



Princess Sumaya University for Technology
King Abdullah II School for Electrical Engineering
Computer Engineering Department

Logic and Computer Organization Lab
(EE 4349)

Experiments Manual

Edited by: Eng. Hussam Kharouf

Spring 2007/2008

Lab. Experiments:

Introduction: A-Understanding IC's.....	2
B- Wiring IC's.....	5
Experiment 1 Basic Logic Gates	10
Experiment 2 Adders and Subtractors	15
Experiment 3 Seven Segment Display Decoder	17
Experiment 4 Decoder and Multiplexer.....	19
Experiment 5 Sequential Circuits	21
Experiment 6 Registers.....	23
Experiment 7 RAM /ROM	25
Experiment 8	28
Appendix 1: Datasheets.....	29

Grading System:

Reports:	50%
Performance:	10%
Prelabs & Quizzes:	20%
Mid term exam:	20%

Introduction

A-Understanding IC's

Logic Gates and Integrated Circuits

Digital circuits are hardware components that are implemented using transistors and interconnections in complex semiconductor devices called *integrated-circuits*. Digital circuits work in binary logic domain which uses two discrete values, **TRUE (High)** and **FALSE (Low)**. We can also refer to these values as **1(High)** and **0 (Low)**.

Logic-Gates are basic building blocks of digital circuits. Using these building blocks, complex functions or larger digital circuits can be built. Examples of the basic logic gates are **AND, OR, NOT, NAND and NOR**. A complex gate such as an **XOR** gate can be built out of these basic gates.

Each logic gate is actually implemented in part of an integrated circuit (IC), with each gate made using several transistors. For the most part we do not need to concern ourselves with the actual circuitry inside each gate, only the interconnections between individual gates. Usually each IC package contains several individual gates. The 7408 chip implements 4 two input AND gates and is commonly referred to as a “Quad two input AND” chip. Similarly the 7432 chip is a “Quad 2 input OR” chip while the 7404 chip is a “Hex Inverter” since it contains 6 inverters. The IEEE standard logic symbols for each of these devices are shown in Figure 1.

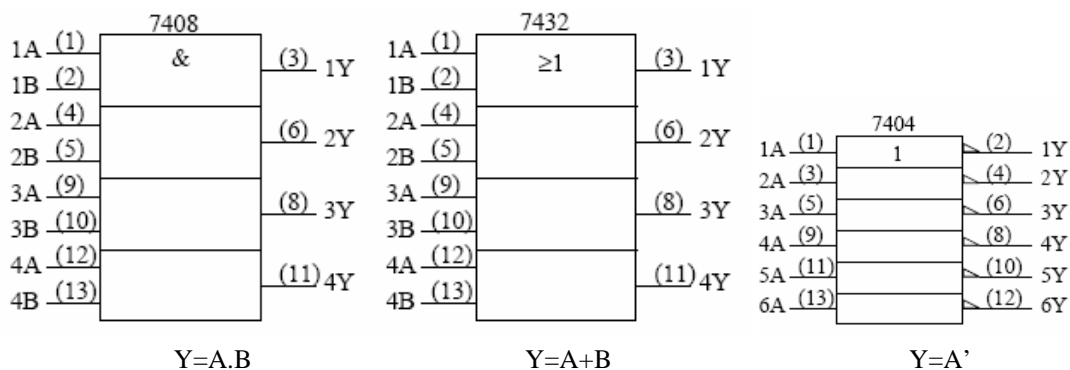


Figure 1: Standard Logic Symbols

Each of the chips in Figure 1 is available in 14 pin Dual-In-Line packages or DIPs. In a popular logic family called TTL (Transistor-Transistor Logic), the low logic level is assigned to 0V and the high logic level is assigned to 5V. Each IC or chip has an ID number that can be referenced in IC Data Book. From the book, you can get the pin configuration of that chip. The pin numbers assigned to each logic signal are shown inside brackets in the figure. The pins are numbered as shown in Figure 2. Pin 1 is usually identified as the pin to the left of an indentation or cutout in one end of the chip

that is visible when the chip is viewed from the top. Occasionally, it is also identified by a printed or indented dot placed just next to it.

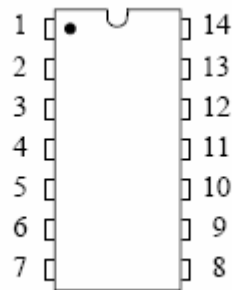


Figure 2: Identifying Pin 1

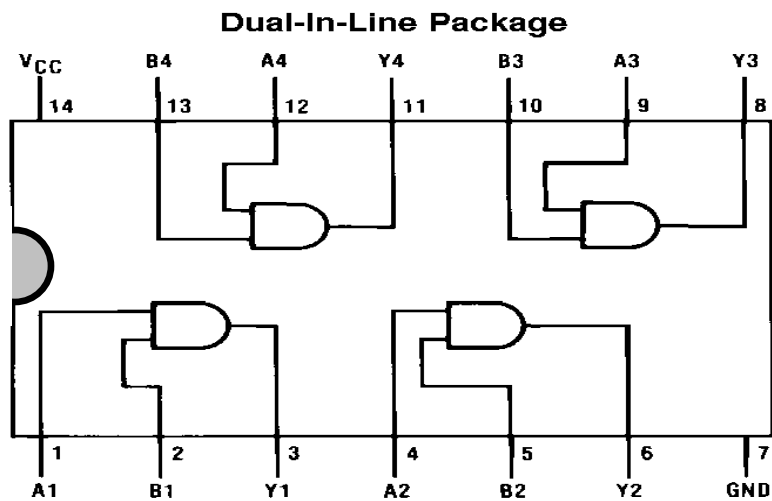


Figure 3: A Quad 2-input AND gate chip

Figure 3, shows an IC which contains four 2-input AND gates. You will notice that pins 7 and 14 show no connections. In 14 pin DIP packages, pin 7 is usually connected to ground (Gnd), and pin 14 is usually connected to the power supply (Vdd). These connections must be made or the chip will not work. Take care not to connect pin 7 to power and pin 14 to ground, or to connect the outputs of two or more gates together. In complex ICs more than one pin can be dedicated for power (VDD) as well as ground. In the simpler gates that we will be using in this experiment, the ICs require only one pin for power (VDD) and another for ground (GND). The power supply (VDD) voltage is typically +5Volts, 3.3 Volts or 2.5 Volts. The ground is typically connected to 0 Volts.

The Breadboard

The circuit is constructed on the breadboard section of the prototyping board. A breadboard is used to rapidly create an experimental or prototype circuit. It consists of an array of holes in which wires or component leads can easily be inserted. Rows of five holes are electronically connected to form a single node, also each 25 horizontal holes forms one node as shown in Figure 4. When a component lead is inserted into one of the holes, anything inserted into any of the remaining four holes will be connected to that lead. Nodes can be connected to each other using wires with 1/4" of insulation stripped from both ends. The holes are spaced 100 mms(.1 inch) apart, which is the standard spacing of the pins on a DIP package. The breadboard has a groove down the center separating one side from the other. When inserting a chip into the breadboard, make sure it fits into the central groove as shown in figure 5; otherwise the pins on opposite sides of the chip will be connected. Press the chip down until it touches the surface of the breadboard.

