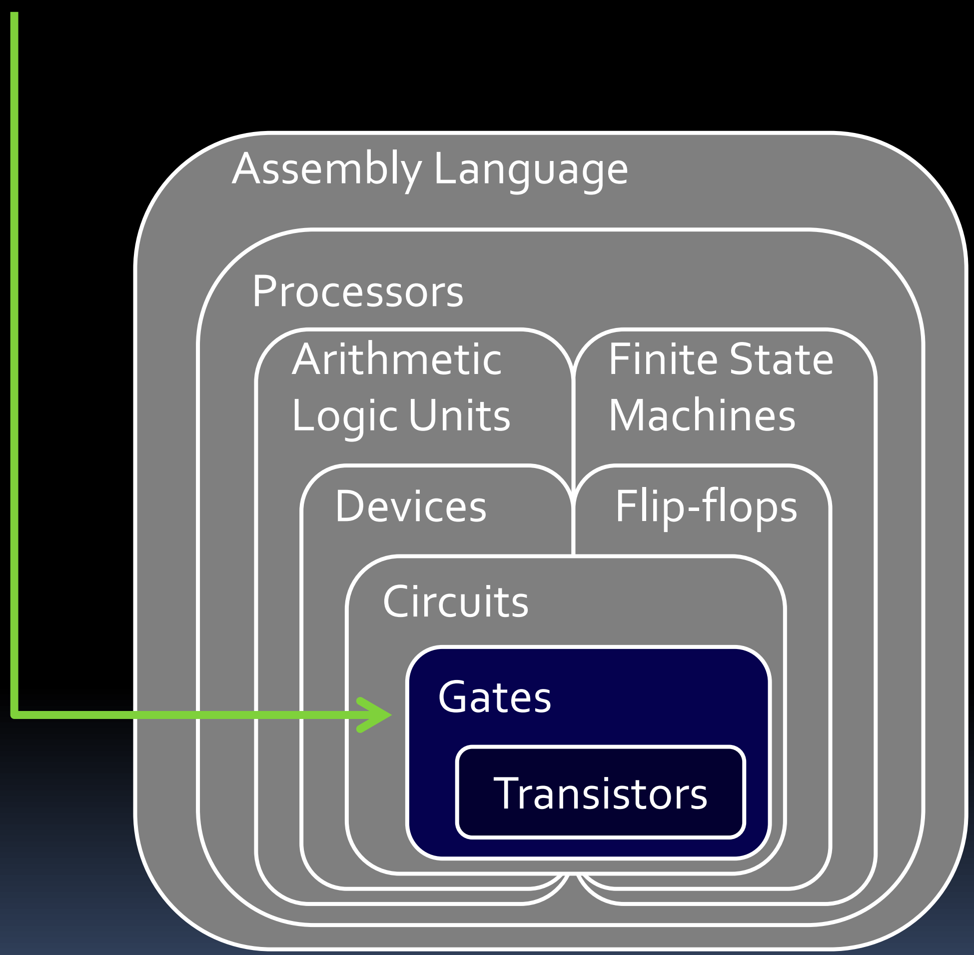
Week 2:
Circuit Creation

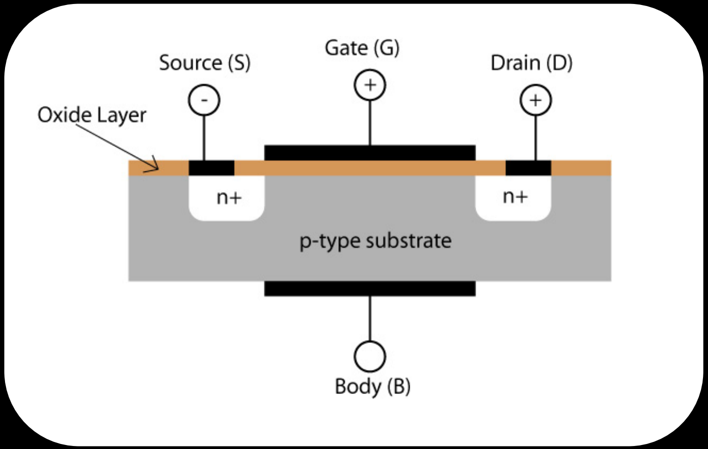
You are here

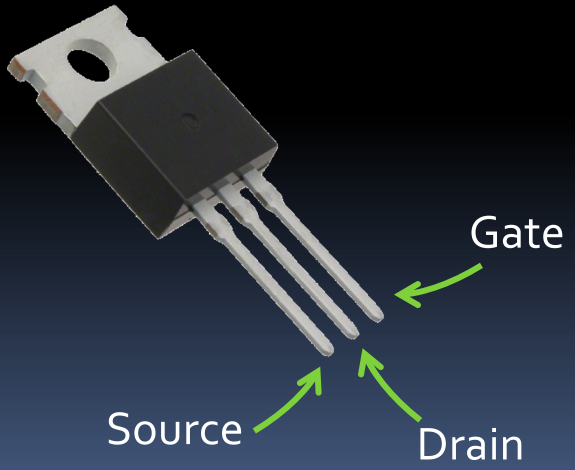


Assembly Language
Processors
Arithmetic Logic Units
Finite State Machines
Devices
Flip-flops
Circuits
Gates
Transistors

From transistors to gates

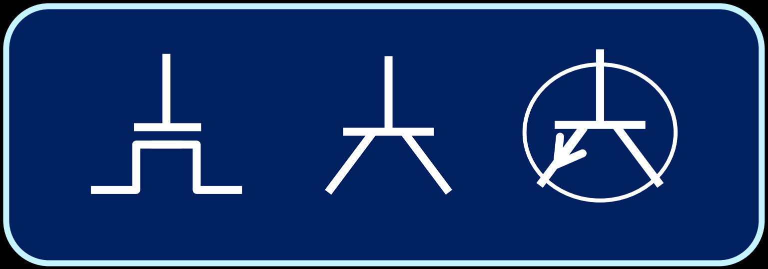
- Transistors are semiconductor circuits that can connect the source and the drain together, depending on the voltage value at the gate.
  - NPN transistors (nMOS) are connected when the gate value is high.
  - PNP transistors (pMOS) are connected when the gate value is low.
- These are then used to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



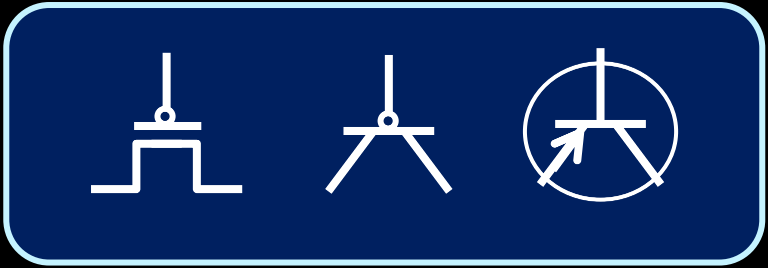


Transistor notation

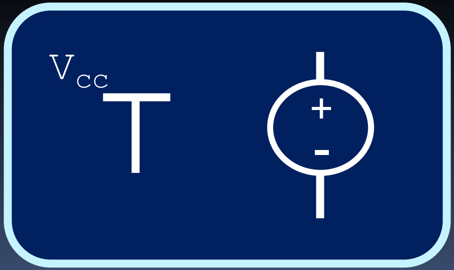
- NPN transistor:

