Alyssa J. Pasquale, Ph.D.
College of DuPage

Introduction to Digital Systems Lab Manual

Spring 2025 Edition

this lab manual belongs to

Table of Contents

License & Attribution Information

Pre-Lab 1

Carefully read the entirety of Lab 1, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Is there a built-in connection on the breadboard between the binding posts and the power rails? If not, how can we make that connection?

2. What two things can happen if the power supply is dialed to a value greater than +5 V?

   (a)

   (b)

3. Why do we use red wires for power and black wires for ground?

4. What might happen if the exposed contacts of red and black wires touch each other?

5. How should you disconnect the power supply from your breadboard during class if you want to temporarily remove the voltage from your circuit?

6. How should you disconnect and turn off the power supply at the end of lab?

7. Why are DIP chips inserted into a breadboard in such a way that the two columns of pins straddle the trench?

8. Why is it important to be careful when inserting and removing DIP components from a breadboard?

9. What are the two purposes of pull-down resistors?

   (a)

   (b)

Lab 1: Introduction to Digital Labs

In this introductory lab, you will learn how to use the lab equipment required to build digital logic circuits. The most important pieces of equipment, that will be used in almost every lab in this class, are the breadboard, power supply, and DIP switch.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-1

1.1 Breadboards and Power Supplies

In days before modern breadboards, people who worked on electronics projects had to find a way to conveniently prototype circuits that allowed them to create removable electrical connections between components. Many folks found that using an actual breadboard (a piece of wood that you cut bread on) was useful. People would drill holes in them and then insert screws or nails in regular spacings in those holes. By wrapping the leads of different components around the nails, they could achieve electrical connections at each individual nail.

The name "breadboard" continues to be used, even in modern versions that are no longer built out of pieces of wood. Breadboards have a bunch of built-in electrical connections. These electrical connections use terminal strips that are hidden on the back side of the breadboard. Along the sides of each panel run power rails. These long columns can be connected to VCC and ground for easy access along the entire breadboard. In-between, rows of holes are connected by terminal strips, with a large trench running between two sets of rows. A terminal strip is a strip of metal that connects each of the input holes. The trench provides electrical isolation between adjacent rows of pins, and is ideal for use with digital logic chips.

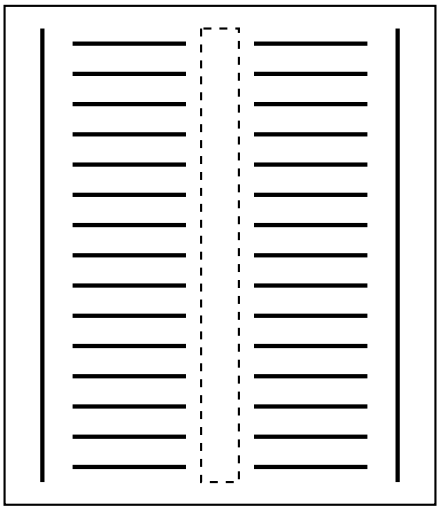


Figure 1.1: Diagram of the electrical connections (thick solid lines) and trench (dashed lines) on a breadboard.

Figure 1.1 shows a simplified layout of a breadboard. The dashed rectangle indicates the trench.

1.2 Digital Logic Signals

A digital logic signal can take on one of two values. This is why the binary number system is used in digital electronics. Binary represents two states, and those two states correspond to electrical signals in a circuit. These values can be represented as a binary value (with the numerals 0 and 1), as a signal (LOW and HIGH), as the positioning of a switch (OFF and ON), as a voltage value (0 V and 5 V), or using circuit notation (GND and VCC). These terms may be used interchangeably. This is shown in table 1.1.

1.2.1 Connecting Power to a Breadboard

Binding posts are used to connect to a power supply to supply digital logic signals. There is no inherent connection between the breadboard power rails and the binding posts. These connections must be made

Binary Value	Signal Value	Switch Status	Voltage	Circuit Notation
0	LOW	OFF (open)	0 V	GND
1	HIGH	ON (closed)	5 V	VCC

Table 1.1: Binary values, signal value, switch status, voltages, and circuit notation in digital logic.

manually. Each binding post can be unscrewed. When unscrewed, there is a hole that can be used to place a wire inside. Place the metal end of a jumper wire in this hole and screw the binding post back down to create a firm electrical connection. (Be careful that the plastic insulating part of the jumper wire isn't inserted into the binding post, only the metal end.) Lightly tug on the wire to ensure that it stays in place. Then, place the other end of the jumper wire into a power rail. Connections from that power rail can then be made to other places on the breadboard as needed. Use a banana plug (named due to the shape of the wire coming out of the end) to connect the power supply to the binding post.

Circuit 1: Connect a power supply to the breadboard. Do not turn the power supply on yet. Use a red jumper wire to connect the red binding post to one of the power rails on your breadboard. Use a black jumper wire to connect the black binding post to a different power rail on the breadboard. Use a red banana plug to connect the + terminal of the power supply to the breadboard, and a black banana plug to connect the – terminal of the power supply to the breadboard.

Note: it is important not to connect red and black wires into connected sections of the breadboard. This would create an electrical short circuit, creating a lot of current which generates heat and can melt the wire, the breadboard, or other circuit components.

When the power supply has been properly connected to the breadboard, demonstrate your setup to your instructor to receive a stamp. When you get the OK, you will be instructed to turn your power supply on and slowly dial it up to 5 V. This is how you will get every circuit started in this class.

Instructor Stamp: _____

1.2.2 Disconnecting and Turning Off the Power Supply

If the power supply needs to be temporarily disconnected from a circuit, do not turn the power supply on and off, as this will stress the power supply. In addition, when the power supply is turned on, the voltage may spike into the tens of volts before settling back to zero. Therefore, if the power supply is turned on and off when connected to a circuit, those voltage spikes may cause damage to anything connected to the supply.

To temporarily disconnect the power supply from the breadboard, either disconnect one of the banana plugs (it doesn't matter which one as any break in the connection to or from the power supply will interrupt the flow of current) or dial the voltage back to zero. When dialing back to 5 V again, do so slowly, as an over-voltage may damage components or melt the breadboard.

At the end of lab, when completely finished using the power supply, follow the indicated steps to turn the power supply off correctly.

1. Dial the voltage knob down to zero.

2. Unplug the banana plugs from the front of the power supply.

3. Turn the switch on the power supply to the OFF position.

1.3 Light-Emitting Diodes

Light-emitting diodes (LEDs) will be used in labs to determine whether the output of a logic functions is a 1 or a 0. A lit LED will correspond to 1 (ON), and an unlit LED will correspond to 0 (OFF). The LED is simply a convenient way to determine logic levels without having to use a voltage probe.

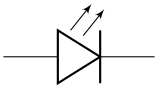


Figure 1.2: Circuit schematic of a light-emitting diode (LED). Current will flow in the same direction as the triangle (from anode to cathode), but will not flow in the opposite direction.

A diode is a semiconductor device that allows the flow of electrons in only one direction. Schematically, current flows in the same direction of the triangle in the circuit symbol (shown in figure 1.2) for the diode or LED. This means that high potential must be present on the anode (left side of the circuit diagram) and low potential must be present on the cathode (right side of the circuit diagram) for the LED to light up.

There are two ways to distinguish the anode from cathode in a physical LED, depicted in figure 1.3. The anode has a longer leg and is connected to the epoxy cap in a section where the cap has a rounded edge. The cathode has a shorter leg and is connected to the epoxy cap in a section where the cap has a flat edge.

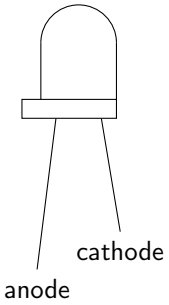


Figure 1.3: Depiction of an actual light-emitting diode (LED). The anode corresponds to the long leg connected to the round end of the epoxy cap. The cathode corresponds to the short leg connected to the flat end of the epoxy cap.

1.3.1 Current-Limiting Resistors

An LED can only tolerate a limited amount of current flowing through it before it overheats and possibly melts. It is therefore important to use a current-limiting resistor in every circuit path that contains an LED. A current-limiting resistor should be connected in series with an LED, and can be connected to either the anode or cathode, as depicted in figure 1.4.

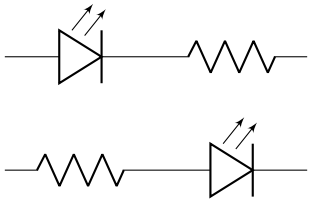


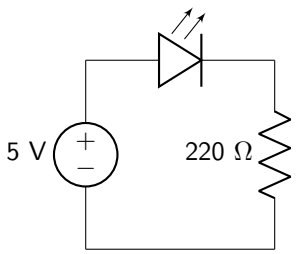
Figure 1.4: A current-limiting resistor can be connected to either the cathode (top) or anode (bottom) of an LED.

While different LED displays may require some calculation to determine the appropriate value of current-limiting resistor to be used, all of the LEDs used in these labs will work with a 220 Ω resistor. Resistors are

identified by reading the colored bands around the ceramic coating. Each color corresponds to a number. 220 Ω resistors are colored RED-RED-BROWN, which means 22×10^1, or 220. The gold-colored stripe indicates the tolerance of the resistor, which means that the actual value of the resistor will fall between 5% of the indicated resistance value.

It is important to note that, unlike LEDs, resistors do not have a polarity to them. In other words, it does not matter which end is connected to high potential and low potential in a circuit.

Circuit 2: Connect the following circuit, which contains the power connections that you wired up previously to the power supply, and an LED with current-limiting resistor. If this successfully works, then you should see the LED light up.



When you have completed your circuit, verify that it works. Then, demonstrate your results to your instructor to receive a stamp.

Instructor Stamp: _____

1.4 DIP Switches

DIP stands for dual in-line package. Any component that has a DIP configuration contains two sets of pins set in parallel (in-line) columns. Each of the pins on a DIP component will be unique. That is, each pin has its own particular function, that depends on the device being used. This fact means that a DIP component must be inserted onto a breadboard such that the columns of pins straddle the isolating trench in the center of the board. This prevents the pins that would otherwise share a row from being shorted together. Virtually all of the components used in this class will be of DIP architecture.

Because the pins in any DIP component are set vertically, they can be very easily bent. For this reason, it is important to be very careful when inserting and removing DIP chips from the breadboard. Use the dental probe to gently wedge the DIP component out from top and bottom, and then pull vertically upward to remove the DIP switch. Do not apply a large force to the DIP component from only one side, as that tends to cause the pins on the opposing side of the pins to bend (for example, if levering up strongly from the bottom, the top pins may bend outward).

DIP switches are a convenient package of toggle switches that are used to control the input signals of a digital logic circuit, which can correspond to the two logic states defined in table 1.1. This allows signal inputs to a digital logic circuit to be controlled in one location without having to continuously move wires from one place to another on the breadboard. A diagram of a DIP switch containing four toggle switches is shown in figure 1.5. The position of the slider indicates the value of each switch. In

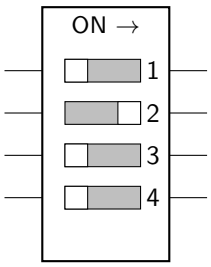


Figure 1.5: Diagram of a DIP switch containing four toggle switches. Switches 1, 3, and 4 are OFF. Switch 2 is ON.

figure 1.5, switches 1, 3, and 4 are in the OFF (open switch) position. Switch 2 is in the ON (closed switch) position.

Digital logic signals can only be generated in a DIP switch if it is connected properly. Electrical connections to both GND and VCC are required to do this. The left side of the DIP switch is therefore connected to VCC. Each of the switches that is to be used must have its own connection to VCC made somewhere in the adjoining row on the breadboard.

A connection to GND is also required to avoid floating values (which indicate that no electrical connection exists). The right side of a DIP switch cannot be directly connected to ground (as that would cause a short circuit when the switch is closed), but instead must contain a pull-down resistor. A pull-down resistor connects the right side of the DIP switch to GND, hence the name "pull-down" (pulling down to zero volts.) The pull-down creates an electrical connection without shorting anything out. The signal wire is always connected between the switch and the pull-down resistor. In this manner, a pull-down resistor accomplishes two functions: it prevents short circuits, and prevents floating values.

Figure 1.6 shows a circuit diagram of the proper wiring of a DIP switch. The left side of the switch is connected directly to VCC. The right side of the switch is connected to GND using a pull-down resistor. When the switch is OFF (figure 1.6, left), the switch is open. Current cannot flow, and the DIP switch signal value is LOW. When the switch is ON (figure 1.6, right), the switch is closed. Current flows from VCC to GND, via a pull-down resistor to protect against short circuits. In this case, the DIP switch signal value is HIGH.

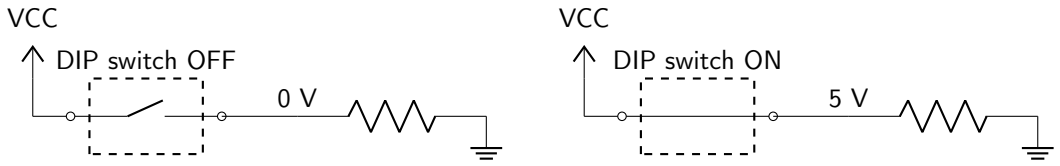


Figure 1.6: Schematic of a DIP switch connected directly to VCC and to GND via a pull-down resistor, both when the slider is in the OFF position (left) and when the slider is in the ON position (right).

Circuit 3: Connect two switches of a DIP switch to two LEDs. In this manner, each of the switches will control one of the LEDs. When you have completed your circuit, verify that it works. Then, demonstrate your results to your instructor to receive a stamp.

Instructor Stamp: _____

1.5 Preparation for Lab 2

Carefully read through lab 2 and watch the YouTube video explaining how to use TinkerCAD. You will use TinkerCAD to build some of the circuits that will be built in lab 2, giving you practice in completing the circuit in a simulation before building it in real life. Your instructor will provide you with a course access code to sign up for TinkerCAD. Using that access code, your circuit can therefore be viewed by your instructor to provide you with feedback prior to the next lab.

Create separate simulation files for each circuit and ensure that they are properly labeled. **You will not need to obtain stamps for these simulations, as your instructor will view them, grade them, and provide feedback after lab 1 ends and before lab 2 begins. Note that you are expected to complete this activity as part of the lab, not as part of the lab homework. Do not leave the lab until you have completed this activity.**

Circuit 4: Build a NOT gate circuit in TinkerCAD using the 7404 chip.

Circuit 5: Build an AND gate circuit in TinkerCAD using the 7408 chip.

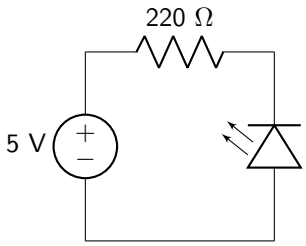
Circuit 6: Build an OR gate circuit in TinkerCAD using the 7432 chip.

Lab 1 Homework

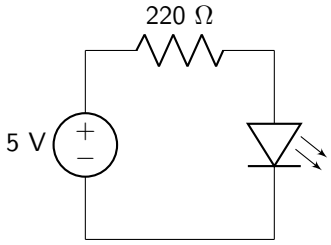
Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Build the following LED circuits in your TinkerCAD account, using the access code provided by your instructor. For each circuit, indicate below whether or not the LED will be lit. If the LED does not light, explain why.

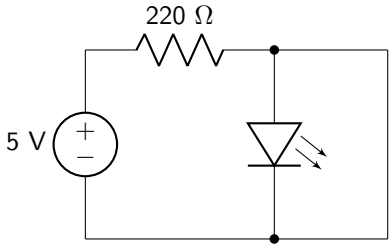
   (a)



   (b)

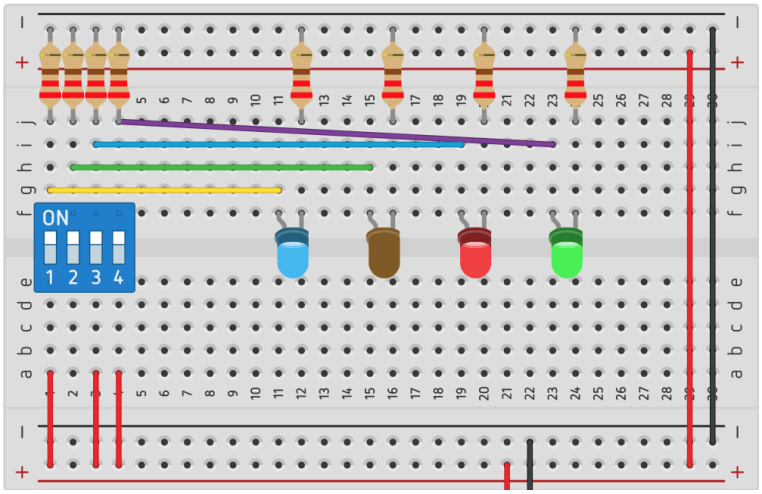


   (c)

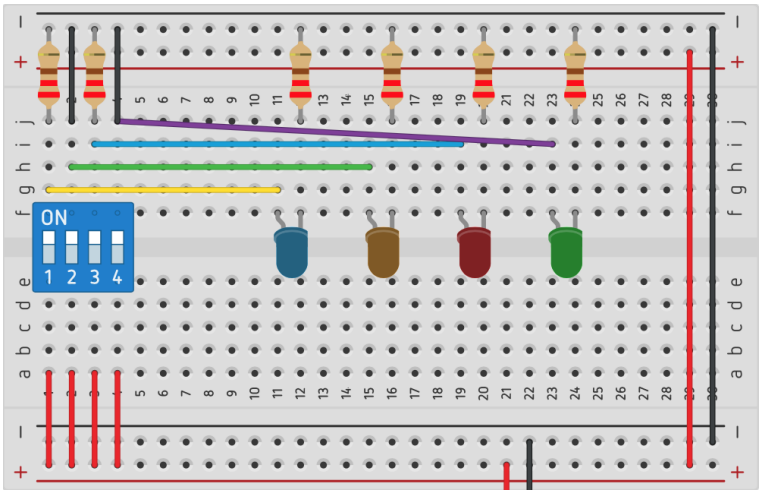


2. For each of the following circuits, explain why the LEDs are not all lit up. All of the power and ground rails are connected properly to a power supply set to 5 V.

   (a) Why is the LED corresponding to input 2 (B) off?



   (b) Why are all of the LEDs off?



Pre-Lab 2

Carefully read the entirety of Lab 2, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Use any reliable resource to determine the truth table for a NOT gate, and record it below. You will use this to verify the functionality of Circuit 1 in this lab.

A	F
0	
1	

2. Use any reliable resource to determine the truth table for an AND gate, and record it below. You will use this to verify the functionality of Circuit 2 in this lab.

A	B	F
0	0	
0	1	
1	0	
1	1	

3. Based on your answer above, determine the truth table for a 3-input AND gate, and record it below. You will use this to verify the functionality of Circuit 4 in this lab.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

4. Use any reliable resource to determine the truth table for an OR gate, and record it below. You will use this to verify the functionality of Circuit 3 in this lab.

A	B	F
0	0	
0	1	
1	0	
1	1	

5. Based on your answer above, determine the truth table for a 3-input OR gate, and record it below. You will use this to verify the functionality of Circuit 5 in this lab.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

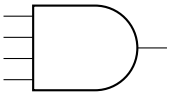
6. How many resistors will you need to use when wiring up Circuit 5? The circuit has three inputs and one output signal.

7. What will happen to a digital logic chip if the power and ground connections are swapped?

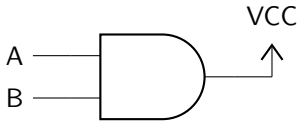
8. What is meant by a floating input?

9. What will happen to the output of a logic chip that has a floating input?

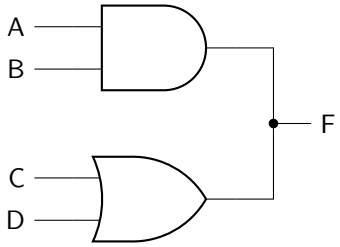
10. What is the fan-in of the following AND gate?



11. Describe the problem with the following circuit.



12. Describe the problem with the following circuit.



13. Identify the identity (name) of each of the following logic chips.

    (a) T74LS283B1

    (b) 74LS00PCQR

    (c) SN74HC595N

    (d) SN74ALS08N

Lab 2: Digital Circuits

This lab focuses on digital circuits, and will introduce basic logic functions. These logic functions will be described using circuit diagrams, truth tables, and expressions. The digital logic chips that contain logic functionality will also be introduced. These logic chips are part of a "family" of logic gates known as TTL. You will learn about how to use these logic chips, and then connect them together in several different circuits.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-2

2.1 Digital Logic Functions

There are three basic universal logic functions that can be used to build any digital circuit, regardless of its complexity. These logic functions are AND, OR, and NOT, and each is explained in detail in the class textbook.

Any digital circuit can be described using a circuit diagram, a truth table, or a Boolean expression. In labs, each of these has a valuable purpose. A circuit diagram provides the blueprint of how a circuit should be built using each individual logic gate and logic chip. The truth table can be used to either verify that a circuit is working as expected (if the circuit is already well-understood), or to record the information about how a circuit functions (if the circuit is not already well-understood). A Boolean expression acts as a convenient shorthand to describe a circuit in a manner that takes up less physical space than a circuit diagram or a truth table.

2.2 Transitor-Transistor Logic

The digital logic chips that will be used in this class are part of the Texas Instruments 7400 series, which are members of the TTL (Transistor-Transistor Logic) family. TTL devices use a particular type of transistor to create the logic gates internal to each chip. TTL chips have particular voltage values that must be present on the input in order to be registered as a logic HIGH or logic LOW value. When these conditions are present, the circuitry will generate a logic LOW or logic HIGH output value within certain voltage values specified by the TTL electronics. These input and output voltage ranges are provided in table 2.1. Note that the output values have tighter voltage values, which enables compatibility when the output of one logic chip is connected to the input of another logic chip.

Signal Type	LOW voltage values (V)	HIGH voltage values (V)
input	0–0.8	2–5
output	0–0.4	2.4–5

Table 2.1: Voltage values present on the input and output of a 5 V TTL logic chip given a logic HIGH or logic LOW condition.

2.2.1 Power Connections

Power connections must be made to every digital logic chip. This establishes the logical HIGH and logical LOW values used on the chip, and also power the internal circuitry, allowing it to function properly. Switching these two connections **will** cause a chip to overheat, and possibly melt. Therefore, special care must be taken to ensure that the chip is inserted properly and that all power connections are made to the correct power rail on the breadboard.

2.2.2 Floating Inputs

All inputs to a TTL logic circuit must have a clearly defined value, as given in table 2.1. Importantly, a logic circuit input cannot be floating. A floating input means that there is no electrical connection at all, either to VCC or GND. If there are any floating inputs on a logic gate (or even a more complicated logic device), the output may behave unpredictably. For this reason, all inputs must be connected to a properly wired DIP switch (or other input device such as a pushbutton, toggle switch, or keypad), or directly to VCC or GND.

2.2.3 Outputs

Different considerations must be made with logic circuit outputs. All logic circuit outputs will assert themselves (establishing either a logical HIGH or logical LOW output value) based on the input values and device functionality. For this reason, an output signal should **never** be directly connected to another signal value, such as VCC, GND, or another logic chip output. If this were to happen, a short circuit could result. This could cause (in the best case) the circuit to malfunction or (in the worst case) could damage the chip, possibly irreversibly.

2.2.4 Reading Part Numbers

The 7400 series of digital logic chips all have part numbers that indicate what the device is used for. These part numbers contain multiple parts. Each follows a similar format: XX74YYZZZPP. Each of these parts is explained below.

- **XX** – manufacturer (optional)
- **74** – 7400 series
- **YY** – device family (optional)
- **ZZZ** – part number
- **PP** – packaging (optional)

The most important value given is the part number. This provides the identity of the chip and its functionality. The part number of every chip used in your digital logic circuits should be verified before placing it into your breadboard.

As an example, consider the part number 74LS126N. This part number does not contain a manufacturer label. The 74 indicates that the device is part of the 7400 series of TTL logic devices and will be compatible with other TTL chips. The device family is LS, which explains exactly how the device is fabricated, and dictates its electrical properties, power consumption, and switching characteristics. The part number is 126. Looking at the list of 7400 chips, that identifies the chip as a Quad, 3-state buffers chip. Finally, the packaging is given by N, which indicates that the device is packaged in a plastic DIP format.

2.3 DIP Chips

As discussed in Lab 1, DIP stands for dual in-line package. DIP switches were used to create connections to digital logic signals. The chips containing digital logic gates also have a DIP architecture. As mentioned, the pins on a DIP chip are very fragile and can be bent and broken quite easily. Visually verify that all of the pins are present and in good shape before inserting a DIP chip into your breadboard. Each pin has a particular function. For this reason, it is important to understand how a DIP chip is oriented and how the pins are numbered. A DIP chip is shown in figure 2.1.

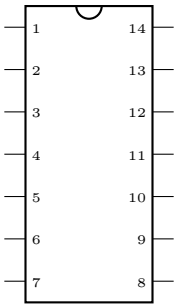


Figure 2.1: Diagram of a DIP chip with 14 pins.

The notch on the DIP chip indicates which end is up. When placing a DIP chip into a breadboard, ensure that the notch points to the top of the breadboard. The pin numbering starts at the pin in the upper left, and continues counter-clockwise around the chip.

A pinout diagram is used to determine the purpose of each of the pins on a logic chip. First, the part number must be established. Second, the notch must be placed in the correct orientation. The pinout diagram can then be used. Pinout diagrams for all of the chips used in your labs are provided in appendix A. An example pinout diagram is provided in figure 2.2 for the 7408 chip.

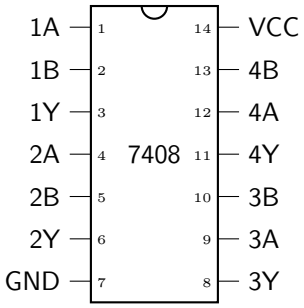


Figure 2.2: Pinout diagram of the 7408 chip.

A datasheet is a more complete set of information describing a digital logic chip. The datasheet contains a pinout diagram, but also a complete description, logic diagrams, electrical properties, switching characteristics, and other important information. A datasheet must be consulted if more detailed information about a digital logic chip is required than just the label corresponding to each pin.

Circuit 1: Refer to the pinout diagram for the 7404 chip. Use it to wire up a NOT circuit, using a DIP switch as the logic circuit input and an LED as the logic circuit output. Draw the circuit diagram. When you have completed your circuit, verify that it works. Then, demonstrate your results to your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 2: Refer to the pinout diagram for the 7408 chip. Use it to wire up an AND circuit, using a DIP switch as the logic circuit input and an LED as the logic circuit output. Draw the circuit diagram. When you have completed your circuit, verify that it works. Then, demonstrate your results to your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 3: Refer to the pinout diagram for the 7432 chip. Use it to wire up an OR circuit, using a DIP switch as the logic circuit input and an LED as the logic circuit output. Draw the circuit diagram. When you have completed your circuit, verify that it works. Then, demonstrate your results to your instructor to receive a stamp.

Instructor Stamp: _____

2.4 Fan-In

Fan-in corresponds to the number of inputs that a logic gate contains. The 7408 and 7432 chips used in the previous circuits have a fan-in of two, as each individual logic gate only has two inputs. In the labs in this class, the fan-in of all logic chips will be two, due to a limited inventory of parts. Therefore, it is important to understand how to use logic gates with a fan-in of two to act as logic gates with larger fan-in (such as three or four, which could be necessary for a 3-input gate or a 4-input gate).

Circuit 4: Determine how to wire up a 3-input AND gate. When you are finished, draw the circuit diagram and wire up the circuit on your breadboard. Use an LED to determine the output values. Demonstrate its functionality to your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 5: Determine how to wire up a 3-input OR gate. When you are finished, draw the circuit diagram and wire up the circuit on your breadboard. Use an LED to determine the output values. Demonstrate its functionality to your instructor to receive a stamp.

Instructor Stamp: _____

2.5 XOR and XNOR

XOR and XNOR logic gates are not part of the family of universal logic gates (NOT, AND, OR), meaning that they cannot be used to build every digital logic circuit. However, they may still provide a useful function in reducing the complexity of a circuit that you need to build. The XOR chip (7486) is available to use in lab. However, there is no XNOR chip. Understanding how to wire this up using the components that are available will be helpful in future labs.

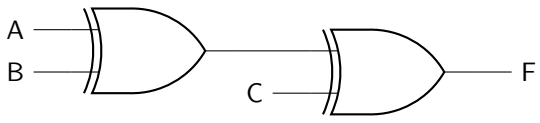
Circuit 6: Determine how to wire up a 2-input XNOR gate. (There is no XNOR chip available to use, so another way to build this will need to be determined.) When you are finished, draw the diagram and wire up the circuit on your breadboard. Use an LED to determine the output values. Fill out the corresponding truth table. Demonstrate its functionality to your instructor to receive a stamp.

