Digital Electronics: Logic and Clocks

1 Goals

In this lab, we will go over the basics of digital electronics. You will utilize some of the most basic components that are used to build more complex circuits, like computers. You will learn to use logic gates, memory circuits, and digital clocks (a kind of oscillator), and you will learn how to combine gates and predict the behavior with Boolean algebra.

This subject can easily span an entire semester, but we will just dip our toes in the water so that you will be equipped to employ some basic components into your final project if needed. The digital clock you will build in this lab, for example, has many applications. It is worth also investigating other kinds of oscillators for your project (if relevant).

1.1 Some Useful Definitions

Duty cycle - percentage of time during one cycle that a system is active (+5V in the case of TTL digital logic)

Truth-table - table that shows all possible input combinations and the resulting outputs of digital logic components

Flip-flop - a circuit that has two stable states and can be used to store state information

2 Intro to Digital Circuits

In almost all experiments in the physical sciences, the signals that represent physical quantities start out as *analog* waveforms. However, in the modern day, all our data eventually ends up on a computer (which is a digital circuit) to store and display. This requires analog-to-digital converters (ADC or A/D converter). There are plenty of commercially available ADCs that read voltages and connect to computers via USB, ethernet, RS-232, PCI, or PCIe; however, most instruments, such as the oscilloscope you use, have ADCs built in and communicate with a computer via USB, RS-232, ethernet, or GPIB. Scientists usually buy their data acquisition equipment rather than build it, so they usually don't have to know too much about the digital circuitry that makes it work.

In lab 5 you built a 3-bit digital-to-analog converter (DAC). Both DACs and ADCs have precision limited by the number of bits. A set of N bits has 2^N possible different values. The precision will be determined by the range of value and the number of possible values. If you try to represent an analog voltage by 7 bits, your minimum uncertainty will be about 1% of the total range, since there are $2^7 = 128$ possible combinations of 7 bits. For higher accuracy you need more bits.

There are three key components of digital circuits:

- logic
- memory
- timing

Transistors are the building blocks of all of these components, but both logic and memory can be exclusively made with **gates**.

DEFINITION: Logic gates - a simple transistor circuit that implements some Boolean logic operation.

In digital circuits, it is easier to abstract the discrete transistor circuits into gate circuits (similar to how we can treat an op-amp as a lumped element completely independent of the exact transistor layout that makes up the op-amp). In computers, gates are CMOS (where gates are designed with MOSFETs), but gates originated as TTL (where gates are made from BJTs). Most gates on DIP chips are still TTL (including the ones we will use in this lab).

Gates can be used to construct arbitrary combinatorial logic (they can generate any *truth-table*), but to create a machine that steps through a sequence of instructions like a computer does, we also need *memory* and a *clock*. The fundamental single-bit memory element of digital electronics is called a *flip-flop*. We will study two types, called SR (or RS) and JK. Both of these can be built exclusively with gates (the ones we use are TTL). A *digital clock* is a repeating digital waveform used to step a digital circuit through a sequence of states. We will introduce the 555 timer chip and use it to generate a clock signal. Digital circuits that are able to step through a sequence of states with the aid of flip-flops and a clock are called sequential logic.

The voltage in a digital circuit is allowed to be in only one of two states: HIGH or LOW. HIGH is taken to mean logical (1) or logical TRUE. LOW is taken to mean logical (0) or logical FALSE. In the TTL logic family (see Figure 1), the "ideal" HIGH and LOW voltage levels are $5\ \mathrm{V}$ and $0\ \mathrm{V}$, respectively, but any input voltage in the range of $2\text{–}5\ \mathrm{V}$ is interpreted as HIGH, and any input voltage in the range of $0\text{–}0.8\ \mathrm{V}$ as LOW. Voltages outside this range are undefined, and therefore "illegal," except if they occur briefly during transitions. If the input to a TTL circuit is a voltage in this undefined range, the response is unpredictable, with the circuit sometimes interpreting it as a "1" and sometimes as a "0." Avoid sending voltages in the undefined range into TTL components.

