# Illinois Institute of Technology
# Department of Electrical and Computer Engineering

## Laboratory Manual for ECE214
**Analog & Digital Laboratory II**

# Introduction

This is a laboratory course for the sophomore level in electrical engineering. The primary purpose of the course is to teach the student laboratory skills that are essential for further study in electrical engineering. The emphasis is upon the use of laboratory equipment so that the student becomes as familiar with using oscilloscopes, function generators, logic analyzers and other instruments, as the student is with the use of a hand-held calculator. The method by which this is accomplished is to demonstrate the application of material learned in lecture courses within a laboratory setting.

The student should be discouraged from thinking that the "real world" does not relate to the "theory" which is being taught in lecture-style courses. Rather, the student should be encouraged to see the links between circuit theory and laboratory measurements, and to realize that if a theoretical analysis disagrees with laboratory results, then the models used for the analysis may be simply inadequate to predict the circuit's function. This then suggests the need for further sophistication of the model rather than an abandonment of theory.

In order to bridge the gap between hand analysis and laboratory measurements, computer tools including the spreadsheet and PSpice are used to sophisticate the analytical approach.

Although the manual contains a combination of digital and analog procedures, the student should be encouraged not to see them as different subjects. Rather, digital circuits should be viewed as analog circuits with specialized functions that are suitable for use in digital systems.

The appendices of the manual contain valuable information to assist the student in completing the course. In particular, the sample Laboratory Report, in Appendix B, should be examined as illustrative of an excellent report. Depending upon the particular laboratory procedure, some portions may need to be emphasized more than others are.

# Counters

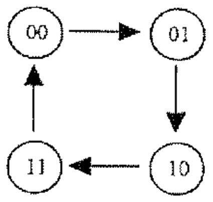
## 13.1 Objective

This experiment will review counters. Students will design counters from flop-flops and combinational logic and explore the capabilities of TTL counter chips.

## 13.2 Background

A *counter* is a sequential logic circuit that goes through a specific sequence of states. The *binary counter* is the best-known form of counter. It goes through a sequence of states in which the flip-flop values represent increasing binary values. For example, a two bit counter goes through states "00", "01", "10", and "11" before returning to state "00". In general, an n-bit binary counter goes through a sequence of states with values 0 to $2^n$-1 before returning to the 0 state. Other counting sequences are also possible. For example, a *Gray code* counter goes through states in a way such that only one bit changes at a time. A two bit Gray code goes though states "00", "01", "11", and "10" before returning to state "00". Most counters used today are *synchronous* – all flip-flops in the counter are connected to a common clock. Asynchronous counters such as "ripple counters" (discussed in ECE218) can be easier to implement but are difficult to use reliably. For this reason, this experiment deals only with synchronous counters.

It is convenient to represent the behavior of sequential logic circuits such as counters using either a *state diagram* or *state transition table*, as shown in Figure 13.1. A state diagram represents each state as a "bubble" that shows the flip-flop values for the state. "Arrows" between bubbles represent *state transitions* that occur when going from one clock cycle to the next clock cycle. Figure 13.1(a) shows the state diagram for a two bit binary counter. It shows that if the current state is "00" then the next state will be "01", if the current state is "01" the next state will be "10", etc. The state transition table provides the same information in a table, as shown in Figure 13.1(b). Given a current state that is specified by flip-flop values QA and QB, it specifies the next state values of the flip-flop, which are labeled QA* and QB*.



(a) State Diagram

| Present State | | Next State | |
|---|---|---|---|
| QA | QB | QA* | QB* |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

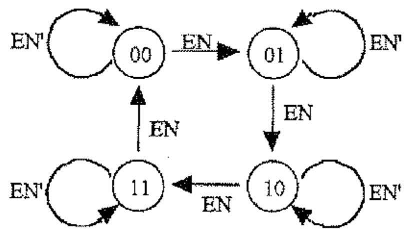
(b) State Transition Table

Figure 13.1 – Two-Bit Binary Counter

Often, state transitions in sequential logic circuits depend on an input as well as the current state. For example, a *binary counter with enable* only counts when an enable input EN is true. When the EN input is false, it stays in the current state. Figure 13.2(a) shows the state diagram for a two bit binary counter with enable.

When state transitions depend on input values, they are labeled with a Boolean expression that specifies the conditions under which the transition is taken. For example, when the counter in Figure 13.1(a) is in the "01" state and EN is true, the next state is

"10". Thus, the arrow between state "01" and "10" is labeled with the expression EN. On the other hand, if the counter is in state "01" and EN is false, then the next state will remain "01". Thus the arrow from state "01" back to itself is labeled with the expression EN' (EN prime). The state transition table describes the information in tabular form. Thus next state values QA* and QB* are described as functions for EN, QA, and QB, as shown in Figure 13.2(b).



(a) State Diagram

| Present State | | | Next State | |
|---|---|---|---|---|
| EN | QA | QB | QA* | QB* |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |

(b) State Transition Table

Figure 13.2 – Two-Bit Binary Counter with Enable

If D flip-flops are used to implement a sequential logic circuit, then the state transition table becomes a truth table for the D inputs of the two flip-flops. These logic functions can be minimized and used to create an implementation of the circuit. If other flip-flop types are used (e.g. JK flip-flops), then an additional step must be taken to determine the values of the flip-flop inputs that will result in the proper flip-flop values. These input values are known as *excitation values*. Section 5-7 of [Mano] describes this process in more detail. You should review this section before beginning the preliminary.

Since counters perform a very useful function, the TTL logic family provides several different counters. Figure 13.3 shows the pinout of the 74LS163, a 4 bit binary counter that is provided in the lab kit. In addition to performing the basic counting sequence, the 'LS163 performs several other useful functions. It has four outputs QA, QB, QC, and QD that are the outputs of the counter flip-flops (QD is the most significant bit).

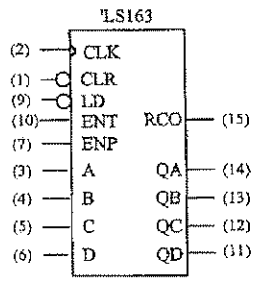


Figure 13.3 – 74LS163 Binary Counter

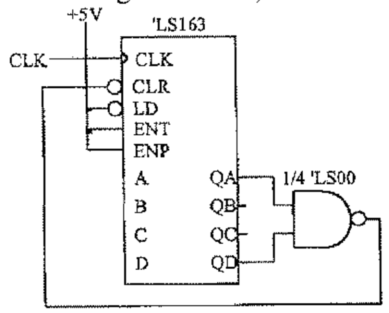


Figure 13.4 – Decade Counter using 74LS163

An additional output called RCO is a "carry" output that is true when the counter is in state "1111" and the ENT input is logic H. ENP ("Enable Parallel") and ENT ("Enable Trickle") act as enable inputs. When either ENT or ENP is logic L, the counter stops counting and remains in the current state. In addition, when ENT is logic L, the RCO output is disabled (logic L) regardless of the current state. Two counters can be combined to create an 8-bit counter by connecting the RCO output of one counter to the ENT input of the second counter.

The CLR input is an active-low "synchronous clear". It is called synchronous because the flip-flops in the counter are reset on the next rising clock edge after CLR is asserted to logic L. This is in contrast to asynchronous clear inputs, which reset the flip-flops immediately when CLR is asserted. Because it is synchronous, it can be used without worrying about the timing problems associated with asynchronous reset inputs.
The LD input loads the A, B, C, and D inputs into QA, QB, QC, and QD, respectively. Like CLR, this signal is active-low and synchronous.

With all of these capabilities, the 'LS163 can be adapted to perform many functions besides the simple binary counter function. For example, the CLR input can be used to "cut short" a counting sequence by returning the counter to state "0000". Figure 13.4 shows the use of the CLR input to create a "decade counter" which counts from 0 to 9. When the counter reaches "1001", the output of the NAND gate becomes logic L. This forces the next state of the counter to be "0000". The LD input can be used to start counting sequences at other values besides "0000". For example, if the output of the NAND was connected to LD instead of CLR, C & D grounded, and A, B & CLR tied to Vcc then counter would count from 3 ("0011") to 9 ("1001").

**13.3 Preliminary**
1. What range of values does a 10-bit binary counter go through?

2. Draw a state diagram for the self-correcting LFSR designed in the "Latches, Flip-Flops, and Shift Registers" Lab.

3. Draw the state diagram and state transition table for a 2-bit Gray code counter with enable input EN. When EN is logic H, the counter should go through the counting sequence for a Gray code. When EN is logic L, the counter stays in the current state.

4. Design the 2-bit Gray code counter with enable using D flip-flops and combinational parts available to you. Use the state transition table developed in step 3 as a truth table for the combinational logic as described previously. Do not use a PAL chip for this circuit.

5. Draw a schematic showing how three 74LS163 counters can be connected to create a 12-bit counter with enable.

6. Using a 74LS163 and other parts in the lab kit design a specialized counter with two inputs C0 and C1 (Table 13.1). Depending on the values of C0 and C1, it should go through the counting sequences shown below. Draw a state diagram for this counter. You may use a PAL with the 74LS163 for this circuit.

Table 13.1 – Counting Sequence for Specialized Counter with Two Inputs C0 and C1

| C0 | C1 | Counting Sequence |
|---|---|---|
| 0 | 0 | 0 to 15 |
| 0 | 1 | 0 to 12 |
| 1 | 0 | 3 to 15 |
| 1 | 1 | 3 to 12 |

## 13.4 In the lab

1. Build the 2-bit Gray code counter with enable designed in the preliminary. Demonstrate its operation to the instructor.

2. Build the specialized counter designed in the preliminary. Demonstrate its operation to the instructor.

## 13.5 Reference

[Mano] M. Morris Mano, *Digital Design*, Third Edition, pp. 207, Prentice-Hall, Inc., 2002.

# Logic Analyzer Familiarization

## 14.1 Objective

In this experiment, you will become familiar with the HP 54620A Logic Analyzer and its use as a debugging tool. Using two simple sequential logic circuits, you will learn how to set up the logic analyzer to trigger on different conditions and display multiple logic signals.

## 14.2 Background

You have already become familiar with the use of an oscilloscope as a tool for displaying voltage versus time. The oscilloscope can be used as a tool for logic circuits as well as analog circuits since different voltage ranges represent different logic values. The biggest drawback of doing this is that oscilloscopes have a limited number of channels (in our case two). In complex logic circuits, we may want to observer several signals simultaneously.

Logic analyzers alleviate this problem. They display logic values (i.e. 1 and 0) that represent the value of digitals versus time. Logic analyzers can display of several these signals simultaneously, making them ideal for debugging complex circuits. The HP 54620A can display up to 16 signals simultaneously (high-end logic analyzers can display many more). Logic analyzers are no substitute for oscilloscopes when debugging tricky analog problems because they can display only logic 1 and 0 values. However, they excel when you are attempting to debug complex sequential circuits.

Logic analyzers operate by repeatedly sampling data inputs and temporarily storing them while searching for a trigger condition. If no trigger condition is detected, stored values are overwritten by new values as they sampled. However, if a trigger condition is detected, then the stored data is shown on the display so that the user can see the values of the signals before, during, and after the trigger condition. In the HP 54620A, these values are displayed as "traces" similar to oscilloscope traces. Other logic analyzers may also display values as vectors of binary, octal, or hexadecimal numbers.

A major advantage of logic analyzers is that they can collect this data at high speed, making it possible to test circuits at full clock speed where errors are most likely to occur.

Logic analyzers are extremely useful when debugging sequential circuits, which may contain many signals, which change at different times. Examining these signals with a logic analyzer allows you t o see if the circuit is behaving properly and, if not, to isolate the source of the problem. Unlike an oscilloscope, which triggers only on a single signal, logic analyzers can trigger only arbitrary patterns of multiple inputs, making it possible to specify exactly the sequence of events that you wish to examine.

Figure 14.1 shows a diagram of the HP 54620A logic analyzer front panel. Notice the similarity between its controls and the controls on an oscilloscope. This similarity is intentional to make the logic analyzer easy to learn and use.

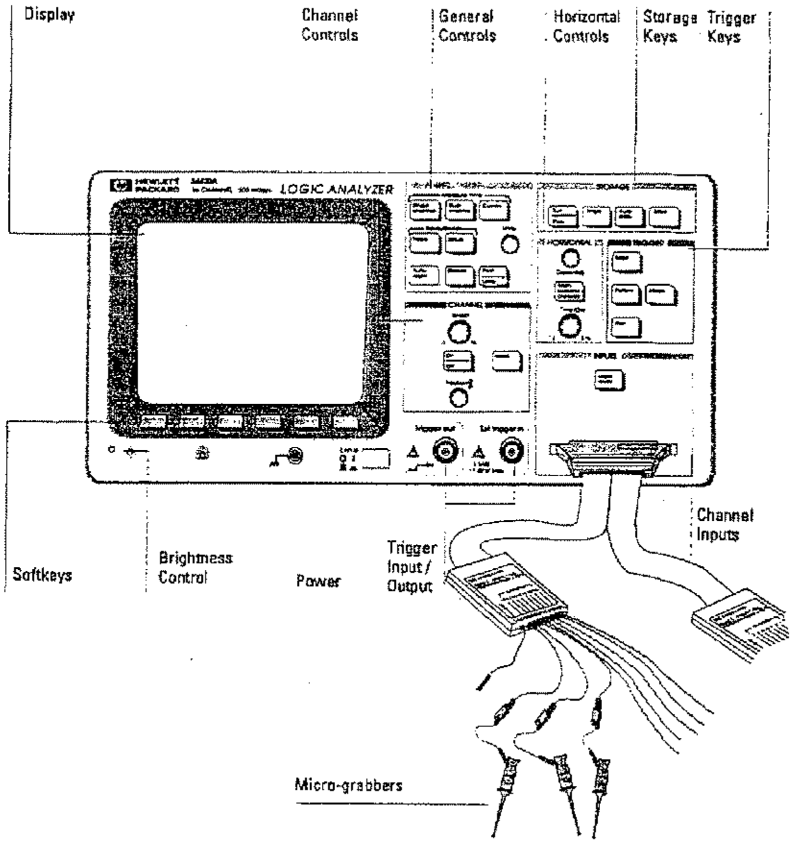


Figure 14.1 – HP56420A Front Panel and Probes

16 *channel probes* are connected to the front of the logic analyzer via a ribbon cable. The probes are grouped into two "pods" of 8 probes each. Each probe has a unique number between 0 and 15 that corresponds to its channel number. There is also a *ground probe* connected to each pod that should be attached to the ground of the circuit under test.

Each probe is connected to a *micrograbber* – a short probe that can be attached to sires in your circuit. Pressing the back of the micrograbber causes two small wires to be extended – these are used to attach the micrograbber to a wire or chip pin. **Please be gentle using the probes and micrograbbers – they are easy to break!** Always use the micrograbbers – do not insert wires directly into the probes since this may damage the springs inside the probes.

To connect a probe to a test point in your circuit, **first turn off the power**. Then, connect the micrograbber to the circuit in one of the following ways: 1) Cut and strip the ends of a short piece of wire. Plug one end into the breadboard at your test point, and attach the micrograbber to the other end, or 2) connect the micrograbber directly to a chip pin that is connected to the test point (this is a little harder). It is important to turn off the power so that you do not short out your circuit when making the connections.

The controls on the front panel of the logic analyzer can be divided into the following groups:

**Softkeys** – these are the bottom of the display and are used in conjunction with the other controls. During operation, a legend appears on the display over each softkey describing its function. This changes depending on which of the other controls are in use.

**Channel Controls** – these are used to select which channels will be displayed, set the position of each channel on the display, and assign labels to each channel. The **Select** knob is used to select one of the 16 channels. If the channel is currently being displayed, then its label on the left side of the display is highlighted; otherwise, the selected channel is shown in the upper left-hand corner of the display. The **On/Off** button is used to add or remove a selected channel to/from the display. The **Position** knob is used to move the position of a selected channel up or down with respect to other displayed channels. The **Label** button is used in conjunction with softkeys to label different channels with signal names.

**Horizontal Controls** – these control the time scale and the delay from the trigger point to the display. The **Time/Div** knob controls the time scale – turning it counter-clockwise lengthens the scale up to a maximum of 1 second/division, while turning it clockwise shortens it to a minimum of 5 ns/division. During default operation, the trigger time of each trace is shown at the center of the display, with an equal time before and after the trigger. The **Delay** knob can be used to shift where the trigger point is displayed. Its use is not discussed in this lab.

**Trigger Controls** – these control how the logic analyzer is triggered. Triggers can be specified in three ways. When the **Edge** button is pressed, softkeys allow you to select a channel and an edge type (i.e. rising, falling) as a trigger. This is similar to an oscilloscope trigger. When the **Pattern** button is pressed, softkeys allow you to specify a multiple-bit value as a trigger. Each displayed input can be specified as a High, Low or Don't Care value using softkeys. Any time the inputs match this condition, the logic analyzer is triggered. This is especially useful when debugging complex sequential circuits. The **Adv** button allows the specification of more advanced triggers and will not be discussed in this lab (see the logic analyzer manual if you are interested).