A	B	F
0	0	
0	1	
1	0	
1	1	

Instructor Stamp: _____

Circuit 7: Wire up the following circuit on your breadboard. Use an LED to determine the output values. Fill out the corresponding truth table. Demonstrate its functionality to your instructor to receive a stamp.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



Instructor Stamp: _____

Lab 2 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

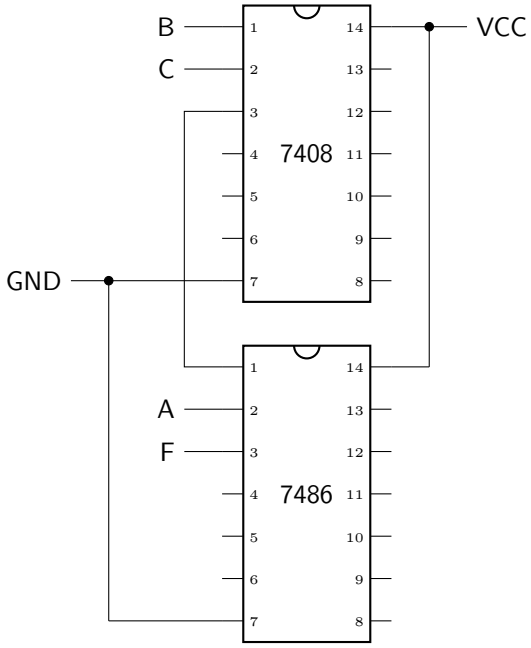
1. The 4069 integrated circuit is an inverter chip using CMOS technology. The logic levels that it uses are shown in the table below.

Signal Type	LOW voltage values (V)	HIGH voltage values (V)
input	0–1	4–5
output	0–0.05	4.95–5

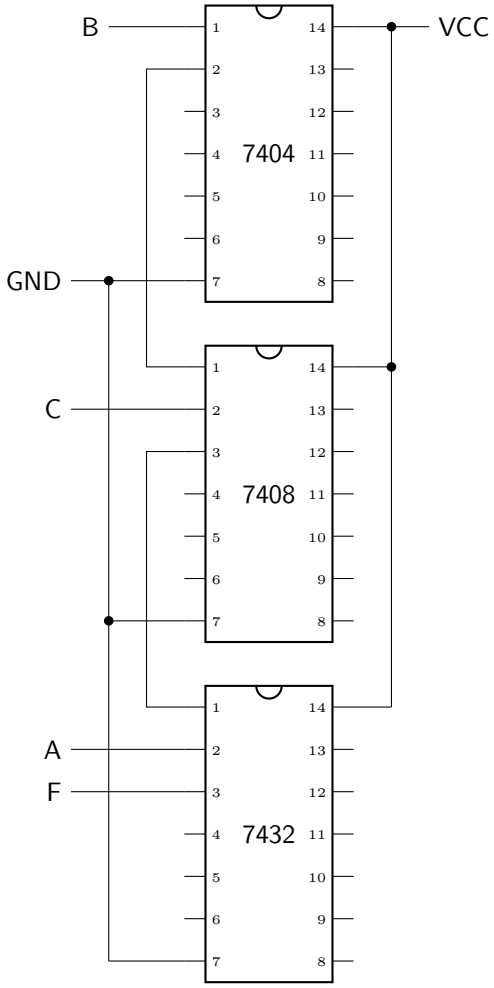
   (a) Can you use the 4069 chip as an input to a TTL logic gate and be certain that the circuit will function correctly? Why or why not?

   (b) Can you use a TTL logic gate as an input to the 4069 chip and be certain that the circuit will function correctly? Why or why not?

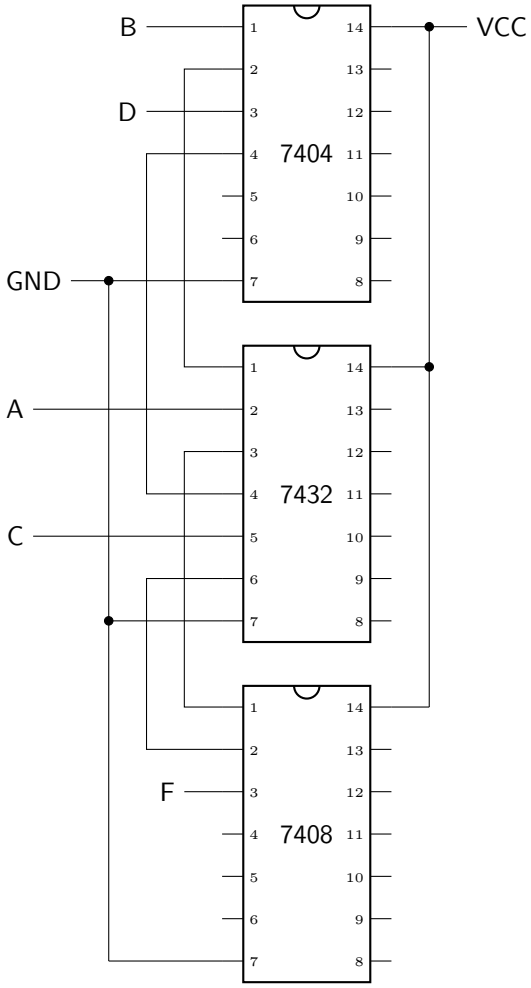
2. What digital logic expression is being implemented by the circuit shown below? Draw a circuit diagram and write an expression.



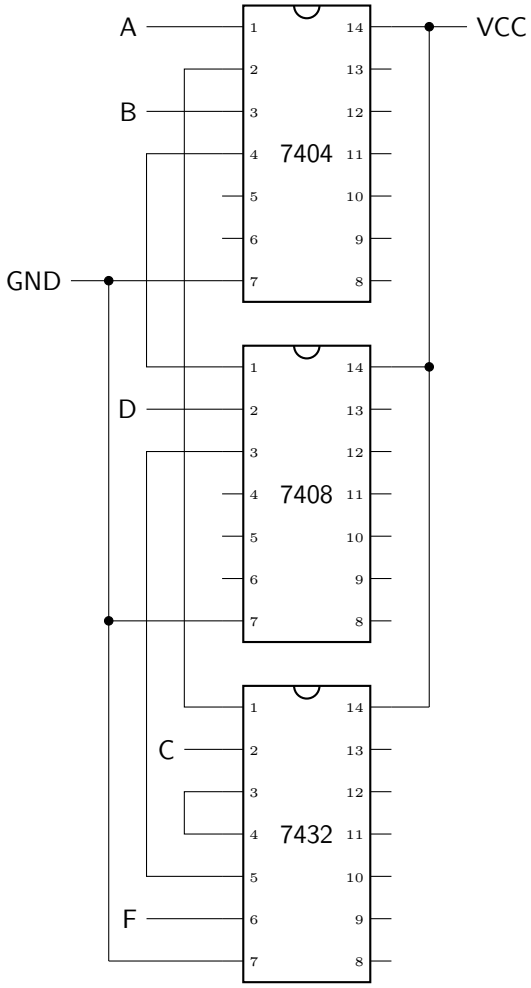
3. What digital logic expression is being implemented by the circuit shown below? Draw a circuit diagram and write an expression.



4. What digital logic expression is being implemented by the circuit shown below? Draw a circuit diagram and write an expression.



5. What digital logic expression is being implemented by the circuit shown below? Draw a circuit diagram and write an expression.



Pre-Lab 3

Carefully read the entirety of Lab 3, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Consult the datasheet for each of the following logic chips. For each signal, determine if it is an active-HIGH or active-LOW input. How do you know?

   (a) Pin 3 (lamp test) on the 7447 chip.

   (b) Pin 4 (preset) on the 74112 chip.

   (c) Pin 1 (enable) on the 74126 chip.

   (d) Pin 1 (clear) on the 74194 chip.

Lab 3: Digital Input and Output Signals

Now that you have a preliminary understanding of digital logic circuits, this lab will introduce some concepts that will be utilized throughout the semester. First, you will explore the difference between active-HIGH and active-LOW input and output signals. Then, you will purposefully make some mistakes with the wiring of your digital logic circuits to determine the effect it has on the output signal. This will be particularly useful in the next lab that specifically covers the topic of troubleshooting.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-3

3.1 Active-HIGH and Active-LOW Inputs

So far in this class, we have worked with active-HIGH signals. An active-HIGH signal means that an input signal is asserted when its value is 1 (VCC) and that an input signal is de-asserted when its value is 0 (GND). Active-LOW means the opposite: an input is asserted when 0 and de-asserted when 1. (Note the use of the term asserted and de-asserted. Saying that an input is "asserted" means that it is activated, which is independent of whether it is active-HIGH or active-LOW. This avoids getting confused about actual signal levels.)

While the concept of active-LOW may seem strange right now, many digital logic chips that you will utilize this semester will have active-LOW input signals. It is therefore important to understand what that means and how to work with them.

An active-LOW input appears on a circuit pinout diagram with an overline over the signal name (example: $\overline{\text{CLR}}$) and a bubble on the pin. For example, in Figure 3.1, pins 1, 4, 10, and 13 are active-LOW input pins. (Pins 6 and 8 are active-LOW output pins.)

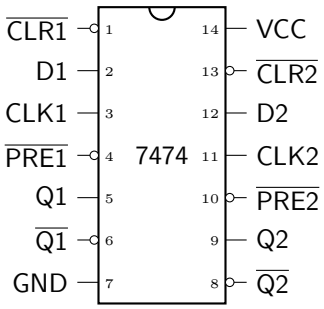


Figure 3.1: Pinout diagram of the 7474 chip.

When connecting an active-HIGH signal that should continuously remain asserted, simply connect it to VCC. To de-assert, connect to GND. The reverse is true for active-LOW inputs that must remain continuously asserted or de-asserted.

For active-HIGH inputs connected to an input device such as a DIP switch or a pushbutton, an active-HIGH configuration using a pull-down resistor, as you have done so far in Labs 1 and 2, is ideal. To connect an active-LOW input, there are two possibilities. The first is using an active-HIGH DIP switch or pushbutton input that has been inverted using a NOT gate (see Figure 3.2, left). The second option is to use an active-LOW input signal by using a pull-up resistor instead of a pull-down resistor (see Figure 3.2, right).

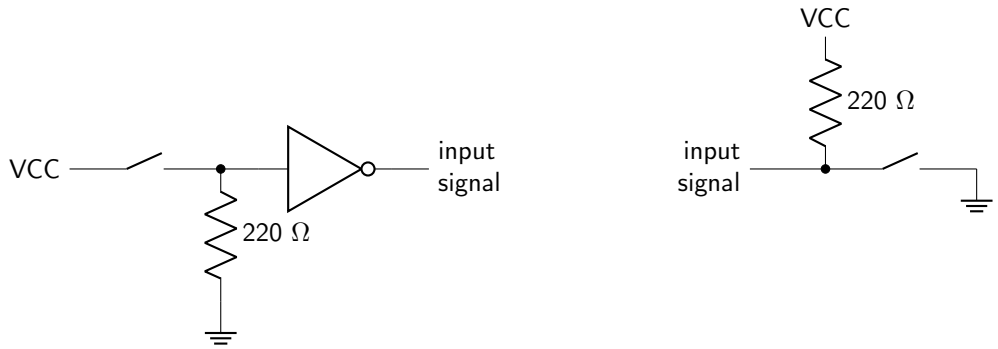


Figure 3.2: Schematic of an active-HIGH DIP switch input connected through an inverter (left). Schematic of an active-LOW DIP switch input (right).

Circuit 1: Refer to the pinout diagram for the 7408 chip. Use it to wire up an AND circuit. First, to remind yourself of the truth table of an AND gate, wire the circuit as you did in Lab 2. (This is for your own recollection, there is no need to demonstrate this to your instructor.)

Once you feel familiar with the "normal" AND gate, change your DIP switch so that both inputs have an active-LOW input configuration using pull-up resistors, as shown in Figure 3.2 right. Fill out the corresponding truth table. Demonstrate its functionality to your instructor to receive a stamp.

A	B	F
0	0	
0	1	
1	0	
1	1	

Instructor Stamp: _____

Circuit 2: Refer to the pinout diagram for the 7432 chip. Use it to wire up an OR circuit. First, to remind yourself of the truth table of an OR gate, wire the circuit as you did in Lab 2. (This is for your own recollection, there is no need to demonstrate this to your instructor.)

Once you feel familiar with the "normal" OR gate, change your DIP switch so that both inputs have an active-LOW input configuration using pull-up resistors. Fill out the corresponding truth table. Demonstrate its functionality to your instructor to receive a stamp.

A	B	F
0	0	
0	1	
1	0	
1	1	

Instructor Stamp: _____

Circuit 3: Using four active-HIGH DIP switch inputs (pull-down resistors), wire up Mystery Box 1. Complete the truth table, and show it to your instructor to receive a stamp. You will use this in the lab homework.

EN	A	B	C	F
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Instructor Stamp: _____

Circuit 4: Using four active-HIGH DIP switch inputs (pull-down resistors), wire up Mystery Box 2. Complete the truth table, and show it to your instructor to receive a stamp. You will use this in the lab homework.

EN	A	B	C	F
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Instructor Stamp: _____

3.2 Active-HIGH and Active-LOW Outputs

Just as input signals can be active-HIGH or active-LOW, output signals can also be active-HIGH or active-LOW. An active-LOW output has a value of 0 when activated, and an active-HIGH output has a value of 1 when activated. While this may also seem strange, many digital logic chips have active-LOW outputs. This is due to historic limitations on the ability to source output current in digital logic chips, which meant that an external power supply connected to an active-LOW output was used to generate that current instead. While today we have output devices (such as LEDs) that require minimal current, and chips that can source more current than previously, these historical limitations still dictate the design of each logic chip's pinout. (For example: once the 74138 chip was designed with active-LOW outputs, all future versions of that same chip must also have active-LOW outputs.)

Sometimes, active-LOW outputs require no modification to interact with other digital circuit components. For example, in Lab 7, you will learn about common anode 7-segment displays, which are designed to work with active-LOW output decoders.

Other times, an active-LOW output signal needs to be modified to interact with another active-HIGH device. In this case, an inverter can be used to convert an active-LOW signal into an active-HIGH signal. When utilizing an output device such as an LED with an active-LOW output device, it is not always necessary to invert the output (which requires an extra logic gate). Instead, an LED can be connected in an active-LOW configuration. In this case, rather than wiring up an LED in an active-HIGH configuration (as you've done in previous labs, and as shown in Figure 3.3, left) wire it instead in an active-LOW configuration (as shown in Figure 3.3, right).

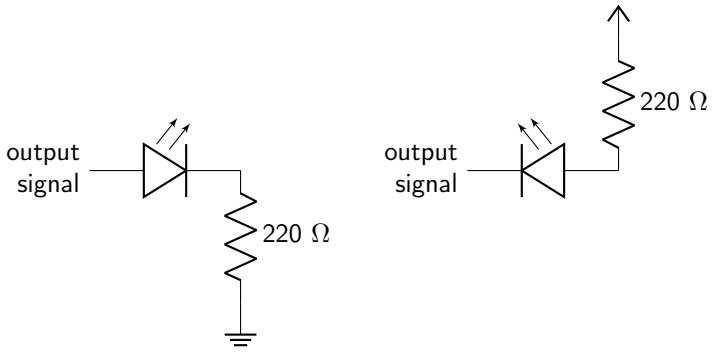


Figure 3.3: Schematic of an LED wired for an active-HIGH output signal (left). Schematic of an LED wired for an active-LOW output signal (right).

Circuit 5: Wire up an AND gate with active-HIGH inputs (pull-downs on the DIP switch) and an LED wired in an active-LOW configuration, as shown in Figure 3.3 right. Fill out the corresponding truth table. Demonstrate its functionality to your instructor to receive a stamp.

A	B	F
0	0	
0	1	
1	0	
1	1	

Instructor Stamp: _____

Circuit 6: Wire up an OR gate with active-HIGH inputs (pull-downs on the DIP switch) and an LED wired in an active-LOW configuration. Fill out the corresponding truth table. Demonstrate its functionality to your instructor to receive a stamp.

A	B	F
0	0	
0	1	
1	0	
1	1	

Instructor Stamp: _____

3.3 Making Mistakes on Purpose

Inevitably, mistakes will be made throughout the semester as you build digital logic circuits. Making mistakes is not a problem in itself, but it can lead to a frustrating troubleshooting process. While the steps of troubleshooting will be explored in the next lab, the purpose of the next few circuits is to determine the symptoms that occur when some particularly common mistakes are made in the wiring of digital logic circuits.

For the three basic logic gates, you will purposefully wire the circuits wrong in different ways. Make observations about the effect these mistakes have on the output. Hopefully, if you notice these symptoms arising in the future, you will be familiar with them and be able to quickly find the cause.

Circuit 7: Refer to the pinout diagram for the 7404 chip. Use it to wire up a NOT circuit, using an active-HIGH DIP switch. First, verify that the circuit works as intended when wired properly as you did in Lab 2. Then, make the following mistakes (only one at a time) and make observations about how the output is affected.

remove the pull-down resistor from the DIP switch input	remove the VCC connection from the DIP switch input	remove the VCC connection from the 7404 chip	remove the GND connection from the 7404 chip

When you have finished collecting this data, discuss it with your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 8: Refer to the pinout diagram for the 7408 chip. Use it to wire up an AND circuit, using an active-HIGH DIP switch. First, verify that the circuit works as intended when wired properly as you did in Lab 2. Then, make the following mistakes (only one at a time) and make observations about how the output is affected.

remove the pull-down resistor(s) from the DIP switch input	**remove the VCC connection(s) from the DIP switch input**	**remove the VCC connection from the 7408 chip**	**remove the GND connection from the 7408 chip**

When you have finished collecting this data, discuss it with your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 9: Refer to the pinout diagram for the 7432 chip. Use it to wire up an OR circuit, using an active-HIGH DIP switch. First, verify that the circuit works as intended when wired properly as you did in Lab 2. Then, make the following mistakes (only one at a time) and make observations about how the output is affected.

remove the pull-down resistor(s) from the DIP switch input	**remove the VCC connection(s) from the DIP switch input**	**remove the VCC connection from the 7432 chip**	**remove the GND connection from the 7432 chip**

When you have finished collecting this data, discuss it with your instructor to receive a stamp.

Instructor Stamp: _____

Lab 3 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Which of the mystery boxes has an active-HIGH enable input? How do you know?

2. Which of the mystery boxes has an active-LOW enable input? How do you know?

3. Consider the 7404 chip wired up with both an active-LOW DIP switch (as shown in Figure 3.2 right) **and** an active-LOW LED (as shown in Figure 3.3 right). Fill out the following truth table for that configuration. (Feel free to test this out in TinkerCad if you're unsure of your answer.)

A	F
0	
1	

4. Consider the 7408 chip wired up with both an active-LOW DIP switch (as shown in Figure 3.2 right) **and** an active-LOW LED (as shown in Figure 3.3 right). Fill out the following truth table for that configuration. (Feel free to test this out in TinkerCad if you're unsure of your answer.)

A	B	F
0	0	
0	1	
1	0	
1	1	

5. Consider the 7432 chip wired up with both an active-LOW DIP switch (as shown in Figure 3.2 right) **and** an active-LOW LED (as shown in Figure 3.3 right). Fill out the following truth table for that configuration. (Feel free to test this out in TinkerCad if you're unsure of your answer.)

A	B	F
0	0	
0	1	
1	0	
1	1	

Pre-Lab 4

Carefully read the entirety of Lab 4, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Derive a truth table for Circuit 1.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

2. Derive a truth table for Circuit 2.

A	B	F
0	0	
0	1	
1	0	
1	1	

3. Derive a truth table for Circuit 3.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

4. Derive a truth table for Circuit 4.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

5. Derive a truth table for Circuit 5.

A	B	F
0	0	
0	1	
1	0	
1	1	

6. Derive a truth table for Circuit 6.

A	B	F
0	0	
0	1	
1	0	
1	1	

7. Write up a rough draft of your process flow for debugging a circuit. Include at least four steps (be as specific as possible), and place them in the order you will use them.

Lab 4: Troubleshooting

Troubleshooting is an important part of building digital circuits. Frequently, one or more errors in a design will lead to incorrect output values, or undesired effects such as short circuits. This lab will present you with multiple circuits that don't work. Your job will be to diagnose the problem(s) that exist in the circuit.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-4

4.1 Circuit Debugging

In order to debug a circuit, it is important to first understand exactly what the circuit is **supposed** to do. With combinational circuits, a truth table can be used to determine what the circuit output(s) should be for each combination of input variables. Next, it is important to understand what the circuit **does** do.

After comparing the ideal circuit output with the real circuit output, then it's time to start going through your list of troubleshooting steps and finding the problem(s) and correcting them.

4.2 Logic Probe

A logic probe can be used to quickly determine the logic level (HIGH or LOW) of any connection in a digital circuit. A thin metal probe can be inserted into a breadboard hole, on top of a logic chip pin, or on the ends of an LED or resistor. The internal circuitry of the logic probe causes a sound to be emitted, and an LED to light up, based on the value of the signal at that location.

Logic probes are superior to using a single LED to debug a circuit because it is able to distinguish between a LOW and a float, which a single LED is unable to do. Some care must be taken on where a logic probe can be placed, however. Only digital logic signals of HIGH and LOW will be valid. Anything that is in between those two voltage ranges does not correspond to a digital logic level and therefore may not read correctly on the logic probe. This will predominantly occur in the case of a current-limiting resistor and LED combination.

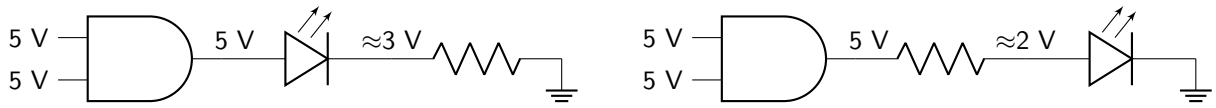
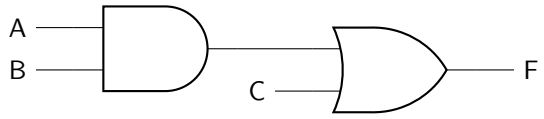


Figure 4.1: Circuit diagrams showing the voltage values present at various connections around a current-limiting resistor an LED. Whether the resistor is placed at the cathode of the LED (left) or anode of the LED (right), the voltage level in between the two circuit elements will be indeterminate.

Figure 4.1 shows voltage values in an example circuit where the output of a logic gate is HIGH. The connection made directly after the output of the logic gate will be 5 V, a well-established logic signal value. However, whether the current-limiting resistor is placed at the cathode of the LED (figure 4.1 left) or anode of the LED (figure 4.1 right), the voltage level in between the resistor and LED will not correspond to any logic level. The readout of a logic probe used at the intersection of an LED and a resistor is therefore indeterminate and should not be probed to determine the output of a logic gate.

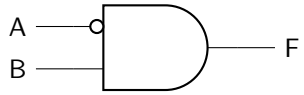
Circuit 1: Circuit 1 is defined by the circuit diagram below. Make as many observations about the circuit as you need to determine what is wrong with the circuit. When you have come to a conclusion, discuss it with your instructor to receive a stamp.



symptom	cause	how you found the cause

Instructor Stamp: _____

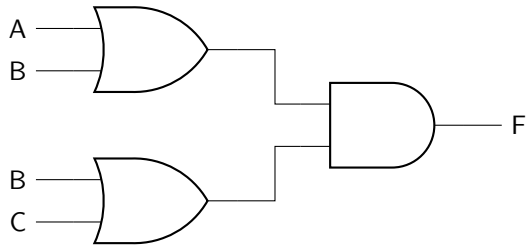
Circuit 2: Circuit 2 is defined by the circuit diagram below. Make as many observations about the circuit as you need to determine what is wrong with the circuit. When you have come to a conclusion, discuss it with your instructor to receive a stamp.



symptom	cause	how you found the cause

Instructor Stamp: _____

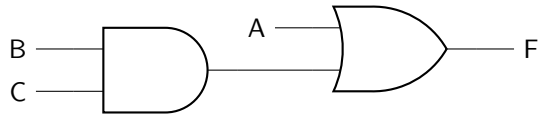
Circuit 3: Circuit 3 is defined by the circuit diagram below. Make as many observations about the circuit as you need to determine what is wrong with the circuit. When you have come to a conclusion, discuss it with your instructor to receive a stamp.



symptom	cause	how you found the cause

Instructor Stamp: _____

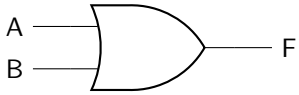
Circuit 4: Circuit 4 is defined by the circuit diagram below. Make as many observations about the circuit as you need to determine what is wrong with the circuit. When you have come to a conclusion, discuss it with your instructor to receive a stamp.



symptom	cause	how you found the cause

Instructor Stamp: _____

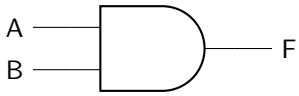
Circuit 5: Circuit 5 is defined by the circuit diagram below. Make as many observations about the circuit as you need to determine what is wrong with the circuit. When you have come to a conclusion, discuss it with your instructor to receive a stamp.



symptom	cause	how you found the cause

Instructor Stamp: _____

Circuit 6: Circuit 6 is defined by the circuit diagram below. Make as many observations about the circuit as you need to determine what is wrong with the circuit. When you have come to a conclusion, discuss it with your instructor to receive a stamp.



symptom	cause	how you found the cause

Instructor Stamp: _____

Lab 4 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. What are 5 possible causes of: the output of your circuit is (or appears to be) always LOW. These 5 causes should be specific, and should have the result of always creating or appearing to have a LOW output. (For example, "using the wrong resistor" is both not specific enough and not necessarily a cause of an appearance of an always LOW output. Which resistor? Does wrong mean too high or too low? Be specific!)

2. What are 5 possible causes of: some outputs of your circuit are correct, while others are not. These 5 causes should be specific, and should have the result of causing the outputs to be sometimes correct and sometimes incorrect, but not always LOW or always HIGH. (For example, "having misplaced wires" is not specific enough. Specifically, which wire(s)? How would those wires be placed incorrectly? Be specific!)

3. What are 3 possible causes of: the "current limited" LED on the power supply lights up. As with the previous questions, your answers must be specific. (For example, "wrong resistor value" is not specific enough and not necessarily a cause of making the current limited LED turn on. Which resistor? Too high or too low a value? Be specific!)

4. Now that you have debugged several circuits, revise your process flow from the pre-lab. Include at least six steps, and place them in the order you will use them. You will use this process flow to help debug future circuits. It is important for these steps to be as specific as possible. For example "troubleshoot the circuit" is not a specific debugging step. Continue adding to this list as you come up with new things to check.

Pre-Lab 5

Carefully read the entirety of Lab 5, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Derive a minimum SOP or minimum POS expression for Circuit 5.

2. Derive a minimum SOP or minimum POS expression for Circuit 6.

3. Derive a minimum SOP or minimum POS expression for Circuit 7.

4. Derive a minimum SOP or minimum POS expression for Circuit 8.

5. Derive a minimum SOP or minimum POS expression for the red LED in Circuit 9.

6. Derive a minimum SOP or minimum POS expression for the yellow LED in Circuit 9.

7. Derive a minimum SOP or minimum POS expression for the green LED in Circuit 9.

Lab 5: Boolean Algebra

This lab will focus on minimizing circuits to make them as simple to build on a breadboard as possible.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-5

5.1 Boolean Algebra

Boolean algebra refers to the type of mathematics that is used to express logical functions (using AND, OR, and NOT). It also contains many laws and theorems that can be used to minimize an expression. A minimum expression has the fewest terms and/or variables possible for any given logic function. The textbook outlines each of these laws and theorems, and explains how to find minimum expressions.

In lab, the goal is to make circuits as easy to wire as possible. Because there are fewer parts in a minimum circuit, there will also be less to troubleshoot and debug.

Circuit 1: Fill out the following truth table for Mystery Box 1. When you have completed the truth table, show it to your instructor to receive a stamp. (You will derive a minimum expression, using Boolean algebra, in the lab homework.)

A	B	F
0	0	
0	1	
1	0	
1	1	

Instructor Stamp: _____

Circuit 2: Fill out the following truth table for Mystery Box 2. When you have completed the truth table, show it to your instructor to receive a stamp. (You will derive a minimum expression, using Boolean algebra, in the lab homework.)

A	B	F
0	0	
0	1	
1	0	
1	1	

Instructor Stamp: _____

Circuit 3: Fill out the following truth table for Mystery Box 3. When you have completed the truth table, show it to your instructor to receive a stamp. (You will derive a minimum expression, using Boolean algebra, in the lab homework.)

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Instructor Stamp: _____

Circuit 4: Fill out the following truth table for Mystery Box 4. When you have completed the truth table, show it to your instructor to receive a stamp. (You will derive a minimum expression, using Boolean algebra, in the lab homework.)

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Instructor Stamp: _____

Circuit 5: Build the circuit corresponding to the minterm expression given below. Draw your circuit diagram. Then, build the circuit and confirm that it operates as you expect it to. Demonstrate its functionality to your instructor to receive a stamp.

$$F(A, B, C) = \Sigma m(2, 3, 4, 5, 7)$$

Instructor Stamp: _____

Circuit 6: Build the circuit corresponding to the minterm expression given below. Draw your circuit diagram. Then, build the circuit and confirm that it operates as you expect it to. Demonstrate its functionality to your instructor to receive a stamp.

$$F(A, B, C, D) = \Sigma m(1, 2, 3, 5, 9, 10, 11, 13, 14)$$

Instructor Stamp: _____

Circuit 7: Build the circuit corresponding to the maxterm expression given below. Draw your circuit diagram. Then, build the circuit and confirm that it operates as you expect it to. Demonstrate its functionality to your instructor to receive a stamp.

$$F(A, B, C, D) = \Pi M(1, 2, 5, 6, 7, 9, 13, 15)$$

Instructor Stamp: _____

Circuit 8: Build the circuit corresponding to the maxterm expression given below. Draw your circuit diagram. Then, build the circuit and confirm that it operates as you expect it to. Demonstrate its functionality to your instructor to receive a stamp.

$$F(A, B, C, D) = \Pi M(0, 1, 4, 5, 6, 7)$$

Instructor Stamp: _____

Circuit 9: A row of parking spaces has 15 spots. If more than half are empty, a green LED (F_G) should light. If fewer than two spots are empty, a red LED (F_R) should light. Otherwise, a yellow LED (F_Y) should light. You receive a 4-bit binary number ($ABCD$) corresponding to the number of empty spaces. Design a circuit to fulfill this function. Draw your circuit diagrams. Then, wire the circuit on your breadboard and fill out the corresponding truth table. Demonstrate its functionality to your instructor to receive a stamp.

