

Laboratory Manual
CSE1003 DIGITAL LOGIC DESIGN

LIST OF EXPERIMENTS

<i>SN #</i>	EXPERIMENT
1	Digital Logic Gates
2	Boolean Algebra
3	Simplification
4	Code Conversion
5	Adders/Subtractors
6	Multiplexers
7	Flip-Flops
8	Counters And Sequential Logic
9	Digital Simulation using ModelSim

ORCAD 9.2 CAPTURE CIS SIMULATION MANUAL

Aim of this manual is to explain beginner users how to do simulations with logic gates. Manual consists of 4 steps and each step is explained with images.

Step 1: Open “Capture CIS”.

Open the Capture CIS program whose icon is shown in . If you can't see this icon on your desktop, search for it from “Start” menu.

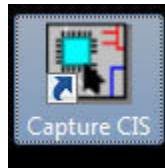


Figure – 1

Step 2: Open a new Project.

When you open the program, **Figure-2** should be seen. As you can see from the figure a new project is opened from File -> New-> Project...

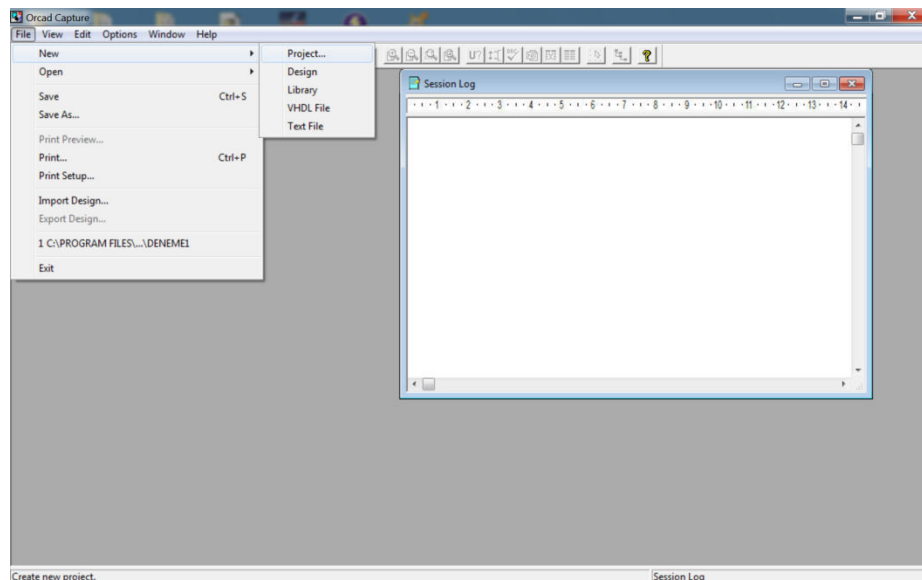


Figure – 2

When you click as shown in **Figure-2**, a new window is opened as seen in **Figure-3**. There are three important points in this window. First one is your project's name; type a meaningful name which describes your project's aim for instance "xor_gate_with_nand_gates", "2_to_1_Mux". Second point is to choose your project type. In our laboratory experiments, you should always choose "Analog or Mixed A/D", otherwise you can't perform simulation. The last point is the location of your project. Create your project to your own student file for easy access.

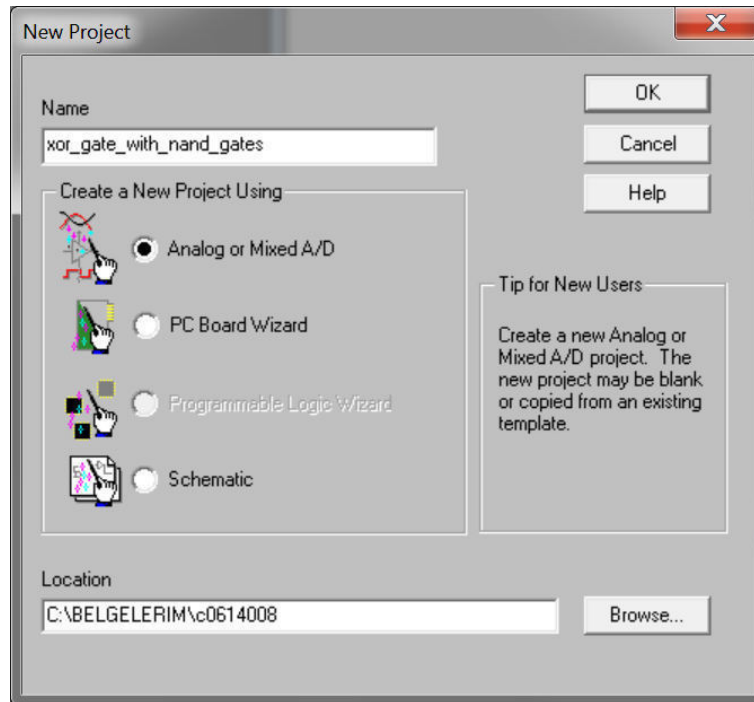


Figure – 3

After pressing "OK" button in Figure-3. Figure-4 is opened which is named as "Create PSpice Project". You should choose "Create a blank project" then press "OK".

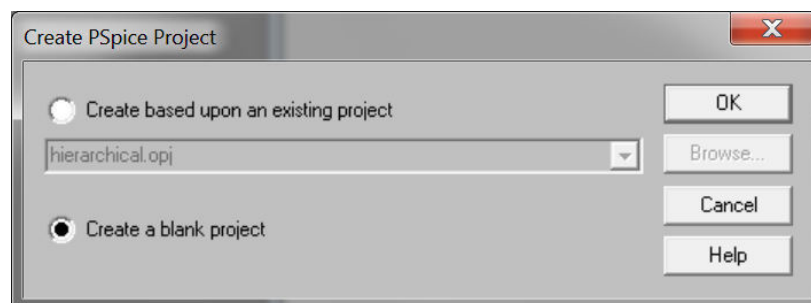


Figure – 4

A new project is opened. Program window should be like **Figure-5**. A schematic sheet is opened.

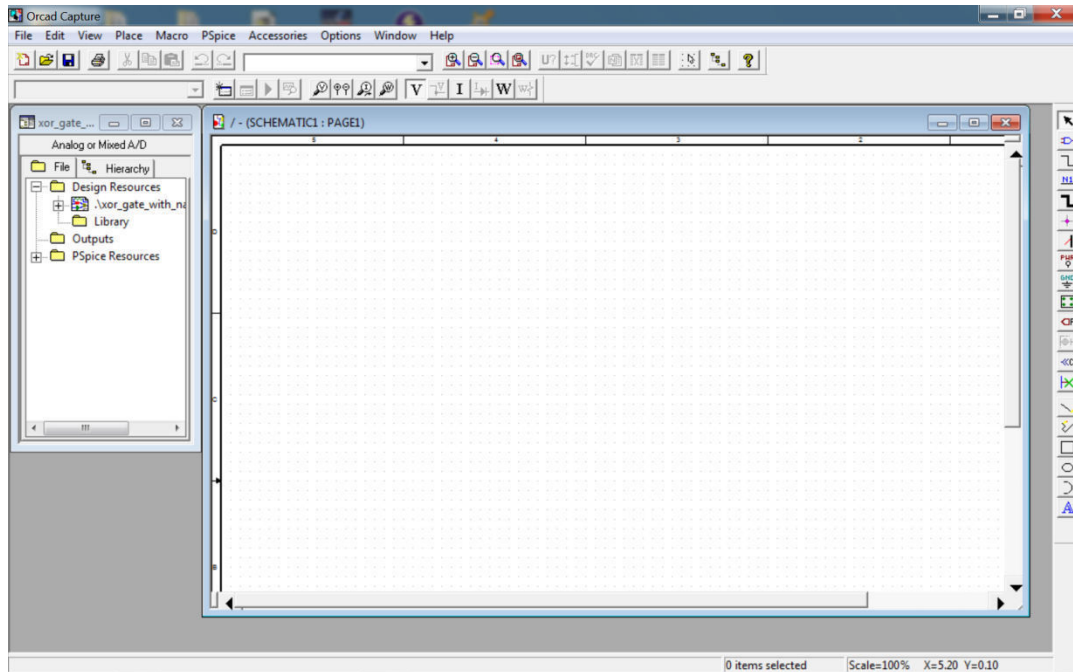


Figure – 5

Step 3: Constructing the logic circuit.

A new project is opened named as “xor_gate_with_nand_gates”. We are going to construct a logic circuit with three NAND gates and two Inverters. Duty of this circuit is performing “xor” gate function. Equation of this circuit is like this:

$$Z = ((X'Y)' (Y'X)')'$$

Eq. 1

To construct the logic circuit first we need to add components to the schematic sheet. There are two ways to add components to the sheet. First one is pressing the place part button. Button is on the rightmost part of the sheet, lies perpendicular to the page, can be seen at **Figure – 6**. Second one is from the Capture Menu by pressing **Place -> Part**.

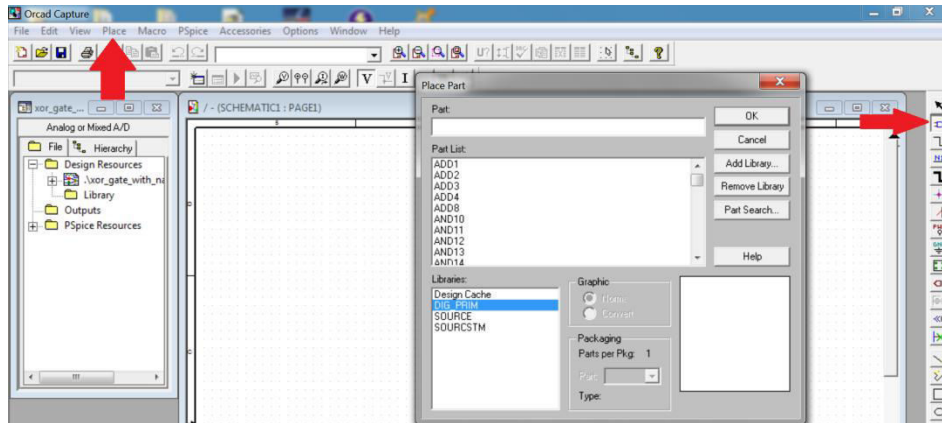


Figure – 6

There are three main component libraries that we are going to use throughout our lab works. Name of these libraries are DIG_PRIM, SOURCE, SOURCSTM. In DIG_PRIM, components like and gate, or gate, nand gate, xor gate, flip_flops, multiplexers are exists. In the SOURCE library, source part of our logic circuits like DigClock,VDC are found. These elements have paramount importance for logic circuit operation; can be thought as power source. The last library is SOURCSTM; we are going to use this library as logic input of our circuits.

Now after introducing the libraries, we are going to add our first element to the circuit. Press place button shown as **Figure -6** and a window like in **Figure - 7** is opened.