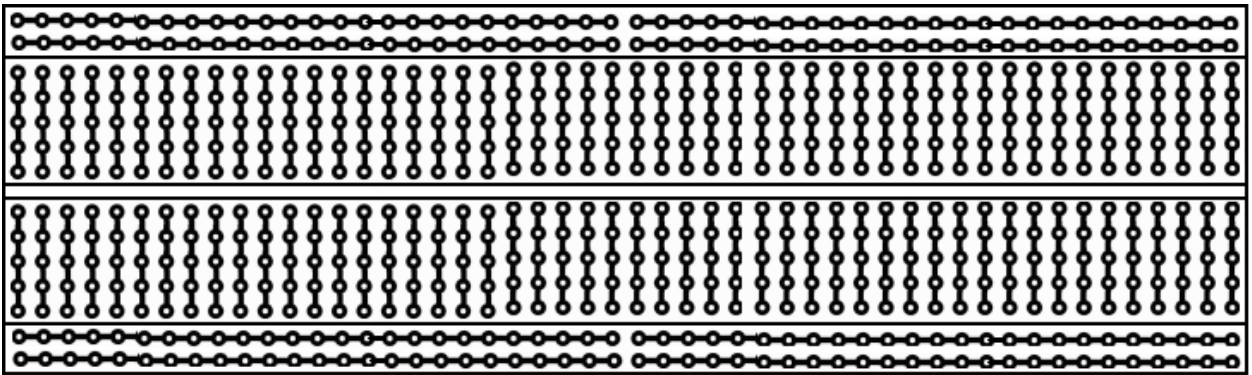


Figure 4: Breadboard hole pattern

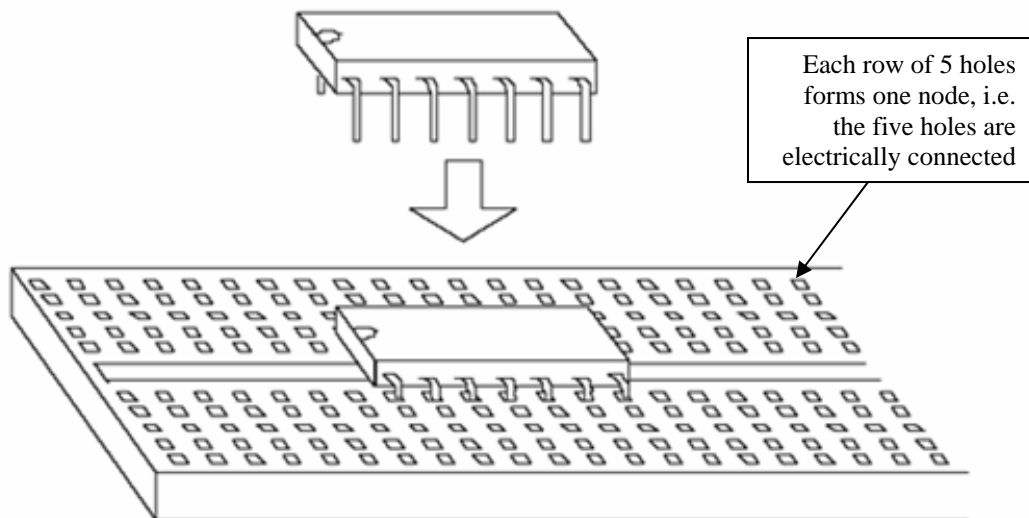


Figure 5: Placing DIP devices on a breadboard

Introduction B-Wiring IC's

In this section we'll walk you through wiring a simple gate circuit using one specific integrated circuit (IC) the 7400 chip. It's a good introduction to some of the more complex logic chips that you'll probably be using later.

Wiring A 7400 Chip

Here's a picture of the 7400 chip in a circuit board. This chip is actually an M74LS00P. The LS tells you that it is a low power Schottky chip. Every manufacturer will embed the 7400 or 74LS00 in other part numbers. Notice that this chip has fourteen pins.

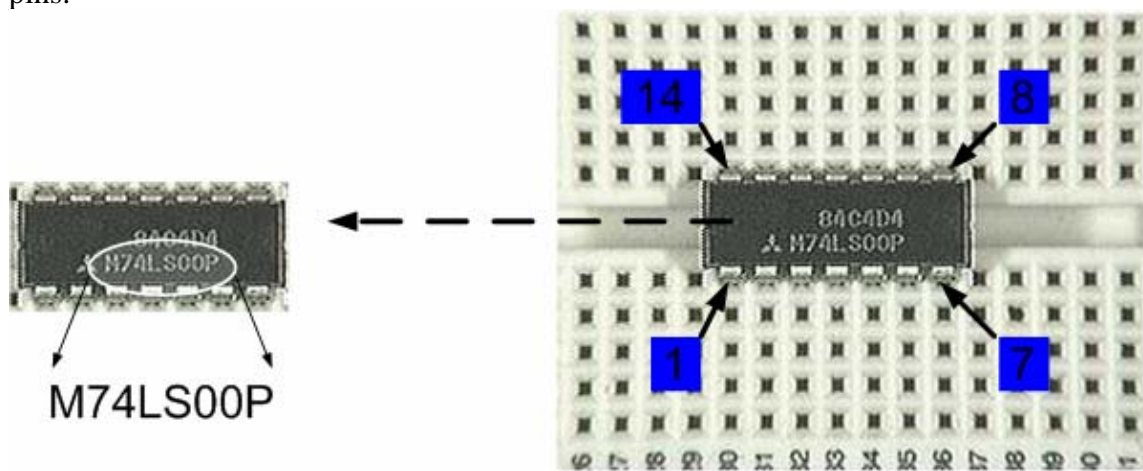


Figure 1: Chip number and pins numbers

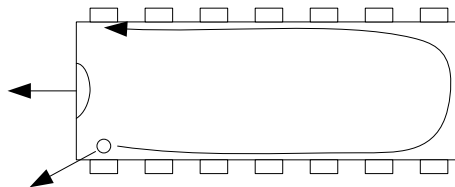


Figure 2: IC pins Numbering

Before starting wiring any IC, you should know the position of every pin. To start counting the pins as shown above, there is a little notch on one side of the Chip to mark pin number "1". That notch could be a half circle on the edge of the chip, or a dot just over pin number one. The pins are numbered counter clock wise starting from pin number "1".

If you want to use an IC chip, then you will always need to know the pinout. That's electrical engineering lingo for describing the way the pins are connected to the internal

circuitry of the chip. You need to know where the power supply is connected and where the gate inputs and outputs are connected. Here's the pinout for a 7400 chip.

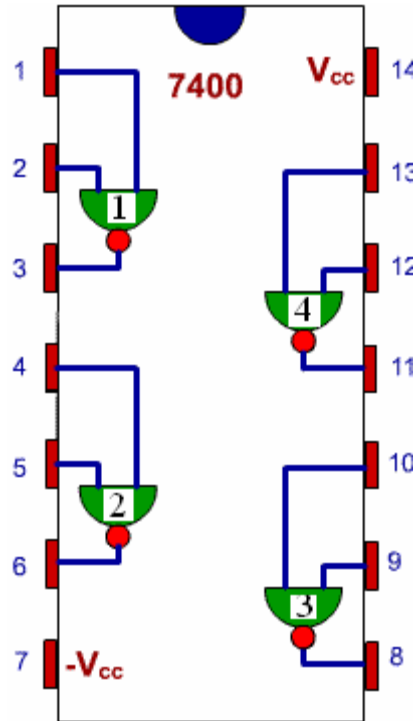


Figure 3: IC internal circuitry and pinout

Using this pinout we can construct any circuit constructed from NAND gates. As any electrical and Electronic system, nothing will work unless it is supplied with necessary voltage and current.

