Georgia Tech College of Engineering
School of Electrical and Computer Engineering

ECE 2020: Karnaugh Maps and CMOS review

Instructor: Samuel Talkington

September 10, 2024

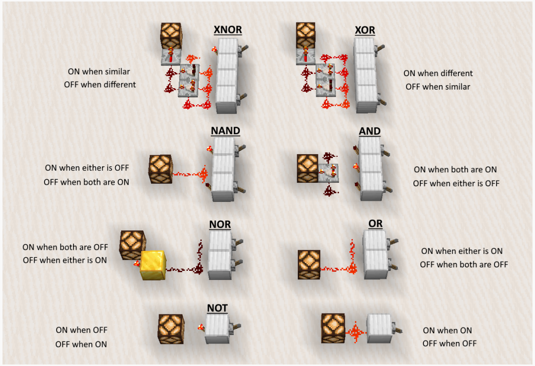
Logistics

- Now available:
    - **HW1: Grades and solutions released.**
    - HW 2—due September 17th **with optional 2 day extension** to Thursday, September 19th.
    - Updated course schedule—more on this later.
    - Pre-lab worksheet—due September 12th before class.
- Schedule for the next two weeks:
    - First lab next Thursday, September 12th, in-class.
    - First exam, September 19th, in-class.

Agenda

Today's agenda

- HW1 review ✓
- Puzzle review ✓
- CMOS review ✓
- Karnaugh maps ✓
- SOP form ✓



Source: r/minecraft

Agenda

- Today, September 10:
  - HW 1 review
  - Course schedule update
  - CMOS logic design review
  - Karnaugh maps and SoP forms
- Thursday, September 12: **Lab**—due Fri. Sep 20th, 11:59pm.
- Tuesday, September 17:
  - (Time permitting): Timing Diagrams (approx. half lecture) ✓
  - Exam review (half lecture, and after class).

Exam schedule

1. Exam 1: Thursday, **September 19, 2024**: 60 Minutes.
2. Exam 2: Thursday, October 10, 2024: 60 Minutes.
3. Exam 3: Thursday, November 07, 2024: 60 Minutes.
4. **Optional redemption exam:** Tuesday, November 26, 2024: 60 Minutes.

Problems will have short solutions. The exam will be designed for 60 minutes.

What to expect on the first exam

The exam will take 60 minutes and will be similar to homework assignments, but more brief and conceptual. There will be 3-4 problems

1. Boolean algebra and logic function simplification
2. Logic gates, operations, and circuits.
3. CMOS circuits

A single page of notes is permitted.

60%-70% HW1 material, 30%-40% HW2 material.

Survey feedback: Redemption

Confession: I sucked (still suck) at exams throughout my engineering education. Based on your survey feedback, we're going to try something new this semester:

- **Fourth exam** is an optional redemption exam.
- **Lowest homework** dropped at end of the semester.

HW1 review

HW1 Problem 1

Smart Thermostat

$X, M \in \{0,1\}$

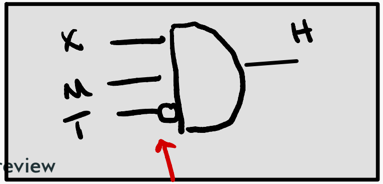
$$T = \begin{cases} 1 & \text{if temp} \geq \text{comfort} \\ \boxed{0 \quad \text{o.w.}} \end{cases}$$

$\overline{T} = 1 \Leftrightarrow \text{temp} < \text{comf.}$

$$H = \begin{cases} 1 & \text{if the house is heating} \\ 0 & \text{o.w.} \end{cases}$$

