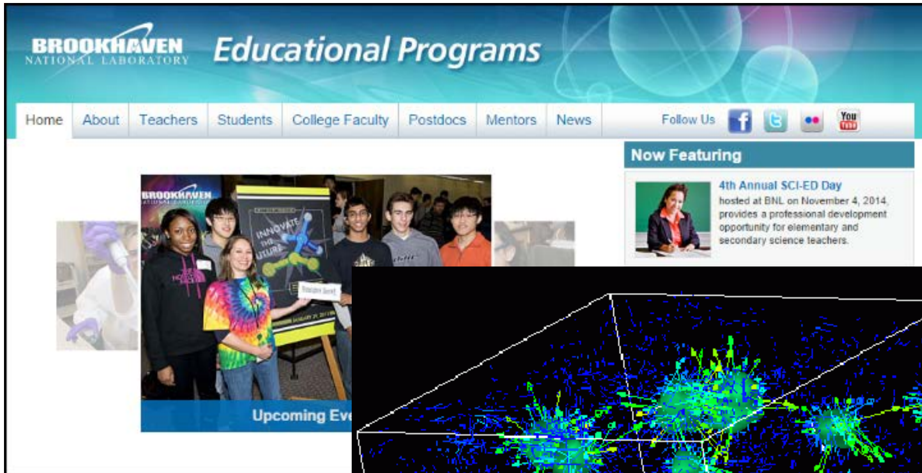
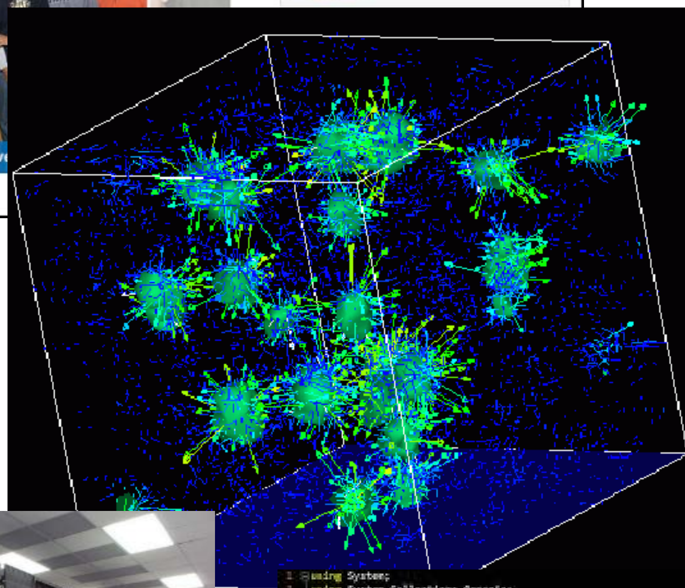
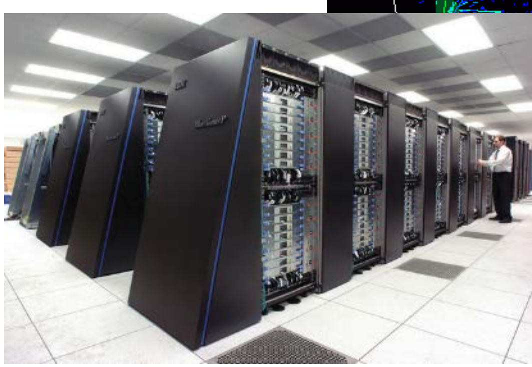






Survey of Scientific Computing

(SciComp 301)

Dave Biersach
Brookhaven National Laboratory
dbiersach@bnl.gov

Session 26

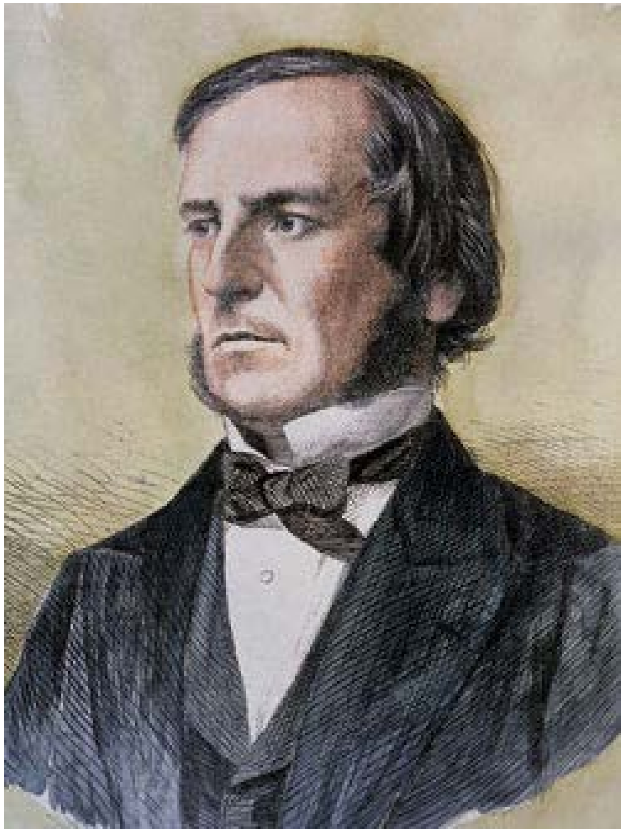
Boolean Algebra, Logic Gates

Session Goals

- Understand the Boolean mathematics of the three basic logic gates: **NOT**, **AND**, and **OR**
- Learn how to draw individual logic gates and how to chain multiple logic gates *together* in a circuit
- Incorporate multiple data input and output **lines** in a circuit
- Develop **truth tables** and analyze logic circuits to calculate the output states given the input states
- Understand how **half-adders** and **full-adders** operate
- Appreciate how **memory** can be constructed from gates

George Boole (1815 – 1864)

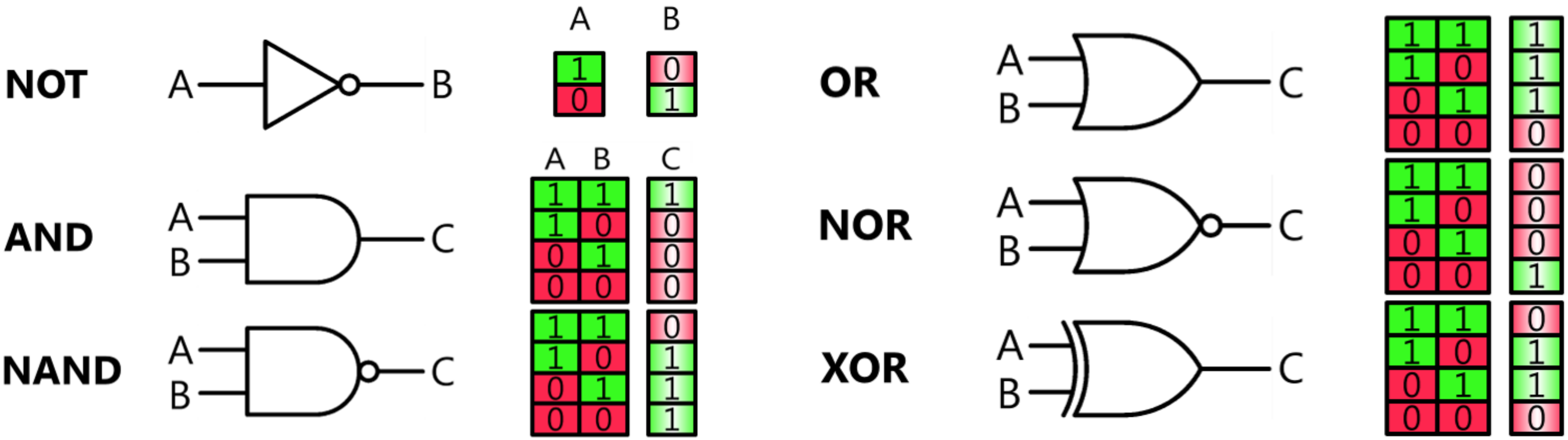
- English mathematician
- 1847 published rules of symbolic logic (“Boolean Algebra”)
- Only person with a data type named after him (“bool”)



Boolean Logic Gates

- Logic Gates
  - Three types: **NOT** (inverter), **AND**, **OR**
  - Gates have **1 or 2 input** lines, and **1 output** line
- Input / Output lines are either:
  - False, F, Cold, Low, **L, 0** (Zero)
  - True, T, Hot, High, **H, 1**
- The <u>left</u> side of a **truth table** is counted using **Base 2** to ensure every possible *permutation* of input states is evaluated across the entire circuit

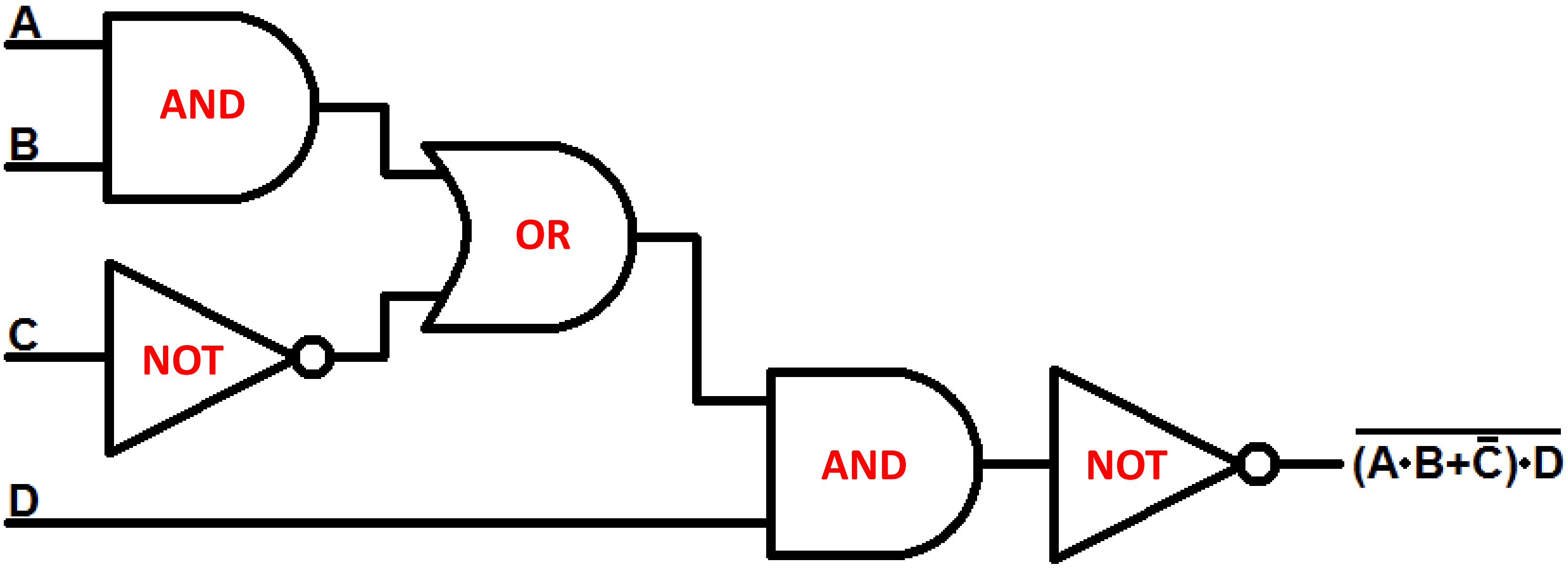
Boolean Logic Gates



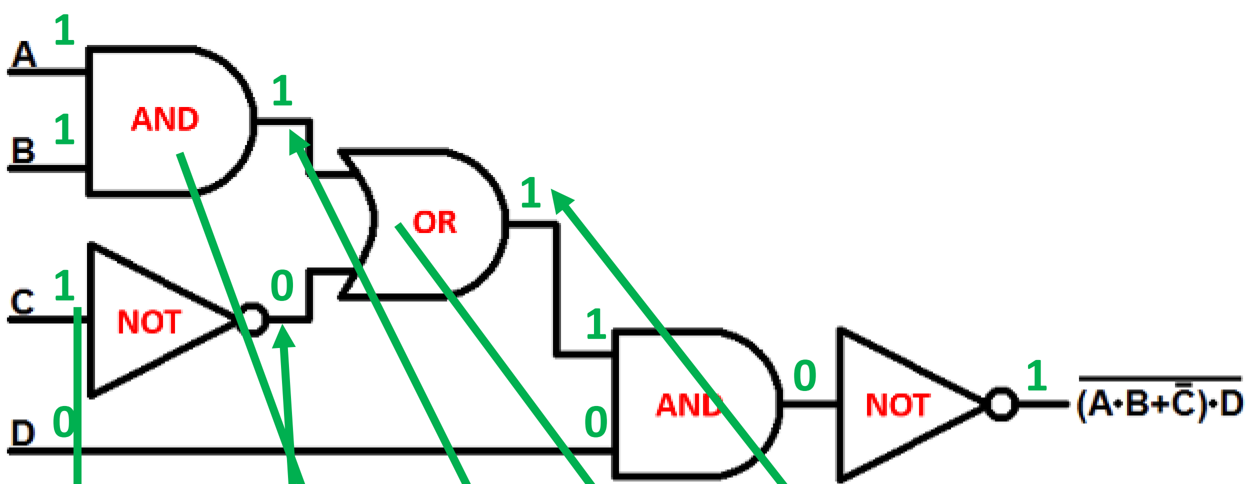
Boolean Logic Gates

- **NOT** (inversion) is sometimes shown as a “bar” on top
- **OR** is a “sum” while **AND** is a “multiply” (modulo 2)
  - OR is sometimes shown as A + B
  - AND is sometimes shown as A • B
- Circuits flow (propagate state) from “**Left to Right**”
  - The <u>output</u> of one gate flows into the <u>input(s)</u> of the *next* gate(s)
  - We can evaluate a gate’s output line **only** when we know the value for every input line entering that gate

Chaining Logic Gates



Truth Tables



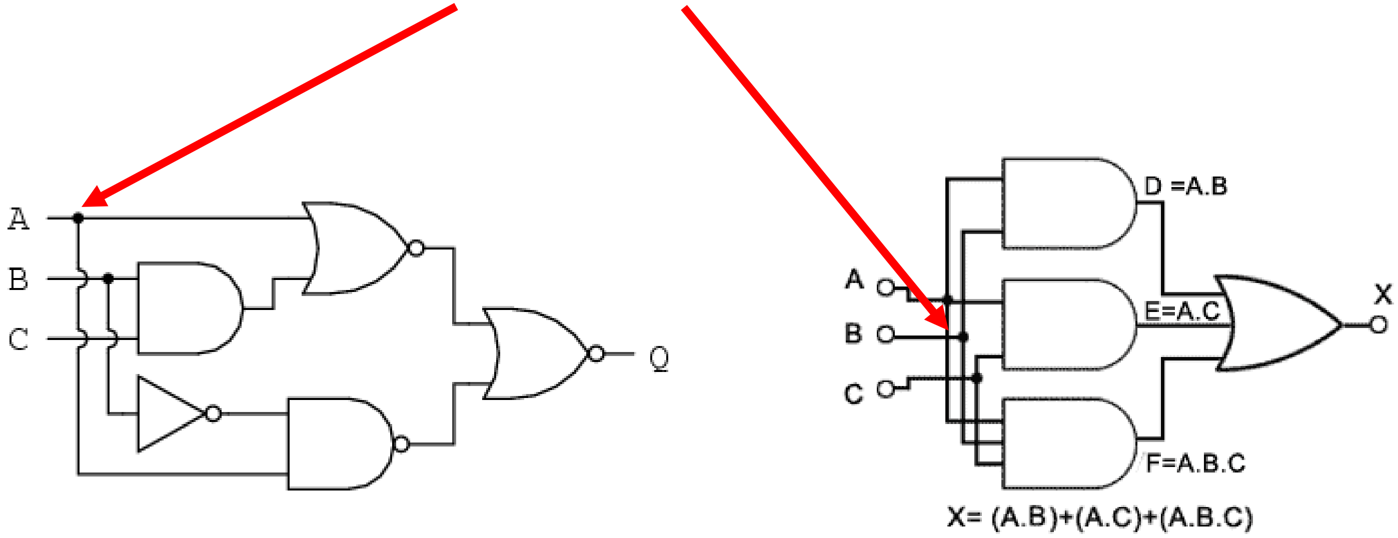
NOT Gate	
INPUT	OUTPUT
0	1
1	0

AND Gate		
INPUT		OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

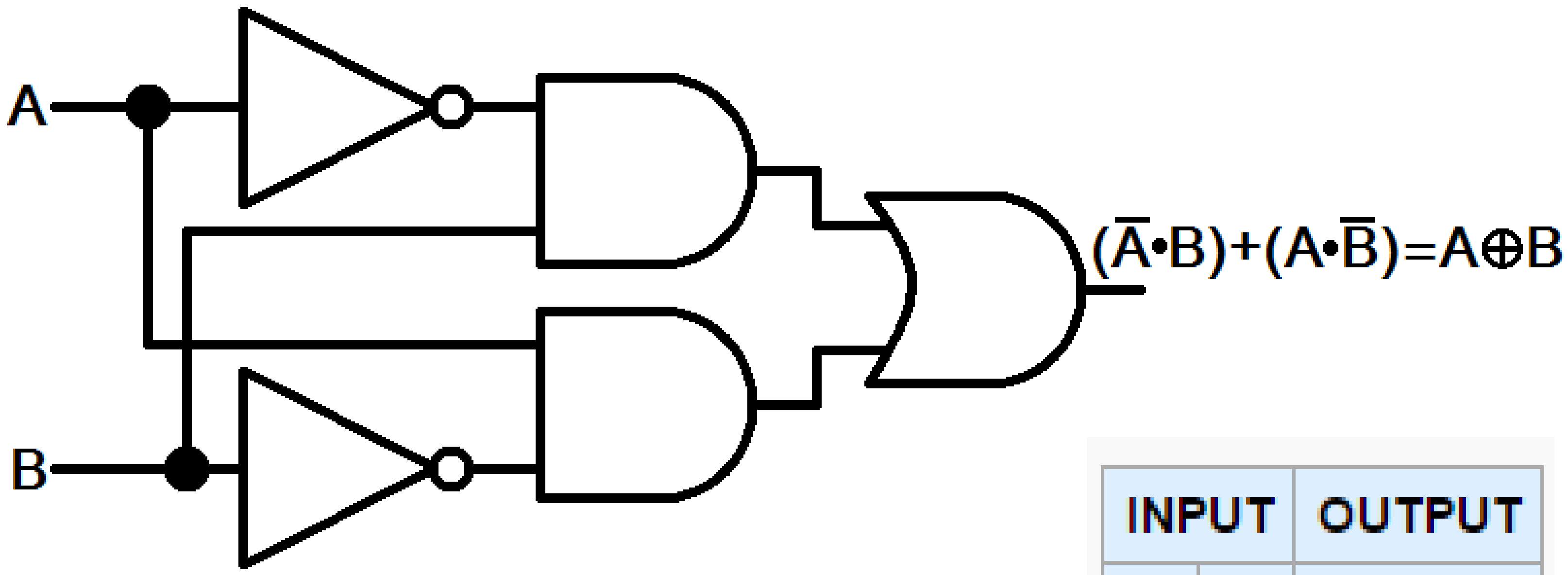
OR Gate		
INPUT		OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

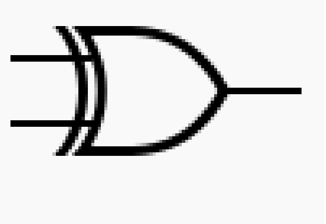
Wire **Overlap** vs. **Intersection**

- Use **filled** circles ***only*** for electrical junction points
- Carry forward current logic state of line into each **branch**



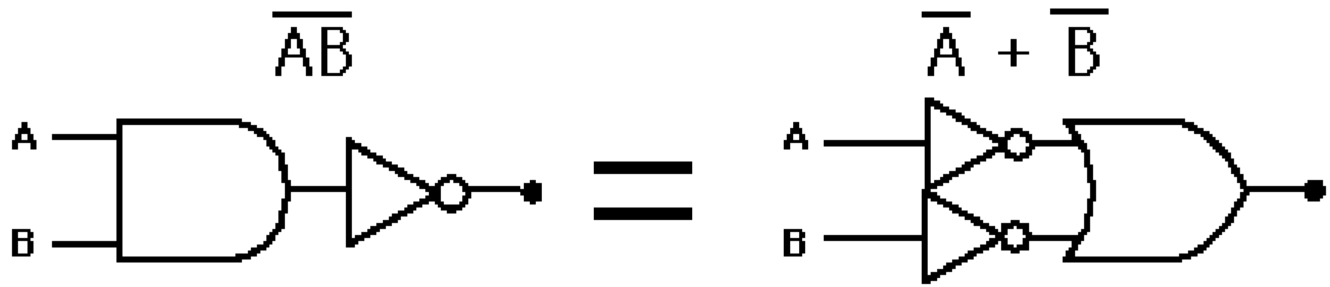
XOR Gate



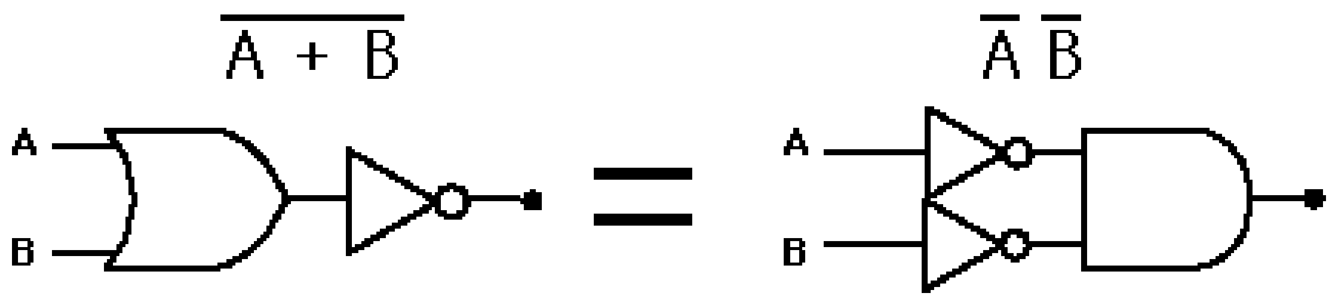


INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

NAND and **NOR** Gates

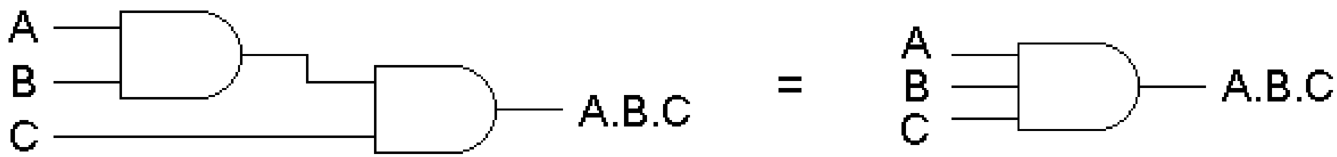


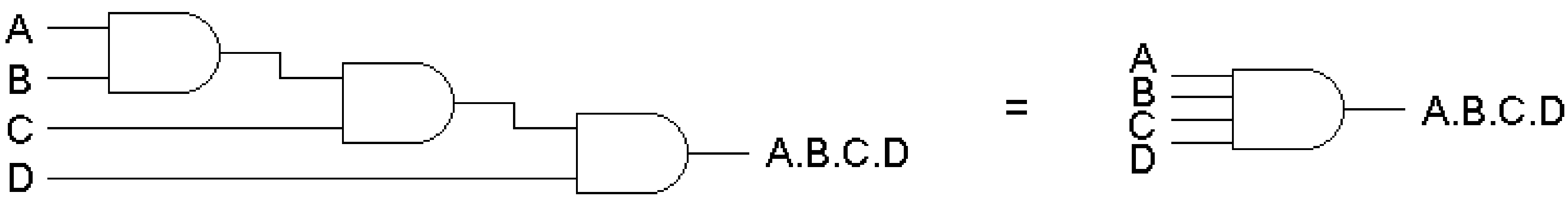
A NAND gate is equivalent to an inversion followed by an OR



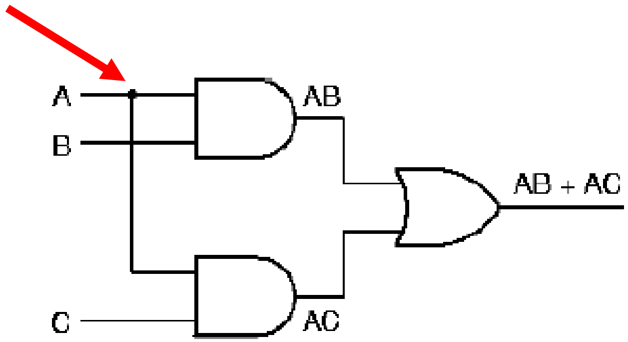
A NOR gate is equivalent to an inversion followed by an AND

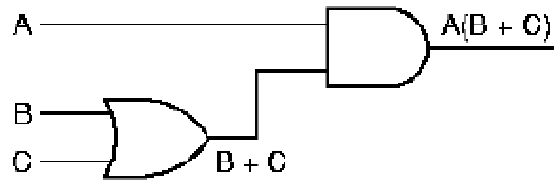
Multi-Input Gates





Truth Tables





$000_2 = \mathbf{0}_{10}$
$001_2 = \mathbf{1}_{10}$
$010_2 = \mathbf{2}_{10}$
$011_2 = \mathbf{3}_{10}$
$100_2 = \mathbf{4}_{10}$
$101_2 = \mathbf{5}_{10}$
$110_2 = \mathbf{6}_{10}$
$111_2 = \mathbf{7}_{10}$

A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

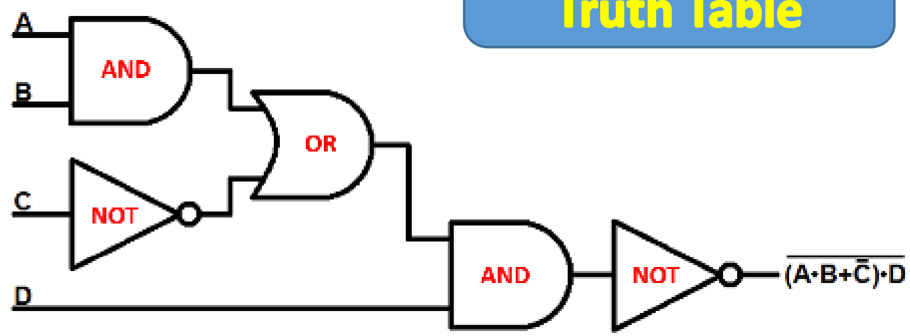
A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

With 3 *input* lines, there should be $\mathbf{2^3 = 8}$ rows in the circuit's truth table (**0-7**)

