EECS 270 W25
Discussion 3

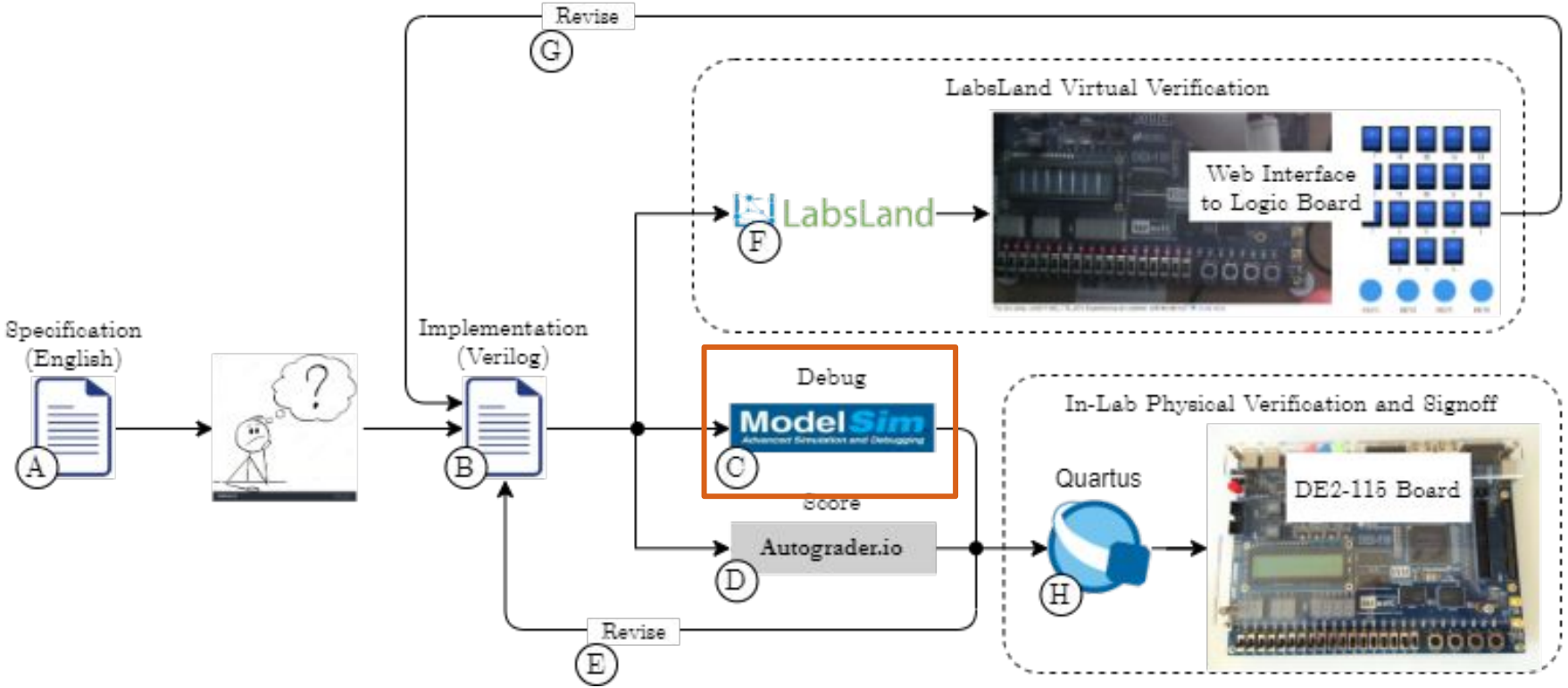
January 23th, 2025
By: Mick Gordinier

Please fill out the Google form if you haven't already! —>
(Link Here As Well)

1. Discussion 2 Recap
2. Boolean Algebra Deep Dive
3. Project 3A “Robbie” Discussion
4. (Bonus) More Algebraically Proving Theorems Walkthrough
5. (Bonus) From Switches to Boolean Algebra
6. (Bonus) Behavioral Verilog Introduction
   (Will be the focus for next week)

Discussion 2 Recap

Overview of the Tools



Why Do We Need to Simulate?

- Main Goal: Need to ensure our code behaves the correct way
  - Verifying the robustness of our design
  - **BEFORE FABRICATING THE PHYSICAL HARDWARE**

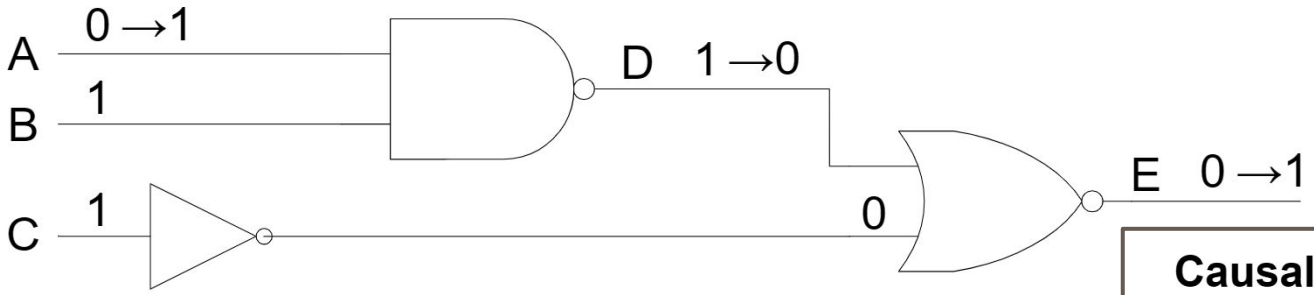
- If we don’t simulate before testing on the board
  - The possible issues could be endless

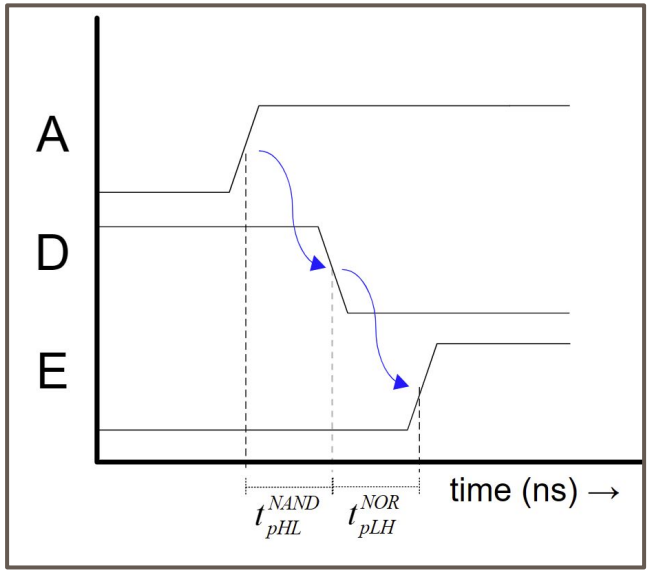
Output Changes are NOT Immediate

- Logical circuitry is built using physical wires and transistors
  - Each have some internal delay
  - In real implementations, need to account for delay

- **Propagation Delay** - Time it takes for the input change to be reflected onto the output

Causality Graphs and Terminology





Causality Arrow

Gate Propagation Delay

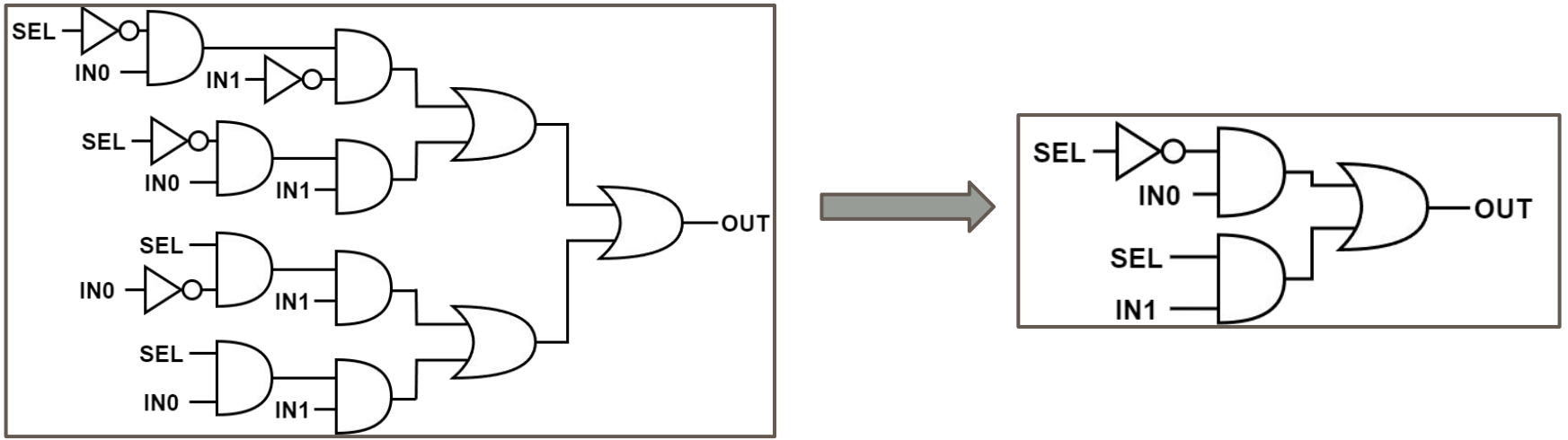
t_{pHL}^{NAND}

gives NAND gate propagation delay from input to output when output is changing from H to L