**Storage Controls** – these control how data is collected and stored. The **Run/Stop** key is used to turn the collection of data on ("run") or off ("stop"). Pressing this key once causes the logic analyzer to continuously wait for a trigger and display data each time a trigger is encountered. Pressing it a second time stops this process. The **Single** key waits for a trigger condition *once* and displays the result. This is useful when you want to see what happens in response to a single trigger event rather than repeated trigger events. The **Autostore** button places the logic analyzer in a mode where the values from previous traces are stored and displayed at half-brightness while the current trace is displayed a full brightness. This can be used to see what happens over multiple triggering events. This is similar to the function of a storage oscilloscope. The *Erase* button erases this previously stored data.

**General Controls** – these are used to set up what is displayed, control the display, and measure time between events on the display. The **Measure time** buttons allow you to measure time between events; these will be discussed later. The **Save/Recall** buttons work with softkeys; they allow you to save configurations and restore them. You will often use the **Setup** button in conjunction with the **Default** softkey to set the logic analyzer to display all channels. The **Autoscale** button is particularly useful because it selects for display all channels on which inputs are active and guesses at an appropriate time scale. This is very useful for quickly setting up the display, but it will not display channels on which there is no activity.

**Channel Inputs** – this section includes the probe connector and a **Logic levels** button that allows you to specify which logic family you will be debugging. You probably will not need to change this.

## 14.3 Prelab

The circuit shown in Figure 14.2 is known as a self-correcting ring counter. During normal operation, it goes through a counting sequence of "1000", "0100", "0010", and "0001". When the counter is initialized with an illegal value (i.e. a value not in the counting sequence), it is guaranteed to eventually reach a legal state.

1. Design a TTL implementation of the self-correcting ring counter using the parts given to you for this course. HINT: the 74LS175 quad flip-flop provides flip-flop outputs in both inverted and un-inverted form.

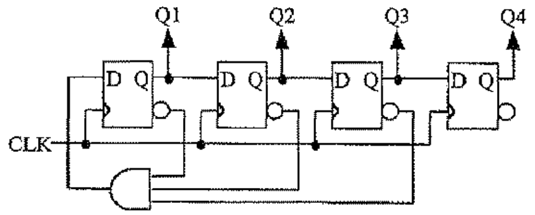


Figure 14.2 – Self-Correcting Ring Counter

2. Assume that the ring counter is initialized in an illegal state of "1111". Show the sequence of values that the ring counter goes through to reach a legal value in the counting sequence. How many clock cycles does it take?

3. Assume that the ring counter is initialized in an illegal state of "1010". Show the sequence of values that the ring counter goes through to reach a legal value in the counting sequence. How many clock cycles does it take?

4. Be familiar with the pin-outs of the integrated circuits used in the self-correcting ring counter before coming to lab.

In the second part of the laboratory, we will use a variant of the linear feedback shift register developed in the previous lab that is shown in Figure 14.3.

5. Design this circuit using parts available to you in the lab. Be prepared to build this circuit in the lab by "adapting" your ring counter circuit, connecting the switch input to a switch on the lab breadboard.

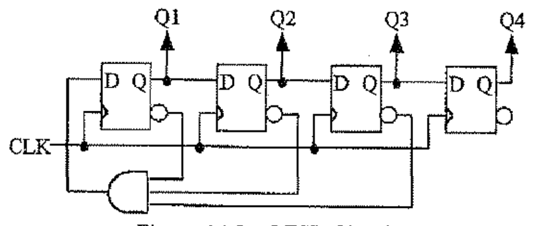


Figure 14.3 – LFSR Circuit

6. Assuming that the switch is "OFF" (0), what will be the normal sequence of states for this circuit? What happens if the circuit is initialized to all zeros?

7. Assuming that the switch is "ON" (1), what will be the normal sequence of states for this circuit? Is there a state that is analogous to the "all zero" state when the switch is on?

**14.4 In the Lab**

1. Using LED indicators and the 1-Hz clock available in the lab, build and debug your ring counter circuit and ensure that is it operating correctly.

2. With **power off**, connect probe 0 of the logic analyzer to CLK, probe 1 to Q1, probe 2 to Q2, probe 3 to Q3, and probe 4 to Q4.

3. Set your clock speed to 100 KHz and turn on the logic analyzer and power supple for your circuit.

4. Press the **setup** key and then press the **default** softkey at the bottom of the display. This will start the logic analyzer displaying all 16 channels while triggering off a rising edge on channel 1.

5. Press the **Autoscale** button. This automatically configures the logic analyzer to display only the active signals and adjusts the time scale so that most pulse widths are "reasonable" to observe.

6. Examine the notations on the top of the display. First, note that the sampling rate is displayed at the top left-hand corner, telling you how fast samples are being collected by the logic analyzer. Next, the display shows the number of time units per division for the current display. The small arrow in the center shows the "trigger point" – the point in the display where the triggering event was detected. The position of the trigger point can be shifted on the display using the **delay** control. Note that values are displayed both before and after the triggering event. This is convenient when you are interested in what happened both before and after the event and is information that you cannot get with an oscilloscope.

7. Note that the display may "jitter" and not show a fixed pattern. This is because the trigger is currently set to be the rising edge of the clock, and data is collected continuously each time the trigger is encountered. Therefore, the displayed data is not necessarily synchronized with the counting sequence of the ring counter. To examine a single triggering event, press the **run/stop** button to stop data acquisition. The train of pulses on Q1, Q2, Q3, and Q4, will probably move around on the display. Press the **single** button a few times and see what happens. You should see more data displayed, but with the counting sequence starting at different points.

8. To properly track the counting sequence in continuous mode, we need to change the trigger of the logic analyzer to a different signal. Specifically, if we trigger on the rising edge of signal Q1, we should see the signals displayed steadily even when the logic analyzer is running continuously. To change the trigger, press the **edge** button. Use the **channel select** knob to select channel 1 for a trigger source and use a softkey to specify a rising edge trigger. Press the **run/stop** button and observe what happens. You should see a steady sequence of pulses on Q1, Q2, Q3, and Q4, even when the logic analyzer is running continuously.

9. Use the **time/div** knob to adjust the time scale of the display. This knob can be used to adjust the time scale from 1 second/division to 2 nanoseconds/division. Adjust the time scale to show two full counting sequences of the ring counter. Write down the values of the sampling rate and time/division when two full counting sequences are shown and include them in your final report.

10. It is sometimes convenient to reposition signals on the display, remove some signals, and add other signals. The **channel select** knob can be used to select individual channels. The **on/off** button can be used to turn individual channels on and off. Use the **channel select** knob to select channel 0 and turn it off with the **on/off** button. Then press it again to turn it on again.

11. The **position** knob is used to reposition signals on the display. Use the **channel select** knob to select channel 0. Then, turn the position knob. Note that the position of channel0 moves up and down as you turn the position knob. Use the position knob to place channel 0 at the *bottom* of the display. Show the resulting display to your TA.

12. The logic analyzers provide *cursors* that can be used to mark events and measure time between events. To gain experience using the cursors, press the **cursors** button in the general controls. Two vertical lines will appear which are superimposed over each other at the trigger point in the center of the display. Use the **Entry** knob to move one of the cursors while leaving the other at the trigger point. Note that the delay between the cursors is shown at the bottom of the display.

13. This capability can be used to measure the delay between events in the circuit. Using the **time/div** knob, adjust the time scale so that the signals are displayed at 5ns/div and the displayed *sampling* period is 2.0 ns. Press the **cursors** button and turn the entry knob counterclockwise until the moving cursor lines up with the first rising edge of the clock

before the trigger point. This shows the *delay* of the flip-flop from when the rising edge of the clock triggers it until a valid logic value is available at the trigger point. Record this value and include it in your final report.

14. Turn the **power off** for your circuit and the logic analyzer. Modify your circuit to build the LFSR circuit shown in Figure 14.3. Reconnect probe 0 of the logic analyzer to CLK, probe 1 to Q1, probe 2 to Q2, probe 3 to Q3, and probe 4 to Q4.

15. Turn the power back on and use the **Autoscale** button to examine the channels with activity. If you see no activity, turn the switch connected to the input to the ON position, then back to the OFF position, and press **Autoscale** again. Note that when the logic analyzer is running continuously, the outputs of the LFSR will not be displayed as stable signals even if the trigger is set to the rising edge of Q1. This is because Q1 makes multiple transitions during each counting cycle and therefore may trigger at different places during the counting cycles. For better results, we will use a *pattern trigger* that triggers on a *combination* of values on different input channels. In a pattern trigger, we specify a trigger condition of either 0, 1 or don't care (i.e. either 0 or 1). To use the pattern trigger, first press the **pattern** button. Then use the **channel select** knob to select individual channels and for each channel, press the softkey marked either **high**, **Low**, or **Don't Care**. Following this procedure, set the logic analyzer to trigger when Q1=H, Q2=L, Q3=L, and Q4=L, CLK=don't care. The logic analyzer should now show the same pattern when running continuously.

16. Write down the values of Q1, Q2, Q3, and Q4 as the circuit goes through its counting sequence with the switch OFF. HINT: an easy way to mark your place while reading these values is to use the cursor described in Step 12. The resulting sequence of values should match the sequence you derived in Step 6 of the Prelab.

17. Repeat Step 16 with the switch ON. The resulting sequence of values should match the sequence that you derived in Step 7 of the Prelab.

18. It is possible that due to wiring errors, your circuit may not work properly. The logic analyzer gives you the opportunity to debug the circuit. If your circuit does not appear to display the proper values, examine a single trace by pressing the **run/stop** button to stop tracing. During each clock cycle that is displayed on the screen, examine the values of the flip-flop Q1-Q4. Are they changing in the sequence described at the beginning of the Prelab? If not, then something is wrong. Isolate the error, fix it, and try again.

19. Once your circuit is operating properly, show it to the instructor with the trigger set as described in 15.

## 14.5 Conclusion

After completing this lab, you should understand how to connect the logic analyzer to a digital circuit and configure it to trigger using either a single signal or a pattern of signals. In addition, you should understand how to adjust the time scale of the analyzer, turn signals on and off, and reposition signals. While the logic analyzer has several other capabilities, these are the most important for use during circuit debugging.

Logic analyzers provide a powerful debugging tool. However, they are not a substitute for careful design and circuit construction. During the debugging process, keep in mind that you are trying isolate some problem that keeps the circuit from working properly. This can be either a design error or a construction error. Starting with a faulty circuit, your job is to isolate where the fault occurs and determine how to fix it once it is isolated. In the following labs, you will have the opportunity to do this.

# Finite State Machines

## 15.1 Purpose

This experiment will review Finite State Machines (FSMs). Students will design and build a FSM using the classical design method.

## 15.2 Background

In Lab 13 on Counters, we used state diagrams to describe the operation of counters. State diagrams are also used to specify other sequential logic circuits that implement a specialized logic function. They allow designers to specify the behavior of the FSM (i.e. the next state and outputs) as a function of current state and inputs.
Next state values are specified by transition arrows, while outputs are specified in one of two ways. First, some output values have a fixed value in each state. Outputs specified this way are called *unconditional outputs* since they depend only on the current state and not the value of the input. Unconditional outputs are specified in a state diagram by labeling the state bubbles with the output values that will appear in each state. Figure 15.1(a) shows a two-bit binary counter with a carry output. The carry output C is true only when the counter is in state "11". FSMs that contain only unconditional outputs are called *Moore machines*.

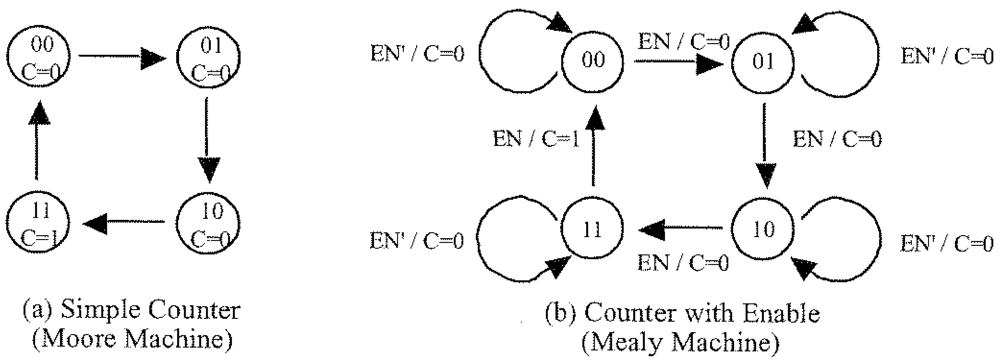


Figure 15.1 - Conditional and Unconditional Outputs in State Diagrams

On the other hand, some output values depend on an input condition during a particular state. Outputs specified this way are called *conditional* outputs since they depend on an input condition as well as the current state. Conditional outputs are specified in a state diagram by labeling the arrows between states with the condition that causes the transition and the output value that should be asserted if that condition is true. Figure 15.1(b) shows the state diagram of a two-bit counter with enable and a carry output. Since the carry output should only be true when the enable input EN is true and the current state is "11", it is a conditional output and the arrows between states are labeled with output values. FSMs that contain conditional outputs are called *Mealy machines*.

In the state diagrams we have worked with so far, we have used binary values for each state. However, it is sometimes useful to give states *symbolic names* that make it easier to keep track of what each state is doing. For example, if a FSM performs a sequence of functions, each state is given a name that refers to the function that it is currently performing. This makes the state diagram easier to understand. Figure 15.2 shows a state diagram with symbolic state names S0, S1, S2, and S3 that is similar to Figure 15.1(b).

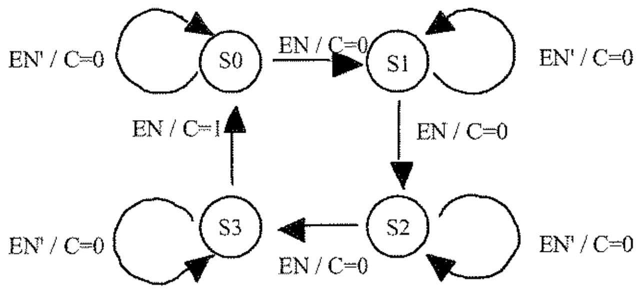


Figure 15.2 – State Diagram with Symbolic State Names

Each symbolic state name still represents a unique state value, although this value may not initially be specified. This value, called a *state code*, is supplied by the designer during a process known as *state assignment*. During state assignment, a state code is chosen for each symbolic state name. Different state assignments for the same state diagram can result in implementations that differ in cost. For this reason, computer-based tools are often used to perform state assignment for large FSMs. In smaller FSMs where cost is not a factor, arbitrary state assignments are often used.

**Implementing Finite State Machines**

Once a state diagram has been completed, the process of designing an implementation is a straightforward, mechanical process. [Mano] describes this process in Section 5-7. In short, the implementation process consists of the following steps:

1. Create a *state transition table* (STT) that corresponds to the state diagram. Each row of the STT corresponds to one transition "arrow" in the state diagram. It specifies an input condition, present state, next state, and output value. Figure 15.3(a) shows the STT that is created from the state diagram of figure 15.2.

2. (Optional) Reduce the number of states in the state diagram by combining any equivalent states (i.e. states with the same outputs and same next state transitions). Section 5-6 of [Mano] describes the details of state minimization.

3. Perform state assignment. This involves choosing the number of flip-flops to be used in the implementation (at least $\log_2$ of the number of states) and assigning a state code to each state that is a unique binary number. Arbitrary state assignments can be used on small state machines, but larger state machines may need a computer-based state assignment tool to reduce the implementation cost. After state assignment is complete, the state codes are substituted for the state names, creating a truth table for the next state and output values. For example, since the state diagram of Figure 15.2 has four states, two flip-flops are needed. Assigning the states to arbitrary state codes results in the table shown in Figure 15.2(b).

4. Choose flip-flop type to be used in the implementation. JK flip-flops sometimes result in lower-cost implementations but are more complicated to work with. State machines using D flip-flops are easier to implement and will be used in this lab.

5. Create an excitation and output logic table. The excitation and output table acts as a truth table for the excitation inputs of each flip-flop and the outputs. For D flip-flops, this is equivalent to the state transition table created in step 3. For JK flip-flops, a new table must be created that specifies the values for the J and K inputs of each flip-flop (see section 5-7 of [Mano] book for more details).

6. Simplify excitation and output logic using Karnaugh Maps, Boolean Algebra, or a computer-based logic minimizer. As discussed in ECE 218, Karnaugh Maps can be used to easily minimize logic functions with four or fewer inputs. Five and six input logic functions can be minimized using multiple Karnaugh maps, but this process is more difficult. Since the inputs to the logic functions include both state variables and FSM inputs, the excitation logic often has more than six inputs. In this case, small functions can be simplified using the rules of Boolean Algebra. For functions that are more complex a computer-based minimizer is essential.

