Boolean Algebra and Circuits

Overview

- Truth Tables
- Boolean Functions
- Logic Gates
- Combinatorial Circuits
- Appendix C (4th edition) or Appendix B (5th and 6th edition)

Truth Table

Boolean Operations

- Consider binary variables, A, B
  - Possible values are 0 (FALSE) and 1 (TRUE)
- Different possible inputs with n operands = 2^n
  - 2 binary operands: $2^2 = 4$ different possible inputs
  - 4 binary operands: $2^4 = 16$ different possible inputs

A
0
1

A	B

A	B	C

Boolean Operations

- Consider binary variables, A, B
  - Possible values are 0 (FALSE) and 1 (TRUE)
- Different possible inputs with n operands = 2^n
  - 2 binary operands: $2^2 = 4$ different possible inputs
  - 4 binary operands: $2^4 = 16$ different possible inputs

A
0
1

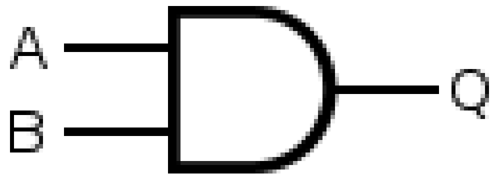
A	B
0	0
0	1
1	0
1	1

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

Boolean Operations

- **AND** operator
- **A · B** is true if A **and** B are both true
- Logical product

LOGIC GATE

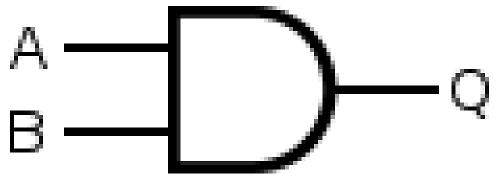


A	B	A · B
0	0	
0	1	
1	0	
1	1	

Boolean Operations

- **AND** operator
- **A · B** is true if A **and** B are both true
- Logical product

LOGIC GATE



A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Operations

- **OR** operator
- A + B is true if
  - A is true
  - B is true
  - A and B are both true
- Logical sum

LOGIC GATE

A	B	A + B
0	0	
0	1	
1	0	
1	1	

Boolean Operations

- **OR** operator
- A + B is true if
  - A is true
  - B is true
  - A and B are both true
- Logical sum

LOGIC GATE

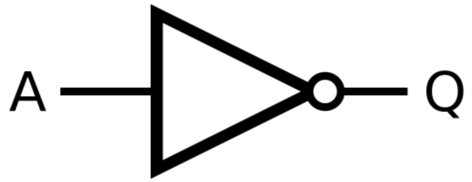
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Operations

- **NOT** operator
- $\bar{A}$ is true if A is false
- $\bar{A}$ is false if A is true

A	$\bar{A}$
0	1
1	0

LOGIC GATE



Boolean Operations

- A **Boolean Function** is an algebraic expression consisting of binary variables and the logic operation symbols
- Any Boolean function can be written in terms of **AND, OR, and NOT**
- Truth table defines all possible outcomes

A	B	A · B	A + B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Note on Notation

OR(A,B)	AND(A,B)	NOT(A)
A + B	A·B	$\bar{A}$
A | B	A & B	~A
A || B	A && B	!A
A v B	A ^ B	¬A
	AB	A’

When do we use which?

Boolean Algebra

identity	$A + 0 = A$	$A \cdot 1 = A$
one and zero	$A + 1 = 1$	$A \cdot 0 = 0$
inverse	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
commutative	$A + B = B + A,$	$A \cdot B = B \cdot A$
associative	$(A + B) + C = A + (B + C)$	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
distributive law:	$A \cdot (B + C) = A \cdot B + A \cdot C$	$(A \cdot B) + C = (A + C) \cdot (B + C)$
De Morgan	$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$

