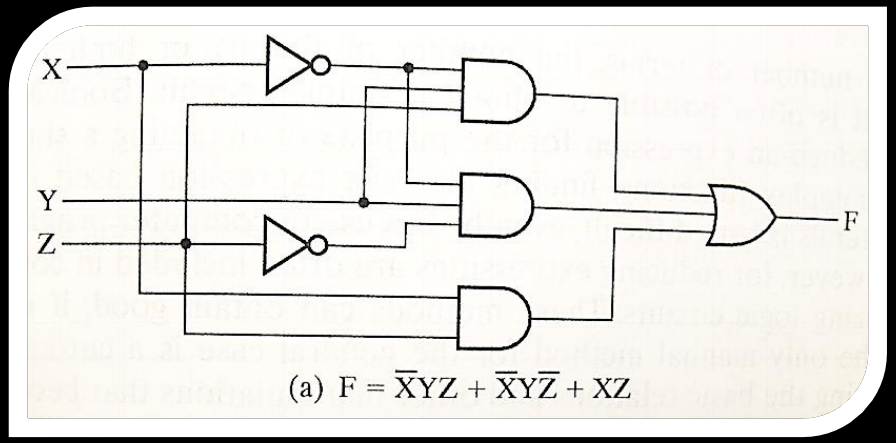
Week 2, Reducing circuits using Boolean algebra

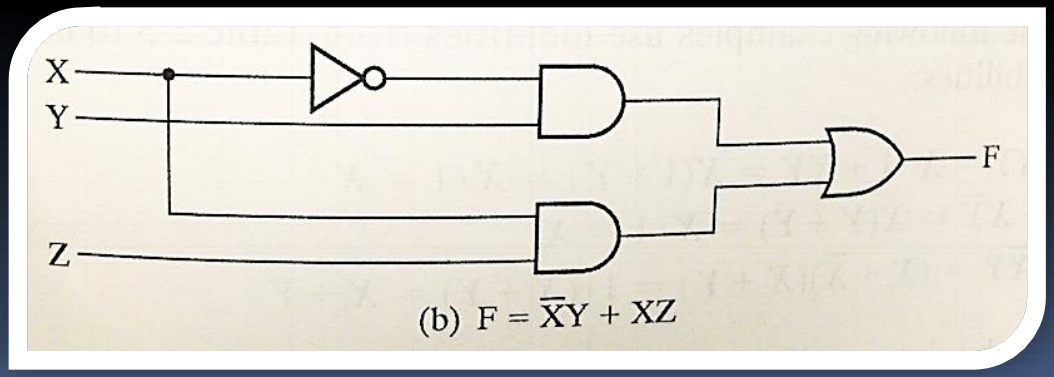
Which is Better?

- Which implementation do you prefer? Why?

A.



B.

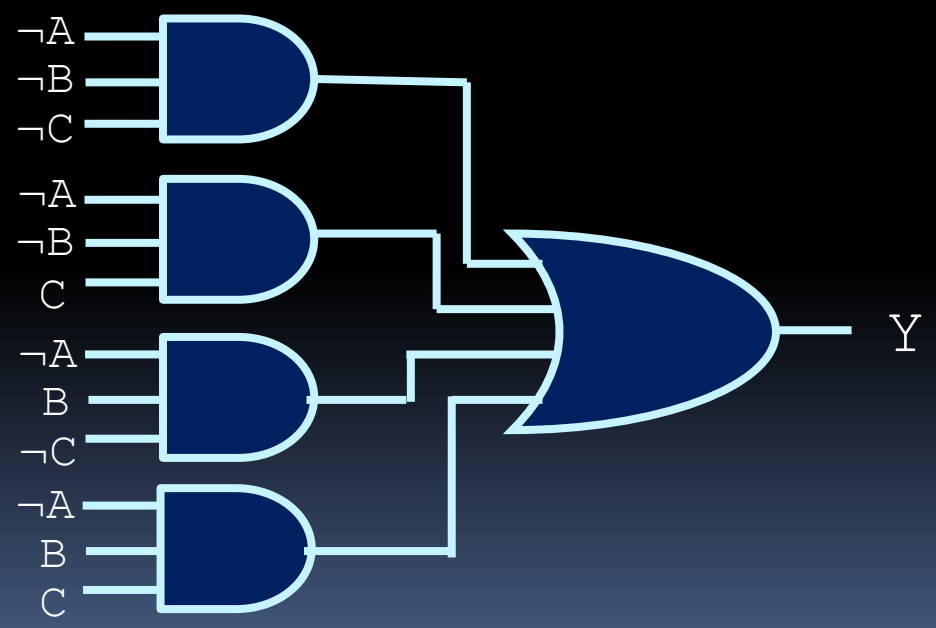


Remember this circuit?