7. Draw schematic diagram of the simplified logic and flip-flops required to implement the FSM.

| IN | PS | NS | OUT |
|---|---|---|---|
| 0 | S0 | S0 | 0 |
| 1 | S0 | S1 | 0 |
| 0 | S1 | S1 | 0 |
| 1 | S1 | S2 | 0 |
| 0 | S2 | S2 | 0 |
| 1 | S2 | S3 | 0 |
| 0 | S3 | S3 | 0 |
| 1 | S3 | S0 | 1 |

(a) State Transition Table (STT)

| IN | Q0 Q1 | D0 D1 | OUT |
|---|---|---|---|
| 0 | 0 0 | 0 0 | 0 |
| 1 | 0 0 | 0 1 | 0 |
| 0 | 0 1 | 0 1 | 0 |
| 1 | 0 1 | 1 0 | 0 |
| 0 | 1 0 | 1 0 | 0 |
| 1 | 1 0 | 1 1 | 0 |
| 0 | 1 1 | 1 1 | 0 |
| 1 | 1 1 | 0 0 | 1 |

(b) STT after State Assignment

Figure 15.3 – State Transition Table

**FSM Implementation Example**

In this section, we walk through an example FSM implementation using the state diagram shown in Figure 15.2 as a starting point. The first step is to create a state transition table (STT), which is shown in Figure 15.3(a). Since there are no equivalent states, in this state machine, state minimization is not attempted. Next, an arbitrary state assignment is performed. Since there are four states, a minimum of two flip-flops are required. If we choose two flip-flops, we can arbitrarily assign state S0 the code "00", state S1 the code "01", state S2 the code "10", and state S3 the code "11". When these codes are substituted into the STT, the result is the table shown in Figure 15.3(b). If D flip-flops are used, we can name the outputs of these flip-flops Q0 and Q1 and the inputs of these flip-flops D0 and D1. In Figure 15.3(b), we see each value of IN, Q0, and Q1 specifies a desired value for D0, D1, and OUT. Thus, this table is a truth table for logic functions producing D0, D1, and OUT. Since each of these functions has three inputs, they can easily be simplified using Karnaugh Maps, as shown in Figure 15.4. Finally, Figure 15.5 shows a schematic diagram of the FSM implementation. Note that the product term IN'*Q1 is common to two different logic functions (D0 and D1) and is reused.

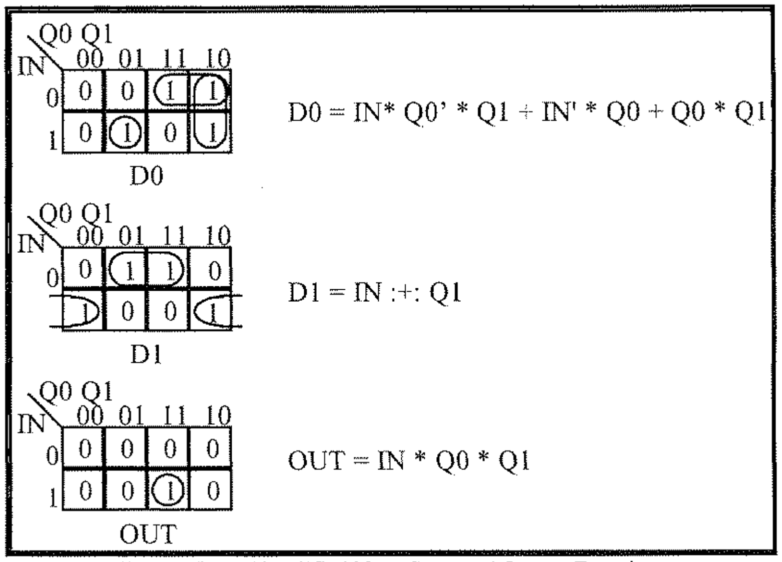


Figure 15.4 – Simplified Next-State and Output Functions

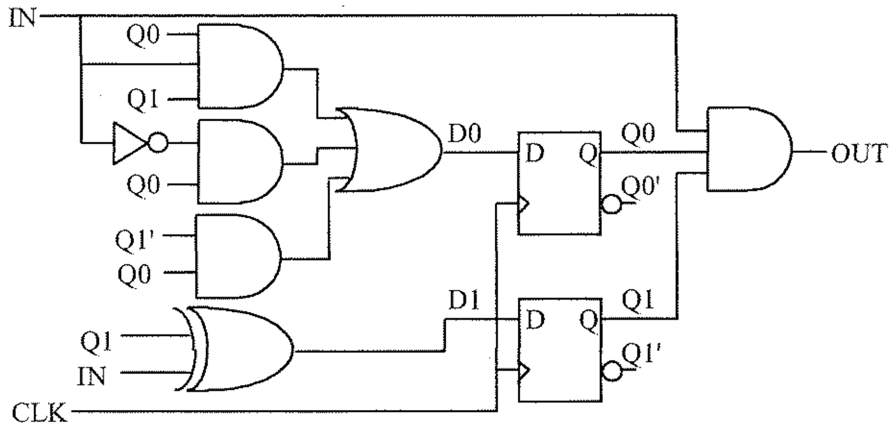


Figure 15.5 - FSM Implementation

**The Turn Signal FSM**

For the design part of this experiment, we will implement an FSM that controls a turn signal and hazard flasher for a car. As shown in Figure 15.6, the rear of a car has three lamps labeled LL, HL, and RL. The FSM that controls these lamps has two inputs, L and R and three outputs that drive lamps LL, HL, and RL. It is also connected to a clock signal (not shown) that has a period equal to the rate at which signals should flash.

When L is high and R is low, the FSM should indicate a left turn by alternately making the LL output high and low during successive clock cycles until L is no longer high.

Similarly, when L is low and R is high, the FSM should indicate a right turn by alternately making the RL output high and low during successive clock cycles until R is no longer high. When L and R are both high, the FSM should initiate a hazard flashing sequence in which LL and RL are both high, followed by LL and RL low and HL high, followed by LL, HL, and RL all low in successive clock cycles until L and R are no longer high.

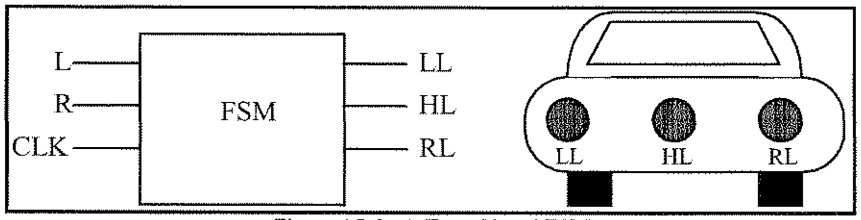


Figure 15.6 - A Turn Signal FSM

Figure 15.7 shows a state diagram for the FSM. Each "bubble" in the state diagram shows the symbolic state name and the values of outputs LL, HL, and RL, respectively. Transition "arrows" are labeled by the input condition that must be true for the transition to be taken (transitions labeled with "1" are always taken).

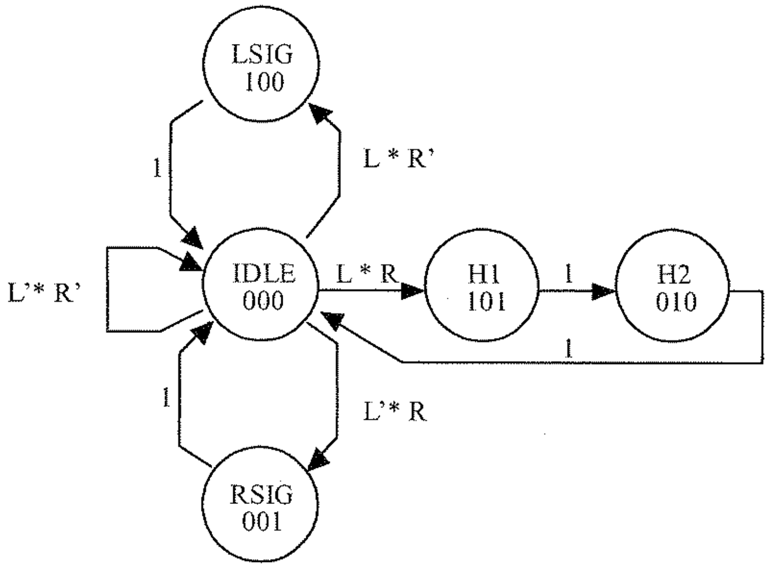


Figure 15.7 - State Diagram for Turn Signal FSM

In the IDLE state, nothing is happening and all lights are off, as shown by the output values "000". The FSM stays in the IDLE state as long as both inputs are false. If the L or R input is asserted true while the other remains FALSE, then the FSM begins a flashing sequence through states either LSIG or RSIG. After entering one of these states, the FSM returns to the IDLE state in the next clock cycle so that the signal flashes on and off. If L and R are both true, then the FSM enters the H1 state and turns on both the LL and RL outputs. In the next clock cycle, it enters state H2 and turns on the HL output before returning to the IDLE state to implement the hazard flashing sequence. It should be noted that this FSM is a "Moore machine" in which the output values depend only on the present state and not the current input values.

## 15.3 Preliminary

1. Implement the state diagram of Figure 15.2 using the state assignment S0="10", S1="11", S2="01", and S3="00". Compare the cost of this implementation to the implementation discussed in the previous section.

2. Using the procedures described in the previous section, create a State Transition Table for the turn signal state diagram shown in Figure 15.7.

3. Since there are five states in this FSM, the implementation of this circuit will require at least three flip-flops. We will implement this circuit using three D flip-flops. Assume that the flip-flop outputs are named Q0, Q1, and Q2 and the flip-flop inputs are named D0, D1, and D2. Given a state assignment of IDLE = 000, LSIG = 100, RSIG = 001, H1 = 101, and H2 = 010, write a truth table for the flip-flop excitation inputs D0, D1, and D2 and outputs LL, HL, and RL in terms of the L and R inputs and the present state values Q0, Q1, and Q2.

4. If you examine the values of the outputs in the truth table, you will see that the values of the outputs LL, HL, and RL are equal to the values of Q0, Q1, and Q2 in every state. This is done intentionally during state assignment and is called an "output-coded" state assignment. Output-coded state assignments are popular with designers because they eliminate output logic. However, they can only be used in Moore-style FSMs because the outputs cannot depend on input values. What else restricts the use of output-coded state assignments in FSM implementation?

5. Simplify the excitation logic for the turn signal FSM and design an implementation using parts available to you in the lab kit. You may wish to simplify the logic using Boolean Algebra instead of a Karnaugh Map. Draw a detailed schematic diagram of your circuit showing all connections and pin numbers and bring it to lab.

## 15.4 In the Lab

Build and debug the Turn Signal FSM that you designed in step 2 of the preliminary. Demonstrate its operation to the instructor.

## 15.5 Reference

[Mano] M. Morris Mano, *Digital Design*, Third Edition, Prentice-Hall, Inc., 2002.

# Sinusoidal Steady State Analysis

## 16.1 Introduction

Sinusoidal steady state analysis is used to examine the behavior of a circuit at a single frequency or over a range of frequencies. Mathematically, we use a technique known as *phasor arithmetic* to solve node or mesh equations of the circuit. In one method of using this technique, the elements of a circuit are assigned expressions derived from the type of element: Resistors assume the value of $Z_R$=R, Inductors assume the value of $Z_L$=jωL, and Capacitors assume the value of $Z_C$=1/jωC. You have been used to writing node or mesh equations for circuits that contains only real impedances, i.e. resistors. For circuits with reactive elements, the procedure is the same, only the solution is accomplished using complex math combined with substitution of variables, matrix techniques, or other methods of solving simultaneous equations.

Sinusoidal steady state analysis is a linear method. That is, if the magnitude of the input source is doubled, so the magnitude of the response doubles. It also ignores the transient nature of the circuit's response. Sinusoidal steady state analysis is used to describe the response that would occur in the circuit if it were excited at individual frequencies for a period of time much longer than the natural response of the solution to the circuit's differential equations.

Since the node and mesh equations for a given circuit usually involve complex numbers, it is important to pay close attention to the rules of complex math. When adding or subtracting complex numbers, it is usually best to express the numbers in their rectangular form, using real and imaginary parts. When dividing or multiplying complex numbers, it is often more convenient to express the numbers in their polar form, using a magnitude and phase. Most calculators contain a built-in function for converting complex numbers between rectangular and polar form. In manipulating the node and mesh equations, it is often best to substitute the numerical values of the components at the final step rather than the initial step of the analysis since the evaluation of complex math expressions for a range of frequencies can be tedious, mistakes often creep into the calculations. Therefore, computational tools are usually used to assist the circuit designer in evaluating the frequency response of a circuit.

Spreadsheets can be thought of as programmable calculators. Expressions can be evaluated in different cells of the spreadsheet, using values from other cells. By evaluating a single expression over a range of input values, it is easy to generate curves that relate the result of an expression, such as a complex voltage transfer function, to an input, such as frequency.

Using PSpice, one can also analyze the frequency response of a circuit by performing a computer simulation of sinusoidal steady state analysis. The PSpice simulation program simply facilitates the solving of the complex node and mesh equations. For a particular circuit, it is best that one solve the circuit equations by hand, if possible. Simulation can be sued to verify and extend the analysis. The advantage to hand analysis lies in the ability to see which components affect certain aspects of the response. Simulation only shows the response, not the analytical relationships to individual components.

One can consider that in a hand calculation, assisted by a spreadsheet, the node or mesh equations are solved by and resulting in a complex expression for the desired response, and the spreadsheet acts as a sophisticated programmable calculator. In a PSpice simulation, the circuit topology is the input to the computer program. The program then solves the mesh and node equations to produce a numerical analysis of the desired function.

**16.2 Purpose of Laboratory Procedure**

In this laboratory procedure, you will gain experience in several areas:

1. Measurements of a circuit at different frequencies.
2. Practice in the use of hand *Phasor* arithmetic to compute impedance computations and the steady state response of active network.
3. use of a Spreadsheet and PSpice simulations to confirm hand calculations.
4. Magnitude and phase measurements.
5. Examination of a lowpass response function.

Completion of the prelaboratory computations and simulations will not only save you considerable time, in the laboratory, but will also allow you to observe the direct relationship between "hand calculations", "computer simulations", and laboratory measurements. In week #1, you will work in the computer lab to gain proficiency in using the Excel Spreadsheet and PSpice simulation program to analyze a circuit. In week #2, you will analyze an active circuit that uses an operational amplifier, as well as build both circuits in the laboratory and confirm your analysis by measurements.

**Week #1**

**16.3 Preliminary**

In Figure 16.1, below, a simple RC circuit is shown which can be described by writing one node equation, at $V_{out}$. To review your proficiency with the phasor technique, solve for the transfer function $V_{out}/V_{in}$, and express the function in terms of a magnitude and phase as a function of frequency. You should obtain two different frequency-dependent expressions, one for the magnitude response and one for the phase response. Prepare your expressions for use in a spreadsheet to calculate the magnitude vs. frequency and phase vs. frequency on a plot. Refer to the tips on using a spreadsheet at the end of this procedure.

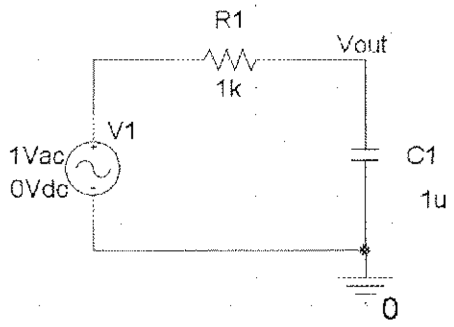


Figure 16.1 – Passive RC Circuit

Use Hz for the frequency units, recalling that f=ω/(2π), and plan on using a frequency range of 10-10,000 Hz, Use degrees for the phase units and dB (20*log(magnitude)) for the magnitude response. You should plan on obtaining a plot as shown in Figure 16.2.

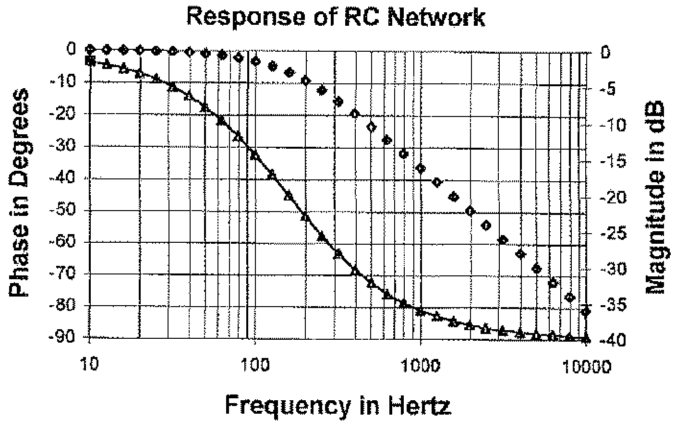


Figure 16.2 – Excel Graph of RC Network

Draw a schematic or write a netlist of the same circuit for simulation using PSpice. In the case of the netlist, use AC analysis by using the ".AC" command. You should plan to obtain a plot as shown in Figure 16.3, below. When you come to the laboratory, you should have a set of equations ready to type into a spreadsheet, and a netlist, ready to type into the PSpice program.

