

Digital Logic

Dr. Charles R. Severance

www.ca4e.com

online.dr-chuck.com

Outline

- What is Computer Architecture
- What is a Central Processing Unit (CPU)
- Start with gates - AND / OR / NOT / XOR ...
- Adding numbers with gates – half and full adders
- Number representation – Base 2 and Base 10 (hexadecimal)
- Storing data with gates – Latches and Flip-Flops

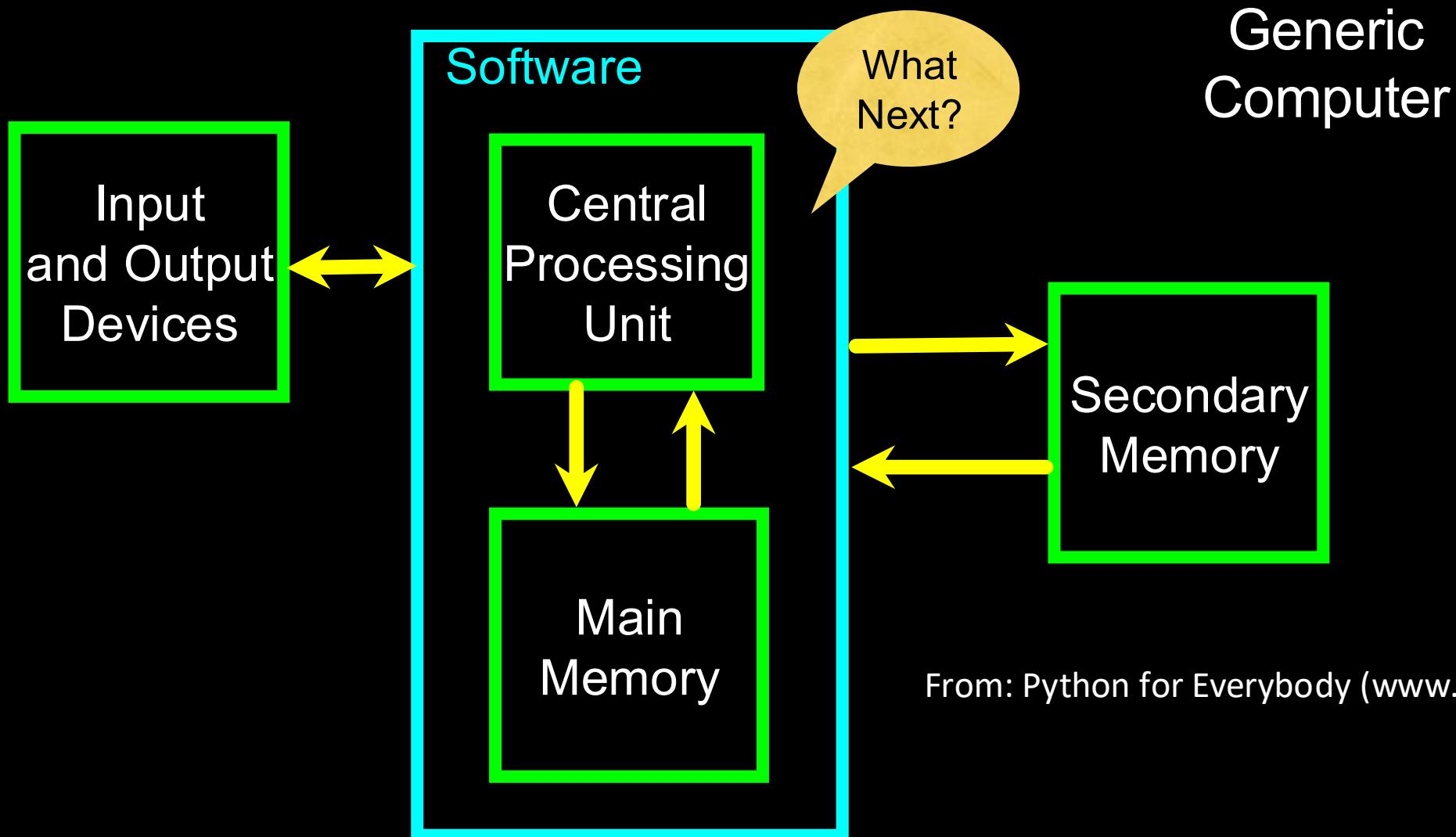
Computer Architecture

- Within the computer data is moved using electric wires
- Each wire generally has a voltage that is either a "0" or "1"
- Wires connect components like the CPU, Memory, or other devices