Boolean Algebra Deep Dive

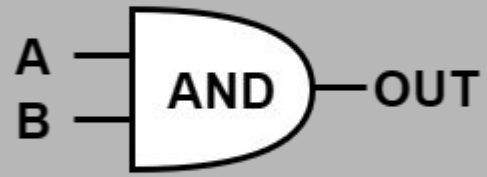
Why Should We Care About Boolean Algebra

- Extremely helpful in finding how we can optimize our logic / circuit!
- **Building and optimizing circuits by doing math**



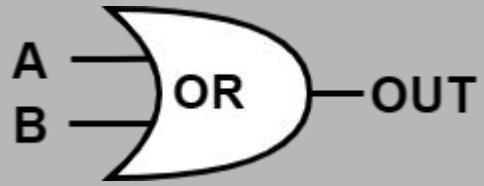
The Big 3 Operations - AND, OR, NOT

AND



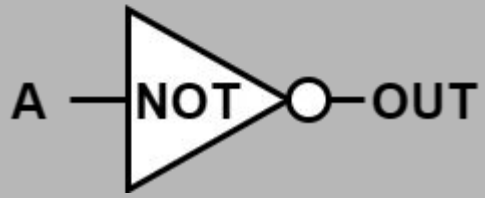
A	B	OUT
0	0	0
0	1	0
1	0	0
1	1	1

OR



A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	1

NOT



A	OUT
0	1
1	0

Axioms of Boolean / Switching Algebra

- Mostly defining how the big three operations works

Postulate	Defines	A	B
P1	Switching Variables	$x = 0$ iff $x \neq 1$	$x = 1$ iff $x \neq 0$
P2	NOT	$0' = 1$	$1' = 0$
P3		$0 \cdot 0 = 0$	$1 + 1 = 1$
P4	AND / OR	$1 \cdot 1 = 1$	$0 + 0 = 0$
P5		$0 \cdot 1 = 1 \cdot 0 = 0$	$0 + 1 = 1 + 0 = 1$

Let's Cover Some Basics (X in a Binary Variable)

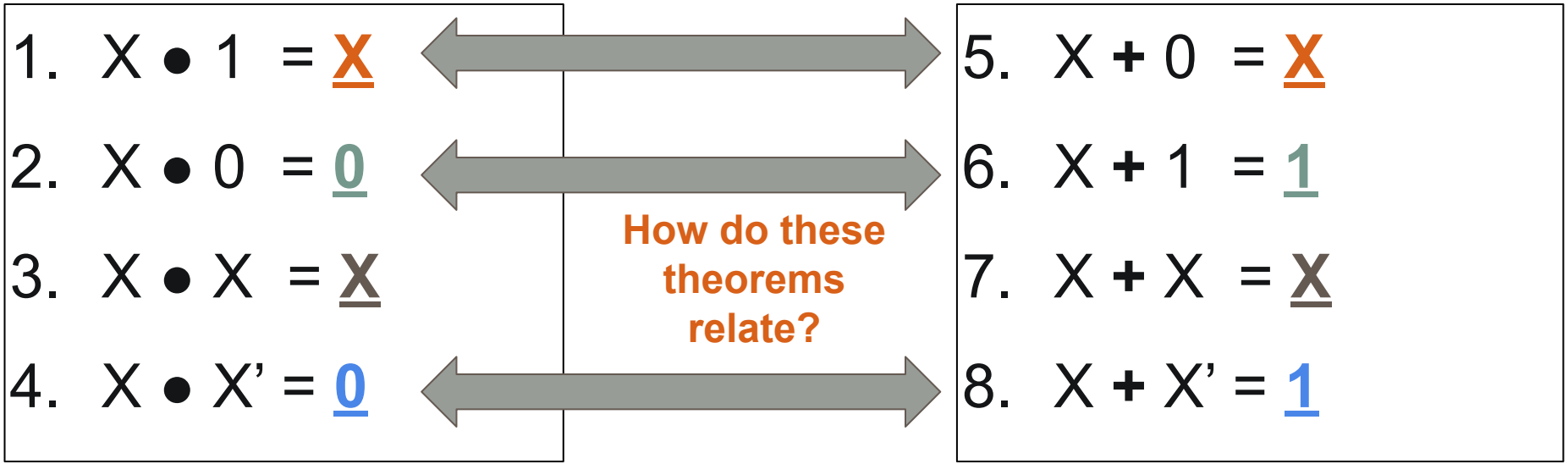
1. $X \bullet 1 =$ ____
2. $X \bullet 0 =$ ____
3. $X \bullet X =$ ____
4. $X \bullet X' =$ ____

5. $X + 0 =$ ____
6. $X + 1 =$ ____
7. $X + X =$ ____
8. $X + X' =$ ____

9. $X'' =$ ____

10. $X''' =$ ____

Let's Cover Some Basics (X in a Binary Variable)



9. X'' = X

10. X''' = X'

Duality (For Proving Theorems)

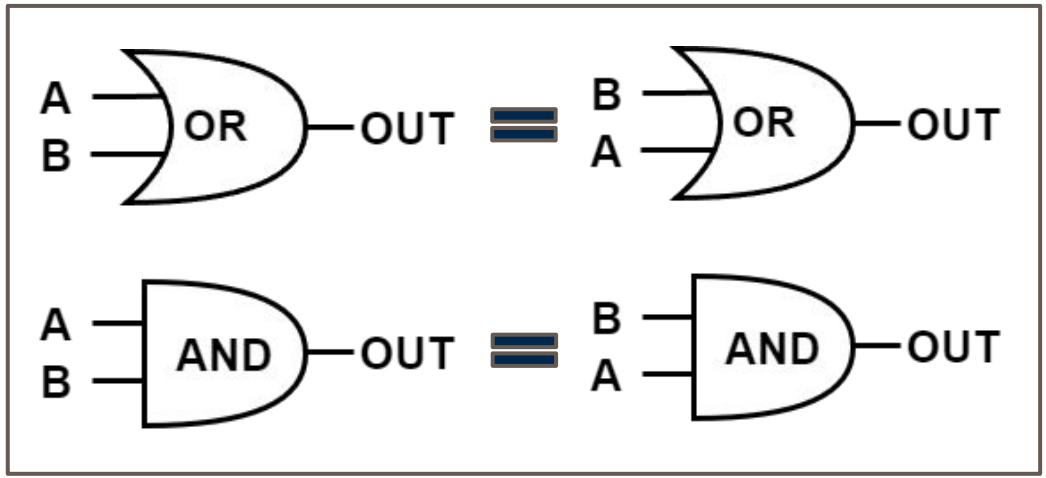
	A	Name	B
T1	$x \cdot 1 = x$	Identities	$x + 0 = x$
T2	$x \cdot 0 = 0$	Null Elements	$x + 1 = 1$
T3	$x \cdot x = x$	Idempotency	$x + x = x$
T4		Involution $(x')' = x$	
T5	$x \cdot x' = 0$	Complements	$x + x' = 1$

- Dual of an Equation - Swap Constants 0 <--> 1 and ● <--> +
  - **NOT inverting literals X <--> X'**

- **If an equation is true, then the dual of the equation is true as well**
  - A and B are duals of each other

More Core Boolean Algebra Theorems

	A	Name	B
T6	$x \cdot y = y \cdot x$	Commutativity	$x + y = y + x$
T9	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	Associativity	$(x + y) + z = x + (y + z)$
T10	$x \cdot (y + z) = x \cdot y + x \cdot z$	Distributivity	$x + (y \cdot z) = (x + y) \cdot (x + z)$



De Morgan's Laws (Really Good One to Know)

- How we distribute a function inversion

T12	De Morgan's $f(x_1, \ldots, x_n, 0,1,\cdot,+)' = f(x_1', \ldots, x_n', 1,0,+,\cdot)$

$$(X \bullet Y)' = X' + Y'$$

$$(X + Y)' = X' \bullet Y'$$

$$
\begin{aligned}
&(X \bullet (X + Y))' \\
&= X' + (X + Y)' \\
&= X' + (X' \bullet Y') \\
&= (X' + X') \bullet (X' + Y') \\
&= (X') \bullet (X' + Y') = \mathbf{X'}
\end{aligned}
$$

$$
\begin{aligned}
&[(W \bullet Y \bullet 1) + ((Z \bullet Y) + 0)]' \\
&= (W \bullet Y \bullet 1)' \bullet ((Z \bullet Y) + 0)' \\
&= (W' + Y' + 0) \bullet ((Z' + Y') \bullet 1) \\
&= (W' + Y') \bullet (Z' + Y') \\
&= \mathbf{Y' + (W' \bullet Z')}
\end{aligned}
$$

	A	Name	B
T1	$x \cdot 1 = x$	Identities	$x + 0 = x$
T2	$x \cdot 0 = 0$	Null Elements	$x + 1 = 1$
T3	$x \cdot x = x$	Idempotency	$x + x = x$
T4		Involution $(x')' = x$	
T5	$x \cdot x' = 0$	Complements	$x + x' = 1$
T6	$x \cdot y = y \cdot x$	Commutativity	$x + y = y + x$
T7	$x \cdot (x + y) = x$	Absorption	$x + (x \cdot y) = x$
T8	$x \cdot (x' + y) = x \cdot y$	No Name	$x + (x' \cdot y) = x + y$
T9	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	Associativity	$(x + y) + z = x + (y + z)$
T10	$x \cdot (y + z) = x \cdot y + x \cdot z$	Distributivity	$x + (y \cdot z) = (x + y) \cdot (x + z)$
T11	$x \cdot y + x' \cdot z + y \cdot z = x \cdot y + x' \cdot z$	Consensus	$(x + y) \cdot (x' + z) \cdot (y + z) = (x + y) \cdot (x' + z)$
T12	De Morgan's $f(x_1, \dots, x_n, 0, 1, \cdot, +)' = f(x_1', \dots, x_n', 1, 0, +, \cdot)$		

Let’s Practice!