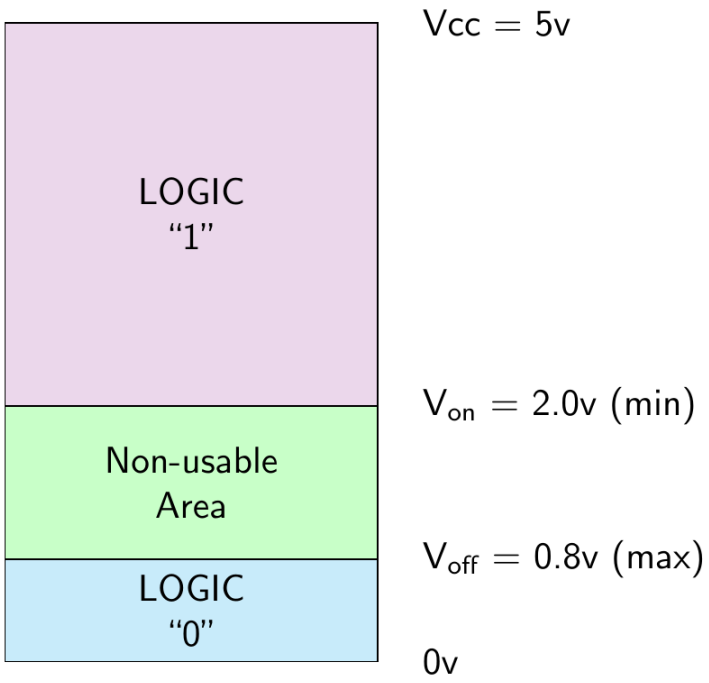


Figure 1: TTL voltage levels

3 Prelab

3.1 Digital Logic Gates

The flow of digital signals is controlled by transistors in various configurations depending on the logic family (we mentioned CMOS and TTL above). For most purposes, we can imagine that the logic gates are composed of several ideal switches with just two states: OPEN and CLOSED. The state of a switch is controlled by a digital signal. The switch remains closed so long as a logical (1) signal is applied. A logical (0) control signal keeps it open.

Logic signals interact by means of gates. The three fundamental gates, AND, OR, and NOT, are named after the three fundamental operations of logic that they carry out. The AND and OR gates each have two inputs and one output. The output state is determined by the states of the two inputs. The NOT gate has one input and one output.

The function of each gate is defined by a truth table, which specifies the output state for every possible combination of input states. The output values of the truth tables can be understood in terms of two switches. If the switches are in series, you get the AND function. Parallel switches perform the OR operation. The most common gates, their symbols, and Boolean notation are shown in Table 1. A small circle after a gate or at an input indicates negation (NOT). The truth tables for all gates are summarized in Table 2.

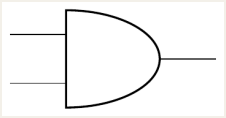
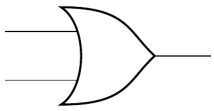
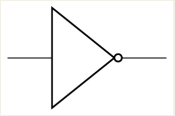
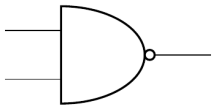
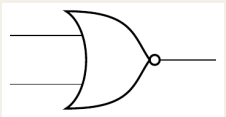
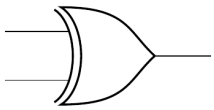
Gate	Boolean Notation	Symbol
AND	$A \cdot B$	
OR	$A + B$	
NOT	$\overline{A}$	
NAND	$\overline{A \cdot B}$	
NOR	$\overline{A + B}$	
XOR	$A \oplus B$	

Table 1: Logic gate symbols and Boolean notation.

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

Table 2: Truth tables for all logic gates. NOT depends only on A.

3.1.a Prelab Question

1. Read Section 6.2 of the lab thoroughly and enter in your lab notebook the circuit diagrams (symbols) and truth tables of all the circuits that you will test: NAND, NOR, NOT (INVERT), and XOR (EXCLUSIVE OR).
2. Read through the appendix. Design a circuit to perform the XOR function using only NAND gates or only NOR gates. Simplify the circuit so that you use the smallest possible number of gates.
3. Prove that your circuit is equivalent to the XOR using the truth tables or with Boolean algebra.

3.2 555 Timer and Digital Clock

Read Steck section 15.1 and 15.1.1.

Figure 2 shows the astable circuit for generating a clock with the 555 timer. The capacitor C charges through R_A and R_B in series and discharges through R_B alone, producing the timing relationships:

$$t_2 = (R_A + R_B)\, C \ln 2 \qquad (\text{output HIGH — capacitor charging}) \tag{1}$$

$$t_1 = R_B\, C \ln 2 \qquad (\text{output LOW — capacitor discharging}) \tag{2}$$

The period is $T = t_1 + t_2$ and the duty cycle is t_2/T. Component limits: $1 \ \text{k}\,\Omega \le R_A, R_B \le 3.3 \ \text{M}\,\Omega$ and $C \ge 500 \ \text{pF}$. The resulting waveforms are shown in Figure 3. Much more information is available in Steck section 15.1.2.

