# Course Description

**Weekly Overview**

The focus this week takes a giant step forward in connecting formal logic with the machine of a computer. Students have now had truth tables, and they have also spent quite a bit of time learning that formal logic alone does not convey truth. They are ready to see how a “computing device” might be built. To do this, we use series and parallel circuits to model *and* and *or*, and we take the next step to logic gates. In all of this, we take our lead from W. Daniel Hillis in *The Pattern on the Stone*.

# Institutional Learning Outcomes

**Main Objectives**

* Understand that parallel and series circuits are physical models of *and* and *or*.
* Apply formal logic and circuits to solve a computational problem.
* Introduce iteration.

# Understand recursion, both mathematic

# Discipline Specific Outcomes

# Student Readings

*The Pattern on the Stone*, D. Hillis (Chapter 2)

**Daily Outline**

Day 1: Rock, Paper, Scissors Wiring Lab

Day 2: Rock, Paper, Scissors Wiring Lab (continued)

Day 3: Logic Gates

Day 4: Exploring Iteration

Day 5: Test

**Included Resources**

Teacher Lab Notes: Circuits and Rock, Paper, Scissors

Lab Assignment: Circuits and Rock, Paper, Scissors

Lecture Notes: Logic Gates

Homework Assignment: Logic Gates Exercises

Lab Assignment: Exploring Iteration

Iteration.java

**Teacher Lab Notes: Circuits and Rock Paper Scissors**

**Main Lecture: Circuits and Logic**

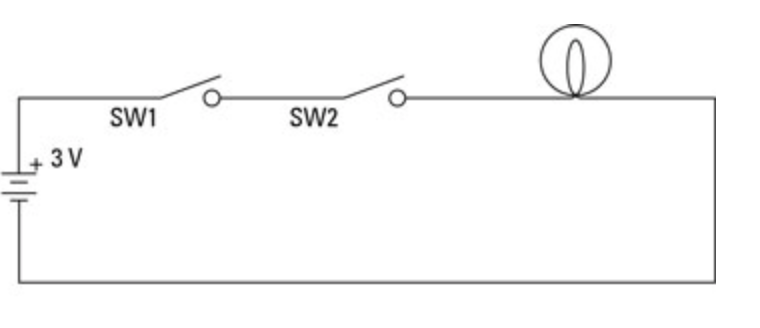
Before launching students into their work on the first part of this lab, you may need to give some instruction on parallel and series circuits. Some of this depends on how much they grasped the material from the Hillis reading the night before.

Note that we are not trying to give in depth notes on circuitry and electricity. The goal, actually, is simple:

1. We want to build a physical model for a logical system that consists of True and False.
2. We also want to model *and* and *or*. (We will ignore *not* for the time being.)
3. This physical model can be done using a variety of mechanisms, but we will be exploring electricity.

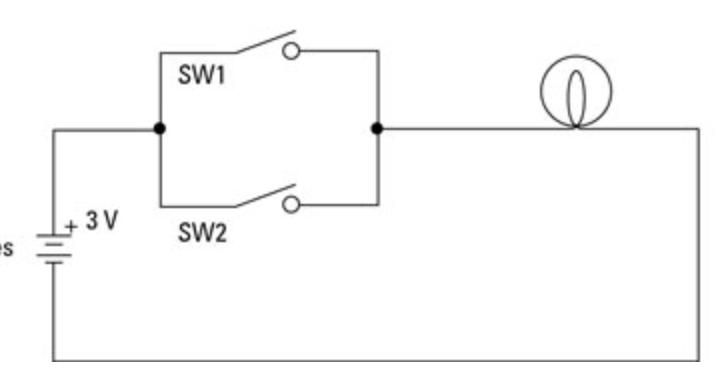
In order to model True and False using electricity, we will think of True as “on” or “current flowing”, and False as “off” or “current not flowing.”

The “and” is modeled using a *series circuit*.



The left side of the diagram is the battery, and the light bulb is, well, the thing that looks like a light bulb. The switches are labeled SW1 and SW2. “True” will be symbolize by a completed circuits with electricity flowing, which is indicate by the bulb being “on.” Notice that *both* switches need to be pushed in order to complete the circuit and turn on the bulb. “Both” is synonymous with “SW1 *and* SW2.”

Contrast this with a *parallel circuit*.



Notice in this diagram, if *either* of SW1 or SW2 (or both!) are down, the circuit is complete and the light bulb is on. This models the logical *or*.

**Lab Purpose**

Once you are comfortable that student grasp the way in which these two types of circuits model *and* and *or*, you can move on to explaining the lab. The lab sheet lays this out, so you can use that to talk students through it. We should note here, though, that the “tough” step in this wiring is using a single “button” to control multiple switches. This can be accomplishes by tying them on to the same rod (popsicle stick, etc.) so that when the rod is pushed down, it closes all the switches attached to it.