https://en.wikipedia.org/wiki/Printed_circuit_board



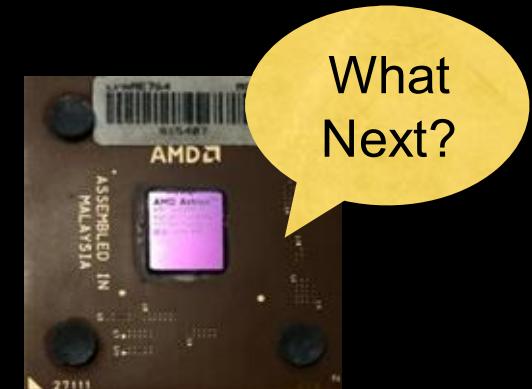
A sound chip from a 1980's Commodore 64 connected to the motherboard

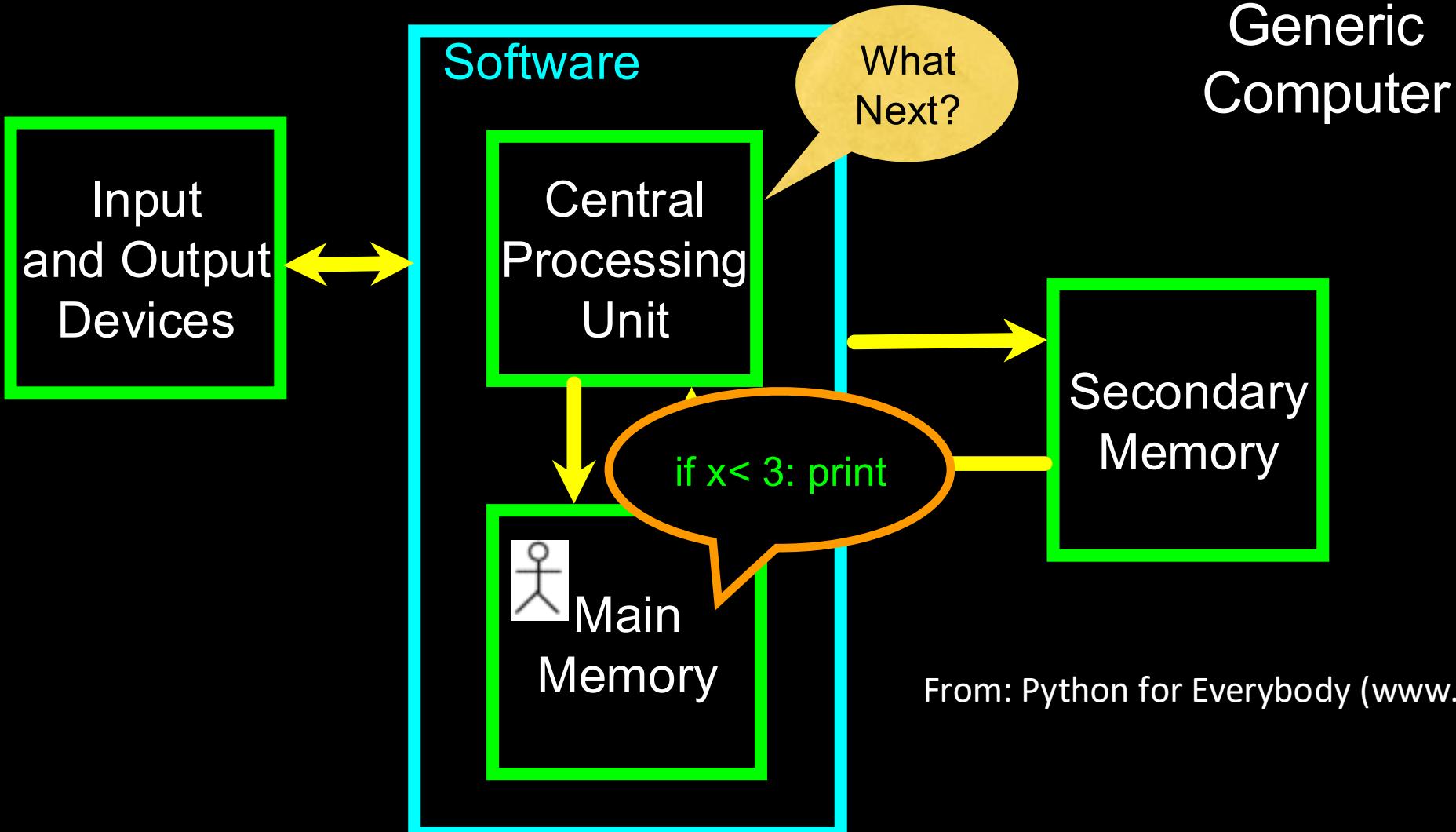


From: Python for Everybody (www.py4e.com)