$\Rightarrow \boxed{H = X \cdot M \cdot \overline{T}}$

house power



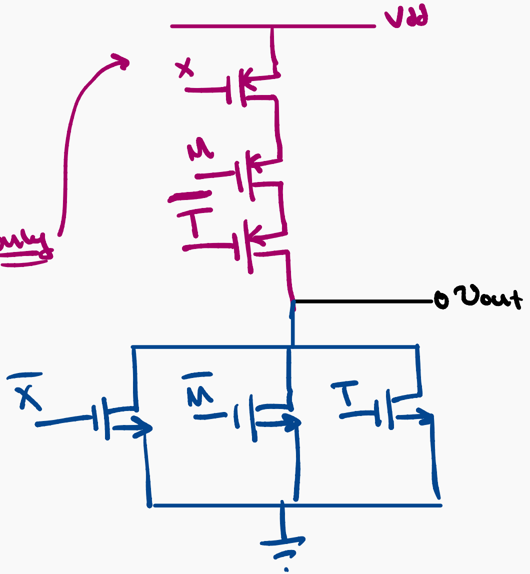
Example: How to make HW1 Problem 1 in real life with CMOS

$H = XM\overline{T}$ . . . pull up logic

$\overline{H} = \overline{XM\overline{T}} = \underline{\underline{\overline{X} + \overline{M} + T}}$

Pull up: Logic for H=1 PMOS only

Pull down: Logic for H=0.



HW1 Problem 2

$$F_1 = AB + C(A+B) + CA + B$$

$AB+B$

Absorption

$$AB + B = \underbrace{(A+C)}_{=1} \cdot B$$

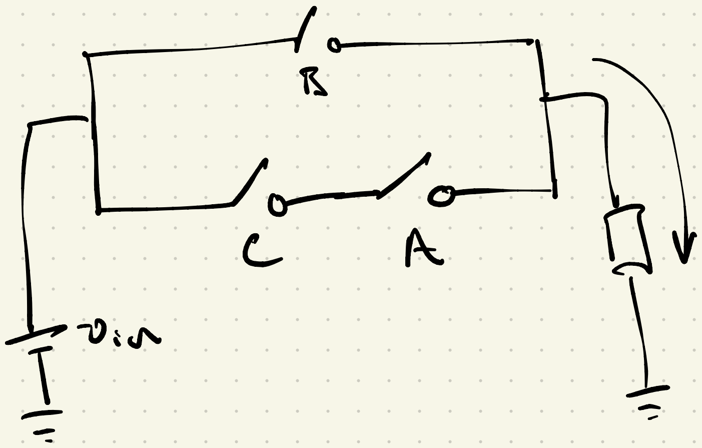
$$= \underline{B}$$

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

$$= B + CA + CB + CA$$

$$= B + \underbrace{CA + CA}$$

$$= B + CA$$


B
C
A
vin

HW1 Problem 3

CMOS Review

Implementing a NOR gate

Suppose you want to implement the NOR gate in CMOS. How do we do this?

CMOS: NOR

Implementing a generic logic function

Implement the following logic function in CMOS:

$$F = A \cdot \overline{(\overline{B} \cdot C)} \checkmark$$

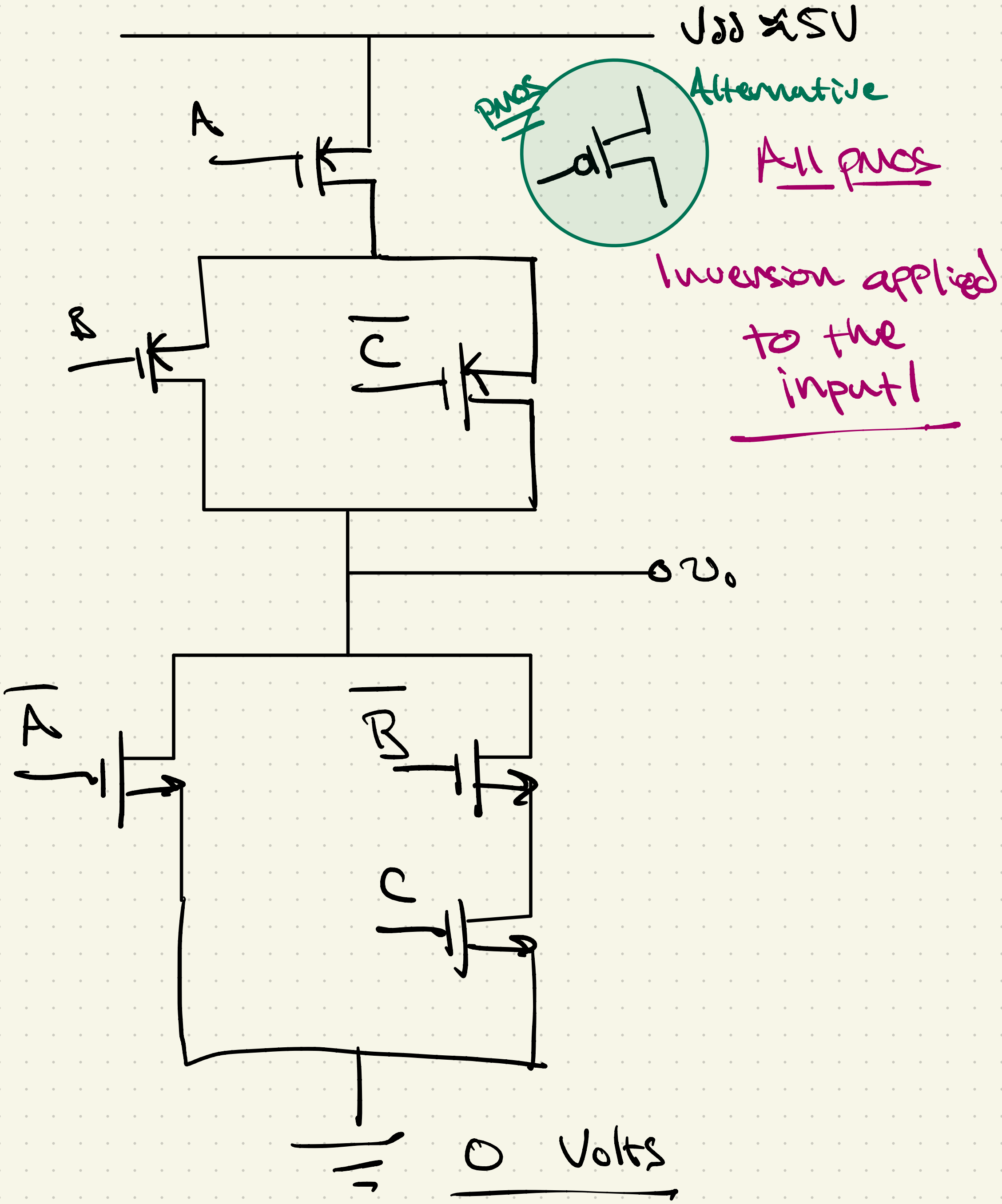
CMOS: Generic logic

Step1 PMOS:

$$F = A \cdot (B + \overline{C})$$

Step2: NMOS (pulldown)

$$\overline{F} = \overline{A \cdot \overline{(\overline{B} \cdot C)}} = \overline{A} + (\overline{B} \cdot C)$$



All PMOS

Inversion applied to the input!

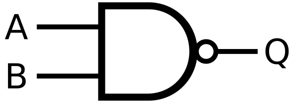
Participation puzzle: implement a CMOS NAND gate

Participation puzzle

Consider our friend, the NAND gate.

- Derive pull-up and pull-down expressions for a NAND gate.
- Construct a CMOS schematic for a NAND gate.
- Indicate the locations of the pull-up and pull-down networks.