Observe that the magnitude of the voltage transfer function reduces at a rate approaching a factor of ten reduction for every factor of ten increase in frequency for frequencies which are above a critical frequency. This critical frequency is known as the "cutoff" or -3 dB frequency. At this critical frequency, the response is reduced by 1/√2 below its value at lower frequencies.

Observe that the shape of the phase response is such that below the cutoff frequency, the phase asymptotes towards zero, at the cutoff frequency the phase passes through 45 Degrees and above the cutoff frequency the phase asymptotes towards -90 Degrees.

Since this network would "pass" low frequencies while attenuating high frequencies, it is often called a "low-pass" network or filter. You will learn more about filters later in the semester.

## 16.4 Laboratory Procedure

Type your spreadsheet equations into the spreadsheet using the tips given at the end of this procedure. Plot a magnitude and phase curve for the passive RC network. You should obtain a plot as shown in Figure 16.2, above. Use the on-line help for the program, but do not be afraid to ask for help when needed.

Next, draw the schematic or type in your PSpice netlist into the PSpice program. Do a small signal linear AC analysis of the circuit (see .AC). Refer to the PSpice Reference

Guide in online manuals to understand the commands in the netlist. Use PROBE to view a plot of your output. Obtain a plot as shown in Figure 16.3, below.

Remember that the purpose of this week's portion of the procedure is **not** to get a plot to turn in! Rather, you should be learning how to use the computer as an analysis and simulation tool. For next week's prelab you will need to do the same type of analysis on a different circuit. Therefore, it is important that you use this week to acquire the necessary understanding and skills.

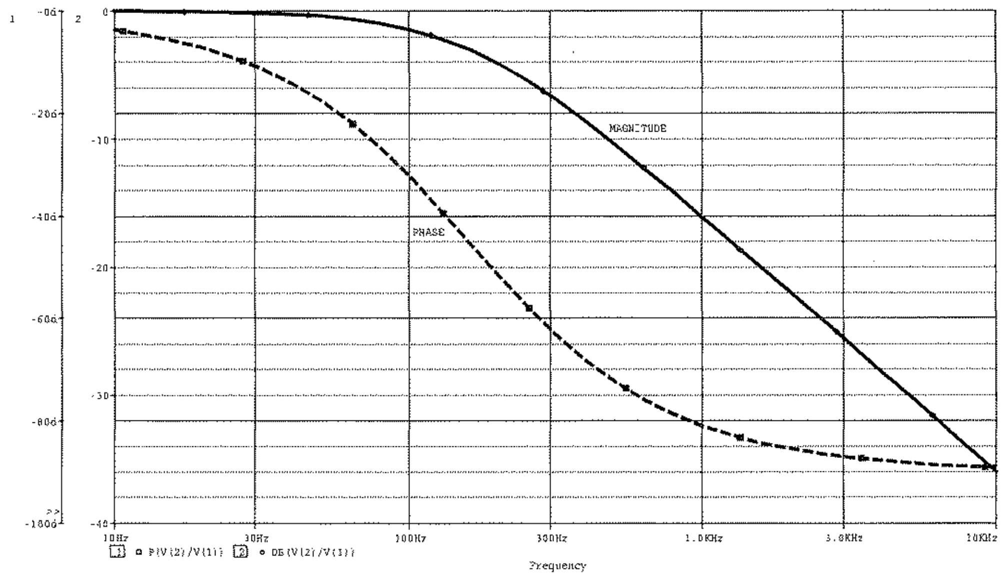


Figure 16.3 – PSpice Plot for RC Network

## 16.5 Postlab

Write a report that describes your experience in the laboratory. State the purpose of the procedure, the steps you followed in the prelab, problems which you experienced running the computer programs, and what you learned. Attach your spreadsheet and PSpice plots. You should be sufficiently confident in your ability to step through this analysis process that you will be able to accomplish the analysis for week #2. You will se your computer models and results for week #2.

**Week #2**

## 16.6 Prelab

There are other circuits, which use active componets, which also are characterized by similar magnitude and phase responses. Often such circuits are used at building blocks or "stages" to comprise a more complex system.

Solve for the magnitude and phase vs. frequency for $V_{out}/V_{in}$ for the circuit shown in Figure 16.4. Write a node equation, at $V_1$ Recall that for an ideal op amp model, the voltage at the inverting node can be set equal to the voltage at the non-inverting node, so in this case $V_1$=0- and $V_2$=0. Obtain an expression for $V_{out}$ in terms of $V_{in}$. Express it as a magnitude and phase function. Use the spreadsheet as in week #1 to solve for 10 points per frequency decade from 10 Hz to 10 kHz.

As before, draw a schematic or write a PSpice netlist for the circuit. In the case of netlist, for the op amp, use a voltage controlled voltage source (VCVS) (Exx in PSpice) with a gain of 1E6. The nodes of VCVS are Vout and ground when the controlling nodes are V1 and V2. Look at the Exx element in PSpice, referring to PSpice Reference Guide in online manuals. Obtain PROBE plots and compare them to your spreadsheet plots of magnitude and phase vs. frequency. Notice that the response is also of a "lowpass" nature.

Compare the values obtained from PSpice with those from the spreadsheet. You should have very close agreement.

Prepare an EXCEL table for taking data of magnitude and phase in the laboratory. In the laboratory you will take data at frequencies of 100, 300, 500, 1000, 3000, 5000, and 10,000 Hz.

When you come to the laboratory you should have your hand calculations, a spreadsheet plot, a PSpice plot and a blank table for the circuit analyzed in week #1 and this week's op amp circuit.

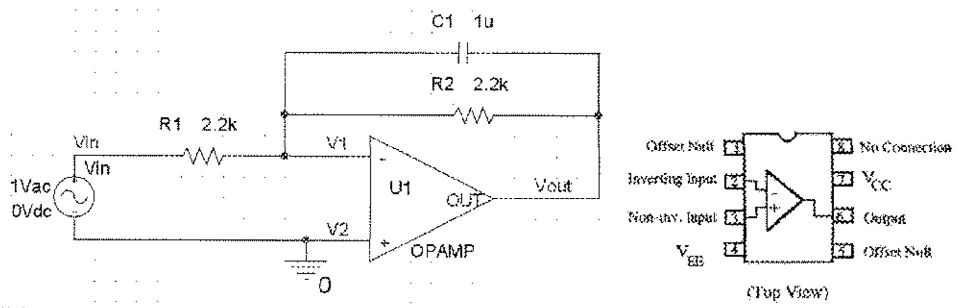


Figure 16.4 – Active Low-Pass (with Perspective View of the MC1741 Op Amp and the Pin Arrangement)

## 16.7 Laboratory Procedure

Using a protoboard, build the passive RC circuit shown in Figure 16.1.

Connect your circuit to the function generator indicated.

Use channel #1 of the oscilloscope to display the output of the function generator that is connected to the input of your circuit. Use channel #2 of the oscilloscope, in dual trace mode, to simultaneously display the output $V_{out}$. Adjust the function generator and oscilloscope to obtain stable waveforms at 1 kHz, with a 1 Volt peak-to-peak input to the circuit. Quickly adjust the frequency down to 100 Hz and up to 10 kHz to verify that the circuit is operating as expected and is characterized by the "lowpass" function. For this type of measurement, it is best to externally trigger the scope from the function generator.

Make precise measurements of the peak-to-peak output voltages $V_{out}$ and $V_{in}$ as well as the phase difference between them for the frequencies listed on your blank data table. Recall that phase differences can be determined by setting the "zero" position for each oscilloscope channel to the same horizontal line (e.g. the centerline) and measuring the time difference between the zero-crossing of the two waveforms. You will find that the Cursor controls on the oscilloscope can be helpful to measure the time differences. Quickly check to see that your data roughly agrees with your calculations and simulations. If time permits, take additional frequency data points.

Next, build your second circuit, Figure 16.4. Refer to the data sheet for the 741-type op amp to identify the correct pin numbers on the integrated circuit that correspond to those on the schematic.

As for the first circuit, check to see that the circuit is operating properly by quickly sweeping the frequency from a low value to a high value, checking for the lowpass function response. Do not take frequency response data unless you are certain that the circuit is functioning properly. Do not forget to turn on and properly adjust the power supply!

## 16.8 Postlab

Plot your laboratory data points, for both circuits, in your spreadsheet program and compare them to the calculated points that you obtained in the prelab. Write a laboratory report that describes the procedures, observations, and problems that you experienced during prelab and during the laboratory session.

## 16.9 Tips on Using a Spreadsheet to Calculate Frequency Response Functions

Create a column of frequencies that will produce equally spaced point when plotted on a logarithmic scale. This can be facilitated by creating a series of points from 0.1-0.9 in increments of 0.1. In the next column, use the first column as the exponent, x, for the function $10^x$. When you plot these points, later, on a log scale they will produce equal increments.

Use the second column of frequencies as input variables for your expressions of magnitude and phase which you derived by hand. Express the magnitude in the form of decibels, i.e., $X_{dB}=20\log(x)$, where x is the value of the magnitude function.

Derive and calculate a function for the phase of the response. Note that spreadsheets and calculators normally use an arctangent function that produces results only in the range of $-\pi/2$ to $\pi/2$. Spreadsheets have a special function called ATAN2 that provides results in the range of 0 to $\pi$. Pay close attention to the sign of the resulting function. Depending upon your expression for the phase angle, you may have to multiply the angle by -1. Calculate your angle in degrees.

On the same plot, display both the magnitude and phase, using a line plot. Use the second axis function to accomplish this.

# Power Measurements and Power Factor Correction

## 17.1 Introduction

The dissipation of power in circuit elements is a major factor to be considered in power distribution and circuit design. The familiar expression P=V*I is used to calculated the power dissipated in a circuit element characterized by a real value of impedance, i.e. a resistor.

Often the heat which results from power dissipated in a resistor is more significant to the designer than the instantaneous value of the power. In examining the heat, the integral of the instantaneous power – the average power, is more significant. This is because the average power dissipated in the resistor must be supplied by the independent sources in the circuit, impacting their size, and that the temperature increase resulting from short time periods of power dissipation can be small. The average power is computed by integrating the product of the instantaneous voltage and the instantaneous current waveforms, then dividing by the integration period.

For voltages and currents that are constant, i.e. DC values, the instantaneous power and the average power are identical. For time-varying voltages and currents, the instantaneous power and the average power differ due to the integration process. As an example, consider the integral of a sinusoid compared to the instantaneous values of the sinusoid.

For resistors, the voltage and current resulting form sinusoidal excitation are in phase. Therefore is it accurate to compute the root-mean-square (RMS) values for the voltage and current separately and then multiple the RMS current and RMS voltage to obtain the average power. The product of the RMS values is called the effective power, as distinguished form the average power.

For capacitors and inductors, a sinusoidal voltage results in a current that is 90 Degrees out of phase with the element voltage. Consequently, the average power dissipated in capacitors and inductors is zero. This can easily be seen if considering the average power for a sinusoidal excitation. However, for these elements, the effective power is not equal to zero.

A simple method of relating the average power to the effective power for any circuit element is to define their ratio. This ratio is called the power factor (PF). For sinusoidal voltages and currents, the PF can be derived as being equal to the cosine of the phase difference between the voltage and the current. Obviously, for capacitors and inductors the PF is zero. For combinations of resistors and, capacitors or inductors, the impedance is complex and the value of the PF will range from 0 to 1.

It is easy to see mathematically that, for sinusoidal excitation, the average power in complex impedance can be determined by first measuring the peak values of the voltage and current, and dividing each by $\sqrt{2}$ to obtain the RMS values, Second, measuring the phase shift between the voltage and the current and using this phase shift to compute the

PF, And third, multiplying the RMS voltage, the RMS current, and the PF to obtain the average power. All of these measurements may be made using an oscilloscope.

For loads that are not entirely resistive, but contain a reactive component, there is a penalty in supplying power to the load. This is because the PF is less than 1 and to obtain same average power for a fixed excitation voltage, the effective value of the current must be higher than it would be for PF=1. This additional current results in increased power dissipation in the lumped resistance consisting of the power line resistance and the Thevenin equivalent of the source.

A common example of this situation occurs when supplying power to motors. A simple model for a motor is an inductor in series with a resistor. The resistor results from two components – the equivalent resistance caused by the mechanical load on the motor, and the motor winding resistance from the wire. Since the PF for the series R-L combination is less than 1, increased power losses occur at the source and within the power distribution system. A capacitor placed in parallel with the R-L load branch can cancel the imaginary portion of the R-L impedance and return the PF close to 1. A further description of this technique can be found in section 9.5 of [Irwin].

In this laboratory procedure, you will analyze the average power dissipated in a circuit model for a motor. You will calculate a suitable capacitor to be placed in parallel with the R-L model for the motor that will return the PF to 1. You will simulate the two circuits, with and without the PF correction. You will then build and measure the circuit in the laboratory.

## 17.2 Prelab

For the following circuit, compute the effective power dissipated in each circuit element. Perform a "power audit" in which you show that the total circuit power dissipated equals the power supplied by the source $V_s$. First, perform this computation for the circuit without the power factor correcting capacitor. Next, calculate a value of capacitance in place in parallel with the R-L load to correct the PF to 1. Then perform the energy audit again for the new circuit, which includes the capacitor.

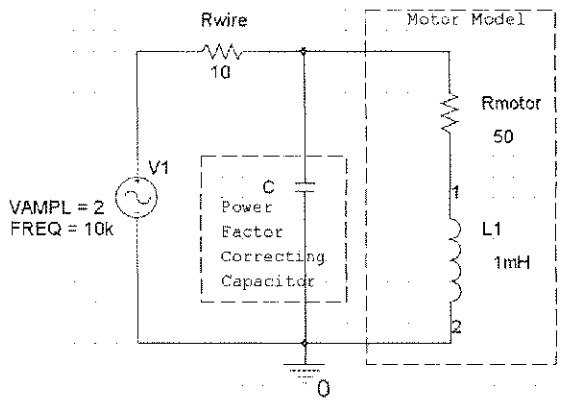


Figure 17.1 – Motor Model

Simulate the original and power factor corrected circuit on PSpice. Use transient analysis at 10 kHz. Be certain to force your simulation to calculate sufficient points per cycle to obtain a smooth curve. This can be accomplished by specifying the 4th parameter in the .TRAN statement. Usually a 50-100 point per cycle is adequate.

In PROBE, you can display the average and effective power for any element. Note that PROBE calculates a running average, and a running RMS value. Therefore, you need at least ten cycles of transient simulation to obtain a measure of the steady state value. When adding a trace in PROBE you may ask for the average value of an expression by using the "AVG( )" function. Similarly you may obtain the effective value by using the "RMS( )" function. For example an expression for the average power might be: AVG (V(1)*I(VS)). And, an expression for effective power might be: RMS(V(1))*RMS(I(VS)). Try this technique on a simple resistive-only circuit to see how to use these functions.

Use PSpice to verify your hand calculations and your power audit for the original and corrected circuit. You should come to the laboratory with your hand energy audit and your PSpice simulations that confirm your hand calculations for each of the two circuits. That is, for the circuit with and without the power factor correcting capacitor.

## 17.3 In the Lab

In the laboratory, you will build the original circuit, without the power factor correction. You will then use the oscilloscope to make peak measurements of voltage and current for each element in the circuit. You will then compute the average values of power and compare to your prelab power audit.

To measure current you will use a device known as a current transformer. This transformer is a 1:1000 turn transformer wound on a toroid core. When terminated with a 1000 Ω resistor, the scale factor is 1 V/Amp. You use the transformer by inserting a wire in which you wish to measure the current through the center hole of the transformer. The oscilloscope is connected across the 1000 Ω terminating resistor. The current is displayed as a voltage waveform on the oscilloscope using the 1 V/Amp scale factor. In using the transformer with your circuit, you will need to move its location within the circuit as necessary to measure the current through each of the components. Note that the load current can be measured in the original circuit by leaving the transformer at one location.

Be certain to adjust the frequency by direct measurement on the oscilloscope rather than reading off the function generator dial. Adjust the frequency for 10 kHz and measure the necessary voltages and current.

Be careful not to connect the ground side of the oscilloscope to any point other than your circuit ground. To measure voltage across the resistors, you may use the two channels of the oscilloscope in differential mode, for which you invert channel two and set the oscilloscope to "add". Your instructor will demonstrate this technique. You will need to use this technique only for the resistors, since the other circuit elements have each have one side connected to circuit ground.

Observe the peak values of voltage and current, and convert them to RMS values. Measure the phase different between the voltage and current for each measurement. You may assume that for the resistors, this phase difference is zero. Measure the R-L load as a single branch, and then measure the individual components. Use the phase angle measurements to convert the effective power to the average power. Compare to your hand and PSpice calculations.

Next, add the PF correcting capacitor. Perform the power measurements as before. For this circuit, treat the R-L-C network as a single element as well as individual elements. Compare your average power computations in the lab to those obtained from the prelab.