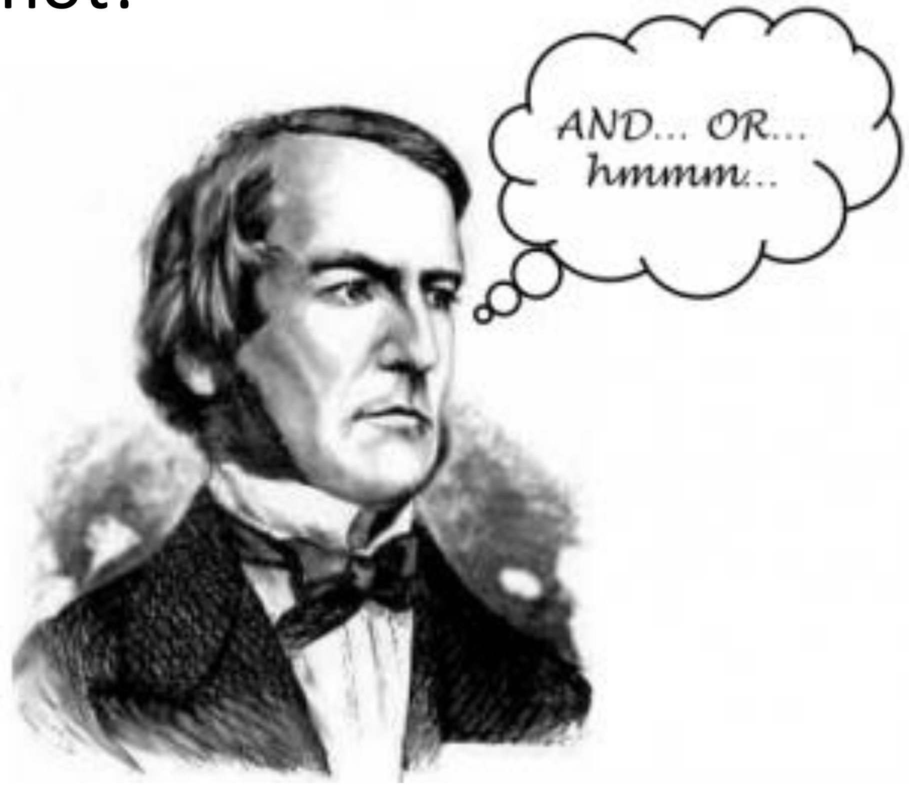
- Rules capture logical reasoning
- Allow for manipulation of expressions

Boolean Algebra

Are two boolean functions equivalent?
Prove it, or find values to show that they are not!

$A \cdot B \cdot C' + A \cdot C \cdot B' + B \cdot C \cdot A'$

$B \cdot (A \cdot C' + C \cdot A')$



Boolean Expression

Derive Boolean Expression

- Consider a logic function with three inputs, *A*, *B*, and *C*, and three outputs, *D*, *E*, and *F*.
  - *D* is true if at least one input is true
  - *E* is true if exactly two inputs are true
  - *F* is true only if all three inputs are true
- Find the Boolean expression for D, E, and F.

Derive Boolean Expression

Find the truth table! *D* is true if at least one input is true

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

Derive Boolean Expression

Find the truth table! *E* is true if exactly two inputs are true

A	B	C	D	E
0	0	0	0	
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	1	

Derive Boolean Expression

Find the truth table! *F* is true only if all three inputs are true

A	B	C	D	E	F
0	0	0	0	0	
0	0	1	1	0	
0	1	0	1	0	
0	1	1	1	1	
1	0	0	1	0	
1	0	1	1	1	
1	1	0	1	1	
1	1	1	1	0	

Derive Boolean Expression

Inputs			Outputs		
A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

What are the Boolean expressions for outputs D, E, and F?

Derive Boolean Expression

Inputs			Outputs		
A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

D is true if at least one input is true

D=

Derive Boolean Expression

Inputs			Outputs		
A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

F is true only if all three inputs are true

F =

Derive Boolean Expression

Inputs			Outputs		
A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

E is true if exactly two inputs are true

E =

Sum of Products, Product of Sums

- **Canonical form** for any logic function
  - Only two levels of gates, one AND, the other OR
  - Possible inversion on the final output.

$E = (A \cdot B \cdot C') + (A \cdot C \cdot B') + (B \cdot C \cdot A')$ Sum-of-Product