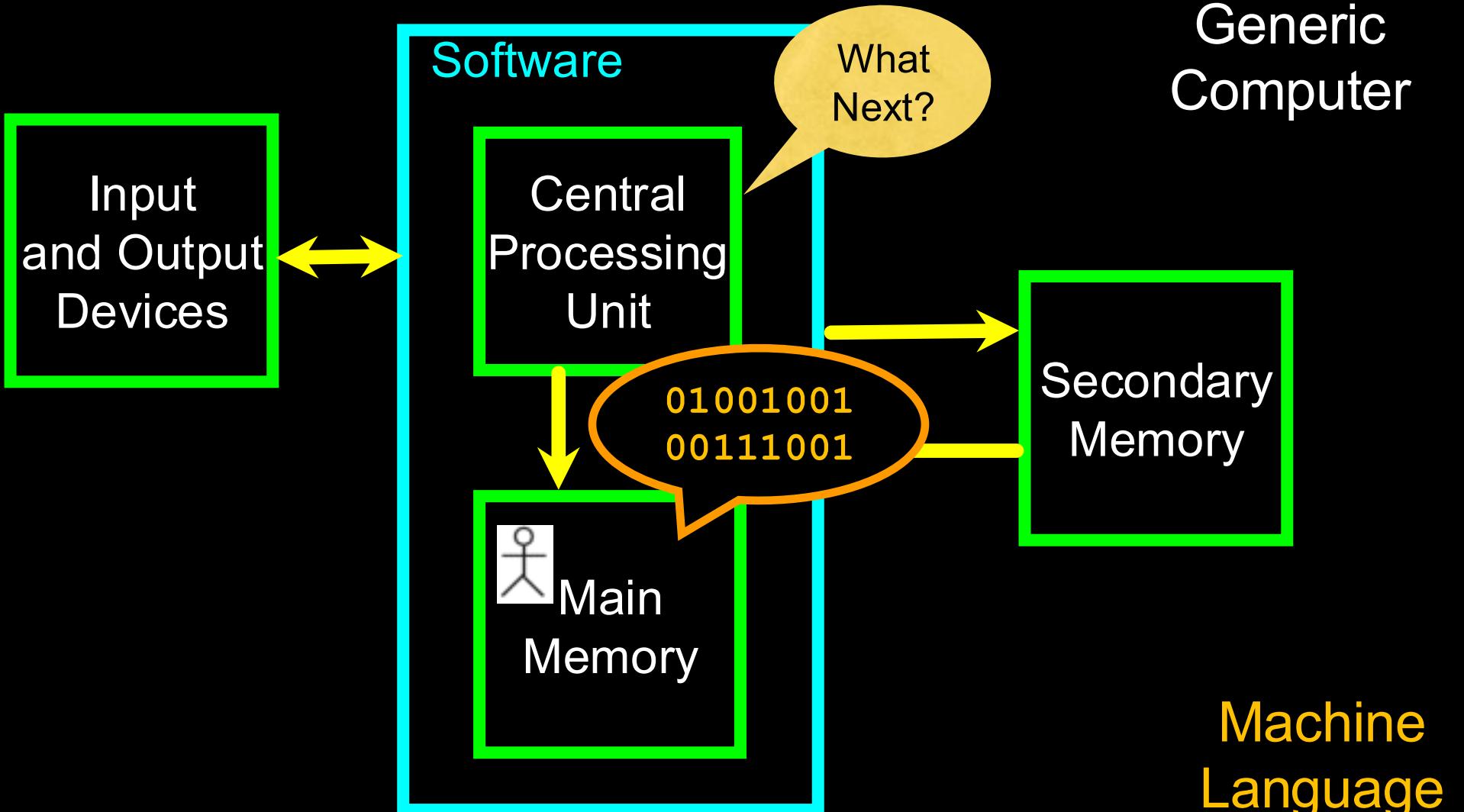
Definitions

- **Central Processing Unit:** Runs the Program - The CPU is always wondering “what to do next”. Not the brains exactly - very dumb but very very fast
- **Input Devices:** Keyboard, Mouse, Touch Screen
- **Output Devices:** Screen, Speakers, Printer, DVD Burner