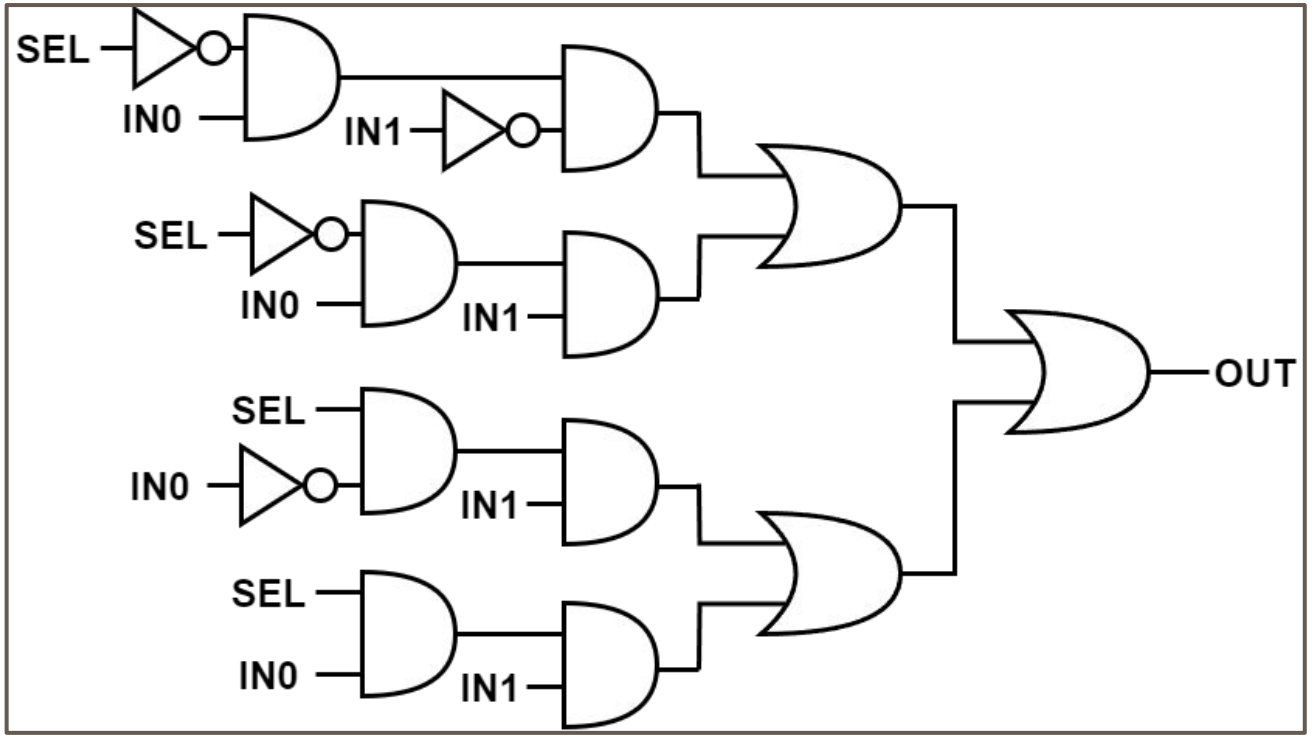
OUT = (~SEL ● ~IN1 ● IN0) + (~SEL ● IN1 ● IN0) + (SEL ● IN1 ● ~IN0) + (SEL ● IN1 ● IN0)

Look familiar?

How many primitive logic gates does the require?

Let's Practice!

OUT = [(~SEL • ~IN1 • IN0) + (~SEL • IN1 • IN0)] + [(SEL • IN1 • ~IN0) + (SEL • IN1 • IN0)]

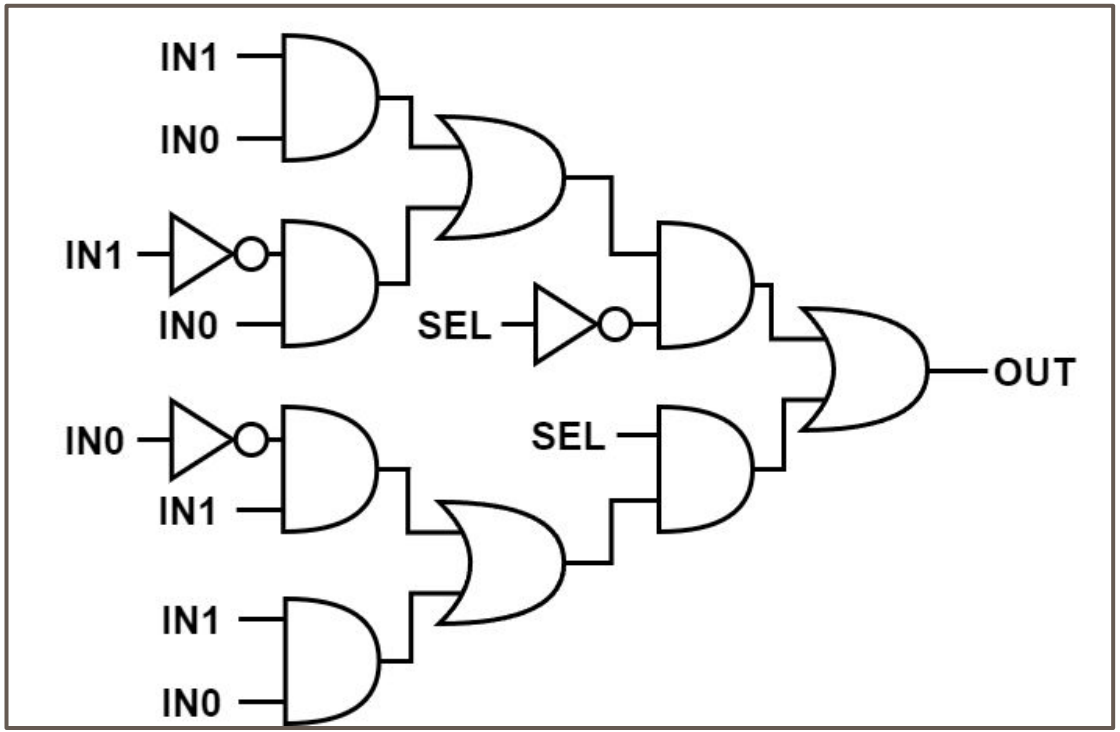


15 Primitive Gates?!?!

What can we simplify here?

Let's Practice!

OUT = [~SEL ● ((~IN1 ● IN0) + (IN1 ● IN0))] + [SEL ● ((IN1 ● ~IN0) + (IN1 ● IN0))]

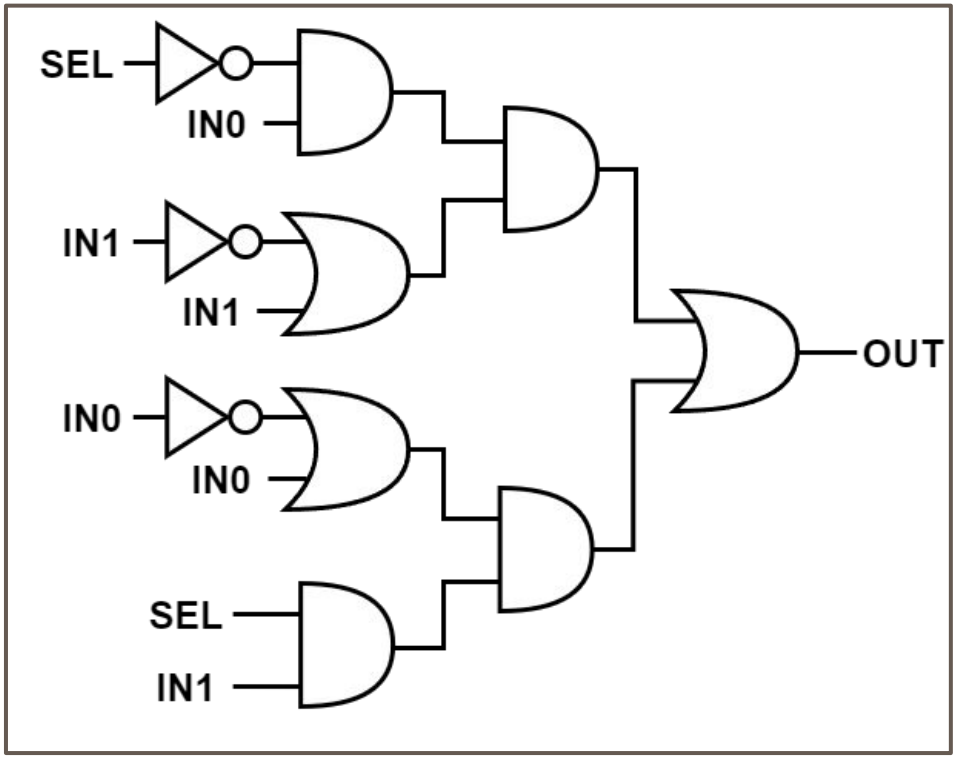


DISTRIBUTION
→ 12 Primitive Gates

We can do better!

