*Use pencil!*

# Class Introduction

**Website:** [**https://www.eecs.umich.edu/courses/eecs270/**](https://www.eecs.umich.edu/courses/eecs270/)
Piazza sign-up: <http://piazza.com/umich/fall2020/eecs270>
Gradescope entry code: 9PVPZE
Zoom lecture: <https://umich.zoom.us/j/96205373417> Passcode: EECS270
Zoom lab talk:

There are lots of other links (office hours Zoom link, office hours queues, etc.), but all of them, including the ones above, can be gotten from the home page.

## Covid--Labs

We are holding hybrid lab sections. In our case that means you can attend lab either in person or remotely as you wish. If you are in lab section 911 (pure remote) things will be a bit different. But in general how lab works is as follows:

**If come in person,** you will need to find an open seat in one of the two lab rooms, 2322 and 2331 EECS. You must wear a mask at all time and no food or drink is allowed. Cleaning supplies will be in each room and you should wipe down your station (keyboard, FPGA board, desk area near you).

The lab instructor will generally talk for a bit at the start of class. They will generally do this from 2322 EECS and broadcast that talk via Zoom (see link below) to the other room You can ask questions via Zoom if you are in the other room. These tend to be fairly short (2-10 minutes) and once done the Zoom link won’t be used again during that lab period.

There is an electronic office hours queue for the lab. If you have questions, you will add yourself to that help queue. When you do so, you should include where you are (which room and which station), and if this is your home lab (that is, the lab you are signed up for). Note: if it isn’t your home lab you have the lowest priority and may sit through the entire lab without any help. Once the lab instructor comes to help you, you ***must*** put on a face shield in addition to your mask. The lab instructor will have one on also.

**If you are remote:** there will be a Zoom link for the lab <https://umich.zoom.us/j/92396114763>. You can ask questions via Zoom during that talk (as can the people in the room). These tend to be fairly short (2-10 minutes) and once done the Zoom link won’t be used again during that lab period.

You will be using CAEN’s VNC and a website called labsland.com to do the labs remotely. See the webpage for more details.

There is an electronic office hours queue for the lab. If you have questions, you will add yourself to that help queue. When you do so, you should include a Zoom link where the lab instructor can join you, and if this is your home lab (that is, the lab you are signed up for). Note: if it isn’t your home lab you have the lowest priority and may sit through the entire lab without any help.

# Lecture start

Say we live in the rather black and white world where things (variables) are either true (**T**) or false (**F**). So if ***S*** is “Mark is going to the Store” and ***C*** is “Mark likes Computer games” then we’ll assume that each phrase is either true or false (as opposed only sort of liking computer games). We can then use connectives to combine the variables.

Mark is going to the store AND Mark likes computer games.

The above statement is only true if both phrases are true. Let that sentence be ***X***. We can now draw the “truth table” for ***X*** (we’ll use the other tables in a minute). When is X true?

|  |  |  |
| --- | --- | --- |
| S | C | X |
| F | F |  |
| F | T |  |
| T | F |  |
| T | T |  |

\_AND\_

|  |  |  |
| --- | --- | --- |
| S | C |  |
| F | F |  |
| F | T |  |
| T | F |  |
| T | T |  |

\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| S | C |  |
| F | F |  |
| F | T |  |
| T | F |  |
| T | T |  |

\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| C |  |
| F |  |
| T |  |

\_\_\_\_\_\_\_\_\_\_\_

What if the statement, ***Y***, were “Mark is going to the store OR Mark likes computer games”? When is that true? How about Mark does NOT like computer games?

(OR vs. XOR)

How about having ***B*** be “Bob’s house is brown”.

|  |  |  |  |
| --- | --- | --- | --- |
| S | C | B |  |
| F | F | F |  |
| F | F | T |  |
| F | T | F |  |
| F | T | T |  |
| T | F | F |  |
| T | F | T |  |
| T | T | F |  |
| T | T | T |  |

S AND C AND B

|  |  |  |  |
| --- | --- | --- | --- |
| S | C | B |  |
| F | F | F |  |
| F | F | T |  |
| F | T | F |  |
| F | T | T |  |
| T | F | F |  |
| T | F | T |  |
| T | T | F |  |
| T | T | T |  |

S OR C OR B

|  |  |  |  |
| --- | --- | --- | --- |
| S | C | B |  |
| F | F | F |  |
| F | F | T |  |
| F | T | F |  |
| F | T | T |  |
| T | F | F |  |
| T | F | T |  |
| T | T | F |  |
| T | T | T |  |

S OR C OR NOT B

|  |  |  |  |
| --- | --- | --- | --- |
| S | C | B |  |
| F | F | F |  |
| F | F | T |  |
| F | T | F |  |
| F | T | T |  |
| T | F | F |  |
| T | F | T |  |
| T | T | F |  |
| T | T | T |  |

S OR (C AND B)