- **Main Memory:** Fast small temporary storage - lost on reboot - aka RAM
- **Secondary Memory:** Slower large permanent storage - lasts until deleted - disk drive / memory stick





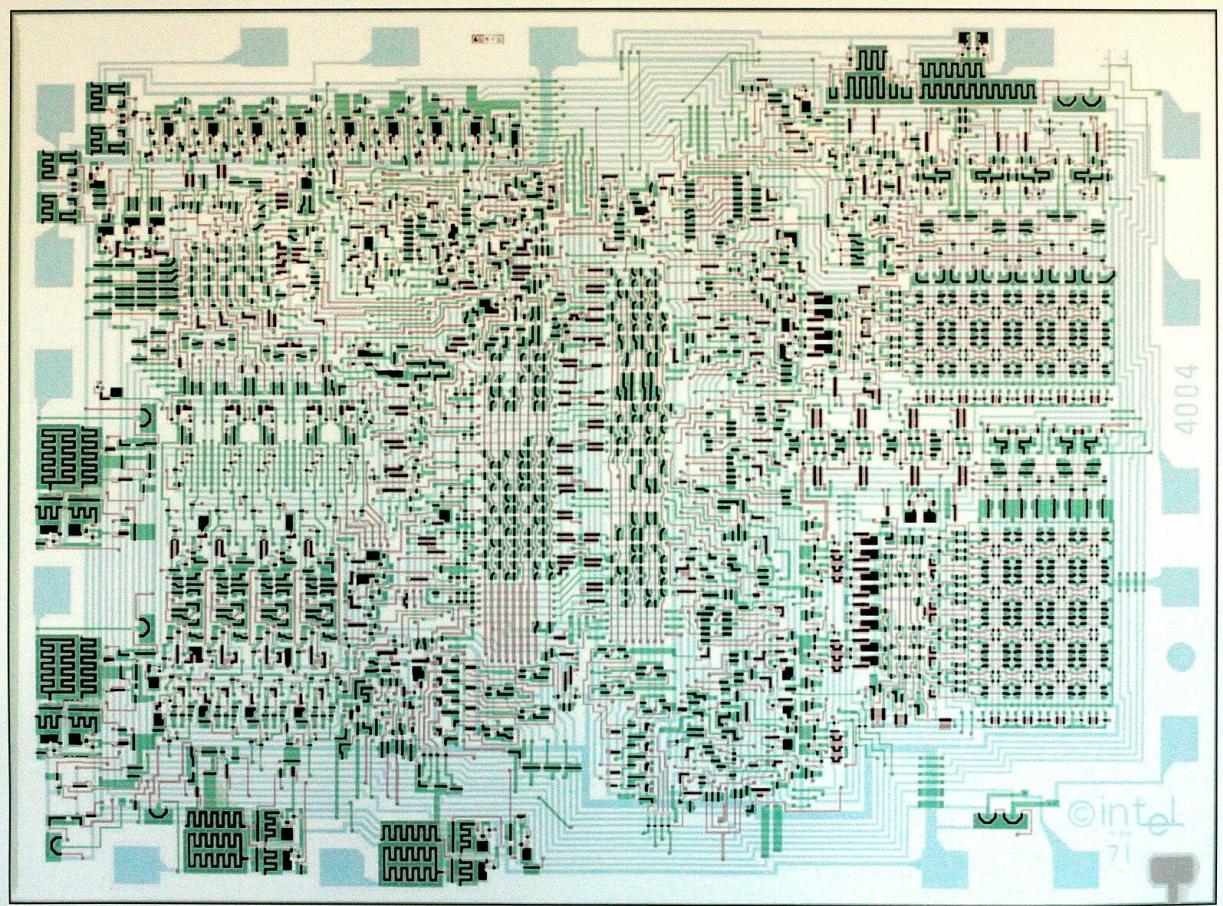
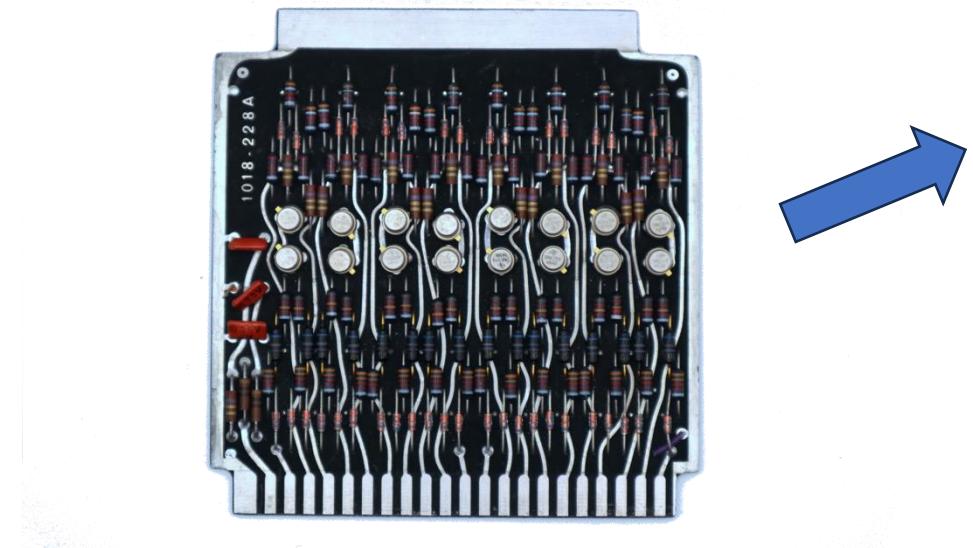
From: Python for Everybody (www.py4e.com)