Let's Practice!

OUT = [~SEL ● IN0 ● (~IN1 + IN1)] + [SEL ● IN1 ● (~IN0 + IN0)]

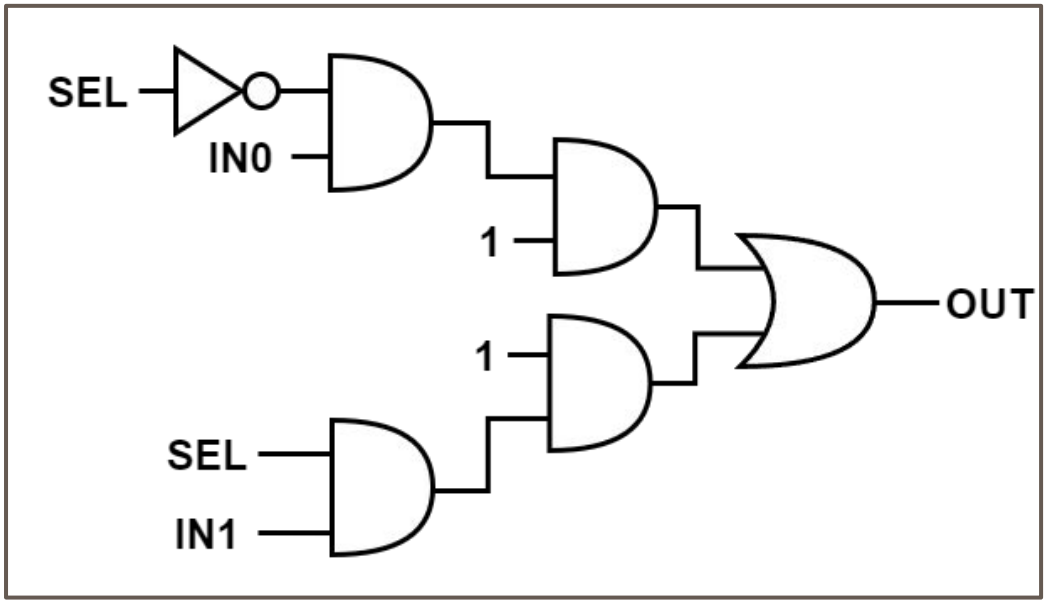


Distribution Again!
→ 10 Primitive Gates

We can do better!!

Let's Practice!

OUT = [~SEL ● IN0 ● (1)] + [SEL ● IN1 ● (1)]

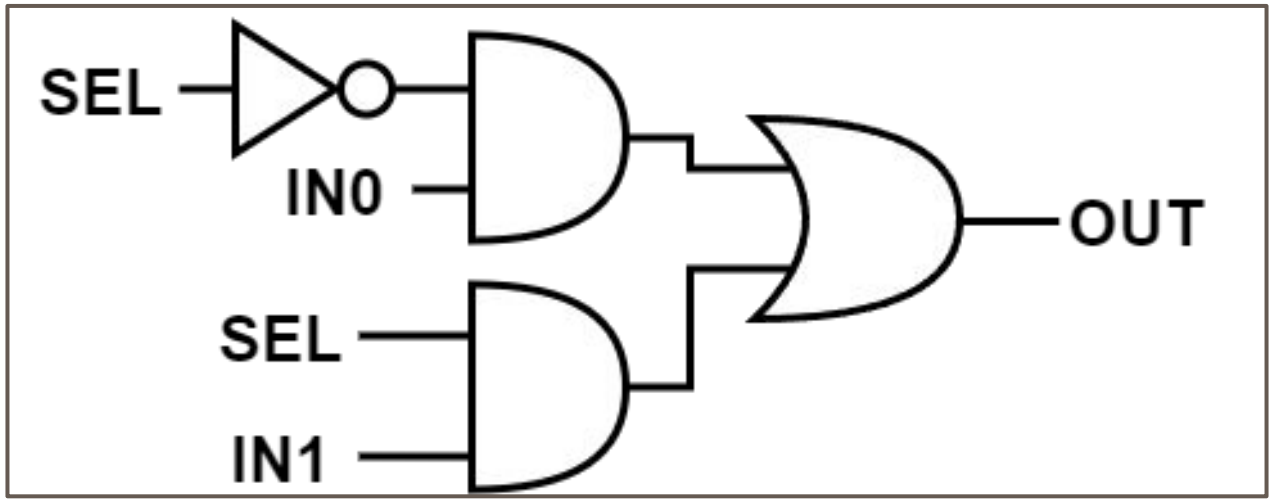


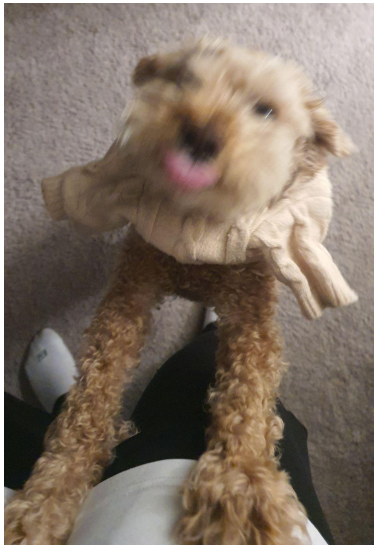
Complements
→ 6 Primitive Gates

BETTER!!!

Let's Practice!

OUT = (~SEL ● IN0) + (SEL ● IN1)





Identity
→ 4 Primitive Gates

1) $a+0=$ _____

2) $\bar{a}\cdot 0=$ _____

3) $a+\bar{a}=$ _____

4) $a+a=$ _____

5) $a+ab=$ _____

6) $a+\bar{a}b=$ _____

7) $a(\bar{a}+b)=$ _____

8) $ab+\bar{a}b=$ _____

9) $(\bar{a}+\bar{b})(\bar{a}+b)=$ _____

10) $a(a+b+c+\ldots)=$ _____

For (11),(12), (13), $f(a,b,c)=a+b+c$

11) $f(a,b,ab)=$ _____

12) $f(a,b,\bar{a}\cdot\bar{b})=$ _____

13) $f[a,b,\overline{(ab)}]=$ _____

14) $y+y\bar{y}=$ _____

15) $xy+x\bar{y}=$ _____

16) $\bar{x}+y\bar{x}=$ _____

17) $(w+\bar{x}+y+\bar{z})y=$ _____

18) $(x+\bar{y})(x+y)=$ _____

19) $w+[w+(wx)]=$ _____

20) $x[x+(xy)]=$ _____

21) $\overline{(\bar{x}+\bar{x})}=$ _____

22) $\overline{(x+\bar{x})}=$ _____

23) $w+(w\bar{x}yz)=$ _____

24) $\bar{w}\cdot\overline{(wxyz)}=$ _____

25) $xz+\bar{x}y+zy=$ _____

26) $(x+z)(\bar{x}+y)(z+y)=$ _____

27) $\bar{x}+\bar{y}+xy\bar{z}=$ _____

Project 3A “Robbie” Overview

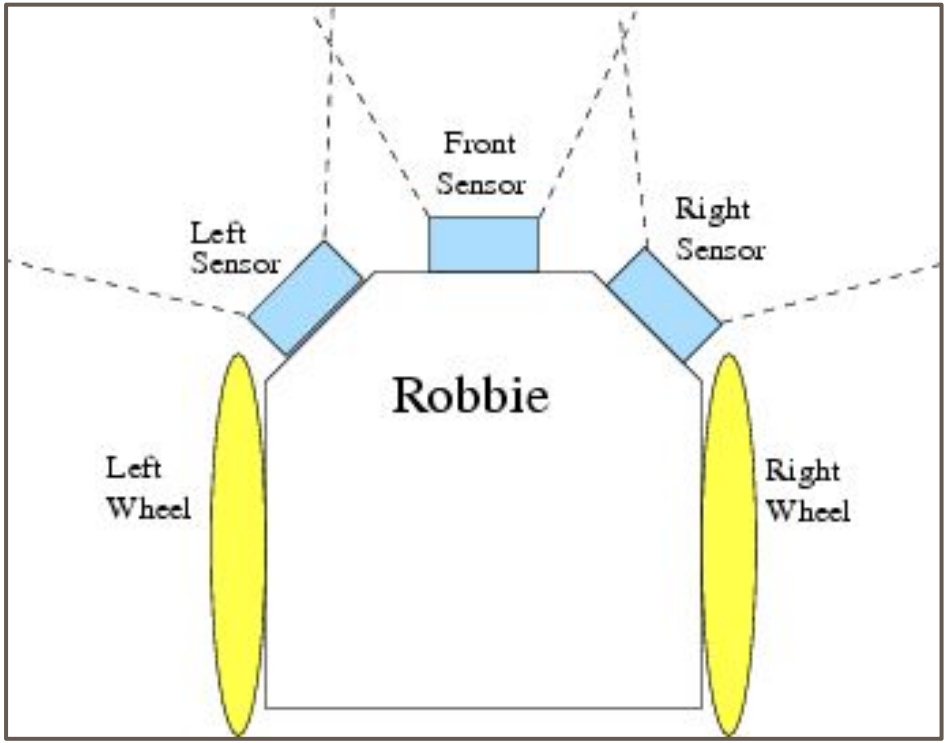
USE STRUCTURAL VERILOG FOR Project 3A!!!

