Lab 2 – Intro to Digital Logic and Transistors

Introduction:

Up until now, everything that you have done has been in the analog realm. By changing the resistance of your simple LED circuit, you have been able to sweep through a continuous range of light intensity. However, in the world of digital electronics, a signal can have only one of two values: ON (HIGH, 1) or OFF (LOW, 0). In this lab, you will become familiar with basic digital logic, and you will implement basic logic functions using pushbutton switches and integrated circuit (IC) chips. You will also learn how to use a transistor as a switch, and in the process, you will learn how to use the function generator to supply a voltage signal and the oscilloscope to look at voltage signals.

Goals:

- Learn how to implement logic gates with switches
- Learn how to read pinout diagrams
- Understand the operation and importance of the half-adder circuit
- Learn how to use the transistor as a switch
- Learn how to use the function generator and oscilloscope

Procedure:

1. Turn an LED on/off using a pushbutton

   - Obtain a four-terminal pushbutton. Figure 1 shows a diagram of a pushbutton and its internal connections. When the button is not being pressed, terminals 1 and 2 are electrically connected to each other (shown in red), and terminals 3 and 4 are electrically connected to each other (also shown in red). When the button is pressed, all four terminals become electrically connected to each other (shown in green).

   ![Figure 1: Pushbutton connections; red = not pressed, green = pressed](image)

   - Turn the output of the power supply off and build the circuit represented in Figure 2. Use Figure 3 as a reference. Note that the four-terminal switch that we are using has been abstracted to a two-terminal switch for simplicity.
- Turn the power supply on. The LED should now only turn on when you press the button. In the digital world, the output is either a 1 (the LED is on) or a 0 (the LED is off).

2. **Build logic gates using pushbuttons**

Logic gates are the basic building blocks of all digital electronics. Logic gates have some number of binary inputs, usually two, and one output. You will now build two basic logic gates, the AND gate and the OR gate, using pushbuttons.

The symbol and truth table for the two-input AND gate are shown in Figure 4.
In a truth table, a ‘0’ represents a low voltage (0V) and a ‘1’ represents a high voltage (in our case, 5V). The output of the AND gate is only ‘1’ when both of the inputs are ‘1’; otherwise, the output is ‘0.’ This AND gate can easily be constructed by placing two switches in series.

- Turn the output of the power supply off and build the circuit represented in Figure 5. Use Figure 6 as a reference.
- Turn the power supply on. The LED should now only turn on when both buttons are pressed. Since the pushbuttons are placed in series, current will only be able to flow from 5V to ground through the LED if both buttons are pressed; otherwise, the circuit is open and current does not flow, so the LED is off.

The symbol and truth table for the two-input OR gate are shown in Figure 7. The output of the OR gate is ‘1’ when either of the inputs is ‘1’; the output is ‘0’ if neither input is ‘1.’ As you might have guessed, the OR gate can be constructed by placing two switches in parallel.

**2-input OR gate**

![Diagram of 2-input OR gate]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*Figure 7: Symbol and truth table for two-input OR gate*

- Turn the output of the power supply off and build the circuit represented in Figure 8. Use Figure 9 as a reference.
- Turn the power supply on. The LED should now only turn on when *either* button is pressed. Since the pushbuttons are placed in **parallel**, current can travel through either or both pushbuttons if they are pressed, thus lighting the LED; otherwise, the circuit is open and current does not flow, so the LED is off.

Using combinations of logic gates, complex operations can be performed. In this lab, you used electromechanical switches to implement simple logic gates. However, in modern electronics, logic gates are implemented with **transistors**, which can be used as electrically activated switches. Anywhere from a few to thousands of transistors can be combined on a single chip to create **integrated circuits (ICs)**.
3. *Build an XOR gate using an IC*

The symbol and truth table for the two-input exclusive-OR (XOR) gate are shown in Figure 10.

![Exclusive-OR gate](image)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

*Figure 10: Symbol and truth table for two-input OR gate*

The output of the XOR gate is ‘1’ when *either but not both* of the inputs is ‘1’; the output is ‘0’ if *neither* input is ‘1’ or if *both* inputs are ‘1.’ Since this gate is slightly more complicated to build with pushbutton switches, you will use an IC chip which has four independent XOR gates built in. (Logic chips like this use complementary metal-oxide-semiconductor [CMOS] technology → take ESE 370 if you want to learn more about how to build logic gates with MOS transistors.)

The IC logic chips that you will see in lab are installed in dual-in-line packages (DIPs). This allows you to easily insert the IC into a breadboard. The 7486 Quad 2-input XOR Gate, as the name implies, features four independent exclusive-OR (XOR) gates. Two of these chips are shown in Figure 11.

![7486 Quad 2-input XOR Gate chips](image)

Since we cannot see inside of these chips, we must consult the datasheet provided by the manufacturer to understand how to connect these chips to the rest of our circuit. The datasheet will provide a *pinout diagram*, such as the one shown in Figure 12, which provides information about each pin’s function and may show some of the chip’s internal circuitry. The datasheet also provides specifications on how much voltage and current should be supplied to the IC.
Notice the notch in the pinout diagram in Figure 12. IC packages usually have a notch or dot on one end of the package to indicate the position of pin 1. As you can see in Figure 12, the pin ordering is counter-clockwise.

