CSC258 Winter 2016
Lecture 2

Lab 1 prep

- You should be registered in a lab section by now; if not, get in now, there is a new section PRA0105 added.
- Lab 1 is about familiarize with everything
  - Create a circuit using Quartus
  - Simulate the circuit using Quartus
  - Burn the circuit into the FPGA board, and test the hardware for real.
- Work in groups of 2 students.

Lab tips

- Read the "Summary of TODOs" part of the handout to quickly know what to do.

6 Summary of TODOs

Below is the summary of the steps to be completed for this lab:

1. Before the lab, read through the Quartus tutorial and/or install the tools on your own computer.
2. Find a partner in the lab to work together.
3. Predict the behaviour of the mystery circuit before implementing it.
4. Implement the mystery circuit using Quartus.
5. Simulate the circuit, explain the waveforms and show them to your TA.
6. Load your the circuit to the DE-2 board, make sure it's working as expected, and show it to your TA.

Evaluation (4 marks in total): 2 marks for attending and making an honest effort; 1 mark for showing and explaining the simulation waveforms; 1 mark for having the circuit working on the DE-2 board.

Lab tips

- Make sure to finish the work that needs to be done **before** the lab.
- Be ready to explain your work to the TA in order to get full mark.
- If your partner can answer but you can't, your partner gets more marks than you.
- Teach your partner! That makes you learn better, too!

Lab 1 tips

- Read the Quartus Tutorial (posted on course website) patiently.
- Follow the tutorial and you'll mostly be mostly fine.
- The DE2 pin assignment posted on course website is useful for loading to DE2 boards, you'll use in future labs over and over again.
- Sometime Quartus crashes unreasonably, just restart it.

More lab tips

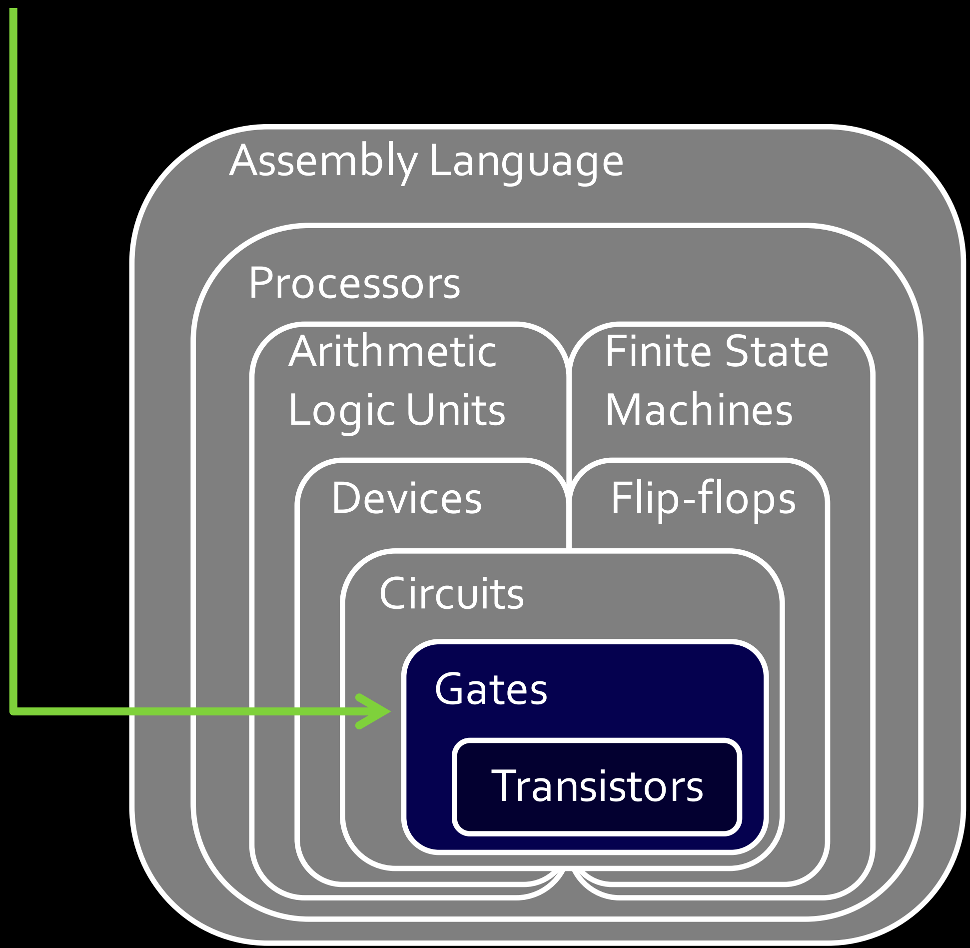
- Back up your work (copy to USB, upload to Dropbox, etc), you may need it for future labs.

- Send feedback using the Weekly Feedback Form after the lab.
  - http://goo.gl/forms/o248ETWgnS
  - Or just drop by my office to chat about how it went, while eating candies.

More lab tips

- Consider switching to the new lab section PRA0105, where you can get more help from the TA.

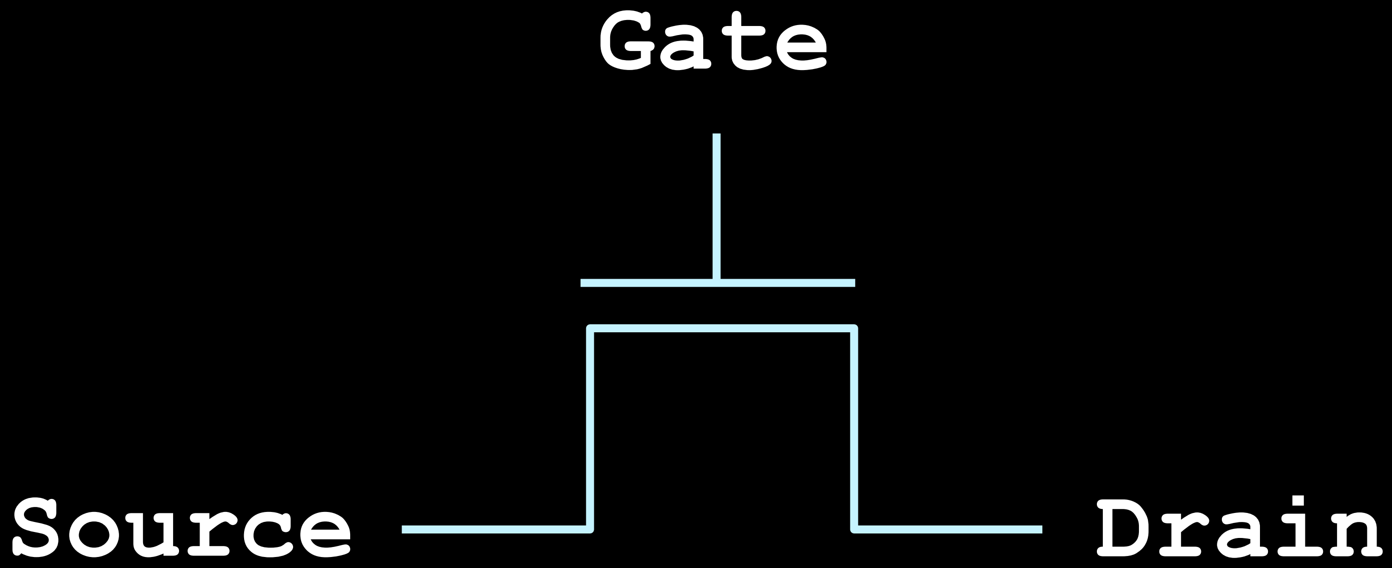
We are here


Assembly Language
Processors
Arithmetic Logic Units
Finite State Machines
Devices
Flip-flops
Circuits
Gates
Transistors

Recap: Transistors

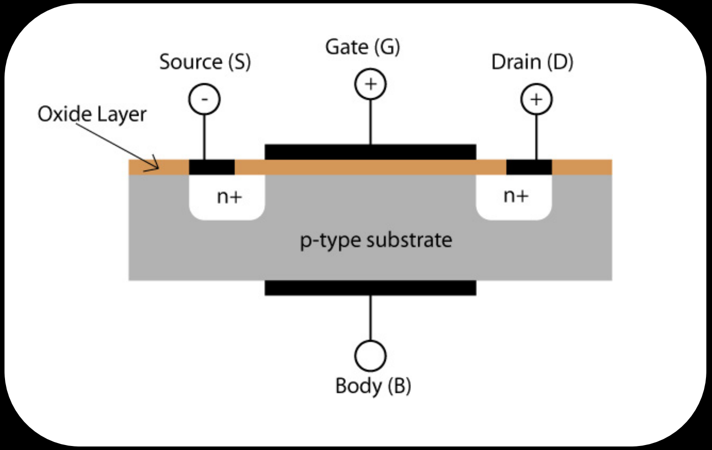
- **Transistors**, made of doped semiconductors put together (PN-Junctions), is like a resistor but can change its resistance.
- It has two state: connected (switched ON) or disconnected (switched OFF)
  - This is the origin of the 1's and 0's that all current computers are built with. (Quantum computer may do it differently)
- The ON/OFF state of a transistor is control by an electrical signal, like in the MOSFET.