- **You SHOULD NOT be using**
  - **Continuous ‘assign’ statements**
  - **Procedural statements ‘always’ or ‘initial’**

- These abstractural statements are used in “Behavioral” Verilog
  - Will be using in Projects 3B - 7

- **You SHOULD ONLY be using**
  - **Primitive gate instantiations as described in previous discussion**
  - **Module instantiations as described in previous discussion**

Robbie Interface



Inputs (Switches)

1. Left Sensor (ls)
2. Front Sensor (fs)
3. Right Sensor (rs)

Input Values

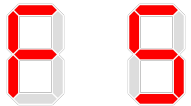
‘1’ - Beacon Detected

‘0’ - No Beacon Detected

Outputs (HEX Displays)

1. Left Wheel
2. Right Wheel

Output Values



THANK YOU

Please fill out form if you haven’t already done so!
(Link Here As Well)

(Bonus) More Algebraically Proving Theorems

Let’s Prove More Theorems (Algebraic Manipulation)

X ● (X + Y) = **What theorem can we apply here?**

Let's Prove More Theorems (Algebraic Manipulation)

$X \bullet (X + Y) = (X \bullet X) + (X \bullet Y)$ (Distribution)

= **Next Simplification Step**

Let's Prove More Theorems (Algebraic Manipulation)

$X \bullet (X + Y) = (X \bullet X) + (X \bullet Y)$ (Distribution)

$= X + (X \bullet Y)$ (Idempotency)

= **What term is shared in both?**

Let's Prove More Theorems (Algebraic Manipulation)

$X \bullet (X + Y) = (X \bullet X) + (X \bullet Y)$ (Distribution)

$= X + (X \bullet Y)$ (Idempotency)

$= (X \bullet 1) + (X \bullet Y)$ (Identity)

= **Now what can we "factor out"?**

Let's Prove More Theorems (Algebraic Manipulation)

$X \bullet (X + Y) = (X \bullet X) + (X \bullet Y)$ (Distribution)

$= X + (X \bullet Y)$ (Idempotency)

$= (X \bullet 1) + (X \bullet Y)$ (Identity)

$= X \bullet (1 + Y)$ (Distribution)

= **Simplify**

Let's Prove More Theorems (Algebraic Manipulation)

$X \bullet (X + Y) = (X \bullet X) + (X \bullet Y)$ (Distribution)

$= X + (X \bullet Y)$ (Idempotency)

$= (X \bullet 1) + (X \bullet Y)$ (Identity)

$= X \bullet (1 + Y)$ (Distribution)

$= X \bullet 1$ (Null Elements)

= **Simplify**

Let's Prove More Theorems (Algebraic Manipulation)

$X \bullet (X + Y) = (X \bullet X) + (X \bullet Y)$ (Distribution)

$= X + (X \bullet Y)$ (Idempotency)

$= (X \bullet 1) + (X \bullet Y)$ (Identity)

$= X \bullet (1 + Y)$ (Distribution)

$= X \bullet 1$ (Null Elements)

$= \underline{X}$ **(Absorption Theorem)**

Another One!! (Algebraic Manipulation)

$X + (X' \bullet Y) = ?$

Another One!! (Algebraic Manipulation)

$X + (X' \bullet Y) = (X + X') \bullet (X + Y)$ (Distribution)

$= ?$

Another One!! (Algebraic Manipulation)

$X + (X' \bullet Y) = (X + X') \bullet (X + Y)$ (Distribution)

$= (1) \bullet (X + Y)$ (Complements)

$= ?$

Another One!! (Algebraic Manipulation)

$X + (X' \bullet Y) = (X + X') \bullet (X + Y)$ (Distribution)

$= (1) \bullet (X + Y)$ (Complements)

$= \underline{X + Y}$ **("No Name" Theorem)**

Proving “No Name” Theorem (Perfect Induction)

- Proving X ● (X’ + Y) = X ● Y

X	**Y**	X’	X’ + Y	**X ● (X’ + Y)**	**X ● Y**
0	**0**	1	1	**0**	**0**
0	**1**	1	1	**0**	**0**
1	**0**	0	0	**0**	**0**
1	**1**	0	1	**1**	**1**

Going Through Proving Consensus

$XY + X'Z + YZ = ?$

$= ?$

$= ?$

$= ?$

$= ?$

$= ?$

$= \mathbf{XY + X'Z}$

Going Through Proving Consensus

$XY + X'Z + YZ = XY + X'Z + (YZ \bullet (1))$ (Identities)

$= ?$

$= ?$

$= ?$

$= ?$

$= ?$

$= \mathbf{XY + X'Z}$

Going Through Proving Consensus

$XY + X'Z + YZ = XY + X'Z + (YZ \bullet (1))$ (Identities)

$= XY + X'Z + (YZ \bullet (X + X'))$ (Complement)

$= ?$

$= ?$

$= ?$

$= ?$

$= \mathbf{XY + X'Z}$

Going Through Proving Consensus

$XY + X'Z + YZ = XY + X'Z + (YZ \bullet (1))$ (Identities)

$= XY + X'Z + (YZ \bullet (X + X'))$ (Complement)

$= XY + X'Z + (XYZ + X'YZ)$ (Distribution)

$= ?$

$= ?$

$= ?$

$= \mathbf{XY + X'Z}$

Going Through Proving Consensus

XY + X'Z + YZ = XY + X'Z + (YZ ● (1))	(Identities)
= XY + X'Z + (YZ ● (X + X'))	(Complement)
= XY + X'Z + (XYZ + X'YZ)	(Distribution)
= (XY + XYZ) + (X'Z + X'YZ)	(Commutativity)
= ?	
= ?	
= **XY + X'Z**	

Going Through Proving Consensus

$$XY + X'Z + YZ = XY + X'Z + (YZ \bullet (1)) \quad \text{(Identities)}$$
$$= XY + X'Z + (YZ \bullet (X + X')) \quad \text{(Complement)}$$
$$= XY + X'Z + (XYZ + X'YZ) \quad \text{(Distribution)}$$
$$= (XY + XYZ) + (X'Z + X'YZ) \quad \text{(Commutativity)}$$
$$= (XY \bullet (1 + Z)) + (X'Z \bullet (1 + Y)) \quad \text{(Distribution)}$$
$$= ?$$
$$= \mathbf{XY + X'Z}$$

Going Through Proving Consensus

XY + X'Z + YZ = XY + X'Z + (YZ ● (1))	(Identities)
= XY + X'Z + (YZ ● (X + X'))	(Complement)
= XY + X'Z + (XYZ + X'YZ)	(Distribution)
= (XY + XYZ) + (X'Z + X'YZ)	(Commutativity)
= (XY ● (1 + Z)) + (X'Z ● (1 + Y))	(Distribution)
= (XY ● (1)) + (X'Z ● (1))	(Null Elements)
= **XY + X'Z**	

Going Through Proving Consensus

$$
\begin{aligned}
XY + X'Z + YZ &= XY + X'Z + (YZ \bullet (1)) && \text{(Identities)} \\
&= XY + X'Z + (YZ \bullet (X + X')) && \text{(Complement)} \\
&= XY + X'Z + (XYZ + X'YZ) && \text{(Distribution)} \\
&= (XY + XYZ) + (X'Z + X'YZ) && \text{(Commutativity)} \\
&= (XY \bullet (1 + Z)) + (X'Z \bullet (1 + Y)) && \text{(Distribution)} \\
&= (XY \bullet (1)) + (X'Z \bullet (1)) && \text{(Null Elements)} \\
&= \mathbf{XY + X'Z} && \text{(Identities)}
\end{aligned}
$$

Adding Consensus Term Cool Trick

Question coming from FA 21 Question 1A (Simplifying Equation)

$WX + W'Y + XYZ = (WX + W'Y + XY) + XYZ$ (Adding Consensus Term XY)

$= WX + W'Y + (XY + XYZ)$

$= WX + W'Y + XY$ (Absorption)

$= \mathbf{WX + W'Y}$ (Removing Consensus Term!)