A	B	C	D	FR	FY	FG
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Instructor Stamp: _____

Lab 5 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Use Boolean algebra to find the minimum SOP or minimum POS expression for Mystery Box 1. Then, draw a circuit diagram of this simplified circuit.

2. Use Boolean algebra to find the minimum SOP or minimum POS expression for Mystery Box 2. Then, draw a circuit diagram of this simplified circuit.

3. Use Boolean algebra to find the minimum SOP or minimum POS expression for Mystery Box 3. Then, draw a circuit diagram of this simplified circuit.

4. Use Boolean algebra to find the minimum SOP or minimum POS expression for Mystery Box 4. Then, draw a circuit diagram of this simplified circuit.

5. Use Boolean algebra to convert $A'B' + B'C' + C'D$ to a minimum POS expression. (Do a conversion; do not simply write up a truth table and derive the POS expression.)

6. Use Boolean algebra to convert $CD + A'C + AC'D'$ to a minimum POS expression. (Do a conversion; do not simply write up a truth table and derive the POS expression.)

7. There is an elevator in a two-story building. Each floor has a button that can be pressed to call for the elevator. You know on which floor the elevator is at any time (it is OK to assume that if the elevator is not on the first floor, then it is on the second floor), and whether or not each button has been pressed. If a button has been pressed on the floor where the elevator already is, the door needs to open. If a button has been pressed on a floor where the elevator isn't present, the elevator motor needs a signal to go either up or down. If both buttons have been pressed, the elevator doors must open before the motor moves in order to let people in on the current floor. Design three circuits (one to instruct the door to open, one to send the elevator up, one to send the elevator down). Clearly define each of the variables, including the significance of a 0 and 1 value. All of the input variables will be shared between the three circuits and do not need to be defined more than once. Show all work.

Pre-Lab 6

Carefully read the entirety of Lab 6, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Derive a minimum expression for the sum of a full adder, $\Sigma(A, B, C_{in})$.

2. Derive a minimum expression for the carry-out of a full adder, $C_{out}(A, B, C_{in})$.

3. What should the carry in pin on the 74283 chip be connected to in Circuit 4?

4. Carefully read Circuit 4. Fill out the following truth table and derive a minimum expression for the overflow signal, $O(A_4, B_4, \Sigma_4)$.

A4	B4	Σ4	O
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Lab 6: Binary Adders

This lab focuses on binary adders, which are a critical part of computer arithmetic and logic units. The lab will start by focusing on simple one-bit adders, connecting them together into a two-bit adder, and then using a 4-bit adder digital logic chip.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-6

6.1 Binary Adders

It is vitally important for computers and other devices to be able to add binary numbers together. Not only is addition one of the basic arithmetic operations, an adder in a computer arithmetic and logic unit (ALU) can be used to subtract, increment, or decrement numbers. These are all important functions for a computer processor.

Binary adders are defined by how many bits each addend can be. A one-bit adder can add together two one-bit binary numbers, and so on. However many bits a binary adder is, it still consists of the basic building blocks of either a half adder or a full adder.

6.2 Half Adder

A half adder is able to take two one-bit binary numbers as inputs (A and B) and output the sum (Σ) and carry-out (C_{out}) term. Because a half adder cannot handle any carry-in terms, it is unable to be cascaded together to create a larger adder, such as a two-bit adder or four-bit adder. However, it is still a useful logic device and helps assist in the understanding of how binary adders work.

Circuit 1: Build a half adder circuit, including both outputs (Σ and C_{out}). Draw a circuit diagram for each output. When you have verified that the circuit works, demonstrate its functionality to your instructor to receive a stamp. **When finished: keep this half adder on the breadboard. You will add to it in Circuit 3.**

Instructor Stamp: _____

6.3 Full Adder

A full adder has three inputs: the two terms being summed together (A and B) and an input carry term (C_{in}). The outputs are the sum (Σ) and carry-out (C_{out}) terms. Because adding binary numbers that are two-bits or larger, each bit with more weight than the least-significant bit (LSB) will need to include the carry out of the previous bit. This carry-out of bit n becomes the carry-in of bit $n+1$.

Circuit 2: Build a full adder circuit, including both outputs (Σ and C_{out}). Draw a circuit diagram for each output. When you have verified that the circuit works, demonstrate its functionality to your instructor to receive a stamp. **When finished: keep this full adder on the breadboard. You will add to it in Circuit 3.**

Instructor Stamp: _____

6.4 Ripple Carry Adder

A ripple carry adder is created when two or more adders are cascaded together to add together binary numbers that are greater than one-bit in length. The carry-out from one stage becomes the carry-in of the next stage. If the first stage represents the LSB and requires no carry-in term, then a half adder can be used for that stage instead of a full adder. A four-bit ripple carry adder is depicted schematically in figure 6.1.

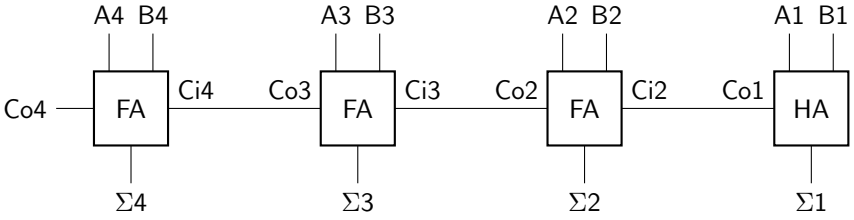


Figure 6.1: Schematic diagram of a four-bit ripple carry adder.

Circuit 3: Connect your half adder and full adder in such a way as to create a two-bit ripple carry adder. There will be two two-bit binary inputs, a two-bit sum (Σ) output and a one-bit final carry-out (C_{out2}) output. Draw a circuit diagram. When you have verified that the circuit works, demonstrate its functionality to your instructor to receive a stamp.

Instructor Stamp: _____

6.5 74283 4-Bit Full Adder

There are logic chips capable of conducting addition, without having to build each individual addition stage. The 74283 logic chip is a four-bit adder, capable of summing together two four-bit numbers. It also contains a carry-in term for the first stage so that two or more chips can be cascaded together to create even larger bit adders. The 74283 adder chip works by adding two binary numbers as follows.

$$\begin{array}{cccccc} C_{out} & C_3 & C_2 & C_1 & C_{in} \\ & A_4 & A_3 & A_2 & A_1 \\ + & B_4 & B_3 & B_2 & B_1 \\ \hline & \Sigma_4 & \Sigma_3 & \Sigma_2 & \Sigma_1 \end{array}$$

In other words, two 4-bit binary inputs $A_4A_3A_2A_1$ and $B_4B_3B_2B_1$ are added together and a value of $\Sigma_4\Sigma_3\Sigma_2\Sigma_1$ is the summation of those two binary numbers.

The first stage carry-in term C_{in} is used either for cascading adders together, or when conducting a two's complement subtraction operation. However, when simply adding together two numbers, as will be done in this lab, the C_{in} term will not be used. The 74283 datasheet indicates what should be done with the input.

> Note that with active HIGH inputs, Carry In can not be left open [floating], but must be held LOW when no carry in is intended.

This means that the carry in pin needs to be connected directly to ground during the following exercise.

Circuit 4: Use the 74283 four-bit full adder to display the sum of two four-bit numbers, and also to detect overflow. Wire up the 74283 adder with eight inputs corresponding to $A_4A_3A_2A_1$ and $B_4B_3B_2B_1$. Display the sum $\Sigma_4\Sigma_3\Sigma_2\Sigma_1$ on four LEDs. In addition, wire up your overflow logic to an additional LED that will light when an overflow occurs.

The block diagram of this circuit is shown in figure 6.2.

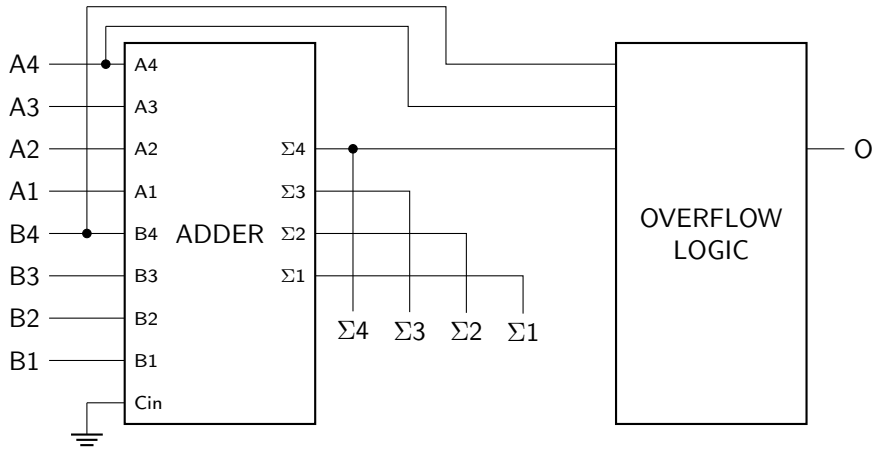


Figure 6.2: Block diagram of the four-bit adder with overflow logic.

Draw a circuit diagram for O on the next page. Verify that your circuit works. Then, demonstrate it to your instructor to receive a stamp by showing four addition examples that lead to an overflow, and four addition examples that do not. Include negative numbers in at least half of your examples. Use the four-bit two's complement table (table 6.1) for reference.

Instructor Stamp: _____

6.6 Four-Bit Two's Complement

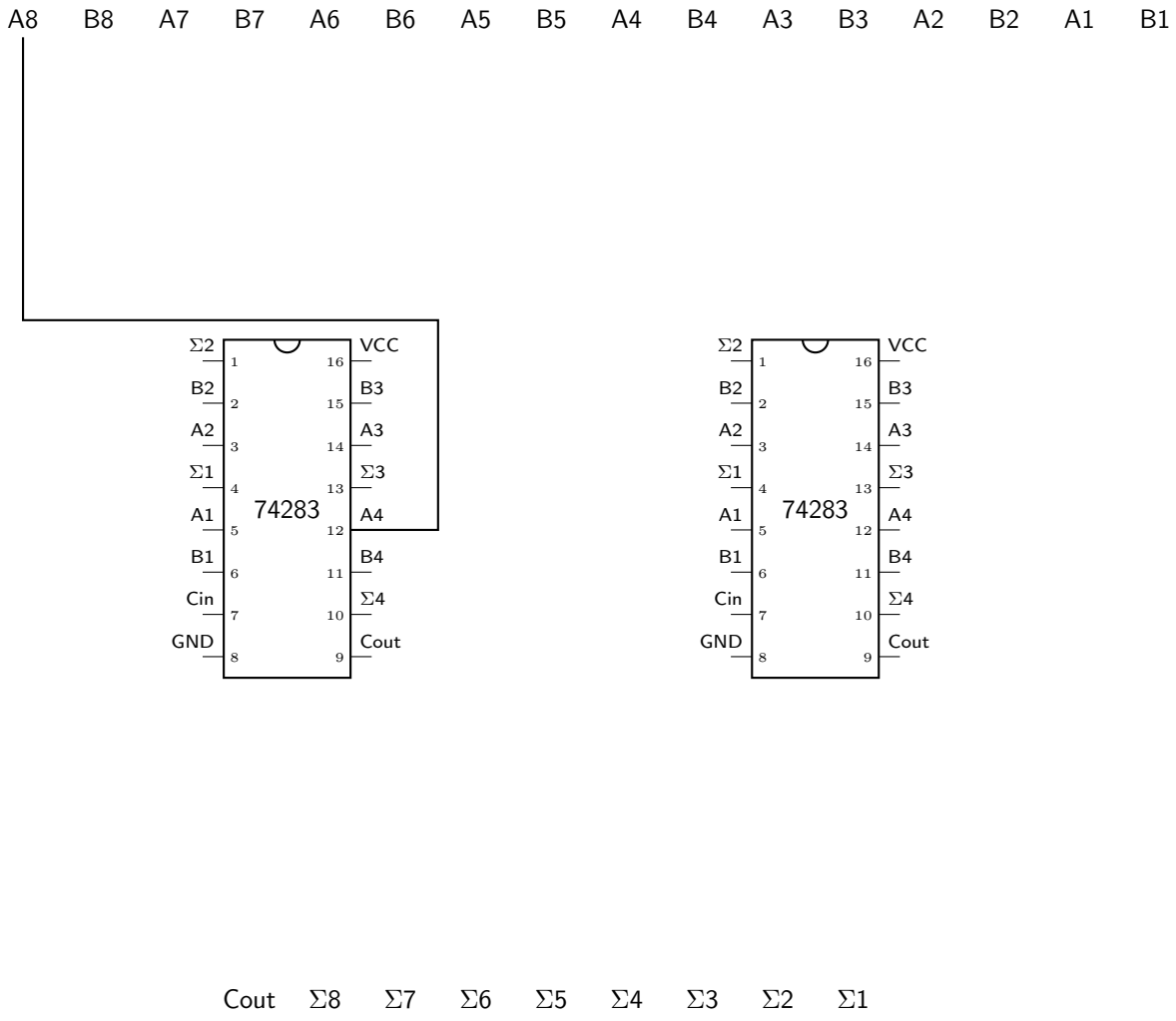
Decimal	Two's Complement
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
−1	1111
−2	1110
−3	1101
−4	1100
−5	1011
−6	1010
−7	1001
−8	1000

Table 6.1: A list of all of the four-bit two's complement numbers.

Lab 6 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Annotate the circuit diagram below to depict how you would make input and output connections to create an eight-bit adder out of two four-bit adders. The connection to $A8$ (the MSB) has been made for you as an example.



2. Using your TinkerCAD classroom account, build a 4-bit 2's complement subtractor using the 74283 chip and any other necessary logic gates. The DIP switch should have inputs in order (A4-A3-A2-A1 and B4-B3-B2-B1), where A4 and B4 are the MSBs (sign bits) and A1 and B1 are the LSBs. All binary values (inputs and the result) are 4-bit 2's complement values. The logic function of the output will be $F = A - B$. Using the notes tool in TinkerCAD, clearly label each input signal (which is A, which is B, and what is the MSB/LSB of each input) and each output LED (which output LED corresponds to the MSB of the output and which corresponds to the LSB). No overflow detection is required in this circuit. Clearly label your circuit so that it can easily be found in your TinkerCAD account. (Hint: you may want to rotate each component so that they are "vertical" as you would build them in lab rather than "horizontal" as is the default in TinkerCAD. This will help you ensure the orientation of each DIP switch is correct, among other things.)

3. How many four-bit adders would it take to add three four-bit numbers together ($\Sigma = A + B + C$)? Draw this as a block diagram.

4. How many four-bit adders would it take to add n four-bit numbers together?

5. If each AND, OR, and XOR chip has a propagation delay of approximately 5 ns, how much delay would there be in changing the input on a 4-bit ripple carry adder before the output is valid? Justify your answer.

6. Consult the datasheet for the 74283 logic chip to explain why the propagation delay of the 74283 logic chip is **only** approximately 15 ns.

Pre-Lab 7

Carefully read the entirety of Lab 7, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. In Circuit 1, you are asked to use the 5-2-2-1 BCD code as the input variable to decode into one segment of a 7-segment display. Use the truth table below to determine the decimal (base-10) value of each of the 5-2-2-1 numeral encodings.

A	B	C	D	Decimal Value
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

2. Fill out the truth table below based on Circuit 2.

A	B	C	D	Intensity (Bels)	Intensity (dB)	F0	F1	F2	F3	F4
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

3. Based on the truth table you filled out in the previous question, find a minimum SOP or minimum POS expression for...

   (a) F_4.

   (b) F_3.

(c) F_2.

(d) F_1.

(e) F_0.

4. What should pins 2, 3, and 4 on the 7485 chip be connected to in Circuit 3?

Lab 7: Combinational Logic

This lab builds off of the knowledge built in all of the previous labs to explore more complicated and interesting circuits. Each of the circuits in this lab has a practical application, and makes use of either a specialized logic device or an interesting input or output device.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-7

7.1 7-Segment Displays

A 7-segment display is used to display base 10 numbers from 0–9. They contain seven LEDs (sometimes more, for decimal points) oriented in a way such that turning on and off segments in particular order can generate any of the decimal numbers. A diagram of a 7-segment display is shown in figure 7.1.

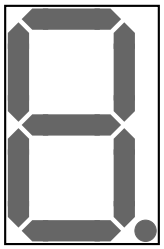


Figure 7.1: A diagram of a 7-segment display.

Because a 7-segment display contains LEDs to illuminate the different numbers, every individual LED must be connected with a current-limiting resistor in order to protect from high current. Failure to do so many result in burning out the display. This means that if all seven segments are used, there need to be seven current-limiting resistors included in the circuit. It is not sufficient to use just one. Information about this is included in the current-limiting resistor video.

A 7-segment display has no notch. However, the pin numbering still works in the same manner (starting in the upper left, and moving counter-clockwise to the upper right). To orient the display, ensure that the decimal point is at the bottom of the display.

7.1.1 Common Cathode

In a common cathode display, the LEDs are connected such that they share a common cathode, as shown in figure 7.2.

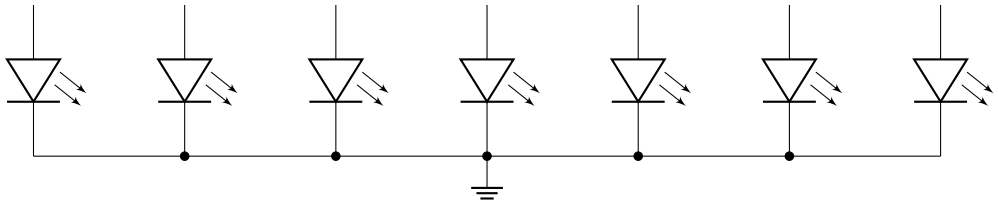


Figure 7.2: Circuit diagram of a common cathode 7-segment display.

In this configuration of LED segments, a HIGH value connected to an anode will cause current to flow, making the segment light up. A LOW value connected to an anode will not cause current to flow (as there will be no voltage drop between the anode and cathode), making the segment remain off. This type of display is used with active-HIGH output logic.

7.1.2 Common Anode

In a common anode display, the LEDs are connected such that they share a common anode, as shown in figure 7.3.

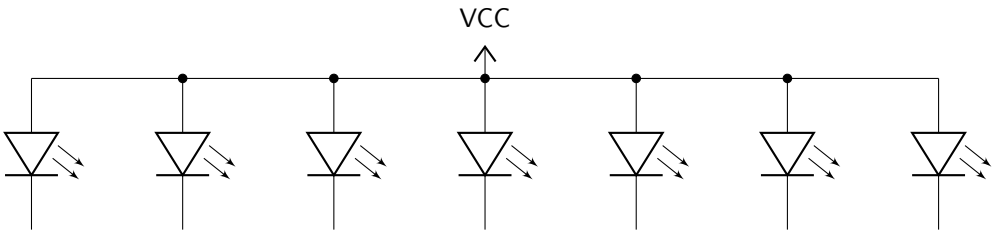


Figure 7.3: Circuit diagram of a common anode 7-segment display.

In this configuration of LED segments, a LOW value connected to a cathode will cause current to flow, making the segment light up. A HIGH value connected to a cathode will not cause current to flow (as there will be no voltage drop between the anode and cathode), making the segment remain off. This type of display is used with active-LOW output logic.

7.1.3 7-Segment Decoder

A device called a 7-segment decoder is used to turn ON and OFF each individual segment in order to illuminate each of the decimal numbers from 0–9. The decoder takes an 8-4-2-1 BCD value and has internal logic that creates the correct output values to send to each segment cathode (for a common anode display) or to each segment anode (for a common cathode display).

Depending on the display type used, the correct decoder chip must be used. The 7447 BCD to 7-segment decoder works with common anode displays. Current-limiting resistors must be used between each output of the 7447 display and the individual segment cathodes. A PCB containing these connections has already been made and is available for you to use in future labs (you will not use it in this lab, as the purpose of this lab is to get you familiar with the concept of decoding just one segment).

The 7448 BCD to 7-segment decoder contains internal resistance such that external resistors don't need to be used (depending on the properties of the 7-segment display, the internal resistance may not be high enough). This chip is used with common cathode displays. This chip usually costs about double the price of the 7447, due to the extra circuitry inside!

Circuit 1: Using **either** the common cathode or the common anode display, create a circuit to light up the segment of your choice given a **5-2-2-1** BCD input value. (If there are multiple encodings of a decimal value, the ones that show up farther down on the truth table should be considered to be don't cares.) **Indicate which display type and segment you choose.** Fill out the truth table and find a minimum expression for the segment output. Draw a circuit diagram. Wire up the circuit on your breadboard. Once you have verified its functionality, demonstrate your circuit to your instructor to receive a stamp.

A	B	C	D	Decimal Number	Segment ON or OFF	F
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Instructor Stamp: _____

7.2 Analog to Digital Conversion

Analog to digital conversion is a process by which analog voltages (which can take on any arbitrary voltage, and is not just limited to 0 V or 5 V) are converted into a binary number. This enables an analog signal to be converted into a quantity that can be processed by a digital logic circuit.

For example, an electronic temperature sensor can be used to measure the temperature of a room. It can be connected into an analog circuit to produce an output voltage that is proportional to the temperature. The voltage can be any value between 0 V and 5 V, but cannot be directly used by a digital logic circuit. After going through the analog to digital conversion process, a binary number can be created that is either proportional to or equal to the actual temperature.

Many microcontrollers contain circuitry that are capable of doing analog to digital conversion. While this is not the subject of this class, a microcontroller can be used to generate binary numbers from an analog quantity, which will be used in Circuit 2.

Circuit 2: You have a 5-LED display that indicates the volume of a stereo system. You are given a 4-bit binary input ($ABCD$) that corresponds to the sound intensity of the stereo system in bels (1 bel = 10 dB). None of the LEDs will be lit when the sound intensity is less than 30 dB; LED 0 will be lit when the sound intensity is greater than or equal to 30 dB; LEDs 0 and 1 will be lit when the sound intensity is greater than or equal to 60 dB; LEDs 0, 1 and 2 will be lit when the sound intensity is greater than or equal to 90 dB; LEDs 0, 1, 2 and 3 will be lit when the sound intensity is greater than or equal to 120 dB. At values of 150 dB and greater, all five LEDs will be lit.

Using the equations you derived in the pre-lab, draw and label a circuit diagram for each output. Then, wire up each output, using the PCB described in figure 7.4 for each output. (Note that 220 Ω current-limiting resistors are already included in the PCB.) First, use a DIP switch for inputs A, B, C, and D to verify that each output functions as expected.

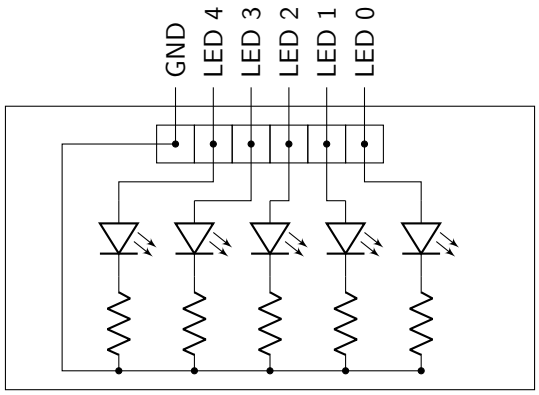


Figure 7.4: Circuit diagram of the PCB that will be used for each of the outputs in this circuit.

Now that you have verified the functionality of your circuit with a DIP switch, disconnect the DIP switch and the pull-down resistors. Instead, use the potentiometer (dial) PCB (depicted in figure 7.5) to generate the input values. The potentiometer can be dialed just like the input to a stereo system. As you twist the potentiometer from one extreme to the next, you should see the LEDs light up in sequence.

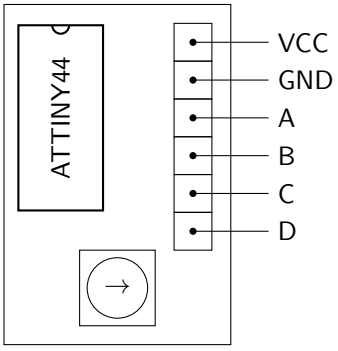


Figure 7.5: Circuit diagram of the PCB that will be used to generate the input signals in this circuit.

Verify the functionality of your circuit, and then demonstrate it to your instructor to receive a stamp.

Instructor Stamp: _____

7.3 RGB LEDs

An RGB LED contains three separate LEDs – one red, one green, and one blue – all contained within the same epoxy cap. Turning on combinations of each LED can create the primary colors, secondary colors, and white. In this manner, an RGB LED can be used to create several different colors using a single device.

Just as 7-segment displays come in both common cathode and common anode formats, RGB LEDs also come in two formats. However, the only RGB LEDs that will be available to use in this lab are common cathode RGB LEDs. This means that all of the LED cathodes are connected together at one lead, which must be connected directly to ground. Each LED anode has its own leg, and each has a different length. The pinout diagram explaining which leg corresponds to which connection is given in Appendix A of this lab manual.

Current-limiting resistors are required between any signal and the anode of each LED. This means that three current-limiting resistors must be used in any RGB LED. This is shown for a common-cathode RGB LED in figure 7.6.

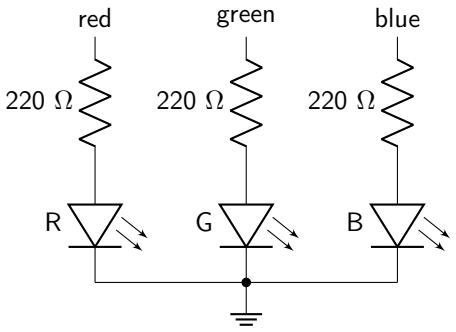


Figure 7.6: Circuit diagram of a common-cathode RGB LED, showing proper connections to each current-limiting resistor.

7.4 Comparison Logic

Comparison logic can be used to determine which of two numbers has a greater magnitude. If a comparator is considering two numbers, A and B, the comparator will generate three output signals, a signal that will be asserted if $A > B$, a signal that will be asserted if $A = B$, and a signal that will be asserted of $A < B$.

This logic is rather simple to build when comparing two one-bit numbers. However, it quickly becomes difficult to build using AND, OR, and NOT gates once A and B each represent two-bit numbers. Thankfully, comparator chips are available to use. In this lab, the 7485 four-bit magnitude comparator will be used. This chip has many important features to be understood before utilizing it in lab.

There are two types of inputs used in the 7485 logic chip.

The parallel inputs correspond to A and B, both of which are four-bit unsigned binary numbers. These are the numbers that will come from an input device such as a keypad or DIP switch. Each input is expressed as $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$, with the variable labeled 3 being the MSB and the variable labeled 0 being the LSB.

Expander inputs are used to cascade together two or more 7485 chips to compare binary numbers that are larger than four-bits. (Just as two four-bit adders can be connected to create an eight-bit adder, so can two four-bit comparators be connected together to create an eight-bit comparator.) The 7485 datasheet indicates what should be done with the expander inputs in case the device will not be expanded.

> For proper compare operation, the Expander Inputs to the least significant position must be connected as follows: IA < B= IA > B = L, IA = B = H.

The three output signals are $O_{A>B}$, which is asserted HIGH if $A > B$, $O_{A=B}$, which is asserted HIGH if $A = B$, and $O_{A<B}$, which is asserted HIGH if $A < B$.

Circuit 3: You have an RGB LED to use to indicate whether or not player A has a higher score than player B. Each player's score will be represented by a four-bit binary number, using a DIP switch. If player A has a higher score, then the LED will be green. If player B has a higher score, then the LED will be red. If both players are tied, the LED will be blue. Use an RGB LED and the 7485 chip to build this circuit. Verify that the circuit works, then demonstrate it to your instructor to receive a stamp.

Instructor Stamp: _____

Lab 7 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Fill out the k-map to find every prime implicant, and then **use a prime implicant table** to derive a minimum SOP expression for **segment g** on a common-cathode 7-segment display. Numerals 6 and 9 do not have a hat/tail. (In other words, segment a is off for numeral 6 and segment d is off for numeral 9.) The input is 8-4-2-1 BCD. Clearly label each of the k-map loops with the correct Boolean term.

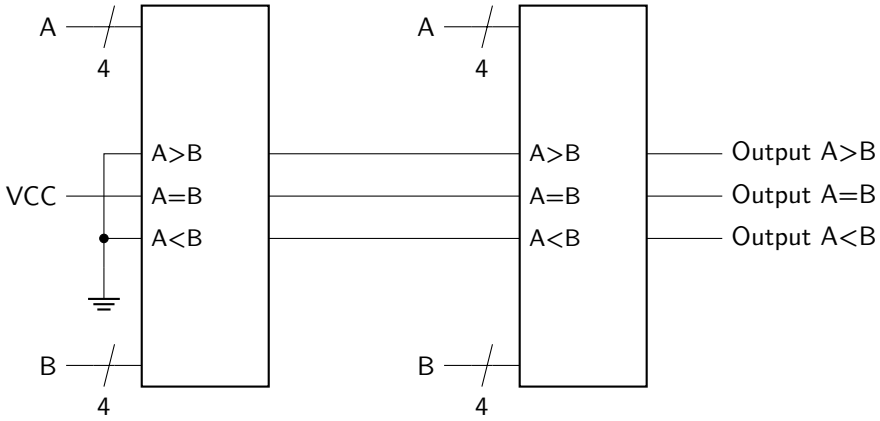
AB CD	**00**	**01**	**11**	**10**
00				
01				
11				
10				

2. Design a circuit that converts from 8-4-2-1 BCD to **6-3-1-1** BCD. If there are two 6-3-1-1 encodings that result in an 8-4-2-1 numeral, select the option with the least significance on the truth table. (For example, if both 0101 and 1000 in 6-3-1-1 are equivalent to a numeral in 8-4-2-1, you would select 0101.) All input values greater than 9 will be don't cares. Derive minimum SOP or minimum POS expressions for each of the four outputs.