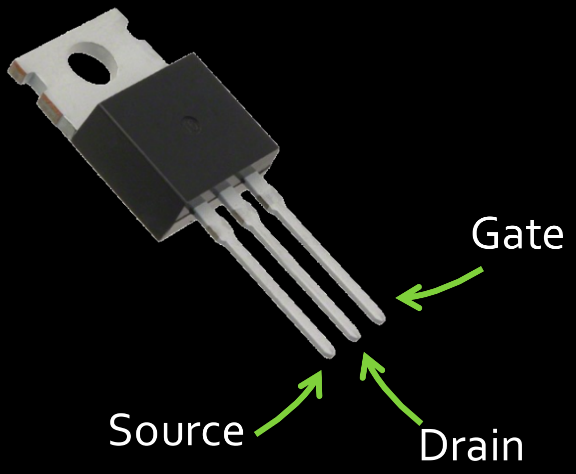
MOSFET



From transistors to gates

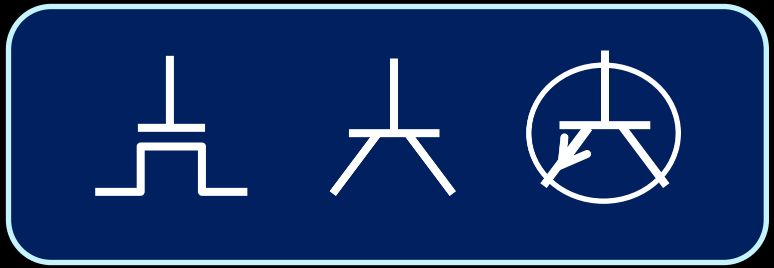
- Transistors are semiconductor circuits that can connect the source and the drain together, depending on the voltage value at the gate.
  - For `NPN` MOSFETs (nMOS), they are connected when the gate value is high.
  - For `PNP` MOSFETs (pMOS), they are connected when the gate value is low.
- These are then used to make digital logic gates.



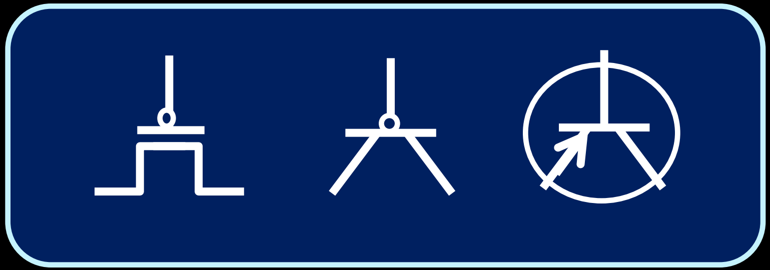


Transistor notation

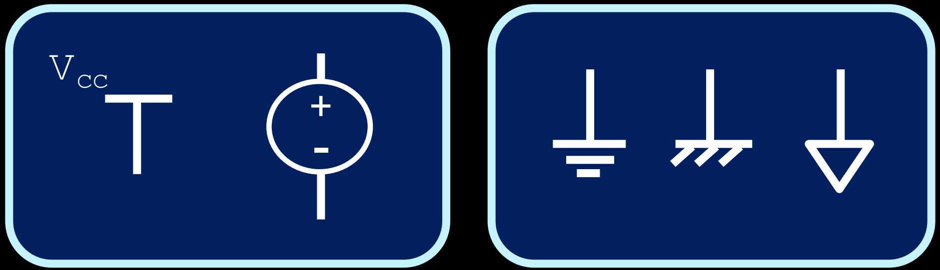
- NPN transistor:



- PNP transistor:

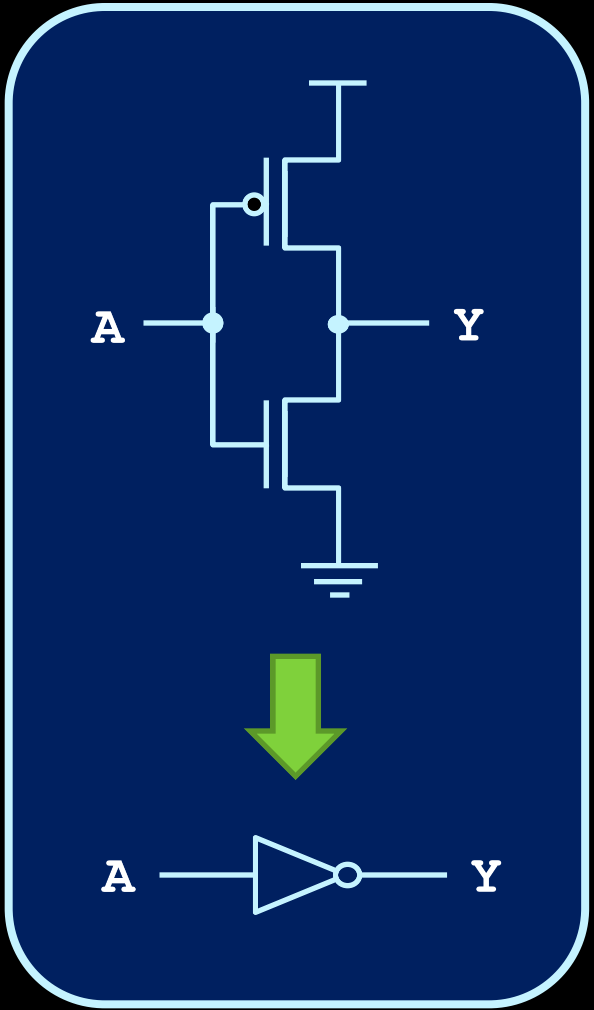


- Voltage values:

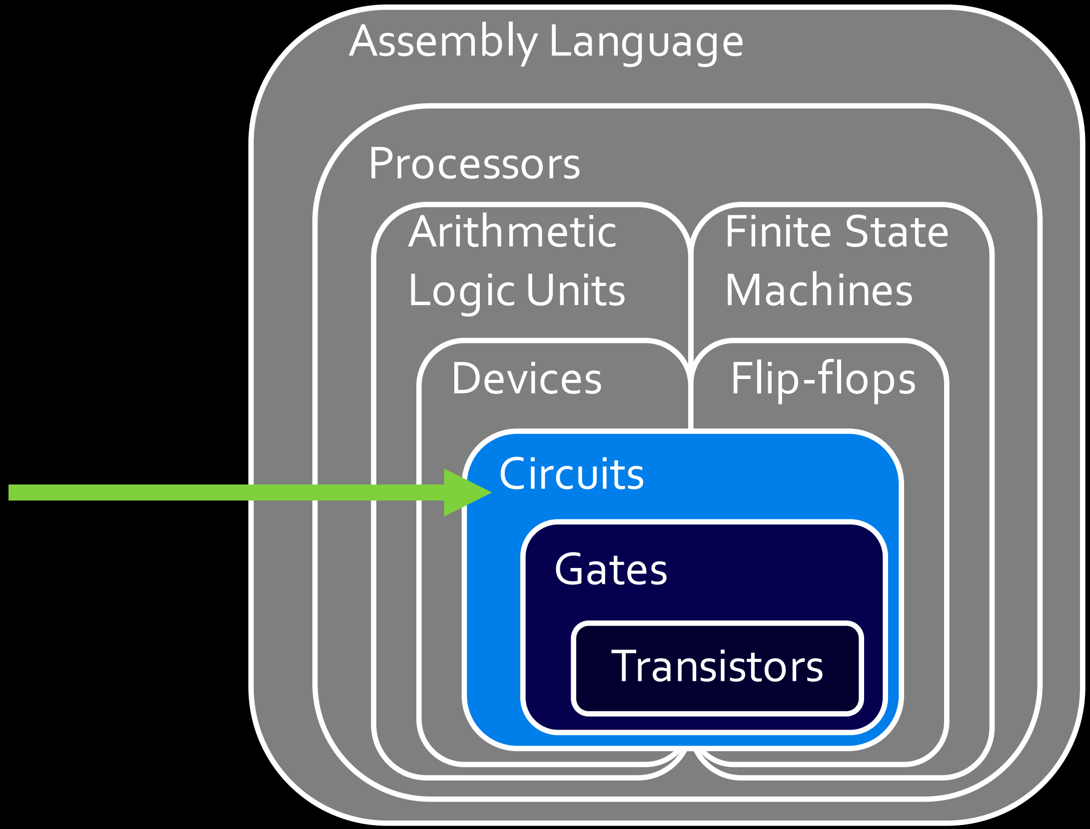


How gates are made

- To create logic gates:
  - Remember that transistors act like faucets for electricity.
  - The inputs to the logic gates determine if the outputs will be connected to high or low voltage.
  - <u>Example:</u> NOT gates:

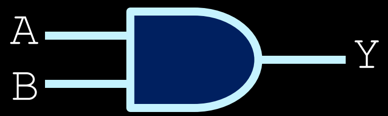


From gates to circuits

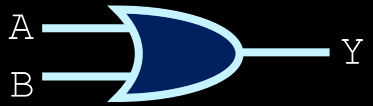


Making logic with gates

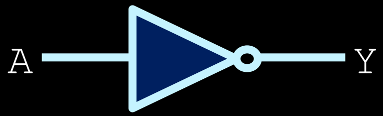
- Logic gates like the following allow us to create an output value, based on one or more input values.
  - Each corresponds to Boolean logic that we've seen before in CSC108 and CSC148:



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



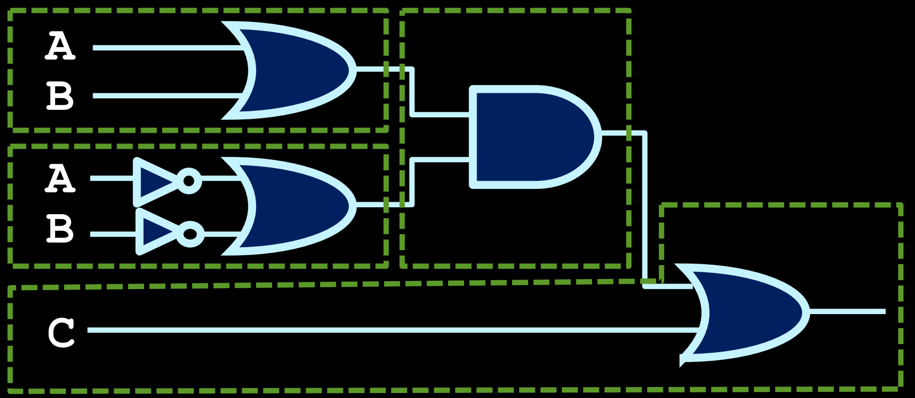
A	Y
0	1
1	0

Making boolean expressions

- So how would you represent Boolean expressions using logic gates?

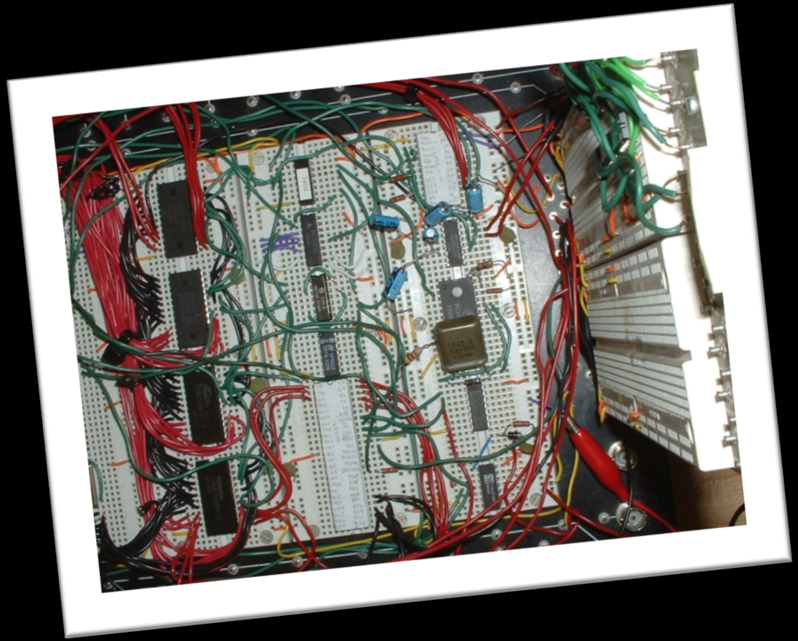
```
Y = (A or B) and (not A or not B) or C
```

- Like so:



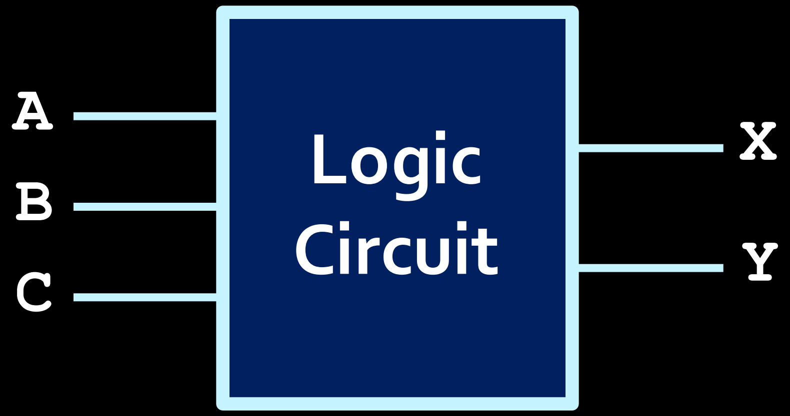
Creating complex circuits

- What do we do in the case of more complex circuits, with several inputs and more than one output?
  - If you're lucky, a truth table is provided to express the circuit.
  - Usually the behaviour of the circuit is expressed in words, and the first step involves creating a truth table that represents the described behaviour.



Circuit example

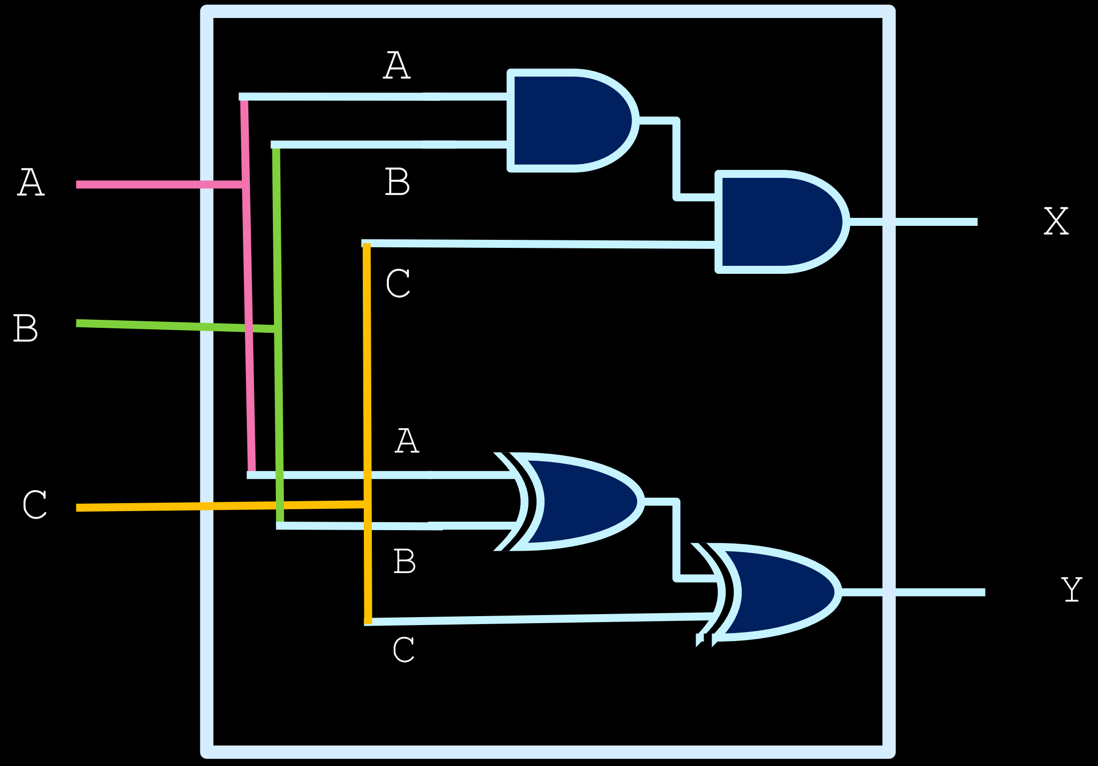
- The circuit on the right has three inputs (A, B and C) and two outputs (X and Y).