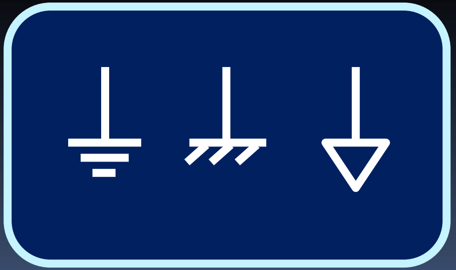


- PNP transistor:



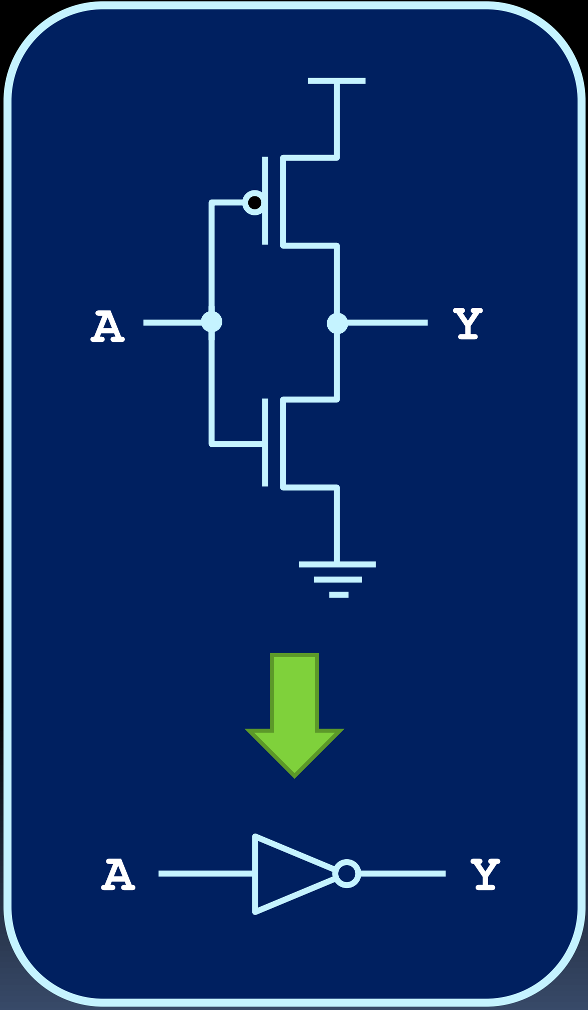
- Voltage values:



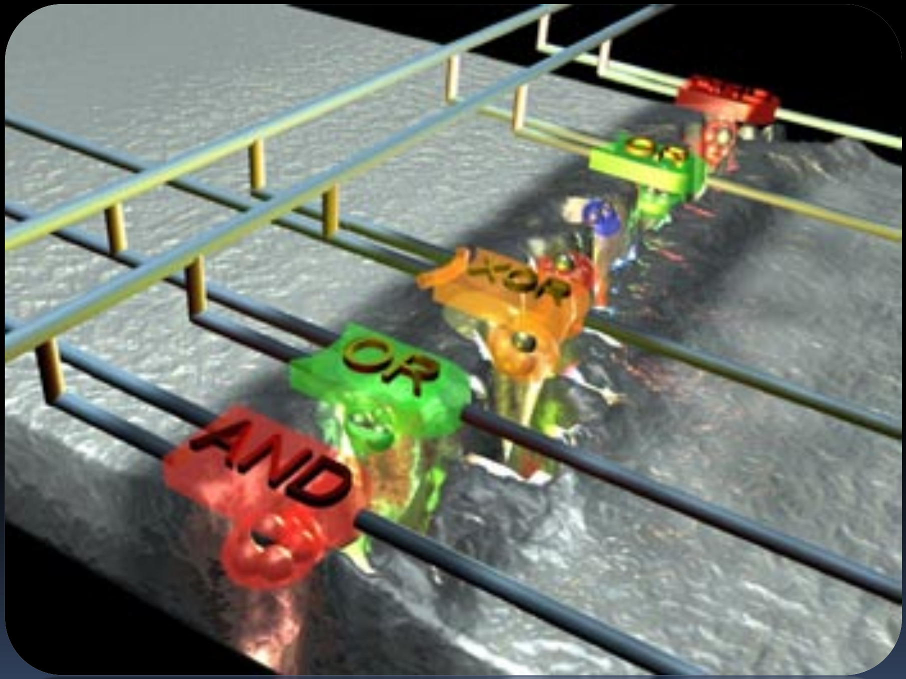


How gates are made

- 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.
  - Example: NOT gates:

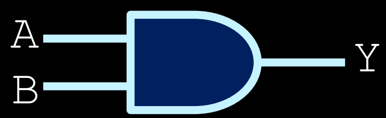


Creating circuits with gates

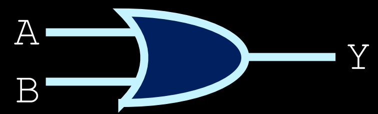


Making logic with gates

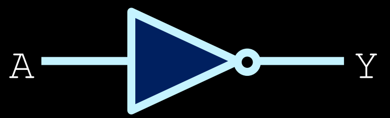
- Logic gates like the following allow us to create an output value, based on one or more input values.
  - Each corresponds to Boolean logic that we've seen before in CSCA08/A48/A67:



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

Aside: notation

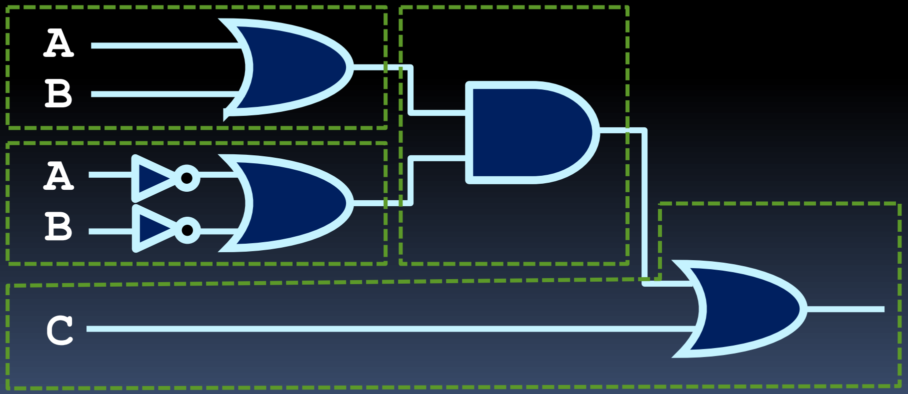
- While we're talking about notation…
  - AND operations are denoted in these expressions by the multiplication symbol.
    - e.g. $A \cdot B \cdot C$ or $A*B*C$ $\approx$ $A \land B \land C$
  - OR operations are denoted by the addition symbol.
    - e.g. $A+B+C$ $\approx$ $A \lor B \lor C$
  - NOT is denoted by multiple symbols.
    - e.g. $\neg A$ or A' or $\overline{A}$
  - XOR occurs rarely in circuit expressions.
    - e.g. $A \oplus B$