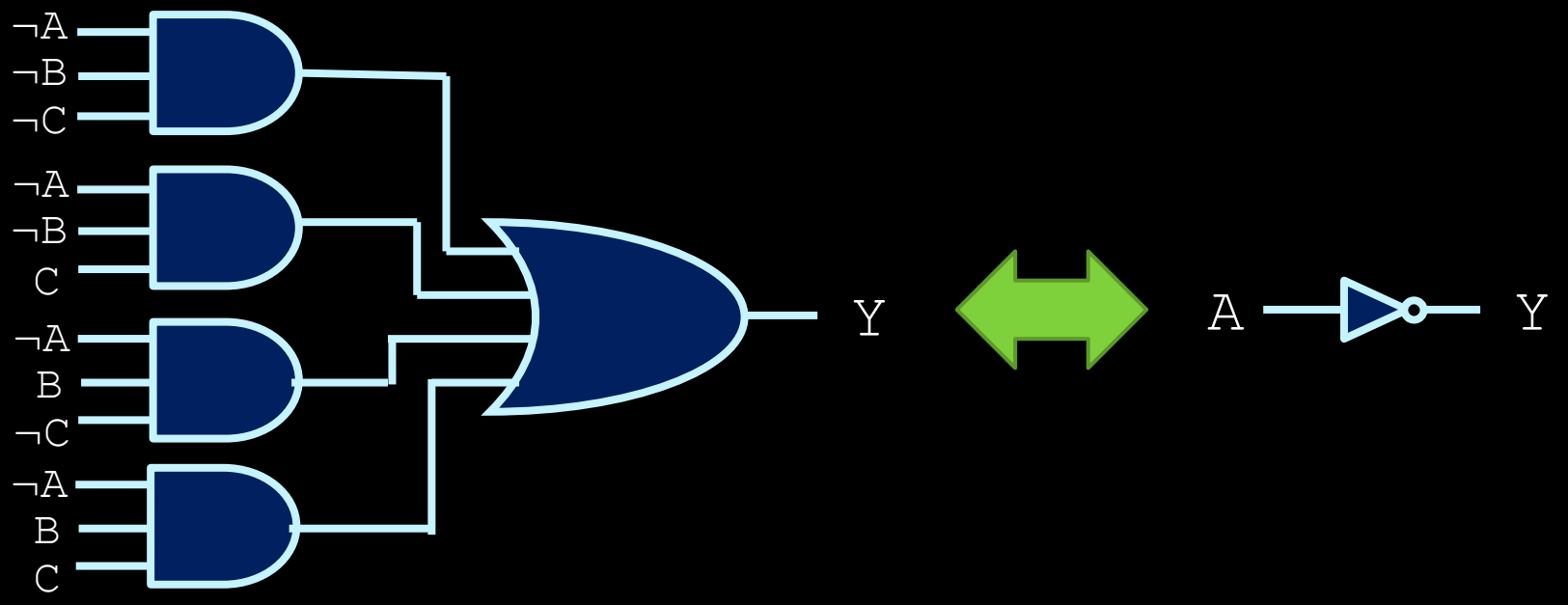
- We previously built this circuit using SOM
- Can we simplify it?

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

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



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 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.

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?

Boolean identities

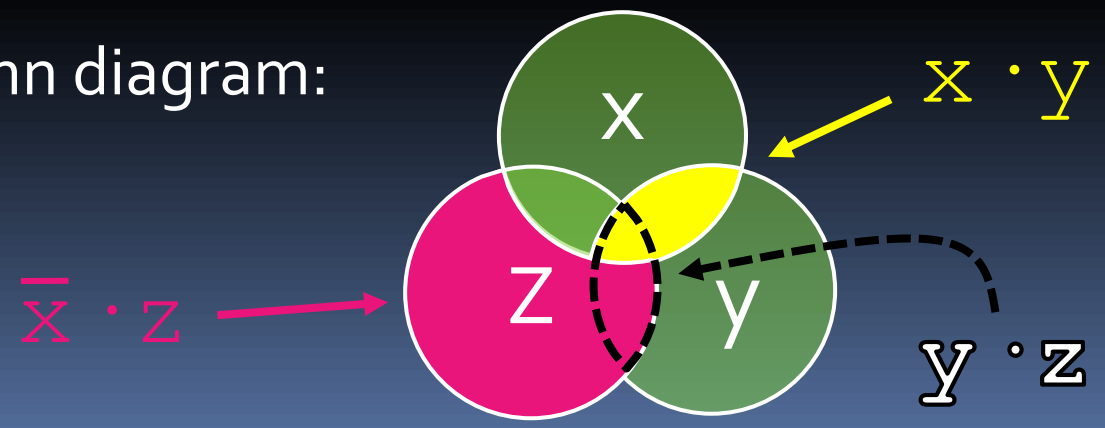
- Simplification Law:

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

- Consensus Law:

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

  - Proof by Venn diagram:



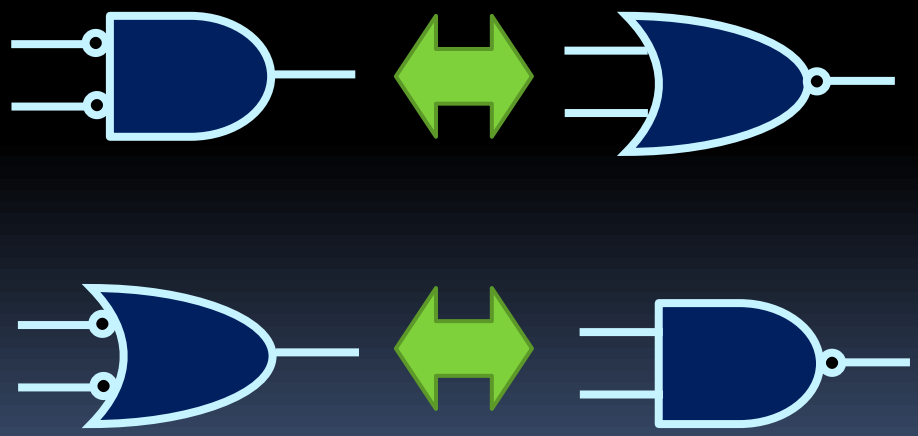
Boolean identities

- Absorption Law:

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

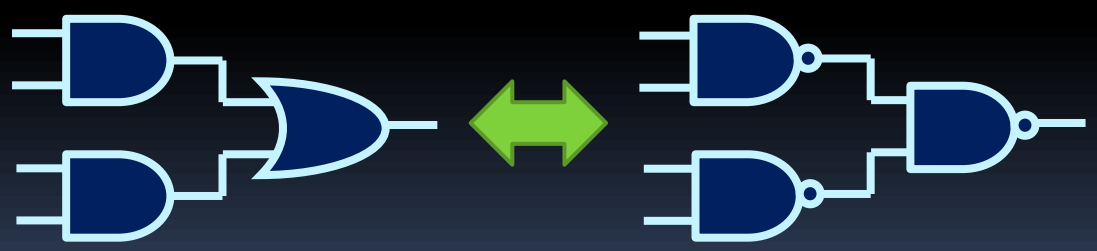
- De Morgan's Laws:

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



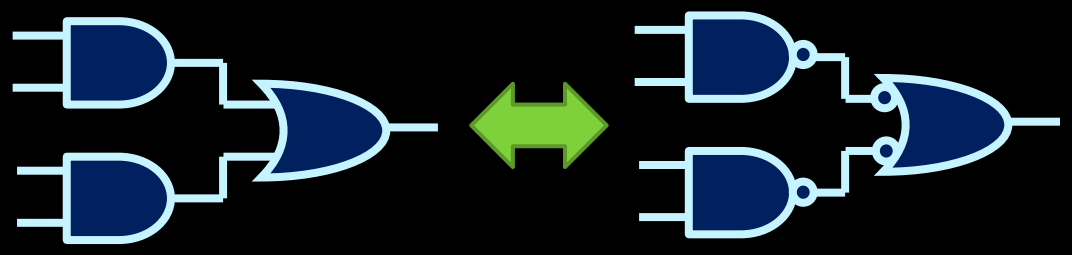
Converting to NAND gates

- De Morgan's Law is important because out of all the gates, NANDs are the cheapest to fabricate.
  - Can convert a Sum-of-Minterms (also known as sum-of-products) circuit to an equivalent circuit of NAND gates:

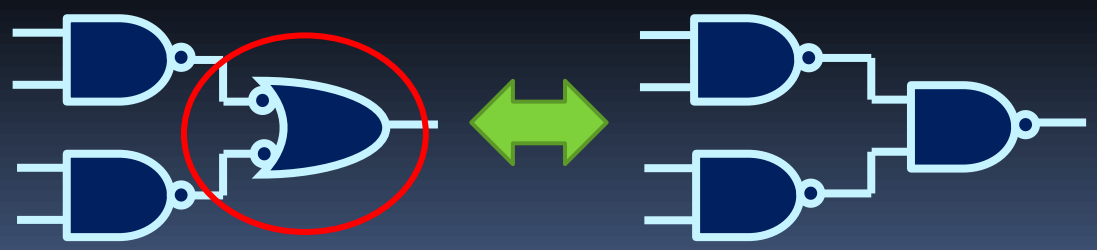


Converting SOM to NAND gates

- Start by adding two NOT gates on internal wires. Recall that `x = ~(~x))`



- Apply de Morgan's law to the final or gate: $\overline{x}+\overline{y} = \overline{x \cdot y} = \mathrm{NAND}(x,y)$



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:

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

Reducing Boolean expressions

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

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

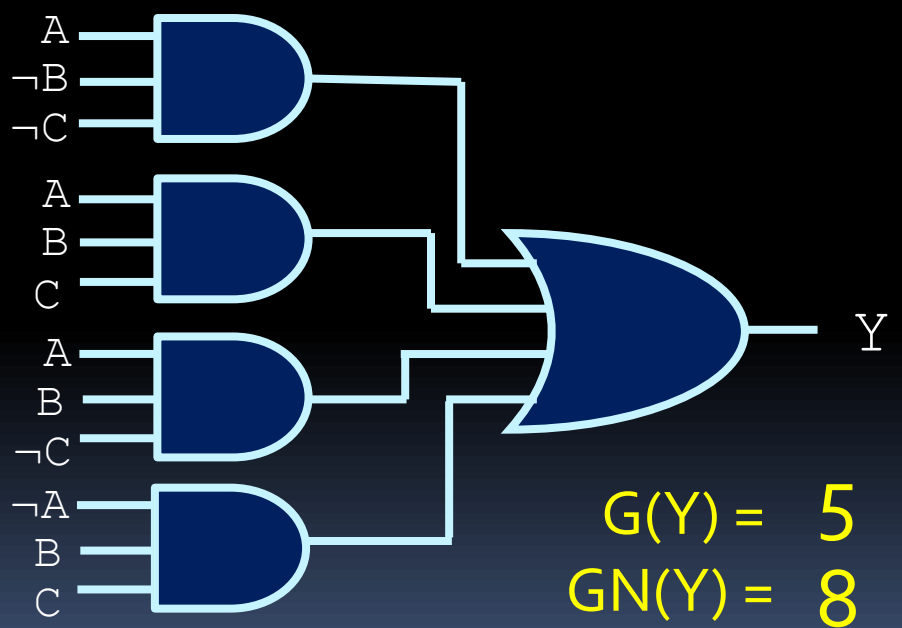
- If you combine the end and middle terms…

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

- 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

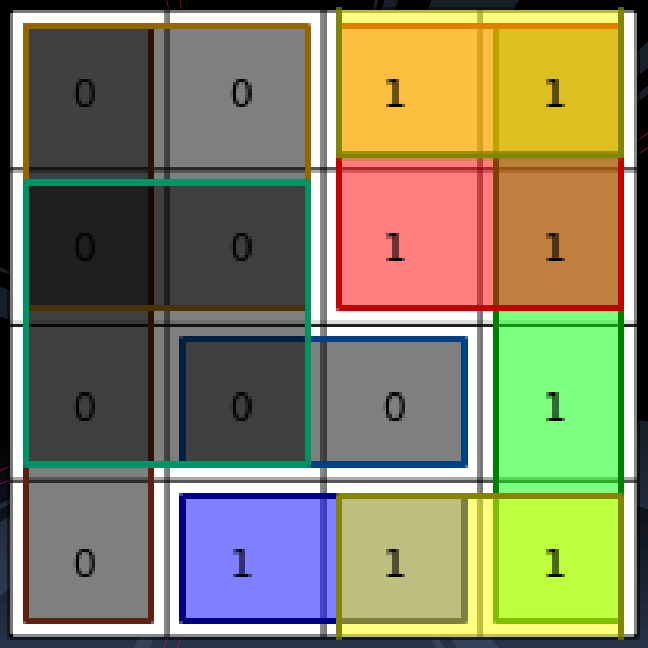
GN(Y) = 8

Don't count ¬C twice!