- What logic is needed to set X high when all three inputs are high?
- What logic is needed to set Y high when the number of high inputs is odd?

Combinational circuits

- Small problems can be solved easily.



X high when all three inputs are high

Y high when number of high is odd

For more complicated circuits,
we need a systematical approach

Creating complex logic

- The general approach
- Basic steps:
  1. Create truth tables based on the desired behaviour of the circuit.
  2. Come up with a "good" Boolean expression that has exactly that truth table.
  3. Convert Boolean expression to gates.
- The key to an efficient design?
  - Spending extra time on Step #2.

First,
a better way to represent
truth tables

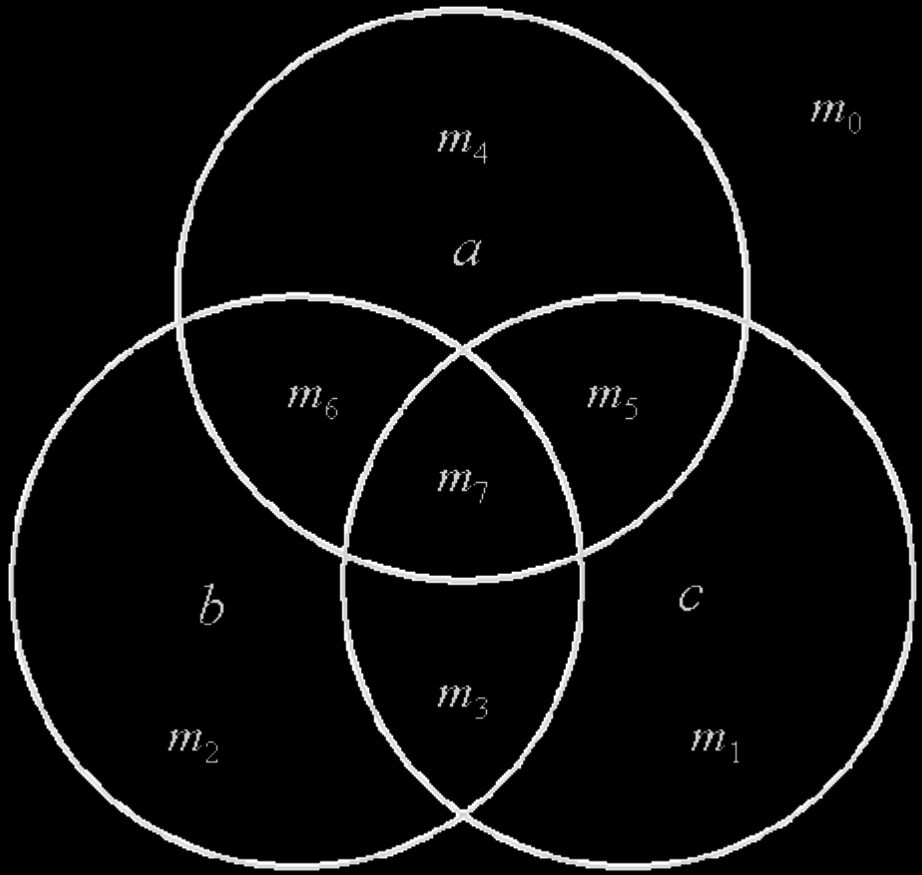
Example truth table

- Consider the following example:
  - *"Y is high only when B and C are both high"*
- This leads to the truth table on the right.
  - Do we always have to draw the whole table?
  - Is there a better way to describe the truth table?

A	B	C	Y
0	0	0	0
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

This is all we needed to express!

Yes, use "Minterms" and "Maxterms"



Minterms, informally

- First, sort the rows according to the value of the number “ABC” represents
- Then for each row, we name the inputs as $m_{\{\text{row number}\}}$
- m_0, m_1, m_2, …, are called **minterms**

Sorted

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Minterm	Y
m_0	0
m_1	1
m_2	1
m_3	1
m_4	1
m_5	0
m_6	1
m_7	0

Minterm, a more formal description

Minterm: an AND expression with every input present in true or complemented form.

$m_0: \overline{A} \cdot \overline{B} \cdot \overline{C}$

$m_1: \overline{A} \cdot \overline{B} \cdot C$

$m_2: \overline{A} \cdot B \cdot \overline{C}$

$m_3: \overline{A} \cdot B \cdot C$

$m_7: A \cdot B \cdot C$

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Minterm	Y
m_0	0
m_1	1
m_2	1
m_3	1
m_4	1
m_5	0
m_6	1
m_7	0

Minterm (m) and Maxterm (M)

Minterm: an **AND** expression with every input present in true or complemented form.

Maxterm: an **OR** expression with every input present in true or complemented form.

M_0: $A+B+C$ M_1: $A+B+\overline{C}$

M_6: $\overline{A}+\overline{B}+C$ M_7: $\overline{A}+\overline{B}+\overline{C}$

Feel something fishy?

Naming!

Minterm is about when the output is 1

m_0 is $\overline{A} \cdot \overline{B} \cdot \overline{C}$

$\overline{A} \cdot \overline{B} \cdot \overline{C}$ is **1** only when A, B, C are 0, 0, 0

M_0 is $A+B+C$

$A+B+C$ is **0** only when A, B, C are 0, 0, 0

Maxterm is about when the output is 0

Exercise: Minterm or Maxterm?

given four inputs: (A, B, C, D)

$A \cdot B \cdot C$

Neither! **Every** input needs to be there, D is missing!

$A+B+D$

Neither! Same reason

$A \cdot B+C \cdot \overline{D}$

Neither! It has to be **only AND** or **only OR**, cannot mix

$A+\overline{B}+C+\overline{D}$

Maxterm, M_5

$A \cdot \overline{B} \cdot C \cdot \overline{D}$

Minterm, M_{10}

Quick fact

- Given `n` inputs, how many possible minterms and maxterms are there?
  - 2^n minterms and 2^n maxterms possible (same as the number of rows in a truth table).

Quick note about notations

- AND operations are denoted in these expressions by the multiplication symbol.
  - e.g. $A \cdot B \cdot C$ or $A*B*C$ ≈ $A \wedge B \wedge C$
- OR operations are denoted by the addition symbol.
  - e.g. $A+B+C$ ≈ $A \vee B \vee C$
- NOT is denoted by multiple symbols.
  - e.g. $\neg A$ or A' or $\overline{A}$
- XOR occurs rarely in circuit expressions.
  - e.g. $A \oplus B$

Use minterms and maxterms

to go from truth table to logic expression

Using minterms

- What are minterms used for?
  - A single minterm indicates a set of inputs that will make the output go high.
  - Example: Describe the truth table on the right using minterm:
    - m_2 A'B'CD'

A	B	C	D	m_2
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Using minterms

- What happens when you OR two minterms?
  - Result is output that goes high in both minterm cases.
  - Describe the truth table with the right-most column of outputs
  - **$m_2 + m_8$**

A	B	C	D	m_2	m_8	m_2+m_8
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	1
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	1	1
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0

We came up with a logic expression that has the desired truth table, easily: A'B'CD' + AB'C'D'

Creating boolean expressions

- Two canonical forms of boolean expressions:
  - Sum-of-Minterms (SOM): AB + A'B + AB'
    - Each minterm corresponds to a single high output in the truth table.
    - Also known as: Sum-of-Products.
  - Product-of-Maxterms (POM): (A+B)(A' + B)(A+B')
    - Each maxterm corresponds to a single low output in the truth table.
    - Also known as Product-of-Sums.