Figure 13 shows a schematic for an XOR gate circuit. R1 and R2 are pull-down resistors which are used to pull the voltage at the gate inputs down to 0V while the buttons are not pressed. When a button is pressed, the voltage at the gate then becomes 5V. The LED is used to indicate the output of the gate; it will light up when only one of the buttons are pressed.

- Build the circuit represented in Figure 13. Consult Figure 14 as a reference.
  - You must place the IC over the partition in the breadboard. (Why?)
  - Supply 5V to pin 14 and 0V to pin 7.
  - Consult Figure 12 for the pinout of the IC. You may use any of the four gates.
- Turn the power supply on. The LED should only turn on when only one of the buttons is pressed; otherwise, it should be off.
4. Binary addition and the half-adder

Digital logic gates are what computers and microcontrollers are made of. Since we’ll be using microcontrollers later this semester, let’s take a little look inside. A simple microcontroller is composed of an Arithmetic & Logic Unit, or ALU, digital circuitry to move data and program information to and from memory, a memory bank, and a number of peripherals to help interface with the outside world, for example input/output, or “I/O” ports.
We’ll focus today on the ALU, which is where all math and logic functions commanded in the program code are executed. Specifically, we’re going to look at one small piece of circuitry that performs addition.

Binary addition is very similar to our normal decimal, or base-10, addition: just as in decimal math where we add each digit in one number (1’s place, 10’s place, etc.) to the corresponding digit in the other number, then carry results of 10 or greater, so is it done with binary. Consider the following examples:

\[
\begin{align*}
0 + 0 &= 0 \\
1 + 0 &= 0 + 1 = 1 \\
1 + 1 &= 10 \\
\text{carry} \rightarrow 11 \\
1101 + 0110 &= 10011
\end{align*}
\]

Looking at a single digit, we can summarize the behavior into a truth table. There are again two inputs (one from each of the ‘addends’, the numbers to add). Unlike before, we now have two outputs – a ‘sum’ (S) and a ‘carry’ (Carry_Out, or Cout).
- Draw up a table like the following to capture the behavior of this adder.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>S</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Truth table for half-adder

With the truth table filled in you can start to see the patterns for each of the two outputs.

- Design a circuit using logic gates only to implement this logic. (You just need to draw the schematic – use logic gate symbols and letters to represent input/output voltages.) This circuit is called a half-adder.

- **Bonus:** If you have time after the next section, build a half-adder circuit using two IC logic gate chips, two pushbuttons as inputs, and two LEDs (with current limiting resistors, of course) to represent the output bits. Consult a TA before building your circuit!

In theory now we could cascade these circuits together, with one feeding into another, to implement addition of binary numbers more than one-bit wide. But you’ll notice we’re missing one crucial piece: how do we account for the carried bit? The circuit we’ve implemented is called a half-adder because it cannot handle the carry-in bit, thus making it impossible to cascade into a useful multi-bit adder. A full-adder, on the other hand, has three inputs: A, B, and Carry_In (or Ci). Thus, the carry-out of one full-adder becomes the carry-in of the next.

- As an optional challenge, draw up a truth table for the full-adder – then, if you’re feeling ambitious, try designing a circuit that uses logic gates to implement the full-adder.

*To learn more about digital logic and design, take ESE 170/171. To learn more about computer architecture, take CIS 240.*

5. **Use a transistor as a switch**

Transistors have two very important applications: amplification and switching. In analog circuit design, transistors are valued more for their ability to amplify signals, while in digital circuit design, they are valued for their ability to rapidly switch voltages on and off. There are two main families of transistors: bipolar junction transistors (BJTs) and field-effect transistors (FETs). Most ICs feature a subset of FETs called metal-oxide-semiconductor FETs, or MOSFETs. The distinction between BJTs and FETs is beyond the scope of this lab. You will be using a BJT in this lab.
To learn more about how transistors work on a physical level, take ESE 218. To learn more about how to design circuits with MOSFETs, take ESE 216. To learn more about how to design circuits with BJTs, take ESE 319.

There are two types of BJTs: NPN and PNP. Most circuits tend to use NPN. The difference between NPN and PNP has to do with the level of *doping* in each of the layers of silicon and will not be discussed in this lab. You will be using a 2N3904 NPN transistor, pictured in Figure 16.

The BJT has three terminals (leads): the emitter, base, and collector (Figure 16). The collector is connected to the supply voltage through a load (in our case, an LED and a current-limiting resistor). The emitter is connected to ground. The base is the lead that is responsible for activating the transistor. By applying a small current at the base, a much larger current will flow from the collector to the emitter. Otherwise, no current will flow. This concept is demonstrated in Figures 17 and 18.