Algebraic Intuition?

- We don't like the fact that we need to rely on "intuition" to reduce using algebra.
  - What if we miss a step?
- Is there a simpler process that is repeatable?
- Next up, Karnaugh maps!

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 input (literal).
  - Values of the grid are the output for that minterm.

	00	01	11	10
	$\overline{B} \cdot \overline{C}$	$\overline{B} \cdot C$	$B \cdot C$	$B \cdot \overline{C}$
$\overline{A}$	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

		00	01	11	10
		$\bar{C}\cdot\bar{D}$	$\bar{C}\cdot D$	$C\cdot D$	$C\cdot\bar{D}$
00	$\bar{A}\cdot\bar{B}$	m_0	m_1	m_3	m_2
01	$\bar{A}\cdot B$	m_4	m_5	m_7	m_6
11	$A\cdot B$	m_{12}	m_{13}	m_{15}	m_{14}
10	$A\cdot\bar{B}$	m_8	m_9	m_{11}	m_{10}

- Once again, remember the trick to guarantee adjacent terms differ by one input:
  00 01 11 10

Karnaugh maps

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

Example:

$Y = m_{15} + m_{14}$

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

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

$= A \cdot B \cdot C$

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

Using Karnaugh maps

- Once Karnaugh maps are created, draw boxes over groups of high output values.
  - Boxes must cover all ones and cannot cover any zero.
  - Boxes must be rectangular, and aligned with map.
  - Size of each box (number of values it contains) must be a power of 2. (1,2,4,8,16,…)
  - 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 values contained in each grouping 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

Cover all the high values (1's) using the **smallest number** of valid groupings that are as **large** as possible.

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

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

So… how to find smallest expression

Cover all the high values (1's) using the **smallest number** of valid groupings that are as **large** as possible.

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

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

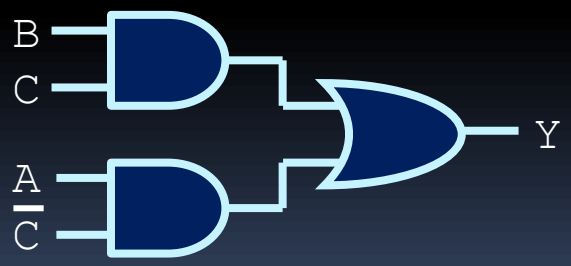
	$\bar{B}\cdot\bar{C}$	$\bar{B}\cdot C$	$B\cdot C$	$B\cdot\bar{C}$
$\bar{A}$	0	1	1	1
A	0	0	1	1

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

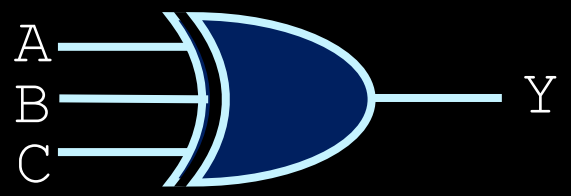
- $B\overline{C} + \overline{B}C\overline{D}$

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!