Lab 1 – Simple Circuit Trace

Complete this
Truth Table

	INPUT				OUTPUT
$Base_{10}$	A	B	C	D	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

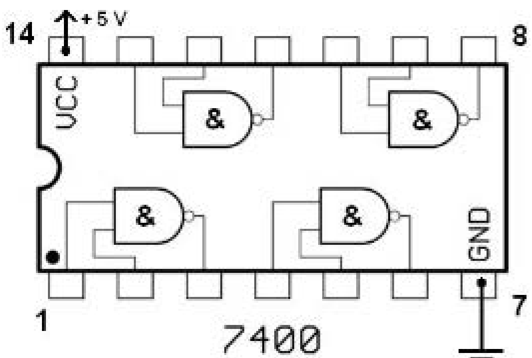


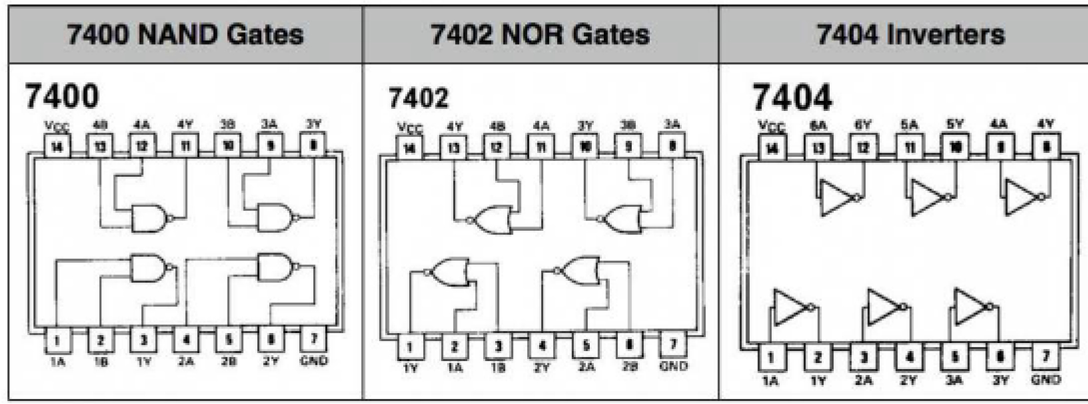
NOT Gate	
INPUT	OUTPUT
0	1
1	0

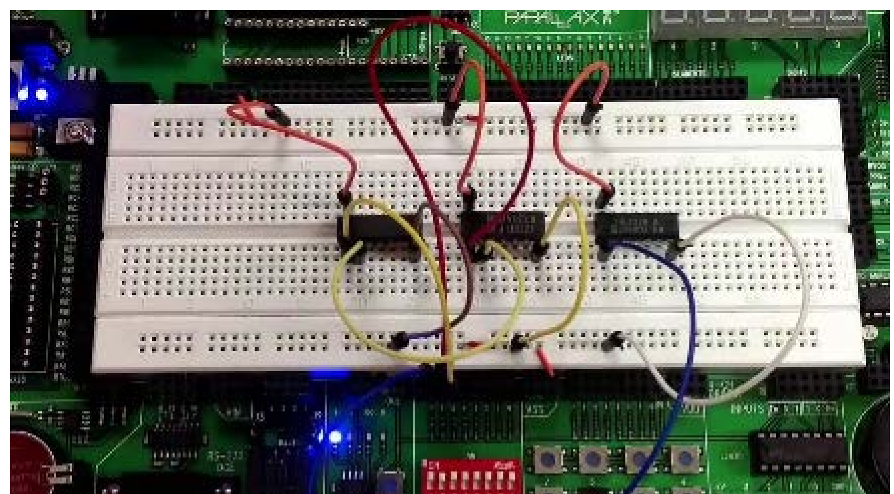
AND Gate		
INPUT		OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate		
INPUT		OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

Logic Gates in the Real World

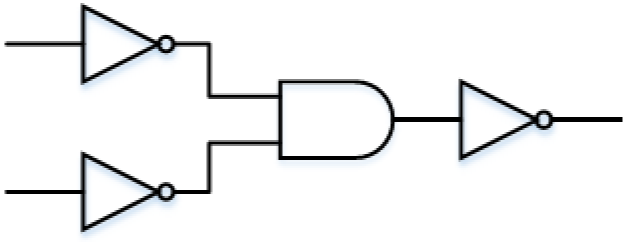




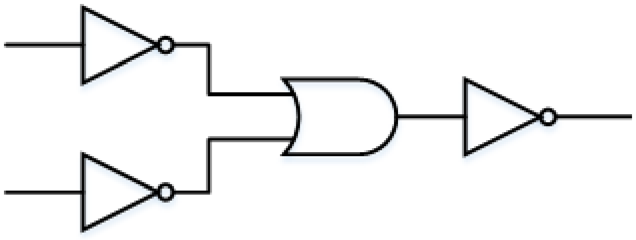


Gate Equivalence

- **Augustus De Morgan's** laws:
    - Can make an AND gate from 3 NOTs and 1 OR
    - Can make an OR gate from 3 NOTs and 1 AND
    - Simply invert both inputs and the output!

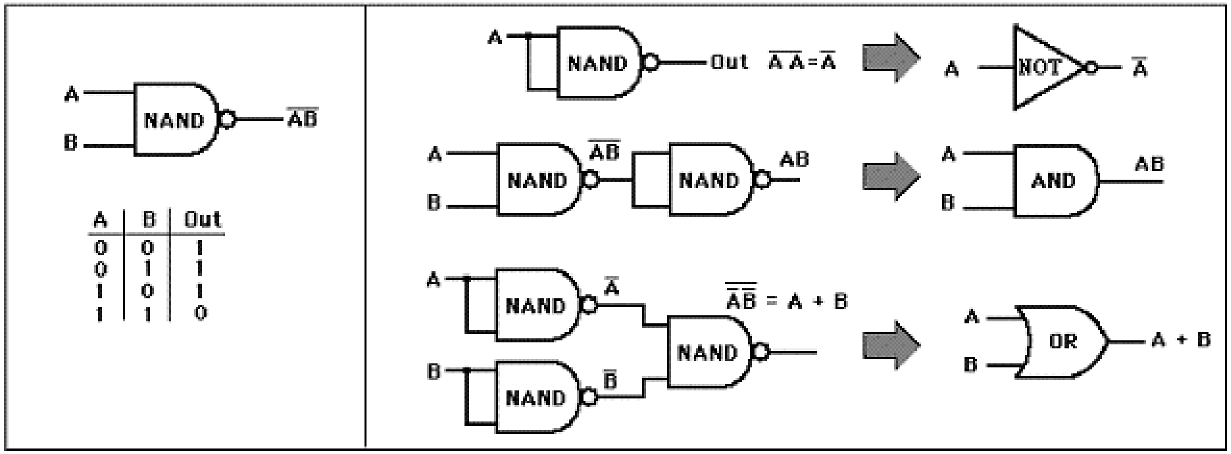


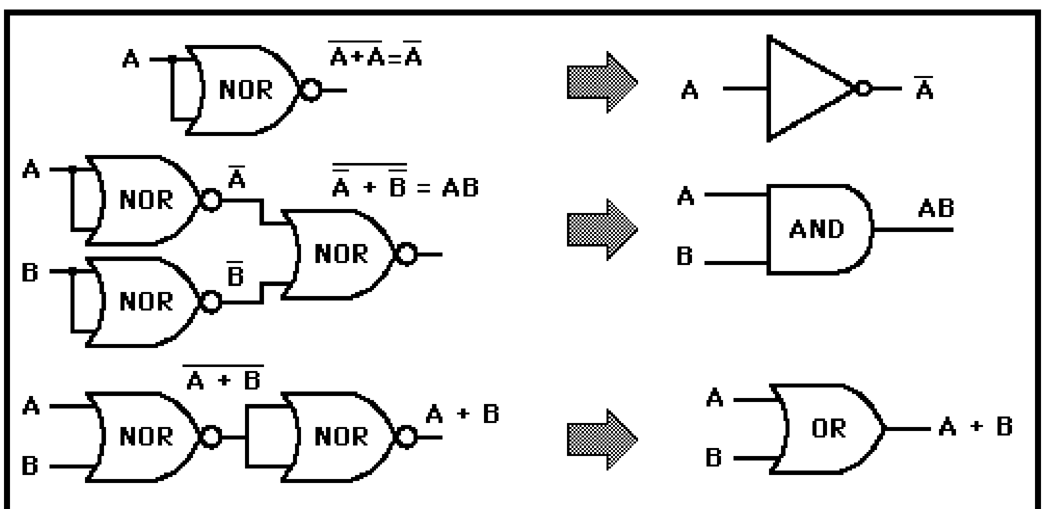
OR from AND



AND from OR

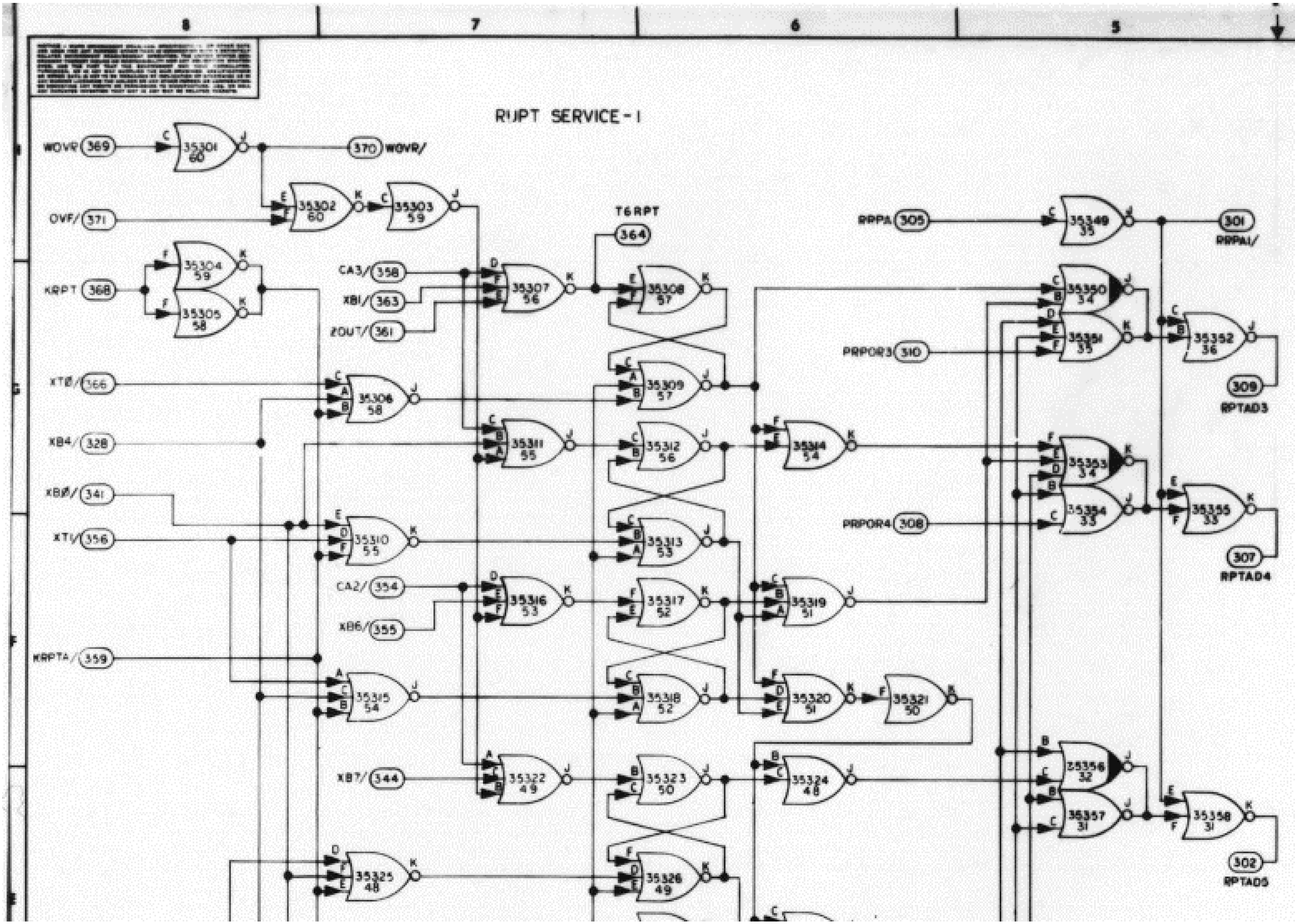
De Morgan's Laws



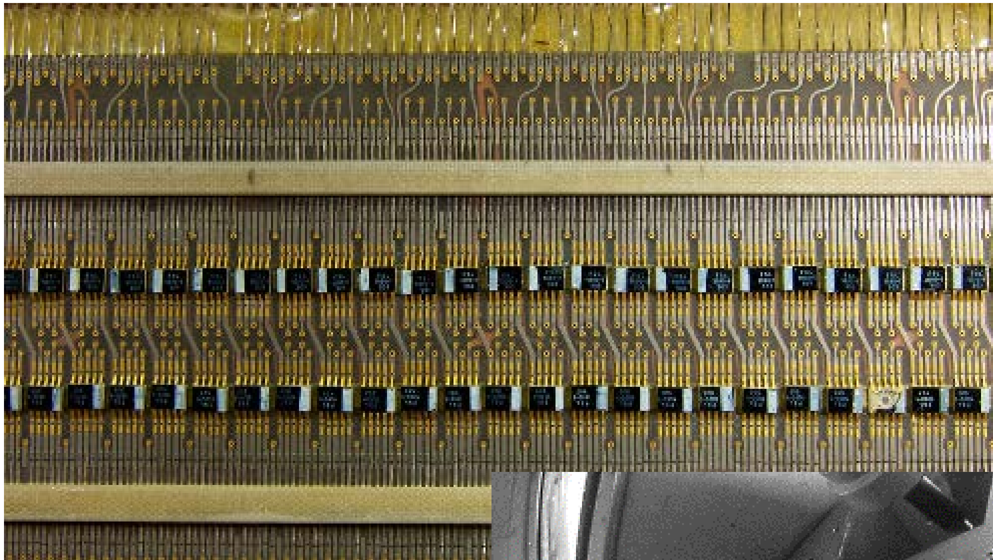


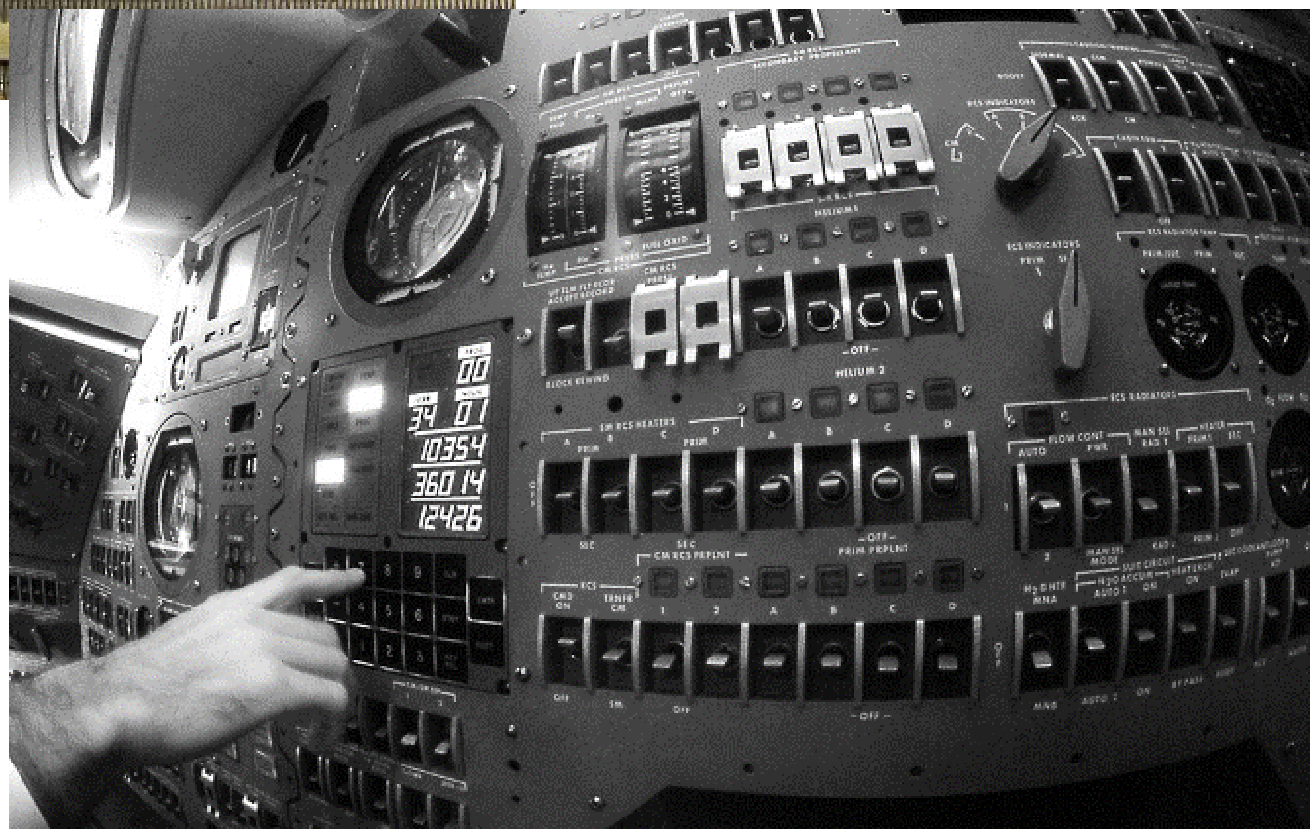
We can build any circuit using just NAND or NOR gates!

Apollo Guidance Computer

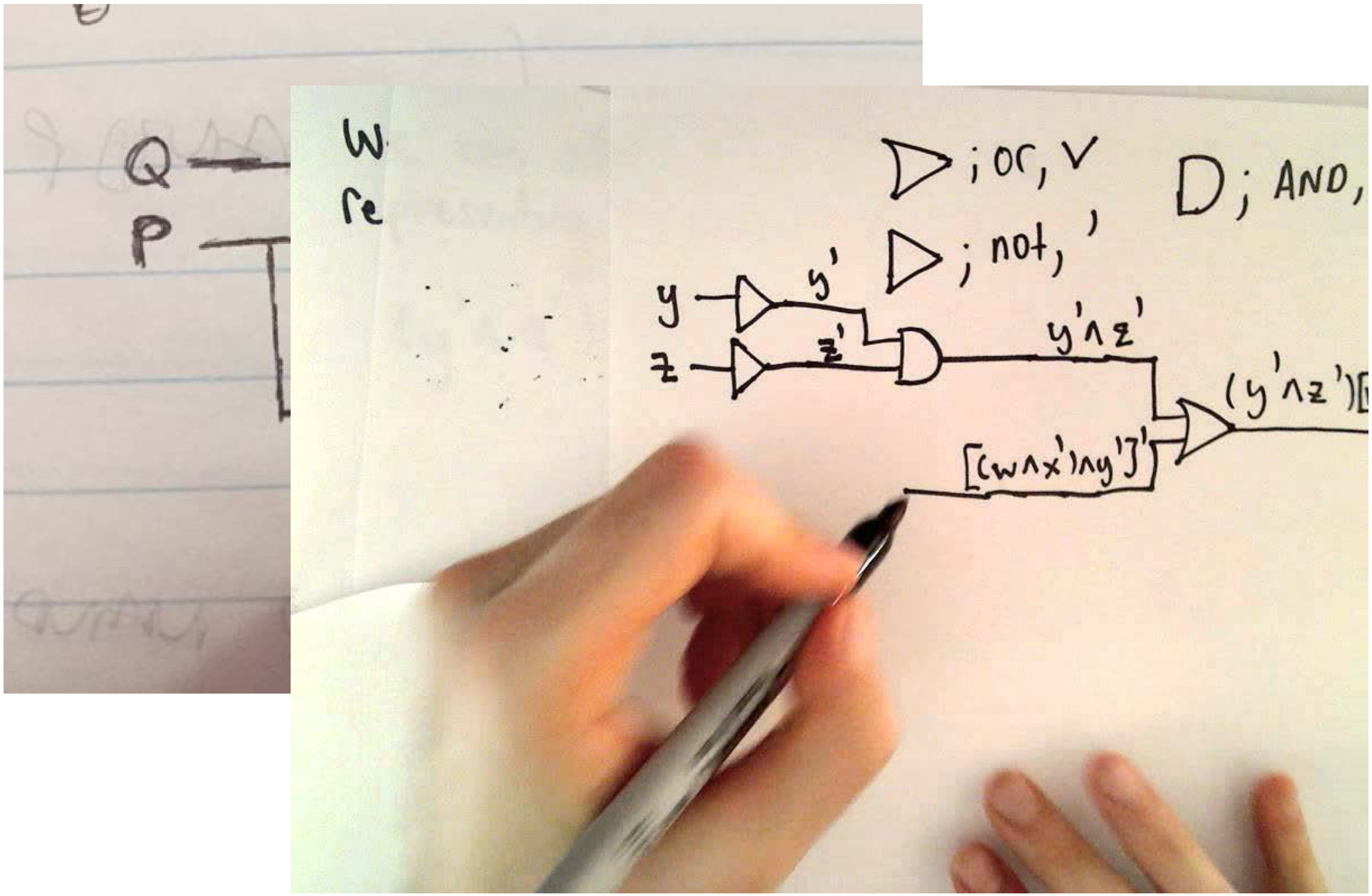


Apollo Guidance Computer





Drawing A Digital Circuit



Logisim

a graphical tool for designing and simulating logic circuits

http://www.cburch.com/logisim

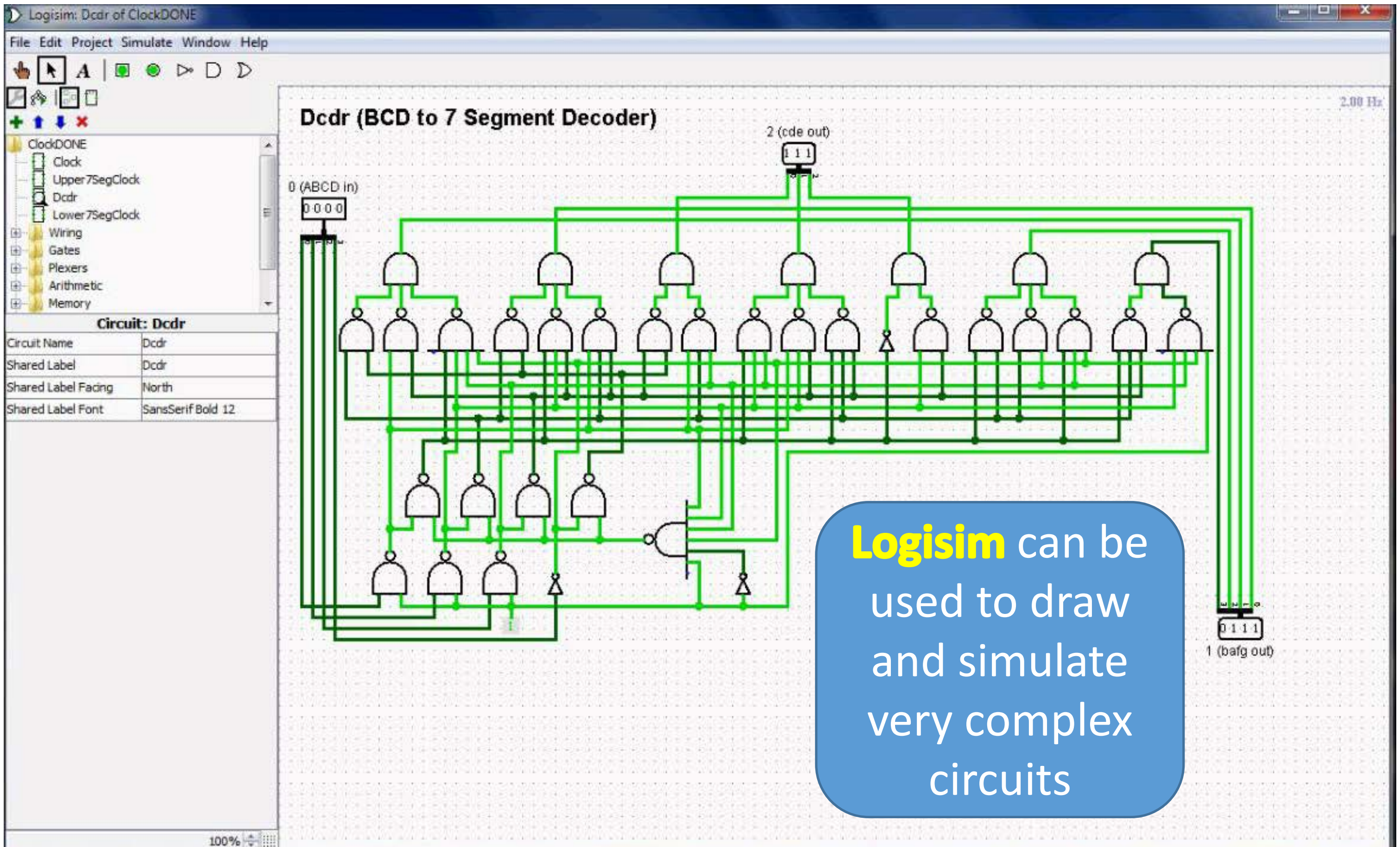
Logisim is an educational tool for designing and simulating digital logic circuits. With its simple toolbar interface and simulation of circuits as you build them, it is simple enough to facilitate learning the most basic concepts related to logic circuits. With the capacity to build larger circuits from smaller subcircuits, and to draw bundles of wires with a single mouse drag, Logisim can be used (and is used) to design and simulate entire CPUs for educational purposes.

Logisim is used by students at colleges and universities around the world in many types of classes, ranging from a brief unit on logic in general-education computer science surveys, to computer organization courses, to full-semester courses on computer architecture.