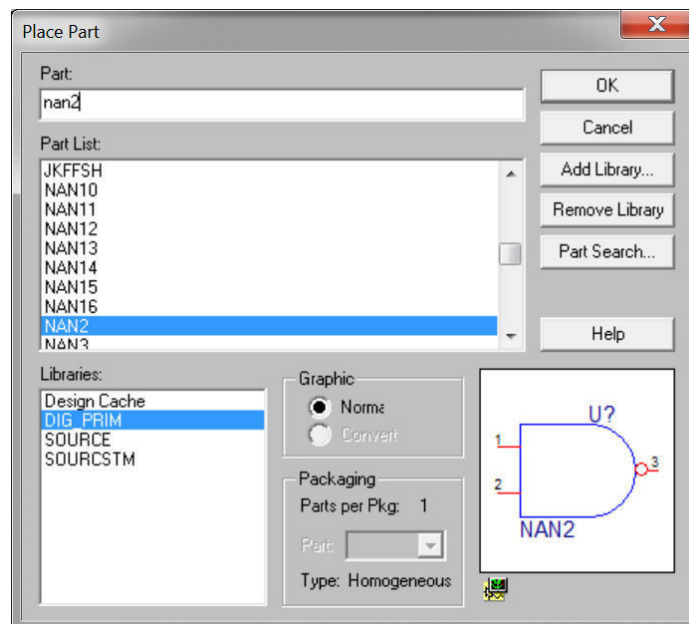


Figure – 7

Choose DIG_PRIM library and type “NAN2” to the search part then press “OK”. Place three NAND gates then to be able to stop placing this component press “ESC” button from the keyboard, **Figure-8**.

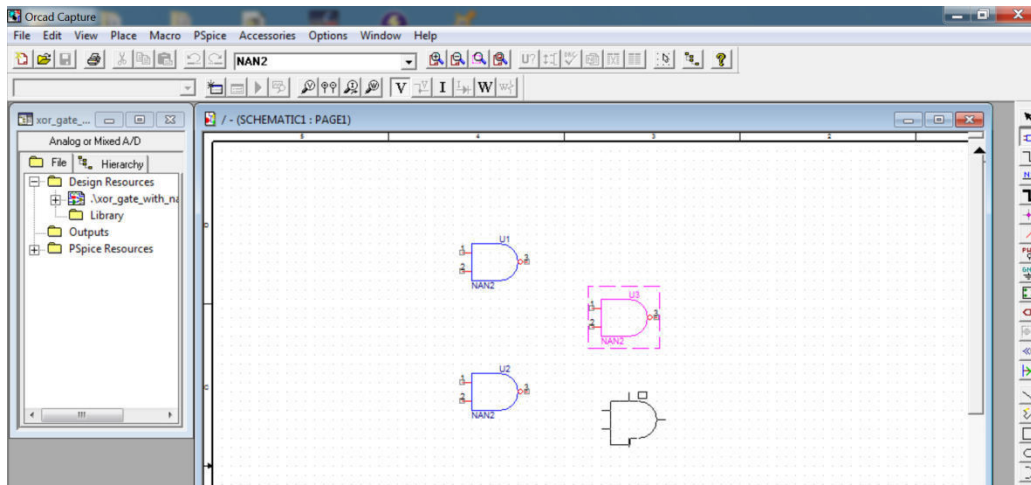


Figure – 8

We need two inverters. Add inverter to the project by typing “INV” to the search part in **Figure - 9**.

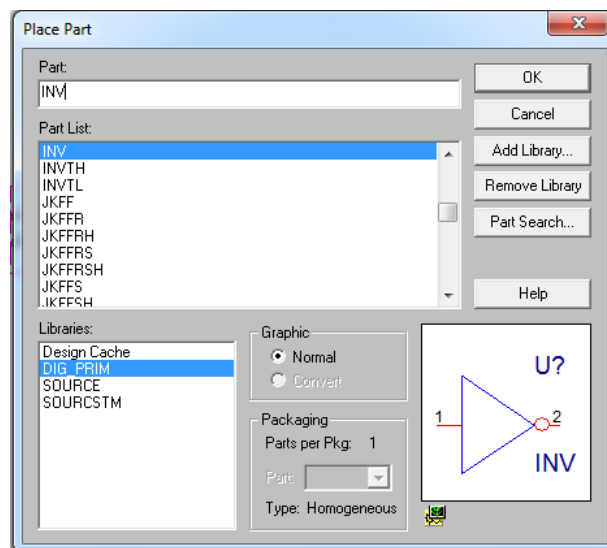


Figure – 9

We place two inverters. Now we need to wire these logic gates to be able simulate the “xor” function. To draw wire lines choose second icon from the rightmost menu as shown in **Figure-10**. Then start wiring as shown, be careful while wiring pin to pin. To make sure they are connected see red dot like in **Figure-10**.

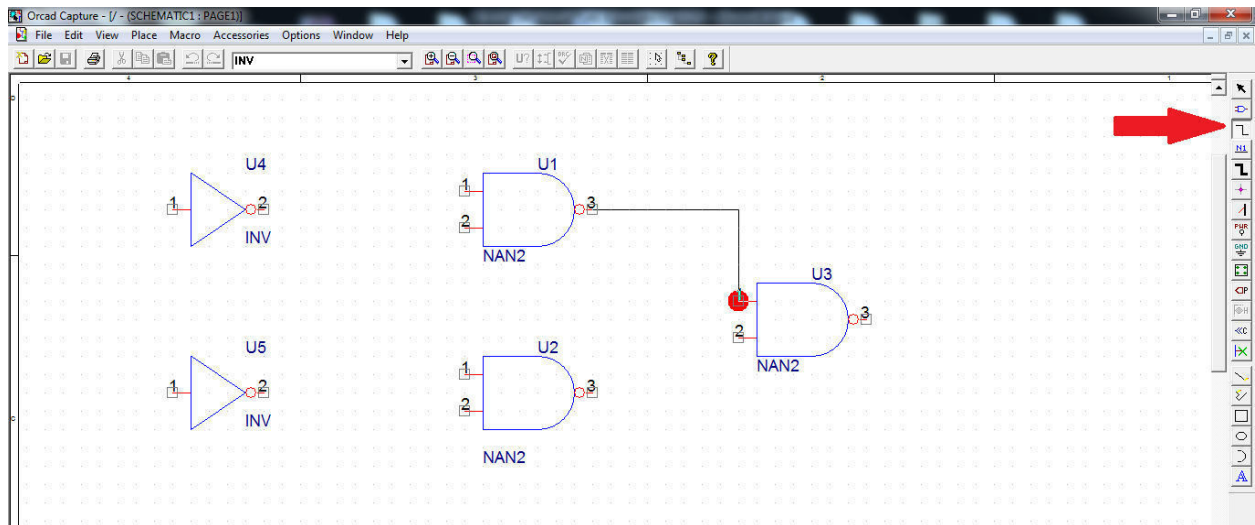


Figure – 10

When you wire the circuit, next step is adding logic inputs x and y to the system. We are going to use two clock signals, which gives us “00”, “01”, “10”, “11” conditions. These conditions are needed to be able to check whether result of “xor” gate function is true or not. To add clock signals to the system, place DigClock from the “SOURCE” library. As a result, circuit at **Figure-11** should be observed. Change OFFTIME and ONTIME values as shown in figure.

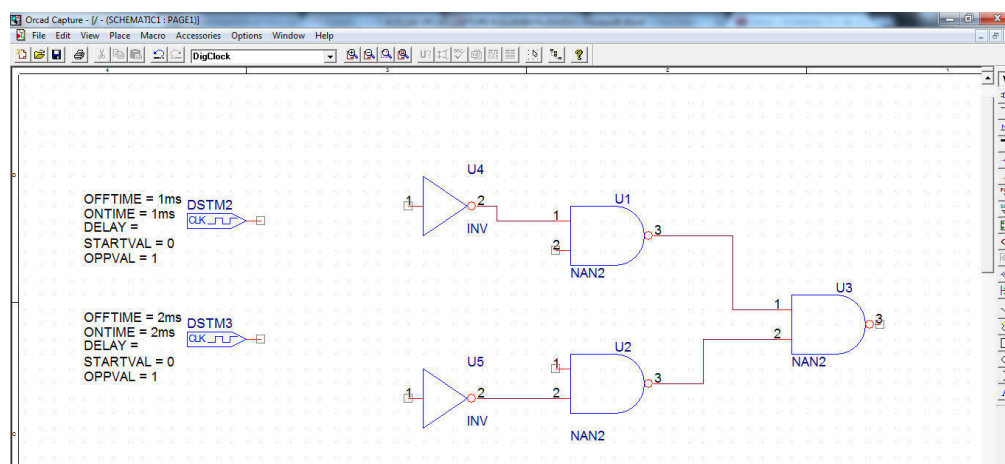


Figure – 11

These clock signals are going to be our input signals of “xor” gate. We can directly connect them to the corresponding places. However, we are going to show a new simple method. By using this method, we simplify the scene of our circuitry. For more complex logic circuits, it will be beneficial.

Press “Place off-page connector” from the rightmost menu, also shown in **Figure-12**.

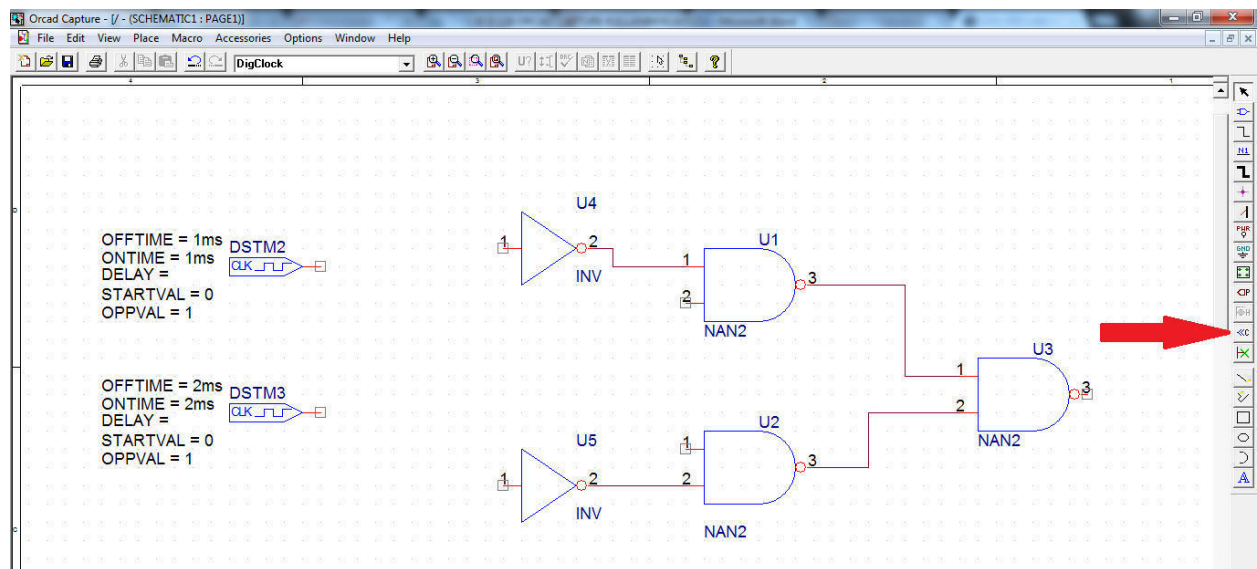


Figure – 12

Choose OFFPAGELEFT-L, **Figure-13.**

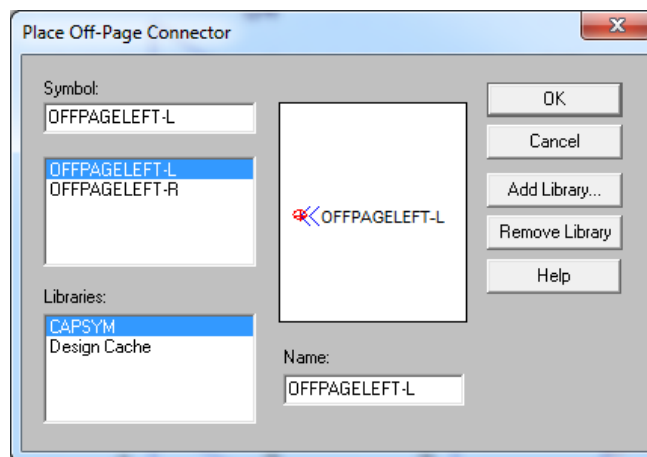


Figure – 13

Place the connector as shown in **Figure-14.** Be careful about red arrows direction. If possible use “R” in keyboard to rotate the connector. Also, as you can see from the figure that connectors are renamed. You should also rename them as shown.