## 17.4 Postlab

Discuss the significance of the two different circuits. For the corrected circuit, did you observe the predicted improvement in supplied power from the source $V_s$? Write a report that directly compares hand, PSpice and lab computations.

## 17.5 Reference

[Irwin] J. David Irwin, *Basic Engineering Circuit Analysis*, seventh edition (a Wiley first edition), pp. 359, John Wiley & Sons, Inc., 2002.

# Sequential Logic Design with PLDs

## 18.1 Purpose

In this lab, you will design and use sequential circuits and FSMs with programmable logic devices (PLDs). The concepts of digital system datapath and controller elements are introduced. To illustrate these concepts, you will design and build a simple "ping-pong" game using a FSM and a specialized shift-register.

## 18.2 Background

In previous experiments, we used programmable logic devices (PLDs) to synthesize combinational logic. PLDs with "registered" outputs (i.e. with flip-flops on their outputs) can be used to implement sequential circuits. Figure 18.1 shows the basic structure of a registered PLD. Sum-of-product outputs from the OR array are connected to flip-flop inputs. The flip-flop outputs are connected to output pins and are internally fed back to the AND array. Thus, each flip-flop can be programmed to be a function of both inputs and current flip-flop outputs, making PLDs ideal for implementing sequential logic circuits. PLDs with configurable output "macrocells" (as the PALCE22V10 used in this lab) can be set to use both registered AND combinational outputs.

To program sequential PLDs using ABEL, logic equations are written in the same way that they are for combinational circuits except that the "=" combinational assignment operator is replaced with the sequential assignment operator ":=". This specifies that the value on the right hand side of the equation will be latched into the flip-flop at the rising edge of the clock.

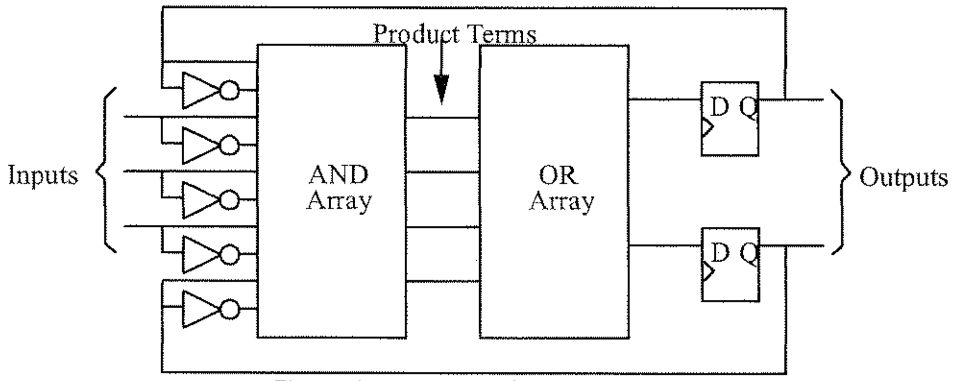


Figure 18.1 - Structure of a Sequential PLD

```
MODULE ShiftReg
"Inputs
CLOCK, S0, S1, LIN, RIN, A, B, C, D PIN;
"Outputs
QA,QB,QC,QD PIN ISTYPE 'REG';

EQUATIONS
QA:= (!S0 & !S1 & QA) # (!S0 & S1 & RIN) # (S0 & !S1 & QB) # (S0&S1&A);
QB:= (!S0 & !S1 & QB) # (!S0 & S1 & QA) # (S0 & !S1 & QC) # (S0&S1&B);
QC:= (!S0 & !S1 & QC) # (!S0 & S1 & QB) # (S0 & !S1 & QD) # (S0&S1&C);
QD:= (!S0 & !S1 & QD) # (!S0 & S1 & QC) # (S0 & !S1 & LIN) # (S0&S1&D);

QA.CLK=CLOCK; QA.OE=1;
QB.CLK=CLOCK; QB.OE=1;
QC.CLK=CLOCK; QC.OE=1;
QD.CLK=CLOCK; QD.OE=1;

TEST_VECTORS
([CLOCK,S0,S1,RIN,LIN,A,B,C,D] -> [QA,QB,QC,QD])
[.C.,1,1,0,0,0,0,0,0] -> [0,0,0,0];
[.C.,0,1,1,0,0,0,0,0] -> [1,0,0,0];
[.C.,0,1,1,0,0,0,0,0] -> [1,1,0,0];
[.C.,1,0,0,1,0,0,0,0] -> [1,0,0,1];
[.C.,1,0,0,1,0,0,0,0] -> [0,0,1,1];
[.C.,0,0,0,0,0,0,0,0] -> [0,0,1,1];
[.C.,1,1,0,0,1,0,1,1] -> [1,0,1,1];

END
```

Figure 18.2 - ABEL File for a Four-Bit Universal Shift Register

Figure 18.2 shows an example of a sequential logic circuit implemented using ABEL. This circuit implements a four-bit "universal shift register" similar to the 74LS194 TTL part. Registered outputs QA, QB, QC, and QD are used as the shift register's flip-flops. Depending on the values of the select signals SO and S1, the circuit will either hold its current value, shift data left using LIN as an input, shift data right using RIN as an input, or load a new value into all four flip-flops using the A, B, C, and D inputs. The EQUATIONS section specifies the logic equations for each flip-flop.

PLDs can be used to implement FSMs using the methods described in the "Finite State Machine" lab to translate the state diagram into excitation logic for the flip-flops and writing equations for this logic using the ":=" operator. However, ABEL supports a more direct method of generating implementations directly from state diagrams.

In the direct implementation method, ABEL allows the user to specify state variables, a state assignment, state transitions, and FSM outputs. Figure 18.3 shows an ABEL specification for the two-bit counter with enable described in the "Finite State Machines" lab.

Digital systems are often divided into two parts: the *datapath*, which stores and manipulates information, and the *controller* - a FSM that sequences the operation of the datapath. Each of the datapath elements has *control lines* that determine which function it will perform. In addition, it may have one or more *status lines* that represent a particular condition. The controller FSM has outputs that connect to the control lines of datapath elements, while status lines become FSM inputs. One metaphor for thinking about how the controller and the datapath work together is that of a puppeteer with a marionette puppet. The puppeteer (controller) pulls the strings to make the puppet (datapath) perform certain functions.

```
MODULE TwoBitCounter
"Inputs
CLOCK, EN PIN;
"Outputs
Q0, Q1 PIN ISTYPE 'REG';

QSTATE=[Q0,Q1];
S0=[0,0];
S1=[0,1];
S2=[1,0];
S3=[1,1];

STATE_DIAGRAM QSTATE
STATE S0: IF (EN) THEN S1 ELSE S0;
STATE S1: IF (EN) THEN S2 ELSE S1;
STATE S2: IF (EN) THEN S3 ELSE S2;
STATE S3: IF (EN) THEN S0 ELSE S3;

EQUATIONS
QSTATE.CLK=CLOCK; QSTATE.OE=1;

TEST_VECTORS
([EN, CLOCK] -> [Q0, Q1])
[1, .C.] -> [0, 0];
[1, .C.] -> [0, 1];
[1, .C.] -> [1, 0];
[1, .C.] -> [1, 1];
[1, .C.] -> [0, 0];
[0, .C.] -> [0, 0];
[1, .C.] -> [0, 1];

END
```

Figure 18.3 - ABEL File for a Two-bit Counter with Enable

To illustrate this idea, consider the following design problem: We wish to design a simple "Ping-Pong" game shown in Figure 18.4. Eight LED's represent the location of the ball on a ping-pong table, while two switches represent the two paddles. Initially, the ball is placed in the leftmost position and the left player "serves" by pressing the "left paddle" button. The ball moves from left to right until it reaches the rightmost LED. If the right player presses the "right paddle" button at the same time that the ball reaches the rightmost position, the ball "bounces" back to the left side. On the other hand, if the right player presses the button too early or too late, he or she "misses" the ball, which goes back to the leftmost position for another serve by the left player. Similarly, when the ball bounces back to the leftmost position, the left player must press the "left paddle" button at exactly the right time to hit the ball back to the right side. If he or she misses, then the ball goes to the rightmost position for a serve by the right player.

Figure 18.5 shows a possible implementation of the ping-pong game using a controller FSM and a datapath consisting of a special-purpose shift register. The outputs of this shift register are tied to LEDs to display the ball position. Control lines STARTL and STARTR set the shift register to display the ball in the leftmost and rightmost positions, respectively. SHIFTL and SHIFTR control the direction of shifting in the shift register. The FSM has inputs from the left paddle (LP) and right paddle (RP) switches and status line inputs from the datapath that specify that the ball is in either the leftmost (LLAMP) or rightmost (RLAMP) positions.

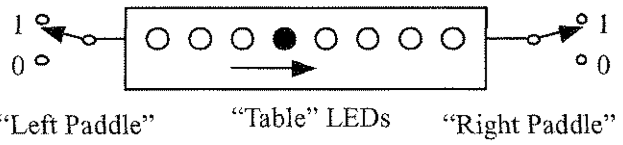


Figure 18.4 - "Ping-Pong" Game

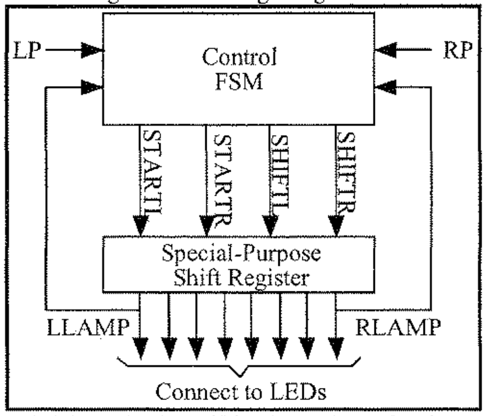


Figure 18.5 Organization of Ping-Pong Game Implementation

Figure 18.6 shows a state diagram for the control FSM *without* specifying the outputs. There are four states in this FSM. In state STOPL, the ping-pong game is waiting for the left player to serve by pressing the LP button. When this happens, it shifts to the RUNR state, which moves the ball from the left to the right until it reaches the rightmost position (as specified by the RLAMP input). If the right player presses the RP button at exactly this time, it goes to state RUNL to move the ball back to the left side. Otherwise, it goes to the STOPL state since the right player "missed the ball". The STOPR and RUNR states work in a similar way, but in the opposite direction.

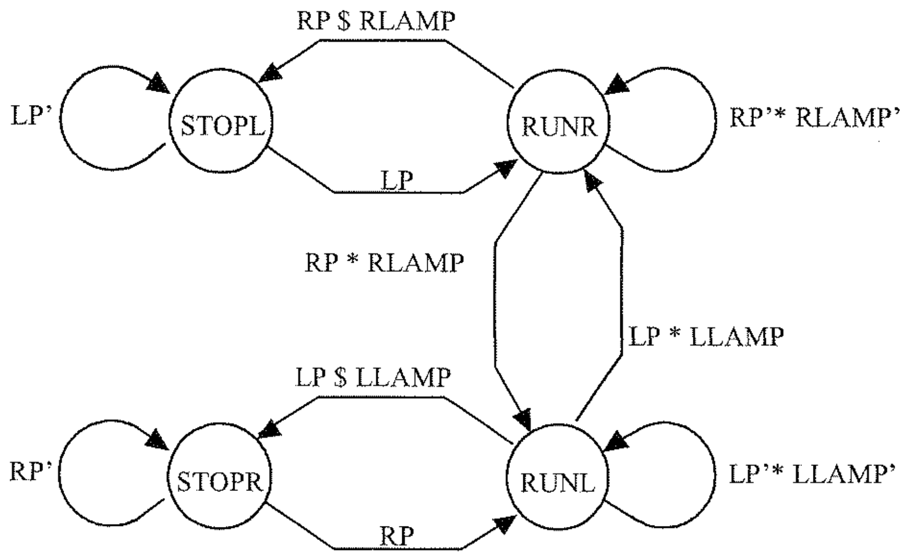


Figure 18.6 - State Diagram for Ping-Pong Game

**18.3 Preliminary**

1. Review the ABEL file for the universal shift register shown in Figure 18.2 and write a "function table" that describes what function it performs for each value of S0 and S1.

2. By using a PLD, design an 8-bit specialized shift register for use in the "Ping-Pong" game. This shift register should have four inputs: STARTL, STARTR, SHIFTL, and SHIFTR. When "STARTL" is true, the shift register should be loaded with the value "10000000" to indicate that the ball starts on the left side of the table. Similarly, when "STARTR" is true, the shift register should be loaded with the value "00000001" to indicate that the ball is on the right side of the table. When SHIFTL is true, the shift register should shift left. When SHIFTR, the shift register should shift right. You should be able to derive the equations for the shift register by inspection after studying Figure 18.2. Simulate your ABEL description to show its proper operation.

3. The control line outputs of the FSM are not specified in Figure 18.6. Determine the proper values for the control lines and label the state diagram with the proper output values.

4. By using a PLD, design an implementation of the FSM including outputs. Simulate your ABEL description to show its proper operation.

5. Draw a schematic diagram showing how the specialized shift register and FSM should be connected to implement the state machine. Be sure to show all pin numbers.

6. When power is turned on in a sequential circuit, the flip-flops may hold random values. Explain what will happen to your circuit when it is turned on if it is initialized to a random value.

**18.4 In the Lab**

1. Show the instructor your simulation for the specialized shift register and have him or her program the PLD for you. Test the circuit in the breadboard and demonstrate its proper operation to the instructor.

2. Show the instructor your simulation for the control FSM and have him or her program the controller PLD for you. Test the circuit in the breadboard with the specialized shift register and demonstrate the proper operation of the ping-pong circuit to the instructor.

# Frequency Response of Active Networks

## 19.1 Introduction

Filters are often used to modify and processes analog signals. Filters are needed in electronic systems to amplify certain frequency components of a composite signal and to attenuate other frequency components. A common example of the use of filters is the tone control on a stereo receiver. A low-pass filter is one that amplifies or "passes" low-frequency components of an audio signal. This would correspond to the "bass control" on a stereo amplifier. A high-pass filter is one that amplifies or passes high-frequency components of an audio signal. This would correspond to the "treble control" on a stereo amplifier. As the bass and treble controls are adjusted, the shape of the corresponding filter voltage transfer function is changed, and the range of frequencies that are passed is similarly changed.

The low-pass filter is used extensively for many electronic circuit applications. You will examine a low-pass filter topology in this laboratory procedure.

The ideal low-pass filter would pass all frequencies below its "cutoff frequency" and completely attenuate all frequencies above it cutoff frequency and the corresponding magnitude response curve would appear as in Figure 19.1. Unfortunately, this ideal desired response cannot be obtained with electronic circuits. Therefore, engineers use different approximations to the ideal response to implement their system designs. For these approximation functions, the rate of attenuation above the filter's cutoff frequency is not abrupt, as in the ideal case. Rather the attenuation rate can be controlled by using a 1st, 2nd, 3rd, or nth order polynomial approximation function. A circuit can then be built whose transfer function matches the desired polynomial.

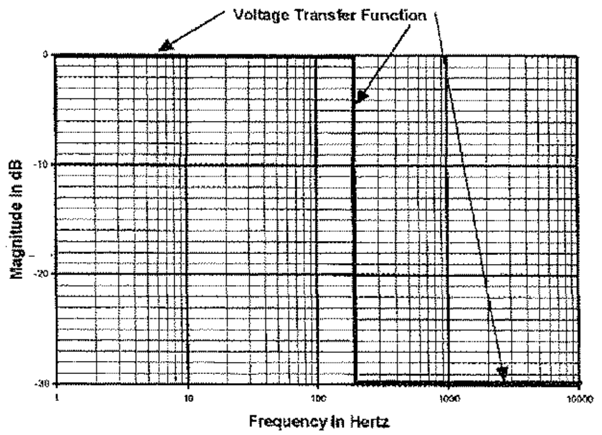


Figure 19.1 – Ideal Low Pass Filter Response

The rate of attenuation for a particular polynomial function depends directly upon the order of the polynomial. For example, a first order polynomial will attenuate frequencies above the cutoff a factor of 10 for each factor of 10 increases in frequency. We usually plot such responses on a log frequency scale as shown in Figure 19.2.

For the first order function, the rate of attenuation, or drop-off, would be 1/10 for each decade of frequency, or 20 dB/decade. For a second order filter, the drop-off rate would be 40 dB/decade, or a factor of 1/100 for each decade increase in frequency. Similarly, a third order filter would drop-off at 60 dB/decade. In general, the higher the order of the filter polynomial, the closer the approximation will be to the desired ideal, abrupt filter response. In practice, the order of the filter is limited by economic and complexity considerations. A higher order filter is more costly because it must use more components to achieve the higher order response. Typically, one reactive element, such as a capacitor, must be added to the circuit for each power-of-ten increase.

