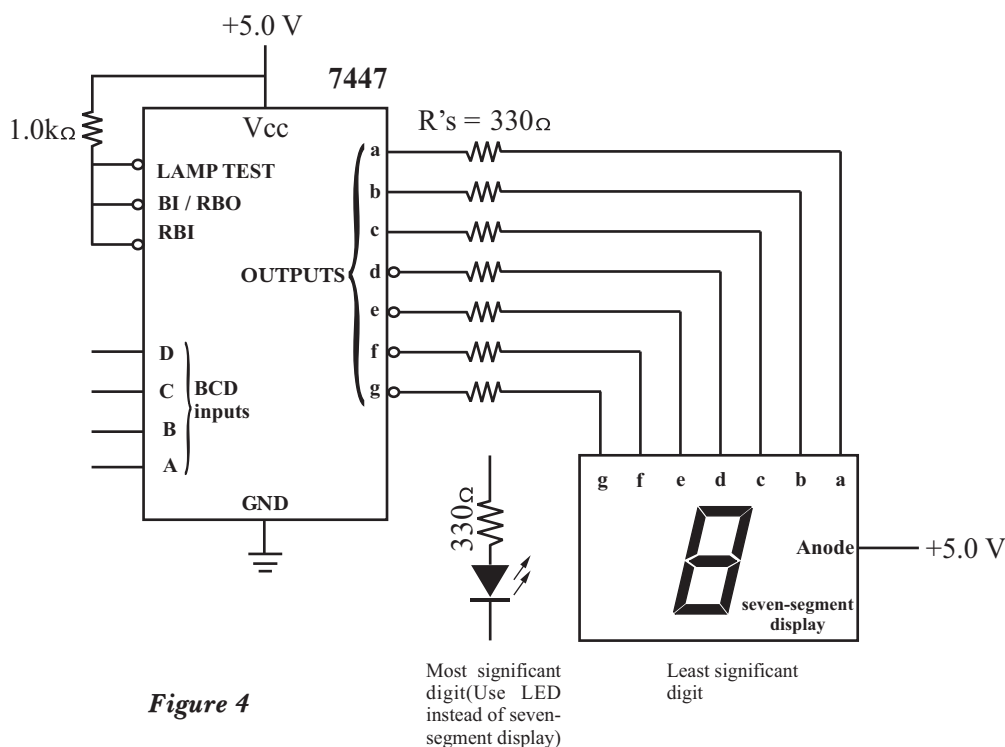
For Further Investigation

As you observed, the 7447A decoder used in this experiment is designed for BCD-to-decimal decoding; however, a slight modification of the circuit can be made to decode a binary number into octal. The largest number that we can show with a 4-bit binary input is octal 17, which requires two seven-segment displays to show both digits.

Recall that the conversion of a binary number to octal can be accomplished by grouping the binary number by threes, starting at the binary point. To display the octal numbers larger than binary 111 would normally require a second decoder and seven-segment display. For this problem, the most significant digit is either a zero or a one; therefore, we can dispense with the extra decoder and we could even use an ordinary LED to represent the most significant digit.

The seven-segment display you have will still show the least significant digit. Modify the circuit Figure 3 so that it correctly shows the octal numbers from 0 to 17. For example, if the switches are set to binary 1011 your circuit should light the LED representing the most significant digit and the seven-segment display should show a three.

A partial schematic is shown in figure 4. Complete the schematic, showing how to connect the circuit to show octal numbers. Construct the circuit, test it, and summarize how it works.



Report Of Experiment 2

Data and Observations:

TABLE 1

Inputs		Outputs
Binary Number	BCD Number	Seven-Segment Display
0000		8
0001		8
0010		8
0011		8
0100		8
0101		8
0110		8
0111		8
1000		8
1001		8
1010	INVALID	8
1011	INVALID	8
1100	INVALID	8
1101	INVALID	8
1110	INVALID	8
1111	INVALID	8

TABLE 2

Trouble Number	Trouble	Observations
1	LED for the c input is open.	
2	A input to 7447 is open.	
3	LAMP TEST is shorted to ground.	
4	Resistor connected to pin 15 of the 7447 is open	

Step 6. Method for causing leading zero Suppression:_____

Results and Conclusion:

Experiment 3

Logic Gates

Objectives

Determine experimentally the truth tables for the NAND, NOR, and inverter gates.

Use NAND and NOR gates to formulate other basic logic gates.

Components

7400 quad 2-input NAND gate

7402 quad 2-input NOR gate

1.0k Ω resistor

7486 quad XOR gate

Summary

Logic deals with only two normal conditions: logic “1” or logic “0”. These conditions are like the yes or no answers to a question. Either a switch is closed (1) or it isn’t (0); either an event has occurred (1) or it hasn’t (0); and so on. In Boolean logic, 1 and 0 represent conditions. In positive logic, 1 is represented by the term HIGH and 0 is represented by the term LOW. In positive logic, the more positive voltage is 1 and the less positive voltage is 0. Thus, for positive TTL logic, a voltage of +2.4 V = 1 and a voltage of +0.4 V = 0.

In some systems, this definition is reversed. With negative logic, the more positive voltage is 0 and the less positive voltage is 1. Thus, for negative TTL logic, a voltage of +0.4 V = 1 and a voltage of +2.4 V = 0

In addition to the AND, OR, and INVERT functions, two other gates are very important to logic designers. These are the NAND and NOR gates, in which the output of AND OR, respectively, have been negated. These gates are

important because of their “universal” property; they can be used to synthesize the other Boolean logic functions including AND, OR, and INVERT functions.

Other commonly used gates are the exclusive-OR (abbreviated XOR) and the exclusive-NOR (abbreviated XNOR) gates. These gates always have two inputs. The output of the XOR gate is HIGH when either A or B is HIGH, but not both (inputs “disagree”). The XNOR is just the opposite; the output is HIGH only when the input are the same (agree). For this reason, the XNOR gate is sometimes called COINCIDENCE gate.”

In this experiment, you will test the truth tables for NAND and NOR gates as well as those for several combinations of these gates. Keep in mind that if any two truth tables are identical, then the logic circuits that they represent are equivalent.

Pin configuration for ICs 7400 and 7402 are shown in figure 3.

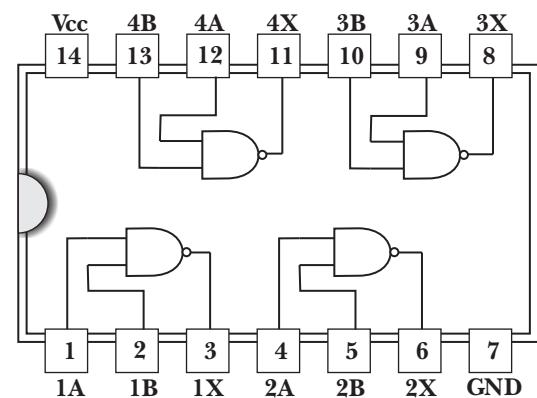


Fig 3(a): 7400 Quad 2-input NAND gate

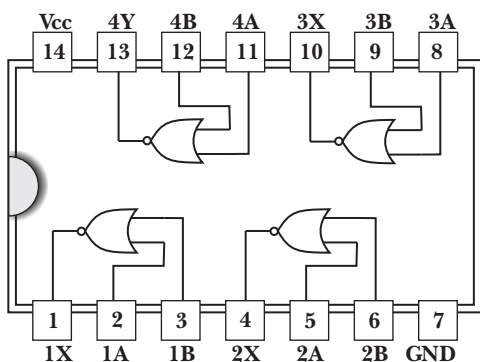


Fig 3(b): 7402 Quad 2-input NOR Gate

Procedure

Logic Functions

1. 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 combinations of inputs, as listed Table 2 of the report. Show the logic output (1 or 0) as well as the measured output voltage in table 2. Use the DMM to measure the output voltage.

2. Repeat Step 1 for one of the NOR gates; tabulate your results in table 3 of the report.

3. Connect the circuits of Figures 4 and 5. Connect the input to a 0 and a 1, measure each output voltage, and complete truth Table 4 and 5 for the circuits.

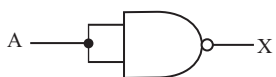


Fig . 4

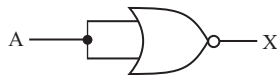


Fig . 5

4. Construct the circuit shown in Figure 6 and complete truth Table 6. This circuit may appear at first to have no application, but in fact can be used as a buffer. Because of amplification within the IC, a buffer provides more drive current.

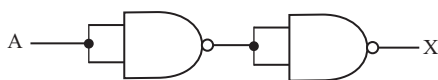


Fig . 6

5. Construct the circuit shown in figure 7 and complete truth Table 7. Notice that the truth table for this circuit is the same as the truth table for one of the single gates.

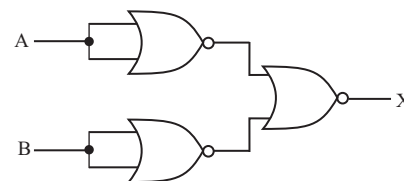


Fig . 7

6. Repeat Step 5 for the circuits shown in figure 8 and 9. Complete truth Tables 8 and 9.

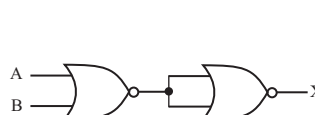


Fig . 8

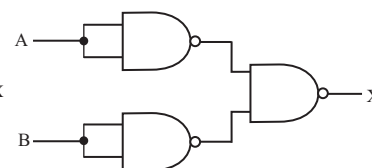


Fig . 9

For Further Investigation

The circuit shown in Figure 10 has the same truth table as one of the tables shown in Figure 4(a) through (e). Test all input combinations and complete truth Table 10. What is the equivalent gate?

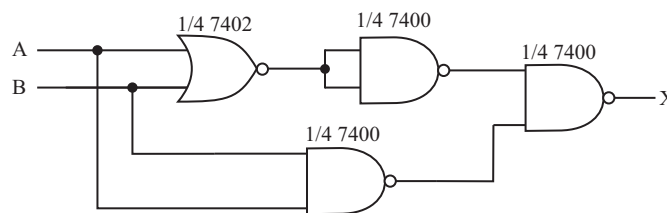
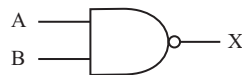


Fig . 10

Report Of Experiment 3

Data and Observations:

Table 2
NAND gate



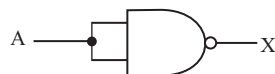
Inputs		Output	Measured Output Voltage
A	B	X	
0	0		
0	1		
1	0		
1	1		

Table 3
NOR gate



Inputs		Output	Measured Output Voltage
A	B	X	
0	0		
0	1		
1	0		
1	1		

Table 4
Truth table for Fig. 4



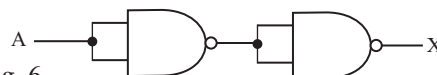
Input	Output	Measured Output Voltage
A	X	
0		
1		

Table 5
Truth table for Fig. 5



Input	Output	Measured Output Voltage
A	X	
0		
1		

Table 6
Truth table for Fig. 6



Input	Output	Measured Output Voltage
A	X	
0		
1		

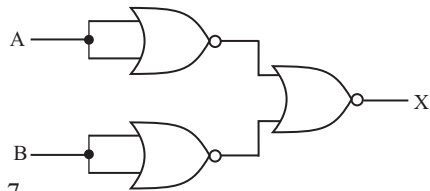


Table 7
Truth table for Fig. 7

Inputs		Output	Measured Output Voltage
A	B	X	
0	0		
0	1		
1	0		
1	1		

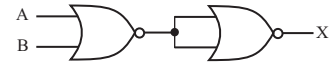


Table 8
Truth table for Fig. 8

Inputs		Output	Measured Output Voltage
A	B	X	
0	0		
0	1		
1	0		
1	1		

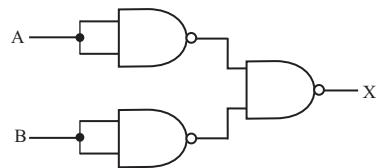


Table 9
Truth table for Fig. 9

Inputs		Output	Measured Output Voltage
A	B	X	
0	0		
0	1		
1	0		
1	1		

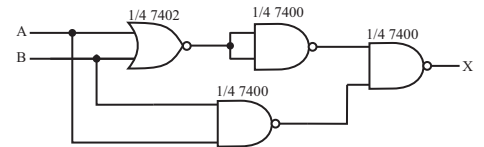


Table 10
Truth table for Fig. 10

Inputs		Output	Measured Output Voltage
A	B	X	
0	0		
0	1		
1	0		
1	1		

Results and Conclusion:

Experiment 4

Boolean Laws And DeMorgan's Theorem

Objectives

- ❑ Experimentally verify several of the rules for Boolean algebra.
- ❑ Design circuits to prove Rules 10 and 11.
- ❑ Experimentally determine the truth tables for circuits with three input variables, and use DeMorgan's theorem to prove algebraically whether they are equivalent.

Components

7432 quad 2-input OR gate
7404 hex inverter
7408 quad 2-input AND gate

Summary

Boolean algebra consists of a set of laws that govern logical relationships. Unlike ordinary algebra, where an unknown can take any value, the elements of Boolean algebra are binary variables and can have only one of two values: 1 or 0.

Symbols used in Boolean algebra include the over bar, which is the NOT or complement; the connective +, which implies logical addition and is read "OR"; and the connective ·, which implies logical multiplication and is read "AND." The dot is frequently eliminated when logical multiplication is shown. Thus $A \cdot B$ is written AB . The basic rules of Boolean algebra are listed in Table 1 for convenience.

The Boolean rules shown in Table 1 can be applied to actual circuits, as this experiment demonstrates. For example, Rule 1 states $A + 0 = A$

(remember to read + as "OR"). This rule can be demonstrated with an OR gate and a pulse generator, as shown in Figure 1. The signal from the pulse generator is labeled A and the ground represents the logic 0. The output, which is a replica of the pulse generator, represents the ORing of the two inputs—hence, rule is proved. Figure 1 illustrates this rule.

Table 1

Basic rules of Boolean algebra.

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $\bar{A} + A = 1$
7. $A \cdot A = A$
8. $\bar{A} \cdot A = 0$
9. $\bar{\bar{A}} = A$
10. $\bar{A} + AB = A$
11. $A + \bar{A}B = A + B$
12. $(A + B)(A + C) = A + BC$

NOTE: A, B, or C represent a single variable or a combination of variables

In addition to the basic rules of Boolean algebra, there are two additional rules called DeMorgan's theorems that allow simplification of logic expressions that an over bar covering more than one quantity. DeMorgan wrote two theorems for reducing these expressions. His first theorem is

The complement of two or more variables ANDed is equivalent to the OR of the complements of the individual variables.

Algebraically, this can be written as

$$\overline{X \cdot Y} = \overline{X} + \overline{Y}$$

His second theorem is

The complement of two or more variables Ored is equivalent to the AND of the complements of the individual variables.

Algebraically, this can be written as

$$\overline{X + Y} = \overline{X} \cdot \overline{Y}$$

As a memory aid for DeMorgan's theorems, some people like to use the rule "Break the bar and change the sign." The dot between ANDed quantities is implied if it is not shown, but it is given here to emphasize this idea.

Procedure

1. Construct the circuit shown in Figure 1. Use the +5.0V power supply. Set a frequency of 10 kHz from the pulse generator. Observe the signals from the pulse generator and the output at the same time on your oscilloscope. If you are using an analog scope, your need to be sure to trigger the scope on one channel only; otherwise a timing error can occur with some signals. The timing diagram an Boolean rule for this circuit has been completed in Table 2 as an example.

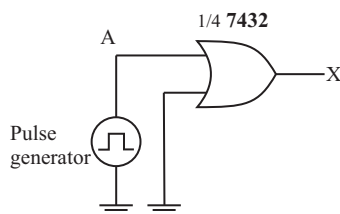


Fig 1

2. Change the circuit for step 1 to that of Figure 2. Complete the second line in Table 2.

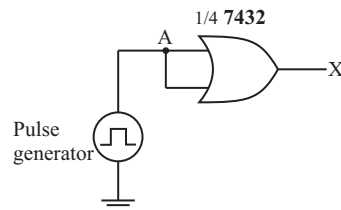


Fig 2

3. Connect the shown in Figure 3. Complete the third line of table 2.

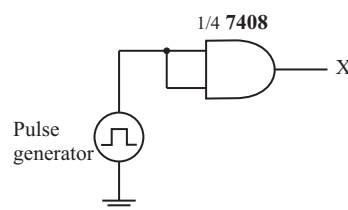


Fig 3

4. Change the circuit in Step 3 to that of Figure 4. Complete the forth line of table 2.

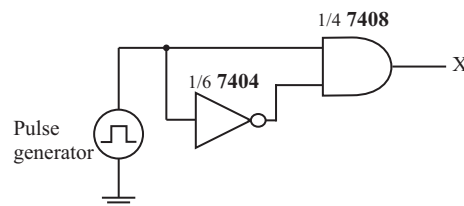


Fig 4

5. Design a circuit that will illustrate Rule 10. The pulse generator is used to represent the A input and a logic switch is used at the B input. Complete the schematic, build your circuit, and draw two timing diagrams in the space provided in Table 3. The first timing diagram is for the condition B = 0 and the second is for the condition B = 1.

6. Design a circuit that illustrates Rule 11. Draw your schematic in the space provided in Table 3. Construct the circuit and draw two timing diagrams for the circuit in Table 4.

For Further Investigation:

1. Build the circuit shown in Figure 5. Test each combination of input variables by closing the appropriate switches as listed in truth Table 5. Test each combination of inputs variables by closing the appropriate switches as listed in truth Table 5 in the report. Using the LED as a logic monitor, read the output logic, and complete Table 5.

2. Construct the circuit of Figure 6. Again, test each combination of inputs and complete truth Table 6 in the report. Observe the two truth tables. Can you prove (or disprove) that the circuits perform equivalent logic?