1) $a+0=$ _____

2) $\bar{a}\cdot 0=$ _____

3) $a+\bar{a}=$ _____

4) $a+a=$ _____

5) $a+ab=$ _____

6) $a+\bar{a}b=$ _____

7) $a(\bar{a}+b)=$ _____

8) $ab+\bar{a}b=$ _____

9) $(\bar{a}+\bar{b})(\bar{a}+b)=$ _____

10) $a(a+b+c+\ldots)=$ _____

For (11),(12), (13), $f(a,b,c)=a+b+c$

11) $f(a,b,ab)=$ _____

12) $f(a,b,\bar{a}\cdot\bar{b})=$ _____

13) $f[a,b,\overline{(ab)}]=$ _____

14) $y+y\bar{y}=$ _____

15) $xy+x\bar{y}=$ _____

16) $\bar{x}+y\bar{x}=$ _____

17) $(w+\bar{x}+y+\bar{z})y=$ _____

18) $(x+\bar{y})(x+y)=$ _____

19) $w+[w+(wx)]=$ _____

20) $x[x+(xy)]=$ _____

21) $\overline{(\bar{x}+\bar{x})}=$ _____

22) $\overline{(x+\bar{x})}=$ _____

23) $w+(w\bar{x}yz)=$ _____

24) $\bar{w}\cdot\overline{(wxyz)}=$ _____

25) $xz+\bar{x}y+zy=$ _____

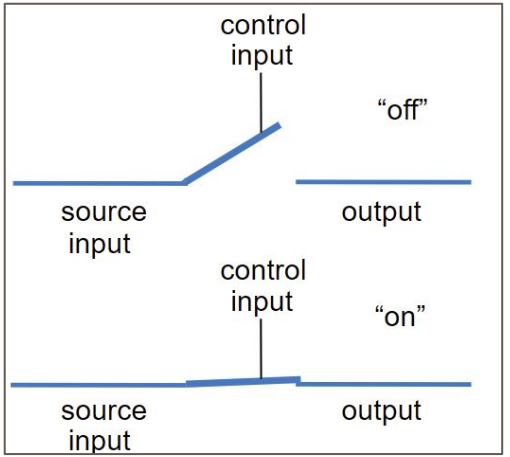
26) $(x+z)(\bar{x}+y)(z+y)=$ _____

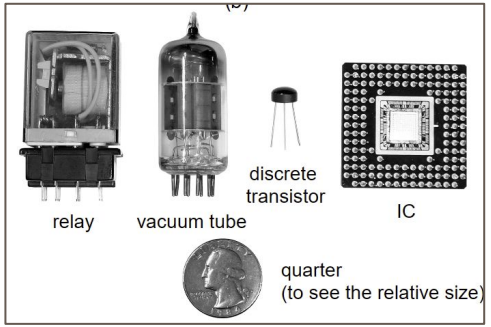
27) $\bar{x}+\bar{y}+xy\bar{z}=$ _____

(Bonus) From Switches to Boolean Algebra

Switches - The Basis of All Binary Digital Circuits

- Each switch acts as a binary bit (0 or 1)
- Has evolved throughout the past century
  - Relays (1930s)
  - Vacuum Tubes (1940s)
  - Discrete Transistors (1950s)
  - Integrated Circuits (1960s - Today!)
    - Small Scale Integration (SSI) (1960s)
    - Medium Scale Integration (MSI) (1960s)
    - Large Scale Integration (LSI) (1970s)
    - **Very Large Scale Integration (VLSI) (1980s - Today)**



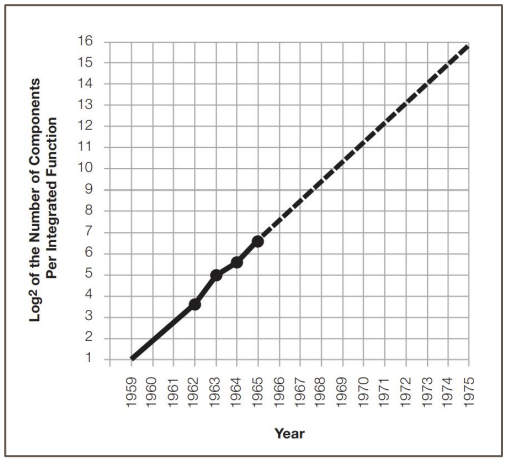


Moore’s “Law” - The Prediction of a Lifetime

- Gordon Moore (Co-founder of Intel)
  - Asked to predict what would happen with ICs in the next decade (1965)

- **Predicted Transistor Density count will double around every 2 years**
  - “Computing power will double every 2 years”
  - “Cost of computing power will half every 2 years”
  - Intel Exec. David House predicted **every 18 months**

- Purely an educated guess
  - Extrapolated by a few data points at the time
  - “Line go up”

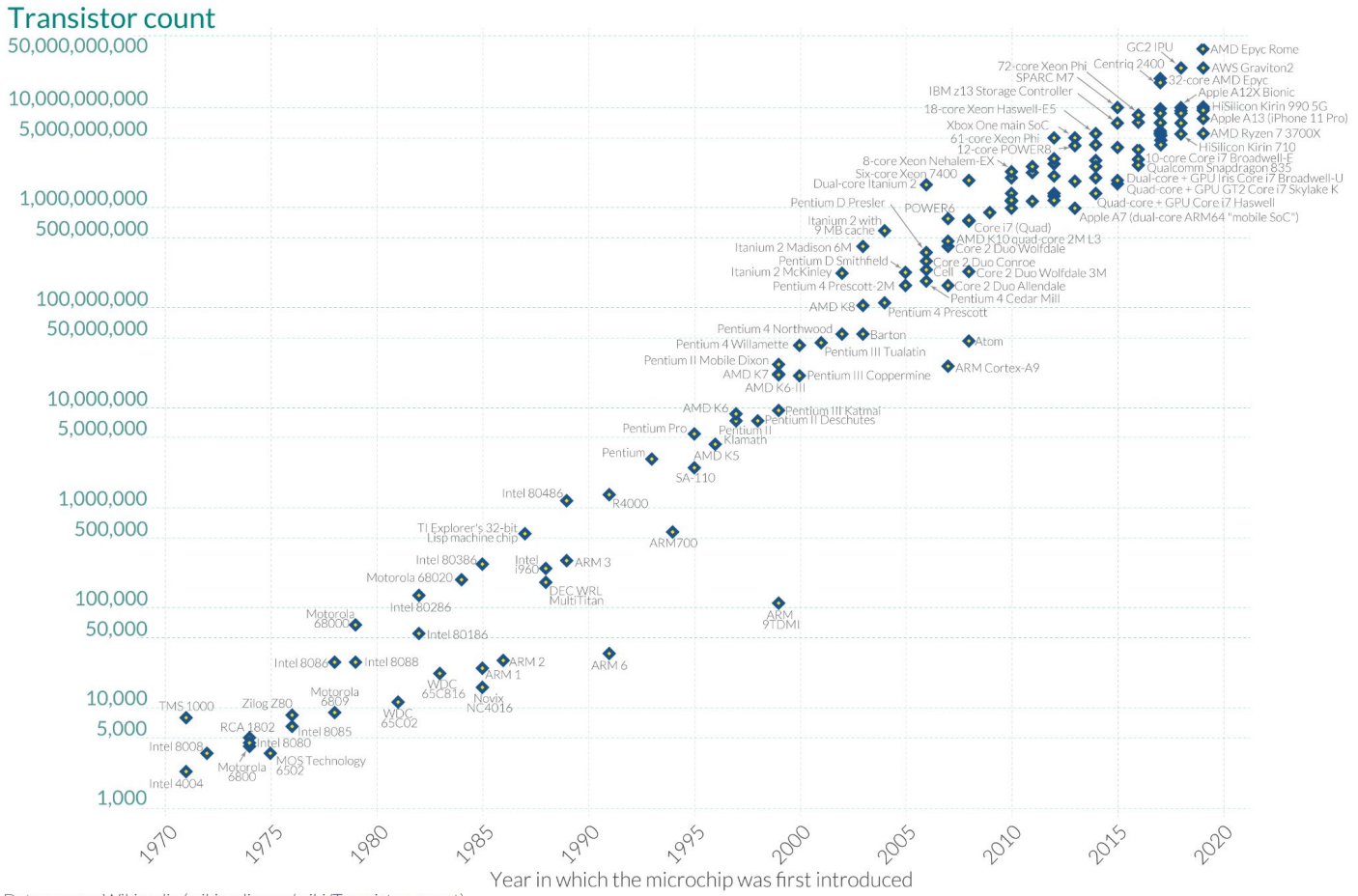


[Read Gordon Moore's 1965 Paper! (It's a Really Cool Quick Read)]

Moore's Law: The number of transistors on microchips has doubled every two years

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



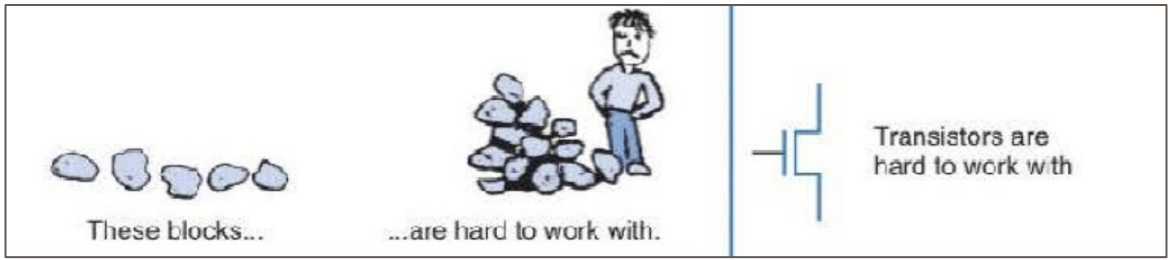
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world's largest problems.

Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

The Problem - Switches as Building Blocks

- Working with switches as building blocks to complex designs is DIFFICULT
- Switches are too low-level as building blocks
- Trial-and-Error Method in Circuit Designing
  - No standardized notation
  - Difficult to analyze
  - Very slow and error prone on complex systems

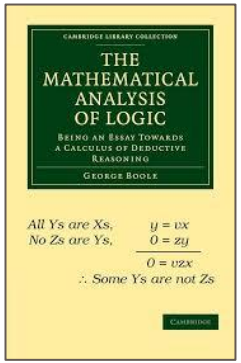


What is Boolean Algebra? (1800s)

- Developed by mathematician and philosopher George Boole
  - 1847 “The Mathematical Analysis of Logic” Book

- Created a scheme using algebraic methods to formulate human logic and thought

- Boolean Variables - Variables whose values can only be 1 (True) or 0 (False)
- Boolean Operators - Operate on boolean variables and return True or False



Claude Shannon AKA The Goat

- Revolutionized many fields through his methodology of simplification
  - Undergraduate at the **University of Michigan Ann Arbor!!!**
- Changed switching circuits through his master's thesis
  - Introduced using Boolean Algebra to Switch-Based Circuits!