Every logic expression can be converted to a SOM, also to a POM.

$$Y = m_2 + m_6 + m_7 + m_{10} \quad \text{(SOM)}$$

A	B	C	D	m_2	m_6	m_7	m_{10}	Y
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	1
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0
0	1	1	0	0	1	0	0	1
0	1	1	1	0	0	1	0	1
1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0

$$Y = M_3 \cdot M_5 \cdot M_7 \cdot M_{10} \cdot M_{14} \text{ (POM)}$$

A	B	C	D	M_3	M_5	M_7	M_{10}	M_{14}	Y
0	0	0	0	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	0
0	1	0	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	0
0	1	1	0	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	0
1	0	0	0	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1
1	0	1	0	1	1	1	0	1	0
1	0	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1

Sum-of-Minterms vs Product-of-Maxterm

- SOM expresses which inputs cause the output to go high.
- POM expresses which inputs cause the output to go low
- SOMs are useful in cases with very few input combinations that produce high output.
- POMs are useful when expressing truth tables that have very few low output cases…

What if we do this using POM?

$$Y = m_2 + m_6 + m_7 + m_{10} \text{ (SOM)}$$

A	B	C	D	m_2	m_6	m_7	m_{10}	Y
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	1
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0
0	1	1	0	0	1	0	0	1
0	1	1	1	0	0	1	0	1
1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0

What if we do this using SOM?

$$Y = M_3 \cdot M_5 \cdot M_7 \cdot M_{10} \cdot M_{14} \text{ (POM)}$$

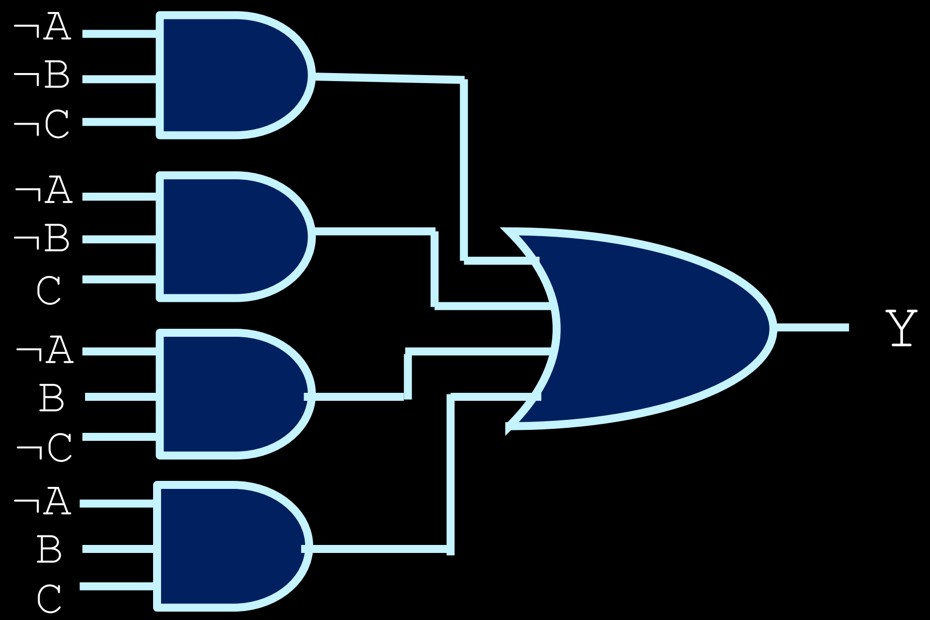
A	B	C	D	M_3	M_5	M_7	M_{10}	M_{14}	Y
0	0	0	0	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	0
0	1	0	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	0
0	1	1	0	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	0
1	0	0	0	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1
1	0	1	0	1	1	1	0	1	0
1	0	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1

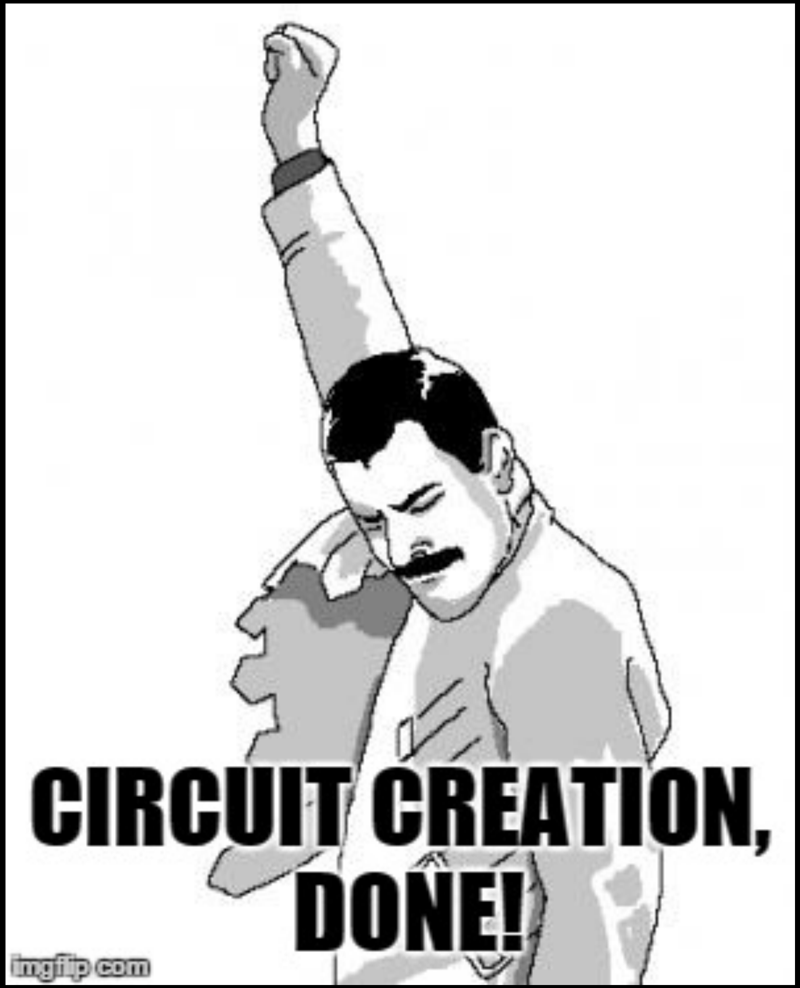
Converting SOM to gates

- Once you have a Sum-of-Minterms expression, it is easy to convert this to the equivalent combination of gates:

$$m_0 + m_1 + m_2 + m_3 =$$

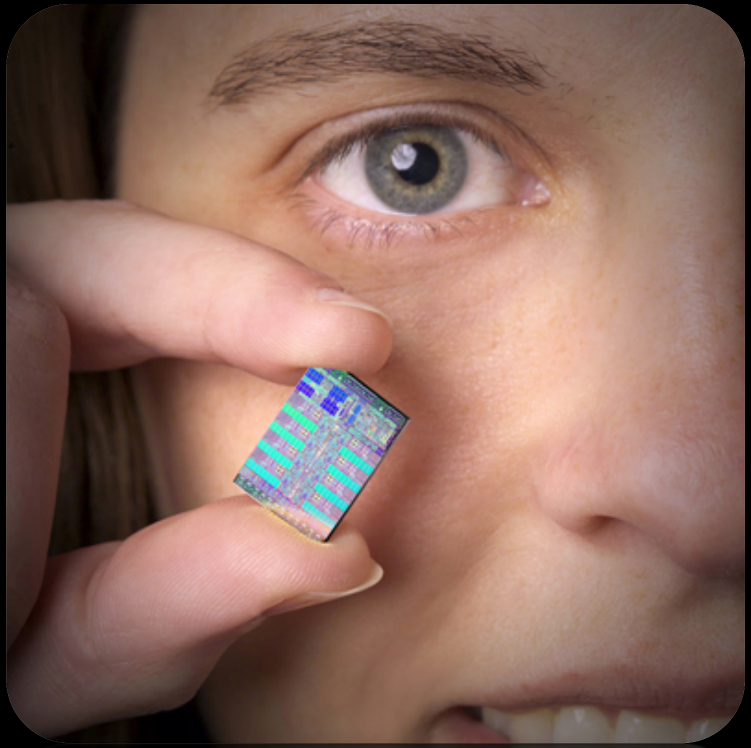
$$\overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot\overline{C} + \overline{A}\cdot B\cdot C$$