Making boolean expressions

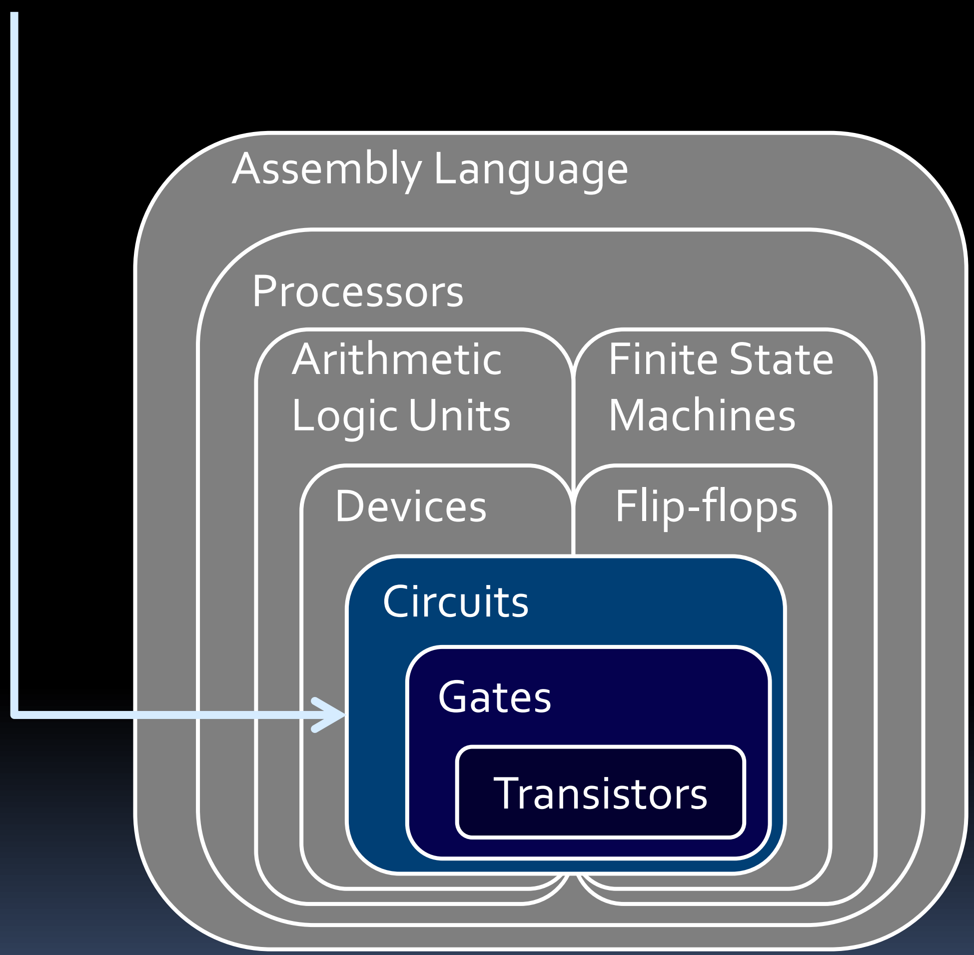
- So how would you represent boolean expressions using logic gates?

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

- Like so:

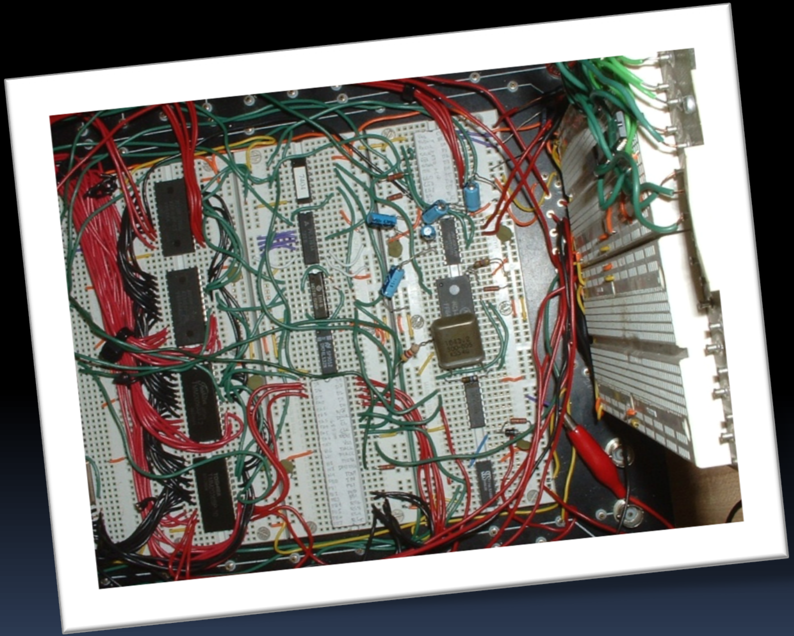


Now you are here



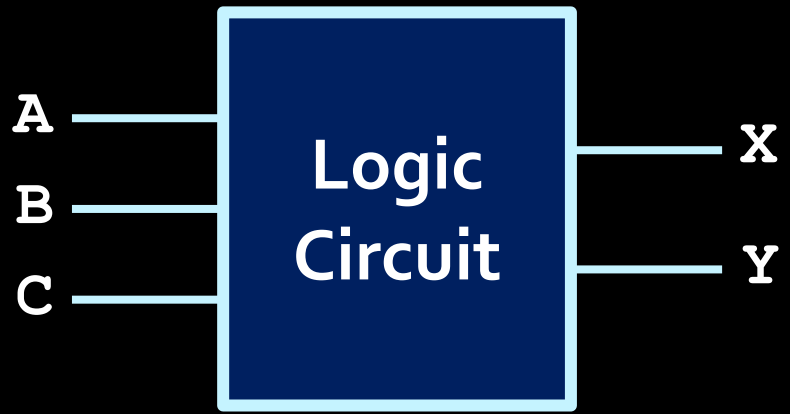
Creating complex circuits

- 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

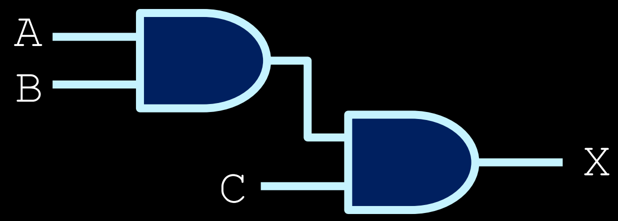
- The circuit on the right has three inputs (A, B and C) and two outputs (X and Y).