Logisim

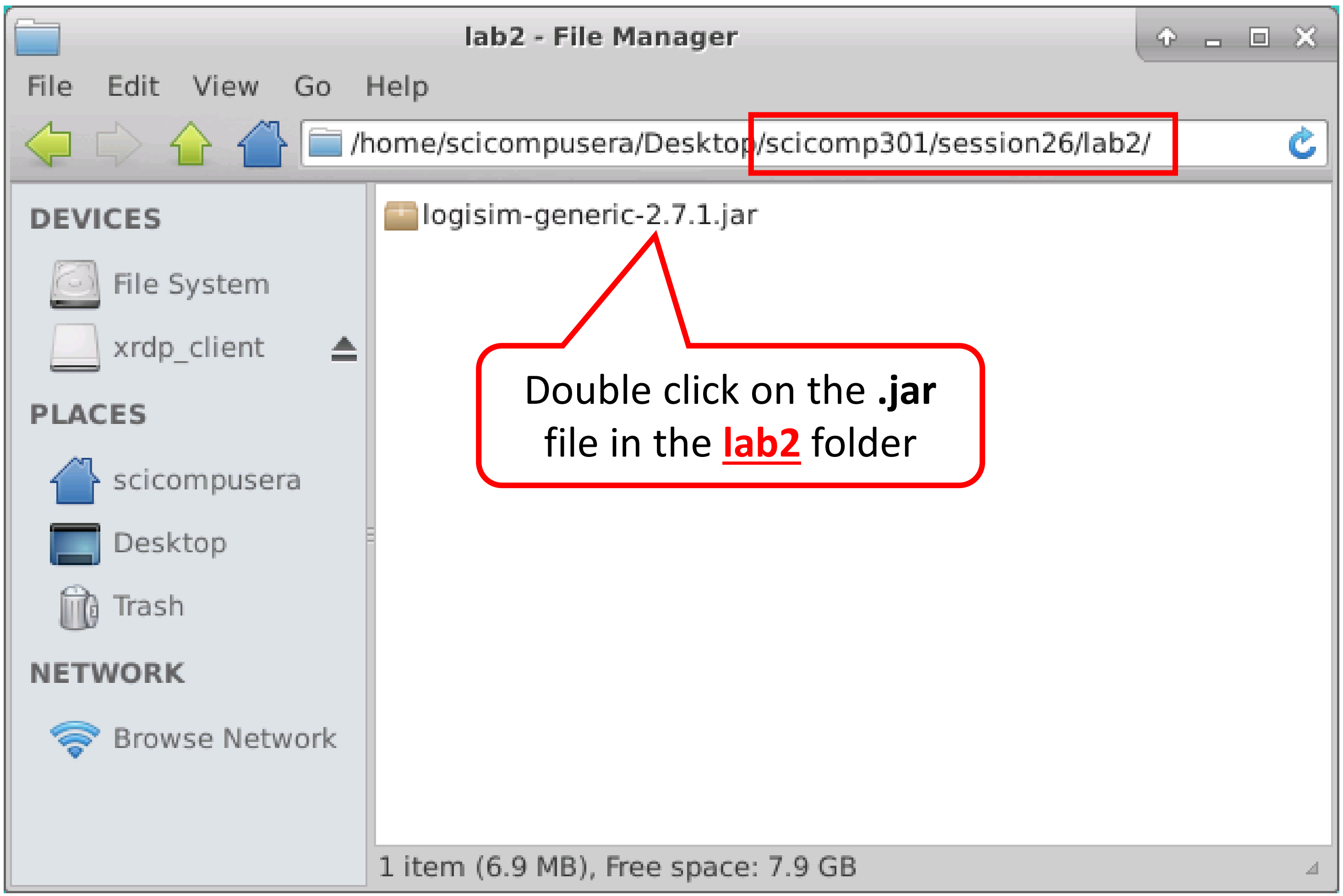
a graphical tool for designing and simulating logic circuits