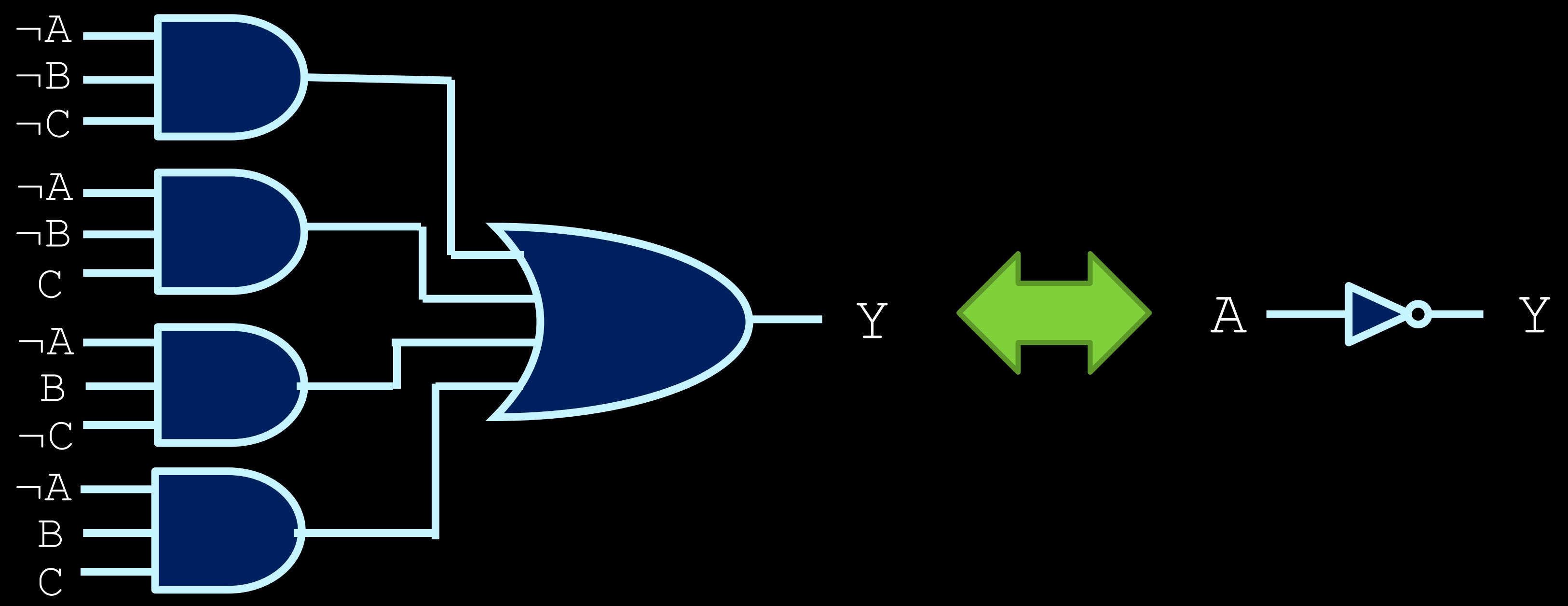

CIRCUIT CREATION,
DONE!
imgflip.com


NO
ITS NOT
memegenerator.net

Reducing circuits



Reasons for reducing circuits



- To minimize the number of gates, we want to reduce the Boolean expression as much as possible from a collection of minterms to something smaller.
- This is where math skills come in handy ☺

Boolean algebra review

- Axioms:

$$0 \cdot 0 = 0 \qquad 0 \cdot 1 = 1 \cdot 0 = 0$$
$$1 \cdot 1 = 1 \qquad \text{if } x = 1,\ \overline{x} = 0$$

- From this, we can extrapolate:

$$x \cdot 0 = 0 \qquad x+1 = 1$$
$$x \cdot 1 = x \qquad x+0 = x$$
$$x \cdot x = x \qquad x+x = x$$
$$x \cdot \overline{x} = 0 \qquad x+\overline{x} = 1$$
$$\overline{\overline{x}} = x$$

Other boolean identities

- Commutative Law:

$$x \cdot y = y \cdot x \qquad x+y = y+x$$

- Associative Law:

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$
$$x+(y+z) = (x+y)+z$$

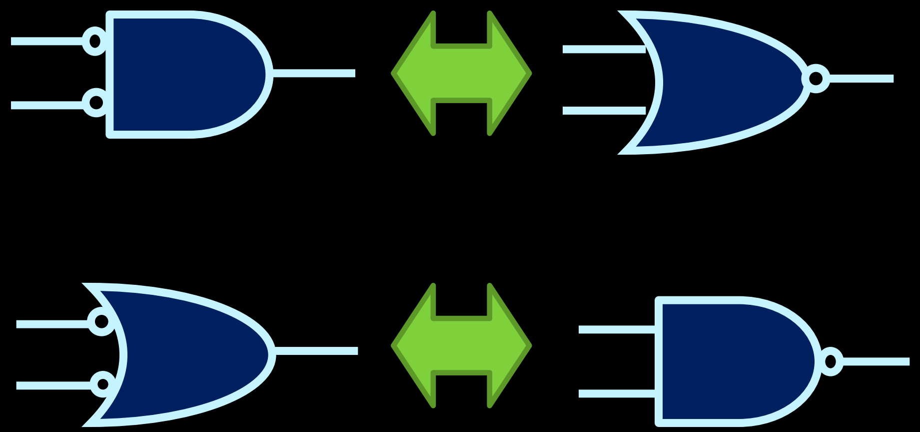
- Distributive Law:

$$x \cdot (y+z) = x \cdot y + x \cdot z$$
$$x+(y \cdot z) = (x+y) \cdot (x+z)$$

Other boolean identities

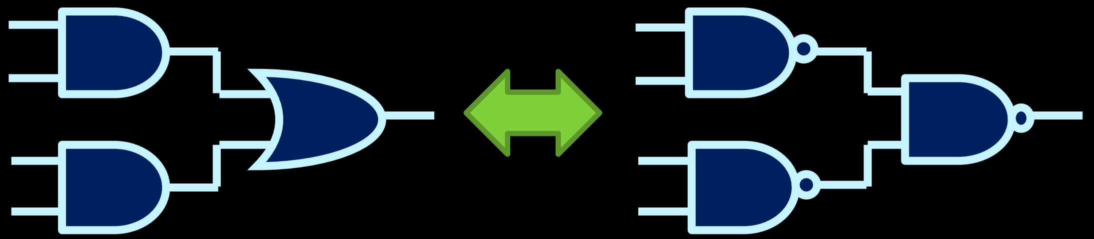
- De Morgan's Laws:

$$\overline{x} \cdot \overline{y} = \overline{x+y}$$
$$\overline{x}+\overline{y} = \overline{x \cdot y}$$

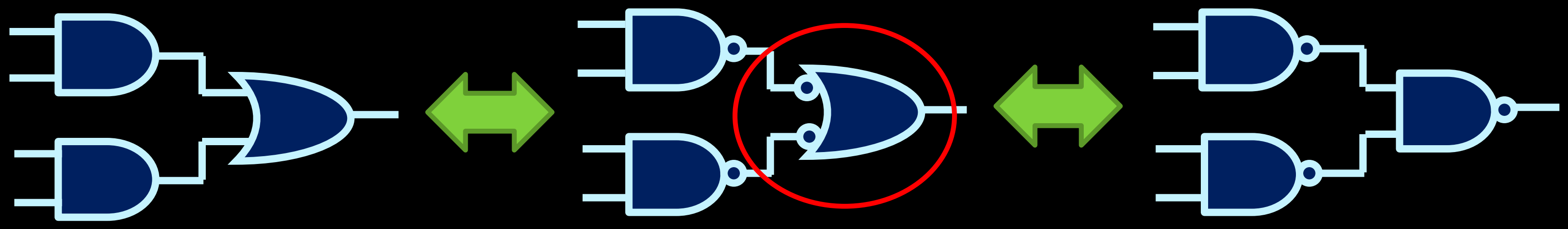


De Morgan and NAND gates

- De Morgan’s Law is important because out of all the gates, NANDs are the cheapest to fabricate.
  - a Sum-of-Products circuit could be converted into an equivalent circuit of NAND gates:



- This is all based on de Morgan’s Law:



Reducing boolean expressions

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

- Assuming logic specs at left, we get the following:

$$m_3 + m_4 + m_6 + m_7$$

$$Y = \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot \bar{C} + A \cdot B \cdot \bar{C} + A \cdot B \cdot C$$

Warming up...

$$A \cdot B + A \cdot \overline{B} \quad = \; A$$

Reduce by combing two terms that differ by a single literal.