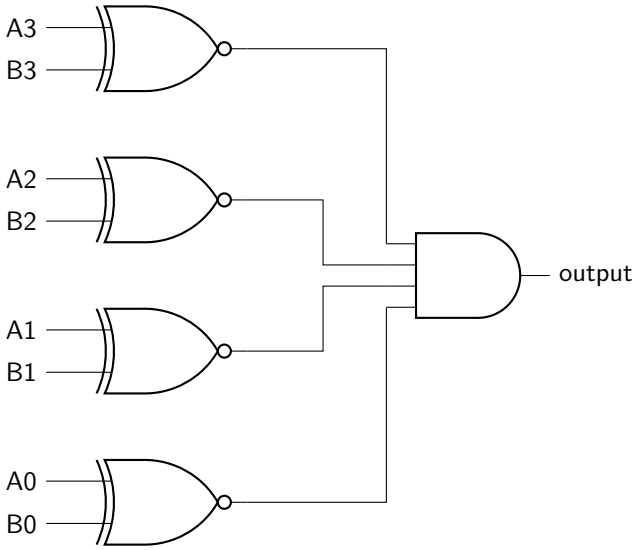
A	B	C	D	W	X	Y	Z
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

3. Is it possible to use a single logic gate to determine if two one-bit numbers are equal? If so, which logic gate would you use?

4. The circuit diagram below shows two 4-bit magnitude comparators connected together to create an 8-bit magnitude comparator. Which of the chips should be connected to the most significant data nibble? Which should be connected to the least significant data nibble? Justify your answer.



5. A 4-bit magnitude comparator is built as shown in the following circuit diagram. Is this circuit output, as wired, capable of determining if $A > B$, $A < B$, and $A = B$? Justify your answer.



Pre-Lab 8

Carefully read the entirety of Lab 8, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Calculate the number of **bytes** of memory in the following ROM chips.
   (a) 27C64

   (b) 27C010

   (c) 27C256

   (d) 27C1024

2. Are EPROM outputs HIGH or LOW by default?

3. How many input variables are needed to express decimal numbers from 0–31?

4. When placed on your breadboard, what value should be sent to the $\overline{\text{PGM}}$ pin (VCC or GND)? Why?

5. What are the binary values that should be written to the EPROM in lab Circuit 2? Q7 is the MSB and Q0 is the LSB of each word. (Note, this truth table is very long and continues on the next page.)

Address	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
00000								
00001								
00010								
00011								
00100								
00101								
00110								
00111								
01000								
01001								
01010								
01011								
01100								
01101								
01110								
01111								
10000								
10001								
10010								
10011								
10100								
10101								
10110								
10111								
11000								
11001								

11010								
11011								
11100								
11101								
11110								
11111								

6. What are the binary values and hexadecimal characters that should be written to the EPROM in lab Circuit 3? Output pins Q7–Q3 will not be used and will therefore be kept at their default HIGH value.

Address	**Q7–Q3**	**Q2** **R**	**Q1** **G**	**Q0** **B**	**Hex Value**
0000	1 1 1 1 1				
0001	1 1 1 1 1				
0010	1 1 1 1 1				
0011	1 1 1 1 1				
0100	1 1 1 1 1				
0101	1 1 1 1 1				
0110	1 1 1 1 1				
0111	1 1 1 1 1				
1000	1 1 1 1 1				
1001	1 1 1 1 1				
1010	1 1 1 1 1				
1011	1 1 1 1 1				
1100	1 1 1 1 1				
1101	1 1 1 1 1				
1110	1 1 1 1 1				
1111	1 1 1 1 1				

7. The following questions pertain to the 74194 4-bit bidirectional universal shift register. Refer to the datasheet to help answer these questions.

   (a) What value of mode control inputs S1 and S0 is required to achieve each of the following functions on the register?

Function	S1	S0
Load		
Hold		
Shift Left		
Shift Right		

   (b) In what direction is data shifted when shifting left? (In other words, does data shift from A–D or from D–A?)

   (c) In what direction is data shifted when shifting right? (In other words, does data shift from A–D or from D–A?)

   (d) What do the SR SER and SL SER pins do?

   (e) Is $\overline{\text{CLR}}$ active-HIGH or active-LOW? How do you know?

Lab 8: Memory

In this lab, we will learn about volatile and non-volatile memory elements and see how they are useful in implementing complicated and/or multiple-output combinational logic circuits. We will first learn how to read data that has already been programmed into a ROM chip, and then we will expand on this knowledge by programming ROM chips to create circuits without using any AND, OR, or NOT logic gates. Finally, an important volatile memory chip will be used to store and shift data.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-8

8.1 Read-Only Memory

Memory is used to store bits of information for use by a computer, microcontroller, or digital logic circuit. Read-only memory refers to non-volatile memory that is not intended to be re-written frequently (if at all) after it is initially programmed. Although the name may suggest otherwise, some types of read-only memory can be written to (and are not in fact "read only"), but it is generally more complicated to write to the memory unit than to read from it. For example, a high voltage that is not available inside of a digital logic circuit may be required to re-write the data on the chip. This helps prevent against accidentally writing over important memory during the routine use of the memory chip.

The two types of ROM that will be used in this lab are UV EPROM and PROM.

8.1.1 UV Erasable Programmable Read-Only Memory (EPROM)

UV EPROM is a type of read-only memory that can be written to with a programmer, and then subsequently erased by exposing the chips to high-intensity UV light for several minutes. UV EPROM chips can be recognized by a quartz window that exposes the underlying silicon chip, allowing it to be erased by UV light if needed. Data is written to the chip using a programmer, which injects electrons into certain areas of the silicon, creating logic LOW values at those locations. When exposed to UV light, the electrons are ejected out of the silicon, causing each memory location to take on its default unprogrammed value of logic HIGH.

UV EPROM chips are a great option for testing out a prototype circuit, as the design can be tested out without committing to the final format. The chip can be erased and reprogrammed if there are issues with the design, or if modifications need to be made. It is also a great chip to use in a lab class, as students can work with the chips several times, erasing after each use, over many semesters without having to purchase new chips.

8.1.2 Programmable Read-Only Memory (PROM)

PROM is read-only memory that is one-time programmable (OTP). It functions almost exactly the same as UV EPROM, except that there is no quartz window to erase data. Once the chip has been programmed, it will store that data for the lifetime of the chip. It cannot be erased or reprogrammed. This makes PROM a great candidate for use in circuits once a design has been finished. Prototype with UV EPROM, and then use PROM for the design once it has been tested and finalized.

8.1.3 Using ROM

It is possible to build any combinational or sequential circuit using a combination of logic gates. However, sufficiently complicated circuits may require a large footprint of devices, consume a lot of power, and take a long time to build and debug. For this reason, it can be useful to use ROM instead. For example, a circuit

used to realize many separate output functions that each require several inputs would be a prime candidate for using ROM. The truth table of each output is stored in the ROM as a sort of look-up table. When the input is applied to the address inputs of the ROM, that line of the "truth table" is then accessed in the memory and the values are sent to the output lines.

8.1.4 EPROM Chips

There are a few EPROM chips available in this class. This lab in particular will only use the EPROM chips with 8-word memory (capable of executing eight output functions simultaneously). Regardless, each of the chips share some common features and pin notation. Explanations of each type of pin follows.

Address Inputs These pins are labeled starting with the letter A, followed by a number. A0 is the least significant address pin. These pins are used as the inputs to the ROM chip. They are called address pins because they tell the ROM which row of data to select and send to the output.

Data Outputs These pins are labeled starting with the letter Q, followed by a number. Q0 is the least significant data output pin. These pins will be asserted with whatever values are programmed into the chip at the address location present on the input. These pins may not work as expected if the chip or output pins are not enabled.

Output Enable This pin is labeled $\overline{\text{OE}}$. This is an active-LOW output enable pin. When asserted, the output pins will function as expected (assuming that the chip itself is also enabled). When not asserted, the outputs will all be in high-Z mode (tri-stated). This tri-state mode will be explored in the next lab. For now it is sufficient to consider a tri-state mode to be a floating output value. (This means you will not get either a HIGH or LOW signal if probing the pins in the tri-state mode.) Because of the tri-state property, this pin is used when connecting two or more EPROM chips together on one data bus.

Chip Enable This pin is labeled $\overline{\text{CE}}$. This is an active-LOW chip enable pin. When asserted, the output pins fill function as expected (assuming that the outputs are also enabled). When not asserted, the outputs will all be in high-Z mode, putting the chip into a standby mdoe.

Program Enable This pin is labeled $\overline{\text{PGM}}$. This is an active-LOW program enable pin. When asserted, the chip can be programmed (which would not be desirable when the chip is placed into the breadboard, ready to be used).

Program Voltage This pin is labeled VPP. This pin is used to connect to a high voltage (usually between 12–15 V) to program the chip and enter new data, assuming that the program enable is asserted.

No Connection Sometimes an EPROM chip contains more pins than functions. Any pin that has no connection is labeled NC.

Circuit 1: The 27C256 chip has already been programmed by your instructor before lab to realize four functions of four variables: $Q_0(A, B, C, D)$, $Q_1(A, B, C, D)$, $Q_2(A, B, C, D)$, and $Q_3(A, B, C, D)$. You will therefore use address bits A3–A0 as inputs $ABCD$, respectively. All other address inputs should be grounded. Fill out the following truth table based on the results of each set of input combinations.

A	B	C	D	Q0	Q1	Q2	Q3
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

Instructor Stamp: _____

8.1.5 Programming EPROM

A programmer will now be used to write data to the EPROM chips. The exact chip that will be used needs to be entered into the software, causing the software to load configuration details such as the chip pinouts, the word size, the number of address locations, program voltages, and other important information.

For each address location, you will enter the hexadecimal value that you wish to have programmed into that location in memory. For chips that have 8-bit words, this corresponds to one byte, or two hexadecimal characters. Programming the chip takes place in a few steps.

- Connect the programmer to the computer using the USB cable, and open the programmer software.
- Click on the button under `Select IC` and search for the corresponding EPROM chip and packaging. When you have found it, click on the `Select` button.
- Click on the `Information` button to see how the chip should be inserted into the ZIF socket, and do so. Lock the socket by pressing down on the lever.
- Enter the hexadecimal characters corresponding to the data that you wish to have programmed. (Entries start in the upper left and proceed left to right, then top to bottom, as if you are reading a book.)
- Under `Options`, deselect the `Check ID` button.
- Program the chip by clicking on the large, red `P` button. (The P looks like it is inside of a surface mount IC chip.)

Circuit 2: Use the programming software to write the necessary hex values to the 27C010 chip to display numerals from 00–31 on two 7-segment displays via the 7-segment display PCB. The pinout diagram for this PCB is provided in Appendix A of this lab manual. Wire up the circuit and verify that it works. Then, demonstrate it to your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 3: Use the programming software to write the necessary hex values to the 27C010 chip to light up the segments on an RGB LED as given below. Refer to Appendix A and Lab 7 for information on how to properly connect the RGB LED.

Input Values	Illuminated LED Color(s)
0–2	red
3–4	red and green
5–7	green
8–9	green and blue
10–11	blue
12–13	blue and red
14–15	red and green and blue

Wire up the circuit and verify that it functions properly. Then, demonstrate it to your instructor to receive a stamp.

Instructor Stamp: _____

8.2 Parallel-In, Parallel-Out (PIPO) Shift Register

While we will not formally learn about registers until later in the semester, a parallel-in parallel-out (PIPO) shift register is an incredibly useful component in digital circuits that require a memory component that can store data on the breadboard (rather than being programmed externally to the rest of the circuitry). Registers are a type of volatile memory. In volatile memory elements, the contents that are stored on the chip are lost when power is removed from the circuit. The memory capacity of a register has to do with how many bits of storage are available on the device. This corresponds to the number of circuit elements known as flip-flops, which will be explored in future labs.

8.2.1 4-Bit Bidirectional Universal Shift Register Chip

The 74194 digital logic chip is a PIPO register with a memory capacity of four bits. It is capable of four functions, listed below.

- Hold (continue holding on to previously stored data)
- Parallel load (store new data)
- Shift right
- Shift left

The function that is selected is controlled by the mode control inputs S1 and S0. The action that is selected will occur when a rising edge occurs on the CLK pin. If an active-HIGH pushbutton is connected to the CLK pin, that means that data will shift, load, or hold when the pushbutton is pressed down.

Circuit 4: Wire up the 74194 chip using a DIP switch to control the data inputs (A–D), mode control inputs (S1 and S0), and serial shift inputs (SR SER and SL SER). Connect an active-LOW pushbutton to the $\overline{\text{CLR}}$ pin. Connect an active-HIGH debounced pushbutton to the CLK input.

Connect each of the data output pins (QA–QD) to an LED. By selecting the mode control inputs, try each of the different functions of the shift register. Then, demonstrate the completed circuit to your instructor to receive a stamp.

Instructor Stamp: _____

Lab 8 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Derive a minimum expression for Q_0 in lab Circuit 1.

2. Derive a minimum SOP or minimum POS expression for Q_1 in lab Circuit 1.

3. Derive a minimum SOP or minimum POS expression for Q_2 in lab Circuit 1.

4. Derive a minimum SOP or minimum POS expression for Q_3 in lab Circuit 1.

5. In the context of lab Circuit 1, explain how ROM simplifies the design and creation of combinational logic circuits.

6. In lab Circuit 2, you designed a 5-bit binary to BCD decoder.

   (a) Determine the minterms of $Q_4(A, B, C, D, E)$.

   (b) Use the Quine-McCluskey method to derive a minimum SOP expression for $Q_4(A, B, C, D, E)$.

Column 1	Column 2	Column 3	Column 4	Column 5

Prime Implicants	

(c) Determine the minterms of $Q_3(A, B, C, D, E)$.

(d) Use the Quine-McCluskey method to derive a minimum SOP expression for $Q_3(A, B, C, D, E)$.

Column 1	Column 2	Column 3	Column 4	Column 5

Prime Implicants	

7. Derive a minimum SOP or minimum POS expression for the red anode of the RGB LED from lab Circuit 3.

8. Derive a minimum SOP or minimum POS expression for the green anode of the RGB LED from lab Circuit 3.

9. Derive a minimum SOP or minimum POS expression for the blue anode of the RGB LED from lab Circuit 3.

Pre-Lab 9

Carefully read the entirety of Lab 9, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Derive the minimum SOP expression for the function given in Circuit 5.

2. Derive the minimum POS expression for the function given in Circuit 6.

3. Is Circuit 7 a minimum circuit? If not, derive either a minimum SOP or a minimum POS expression that is equivalent to the circuit.

Lab 9: NAND and NOR Circuits and Introduction to Tri-State Outputs

In this lab, two new types of gates will be used: NAND and NOR. Each of these logic gates is a universal gate, as it can be used to create any other type of logic gate. At the conclusion of this lab, you will utilize a tri-state buffer to connect outputs together in a manner that would not be allowed with "regular" digital logic devices.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-9

9.1 NAND and NOR Gates

Two logic functions that can be used to implement a digital logic circuit are NAND (not and) and NOR (not or). Each of these logic functions is universal. That is, any digital logic circuit can be made with only NAND gates. Also, any digital logic circuit can be made with only NOR gates.

Circuit 1: Determine how to create a NOT gate using only NAND gates. Draw a circuit diagram. Then, wire it up on your breadboard. Demonstrate the functionality to your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 2: Determine how to create a NOT gate using only NOR gates. Draw a circuit diagram. Then, wire it up on your breadboard. Demonstrate the functionality to your instructor to receive a stamp.

Instructor Stamp: _____

9.1.1 NAND and NOR Fan-In

The NAND and NOR gates available to use in this lab have a fan-in of two. It is therefore important to determine how to create gates with a fan-in of three using the chips that are available in lab.

Circuit 3: Determine how to create a 3-input NAND gate using only 2-input NAND gates. Draw a circuit diagram. Then, wire it up on your breadboard. Demonstrate the functionality to your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 4: Determine how to create a 3-input NOR gate using only 2-input NOR gates. Draw a circuit diagram. Then, wire it up on your breadboard. Demonstrate the functionality to your instructor to receive a stamp.

Instructor Stamp: _____

9.1.2 NAND-NAND and NOR-NOR Circuits

Minimum SOP and POS expressions can be modified to derive NAND-NAND and NOR-NOR circuits. This can be accomplished either using the double prime method or with the bubble method. Both of these methods are described in the textbook.

Circuit 5: Determine the equivalent NAND-only circuit for the equation you derived for the given function. Wire it up using **only** NAND gates. Demonstrate its functionality to your instructor to receive a stamp. Draw a circuit diagram of your NAND-NAND circuit.

$$F(A, B, C, D) = \Sigma m(2, 3, 8, 10, 13, 15) + \Sigma d(9, 11)$$

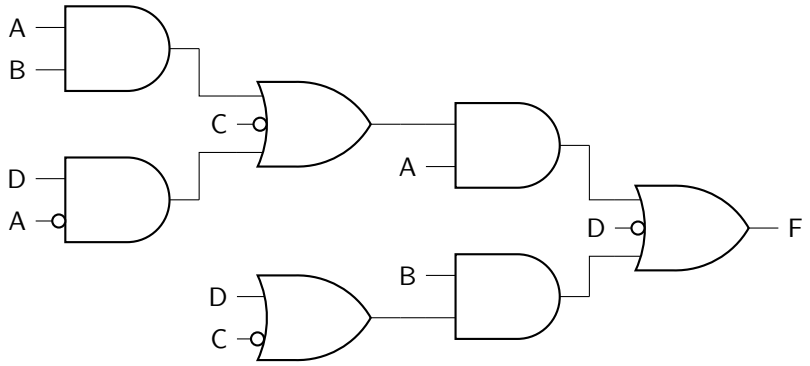
Instructor Stamp: _____

Circuit 6: Determine the equivalent NOR-only circuit for the equation you derived for the given function. Wire it up using **only** NOR gates. Demonstrate its functionality to your instructor to receive a stamp. Draw a circuit diagram of your NOR-NOR circuit.

$$F(A, B, C, D) = \Sigma m(4, 5, 6, 7, 8, 12, 13)$$

Instructor Stamp: _____

Circuit 7: Create either a NAND or NOR equivalent (indicate which you choose) of the circuit provided below. Using only the gate that you have chosen (i.e., no inverters, AND, OR, or XOR gates), wire it up on your breadboard. Demonstrate its functionality to your instructor to receive a stamp. Draw a circuit diagram of your final circuit.



Instructor Stamp: _____

9.2 Tri-State Devices

The term tri-state refers to a third possible output value that can be present on a digital logic device. Previously we have considered signals that are HIGH (5 V) or LOW (0 V). This third value is an electrical disconnection, which means there is no voltage level present at all on the output. The output is said to be floating (or tri-stated). Sometimes this output value is known as high-impedance or high-Z, due to the fact that impedance is another word for resistance, and resistance is effectively infinite when there is an electrical disconnection. The tri-state output appears on a truth table as a Z symbol.

Devices with tri-state outputs have a function that is not possible with HIGH and LOW values: tri-stated outputs can be connected directly together, but only when one of the tri-stated outputs is enabled at a time!

To determine if a digital logic chip has tri-state outputs, the datasheet must be consulted. (Unlike with active-LOW signals, there is not a universally used pinout diagram signal for tri-state outputs.) Sometimes the description on the datasheet will explain if outputs can be tri-stated. At times, a truth table must be consulted to determine if tri-state outputs can be enabled.

9.2.1 Tri-State Buffer Chip

The 74126 chip is a quad tri-state buffer chip. Each buffer has an active-HIGH enable. When asserted, the buffer will act as a typical buffer gate. When de-asserted, the output of the buffer will be held in a tri-state mode, as shown in the truth table below.

EN	A	F
0	0	Z
0	1	Z
1	0	0
1	1	1

Circuit 8: Use a DIP switch to generate two 4-bit binary numbers. Using tri-state buffers, connect both sets of binary numbers to a single 7-segment display (using the 7447 7-segment decoder chip). **Be sure that only one set of tri-state buffers is enabled at a time.** (If not, you may get a short on your breadboard. Be aware of the current draw from the power supply and be prepared to disconnect power from your breadboard if this occurs.) Use a toggle switch to select which of the two 4-bit values gets sent to the 7-segment display. A high-level schematic is shown in Figure 9.1. When you've successfully completed the circuit, demonstrate it to your instructor to receive a stamp.

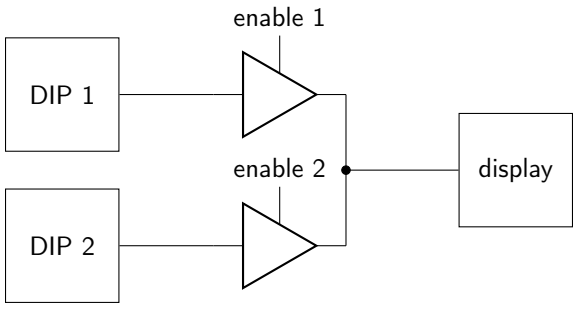


Figure 9.1: High-level schematic of the tri-state buffer circuit.

Instructor Stamp: _____

Lab 9 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Using your TinkerCAD classroom account, build an AND gate using only NAND gates. This should require a single 7400 chip.
2. Using your TinkerCAD classroom account, build an OR gate using only NAND gates. This should require a single 7400 chip.
3. Using your TinkerCAD classroom account, build an AND gate using only NOR gates. This should require a single 7402 chip.
4. Using your TinkerCAD classroom account, build an OR gate using only NOR gates. This should require a single 7402 chip.
5. Draw a circuit diagram that shows how you could build a NAND gate with fan-in of four using only NAND gates that have a fan-in of two. Do the best you can to equalize propagation delay for each input signal.
6. To build an extremely streamlined processor, a circuit designer creates hardware to perform AND and NOT operations, but nothing to do OR operations. Can this processor carry out all possible logical operations? Why or why not?
7. Draw a circuit diagram for a NOR-only circuit that carries out a 2-input exclusive OR (XOR) operation.
8. Draw a circuit diagram for a NAND-only circuit that carries out a 2-input equivalence (XNOR) operation.

9. Convert the following expression to a NAND-only circuit using the **double prime method**. (Do not alter the form of the expression, convert it as-is. In addition, do not change the fan-in of any of the logic gates.)

$$F(A, B, C, D) = B + C(D + A')$$

10. Convert the above expression (from the previous question) to a NAND-only circuit using the **bubble method**. Assume you have access to NAND gates with whatever fan-in is required. (Do not alter the form of the expression, convert it as-is. In addition, do not change the fan-in of any of the logic gates.)

11. Convert the following expression to a NOR-only circuit using the **double prime method**. (Do not alter the form of the expression, convert it as-is. In addition, do not change the fan-in of any of the logic gates.)

$$F(A, B, C, D) = A'(C' + DB(A + C))$$

12. Convert the above expression (from the previous question) to a NOR-only circuit using the **bubble method**. Assume you have access to NOR gates with whatever fan-in is required. (Do not alter the form of the expression, convert it as-is. In addition, do not change the fan-in of any of the logic gates.)

Pre-Lab 10

Carefully read the entirety of Lab 10, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. The following questions refer to Circuit 5.

   (a) Fill out the k-map for the circuit.

AB CD	00	01	11	10
00				
01				
11				
10				

   (b) Which control bits will lead to a minimum number of external logic gates?

   (c) What is the MUX equation corresponding to those control bits?

2. The following questions refer to Circuit 6.

   (a) Fill out the k-map for the circuit.

AB CD	00	01	11	10
00				
01				
11				
10				

   (b) Which control bits will lead to a minimum number of external logic gates?

   (c) What is the MUX equation corresponding to those control bits?

3. In this lab, you will be asked to design a 3 to 1 MUX using only AND, OR and NOT gates.

   (a) How many control bits will you require for this circuit?

   (b) With the control bits acting as inputs, write a truth table for the output of the 3 to 1 MUX.

Control Bits	F

   (c) Write out the circuit equation.

Lab 10: Multiplexers

In this lab, multiplexers and their various applications in digital circuit design will be explored.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-10

10.1 Multiplexers

A multiplexer (MUX) is a logic device that selects one out of two or more data inputs and sends that input to the output. The input that is selected is chosen based on the value of one or more control inputs. n control inputs are required to select among 2^n data inputs. A circuit diagram of a 2 to 1 MUX is shown in figure 10.1. The two data inputs are I_1 and I_0, the control input is A, and the output is Z.

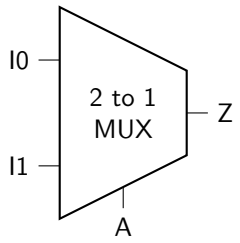


Figure 10.1: Circuit diagram of a 2 to 1 MUX.

Multiplexers are used in applications where many data streams have to share a single wire (for example, many cable signals are sent through a single fiber optic cable, and are then demultiplexed when they arrive at the subscriber's home rather than hooking up a separate fiber optic line for every subscriber). Multiplexers can also be used to implement Boolean functions. A MUX with four data inputs can be used to implement a Boolean function of four variables.

Before using a logic chip built to perform the function of a multiplexer, you will first discover different ways to implement them using different types of logic gates.

Circuit 1: Determine how to build a 2 to 1 MUX using only AND, OR and NOT gates. Draw the circuit diagram. Then, wire it up on your breadboard. Demonstrate its functionality to your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 2: Determine how to build a 3 to 1 MUX using only AND, OR and NOT gates. Draw the circuit diagram. Then, wire it up on your breadboard. Demonstrate its functionality to your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 3: Determine how to build a 2 to 1 MUX using tri-state buffers. Draw the corresponding circuit diagram. Then, wire it up on your breadboard. Demonstrate its functionality to your instructor to receive a stamp.

Instructor Stamp: _____

10.2 Dual 4 to 1 Multiplexer Chip

Now that you have learned how to make multiplexers with AND/OR/NOT gates, as well as tri-state buffers, you will be able to use the 74153 multiplexer chip to implement more complicated circuits.

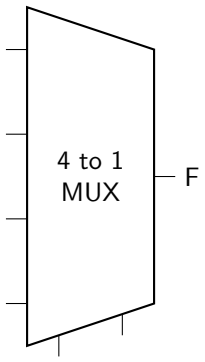
Circuit 4: Determine how to build a **6 to 1 MUX** using only 4 to 1 multiplexers. Draw the corresponding circuit diagram. Then, wire it up on your breadboard. Demonstrate its functionality to your instructor to receive a stamp.

Instructor Stamp: _____

Circuit 5: Use a 4 to 1 MUX with a minimum number of gates on the input to realize the following Boolean function.

$$F = \Sigma m(0, 1, 4, 7, 8, 13, 15) + \Sigma d(3, 5, 6)$$

Using the equation that you derived in the pre-lab, fill out the corresponding circuit diagram. Then, wire it up on your breadboard. Verify that the circuit works. Then, demonstrate its functionality to your instructor to receive a stamp.

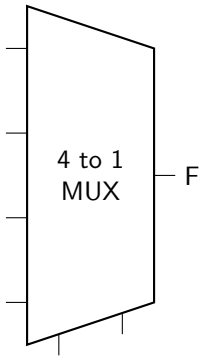


Instructor Stamp: _____

Circuit 6: Use a 4 to 1 MUX with a minimum number of gates on the input to realize the following Boolean function.

$$F = \Sigma m(0, 1, 3, 5, 7, 10, 14) + \Sigma d(2, 9, 12, 13)$$

Using the equation that you derived in the pre-lab, fill out the corresponding circuit diagram. Then, wire it up on your breadboard. Verify that the circuit works. Then, demonstrate its functionality to your instructor to receive a stamp.



Instructor Stamp: _____

Lab 10 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

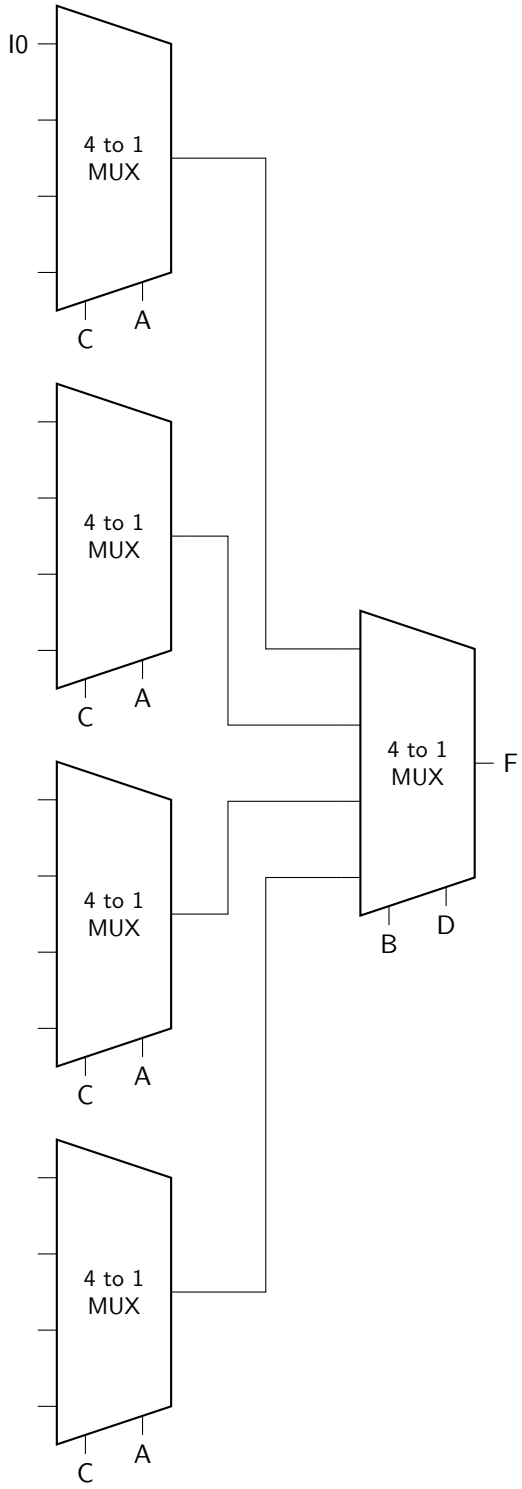
1. Assume you don't have any logic gates that have enable bits. Design a three-input AND gate (using only AND, OR and NOT gates) that has an active low enable ($\overline{E}$). Write the equation and draw the circuit diagram.