http://www.cburch.com/logisim

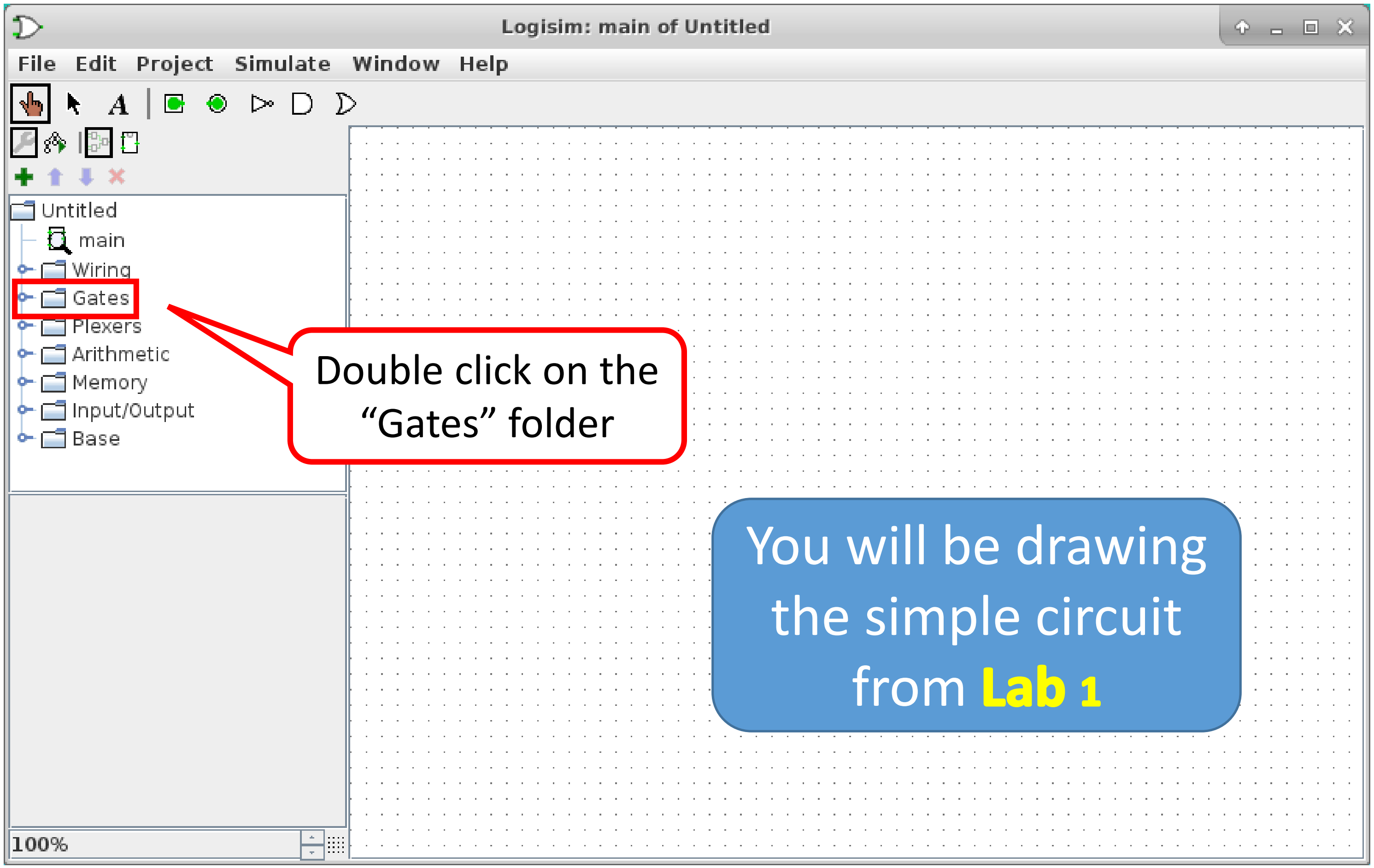


Logisim can be used to draw and simulate very complex circuits

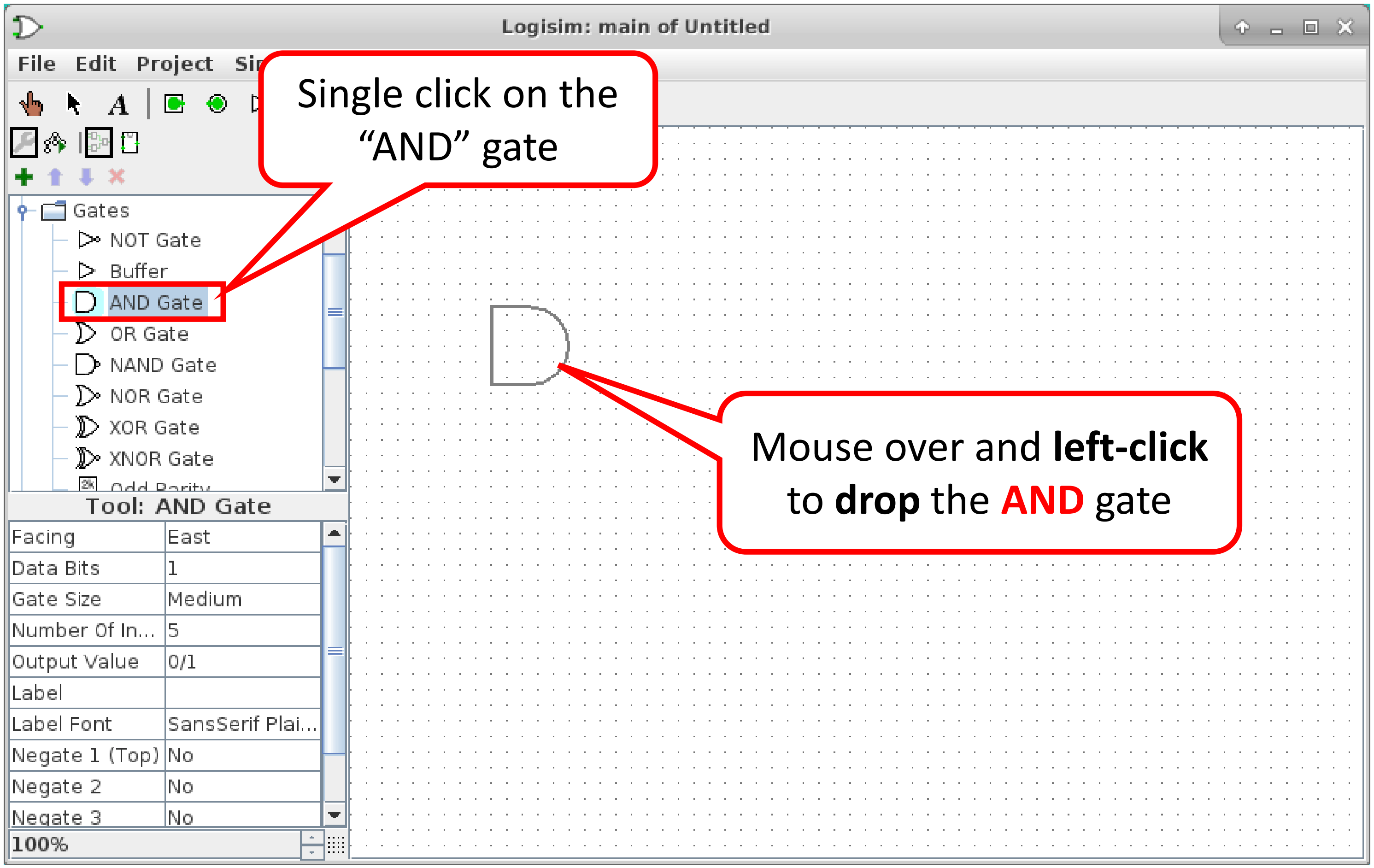
Lab 2 – Simple Circuit in Logisim



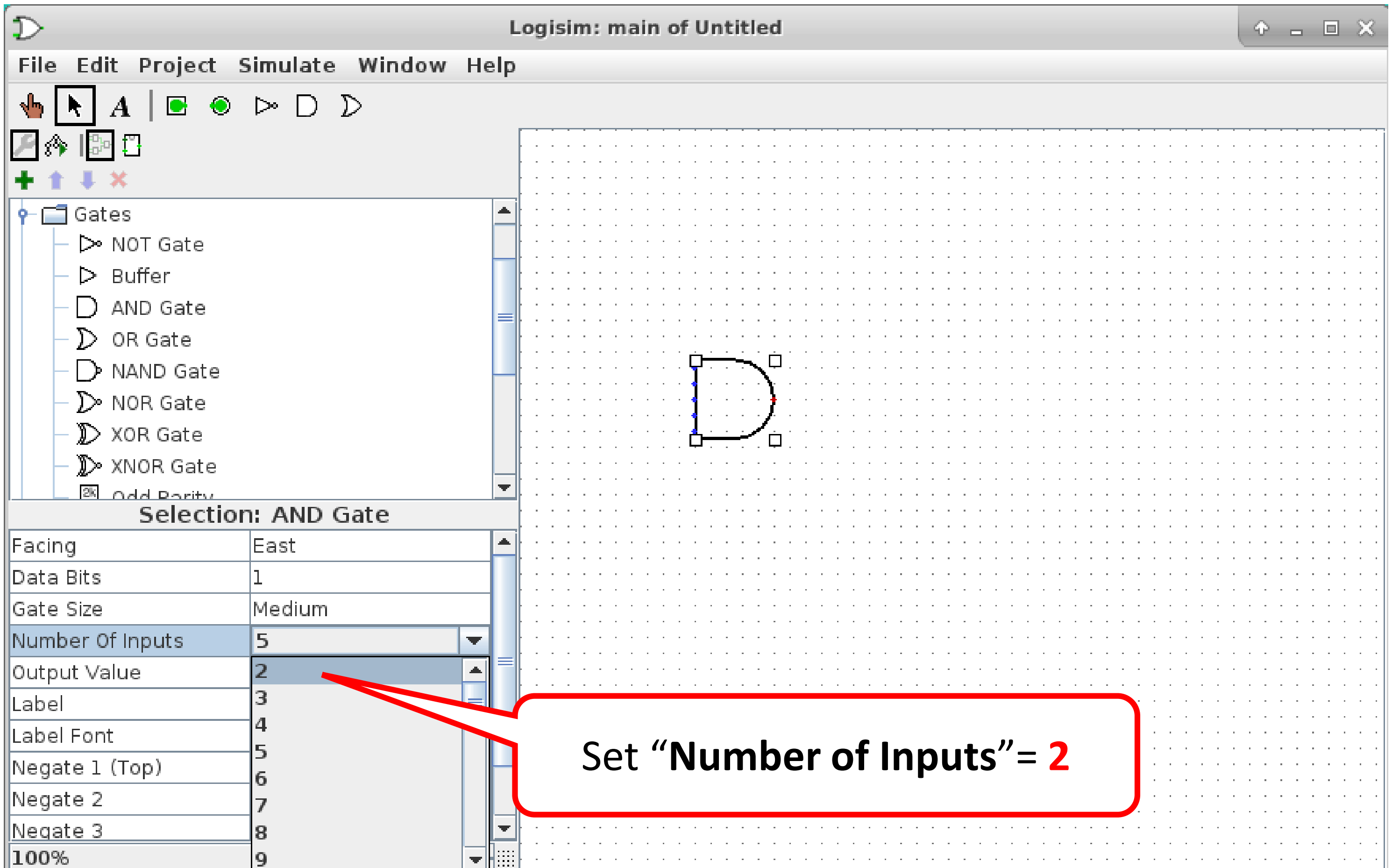
Lab 2 – Simple Circuit in Logisim



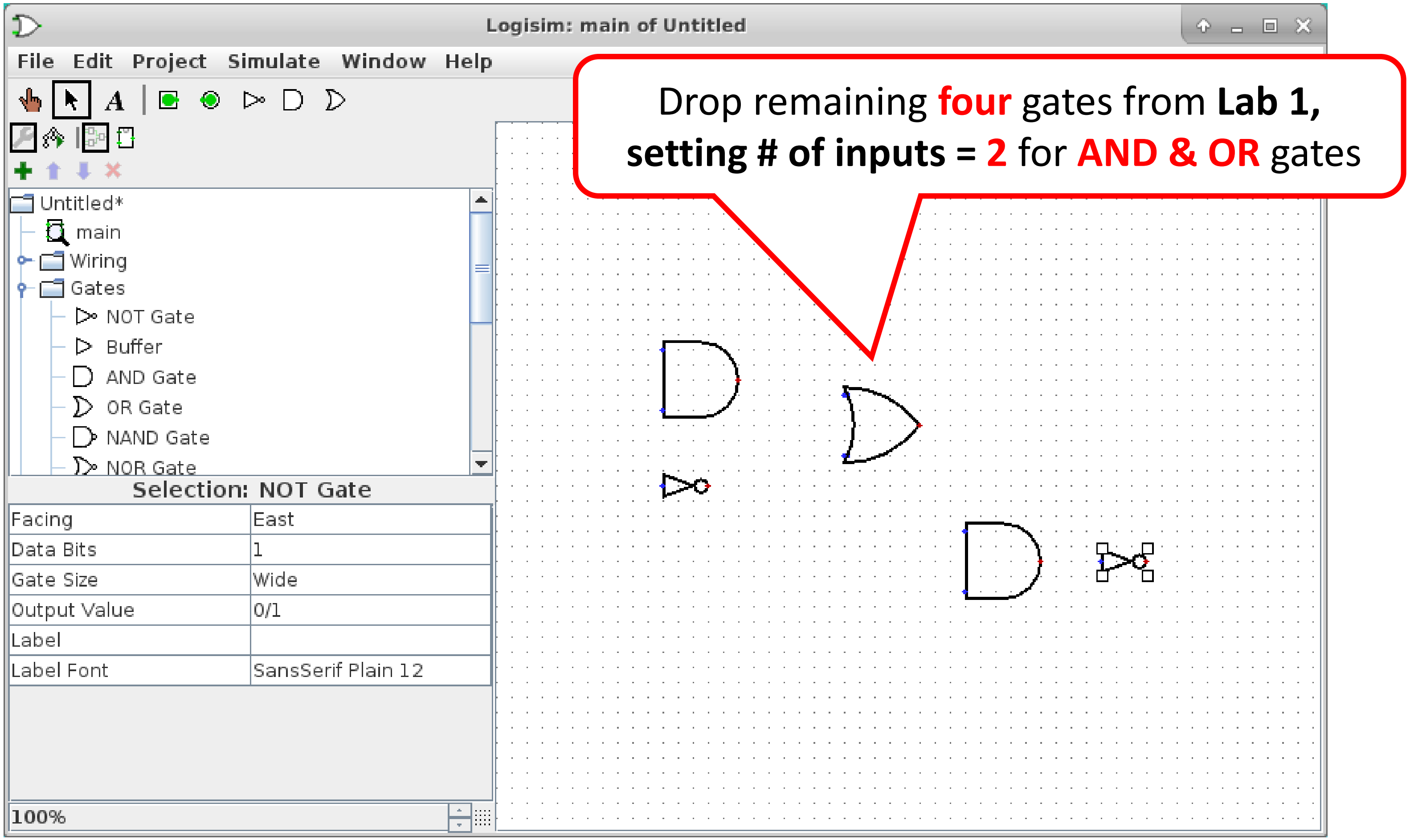
Lab 2 – Simple Circuit in Logisim



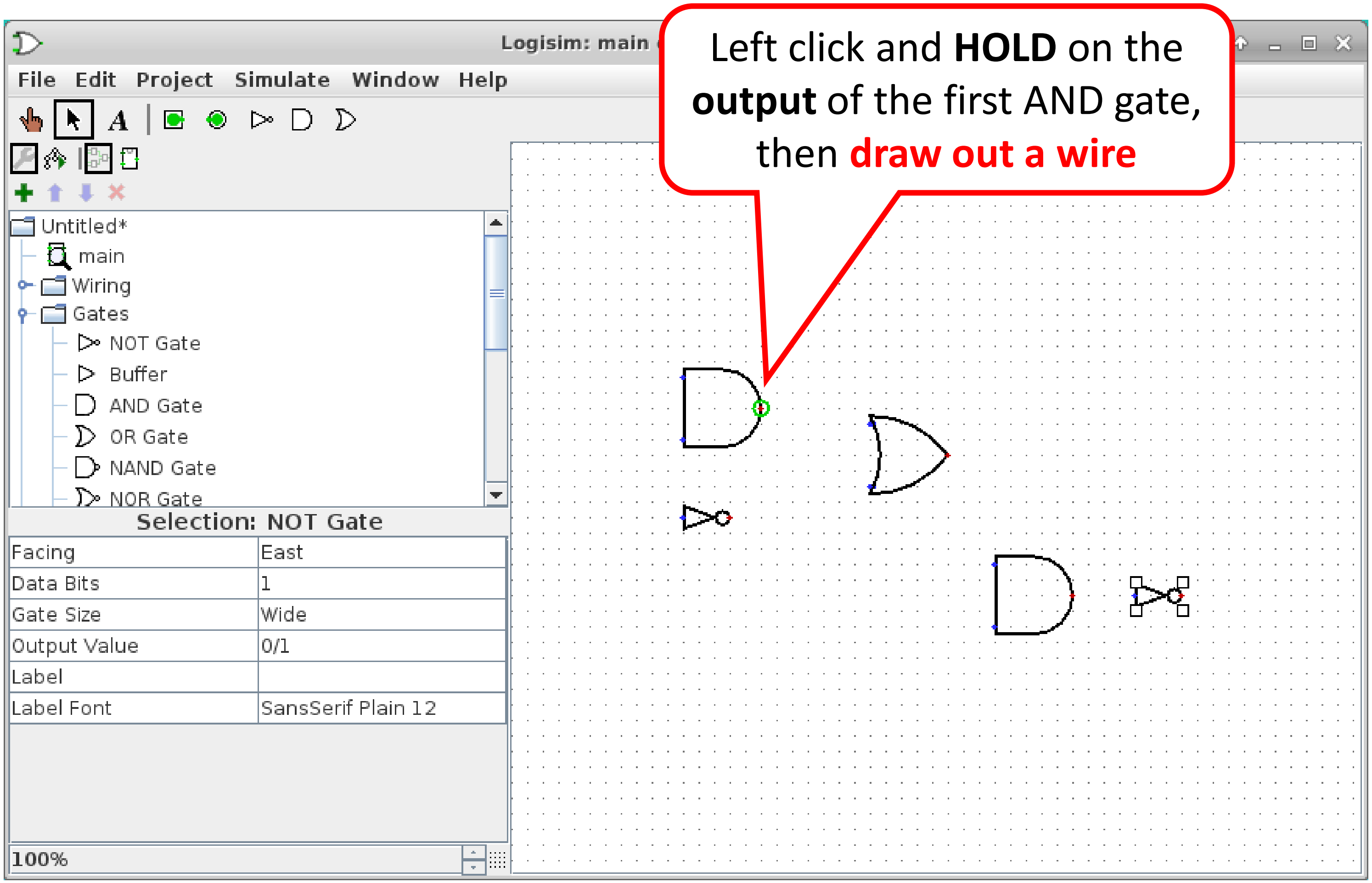
Lab 2 – Simple Circuit in Logisim



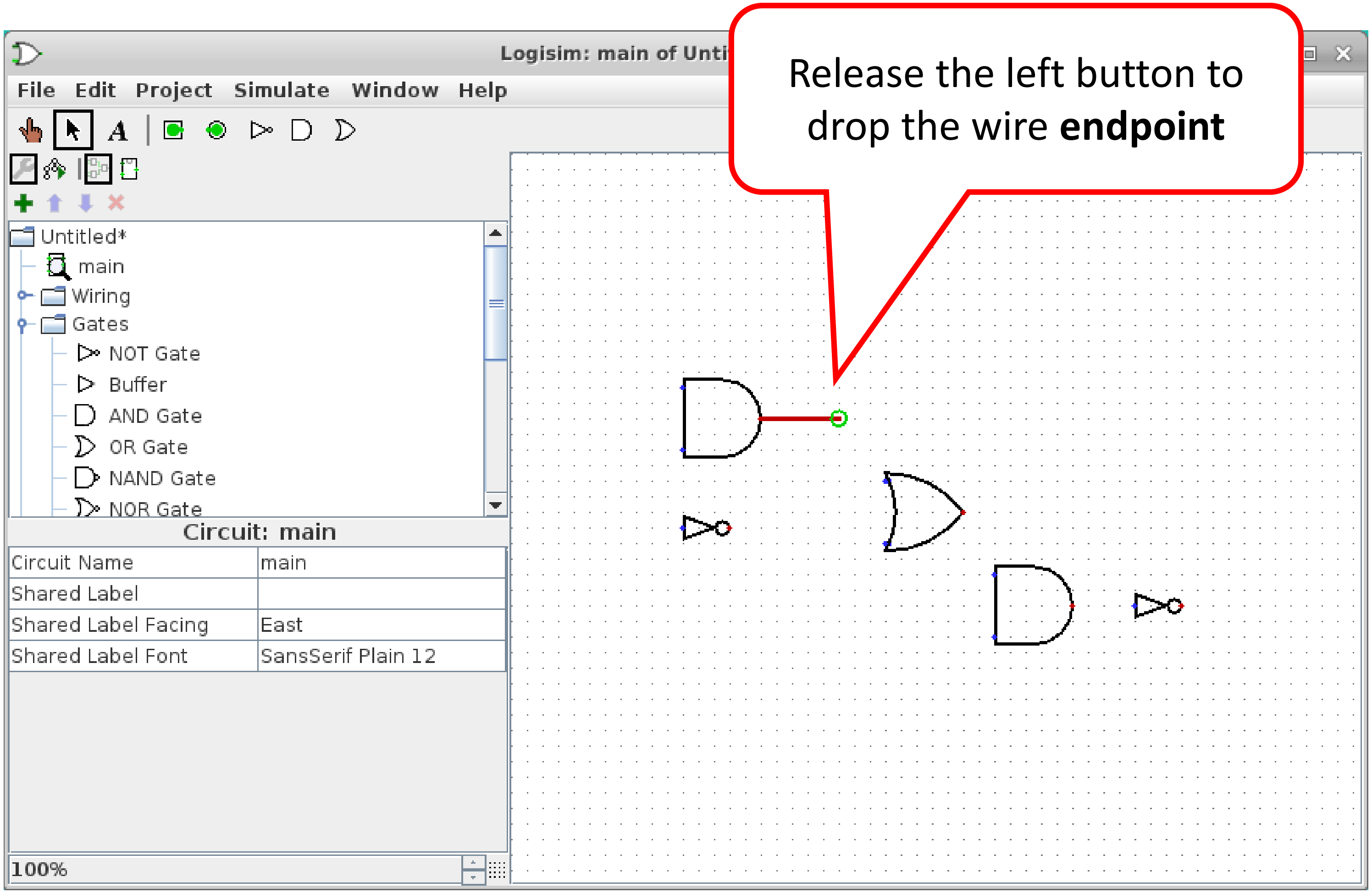
Lab 2 – Simple Circuit in Logisim



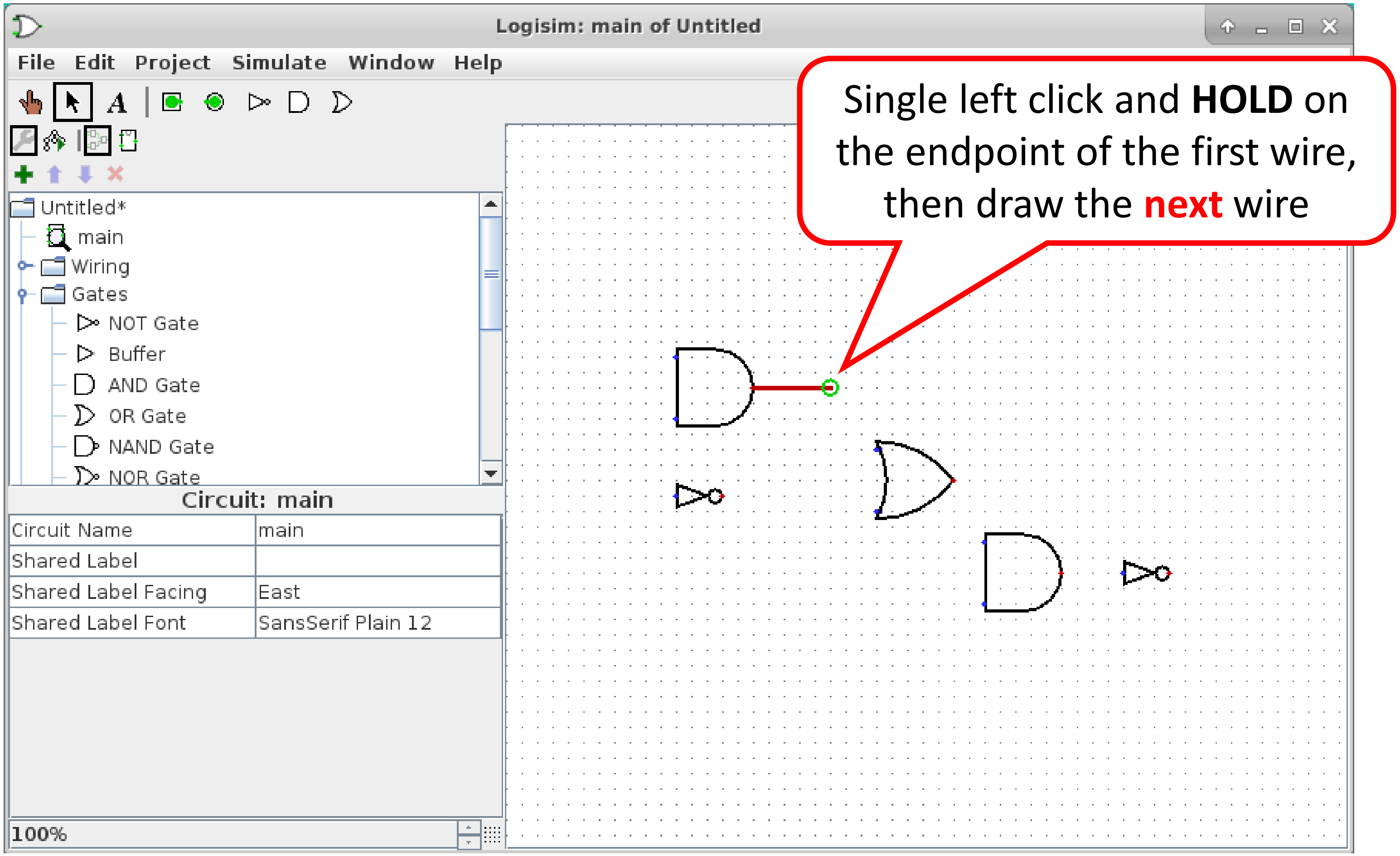
Lab 2 – Simple Circuit in Logisim



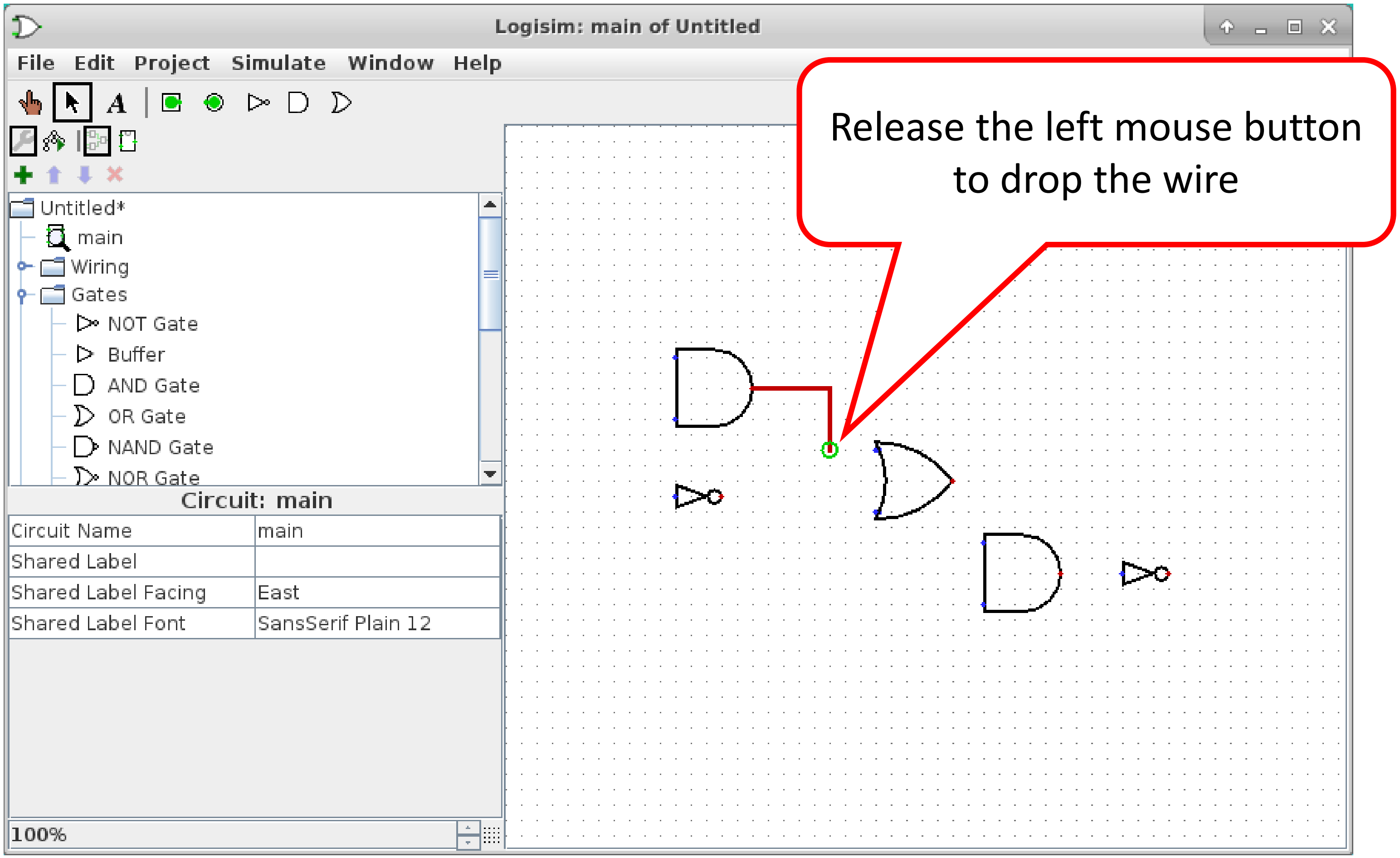
Lab 2 – Simple Circuit in Logisim



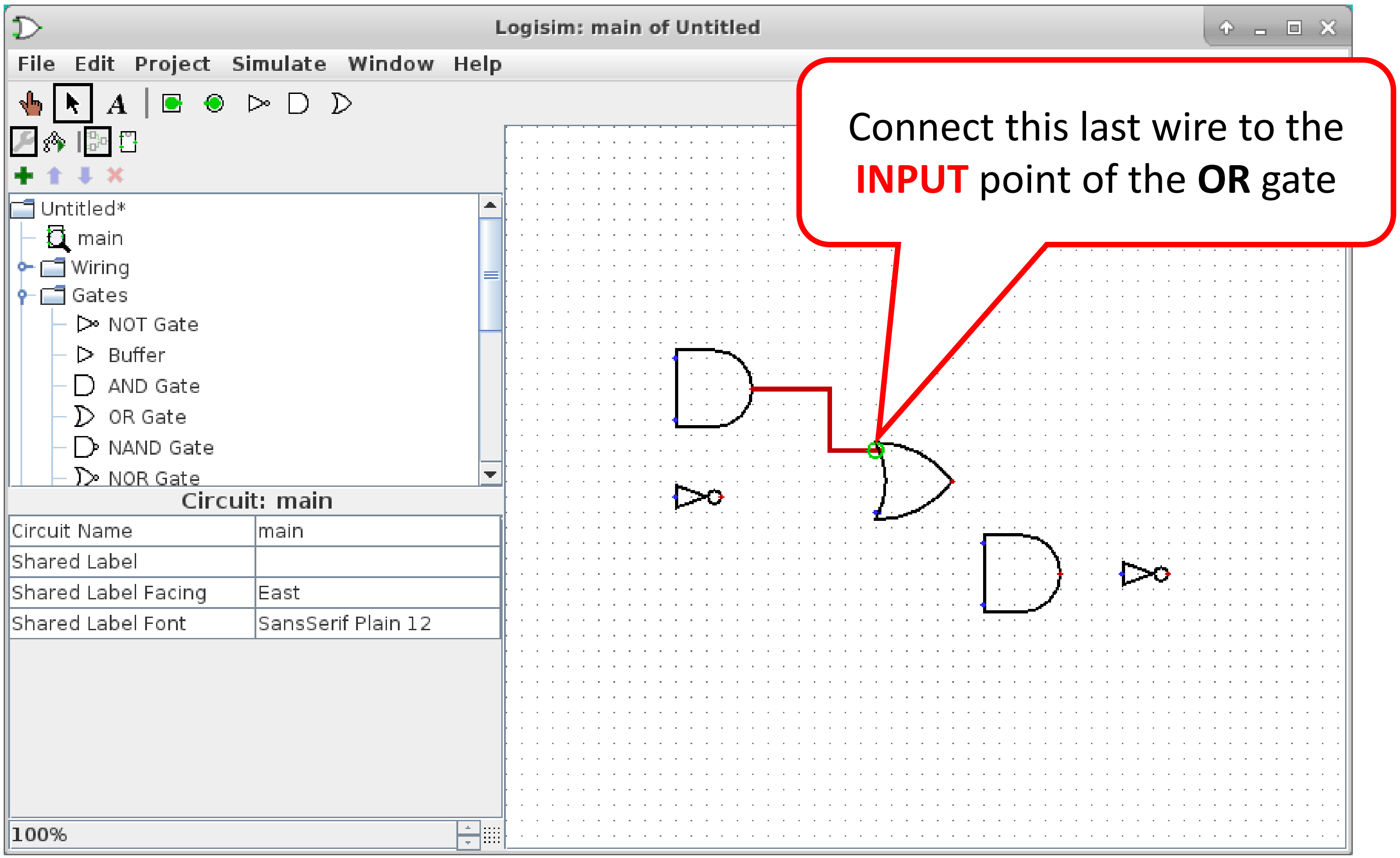
Lab 2 – Simple Circuit in Logisim



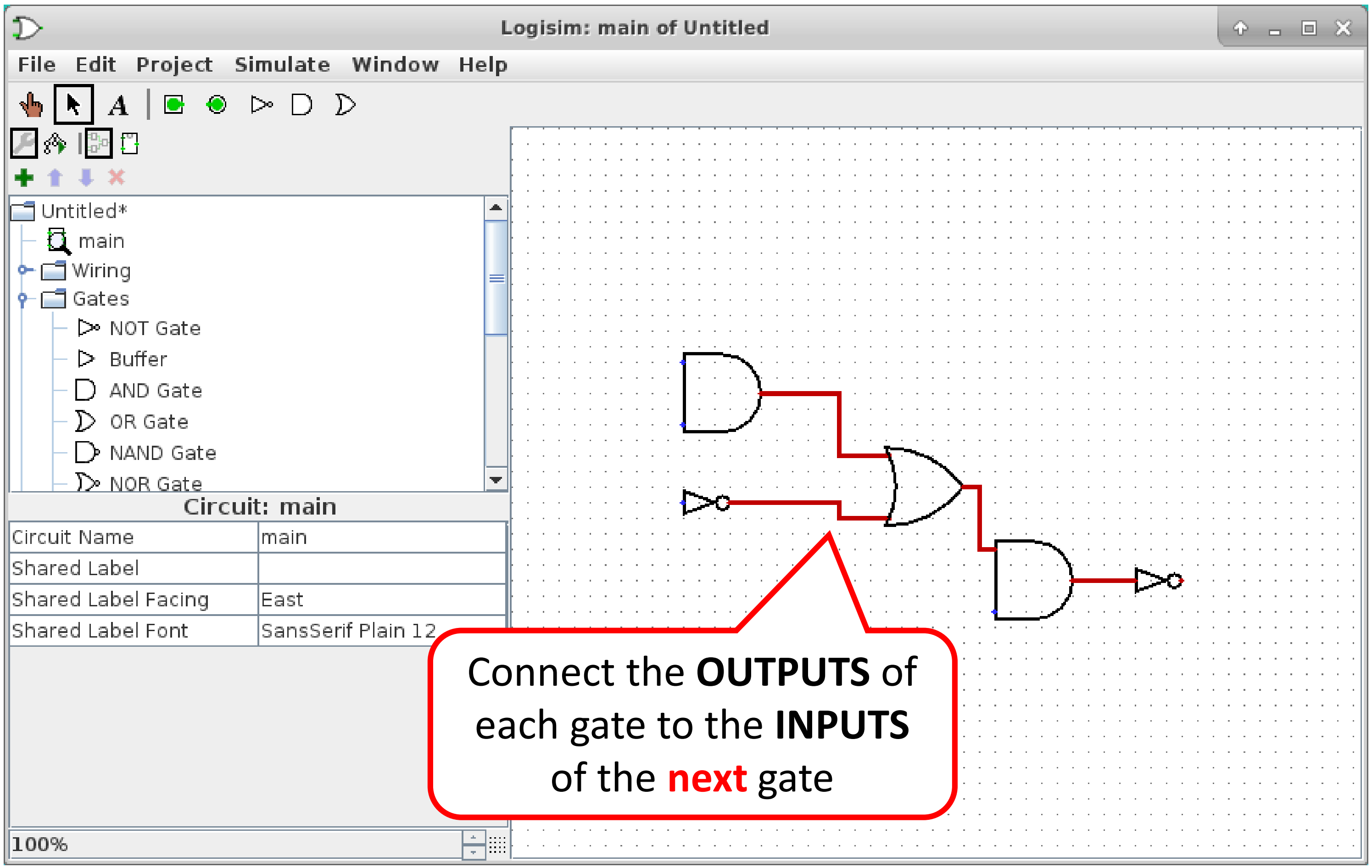
Lab 2 – Simple Circuit in Logisim



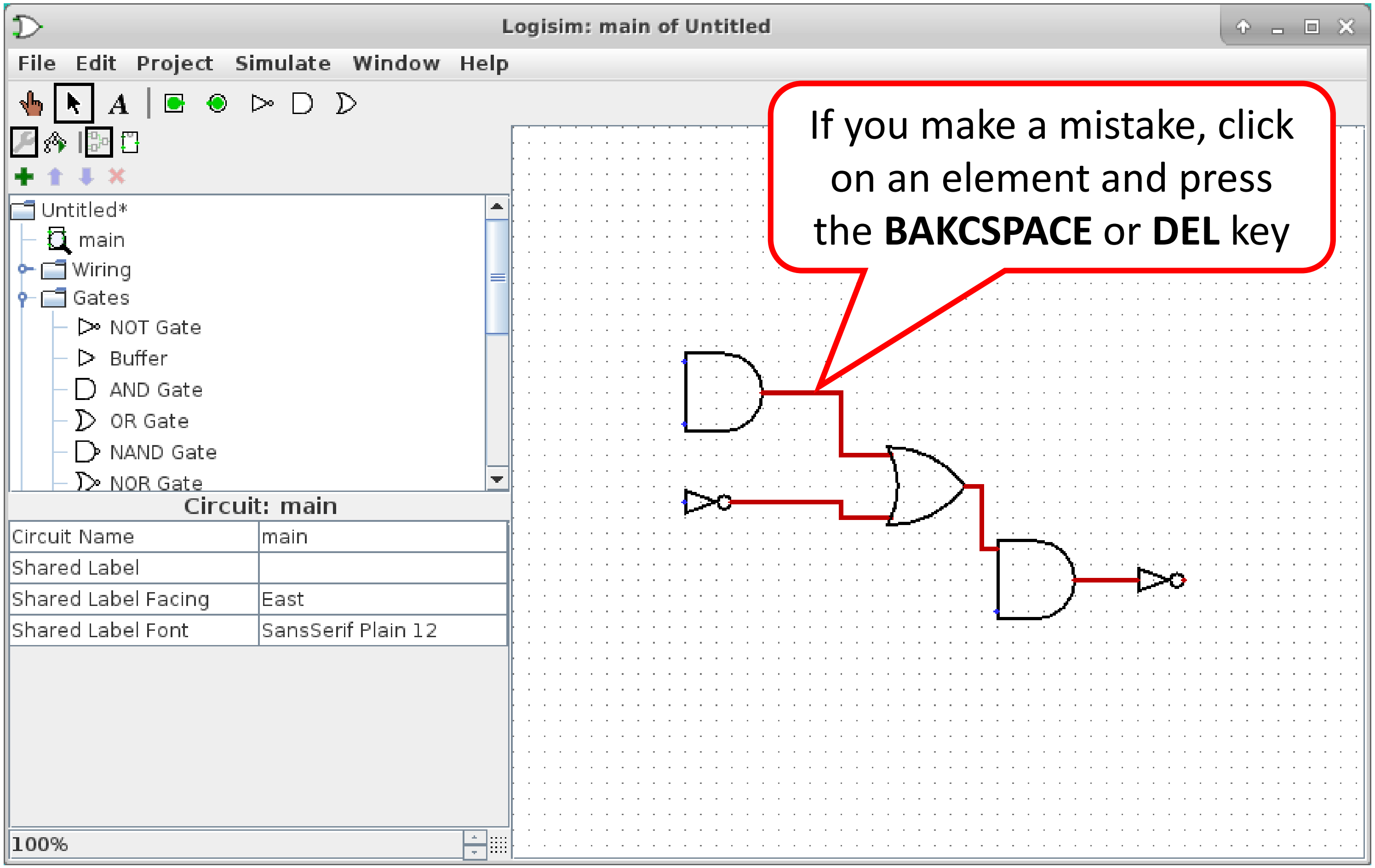
Lab 2 – Simple Circuit in Logisim



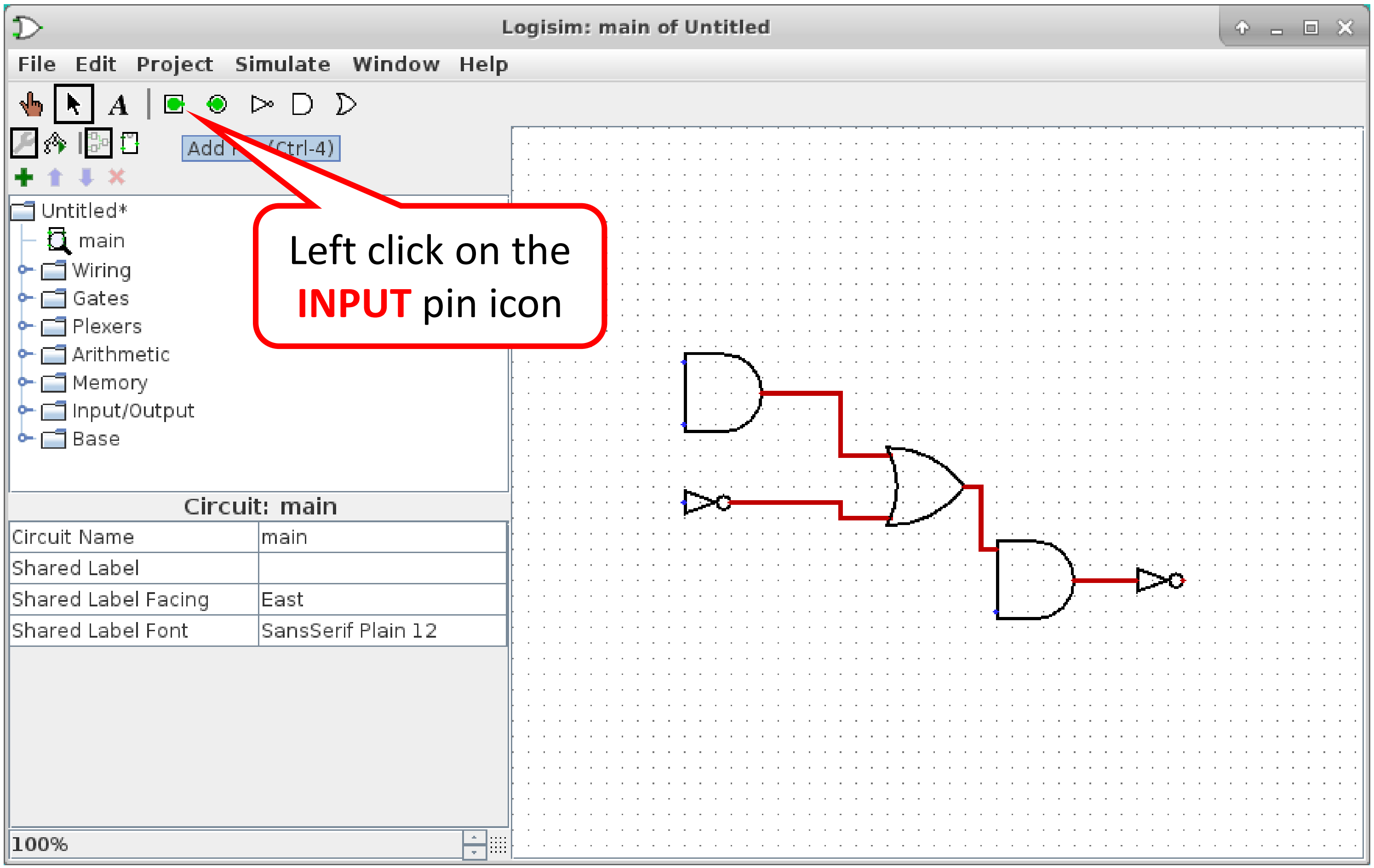
Lab 2 – Simple Circuit in Logisim



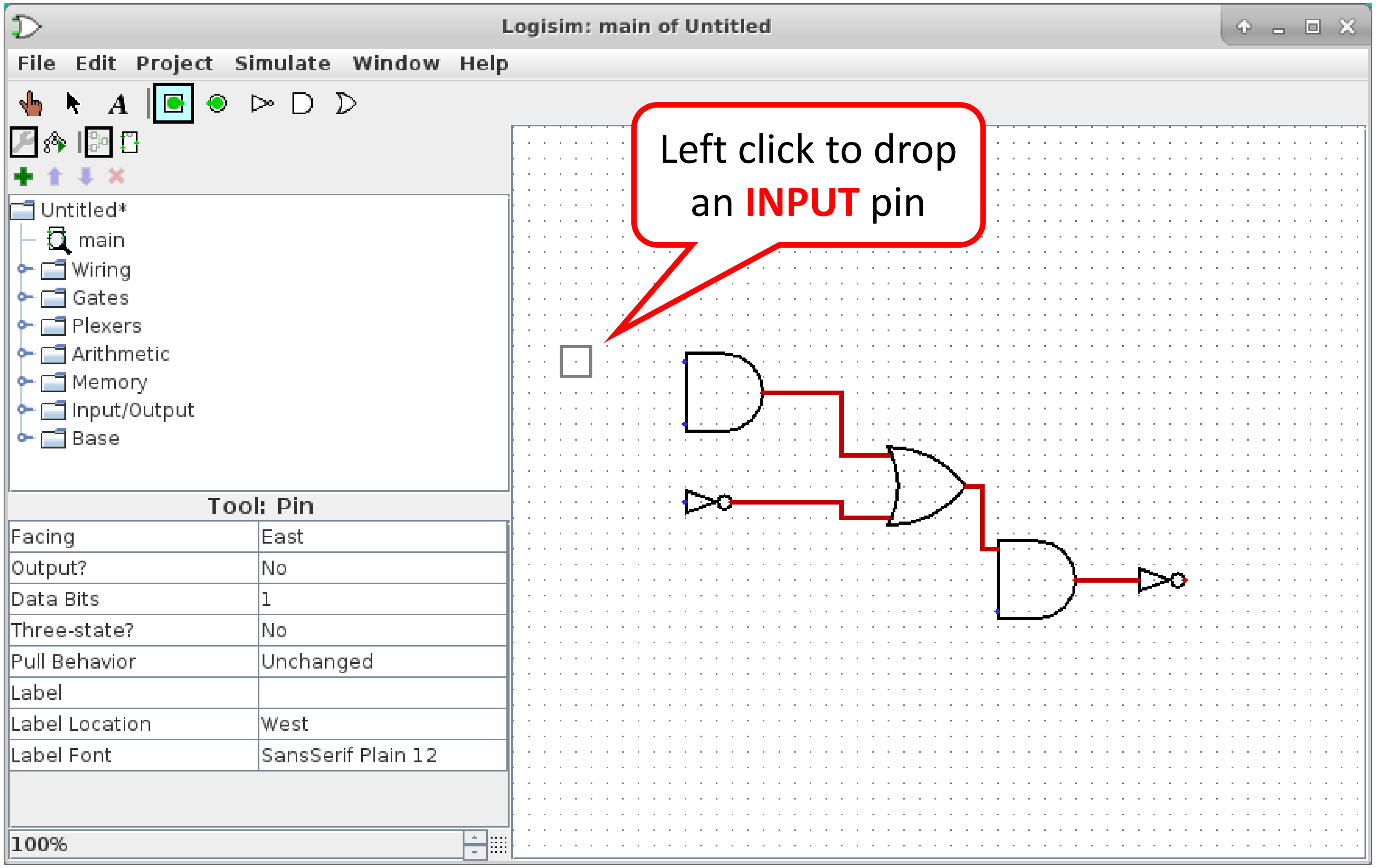
Lab 2 – Simple Circuit in Logisim



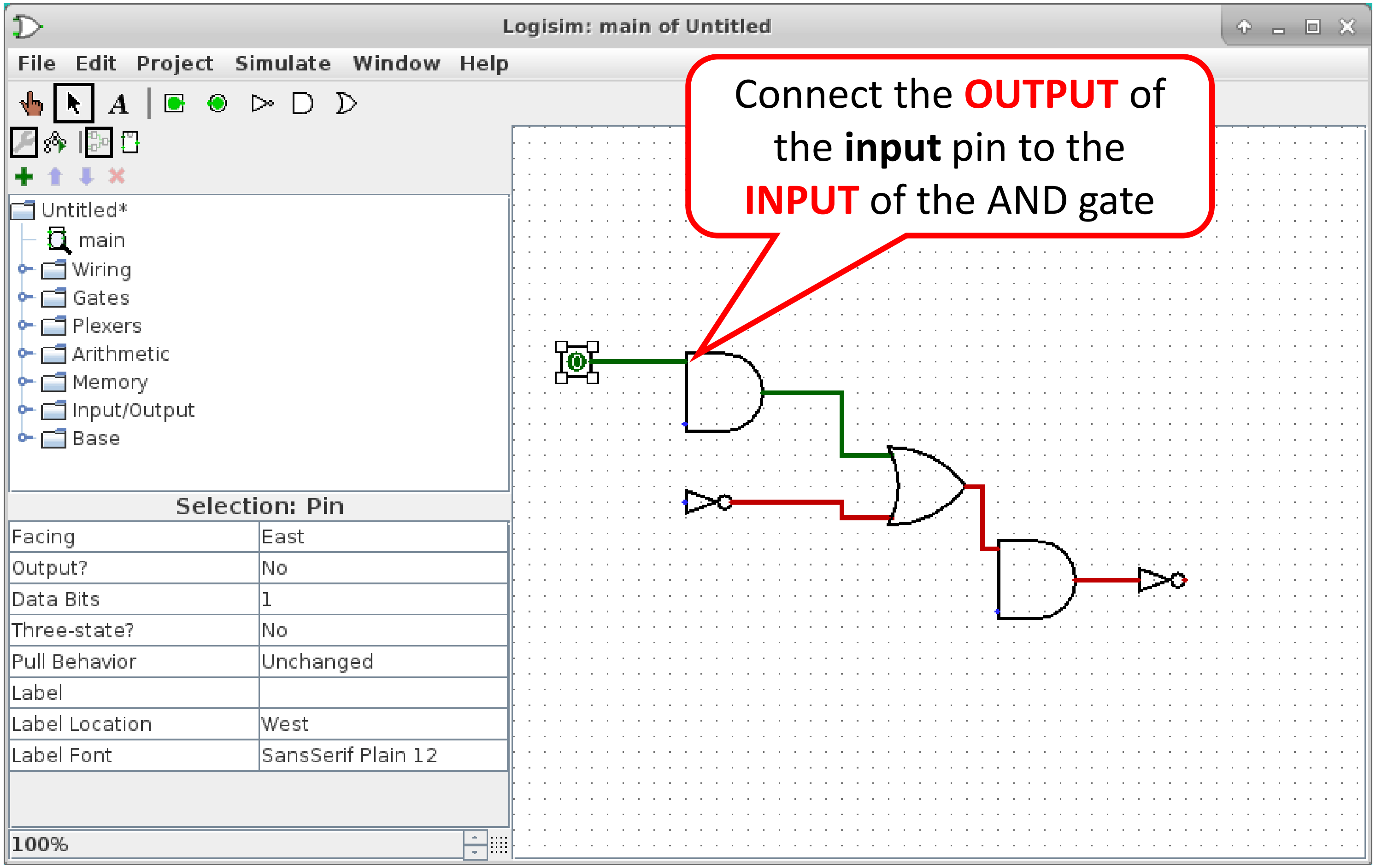
Lab 2 – Simple Circuit in Logisim



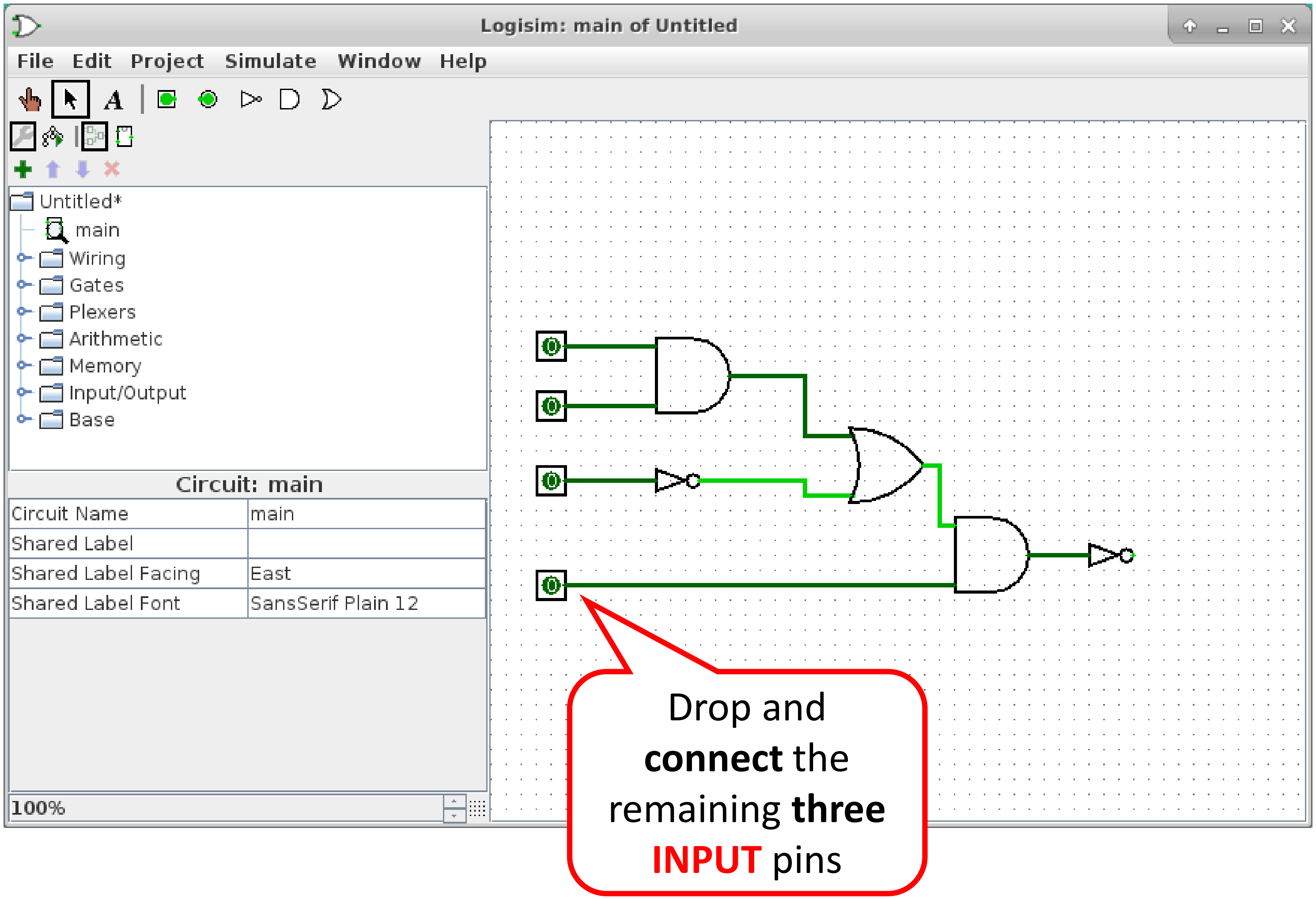
Lab 2 – Simple Circuit in Logisim



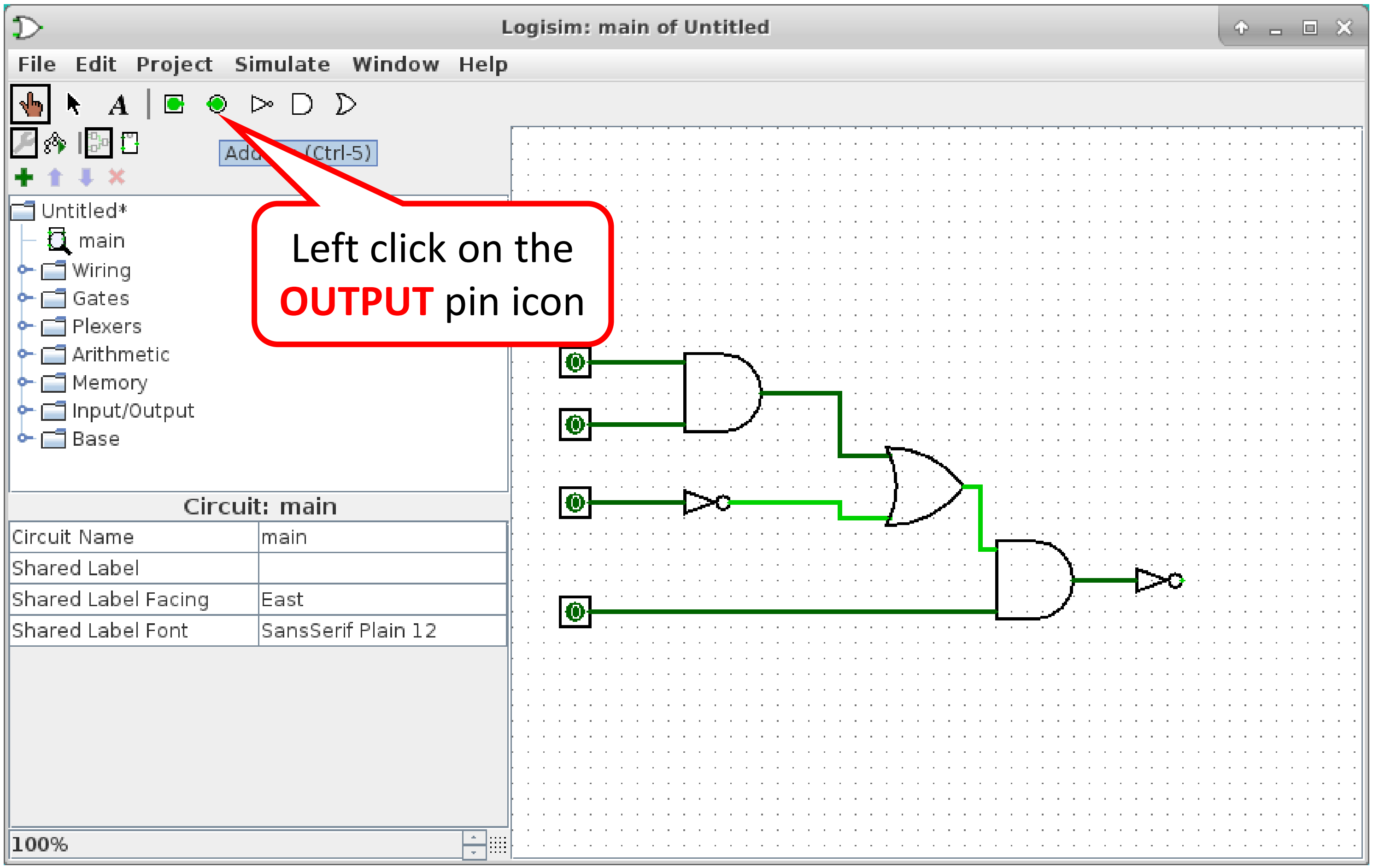
Lab 2 – Simple Circuit in Logisim



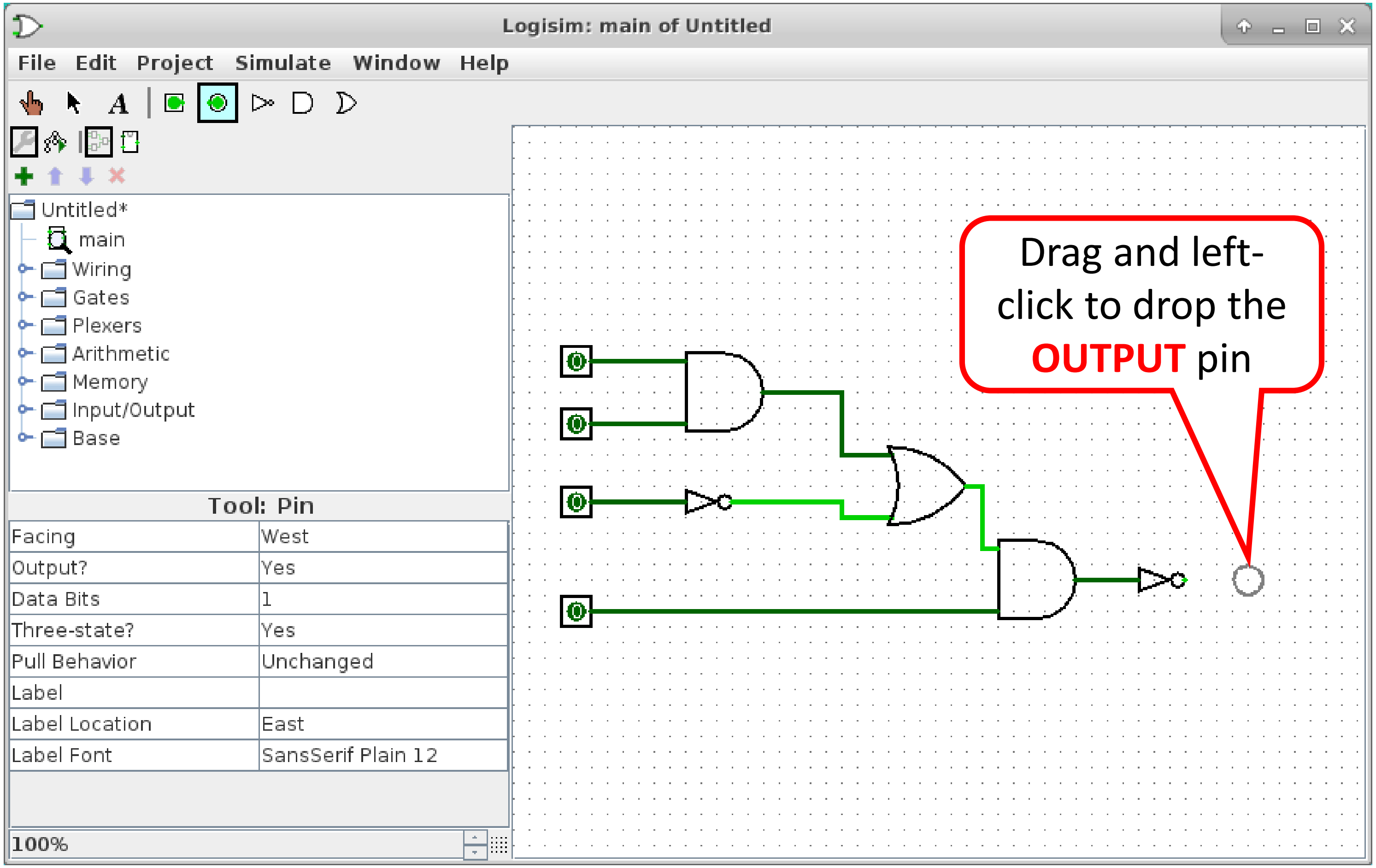
Lab 2 – Simple Circuit in Logisim



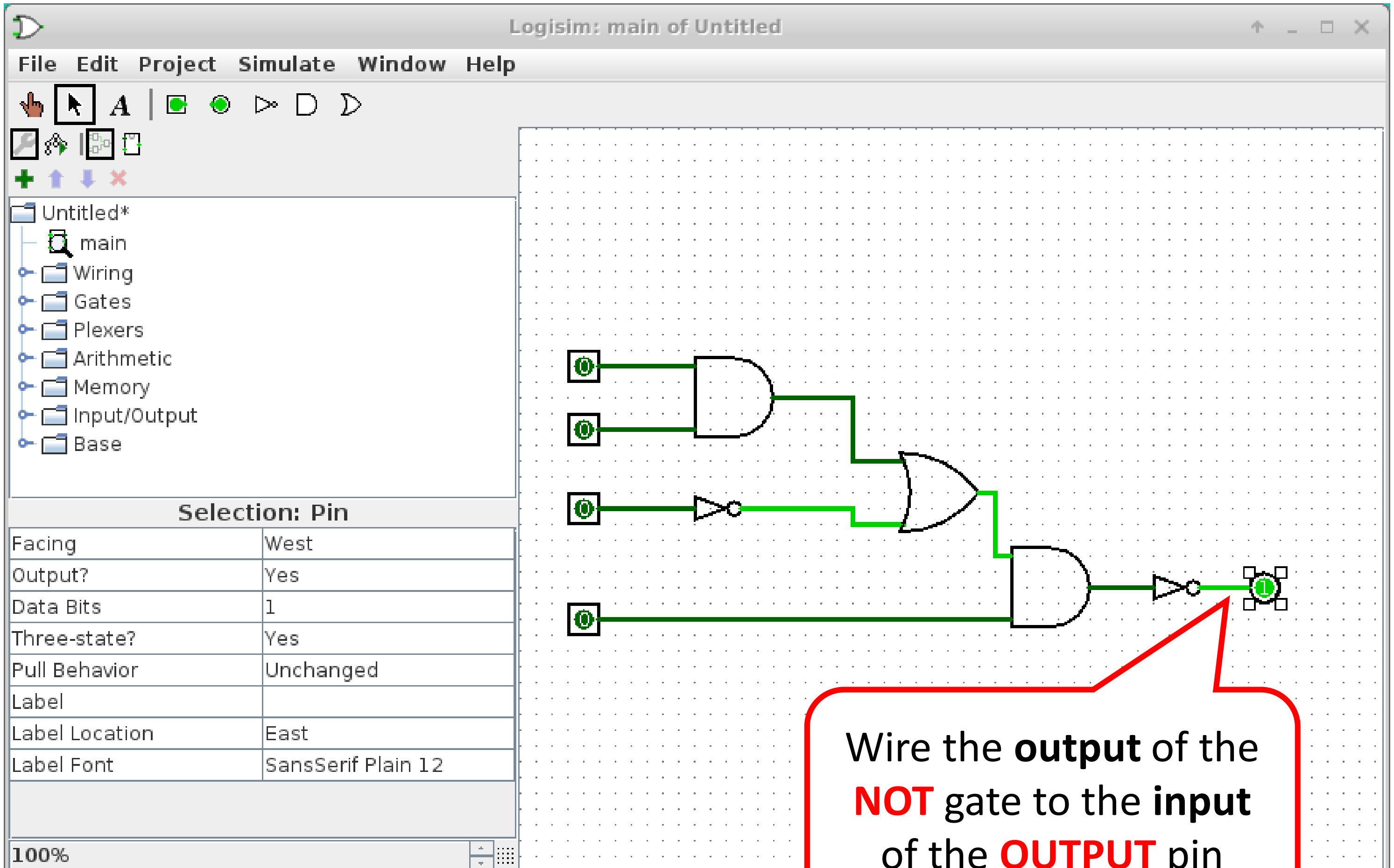