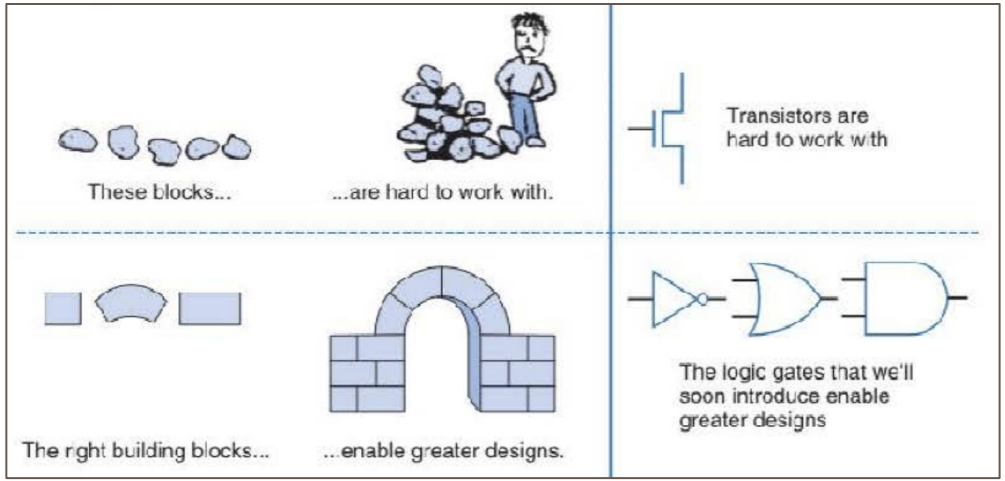
This could be you!!!

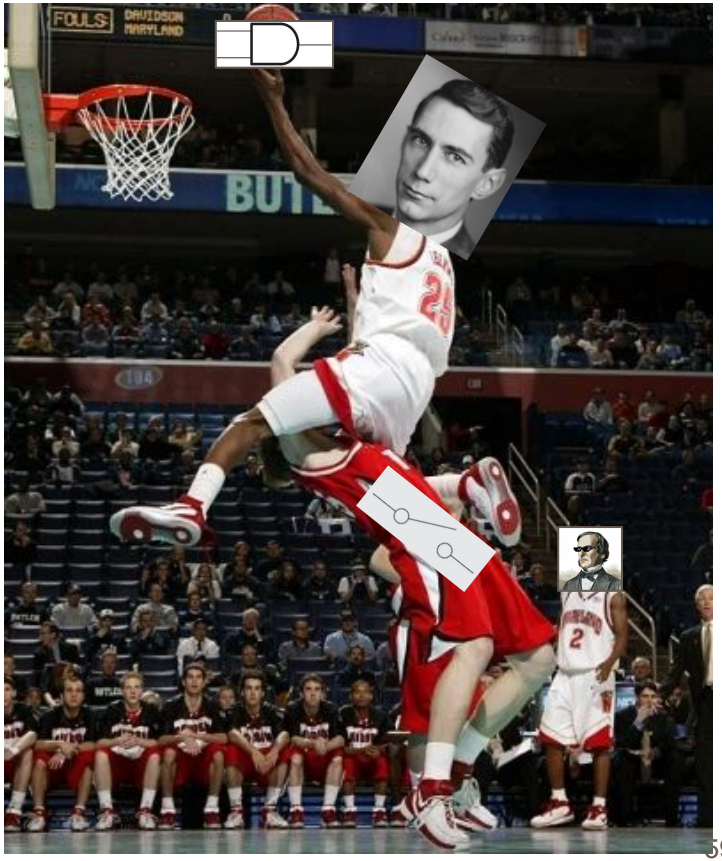
"We can build circuits by doing math"

[Great Paper about Claude Shannon and His Work]
[Claude Shannon's Master Thesis]