Building Microprocessors

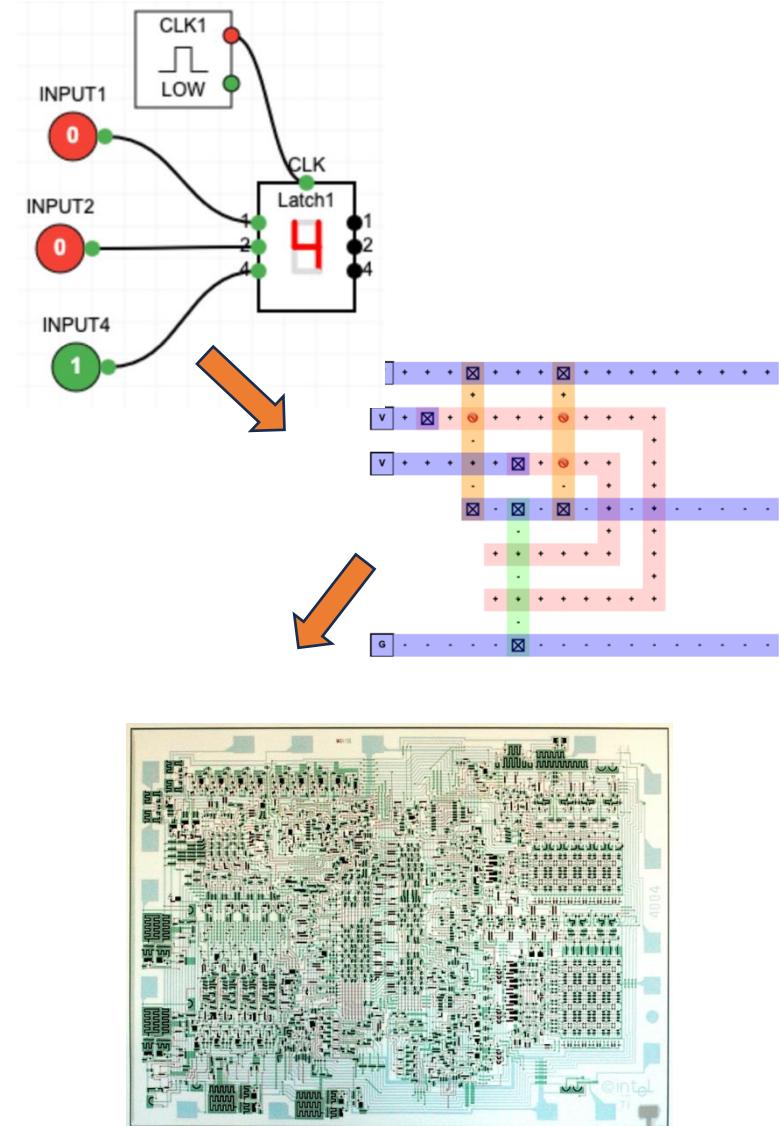
Very Large Scale Integration

- We went from 16 transistors on a board 4 inches on a side to billions of transistors in a chip 0.5 inches on a side



CPU Design Process

- We start with gates and build reusable components and then combine those components to implement programmable digital logic
- Software translates our abstract design into transistors and then creates an optimal layout for our chip
- Photo lithography is then used to build the chip



Adding numbers with gates

Gates are now our low-level building blocks – we can combine them to solve problems

Classic Mathematical Logic Gates

Not



A	Q
0	1
1	0

And



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Or



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Exclusive Or



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Representing Numbers with 0's and 1's