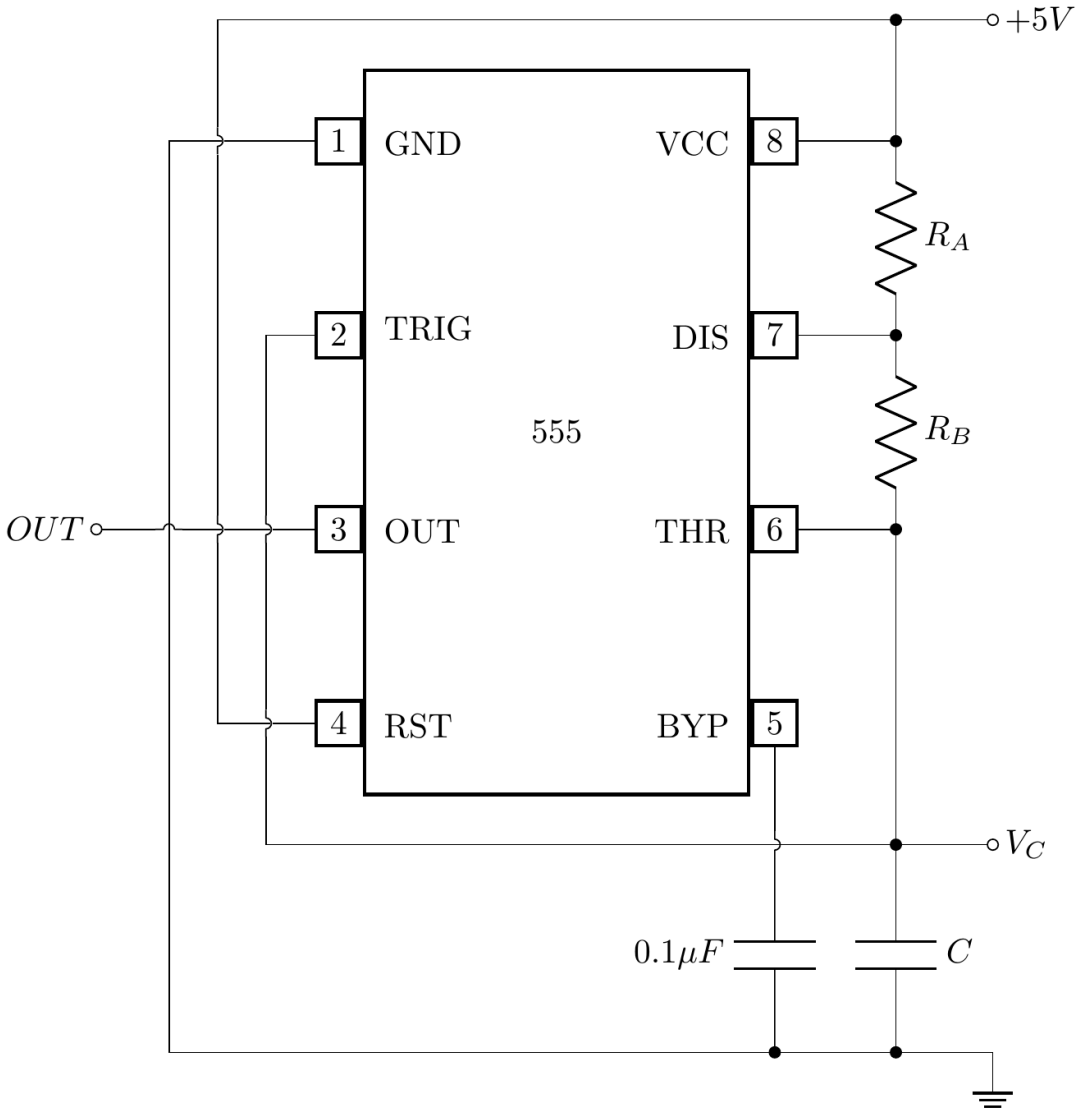


Figure 2: 555 timer astable (free-running) oscillator circuit for generating a digital clock signal.

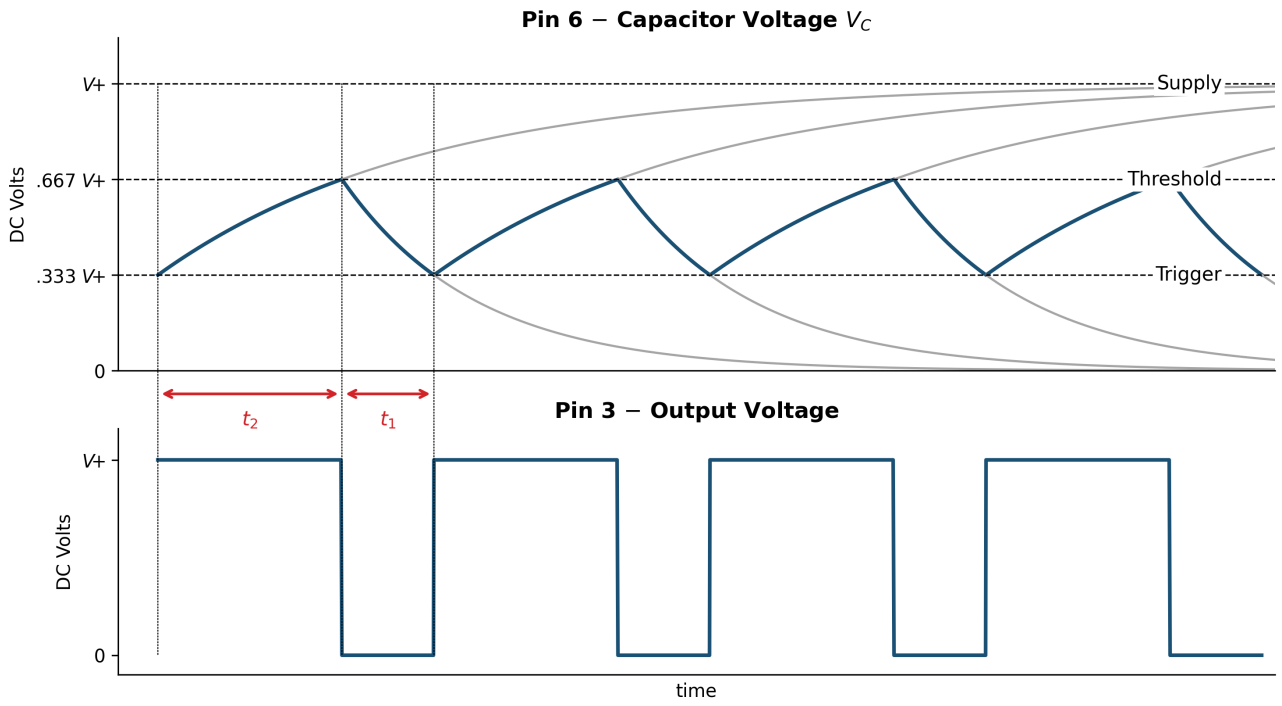


Figure 3: Capacitor voltage and output waveforms for the 555 timer astable circuit.