Logic gate and truth table

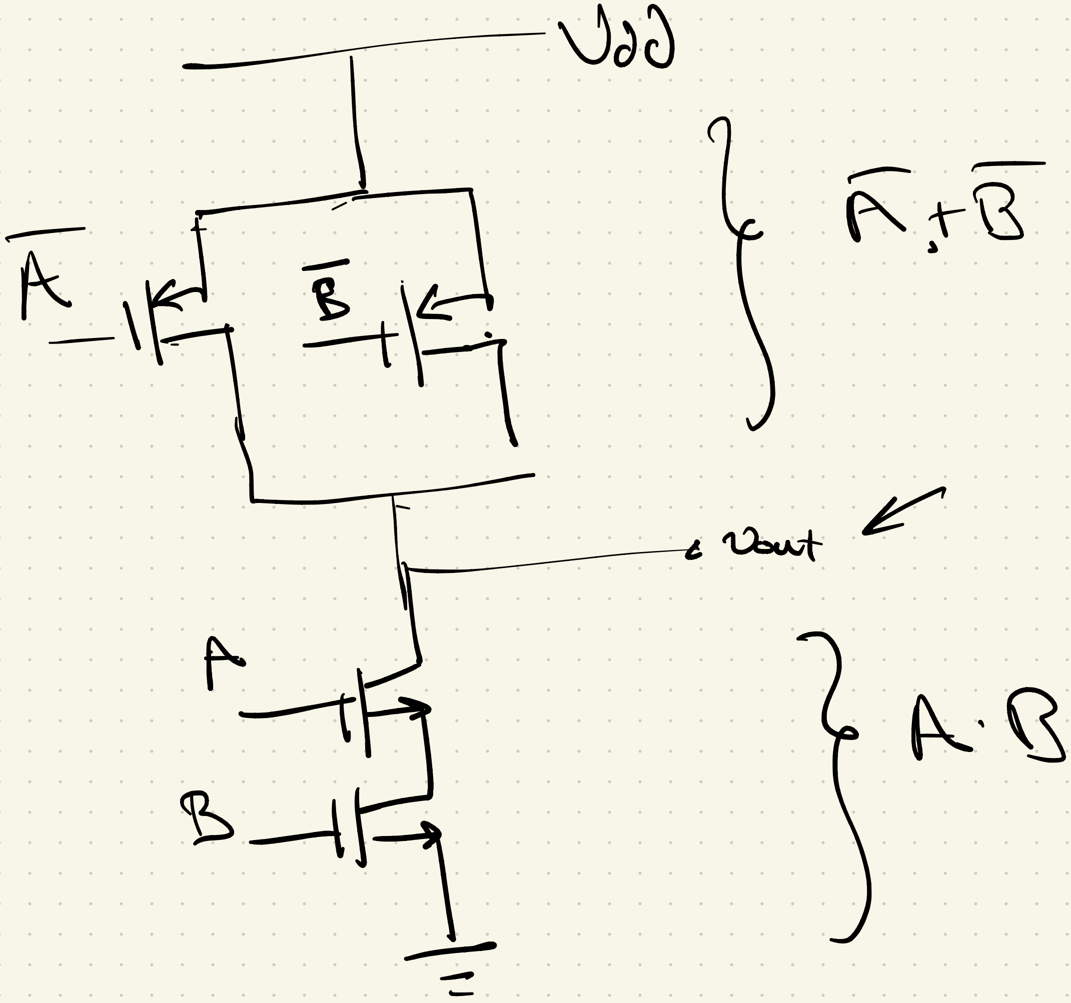


The NAND gate

A	B	$Q = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

$Q = \overline{A \cdot B} = \overline{A} + \overline{B}$

$\overline{Q} = A \cdot B$



Karnaugh maps

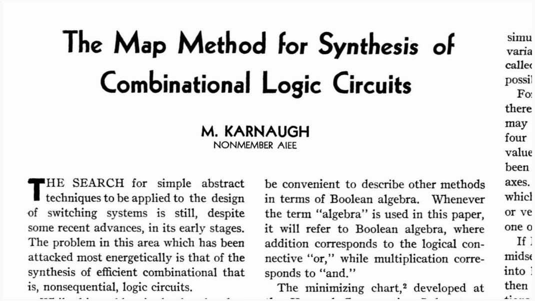
Kmap history

You, c. 1950s:

“I spent all day doing Boolean Algebra trying to make my computer with the least amount of gates! This sucks!”