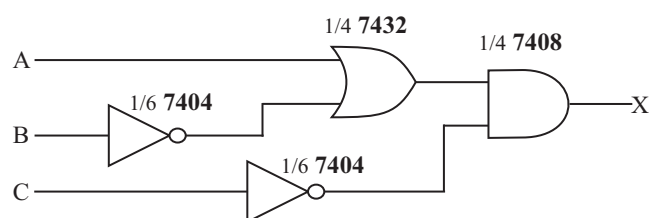


Fig 5

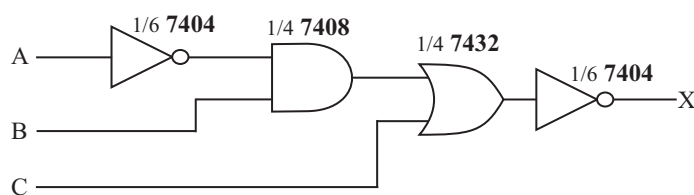


Fig 6

Table 5

Inputs			Outputs
A	B	C	X
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Table 6

Inputs			Outputs
A	B	C	X
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Report Of Experiment 4

Table 2

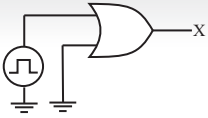
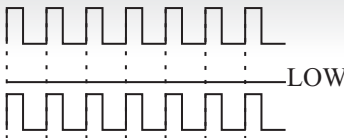
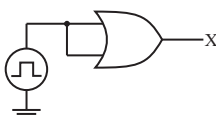

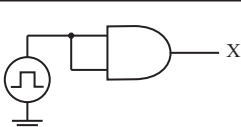
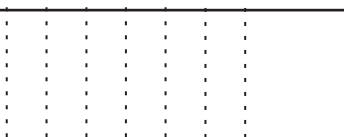
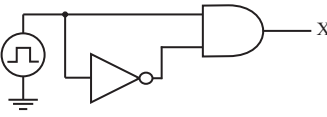
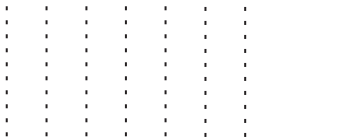
Schematic	Timing Diagram	Boolean Rule
	Inputs { 	$A + 0 = A$
	Inputs { 	
	Inputs { 	
	Inputs { 	

Table 3

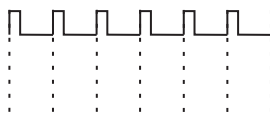

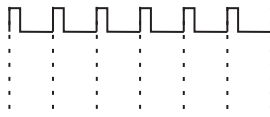
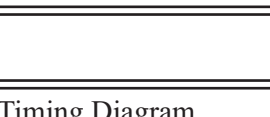
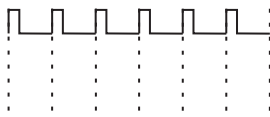

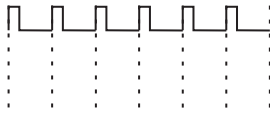
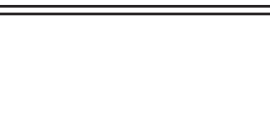
Schematic	Timing Diagram
Rule 10: To Signal generator A —○— Logic switch B —○—	Timing diagram for B=0: Inputs {  Output  Timing diagram for B=1: Inputs {  Output 

Table 4

Schematic	Timing Diagram
Rule 11: To Signal generator A —○— Logic switch B —○—	Timing diagram for B=0: Inputs {  Output  Timing diagram for B=1: Inputs {  Output 

Experiment 5

Logic Circuit Simplification

Objectives

- ❑ Develop the truth table for a BCD-invalid code detector.
- ❑ Use a Karnaugh map to simplify the expression.
- ❑ Build and test a circuit that implements the simplified expression.
- ❑ Predict the effect of “faults” in the circuit.

Components

As determined by the Student.

Summary

With combinational logic circuits, the outputs are determined solely by the inputs. For simple combinational circuits, truth tables are used to summarize all possible inputs and outputs; the truth table completely describes the desired operation of the circuit. The circuit may be realized by simplifying the expression for the output function as read from the truth table.

A powerful mapping technique for simplifying combinational logic circuits was developed by M. Karnaugh and was described in a paper he published in 1953. The method involved writing the truth table into a geometric map in which adjacent cells (squares) differ from each other in only one variable. (Adjacent cells share a common border horizontally or vertically.) When you are drawing a Karnaugh map, the variables are written in a Gray code sequence along the sides and tops of the map. Each cell on the map

corresponds to one row of the truth table. The output variables appear as 0's and 1's on the map in positions corresponding to those given in the truth table.

Table 1

Truth table for comparator.

Inputs				Outputs
A ₂	A ₁	B ₂	B ₁	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
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	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

As an example, consider the design of a 2-bit comparator. The inputs will be called A₂ A₁ and B₂ B₁.

The desired output is HIGH if A_2A_1 is equal to or greater than B_2B_1 . To begin, the desired output is written in the form of a truth table, as given in Table 1. All possible inputs are clearly identified by the truth table and the desired output for every possible input is given.

Next the Karnaugh map is drawn, as shown in Figure 1. In this example, the map is drawn using numbers to represent the inputs. The corresponding values for the output function are entered from the truth table. The map can be read in sum-of-products (SOP) form by grouping adjacent cells containing 1's on the map. The size of the groups must be an integer power of 2 (1, 2, 4, 8, etc.) and should contain only 1's on the map. The largest possible group should be taken; all 1's must be in at least one group and may be taken in more than one group if helpful.

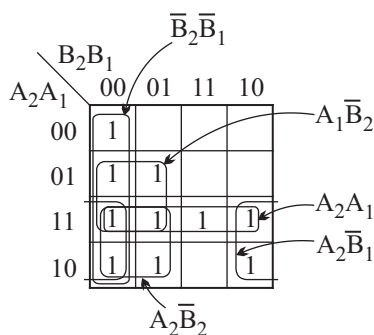


Fig 1:
Karnaugh map for the truth table shown in Table 1.

After grouping the 1's on the map, the output function can be determined. Each group is read as one of the product terms in the reduced output function. Within each group larger than one, adjacent boundaries will be crossed, causing the variable that changes to be eliminated from the output expression. A group of two adjacent 1's will have a single adjacent boundary and will eliminate one variable. A group of four 1's will eliminate two variables and a group of eight 1's will eliminate three variables. Figure 1 shows the grouping for the 1-bit comparator. Since each group in this case is a group of four 1's, each product term contains two variables (two were eliminated from each term). The resulting expression is the sum of all of the product terms. The circuit can be drawn directly, as shown in Figure 2.

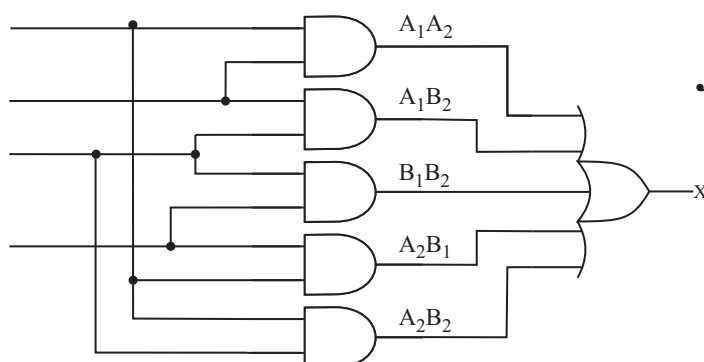


Fig 2:
Circuit implementation of the comparator given by truth table 1.

In this experiment you will use the Karnaugh mapping method, similar to the one described previously, to design a BCD-Invalid code detector. You will design a circuit to assure that only valid BCD codes are present in Register A and to signal a warning if an invalid BCD code is detected. Your circuit will be designed for 4-bits, but could easily be expanded to 8-bits for Register A.

Procedure

BCD Invalid Code Detector

1. Complete the truth table shown as Table 2 in the report. Assume the output for the ten valid BCD codes is a 0 and for the six invalid BCD codes is a 1. As usual for representing numbers, the most significant bit is represented by the letter D and the least significant bit by the letter A.
2. Complete the truth table shown as Table 4 in the report. Group the 1's according to the rules. Find the expression of the invalid codes by reading the minimum SOP from the map. Write the Boolean expression in the space provided in the report.
3. If you have correctly written the expression in Step 2, there are two product terms and you will see that the letter D appears in both terms. This expression could be implemented directly as a satisfactory logic circuit. By factoring D from each term, you will arrive at another expression for invalid codes. Write the new expression in the space provided in the report.

4. Although the circuit shown in Figure 3 satisfies the design requirements with only two gates, it requires two different ICs.

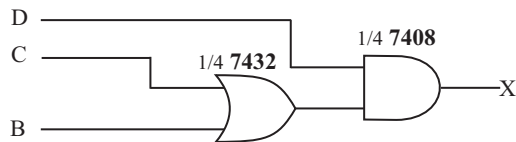


Fig 3:

5. Construct the circuit your drew in Step 5. Test all combinations of the inputs and complete truth Table 3 in the report. If you have constructed and tested the circuit correctly, the truth table will be the same as Table 2.

For Further Investigation

Design a circuit that will indicate if a BCD number is evenly divisible by threee (3, 6, or 9). The input is a valid BCD number have already been tested and rejected by your earlier circuit! Since invalid

numbers are now impossible, the Karnaugh map will contain “don’t care” entries. An “X” on the map means that if the input is not possible, then you don’t care about the outcome.

1. Complete the truth Table 5 in the report for the problem stated above. Enter 0’s for BCD numbers that are not divisible by three and 1’s for BCD numbers that are divisible by three. Enter an “X” to indicate a invalid BCD code.

2. Complete the Karnaugh map shown as Figure 6. Group 1’s on the map in groups of 2, 4, 8, etc. Do not take any 0’s but do take X’s if you can obtain a larger group. Read the minimum SOP from the map and show your expression in the space provided in the report.

3. Draw a circuit using only NAND gates that will implement the expression. The LED should be turned ON with a LOW output. Build the circuit and test each of the possible inputs to see that it performs as expected.

Report for Experiment 5

Data and Observations:

Table 2

Truth table for BCD-invalid code detector.

Inputs				Outputs
D	C	B	A	X
0	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	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	

DC \ BA				
	00	01	11	10
00				
01				
11				
10				

Fig 4:
Karnaugh map of truth table for invalid BCD code detector.

Minimum sum-of-products read from map:

X = _____

Factoring D from both product terms gives:

X = _____

Step 5. Circuit for BCD-invalid code detector:

Table 3

Truth table for BCD-invalid code detector constructed in step 6.

Inputs				Outputs
D	C	B	A	X
0	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	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	

DC \ BA				
	00	01	11	10
00				
01				
11				
10				

Fig 6:
Karnaugh map for BCD-invalid code detector

Minimum sum-of-products read from map:

X = _____

Circuit:

Table 5

Truth table for BCD numbers divisible by three.

Inputs				Outputs
D	C	B	A	X
0	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	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	

DC	BA			
	00	01	11	10
00				
01				
11				
10				

Fig 4:
Karnaugh map of truth table for invalid BCD numbers divisible by three.

Minimum sum-of-products read from map:

X = _____

Circuit:

Experiment 6

The Perfect Pencil Machine

Objectives

- ❑ Design and build the combinational portion of a logic circuit that delivers a “pencil” and “change” based on the value of “coins” that are entered.
- ❑ Write a formal laboratory report describing your circuit and results.

Components

Materials as determined by student.

Summary

The methods of designing a combinational logic circuit using the truth table, the equation of the function, and Karnaugh maps. These methods are important tools for the design of simple combinational circuits.

They require the ability to respond to a series of events. For example, a coin operated machine must “remember” the amount of money that has been inserted previously and compare it with the price of the product. When the amount of money is equal or greater then the price of the product and any necessary change. This is an example of a sequential machine; the key difference between it and a combinational machine is memory. Any circuit that contains memory is considered to be a sequential circuit.

For analysis purpose, sequential circuits can be broken into memory units and combinational units. A coin-operated machine actually contains both sequential (memory) and combinational portions. For simplification, we will focus only on the combinational logic portion of a coin-

operated machine that has multiple outputs. The outputs consist of a product and various coins for change.

To design the combinational logic portion of the machine, you will need a separate Karnaugh map for each output. Keep in mind that logic needed for one of the outputs sometimes appears in the equation for the other output, and this logic may simplify the total circuit.

As an example, the Model-1 perfect pencil machine design is shown in figure 1. This machine (which was shortly after the invention of the pencil) uses combinational logic to determine if the mechanical hand holding the pencil should open. Back then, pencil sold for ten cents and the original pencil machine could accept either two nickels(5 cent coins) or one dime(10 cent coins) but nothing else. If someone first dropped in a nickel and then a dime, the machine was smart enough to give a nickel change.

The Model-1 perfect pencil machine was designed by first filling out a truth table showing all possibilities for the input coins. There were two switches for nickels, labeled N_1 and N_2 , and one switch for a dime, labeled D . The truth table is shown as Table 1. The truth table lists two outputs; a pencil, labeled P , and a nickel change, labeled N_C . A 1 on the truth table indicates that the coin has been placed in the machine or a product (pencil or change) is to be delivered; a 0 indicates the absence of a coin or that no product is to be delivered; a 0 indicates the absence of a coin or that no product is to be delivered.

In the Model-1 machine, the two nickel switches are stacked on top of each other (see figure 1), forming a mechanical version of sequential logic.

It is not possible for the second nickel to be placed in the machine until first nickel has been placed in the machine sequentially; as soon as ten cents is added, the pencil is delivered. It is not possible for three coins to be placed in the machine as any combination of two coins will deliver a pencil. For these reasons, several lines on the truth table are shown as “don’t care” (X). “Don’t cares” appear on the table because if the input is not possible, then we don’t care what the output does.

Table 1

Truth table for the Model-1 perfect pencil machine

Inputs			Outputs	
N_1	N_2	D	P	N_C
0	0	0	0	0
0	0	1	1	0
0	1	0	X	X
0	1	1	X	X
1	0	0	0	0
1	0	1	1	1
1	1	0	1	0
1	1	1	X	X

The information from the truth table was entered onto two Karnaugh maps __one for each output see figure 2. The maps were read by taking “don’t care wherever it would help. This, of course, greatly simplifies the resulting logic. The Boolean expressions that resulted surprised

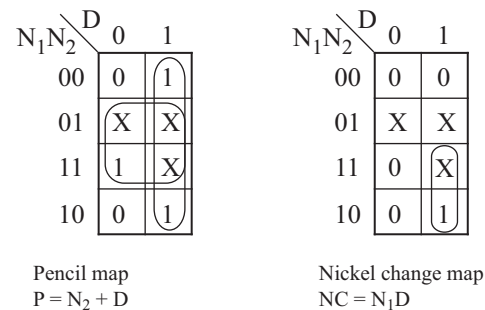


Fig 2 : Karnaugh maps for Model-1 perfect pencil machine

no one__not even the company president! From the Boolean sum-of-products (SOP), it was a simple matter to implement the Model-1 perfect pencil machine, as shown in Figure 3.

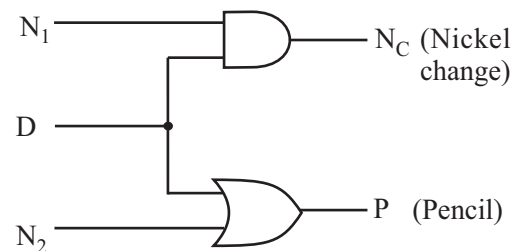
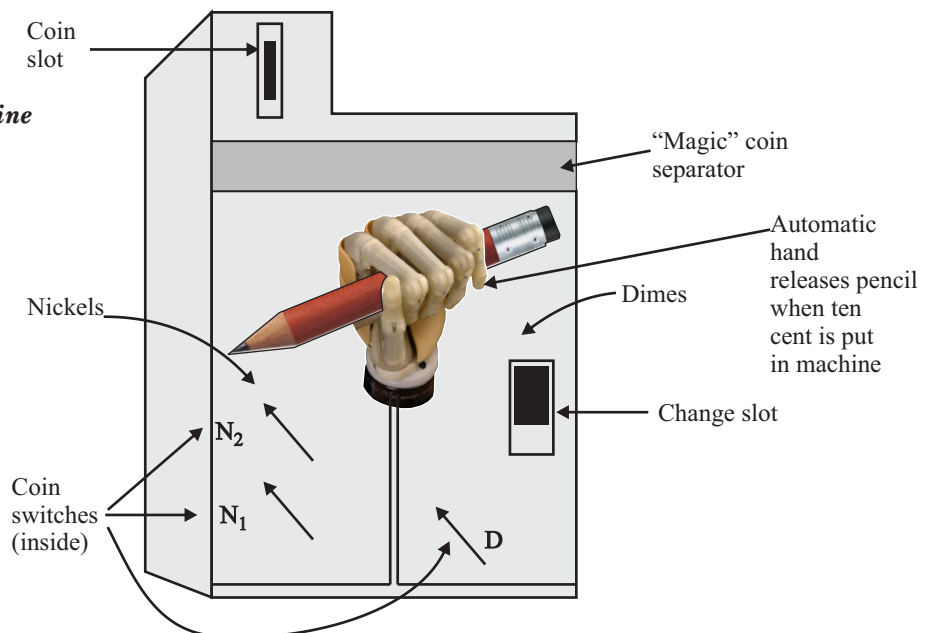


Fig 3 : Model-1 perfect pencil machine

Fig 1 : Model-1 perfect pencil machine



Procedure

The Model-2 Perfect Pencil Machine

1. The logic for the new Model-2 perfect pencil machine must be designed. Your job is to design the combinational logic portion of the new machine and test it (using LEDs for the outputs) to assure that it works. You must then write a report for the director of engineering summarizing your design and test results. The problem statement is given as follows.

Problem statement:

The new Model-2 perfect pencil machine sells pencils for fifteen cents (due to inflation). The machine will be designed to accept one nickel, two dimes (10 cents), a quarter (25 cents), or certain combinations of these. (Anyone who put three nickels in the machine deserved to lose them!). The coins for the combinational logic are represented by four input switches—a nickel (N_1), two dimes (D_1 and D_2), and a quarter (Q). The first dime in the machine will always turn on the D_1 switch and second dime in the machine will always turn on the D_2 switch. As in the Model-1 design, there are certain combinations that are impossible. This is due to the fact that the automatic hand (holding the pencil) begins to open as soon as 15 cents in the machine. This implies that no more than 2 coins can be put in the machine. Also, it is not possible to activate the second dime switch before the first dime switch. To clarify the combinations, the director of engineering has started filling out the truth table (see Table 2 in the Report section) but was called to a meeting.

There are four outputs from the Model-2 perfect pencil machine, as listed on the truth table. They are pencil (P), nickel change (N_C), the first dime change (DC_1), and the second dime change (DC_2). Your will need to determine what these outputs should be to deliver the correct product and change.

Oh! and the director of engineering says she would like to have you use no more than two ICs! Good luck!

For Further Investigation:

After the Model-2 perfect pencil machine was designed, the purchasing department found out that 2-input NAND gates were on sale and they purchased several million of them. The problem is that you will need to modify your circuit to use the 2-input NAND gates throughout. Your boss muttered, “It may mean more IC’s, but we’ve got to use up those NAND gates!” Show in the report how you will help the company in your design.

Table 2

Truth table for the Model-2 perfect pencil machine

Inputs				Outputs			
N_1	D_1	D_2	Q	P	N_C	DC_1	DC_2
0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0
0	0	1	0	X	X	X	X
0	0	1	1	X	X	X	X
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	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				

Report for Experiment 6

Data and Observations:

Table 3

Truth table for the Model-2 perfect pencil machine shown with impossible inputs

Inputs				Outputs			
N ₁	D ₁	D ₂	Q	P	NC	DC ₁	DC ₂
0	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	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	---	---	---	---

Note: Impossible and invalid inputs are crossed in the table

Results and Conclusion:

Experiment 7

The Fillmore Furniture Factory

Objectives

Design the combinational logic for control of a process in a furniture factory.

Submit a written laboratory report describing your circuit and results.

Components

Materials as determined by the student.

Summary

Control systems typically use sophisticated programmable controllers to provide flexibility in implementing changes. For very simple systems, the system could be designed in a permanent manner. The problem posed in this experiment is a simple system that will provide you with an opportunity to test your skill at designing, testing, and reporting on a combinational logic problem with a limited number of inputs and outputs.

Procedure

The Fillmore Furniture Factory wishes to automate its lawn chair production line with new control logic. Your job is to complete the design of the system, build and test it, and submit a written report. You will need to design the logic for controlling each of four motors (lube, conveyor, band saw, and cross-cut saw). The motors are to be simulated with LEDs turned on whenever the motor is on. The motors are controlled by four switches (S_1 , S_2 , S_3 and S_4) in

accordance with rules given in the section “Operational Requirements.” The lubrication motor design has been completed for you as an example.

Basic System Operation

The system uses four manual ON/OFF switches, control logic, and a motor drive interface to control the conveyor lubrication pump, the conveyor motor, the cross-cut saw motor, and the band saw motor, as indicated by the block diagram in Figure 1. The focus of this system application is the control logic portion of the system.