3.2.a Prelab Question

1. Design a 4 kHz clock using the 555-timer chip. Make the low level $1/4$ of the output period (a 75% duty cycle: 25% low, 75% high).
2. How large a capacitor would you need to add in parallel with your existing capacitor in order to modify your clock to run at 2 Hz (e.g. for visual observation of LEDs), keeping all other components fixed?

3.3 Memory Elements and Flip-Flops

In sequential logic circuits, the output depends upon previous values of the input signals as well as their present-time values. Such circuits necessarily include memory elements that store the logic values of the earlier signals. The fundamental memory circuit is the RS memory element. The JK flip-flop has an RS flip-flop at its core, but it adds circuitry that synchronizes output transitions to a clock signal. Timing control by a clock is essential to most complex sequential circuits.

3.3.a RS memory circuit

The truth table for the RS memory element shows how the circuit remembers. Suppose it starts in a state with $Q = 0$ and $R = S = 0$. A positive pulse S at the input sets it into the state $Q = 1$, where it remains after S returns to zero. A later pulse R on the other input resets the circuit to $Q = 0$, where it remains until the next S pulse.

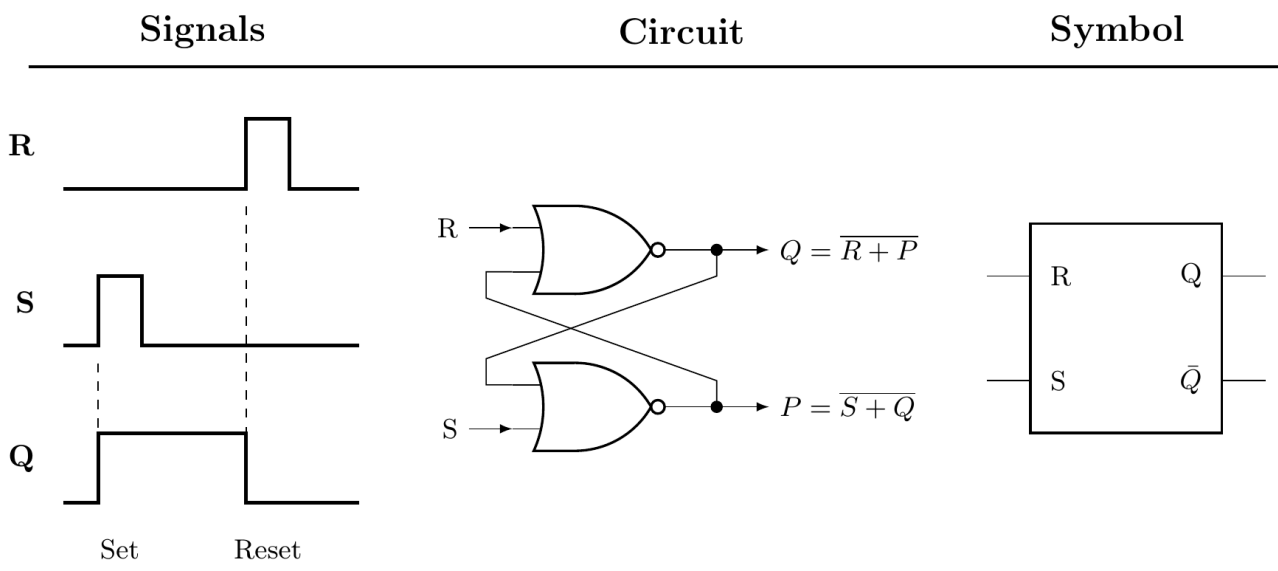


Figure 4: RS memory element: signal waveforms, cross-coupled NOR gate circuit, and block symbol.

R	S	Q	$P = \bar{Q}$
0	0	stays the same	
0	1	1	0
1	0	0	1
1	1	0	0

Table 3: RS memory truth table. $R = S = 1$ is disallowed because $P \neq \bar{Q}$.

3.3.b JK flip-flop (74107)

There are three kinds of inputs to the JK flip-flop:

1. Data inputs J and K
2. The clock CK
3. The direct input clear CLR

There are two outputs: Q and its complement $\overline{Q}$ (not Q).

The index n counts the number of clock pulses since the start of the experiment. In the absence of a clock pulse, the output remains unchanged at the previously acquired value, Q_n, which is independent of the present-time data inputs J and K. Only on the arrival of a clock pulse, CK, can the output change to a new value, Q_{n+1}. The value of Q_{n+1} depends on the J and K inputs in the way specified in Table 4. The change occurs at the falling (trailing) edge of the clock pulse, indicated by a downward arrow in Figure 5 and Table 4. The direct input, CLR, overrides the clock and data inputs. During normal operation, $\mathrm{CLR} = 1$. At the moment CLR goes to zero, the output goes to zero and remains there as long as $\mathrm{CLR} = 0$.