Lab 2 – Simple Circuit in Logisim



Lab 2 – Simple Circuit in Logisim



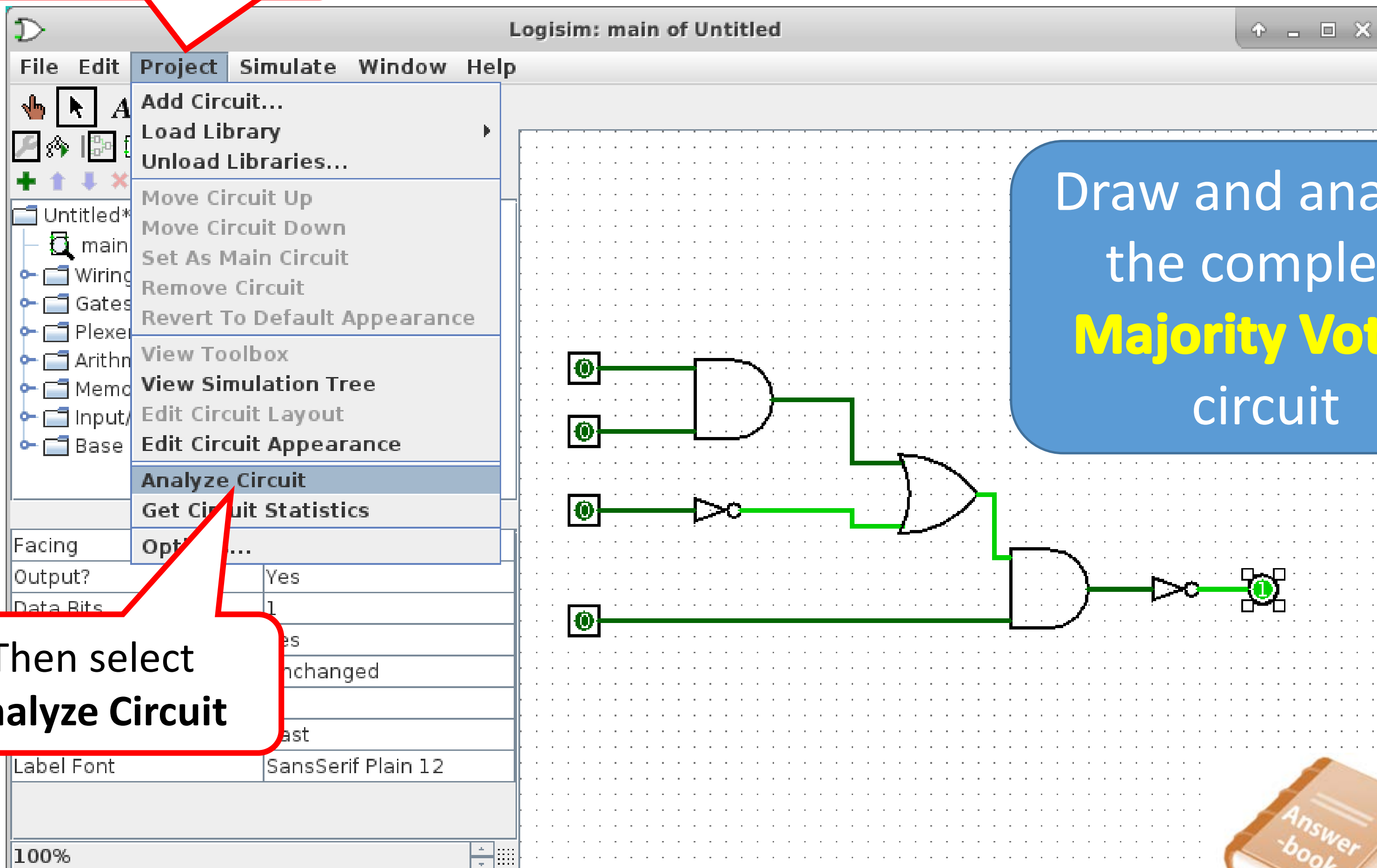
Lab 2 – Simple Circuit in Logisim



Wire the **output** of the **NOT** gate to the **input** of the **OUTPUT** pin

Lab 2 – Simple Circuit in Logisim

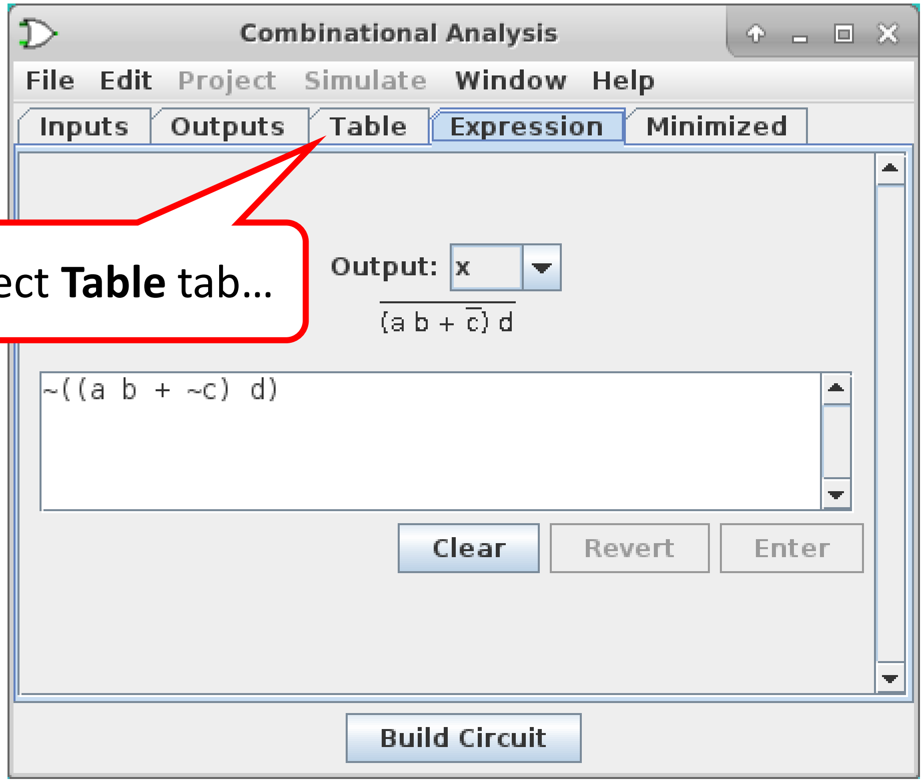
Select **Project** menu…



Draw and analyze the complete **Majority Voting** circuit

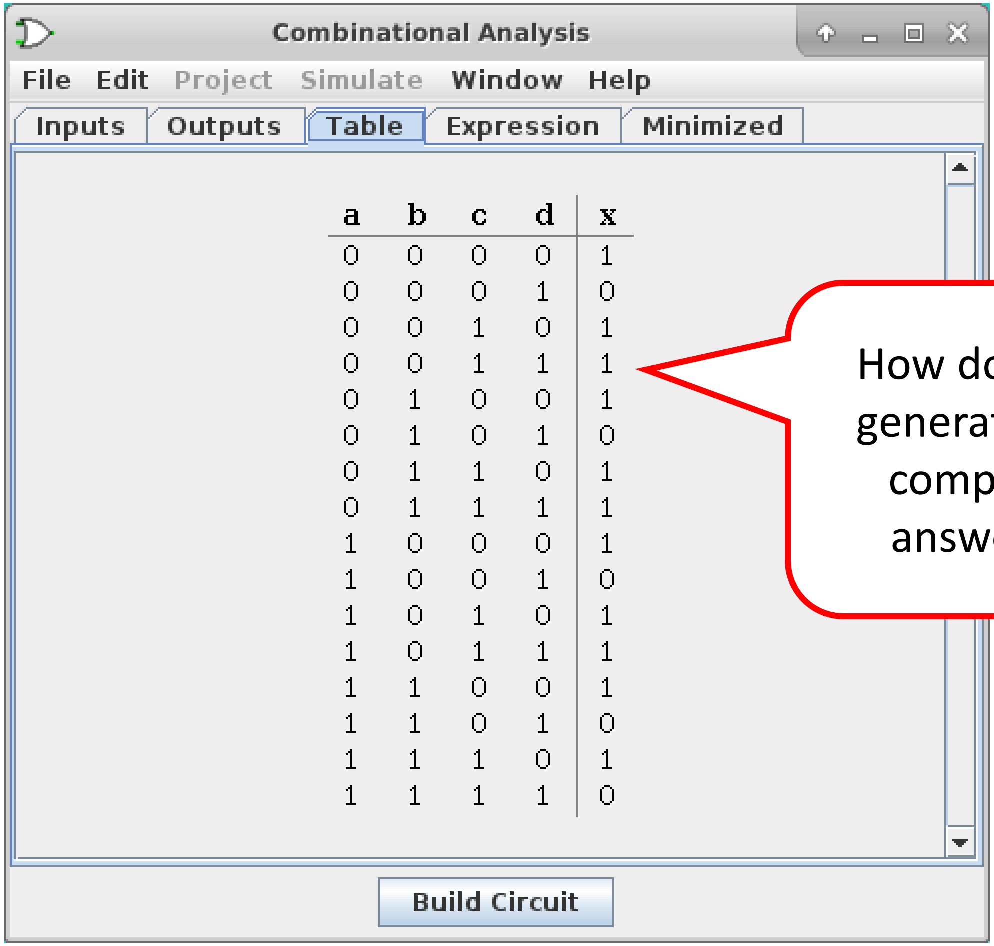
Then select **Analyze Circuit**

Lab 2 – Simple Circuit in Logisim



Select **Table** tab…

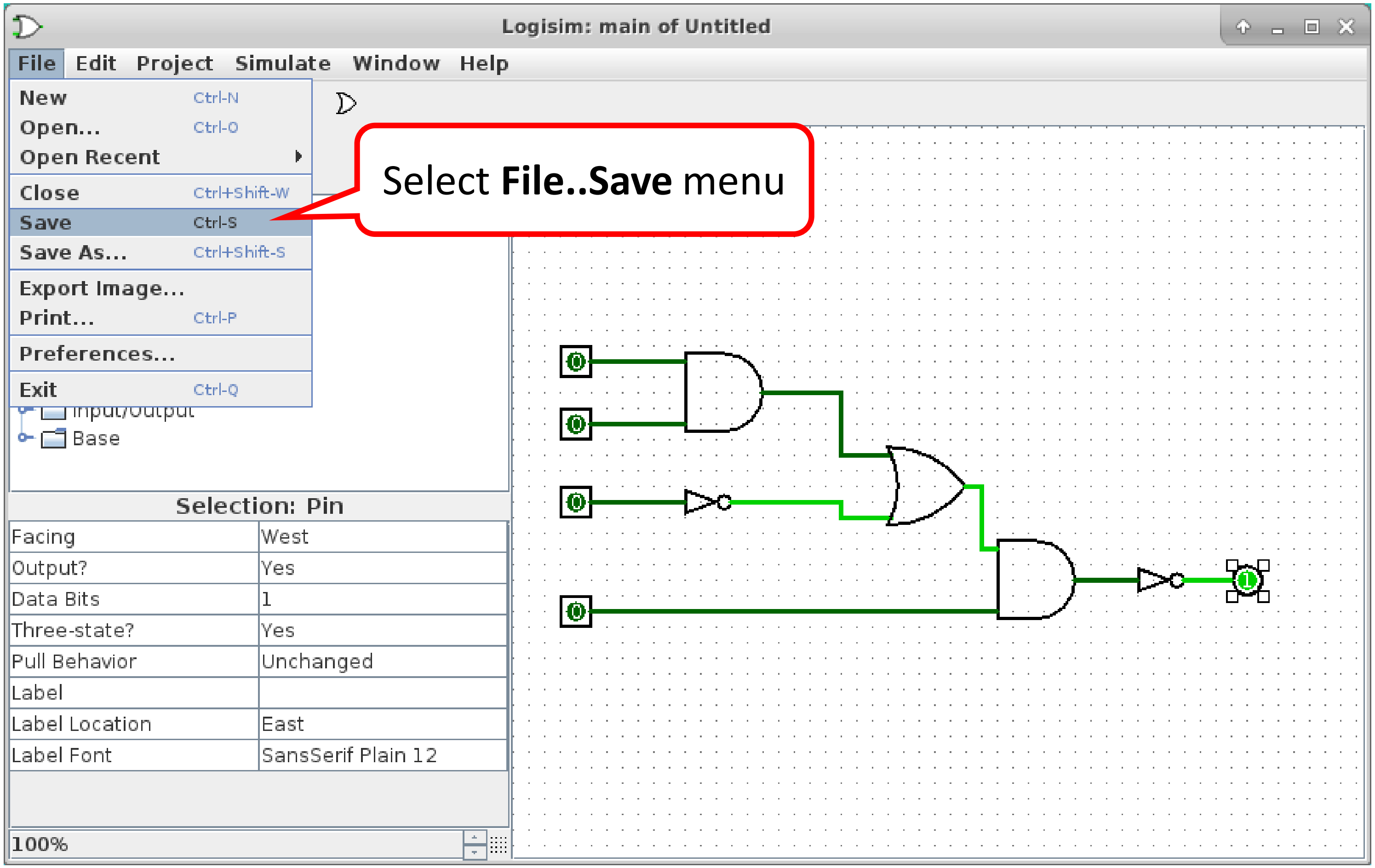
Lab 2 – Simple Circuit in Logisim



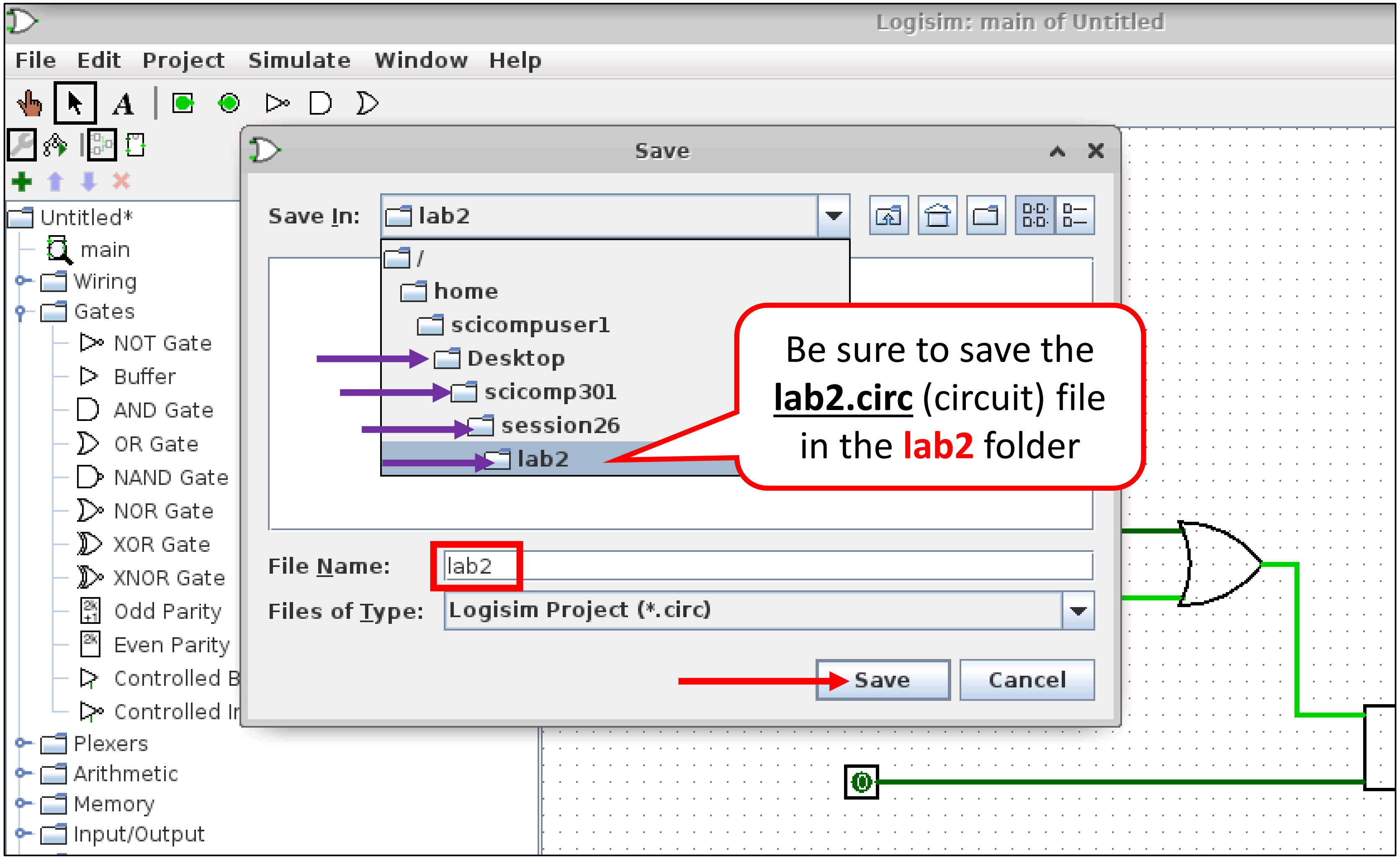
a	b	c	d	x
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

How does the **Logisim** generated **Truth Table** compare with your answers for **Lab 1**?

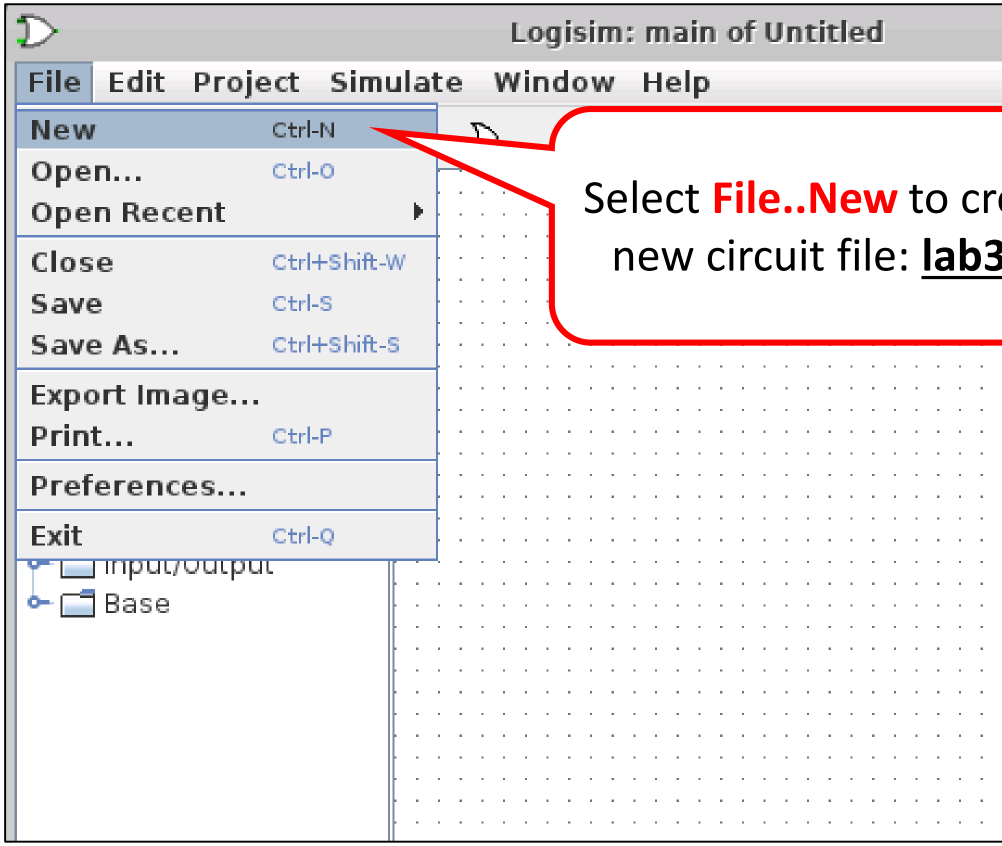
Lab 2 – Simple Circuit in Logisim



Lab 2 – Simple Circuit in Logisim



Lab 3 – Majority Voting (2 of 3)



Select **File..New** to create a new circuit file: **lab3.circ**

Lab 3 – Majority Voting (2 of 3)

- Create a truth table with input variables (A, B, C) that represent the vote (yeah or nay) of **three** people
- Design a circuit that emits **1** as output ***if and only if*** at least 2 out of the 3 input lines are also **1**
  - Start out by making a truth table for all 8 possible input permutations
  - Then use **AND** gates to select only those input permutations that that represent valid (1/high/true) “majority” output