2. Again, assuming you don't have any logic gates with enable bits, design a two-input OR gate (using only AND, OR and NOT gates) that has an active high enable (E). Write the equation and draw the circuit diagram.

3. How would you build a 4 to 1 MUX using only 2 to 1 MUX chips? Draw the circuit diagram below.

4. How would you build a 10 to 1 MUX using only 4 to 1 MUX chips? Draw the circuit diagram below.

5. A 16 to 1 MUX was wired up using 4 to 1 MUX chips as follows. Label all of the MUX inputs with the corresponding binary value. (For example, if the control bits are `0000`, the input that will be passed through to the output will be labeled as $\texttt{I}_0$. This input has been labeled for you on the diagram below.) The significance of the control inputs is ABCD (A is the most significant control input).



6. What is the minimum SOP expression for the function given in Circuit 5 of the in-lab activity? (Hint: You will need to derive the minimum SOP expression, which will not be the same as the MUX equation!) Draw the circuit diagram. How many logic gates were saved by implementing this with a MUX?

7. What is the minimum SOP expression for the function given in Circuit 6 of the in-lab activity? (HINT: You will need to derive the minimum SOP expression, which will not be the same as the MUX equation!) Draw the circuit diagram. How many logic gates were saved by implementing this with a MUX?

Pre-Lab 11

Carefully read the entirety of Lab 11, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

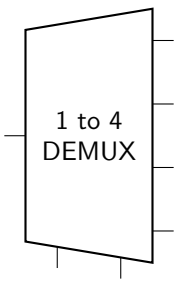
1. Carefully read Circuit 1, as well as the datasheet for the 74154 chip. Then, answer the following questions.

   (a) Fill out the truth table for an active HIGH output 1 to 4 DEMUX. X and Y are the control inputs; W is the data input, and all of the signals starting with a Z correspond to the outputs.

X	Y	W	Z0	Z1	Z2	Z3
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

   (b) Which of the input signals is the MSB of the decoder chip? D or A?

   (c) Annotate the DEMUX circuit diagram below, indicating which pin (use the symbol or name, not the pin number) you will use from the decoder chip to emulate each component of the DEMUX. (Note: there is more than one correct answer to this question.)

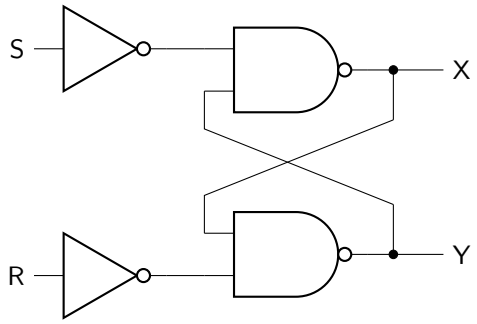


2. The following questions relate to Circuit 3.

   (a) Fill out the truth table for an active-HIGH output SR latch.

S	R	Q	Q+
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

   (b) To create the functionality you described in the previous question, based on the circuit diagram below, which output (X or Y) corresponds to Q? Which output corresponds to $\overline{Q}$?



3. If there is no notch on the 555 timer, how do you know which end is up?

4. When creating a clock, you will be asked to create one with a frequency f of 1 Hz. Use equation 11.1 to calculate the resistor and capacitor values required to create this frequency. Refer to the inventory of parts to select single component values for R_1, R_2, and C to achieve this within 10% of the desired frequency. Be sure that resistor values are at least 10 kΩ.

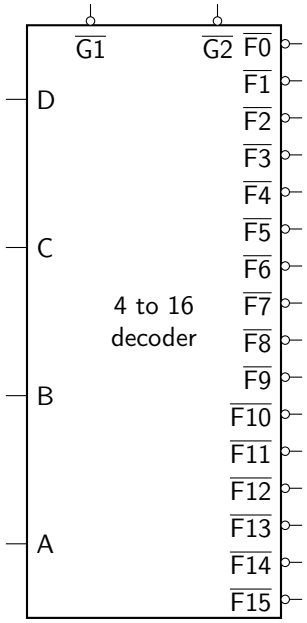
Lab 11: Decoders, Latches, and Flip-Flops

In this lab, we will learn about decoders and how they can be used to create various types of digital circuits and in implementing Boolean expressions. Then, we will learn about and design set-reset (SR) latches. We will introduce the LM555 timer, which is used to create an oscillating clock signal. Finally, we will create an edge-detecting circuit that will allow us to build flip-flop (synchronous) circuits.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-11

11.1 Demultiplexers

Circuit 1: Using the 4 to 16 decoder, create a 1 to 4 DEMUX. Annotate the decoder diagram below corresponding to your circuit diagram, clearly indicating which are your control bits, which is your DEMUX input, and which are your DEMUX outputs. Show power and ground connections where necessary. Wire it up on your breadboard and verify that the circuit works. Then, demonstrate its functionality to your instructor to receive a stamp.



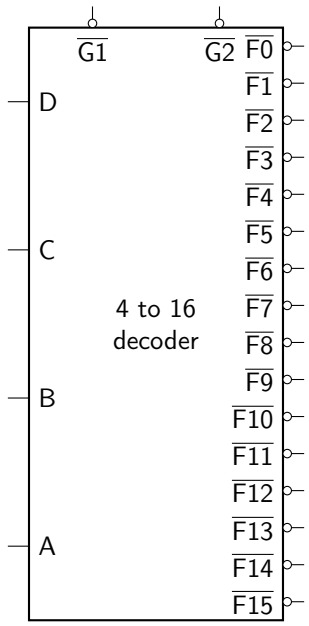
Instructor Stamp: _____

11.2 Decoders

A decoder takes n inputs and accordingly sets the values of 2^n outputs. This is useful when there is a need to select among one of many possible signals or values.

Circuit 2: Using the 4 to 16 decoder and a minimum number of external gates, realize the following Boolean function. Annotate the decoder diagram below corresponding to your circuit diagram, then wire it up on your breadboard and verify that the circuit works. Then, demonstrate its functionality to your instructor to receive a stamp.

$$F(A, B, C, D) = \Sigma m(0, 1, 4, 5, 7, 9, 14) + \Sigma d(2, 6, 8)$$



Instructor Stamp: _____

11.3 Latches

Latches are feedback circuits that have stable outputs. They are asynchronous, which means that output signals update whenever there is a change in the input values. A set-reset (SR) latch has two inputs. A circuit symbol for an SR latch is shown in figure 11.1.

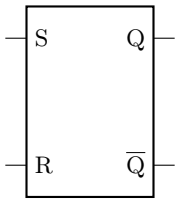


Figure 11.1: Circuit symbol of an SR latch.

Circuit 3: Using only NAND gates, create an SR latch. Display the value of Q on a 7-segment display. Draw the circuit diagram below. Then, wire it up on your breadboard and verify its functionality. Then, demonstrate the circuit to your instructor to receive a stamp. **When finished: keep this circuit on your breadboard. You will add to it in the next circuit.**

Instructor Stamp: _____

11.3.1 Gated Latches

Gated latches are latches with an enable bit (EN). When the enable bit is not asserted, the output Q does not change regardless of the values of S and R. However, when the enable bit is asserted, the latch acts in its normal functionality.

Circuit 4: Building off of your previous circuit, and using only NAND gates, create a gated SR latch. Display the value of Q on a 7-segment display. Draw the circuit diagram. Then, wire it up on your breadboard and demonstrate its functionality to your instructor to receive a stamp. **When finished: keep this circuit on your breadboard. You will add to it in Circuit 6.**

Instructor Stamp: _____

11.4 Flip-Flops

Flip-flops are feedback circuits that have bi-stable outputs (i.e. the output can take on one of two stable states: 0 or 1). They are synchronous, which means that they update in synch with an external oscillator (clock). An edge-triggered flip-flop changes its state at either the rising or falling edge of the clock signal. Therefore, the values of the inputs does not matter at any moment in time other than when the clock signal is changing.

11.4.1 555 Timer

A clock signal can be created using a 555 timer integrated circuit chip. The 555 timer has two modalities: astable and monostable modes. Astable mode will be used to create a clock signal. Appendix A in the lab manual provides the pinout diagram for the 555 timer, and Appendix B shows how to connect the 555 timer in astable mode.

Not all 555 timer chips have a notch to indicate which way points up. Whenever there is a notch on a chip, that notch indicates the top of the chip, and the pin to the left of that notch is always pin 1. In the absence of a notch, a dot will be etched into the plastic or ceramic package. That dot will indicate the top of the chip, and the pin to the left of that dot is pin 1. (Note: sometimes chips will have both a notch and a dot, sometimes at opposite ends! The notch always wins.)

The frequency of the clock signal produced by the 555 timer is given by equation 11.1, where R_1 and R_2 are resistor values, and C is the value of the capacitor.

$$f = \frac{1}{\ln(2)(C)(R_1 + 2R_2)} \tag{11.1}$$

11.4.2 Capacitors

Capacitors are analog circuit components that can store energy in the form of an electric field. In this lab, the larger capacitors that may be used in the 555 timer to generate slow frequencies may be aluminum electrolytic capacitors. All electrolytic capacitors (including aluminum electrolytic capacitors) have a polarity to them. This means that it is important to connect the anode to the high potential side of the circuit, and the cathode must be connected to the low potential side of the circuit. The cathode leg of the capacitor is indicated with a negative symbol. Be careful when hooking this up! Failure to respect the polarity of the device may lead to a "popped cap." An aluminum electrolytic capacitor can be recognized due to its cylindrical shape, with two radial (coming out of the same end) leads. This is depicted in Figure 11.2 left.

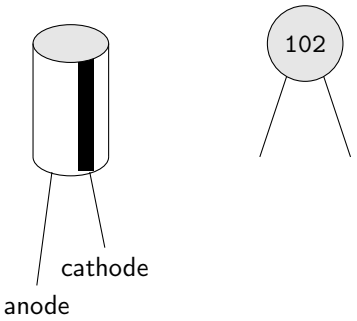


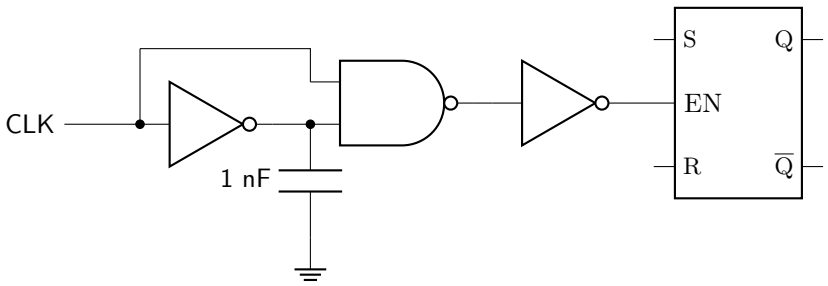
Figure 11.2: Left: Diagram of an aluminum electrolytic capacitor. The long lead is the anode (connects to higher voltage) and the short lead is the cathode (connects to lower voltage). Right: Diagram of a ceramic capacitor. The numerical code indicates the value of the capacitance.

Smaller capacitors that are shaped like a pancake (as shown in Figure 11.2 right) are known as ceramic capacitors. Not only are these physically smaller than aluminum electrolytic capacitors, they also typically have smaller capacitance values. These are frequently used to remove noise from digital circuits. Ceramic capacitors have no polarity (therefore it does not matter which way they are connected in a circuit), and are generally very safe and easy to use in circuits.

Circuit 5: Using the 555 timer, create a clock (oscillator) with a frequency of approximately 1 Hz. Use an LED to verify this timing. Wire it up on your breadboard and demonstrate its functionality to your instructor to receive a stamp. **When finished: keep this circuit on your breadboard. You will add to it in Circuit 6.**

Instructor Stamp: _____

Circuit 6: Tying all of this together, use your clock to power an edge-detection circuit that acts as an enable to your SR latch. This device is known as an SR flip-flop. Display the value of Q on a 7-segment display. You have already built the clock signal as well as the gated SR latch. You will include an edge detector circuit, shown below, to act as the clock trigger for the flip-flop. Wire it up on your breadboard and verify its functionality. Then, demonstrate the circuit to your instructor to receive a stamp.



Instructor Stamp: _____

Lab 11 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. What is one advantage of implementing a Boolean algebra expression with a decoder vs. a multiplexer?

2. What is one advantage of implementing a Boolean algebra expression with a multiplexer vs. a decoder?

3. The 7447 BCD to 7-segment decoder has active-LOW outputs for displaying a BCD character on a common-anode 7-segment display. Each of the encodings is shown in figure 11.3. Given these values, determine what output values exist for each of the following 7447 logic chips. The inputs of the 7447 chip are in order of $DCBA$.

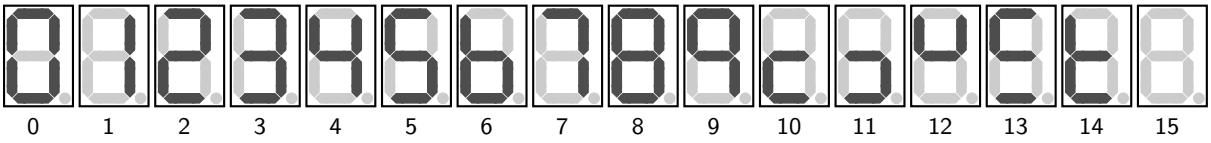
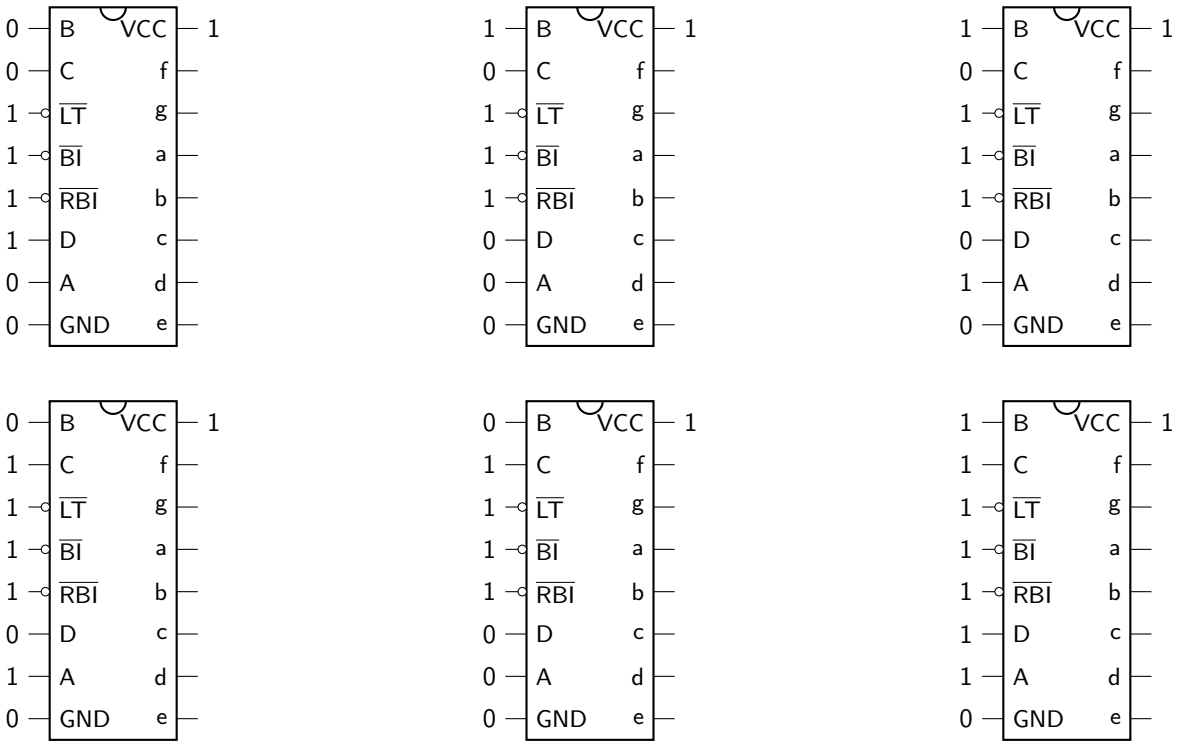


Figure 11.3: Numeral encodings for the 7447 BCD to 7-segment decoder chip.

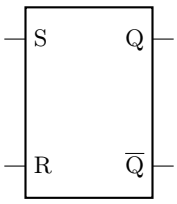


4. You wish to build an improved version of an SR latch that has the function table given in table 11.1.

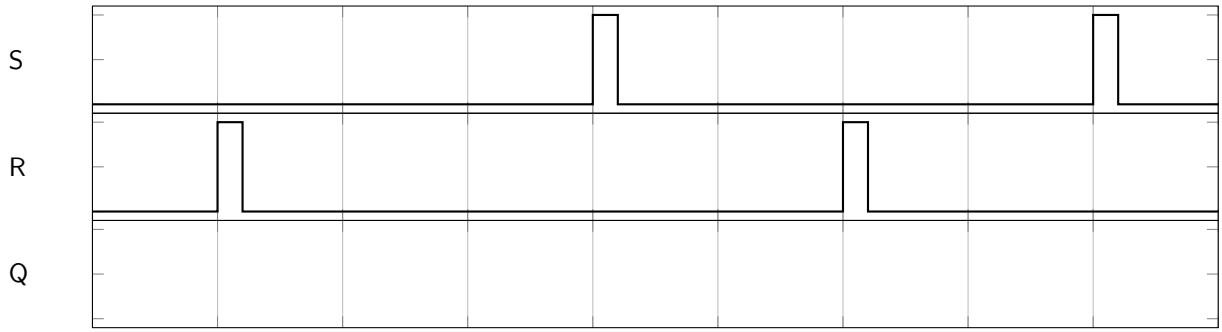
S	R	Action
0	0	hold
0	1	reset
1	0	set
1	1	hold

Table 11.1: Function table for an improved SR latch.

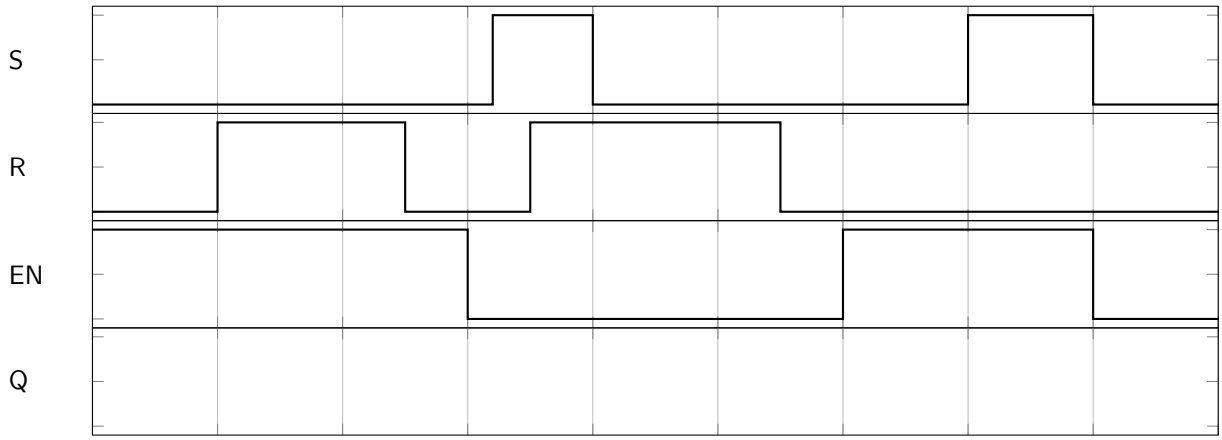
Draw a circuit diagram indicating how you would build this out of an existing SR latch (shown), as well as AND, OR, and NOT gates as needed.



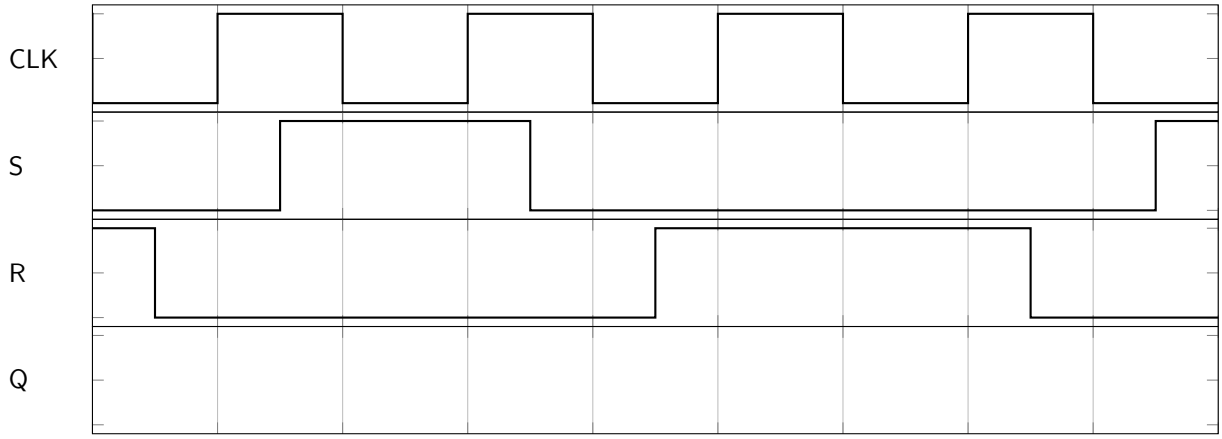
5. Given the following input values for an SR latch, draw the timing diagram for the output Q. $Q(0) = 1$. You may ignore propagation delays.



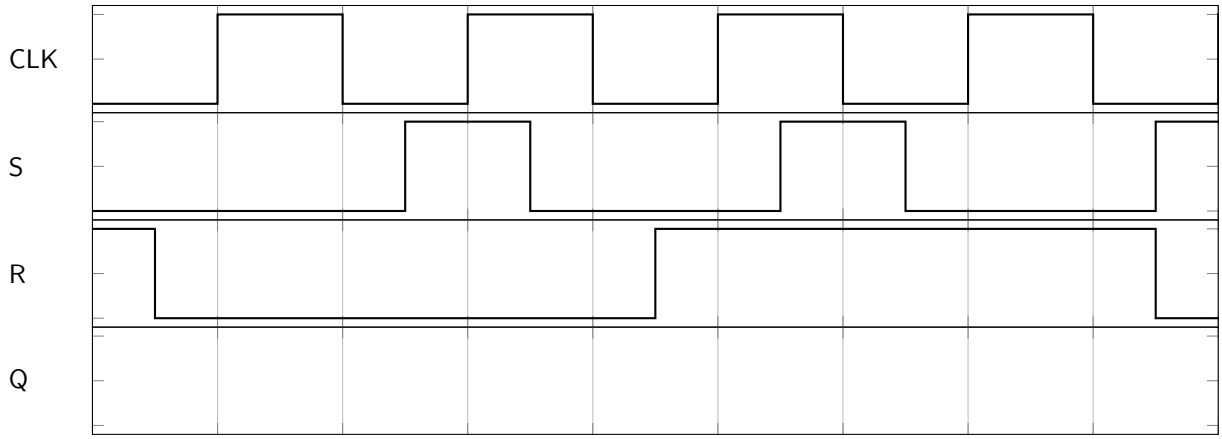
6. Given the following input values for a gated SR latch, draw the timing diagram for the output Q. $Q(0) = 1$. You may ignore propagation delays.



7. Given the following input values and clock signal for a falling-edge triggered SR flip-flop, draw the timing diagram for the output Q. $Q(0) = 1$. You may ignore propagation delays.



8. Given the following input values and clock signal for a rising-edge triggered SR flip-flop, draw the timing diagram for the output Q. $Q(0) = 0$. You may ignore propagation delays.



Pre-Lab 12

Carefully read the entirety of Lab 12, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Carefully read Circuit 1.

   (a) Fill out the following transition table to convert a JK flip-flop to a T flip-flop.

T	Q	Q+	J	K
0	0			
0	1			
1	0			
1	1			

   (b) Derive a minimum expression for J.

   (c) Derive a minimum expression for K.

2. Carefully read Circuit 2.

   (a) Fill out the following transition table to convert a D flip-flop to a T flip-flop.

T	Q	Q+	D
0	0		
0	1		
1	0		
1	1		

   (b) Derive a minimum expression for D.

3. Carefully read Circuit 3.

   (a) Derive a minimum expression for the input to flip-flop A (you will treat it like a D flip-flop) to obtain the stated functionality.

   (b) Derive a minimum expression for the input to flip-flop B (you will treat it like a D flip-flop) to obtain the stated functionality.

   (c) Derive a minimum expression for the input to flip-flop C (you will treat it like a D flip-flop) to obtain the stated functionality.

   (d) Derive a minimum expression for the input to the clock signal to obtain the stated functionality.

4. Carefully read Circuit 4.

   (a) Decide which type of flip-flop you would like to use (D or JK).

   (b) Decide if you would like your counter to count up or count down.

   (c) Fill out the following transition table for your counter. (The right-most column can be used for flip-flop values, as needed.) Treat all unused states as don't cares.

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

(d) Derive a minimum expression for either D_A or J_A and K_A.

(e) Derive a minimum expression for either D_B or J_B and K_B.

(f) Derive a minimum expression for either D_C or J_C and K_C.

(g) Derive a minimum expression for either D_D or J_D and K_D.

Lab 12: Registers and Counters

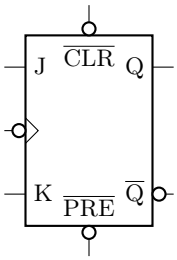
This lab will introduce two important sequential circuits: registers and synchronous counters. Registers are used to store and shift multiple bits of data. The synchronous counter built in this lab is a decade counter, which counts from 0–9 or 9–0 over and over again.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-12

12.1 Flip-Flops

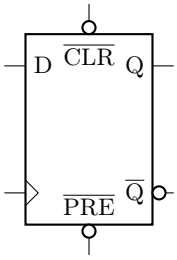
A flip-flop is a synchronous (edge-triggered) building block for sequential circuits. Each flip-flop can do two or more of the following: hold ($Q^+ = Q$), reset ($Q^+ = 0$), set ($Q^+ = 1$), toggle ($Q^+ = Q'$). The two flip-flop chips available in lab are the 7474 (dual D flip-flop) and 74112 (dual JK flip-flop). There is a third common type of flip-flop, known as a T flip-flop, that we do not have. However, it is possible to convert flip-flops from one form to another. These first two circuits will allow you to determine how to convert between flip-flop forms, using both the JK and D flip-flops as a T flip-flop.

Circuit 1: In the pre-lab, you determined how to wire up a JK flip-flop to act as a T flip-flop. Use a clock frequency of approximately 1 Hz, wire up the circuit and display the output on a 7-segment display. Annotate the diagram below. Demonstrate its functionality to your instructor to receive a stamp.



Instructor Stamp: _____

Circuit 2: In the pre-lab, you determined how to wire up a D flip-flop to act as a T flip-flop. Use a clock frequency of approximately 1 Hz, wire up the circuit and display the output on a 7-segment display. Annotate the diagram below. Demonstrate its functionality to your instructor to receive a stamp.



Instructor Stamp: _____

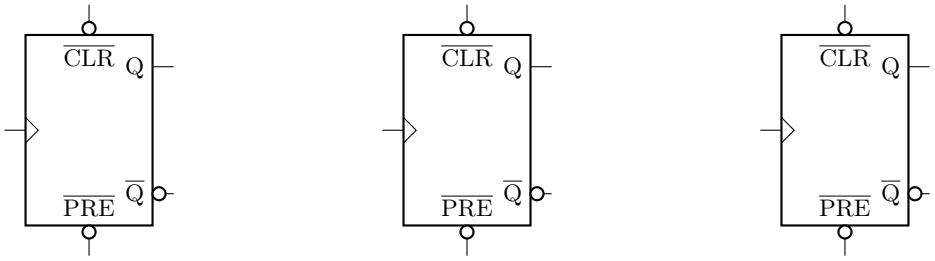
12.2 Registers

Registers are digital logic devices that store and shift binary data. They are comprised of flip-flops. The storage capacity of a register denotes how many ones and zeros can be stored at any given time. Shifting refers to the movement of bits from one flip-flop to another.

Circuit 3: Using either three D flip-flops or three JK flip-flops, create a 3-bit parallel in / parallel out shift register. Each output (Q_A, Q_B, and Q_C) will have its own individual LED. The design will include two pushbuttons as inputs. If one of the pushbuttons is pressed, the register will LOAD inputs from a DIP-switch directly to each flip-flop. If the other pushbutton is pressed, the register will SHIFT from Q_A toward Q_C. If neither pushbutton is pressed, the register will HOLD (not do anything).