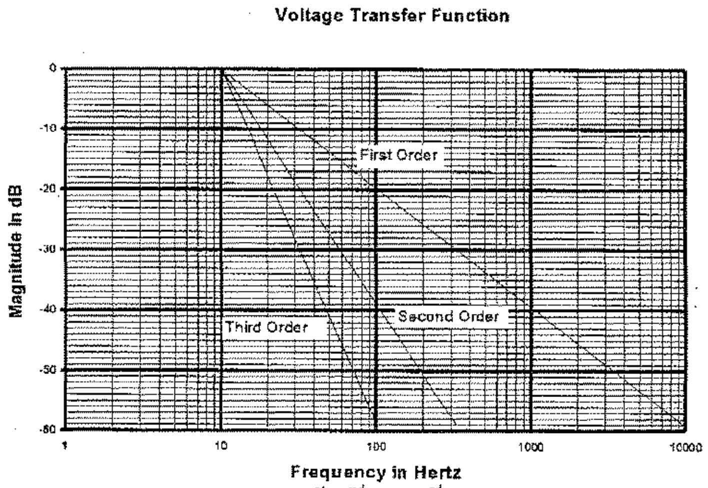


Figure 19.2 – Response of 1st, 2nd, and 3rd Order Low-Pass Filters

There is a wide range of nth order polynomials that can be used to implement filter functions. Circuit designers have historically used three or four different ones to achieve a range of approximations to the ideal response. Two of these are called "Max-flat Butterworth" and "Chebyshev", named after the individuals who recognized their properties. The Butterworth function better approximates the ideal response in the region below the cutoff frequency by keeping the response "flat". The Chebyshev function better approximates the ideal response at, and just above, the cutoff frequency by providing a faster drop-off rate, compared to the Butterworth.

In this laboratory procedure, you will design, analyze, construct and measure two low-pass filters. Each will have the same cutoff frequency, but will use different polynomial functions, Butterworth and Chebyshev. You will see the effect of changing a circuit parameter called "Q", or quality factor, in selecting which function is used.

## 19.2 Prelab

In this laboratory procedure, you will design two 2nd-order filter stages to implement the Butterworth and Chebyshev 2nd-order functions.

A second order voltage transfer function for a low-pass filter can be expressed as:

$$\frac{V_2(s)}{V_1(s)} = \frac{H_0\omega_n^2}{s^2 + (\omega_n / Q)s + \omega_n^2}$$

The parameters of $\omega_n$ and Q control the shape of the response and the type of the 2nd-order function that is implemented. For a 1 kHz cutoff Butterworth filter, $\omega_n$=6283 radians/sec and Q=0.707. For a 1 kHz cutoff Chebyshev filter, $\omega_n$=5282 radians/sec and Q=1.306. Use the circuit and its equations below, and design a 1 kHz Butterworth and a 1 kHz Chebyshev low pass filter. Use PSpice to analyze your designs and verify the shape of the filter function. Come to lab with your computations, simulations, and circuits so that you may use the lab time to measure your circuits.

$$\omega_n = 1/RC$$
$$1/Q = 3 - K$$
$$H_0 = K$$

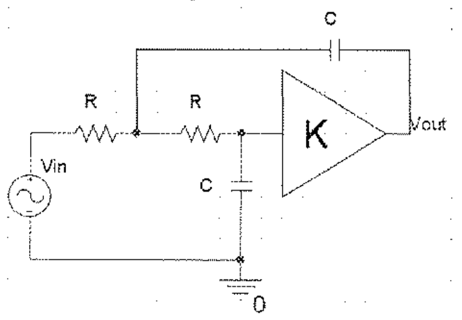


Figure 19.3 – Sallen-Key Low-Pass Filter

Note that for this circuit, both resistors are equal and both capacitors are equal. $H_0$, is not a free parameter but is constrained to the value of K. The K-block amplifier is a non-inverting amplifier with a gain equal to K, and can be implemented using an op-amp, with two resistors, in a non-inverting amplifier configuration. You may need to look up the non-inverting amplifier topology.

## 19.3 Laboratory

In the laboratory, you will build each of the two filters. Measure the magnitude vs. frequency and phase vs. frequency response. Be certain to select an appropriate range and density of frequency measurements to observe the differences between the two circuits. Plot on log graph paper, or use a spreadsheet to make your plots.

## 19.4 Postlab

Compare your Prelab calculations and simulations to the lab data. What is the difference between the two filter responses? Comment on the effect of the Q on the filter response.

# Transformer Principles

## 20.1 Introduction

You have learned about the concept of a mutual inductance. A mutual inductance can be used to model 2 inductors which share a common magnetic path. Often, this combination of circuit elements is referred to as a "transformer".

There are many uses for transformers. Transformers are frequently used to convert a higher voltage to a lower voltage, such as in the power supplies of computers, or on the utility pole near houses. Transformers can also be used to convert a lower voltage to a higher voltage, such as in the high-voltage power supply of a computer monitor, in which 50 V is converted to 10,000 V.

Another application for transformers is called impedance matching. Often it is necessary to supply power to a load whose value is much lower than the Thevenin equivalent resistance of the source. One familiar example of this situation is impedance matching for HiFi stereo speakers. Frequently a transformer is used in the output section of the stereo amplifier to match its impedance to the 8 Ω impedance of the speaker.

You have learned about the linear and ideal transformers models. The linear transformer may be considered an ideal transformer if the value of the mutual inductance is large compared to the self-inductances, and the magnitudes of the self-inductances are large compared to the source and load resistances. You may review these concepts in Chapter 8 of [Irwin]. Note that for the ideal transformer, the voltage transfer function may be derived as being equal to the turns-ratio. It may also be shown that a resistance placed as a load across the secondary winding of an ideal transformer "reflects" into the primary winding by a multiplicative factor equal to the square of the turns-ratio. Review Chapter 8 of [Irwin] to better understand these concepts.

In this laboratory procedure, you will construct a linear transformer that can be used as an approximation to an ideal transformer. You will see how the turns-ratio can be used to predict the voltage transfer function. You will also measure the effects of impedance matching on power transfer from a source.

## 20.2 Preliminary

Review Chapter 8 of [Irwin]. In particular, read section 8.3 on ideal transformers. In this laboratory, you will make a 2:1 turns-ratio transformer.

First, consider the following circuit:

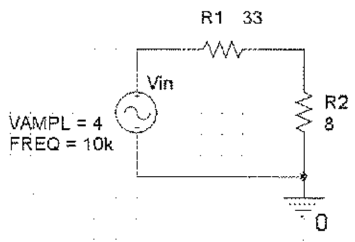


Figure 20.1 – Resistive Divider

Calculate the power that would be dissipated in the 8 Ω load resistor.

Next consider the modified circuit which uses a 2:1 impedance matching transformer:

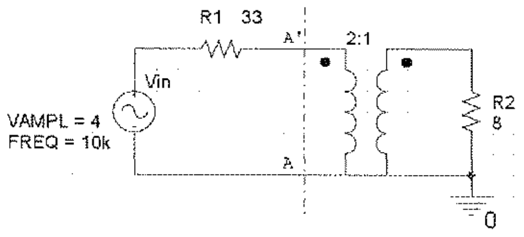


Figure 20.2 – Resistive Divider with Impedance Matching Transformer

At the terminals A'-A of the transformer what is the reflected value of the load resistor? Using this value calculate the power which would be dissipated in the 8 Ω resistor. Why is the power dissipated in the load different for this circuit from for the previous circuit? For both circuits what is the power supplied by the source?

When you come to lab, bring an analysis of the two circuits that shows the benefit of using the impedance matching transformer in terms of the power supplied by the source and the power delivered to the load.

**20.3 In the Lab**

In the laboratory, you will wind wire on a core to fabricate the impedance matching transformer. You will use a type of core known as a toroid. This core is circular in shape. The wire should be wound on the core as shown in the figure below.

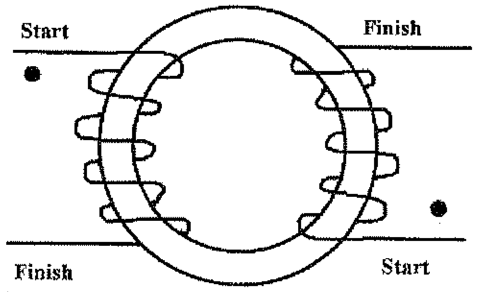


Figure 20.3 – Winding Configuration of Transformer

1. First wind the primary. Wind exactly 50 turns on the core. Examine the laboratory model to see the physical configuration of the winding. Identify the start and finish of the

winding. The distribution of the wire around the core is not critical to the transformer's function. Next wind the secondary. Wind exactly 25 turns on the core. Refer to the model. Identify the start and finish the winding.

Verify operation of your transformer by connecting it in the circuit as shown below:

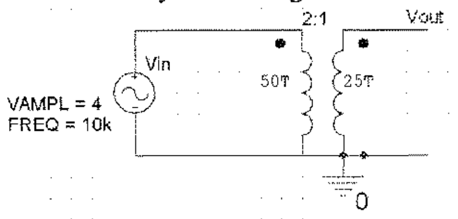


Figure 20.4 – Test Circuit for Transformer

Measure the voltage transfer, magnitude and phase, from the primary to the secondary winding at a frequency of 10 kHz.

2. Reverse the start and finish of the primary. Measure the voltage transfer, magnitude and phase, from the reversed primary to secondary. Compare the magnitude and phase of this measurement with the previous one. Make two more measurements, reversing the phase of the secondary. Observe how the phase is related to the polarity of the windings with respect to their start and finish.

3. Construct the 33 Ω-to-8 Ω divider shown in Figure 20.1. Measure the power dissipated in the 8 Ω resistor when the function generator is set to a value of 4 volts-peak.

4. Construct a second circuit that uses the impedance transformer as shown below.

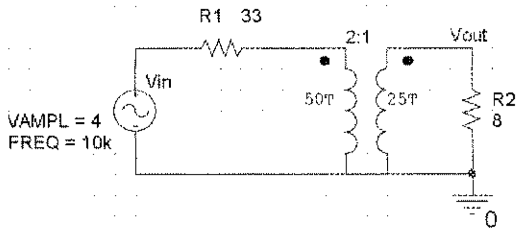


Figure 20.5 – Laboratory Circuit for Resistive Divider with Impedance Matching Transformer

Set the function generator to produce 4 volts-peak at 10 kHz. Measure the power dissipated in the 8 Ω resistor for this circuit. Also, measure the reflected value of the 8 Ω resistor at the primary terminals of the transformer. You may do this by making a voltage measurement across the transformer terminals and using the relationships for a voltage divider to calculate the reflected resistance.

## 20.4 Post-lab

Discuss your laboratory experience and compare your Pre-lab calculations with your Laboratory data. Is there close agreement? If not speculate, why not? Is there an advantage to reflecting the secondary into the primary as opposed to reflecting the primary into the secondary in terms of facilitating the computations?

**20.5 Reference**

[Irwin] J. David Irwin, *Basic Engineering Circuit Analysis*, seventh edition (a Wiley first edition), John Wiley & Sons, Inc., 2002.

# Appendix A – The Oscilloscope and the Function Generator

## A.1 Objective

In this experiment, you will learn to use the oscilloscope, the fundamental test and measurement instrument for electrical engineers. You will make measurements of voltage, time delay, and phase delay for sinusoidal signals. You will also be introduced to the function generator, an instrument used to generate periodic voltage waveforms.

## A.2 Background material

Read [Wolf and Smith], pp. 8-12, 145-156, 159-164,172-184.

In this laboratory, you will be learning the basic operation of the oscilloscope. This instrument is well described in [Wolf and Smith], and you must read the sections listed above before attending laboratory. This short section is intended only to be a brief overview of scope operation. During the laboratory, your instructor will walk you through a sequence of operations with the oscilloscope.

A diagram of the Tektronix TAS 220 oscilloscope is shown below. There are four primary sections: the cathode ray tube (CRT), the vertical section, the horizontal section, and the trigger section. (A fifth section, the cursor section will be discussed in a future lab.)

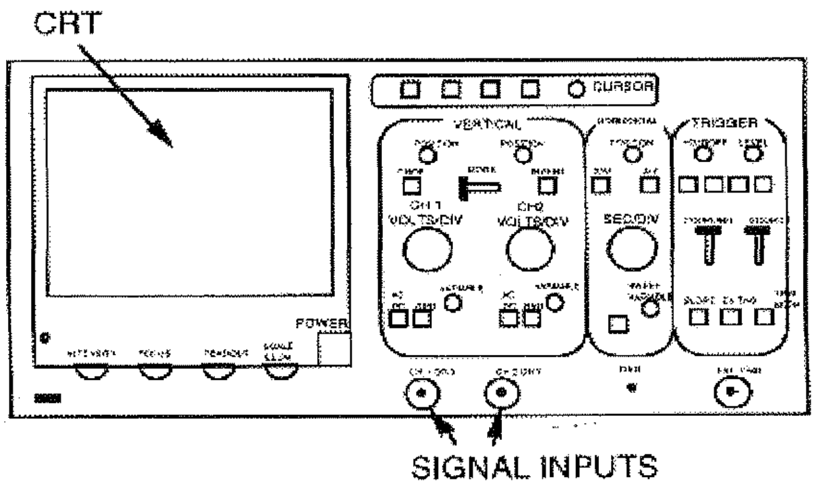


Figure A.1 – The Tektronix TAS 220 Oscilloscope

The CRT operates exactly like a television screen. An electron beam generated behind the screen is deflected by control voltages and strikes a position on the screen. The CRT controls adjust the properties of this spot. The intensity and focus controls do exactly what you would expect. As the spot is scanned across the screen at high rates, it will appear as a continuous line, called the trace. {The intensity should be adjusted so that you can easily see the trace, but don't overdo it; the screen phosphors may be damaged if too great an intensity is left on the scope for an extended period of time.}

The spot position is determined by deflection voltages in the CRT. These voltages are in turn derived from input voltages (the vertical deflection) or perhaps by an internally generated voltages ramp that sweeps the beam across the screen at regular voltage versus time, the CRT display may simply be thought of as a graph of voltage versus time. It is up to the operator to set the scaling of this graph and to be certain that the reference levels are accurate.

The vertical position of the trace is determined by two independent controls (for each of the two input channels), the position, and the sensitivity or Volts/division setting. A channel refers to one of the two voltage inputs, coupled to the scope through a BNC female connector on the front panel. The position is just a bias voltage supplied by the scope, independent of the signal input. With the input grounded (i.e., the signal in = 0V), this knob is used to place the trace at a convenient zero point, usually along a horizontal grid line.) If the signal is then applied to the input, the trace will deviate from this base line. The amount of deviation depends upon the input voltage and the Volts/div setting for the channel. For example, if a 1-volt signal is applied with the channel set a 1 V/div, the trace will rise by one division, or one horizontal grid line. If, however, the channel were set to 0.5 V/div, the trace would rise by two divisions above the ground point.

IMPORTANT NOTE: The preceding examples for the scaling of voltages works only if the oscilloscope is calibrated. The input signals are amplified by an amount that depends upon the volts/division setting. The amplification can be varied within a V/div setting using the knob labeled "VARIABLE". If this knob is turned all the way clockwise, it will click into place at a setting labeled "CAL". This means that the gain is calibrated, and the V/div setting is accurate. Without this calibration, the measured voltage will be less than the actual voltage. Normally, people leave the scope in a calibrated setting. However, here a scope will have multiple users, and you cannot always be sure that the previous user left the scope in a calibrated condition. Be sure that your oscilloscope is calibrated before making any measurements of voltage.

The scopes used in this laboratory are dual trace oscilloscopes, meaning that you can simultaneously display up to two independent inputs. A switch will allow you to display Channel 1, Channel 2, both, or the sum of the two in the dual trace mode. You may also invert the Channel 2 signal (i.e., multiply the voltages by -1), so that the sum becomes the difference. Some final notes on the vertical system: at the input, two buttons allow you to select DC, AC, or ground. The ground setting is used to position the reference level before making your measurement. The ground position may change if you change the Volts/div setting, so be sure to check this if you change scales. The DC setting allows display of the precise input signal. The AC setting removes the low-frequency components (i.e., steady-state values) and displays only time-varying signals. This is useful when looking at small voltage deviations on a large constant voltage.

**The Horizontal System**

This system generates the sweep of the trace across the screen. One way to move the trace is to supply a voltage signal at the input, just as in the vertical system. This mode, called X-Y mode, is used in applications such as plotting current versus voltage and measuring the relative phase between signals. The most common use, however, is to use an internally generated sweep signal that moves the trace across the screen in a specified time. This time is set by a knob labeled Time/div. Since the scope has 10 divisions horizontally, the total time for the sweep is ten times this value. This setting allows one to "zoom in" on rapidly varying signals (i.e., to look at smaller time intervals). The horizontal trace position may be set with a knob similar to the vertical system's, but since the absolute time is set by the trigger system, this knob rarely needs to be used. The trigger system will provide greater control over the origin of the time sweep.