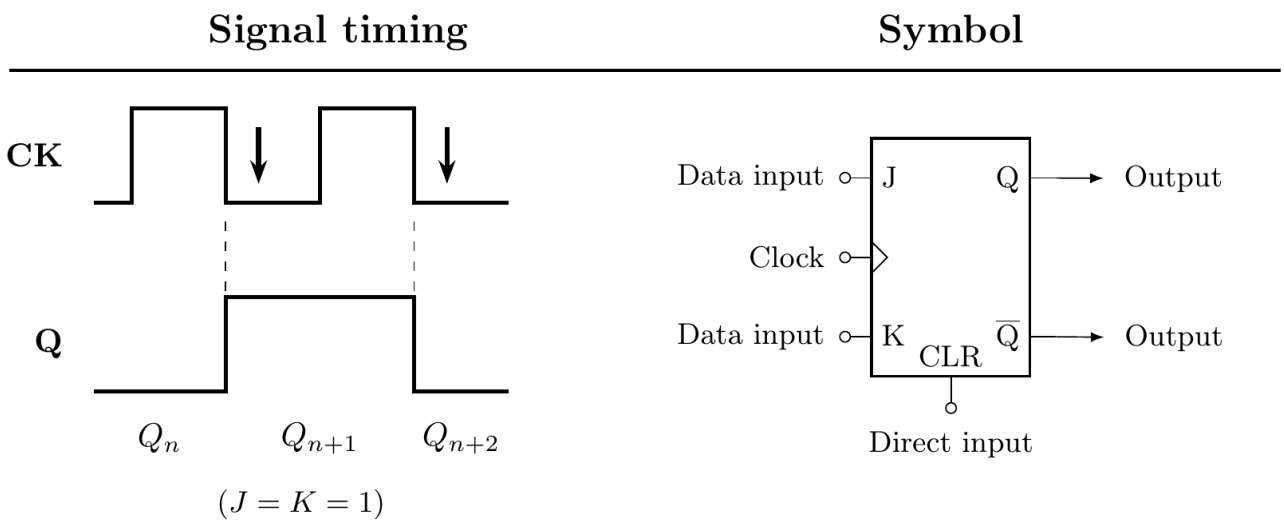


Figure 5: JK flip-flop: signal timing diagram and block symbol.

CLR	CK	J	K	Q_{n+1}	$\overline{Q}_{n+1}$
1	↓	0	0	Q_n	$\overline{Q}_n$
1	↓	1	0	1	0
1	↓	0	1	0	1
1	↓	1	1	$\overline{Q}_n$	Q_n
0	anything			0	1

Table 4: JK flip-flop truth table. $J = K = 0$: no change; $J = K = 1$: toggle; $\mathrm{CLR} = 0$: reset.

3.3.c Prelab Question

1. A JK flip-flop with $J = K = 1$ and $\mathrm{CLR}{=}1$ is driven at the clock input by the 4 kHz clock you designed in Section 3.2.a. Draw the waveforms for the clock and the Q output (labeling each) vs. time using the same time scale (making sure the times are indicated on the x-axis). Include enough periods of the clock signal to see the full behavior of the flip-flop's output.

3.4 Lab activities

3.4.a Prelab Question

Please review the lab activities so that you're better prepared when you arrive at your lab section.

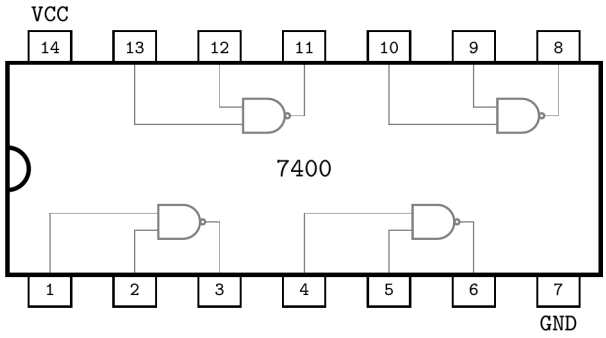
4 Useful Readings

You can find more on digital circuits in these recommended sources:

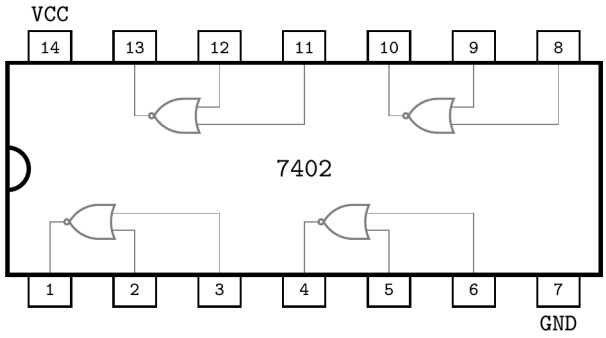
1. Steck: Sections 9.1, 9.2, 9.3, 10.1, 10.2, 11.4, 13.1, 13.2, 15.1.1, 15.1.2
2. Fischer-Cripps Chapter 11
3. Horowitz and Hill 2nd Ed. Chapter 8
4. Horowitz and Hill 3rd Ed. Chapter 10