**Solutions**

(Note again that the circular button needs to close multiple switches at once. Students will need a hint on this.)

*Solution 1*

2

1

R2

R1

P2

P1

S1

S2

*Solution 2*

Note on this that I have included an extra switch in each set of Player 2 Buttons. These are not necessary, but can be used to implement “ties” with an extra light bulb (see “Extra Credit” on the assignment sheet).

R2

R1

1

P2

P1

2

S2

S1

**Homework (Day 1):** Complete the wiring diagram

**Homework (Day 2):** Read Hillis, Chapter 2

**Lab Assignment: Circuits and Rock Paper Scissors**

**Lab Purpose**

The goal is to model the game Rock, Paper, Scissors using parallel and series circuits.

1. There are two light bulbs that represented “Win” for the two players.
2. There is one battery.
3. There are six buttons (R1, P1, S2, R2, P2, and S2), three for each player representing the three moves (Rock, Paper, Scissors).
4. You may use more than one switch under a single button. In other words, the single press of the R1 button might close two separate switches simultaneously.

The light bulbs should light up as follows (unless noted, the bulb is “off”):

|  |  |  |  |
| --- | --- | --- | --- |
|  | **R1 Pushed** | **P1 Pushed** | **S1 Pushed** |
| **R2 Pushed** |  | Player 1  Light Bulb  On | Player 2  Light Bulb  On |
| **P2 Pushed** | Player 2  Light Bulb  On |  | Player 1  Light Bulb  On |
| **S2 Pushed** | Player 1  Light Bulb  On | Player 2  Light Bulb  On |  |

**Stage 1**

On the first day, you are responsible for working in pairs to produce a wiring diagram that correctly models the logic necessary to light the correct bulbs given the correct input. *You may assume that only legal situations are being played, e.g. Player 1 will not press both R1 and P1, nor will the player fail to press any button.*

For extra credit, include a third bulb that lights in the case of a “Tie”. For extra, extra credit, wire a fourth bulb for “Error” to deal with a player that presses more than one button.

**Stage 2**

You will be given the materials necessary to actual wire and test your circuit. You have until the end of the period to make it work.

**Lecture Notes: Logic Gates**

**Bell Work (5 minutes)**

Write a formal logical statement that represents the following circuit. The lower case letters represent switches.

*t*

*q*

*r*

*p*

**Answer:** *p* | [*r* & (*t* | *q*)]

**Main Lecture: (45 minutes)**

It is something of an oversimplification to say that the computer consists only of parallel and series circuits. What *is* accurate is to say that the computer presents a physical model of the logical system consisting of True, False, *and*, *or*, and *not*. We have seen from our Hillis readings that this can be accomplished using electricity, fluid, or even level-like systems. Electricity has the advantage of fast and not suffering nearly as much from signal distortion.

While the computer does utilize electricity to model Boolean logic, is more accurate to say that the computer consists of something we call *logic gates*. Fundamentally, there are three types of logic gates.[[1]](#footnote-1)

**AND NOT OR**

The operation of these gates is relatively self-explanatory. For the AND gate, the current comes out only if current is coming in to both wires. (Actually, the current in these gates is not really “ON” or “OFF”, but rather “HIGH” or “LOW”.) Notice that this is a bit different than the series circuit, which pays attention to switches being open and closed. Similarly, the OR gate has (high) current flowing out of it if one or the other (or both) of the inputs has (high) current flowing in. The NOT gate, known as an “inverter”, take high current and outputs low current, and takes low current and amplifies it to high current.

We will not get into the physics that designs these gates, but the construction can be accomplished in a variety of ways, including using transistors or something called diodes. Underneath the hood, however, the basic wiring in the gates for AND and OR utilizes series and parallel wiring concepts. That is why I said that the use of parallel and series circuits is “something of an oversimplification”, but I did not say that it is “inaccurate.”

The big missing piece from our logical puzzle is now to wire the concept of “NOT”. Notice that we did not talk about this during the Rock, Paper, Scissors lab, nor was it needed to implement the solution. It is, however, required for a full picture of a “physical/mechanical logical system.” In fact, it is *more* required than having both an AND and an OR gate. We have seen DeMorgan’s Laws, so we know that we could wire up an AND gate using OR and NOT gates:

***P* & *Q* = !(!*P* | !*Q*) *P* | *Q* = !(!*P* & !*Q*)**

However, the NOT gate *cannot* be made from a combination of AND and OR gates. This is another oversimplification, but the basic construction of a NOT gate takes advantage of a transistor being a sort of electrically-powered switch. If a high current is coming through, it flips the switch to “OFF”, preventing a main secondary line of power from flowing. If a low current is coming through, the switch does not flip and remains “ON”, allows the secondary line to flow. Therefore HIGH leads to a LOW output, and LOW leads to a HIGH output. An oversimplification? Definitely, but it is useful for describing to students how this sort of thing might be accomplished in circuitry.