The first step in wiring the 7400 is to connect the positive power supply. Use a five volt (5v) power supply and don't turn it on yet. Connect a lead to pin 14 as shown below, and connect the other end of that lead to a 5v supply. **Keep the power supply turned off until you have everything connected.** Here's what that looks like when the positive supply voltage to the chip is wired.

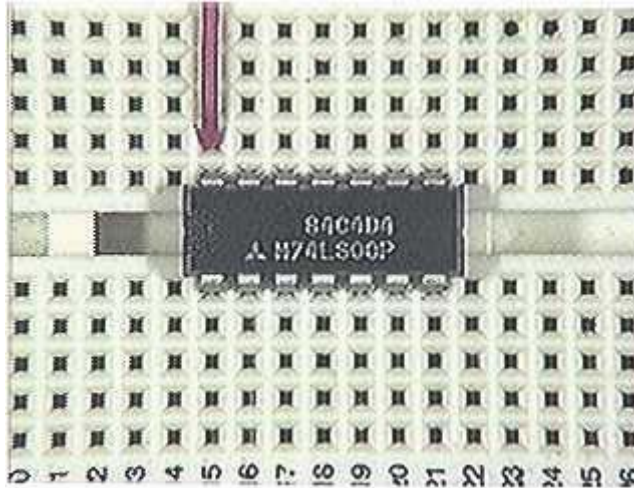


Figure 4: Connecting power to the chip

The next step in wiring the 7400 is to connect the ground connection. Connect a lead to pin 7 as shown below, and connect the other end of that lead to ground.

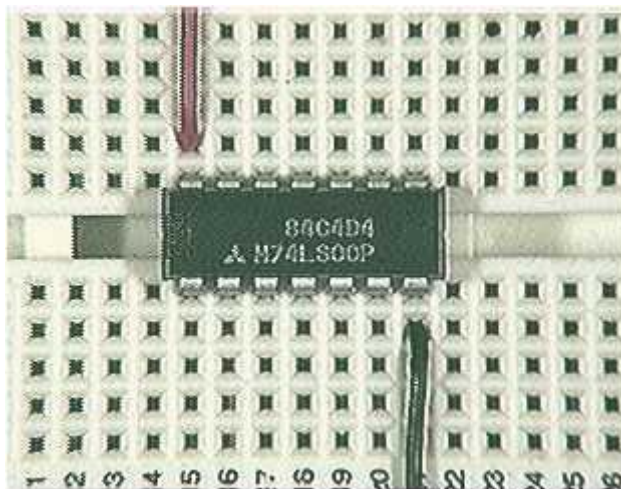


Figure 5: Connecting ground to the chip

Notice the pattern to this connection. The power to this digital logic chip goes to the corners. Remember, **power to the corners** for logic chips.

To move on, consider the following circuit, we need three NAND gates to construct this circuit, we will need one 7400 chip (each 7400 chip contains 4 NAND gates).

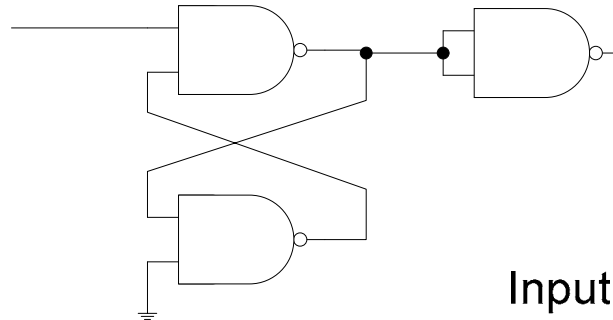


Figure 6: Simple logic circuit

To start connecting this circuit, we may use NAND gates 1,2 and 3. the NAND gate number 1 has the following pinout numbers, 1 and 2 as inputs and 3 as output. NAND gate number 2 has the following pinout numbers, 4 and 5 as inputs and 6 as output. Finally the last gate pinout numbers are 10 and 9 as input and 8 as output. Redraw the above circuit showing the gates numbers and the pinout numbers as shown below:

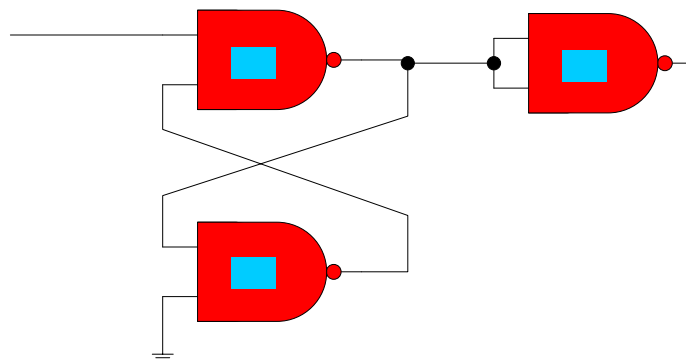


Figure 7: numbering the circuit

After numbering the circuit and locating the gates to be used and indicate the pinout of each gate, we can proceed to connect this circuit.

Input

1

2⁸ N 1

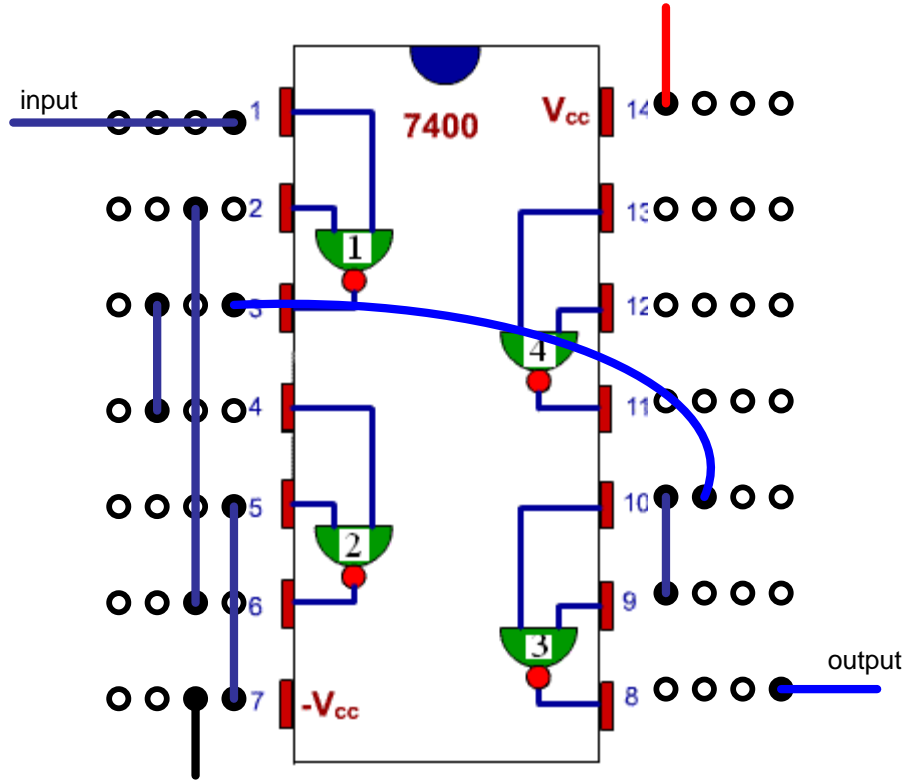


Figure 8: Wiring the circuit on the breadboard

The input to pin number 1 maybe a switch and the output from pin 8 may go to LED to see the output by your own eyes.