  - Then use **OR** gates to gather the output of those **AND** gates into a single output line

Lab 3 – Majority Voting (2 of 3)

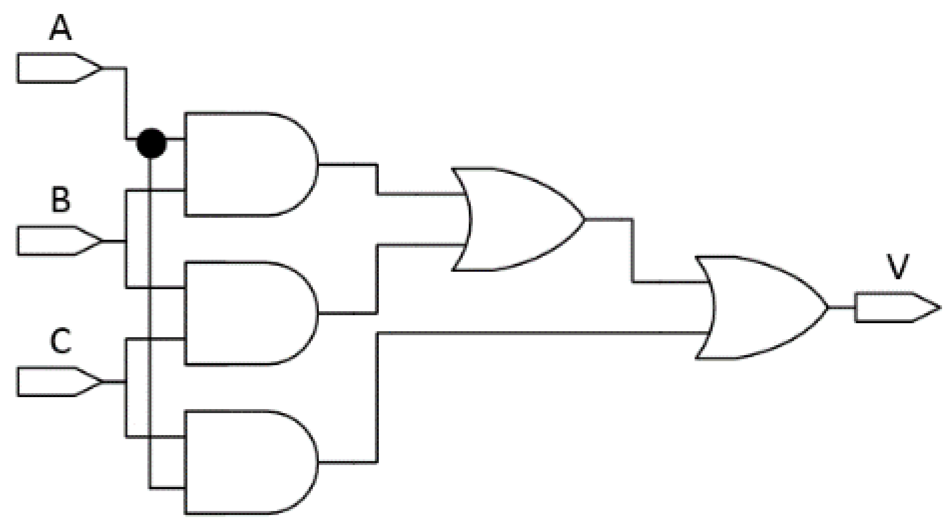
2 of 3 Majority Voting Truth Table

INPUT				OUTPUT
A	B	C		V
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		1
1	1	1		1

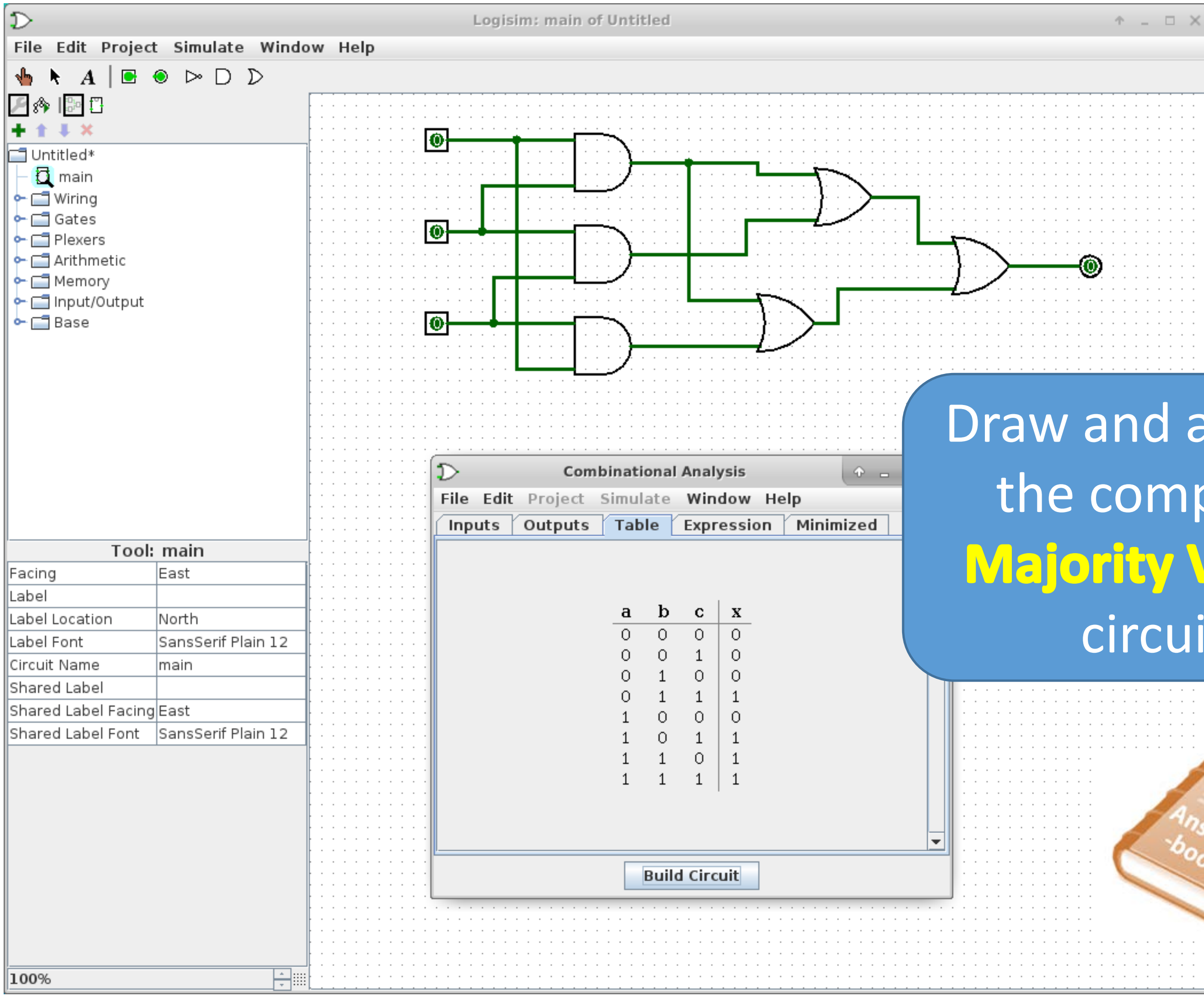
Lab 3 – Majority Voting (2 of 3)

2 of 3 Majority Voting Truth Table

INPUT				**OUTPUT**
A	B	C		V
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		1
1	1	1		1



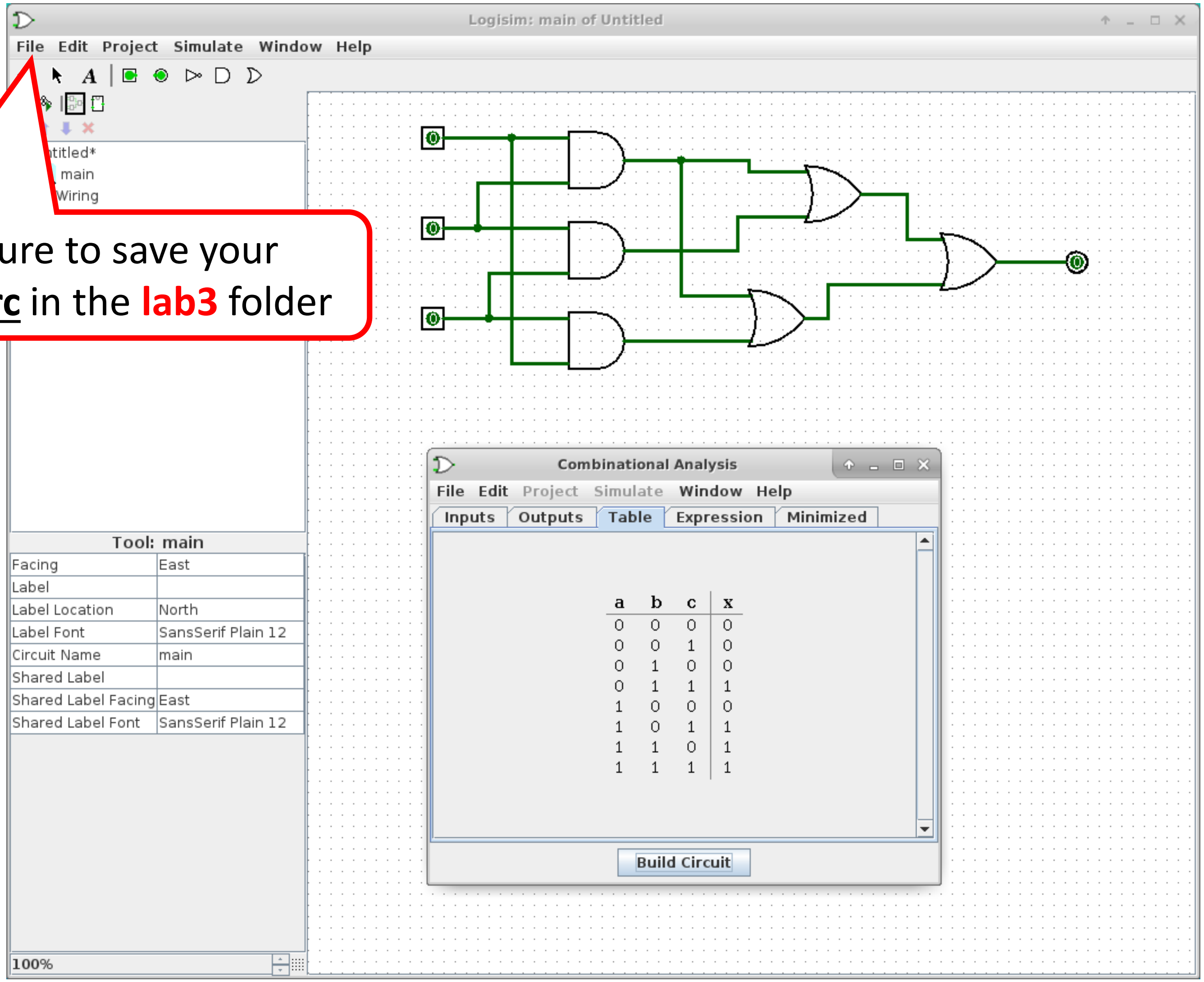
Lab 3 – Majority Voting (2 of 3)



a	b	c	x
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Draw and analyze the complete **Majority Voting** circuit

Lab 3 – Majority Voting (2 of 3)

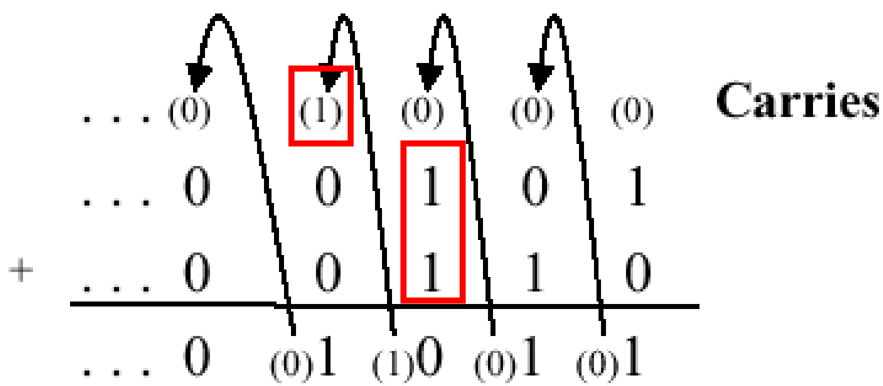


Be sure to save your **lab3.circ** in the **lab3** folder

Binary Addition

- $5_{ten} + 6_{ten}$

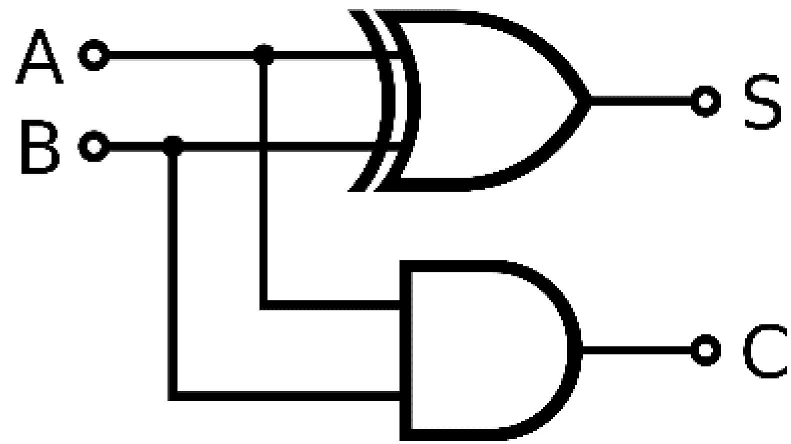
$$
\begin{array}{rl}
 & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ \boxed{0101} \quad (5_{ten}) \\
+ & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ \boxed{0110} \quad (6_{ten}) \\
\hline
= & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ \boxed{1011} \quad (11_{ten})
\end{array}
$$



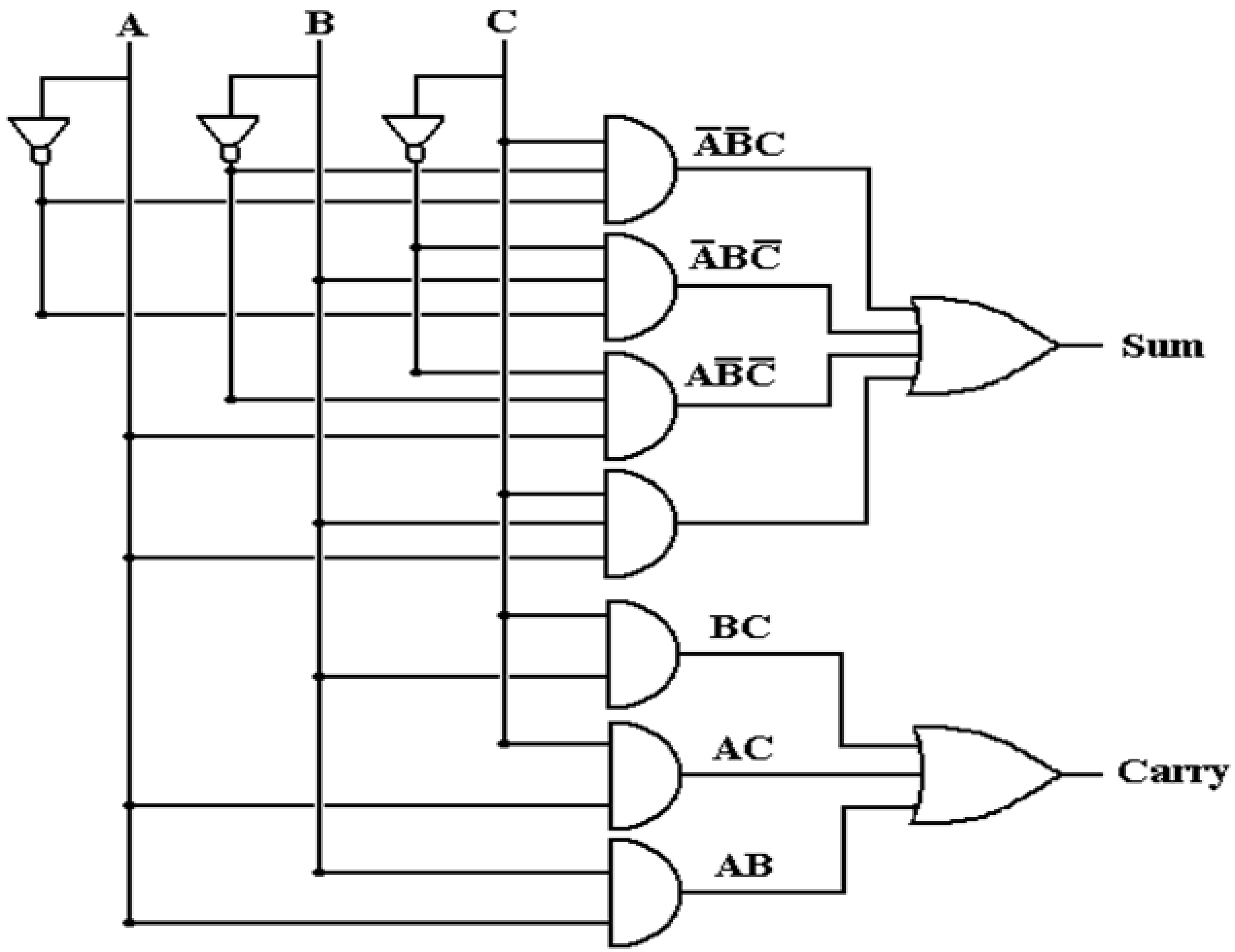
$$
\begin{array}{rr}
{\color{red}0\,1\,1\,1} & \\
00111 & 7 \\
10101 & 21 \\
\hline
11100 & {\color{red}=}\,28
\end{array}
$$