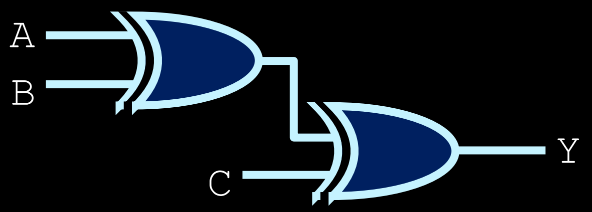


- What logic is needed to set X high when all three inputs are high?
- What logic is needed to set Y high when the number of high inputs is odd?

Combinational circuits

- Small problems can be solved easily.

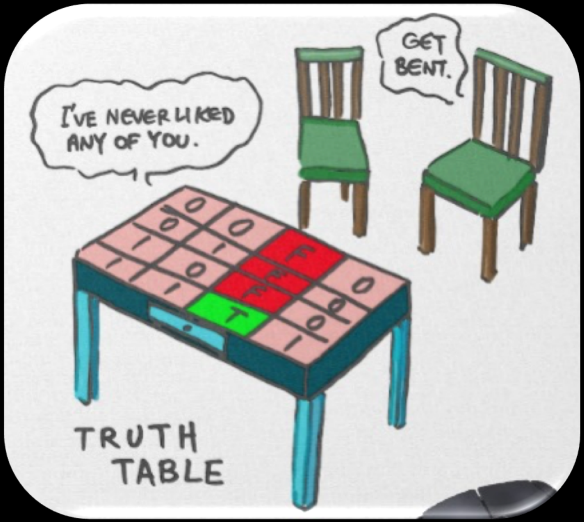




- Larger problems require a more systematic approach.
  - <u>Example:</u> Given three inputs `A`, `B`, and `C`, make output `Y` high in the case where all of the inputs are low, or when `A` and `B` are low and `C` is high, or when `A` and `C` are low but `B` is high, or when `A` is low and `B` and `C` are high.

Creating complex logic

- How do we approach problems like these (and circuit problems in general)?
- Basic steps:
  1. Create truth tables.
  2. Express as boolean expression.
  3. Convert to gates.
- The key to an efficient design?
  - Spending extra time on Step #2.

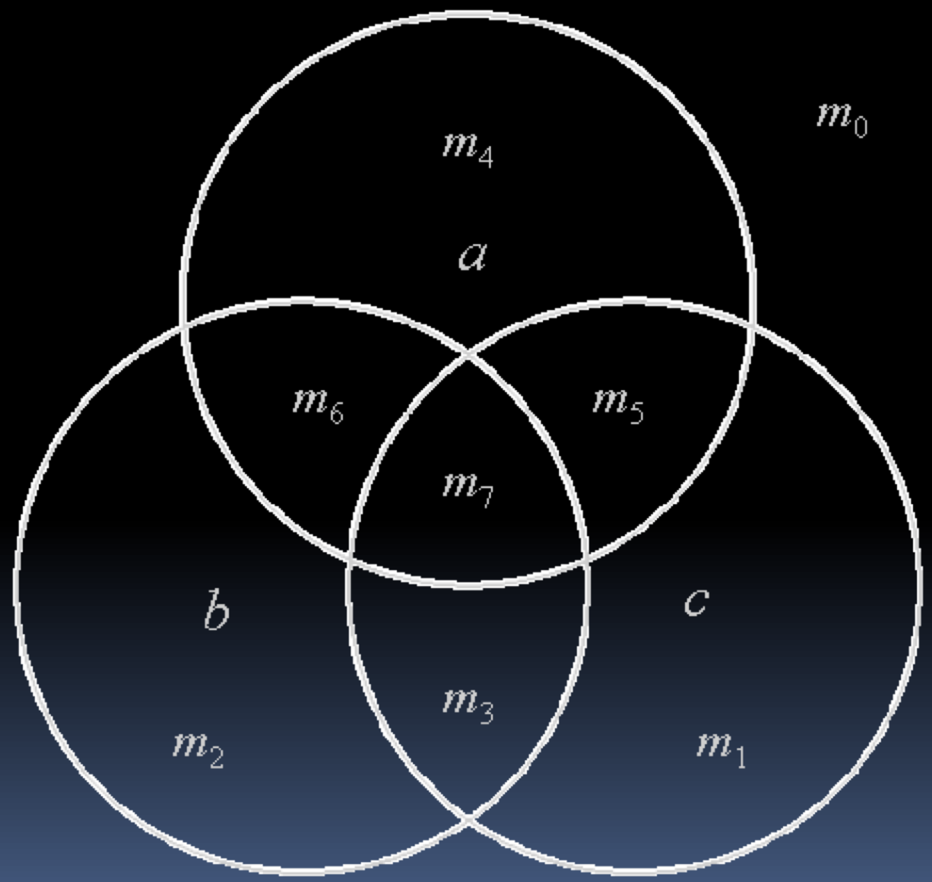


Example truth table

- Consider the following example:
  - *"Given three inputs A, B, and C, make output Y high wherever any of the inputs are low, except when all three are low or when A and C are high."*
  - This leads to the truth table on the right.
  - Is there a more compact way to describe this?

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

Minterms and Maxterms



Minterms

- An easier way to express circuit behaviour is to assume the standard truth table format, and then list which input rows cause high output.
  - These rows are referred to as minterms.

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

Minterms and maxterms

- A more formal description:
  - 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.
  - For example, given four inputs (A, B, C, D):
    - Valid minterms:
      - $A \cdot \overline{B} \cdot C \cdot D$, $\overline{A} \cdot B \cdot \overline{C} \cdot D$, $A \cdot B \cdot C \cdot D$
    - Valid maxterms:
      - $A+\overline{B}+C+D$, $\overline{A}+B+\overline{C}+D$, $A+B+C+D$
    - Neither minterm nor maxterm:
      - $A \cdot B+C \cdot D$, $A \cdot B \cdot D$, $A+B$

What is This For?