![Image](image_url)

Figure 17: When no voltage (or a tiny voltage) is applied through Rin, no significant current travels through the base-emitter junction. Therefore, no current travels from the collector to the emitter.

![Image](image_url)

Figure 18: When a voltage is applied through Rin, a small current is formed at the base. This activates the current flow from the collector to the emitter.
You will use the function generator to supply a time-varying voltage to the base of the transistor to control the behavior of an LED connected to the collector. The LED should only turn on once a sufficient positive current is applied through the base-emitter junction.

- Build the circuit shown in Figure 19. Use Figure 20 as a reference. Consult Figure 16 for clarification on how to connect the transistor to your circuit.
- To connect the function generator to your circuit (the signal represented by Vs), use a cable with two test clips on one end and a BNC connector on the other. The BNC connector plugs into the output port on the face of the function generator (the connector on the right). Connect the red test clip to the base of the transistor through the 100-kΩ resistor (Brn-Blk-Yel-Gld), and connect the black test clip to ground.

![Figure 19: BJT switching circuit schematic](image)
Using the Agilent 33521A Function Generator

A function generator (also called an arbitrary waveform generator) is a piece of lab equipment that supplies an electrical signal to a circuit. An electrical signal is a time-varying voltage, which is represented mathematically as a voltage function of time (hence the name “function generator”). The signals produced by the function generator are periodic, meaning that they repeat with a set frequency. Another word for a periodic, time-varying function is a waveform. You may be familiar with some of the more common types of wave, such as sine waves and square waves.

1. Turn on the function generator. The power button is located at the bottom left corner of the instrument’s front panel.
2. Select the type of waveform you want to create by pushing the “Waveform” button (to the right of the screen) and then selecting the button beneath the wave you want (in this case, Sine).
3. Press the “Parameters” button (below “Waveform”) and select the parameter you want to edit by pressing the button underneath its name at the bottom of the screen.
4. Enter 1 kHz for the frequency, 5Vpp for the amplitude, 2.5V for the offset, and leave the phase at 0 degrees. Note that “Vpp” represents a peak-to-peak voltage amplitude; that is, the amplitude of the wave is 5V from the highest point in the wave to the lowest point in the wave. The 2.5-V offset shifts the wave “up” by 2.5V. This offset value eliminates negative voltages in the signal, as the oscillation will now be from 0V to 5V instead of −2.5V to 2.5V.

5. Press the “Channel” button (above the output port), and then turn the output of the function generator on by pressing the leftmost button under the screen.

Using the Agilent MSO7034B Oscilloscope

The simplest way to measure a voltage is to use a multimeter. This works well if the voltage isn’t changing in time, or is changing very slowly. But how do you measure a voltage that is changing hundreds, or even millions of times a second?

The oscilloscope is a piece of equipment designed to do exactly that. The screen of the oscilloscope displays a plot of voltage (y-axis) versus time (x-axis). By actually plotting the signal, you can actually observe it in a way that is simply impossible if you are just reading numbers off of a multimeter.

The Agilent oscilloscopes that we have in lab are capable of displaying 4 different signals at the same time, so that you can measure and compare multiple waveforms. They are also able to automatically perform many different kinds of measurement and mathematical operation that you will not need for this class.

1. Turn on the oscilloscope (the power button is at the bottom left of the front panel).
2. At the bottom right of the front panel there is a BNC connector for each of the four ports of the oscilloscope. Plug a scope probe into the connector corresponding to channel 1. Plug a second scope probe into the connector corresponding to channel 2.
3. Clip the probe to the point in the circuit that you would like to measure. In this case, clip the first probe (channel 1) to the same wire as the input signal from the function generator, and clip the second probe (channel 2) to the collector of the transistor to measure the output signal.
4. Press the “Auto Scale” button (top right corner of the front panel). This adjusts the x-axis and y-axis scales to make the waveform easy to see. If you are using more than one channel, the Auto-Scale function is less useful. There are controls on the face to adjust the vertical (voltage) scales for each channel individually, as well as controls to adjust the horizontal (time) scale for all of the signals.
   a. Adjust the voltage scale so that both signals are on a 5-V scale (use the big, color-coded knobs right above the buttons with the channel numbers).
   b. Adjust the time scale so that you can get a clear picture of the waveform (use the big knob at the top-left of the button panel).
c. If you need to, you can also adjust the vertical positions of the waveforms (use the smaller, color-coded knobs right under the buttons with the channel numbers) or the horizontal position of the waveforms (use the smaller knob to the right of the knob used to adjust the time-scale).

5. Observe the behavior of the LED and compare it to the output waveform on the oscilloscope. Why does the LED look like it is constantly on?

6. Change the frequency of the sine wave to 10Hz. Can you now see the LED flashing on and off? Keep increasing the frequency until you can no longer notice the LED flashing.

7. Spend a few minutes changing the parameters of the input signal and observe how your changes affect the LED behavior and the output waveform on the oscilloscope. Try changing the following parameters:
   a. Waveform shape (sine wave, square wave, etc.)
   b. Frequency
   c. Amplitude
   d. Offset