Half-Adder Circuit

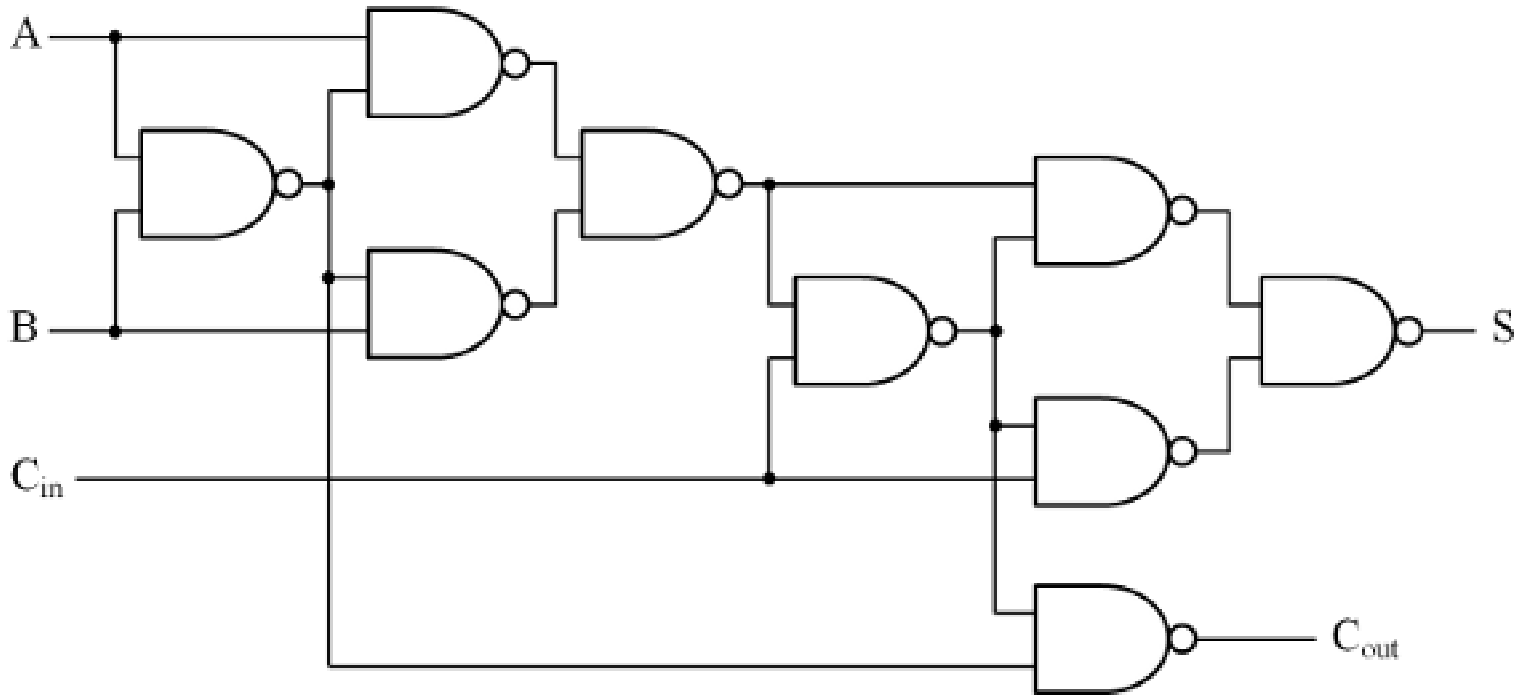
Half ADDER Truth Table				
INPUT			OUTPUT	
A	B		S	C_{out}
0	0		0	0
0	1		1	0
1	0		1	0
1	1		0	1



Full Adder Circuit



Full Adder Circuit using **NAND** gates



Lab 4 - Full Adder using **XOR** gates

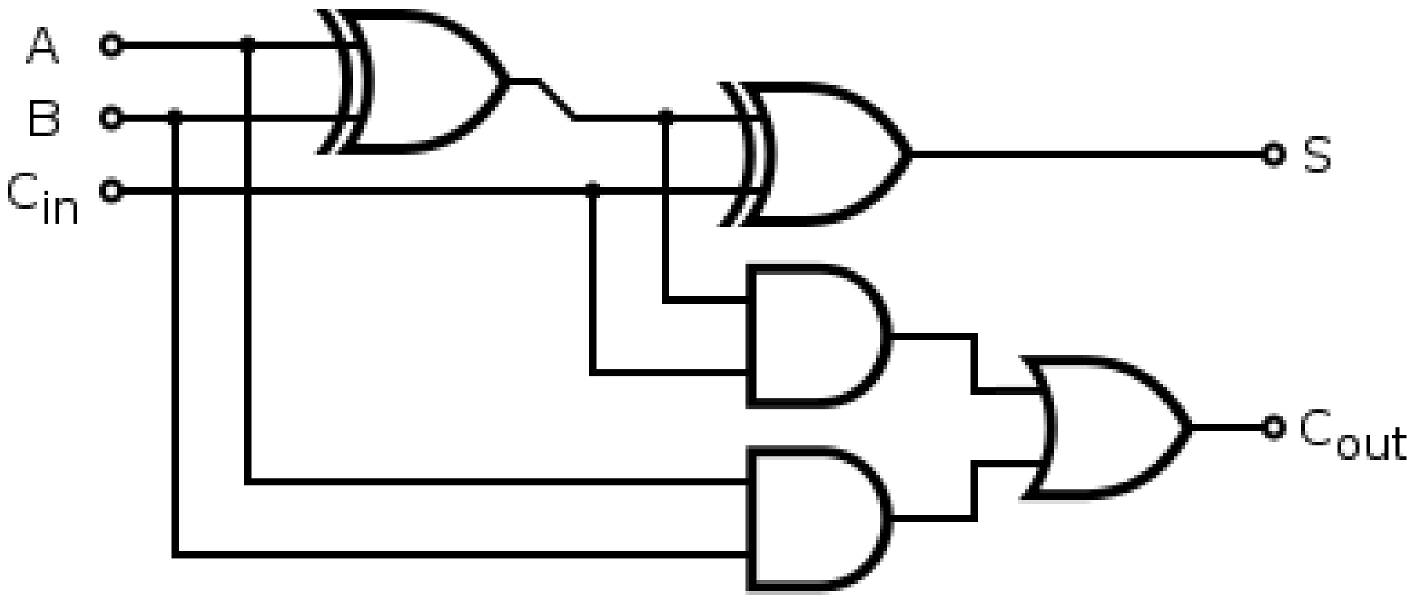
- Using <u>only</u> 1 **OR**, 2 **XOR**, and 2 **AND** gates, design a **FULL ADDER** circuit
- The circuit has **3 input lines**, and **2 output lines** to indicate the sum and if there needs to be a carry to the next column
- Consider how FULL ADDERs can be chained to **sum two 3-bit numbers**

FULL ADDER Truth Table

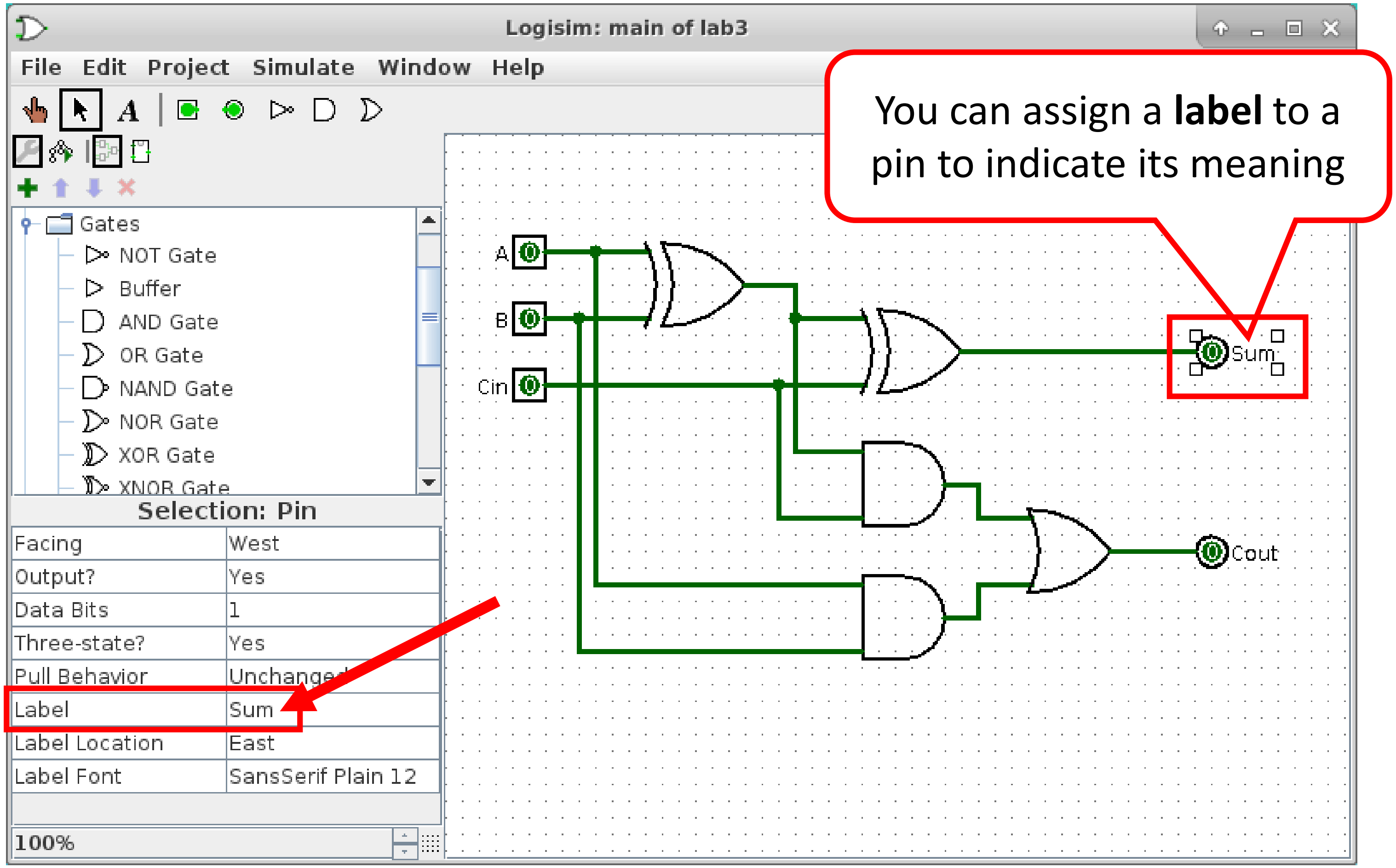
INPUT			OUTPUT	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Lab 4 - Full Adder using **XOR** gates

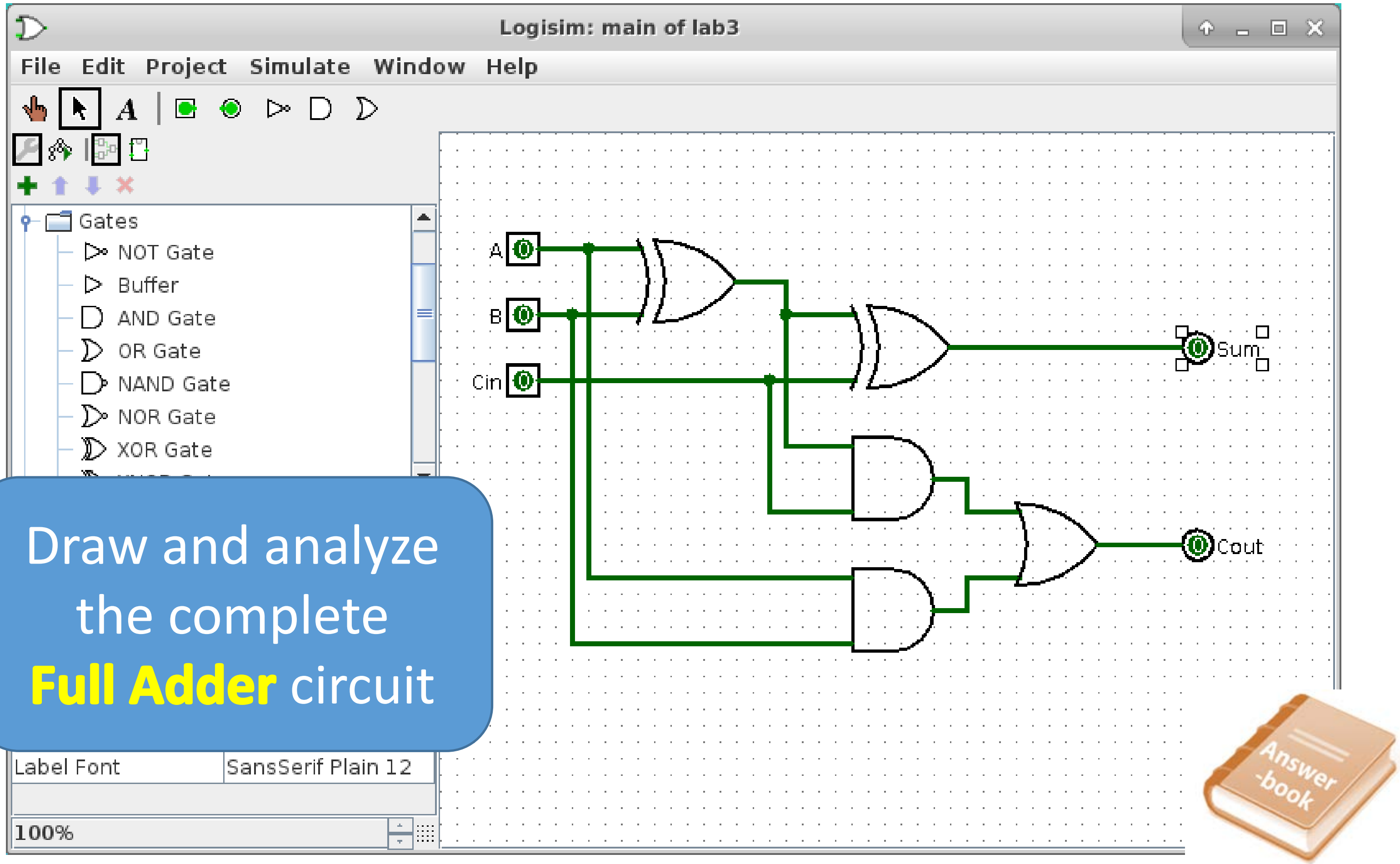
Lab 4 - Full Adder using **XOR** gates



Lab 4 - Full Adder using XOR gates

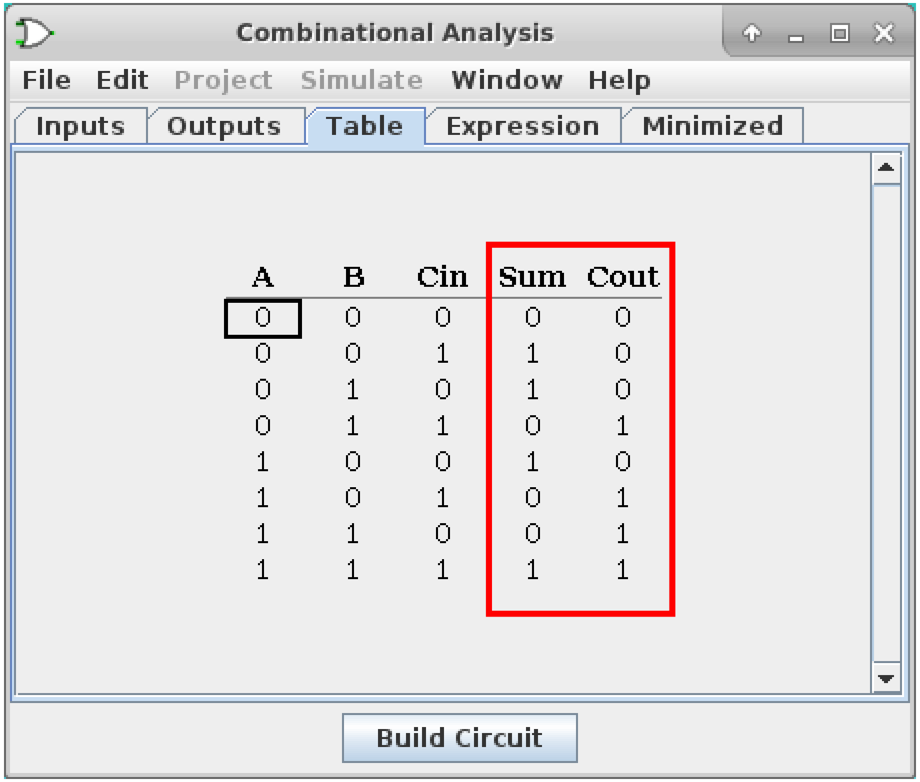


Lab 4 - Full Adder using XOR gates

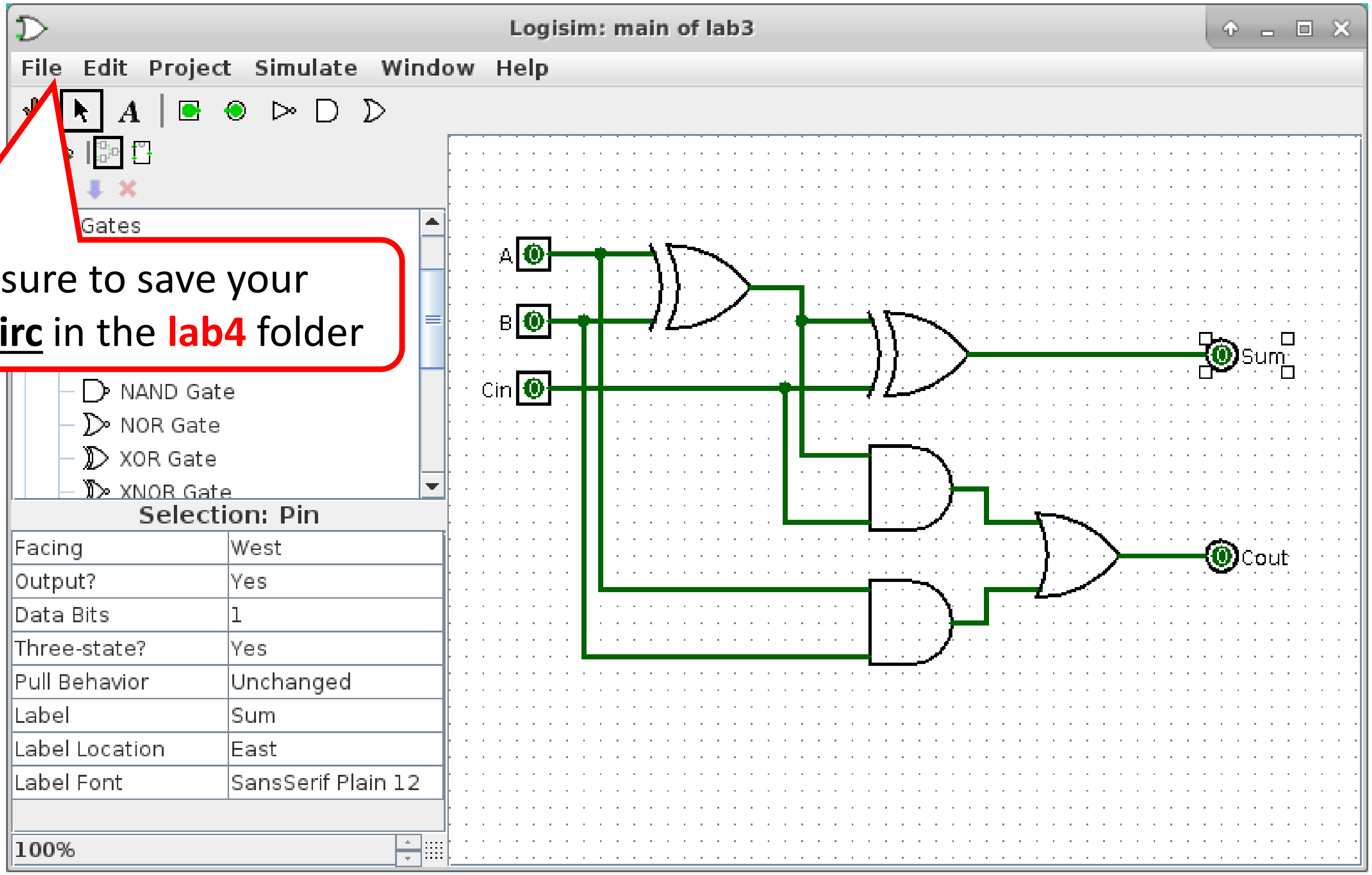


Draw and analyze the complete **Full Adder** circuit

Lab 4 - Full Adder using **XOR** gates

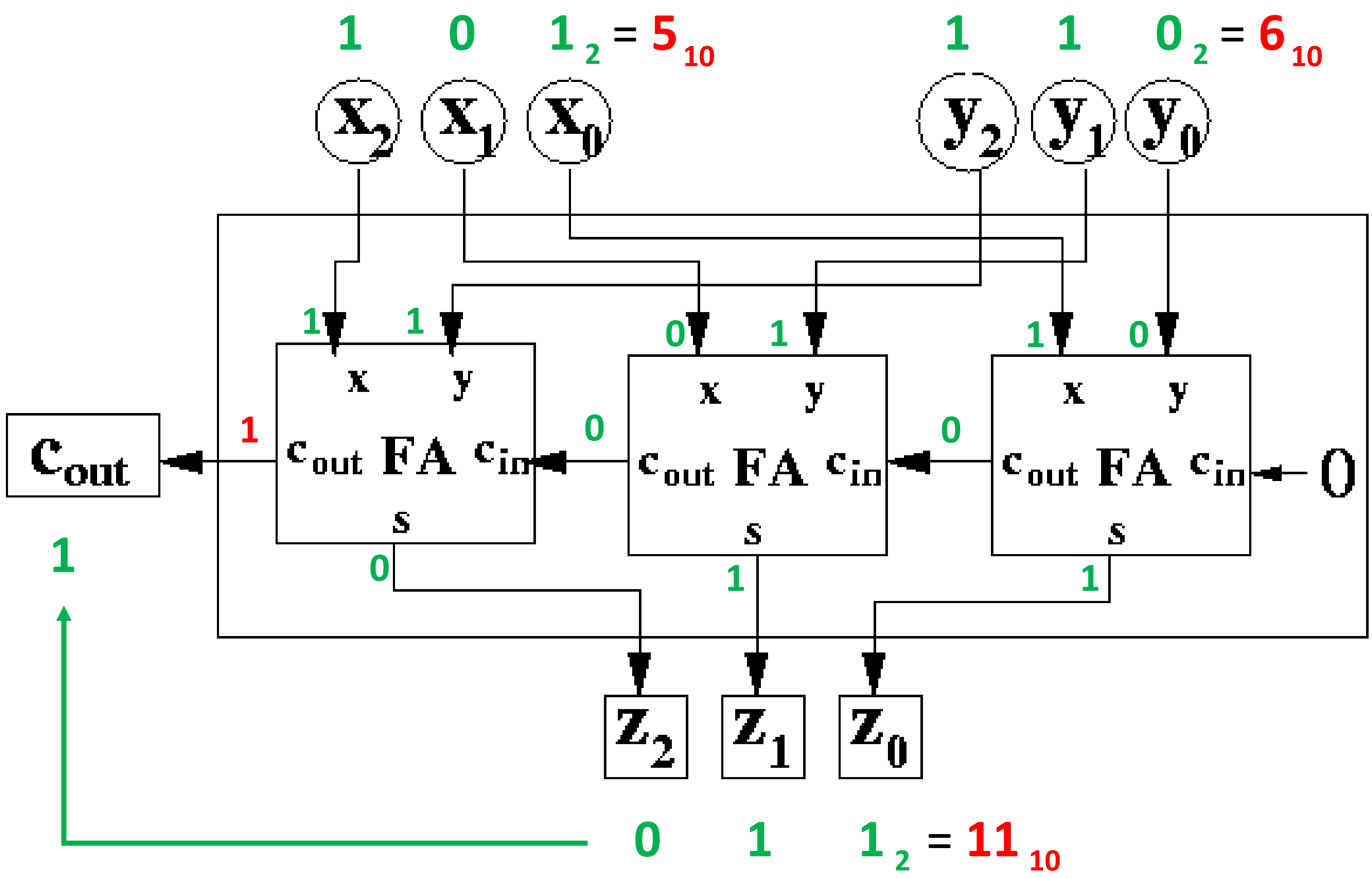


Lab 4 - Full Adder using **XOR** gates

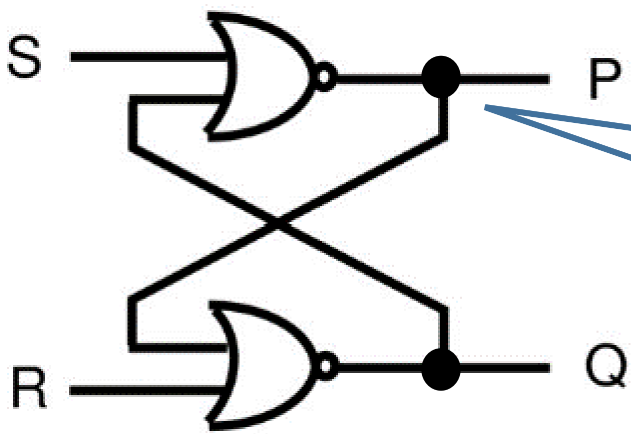


Be sure to save your **lab4.circ** in the **lab4** folder

Chaining Full Adders with Ripple Carry



1 Bit Memory : Set-Reset Latch



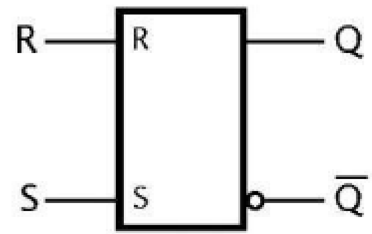
The key was to send the output back into the input!