Operational Requirements

Switch S_1 controls the lubrication pump motor, M_1 . Switch S_2 controls the conveyor motor, M_2 . Switch S_3 controls the band saw motor, M_3 . Switch S_4 controls the cross-cut saw motor, M_4 .

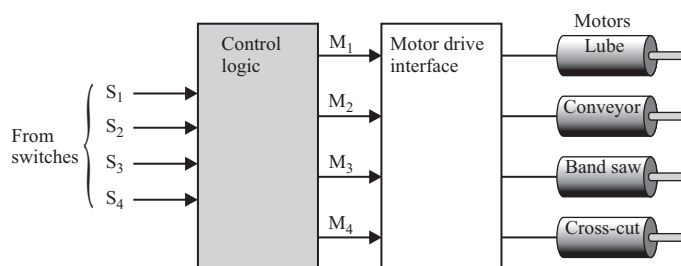


Fig 1 : Basic block diagram for the control system

The motor that provides for lubrication of the conveyor (M_1) must be ON when the conveyor is ON. The motor that drives the conveyor (M_2) is ON only when both switches S_1 and S_2 are ON. The band saw motor is ON when switch S_1 is ON, and the cross-cut saw motor M_4 is ON when switch S_4 is ON.

The band saw and cross-cut saw motors require no lubrication, but they must never be ON at the same time. If switches S_3 and S_4 are both turned ON at the same time, the system must be completely shut down, including the conveyor and lubrication motors. In addition, the cross-cut saw and the conveyor cannot be ON at the same time. The purpose of the control logic is to control the motors to prevent any of the unallowed conditions from occurring should the switches be improperly operated.

Truth Table for the Control Logic

Table 1 is a truth table that shows the states of the switches and motors based on the conditions just described. A logic 1 indicates the ON state and a logic 0 indicates the OFF state. Since there are four switches, there are sixteen possible ON/OFF combinations of the switches and, as a result, sixteen possible ON/OFF combinations for the motors. The states of the switches are the input variables and the states of the motors are the output variable. The unallowed conditions in this case are not don't cares (X). Instead, they are conditions that should not occur but if they do, the system must be shut down.

Table 1

Truth table for the control logic.

Inputs				Outputs				Comments
N_1	D_1	D_2	Q	M_1 Lube	M_2 Conv	M_3 Band	M_4 Cross	
0	0	0	0	0	0	0	0	shut down
0	0	0	1	0	0	0	1	OK
0	0	1	0	0	0	1	0	OK
0	0	1	1	0	0	0	0	Unallowed (2 saws on) __ shut down
0	1	0	0	0	0	0	0	Unallowed (no lube with conveyor) __ shut down
0	1	0	1	0	0	0	0	Unallowed (conveyor and cross cut) __ shut down
0	1	1	0	0	0	0	0	Unallowed (no lube with conveyor) __ shut down
0	1	1	1	0	0	0	0	Unallowed (2 saws and conveyor) __ shut down
1	0	0	0	1	0	0	0	OK
1	0	0	1	1	0	0	1	OK
1	0	1	0	1	0	1	0	OK
1	0	1	1	0	0	0	0	Unallowed (2 saws) __ shut down
1	1	0	0	1	1	0	0	OK
1	1	0	1	0	0	0	0	Unallowed (conveyor and cross cut) __ shut down
1	1	1	0	1	1	1	0	OK
1	1	1	1	0	0	0	0	Unallowed (all on) __ shut down

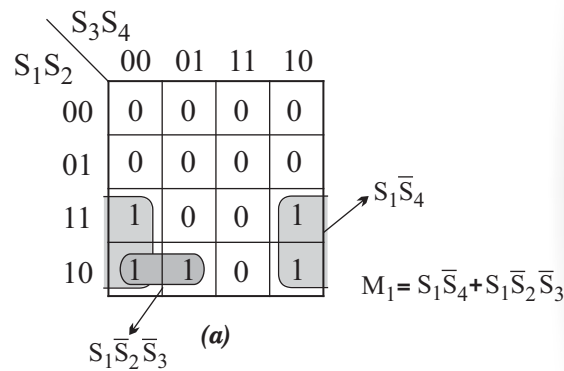
Design of the Control Logic

There will be four separate logic circuits, one for each of the motors. The logic for the lubrication pump motor (M_1) is shown as an example. The first step is to transfer the data from the truth table to a Karnaugh map and develop an SOP expression.

The switch variables S_1 , S_2 , S_3 , and S_4 are map variables, and the states of M_1 are plotted and grouped as shown in Figure 2(a). The result-

ing sum-of-products expression for the lubrication pump motor logic results in the NAND implementation shown in part (b). Complete the design, showing your schematic in your report. Blank Karnaugh maps have been inserted to help you get started. After completing the design, construct your circuit, test it, and summarize your results.

Fig 2: Karnaugh map simplification and implementation for the lubrication motor logic

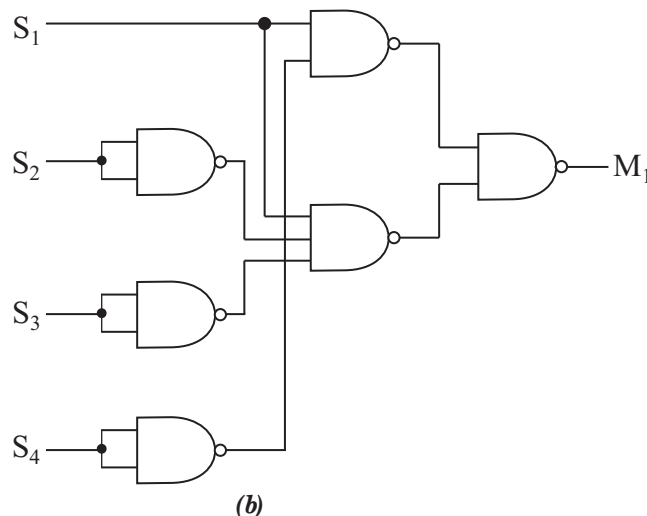


M_1 Can be implemented using NAND gate as:

$$M_1 = S_1\bar{S}_4 + S_1\bar{S}_2\bar{S}_3$$

$$= \overline{\overline{S_1\bar{S}_4} \cdot \overline{S_1\bar{S}_2\bar{S}_3}}$$

$$M_1 = \overline{\overline{S_1\bar{S}_4} \cdot \overline{S_1\bar{S}_2\bar{S}_3}}$$



Report for Experiment 7

Data and Observations:

$S_1S_2 \backslash S_3S_4$	00	01	11	10
00				
01				
11				
10				

$S_1S_2 \backslash S_3S_4$	00	01	11	10
00				
01				
11				
10				

$S_1S_2 \backslash S_3S_4$	00	01	11	10
00				
01				
11				
10				

Results and Conclusion:

Adder and Magnitude Comparator

Objectives

- ❑ Complete the design, build, and test a 4-bit binary to Excess-3 code converter.
- ❑ Complete the design of a signed number adder with overflow detection.

Components

7483A 4-bit binary adder
7485 4-bit magnitude comparator
BCD to 7-segment decoder with Display.
As determined by the student.

Summary

This experiment introduces you to two important MSI circuits — the 4-bit adder and the 4-bit magnitude comparator. The TTL version of a 4-bit adder, with full look-ahead carry, is the 7483A. The TTL version of the 4-bit magnitude comparator is the 7485 which includes $A > B$, $A < B$, and $A = B$ outputs. It also has $A > B$, $A < B$, and $A = B$ inputs for the purpose of cascading comparators. Be careful to note inputs and outputs when you are connecting a comparator.

One difference in the way adder and comparators are labeled needs to be clarified. A 4-bit adder contains a carry-in, which is considered the least significant bit so it is given a zero subscript (C_0). The next least significant bits are contained in the two 4-bit numbers to be added, so they are identified with a one subscript (A_1, B_1). On a comparator, there is no carry-in, so the least significant bits are labeled with a zero subscript (A_0, B_0).

In this experiment, an adder and comparator are used to convert a 4-bit binary code to Excess-3 code. To familiarize you with the experiment, a similar circuit is described in the following example.

Example: A 4-bit binary to BCD Converter

Recall that a 4-bit binary number between 0000 and 1001 (decimal 9) is the same as the BCD number. If we add zero to these binary numbers, the result is unchanged and still represents BCD. Binary numbers from 1010 (ten) to 1111 (fifteen) can be converted to BCD by simply adding 0110 (six) to them.

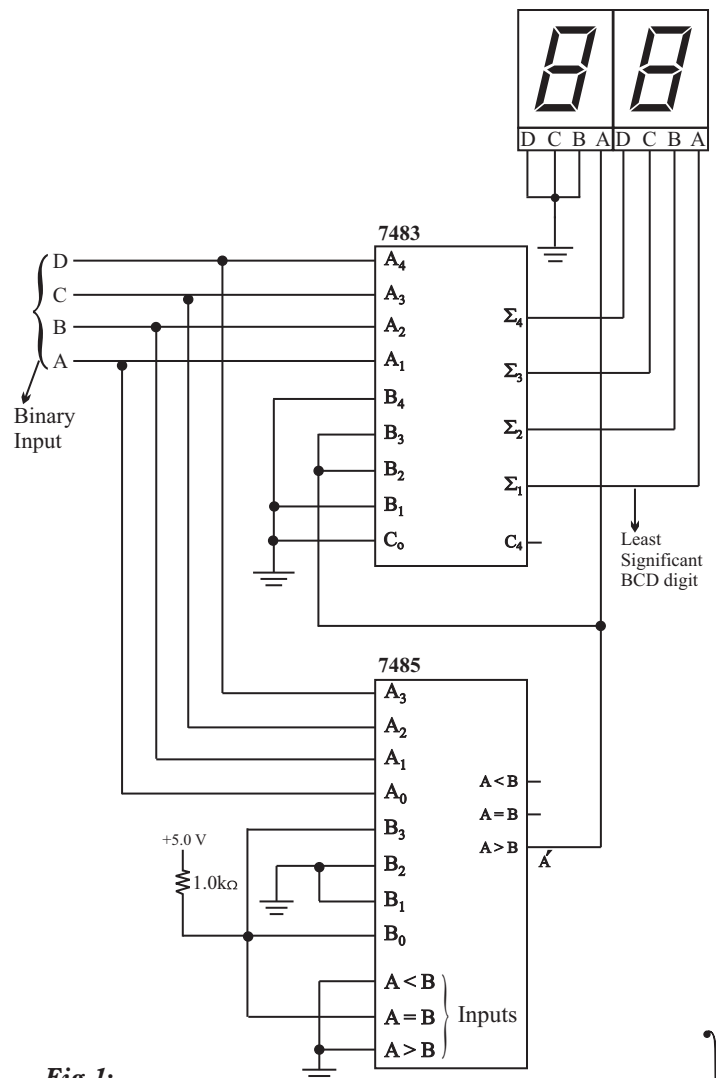


Fig 1:  **Binary to BCD converter.**

Table 1
Binary to BCD.

Comparator A > B Output	Adder Input				Comment
	B ₄	B ₃	B ₂	B ₁	
0	0	0	0	0	input is less than ten, add 0000
1	0	1	1	0	input is greater than nine, add 0110

The circuit that accomplishes this is illustrated in Figure 1. Notice that the B input of the comparator is connected to 1001. If the binary number on the A input of the comparator is greater than 1001, the A > B output is asserted. This action causes the adder to add 0110 to the binary input. Notice how the A > B output of the comparator is connected to the B side of the adder. Only bits B₂ and B₃ will be high when A > B is asserted; otherwise, all bits are low. This causes the adder to either add 0000 or 0110 to the binary input. The situation is summarized in Table 1. Notice in the table that bits B₂ and B₃ have the same “sense” as the A > B is 1. Bits B₄ and B₁ are always 0, hence these inputs are connected to ground.

Procedure

Figure 2 (in report) shows a partially completed schematic of a binary to Excess-3 code conversion circuit. The adder must add 0011 (decimal 3) to the binary input number if it is between 0000 and 1001 but add 1001 (nine) to the number if it is greater than nine in order to convert the 4-bit binary number to Excess-3. The problem is summarized in the table 2. Decide how to connect the open inputs on the 7483A, and complete the schematic.

From your schematic, build the circuit. Test all possible inputs as listed on truth Table 4 (in report). The outputs can be read directly from the display.

Table 2
Binary to Excess-3.

Comparator A > B Output	Adder Input				Comment
	B ₄	B ₃	B ₂	B ₁	
0	0	0	1	1	input is less than ten, add 0011
1	1	0	0	1	input is greater than nine, add 1001

For Further Investigation

Overflow Detection

Fixed-point signed numbers are stored in most computers in the manner illustrated in Table 3. Positive numbers are stored in true form and negative numbers are stored in 2’s complement form. If two numbers with the same sign are added, the answer can be too large to be represented with the number of bits available. This condition, called *overflow*, occurs when an addition operation causes a carry into the sign bit position. As a result, the sign bit will be in error, a condition easy to detect. When two numbers with the opposite sign are added, overflow cannot occur, so the sign bet will always be correct. Figure 3 illustrates overflow for 4-bit numbers.

In this part of the experiment we will step through the design of a 4-bit adder for signed numbers that detects the presence of an overflow error and lights an LED when overflow occurs. We can start with the 7483 adder and a 7404 hex inverter as shown in Figure 4 (in the report).

Consider the problem of detecting an overflow error. We need to consider only the sign bit for each number to be added and the sign bit for the answer. Complete truth Table 5 for all possible combinations of the sign bit, showing a 1 whenever an overflow error occurs.

Complete the Karnaugh map of the output (shown in the report as Figure 5) to see whether minimization is possible.

Write the Boolean expression for detection of an overflow error in your report.

Note that the signals going into the box in Figure 4 are A_4 , B_4 and Σ_4 . If you apply DeMorgan's theorem to one term of your Boolean expression, you can draw a circuit that uses only these inputs. Draw the circuit in the box. If directed by your instructor, build and test your circuit.

Numbers with opposite signs:

Overflow into sign position cannot occur.

Carry out indicates answer is positive. $\begin{array}{r} 0111 = +7 \\ 1110 = -2 \\ \hline 10101 = +5 \end{array}$

No carry out indicates answer is negative and in 2's complement form. $\begin{array}{r} 1001 = -7 \\ 0010 = +2 \\ \hline 10101 = -5 \end{array}$

Number with the same sign:

Overflow into sign position can occur.

Sign bit has changed. $\begin{array}{r} 1001 = -7 \\ 1110 = -2 \\ \hline 10111 = \text{Error!} \end{array}$

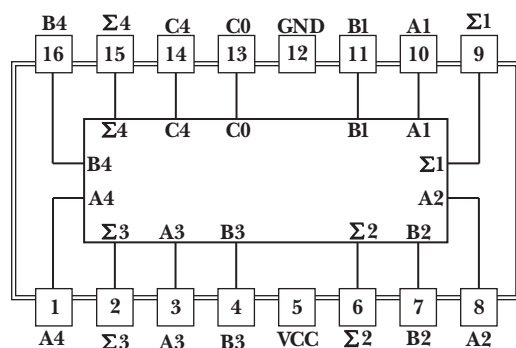
Sign bit has changed. $\begin{array}{r} 0111 = +7 \\ 0010 = +2 \\ \hline 01001 = \text{Error!} \end{array}$

Table 3

Representation of 4-bit signed numbers.

Base Ten Number	Computer Representation
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

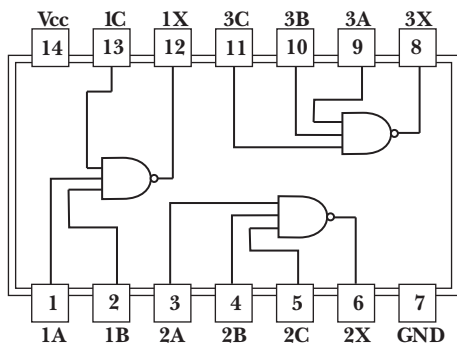
↑
Sign bit



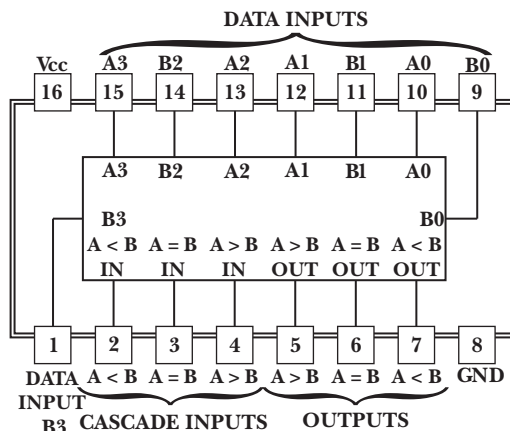
7483 4-bit binary adder

Fig 3:

Overflow that can occur with 4-bit signed numbers. The concept shown here is applied to large binary numbers.



7410 triple 3-input NAND Gate



7485 4-Bit Magnitude comparator.

The circuit diagram illustrates an Excess-3 converter. It consists of two 7483 4-bit adders and a 7485 4-bit magnitude comparator.

- 7483 Adder 1:** Its inputs A_4, A_3, A_2, A_1 are connected to the inputs D, C, B, and A, respectively. Its inputs B_4, B_3, B_2, B_1 are connected to the outputs $\Sigma_4, \Sigma_3, \Sigma_2, \Sigma_1$ of the second 7483 adder. Its carry input C_0 is connected to the output A' of the 7485 comparator. Its output C_4 is connected to the input A_0 of the 7485 comparator.
- 7483 Adder 2:** Its inputs A_3, A_2, A_1, A_0 are connected to the inputs D, C, B, and A, respectively. Its inputs B_3, B_2, B_1, B_0 are connected to the inputs D, C, B, and A, respectively. Its carry input C_0 is connected to the output A' of the 7485 comparator. Its output C_4 is connected to the input A_0 of the 7485 comparator.
- 7485 Comparator:** It has three outputs: $A < B$, $A = B$, and $A > B$. The $A < B$ output is connected to the input A_4 of the first 7483 adder. The $A = B$ output is connected to the input A_3 of the first 7483 adder. The $A > B$ output is connected to the input A_2 of the first 7483 adder.
- Power Supply:** A +5.0 V supply is connected to the inputs A_3, A_2, A_1, A_0 of the 7485 comparator through a 1.0 k Ω resistor. The inputs B_3, B_2, B_1, B_0 of the 7485 comparator are connected to ground.

The final output of the Excess-3 converter is the 4-bit BCD digit $A'B'C'D$, where A' is the output of the 7485 comparator and $B'C'D$ are the outputs $\Sigma_4, \Sigma_3, \Sigma_2$ of the second 7483 adder.

Table 4

Truth table for Figure 2.

Inputs (Binary)				Outputs Excess-3			
D	C	B	A	\bar{A}	D	C	B A
0	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	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				

Table 5

Truth table for overflow error.

Sign Bits			Error
A_4	B_4	Σ_4	X
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

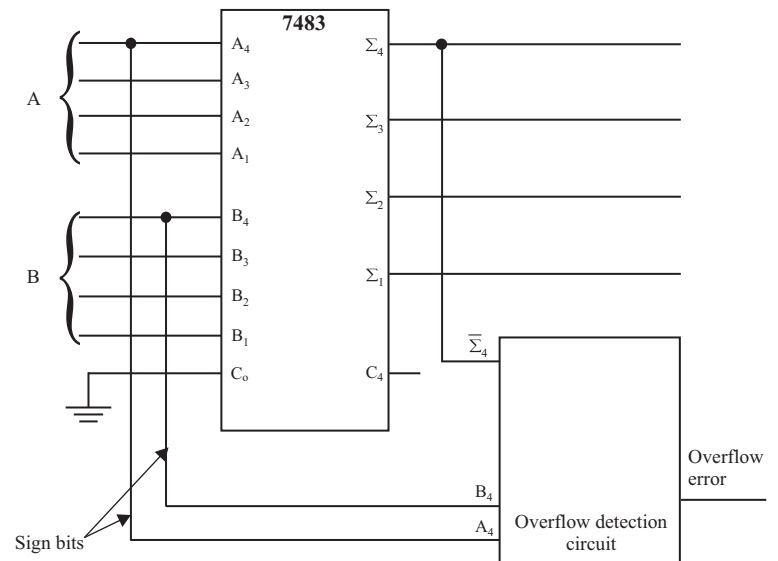


Fig 4: Signed number adder.