- If we are going to do significant computations, we will need a way to represent numbers as electrical values
- "Normal" numbers are Base-10
- We know the place values of Base-10 numbers
- To manipulate numbers with gates, we use Base-2 numbers

Base-10

$$\begin{aligned}123_{10} &= 1 \times 10^2 + 2 \times 10^1 + 3 * 10^0 \\&= 1 \times 100 + 2 * 10 + 3 * 1 = 123_{10}\end{aligned}$$

Base-2

$$\begin{aligned}110_2 &= 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = \\&= 4 + 2 + 0 = 6_{10}\end{aligned}$$

Keep it simple for now

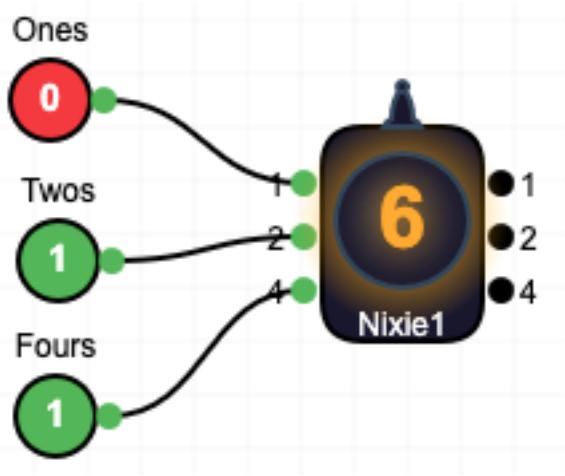
- Base-2 to Base-10 for numbers less than 8
- A.k.a. 3-bit numbers

4s 2s 1s

1 1 0 = 6

Base-10	Base-2
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111

Nixie Tubes – Binary to Decimal Display

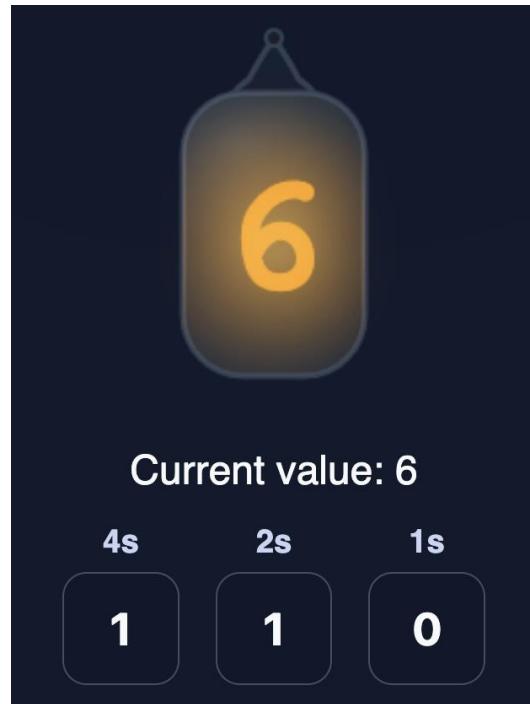


4 2 1

1 1 0 = 6



https://en.wikipedia.org/wiki/Nixie_tube



NIXIE CHALLENGE

Add binary numbers one bit at a time and track the carry.