Karnaugh:

“yikes, you can just circle these 1s instead dude...”



The Map Method for Synthesis of Combinational Logic Circuits

M. KARNAUGH
NONMEMBER AIEE

THE SEARCH for simple abstract techniques to be applied to the design of switching systems is still, despite some recent advances, in its early stages. The problem in this area which has been attacked most energetically is that of the synthesis of efficient combinational that is, nonsequential, logic circuits.

be convenient to describe other methods in terms of Boolean algebra. Whenever the term “algebra” is used in this paper, it will refer to Boolean algebra, where addition corresponds to the logical connective “or,” while multiplication corresponds to “and.”

The minimizing chart,[2] developed at

simu
varia
callec
possil
Fo
there
may
four
value
been
axes.
whicl
or ve
one o
If
midse
into
then

Figure 1: A genius idea

What is a Karnaugh Map? Motivation.

K-Maps: What?

- **Resource efficiency:** Minimize the number of gates/transistors to implement a logic function (#1 most important reason, but mostly computers do this now)
- **“Seeing” Boolean algebra:** Graphical depiction of boolean algebra minimization.
- **Some folks are visual:** You can use K-maps to do tons of algebra really fast.
- **They’re cool:** You can use geometric pattern matching to do Boolean algebra.

Graphical illustration of Karnaugh Maps

How do K-maps work?

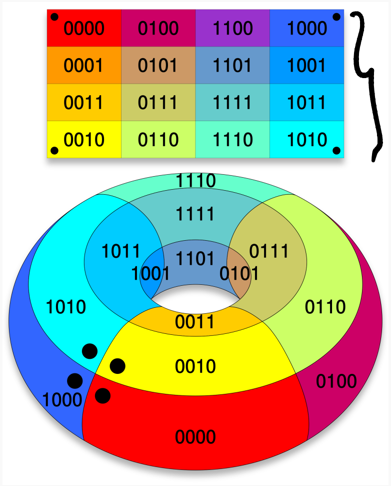
Basically, you just twist the truth table into a hyperdimensional donut...

⋮

what..?—**Yes... really.**

⋮

I know, it's wild!



Source: Wikipedia

The truth... and why we're still covering this

- K-maps were once essential for making efficient computers.
- In the past few decades, they have largely been replaced by computers themselves. (Check out the research area of *Automated Theorem proving*).
- We will only be covering the basics in this course.

What?

I can't with this donut thing, let's do some examples

Example from the ground up

Example from the ground up

Suppose you want to build a *digital system*. Suppose that you figure out how to express the thing that you want your system to **do in words** like this:

$$\longrightarrow F = \left\{\begin{array}{ll} 1 & \text{at least two inputs are false} \\ 0 & \text{otherwise} \end{array}\right\}$$

You all now have the tools to design the entire system! Steps:

1 Find the simplest way to represent F using a K-map

2 Build a gate-level schematic for your design

3 Build a CMOS level schematic for your design

Big-picture example of Exam 1 material i

idea

Truth table

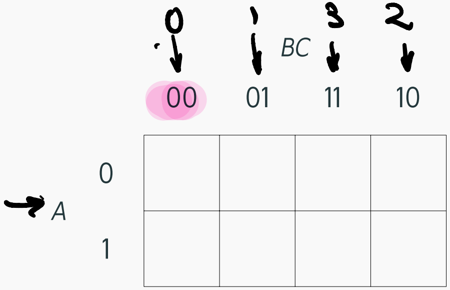
$$F = \begin{cases} 1 & \text{at least two inputs are false} \\ 0 & \text{otherwise} \end{cases}$$

Step 1: construct truth table using word logic

$\overbrace{\hphantom{A\quad B\quad C}}^{3}$

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

Big-picture example of Exam 1 material ii



$$N_{squares} = 2^{\# inputs} = 2^3 = 8$$

Step 2: Create an empty K-map to populate with your truth table.