$A_4 \backslash B_4$	Σ_4	0	1
0	0		
0	1		
1	1		
1	0		

Fig 5: Karnaugh map for overflow error.

Boolean expression for overflow error:

X = _____

Experiment 9

Combinational Logic Using Multiplexers

Objectives

- ❑ Use a multiplexer to construct a comparator and a parity generator and test the circuit.
- ❑ Use an N -input multiplexer to implement a truth table containing $2N$ inputs.

Components

74151A data selector/multiplexer.

7404 hex inverter.

Other determined by student.

Summary

The *multiplexer* or *data selector* connects any one of several inputs to a single output. The opposite function, in which a single input is directed to one of several outputs, is called a *demultiplexer* or a *decoder*. These definitions are illustrated in Figure 1. Control is determined by additional logic signals called the select (*or address*) inputs.

Multiplexers (MUXs) and demultiplexers (DMUXs) have many applications in digital logic. One useful application for MUXs is implementation of combinational logic functions directly from the truth table. An overflow error detection circuit with the truth table shown in Figure 2(a). Each output row containing a “1” represents a minterm. (*Minterm* is a Boolean term in which all input variables appear either in “NOTed” or “true” form. For example, if there are 4-inputs labeled A, B, C, and D, then $\overline{A}BC\overline{D}$ and $\overline{A}BCD$ are both minterms but $\overline{A}BC$ is not.) If the logic for that minterm is connected to the data inputs of a MUX, and if the data selected are controlled by the input variables, the truth table has been

implemented directly. Figure 2 illustrates this idea conceptually for the overflow detector.

Actually, an 8-input MUX is not required to implement the overflow detection logic. Any N -input MUX can generate the output function for $2N$ inputs. To illustrate, we reorganize the truth table in pairs, as shown in Figure 3(a). The inputs labeled A_4 and B_4 are used to select a data line. Connected to that line can be a logic 0, 1, Σ_4 , or $\overline{\Sigma}_4$. For example, from the truth table, if $A_4 = 0$ and $B_4 = 1$, the D_1 input is selected. Since both outputs are the same (in this case a logic 0), then D_1 is connected to a logic 0. If the outputs were different, such as in the first and fourth rows, then the third input variable, Σ_4 , would be compared with the output. Either the true (or NOT) form of that variable then would be selected. The results are shown conceptually in Figure 3(b), which is equivalent to but simpler than the circuit in Figure 2(b).

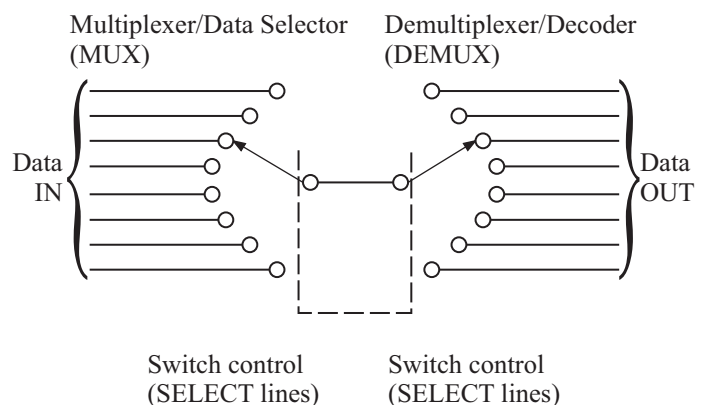
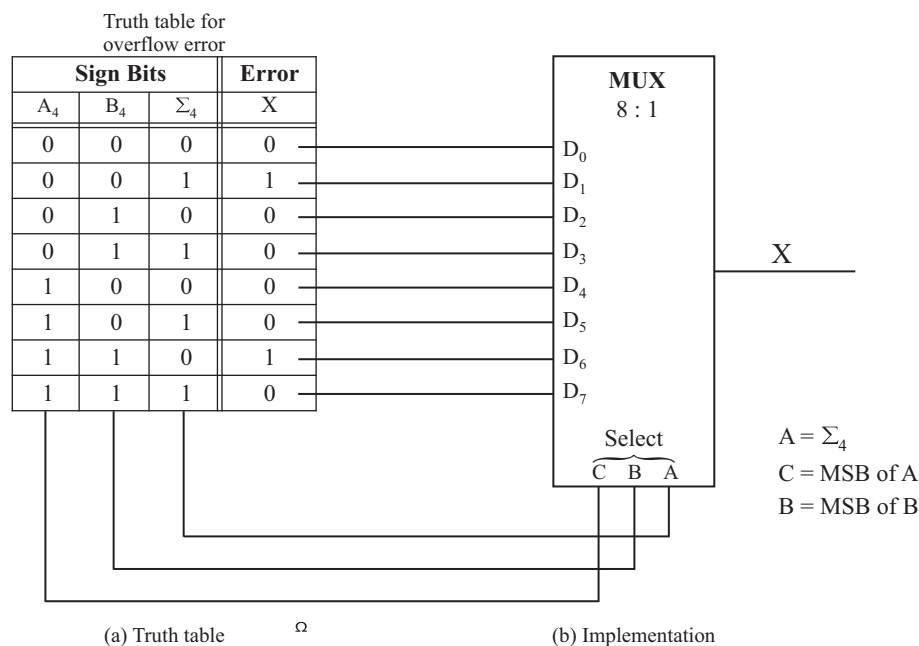


Fig 1

Fig 2:



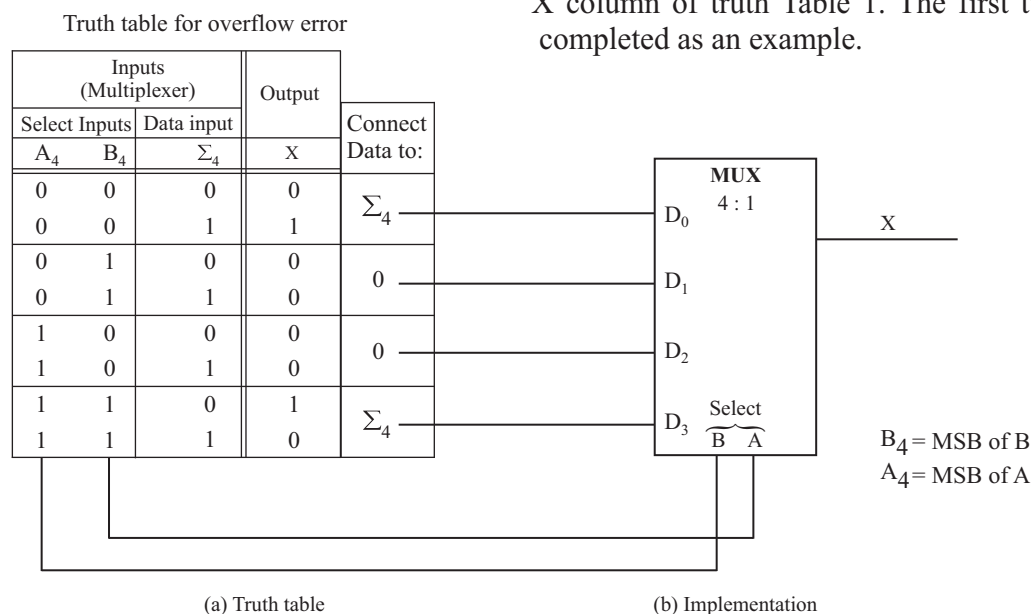
In this experiment you will use an 8 : 1 MUX to implement a 4-input truth table (with 16 combinations). First you will develop the circuit to implement a special comparator. In the Further Investigation section the circuit is modified to make a parity generator for a 4-bit code. *Parity* is an extra bit attached to a code to check that the code has been received correctly. *Odd parity means* that the number of 1's in the code, including the parity bit, is an odd number. Odd or even parity can be generated with exclusive-OR gates, and parity generators are available in IC form. However, the implementing of an arbitrary truth table using MUXs is the important concept.

Procedure

Special 2-Bit Comparator

1. Assume you needed to compare two 2-bit numbers called A and B to find whether A is equal to or greater than B. You could use a comparator and OR the $A > B$ and $A = B$ outputs. The partially completed truth table for the comparator is shown as Table 1 in the report. The inputs are A_2, A_1 and B_2, B_1 , representing the two numbers to be compared. Notice that the A_2, A_1 , and B_2 bits are connected to the SELECT the inputs of the MUX. The B_1 bit is available to be connected as needed. It is therefore listed in a separate column of the truth table. Determine the logic in which the output represents $A > B$ and complete the X column of truth Table 1. The first two entries are completed as an example.

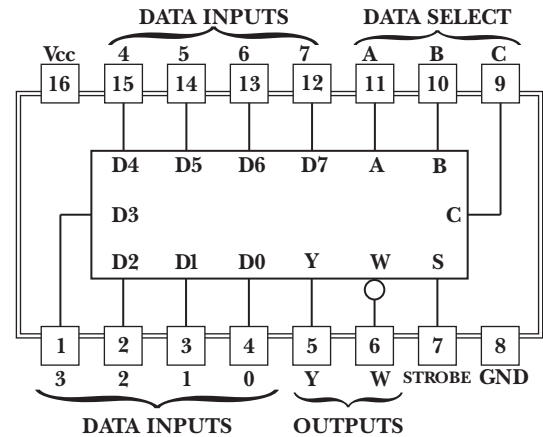
Fig 3:



2. Look at the output X, in groups of two. The first pair of entries in X is the complement of the corresponding entries in B_1 ; therefore, the data should be connected to \overline{B}_1 , as shown in the first line. Complete Table 1 by filling in the last column with either 0, 1, B_1 , or \overline{B}_1 .

3. Using the data from Table 1, complete the circuit drawing shown as Figure 4 in the report. Construct the circuit and test its operation by checking every possible input. Demonstrate your working circuit to your instructor.

The pin configuration for the IC 74151 is Shown Below.



74151 data selector/multiplexer

Data and Observations:

Table 1

Truth table for special 2-bit comparator, $A \geq B$.

Inputs				Output	Connect Data to:
A_2	A_1	B_2	B_1	X	
0	0	0	0	1	\overline{B}_1
0	0	0	1	0	
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	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		

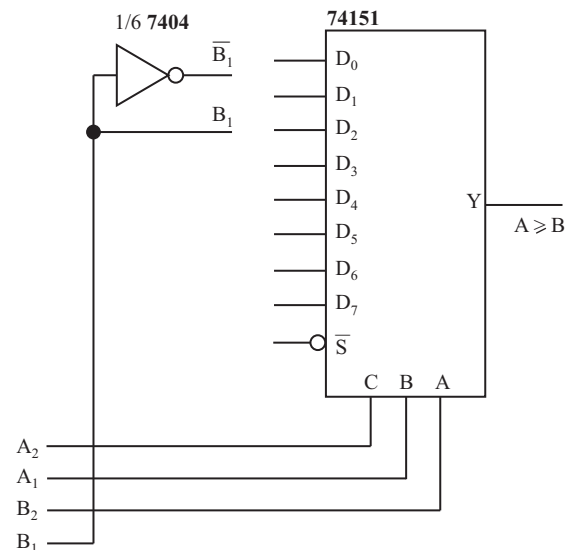


Fig 4

For Further Investigation

Parity Generator Using a Multiplexer

1. The technique to implement an arbitrary function can also generate either odd or even parity. The MUX can generate both odd and even parity at the same time because there are two complementary outputs. One interesting aspect of the parity generator circuit is that any of the four inputs can turn the more than one location. The truth table is shown in the report as Table 2. Four of the bits (A_3 through A_0) represents the parity bit. The requirement is for a circuit that generates both odd and even parity; however, the truth table will be set up for even parity. Even parity means that the sum of the 5 bits, *including the output parity bit*, must be equal to an even number. Complete truth Table 2 to reflect this requirement. The first line has been completed as an example.

2. Using the truth table completed in Step 1, complete the schematic for the even parity generator that is started in Figure 5 of the report. Change your original circuit into the parity circuit and test its operation.

Table 2

Truth table for even parity generator.

Inputs				Output	Connect Data to:
A_3	A_2	A_1	A_0	X	
0	0	0	0	0	A_0
0	0	0	1	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	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		

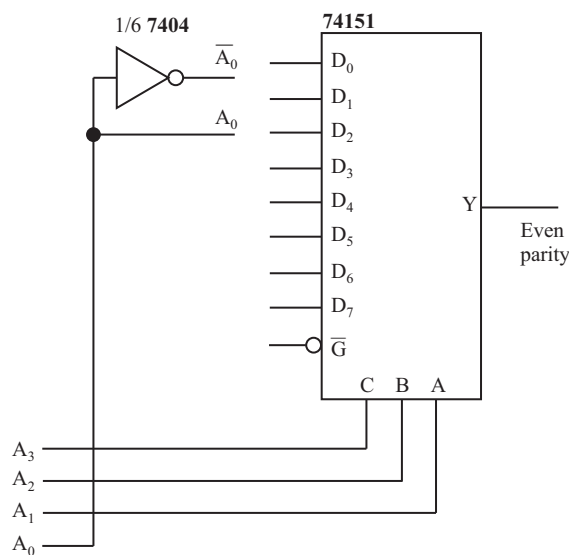


Fig 5

Experiment 10

The D Latch and D Flip-Flop

Objectives

- ❑ Demonstrate how a latch can debounce an SPDT switch.
- ❑ Construct and test a gated 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

7486 quad XOR gate
7400 quad NAND gate
7404 hex inverter
7474 dual D flip-flop

Summary

As you know, combinational logic circuits are circuits in which the outputs are determined fully by the inputs. *Sequential* logic circuits contain information about previous conditions. The difference is that sequential circuits contain memory and combinational circuits do not.

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

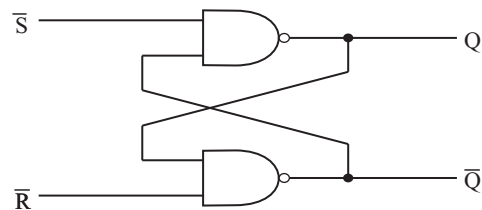


Fig 1:
 \bar{S} - \bar{R} latch.

Equation for top NAND gate:

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

Applying De Morgan's theorem:

$$Q = \bar{S} + \bar{Q}$$

Thus, Q appears on both sides of the equation.

If $S = 1$, then $\bar{S} = 0$ and $\bar{Q} = 0 + Q$ (Q is previous state) output is latched.

A simple modification of the basic latch is the addition of steering gates and an inverter, as shown in Figure 2. This circuit is called a gated D (for Data) latch. An enable input allows data present on the D input to be transferred to the output when Enable is asserted. When the enable input is not asserted, the last level—Q and \bar{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.

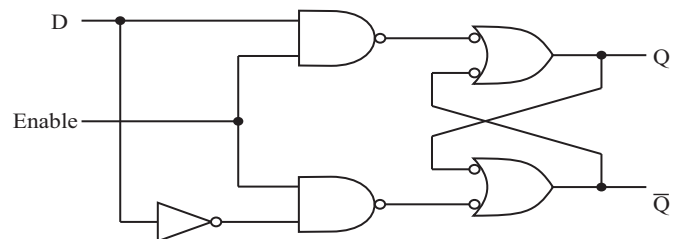


Fig 2:
Gated D latch.

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 either 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. In this experiment, you will also test this IC.

It is useful to review oscilloscope timing. If you are using an analog dual-trace oscilloscope, you should trigger the scope from the channel with the *slowest* of two waveforms that are being compared to be sure to see the correct time relationship. A digital scope will show it correctly for either trigger channel.

Procedure

$\bar{S}\text{-}\bar{R}$ latch

1. Build the $\bar{S}\text{-}\bar{R}$ latch shown in Figure 3. You can use a wire to simulate the single-pole, double-throw (SPDT) switch.

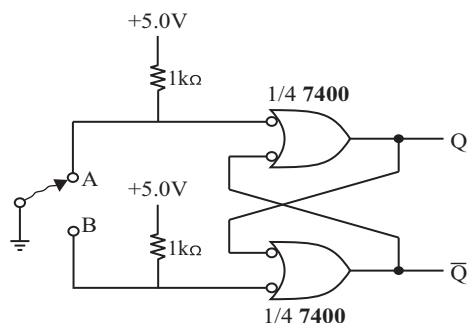


Figure 3:
SPDT switch debounce. The NAND gates are drawn as negative-input OR gates to emphasize the active-LOW.

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.

3. After touching A several times, 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). Summarize your observations of the $\bar{S}\text{-}\bar{R}$ latch used as a switch debounce circuit in the report.

D Latch

4. Modify the basic $\bar{S}\text{-}\bar{R}$ latch into a D latch by adding the steering gates and the inverter shown in Figure 4. Connect the D input to a TTL level pulse generator set for 1 Hz. Connect the enable input to a HIGH (through a 1k resistor). Observe the output; then change the enable to a LOW.

5. Leave the enable LOW and place a momentary short to ground first on one output and then on the other. Summarize your observations of the gated D latch in the report.

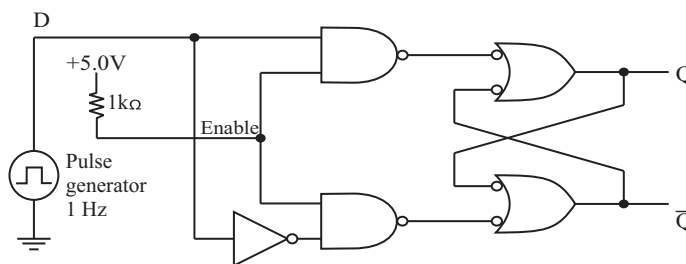


Figure 4:
Gated D Latch.

6. Now make the simple burglar alarm shown in Figure 5. The data input represents switches connected to windows and doors. The enable input is pulled HIGH so the system is activated. To reset the system, put a momentary ground on the Q output as shown. Summarize your observations in the report.

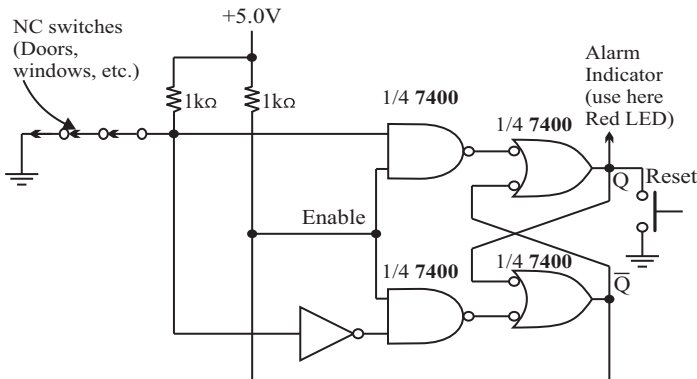


Figure 5:
Simple Burglar Alarm.