5 Digital Logic Chip Pin-Outs

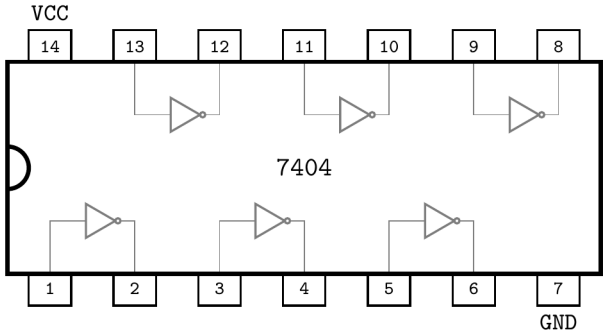
Each chip has a dot or notch to indicate the end where pins 1 and 14 are located. The pin numbers increase sequentially as you go counterclockwise around the chip viewed from above. In 74xx family logic chips, pin 7 is always grounded $(0\ \mathrm{V})$ and pin 14 is always connected to the $+5\ \mathrm{V}$ supply. You connect these to the breadboard the same way as the op-amp (across the groove in the middle of the breadboard). Decoupling capacitors for the power to these chips are not needed in steady state use of these chips, but it's typically a good idea to use them when dealing with switching on/off states rapidly.



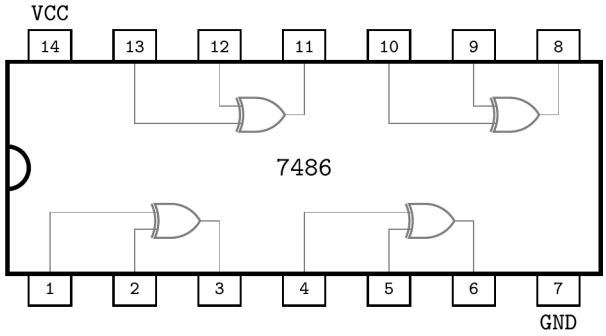
(a) 7400 Quad 2-Input NAND



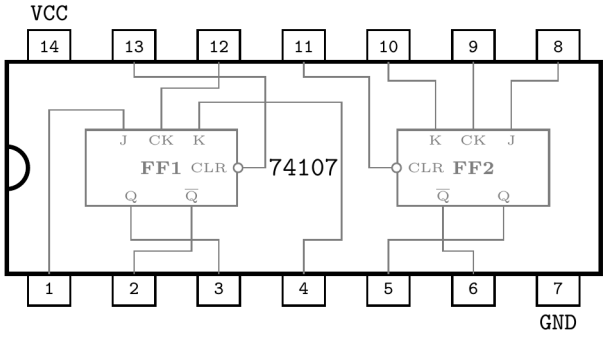
(b) 7402 Quad 2-Input NOR



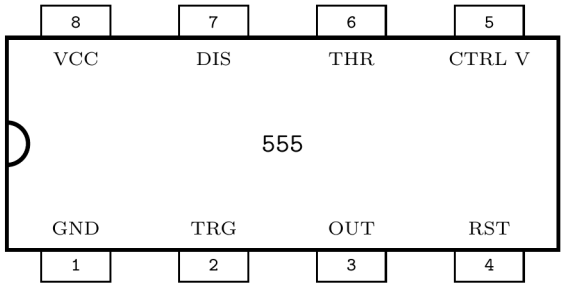
(c) 7404 Hex Inverter



(d) 7486 Quad 2-Input XOR



(e) 74107 Dual JK Flip-Flop



(f) 555 Timer (8-Pin DIP)

Figure 6: Logic chip pin-outs.

6 Lab Activities

6.1 Switches

Throughout this lab you will switch inputs between $0\ \mathrm{V}$ and $5\ \mathrm{V}$. This can be done by moving wires between two rails (one at ground and one at $5\ \mathrm{V}$); however, for the memory circuits, it can cause problems to leave an input floating for too long (the time it takes to move the wire). In these cases, it is crucial to use a switch to switch between these voltages.

1. The middle pin of a switch is connected to either of the other two of the pins depending on the position of the lever. Connect one outer pin of the switch to ground and the other to $5\ \mathrm{V}$.
2. Measure the middle pin with your DMM with respect to ground and confirm that the switch switches between the two desired voltages.

You can monitor the logic level of any state (input, output, or any intermediate state) using a light emitting diode (LED). When a state is (1), $5\ \mathrm{V}$ will be applied to one end, and the LED will light up, and when a state is (0), there won't be a voltage across the LED, so it won't light up. This is a nice way to visualize what is happening in your circuit without needing to endlessly probe around with your DMM. You will likely want quite a few LEDs for this purpose, so you may want to use at least one 10-LED bank (MV57164). It can be nice to use individual LEDs (HLMP-C625) to be able to place each one physically where you want in the circuit.

1. To limit the amount of current through each diode, place a resistor in series with it. What value of resistor should you use to limit the current to $20\ \mathrm{mA}$ (don't forget that the LED has a voltage drop of about $2\ \mathrm{V}$)? Record your calculation. Check that the LED lights up appropriately. If you don't see light, try reversing the $+5\ \mathrm{V}$ and ground connections. You can review Lab 6 for more information on LEDs.
2. Connect an LED to the output of each switch and toggle them to confirm you can visualize the state of each switch. These switches can be used as the inputs for all the circuits you will build moving forward. You should use other LEDs to visualize output states of your circuits as well.