K-maps

- It is up to your taste which variables to put on which side.
- Split your variables split to form a $2^{\# \text{inputs}}$ rectangle.
- The ordering of the binaries is called the "Gray code"

Big-picture example of Exam 1 material iii

The Gray Code

The **Gray Code** for ordering binary numbers refers to ordering the binary numbers such that two values differ by only **one bit.**

Examples:

(2 bits) 00, 01, 11, 10, ...

(3 bits) $\underline{000, 001, 011, 010, 110, 111, 101, 100,}$ ...

We use the Gray Code binary number ordering for K-Maps.

Big-picture example of Exam 1 material iv

- **Question:** Why Gray Code?
- **Answer:** Because we only have to physically change **one switch at a time** to move through Gray Coded binary numbers, and this maintains the donut structure.
- **Foreshadowing:** What are other reasons it can be useful to have only one switch move at a time? What if there are *switch delays over time*?

Big-picture example of Exam 1 material ∨

A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	0	0

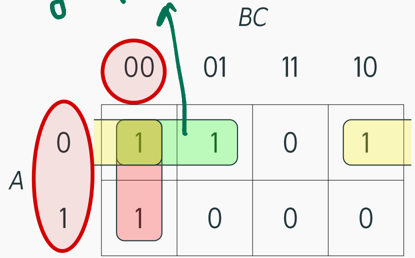
010

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

Step 3: Populate your K-Map using Gray Coding.

Big-picture example of Exam 1 material vi

each group represents a product term $\{m_i\}$ "minterms"



	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$$F = \underbrace{\text{group1}} + \underbrace{\text{group2}} + \cdots$$

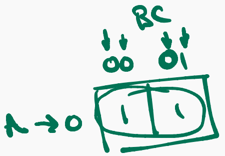
Step 4: Find the simplest form by grouping graphically:

$$\boxed{F = \overline{A}\cdot\overline{B} + \overline{A}\cdot\overline{C} + \overline{B}\cdot\overline{C}.}$$

K-maps

- It is up to your taste which variables to put on which side.
- Note the **"donut action"** creating the yellow group between m_0, m_2 squares across a 3rd dimension.
- **Warning:** You can only create 1s groups that sizes of powers of 2, e.g.: $1, 2, 4, \ldots$

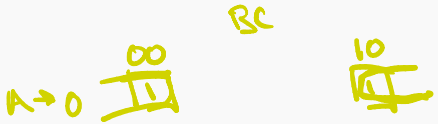
Note on "Don't-Cares"



"Don't Care"

↳ C is not included in the term
↳ b/c $C \in \{0,1\}$

$m_1 = \overline{A} \cdot \overline{B}$



B = don't care, $B \in \{0,1\}$

$A, C \in \{0\}$

$m_2 = \overline{A} \cdot \overline{C}$ →

Gate level schematic

Gate-level schematic:

CMOS schematic

CMOS schematic:

Suggested exercise

Example 2 Problem statement

A **half adder** circuit accepts 2 binary numbers $A, B \in \{0, 1\}$ and has 2 outputs:

- The sum of A and B, which we represent by the operation $S = A \oplus B$
- The carry of the sum, $C = A \cdot B$.

Truth table:

A	B	$S = A \oplus B$	$C = A \cdot B$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 1: The half adder

Suggested bonus

Suggested bonus exercise

A **full adder** circuit accepts 2 binary numbers $A, B \in \{0,1\}$, and a “carry in” term $C_{\text{in}} \in \{0,1\}$ that can come from another sum. The full adder returns the sum of A and B, and the *carry out*, C_{out}. The truth table is on the following slide.

Questions:

- Derive the simplest possible forms of the full adder sum signal S, and carry signal C_{out} using a 2×4 K-map for each.
- Draw a gate-level schematic for the full adder.

Full adder truth table

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1