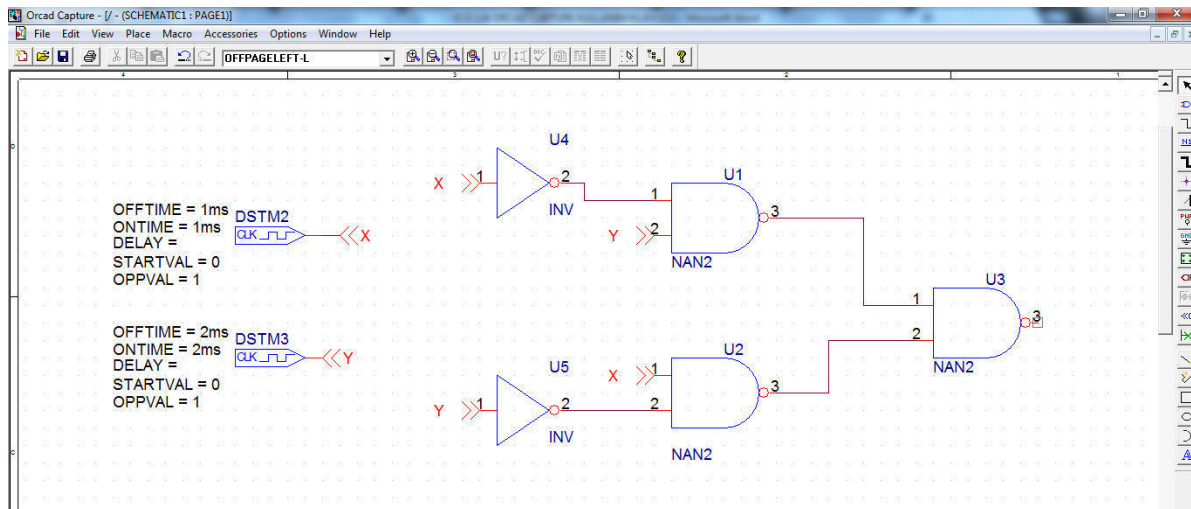


Figure – 14

After this operation our circuit is ready for simulation. We implemented **Eq. 1** completely.

Step 4: Simulation

Simulation is going to be performed from PSpice. To make simulation, firstly we need to open a new simulation profile. Open it, as seen from **Figure-15**.

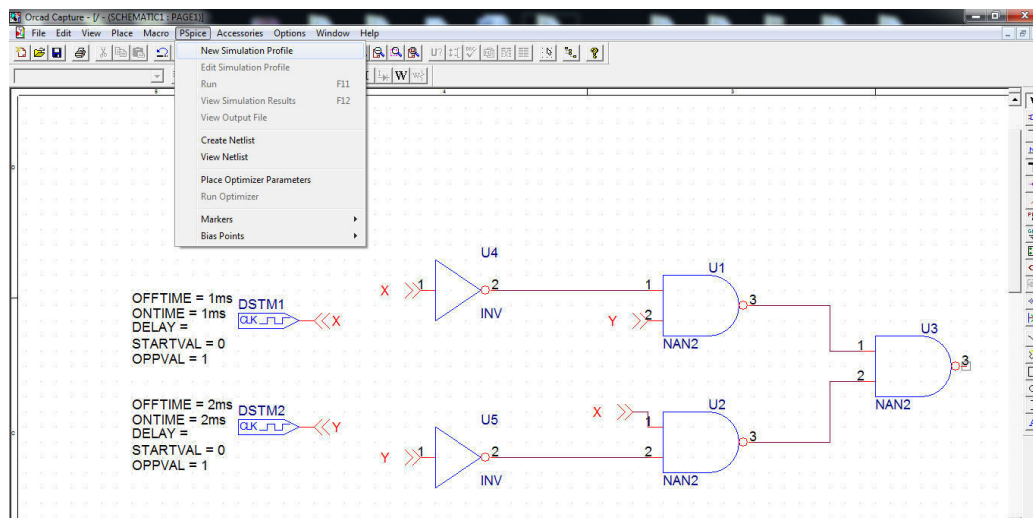


Figure-15

A new window will open. Name your simulation. In this manual, it is named as “xor_sim”, **Figure-16**.

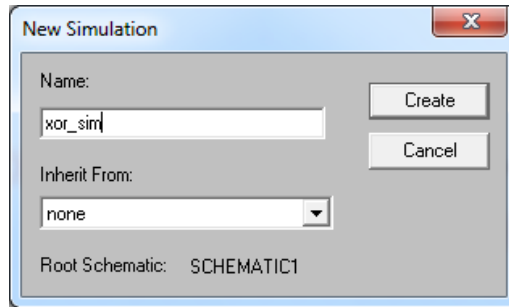


Figure-16

When you press “Create” button a new window is going to be opened, **Figure-17**. Type “Run to time” to 4ms. This value shows us simulation total time. 4ms is sufficient for this experiment. Click “OK” button.

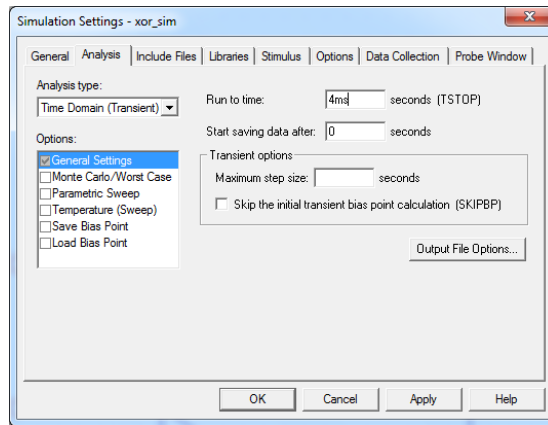


Figure-17

Now, probes should be placed to observation locations. In this example, we want to see two inputs of the system and one output. Probes can be added from the top menu shown in **Figure-18**.

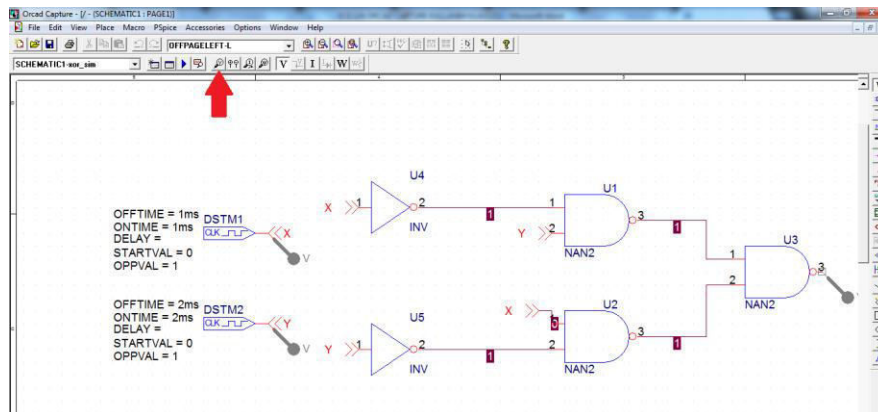


Figure-18

Now, simulation is ready. Press “Run PSpice” button from the top menu, **Figure-19**. A new window is going to be open with corresponding signals.

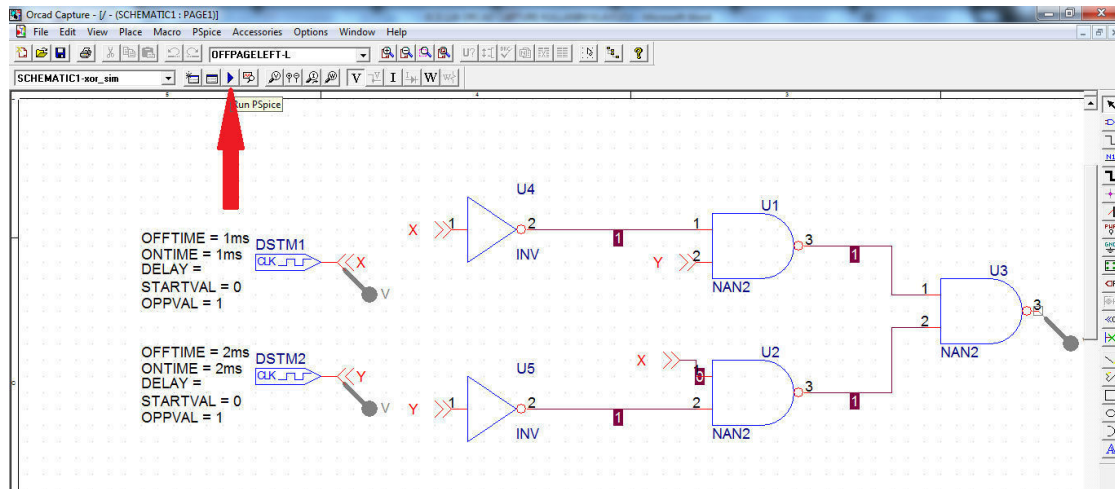
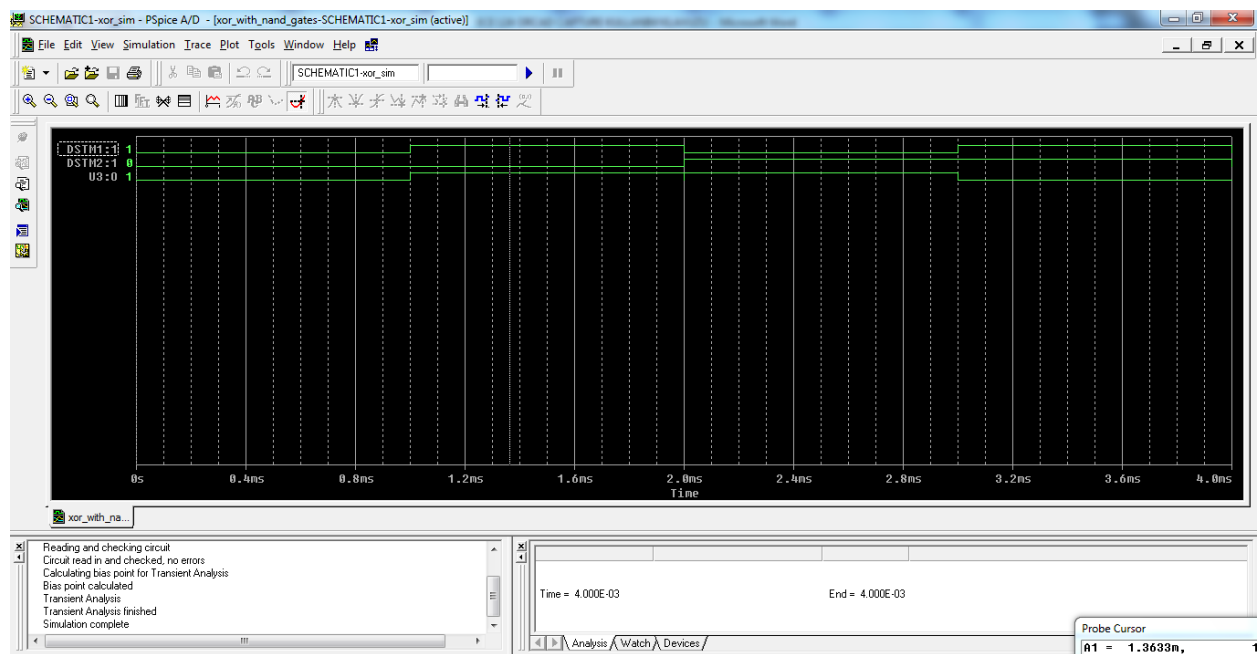


Figure-19

Observation of **Figure-20** shows us that we have achieved the function of “xor” gate.