# Representation of Boolean Logic (section 2.6)

Using AND, OR, NOT and XOR gets old. So symbols have been used to represent these notions for quite a while. We’ll hit three different representations today:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Math/Philosophy  | Electrical/Computer Engineering | Gate  |
|  Y AND Z |  |  |  |
|  Y OR Z  |  |  |  |
| NOT Y |  |  |  |
| Y XOR Z |  |  |  |

Order of operations?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In digital logic we usually think of “1” as being TRUE and “0” as being “FALSE”…

## Methods of representation and terminology (p53-54, 64-68))

We can represent logic functions in a number of ways at this point. We can write logic equations, draw gates, and write truth tables. Going between these methods of representation (mostly to/from truth tables) can be interesting. Consider the following truth table. Write a logical formula that is equivalent.

|  |  |  |  |
| --- | --- | --- | --- |
| S | C | B | X |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Terms: (p54-55,69)

* Variable
* Literal
* Product term
* Sum-of-Products
* Minterm
* Canonical Sum-of-Products

Now write a truth table for the following word problem:

Consider a device with three inputs: A, B and S as well as one output M. M should be equal to A if S=0, else M should be equal to B.

Now, without looking at the Truth table, can you draw gates for this? Hint: it can be done with 4 gates (2 AND, 1 OR, 1 NOT)

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | S |  |
| 0 | 0 | 0 |  |
| 0 | 0 | 1 |  |
| 0 | 1 | 0 |  |
| 0 | 1 | 1 |  |
| 1 | 0 | 0 |  |
| 1 | 0 | 1 |  |
| 1 | 1 | 0 |  |
| 1 | 1 | 1 |  |

This device is called a multiplexor (MUX), you will see it in lab 1!

# Manipulation of logic (2.5, The Axioms of Logic handout on webpage)

There are lots of logic properties, and most have a dual. You can prove these in a number of ways, but the easiest is to show that the truth tables of both are equivalent. This is informally called “proof by truth table” and formally called “perfect induction”

|  |  |  |
| --- | --- | --- |
| **Property** |  |  |
| Commutative | a+b = b+a | a\*b=b\*a |
| Associative | (a+b)+c = a+(b+c) |  |
| Combining\* | a\*b+a\*b’=a |  |
| DeMorgan’s | a’\*b’=(a+b)’ |  |
| Distributive | a\*b+a\*c=a\*(b+c) |  |
|  |  |  |

\*This one isn’t in our book AFAIK.

Can we use Combining to go from Canonical SoP to our other solution for the MUX?

There is a lot more in this section, you’ll need to read it.

# And now for something completely different… (maybe)Binary (and Hex) numbers (1.2)

Consider the number 123

1 2 3

Each *place* has a value. We normally work in base 10, so each place is 10 times bigger than the last.

In binary we work in base 2. Consider the number 100102 (the subscript indicates the base).

1 0 0 1 0

Now what do you suppose the value of 10.112 is?

1 0 . 1 1

Let’s convert 21 into base 2.

*(Time allowing we’ll cover this in class. You need to know it in any case.)*

Now consider base 16. We need symbols for 0-15 but only have them for 0-9 in decimal. So we’ll use A=10, B=11, C=12, D=13, E=14, F=15.

What is 1F16 in decimal? What is 44 in hexadecimal (base 16)?

Converting between base 2 and base 16 is easy. Just group the binary digits into groups of 4 starting at the decimal point. So what is 100110012 in hex? We commonly use hex to represent large numbers which we’d prefer to use binary for just to make it more readable…

# And the payoff

Why is binary

# And the payoff…

Why did we do binary numbers on the first day? How are they relevant to digital logic?

The point is that we can represent binary numbers using basic logic. Consider a device that adds two one-digit binary numbers and outputs a 2 digit binary number. Let the inputs be A and B and the output be R[1:0].

(The R[1:0] is a way two write that there are two outputs, R1 and R0. We might also write R[7:0] to indicate that there are 8 outputs: R7, R6, R5, R4, R3, R2, R1, and R0.)

 A

+ B

====

 R1R0

Write the truth table for this adder. R1 is to be the most significant digit (farthest to the left in the 2’s place in this example) while R0 is to be the least significant digit (farthest to the right, in the 1’s place). Then draw the logic gates.

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | R1 | R0 |
| 0 | 0 |  |  |
| 0 | 1 |  |  |
| 1 | 0 |  |  |
| 1 | 1 |  |  |

The point is that we can do arithmetic using basic logic. You may say “great, I can add two one-bit numbers”. But it turns out we can use this basic idea of using logic states to represent numbers to do all kinds of math. A modern computer can easily do 5-10 billion additions of 64-bit numbers in a second! All based on this basic idea.

Consider adding two 3-digit binary numbers.

* If we used the truth table scheme, how many rows would there be?
	+ What about for a 64-bit addition?
* Can we break the problem down in a way that works better?
	+ How?
* Let’s work on it…