- **Flip-Flop A** – The input to this flip-flop will be equal to A if the load pushbutton is pressed. The input will be 0 if the shift pushbutton is pressed. If neither pushbutton is pressed, the clock signal will be inhibited.
- **Flip-Flop B** – The input to this flip-flop will be equal to B if the load pushbutton is pressed. The input will be equal to the output of Q_A if the shift pushbutton is pressed. If neither pushbutton is pressed, the clock signal will be inhibited.
- **Flip-Flop C** – The input to this flip-flop will be equal to C if the load pushbutton is pressed. The input will be equal to the output of Q_B if the shift pushbutton is pressed. If neither pushbutton is pressed, the clock signal will be inhibited.

Draw a circuit diagram below using the flip-flops shown. Label the inputs so it is clear if you are using a D or JK flip-flop. Demonstrate its functionality to your instructor to receive a stamp.



Instructor Stamp: _____

12.3 Counters

Synchronous counters are simply devices that are synched to an external clock so that all flip-flops change simultaneously. When this happens, the output cycles through values (for example, a decade counters counts 0 – 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 –9 and then loops back through again).

Circuit 4: In the pre-lab, you made design decisions regarding a decade counter. Now it's time to build it! Use your selected flip-flop type to create a decade counter. The output should display on a single 7-segment display. A block diagram of the circuit is shown in figure 12.1.

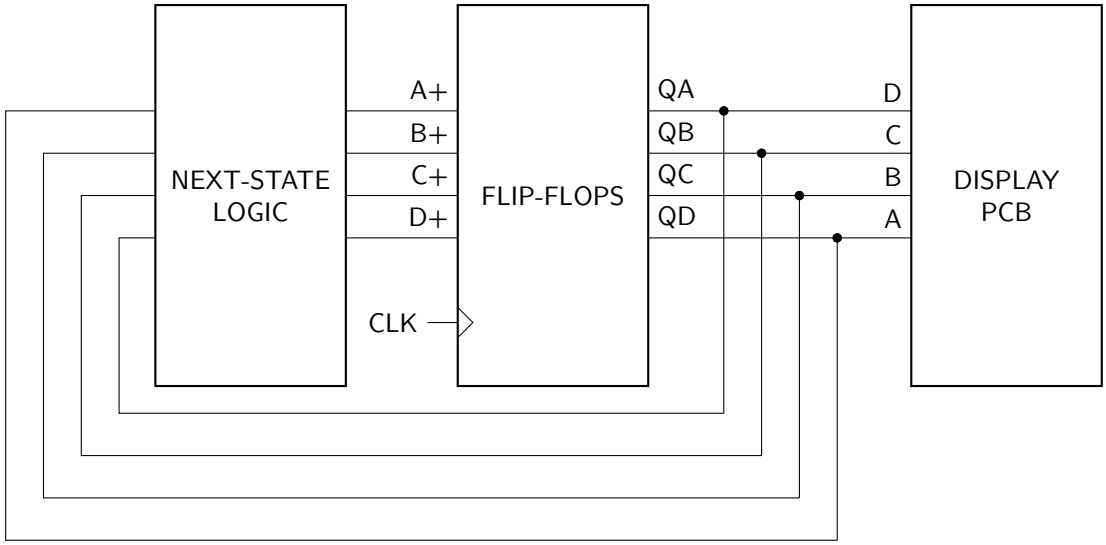
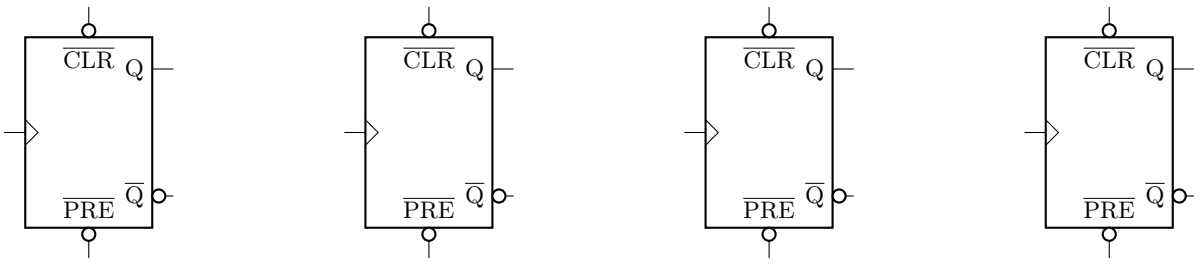


Figure 12.1: Block diagram of the synchronous counter circuit.

Draw a circuit diagram of your completed circuit. Label the inputs so it is clear if you are using a D or JK flip-flop. Demonstrate the functionality of your counter to your instructor to receive a stamp.

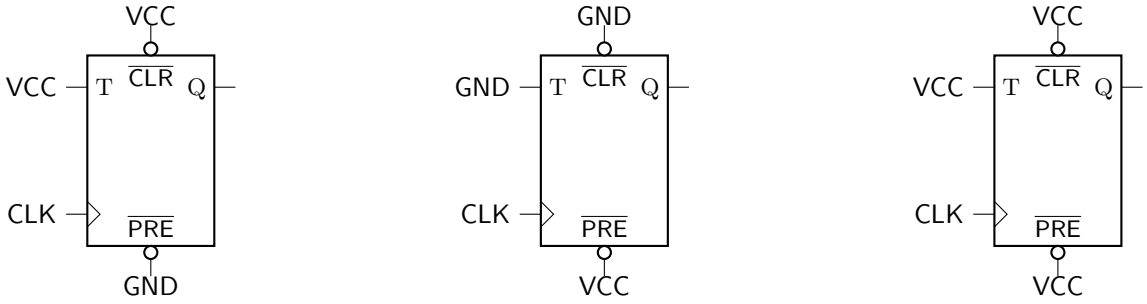


Instructor Stamp: _____

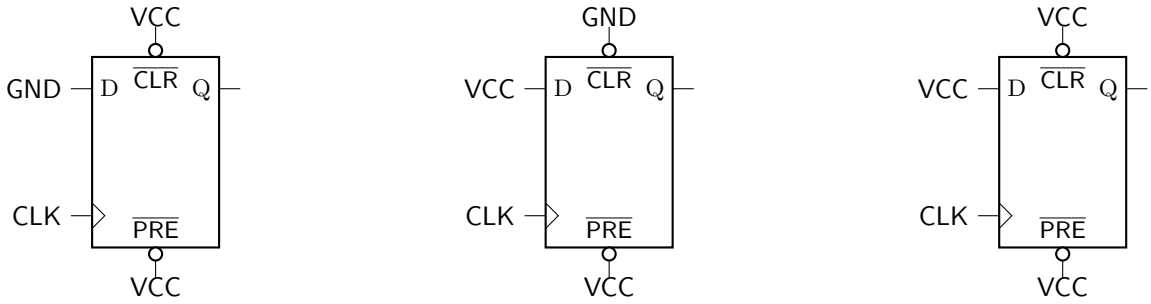
Lab 12 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

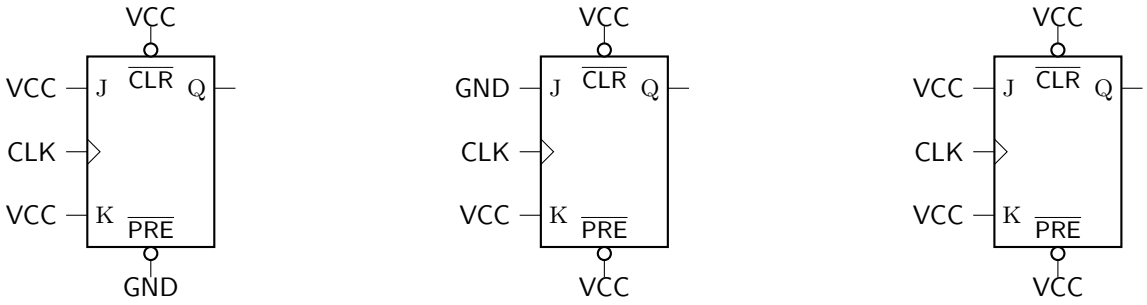
1. For each of the following T flip-flops, indicate if the value of the output is 0, 1, or will toggle.



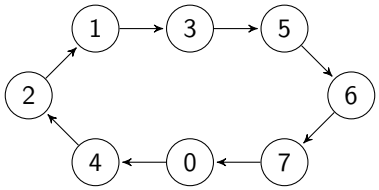
2. For each of the following D flip-flops, indicate if the value of the output is 0, 1, or will toggle.



3. For each of the following JK flip-flops, indicate if the value of the output is 0, 1, or will toggle.



4. Design a 3-bit counter that operates as shown in the state diagram below.



(a) Fill out the transition table below. You will use each of three (D, T, and JK) flip-flops to determine the lowest cost implementation of this synchronous counter. (As mentioned at the header of every lab homework, you are expected to show your work. Attach extra pages to your submission as needed to do so, as there are no blank k-maps provided on these pages.)

A	B	C	A^+	B^+	C^+	T_A	T_B	T_C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0												
0	0	1												
0	1	1												
0	1	0												
1	0	0												
1	0	1												
1	1	1												
1	1	0												

(b) Derive a minimum SOP expression for D_A.

(c) Derive a minimum SOP expression for D_B.

(d) Derive a minimum SOP expression for D_C.

(e) Derive a minimum SOP expression for T_A.

(f) Derive a minimum SOP expression for T_B.

(g) Derive a minimum SOP expression for T_C.

(h) Derive a minimum SOP expression for J_A.

(i) Derive a minimum SOP expression for K_A.

(j) Derive a minimum SOP expression for J_B.

(k) Derive a minimum SOP expression for K_B.

(l) Derive a minimum SOP expression for J_C.

(m) Derive a minimum SOP expression for K_C.

(n) Assuming that you have access to gates with as many inputs as you need, what type of flip-flop leads to the lowest cost circuit?

Pre-Lab 13

Carefully read the entirety of Lab 13, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Carefully read Circuit 1.

   (a) Define all of the necessary states (**and their corresponding outputs**) required to create the specified Moore machine.

   (b) Draw a state diagram.

   (c) Fill out the following state table.

Current State	**Next State**		**Output**
	X=0	**X=1**	

2. Carefully read Circuit 2 in the lab.

   (a) Define all of the necessary states required to create the specified Mealy machine.

   (b) Draw a state diagram.

   (c) Fill out the following state table.

Current State	Next State		Output	
	X=0	X=1	X=0	X=1

3. Carefully read Circuit 3 in the lab.

   (a) Fill out the following state table using the state diagram given in the lab. (Note the use of full Gray code in the state table!)

CS	**Next State**								**Output**							
	XYZ=								**XYZ=**							
	000	001	011	010	110	111	101	100	000	001	011	010	110	111	101	100
S0																
S1																
S2																

   (b) Decide what type of flip-flop (D or JK) you want to use to implement this finite state machine.

   (c) Create state assignments for each state. State S_0 should be assigned as 00.

(d) Represent the values of each next-state and output variable in the following truth table. Note that, depending on your selection of flip-flop, you may use either 3 output functions (D flip-flop) or 5 output functions (JK flip-flop). Label the corresponding output columns with the values JA/KA/JB/KB or DA/DB as needed. All unused outputs should have a value of 1. **There should be no don't care values on this truth table.** (Note, this truth table is very long and continues on the next page.)

	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	
XYZAB	–	–						T	HEX
00000	1	1							
00001	1	1							
00010	1	1							
00011	1	1							
00100	1	1							
00101	1	1							
00110	1	1							
00111	1	1							
01000	1	1							
01001	1	1							
01010	1	1							
01011	1	1							
01100	1	1							
01101	1	1							
01110	1	1							
01111	1	1							
10000	1	1							
10001	1	1							
10010	1	1							
10011	1	1							
10100	1	1							
10101	1	1							
10110	1	1							

10111	1	1							
11000	1	1							
11001	1	1							
11010	1	1							
11011	1	1							
11100	1	1							
11101	1	1							
11110	1	1							
11111	1	1							

(e) Use equation 13.1 to calculate the resistor and capacitor values required to create a pulse of approximately 4 s duration. Refer to the inventory of parts to select one resistor and one capacitor to achieve this within 10% of the desired pulse time.

Lab 13: Sequence Detectors

This lab will focus on sequence detecting finite state machines. Both synchronous (Moore) and asynchronous (Mealy) sequence detectors will be explored. Finally, a more complicated password unlock circuit will be built using a keypad as an input device.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-13

13.1 Moore Machines

Moore machines are finite state machines in which the output(s) only depends on the current state of the device, and not on any of the input value(s). The outputs of a Moore machine are therefore said to be synchronous. A Moore machine will have the same output (whose value is defined by the state table) throughout the duration of a state and will not change until the clock ticks and changes the state of the finite state machine.

Given the state diagram shown in figure 13.1, the timing diagram in figure 13.2 will result, given rising-edge triggered flip-flops. Note that the output is HIGH for the entire duration that the finite state machine is in state S2.

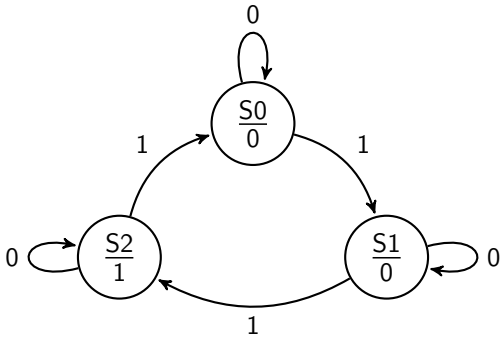


Figure 13.1: Example Moore machine state diagram.

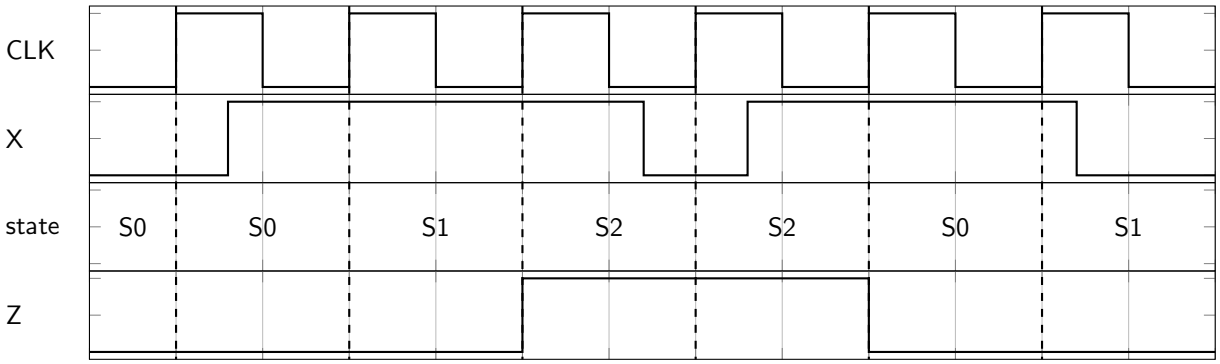


Figure 13.2: Timing diagram corresponding to the example state diagram shown in figure 13.1.

A Moore machine is therefore slightly easier to troubleshoot than a Mealy machine. Because the outputs are synchronous, the timing of the output signal is steadier and simpler to confirm using the troubleshooting methods outlined below. However, because Moore machines have outputs defined by their states, a Moore machine will have more states than an otherwise equivalent Mealy machine. This can lead to more difficulty in the initial design and setup of the circuit, and can lead to more complicated expressions for each flip-flop input.

13.2 Mealy Machines

Mealy machines are finite state machines in which the output(s) depends on **both** the current state of the device and on the input value(s). The outputs of a Mealy machine are therefore said to be asynchronous. A Mealy machine output will change as soon as the input changes to a value that would lead to a changing output based on the state table.

Given the state diagram shown in figure 13.3, the timing diagram in figure 13.4 will result, given rising-edge triggered flip-flops. This finite state machine has a very similar design to the Moore machine shown in figure 13.1, and the timing diagram has the same clock and input values as the Moore machine shown in figure 13.2. Note that the output is HIGH only when the input changes asynchronously to a HIGH value in state S2, and becomes LOW again immediately upon the next clock tick.

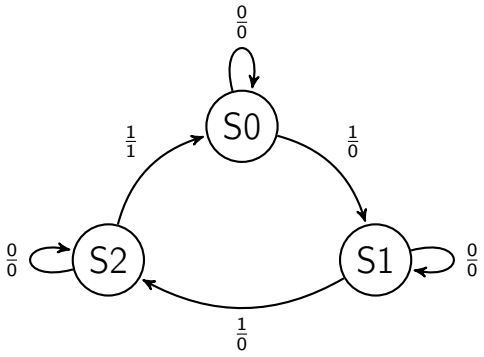


Figure 13.3: Example Mealy machine state diagram.

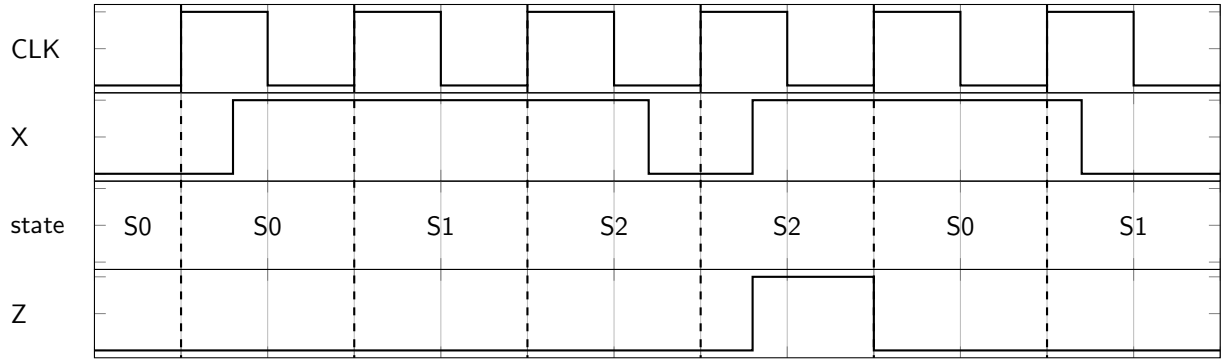


Figure 13.4: Timing diagram corresponding to the example state diagram shown in figure 13.3.

Mealy machines are therefore slightly more difficult to troubleshoot than Moore machines. The timing of the input is important to consider when building and debugging a Mealy machine. Because the output will change asynchronously with the input, it is important to note that the output may change only for a very short period of time, especially if the clock is about to tick and cause the state to change and cause the output to change back to its initial value. Output values can appear as a glitch in these cases. Having a solid understanding of the timing details is important.

13.3 Troubleshooting a Finite State Machine

Troubleshooting a synchronous circuit is generally more complicated than troubleshooting a combinational circuit. Because the circuit relies on feedback, and states are constantly changing, it can be difficult to obtain a steady state of input and output signals to verify using a logic probe. Following are a few troubleshooting tips that you can use with synchronous circuits.

- Use a slow clock signal to observe flip-flop and output values. (In this lab, you will be using a pushbutton or keypad as a clock signal which means the flip-flop and output values can be probed without worrying about fast transitions between states.)
- Place debug LEDs on the output of each flip-flop. They should cycle through values based on the state table and state assignments every time the clock ticks.

Circuit 1: Using a debounced pushbutton as a clock source, a DIP switch as input signal, and D or JK flip-flops (either way, ensure you are triggering state transitions on a rising edge), create a Moore machine to detect the binary sequence **001**. This should be a sliding window detector type circuit. Include a listing of each state's binary state assignment, a transition table, and next-state and output expressions. (Note that you will not receive credit for any of the above items without including state assignments, and creating a transition table and expressions based on those state assignments.) Display the output of each flip-flop on LEDs (so that you can see the transitions between states). In addition, display the output signal on an LED. Wire up the circuit and verify its functionality. A separate pushbutton should be included to asynchronously reset the FSM and place it into the reset state. Then, demonstrate it to your instructor to receive a stamp.

State Assignments:

Transition Table:

Current State	Next State		Flip-Flop		Output
	X=0	X=1	X=0	X=1	

Expressions:

Flip-Flop A equation(s): _____

Flip-Flop B equation(s): _____

Z = _____

Instructor Stamp: _____

Circuit 2: Using a debounced pushbutton as a clock source, a DIP switch as input signal, and D or JK flip-flops (either way, ensure you are triggering state transitions on a rising edge), create a Mealy machine to detect the binary sequence **001**. This should be a sliding window detector type circuit. Include a listing of each state's binary state assignment, a transition table, and next-state and output expressions. (Note that you will not receive credit for any of the above items without including state assignments, and creating a transition table and expressions based on those state assignments.) Display the output of each flip-flop on LEDs (so that you can see the transitions between states). In addition, display the output signal on an LED. Wire up the circuit and verify its functionality. A separate pushbutton should be included to asynchronously reset the FSM and place it into the reset state. Then, demonstrate it to your instructor to receive a stamp.

State Assignments:

Transition Table:

Current State	Next State		Flip-Flop		Output	
	X=0	X=1	X=0	X=1	X=0	X=1

Expressions:

Flip-Flop A equation(s): _____

Flip-Flop B equation(s): _____

Z = _____

Instructor Stamp: _____

13.4 Using EPROM as Next-State and Output Logic

Rather than connecting external combinational logic gates to generate next-state and output logic, an EPROM chip can be used instead. This has a few benefits. First, as there are many output pins on a ROM chip (typically 8, but sometimes 16), all next-state and output logic can be implemented on one chip. Second, because an EPROM acts as a truth table of stored binary values, there is no need to find minimized expressions for each function. This also means that the choice of state assignments will not meaningfully affect the complexity of the circuit, so there is no need to find optimum state assignments.

To use EPROM in this manner, first determine your inputs (any state machine inputs will be included, along with current state values) and decide which address pins you will connect them to. It is strongly recommended to use the least significant address pins for your inputs and ground any unused address pins. Then, determine which output pins you will use for your next-state values and any output values. Based on your address pins, generate a truth table and program the corresponding HEX values onto the chip. Unused output pins should be left in the unprogrammed (HIGH) state, and unused output pins should be left floating.

13.5 Keypad

A keypad is a device with several pushbuttons (usually 12 or 16 of them) that correspond to different values. Used with an encoder, the output of a keypad circuit will correspond to the binary value of the button that was pressed. A side-by-side comparison of a keypad (used with an encoder) and a DIP switch is presented in table 13.2.

Keypad with Encoder	**DIP Switch**
No knowledge of binary is needed	Requires knowledge of binary
The output values all change at once	The output value only changes one bit at a time as the user toggles each switch
A keypad requires an encoder to scan each row and column for a button press	Does not require an encoder because each switch contains connections to VCC and GND
The output value persists until the next button press	The output value persists until a switch is toggled
The DATA pin on the encoder creates an asynchronous signal to alert that an input has been selected	External circuitry or a pushbutton would be required to alert that an input has been selected

Table 13.2: Side-by-side comparison of a keypad with encoder and a DIP switch.

The keypad used in this lab has 12 buttons corresponding to values 0–9, as well as symbols that output binary values corresponding to decimal values of 10 and 11. Each keypad has been wired up on a PCB with the 74922 keypad encoder used in asynchronous data entry mode. Schematics of the 74922 encoder, the 12-character keypad, and the keypad PCB are all provided in Appendix A.

The keypad encoder has an output pin (DATA) that causes a temporary HIGH signal any time a keypad button is pressed. This pin is the perfect candidate to act as a clock source for a finite state machine requiring an input value from a keypad. Rather than synching user input to a clock signal, the state can change asynchronously based on the user button press. This creates a much more user-friendly interface, and does not require any extra circuitry that would cause the finite state machine to go into a WAIT or IDLE mode while waiting for the user to input a new value.

13.6 Monostable Timer

Previously, a 555 timer had been used in astable mode to create a signal that continuously oscillates between LOW and HIGH values. This was used as a clock signal for flip-flops.

In this lab, a 555 timer will be used in **monostable** mode to create a single HIGH pulse (sometimes called a one-shot) upon a triggering event. When the trigger pin on the timer (in monostable mode) is set to logic LOW, the timer will generate a HIGH output signal for the time period described by equation 13.1, otherwise the signal output will be LOW.

$$t \approx 1.1RC \tag{13.1}$$

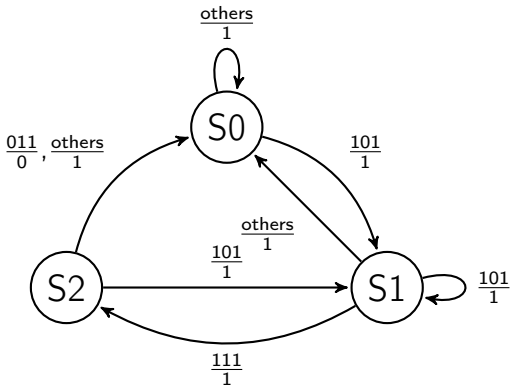
A monostable timer is the ideal solution for a circuit that requires a finite interval of time to elapse between events. Rather than relying on a clock signal to synchronize the timing of states for that duration, a one-shot pulse can be used instead. Because the monostable timer requires a LOW signal to generate a pulse, that active-LOW property will need to be taken into account in the design of a circuit.

Circuit 3: Design a three digit Mealy Machine sequence detector with input coming from a keypad wired up to a keypad encoder chip. Use either D or JK flip-flops. A an RGB status LED will be red when the system is locked and green when it is unlocked. The system will unlock for 4 seconds when the sequence **5–7–3** is entered into the keypad. The keypad numeral will also be displayed simultaneously on a 7-segment display. The clock signal will come from pin 12 of the keypad encoder (labeled `DATA`). (Note that, to avoid setup issues, the flip-flops will require a falling-edge trigger.) It is possible that other combinations (such as 5–7–2) may cause a noise issue on the monostable timer, causing the trigger pin to become LOW unexpectedly. If this happens, place a 100 nF ceramic capacitor between the trigger pin of the 555 timer and ground. A separate pushbutton should be included to asynchronously reset the FSM and place it into the reset state.

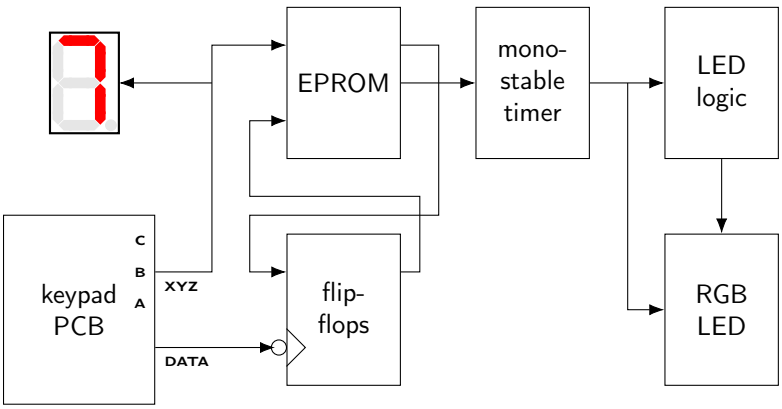
Wire up the circuit and verify its functionality. Demonstrate it to your instructor to receive a stamp. A high-level schematic of this circuit is provided below.

State Definitions and Diagram:

- S_0 – Reset state (no numbers toward the sequence have been detected, or, the sequence has been successfully completed)
- S_1 – 5 (the numeral 5 has been detected toward completing the sequence)
- S_2 – 5–7 (the numeral 7 has been detected after the numeral 5)



High-Level Schematic:



Instructor Stamp: _____

Lab 13 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Design a **non-overlapping sliding window** Moore machine detector that has an output $Z = 1$ if the sequence **0110** is detected, and otherwise the output $Z = 0$.

   (a) Define all of the states required to implement this.

   (b) Draw a state diagram

   (c) Fill out the following state table

Current State	Next State		Output
	X=0	X=1	

(d) Define each of the state assignments. Ensure that the reset state is properly assigned.

(e) Fill out the following transition table

Current State	**Next State**		**Output**
	X=0	**X=1**	
000			
001			
011			
010			
100			
101			
111			
110			

(f) Derive a minimum expression for A^+ using a D flip-flop.

(g) Derive a minimum expression for B^+ using a D flip-flop.

(h) Derive a minimum expression for C^+ using a D flip-flop.

(i) Derive a minimum expression for Z.

2. Design an **overlapping sliding window** Mealy machine detector that has an output $Z = 1$ if either the sequence **100** or **1001** is detected. Otherwise the output $Z = 0$.

   (a) Define all of the states required to implement this.

   (b) Draw a state diagram

   (c) Fill out the following state table

Current State	Next State		Output	
	X=0	X=1	X=0	X=1

(d) Define each of the state assignments. Ensure that the reset state is properly assigned.

(e) Fill out the following transition table

Current State	Next State		Output	
	X=0	X=1	X=0	X=1
000				
001				
011				
010				
100				
101				
111				
110				

(f) Derive a minimum expression for A^+ using a D flip-flop.

(g) Derive a minimum expression for B^+ using a D flip-flop.

(h) Derive a minimum expression for C^+ using a D flip-flop.

(i) Derive a minimum expression for Z.

Pre-Lab 14

Carefully read the entirety of Lab 14, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Using the provided state diagram, fill out the following state table.

C.S.	Next State				Asynchronous Outputs TL/TS				Synchronous Outputs					
	LS=				LS=									
	00	01	11	10	00	01	11	10	MR	MY	MG	SR	SY	SG
S0														
S1														
S2														
S3														

2. You will implement this finite state machine using D flip-flops. Use the guidelines to determine state assignments, and note the state assignments below. State S_0 will be the reset state.