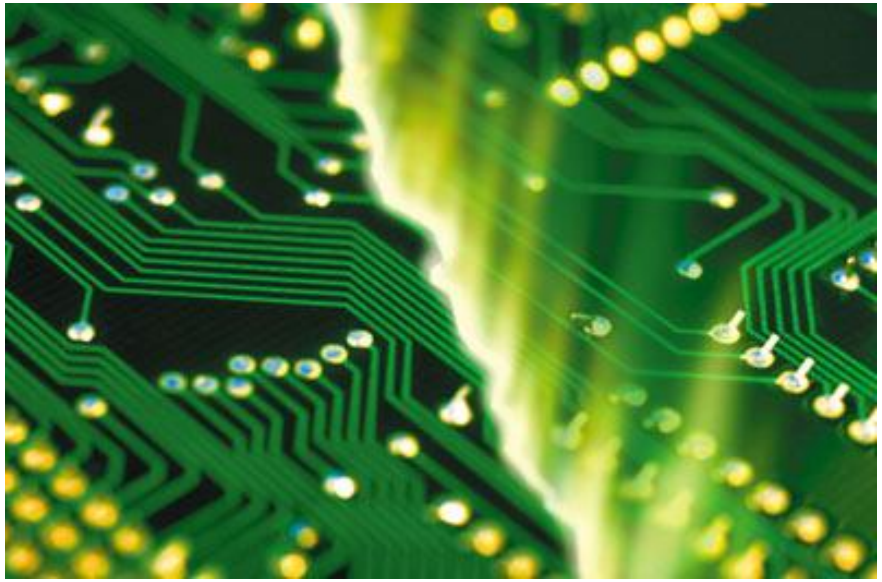
$E' = (A'+B'+C) \cdot (A'+C'+B) \cdot (B'+C'+A)$ Product-of-sum

- Why? Permits an easy/direct/naïve implementation of any Boolean function in hardware

Derive Boolean Expression

- How to derive a Boolean expression
  - Step 1: Write down the truth table
  - Step 2: Find the sum-of-product or product-of-sum
  - Step 3: Simplify it (optional)

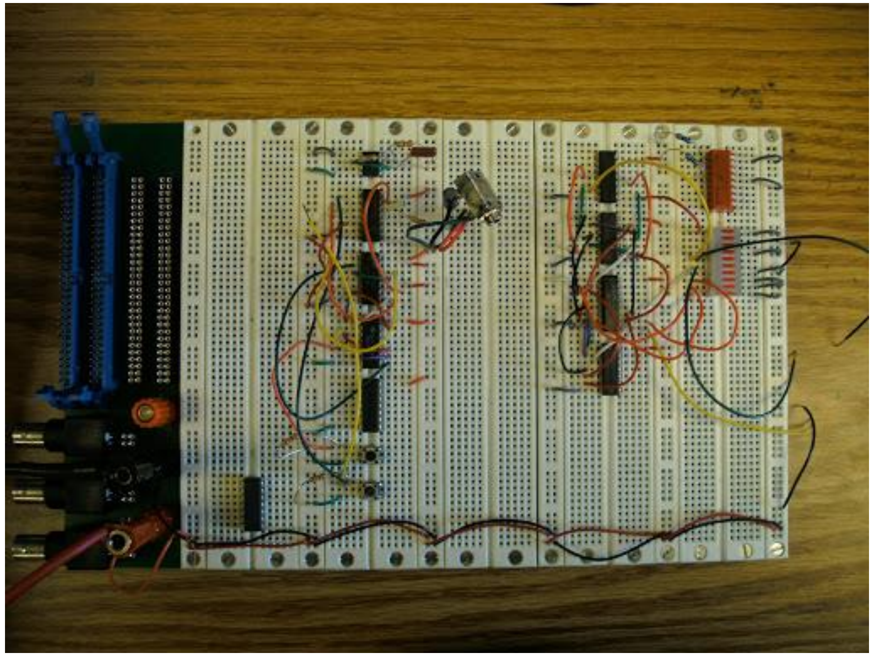
Circuits



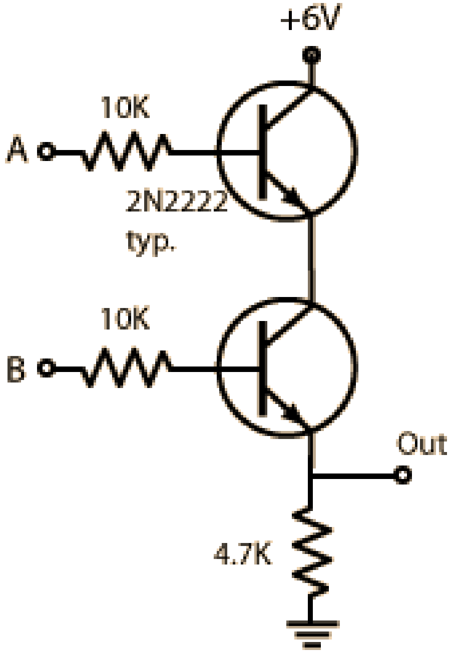
Logic Gates

- **Logic Gates:** Abstraction for building Boolean expressions in hardware
- Gates are made from **transistors**