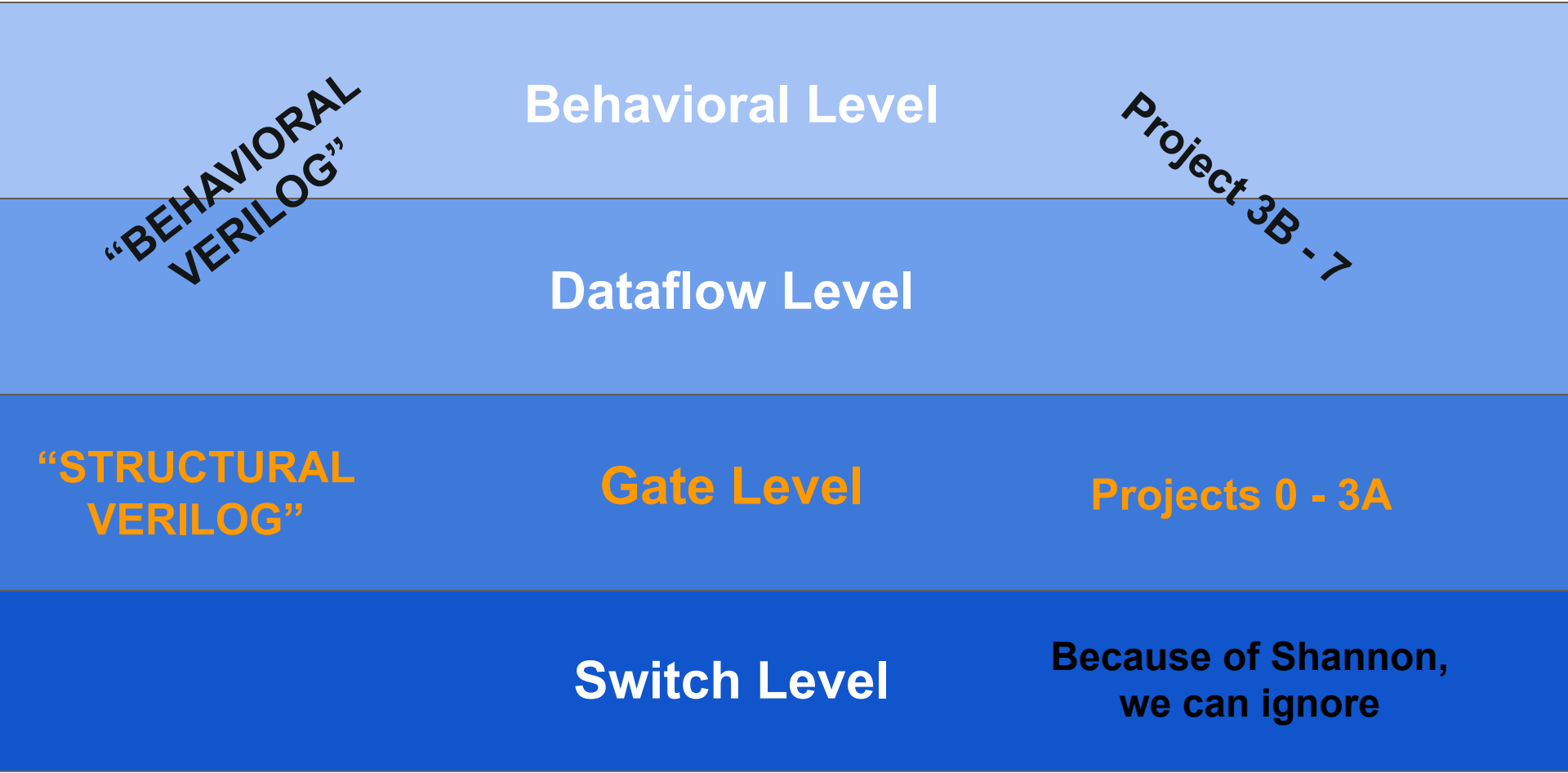
The New Building Blocks - Logic Gates

1. Implement AND, OR, NOT w/ small circuit of switches
2. **Forget about switches**
3. Win



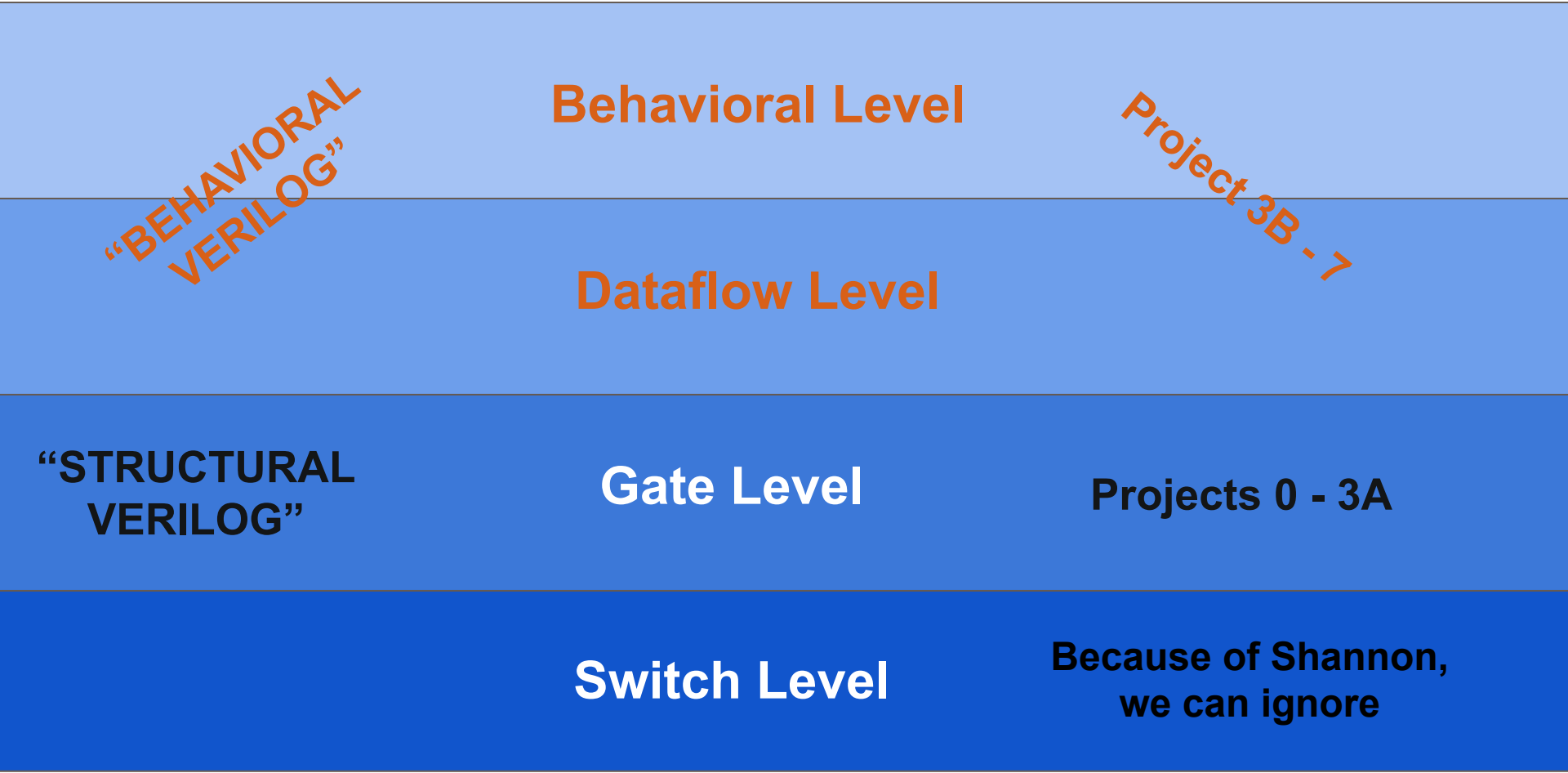


Verilog's Many Levels of Abstraction



(Bonus) Behavioral Verilog Introduction (For Project 3B “Renee”) **(Will be covered next week)**

Verilog's Many Levels of Abstraction



Behavioral Verilog Overview

- Gate-Level “Structural Verilog” Modelling
  - We can instantiate every primitive gate individually and connect them together
  - Helpful in building our knowledge of digital logic design
  - PROBLEM: Can get very difficult and tedious will more complex logic designs

- **Dataflow and Behavioral Level “Behavioral Verilog” Modelling**
  - Higher level abstraction of Verilog
  - **Allows us to focus more on the algorithmic approach than the individual gates**

- Logical Synthesis compiles our design to low-level Verilog to be understood by the board

Declaring and Assigning Wires (Behavioral)

```
// Declaring 1-bit wires
wire out_and, out_or_3, one_bit_mux;

// Continious assignment to wire 'out_and'
// Performing BIT-WISE and of i1 and i2
// NOTE: Left side MUST be a wire (or some net)
assign out_and = i1 & i2;

// Able to perform multiple operations in a single assign
assign out_or_3 = i1 | i2 | i3;

// Use of parenthesis to explicitly define ordering
assign one_bit_mux = (sel & i1) | (~sel & i2);
```

Logical vs. Bitwise Operations (Behavioral)

Logical Operations

- Always evaluate to a 1-bit value
  - 0, 1, or X
- Logical And (&&), Or (||), Not (!)

```
// out1 = (true) || (true) = (true) = 1'b1
assign out1 = 4'b1001 || 4'b0011;

// out2 = (true) || (false) = (true) = 1'b1
assign out2 = 4'b1001 || 4'b0000;

// out3 = !(true) = (false) = 1'b0
assign out3 = !(4'b1001)

// out4 = (true) && (false) = (false) = 1'b0
assign out4 = 4'b1001 && 4'b0000;
```

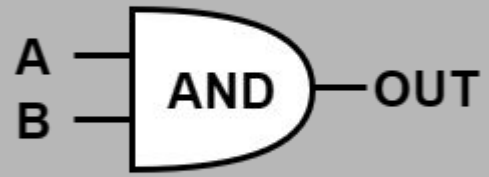
Bitwise Operations

- Perform bit-by-bit operations
  - Will zero-extend the less-sized number
- Bitwise And (&), Or (|), Not (~)
- Bitwise Xor (^), Xnor (~^)

```
// out1 = (4'b1001) | (4'b0011) = 4'b1011
assign out1 = 4'b1001 | 4'b0011;

// out2 = (4'b1001) | (4'b0000) = 4'b1001
assign out2 = 4'b1001 | 4'b0000;

// out3 = ~(4'b1001) = 4'b0110
assign out3 = ~(4'b1001)

// out4 = (4'b1001) & (4'b0011) = 4'b0001
assign out4 = 4'b1001 & 4'b0011;
```

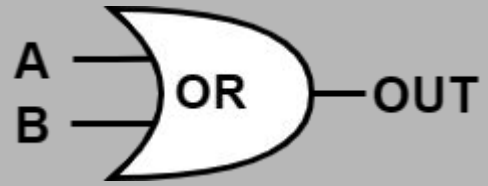
The Bitwise Operators - AND, OR, NOT

AND



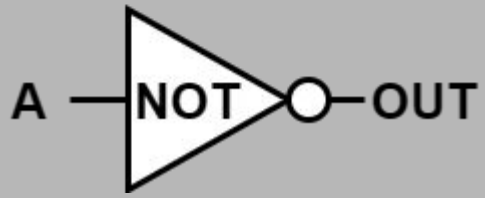
A	B	OUT
0	0	0
0	1	0
1	0	0
1	1	1

OR



A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	1

NOT



A	OUT
0	1
1	0

The Bitwise Operators - NAND, NOR, XOR

- Will cover and practice more in-depth in later discussions

NAND

A	B	OUT
0	0	1
0	1	1
1	0	1
1	1	0

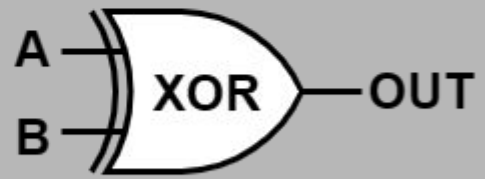
Inverted AND

NOR

A	B	OUT
0	0	1
0	1	0
1	0	0
1	1	0

Inverted OR

XOR



A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0

Inequality Operator

The Bitwise Operators - XNOR, BUF

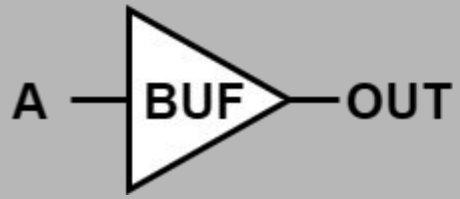
- Will cover and practice more in-depth in later discussions

XNOR

A	B	OUT
0	0	1
0	1	0
1	0	0
1	1	1

Equality Operator

BUF



A	OUT
0	0
1	1

Pass-Through Operator

Verilog Ternary Operator (Behavioral)

- Similar to a MUX operation (Built in project 1)
  - If-Else Statement
- Can assign multiple bits at a time!

```
// Conditional Ternary Operator
// If   sel = 1'b1 --> out = 4'b1001
// Else sel = 1'b0 --> out = 4'b0011
wire [3:0] out;
assign out = sel ? (4'b1001) : (4'b0011);
```

Reduction Operators (Behavioral)

- Applying operation on every single bit in a bitvector

```
// Assume x = 4'b1010
assign out1 = &X //Equivalent to 1 & 0 & 1 & 0. Results in 1'b0
assign out2 = |X //Equivalent to 1 | 0 | 1 | 0. Results in 1'b1
assign out3 = ^X //Equivalent to 1 ^ 0 ^ 1 ^ 0. Results in 1'b0
```

Verilog Concatenation (Behavioral)

```
// Declaring 4-bit wires
wire [3:0] i1, i2;

assign i1 = 4'b1010;
assign i2 = 4'b0011;

// Declaring 8-bit wire
wire [7:0] o1;

// Using concatenation for assignment
// Appending the inputs into one bitvector
// o1 = {i2, i1} = {4'b0011, 4'b1010} = 8'b0011_1010
assign o1 = {i2, i1};
```

PLEASE USE PARENTHESES

- **To ensure the exact ordering of operations to occur, use parenthesis!!**

```
// The ordering is very hard to determine
// Verilog has an internal ordering
// DO NOT ASSUME
assign out = i1 ^ i2 & i3 | i4 & i5 ~^ i6;

// Explicitly define order with parenthesis :)
assign safe_out = ((i1 ^ i2) & i3) | (i4 & (i5 ~^ i6));
```

Go through these practice problems!

- Attempt these problems!
  - Can now use Behavioral Verilog!

- HDLBits Sections (https://hdlbits.01xz.net/wiki/Problem_sets)
  - Getting Started
  - Verilog Language - Basics
  - Verilog Language - Vectors
  - Verilog Language - Modules: Hierarchy

- ASK QUESTIONS IF YOU HAVE ANY!