- Minterms and maxterms are a shorthand to refer to rows of the truth table.
- minterms describe rows where output is high.
- maxterms describe rows where output is low.’
- We then OR minterms or AND maxterms.
  - Don’t mix them both

Back to minterms

- Circuits are often described using minterms or maxterms, as a form of logic shorthand.
  - Given n inputs, there are 2^n minterms and maxterms possible (same as rows in a truth table).
  - Naming scheme:
    - Minterms are labeled as m_x, maxterms are labeled as M_x
      - The x subscript indicates the row in the truth table.
      - x starts at 0, and ends with $n-1$.
  - Example: Given 3 inputs –
    - Minterms are m_0 ($\overline{A} \cdot \overline{B} \cdot \overline{C}$) to m_7 ($A \cdot B \cdot C$)
    - Maxterms are M_0 ($A+B+C$) to M_7 ($\overline{A}+\overline{B}+\overline{C}$)

Quick Exercises

- Given 4 inputs A, B, C and D write:
  - m_9
  - m_{15}
  - m_{16}
  - M_2
- Which minterm is this?
  - $\overline{A} \cdot B \cdot \overline{C} \cdot \overline{D}$
- Which maxterm is this?
  - $A+B+C+\overline{D}$

m_0 vs M_0

- m_0 is $\bar{A}$ and $\bar{B}$ and $\bar{C}$
  - $m_0 = 1$ iff $A = B = C = 0$ (row 0)
- M_0 is A or B or C
  - $M_0 = 0$ iff $A = B = C = 0$ (row 0)
- Minterms tell us when the output is 1
- Maxterms tell us when the input is 0

Using minterms and maxterms

- What are minterms used for?
  - A single minterm indicates a set of inputs that will make the output go high.
  - Example: m_2
  - Output only goes high in third row of truth table.

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 and maxterms

- What happens when you OR two minterms?
  - Result is output that goes high in both minterm cases.
  - For m_2+m_8, both third and ninth rows of truth table result in high output.

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

Creating Boolean expressions

- Two canonical forms of Boolean expressions:
  - Sum-of-Minterms (SOM):
    - Since each minterm corresponds to a single high output in the truth table, the combined high outputs are a union of these minterm expressions.
    - Also known as: Sum-of-Products.
  - Product-of-Maxterms (POM):
    - Since each maxterm only produces a single low output in the truth table, the combined low outputs are an intersection of these maxterm expressions.
    - Also known as Product-of-Sums.

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

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

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

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

Using Sum-of-Minterms

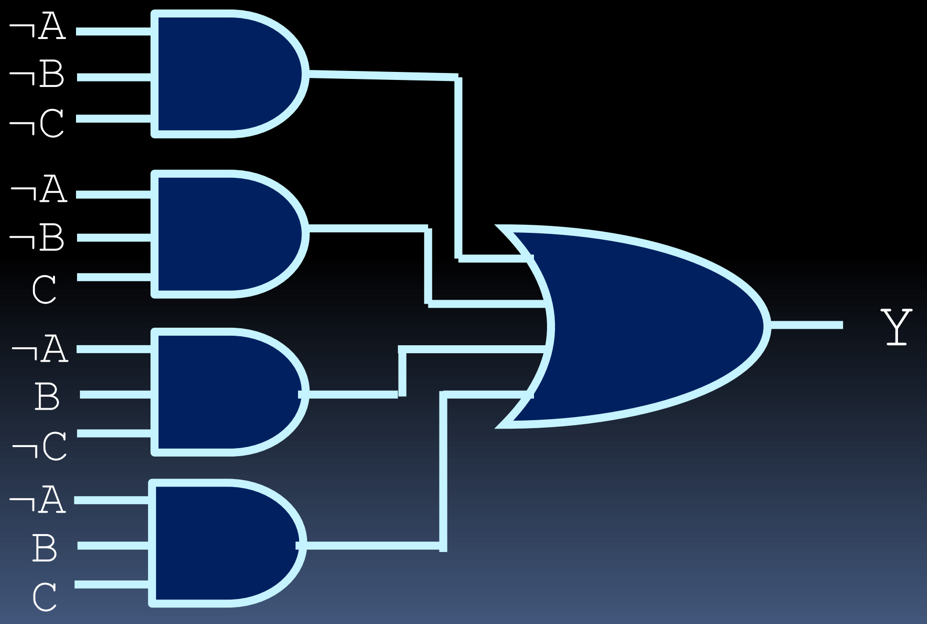
- Sum-of-Minterms is a way of expressing which inputs cause the output to go high. Product-of-Maxterms is a way of expression which inputs cause the output to go low.
  - Assumes that the truth table columns list the inputs according to some logical or natural order.
- Minterm and maxterm expressions are used for efficiency reasons:
  - More compact that displaying entire truth tables.
  - Sum-of-minterms are useful in cases with very few input combinations that produce high output.
    - Product-of-maxterms useful when expressing truth tables that have very few low output cases…

Converting SOM to gates

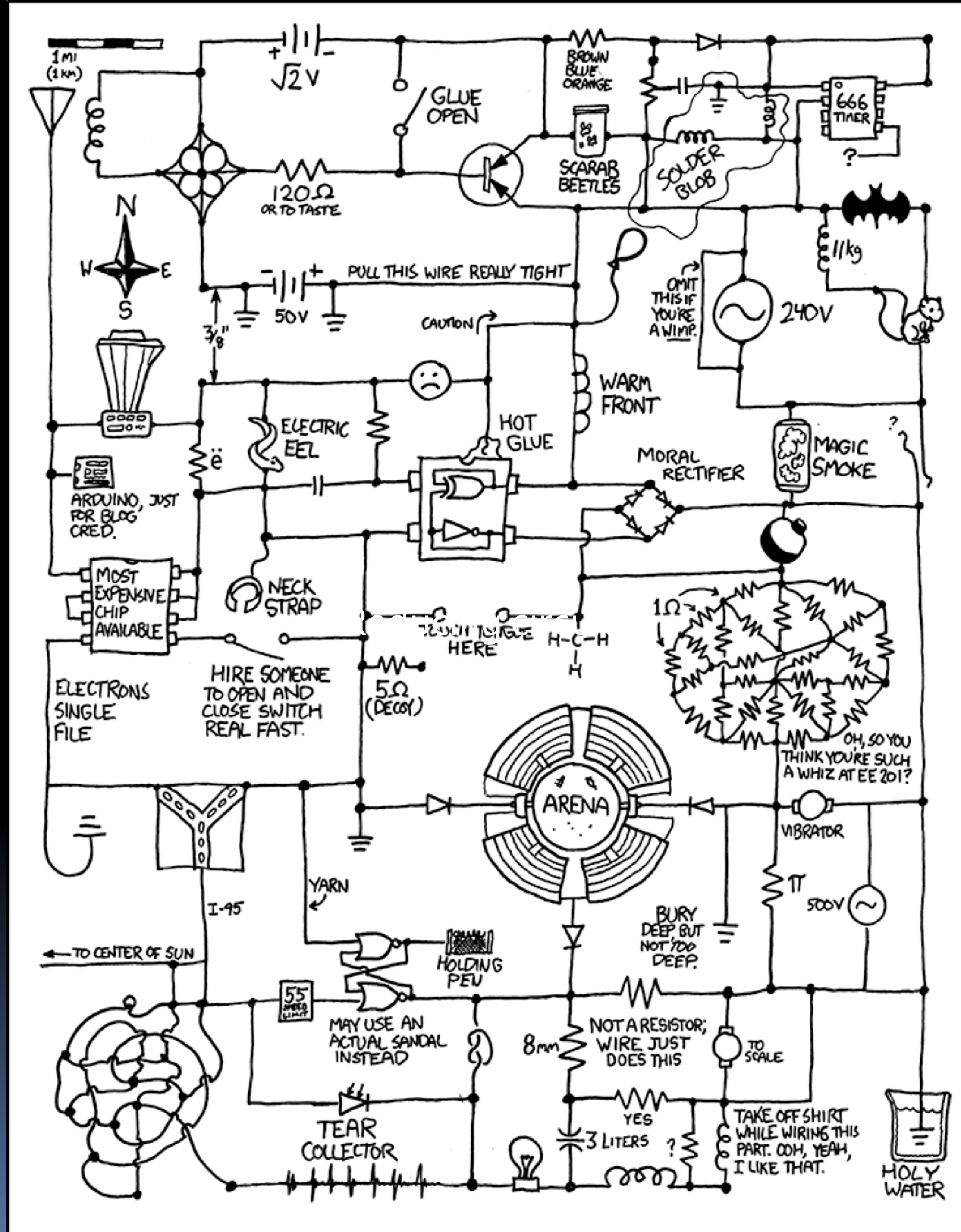
- Once you have a Sum-of-Minterms expression, it is easy to convert this to the equivalent combination of gates:

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

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

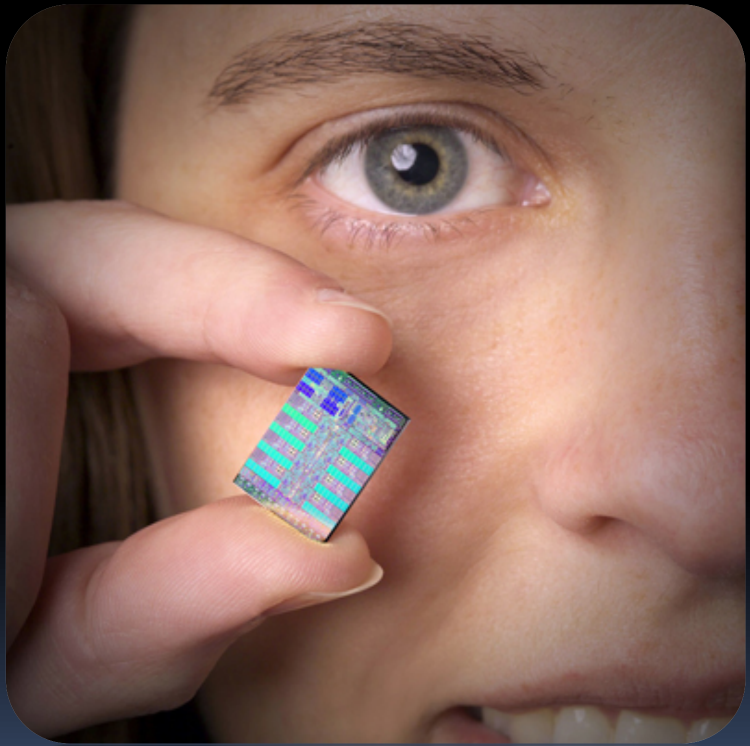


B
r
e
a
k



1 MI
(1 KM)
$\sqrt{2}$ V
GLUE OPEN
BROWN
BLUE
ORANGE
666 TIMER
?
120 Ω
OR TO TASTE
SCARAB BEETLES
SOLDER BLOB
N
W
E
S
50V
PULL THIS WIRE REALLY TIGHT
3/8"
CAUTION
OMIT THIS IF YOU'RE A WIMP.
240V
11kg
WARM FRONT
ELECTRIC EEL
HOT GLUE
ë
MORAL RECTIFIER
MAGIC SMOKE
?
ARDUINO, JUST FOR BLOG CRED.
MOST EXPENSIVE CHIP AVAILABLE
NECK STRAP
HERE
H–C–H
H
1Ω
ELECTRONS SINGLE FILE
HIRE SOMEONE TO OPEN AND CLOSE SWITCH REAL FAST.
5Ω
(DECOY)
OH, SO YOU THINK YOU'RE SUCH A WHIZ AT EE 201?
ARENA
VIBRATOR
YARN
I-95
π
500V
BURY DEEP BUT NOT TOO DEEP.
TO CENTER OF SUN
HOLDING PEN
55
MAY USE AN ACTUAL SANDAL INSTEAD
8mm
NOT A RESISTOR; WIRE JUST DOES THIS
TO SCALE
TEAR COLLECTOR
YES
3 LITERS
?
TAKE OFF SHIRT WHILE WIRING THIS PART. OOH, YEAH, I LIKE THAT.
HOLY WATER

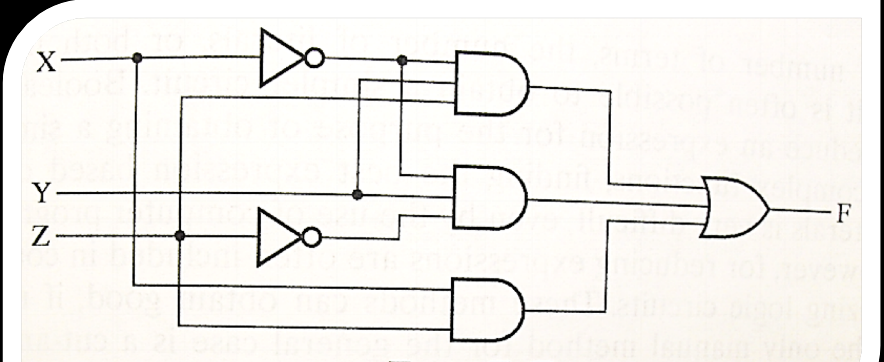
Reducing circuits



Which is Better?

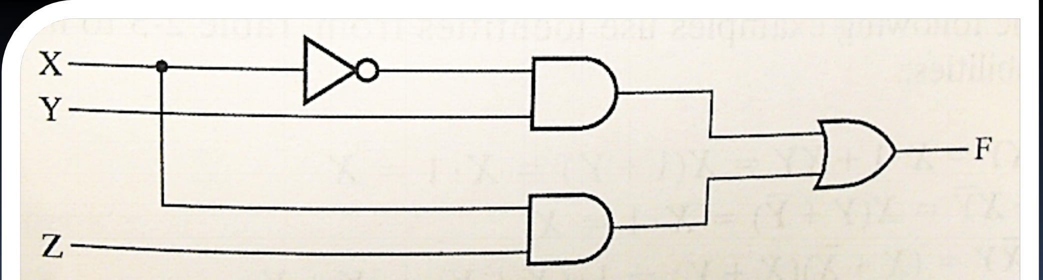
- Which implementation do you prefer? Why?

A.