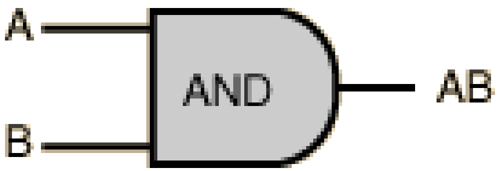
Electric Board



TRANSISTORS

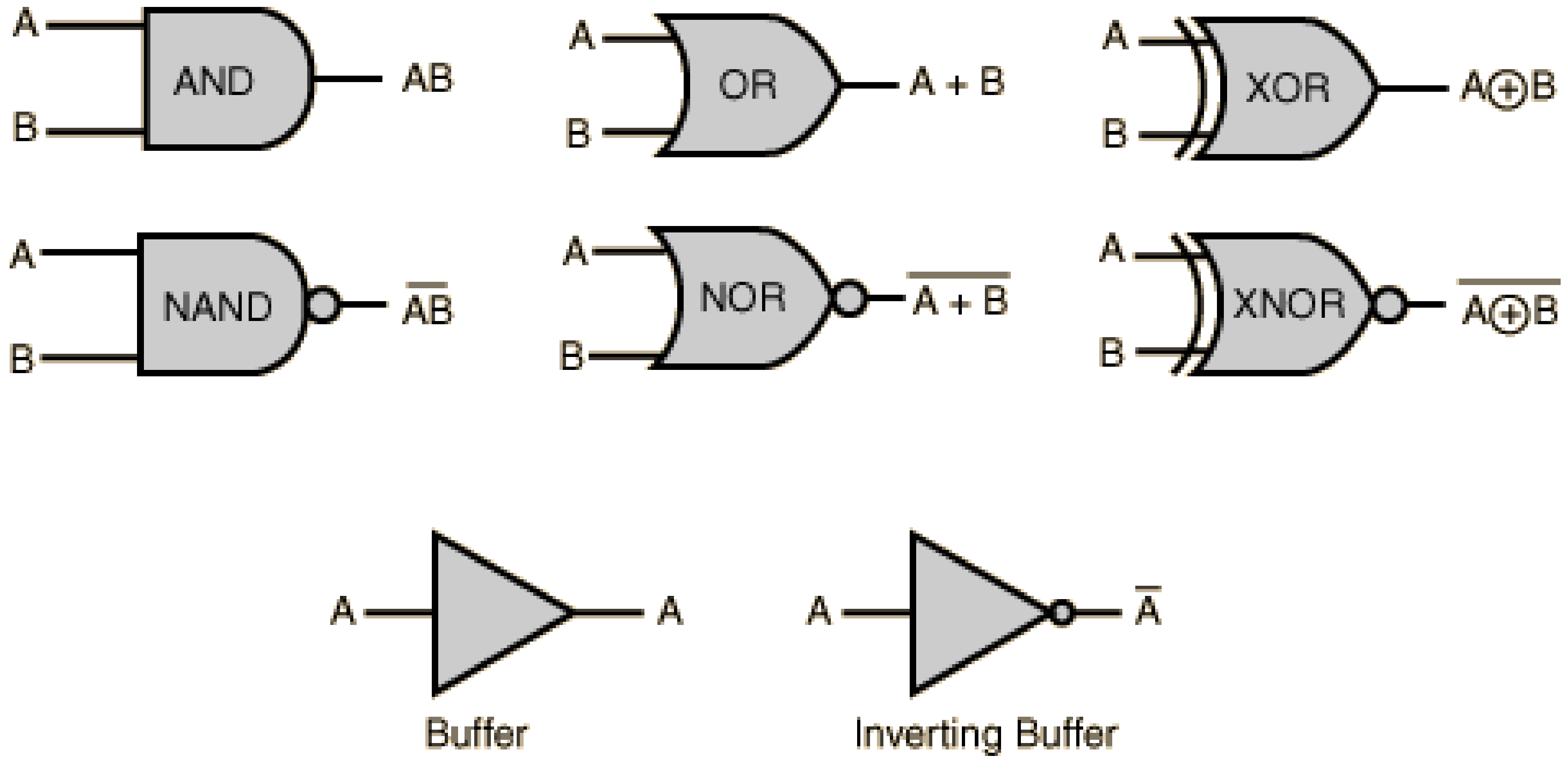


LOGIC GATE



Common Logic Gates

- AND, OR, XOR, and inverted versions



Common Binary Operations

- Three other very common operations:
  - NAND: Not And
  - NOR: Not OR
  - XOR: Exclusive OR