Exercise

Do the wiring for the following circuit

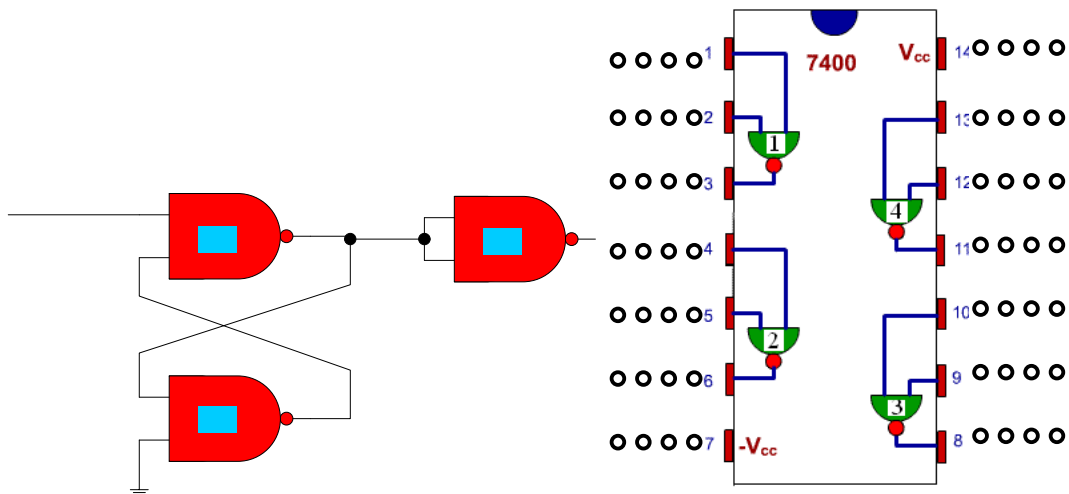


Figure 9: Exercising circuit

Experiment 1

Basic Logic Gates

Introduction to Logic Gates:

There are many things in this world that most electrical and computer engineer can do without. Unfortunately logic gates are not one of them. Good working knowledge of logic gates is essential as they are used in almost all electrical circuits in some form or the other.

This lab has the following objectives.

- Understand how to use the breadboard to patch up, test your logic design and debug it.
- Understand the operation of logic gates such as AND, OR, NOT, NAND, NOR, XOR.
- Understand how to implement a circuit based on a schematic diagram.

Wiring Guidelines:

- Arrange the IC chips on the breadboard so that only short wire connections are needed.
- Try to keep the wire as short as possible to avoid a jungle of wires.
- Try to maintain a low wiring profile so that the pins of the chips can be reached and the chip replaced, if necessary. The best connections are those that lie flat on the board.
- Put the wiring in a numeric order and identify them with different color to make debugging convenient.

Pay extra attention to power and ground. If you find your chips are getting super hot then there is probably a short circuit. Turn off power immediately and wire them correctly.

The logic chips used in this class require a power source and ground. The power source (V_{cc}) is +5V and the ground is 0V, which can be drawn from the Digital Logic Trainer. Normally, pin 14 of the IC chip is connected to +5V and pin 7 to ground.

Objective:

The purpose of this experiment is to introduce you to the equipments and to some of the components that you will be using during the semester.

Components:

- 7404: Hex-Inverter

The Inverter (NOT circuit) performs a basic logic function called inversion or complementation. The purpose of the inverter is to convert one logic level to other logic level. In terms of binary digits, it changes a 1 to a 0 or a 0 to a 1.

- 7408: Quad 2-input AND gate
The logical operation of the AND gate is such that the output is HIGH only when all of the inputs are HIGH.
- 7432: Quad 2-input OR gate
The logical operation of the OR gate is such that a HIGH only on the output is produced when any of the inputs are HIGH.
- 7400: Quad 2-input NAND gate
The NAND gate (NOT-AND) implies an AND function with a complemented output.
- 7402: Quad 2-input NOR gate
The NOR gate (NOT-OR) implies an OR function with a complemented output.
- 7486: Quad 2-input XOR gate
The XOR gate is another important function widely used because of some special arithmetic properties. It is a 2-input gate (A and B) with an output $Y = \overline{A}B + A\overline{B}$.

One Important feature about NAND and NOR gates is that they both can be used as an inverter and a buffer. For an input A, the inverter generates the complement of the input, \overline{A} , and the buffer generates the input as it is, A.

Procedure:

- 1- Insert the 7404 IC on the breadboard socket, connect pin 14 to +5V and pin 7 to GND.
The pin configuration of the 7404 is shown below.

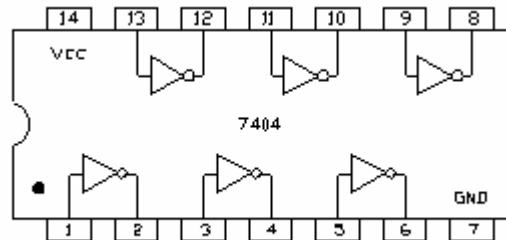


Figure 1: 7404 Pin configurations

- 2- Using any of the gates in the IC, connect the circuit shown in figure 2.

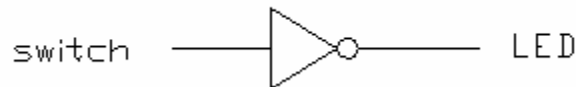


Figure 2

Experimentally verify that this NOT gate is working properly by determining its truth table – Test one gate only.

Connect the output of the first NOT gate to the input of the second NOT gate. Draw the schematic representation of the circuit and determine the truth table.

- 3- Insert the 7408 IC into the bread-boarding socket, connect pin 14 to +5V and pin 7 to GND. Figure 3 shows the pin configuration for this IC.

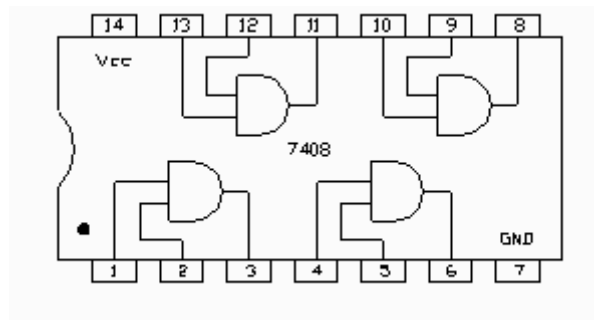


Figure 3: 7408 Pin configurations

Experimentally verify that this AND gate is working properly by determining its truth table. You can do this by connecting the inputs to switches and the output to one of the LED's on the Trainer.

- 4- Insert the 7432 IC into the bread-boarding socket, connect pin 14 to +5V and pin 7 to GND. Figure 4 shows the pin configuration for this IC.

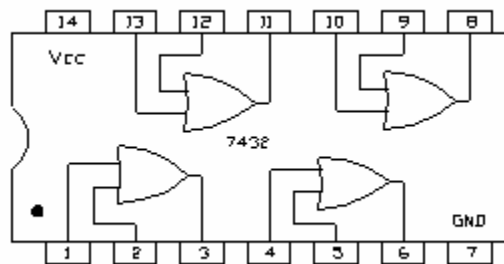


Figure 4: 7432 Pin configurations

Experimentally verify that this OR gate is working properly by determining its truth table. You can do this by connecting the inputs to switches and the output to one of the LED's on the Trainer.

- 5- Insert the 7400 IC into the bread-boarding socket, connect pin 14 to +5V and pin 7 to GND. Figure 5 shows the pin configuration for this IC.