Let's reduce this

$$Y = \overline{A}\cdot B\cdot C + A\cdot\overline{B}\cdot\overline{C} + \underline{A\cdot B\cdot\overline{C} + A\cdot B\cdot C}$$

Combine the last two terms…

$$Y = \overline{A}\cdot B\cdot C + A\cdot\overline{B}\cdot\overline{C} + A\cdot B$$

Combine the middle two and the end two …

$$Y = B\cdot C + A\cdot\overline{C}$$

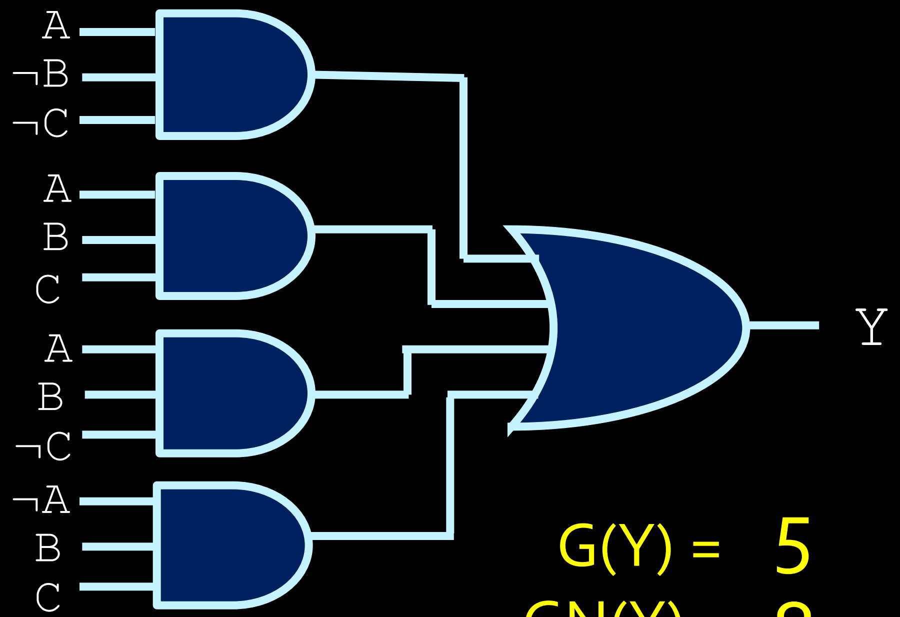
There could be different ways of combining, some are **simpler** than others.

How to get to the simplest expression?

Wait … What does “simplest” mean?

What is “simplest”?

- In this case, “simple” denotes the lowest gate cost (G) or the lowest gate cost with NOTs (GN).
- To calculate the gate cost, simply add all the gates together (as well as the cost of the NOT gates, in the case of the GN cost).

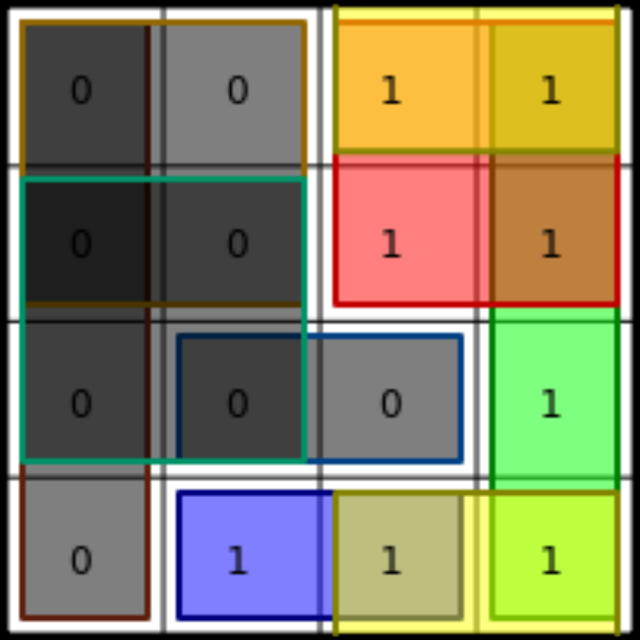


G(Y) = 5
GN(Y) = 8

Don’t count ¬C twice!

Karnaugh maps

Find the simplest expression, systematically.



Reducing boolean expressions

- How do we find the “simplest” expression for a circuit?
  - Technique called Karnaugh maps (or K-maps).
  - Karnaugh maps are a 2D grid of minterms, where adjacent minterm locations in the grid differ by a single literal.
  - Values of the grid are the output for that minterm.

	$\overline{B} \cdot \overline{C}$	$\overline{B} \cdot C$	$B \cdot C$	$B \cdot \overline{C}$
$\overline{A}$	0	0	1	0
A	1	0	1	1

Compare these…

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

	$\overline{B} \cdot \overline{C}$	$\overline{B} \cdot C$	$B \cdot C$	$B \cdot \overline{C}$
$\overline{A}$	0	0	1	0
A	1	0	1	1

$$Y = \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot \overline{C} + A \cdot B \cdot C$$

Karnaugh maps

- Karnaugh maps can be of any size, and have any number of inputs.
- Since adjacent minterms only differ by a single literal, they can be combined into a single term that omits that value.

	$\overline{C}\cdot\overline{D}$	$\overline{C}\cdot D$	$C\cdot D$	$C\cdot\overline{D}$
$\overline{A}\cdot\overline{B}$	m_0	m_1	m_3	m_2
$\overline{A}\cdot B$	m_4	m_5	m_7	m_6
$A\cdot B$	m_{12}	m_{13}	m_{15}	m_{14}
$A\cdot\overline{B}$	m_8	m_9	m_{11}	m_{10}

Using Karnaugh maps

- Once Karnaugh maps are created, draw boxes over groups of high output values.
  - Boxes must be rectangular, and aligned with map.
  - Number of values contained within each box must be a power of 2.
  - Boxes may overlap with each other.
  - Boxes may wrap across edges of map.

	$\overline{B}\cdot\overline{C}$	$\overline{B}\cdot C$	$B\cdot C$	$B\cdot\overline{C}$
$\overline{A}$	0	0	1	0
A	1	0	1	1

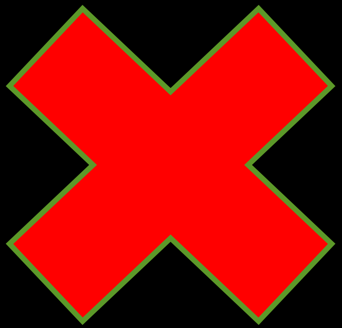
	$\overline{B}\cdot\overline{C}$	$\overline{B}\cdot C$	$B\cdot C$	$B\cdot\overline{C}$
$\overline{A}$	0	1	1	0
A	0	0	1	0

Must be rectangle!

	$\overline{B} \cdot \overline{C}$	$\overline{B} \cdot C$	$B \cdot C$	$B \cdot \overline{C}$
$\overline{A}$	0	1	1	0
A	0	0	1	0

Two boxes overlapping each other is fine.

	$\overline{B} \cdot \overline{C}$	$\overline{B} \cdot C$	$B \cdot C$	$B \cdot \overline{C}$
$\overline{A}$	0	1	1	1
A	0	0	0	0



Number of value contained must be power of 2.

