Lab 9

Digital Electronics: Logic and clocks

Department of Physics | University of Colorado Boulder

2022-12-07

Contents

1 Goals

In this lab, you will learn about the most basic elements of digital electronics, from which more complex circuits, including computers, can be constructed. Proficiency with new equipment and approaches:

- Logic gates, memory circuits, digital clocks
- Combining components & Boolean logic

2 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

Logic gates - a physical device that implements some Boolean logic operation

3 Digital circuits - General

In almost all experiments in the physical sciences, the signals that represent physical quantities start out as *analog* waveforms. To display and analyze the information contained in these signals, they most often are converted into *digital* data. Often this is done inside a commercial instrument such as an oscilloscope or lock- in amplifier, which is then connected to a computer through a digital interface. In other cases, data acquisition cards are added to a computer chassis, allowing analog signals to be input directly to the computer. 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. Almost all data are eventually analyzed digitally with a computer. We emphasize analog electronics in this course because scientists usually have to know much more about it to design and build their experiments.

Analog information can be translated into digital form by a device called an Analog-to-Digital Converter (A/D converter or ADC). A set of N bits has 2^N possible different values, as you might recall from Lab 5. If you try to represent an analog voltage by 7 bits, your minimum uncertainty will be about 1%, since there are 27 = 128 possible combinations of 7 bits. For higher accuracy you need more bits. The

corresponding device that converts digital data back into an analog waveform is a Digital-to-Analog Converter (D/A converter or DAC), which you built in Lab 5.

Logic gates alone 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. The flip-flops we have chosen are from the TTL (Transistor- Transistor Logic) family. 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.

4 Digital Logic States

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 V and 0 V, respectively, but any input voltage in the range of 2–5 V is interpreted as HIGH, and any input voltage in the range of 0–0.8 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.

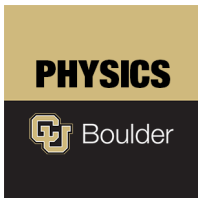

Figure 1: TTL voltage levels

5 Digital Logic Gates

The flow of digital signals is controlled by transistors in various configurations depending on the logic family. 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 are shown in Figure 2 and Figure 3. A small circle after a gate or at an input indicates negation (NOT).

Figure 2: Simple logic gates

Figure 3: Compound logic gates

6 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.

6.1 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.

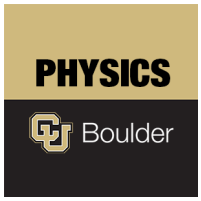

Figure 4: RS memory element

6.2 JK flip-flop (TTL74107)

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 it's complement (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, C, can the output change to a new value, Q_{n+1}. The value of Q_n depends on the J and K inputs in the way specified in the truth table. The change occurs at the falling (trailing) edge of the clock pulse, indicated by a downward arrow in the truth table in Figure 5. The direct input, CLR, overrides the clock and data inputs. During normal operation, CLR = 1. At the moment CLR goes to zero, the output goes to zero and remains there as long as CLR = 0.

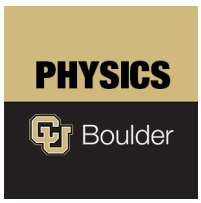

Figure 5: JK flip-flop

6.3 555 timer and digital clock

See Steck section 15.1.1 for a description of the guts of the 555 timer chip. Figure 6 shows the circuit for generating a clock with the 555 and summarizes the formulas relating the resistor and capacitor values to the output low time T1 and the output high time T2. Much more information is available in Steck section 15.1.2.

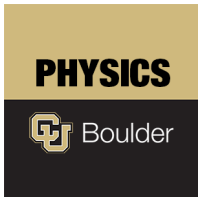

Figure 6: Summary of information for generating a clock with a 555 Timer

7 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 V) and pin 14 is always connected to the +5 V supply. You connect these to the breadboard the same way as the op-amp (across the groove in the middle of the breadboard).

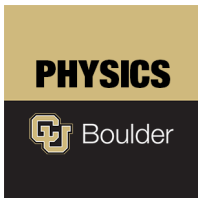

Figure 7: Some descriptive text

8 Useful Readings

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

9 Prelab

Answer the following questions using Mathematica or by hand.

9.1 Basic digital logic

1. Read the lab thoroughly and enter in your lab book the circuit diagrams (symbols) and truth tables of all the circuits that you will test: NAND, NOR, NOT (INVERT), and XOR (EXCLUSIVE OR).
2. 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 Boolean algebra method (see the Appendix).