The D Flip-Flop

7. The 7474 is a dual, positive edge-triggered D flip-flop containing two asynchronous inputs labeled $\overline{\text{PRE}}$ (preset) and $\overline{\text{CLR}}$ (clear). Construct the test circuit shown in Figure 6. Connect the clock through the delay circuit. The purpose of the delay is to allow setup time for the D input. Let's look at this effect first. Observe both the delayed clock signal and the Q output signal on a two-channel oscilloscope. View the delayed clock signal on channel 1, and trigger the scope from channel 1. You should observe a dc level on the output (channel 2).

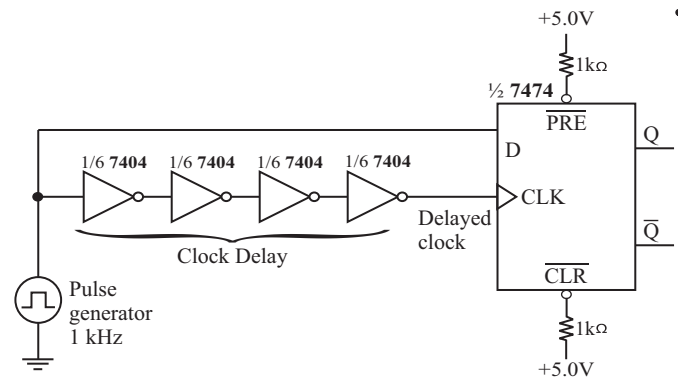


Figure 6

8. Now remove the clock delay by connecting the clock input directly to the pulse generator. The output dc level should have changed because there is insufficient setup time. Explain your observations in the report.

9. Reinstall the clock delay circuit and move the $\overline{\text{PRE}}$ input to a LOW and then a HIGH. Then put a LOW on the $\overline{\text{CLR}}$ input followed by a HIGH. Next repeat the process with the clock pulse disconnected. Determine if $\overline{\text{PRE}}$ and $\overline{\text{CLR}}$ are synchronous or asynchronous inputs.

10. Leave the clock delay circuit in place, but disconnect the D input. Attach a wire from $\overline{\text{Q}}$ to the D input. Observe the waveforms on a scope. Normally, for relative timing measurements, you should trigger the scope using the channel that has the *slowest* waveform as the trigger channel. Summarize, in the report, your observations of the D flip-flop. Discuss setup time, $\overline{\text{PRE}}$ and $\overline{\text{CLR}}$ inputs, and your timing observation from this step.

For Further Investigation

11. The circuit shown in Figure 7 is a practical application of a D flip-flop. it is a parity test circuit that takes serial data (bits arriving one at a time) and performs an exclusive-OR on the previous result (like a running total). The data are synchronous clock; that is , for every clock pulse a new data bit is tested. Construct the circuit and set the clock for 1 Hz. Move the data switch to either a HIGH or a LOW prior to the clock pulse, and observe the result. If a logic 1 is received, what happens when a logic 0 is received?

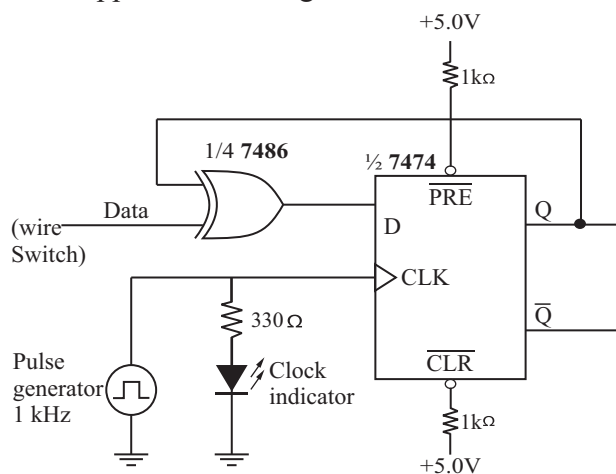
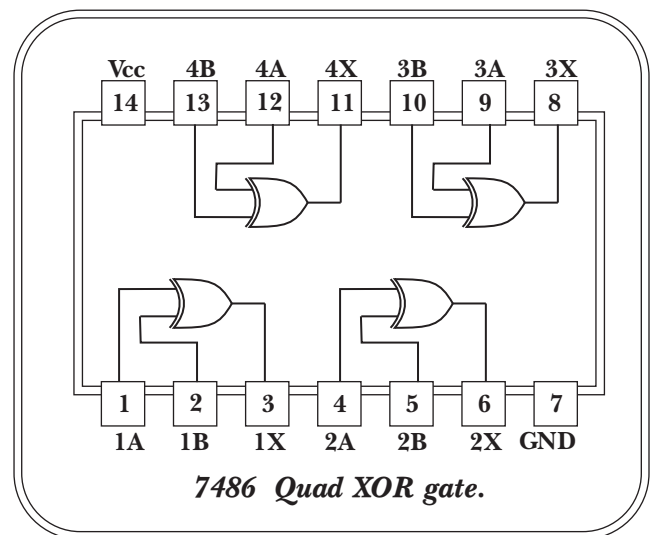
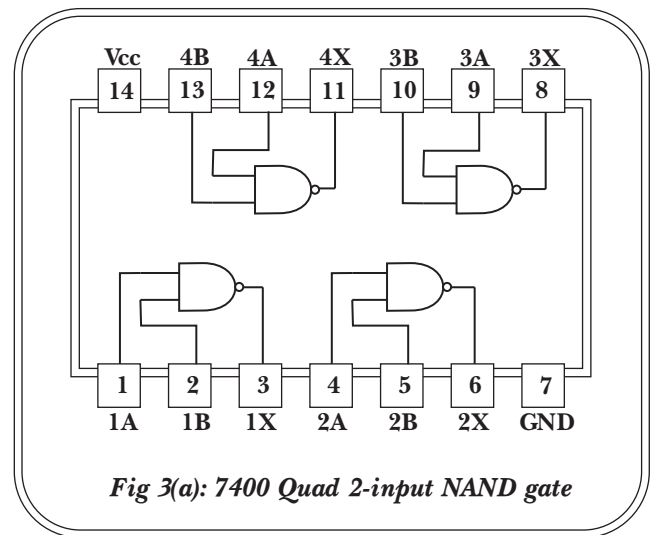
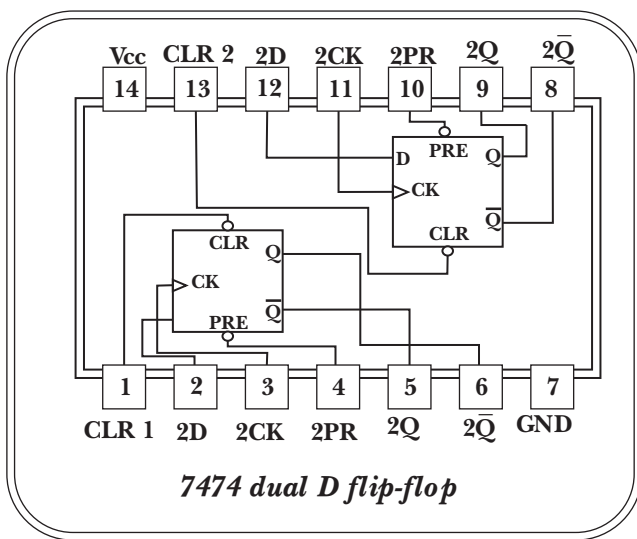


Figure 7



Experiment 11

The J-K Flip-Flop

Objectives

- ❑ Test various configurations for a J-K flip-flop, including the asynchronous and synchronous inputs.
- ❑ Observe frequency division characteristics in the toggle mode.
- ❑ Measure the propagation delay of a J-K flip-flop.

Components

7476 dual J-K flip-flop.

Other material as determined by student.

Summary

The D flip-flop is an edge-triggered device that allows the output to change only during the active clock edge. The D flip-flop can only be set and reset, a limitation for some applications. Furthermore, it cannot be latched unless the clock pulses are removed—a condition that limits applications for this device. The S-R flip-flop could be latched but one combination of inputs is disallowed ($S = 1, R = 1$). A solution to these problems is the J-K flip-flop with additional logic to replace the S-R invalid output states with a new mode called *toggle*. 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; otherwise, it will open.

The J-K flip-flop is the most versatile of the three basic flip-flops. All applications for flip-flops can be accomplished with either the D or

the J-K flip-flops. The clocked S-R flip-flop is seldom used; it is used mostly as an internal component of integrated circuits. The truth tables for all three flip-flops are compared in Figure 1. The inputs are labeled J (the set mode) and K (the reset mode) to avoid confusion with the S-R flip-flop.

S-R flip-flop			D flip-flop		J-K flip-flop		
Inputs		Output	Input	Output	Inputs		Output
S	R	Q	D	Q	S	R	Q
0	0	Latched	0	0	0	0	Latched
0	1	0	1	1	0	1	0
1	0	1			1	0	1
1	1	Invalid			1	1	Toggle

Fig. 1
Comparison of basic flip-flops.

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 of the logic family. In the Further Investigation section, you will investigate the propagation delay for a J-K 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 the preferred method. 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 may be changed. In principle,

Procedure

The J-K Pulse-Triggered Flip-Flop

1. Construct the circuit of Figure 2. The LEDs are logic monitors and are ON when their output is LOW. Select the inactive level (HIGH) for $\overline{\text{PRE}}$ and $\overline{\text{CLR}}$. Select the "set" mode by connecting J to a logic 1 and K to a logic 0. With the clock LOW (not active), test the effect of $\overline{\text{PRE}}$ and $\overline{\text{CLR}}$ by putting a logic 0 on each, one at a time. Are preset and clear inputs synchronous or asynchronous?

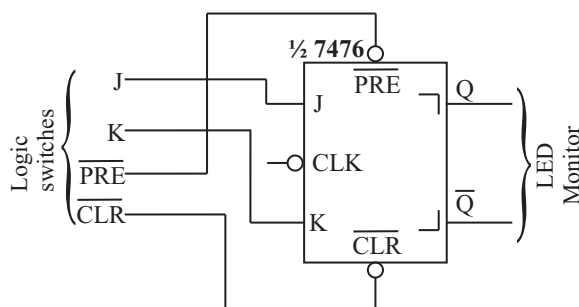


Fig. 2

put $\overline{\text{CLR}}$ on LOW; then pulse the clock by putting a HIGH, then a LOW, on the clock. Observe that the $\overline{\text{CLR}}$ input overrides the J input.

2. Determine what happens if both $\overline{\text{PRE}}$ and $\overline{\text{CLR}}$ are connected to a 0 at the same time. Summarize your observations from this step in the report.

3. Put both $\overline{\text{PRE}}$ and $\overline{\text{CLR}}$ on a logic 1. Connect a TTL level pulse generator set to 1 Hz on the clock input. Add an LED clock indicator to the pulse generator, as shown in Figure 3, so that you can observe the clock pulse and the output at the same time. Test all four combinations of J and K inputs while observing the LEDs.

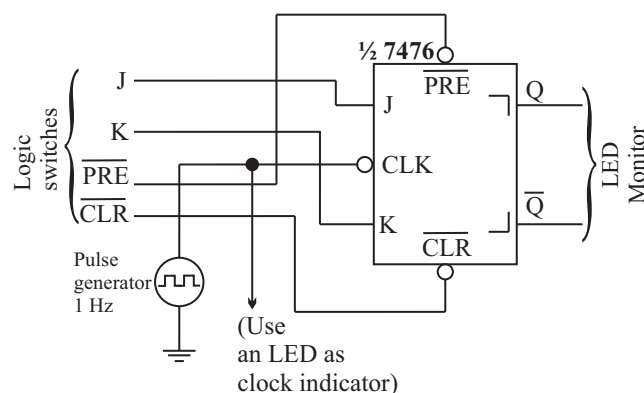


Fig. 3

Are data transferred to the output on the leading or the trailing edge of the clock?

4. Observe that the output frequency is not the same as the clock frequency in the toggle mode. Also note that the output duty cycle in the toggle mode is not the same as the clock duty cycle. This is a good way to obtain a 50% duty cycle pulse. Summarize your observations in the report. Include a discussion of the truth table for the J-K flip-flop.

5. Look at the circuit shown in Figure 4. From your knowledge of the truth table, predict what it will do; then test your prediction by building it. Summarize your observations.

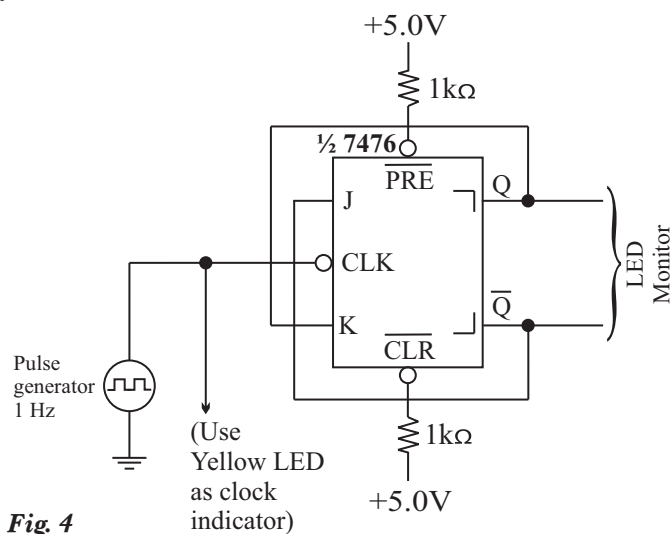
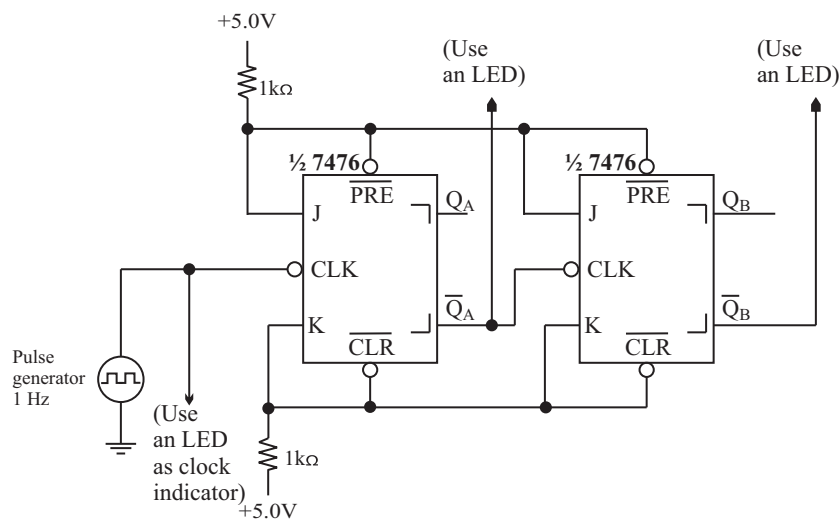


Fig. 4



6. An application of the toggle mode is found in certain counters. Cascaded flip-flops can be used to perform frequency division in a circuit called ripple counter. Figure 4 illustrates a *ripple counter* using the two flip-flops in the 7476. Connect the circuit and sketch the Q_A and Q_B outputs on Plot 1 in the report.

7. Notice that when an LED is ON, the Q output is HIGH. The red and green LEDs indicate that the pulse generator frequency has been changed by the flip-flops.

For Further Investigation

Note: The measurement of a parameter such as t_{PLH} is done differently for analog and digital scopes.

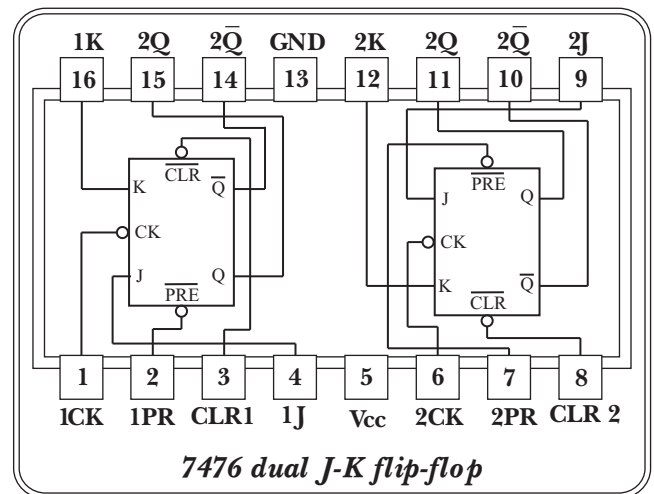
1. Set up the J-K flip-flop for toggle operation. Set the clock frequency for 100 kHz and view the clock on channel 1 and the Q output on channel 2 of your oscilloscope. Set the scope sweep time for 5 s/div to observe the complete waveforms of both the clock and the Q output. Set the VOLTS/DIV control on each channel to 2 V/div and center the two waves across the center graticule of the display.

2. With an analog scope, you will need to trigger the scope on the earlier signal(the clock). Trigger the scope using CH 1 and select falling-edge triggering from the trigger controls. Then increase the sweep speed to 5ns/div (or use the fastest available sweep time if 5 ns/div is not possible). You may need to adjust the trigger

LEVEL control to see the entire clock waveform. You should see a falling edge of the clock and either a rising or falling edge of the Q output. You can observe the LOW-to-HIGH transition of the output by adjusting the HOLDOFF control. When you have a stable trace, go to step 3.

3. With a digital scope on the slower waveform TDS210, you can trigger on the slower waveform(the output) which is on channel 2. This is because the DSO can show signals before the trigger event. From the trigger menu, select Ch2 triggering and select SET LEVEL TO 50%. Then increase the sweep speed to 5 ns/div. In the trigger menu, you can choose between RISING SLOPE or FALLING SLOPE to observe to t_{PLH} or t_{PHL} , respectively.

4. Measure the time from the 50% level of the falling clock signal to the 50% level on the output signal for both a rising and falling output signal. Record your time in the report and compare it to the manufacturer's specified maximum values from the data sheet.



Experiment 12

One-Shots and Astable Multivibrators

Objectives

- ❑ Specify components and trigger logic for a 74121 one-shot to produce a specified pulse and trigger mode.
- ❑ Measure the frequency and duty cycle of a 555 timer configured as an astable multivibrator.
- ❑ Specify components for a 555 timer configured as an astable multivibrator and test your design.

Components

74121 one-shot
7474 dual flip-flop
555 timer
Two $0.01\mu\text{F}$ capacitors
Signal diode (1N914 or equivalent)
And other as determined by the student.

Summary

There are three types of multivibrators: the bistable, the monostable (one-shot), and the astable. The name of each type refers to the number of stable states. The bistable is simply a latch or flip-flop that can be either set or reset and will remain in either state indefinitely. The one-shot has one stable (or inactive) state and one active state, which requires an input trigger to assert. When triggered, the one-shot enters the active state for a precise length of time and returns to the stable state to await another trigger. Finally, the astable multivibrator has no stable state and alternates (or “flip-flops”) between HIGH and LOW by itself. It frequently functions as a clock generator, since its output is a constant stream of pulses. Many systems require one-shot

or astable multivibrators. In this experiment, you will specify the components for the astable multivibrator and test the frequency and duty cycle. In the Further Investigation section, you will design the one-shots.

Most applications for one-shots can be met with either an IC timer or an IC one-shot. A 555 timer is a general-purpose IC that can operate as an astable or as a one-shot. As a one-shot, the timer is limited to pulse widths of not less than about $10\mu\text{s}$ or frequencies not greater than 100KHz. For more stringent applications, the IC one-shot takes over. The 74121, which you will test in this experiment, can provide pulses as short as 30 ns. In addition, integrated circuit one-shots often have special features, such as both leading and trailing edge-triggering and multiple inputs that can allow triggering only for specific logic combinations. As you will see later, these can be extremely useful features. The logic circuit and function table for the 74121 are shown in figure 1.