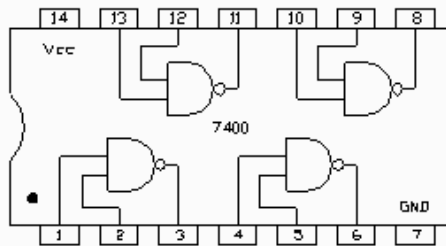


Figure 5: 7400 Pin configurations

Verify the function of a single NAND gate and develop the truth table.

Wire both inputs together and apply only one signal to them. Check the logic at the output and record its truth table. Name the gate you've built.

- 6- Insert the 7402 IC into the bread-boarding socket, connect pin 14 to +5V and pin 7 to GND. Figure 6 shows the pin configuration for this IC.

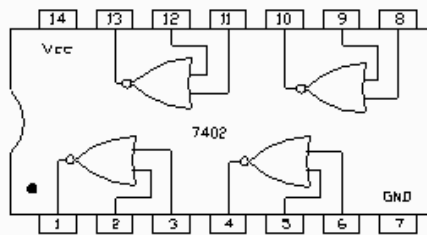


Figure 6: 7402 Pin configuration

Verify the function of a single NOR gate and develop the truth table.

- Can a NOR gate be used as an inverter? As a buffer? How?

- 7- Insert the 7486 IC into the bread-boarding socket, connect pin 14 to +5V and pin 7 to GND. Figure 7 shows the pin configuration for this IC.

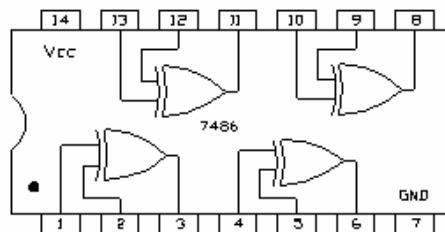


Figure 7: 7486 Pin configuration

Verify the function of a single XOR gate and develop its truth table..

Simple Design:

- 8- Construct the following circuit using the gates within the chips.
All inputs to the circuit are to be connected to the switches and the outputs to the LED's on the trainer.
- Ensure that all IC chips are properly powered and grounded.
 - Ensure that all interconnections are correct.
 - Switch on the power and test the circuit.

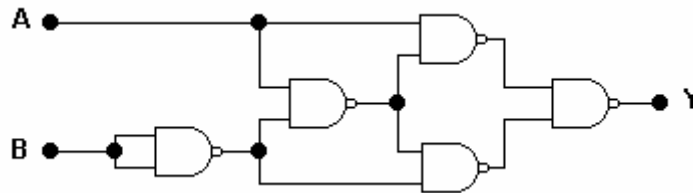


Figure 8

Apply test signals to the input of the circuit and record the values in a truth table.

- Analyze the circuit above and write a Boolean expression for the output as a function of the inputs. Simplify this expression to show the equivalent.

Experiment 2 Adders and Subtractors

Introduction:

The half adder is a dual input, dual output logic circuit that adds two binary bits and gives the sum of these bits and a carry.

The full adder combines two half adders taking the OR of their carries with the sum of the second being used as the output.

The addition and subtraction operations can be combined into one circuit with one common binary adder. This is done by including an exclusive-OR gate with each full adder. A 4-bit adder-subtractor circuit is shown in Figure 1. The mode input M controls the operation. When $M = 0$, the circuit is an adder, and when $M = 1$, the circuit becomes a subtractor (explain?). Each exclusive-OR gate receives input M and one of the inputs of B.

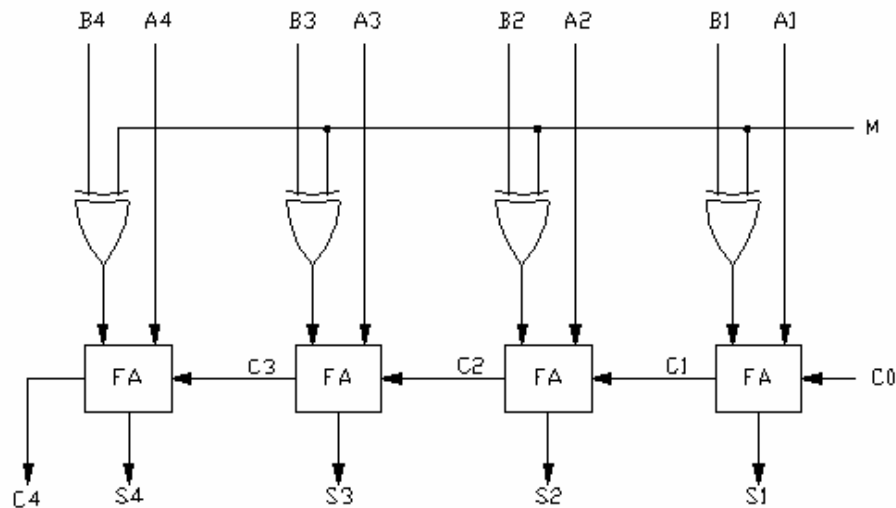


Figure 1: Adder and Subtractor circuit

Objective:

To examine the concept of half and full adders, and to determine the functions of the 4-bit adder-subtractor.

Components:

7400, 7486, 7483.

Prelab:

- 1- Design one bit half-adder using XOR & NAND gates only.
- 2- Design one bit full-adder using XOR & NAND gates only.
- 3- Test your previous designs using EWB software using both logic gates and IC's.

4- Procedure:

1- Design, construct and test a half adder circuit using XOR and NAND gates.

2- Design, construct and test a full adder circuit using XOR and NAND gates.

IC type 7483 is a 4-bit binary parallel adder. The pin assignment is shown in figure 2. The two 4-bit input binary numbers are A1 through A4 and B1 through B4. The 4-bit sum is obtained from S1 through S4. C0 is the input carry and C4 is the output carry.

3- Test the 4-bit binary adder 7483. Connect the power supply and the ground terminals, and then connect the four A inputs to a fixed binary number such as 0110, and the B inputs and the input carry to five toggle switches. The outputs are applied to indicator lamps.

Perform the addition of a few binary numbers and check that the output sum and output carry give the proper values. Show that when the input carry is equal to 1, it adds 1 to the output sum.

4- Figure 2 shows an adder/subtractor circuit depending on the mode select M. When M=0, the circuit will function as an adder, and when M=1, it will function as a subtractor.

- Construct and test the circuit for proper operation.
- Explain how M controls the circuit's operation.

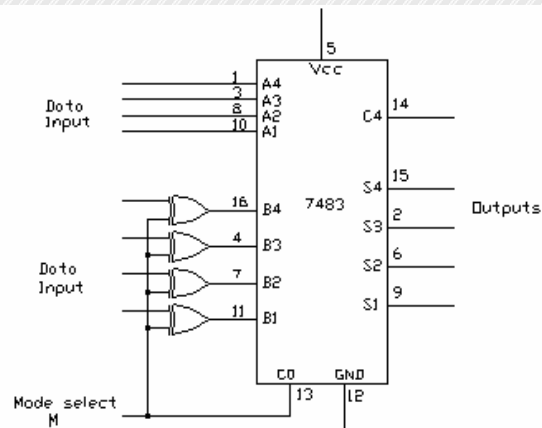


Figure 2: 4-bit adder-subtractor

Experiment 3

Seven Segment Display Decoder

Introduction:

The 7-segment display is used to display numbers and some letters. As its name implies, the 7-segment consists of 7-LEDs (segments) arranged in a shape such that any number can be displayed when the correct segments are activated.

Since numbers in digital systems are represented in binary, it is needed to convert numbers from binary to a code suitable for the 7-segment. This function is called the 7-segment decoder.