3. Fill out the transition table.

C.S.	Next State				Asynchronous Outputs TL/TS				Synchronous Outputs					
	LS=				LS=									
	00	01	11	10	00	01	11	10	MR	MY	MG	SR	SY	SG
00														
01														
11														
10														

4. Derive a minimum expression for D_A.

5. Derive a minimum expression for D_B.

6. Derive a minimum expression for T_L.

7. Derive a minimum expression for T_S.

8. Derive a minimum expression for M_R.

9. Derive a minimum expression for M_Y.

10. Derive a minimum expression for M_G.

11. Derive a minimum expression for S_R.

12. Derive a minimum expression for S_Y.

13. Derive a minimum expression for S_G.

14. Use equation 13.1 to calculate the resistor and capacitor values required to create a pulse of approximately 5 s duration. Refer to the inventory of parts to select one resistor and one capacitor to achieve this within 10% of the desired pulse time.

15. Use equation 13.1 to calculate the resistor and capacitor values required to create a pulse of approximately 2 s duration. Refer to the inventory of parts to select one resistor and one capacitor to achieve this within 10% of the desired pulse time.

Lab 14: Traffic Light State Machine

This lab will build on knowledge from previous lectures and labs to create a circuit based on a real-life example: a simple traffic light. The circuit uses a hybrid Mealy-Moore machine approach to integrate synchronous and asynchronous outputs to generate a completely autonomous (hands-off) circuit.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-14

Circuit 1: Create a finite state machine to control the traffic lights at an intersection. The green light on each street will be on for 5 seconds, and the yellow light will be on for 2 seconds. The pulses that set the green (L) and yellow (S) light timing will also serve as device inputs. When one traffic light is either green or yellow, the other traffic light must be red. This machine will be a hybrid of a Mealy and Moore machine in that some of the outputs will be asynchronous and others will not. For example, timing trigger outputs T_L, which triggers the 5 second pulse, and T_S, which triggers the 2 second pulse are asynchronous (Mealy-type) outputs. The color of the traffic lights, however, are synchronous (Moore-type) outputs and do not depend on inputs L and S.

A block diagram of the circuit is shown in figure 14.1.

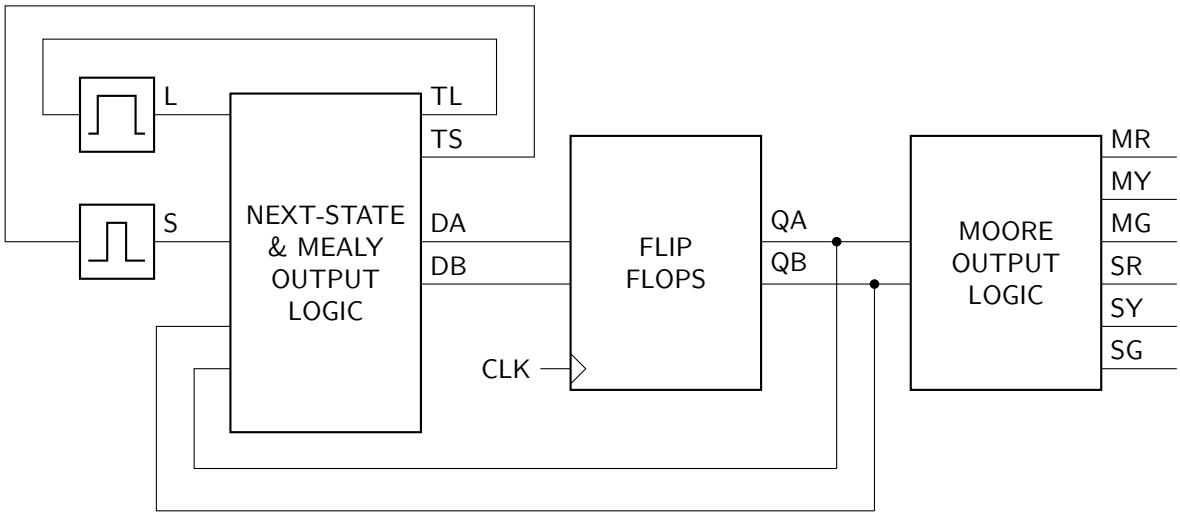


Figure 14.1: Block diagram of the traffic light circuit.

The state diagram in figure 14.2 will give you a starting point for your flip-flop and output equations. The asynchronous outputs are in form $T_L T_S$, and the synchronous outputs are in the form $M_R M_Y M_G S_R S_Y S_G$.

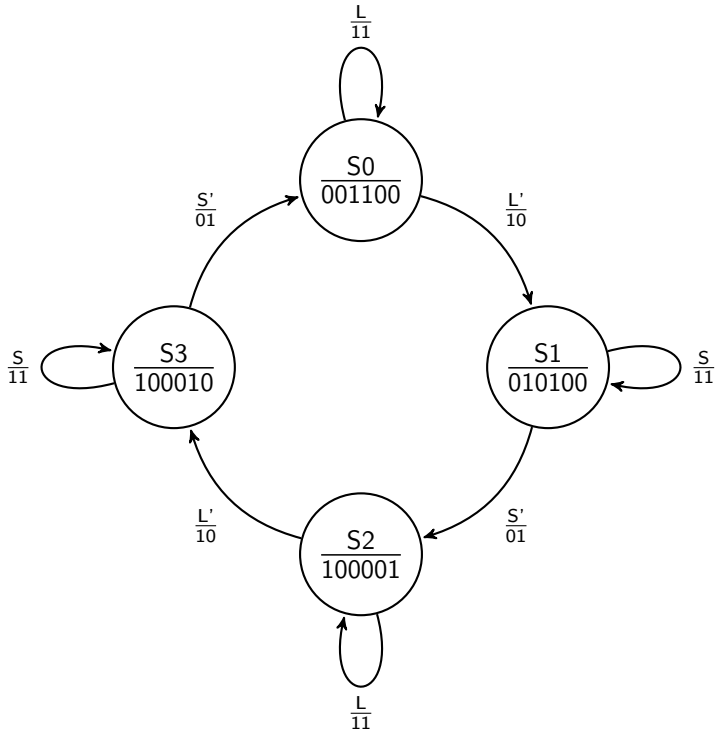


Figure 14.2: State diagram of the traffic light circuit.

You may use a function generator to create a clock frequency of **at least** 100 Hz and D flip-flops. Use the template below to draw a circuit diagram of the completed circuit. Wire up the circuit and demonstrate its functionality to your instructor to receive a stamp.

Instructor Stamp: _____

Lab 14 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

A vending machine requires 30¢ to dispense a can of soda. Users can insert nickels (N), quarters (Q) or dimes (D) in any combination. The vending machine can be implemented as a Mealy machine. There are two outputs to the state machine: V (vend) is 1 if the amount of money inserted is greater than or equal to 30¢, and H (change) is 1 if the amount of money inserted is greater than (but NOT equal to) 30¢. The state diagram is shown in figure 14.3.

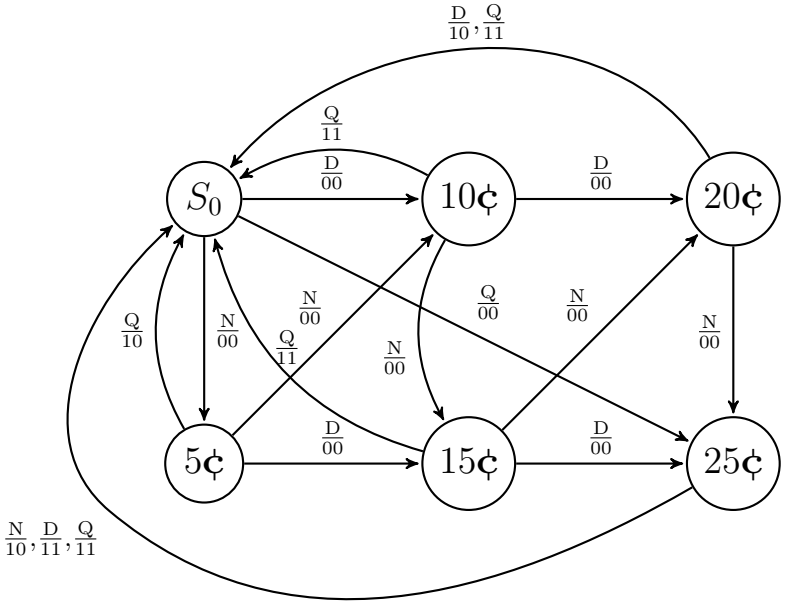


Figure 14.3: State diagram for the vending machine circuit.

1. Use figure 14.3 to fill out the state table for the circuit.

C.S.	Next State				Outputs (VH)			
	N 00	D 01	Q 11	× 10	N 00	D 01	Q 11	× 10
S0				×				×
5¢				×				×
10¢				×				×
15¢				×				×
20¢				×				×
25¢				×				×

2. How many flip-flops are required to implement this circuit?

3. Which state should be the reset state, and why?

4. Use the guidelines for state assignment to determine which states should be adjacent. Then, use a k-map to determine the state assignments. Indicate the binary values for each state.

5. Based on your state assignments, fill out the following transition table. Use D flip-flops.

C.S.	Next State				Outputs (VH)			
	N 00	D 01	Q 11	× 10	N 00	D 01	Q 11	× 10
S0				×				×
5¢				×				×
10¢				×				×
15¢				×				×
20¢				×				×
25¢				×				×

6. Derive a minimum expression for the output V.

XYA BC	000	001	011	010	110	111	101	100
00								
01								
11								
10								

7. Derive a minimum expression for the output H.

XYA / BC	000	001	011	010	110	111	101	100
00								
01								
11								
10								

8. Derive a minimum expression for next state A^+.

XYA / BC	000	001	011	010	110	111	101	100
00								
01								
11								
10								

9. Derive a minimum expression for next state B^+.

XYA BC	000	001	011	010	110	111	101	100
00								
01								
11								
10								

10. Derive a minimum expression for next state C^+.

XYA BC	000	001	011	010	110	111	101	100
00								
01								
11								
10								

Pre-Lab 15

Carefully read the entirety of Lab 15, then answer the following questions. Attach a separate sheet of paper, if necessary, to show all work and calculations.

1. Read the datasheet for the 74194 4-bit bidirectional universal shift register.

   (a) Is there an MSB for the data inputs? If so, which pin is it, and how do you know? If not, why isn't there an MSB and LSB?

   (b) As you will not be using the `SR SER` and `SL SER` pins, what should you do with them?

   (c) Does the register update outputs on rising or falling clock edges? How do you know?

   (d) Is it possible to asynchronously clear the register? If so, how?

   (e) Is it possible to asynchronously set the register? If so, how?

2. Based on the datasheet for the 74283 4-bit full adder, is there an MSB for the input and output pins? How do you know? If so, which pin(s) correspond to the most significant bits? If not, why isn't there an MSB and LSB?

3. Derive an equation to determine if the adder has overflowed. This is **not** the same as the overflow logic for adding signed binary numbers!

4. You want to clear the flip-flops and 74194 register when either the adder has overflowed or the reset pushbutton has been pressed. Derive an equation for an active LOW reset signal ($\overline{\text{RESET}}$) that will be 0 when that condition has been satisfied.

5. Fill out the state table using the state diagram shown in figure 15.2

CS	NS	Outputs	
		OS	CS
A			
B			
reset			

Lab 15: Register Control Logic

In this lab, the 74194 bidirectional shift register and 74283 parallel adder will be used to create a device that is capable of adding numbers together in sequence. It will require an input device, control logic, user-input buttons, memory, and output displays. This lab will integrate the topics of registers, adders, finite state machines, memory, and combinational logic.

For lab resources and information, go to the following URL or scan the QR code. doctor-pasquale.com/digital-systems-lab-15

15.1 Register Control

By controlling and utilizing registers, it is possible to repeat actions sequentially. This is in comparison to performing all actions simultaneously using combinational logic, which requires many more logic gates. For example, adding together four 4-bit numbers using combinational logic would require at least three adders. In general, $n-1$ adders are needed to add n numbers together. This quickly becomes unrealistic if many numbers need to be added together.

Instead of using a completely combinational logic approach, a single adder can be used along with a register. This allows numbers to be loaded and added together sequentially. The control logic tells the register when to load data and when to hold data. While the control logic adds a level of complexity to the circuit, it allows the device to be scalable. In essence, it becomes a microprocessor (a finite state machine capable of carrying out instructions), albeit very small and very specialized.

15.2 4-Bit Register Adder

This lab will feature a 4-bit adder that uses register control logic to sequentially add a 3-bit number input to the current 4-bit output value. Two pushbuttons will also be used to asynchronously control the operation of the adder. One of the pushbuttons will be used to asynchronously clear all of the registers. The other pushbutton will be used to start the register control logic in order to load and add the most recent number to the currently stored sum. The data will be output using EPROM and 7-segment displays. Combinational logic will reset all of the registers when the output will yield a value greater than 15.

The functional block diagram for the device is given in figure 15.3. Each of the functional parts will be described in more detail below.

15.2.1 User Input

The user can input a 3-bit number using a DIP switch. This number displays on a 7-segment display and is loaded directly into a 4-bit adder.

15.2.2 Load Logic

When the user presses the load button, a few things happen. The finite state machine (FSM) is set into motion, causing the current sum register (`CS`) to load with the value that is currently held at the output of the 4-bit adder. The next state of the FSM loads the old sum register (`OS`) with that same value. This keeps the current sum one step ahead of the previous sum, and prevents the adder from continuously being inundated with new data. The final state of the FSM is to put both registers into hold mode, preventing them from obtaining any new data until the load button is pressed again. The control logic block in the diagram represents the flip-flop(s) used to create the hold and load instructions for each register in order.

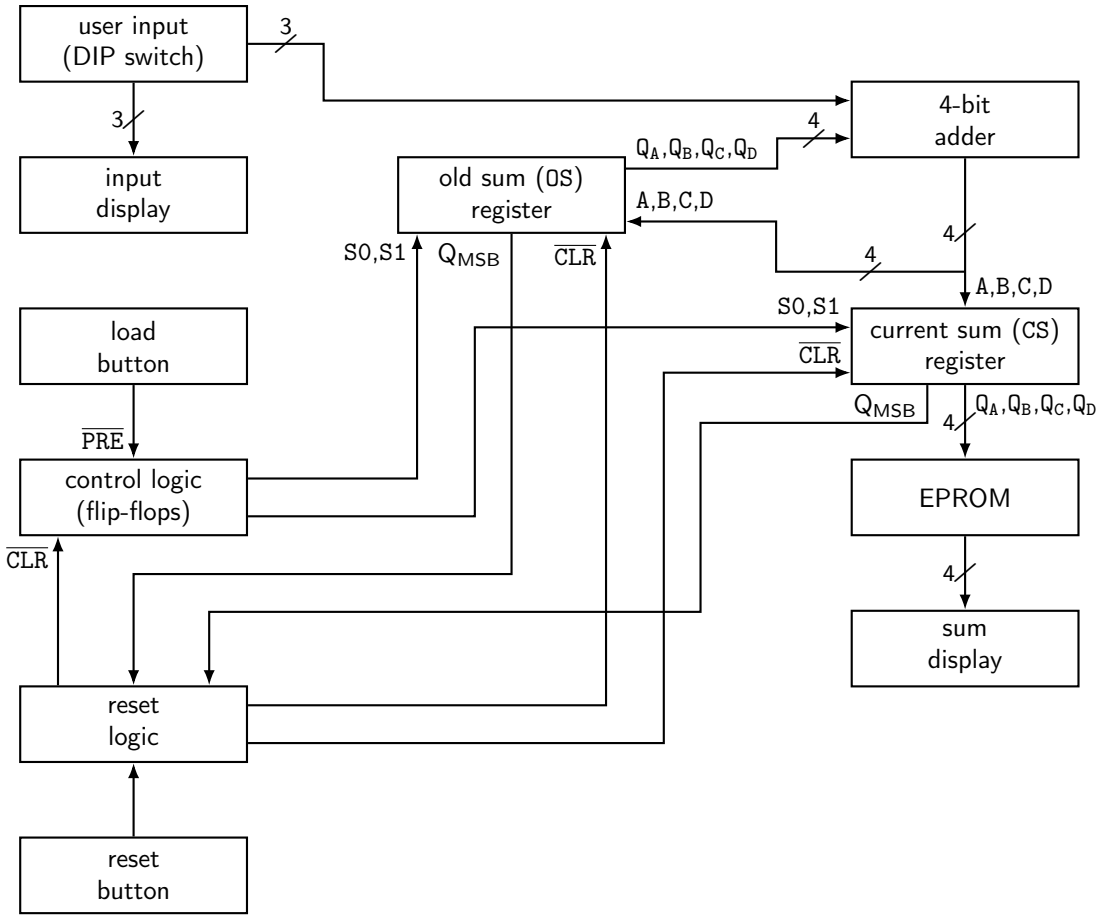


Figure 15.1: Block diagram of the register adder circuit.

15.2.3 Reset Logic

There are two ways for the adding machine to reset. A reset will cause both of the registers to be cleared and the FSM to go to its default (reset) state. One way to reset the adding machine is asynchronously upon pressing a reset pushbutton. This allows the user to reset the sum to zero at any time. The other situation that resets the adding machine is when the output is about to overflow (for example, the current sum is 13 and the user wants to add 7 to that value). This overflow situation can be detected when the MSB of the old sum register (`OS`) is 1 (indicating a number between 8–15 is currently stored in the register) and the MSB of the current sum register (`CS`) is 0 (indicating that the new value has overflowed and is reading artificially as less than 8).

15.2.4 Sum Display

Displaying the sum of all previously added numbers is the last step. The value in the current sum register (`CS`) is a 4-bit binary number capable of storing decimal numbers between 0–15. Two 7-segment displays are therefore needed to accommodate all ranges of values. EPROM is used to decode the 4-bit binary number into BCD numbers, using a similar method that was used in the memory lab. Those BCD numbers are then used as inputs for each of the 7-segment displays.

15.2.5 Finite State Machine (FSM) States

Each of the state definitions for the register adder control logic finite state machine follow.

- **Reset State:** When the system is first powered up, the default (reset) state is for both registers `CS` and `OS` to hold their current values of 0. This state is also triggered by the reset logic. This ensures that nothing happens until the load pushbutton is asserted.
- **State A:** When the load button is asserted, the FSM sends a load signal to `CS` while simultaneously sending a hold signal to `OS`. After this state, state B occurs.
- **State B:** The FSM sends a hold signal to `CS` while simultaneously sending a load signal to `OS`. After this state, the FSM returns to its reset state.

The state diagram corresponding to the register adder is shown in figure 15.2.

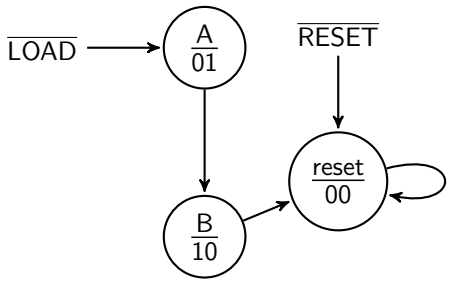


Figure 15.2: State diagram of the register adder circuit.

15.2.6 Pushbutton Inputs

The inputs are $\overline{\texttt{LOAD}}$ (which comes from the pushbutton) and $\overline{\texttt{RESET}}$ (which comes from the reset logic). These inputs asynchronously change the state of the FSM. For this reason, it is useful to have one of inputs **clear** the state logic and the other input **set** the state logic. If $\overline{\texttt{LOAD}}$ is configured to set the FSM, then state A would have a binary assignment of 11. Therefore $\overline{\texttt{RESET}}$ would clear the FSM and the reset state would have a binary assignment of 00. Because the clear and preset inputs of the 7474 flip-flops are active LOW, it is highly recommended that $\overline{\texttt{LOAD}}$ and $\overline{\texttt{RESET}}$ be active LOW signals! (This explains why they have been written in this lab with bars over them.)

Note that it is important to prevent the adder from generating an updated sum when the $\overline{\texttt{LOAD}}$ pushbutton is pressed. For this reason, your logic for the current sum mode select inputs (designated as `CS` in this lab) should be zero when the $\overline{\texttt{LOAD}}$ pushbutton is pressed.

Circuit 1: Select binary state assignments for states A and reset. (You are not given a choice for state B.)

State	Assignment
A	
B	01
reset	

Fill out the corresponding transition table.

CS	NS	Flip-Flop Values	Outputs	
			OS	CS
00				
01				
11				
10				

Derive a minimum expression for...

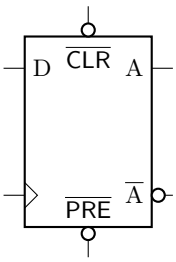
1. D_A

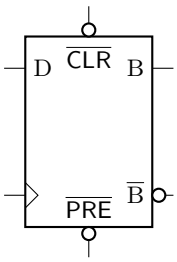
2. D_B

3. OS

4. CS

Build the adder using all of the design criteria given in the lab and the decisions that you have made so far. You may use a function generator to create a clock signal, using a frequency of **at least 100 Hz**. Use D flip-flops for all control logic. Draw the circuit diagram for the FSM and synchronous outputs. Wire up the circuit and demonstrate its functionality to your instructor to receive a stamp.





Instructor Stamp: _____

Lab 15 Homework

Carefully read each question before answering. Show all work or justify your answers to receive credit. Attach a separate sheet of paper, if necessary, to show all work and calculations.

You are building a very simple microprocessor that is capable of performing various operations on 8-bit numbers. The components of this microprocessor include the following, which are depicted in a block diagram shown in figure 15.3.

- A **program counter** that stores the address of the next instruction to be executed.
- **Memory** that contains all of the program instructions that tell the microprocessor what to do.
- A **memory register (MR)** that contains the contents of memory that will be executed next.
- An **instruction decoder (ID)** that figures out how to convert the instruction into command signals to add, subtract, or do any of the other operations.
- An **arithmetic and logic unit (ALU)** that executes the operation.
- **General purpose (GP) registers** that store the operands and the result of the operation.
- A **data bus** that contains information to be accessed by other microprocessor components.

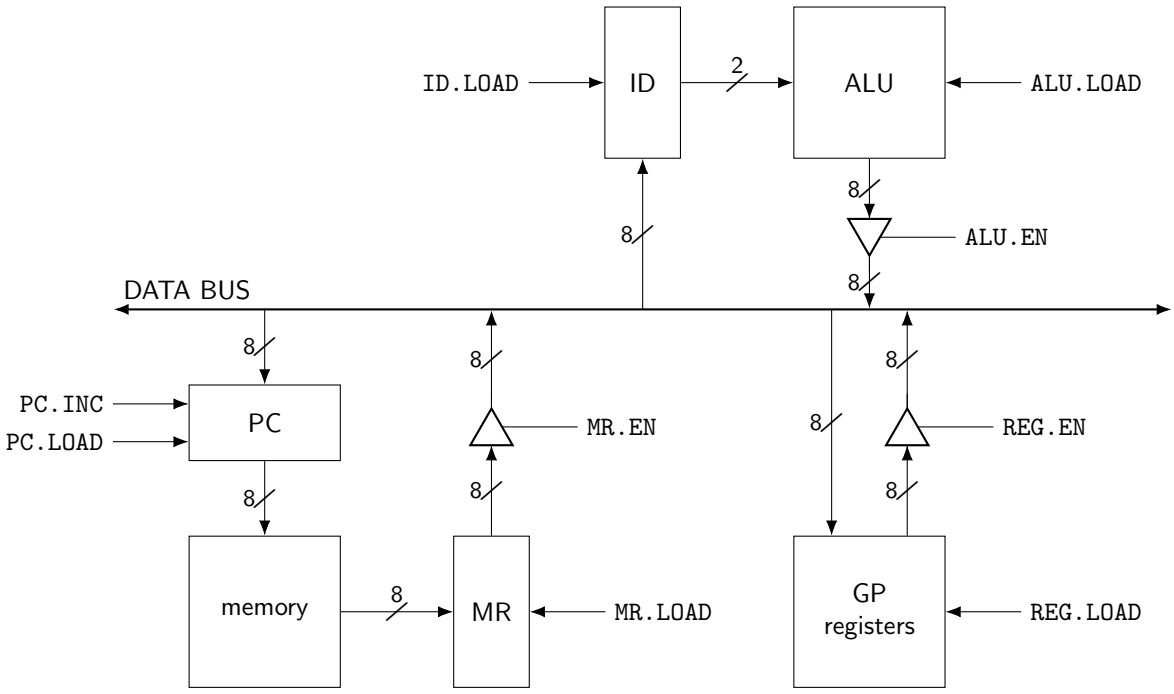


Figure 15.3: Block diagram of a microprocessor.

The finite state machine has the following states to execute these actions. All signals are active HIGH enable.

1. State `S0` – Load the memory with the contents of the program counter by asserting `PC.LOAD`.
2. State `S1` – Increment the program counter by asserting `PC.INC`, and load the memory data register by asserting `MR.LOAD`.
3. State `S2` – Load the instruction decoder by asserting `MR.EN` and `ID.LOAD`.
4. State `S3` – (Go here if `x = 1` which indicates that an operation will be carried out, otherwise return to start if `x=0`, indicating no operation will be carried out.) Fetch the operands from the GP register by asserting `REG.EN`, and write them to the ALU by asserting `ALU.LOAD`.
5. State `S4` – (Go here only after state `S3`.) Execute the operation and write the result to the GP register by asserting `ALU.EN` and `REG.LOAD`.

To clarify what the input variable x means: x is 0 if the instruction that is to be carried out does not require the ALU (arithmetic and logic unit). x is 1 if the instruction does require the ALU. All instructions must go through states S0–S1–S2. x is not a "stay put" or "return to S0" variable.

1. Is this a Mealy machine or a Moore machine? How do you know?

2. How many flip-flops will be needed to create this finite state machine?

3. Fill out the corresponding state table.

C.S.	Next State		Outputs								
	x=0	x=1	PC.LOAD	PC.INC	MR.LOAD	MR.EN	ID.LOAD	REG.EN	ALU.LOAD	ALU.EN	REG.LOAD
S0											
S1											
S2											
S3											
S4											

4. Decide which type of flip-flop you will use to implement this finite state machine.

5. Use the guidelines for state assignment, determine which states should be adjacent.

6. Which state should be the reset state, and why?

7. Based on the decisions made above, derive binary state assignments for each state. Clearly indicate the binary values used for each state.

8. Fill out the corresponding transition table.

C.S.	Next State		Flip-Flop Value		Outputs								
	x=0	x=1	x=0	x=1	PC.LOAD	PC.INC	MR.LOAD	MR.EN	ID.LOAD	REG.EN	ALU.LOAD	ALU.EN	REG.LOAD
000													
001													
011													
010													
100													
101													
111													
110													

9. Derive minimum next-state and output expressions. Show all of your work. You can double check your expressions with LogicAid, but you must derive the expressions by hand.