Karnaugh map review

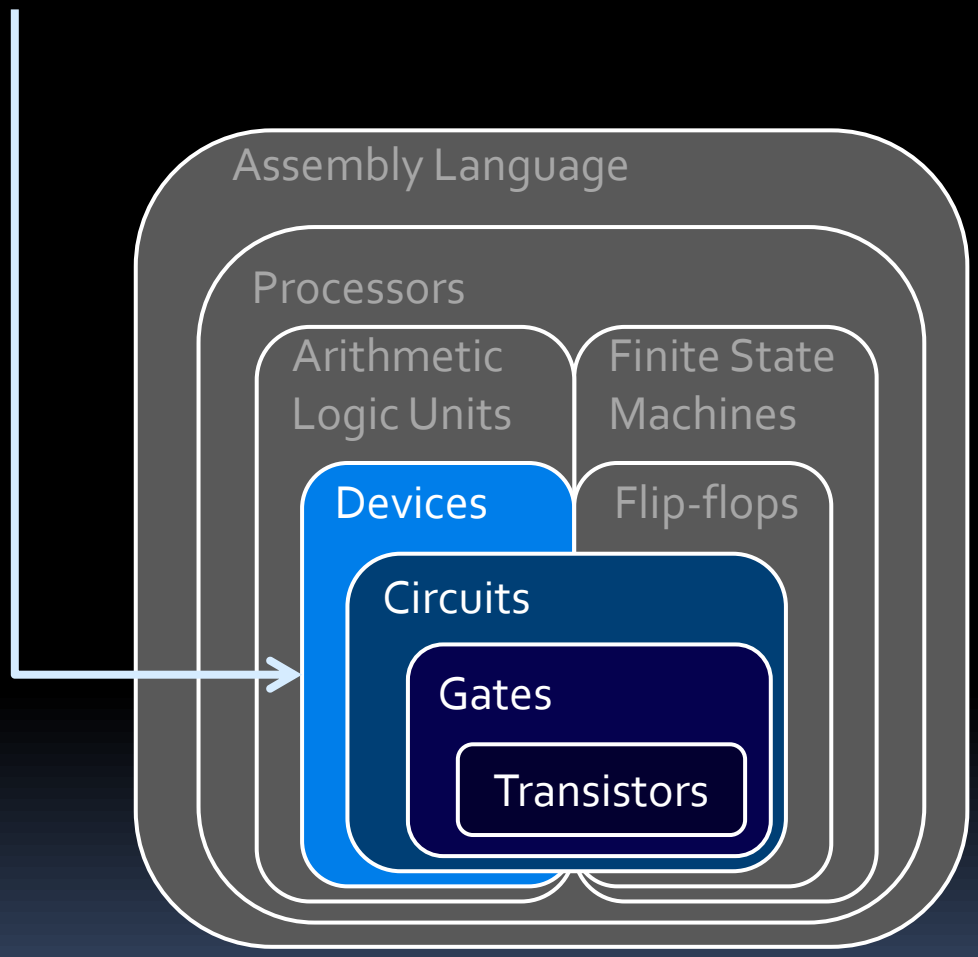
- <u>Note:</u> There are cases where no combinations are possible. K-maps cannot help these cases.
- <u>Example:</u> Multi-input XOR gates.



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

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

We are here


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

Building up from gates…

- Some common and more complex structures:
  - Multiplexers (MUX)
  - Adders (half and full)
  - Subtractors
  - Comparators
  - Decoders
    - Seven-segment decoders

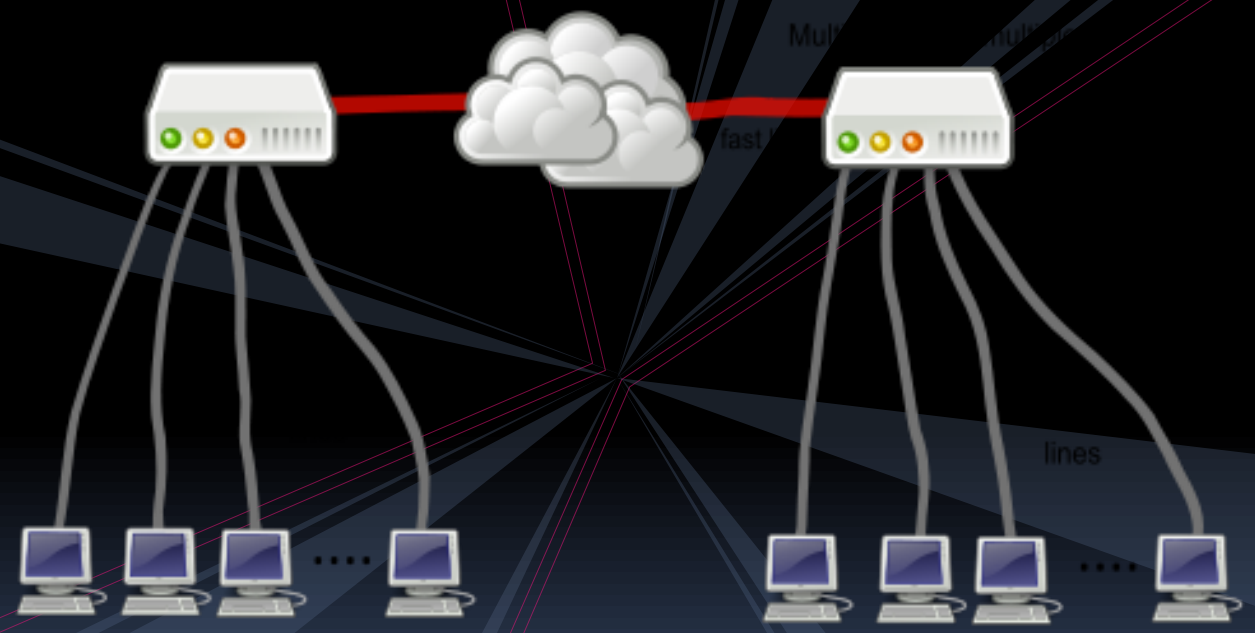
These are all combinational circuits

Combinational Circuits

- *Combinational Circuits* are any circuits where the outputs rely strictly on the inputs.
  - Everything we've done so far and what we'll do today is all combinational logic.