There are two types of 7-segment display configuration:

1) Common Anode:

The anodes of the LEDs are connected together to the Vcc. Individual inputs are inserted to the cathode of each LED through a limiting resistor, so that it will light *if the input is Low*.

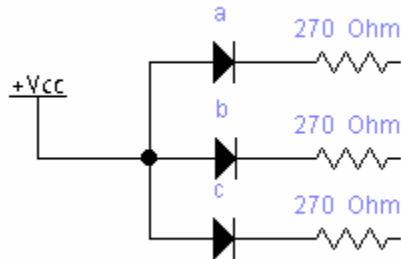


Figure 1: Common Anode Configuration

2) Common Cathode

Connecting the cathodes together to ground will make the LED light *if the input is High*, so the display is called common cathode. Limiting resistors are needed for each input to limit the current through the diodes.

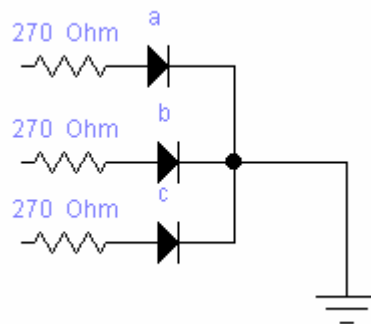


Figure 2: Common Cathode Configuration

Objective:

To construct a circuit that takes a 3-bit binary integer with a decimal value between 0 and 7, and produces the corresponding 7-segment code.

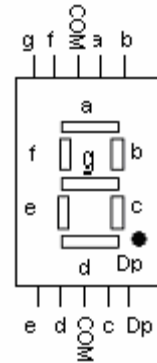
Pre-Lab.:

(To be submitted at the beginning of the lab session.)

The required 7-segment decoder should have 3-inputs (which are the bits of the binary number desired to be designed, call them A,B,C), and 7 outputs (the 7 segments of the display unit which are a, b, c, d, e, f & g).

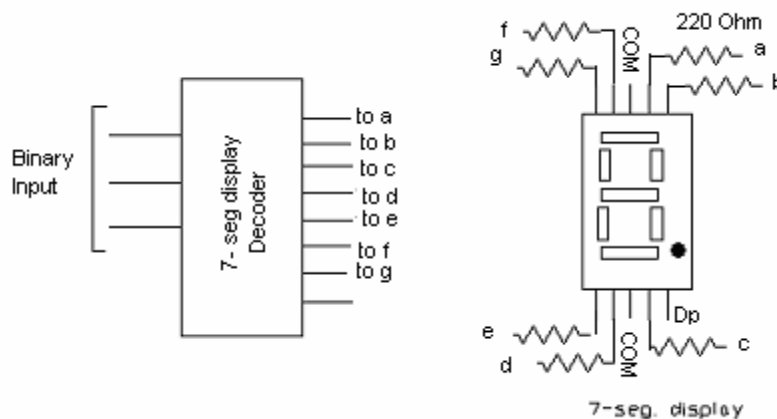
The 7-segment to be used is of **common anode type**. Consequently, **any segment will be ON if its input is Low**, meaning that for displaying 0 the segments inputs (a,b,c,d,e,f,g) should be (0000001), or g will be OFF while all the others are ON.

- 1- Make a table explaining the inputs and the corresponding outputs for the 6 combinations input (000....101), assuming the other two combinations as don't care.
- 2- Find the output as a function of the inputs (A,B,C) using K-map to minimize the expressions
- 3- Show your design using 2-input, and 3-input NAND gates, and inverter.
- 4- Test your design using EWB software.



Procedure:

- 1- Build the designed circuit using 2-input NAND gates (7400), and 3-input NAND gates (7410).
(Use one inverter IC to provide the complement of the inputs).
- 2- Test the circuit on the 7-segment display and record the displays. **Connect ONLY one of the common pin (com) to Vcc leaving the other empty**, and connect the outputs of the decoder to 220Ω resistors to the inputs of the 7-segment as shown in the attached figure.
- 3- Test your circuit for proper operation.



Experiment 4

Decoder and Multiplexer

Introduction:

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. The decoders to generate 2^n minterms of n input variables are called n -to- m line decoders.

Combinational logic implementation: A line decoder provides the 2^n minterms of n -input variables. Since any Boolean function can be expressed in sum of minterms canonical form, one can use a decoder to generate the minterms and the external OR gate to form the sum. In this way any combinational circuit with n inputs and m outputs can be implemented with a n -to- 2^n line decoder and m OR gates.

A digital multiplexer is a combinational circuit that selects binary information from many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally there are 2^n input lines and n selection lines whose bit combinations determine which bit is selected.

Multiplexer ICs may have an enable input to control the operation of the unit.

Objective:

- To construct combinational circuits for the decoder and multiplexer.
- To implement logical functions using decoder and multiplexer.

Components:

74138: 3-to-8 decoder
74151: 8-to-1 multiplexer
Combinational logic gates

PreLab:

1. Using 2-input AND gates (7408) and Inverters, design a circuit that performs the operation of 2-to-4 decoder
2. Design a 2-to-1 line multiplexer using only 2-input NAND gates.
3. Test your previous designs using EWB software using both logic gates and IC's.

Procedure:

1. Analysis of 74138: Build the circuit of Figure 1.

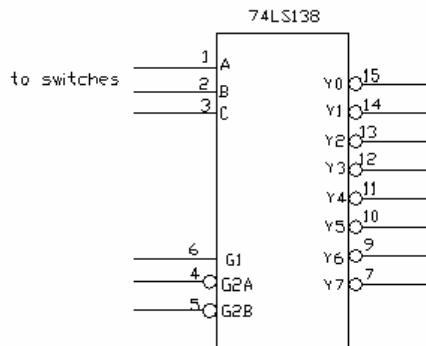


Figure 1

- Set the switches to all possible combinations, and record the outputs in a complete truth table. (Note that the 3-inputs G1, G2a, G2b are chip enables and should be active for the chip to operate as a decoder, i.e. G1=High, G2a=G2b= Low).
2. Build your design in prelab.1. Set the inputs of your design in a way similar to that of part 1, and record the outputs in a complete truth table.
 3. **In your report**, design -using 74154 Decoder and **OR gates**- a combinational circuit that has 4-inputs A,B,C,D, and F1 as the output:

$$F1(A,B,C,D) = \Sigma (0,3,7,9,14)$$

4. Analysis of 74151: Build the circuit of Figure 2.

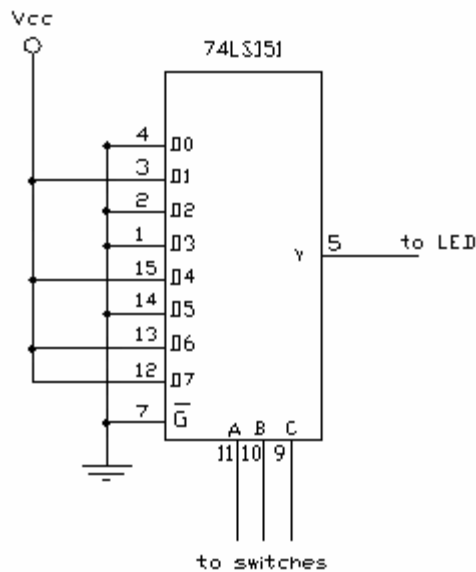


Figure 2

- Set the switches to all possible combinations, and record the outputs in a complete truth table.
5. Build your design in prelab.2, and record the outputs in a complete truth table.
 6. **In your report**, design - using an 8-to-1 multiplexer- a combinational circuit that has 4-inputs A,B,C,D, and F2 as the output:

$$F2(A,B,C,D) = \Sigma (1,2,5,6,7,10,11,14)$$

Experiment 5

Sequential Circuits

Introduction:

The outputs of sequential circuits depend on both their current input and previous outputs. There are two types of sequential circuits:

- Synchronous: output is dependent on a clock signal.
- Asynchronous: output is independent on the clock signal.