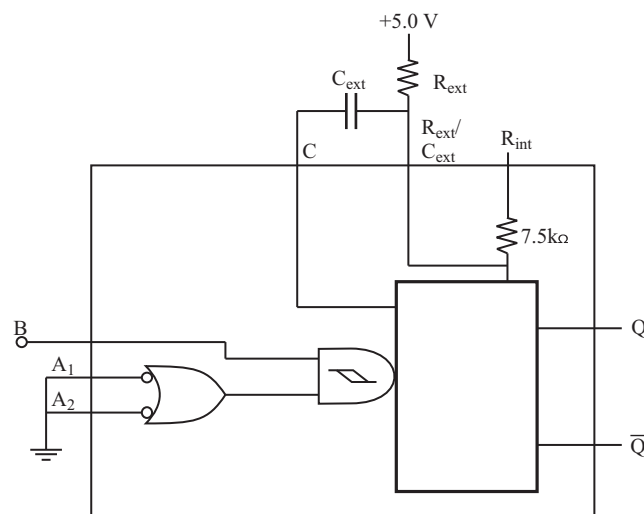


Fig. 1

Inputs			Output	
A ₁	A ₂	B	Q	\bar{Q}
L	X	H	L	H
X	L	H	L	H
X	X	L	L	H
H	H	X	L	H
H	↓	H	⌋	⌋
↓	H	H	⌋	⌋
↓	↓	H	⌋	⌋
L	X	↑	⌋	⌋
X	L	↑	⌋	⌋

H = High logic level
 L = Low logic level
 X = can be either low or high
 ↑ = positive going transition
 ↓ = negative going transition
 ⌋ = a positive pulse
 ⌋ = a negative pulse

Fig. 2

The circuit is triggered by rising pulse on the output of the Schmitt AND gate. The purpose of the Schmitt AND gate is to allow slow rise-time signals to trigger the one-shot. In order for B to trigger it, the input must be a rising pulse, and either A₁ or A₂ must be held LOW, as shown in the last two lines of the function table. If B is held HIGH, then a trailing edge trigger on either A₁ or A₂ will trigger the one-shot provided the other A input is HIGH. Other combinations can be used to inhibit triggering.

Note: You can use A₁ = L, A₂ = L and B input for the edge triggering

This experiment includes an introduction to the 555 timer, the first and still the most popular timer. It is not a TTL device but can operate on +5.0V (and up to +18V), so it can be TTL- or CMOS-compatible. This timer is extremely versatile but has limited triggering logic. Some applications include accurate time-delay generation, pulse generation, missing pulse detectors, and voltage-controlled oscillators(VCOs)

Procedure

Monostable Multivibrator Using the 74121

1. The 74121 contains an internal timing

resistor of 2.0k . You can select the internal resistor for the timing resistor by connecting R_{int} to V_{CC}, or you can select an external resistor. To (VCOs). use an external timing resistor, connect it as shown in Figure 1 with R_{INT} (pin 9) left open. The capacitor is an external component but can be eliminated for very short pulses. The equation that gives the approximate pulse width t_w is

$$t_w = 0.7C_{ext}R_{int}$$

where R_T is the appropriate timing resistor, either internal or external, and C_{ext} is in pF, R_T is in k , and t_w is in ns. Using a 0.01 F capacitor, calculate the required timing resistor to obtain a 50 s pulse width. Obtain a resistor near the calculated value. Measure its resistance and measure the capacitance C_{ext}. Record the computed R_{int} and the measured values of R_{int} and C_{ext} in Table 1 of the report.

2. Using the measured values of R_{int} and C_{ext}, compute the expected pulse width, t_w. Record the computed value in Table 1.

3. Assume that you need to trigger the one-shot using a leading-edge trigger from the pulse generator. Determine the required connections for A₁, A₂, and B. List the input logic levels and the generator connection in your report. Build the circuit.

4. Apply a 10 kHz TTL-compatible signal from the pulse generator to the selected trigger input. Look at the pulse from the generator on channel 1 of two-channel oscilloscope and the Q output on channel 2. Measure the pulse width and compare it with the expected pulse width from Step 1.(You may need to adjust R.) Record the measured pulse width in Table1.

5. Increase the frequency slowly to 50 KHz while viewing the output on the scope. What evidence do you see that the 74121 is not retriggerable? Describe your observations.

The 555 Timer as an Astable Multivibrator

6. One of the requirements for many circuits is a clock, a series of pulses used to synchronize the various circuit elements of a digital system. In the astable mode, a 555 timer can serve as a clock generator.

A basic astable circuit is shown in Figure 3. The capacitor is charged through two timing resistors R_1 and R_2 but is discharged only through R_2 . The duty cycle, which is the ratio of the output HIGH time t_H divided by the total time T , and the frequency f are found by the following equations:

$$\text{Duty cycle} = \frac{t_H}{T} = \frac{R_1 + R_2}{R_1 + 2R_2}$$

$$f = \frac{1.44}{(R_1 + 2R_2)C_1}$$

Measure the value of two resistors R_1 and R_2 and capacitor C_1 with listed values as shown in Table 2. Record the measured values of the components in Table 2. Using the equations given in Step 6, compute the expected frequency and duty cycle for the 555 astable multivibrator circuit shown in Figure 3.

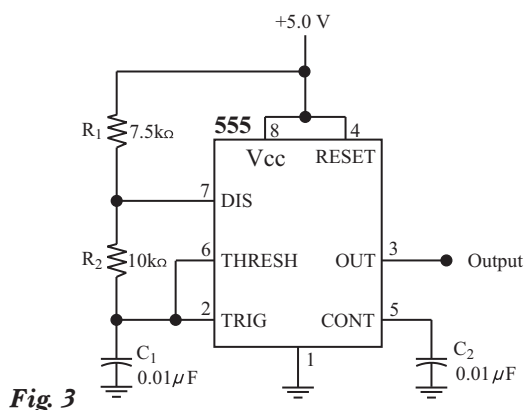


Fig 3

Enter the computed frequency and duty cycle in Table 2 in the report section.

7. Construct the astable multivibrator circuit shown Figure 3. Using an oscilloscope, measure the frequency and duty cycle of the circuit and record Table 2.

8. With the oscilloscope, observe the waveforms across capacitor C_1 and the output waveform at the same time. On Plot 1, sketch the observed waveforms.

9. While observing the waveforms from Step 8, try placing a short across R_2 . Remove the short and write your observations in space provided in the report.

10. A clock oscillator signal, generated from an astable multivibrator. The specified oscillator frequency is 10kHz. The circuit in Figure 3 oscillates at too low a frequency. Modify the design of this circuit so that it oscillates at 10kHz (the duty cycle is not critical). Show the circuit in the space provided in the report.

For Further Investigation

The traffic light control system requires two one-shots, shown in the system as the short timer and the long timer (Figure 4). When the state decoder changes from LOW to HIGH, it causes the trigger logic to change from HIGH to LOW (trailing edge). It is this HIGH-to-LOW transition (trailing edge) that is used to trigger the timers. The short timer must have a 4 s positive pulse and the long timer must have a 25 s positive pulse. Design and build the circuits. An LED (with 330 current-limiting resistor) can be used as an indicator. Test your design to see that the pulse width is approximately correct. Draw the circuits in the report and indicate your test results.

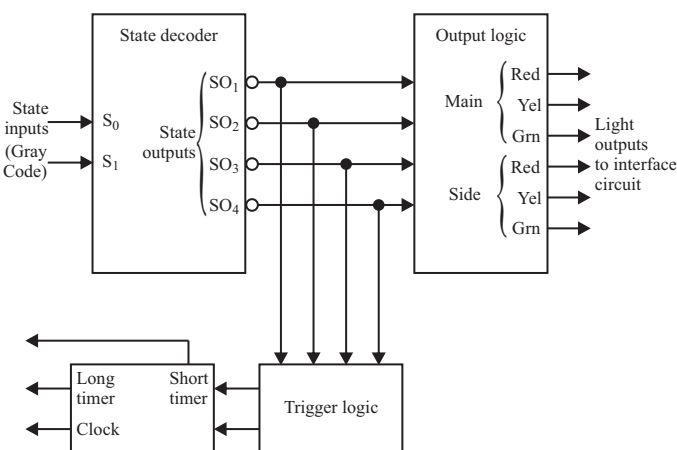


Fig 4

Report for Experiment 12

Data and Observations:

TABLE 1

Data for 74121 monostable multivibrator.

Quantity	Computed Value	Measured Value
Timing Resistor, R_T		
External Capacitor, C_{EXT}	$0.01\mu F$	
Pulse Width, t_w		

TABLE 2

Data for 555 timer as an astable multivibrator.

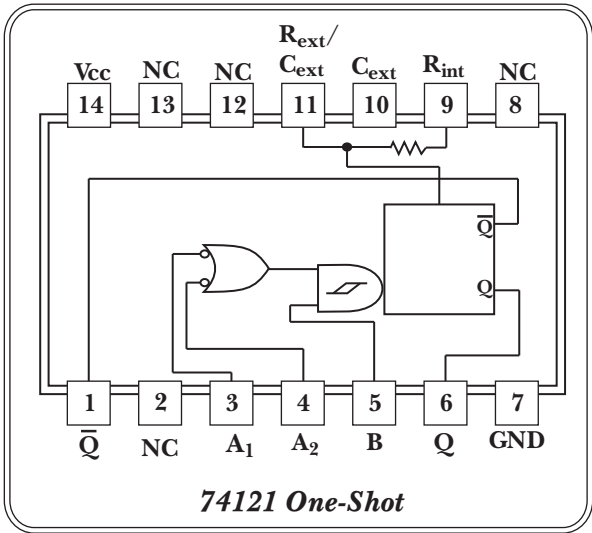
Quantity	Computed Value	Measured Value
Resistor, R_1	$7.5K\Omega$	
Resistor, R_2	$0.01\mu F$	
Capacitor, C_1	$0.01\mu F$	
Frequency		
Duty Cycle		

Step 3. Input logic levels and generator connection:

Step 5. Observations as frequency is raised to 50kHz:

Step 8.

Capacitor waveform:							
Output waveform:							



Result and Conclusion:

Experiment 13

D/A Conversion

Objectives

- ❑ Construct a circuit using a binary counter and a digital-to-analog converter (DAC).
- ❑ Check the calibration of the circuit, measure its resolution in volts/step, and observe its waveforms on an oscilloscope.

Components

74191 up/down counter
MC1408 digital-to-analog converter
seven-segment display with decoder
other material as determined by the student.

Summary

Most physical quantities (pressure, temperature, weight) are analog in nature. Sensors are used to convert these quantities into a proportional voltage or current for electronic control and measuring systems. Most electronic systems use digital techniques for control and processing of data; hence, it is necessary to convert the analog voltage or current into a digital quantity. Conversion from an analog input into a digital converter (ADC). A digital-to-analog converter (DAC) does the reverse — it converts a digital quantity into a voltage or current that is proportional to the digital quantity. (Romance (1999))

The selection of a particular one depends on requirements such as conversion speed, resolution, accuracy, and cost. Resolution is the number of steps the full-scale signal is divided into. It may be expressed as a percentage of full-scale, as a voltage per step, or simply as the number of bits used in

the conversion. Accuracy, which is often confused with resolution, is the percentage difference between the actual and the expected output.

In this experiment, you will test an integrated circuit DAC (the MC 1408) using a binary up/down counter for a digital input. A reference current of nearly 1.0 mA is set up by a 5.1 k resistor connected to +5.0 V. This current is used to set the full-scale output in the 2.0k load resistor.

Procedure

Multiplying DAC

1. Construct the circuit shown in Figure 2. The MC1408 DAC has 8-bit resolution, but we will use only the 4 most significant bits and not bother with a binary-to-BCD decoder. Note how the input pins on the DAC are numbered; the MSB is the lowest number. Set the pulse generator for a TTL-level 1 Hz, and close S_1 . Observe the waveforms at the output of the counter. Sketch the observed waveforms in Plot 1 of the report.
2. Open S_1 . Observe the waveforms from the counter and draw them in Plot 2.
3. Apply a short to ground on the LOAD input of the counter. This parallel-loads the counter to all 1's. Now check the calibration of the DAC. With the short in place, measure the output voltage on pin 4. This represents the full-scale output. Determine the volts per step by dividing the full-scale output voltage by 15 (since there are 15 steps present). Record the voltage and the volts per step in Table 1.

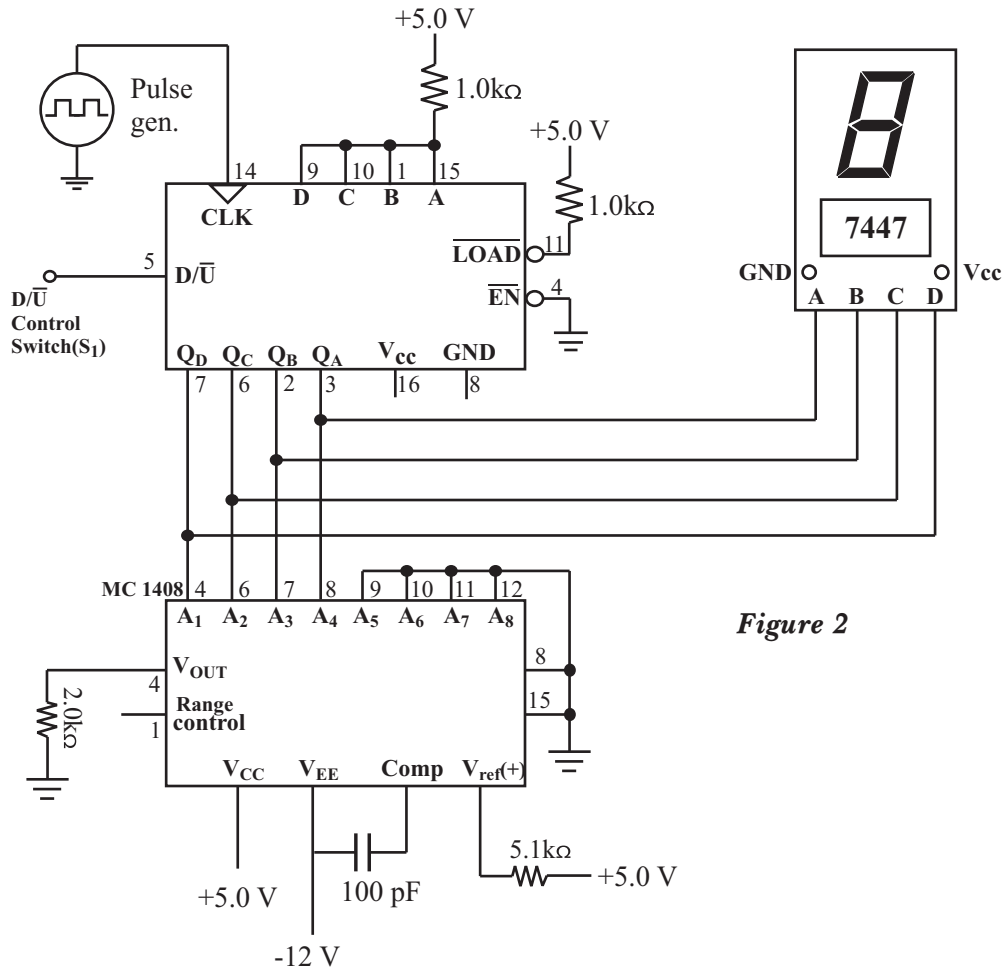
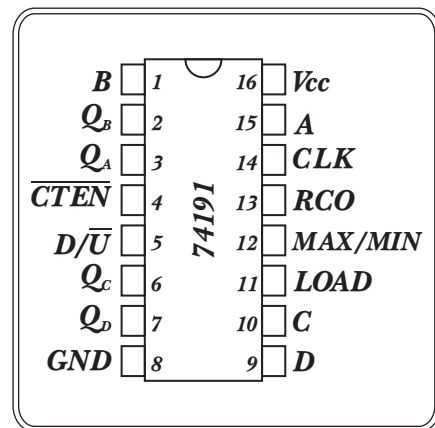


Figure 2

4. Disconnect the short to ground from the $\overline{\text{LOAD}}$ input of the counter. Speed the pulse generator up to 1 kHz. With an oscilloscope, observe the analog output (pin 4) from the DAC. On Plot 3 sketch the waveform you see. Label the voltage and time on your sketch.

5. Close S_1 and observe the analog output from the DAC. Sketch the observed waveform on Plot 4.



Synchronous 4-Bit Up/Down Counter



Experiment 12

Semiconductor Memories

Objectives

- ❑ Complete the design of a memory that stores a series of BCD numbers.
- ❑ Store and read data from the memory.
- ❑ Design a counter to sequence automatically through specified addresses.

Components

Seven-segment display with decoder
74121 one-shot
79LS189 (16×4) tristate RAM
other materials as determined by student

Summary

Memories are circuits that can store binary information either temporarily or long term.

In memories, the number of bits treated as one entity is considered the word size. A word is the number of bits that are accessed at one time and can vary from as little as 1 bit to as many as 64-bits. Keep in mind that for computer applications, a word generally means exactly 16 bits; however, this definition is different for memories.

The outputs of memory and other devices are frequently connected to a common set of wires known as a bus. In order for more than one output to be connected to a bus, the outputs of each device are either tristate or open-collector types. Essentially, tristate or open-collector devices can be electrically disconnected from the bus when not active. Open-collector devices

require a single pull-up resistor is common to all devices that are connected to that same line. Tristate devices do not require a pull-up resistor.

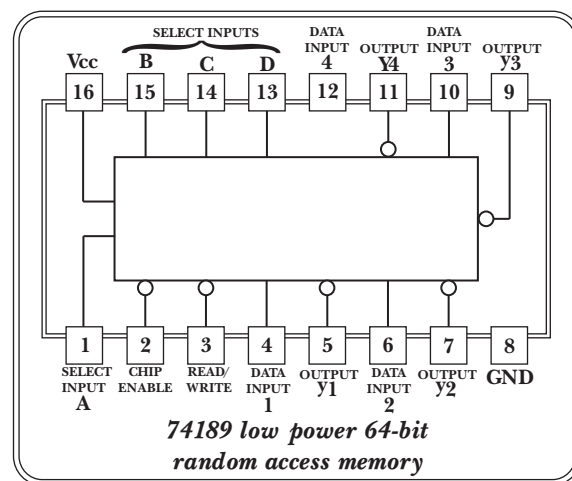
In this experiment, you will the system with a readout of the entry sequence. The memory is a 74LS189 tristate static RAM, a 64-bit memory organized as sixteen 4-bit words. An advantage of the static RAM (SRAM) is that it does not require refresh circuitry. There are four address lines (labeled as SELECT inputs), four data inputs, and four output lines (labeled Y1 through Y4) plus chip enable (\overline{CE}) and a read/write (R/\overline{W}) control line. The output lines are shown with inverting bubbles, hence the complement of the input data is present at the output when reading the data. The tristate outputs are in the high impedance state when the R/\overline{W} line is LOW. Read and write functions for the 74LS189 are summarized in the function table shown as Table 1. To write data, the address is placed on the address bus, data are placed on the data bus, and \overline{CE} is asserted LOW. Then the R/\overline{W} line is pulsed LOW to WRITE. Data are written at the end of this pulse. To read data, the address is placed on the address bus, \overline{CE} is asserted LOW, and the R/\overline{W} line is left HIGH to READ.

Procedure

1. Figure 1 in the report shows a partially completed schematic. Apply negative pulse to the R/\overline{W} line of the memory.