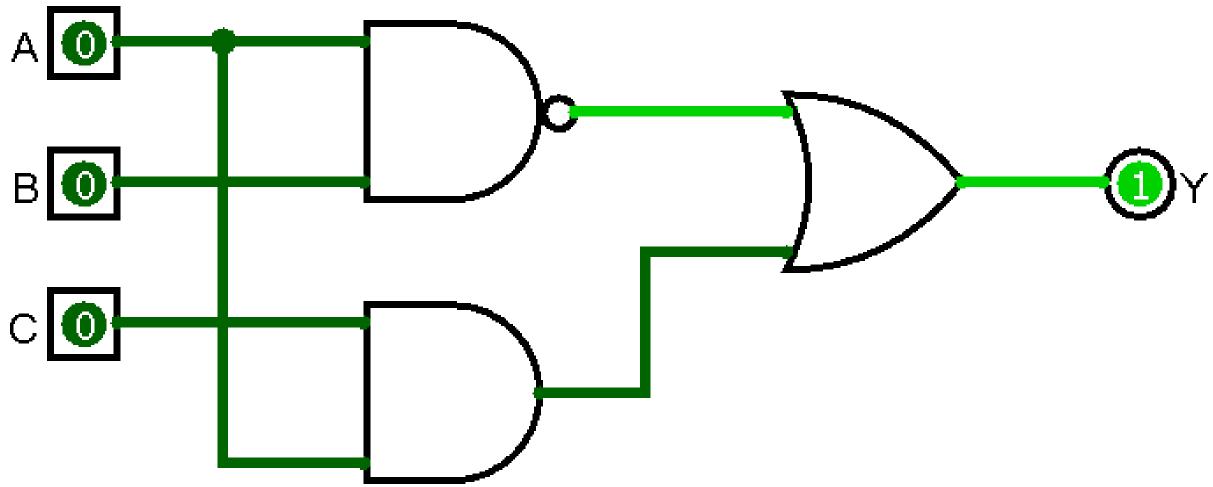
A	B	NAND	NOR	XOR
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

- Using AND, OR, NOT, how would you write expressions for each of these functions?

Combinational Circuits

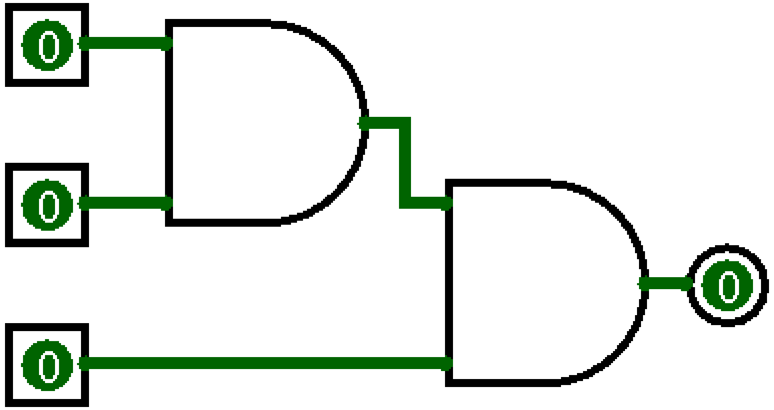
- **Combinatorial circuit:** a circuit that implements a Boolean function