In the simulation window, for better observation you can use cursor property. Cursor can be opened like in **Figure-21**.

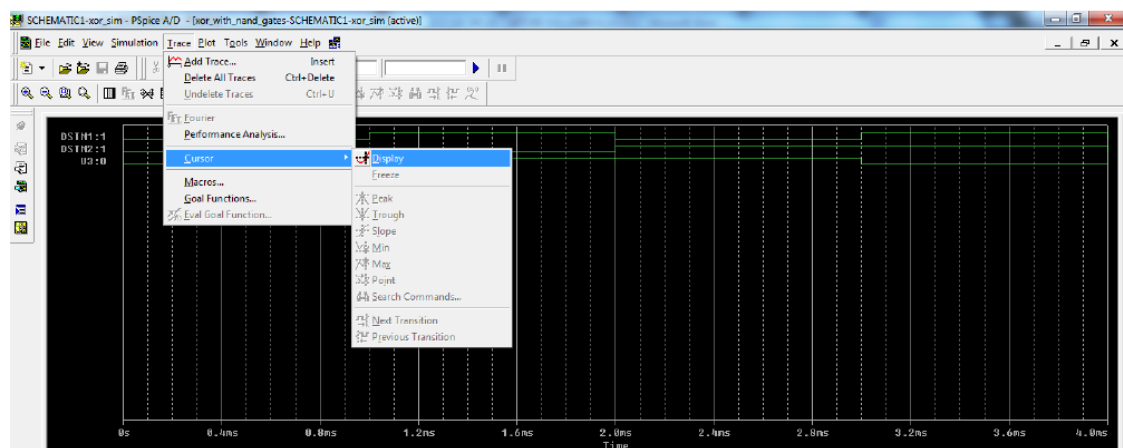


Figure-21

DIGITAL LOGIC GATES

OBJECTIVE:

- To study the basic logic gates: AND, OR, INVERT, NAND, and NOR.
- To study the representation of these functions by truth tables, logic diagrams and Boolean algebra.
- To observe the pulse response of logic gates.
- To measure the propagation delay of logic gates.

APPARATUS:

- IC Type 7400 Quadruple 2-input NAND gates
- IC Type 7402 Quadruple 2-input NOR gates
- IC Type 7404 Hex Inverters
- IC Type 7408 Quadruple 2-input AND gates
- IC Type 7432 Quadruple 2-input OR gates
- IC Type 7486 Quadruple 2-input XOR gate
- IC Type 7493 4-bit ripple counter
- Digi-Designer Logic Board
- Dual-trace oscilloscope

THEORY:

AND	A multi-input circuit in which the output is 1 only if all inputs are 1. The symbolic representation of the AND gate is shown in Fig. 1a.
OR	A multi-input circuit in which the output is 1 when any input is 1. The symbolic representation of the OR gate is shown in Fig. 1b.
INVERT	The output is 0 when the input is 1, and the output is 1 when the input is 0. The symbolic representation of an inverter is shown in Fig. 1c.
NAND	AND followed by INVERT. The symbolic representation of the NAND gate is shown in Fig 1d.
NOR	OR followed by INVERT as shown in Fig 1e.
EX-OR	The output of the Exclusive –OR gate, is 0 when it's two inputs are the same and it's output is 1 when its two inputs are different.

Truth Table Representation of the output logic levels of a logic circuit for every possible combination of levels of the inputs. This is best done by means of a systematic tabulation.

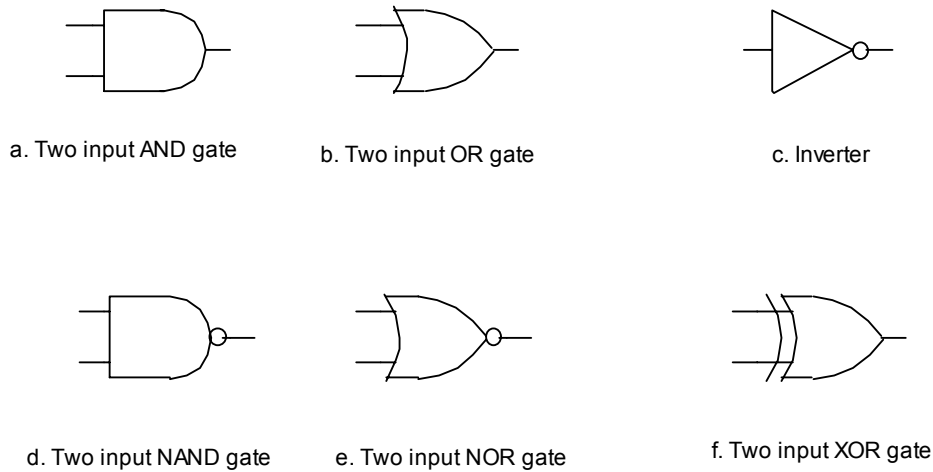


Fig.1 Symbols for digital logic gates

Part 1: Logic Functions

I. AND, OR, NAND, and NOR gates.

1. Use one gate for each IC 7400 (NAND), 7402 (NOR), 7408 (AND), 7432 (OR), 7486 (XOR). Each has input pins, 1 and 2, and output pin 3.
2. Connect pin 1 to switch S1-1, pin 2 to switch S1-2, and pin 3 to LED-1 for every gate as shown in Fig 2 as an example for the NAND gate.

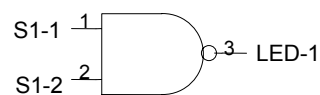


Fig.2 Two input NAND gate

3. Using logic switches S1-1 and S-2, apply the logic levels 0 and 1 to gate inputs (pin 1, pin 2), in the sequence shown in table 1. Record the output logic levels (see lamp LED-1) in table 1. Repeat the recordings for each gate.

Remember: Lamp ON = Logic 1, (High)
Lamp OFF = Logic 0 (Low)

Table 1

Pin 1	Pin 2	Pin 3

4. Use an inverter gate from IC 7404 whose input pin is pin 1 and whose output pin is pin 2.

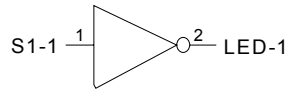


Fig.3 Inverter gate

5. Using logic switches S1-1, apply the logic levels 0 and 1 in the sequence shown in table 2. Record the output logic levels in table 2

Table 2.

Pin 1	Pin 2
0	
1	

Part-2: Response of Logic Gates:

Connect the circuits of figures 4 and 5 and write the corresponding truth tables 3 and 4, respectively.

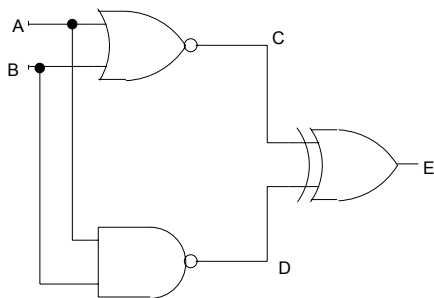


Fig. 4

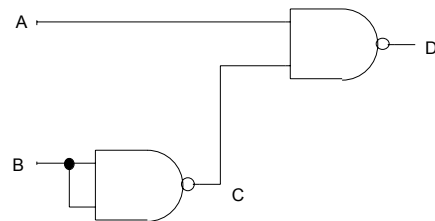


Fig. 5

Table 3.

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

Table 4.

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

Part-3: Propagation Delay in Logic Gates:

Connect all inverters inside two 7404 Ics in cascade. The output will be the same as the input except that it will be delayed by the time it takes the signal to propagate through all six inverters. Set S2 to 100 kHz and apply clock pulses to the input of the first inverter (connect pin 1 to j14) record the wave forms and determine the time delay from the input to the sixth inverter. This is done with a dual trace oscilloscope by applying the input clock pulses to one of the channels and the output of the sixth inverter to the second channel and measuring the delay between the two signals as shown in Fig 6. By using measured delay between two signals calculate the propagation delay for each inverter gate.

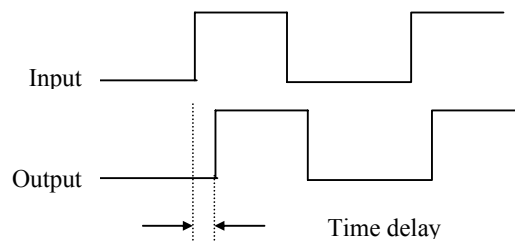
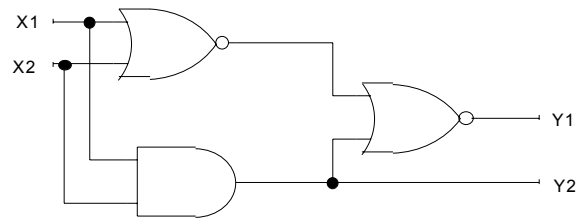
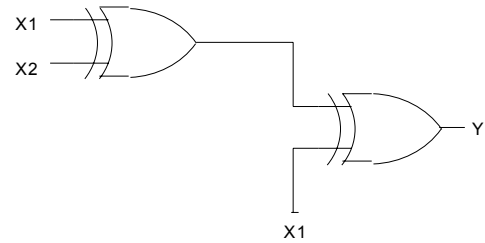
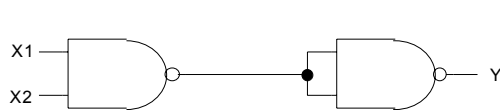
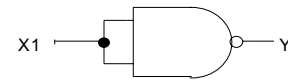
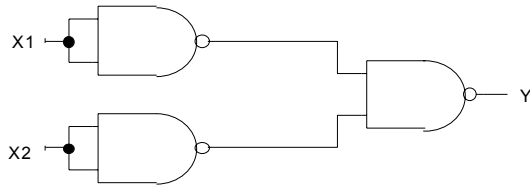


Fig. 6 Propagation delay

Part 4: Review Questions:

1. Write a truth table for each circuit. Derive Boolean expressions for all outputs.



2. A burglar alarm for a car has a normally low switch on each of four doors. If any door is opened the output of that switch goes HIGH. The alarm is set off with an active-LOW output signal. What type of gate will provide this logic? Support your answer with an explanation.

BOOLEAN ALGEBRA

OBJECTIVE:

- To verify the rules and regulations of Boolean Algebra
- To simplify and modify Boolean logic functions by means of Demorgan's theorem.
- To design and implement a logic circuit.