Objective:

In this experiment you will build a simple RS flip-flop, observe the behavior of a 7476 JK flip-flop, and build a 4-input up counter using JK-flip flops.

Also you will test a synchronous 4-bit counter 74161 which is fully programmed

Components:

- 1 x 7402 Quad NOR
- 1 x 7408 Quad AND
- 2 x 7476 Dual JK flip-flop
- 1 x 74161 synchronous 4-bit counter

PreLab.

Design a 4-bit up counter that counts from 0000 to 1111 and repeat, using JK flip flops.

- 1- Determine the number of flip flops required for your counter
- 2- Draw the excitation (truth) table for the JK flip flop.
- 3- Draw the state table and the state diagram for the JK based counter.
- 4- Using the 74LS76 and minimum number of external gates, design your counter.
- 5- Draw the schematic for your counter. If you are not using the preset and clear inputs connect them to their inactive state.
- 6- Test your design using EWB software using both logic gates and IC's.

Procedure:

1- Insert a 7402 chip into the breadboard. Connect power and ground to pins 14 and 7, respectively. Construct the circuit shown in Figure 1 (a).

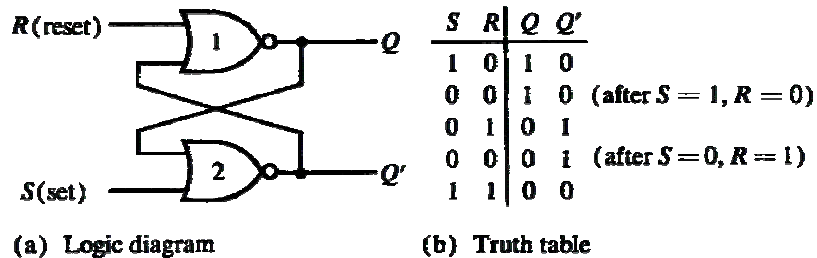


Figure 1: Basic RS flip flop

2- Confirm that your circuit produces the outputs shown in Figure 1 (b).

3- Insert a 7476 chip into the breadboard on your trainer. Connect power and ground to pins 5 and 13, respectively. Connect asynchronous inputs preset and clear to high state, inputs J, K to switches, and don't forget the clock input. Observe your circuit carefully and confirm it is functioning as expected

4- Build the 4-bit counter that you designed in the pre-lab and record the outputs.

5- Insert a 74161 IC into the bread board, connect power and ground as indicated by the data sheet, and program the counter to count from 0110 to 1101. Use any external gate if necessary. Connect the two enable pins (ENP and ENT) to high state while counting, and CLR to inactive state.

Experiment 6

Registers

Introduction:

A register is a group of flip-flops. Each flip-flop is capable of storing one-bit of information. A shift register is capable of shifting its binary information in one or both directions. Commonly used shift registers include serial in-serial out shift registers, serial in-parallel out shift register, parallel in-serial out shift register, and parallel in-parallel out shift register.

In this experiment we will build two simple forms of registers, and also build a parallel-in-serial-out shift register.

Components:

- 2 x 7474 Dual D-Flip Flop
- 2 x 74153 Dual 4X1 multiplexer
- 1 x 7432 Quad OR
- 1 x 7408 Quad AND
- 1 x 7404 Hex inverter
- 1 x 74161 synchronous 4-bit counter
- 1 x 74166 8-bit shift register

PreLab.

- 1- Design 2-bit register with parallel load control input, the load input determines the action to be taken with each clock pulse. When the load input is 1, the data at the input is transferred into the register with the next positive edge of the clock, but the output will not change if the load input equals to 0.
- 2- Test your design using EWB software using both logic gates and IC's.
- 3- Study the circuit in Figure 3 which represents parallel-to-serial converter. The counter provides timing to the 8-bit register (74166) such that the data at the inputs are inserted once, and then shifted out serially from the serial output QH. **Determine the necessary logic levels for all the pins (especially the control pins) for proper operation and record them on the figure,** (Don't simulate the circuit on EWB).

Procedure:

- 1- Build the 4-bit parallel-in parallel out register shown in Figure 1, use 4-switches as inputs and LEDs as the outputs. Change the inputs and record your results.

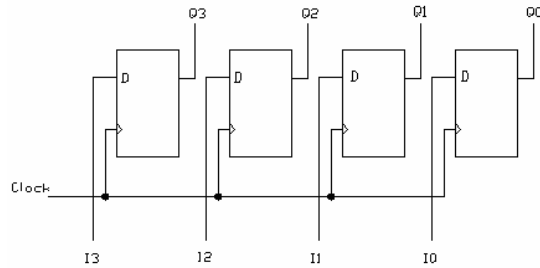


Figure 1: Simple 4-bit register

- 2- Build the 2-bit register with parallel load, designed in pre-lab 1, and record the results.
- 3- Build the parallel to serial converter shown in Figure 3, apply inputs 10101010 to the input data and record the output QH after each clock pulse.

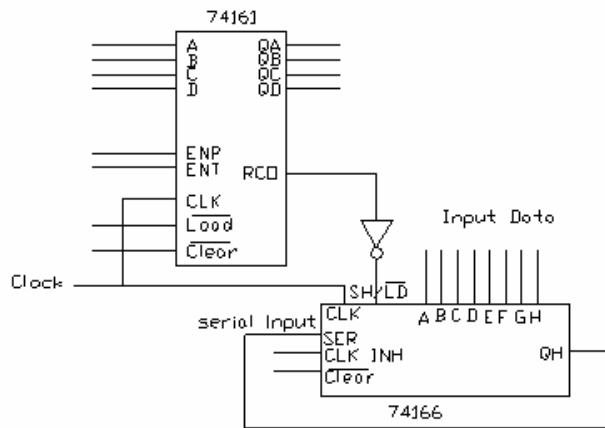


Figure 3: parallel-to-serial converter

- 4- Build carefully the circuit shown in Figure 2, record the outputs, and discuss its function and operation.

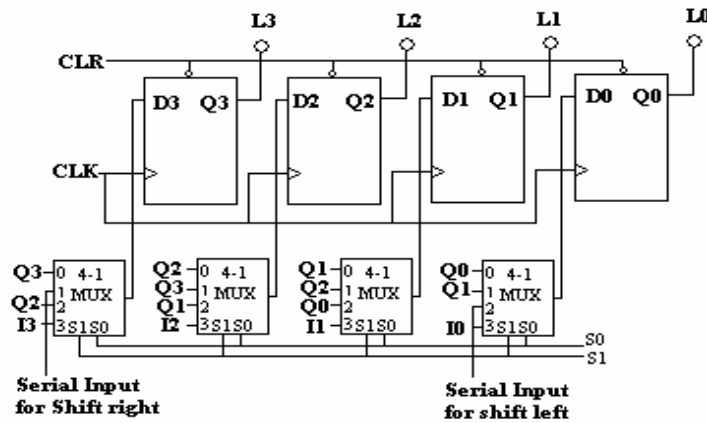


Figure 2: Register

Experiment 7

RAM /ROM

Objective:

To investigate the storage capability of Random Access Memory (RAM) and Read Only Memory (ROM)

Introduction:

Memories are divided into two types:

- 1- Read Only Memory (ROM), which supply fixed data to be read out of the device.
- 2- Random Access Memory (RAM), where data can be read or changed in the same circuit.

ROMs are storage units in which data is stored once and can not be changed then. Yet, some kinds of ROMs are designed to be capable of changing the stored data using special programming equipment apart from the reading application where the data can be read many times. These are called PROMs (Programmable ROMs). Another facility is the ability to erase the ROM contents either by ultra-violet radiation (EPROM) or electrically (EEPROM).

ROMs differ from RAM in that they can keep the information even if power is not available to the chip. This makes them suitable for applications where permanent information is needed, such as the BIOS programs in the PCs.

D27128A is a 16kX8 EPROM. The pin assignment of the chip is shown in Figure 1. The address lines are A0 through A13, and the data outputs are D0 to D7. V_{pp} and \overline{PGM} are used only during programming. For reading operation should be connected to V_{cc} . The chip enable \overline{CE} enables the chip or leaves it in standby mode. In addition, the output enable \overline{OE} enables or disables the output.

IC type of 6116 is a 2KX8 static RAM. Its pin assignment is shown in Figure 2. The address inputs select one of the words in the memory. The write enable \overline{WE} input determines the type of operation. The write operation is performed when $\overline{WE} = 0$. This is a transfer of the binary number from the data inputs into the selected word in memory.

The read operation is performed when $\overline{WE} = 1$.

The data lines (D0...D7) are input/output lines, so that the stored data can be read or written using these lines. The control pin \overline{CS} (Chip Select) selects or enables the chip so that if it is high, the chip is disabled. The \overline{CS} makes it possible to deactivate the chip as if it is not present, so that other devices connected to the same data bus or address bus can utilize them freely. This mechanism is used in microprocessor systems where many chips share the same data and address buses. Also, the output enable \overline{OE} can link the internal data to output lines or can leave these lines in high-impedance state.

Pre-Lab. 1 (ROM)

Prepare a table converting binary into the code suitable for showing the binary number on a 7-segment display (use only 4-bits). The 7-segment should display the hexadecimal digits (0,1,2,...,E,F) in response to the equivalent binary numbers (0000, 0001, ... , 1110, 1111).

Note that the address lines of the ROM will serve as the input binary code, while 7 out of 8 output data lines will provide the required state of the 7-segment inputs (a,b,c,d,e,f,g).

Remember that the 7-segment display that you have is a common-anode one.

Design a block diagram to read the contents of the ROM using 7-segment display.

Pre-Lab. 2 (RAM)

Design a block diagram such that:

The 4-bit counter 74193 will supply the address to the RAM.

The 4-logic indicators will be used to show the address.

The 7-segment display will show the data in decimal format (BCD) through the BCD-to-7 segment decoder 7447, which accepts BCD numbers (4-bits) and provides the needed outputs for the 7-segments.

A switch will be used to control the Read/Write function.

Note: use only the least significant 4-address lines.

Components:

D27128:	16kX8 EPROM
6116	2KX8 RAM
7-segment display (C.A.)	
7447	BCD-to-7 segment decoder
74193	4-bit counter

Procedure:

1) ROM:

Note: The EPROM should not be moved in or out of the circuit while the power is ON.

- 1- Program the EPROM with the data you prepared in the pre-lab using the programming equipment which includes necessary hardware and software.
For programming the chip:
 - a) Enter the programmer utility from the main menu.
 - b) Determine the manufacturer, type, and number of the device.
 - c) Determine the desired area in the device to program.
 - d) Enter the buffer, and write the desired data.
 - e) Insert the chip, and activate the programming operation.
- 2- Construct your circuit that you designed in pre-lab1, read the stored data in the chip, and make sure that the required hexadecimal digits are displayed on the 7-segment.
- 3- Turn the power off, and on again, repeat the reading process of the chip.
- 4- Make $\overline{CS}=1$, what is the output? Explain.

2) RAM

- 1- Construct the circuit that you designed in pre-lab 2, use the 74193 counter to supply the addresses to the chip (A0 to A3) and show them on four LEDs. Connect other address lines to GND. Use the data outputs (D0 to D3) to be displayed on the 7-segment. Record the contents of the RAM.
- 2- Modify the circuit to connect the 6116 I/O lines to the data switches instead of the 7447.
- 3- Store your registration numbers (yours and your partner's), so that the digits are displayed successively in addresses 0000 up to 1111, and store (F) between the numbers. In writing, the desired address is applied, the required data word is formed through the switches, then a low pulse should be applied to the \overline{WE} .
- 4- **Do not turn the power off.** Rewire your circuit, replacing the data switches with the 7447 decoder. At each address, observe the contents of that memory location. Record your result in a table.
- 5- What will happen if the power is turned off for seconds then on?

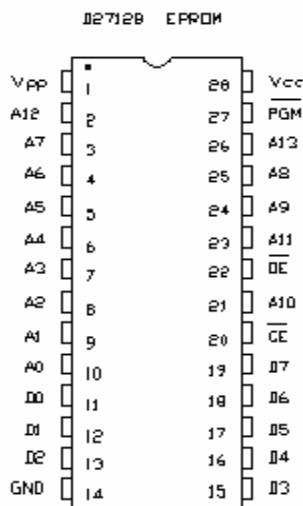


Figure 1: ROM pin configuration

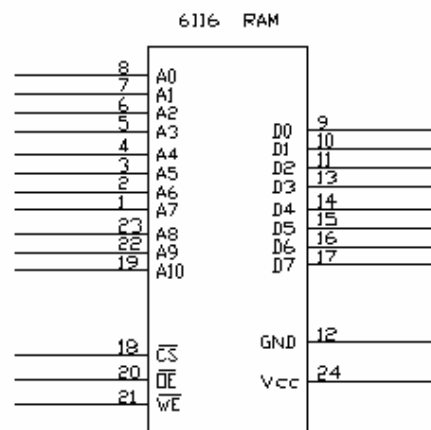


Figure 2: RAM pin configuration

Experiment 8

Will be posted later during the semester.

Appendix 1 Data Sheets

<u>IC Name</u>	<u>IC Number</u>	<u>Page</u>
Quad 2-Input NAND Gate	7400	A-1
Quad 2-Input NOR Gate	7402	A-3
Hex Inverter	7404	A-4
Quad 2-Input AND Gate	7408	A-7
Triple 3-Input NAND Gate	7410	A-8
Dual 4-Input NAND Gate	7420	A-9
Quad 2-Input OR Gate	7432	A-10
BCD-to-7 Segment Decoder	7447	A-11
Dual D-Flip Flop	7474	A-13
Dual JK Flip-Flop	7476	A-14
4-Bit Binary Adder	7483	A-15
Quad 2-Input XOR Gate	7486	A-16
3-to-8 Decoder	74138	A-18
8-to-1 Multiplexer	74151	A-20
Dual 4-to1 Multiplexer	74153	A-22
4-to-16 Decoder	74154	A-23
Quadruple 2-to-1 Multiplexer	74157	A-25
Synchronous 4-Bit Counter	74161	A-26
8-Bit Shift Register	74166	A-28
4-Bit Up/Down Counter	74193	A-30
4-Bit Register	74194	A-32
2K*8 RAM	6116	A-35
8K*8 EPROM	D27C64	A-36
16K*8 EPROM	D27C128	A-38
GAL 16V8	-----	A-40
GAL 20V8	-----	A-41