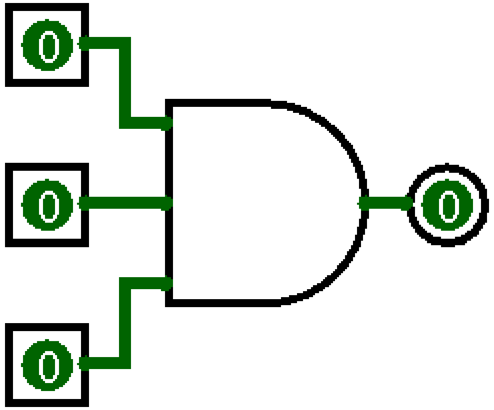
Draw the circuit for Y = (AB)’ + AC



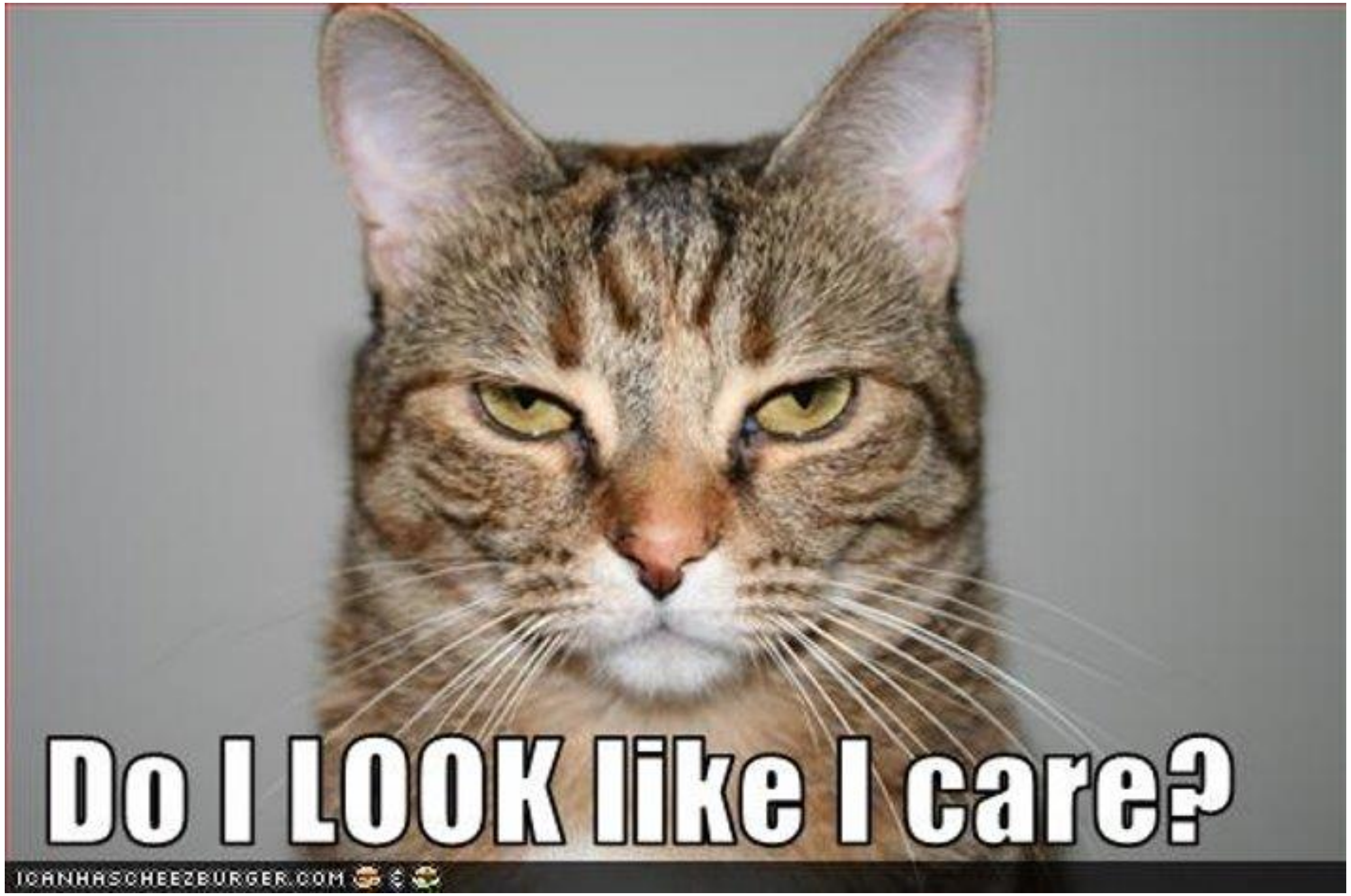
Multiple Inputs

- Can put > 2 inputs on an AND or and OR gate
- Allowed because of the associative law
- Commutative law of Boolean algebra: the ordering of the wires into a AND and OR gates doesn't matter
- Example: 3 input AND gate *(these functions are **symmetric**)*





Don't Care!