6.2 TTL Gates

6.2.a Truth tables

1. Check your power supply before connecting to the circuit board. Often supplies have a fixed $5\ \mathrm{V}$ output for powering TTL digital circuits. However, our supplies do not. Choose a channel to set to $5\ \mathrm{V}$.
2. Input logical values can be set by connecting wires from the gate inputs to either $0\ \mathrm{V}$ (logical 0) or $+5\ \mathrm{V}$ (logical 1). Use one long rail on your prototyping board for $0\ \mathrm{V}$ and one for $+5\ \mathrm{V}$. **Note: Disconnecting an input from the $+5\ \mathrm{V}$ rail is not the same as connecting it to $0\ \mathrm{V}$. If it is disconnected, the input can float up to $+5\ \mathrm{V}$ on its own.**
3. Configure an LED to observe the output of the circuit you are testing.
4. Record the measured truth tables for the NAND (7400), NOR (7402), and INVERT (7404) gates, using the LED indicator for your measurements.

6.2.b Modifying basic gates

1. Connect a NAND gate so that it performs the INVERT function. Do this for a NOR gate also.
2. Record your circuit and measured truth table.

6.2.c Exclusive OR

1. Verify the truth tables for an XOR chip (7486).
2. Build and test the XOR circuit you designed using only NAND or only NOR gates in Section 3.1.a, item 2.

6.3 Sequential Logic

6.3.a RS memory circuit

1. Build an RS memory circuit from two NOR gates. Draw a schematic of your circuit.
2. Demonstrate the memory property by going through a complete memory cycle: Set $(R = 0, S = 1)$, Store $(0, 0)$, Reset $(1, 0)$, Store $(0, 0)$, Set $(0, 1)$. Record all inputs and outputs for each cycle. You can determine the output by using one LED for each output or measuring the voltage of each output. Do your results agree with your predictions?
3. Examine the effect of the "illegal" input $(R = 1, S = 1)$ for different initial states of the RS system. Describe the outcomes of the illegal operation.

6.3.b Digital clock with 555 timer

1. Build the $\sim 4 \ \mathrm{kHz}$ digital clock using a 555 Timer according to your design in Section 3.2.a, item 1. Measure the frequency, the pulse length (time the output is high), the duty cycle, and the nominal $5 \ \mathrm{V}$ amplitude. Include a screen shot showing the results. Do your measurements agree with your predictions using the measured values of your components?
2. Check that a suitable large capacitor placed in parallel with the existing one converts the clock to $2 \ \mathrm{Hz}$.

6.3.c JK flip-flop

1. Test a JK flip-flop by constructing a truth table utilizing different inputs for J and K and creating a clock transition by switching the voltage from $0 \ \mathrm{V}$ to $5 \ \mathrm{V}$ to $0 \ \mathrm{V}$. This is a perfect opportunity to use the switch on the front panel of your setup. Connect LEDs to both outputs to record your data. Since the output depends upon the previous state, Q_n, you will need to tabulate Q_{n+1} for both possible previous states, $Q_n = 0$ and $Q_n = 1$. You should add an additional column, Q_{n+2}, to get a better feel for the behavior of the flip-flop.
2. Set $\mathrm{CLR} = 1$ and $J = K = 1$. Now drive the clock input of the flip-flop with $4 \ \mathrm{kHz}$ pulses from your clock circuit. Use the oscilloscope to measure the clock input and the output, Q, of the flip-flop. You may find Figure 7 helpful to set up this part. Record your measurements and compare with your prediction from Section 3.3.c. Include a screen shot showing the results.
3. Do the same test with $J = K = 0$ and record what happens.

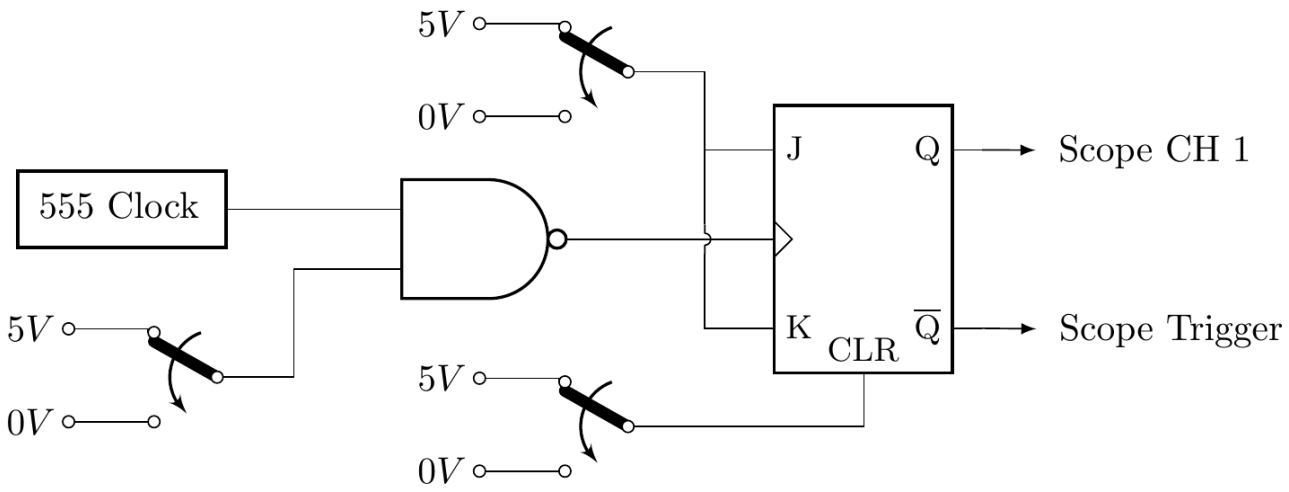


Figure 7: JK flip-flop test setup

7 Appendix: Boolean Algebra

7.1 Fundamental laws

We imagine a logical variable, A, that takes on the values 0 or 1. If $A = 0$ then $\overline{A} = 1$ and if $A = 1$ then $\overline{A} = 0$. Here are some obvious identities using the AND, OR and NOT operations. Looking at these identities you can see why the 'plus' $(+)$ symbol was chosen for OR and 'times' $(\cdot)$ for AND.