*Note to teacher: It is useful to present something of the physical construction of these gates to students to “demystify” the black box, but they are not to be held responsible for the material.*

**Class Exercise**

1. Write a formal logical expression for the following circuit. (Note: There is no problem wiring the same *x* input to two different wires.)
2. Fill in a truth table for the circuit

*x*

*x*

*y*

*y*

How do we take a truth table and find a circuit that models it. The answer is to utilize something called *disjunctive normal form*. Let’s start with a truth table with three inputs. (*p* & !*q*) | !*r*

|  |  |  |  |
| --- | --- | --- | --- |
| ***p*** | ***q*** | ***r*** | **Output** |
| T | T | T | F |
| T | T | F | T |
| T | F | T | T |
| T | F | F | T |
| F | T | T | F |
| F | T | F | T |
| F | F | T | F |
| F | F | F | T |

The output is True (or “ON”, or “HIGH”, or even “1”) precisely five times (see the shaded rows):

1. *p* is True, *q* is True, and *r* is False
2. *p* is True, *q* is False, and *r* is True
3. *p* is True, *q* is False, and *r* is False
4. *p* is False, *q* is True, and *r* is False
5. *p* is False, *q* is False, and *r* is False

Note that each of these is essentially an AND statement:

1. *p* & *q* & !*r*
2. *p* & !*q* & *r*
3. *p* & !*q* & !*r*
4. !*p* & *q* & !*r*
5. !*p* & !*q* & !*r*

If any of 1, 2, or 3 are true, then the output is true, which is essentially an OR statement. Therefore, the logical statement representing this table is:

**Output = (***p* & *q* & !r) | (p & !*q* & *r*) | (*p* & !*q* & !*r*) | (!*p* & *q* & !*r*) | (!*p* & !*q* & !*r*)

**Class Exercise**

Draw the circuit with three inputs that represents this statement/table.

Note to the students that (1) this process leads to the form known as *disjunctive normal form*, but (2) this may not be the simplest form. In fact, electrical and computer engineers spend quite a bit of time trying to *minimize* the number of gates to produce the same circuit.

**Class Exercise**

Use a truth table to show that the expression (*p* & !*q*) | !*r*. Draw the circuit.

Even though it is not the simplest, the disjunctive normal form show that *any* truth table, and therefore *any* statement of formal logical, can be modeled mechanically. This is the beginning of *computation* and therefore the essence of a *computer*.

(We will see later that the missing piece in order to complete a full computational device is *memory*, or the ability to not only calculate an answer, but to store it and retrieve it. Once we have that, we have *algorithm*.)

**Homework:** Logic Gates Exercises

**Homework: Logic Gates**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Draw a circuit that represents the logical statements.

1. *p* | (!*q* & !*p*)
2. *x* & ![*y* | (*x* &*z*)]
3. *p* | ![*q* & !(*q* | !*p*)]
4. *x* | (*y* & *z* & !*x*) & !*z*
5. (!*p* & *q* & !*r*) | (*p* & *q* & *r*) | (*p* & !*q* & *r*) | (!*p* & !*q* & !*r*)

Write a logic statement and draw a circuit that represents the truth tables.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6. | ***p*** | ***q*** | **Output** |  |  |  |  | 8. | ***p*** | ***q*** | ***r*** | ***s*** | **Output** |
|  | T | T | F |  |  |  |  |  | T | T | T | T | F |
|  | T | F | T |  |  |  |  |  | T | T | T | F | F |
|  | F | T | T |  |  |  |  |  | T | T | F | T | T |
|  | F | F | T |  |  |  |  |  | T | T | F | F | F |
|  |  |  |  |  | |  |  |  | T | F | T | T | F |
|  |  |  |  |  |  | T | F | T | F | T |
| 7. | ***p*** | ***q*** | ***r*** | **Output** |  |  |  |  | T | F | F | T | F |
| T | T | T | T |  |  |  |  | T | F | F | F | F |
| T | T | F | F |  |  |  |  | F | T | T | T | T |
| T | F | T | T |  |  |  |  | F | T | T | F | F |
| T | F | F | F |  |  |  |  | F | T | F | T | F |
| F | T | T | F |  |  |  |  | F | T | F | F | F |
| F | T | F | T |  |  |  |  | F | F | T | T | F |
| F | F | T | F |  |  |  |  | F | F | T | F | F |
| F | F | F | F |  |  |  |  | F | F | F | T | F |
|  |  |  |  |  |  |  |  | F | F | F | F | T |

Write a logical expression that is represented by the circuit.

*p*

*q*

*q*

9. 10.

*q*

*p*

*p*

*r*

*r*

*q*

11.

*p*

*p*

*q*

*r*

**Lab Assignment: Iteration**

Blah.

//Iteration.java

1. There are actually more than these three, e.g. the *nand* gate or the *xor* gate, but these can be built from the basic three, and our logical system in this course focuses only on these three Boolean operators. [↑](#footnote-ref-1)