APPARATUS:

- PB-503
- 7400 Quadruple 2 input NAND gates.
- 7402 Quadruple 2 input NOR gates
- 7408 Quadruple 2 input AND gates
- 7432 Quadruple 2 input OR gates
- 7404 Hex inverters
- 7411 Triple 3-input AND gate

THEORY: (See chapter 2 of the textbook)

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

Procedure 1:

- a. Prove rule 1 using LogicWorks. The procedure is:
 - I. Open a new design window
 - II. Choose "ALL LIBRARY" in the Parts Palette
 - III. Put "OR" in the Filter window
 - IV. Select and double click on OR-2
 - V. Move to the cursor back into the circuit window. The cursor on the screen will now be replaced by a moving image of an OR gate.

- VI. Position the OR gate near the center of the circuit window and click the mouse button.
- VII. Press the spacebar to return to point mode.
- VIII. Move again to the Parts Palette and type on the Filter “switch” or part of the word switch e.g. “sw”.
- IX. Select Binary switch and connect it to an input of the OR gate in the design window. (If you want to move the binary switch around, press the shift key while moving it).
- X. Move again to the Parts Palette and select ground to be connected to the other input of the OR gate.
- XI. Using the same method get a Binary Probe and connect it to the output of the OR gate
- XII. Click on the binary switch to change it between 0 and 1 and notice how the rule $A+0 = A$ is satisfied.

In the lab connect the circuit as shown in the figure using the switch S1-1 and LED-1 to verify the rule.

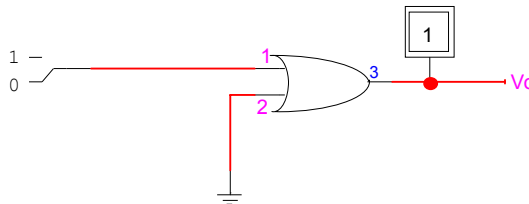


Fig.1 Verifying Rule 1

- b. Connect the circuit of Fig.2 Using LogicWorks. Which rule does this circuit illustrate?

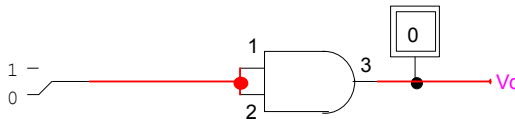


Fig.2

In the lab connect the circuit as shown in the figure using the switch S1-1 and LED-1 to verify the rule.

- c. Design a circuit that illustrates rule 10. Use clock generator of the PB-503 for A and one of the logic switches of S1 for B. Copy the circuit from LogicWorks and paste it in your lab report.
- d. Rule 6 illustrates that $A+A'$ could be replaced with a wire to Vcc. What does rule 8 illustrate?
- e. Rule 11 states that $A+A'B = A+B$. Using LogicWorks design a circuit that illustrates each of these expressions.

$$\begin{aligned} &A+A'B \\ &A+B \end{aligned}$$

Prove that these two circuits perform equivalent logic. (Connect two circuits and show that their outputs are the same).

Procedure 2: Demorgan's Theorem

Proof of equation (1)

Using LogicWorks construct the two circuits given in Figs.3 and 4 corresponding to the functions A' , B' and $(A+B)'$ respectively.

Show that for all combinations of A and B, the two circuits give identical results.

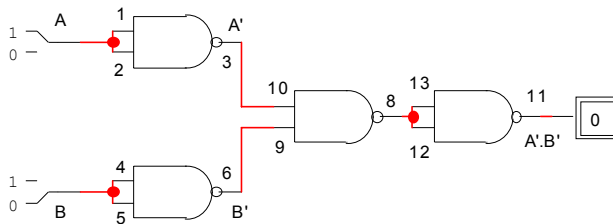


Fig.3

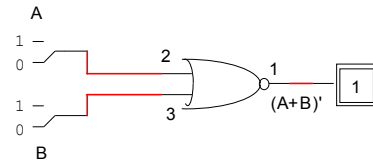


Fig.4

Proof of equation (2)

Using LogicWorks construct two circuits given in Figs. 5 and 6, corresponding to the functions $A'+B'$ and $(A.B)'$ respectively.

Show that, for all combinations of A and B, the two circuits give identical results.

In the lab connect these circuits and verify their operations.

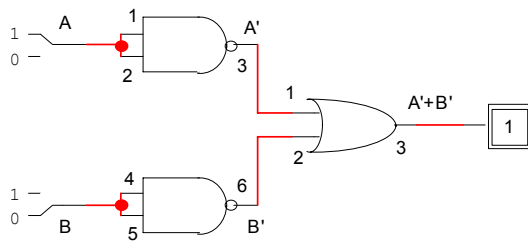


Fig. 5

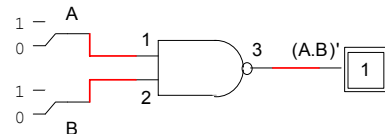


Fig. 6

II. Design of a Digital Circuit

Consider the following problem:

Four chairs A, B, C, and D are placed in a row. Each chair may be occupied ("1") or empty ("0"). A Boolean function F is "1" if and only if there are two or more adjacent chairs that are empty.

1. Give the truth table defining the Boolean function F
2. Express F as a minterm expansion (standard sum of product)
3. Express F as a maxterm expansion (standard product of sum)

4. Using postulates and theorems of Boolean algebra, simplify the minterm expansion of F to a form with as few occurrences of each as possible.
5. Implement on LogicWorks for the pre-lab and then on PB-503, the simplified Boolean function with logic gates and check the operation of the circuit.

Notes:

- In LogicWorks use Binary Switches to represent the four chairs and connect the output of the circuit to a Binary Probe. Check that the Probe is “1” if and only if there are two or more adjacent chairs that are empty.
- For the hardware circuit in the lab, use logic switches S1-1, S1-2, S1-3, and S1-4 to represent the chairs and connect the output of the circuit to LED-1

Result:

Show all truth tables, circuits (using LogicWorks), etc. used in completing this experiment.

**SIMPLIFICATION OF BOOLEAN FUNCTIONS USING K-MAP
TECHNIQUES**

OBJECTIVE:

- To develop the truth table for a combinational logic problem
- To use Karnaugh map to simplify Boolean expressions.
- To draw and simplify sum of products expressions.
- To draw logic diagrams using NAND gates.

APPARATUS:

- PB-503
- 7400 Quadruple 2 input NAND gates.
- 7404 Hex inverters
- 7410 Triple 3-input NAND gates
- 7420 Dual 4-input NAND gates

THEORY:

See chapter 3 of the text, "simplification of Boolean functions"

Procedure:

Part 1: BCD invalid code detector

BCD is a 4-bit binary code representing the decimal numbers 0 through 9.

The binary numbers 1010 through 1111 are not used in BCD.

- a) Construct a truth table containing all possible inputs and desired output. Assume that the desired output for a valid code is a 1, and for an invalid code is 0. Complete the truth table as shown in Table 1. A is the most significant bit, and D is the least significant bit.
- b) Draw the Karnaugh map, and write the simplified Boolean expression for the valid codes as sum of products.

A	B	C	D	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	

- c) Draw the circuit for the above simplified Boolean expression.
- d) Using the universal property of the NAND gate connect an equivalent circuit for these codes that uses only NAND gates.

Part 2: Boolean Functions (1)

1. Simplify the following two Boolean functions by means of Karnaugh maps.

$$F_1(A, B, C, D) = \sum m(0, 1, 4, 5, 8, 9, 10, 12, 13)$$

$$F_2(A, B, C, D) = \sum m(3, 5, 7, 8, 10, 11, 13, 15)$$

2. Draw the logic diagrams for outputs F_1 and F_2 in terms of the inputs A, B, C, and D.
3. Implement and draw the two functions F_1 and F_2 together by using minimum number of NAND gates.
4. Connect the circuit and verify its operation by preparing a truth table for F_1 and F_2 similar to Table 1.

Part 3: Boolean Functions (2)

1. Derive a truth table for the following Boolean Functions.

$$F = A'D + B'D + BC + AB'D$$

2. Draw a Karnaugh map.
3. Combine all the 1's to obtain the simplified function for F .
4. Combine all the 0's to obtain the simplified function for F' .
5. Using logicWorks, implement both F and F' using NAND gates and connect two circuits to the same input switches but to separate output LED's. Prove that both circuits are complement of each other. In the lab implement and verify the operations of the circuit.
6. Draw both circuits.

Part 4: A Majority

A nine member legislative committee requires a 2/3 vote to spend a billion dollars. The vote is tabulated and converted to BCD code. If 2/3 of the committee is in favor, the vote will be the BCD representation of 6, 7, 8, or 9.

1. Derive a truth table for the problem, Table 2.
2. Derive a minimum sum of products expression from the map. {Enter the invalid BCD codes on the map as don't cares (x)}.
3. Using LogicWorks, design a circuit that lights an LED if a majority has voted in favor of spending the billion dollars. Implement this circuit and verify its operation in the lab using hardware.

DESIGN OF CODE CONVERTERS

OBJECTIVE:

1. Design and build gray code to binary converter.
2. Design and build BCD-to-7 segment converter.

APPARATUS:

- Seven segment display.
- SN 7400 quad 2-input NAND gates (1)
- SN 7410 triple 3-input NAND gates (4)
- SN 7420 dual 4-input NAND gates (4)
- SN 7404 HEX inverter (1)
- SN 7446 BCD-to-seven segment decoder.

THEORY:

The conversion from one code to another is common in digital systems. Sometimes the output of a system is used as the input to the other system. A conversion circuit is necessary between 2 systems if each system uses different codes for the same information.

In this experiment you will design and construct 3-combinational circuit converters:

See section 4-5 in your book for further information.

Procedure:

1. *Gray code to Binary converter:*

Gray code is one of the codes used in digital systems. It has the advantage over binary numbers that only one bit in the code word changes when going from one number to the next. (See Table 1).

Design a combinational circuit with 4 inputs and 4 outputs that converts a four-bit gray code number into an equivalent four-bit Binary number. Use Karnaugh map technique for simplification. Use LogicWorks for pre-lab demonstrations. Select the library “7400dev.clf” in the Parts Palette and then select the XOR chip 74-86. This would give you a set of 4 XOR’s as shown in Fig. 1, just like the hardware chip 74-86. You could use as many as needed from these XOR gates in your design. Get back to ALL LIBRARIES and select switches for the inputs and Binary Probes as indicators of the outputs. Verify your design in the pre-Lab. During the Lab construct the circuit and verify its operations.

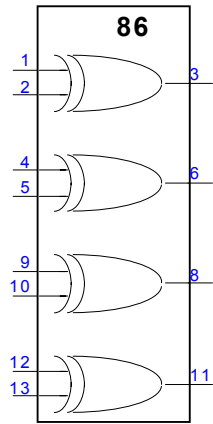


Figure. 1 XOR chip74-86

Decimal	Gray	Binary
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

2. BCD-to-seven Segment converter:

A light emitting Diode (LED) is a PN junction diode. When the diode is forward biased, a current flows through the junction and the light is emitted. See Fig.2.

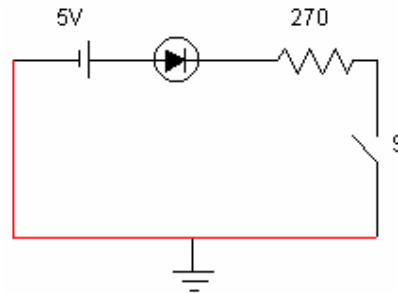


Figure.2

A seven segment LED display contains 7 LEDs. Each LED is called a segment and they are identified as (a, b, c, d, e, f, g) segments. Figure 3.