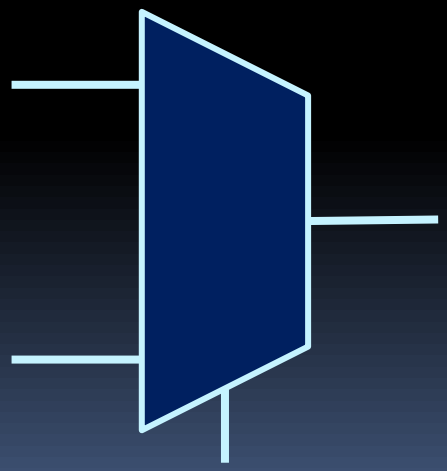
- Another category is *sequential circuits* that we will learn in the next few weeks.

Multiplexers



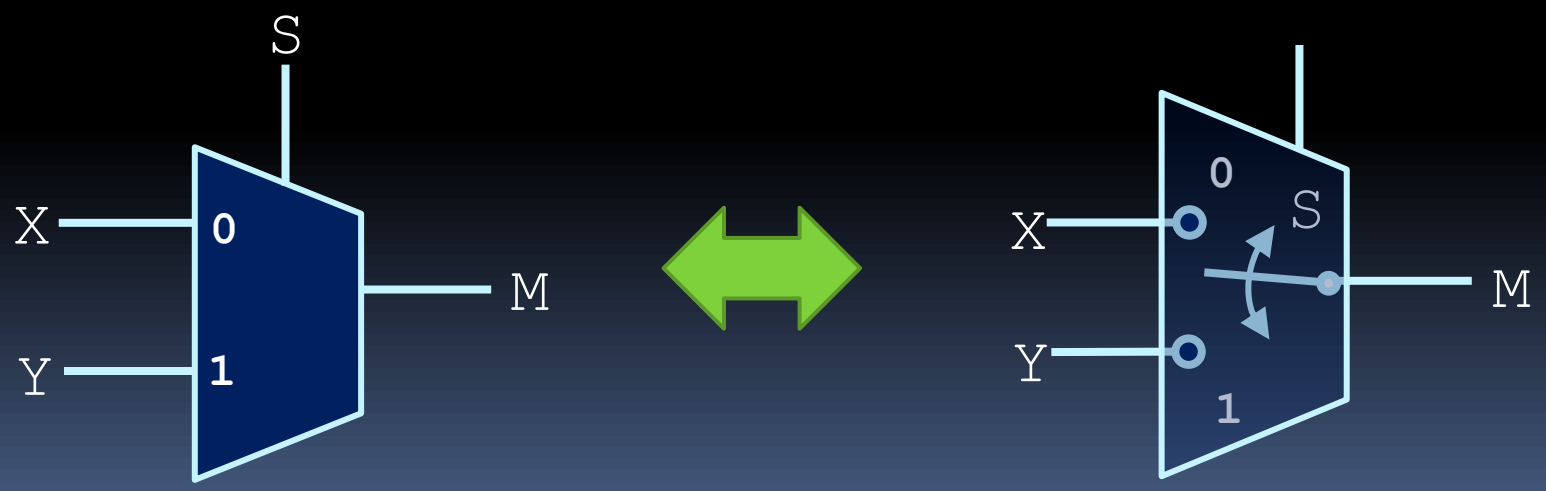
Mux Symbol

- Some circuits are so common to they have their own drawing.
- One of them is the multiplexor, or mux.