Binary Addition

Score: 1/10

Add the two binary numbers. Enter each sum bit (include the carry on the left).

	8s	4s	2s	1s	
A		0	1	1	(3)
B		1	1	0	(6)
Sum	1	0	0	1	(9)

<https://www.ca4e.com/tools/nixie/>

Base-2 Math

$$\begin{array}{r} 1 \quad (1) \\ + \quad 1 \quad (1) \\ \hline 1 \quad 0 \quad (2) \end{array}$$

$$\begin{array}{r} 1 \quad 0 \quad (2) \\ + \quad 1 \quad 0 \quad (2) \\ \hline 1 \quad 0 \quad 0 \quad (4) \end{array}$$

$$\begin{array}{r} 1 \quad 0 \quad 0 \quad (4) \\ + \quad 1 \quad 0 \quad 0 \quad (4) \\ \hline 1 \quad 0 \quad 0 \quad 0 \quad (8) \end{array}$$

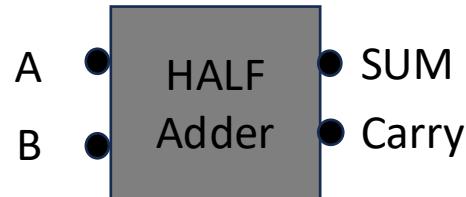
$$\begin{array}{r} 1 \quad 1 \quad (3) \\ + \quad 1 \quad (1) \\ \hline 1 \quad 0 \quad 0 \quad (4) \end{array}$$

$$\begin{array}{r} 1 \quad 1 \quad (3) \\ + \quad 1 \quad 0 \quad (2) \\ \hline 1 \quad 0 \quad 1 \quad (5) \end{array}$$

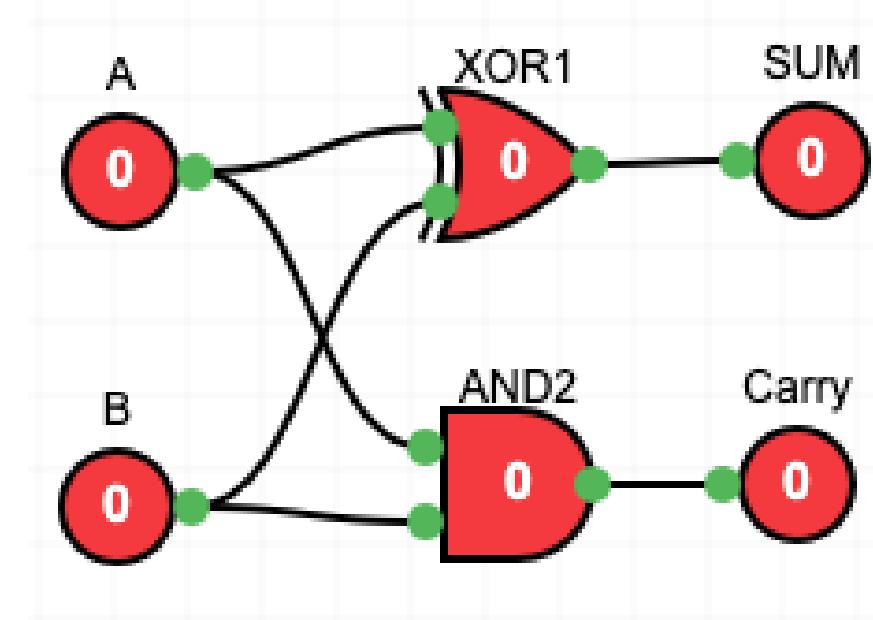
$$\begin{array}{r} 1 \quad 1 \quad 1 \quad (7) \\ + \quad 1 \quad (1) \\ \hline 1 \quad 0 \quad 0 \quad 0 \quad (8) \end{array}$$

Half Adder

- Compute the sum of two base-2 digits and produce a sum and carry.