OR	AND	NOT
$A + 0 = A$	$A \cdot 0 = 0$	$A + \overline{A} = 1$
$A + 1 = 1$	$A \cdot 1 = A$	$A \cdot \overline{A} = 0$
$A + A = A$	$A \cdot A = A$	$\overline{\overline{A}} = A$
$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$	

Table 5: Fundamental laws of Boolean algebra.

7.2 Equality

Two Boolean expressions are equal if and only if their truth tables are identical.

7.3 Associative laws

$$(A + B) + C = A + (B + C) \tag{3}$$

$$(AB)C = A(BC) \tag{4}$$

7.4 Distributive laws

$$A(B + C) = AB + AC \tag{5}$$

Related identities:

$$(A + AB) = A \tag{6}$$

$$(A + \overline{A}B) = A + B \tag{7}$$

$$(A + B) \cdot (A + C) = (A + BC) \tag{8}$$

7.5 DeMorgan's theorems

$$\overline{A \cdot B \cdot ...} = \overline{A} + \overline{B} + ... \tag{9}$$

$$\overline{A + B + ...} = \overline{A} \cdot \overline{B} \cdot ... \tag{10}$$

7.6 Example proof

Each of the above equalities is a theorem that can be proved. Let's do an example by directly comparing the truth tables for the left and right sides. We take on DeMorgan's first theorem for two variables, $\overline{AB} = \overline{A} + \overline{B}$:

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

Table 6: Truth table for $\overline{AB}$.

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

Table 7: Truth table for $\overline{A} + \overline{B}$.

The last columns of the truth tables are identical. Thus, the first theorem is proven for two variables.

7.7 Example of simplification

Boolean algebra can be used to simplify logical expressions and reduce the number of gates required in a circuit. Consider the expression $Y = A + \overline{A}BC$. Figure 8 shows a direct implementation using NOT, NOR, and NAND gates.

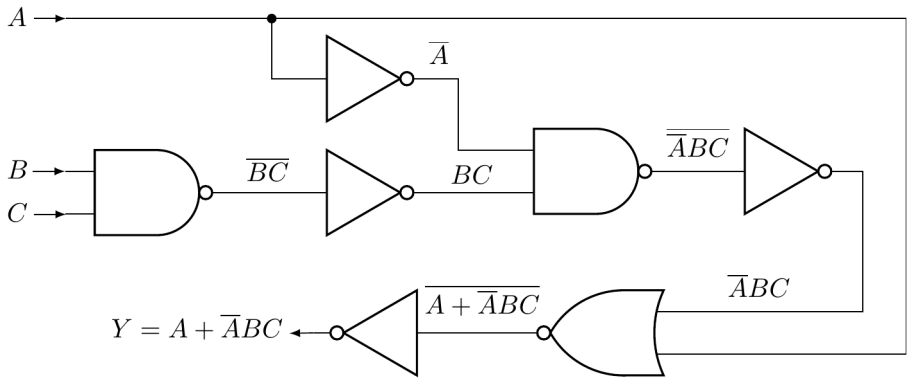


Figure 8: Direct implementation of $Y = A + \overline{A}BC$ using NOT, NOR, and NAND gates, with intermediate signals labeled at each stage.

We can simplify this expression using Boolean identities:

$$
\begin{aligned}
Y &= A + \overline{A}BC \\
&= A + BC \quad (\text{by identity} \quad A + \overline{A}B = A + B\ ) \\
&= \overline{\overline{A + BC}} \quad (\text{double negation}) \\
&= \overline{\overline{A} \cdot \overline{BC}} \quad (\text{De Morgan's Law})
\end{aligned}
$$

The final expression is a NAND of $\overline{A}$ and $\overline{BC}$, so the entire circuit can be built from just three NAND gates, as shown in Figure 9. Note that a NAND gate with its inputs tied together acts as a NOT gate ($\overline{A \cdot A} = \overline{A}$), so the first gate inverts A.

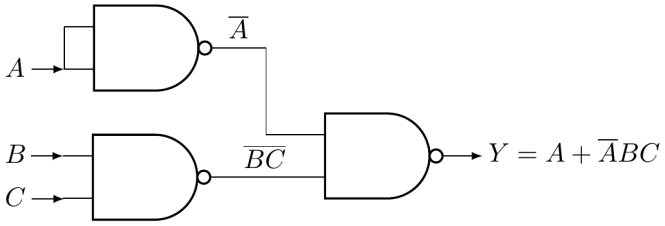


Figure 9: Simplified implementation of $Y = A + \overline{A}BC$ using three NAND gates. The first NAND gate has its inputs tied together, acting as a NOT gate to produce A-bar.