**The Trigger System**

One problem with observing signals on the oscilloscope is that the response time of the eye is slow. Although the oscilloscope could easily display signals at megahertz frequencies, you would have no hope in seeing a signal trace. What you do see is several traces superimposed on each other. For random signals, the trace will be blurring of randomness, called noise. However, for periodic signals (i.e., those that exactly repeat themselves after a fixed time interval), one can get a stable fixed trace if the horizontal sweep is precisely initiated at the same point in the signal's period. Then all of the superimposed traces fall exactly on the top of one another, and the trace appears fixed in time. This is the function of the trigger system.

The trigger is just a signal that the oscilloscope uses to begin its horizontal sweep. The trigger may be supplied from an outside source (the external trigger, with a BNC input); a switch allows one to select this mode. In addition, the trigger may be obtained from the vertical display voltages. A switch may be used to select the input channel from which the trigger signal is to be obtained. The trigger works by comparing the trigger signal with a preset reference. When the level of the signal equals the preset value, a horizontal sweep is initiated. (In order that the trigger occurs only once per period, the slope of the signal is also set, so that trigger occurs only on the rising edge or falling edge of the periodic signal.) The level and slope are set by two controls in the trigger system. If you cannot obtain a stable trace, this is usually the first thing you need to adjust!

**A.3 Function Generators**

A function generator is a voltage source that produces periodic signals. It may be easiest to think of these signals as purely mathematical functions. In this laboratory, you will be using sine waves, in which the voltage varies like

$$v(t) = V_{DC} + V_{\max} \sin(2\pi f t)$$

where VDC is called the offset voltage, Vmax is the peak voltage amplitude, f is the frequency of the signal, and t is the time. (See pages 8-12 of Wolf & Smith.)

The function generator that you will use will have several different types of waves: sine waves, square waves, and triangle waves. You will also find knobs that adjust the amplitude of the sine wave (Vmax) and the offset (VDC). For this experiment, the offset VDC should be set to zero using the function generator controls.

## A.4 Examples

These examples show how a signal might appear on an oscilloscope.

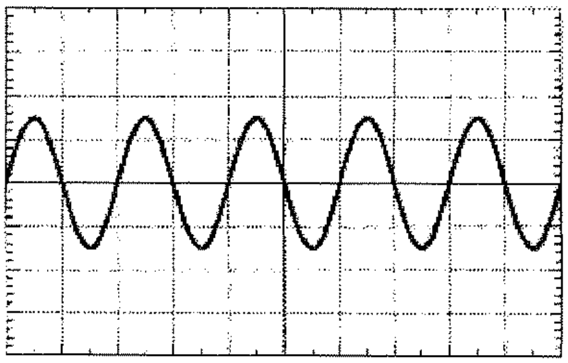
**Example 1: The sinusoidal wave**

Suppose that a sinusoidal voltage is applied to the input of an oscilloscope. The signal varies like

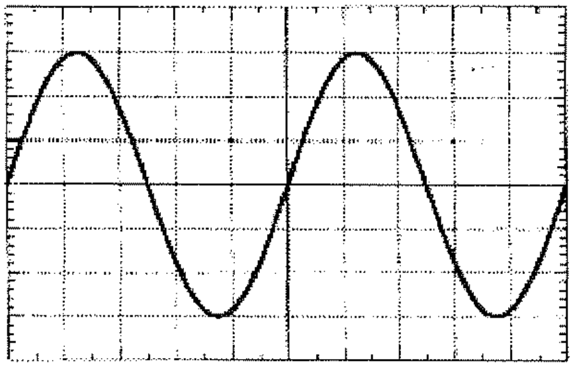
$V(t) = 1.5V \sin(2\pi(1kHz)t)$

Notice that the frequency is 1 kHz = 1000 Hz, so the period of the signal is 1/1000 sec, or 1ms.

A.) Suppose that the oscilloscope is set to 1 V/div (vertical scale) and 0.5 ms/div (time base). The trace would appear as below if ground is set to the center horizontal line. (Of course, the exact time reference point would be determined by the trigger. Here I've assumed that zero phase occurs at the left-hand side of the grid.) Notice that the period of the function is exactly 2 divisions (1ms), and the vertical scale goes from -1.5 divisions to +1.5 divisions.



B.) Now suppose that we change the vertical scale to 0.5 V/div, and the time base to 0.2 ms/div. This simple "zooms" in on the previous graph by a factor of 2 in the vertical direction and a factor of 2.5 in the horizontal direction. The CRT trace is shown below.



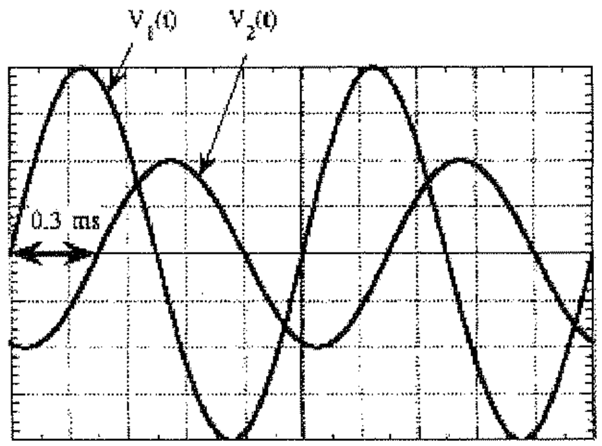
**Example 2: Time delays and relative phases**

The two inputs of a scope can be used to simultaneously display two signals. If these signals have the same frequency (from the same source), the scope can be triggered to provide stable traces of both signals. For sinusoidal waveforms, the signals will, in general, have the forms

$$V_1(t) = A\sin(2\pi f t)$$
$$V_2(t) = B\sin(2\pi f(t - t_0)) = B\sin(2\pi f t - \theta)$$

Here, A and B represent the peak voltages, and t0 is a time delay between signal V1 and signal V2. This time delay can be equated to a phase lag θ= 2πft0 between the signals. From the oscilloscope, we can directly measure f and t0, so we can determine the phase lag.

Suppose that our signals have the following values: A = 2 V, B = 1V, f =1 kHz, and t0 = 0.3 ms. (Thus θ = 0.6π.) We will set the scope to the same settings (0.5 V/div, 0.2ms/div). If we plot the two traces simultaneously, and the ground positions of both traces are set to the center grid line, we should observe the following traces.



**A.5 Reference**

[Wolf and Smith] Stanley Wolf and Richard F. M. Smith, *Student Reference Manual for Electronic Instrumentation Laboratories*, second edition, Pearson Education, Inc., 2004.

# Appendix B - The Laboratory Report

**B.1 Introduction**

In any experiment environment, whether it is an academic teaching laboratory, a research laboratory, or an industrial laboratory, it is important to be extremely careful in the acquisition of data and in the documentation describing your experimental technique. That is the purpose of the laboratory notebook. However, just as important as the method by which you disseminate this information, whether it is to your teaching assistant, your research adviser, your supervisor, or the general engineering community. An important point to remember when describing your experimental results is that you should provide enough information so that another worker could duplicate your experimental procedure and (one would expect) duplicate your results within the limitations of experimental error.

In this section, we will discuss the proper method for presenting an informal laboratory report. You should use these guidelines in the preparation of your own reports. A sample report is included at the end of the section.

When several styles of report are acceptable, some styles may be more appropriate for an academic laboratory report than for a formal report. These reports should posses some basic features. First, they should be written in a way that an intelligent, but perhaps uninformed, reader could follow the procedure. It is important to be clear and concise in your writing, and to be quantitative whenever possible. Be cautious when you find yourself using adverbs in a report: for example, "The transient voltage decayed extremely quickly, with very fast oscillations," would be inappropriate. Rather, one should measure the values and write something of the form, "The transient voltage decayed in an exponential fashion, with a time constant of 2.4 ms, with 5.8 kHz oscillations observed in the decaying signal."

Finally, always provide a brief introduction to the material, followed by the information you wish to discuss, and end with a review of the results presented. This should be done not only for the entire document, but also within the various sections of the document.

On the following page, you will find an outline that you should follow when preparing your laboratory reports.

## B.2 Laboratory Report Outline

**I.** Cover Page
- Your name
- Your laboratory partner's name
- Course number and section
- Experiment number and title
- Date of experiment
- Date of due

**II.** Introduction
- State the objective of the experiment.

**III.** Theory
- Briefly discuss the theory relevant to this experiment. This may include preliminary work that was required prior to the laboratory.

**IV.** Experimental Procedure
- Briefly outline the methods you used in the laboratory
- Provide schematics of the circuits used in the experiment.

**V.** Equipment List
- Provide manufacturer's name and equipment model number.

**VI.** Data Section
- Provide tabulated data, graphs, and sample calculations on the data in this section.

**VII.** Analysis Section
- Provide a technical discussion of your data.
- Compare your results with the expected results.
- Discuss sources of experimental errors.

**VIII.** Conclusions

## B.3 Sample Laboratory Exercise and Report

On the next few pages are an example laboratory assignment and a resulting laboratory report worthy of an A grade. Although the sample report is rather lengthy, note that a great deal of the work was performed in the prelab and simply included in the *Theory* section. Your report should follow this style, although neat, hand-drawn figures and graphs plotted on graphing paper are fully acceptable in place of the computer-generated graphics. However, with the availability of word-processing software and spreadsheets, hand-written reports are discouraged. You will also find that data plotting is trivial with most available spreadsheet programs.

# Sample Laboratory Assignment

ECE212
Experiment 3
Thevenin Equivalent Circuits

## 1. Introduction

In this experiment, you will measure the Thevenin equivalent circuit for a source considering of an (assumed) ideal voltage source and a T-network of resistors. You will measure the load voltage and load current produced by this source for several resistive loads.

## 2. Prelaboratory Exercise

1. For the source of Figure 1, find the Thevenin equivalent as a function of $V_s$, $R_1$, $R_2$, and $R_3$.
2. Suppose that $V_s = 8V$, $R_1 = R_2 = R_3 = 1\ k\Omega$, and a load RL is connected to the output terminals. Calculate the values for your Thevenin equivalent circuit. Tabulate the load voltage, load current, and power delivered to the load for the following load resistances: $0\Omega$, $10\Omega$, $100\Omega$, $1k\Omega$, $1.5k\Omega$, $2k\Omega$, $10k\Omega$, $100k\Omega$ and $1M\Omega$.
3. Plot the data calculated in part 2 on a semi-log graph. (Note that you will not be able to plot zero resistance this way.)

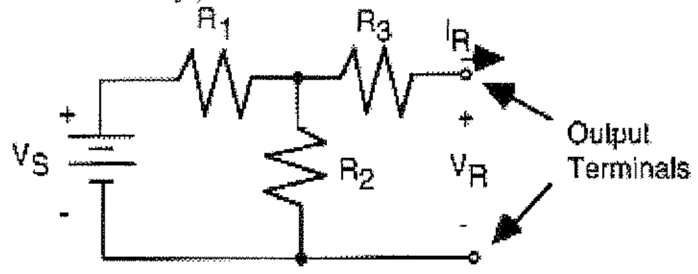


Figure 1 – Source circuit for experiment

## 3. Experimental Procedure

1. Build the source of figure 1 in the following manner. First, set a DC voltage supply to 8V. Next, construct the T-network using 1-k$\Omega$ resistors for $R_1$, $R_2$ and $R_3$. (Measure the actual resistances prior to building the circuit.)
2. Measure the open-circuit voltage and short-circuit current of your source using the digital multimeter. Remember that voltage measurements are made in parallel with the circuit element, and current measurements are made in series with the element. Calculate the Thevenin resistance and compare with the value calculated in your prelaboratory work.
3. From your parts kit, select the load resistances from part 2 of the prelab. Measure the load voltage and load current produced by your source when each load is placed at the output terminals of your source.

## <u>Sample laboratory Report</u>

Ann Undergrad (999-99-999)

Lab Partner: Willie Pass

ECE212-007

**Experiment 3: Thevenin Equivalent Circuits**

Exp date: February 29, 2005
Due date: March, 3, 2005

## Introduction

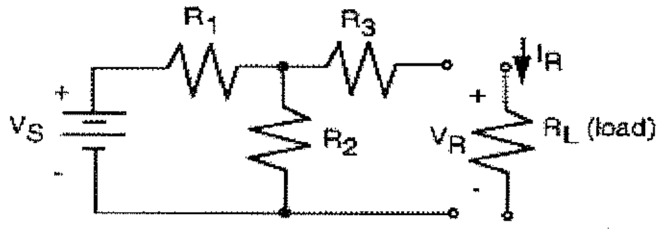
The objective of this experiment was to measure the Thevenin equivalent for a DC circuit and verify that this circuit agrees with that predicted from simple circuit analysis.

## Theoretical Background

Thevenin's theorem states that any circuit composed of linear elements may be reduced to an equivalent circuit consisting of a voltage source and a series impedance. It is useful in analyzing the voltage across or current through a dingle element in the circuit by replacing the remaining circuit with its Thevenin equivalent circuit.

In the prelaboratory section of this experiment, we were to find the Thevenin equivalent circuit for the source in Figure 1. We first find the Thevenin equivalent resistance by shorting the voltage source and calculating the resistance seen looking into the source terminals. This is

$$R_{th} = R_3 + (R_2 // R_1)$$



(Source)

Figure 1 – Circuit diagram for Thevenin equivalence problem

The Thevenin source voltage Vth is the open-circuit voltage at the terminals (with no load connected). In this case, no current flows through $R_3$, so the open-circuit voltage is found by a voltage divider formed by the $R_1$ and $R_2$ resistors:

$$V_{th} = V_{OC} = V_S \frac{R_2}{R_1 + R_2}$$

When a load $R_L$ is connected to this circuit, as in Figure 2, the voltage across the load $V_R$ and the current through the load $I_R$ are found by using the Thevenin equivalent circuit to be

$$V_R = V_{th} \frac{R_L}{R_L + R_{th}}$$

$$I_R = \frac{V_{th}}{R_L + R_{th}}$$

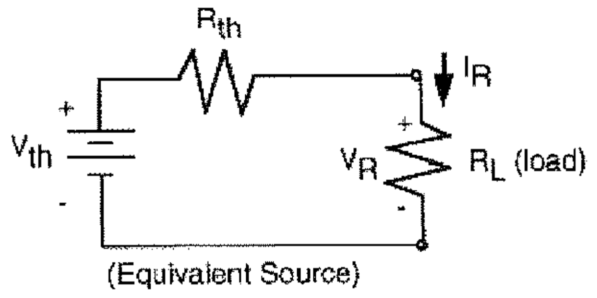


Figure 2 – Thevenin equivalent circuit, including load resistor

Using the values in the prelaboratory exercise, $R_1 = R_2 = R_3 = 1$ kΩ, we find an equivalent resistance of $R_{th} = 1.5$ kΩ. And using a supply voltage of $V_s = 8V$, we find the equivalent source value to be 4V.

Table 1- Load voltage, current and power versus resistance

| Load Resistance | $V_R$ | $I_R$ | $P_R$ |
|---|---|---|---|
| 0 Ω | 0 V | 2.667 mA | 0 W |
| 10 Ω | 0.0265 V | 2.649 mA | 70.17 μW |
| 100 Ω | 0.2500 V | 2.50 mA | 625 μW |
| 1 kΩ | 1.600 V | 1.6 mA | 2.56 mW |
| 1.5 kΩ | 2.000 V | 1.333 mA | 2.667 mW |
| 2 kΩ | 2.286 V | 1.143 mA | 2.612 mW |
| 10 kΩ | 3.478 V | 0.3478 mA | 1.210 mW |
| 100 kΩ | 3.941 V | 39.41 μA | 155.3 μW |
| 1MΩ | 3.994 V | 3.994 μA | 15.95 μW |

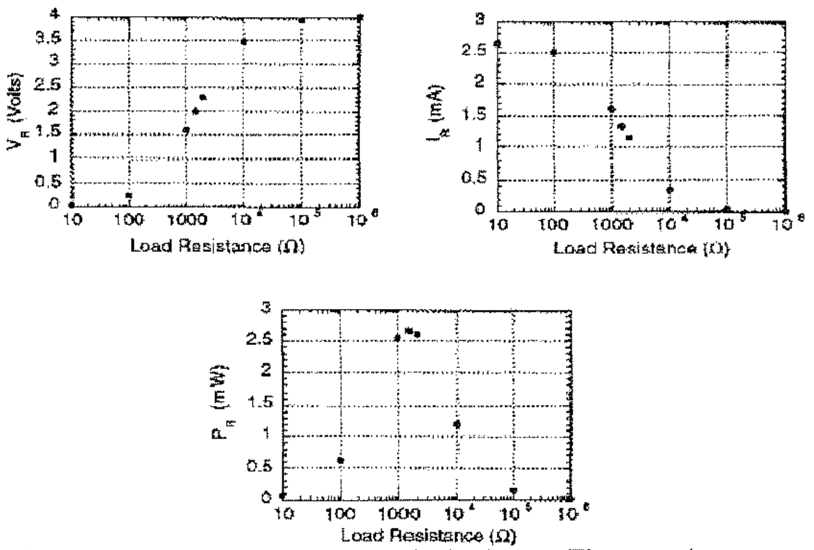


Figure 3: Load voltage, current, and power versus load resistance. (The zero resistance case cannot be plotted on the logarithmic scale.)