3. After you have programmed the combination into the first five location(0000 through 0100), set the data to cause the display to be blank for address 0101. Then test the circuit by setting the addresses for binary sequence 0000 through 0101 and observing the combination in the seven-segment display followed by the blank display. Demonstrate your working circuit to your instructor.

$\overline{\text{CE}}$	$\overline{\text{WE}}$	Operation	Condition of Outputs
L	L	Write	Complement of data inputs
L	L	Read	Complement of selected word
H	H	Inhibit Storage	Complement of data inputs
H	H	Do Nothing	HIGH



Report for Experiment 12

Data and Observations:

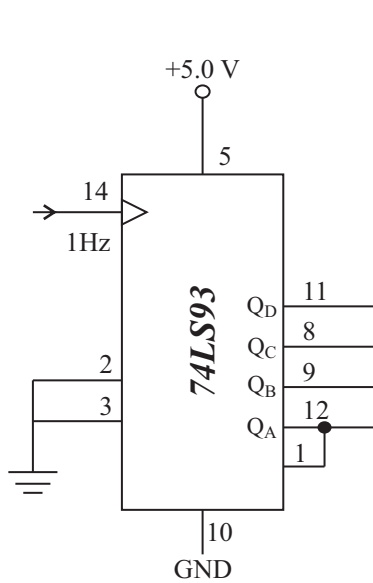


Figure 2

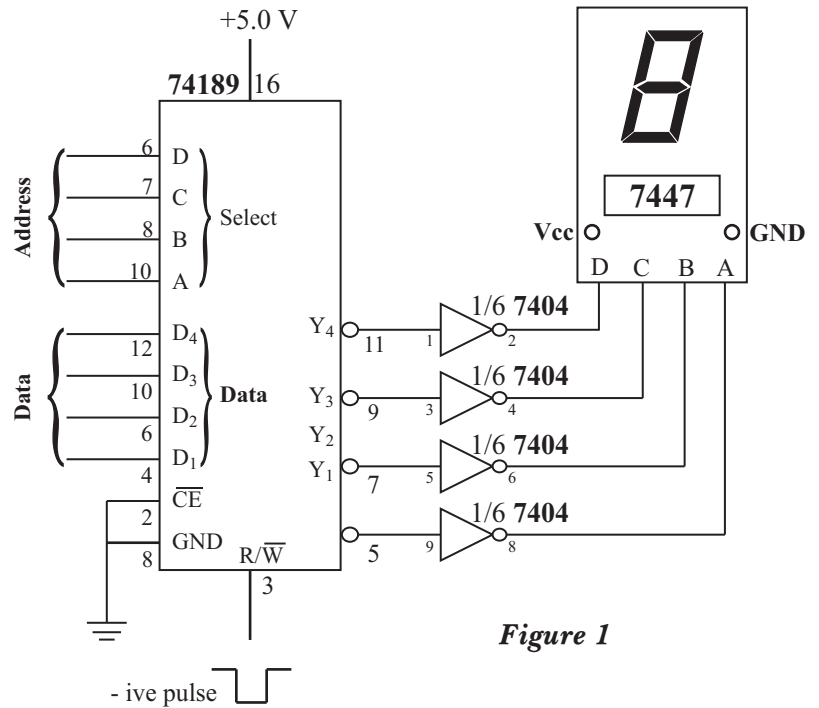


Figure 1

Result and Conclusion:

Further Investigation Results

Experiment 14

Four Digit Decade Counter

Objectives

- ❑ Study the operation of the counter that are available in IC form .
- ❑ Analyze the counter circuit.
- ❑ Cascade the IC counter to achieve higher modulus operation.

Components

4-digit Seven-segment display.
7490 Decade counter.(4)
other materials as determined by student

Summary

A counter is a commonly a digital circuit made up of flip-flop that tells the number of pulses arriving at their inputs counter are classified into tow categories

- (1) Asynchronous counter
- (2) Synchronous counter

Regular binary counter progress through all of its possible states. The maximum modulus of binary counter is a 2^n where n is the no. of flip-flops of the counter. Counter can also be designed to achieve desired number of states in their sequence is called truncated sequence. One common modulus for the counter with truncated sequence is ten. Counter with ten states in their sequence are called decade counter. A decade counter with a count sequence of zero (0000) through nine (1001) is a BCD decade counter.

7490 decade (0-9) ripple counter

The Q_A , Q_B , Q_C , Q_D are the state outputs of 4 internal JK master-Slave flip-flops connected as a standard BCD counter. And additional gating is provided for a divide-by two counter and a three-stage binary counter for which the count cycle length is divide-by-five.

The counter is in two sections: Clock A for Q_A and clock B for the chain of Q_B - Q_C - Q_D . For normal use connect the external clock signal to clock A and connect Q_A to clock B to link the two sections. The output states are as described in table 1.

Table 1
BCD count sequence

Count	Outputs			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

Table 2
Reset/Count function table

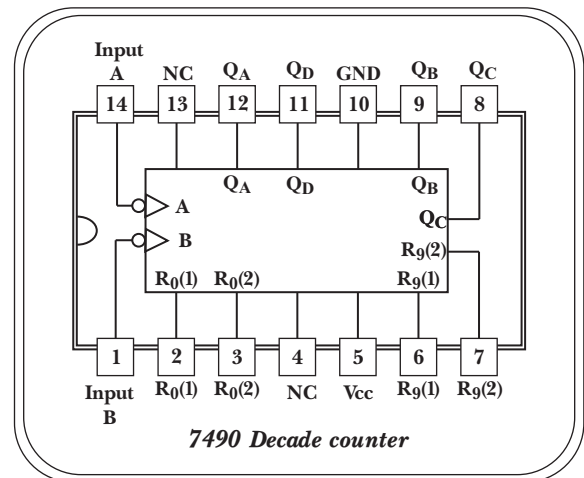
Reset Input				Outputs			
R ₀ (1)	R ₀ (2)	R ₉ (1)	R ₉ (2)	Q _D	Q _C	Q _B	Q _A
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
X	L	X	L	COUNT			
L	X	L	X	COUNT			
L	X	X	L	COUNT			
X	L	L	X	COUNT			

Procedure

- 1 To create a **divide-by-10 counter**, you first connect pin 5 to +5 volts and pin 10 to ground to power the chip.
2. First construct single digit decade counter by connecting pin 12 to pin 1.(Fig. 1)
3. Connect pins 2, 7 to logic 0 and connect pins 3, 6 to logic 1. You run the input clock signal from the clock source in on pin 14.
4. The output appearing on Q_A, Q_B, Q_C and Q_D. Connect these outputs to the A, B, C, D inputs of the seven segment display respectively.
4. For normal operation at least one of the **reset** input should be low.
- 5 Counter can be reset by making both high resets (R₀(1), R₀(2)) High.(see Table 2)

Cascading the counter

To cascade means to arrange two or more similar digital circuits or ICs in a manner such that the output of one circuit or chip provides the input of the next. In the figure 2, four cascaded 7490 decade counters. This circuit, can count from decimal 0 to 9999. The key connection needed to cascade the four 7490 counters: connect pin 11, the D output, of 7490 counter of previous stage to the clock input pin 14 of 7490 counter of next stage. Each counter is connected 7-segment decoder to display the result.



Report for Experiment 14

Data and Observations:

(Apply positive pulse to clear the counter at any stage. (see Table2)

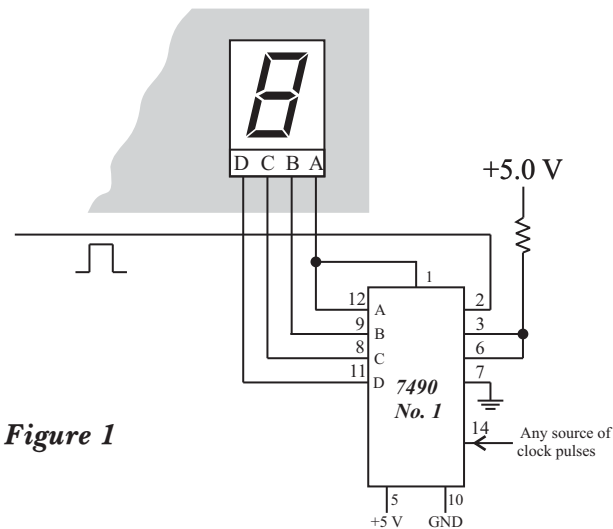


Figure 1

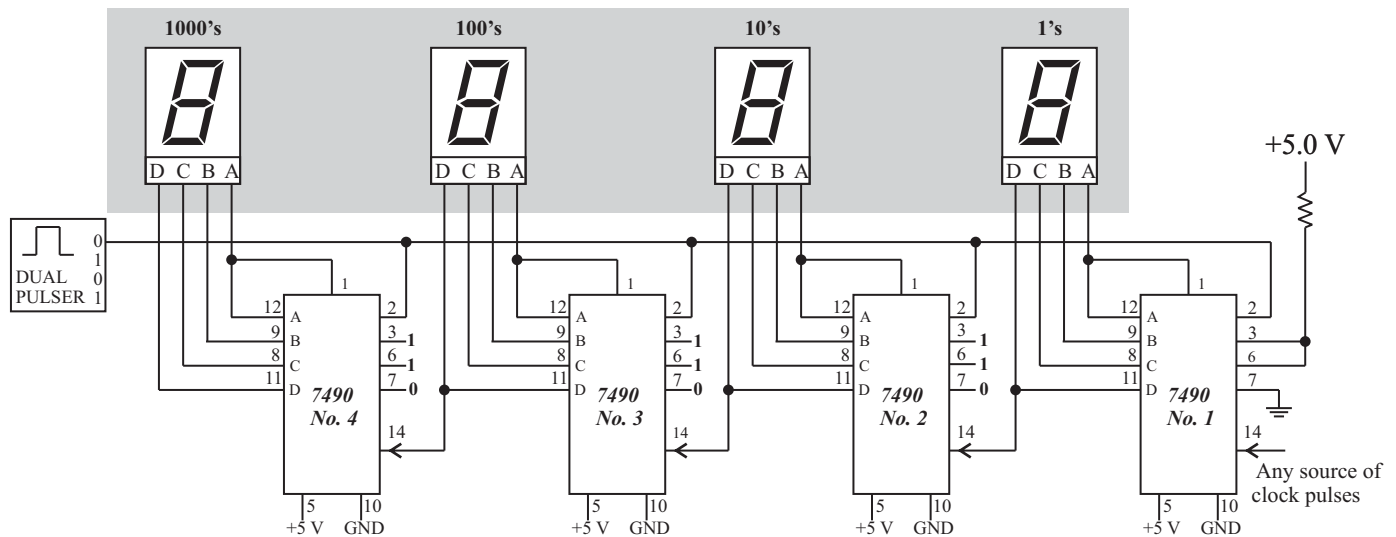


Figure 2

Result and Conclusion:

Experiment 15

Up/Down Counter

Objectives

- ❑ Understand the basic operation of an up/down counter.
- ❑ Use 74190 IC generating the forward and reverse decimal sequence.

Components

Seven-segment display.
74190 Synchronous up/down counter.
other materials as determined by student

Summary

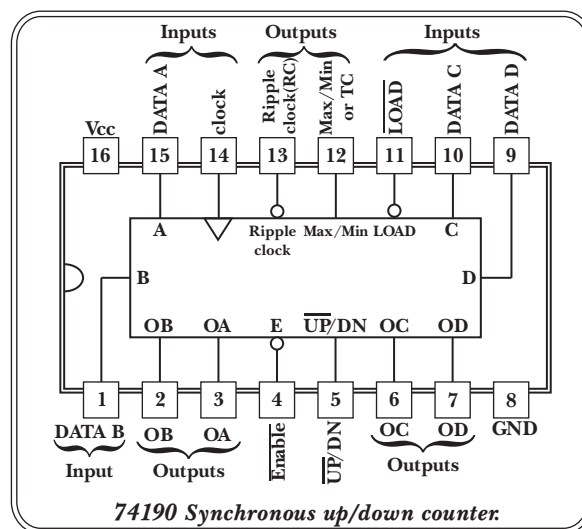
An up/down or bidirectional counter is one which can count increasing as well as in decreasing decimal order. Since up/down counter are able to add as well as subtract from stored count they are often called add/subtract counter. In general most up/down counters can be reversed at any time at any point in their sequence.

74190 up/down Decade counter

The IC 74190 is asynchronously presettable up/down BCD decade counter. It contains four master/slave flip-flops with internal gating and steering logic to provide asynchronous preset and synchronous count-up and count-down operation.

Asynchronous parallel load capability permits the counter to be preset to any desired number. Information present on the parallel data inputs (D_0 to D_3) is loaded into the counter and appears on the outputs when the \overline{LOAD} input is LOW. As indicated in the function table 1, this operation overrides the counting function.

Counting is inhibited by a HIGH level on the count \overline{ENABLE} input. When \overline{ENABLE} is LOW, inter state changes are initiated synchronously by the LOW-to-HIGH transition of the clock input. The up/down input signal determines the direction of counting as indicated in the function table. The \overline{ENABLE} input may go LOW when the clock is in either state, however, the LOW-to-High \overline{ENABLE} transition must occur only when the clock is HIGH. Also, the up/down input should be changed only when either \overline{ENABLE} or clock is HIGH. Over flow/underflow indications are provided by two types of outputs, the terminal count (TC) and ripple clock (\overline{RC}). The TC output is normally LOW and goes HIGH when a circuit reaches zero in the count-down mode or reaches "9" in the count-up-mode. The TC output will remain HIGH until a state change occurs, either by counting or presetting, or until up/down is changed. Do not use the TC output as a clock signal because it is subject to decoding spikes.



Procechure

1. Up/down counter (synchronous) are available in IC form 74LS190 is an example of an IC up/down counter. The count capacity of this counter is increased by cascading two or more ICs.
2. First set up the circuit for single digit as in figure 1. Test the counter for counting in both directions by changing the logic level at pin 5 through a logic switch.
3. A high pulse is produced on Max/Min output when the terminal count (1001) in the up mode and terminal count (0000) in the down mode is reached. You can test this by connecting pin 12 to LED monitor.
4. To preset a value a low level logic is provided on the $\overline{\text{LOAD}}$ input. Apply a BCD digit from the logic switches (say(0111) and load it by applying a negative pulse on pin 11($\overline{\text{LOAD}}$).

5. The TC signal is used internally to enable the $\overline{\text{RC}}$ output. When TC is HIGH and ENABLE is LOW, $\overline{\text{RC}}$ output follows the clock pulse. This feature simplifies the design of multistage counter as shown in figure 2.

6. In this IC, each RC output is used as the clock input to the next higher stage. It is only necessary to inhibit the first stage to prevent counting in all stages, since a HIGH on ENABLE inhibits the RC output pulse as indicated in the function table 1.

7. Pre load a two digit decimal no. the logic switches and confirm the counting from that number.

Table 1

Function Table

OPERATION MODE	INPUTS					OUTPUTS
	$\overline{\text{LOAD}}$	$\overline{\text{U/D}}$	$\overline{\text{ENABLE}}$	Clock pulse	D_n	Q_n
parallel load	L	X	X	X	L	L
	L	X	X	X	H	H
count up	H	L		↑	X	count up
count down	H	H		↑	X	count down
Hold(do nothing)	H	X	H	X	X	no change

Notes

H = HIGH voltage level

L = LOW voltage level

| = LOW voltage level one set-up time prior to the LOW-to-HIGH clock pulse transition

X = don't care

↑ = LOW-to-HIGH clock pulse transition

Report for Experiment 15

Data and Observations:

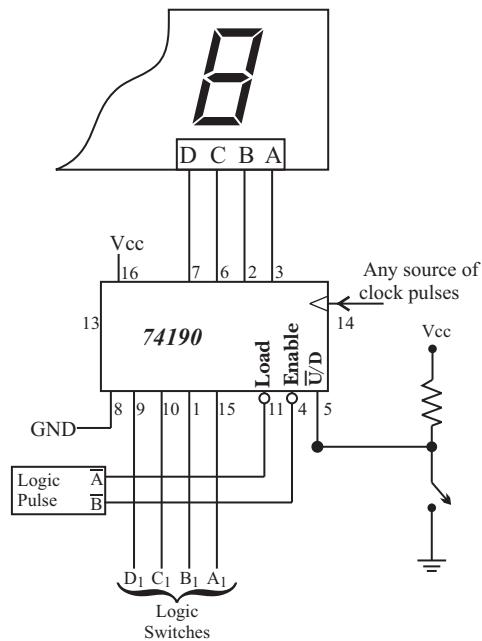


Figure 1

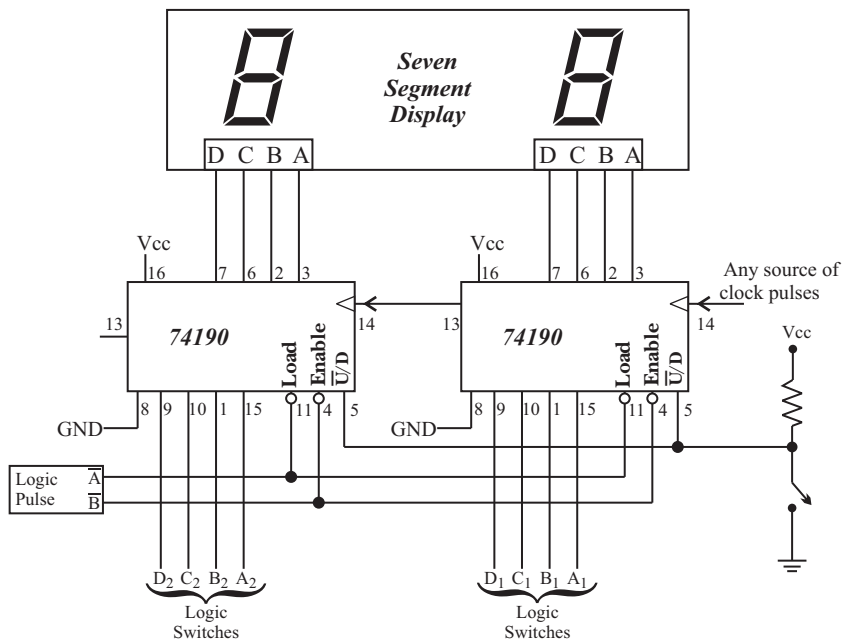


Figure 2

Result and Conclusion:

Experiment 16

24 - Hour Digital Clock

Objectives

Construct the circuit of the 24-hours digital clock using counter IC's 7490 and 7492.

Understanding the operation of the circuit.

Components

An IC 7490(3)

IC 7492(1)

IC 7408(1)

4 Digit Seven segment Display

other material as determined by the students

Summary

Due to advancement in technology many functions that would ordinary require a large number of gates, flip-flop and counters are now provided by some manufacture as a single integrated circuit device. This experiment focuses on the design of a simple digital time clock using TTL devices with display. Counters are the heart of the digital clock system. Many clocks use like the frequency of 50Hz or the main power as their input frequency. This frequency is divided in the seconds, minutes and hours by frequency divider section of the clock. One pulse per second, per minute, per hour is then counted and stored in the count accumulator section of the clock. In this experiment we are going to construct the circuit of minutes and hours of a 24-hour digital clock using the counters.

7492 (Divide-by-twelve counter)

The 7492 is a 4-stage ripple counter containing a high speed flip-flop acting as a divide-by-two and three flip-flops connected as a divide-by-six. HIGH signals on the Master Reset (MR) inputs override the locks and force all outputs to the LOW state.