9.2 555 timer

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 substitute in order to modify your clock to run at 2 Hz (e.g. for visual observation of LEDs), keeping all other components fixed?

9.3 JK flip-flop

1. A JK flip-flop with J=K=1 and CLR=1 is driven at the clock input by the 4 kHz clock you designed in section 9.2. 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.

9.4 Lab activities

1. Read through all of the lab steps and identify the step (or sub-step) that you think will be the most challenging.
2. List at least one question you have about the lab activity.

10 TTL Gates

10.1 Truth tables

1. Check your power supply before connecting to the circuit board. Some of the supplies have a fixed 5 V output that you should use to power the TTL digital circuits. Supplies with a varying output up to 6 V should be configured to output 5 V. The logic chips will burn out at around 6 V.
2. Input logical values can be set by connecting wires from the gate inputs to either 0 V (logical 0) or +5 V (logical 1). Use one long rail on your prototyping board for 0 V and one for +5 V. ***Note: Disconnecting an input from the +5 V rail is not the same as connecting it to 0 V. If it is disconnected, the input can float up to +5 V on its own.**
3. The logic level of the output can be observed using a light emitting diode (LED), which is connected from the output to ground. The LED lights up when the output is +5 V and is off when the output is 0 V. To limit the amount of current though the diode, place a resistor in series with it. What value of resistor should you use to limit the current to 20 mA (don't forget that the LED has a voltage drop of about 2 V)? Record your calculation. You can either use the single LED (HLMP-C625) or one of the LEDs from the 10-LED bank (MV57164). Check that the LED lights up appropriately. If you don't see light, try reversing the +5 V and ground connections. You can review Lab 6 for more information on LEDs.
4. Record the measured truth tables for the NAND (7400), NOR (7402), and INVERT (7404) gates, using the LED indicator for your measurements.

10.2 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.

10.3 Exclusice 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 9.1.2.

11 Sequential Logic

11.1 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.

11.2 Digital clock with 555 timer

1. Build the ~4 kHz digital clock using a 555 Timer according to your design in section 9.2.1. Measure the frequency, the pulse length (time the output is high), the duty cycle, and the nominal 5 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 Hz.

11.3 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 V to 5 V to 0 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 CLR = 1 and J = K = 1. Now drive the clock input of the flip-flop with 4 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 8 helpful to setup this part. Record your measurements and compare with your prediction from section 9.3. Include a screen shot showing the results.
3. Do the same test with J = K = 0 and record what happens.

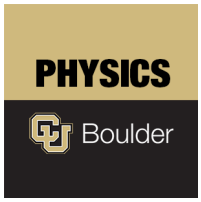

Figure 8: JK flip-flop test setup

12 Appendix: Boolean Algebra

12.1 Fundamental laws

We imagine a logical variable, A, that takes on the values 0 or 1. If $A = 0$ then $\bar{A} = 1$ and if $A = 1$ then $\bar{A} = 0$. 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' (•) for AND.

Table 1: Caption.

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

12.2 Equality

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

12.3 Associative laws

$$(A + B) + C = A + (B + C)$$

$$(AB)C = A(BC)$$

12.4 Distributive laws

$$A(B + C) = AB + AC$$

Related identities:

$$(A + AB) = A$$

$$(A + \bar{A}B) = A + B$$

$$(A + B) \cdot (A + C) = (A + BC)$$

12.5 DeMorgan's theorems

$$\overline{A \cdot B \cdot ...} = \bar{A} + \bar{B} + ...$$

$$\overline{A + B + ...} = \bar{A} \cdot \bar{B} \cdot ...$$

12.6 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, $\overline{AB} = \bar{A} + \bar{B}$:

Table 2: Caption.

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

Table 3: Caption.

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

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

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

12.7 Example of simplification

Boolean algebra can be used to simplify logical expressions and reduce the number of gates required in a circuit. In the figure below, we show two ways to implement the expression, $Y = A + \bar{A}BC$.

Direct implementation using NOT, NOR, and NAND

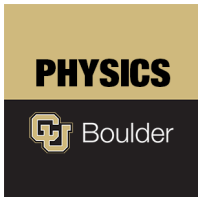

Figure 9: Logic

Simplified circuit