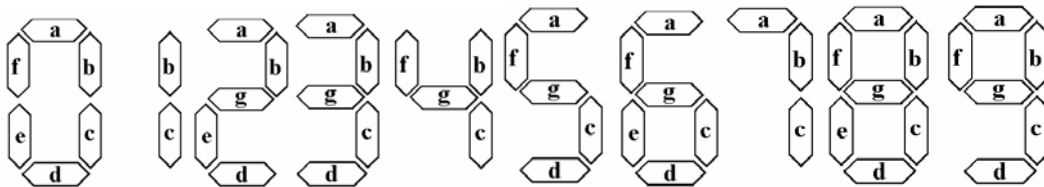


Figure 3. Digits represented by the 7 segments

The display has 7 inputs each connected to an LED segment. All anodes of LEDs are tied together and joined to 5 volts (this type is called common anode type). A limiting resistance network must be used at the inputs to protect the 7-segment from overloading.

BCD inputs are converted into 7 segment inputs (a, b, c, d, e, f, g) by using a decoder, as shown in Fig.4.

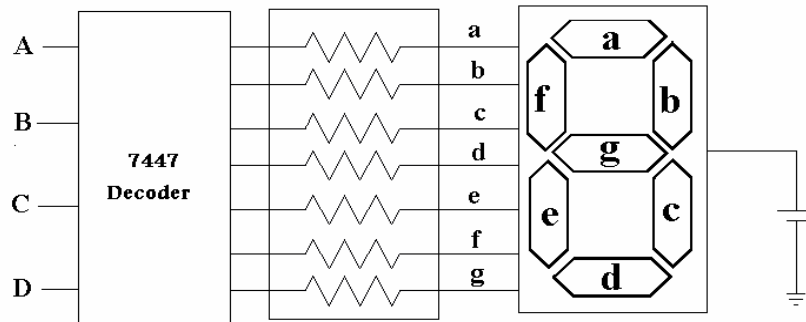


Figure. 4

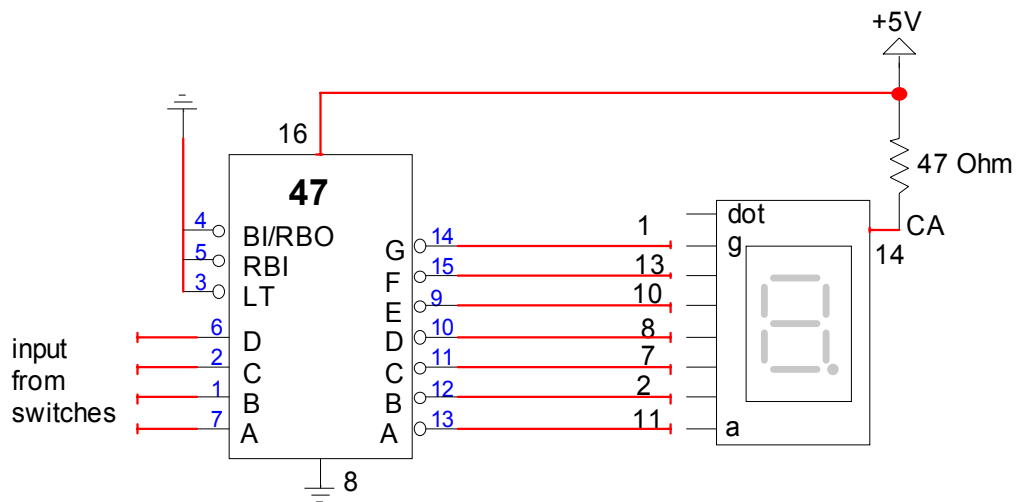
A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n output lines. The input to the decoder is a BCD code and the outputs of the systems are the seven segments a, b, c, d, e, f, and g. For further information and pin connections, consult the specification sheet for decoder and 7-segment units.

First design a combinational circuit which would simulate the decoder function for only the segment “a”, of the display. This can be done in the following steps:

- Write down the truth table with 4 inputs and 7 outputs (Table 2)
- For only the output “a”, obtain a minimum logic function. Realize this function using NAND gates and inverters only. For example if decimal 9 is to be displayed a, b, c, d, f, g must be 0 and the others must be 1 (For common anode type display units), if decimal 5 is to be displayed then a, f, g, c, d must be 0 and the others must be 1.
- Connect the output “a” of your circuit to appropriate input of 7-segment display unit. By applying BCD codes verify the displayed decimal digits for that segment for “a” of the display.
- Replace your circuit by a decoder IC 7447 for all of the seven segments. Observe the display and record the segments that will light up for invalid inputs sequence.
- Comment on the design if you don’t want to see any digit for invalid input sequence.

Table 2

Dec.	BCD				Outputs						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0							
1	0	0	0	1							
2	0	0	0	0							
3	0	0	1	1							
4	0	1	0	0							
5	0	1	0	1							
6	0	1	1	0							
7	0	1	1	1							
8	1	0	0	0							
9	1	0	0	1							



BCD-to-Seven Segment Decoder and 7-segment display

Note: In an actual 7-segment display the dot is on the left

ADDERS, SUBTRACTORS AND MAGNITUDE COMPARATORS

Objectives:

- To construct and test various adders and subtractor circuits.
- To construct and test a magnitude comparator circuit.

Apparatus:

- IC type 7486 quad 2-input XOR gates
- IC type 7408 quad 2-input AND gates
- IC type 7404 HEX inverter
- IC type 7483 4-bit binary adder
- IC type 7485 4-bit magnitude comparator.

Theory:

See Sections 1-5,4-3,5-2,5-4 of your textbook.

a) *Addition:*

IC type 7483 is a 4-bit binary adder with fast carry. The pin assignment is shown in Fig 1. The two 4-bit input binary numbers are A_1 through A_4 and B_1 through B_4 . The 4-bit sum is obtained from S_1 through S_4 . C_i is the input carry and C_o the out carry. This IC can be used as an adder-subtractor as a magnitude comparator.

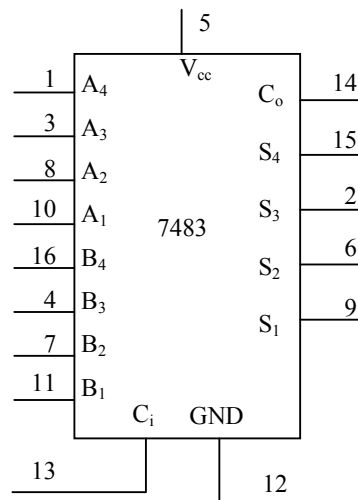
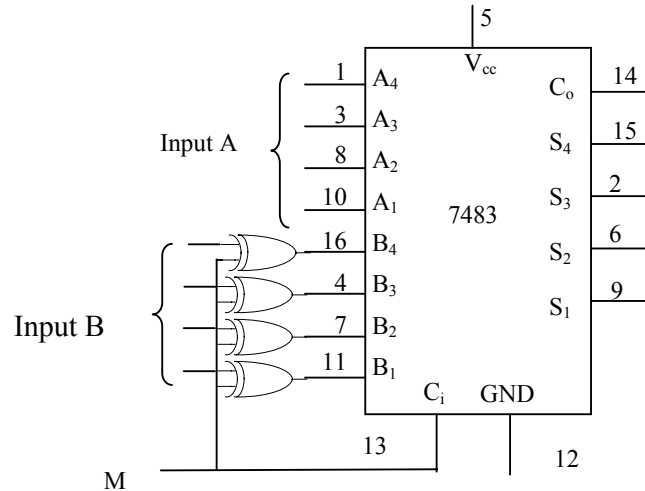


Fig.1 IC type 7483 4-bit adder

b) *Subtraction:*

The subtraction of two binary numbers can be done by taking the 2's complement of the subtrahend and adding it to the minuend. The 2's complement can be obtained by taking the 1's complement and adding 1.

To perform $A - B$, we complement the four bits of B , add them to the four bits of A , and add 1 to the input carry. This is done as shown in Fig 2.



$M = 0$ for add and $M = 1$ for subtract

Fig. 2 4-bit adder/subtractor

Four XOR gates complement the bits of B when the mode select $M = 1$ (because $x \oplus 1 = x'$) and leave the bits of B unchanged when $M = 0$ (because $x \oplus 0 = x$) thus, when the mode select M is equal to 1, the input carry C_i is equal to 1 and the sum output is A plus the 2's complement of B . When M is equal to 0, the input carry is equal to 0 and the sum generates $A + B$.

c) *Magnitude comparison*

The comparison of two numbers is an operation that determines whether one number is greater than, equal to, or less than the other number.

The IC 7485 is a 4 bit magnitude comparator. It compares two 4-Bit binary numbers (labeled as A&B) generates an output of 1 at one of three outputs labeled $A > B$, $A < B$, $A = B$. Three inputs are available for cascading comparators. see Fig.3.

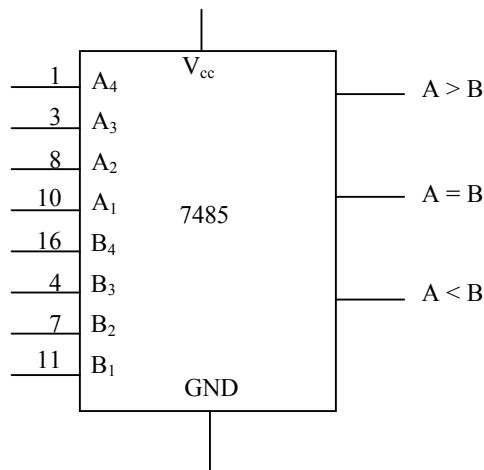


Fig. 3 4-bit magnitude comparator

Procedure:

- Design using LogicWorks a half adder circuit using only XOR gates and NAND gates. Then during the Lab construct the circuit and verify its operation.
- Design using LogicWorks a full adder circuit using only XOR gates and NAND gates. Then during the Lab construct the circuit and verify its operation.
- Use IC 7483 to add the two 4-bit numbers A and B shown in Table1. In LogicWorks, select the chip 74-83 and use Binary switches for the bits of the two numbers and the input carry and use Binary Probe for the sum and carry out.

Table 1.

A3	A2	A1	A0	B3	B2	B1	B0	Sum				Carry out
1	0	0	1	0	0	1	0					
0	1	1	0	1	0	1	1					
1	1	0	0	1	0	1	0					

Input carry C_i is taken as logic 0. Show that if the input carry is 1, it adds 1 to the output sum.

In the Lab use switches S1-1 to S1-8 for the two numbers and use the SPDT S2 for the input carry C_i . For sum and carry out, use LED-1 to LED-5.

- d) Connect the adder-subtractor circuit as shown in Fig 2. Perform the following operations and record the values of the output sum and the output carry C_o .

Table 2.

Decimal A B	Output sum				Carry Out C_o
9 + 5					
9 - 5					
9 + 13					
9 - 9					
10 + 6					
6 - 10					

- Show that $C_o = 1$ when sum exceeds 15.
 - Comment on sum and C_o for the subtraction operations when $A > B$ and $A < B$.
- e) Use IC7485 to compare the following two 4 bit numbers A and B. Record the outputs in table 3. Note that in LogicWorks you need to connect $(A = B)$ input to logic 1 (as an indication that previous stages are equal in multi-digit numbers) for correct results while this is not necessary for the hardware.

Table 3.