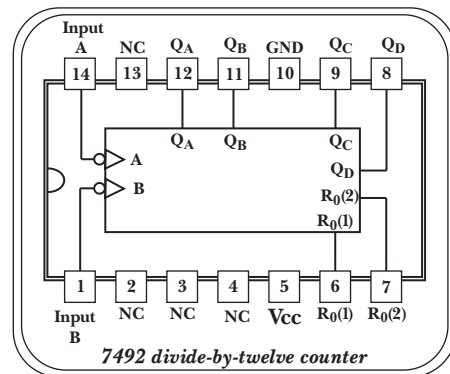


Table 1

Function table for 7492

Reset Input		Outputs			
MR ₁	MR ₂	Q ₀	Q ₁	Q ₂	Q ₃
H	H	L	L	L	L
L	H	COUNT			
H	L	COUNT			
L	L	COUNT			

MR = Master reset

H = HIGH Voltage level

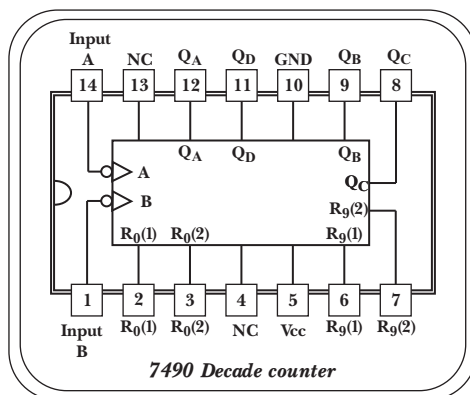
L = LOW Voltage level

Table 2
Reset/Count function table of 7490

Reset Input				Outputs			
R ₀ (1)	R ₀ (2)	R ₉ (1)	R ₉ (2)	Q _D	Q _C	Q _B	Q _A
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
X	L	X	L	COUNT			
L	X	L	X	COUNT			
L	X	X	L	COUNT			
X	L	L	X	COUNT			

7490 decade (0-9) ripple counter

The Q_A, Q_B, Q_C, Q_D are the state outputs of 4 internal JK master-Slave flip-flops connected as a standard BCD counter. And additional gating is provided for a divide-by two counter and a three-stage binary counter for which the count cycle length is divide-by-five.

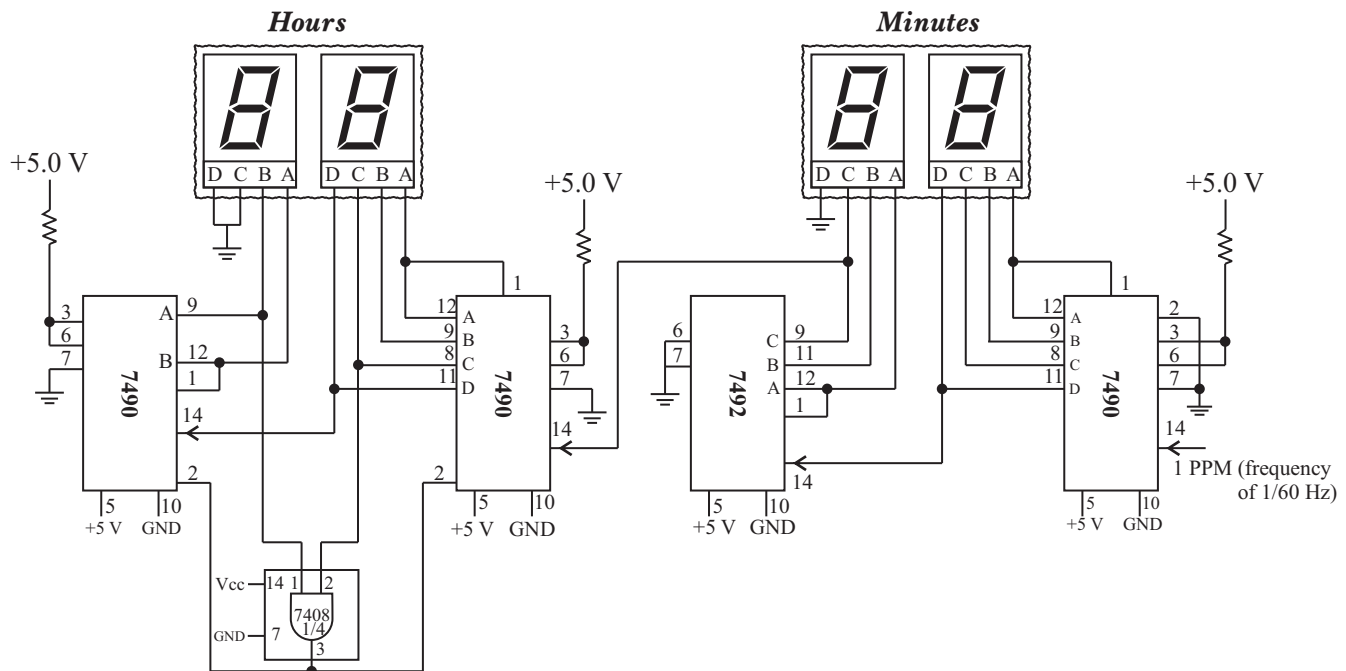


Procedure

1. Construct Digital clock as shown in the figure is a simplified logic diagram of a digital clock that displays minutes and hours.
2. The minutes counter is produced by divide by 60 counter. This counter count from 0 to 59 and then recycle to 00, 7490 counter is used counting 0-9, the first digit and 7492 is used for counting from 0 to 5 for the 2nd digit.
3. Every time minute recycle from 59 to 00. It triggers the hours counters of the clock. The hour's network is made by two cascade divide by 10 counters that go from 00 to 99 but an AND gate (7408) is used to decode 24 hours and reset the both 7490 counters to zero, thus displaying 00 hours.

Report for Experiment 16

Data and Observations:



Result and Conclusion:

Experiment 17

32-Function Arithmetic/Logic Computer

Objectives

Assemble a 32-function digital computer using SC74181 IC.

Test its different operating modes.

Components

IC Sn74181 32-function Arithmetic/Logic Computer

IC 7400 Quad 2-input NAND gate

other material as determined by the students

Summary

In this experiment you will assemble a 32-bit function digital computer using an SN74181 integrated circuit. This IC contains the equivalent of about 75 logic gates that are internally interconnected so as to conveniently provide 32 arithmetic and logic functions. The functions are selected by “setting” four external FUNCTION-SELECT inputs to various combinations of +5 volts (logic 1) and ground (logic 0). You have already become familiar with logic functions such as AND, NAND, OR, NOR, INVERTER and others. Here, now, is a single circuit that can be used to perform these basic logic operations as well as other, more complex ones.

Procedure

1. Plug an Sn74181 integrated circuit and an SN7400 integrated circuit into the socket.

2. Make the connections given in the figure 1, with IC terminals 7 and 8 both connected to +5 volts. This sets up the circuit to perform logic operations. Specific logic functions are selected by making the appropriate settings with S_3 , S_4 , S_5 & S_6 according to table 1. Thus, for example, setting all four FUNCTION-SELECT switches to GROUND (according to line 1 of table 1) yields an INVERTER function involving only the four “A” inputs. For each “A” input, the corresponding output will be its INVERSE.

3. Connect

A_1 input to (logic 1)

A_2 input to (logic 1)

A_4 input to (logic 0)

A_8 input to (logic 0)

i.e. $A_8 A_4 A_2 A_1 = 0 0 1 1$

and set S_3 through S_6 FUNCTION-SELECT switches to GROUND. Each LED MONITOR should indicate the INVERSE of its corresponding “A” input. Note that a given “A” input will not have any effect on the operation of the other “A” inputs or their outputs, i.e., there is not internal carry from one data bit to the next.

4. Set S_3 and S_6 FUNCTION-SELECT switches to +5V (line 16 of table 1). LED MONITORS should now indicate logic levels identical to those of their respective “A” inputs

5. Set S_3 through S_6 FUNCTION-SELECT switches to other combinations of logic 0 and logic 1 so as to generate other logic functions given in table 1. Enter "A" and "B" inputs and record the resultant output levels. Compare your observed results with those that would be predicted on the basis of the Boolean expression involved in table 1.

6. Move the lead on IC terminal 8 to GROUND. This converts the computer from a LOGIC mode of operation to its arithmetic mode. Now, the FUNCTION-SELECT switches, S_3 through S_6 will select arithmetic functions from table 2. Inputs are entered and outputs are indicated in the same manner as before. However, an internal carry is now operative. If, in performing a given arithmetic operation, a carry bit is generated, it will be automatically carried to the next more significant data column, as was described earlier in the section on binary addition and subtraction of numbers. If a carry bit is generated in the column representing the most significant data bit (binary value of 8) it will appear at the Carry OUTPUT (C_{N+4}) is generated in the SN74181 integrated circuit as an inverse function. Therefore, an INVERTER (using a NAND gate in the SN7400) is used to generate a normal CARRY OUTPUT. In this way, the presence of a carry bit is indicated by a "1" (LED monitor ON).

7. Select an arithmetic function by setting FUNCTION-SELECT switches (S_3 through S_6) to the appropriate positions indicated in the table 2.

8. Enter two four-bit binary numbers, A and B, by connecting their corresponding IC terminals to +5 volts for a "1" and to GRD for a "0". The LED monitors will now indicate the arithmetic result according to the function given in table 2 that corresponds to the settings used for S_3 through S_6 .

REMEMBER: In the arithmetic mode of operation, an internal carry is operative. Always check the CARRY OUTPUT for a final carry bit.

Comparators

In addition to the 32 arithmetic and logic functions, the SN74181 also provides two COMPARATOR functions. One of these merely indicates when A equals B. The computer is set up as for subtraction. An LED MONITOR connected to the A = B output at IC terminal 14 will be "ON" whenever A equals B. A "pullup" resistor is required between the A = B output and +5V because it has not been included within the IC. This output's collector has been left "open" by the manufacturer. The second COMPARATOR function is also made operable by setting S_3 through S_6 as for subtraction. IC terminal 7 (C_{IN}) is then connected either to logic 0 or logic 1 according to table 3. The relative magnitudes of A and B are then indicated by a LED MONITOR connected to the CARRY OUTPUT (terminal 14)

9. A = B Comparator

Set $S_6 S_5 S_4 S_3 = 0 1 1 0$

Connect IC terminal 8 to GRD (i.e. $M = 0$). Enter two identical 4-bit numbers into "A" and "B" inputs. The A = B LED MONITOR, should be "ON". Now enter two different 4-bit numbers. This LED should be "off".

10. Magnitude Comparator

Set $S_3 S_4 S_5 S_6 = 0 1 1 0$

Connect IC terminal 8 to GRD (i.e. $M = 0$). Connect terminal 8 of 7400 to CARRY OUTPUT LED. Enter two 4-bit numbers into the "A" and "B" inputs. Now, connect IC pin 7 first to logic 0 and then logic 1, each time noting the condition of C_{IN} . Refer to table 3 to determine the relative magnitudes of A and B.

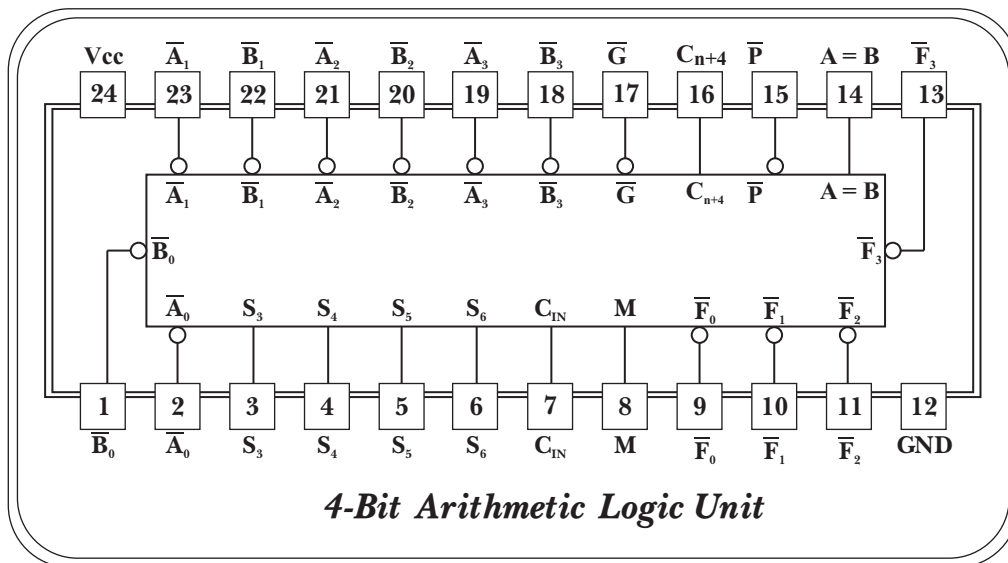
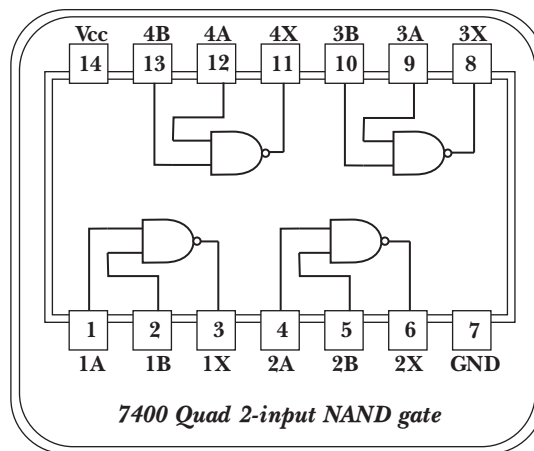
Function-Select

<u>S₃</u>	<u>S₄</u>	<u>S₅</u>	<u>S₆</u>	<u>Function</u>
0	0	0	0	F = \overline{A} (Inverter)
0	0	0	1	F = $\overline{A + B}$ (NOR)
0	0	1	0	F = $\overline{A}B$
0	0	1	1	F = Logical 0
0	1	0	0	F = \overline{AB} (Nand)
0	1	0	1	F = \overline{B} (Inverter)
0	1	1	0	F = A + B(Exclusive -OR)
0	1	1	1	F = $A\overline{B}$
1	0	0	0	F = $\overline{A + B}$
1	0	0	1	F = $\overline{A + \overline{B}}$ (Exclusive -NOR)
1	0	1	0	F = B
1	0	1	1	F = AB(And)
1	1	0	0	F = Logical 1
1	1	0	1	F = A + \overline{B}
1	1	1	0	F = A + B(OR)
1	1	1	1	F = A

Function-Select

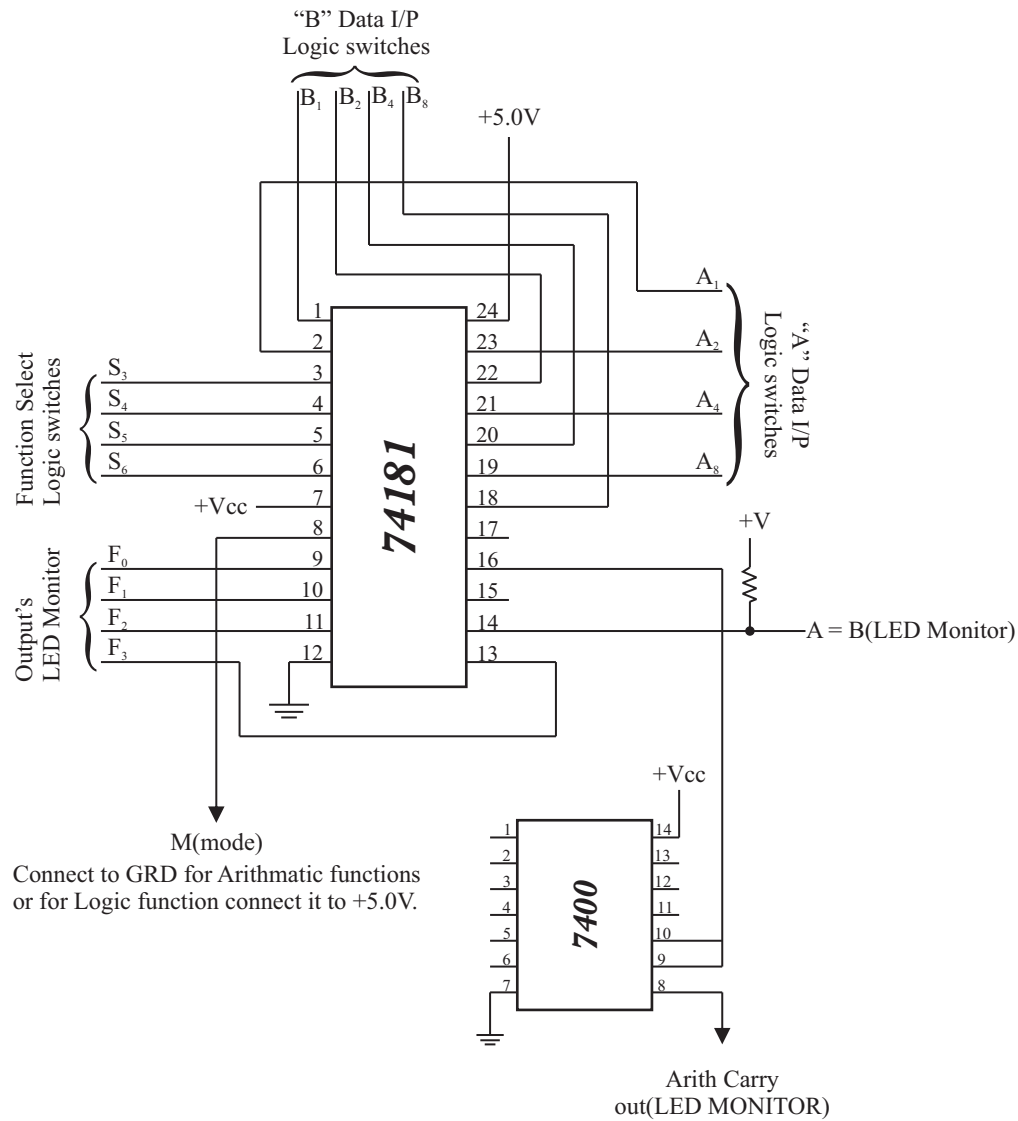
<u>S₃</u>	<u>S₄</u>	<u>S₅</u>	<u>S₆</u>	<u>Function</u>
0	0	0	0	F = A
0	0	0	1	F = A + B
0	0	1	0	F = A + \overline{B}
0	0	1	1	F = Minus 1 (as its 2's complement)
0	1	0	0	F = A plus $A\overline{B}$
0	1	0	1	F = [A + B]plus $A\overline{B}$
0	1	1	0	F = A minus B minus 1
0	1	1	1	F = $A\overline{B}$ minus 1
1	0	0	0	F = A plus AB
1	0	0	1	F = A plus B
1	0	1	0	F = [A + \overline{B}]plus AB
1	0	1	1	F = AB minus 1
1	1	0	0	F = A + A (This results in each bit of binary A shifting to the next more significant position)
1	1	0	1	F = [A + B]plus AB
1	1	1	0	F = [A + \overline{B}]plus AB
1	1	1	1	F = A minus 1

Function-Select				C_{IN}	C_{N+4}	This Means
$\underline{S_3}$	$\underline{S_4}$	$\underline{S_5}$	$\underline{S_6}$	<u>IC Term. 7</u>	<u>If Carry is</u>	<u>That</u>
0	1	1	0	0	0	$A < B$
0	1	1	0	1	1	$A > B$
0	1	1	0	1	0	$A \leq B$
0	1	1	0	0	1	$A \geq B$



Report for Experiment 17

Data and Observations:



Result and Conclusion: