Engineering Sciences 50

BASIC LOGIC DEVICES/CIRCUITS

Laboratory 1 (Formerly 2)

Purpose: These exercises are meant to acquaint you with the basic logic gates used to implement small logic designs.

Background: The primitive gates used in logic designs are the AND, NAND, OR, NOR and inverter. The following exercises will ask to verify their truth tables, build a simple logic circuit and explore the use of a medium scale chip in conjunction with a common display device.

You will use a designer board kit (aka a lab board) to assemble and test your circuits. The kit is a versatile self-sufficient package. Appendix A attached describes the list in great detail and should be read before coming to lab.

The exercises are divided into two main parts, each with sub-parts. Part 1 focus is on simple primitives, i.e. the basic gates. Part 2 focuses on a higher level primitive (the XOR, aka EXOR gate), a one bit adder, and some MSI (Medium Scale Integrated) chips dedicated functions.

Lab 2, Part1: Simple Logic Gates,

Purpose:

1. To become familiar with the operational features of the CMOS/TTL Designer.
2. To learn about simple logic gates.

Parts:

74HCT00 Quad 2-input NAND gate.
74HCT02 Quad 2-input NOR gate.
74HCT04 Hex inverter.
74HCT08 Quad 2-input AND gate.
74HCT32 Quad 2-input OR gate.

Introduction

In this experiment we will examine basic digital logic gates and a display analog component. We will be looking at nearly all of the basic logic functions that are in common use. They are the AND, OR, NOT, NAND, and NOR gates. We will be using these gates to build a wide variety of digital logic circuits. The purpose of this lab is to verify that each gate does what is expected and to become familiar with the use of the lab board. Please read the Introduction to the ES50 labs before proceeding. The information about using the lab board found in this introduction is necessary to complete this experiment.

Part 1: Basic Digital Logic Gates

A. The AND gate

Look at the ICs provided for this experiment. One will have the number 74HCT04; another will have 74HCT08. Two sets of numbers are printed on each IC: the 74HCTXX is the TTL (Transistor to Transistor Logic) part number; the other number is generally, but not always, the year the device was made (9043

means the 43rd week of 1990). Different families have different characteristics. The HCT refers to the family of TTL compatible logic, which we will be using in this course. Note:

> **The acronym HCT actually means High-Speed CMOS that is TTL compatible. We will not make much distinction between actual TTL chips like the Low-Power Schottky (74LSXX) line and TTL compatible chips like the HCT line. We will refer to both as TTL.**

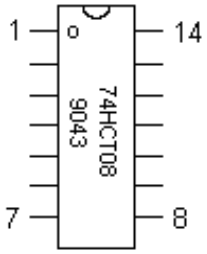


Figure 1-1

The earliest TTL parts had no family designation (*i.e.* 74XX). The pin diagrams for most of the chips (ICs) you will be using (also called pinouts) are the same as the original 74XX family. This is useful if you need to look up the reference in your data manual.

Step 1:

After you have located the 74HCT08, look at the pin configuration in your data manual. The 74HCT08 is in a 14-pin Dual In-Line Package (DIP). Chips come in 8-pin, 14-pin, 16-pin, 24-pin, and 40-pin versions. Looking down on the IC, you will notice either a notch at one end, a dot next to one of the pins, or both. These are the keys to the pin numbering sequence. If you hold the IC so that the notch is at the top, pin #1 is located to the left of that notch. If you see a dot, the dot is next to pin #1. From that point, pins are numbered consecutively in a counter-clockwise direction ending with pin #14 opposite pin #1. The 74HCT08 is a QUAD 2-input AND gate. There are four AND gates in the chip. Each gate can be used independently. The power connections are common to all the gates in the chip. Pin #14 is the V+ connection for all of the gates, and pin #7 is the ground connection for all of the gates. **Remember that each chip must have power (both V+ and GND) to work**.

Step 2:

Plug the 74HCT08 into the breadboard, straddling the center cavity. This way you can make connections to each pin independently. Make sure that all the pins are straight before proceeding. Use a pair of needle nose pliers if necessary to straighten out any bent pins. A rocking motion back and forth helps to slide

the pins into the breadboard. The pins will fit snugly into the holes. Check that all the pins are in place. The first time a chip is inserted into a breadboard is the hardest since the pins are bent slightly outward at the factory. If you notice this, try bending the pins inward against a tabletop or with the pliers until they are parallel. After the first time, the chip should just pop in.

Step 3:

Do all the wiring on the lab-board with the power off. Only turn the power on after you have completed each circuit or subcircuit. With the switch off, connect power and ground to the long parallel strips at either side of the board. Note that the each strip is independent and that there is a break in the middle. If you want to connect them together, you must do it externally. **Figure 1-11** (attached at the end of this lab) shows the wiring diagram for this circuit. Connect pin #7 to the ground bus. Any of the 4 holes below pin 7 could be used. However, using the furthest hole first can reduce clutter and tangled wires on the board. Connect pin #14 to V+. The truth table for the AND gate is as follows:

The And gate

A	B	A • B
0	0	0
0	1	0
1	0	0
1	1	1

A and B are the inputs. A • B is the output. The "•" is used in logic equations to represent the AND function. Verify this truth table, using the logic switches as inputs to test each case and a lamp monitor to observe the results. Remember that V+ is a logic 1 and GND is a logic 0. Never assume that an unconnected pin will be a logic 0; a logic 0 requires that the pin be grounded and not left floating.

Step 4:

Record the wiring diagram and the verified truth table in your notebooks. Verify the operation of one of the other 3 AND gates in the 74HCT08 package.

Step 5:

Oftentimes, AND gates are used as controllers. For example, if input B=O, then the input from A cannot get through regardless of what it is. If input B=1, then the output of the gate is input A; that is, input A passes directly through. Verify this from the truth table.

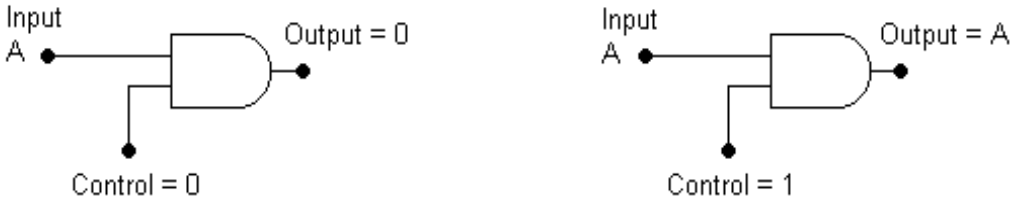


Figure 1-2

In the left drawing of **figure 1-2**, the input A is blocked, but on the right, input A passes through the gate. To display this control function, set the clock to 1Hz and connect the CLK output to the A input (just move input A from the logic switch to the CLK terminal). Also, connect input A, the CLK signal, directly to an adjacent lamp, so we can see the clock signal. Now, connect the other input of the AND gate to a switch and the output to another lamp. Now, you can see that, by turning the switch on, the clock pulses are passed through to the output, but when the switch is off, the clock pulses are not communicated. It is occasionally useful to think of an AND gate as a controller of information flow. We will see repeated usage of such a logic control function.

B. The OR gate

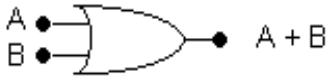


Figure 1-3

The OR gate is another primitive gate that we will be using. Its output is true (logic 1) if **any** of its inputs are true. Its output is false (logic 0) if **all** inputs are false. Here is the truth table for the OR function represented by the "+" symbol.

The OR gate

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

The pinout of the 74HCT32 quad 2-input OR gate chip is identical to the 74HCT08's pinout. Only the logic function has changed. Verify the truth table

with your logic board. You should not have to do any more wiring beyond what was needed to verify the AND gate because the pinouts are the same. Just remove the '08 and insert the '32 in its place. Remember to turn the power off first!

C. The NOT gate (a.k.a. the Inverter)

You have already been introduced to the notion of the inversion of a particular logic level in the CLK and $\overline{\text{CLK}}$ outputs. The bar over the function CLK means the inverse of the logic state of the unbarred variable. How that was done was not explained. A little of the mystery will be removed by discovering that there is a primitive operation called NOT. The symbol for the NOT gate or inverter is shown below:

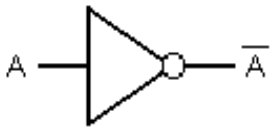


Figure 1-4

If the input is a logic 1, the output is a logic 0 and vice versa. The TTL chip which does this is a 74HCT04 hex inverter, meaning that there are six independent inverters packaged on the same chip. Check the pin-out in the data manual.

Step 1: Insert the 74HCT04 into the breadboard, and power up the chip by connecting pin 7 to ground and pin 14 to V+. Connect pin 1 to a logic switch and pin 2 (the output) to a lamp, and verify the truth table. The pinout of an inverter is different from that of the other gates.

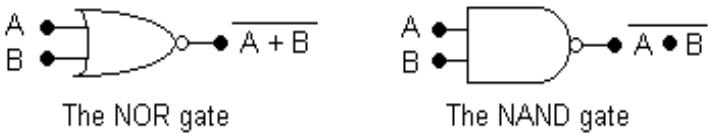
The Inverter

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

Step 2: Set the CLK output to 1Hz. Connect the CLK output to the input of an inverter and also to a lamp. Connect the output of that inverter to another lamp. Compare the light patterns. Notice that only one lamp will be on at any given time.

D. The NAND and NOR gates

The NAND and NOR gates are just the AND and OR gates we have used with an inverter at the output, represented by a bubble on the symbols:



The NOR gate The NAND gate

Figure 1-5

The truth tables for the NOR (NOT OR) and NAND (NOT AND) gates are easily derived from the AND and OR gates.

The NOR gate.

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

The NAND gate.

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

Verify the truth tables on your lab board with the 74HCT02 NOR gate and the 74HCT00 NAND gate. The pinouts for each of these devices are available in your data manual. Hint: compare the pinouts to the 74HCT08 AND gate used earlier. The NAND gate is considered the basic building block of TTL logic. Using the NAND gate you can make an inverter or combine them to make OR,

NOR, or AND gates. This can also be done with the NOR gate. Do you see how?

E. DeMorgan's Theorem

Let's use your newfound experience in digital electronics. Construct the following circuit:

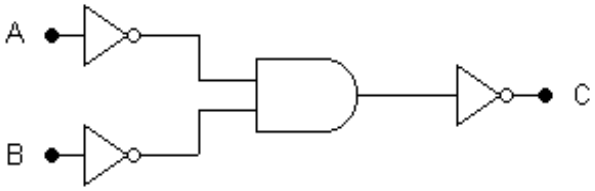


Figure 1-6

Find, experimentally, the truth table for C. Yes, it is an OR gate! This is the simplest example of the famous and extremely useful DeMorgan's Theorem. For a more in depth explanation of DeMorgan's Theorem, consult the textbook. Is it possible to build an AND gate out of OR and NOT gates? As a paper experiment, or a review exercise, you may want to try to synthesize all of the primitive gates you have seen out of NAND gates and NOR gates.

Lab 2 Part2 : Binary Addition/ Subtraction and more

Purpose:

1. To Introduce the higher level primitive EXOR (exclusive-OR) gate.
2. To build a 1-bit adder circuit from the primitive gates.
3. To build an adder with EXORs and other gates.
4. To design a 2-bit adder circuit with EXORs and other gates.
5. To verify that a MSI adder chip will perform these functions.
6. To learn how to connect a seven-segment display for readout of BCD numbers.
7. To use a 7447 to decoder/ driver to facilitate the use of the seven-segment display.

Parts:

74HCT00 NAND gate.
74HCT02 NOR gate.
74HCT04 Inverter.
74HCT08 AND gate.
74HCT32 OR gate.
74LS83 4-bit full-adder (hard to find in HCT).
74HCT86 Quad 2-input EXOR gate.
BCD seven-segment display
7447 seven-segment decoder/ driver.
7 - 330Ω Resistors.

Introduction

Higher level gates provide for abstraction in designing digital circuits. The most common higher level primitive gate, which is investigated in this lab, is the EXOR. As will soon become evident, this gate is very useful in designing adders and subtractors and greatly simplifies circuit design by supplying a layer of abstraction for the gates that it uses.

In Section B of this lab, we will explore the use of analog components in digital circuits. This basic introduction will use the concepts explored in the first lab. In addition to this, we will be using a medium scale integrated circuit to perform the necessary operations. This chip, the 7447 BCD to seven-segment decoder/ driver, also provides a very useful layer of abstraction, as well as saving space on the breadboard.

Section A: Using Higher Level Gates.

Part I: The EXOR gate

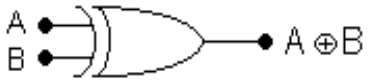


Figure 2-1

Before we proceed on to binary addition, we should introduce the higher level primitive EXOR (exclusive-OR) gate. The EXOR gate (sometimes abbreviated as XOR) is not usually considered a primitive gate since it is internally constructed out of other primitive gates, but it is quite useful to consider it as a single gate. The symbol for the EXOR function is a $\oplus$ as opposed to the + for an OR gate. The 74HCT86 is a quad 2-input EXOR gate. An EXOR gate's output is true **if and only if** one of its inputs is true:

The EXOR Gate

A	B	$A \oplus B$	Minterms
0	0	0	
0	1	1	$\overline{A} \bullet B$
1	0	1	$A \bullet \overline{B}$
1	1	0	

In part 1, we looked at the AND gate as a form of on-off switch. The EXOR gate can also be used as a controller, not as an on-off switch, but as an invert/ no invert switch. Confirm this from the truth table or experimentally. You will need this function before the lab is over. How would you construct an EXOR out of other gates?

Part 2: Binary Adders

We will now look at binary math and design a circuit to add two one-bit binary numbers. Binary representation is just base 2. The only digits used are 0 and 1. Counting in binary is straightforward:

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000

An example of the binary addition of 2 and 3:

Binary	Decimal
010	2
+011	+3
101	5

The procedure for addition in binary is identical to that for decimal; start at the rightmost bit (the least significant bit or LSB) and add that column. The LSB has no carry-in, so we can add them directly. In our example, 0 + 1 = 1, so there is no carry-out to the next most significant bit. Adding the 1 + 1 of the second bit equals 10 in binary, so we get a result of 0 with a carry of 1. This carry is added to the 0 + 0 in the leftmost bit (most significant bit or MSB) to give 1 + 0 + 0 = 1, so the result is 101.

Half-adder

The circuit that performs the adding function with no carry-in, is called a half-adder. The circuit with a carry-in is a full-adder. To design a half-adder, we should first write a truth table of what we want the circuit to do. If we have inputs A and B, an output called Σ, and a carry-out C_o we get the following truth table:

The Half-Adder.

A	B	Σ	Σ minterms	C_o	C_o minterms
0	0	0		0	
0	1	1	$\overline{A} \bullet B$	0	
1	0	1	$A \bullet \overline{B}$	0	
1	1	0		1	$A \bullet B$

Therefore,

$$\mathbf{F}(\Sigma) = A \bullet \overline{B} + \overline{A} \bullet B \qquad \mathbf{F}(C_O) = A \bullet B$$

From these equations the circuit diagram can be drawn:

Half Adder

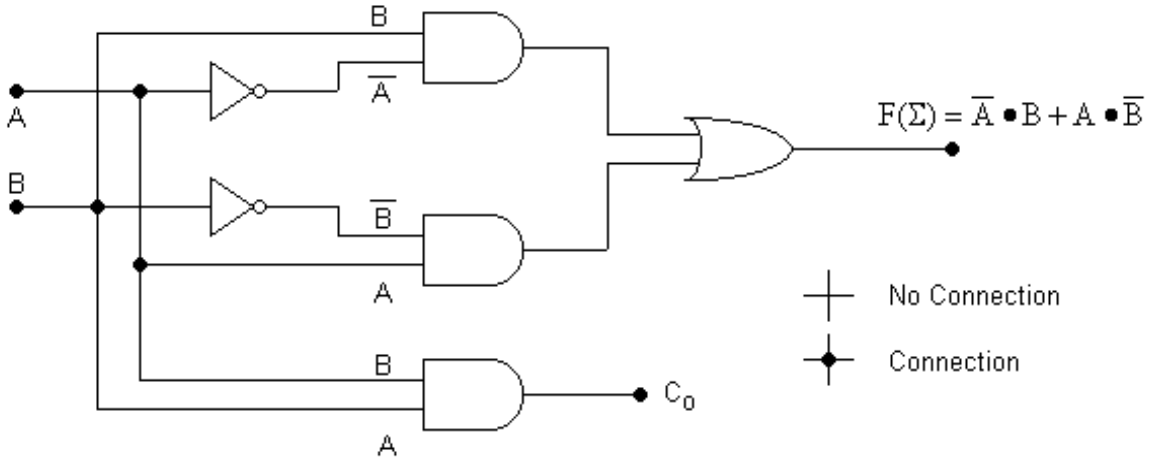


Figure 2-2

Connect up the circuit and confirm the truth table. You need 3 AND gates, 2 inverters, and an OR gate, and since there are 4 AND gates and 4 OR gates on each 74HCT08/ 74HCT32 chip, respectively, and 6 inverters in a 74HCT04 chip, you need one of each chip. This circuit is not the only way to design a half-adder. Perhaps as a review for lab exams you might find others. Build and test this circuit, remembering to keep the wires as far as possible from the chips. This will make your debugging easier, and it will help us to look at your circuits if you need help.

A close comparison between the minterms for the EXOR gate and the half-adder's F(Σ) output reveals that only an EXOR gate is required to generate the F(Σ) output, reducing the total number of gates down to two. The AND gate is still required to generate the carry-out.

Half adder with EXOR

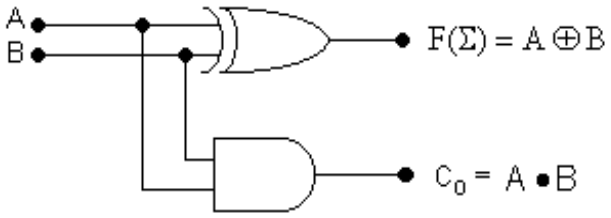


Figure 2-3

Full-adder

A half-adder is not complete because it cannot handle any carry-ins. We need to design an adder that can handle the required addition, a carry-in, and a carry-out. Such a circuit is called a 1-bit full-adder. To create larger adders, we start with a half-adder to take care of the LSB (the LSB will not have a carry-in) and then cascade full-adders with it. Cascading a half-adder with a full-adder results in a 2-bit full-adder. An additional full-adder produces a 3-bit adder, and so on. The truth table for a full-adder has a carry-input, C_i in addition to the other entries that the half-adder had too:

The Full-Adder

A	B	C_i	Σ	Σ Minterms	C_o	C_o Minterms
0	0	0	0		0	
0	1	0	1	$\overline{A} \bullet B \bullet \overline{C_i}$	0	
1	0	0	1	$A \bullet \overline{B} \bullet \overline{C_i}$	0	
1	1	0	0		1	$A \bullet B \bullet \overline{C_i}$
0	0	1	1	$\overline{A} \bullet \overline{B} \bullet C_i$	0	
0	1	1	0		1	$\overline{A} \bullet B \bullet C_i$
1	0	1	0		1	$A \bullet \overline{B} \bullet C_i$
1	1	1	1	$A \bullet B \bullet C_i$	1	$A \bullet B \bullet C_i$

Reading the minterms off the table, we get:

$$\begin{aligned} \mathbf{F}(\Sigma) &= \overline{A} \bullet B \bullet \overline{C_i} + A \bullet \overline{B} \bullet \overline{C_i} + \overline{A} \bullet \overline{B} \bullet C_i + A \bullet B \bullet C_i \\ &= \overline{B}\,(A \bullet \overline{C_i} + \overline{A} \bullet C_i) + B\,(\overline{A} \bullet \overline{C_i} + A \bullet C_i) \\ &= \overline{B}\,(A \oplus C_i) + B\,(\overline{A \oplus C_i}) \\ &= B \oplus A \oplus C_i \end{aligned}$$

The **F**(Σ) output can be implemented with just 2 EXOR gates:

F(Σ) output of the full-adder

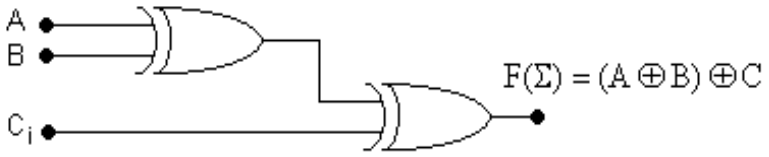


Figure 2-4

Please note that the circuit in **figure 2-4** is not the complete full-adder. We still need additional gates to generate the carry-out. However, this circuit suggests using half-adders to build a full-adder. Note how we use two EXORs, which are also used in half-adders. Using a half-adder that has two inputs, an output F(σ) (the output resulting from summing) and a carry-out "Y", we can cascade two half-adders to generate the F(Σ) output, as is seen in **figure 2-5**. However, this figure, like the one above, does not have the complete implementation of the carry-out yet.

Full-adder **F**(Σ) output constructed with half-adders

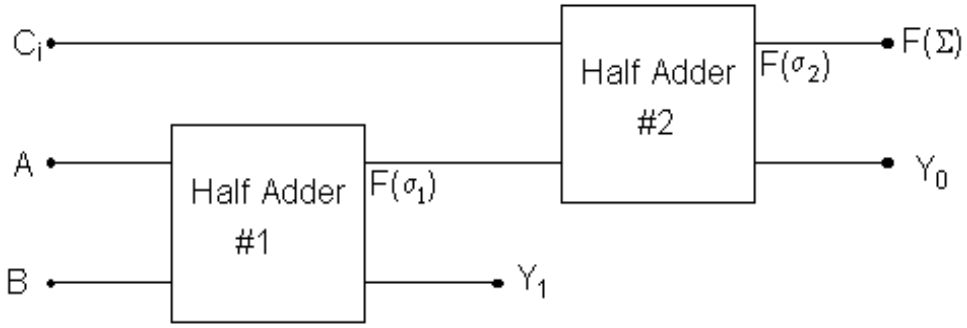


Figure 2-5

The carry output of the full-adder can be determined by going back to the truth table:

$$\mathbf{F}(C_o) = A \bullet B \bullet \overline{C_i} + \overline{A} \bullet B \bullet C_i + A \bullet \overline{B} \bullet C_i + A \bullet B \bullet C_i$$
$$= A \bullet B \bullet \overline{C_i} + C_i(\overline{A} \bullet B + A \bullet \overline{B}) + A \bullet B \bullet C_i$$
$$= A \bullet B\,(\overline{C_i} + C_i) + C_i(A \oplus B)$$
$$= A \bullet B \; + \; C_i(A \oplus B)$$

This would lead us to construct the following circuit:

The complete full-adder

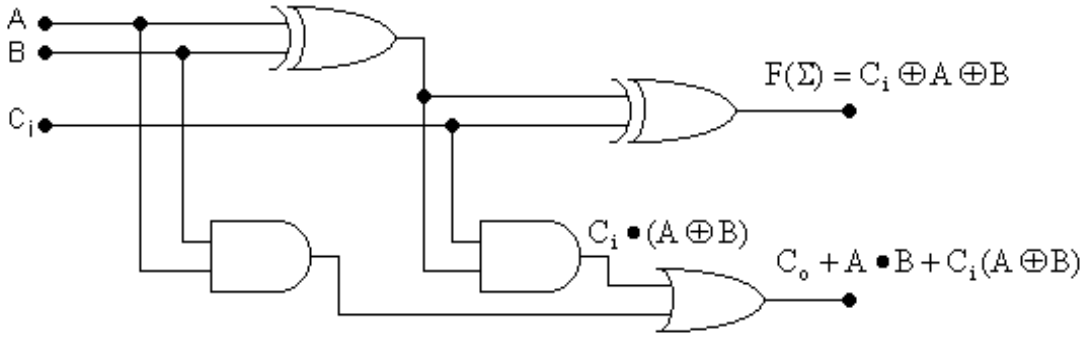


Figure 2-6

Build this circuit and test it. Note that it involves a large number of gates, even with the high level primitive EXOR gate. Without an EXOR gate, it would take even more gates. It is perhaps helpful to look at this circuit in terms of half-adder elements:

The full-adder using half-adder elements

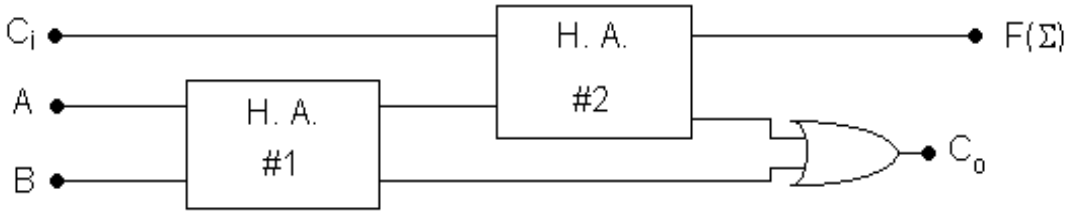


Figure 2-7

Part 3: A 2-bit adder

In building large digital circuits, it is generally possible to design and build them using a number of smaller modules. This is obviously true of adder circuits. As a preliminary exercise, write down the truth table for a 2-bit adder (you need not have a carry-in on the LSB). The inputs should be A1, A2, B1, B2, and the outputs: S1, S2 and Carry. Then, using half-adders as modules, draw in your lab book (do not build) a circuit diagram of a 2-bit adder. Then repeat using full-adders. The size of the truth table should impress upon you the relative complexity of building a 2-bit or larger adder out of gates, even if using EXORs.

The 7482 is a 2-bit adder, but, unfortunately, it is no longer available.

We are therefore forced to use a 74LS83, which is a 4-bit adder. You will be using it as a 2-bit adder for this lab. Notice the block diagram of the internal circuitry on your 74LS83 data sheet. Connect the 74LS83 to switches and lamps and test it. Note that the power and GND connections are not the standard pin 8/16. Why are the unused B inputs (B3 and B4) tied to logic 1 and **not** logic 0? Note that the data book uses the Greek capital letter "Σ" (sigma) to represent the sum outputs instead of an "S".

Pinout of the 74LS83

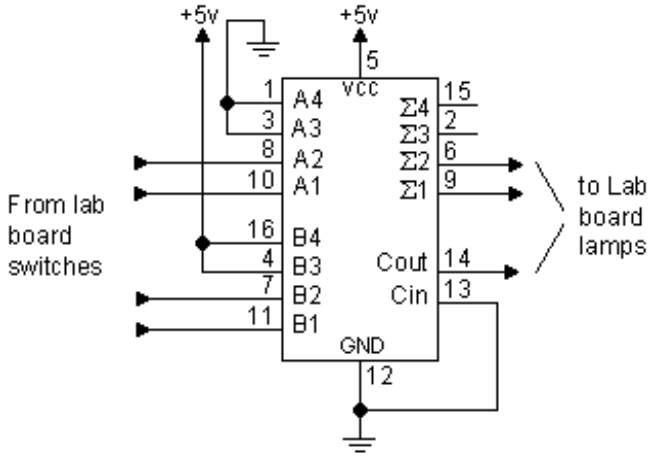


Figure 2-8

Part 4: Binary subtraction

Subtraction in binary, as in decimal, can be reduced to the addition of positive and negative numbers. The method of representing negative numbers on a bit-by-bit basis must be chosen carefully. The most convenient method is known as 2's complement. To represent a number in 2's complement, take the binary number and invert each bit. Then add 1. For example:

Decimal	Binary	2's compl.	Addition
4	100		100
-3	011	101	+101
1			(1)001

Disregard the final carry (not final digit) when performing binary addition of 2's complement numbers. You will often not have a final carry, and thus, often will not have to disregard the last digit. This table shows that 4 - 3 = 1. Try a few more to convince yourself, but note that you cannot go out of the range {-8 < result < +7} with 3-bit numbers.

Your task for this section is to build a circuit that can do both addition and subtraction on demand. From the description above of 2's complement math, we see that we need to be able to invert one of the input numbers and to add to it a 1. To accomplish this you will have a "command" line that will cause one set of inputs (the subtrahend) to be inverted when the line is high (1) and cause no change when the line is low (0). Do you remember the EXOR invert/ noninvert function from the beginning of the lab?. The "add a 1" function can be accomplished using the carry-in line on the LSB. It is recommended that you use the switches for the A and B inputs, a piece of wire to either V+ or GND for the command signal, and the lamp monitors for the Σ_i outputs and the carry-out. Do a few addition and subtraction problems with your circuit to get a better understanding of binary math.

Section B: Using Analog Components with MSI Components

The Seven-Segment Display

In this section we study the common seven-segment display, a device for displaying decimal numbers. The device itself is quite simple, consisting of seven LEDs for the segments and perhaps one or two more for a decimal point. We will not be using the decimal points.

The display is a simple extension of the LED we looked at earlier. It is a group of LEDs arranged in an "8" pattern, with each LED capable of being lit independently. The segments are designated as follows:

Seven-Segment Display

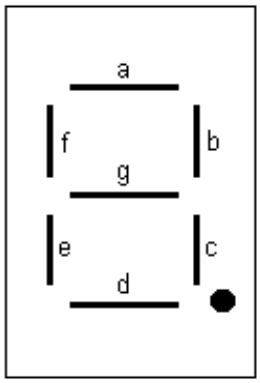


Figure 2-9

There are two possible ways of connecting the display LEDs: <u>common cathode</u> and <u>common anode</u>:

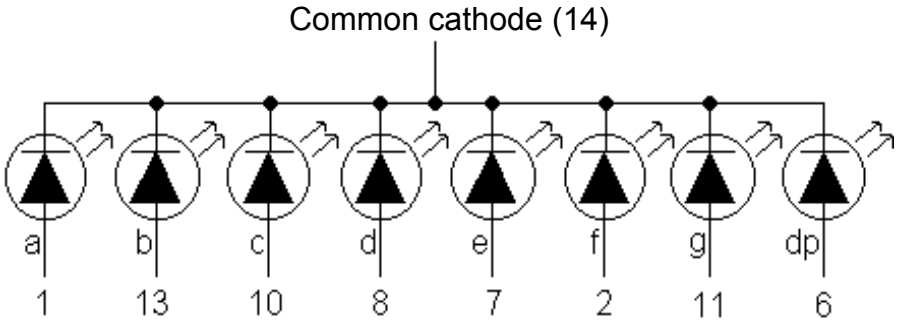


Figure 2-10

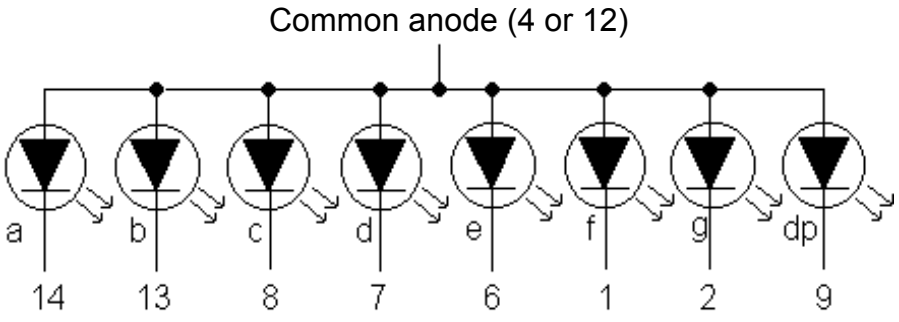


Figure 2-11

For a common cathode display, the common pin is connected to ground, and the segments are lit by connecting a positive voltage to the other pins (through a current limiting resistor, of course). In a common anode display, the common lead is connected to V+ and a segment is lit by bringing the appropriate

line to ground, through a resistor. This is opposite of how we light the lamps on our lab board.

To drive the segments of a common anode display we need logic that will convert a binary number to the segment codes. Remember that a logic 0 means that the segment is ON.

Decimal Number	BCD Code				Line Segment code (0 = ON)						
	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	1	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	1	1	0	0

Task 1:

Writing out Karnaugh maps for each segment is possible, and one set of reduced equations might be the following. Verify at least one of these from the truth table (use a Karnaugh map):

F(a) = $\overline{D} \bullet (C\,\overline{A} + \overline{C}\,\overline{B}\,A)$

F(b) = $\overline{D}\,C \bullet (B \oplus A)$

F(c) = $\overline{D}\,\overline{C}\,B\,\overline{A}$

F(d) = $\overline{A}\,\overline{B} \bullet (D \oplus C) + \overline{D}\,A \bullet \overline{(C \oplus B)}$

F(e) = $A + \overline{D} \bullet C \bullet \overline{B}$

F(f) = $\overline{D}\,\overline{C} \bullet (B + A) + \overline{D}\,B\,A$

F(g) = $\overline{D}\,\overline{C}\,\overline{B} + \overline{D}\,C\,B\,A$

Task 2:

To build this circuit ourselves would require a large number of logic gates. As a lab exercise, design and build a circuit that will drive 3 of the seven segments. You can choose which three, but choosing carefully will greatly

simplify things. Look for overlap in the equations. Use any gates you wish. As always, record your circuit and results in a notebook; it will be a useful study guide.

First you must connect up the display using the 7 330Ω resistors provided. Do you remember why they are needed?

Common anode 7-segment display pinout

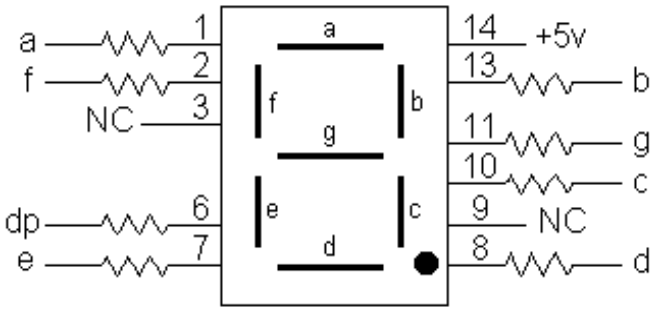


Figure 2-12

Task 3:

For this part of the lab, we will be using the 74HCT47 BCD to seven-segment decoder/ driver. Using larger scale components is the first step toward designing more complicated circuits. Wire up a complete display:

Seven-segment decoder/ driver circuit

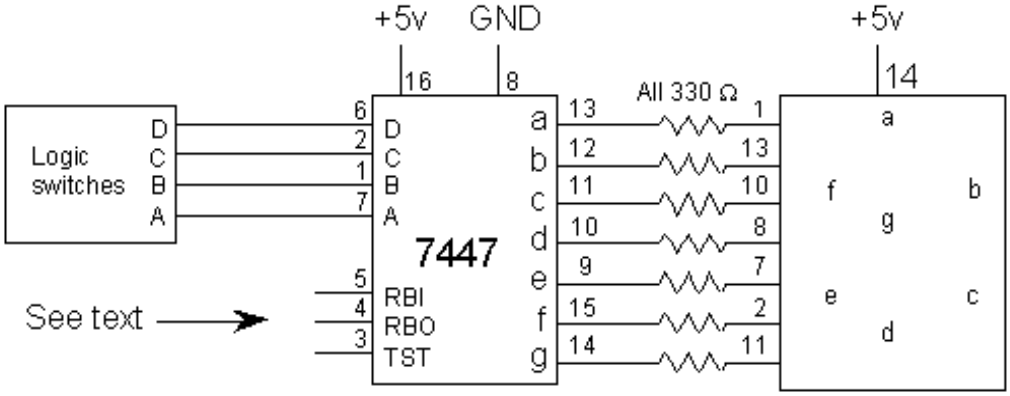


Figure 2-13

Use care in making sure that the resistor leads are not touching each other.

1. With pins 3,4 and 5 high, observe the display patterns for all possible inputs D, C, B, A. Note that 0-9 are the digits and 10-15 are symbols.
2. With pins 4 and 5 high, observe the display when pin 3 is low (grounded). What is pin 3's function?
3. With pins 3 and 5 high, observe the display when pin 4 is grounded. What is pin 4's function?
4. Set pin 3 high and connect pin 4 to a lamp monitor. Observe the monitor and display outputs for various inputs D, C, B, A, and pin 5 first tied high, then low. In particular, what happens when DCBA = 0000 and pin 5 is low? Pin 5 is the "ripple blanking input", and pin 4 is the "ripple blanking output". How might this feature be useful?

Supplementary Questions

1. Why does the seven-segment display need a resistor at each cathode, instead of one at the anode (Hint: think about V=IR and the LED's voltage and current requirements).

APPENDIX A

In this introduction you will examine some of the basic electronics design tools in the lab kit which you will soon use to build circuits utilizing integrated circuits or "IC's". You will learn some simple precautions, basic techniques, and rules, but above all we hope that you will begin to appreciate how relatively simple it is to make use of powerful electronic techniques.

You will find the prototyping digital lab-board, which will be used throughout the course, to be a valuable electronic “sketch-pad” for trying out new ideas. Some of you may find this section somewhat elementary if you have taken other electronics courses before. Although the lab-board is a fairly simple device, no electronics designer worthy of the name would be far from such an instrument.

The physical layout is shown in an accompanying figure. Details of the connections on the breadboard are also shown on a separate figure.

The Lab-Board

The CMOS TTL Designer, which will be referred to as the lab-board, will be divided for discussion into several parts (see **figure 0-1** attached at the end of the these notes).

Part I: Power Switch and Fuse

Plug the lab-board into any standard 3-prong grounded outlet. If a three-prong outlet is not available, use the adapter plug in your briefcase. When the power switch is on, the red monitor light in the switch should be lit. If it is not, check the fuse in the following manner: Unplug the lab-board, release the fuse holder by pushing, then lift the holder and take the fuse out. Examine the fuse, which is a glass tube with a very thin wire running through it. This wire melts if the current flowing to the lab-board exceeds the maximum amount the lab-board can safely carry. If the thin wire appears to be broken or missing, the fuse is “blown” and must be replaced with a fuse of the proper amperage rating. Another, more reliable, method of checking the fuse is to measure the “resistance” through the use of a “multimeter” (which you should already know how to use). The resistance should be no more than a few ohms for a good fuse.

If the lab-board still does not turn on and the fuse appears to be okay or if you are not sure how to check it with the multimeter, bring the board to a course staff member to be checked. Note: the fuse does not prevent one from burning out an IC (Integrated Circuit) device. Only care on your part can prevent damage to an individual device. If you suspect a device is burned out, return it to a course staff member for a replacement.

Part II: V+ and GND Terminals

The V+ switch (TTL / CMOS) to the right of the fuse determines what voltage appears between the V+ terminal and ground (located to the left of the breadboard). This voltage is used to power all the circuits you will build. In the TTL position the terminal is supplied with a constant voltage of approximately 5 volts with respect to the GND (ground) terminal. In the CMOS position up to 15 volts may appear at the V+ terminal, depending on the position of the control to the left of V+ and ground terminals. You will be working with Transistor-Transistor Logic (TTL) which will burn out if supplied with more than 5 volts. Therefore, the V+ switch should always be in the TTL position unless otherwise directed. We have placed a metal tab across the switch to prevent it from being hit accidentally. Should this metal tab become loose, position the tab so that the switch cannot be put in the CMOS position and tighten the screw securing the tab with the screwdriver supplied in your kit.

To prevent any possible damage to the lab-board, no external power supply should be connected to the board.

In further discussion, Logic 1 will mean "near 5 volts," and Logic 0 will mean "near 0 volts."

Part III: Lamp Monitors

There are 4 logic level lamp monitors, which we will call lamps, and a wire terminal below each lamp. The lamps will light if a logic 1 voltage is supplied to the terminal below them. Demonstrate this for yourself by connecting a lamp to a logic 1 voltage. In order to do this, it will be necessary to cut a piece of hookup wire from the coils provided. The convention used in this course is a standard, namely red wire for V+ supply and black wire for ground. Other colors are used for the signals in your circuits. Cut a piece of red wire long enough to connect the

V+ terminal to any of the 4 lamps. Use wire strippers to remove the insulation from about **3/16"** of the wire on each end. The stripped length of the wire is very important! If the stripped wire is too long you may make unwanted connections or "shorts" under the breadboard. If the stripped wire is too short you may not get a connection. Put one end into the V+ terminal and the other into the lamp terminal. If the lab-board is on, the lamp should light. The rest of the lamp circuit to ground is made inside your lab-board. Verify that each light works. The lamps are actually Light Emitting Diodes (LEDs) with some circuitry that powers the lamps whenever the proper logic level is applied to the lamp terminal. None of the voltage levels on the lab-board can damage the LEDs, but an external supply could!

Part IV: Logic Switches

There are 4 independent switches with a wire terminal above each of them. The logic level (or voltage above ground) of each terminal is determined by the position of the switch. Cut a wire and connect a switch to a lamp. Verify that the lamp lights when the switch is in the V+ (logic 1) position and does not light in the GND (logic 0) position. Verify that each switch operates correctly by making the appropriate connections. These switches will be very useful in constructing truth tables for TTL devices by allowing up to four inputs or controls to be easily set at levels O or 1. The device output levels can be determined by whether or not the lamps light.

Part V: Clock (CLK)

The CLOCK circuit behind the indicated section of the lab-board causes the logic level at the CLK terminal to alternate between a logic 1 and logic 0 at a rate determined by setting the switch. Connect a piece of wire from the CLK terminal to a lamp terminal and set the clock switch to 1Hz (1Hz = 1 cycle per second). The lamp should light up about once per second. There is a second terminal above the clock switch marked $\overline{\text{CLK}}$. This terminal gives the opposite logic level from that of the CLK terminal. Connect the CLK terminal to a neighboring lamp. When the lamp connected to the CLK terminal is lit, the lamp connected to the $\overline{\text{CLK}}$ terminal is off and vice-versa. The logic level of $\overline{\text{CLK}}$ is the logical inverse of CLK.

Notice that the lamp is not on the same amount of time that it is off. This is nothing to worry about. Change the clock switch to 10Hz and notice that the lights still appear to flicker on and off. At 100Hz they appear to be always on. The eye cannot observe flicker rates above about 20Hz (That is why movies and television work at the frame rate or refresh rate at which they operate); you would need an oscilloscope to look at signals this fast.

Part VI: The Pulsers

The electronic circuits connected to the pulsers produce 2 different pulses; the upper terminal goes from 0 to 1 when pushed and back to 0 when released, and the lower terminal goes from 1 to 0 and back. Test the pulsers using the lamps. You might notice that you cannot make the lights flicker on and off if you press the buttons gently. These pulsers are “debounced” to prevent unwanted logic transitions with a sequential logic circuit that you will see in the text.

Part VII: The Breadboard

The white plastic board, the breadboard, is used to mount the IC's and to make power, ground, and signal connections to switches, lamps, and pulsers. **Figure 0-2** is a blown up drawing of the breadboard showing the electrical connections made just below the surface. Notice the long groove running through the middle of the board parallel to the long side. On each side of the groove there are about 64 vertical groups of holes, each group consisting of five holes. **Each group of five holes on one side of the center groove are electrically interconnected, but the center groove electrically isolates the groups on either side of it.** Connect a wire from a switch to any hole in a group of five. Connect another wire from any of the other four holes in that group to a lamp. The lamp should light when the switch is in +V and will not light in GND position. If you have trouble connecting the wire, use the pair of pliers supplied.

On the top and bottom edge of the breadboard there are two parallel rows of holes in groups of five. **Figure 0-2** shows the internal connections as solid lines. The horizontal rows are all interconnected, **except across the very middle of the breadboard**, above the center top screw, and below the center bottom screw. The top rows of holes should be used as the V+ voltage bus (common connection for many elements or sub-circuits), and the bottom rows of

holes are used as the ground bus. Connect a black wire from the GND terminal on the left to one of the horizontal rows of holes and connect a red wire from V+ to one of the top horizontal rows. **IMPORTANT:** In making these connections, use a short length of wire. Tall loops of wire sticking up get in the way of future connections. You may wish to connect the left and right hand sides of the V+ bus and the ground bus. Doing so will make it possible to connect IC's anywhere on the board. You have now wired up a ground and power bus.

Part VIII: I/O

These terminals (4 in all) allow you to connect signals easily on your lab-board to the outside world. Wires directly from the breadboard would of course work, but could become inconvenient. There are two tall plastic screw terminals called banana jacks to the right of the breadboard, and two shorter metal “BNC connectors” labeled IN/OUT. These connectors may become useful in your projects if two lab-boards are used. When sharing signals between boards or devices, **remember to tie the grounds together.**

General Hints:

Here are a few general hints that may be helpful in wiring your circuits:

- Keep the wires as far as possible from the chips. If you have to replace a chip, you don't want to have to remove all of your wires first.

- Keep lengths of wire as short and neat as possible. Excess wire makes the circuit harder to debug and can get in the way.

- Do not forget the Vcc (V+) or ground (GND) connections. They are sometimes left off schematic diagrams for clarity.

- If the circuit does not work at first, check over your wiring thoroughly. Faulty wiring is the most common source of problems in circuits. Before panicking when your circuit does not work, make sure that no wires have broken. Sometimes when stripping a wire, you gouge the wire itself, making it prone to snap off when you plug it into the breadboard.

- As circuits get more complicated, spending a few minutes thinking about the layout of the circuit becomes vital. With a little planning, you can reduce the amount of messy wiring and make your circuit easier to test.

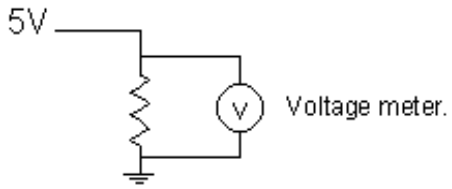
- **Do not** directly connect V+ to GND through a piece of wire. It would cause a very large current to flow through the board and blow the fuse.

- You should wire your circuit with the power off! When you have completed it, turn the power on. Wiring the circuit with the power on could possibly damage some IC's. When making any changes to the circuit, you should also turn the power off. If you are simply providing inputs by connecting a wire from place to place, you do not have to turn the power off.

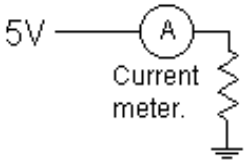
ES-50: DMM instruction sheet:

As the instruction sheet for the DMM is not very complete, we would like to offer a few helpful hints:

a) Voltage measurement is a parallel measurement -- that is the meter leads are placed <u>across</u> the two points that you wish to measure:



b) Current measurement is a series measurement -- that is you must interrupt the circuit and insert the meter in series with the rest of the circuit. You will also notice that the red meter lead must be moved from the Voltage/ Resistance connection (marked V Ω) to the current connection (marked A).



c) When making resistance measurements, be sure that the power is <u>OFF</u> in the circuit and that there is nothing else connected to your circuit that might influence the reading.

d) If you are unsure of the magnitude of the measurement you are about to make, start at the highest range on the meter and decrease the range until you have a reading.

e) When the meter reads "1 . ", it is an indication that the value you are measuring is much greater that the range you are using. For example, trying to measure 15V on the 2V scale or trying to measure 100KΩ on the 2KΩ scale will both produce a reading of "1 . "

f) Be sure to turn the meter off when finished, or you may run down the battery.