A	B	Outputs
1001	0110	
1100	1110	
0011	0101	
0101	0101	

- f) A magnitude comparator can be constructed by using a subtractor as in Fig 2. and an additional combinational circuit. This is done with a combinational circuit which has 5 inputs S_1, S_2, S_3, S_4 , and C_o , and three outputs X, Y, Z see Fig.4

$X = 1$ if $A = B$ Where $S = 0000$

$Y = 1$ if $A < B$ Where $C_o = 0$

$Z = 1$ if $A > B$ Where $C_o = 1$ $S \neq 0000$

Design and construct this logic circuit with minimum number of gates. Check the comparator action using Part (e). In the Lab verify your LogicWorks simulation.

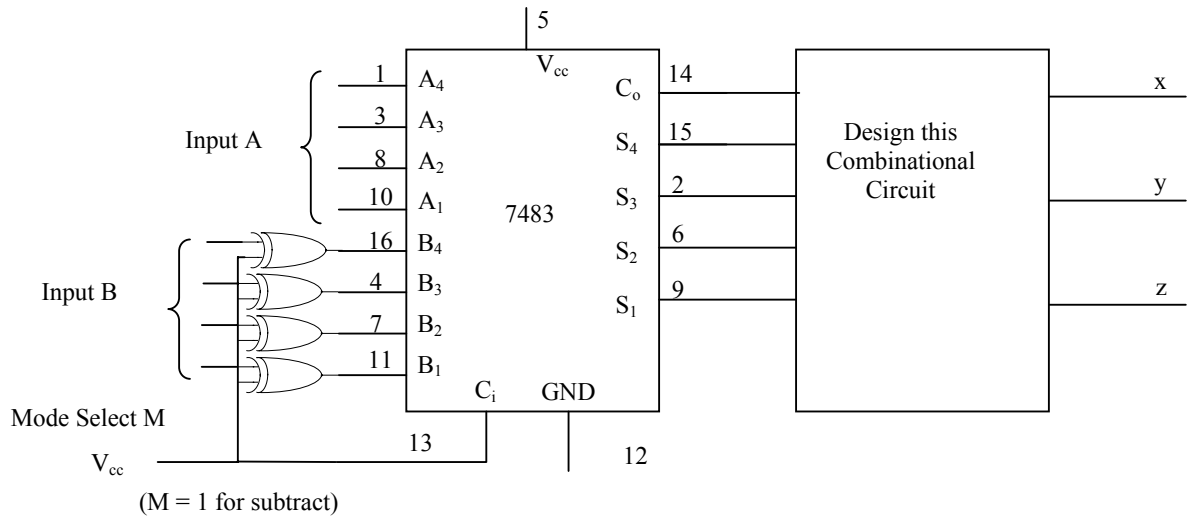


Fig.4 A magnitude comparator using a subtractor

DESIGN WITH MULTIPLEXERS

Objectives:

To design a combinational circuit and implement it with multiplexers. To use a demultiplexer to implement a multiple output combinational circuit from the same input variables.

Apparatus:

- IC type 7404 HEX inverter
- IC type 7408 quad 2-input AND gate
- IC type 74151 8x1 multiplexer (1)
- IC type 74153 dual 4x1 multiplexer (2)
- IC type 7446 BCD-to-Seven-Segment decoder (1)
- Resistance network (1)
- Seven-Segment Display (1)

Theory: see section 5.6 of your text.

IC Description:

74151 is a 8 line-to-1 line multiplexer. It has the schematic representation shown in Fig 1. Selection lines S_2 , S_1 and S_0 select the particular input to be multiplexed and applied to the output.

Strobe S acts as an enable signal. If strobe =1, the chip 74151 is disabled and output $y = 0$. If strobe = 0 then the chip 74151 is enabled and functions as a multiplexer. Table 1 shows the multiplex function of 74151 in terms of select lines.

Table 1.

Strobe	Select Lines			Output
S	S_2	S_1	S_0	Y
1	X	X	X	0
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7

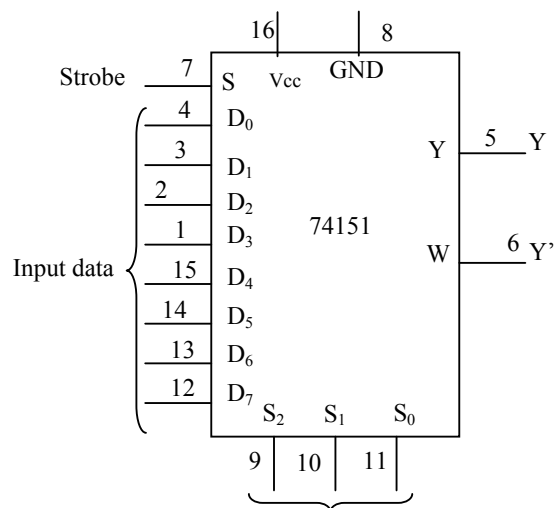


Fig.1 IC type 74151 Multiplexer 8x1

74153 is a dual 4 line-to-1 line multiplexer. It has the schematic representation shown in Fig 2. Selection lines S_1 and S_0 select the particular input to be multiplexed and applied to the output $1Y \{1 = 1, 2\}$.

Each of the strobe signals $1\overline{G} \{I = 1, 2\}$ acts as an enable signal for the corresponding multiplexer.

Table 2. shows the multiplex function of 74153 in terms of select lines. Note that each of the on-chip multiplexers act independently from the other, while sharing the same select lines S_1 and S_0 .

Table 2

Multiplexer 1			
Strobe	Select lines		Output
$1\overline{G}$	S_1	S_0	$1Y$
1	X	X	0
0	0	0	$1D_0$
0	0	1	$1D_1$
0	1	0	$1D_2$
0	1	1	$1D_3$

Multiplexer 2			
Strobe	Select lines		Output
$2\overline{G}$	S_1	S_0	$2Y$
1	X	X	0
0	0	0	$2D_0$
0	0	1	$2D_1$
0	1	0	$2D_2$
0	1	1	$2D_3$

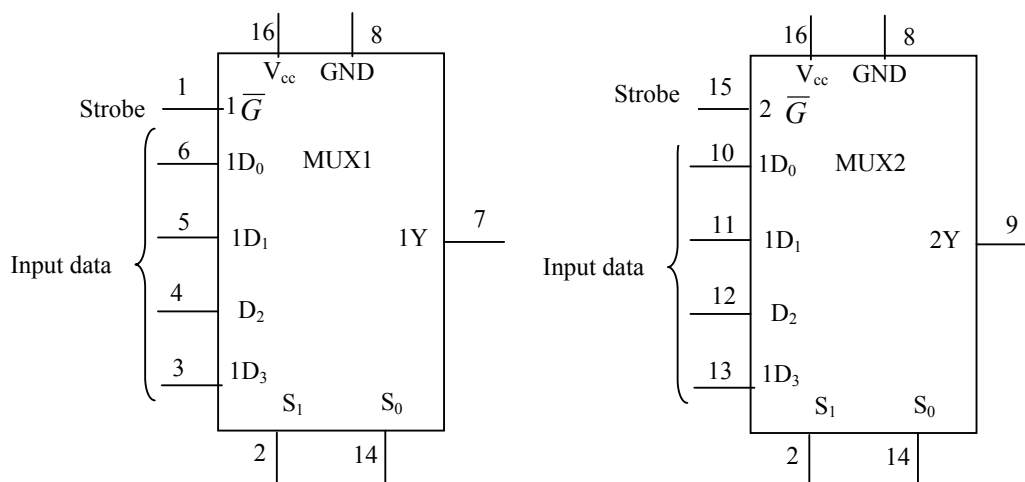


Fig.2 Chip 74153

IC 7446 is a BCD to seven segment decoder driver. It is used to convert the combinational circuit outputs in BCD forms into 7 segment digits for the 7 segment LED display units. See experiment #5.

Procedure:

Part I: Parity Generator:

- a) Design a parity generator by using a 74151 multiplexer. Parity is an extra bit attached to a code to check that the code has been received correctly.

Odd parity bit means that the number of 1's in the code including the parity bit is an odd number. Fill the output column of the truth table in Table 2 for a 5-bit code in which four of the bits (A,B,C,D) represents the information to be sent and fifth bit (x), represents the parity bit. The required parity is an odd parity.

The inputs B,C and D correspond to the select inputs of 74151. Complete the truth table in Table 3 by filling in the last column with 0,1,A or A'.

- b) Simulate the circuit using LogicWorks, use 74-151 multiplexer and Binary

Inputs				Outputs	Connect data to
A	B	C	D	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		

switches for inputs and Binary Probes for outputs. The 74151 has one output for Y and another inverted output W. Use A and A' for providing values for inputs 0-7. The internal values "A, B, C" are used for selection inputs B,C, and D. Simulate the circuit and test each input combination filling in the table shown below. In the Lab connect the circuit and verify the operations. Connect an LED to the multiplexer output so that it represents the parity bit which lights any time when the four bits input have even parity.

Part 2: Vote Counter:

A committee is composed of a chairman (C), a senior member (S), and a member (M). The rules of the committee state that:

- The vote of the member (M) will be counted as 2 votes
- The vote of the senior member will be counted as 3 votes.
- The vote of the chairman will be counted as 5 votes.

Each of these persons has a switch to close ("1") when voting yes and to open ("0") when voting no.

It is necessary to design a circuit that displays the total number of votes for each issue. Use a seven segment display and a decoder to display the required number.

If all members vote no for an issue the display should be blank. (Recall from Experiment #5, that a binary input 15 into the 7446 blanks all seven segments).

If all members vote yes for an issue, the display should be 0. Otherwise the display shows a decimal number equal to the number of 'yes' votes. Use two 74153 units, which include four multiplexers to design the combinational circuit that converts the inputs from the members' switch to the BCD digit for the 7446.

In LogicWorks use +5V for Logic 1 and ground for Logic 0 and use switches for C, S, and M. Use two chips 74153 and one decoder 7446 verify your design and get a copy of your circuit with the pin numbers to Lab so that you could connect the hardware in exactly the same way.

FLIP-FLOPS

Objectives:

1. To become familiar with flip-flops.
2. To implement and observe the operation of different flip-flops.

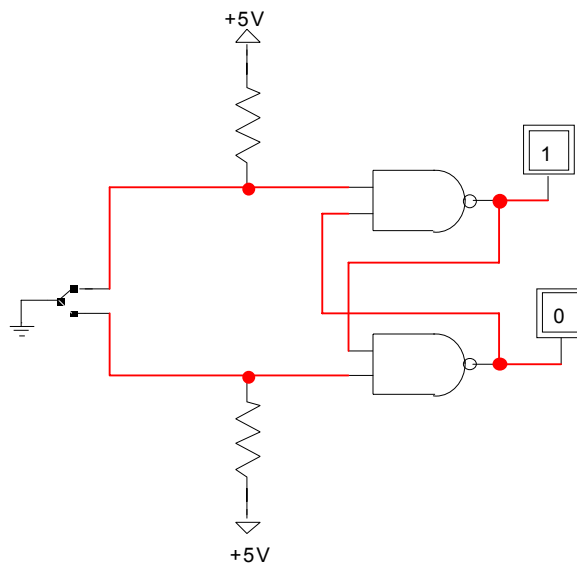
Apparatus:

- IC type 7400 quad 2-input NAND gate
- IC type 7410 triple 3- input NAND gate
- IC type 7476 dual JK master-slave flip-flops.
- IC type 7474 dual D positive-edge-triggered flip-flops.
- Dual trace oscilloscope.