Using these values, we tabulate the load voltage and load current ($V_R$ and $I_R$), and the load power ($P_R = V_R I_R$) for load resistances ranging from 0 Ω to 1 MΩ, as required for the prelaboratory. These values are shown in Table 1, and plotted in Figure 3.

**Experimental Procedure**

The experimental procedure consisted of three steps. First, we built the source to be tested. Next, we directly measured the open-circuit voltage and short-circuit current, established the Thevenin equivalent source values. Finally, we placed a variety of load resistances into the circuit and measured the load voltage and current versus resistance.

We first built the circuit of Figure 1 on a protoboard, with no load connected initially. For $R_1$, $R_2$, and $R_3$ we used 5%, 1/4-Watt resistors with nominal resistances of 1.0 kΩ. We then connected a DC voltage source to the circuit, also as pictured in Fig. 1. Prior to connection, the open-circuit voltage of this source was set to 8.00V using a DC voltmeter with an input impedance of 10 MΩ.

The open-circuit voltage of the modified source, including the added resistors, was then measured using the same DC voltmeter. The set-up is pictured in Fig. 4a. This voltage corresponded to our Thevenin equivalent source voltage.

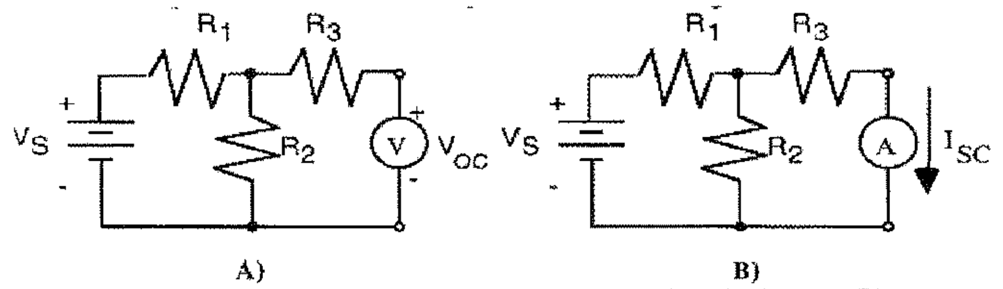


Figure 4: Measurement of open-circuit voltage (A) and short-circuit current (B)

The Thevenin resistance was found indirectly by measuring the short-circuit current, $I_{SC}$. Then $R_{th} = V_{OC} / I_{SC}$. This set-up is pictured in Fig. 4b, where a DC ammeter is connected at the output terminals of the source.

Finally, we constructed the circuit of Fig. 5, where load resistors with nominal values listed in Table 1 were connected at the load terminals of the supply. A DC voltmeter was placed in parallel with the load resistance, and a DC ammeter was placed in series with the load.

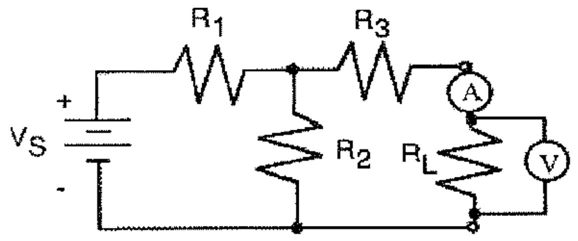


Figure 5: Circuit for measurement of load current and voltage

## Equipment List

The experiment employed the following equipment and parts:
Warsteiner Electronics Model 422 Triple Output Power Supply
Theakston Systems Model 100 Digital Multimeter (3 1/2 digits)
Protoboard
5%, 1/4 -Watt Carbon Resistors

## Data

The experimentally measured values for resistors $R_1$, $R_2$, and $R_3$ as defined in Fig.1.were measured by the digital multimeter, set on a 1 kΩ scale, to be

$$R_1 = 0.979\,k\Omega$$
$$R_2 = 1.020\,k\Omega$$
$$R_3 = 0.971\,k\Omega$$

These values were well within the stated tolerance.

The open-circuit voltage of the modified source of Fig. 4a was measured to be

$$V_{th} = 4.04\,V$$

The short-circuit current of this source was found by the measurement of Fig. 4b to be

$$I_{SC} = 2.74\,mA$$

Thus, we find the Thevenin-equivalent resistance of the source:

$$R_{th} = \frac{Vth}{Isc} = 1.47k\Omega$$

The measured values of the load resistors, the load voltage, and the load current are tabulated below for the various resistances employed.

Table 2 - Measured load resistance, voltage, and current

| Load Resistance | $V_R$ | $I_R$ |
|---|---|---|
| 10.22 Ω | 0.027 V | 2.65 mA |
| 99.5 Ω | 0.262 V | 2.63 mA |
| 1.014 kΩ | 1.591 V | 1.570 mA |
| 1.502 kΩ | 2.03 V | 1.351 mA |
| 1.973 kΩ | 2.30 V | 1.166 mA |
| 10.09 kΩ | 3.48 V | 0.344 mA |
| 100.8 kΩ | 3.95 V | 39.2 µA |
| 0.983 MΩ | 4.02 V | 4.01 µA |

## Analysis

A direct comparison of our measured data with the theoretically predicted results from the prelaboratory shows good agreement, as illustrated in Fig. 6a. We have also recalculated the expected $V_R$ and $I_R$ using the measured load resistances, and a comparison of these theoretical values with the measured values is shown in Fig. 6b.

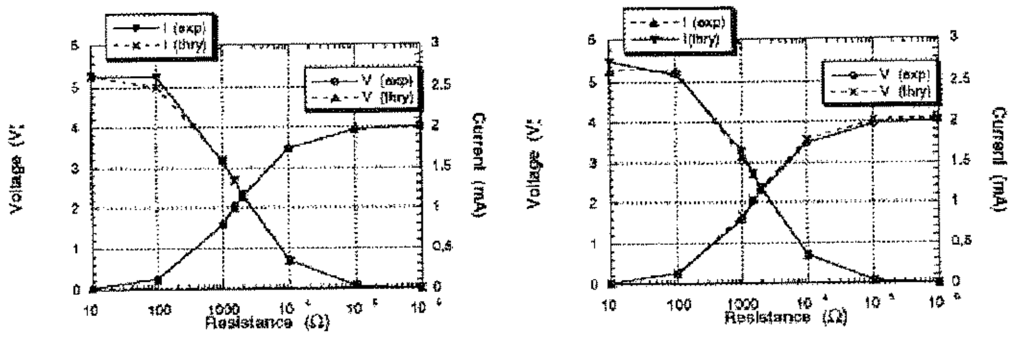


Figure 6: Comparison of experimentally measured load voltages and currents versus load resistance. A) Experiment versus prelab values. B) Experiment versus theory using actual measured values of resistances

The slight deviation from theory may be caused by several sources of error. The errors in the measured voltage and current were less than the last digit on the meter, which is less than 0.5% in all cases. This small error would not be observable on the scale of the graphs.

Another possible cause of the deviation of our results from the theoretical results is our neglecting of internal resistance in the power supply. These value would add to R1, slightly change the Thevenin equivalent values and the results of our analysis. For example, a 10 Ω internal resistance would alter $R_1$ by 1%.

Finally, the loading effects of the digital meter have been neglected in this analysis. The Theakston meter had a specified input resistance of 10 MΩ in the voltage and resistance measuring modes. This would cause significant change only in the case of the 1MΩ load, where the resistance would be changed to 0.91 MΩ When used as an ammeter, the specified series resistance is 0.5 Ω. This would alter the short-circuit and 10-Ω load current measurements slightly.

## <u>Conclusions</u>

In conclusion, we have constructed a source using an ideal voltage source with a T-network of resistors, verifying that the Thevenin equivalent circuit provides an accurate model for this source. The Thevenin values were measured directly, and the voltage-current relationships were measured when the source was applied to a wide range of loads. The experimental and theoretical results showed excellent agreement, and the slight deviation in the data may be explained by source and meter effects for which the theory does not account.

## Appendix C - Notes for Using the Data/IO Chiplab Logic Programmer

1. The Chiplab programmer must be attached to a PC through the parallel port connector and the Chiplab software installed on this PC prior to use.

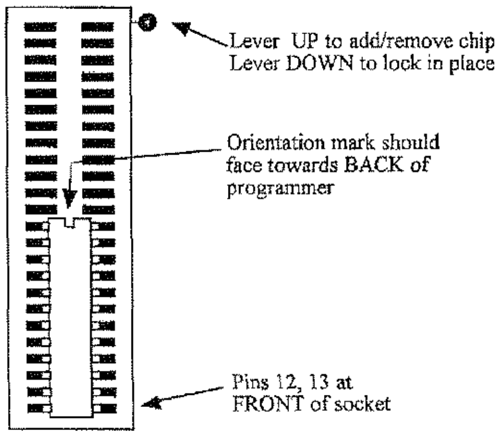
2. ABEL programs can be compiled on any PC. The file to use for programming is the "JEDEC" file, which will be named with extension ".JED". You can transfer files to the PC with the programmer using a floppy disk.

3. To start the programmer, turn on the Chiplab unit and the PC and run the "CHIPLAB" program (TaskLink for Windows, Advanced Device Programming Solutions, version 5.23). Note that it will take a few minutes to initialize.

4. Select the device type of the chip to be programmed. (Menu: Setup/Select Device...)

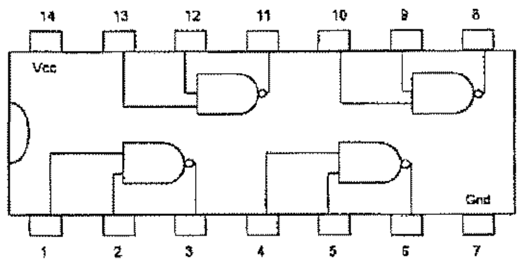
5. Load the JEDEC file for the compiled ABEL description. (Menu: Data/Load RAM from PC file)

6. Program the device:
   a. Make sure that the lever of the programmer socket is in the open (UP) position.
   b. Drop the device into the socket so that the orientation mark (the notch on the edge) points toward the BACK of the programmer and the pins go in the holes that are closest to the FRONT of the socket, as shown below. The device should drop in easily – if you need to force it, then something is wrong.
   c. Move the lever on the programmer socket to the closed (DOWN) position.
   d. Program the chip. If programming is successful, you will get a message on the screen saying so. (Menu: Process/Program Verify)
   e. Move the lever on the programmer socket to the open (UP) position and remove the device. The student may now place the device in his or her circuit for testing.

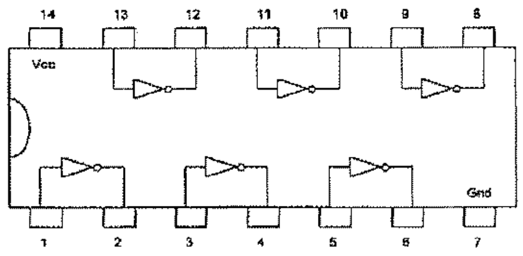

Lever UP to add/remove chip
Lever DOWN to lock in place
Orientation mark should face towards BACK of programmer
Pins 12, 13 at FRONT of socket

## Appendix D – Parts Outlines

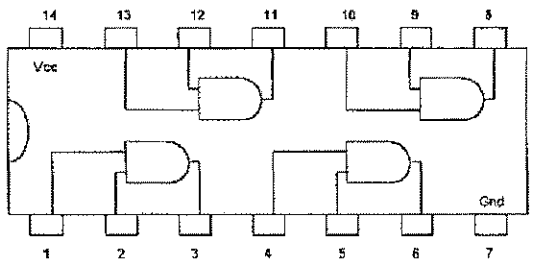
74LS00 Quad 2-input NAND



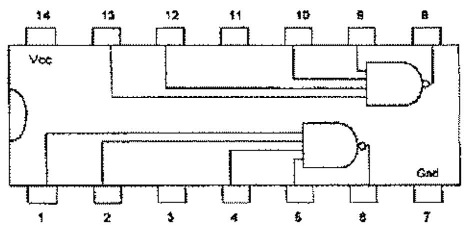
74LS04 Hex Inverter



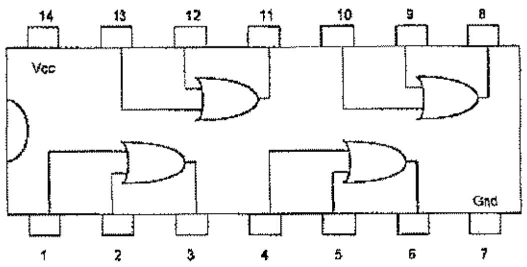
74LS08 Quad 2-Input AND



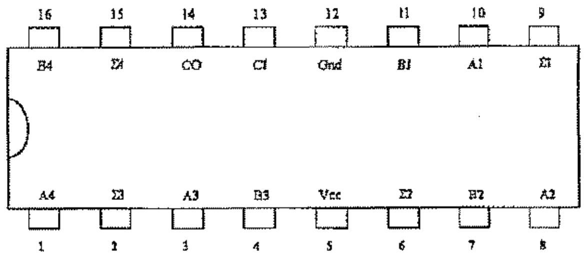
74LS20 Dual 4-input NAND



74LS32 Quad 2-Input OR

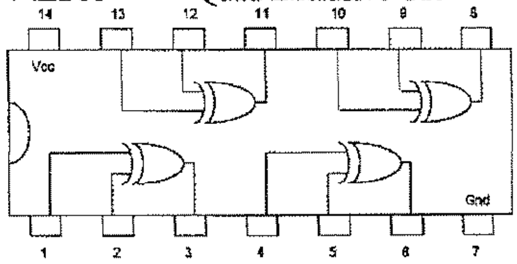


74LS83 4-bit Full Binary Adder

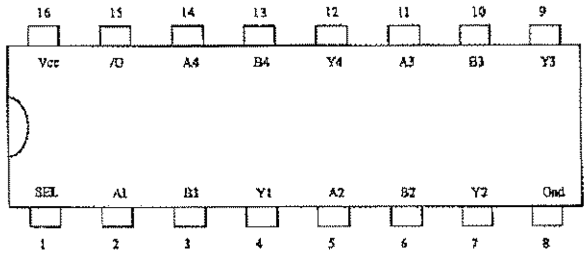


NOTE: VCC and Ground are not on corner pins. 4's are most significant.

74LS86 Quad Exclusive OR

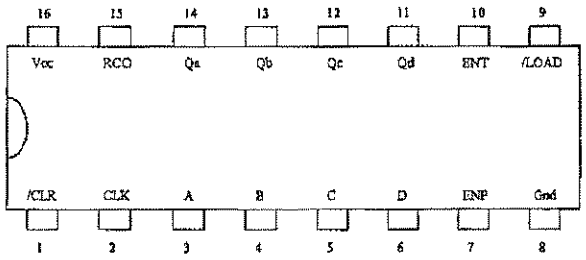


74LS157 Dual 2-to-1 Multiplexer



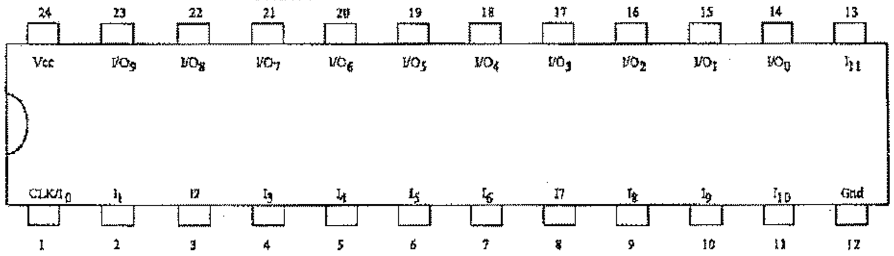
NOTE: /G is an active low enable. Output is B when SEL = 1.

74LS163 4-bit Synchronous Counter

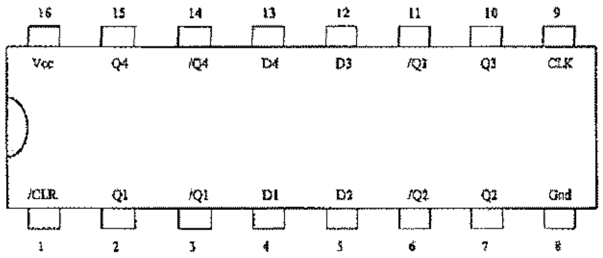


NOTE: CLR and LOAD are active low. ENP and ENT are active high.

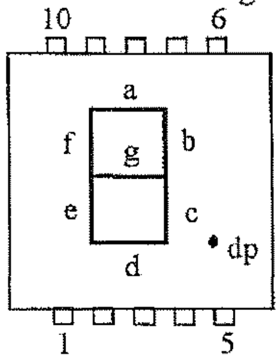
PALCE22V10 Ereasable PLD



74LS175 Quad D Edge Triggered FF



FND507 Seven Segment Display



| Pin | Segment |
|---|---|
| 1 | e |
| 2 | d |
| 3 | CA |
| 4 | c |
| 5 | dp |
| 6 | b |
| 7 | a |
| 8 | CA |
| 9 | f |
| 10 | g |