Input		Output	
S	R	P	Q
0	0	Hold Output	
0	1	1	0
1	0	0	1
1	1	Invalid Input	

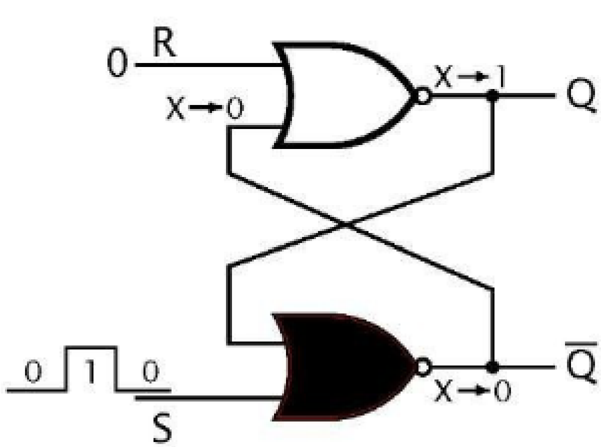
1 Bit Memory : Set-Reset Latch

R S	Q	
0 0	Q	(no change)
0 1	1	(set)
1 0	0	(reset)

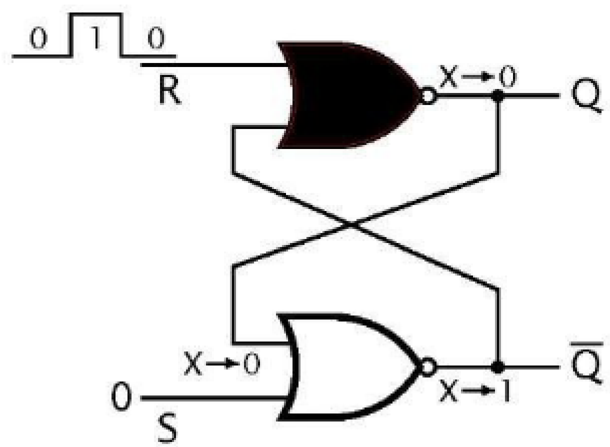
(a) Defining RS latch truth table



(b) Logic symbol with true/complement outputs

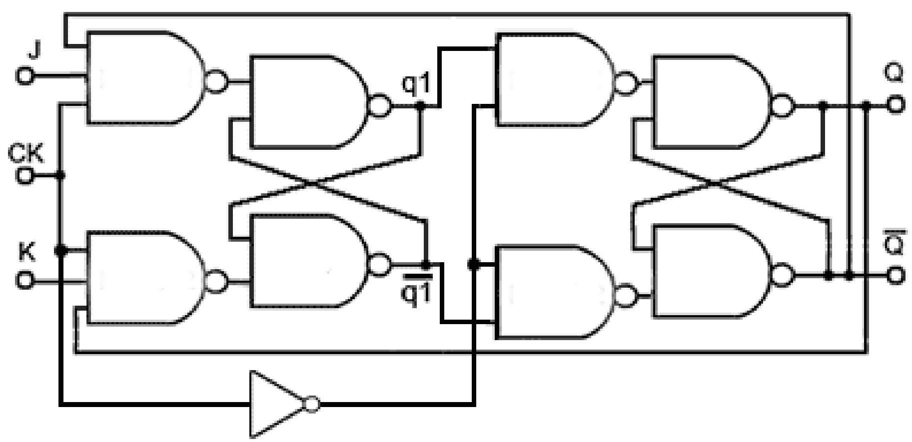


(c) Setting the latch

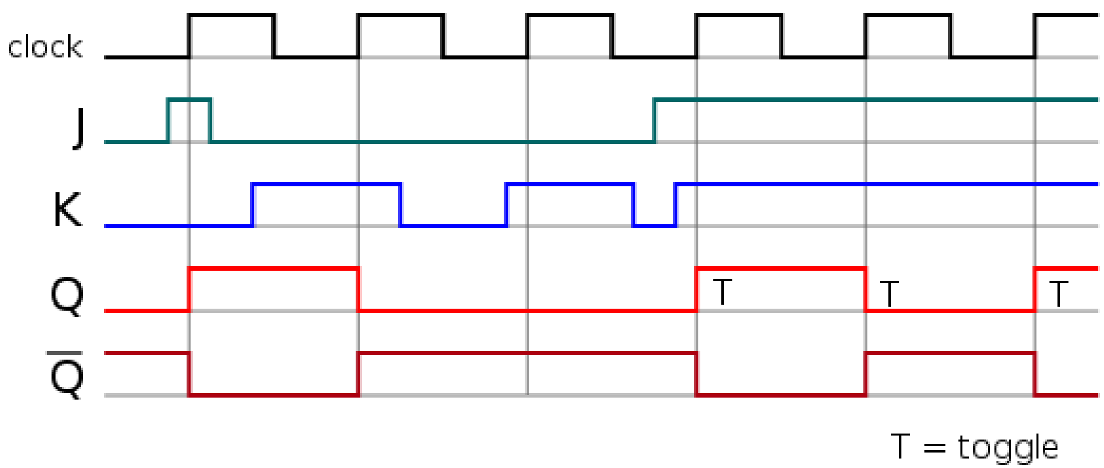


(d) Resetting the latch

1 Bit Memory: A <u>**clocked**</u> J-K Flip-flop



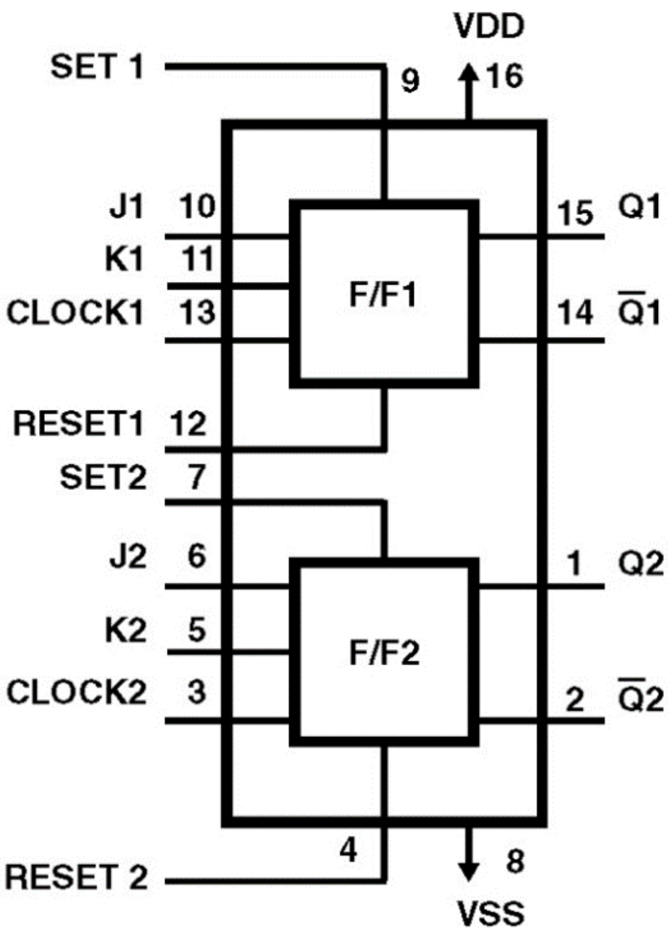
C	J	K	Q	$\overline{Q}$
↑	0	0	latch	latch
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	toggle	toggle
x	0	0	latch	latch
x	0	1	latch	latch
x	1	0	latch	latch
x	1	1	latch	latch

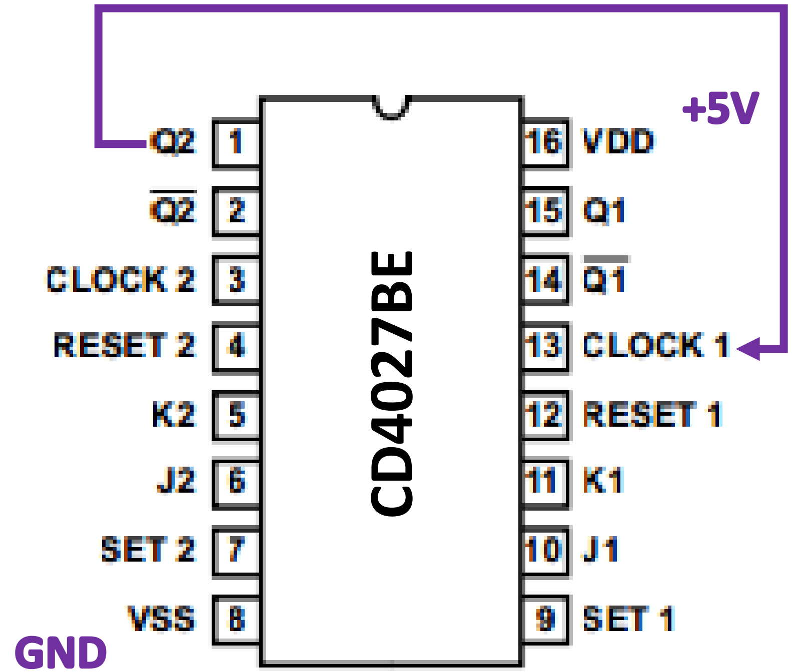


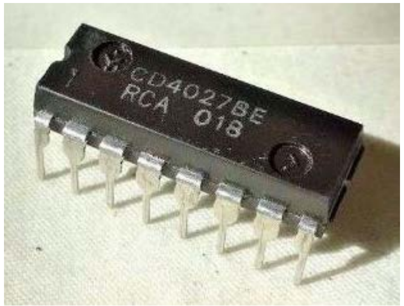
A J-K flip-flop can be set/reset/toggled only during the *rising edge* of the clock signal

When the clock is low a **latch** maintains its prior output value

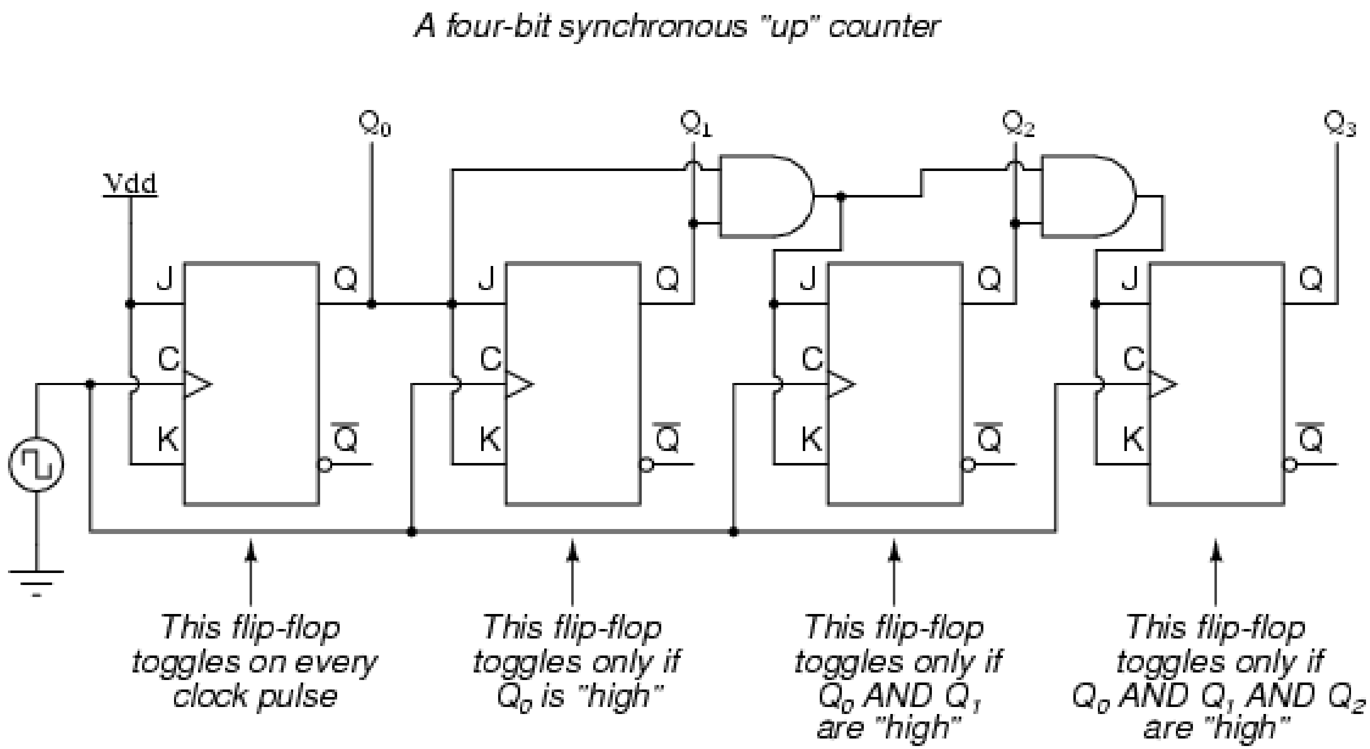
CD4027 Dual J-K Flip-Flop



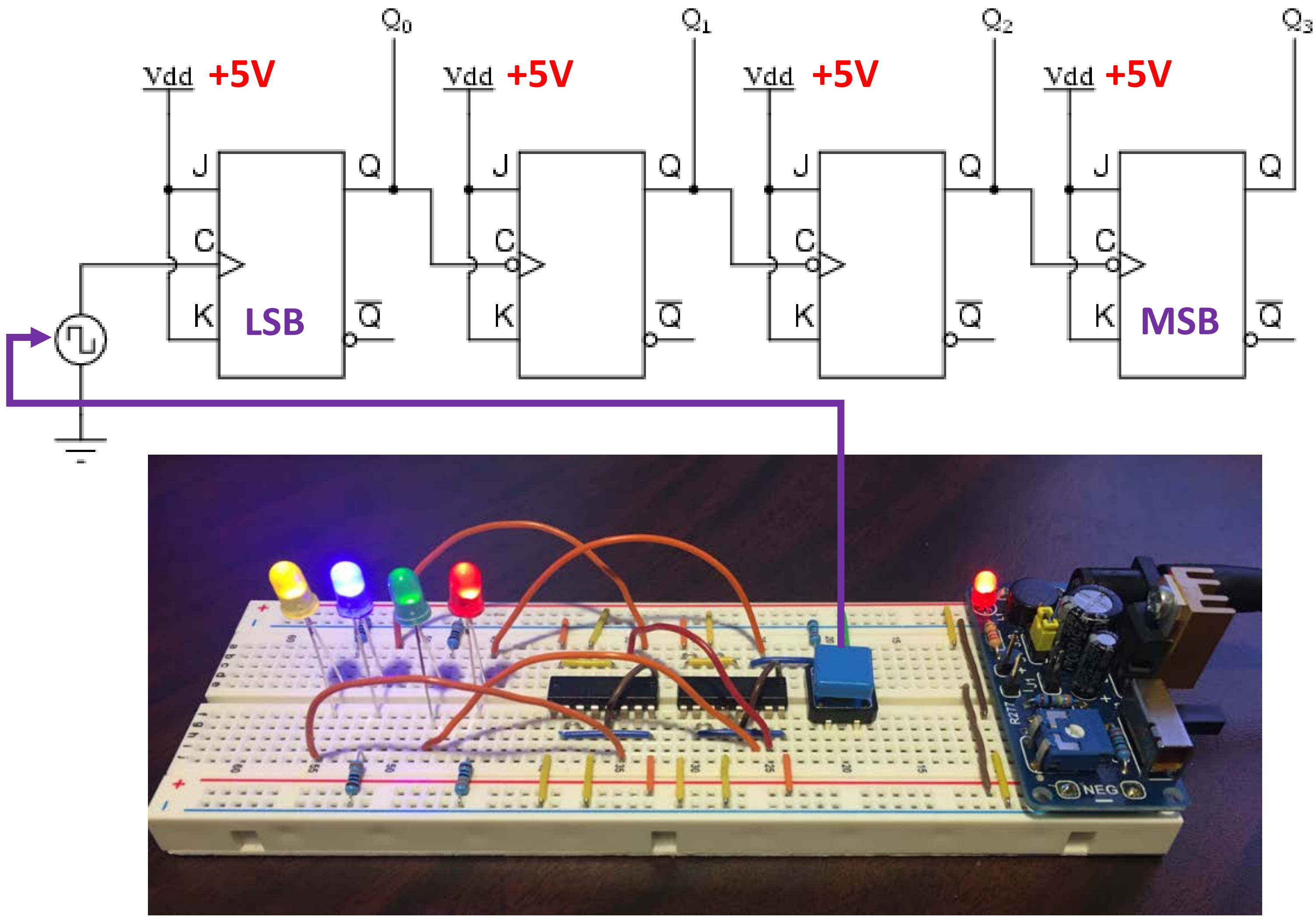




A 4-Bit Counter Using Clocked J-K Flip-Flops

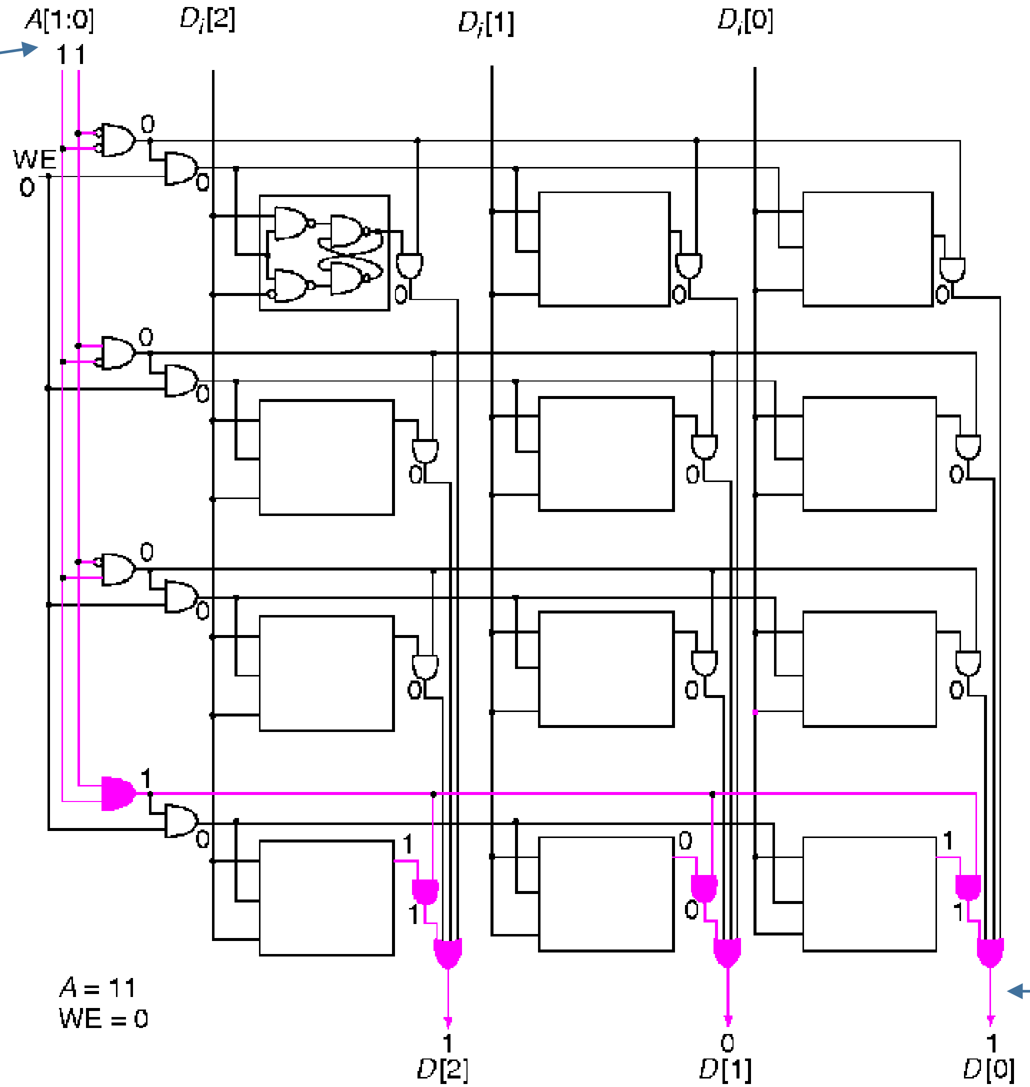


A 4-Bit Counter Using Clocked J-K Flip-Flops



Reading 3 bits from a 4 address memory

Address Lines

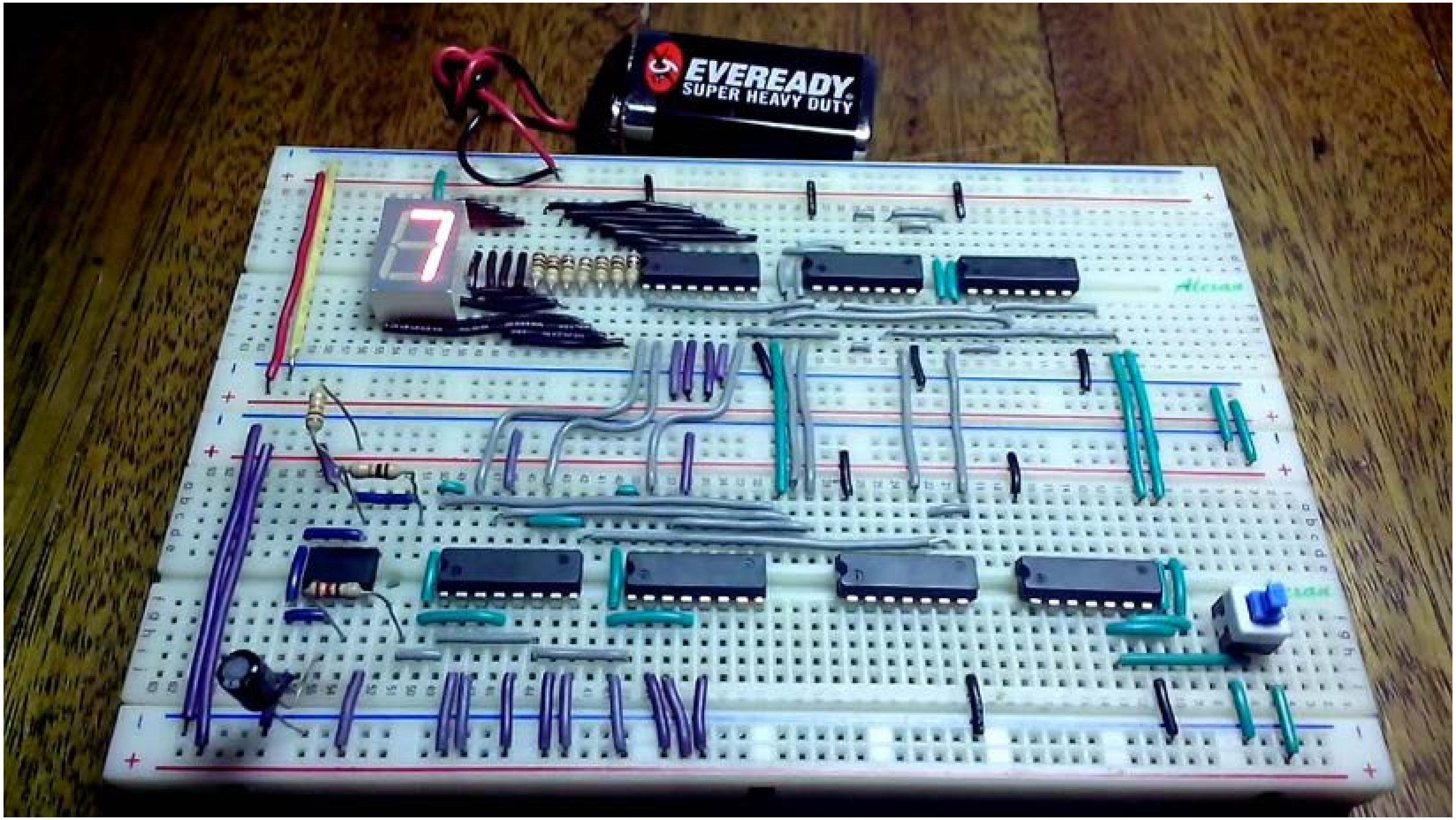


Imagine: a typical 256 GB smart phone has 2×10^{12} bits of memory!

This 12-bit memory is made from a 4 x 3 matrix of flip-flops

Data Lines

A Computer = An **Adder** *with* **Memory**



Invest in Your Own Future

Elegoo EL-KIT-003 UNO Project Super Starter Kit with Tutorial for Arduino

by ELEGOO

865 customer reviews | 151 answered questions

Price: $35.00 prime

- Free PDF tutorial(more than 22 lessons) and clear listing in a nice package
- The most economical way to starting Arduino programming for those beginners who are interested
- Lcd1602 module with pin header (not need to be soldered by yourself)
- This is the upgraded starter kits with power supply module, 9V battery with dc
- High quality kite with uno R3, 100 percent compatible with Arduino uno R3

Invest in Your Own Future

SparkFun Inventor's Kit - v4.0

by SparkFun

6 customer reviews | 5 answered questions

Amazon's Choice for "sparkfun inventor's kit - v4.0"

List Price: $99.95
Price: $85.95 prime
You Save: $14.00 (14%)

- Starter Kit to Learn Arduino
- Includes the SparkFun RedBoard (Arduino compatible)
- ATmega328P
- Complete 16 exciting projects, including Simon Says
- Experiment with Sound, Light, Motion, Display and Robot

Closing thoughts…

- Boundless natural curiosity is the mark of a great scientist
- Be most proud of your greatest questions
- Never be satisfied with the security of mediocrity
- Be a lifelong learner – glide with technology change

Now You Know…

- Digital Logic Circuits
    - All computers are made from **chains** of simple logic gates
    - **NAND** and **NOR** gates are *universal* → they can make all other gates!
- How a computer performs arithmetic = **full adders**
    - Subtraction is just addition with **inverted** logic
    - Multiplication is just repeated addition
    - Division is just repeated subtraction
- How a computer stores numbers in memory = **flip-flops**
    - Four gates make a bit, eight bits make up a byte
    - Imagine how many gates are in your 32GB smartphone!