Theory: See sections 6-2 and 6-3 of your text.

Procedure:

1. In the pre-lab using LogicWorks construct the circuit shown in Fig.1



Where we could use generic NAND gates or 74-00 and Binary Probes to simulate LEDs. Finally, we use SPDT for the bouncing switch. Using the simulated circuit fill in the truth table.

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

In the Lab, Build the RS latch shown in fig.2. Use SPDT switch S2 as a bouncing switch. Q and Q' Outputs are connected to LED'S of the PB-503. Verify the truth table experimentally.

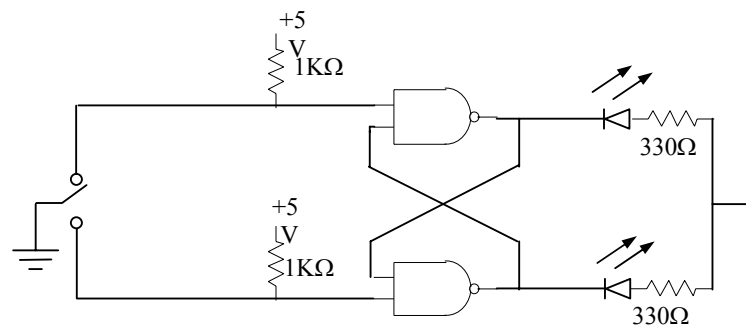


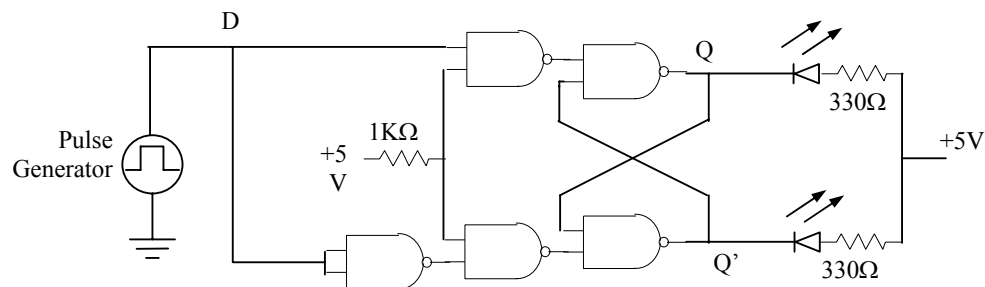
Fig. 2

2. Modify the basic R-S into a D latch by adding the steering gates and the inverter shown in Fig 3.

Connect the D input to the pulse generator of the digi designer and set it at 1 Hz.

Connect the enable input to a high through 1k resistor. Observe the output; obtain the truth table experimentally then change the enable to a low.





Is the enable an active high or an active? Leave the enable low and place a momentary short to ground first on one output and then on the other. What happens?



3. The 7476 is a dual JK master-slave flip-flops with preset and clear inputs. The function table given in table 1 defines the operation of the flip-flop. The +ve transition of the CLOCK (CP) pulse changes the master flip-flop, and the (-ve)

transition changes the slave flip-flop as well as the output of the circuit. In LogicWorks the chip 7476 is not available, however, the generic JK flip-flop behave in exactly the same way as the 7476. The “S” represents the Preset, the “R” represents the Clear, and C represents the clock pulse (CP). Verify the table by connecting Binary switches to R, S, J, K, and C. Notice that only the negative edge of the clock affects the outputs (Q, and Q’).

Table 1

Input					Output	
Preset	Clear	Clock	J	K	Q	Q'
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1	1
1	1		0	0	No change	
1	1		0	1	0	1
1	1		1	0	1	0
1	1		1	1	Toggle	

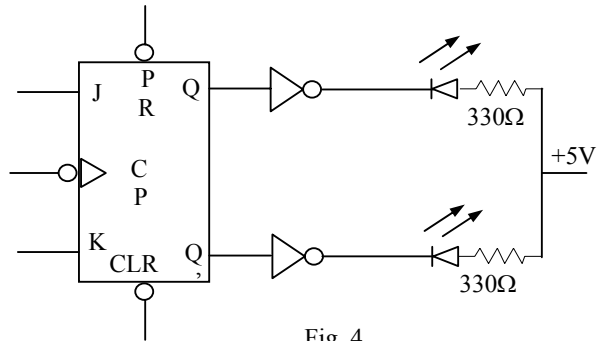


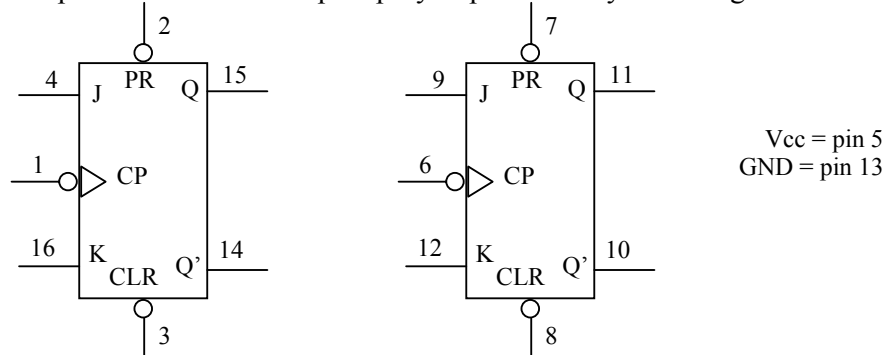
Fig. 4

In the Lab, Construct the circuit of Fig 4. Look at the data sheet for the 7476 and determine the inactive logic required at the PRE and CLR inputs.

Connect the 7476 for the SET mode by connecting J = 1, K = 0. With CLOCK (CP) = 0; test the effect of PRE, CLR by putting a 0 on each, one at a time.

Put CLR = 0, then pulse the clock (CP) by putting a HIGH then a LOW, on the clock. Does the CLR input override J input?

Verify the operation of the JK flip flop by experimentally obtaining the characteristic table.



CLOCKED SEQUENTIAL CIRCUITS AND COUNTERS

OBJECTIVE:

- To design, build and test synchronous sequential circuits.
- To design, build, and test synchronous counters
- To design, build and test asynchronous counters

APPARATUS:

- IC type 7476 dual JK master-slave flip-flops
- IC type 7400 quad 2-input NAND gates

THEORY:

See sections 6-6, 6-7, 6-8, 7.2 and 7.5 of your own text.

PROCEDURE:

1. *SYNCHRONOUS SEQUENTIAL CIRCUITS:*

- a) Design, construct and test a sequential circuit whose state is shown in Fig.1. Use JK flip-flops in the design.

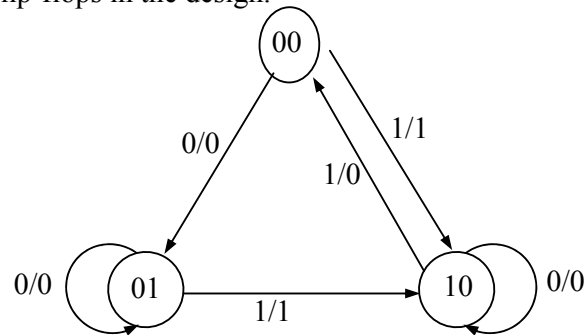


Fig. 1

The circuit has two flip-flops A, B, one input x and one output y. The circuit is to be designed by treating the unused states as don't care conditions. The final circuit must be analysed to ensure that it is self-correcting. If not suggest a solution.

- b) Complete the excitation table shown in Table 1.

Table 1.

Present state		Input	Next state		Output	Flip-flop input functions			
A	B	X	A	B	Y	JA	KA	JB	KB
0	0	0	0	1	0	0	X	1	X
0	0	1	1	0	1	1	X	0	X
0	1	0							
0	1	1							
1	0	0							
1	0	1							
1	1	0							
1	1	1							

- c) Using Karnaugh maps obtain minimal expressions for the flip-flop input functions JA, ..., KC
- d) Simulate the circuit using LogicWorks. LogicWorks does not have the JK master-slave flip-flop IC 7476. Use instead the generic JK flip-flop as you did in experiment 9. In the Lab, build the circuit and check the output to verify the state table values.

2. Synchronous Counters

Synchronous counters have all clock lines tied to a common clock causing all flip-flops to change at the same time. The count sequence of a counter can be analysed by placing the counter into every possible number in the sequence and determining the next number in the sequence state diagram is developed as the analysis proceeds. (A state diagram is an illustration of the transitions that occur after each clock pulse).

- a) In the pre-lab using LogicWorks and then in the lab using hardware chips, design a 2-bit gray code counter using JK flip-flops. The required sequence is the binary equivalent of (0-1-3-2-0). A state diagram for this counter is given in Fig. 2.

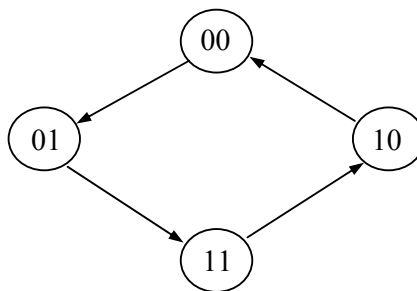


Fig. 2

- b) Complete the excitation table (Table 2) for the counter and obtain logic expression for the JK flip-flop input functions.

Table 2.

Present state		Next state		Flip-flop input functions			
A	B	A	B	JA	KA	JB	KB
0	0						
0	1						
1	1						
1	0						

Flip-flop input functions are:

JA=

KA=

JB=

KB=

- c) In the lab, build the circuit and test it by pulsing it from the PB-503. Check that the output is the designed sequence

3. A Synchronous Counters

Asynchronous counters are a series of flip-flops each clocked by the previous state, one after the other. Since all the stages of the counter are not clocked together, a ripple effect propagates as various flip-flops are clocked. For this reason they are called ripple counters. The modulus of a counter is the number of different output states the counter may take (i.e. Mod 4 means the counter has four output states).

- a) In the pre-lab construct a 4-bit asynchronous counter shown in Fig.3. (It is also called binary ripple counter). Use four generic JK flip-flops. Connect four Binary Probes to Q outputs. Connect all R and S inputs to Logic 1 and connect a switch to the CP input.

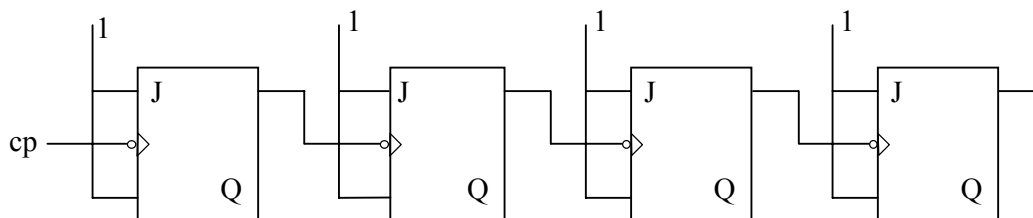


Fig. 3 4-bit ripple counter

- b) In the Lab use two 7476 ICs to implement the design. Connect Q outputs of flip-flops to indicator lamps of the PB-503. Connect all clear (CLR) and preset (PRE) inputs to logic 1. Connect the CP input to the pulse output of the PB-503 and check the counter for proper operation.

- c) Write down the count sequence in Table 3. Identify this count sequence (up or down). Comment on what happens after the application of 15 pulses to CP input.

Table 3. Count sequence for the 4-bit ripple counter.

A	B	C	D