Don’t Care

Situations where we do not care about the value of an output

- Because another output is true

  e.g., If A is true, then A + B is true. We don't need to check B

  e.g., if A is false, then A . B is false. We don't need to check B

- Or we don’t care about the output for specific input compinations

Don’t Care

- Can be easier to specify the truth table
- Write the truth table, marking X for “Don’t Care”
  - Inputs S, A, B
  - Output C
  - If S is true, output is A
  - If S is false, output is B

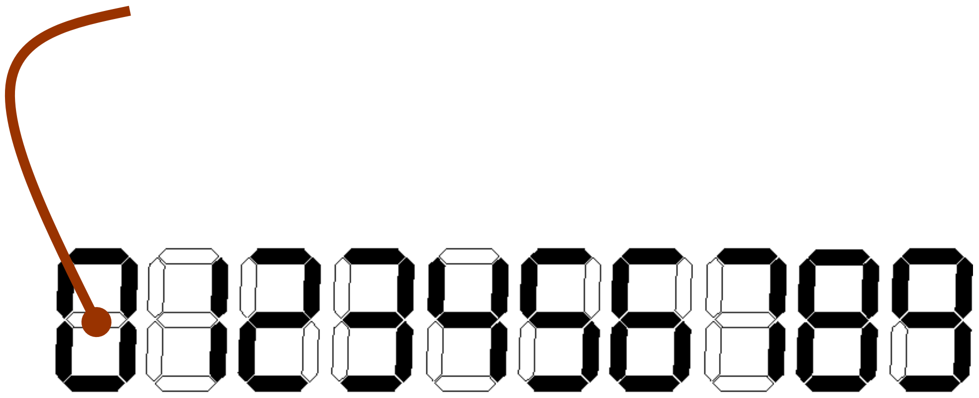
S	A	B	C
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

S	A	B	C
0	x	0	0
0	x	1	1
0	x	0	0
0	x	1	1
1	0	x	0
1	0	x	0
1	1	x	1
1	1	x	1

S	A	B	C
0	x	0	0
0	x	1	1
1	0	x	0
1	1	x	1

Don’t Care

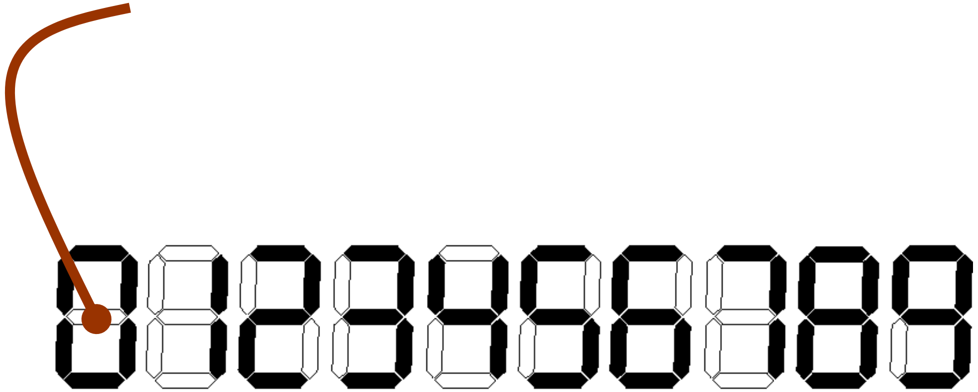
- We can also build simpler circuits by letting outputs take on any convenient value
  - Write the truth table for the middle segment of BCD to a 7-segment display decoder



A3	A2	A1	A0	m
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

Don't Care

- We can also build simpler circuits by letting outputs take on any convenient value
  - Write the truth table for the middle segment of BCD to a 7 segment display decoder



A3	A2	A1	A0	m
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	x	x
1	1	x	x	x

M = (A3'A2'A1'A0' + A3'A2'A1'A0 + A3'A2A1A0)'

Circuit Minimization

- The problem of obtaining the smallest logic circuit (Boolean formula) that represents a given Boolean function or truth table.
- The problem is believed to be ***hard***, but there are effective heuristics (Karnaugh maps, the Quine–McCluskey algorithm).
- Beyond the scope of this course…
  – See Randy H. Katz, "Contemporary Logic Design"

Review and more information

- Truth Tables
- Boolean Functions
- Logic Gates
- Combinatorial Circuits
- The best thing about a Boolean function is that even if you are wrong, you are only off by a bit

- This class covers topics in textbook
  - Appendix C of 4th edition
  - Appendix B of 5th edition