   (a) D_A

   (b) D_B

   (c) D_C

   (d) `PC.LOAD`

   (e) `PC.INC`

   (f) `MR.LOAD`

   (g) `MR.EN`

   (h) `ID.LOAD`

   (i) `REG.EN`

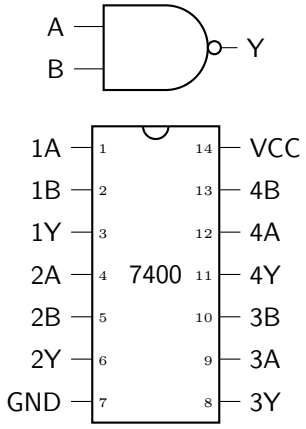
   (j) `ALU.LOAD`

   (k) `ALU.EN`

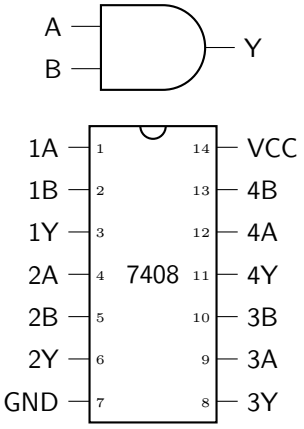
   (l) `REG.LOAD`

Appendix A: Pinout Diagrams

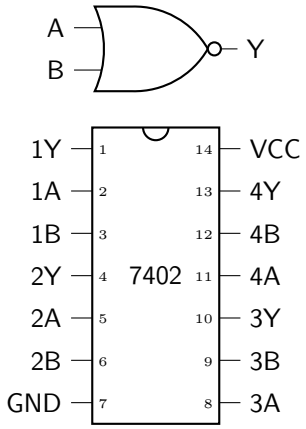
7400: Quadruple 2-Input NAND Gates



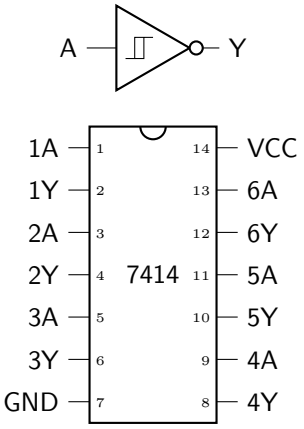
7408: Quadruple 2-Input AND Gates



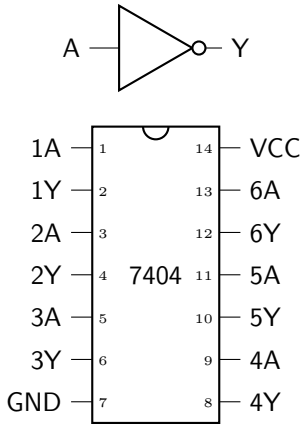
7402: Quadruple 2-Input NOR Gates



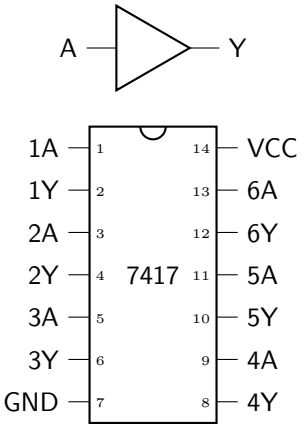
7414: Hex Inverters, Schmitt trigger inputs



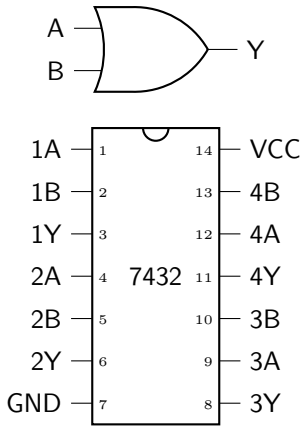
7404: Hex Inverters



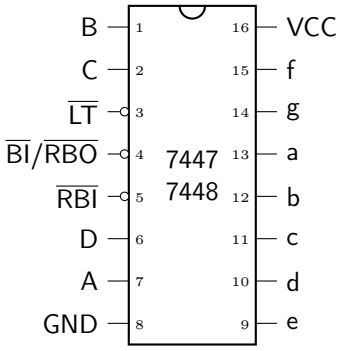
7417: Hex Buffers



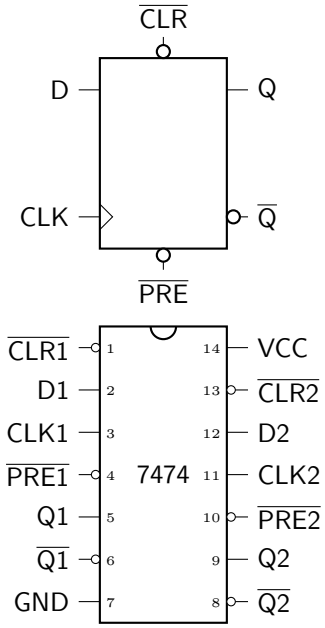
7432: Quadruple 2-Input OR Gates



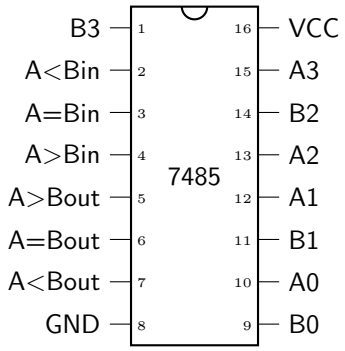
7447/7448: BCD-to-7-Segment Decoder



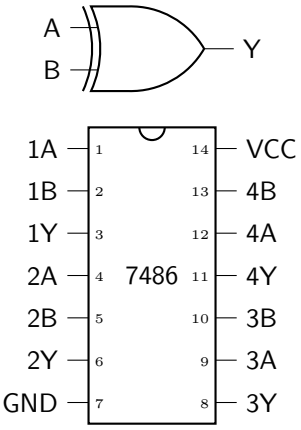
7474: Dual D-Type Flip-Flops with Clear and Preset



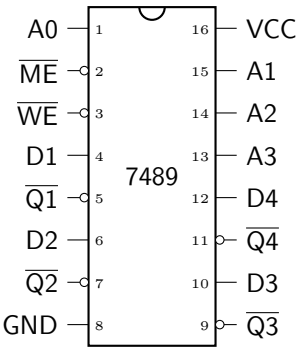
7485: 4-Bit Magnitude Comparators



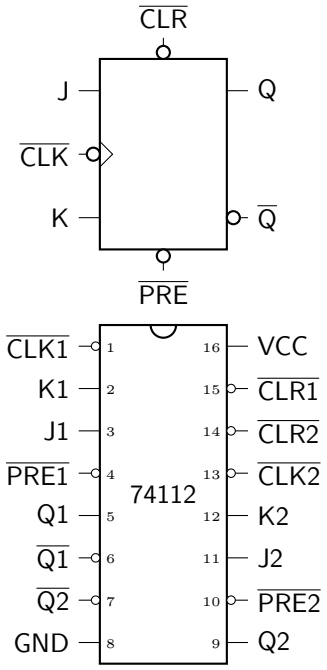
7486: Quadruple 2-Input Exclusive-OR Gates



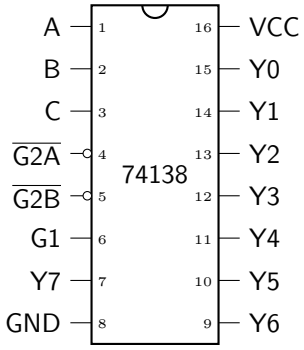
7489: 64-Bit Random Access Read/ Write Memory



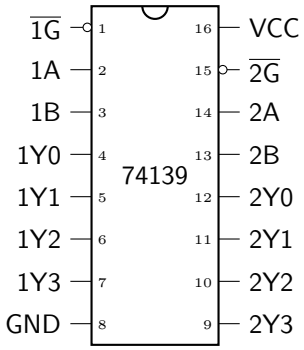
74112: Dual J-K Negative-Edge Triggered Flip-Flops with Clear and Preset



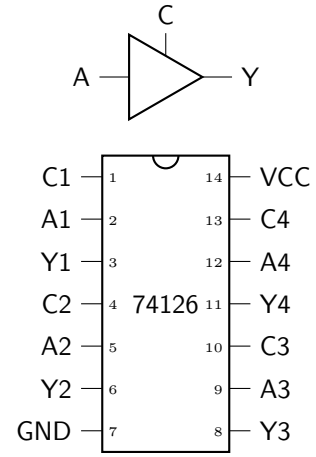
74138: 3-Line to 8-Line Decoders



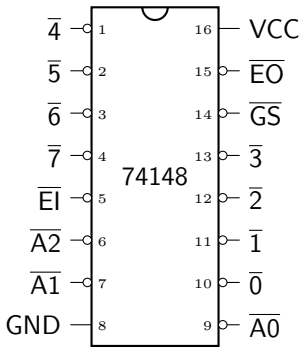
74139: Dual 2-Line to 4-Line Decoders



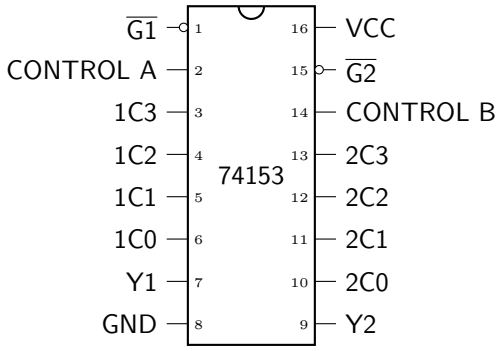
74126: Quadruple 3-State Buffer



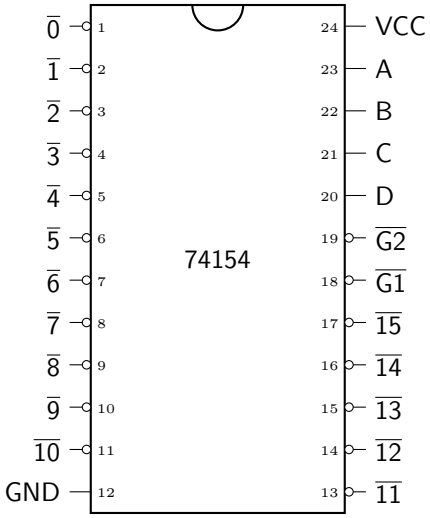
74148: 8-Line to 3-Line Priority Encoder



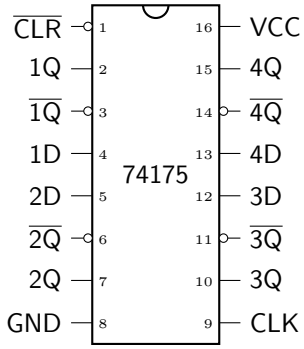
74153: Dual 4-Line to 1-Line Multiplexers



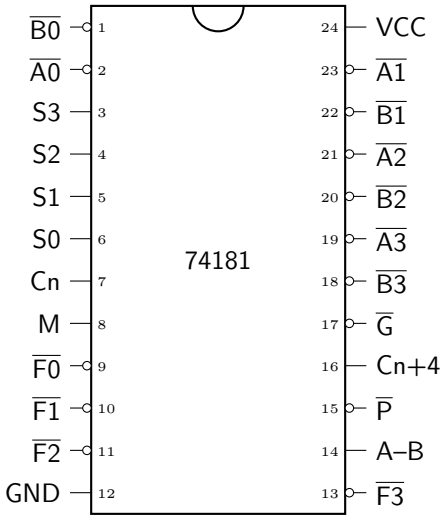
74154: 4-Line to 16-Line Decoder



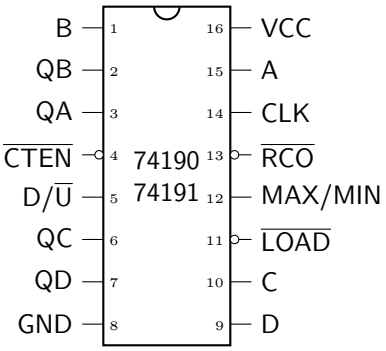
74175: Quadruple D-Type Flip-Flops with Clear



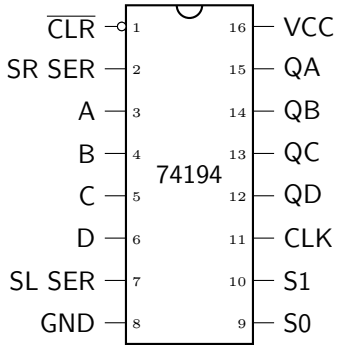
74181: ALU / Function Generator



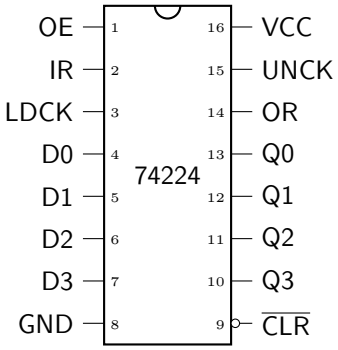
74190/74191: BCD/Binary Up/Down Counters with Mode Control



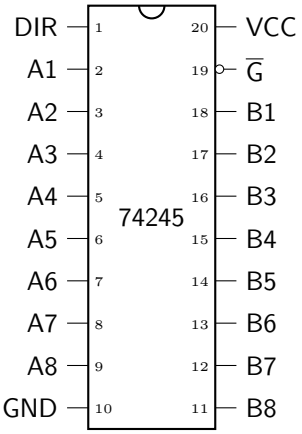
74194: 4-Bit Bidirectional Universal Shift Registers



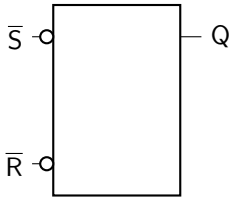
74224: 16x4 Synchronous First-In, First-Out Memories

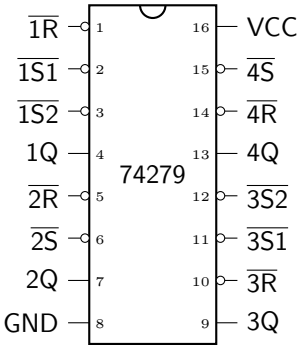


74245: Octal Bus Transceivers with 3-State Outputs

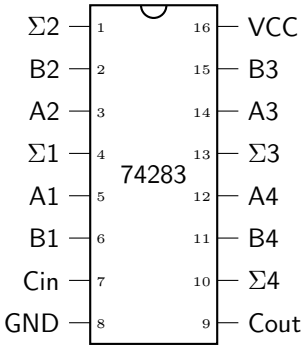


74279: Quad $\overline{\text{S}}$ $\overline{\text{R}}$ Latches

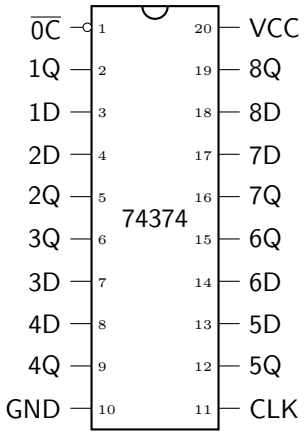




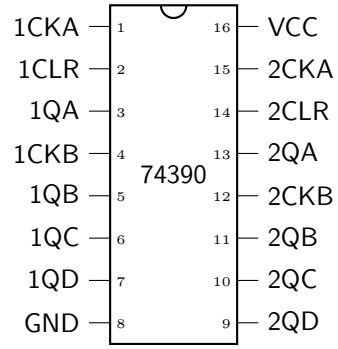
74283: 4-Bit Binary Full Adder with Fast Carry



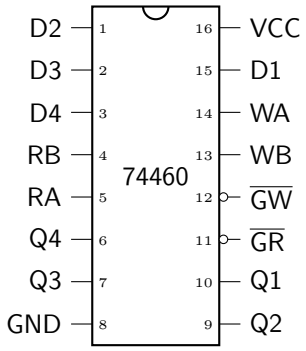
74374: Octal D-Type Edge-Triggered Flip-Flops



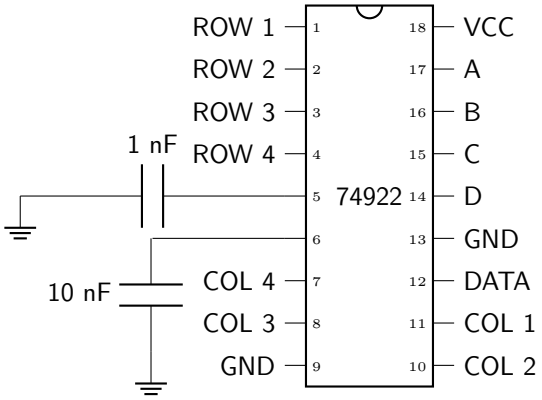
74390: Dual 4-bit decade counters



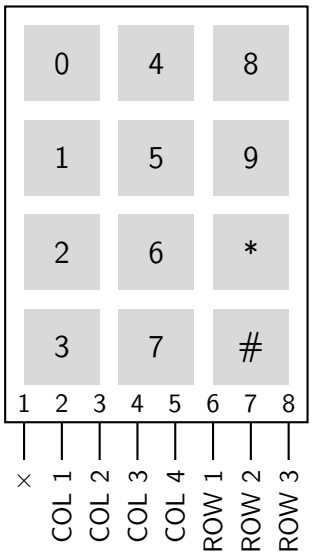
74670: 4-by-4 Register Files with 3-State Outputs



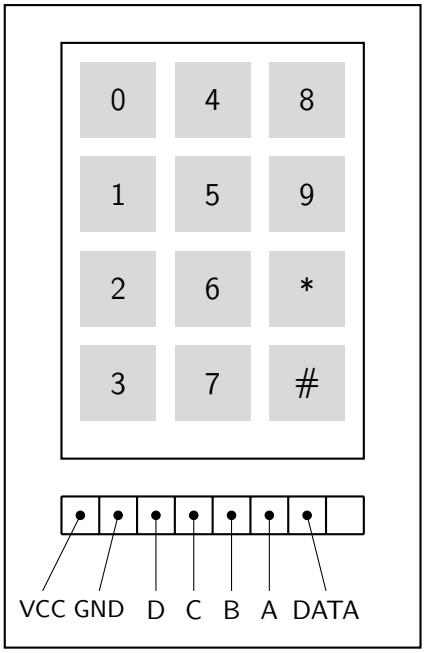
74922: 16 Key Encoder (Asynchronous Data Entry Mode)



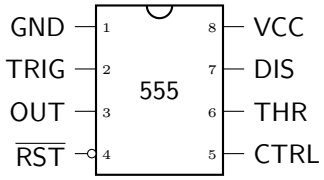
12-Character Keypad



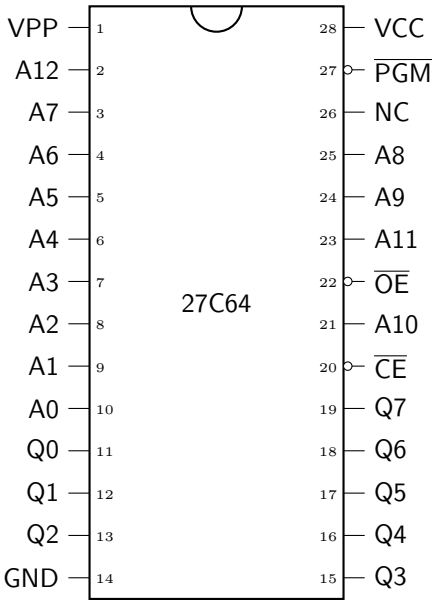
12-Character Keypad PCB



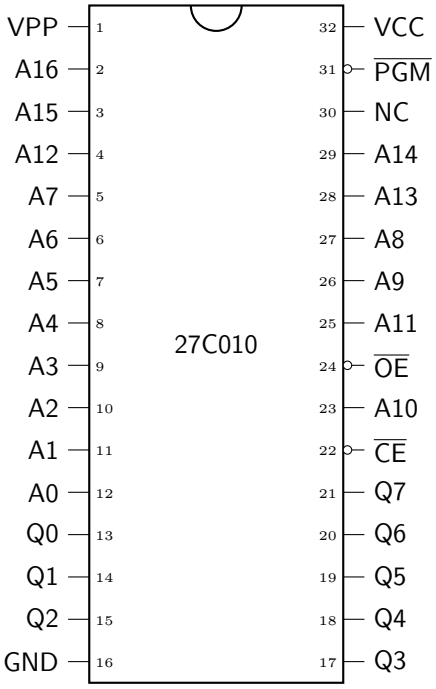
555 Timer



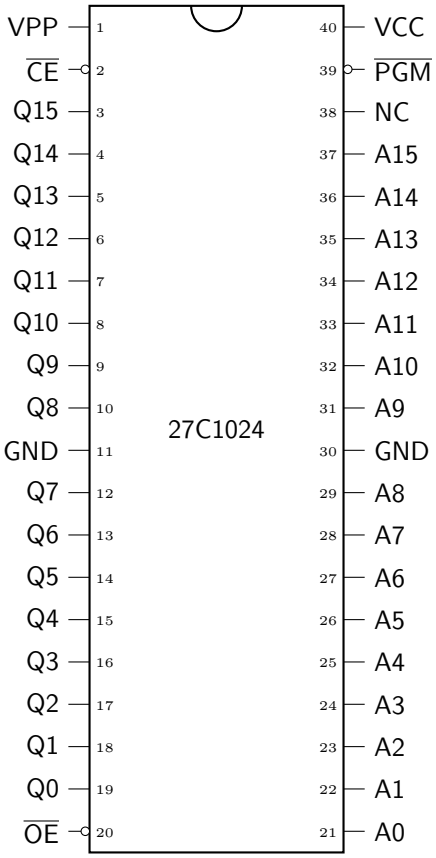
27C64: EPROM 8k × 8



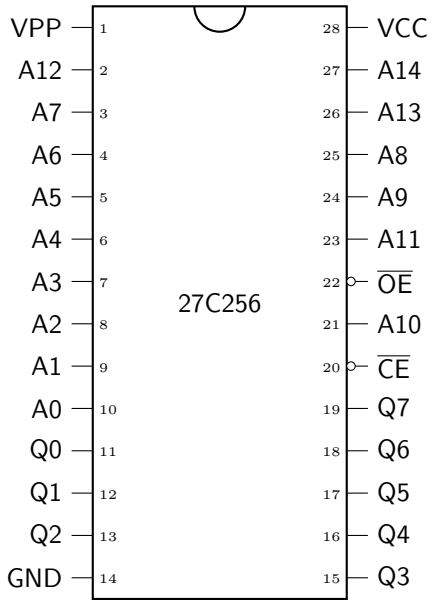
27C010: EPROM 128k × 8



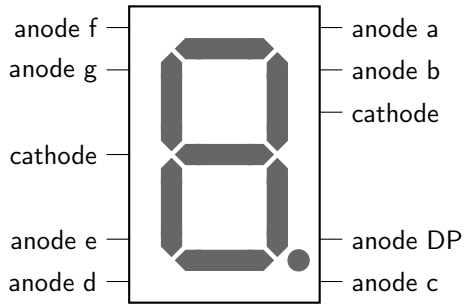
27C1024: EPROM 64k × 16



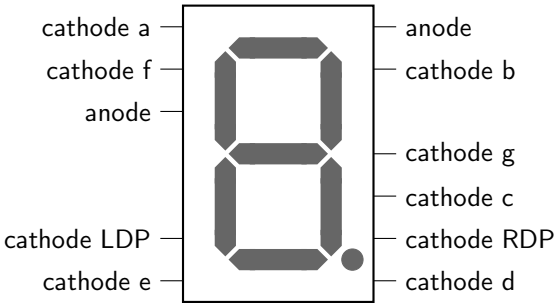
27C256: EPROM 32k × 8 (OTP)



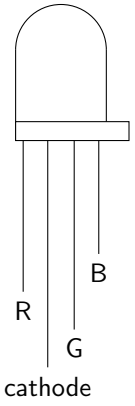
Common Cathode 7-Segment Display



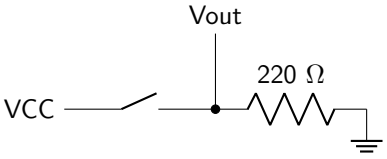
Common Anode 7-Segment Display



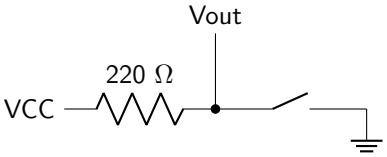
RGB LED



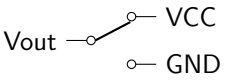
DIP Switch, Active-HIGH



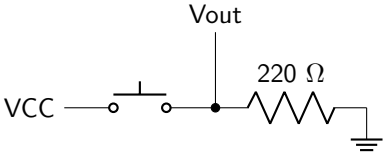
DIP Switch, Active-LOW



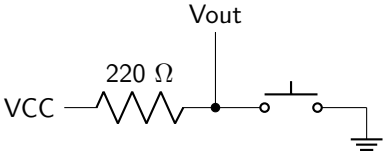
Toggle Switch



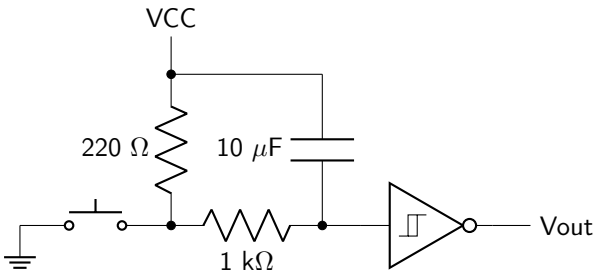
Pushbutton, Active-HIGH



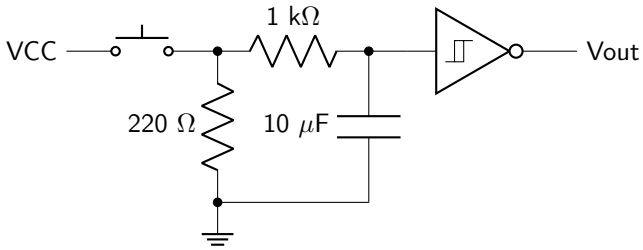
Pushbutton, Active-LOW



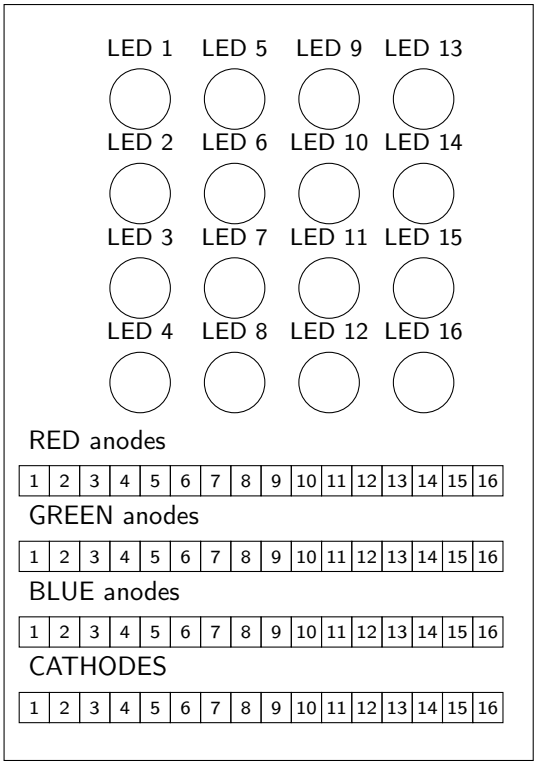
Debounced Pushbutton, Active-HIGH



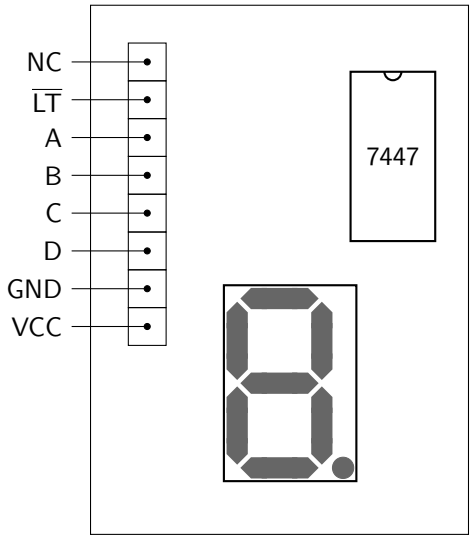
Debounced Pushbutton, Active-LOW



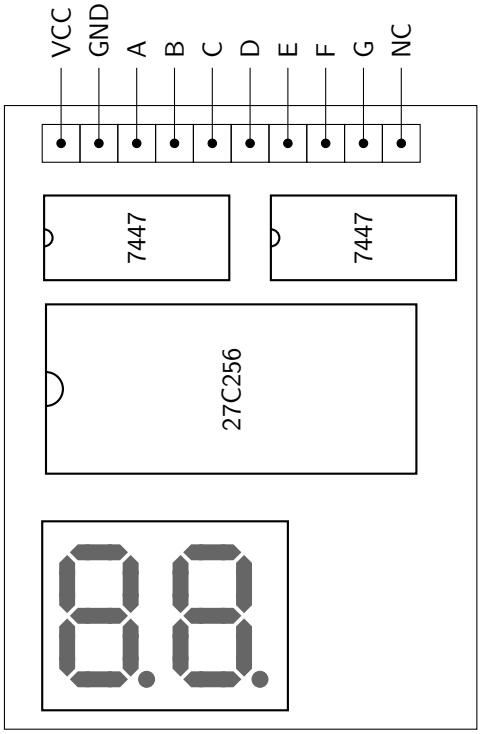
4x4 RGB LED Matrix PCB



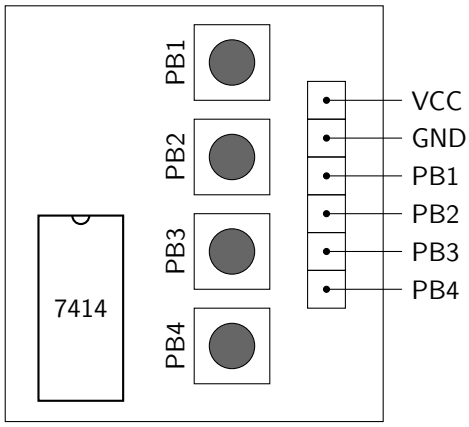
7-Segment Display PCB



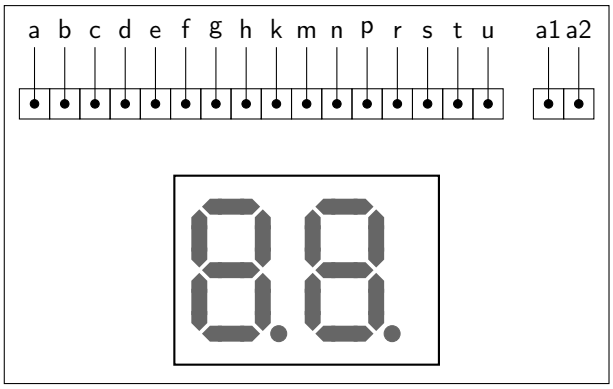
Binary to 0–99 Display PCB



Debounced Pushbutton PCB

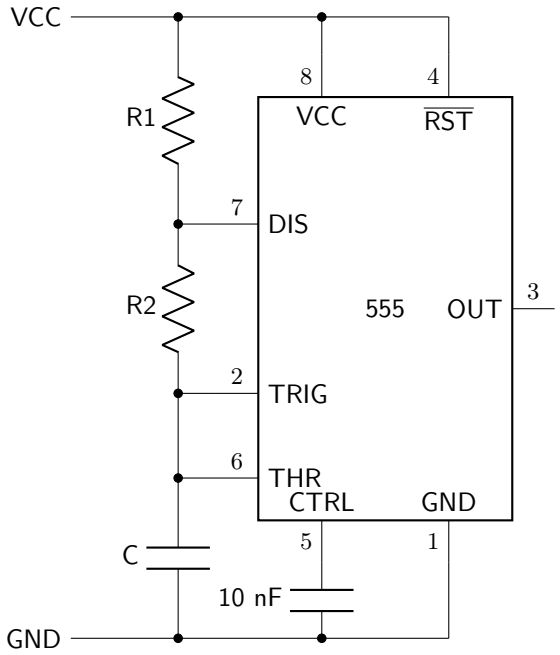


16-Segment Display PCB



Appendix B: 555 Timer Wiring Diagrams

Astable Operation



Monostable Operation