	$\overline{B} \cdot \overline{C}$	$\overline{B} \cdot C$	$B \cdot C$	$B \cdot \overline{C}$
$\overline{A}$	0	1	1	1
A	0	0	0	0

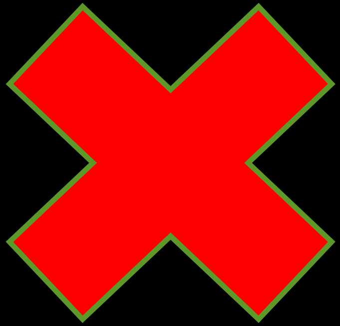
1 is a power of 2

$1 = 2^0$

	$\overline{B}\cdot\overline{C}$	$\overline{B}\cdot C$	$B\cdot C$	$B\cdot\overline{C}$
$\overline{A}$	0	1	1	0
A	0	1	1	0

Rectangle, with power of 2 entries

	$\overline{B}\cdot\overline{C}$	$\overline{B}\cdot C$	$B\cdot C$	$B\cdot\overline{C}$
$\overline{A}$	0	1	0	0
A	0	0	1	0



Must be aligned with map.

	$\overline{B}\cdot\overline{C}$	$\overline{B}\cdot C$	$B\cdot C$	$B\cdot\overline{C}$
$\overline{A}$	0	0	0	0
A	1	0	0	1

Wrapping across edge is fine.

So… how to find smallest expression

Minterms in one box can be combined into one term

	$\overline{B} \cdot \overline{C}$	$\overline{B} \cdot C$	$B \cdot C$	$B \cdot \overline{C}$
$\overline{A}$	0	0	1	0
A	0	0	1	0

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

So… how to find smallest expression

The simplest expression corresponds to the smallest number of **boxes** that cover all the high values (1’s).

	$\overline{B}\cdot\overline{C}$	$\overline{B}\cdot C$	$B\cdot C$	$B\cdot\overline{C}$
$\overline{A}$	0	0	1	0
A	1	0	1	1

	$\overline{B}\cdot\overline{C}$	$\overline{B}\cdot C$	$B\cdot C$	$B\cdot\overline{C}$
$\overline{A}$	0	0	1	0
A	1	0	1	1

$$Y = \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot \overline{C} + A \cdot B \cdot C$$

	$\overline{B} \cdot \overline{C}$	$\overline{B} \cdot C$	$B \cdot C$	$B \cdot \overline{C}$
$\overline{A}$	0	0	1	0
A	1	0	1	1

$$Y = \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot B$$

	$\overline{B} \cdot \overline{C}$	$\overline{B} \cdot C$	$B \cdot C$	$B \cdot \overline{C}$
$\overline{A}$	0	0	1	0
A	1	0	1	1

$$Y = B \cdot C + A \cdot \overline{C}$$

K-map: the steps

Given a complicated expression

1. Convert it to Sum-Of-Minterms
2. Draw the 2D grid
3. Mark all the high values (1’s), according to which minterms are in the SOM.
4. Draw boxes that cover 1’s.
5. Find the smallest set of boxes that cover all 1’s.
6. Write out the simpfied result according to the boxes found.

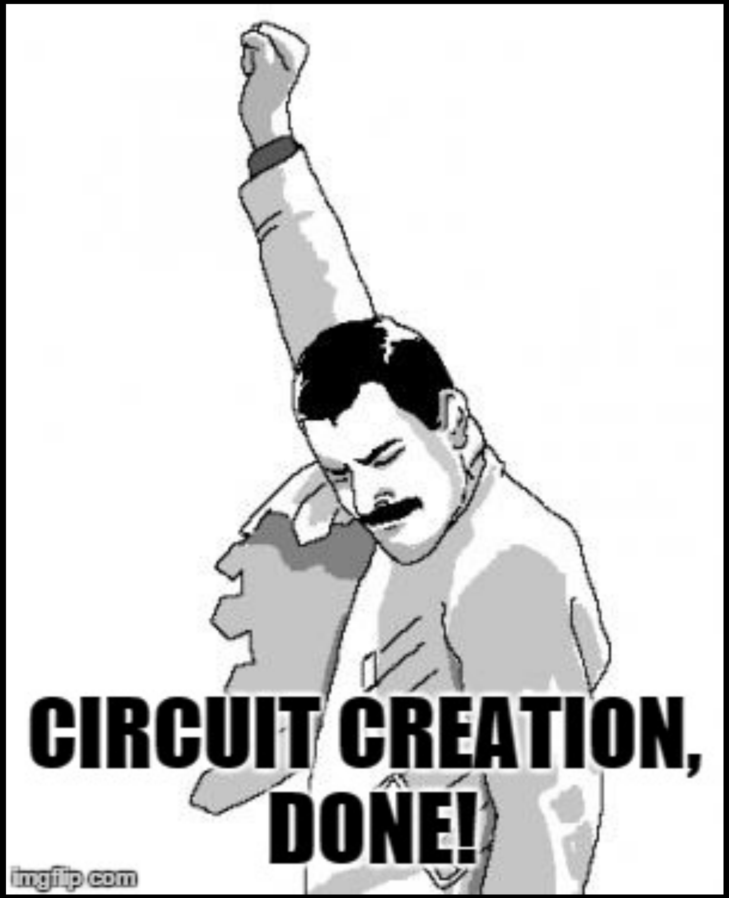
Everything can be done using Maxterms, too

- Can also use this technique to group maxterms together as well.
- Karnaugh maps with maxterms involves grouping the zero entries together, instead of grouping the entries with one values.

	$C+D$	$C+\overline{D}$	$\overline{C}+\overline{D}$	$\overline{C}+D$
$A+B$	M_0	M_1	M_3	M_2
$A+\overline{B}$	M_4	M_5	M_7	M_6
$\overline{A}+\overline{B}$	M_{12}	M_{13}	M_{15}	M_{14}
$\overline{A}+B$	M_8	M_9	M_{11}	M_{10}

Circuit creation – the whole flow

1. Understand desired behaviour
2. Write the truth table based on the behaviour
3. Write the SOM (or POM) of that truth table
4. Simplify the SOM using K-map
5. Translate the simplified logic expression into circuit with gates.



Today we learned

- How to create a logic circuit from scratch, given a desired digital behaviour.
- Minterm & Maxterm
- K-Map use to reduce the circuit

Next Week:

- Logical Devices

Quiz Time!

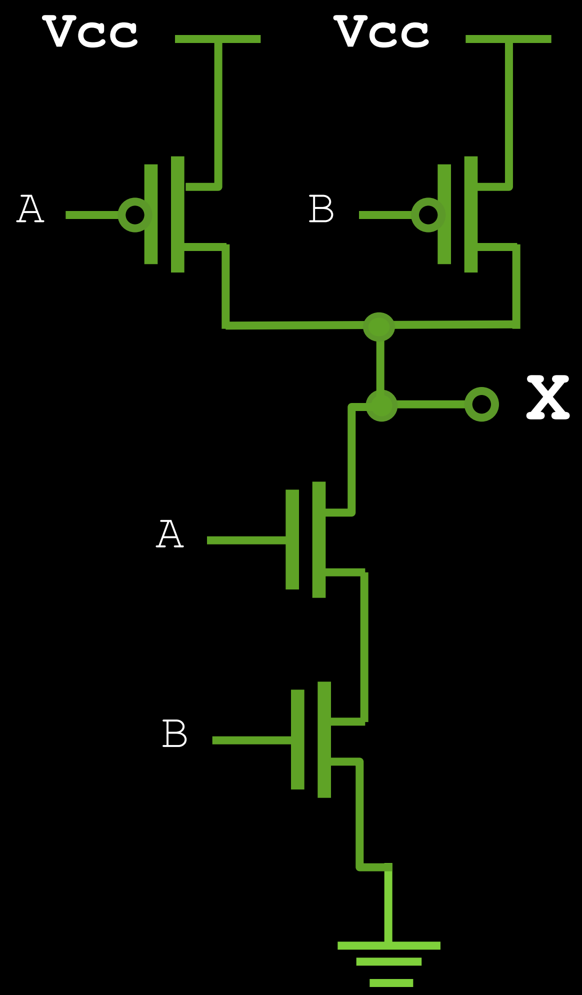
Question 1

After doping a piece of silicon with n-type impurity (phosphorus), there overall electric charge of the material is

A. Positive
B. Negative
C. **Neutral**
D. None of above

Question 2

- What gate is this?
- A and B are inputs, X is output.



NAND

Question 3

- Write the following truth table as Sum-of-Minterms.

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Write in this form

A’B’C + A’BC’

Flip your quiz face-down and pass it to your right.