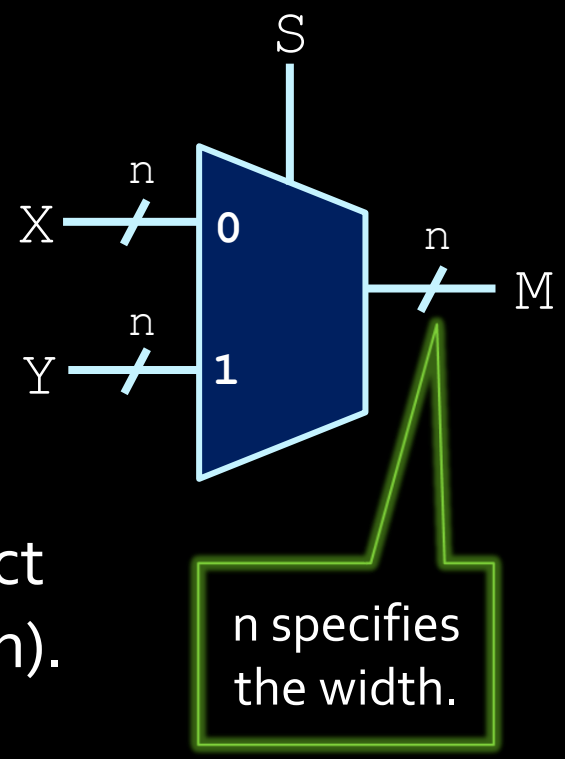
Multiplexer, or mux

- Switches between inputs:
  - Select one of multiple inputs.
  - Connect that input to the single output.
- A 2-to-1 mux will output `X` if `S` is `0`, and will output `Y` if `S` is `1`.



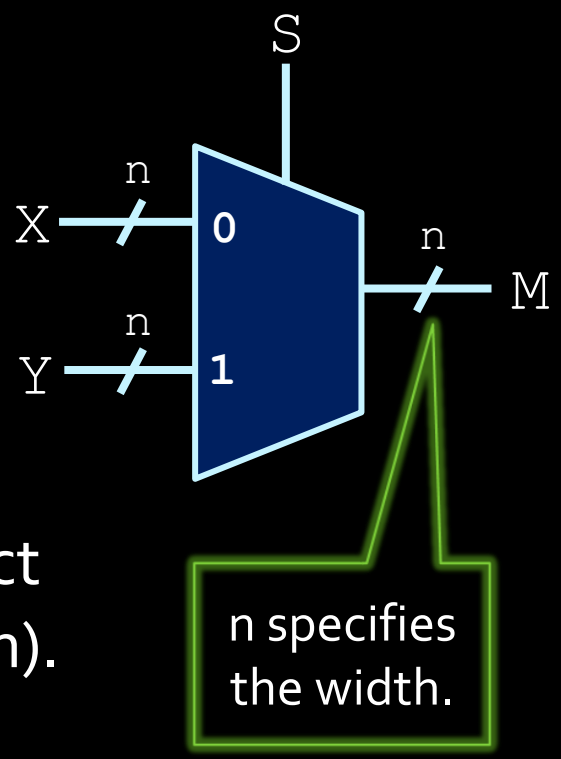
Multiplexer

- S is called the select input.
- X and Y are the data inputs.
- X and Y can have n data bits.
  - Note the number of select bits is distinct from the number of data bits (the width).

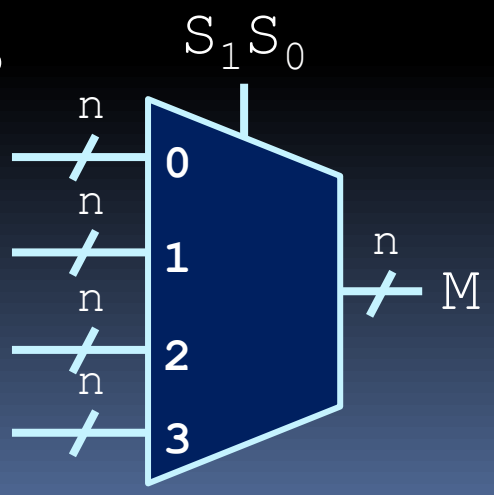


Multiplexer

- S is called the select input.
- X and Y are the data inputs.
- X and Y can have n data bits.
  - Note the number of select bits is distinct from the number of data bits (the width).



- A 4-to-1 mux would have 2 select bits
  - And as many data bits as we want!
- 8-to-1 mux → 3 select bits.



Multiplexer uses

- Muxes are very useful whenever you need to select from multiple input values.
- Your TV has at least one! You can select different input sources.
- More examples:
  - surveillance video monitors
  - digital cable boxes
  - routers.

Multiplexer design

X	Y	S	M
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Multiplexer design

X	Y	S	M
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

	$\bar{Y}\cdot\bar{S}$	$\bar{Y}\cdot S$	$Y\cdot S$	$Y\cdot\bar{S}$
$\bar{X}$	0	0	1	0
X	1	0	1	1

$$M = Y\cdot S + X\cdot\bar{S}$$