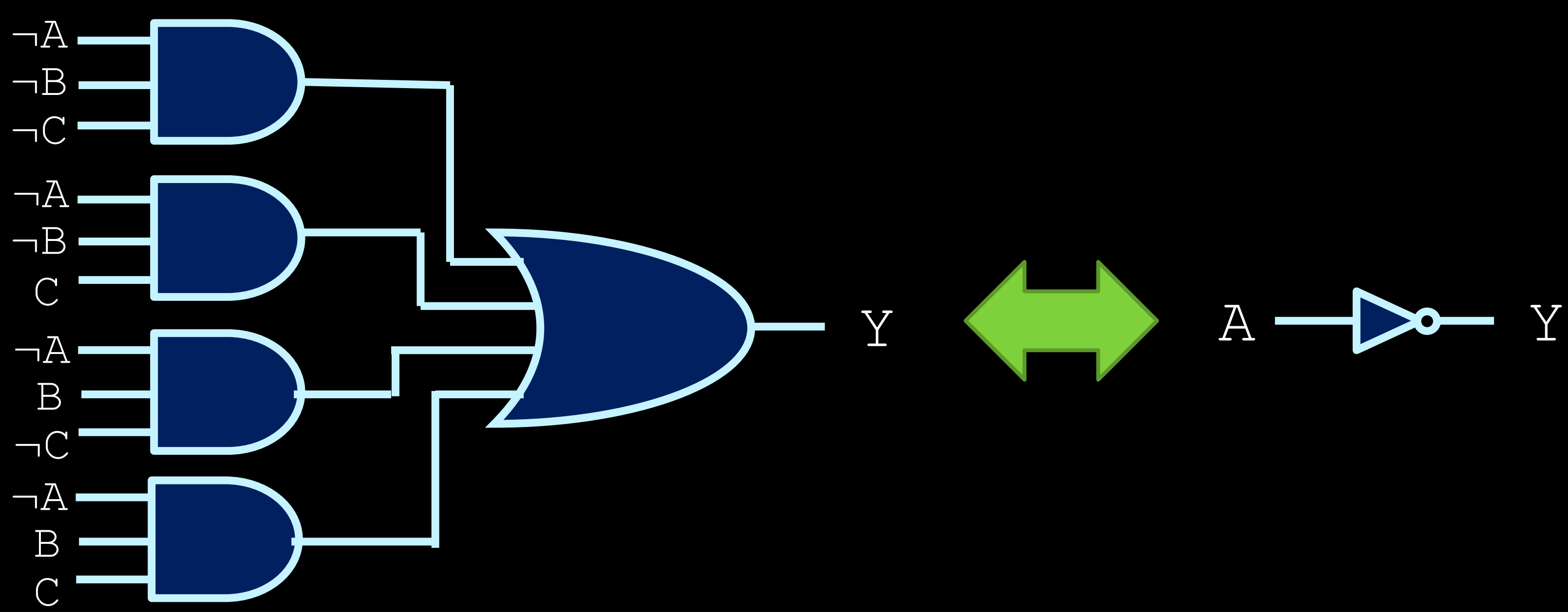
(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$

B.



(b) $F = \overline{X}Y + XZ$

Reasons for reducing circuits



- Note example of Sum-of-Minterms circuit design.
- 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 CSCA67 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:

If one input of a 2-input AND gate is 1, then the output is whatever value the other input is.

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

If one input of a 2-input OR gate is 0, then the output is whatever value the other input is.

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:

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

Does this hold in conventional algebra?

Other boolean identities

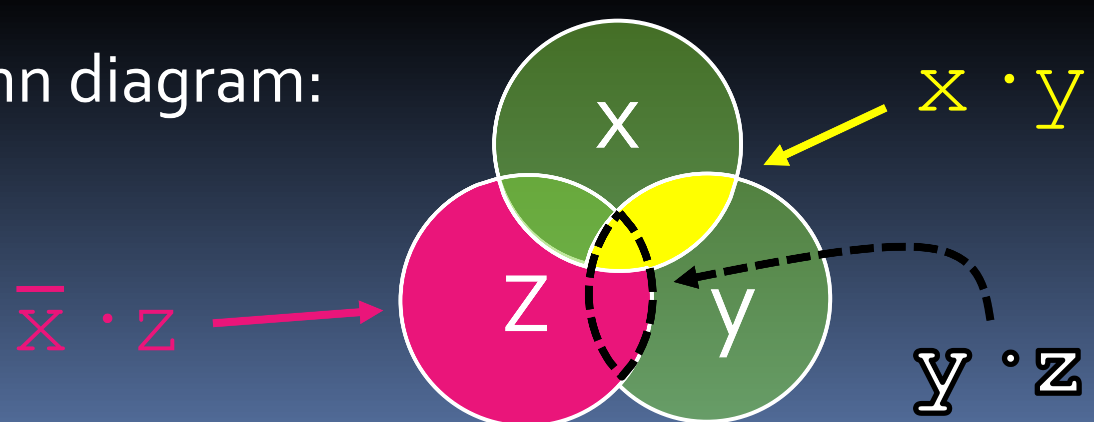
- Simplification Law:

$$x+(\overline{x}\cdot y) = x+y \qquad x\cdot(\overline{x}+y) = x\cdot y$$

- Consensus Law:

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

  - Proof by Venn diagram:



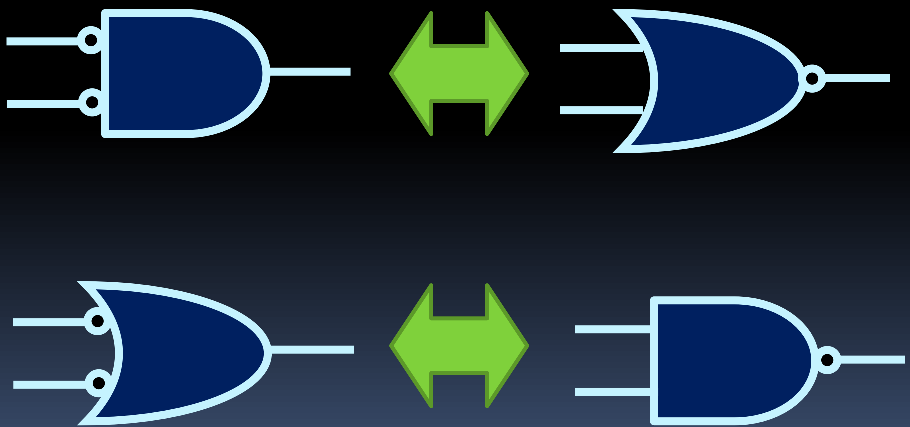
Other boolean identities

- Absorption Law:

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

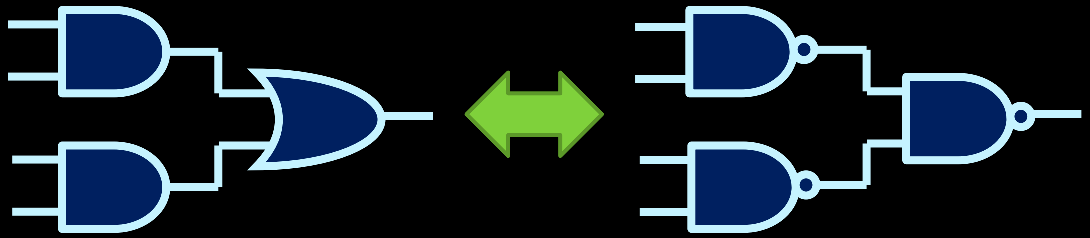
- De Morgan's Laws:

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

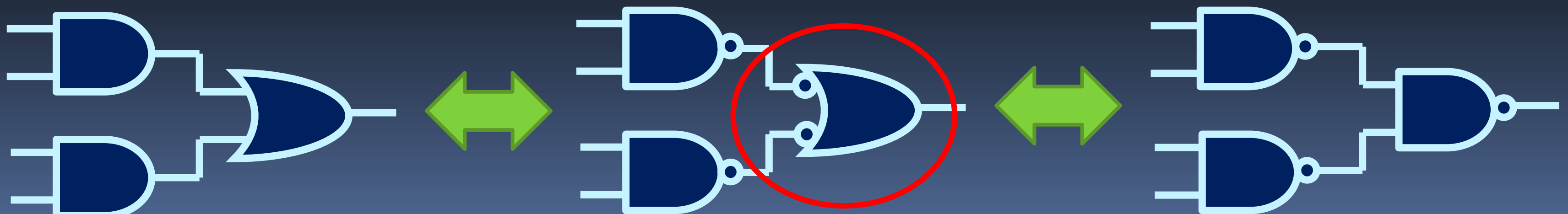


Converting to NAND gates

- 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

- Using SOM:

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

- Now start combining terms, like the last two:

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

Reducing Boolean expressions

- Different final expressions possible, depending on what terms you combine.
- For instance, given the previous example:

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

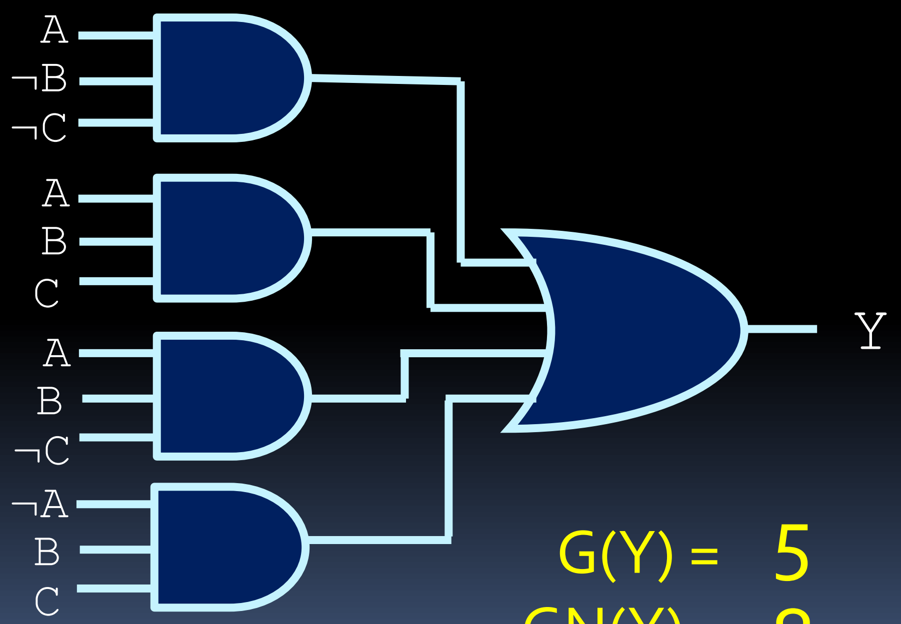
- If you combine the end and middle terms…

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

- Which reduces the number of gates and inputs!

Reducing Boolean expressions

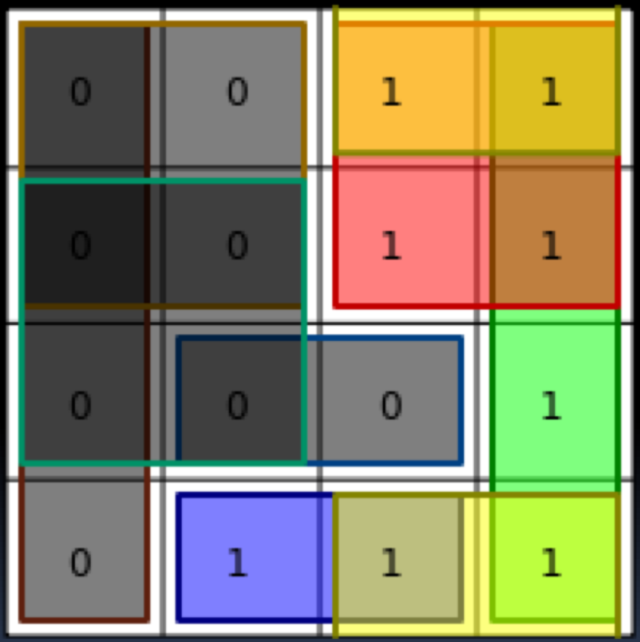
- What is considered the "simplest" expression?
  - In this case, "simple" denotes the lowest gate cost (G) or the lowest gate cost with NOTs (GN).
  - To calculate the gate cost, simply add all the gates together (as well as the cost of the NOT gates, in the case of the GN cost).
  - In this example the cost per gate is 1



G(Y) = 5

GN(Y) = 8

Karnaugh maps



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

Karnaugh maps

- Karnaugh maps can be of any size, and have any number of inputs.
  - 4 inputs here

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

- Since adjacent minterms only differ by a single value, they can be grouped into a single term that omits that value.

Using Karnaugh maps

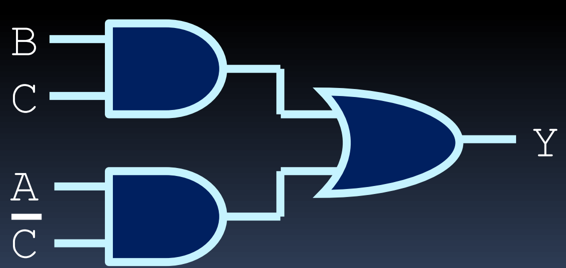
- Once Karnaugh maps are created, draw boxes over groups of high output values.
  - Boxes must be rectangular, and aligned with map.
  - Number of values contained within each box must be a power of 2.
  - Boxes may overlap with each other.
  - Boxes may wrap across edges of map.

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

Using Karnaugh maps

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

- Once you find the minimal number of boxes that cover all the high outputs, create Boolean expressions from the inputs that are common to all elements in the box.
- For this example:
  - Vertical box: $B \cdot C$
  - Horizontal box: $A \cdot \overline{C}$
  - Overall equation: $Y = B \cdot C + A \cdot \overline{C}$



Karnaugh maps and maxterms

- 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}

Quick Exercise

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

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

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

Circuit Creation Algorithm

- Understand desired behaviour
- Write truth table
- Write SOM (or POM) for truth table
- Simplify SOM using K-Map
- Translate simplified SOM into Circuits
- Celebrate!