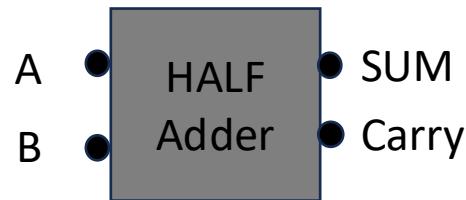


A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

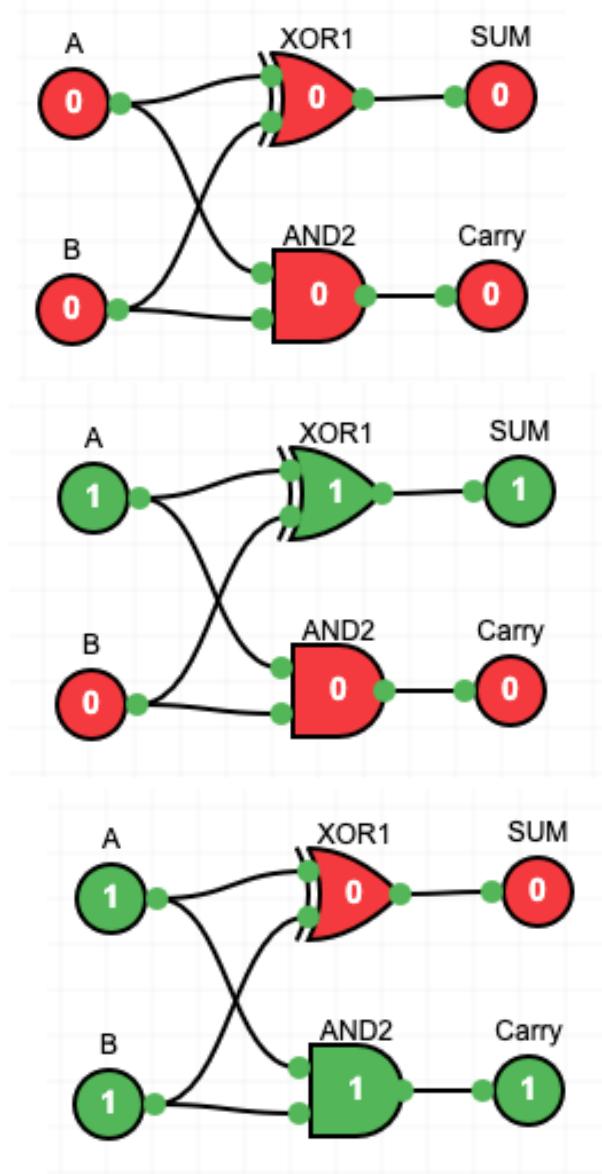


Half Adder

- Compute the sum of two base-2 digits and produce a sum and carry.

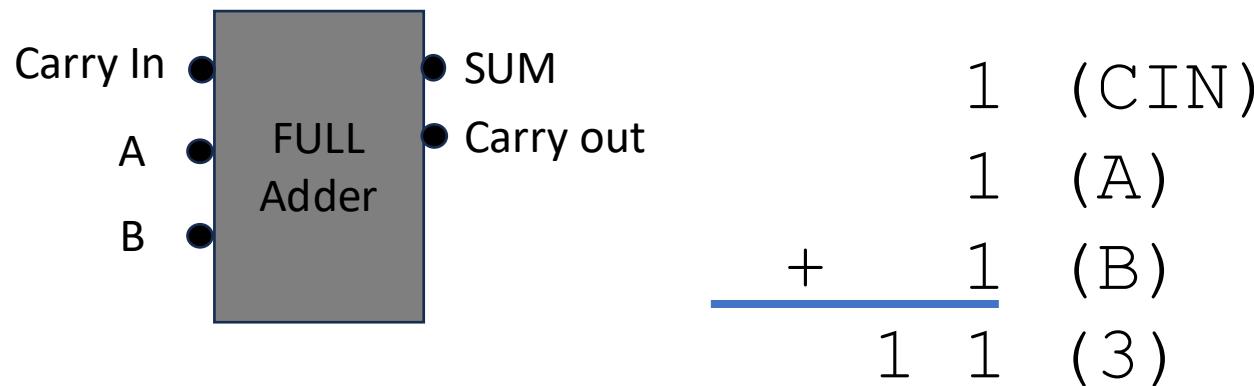


A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



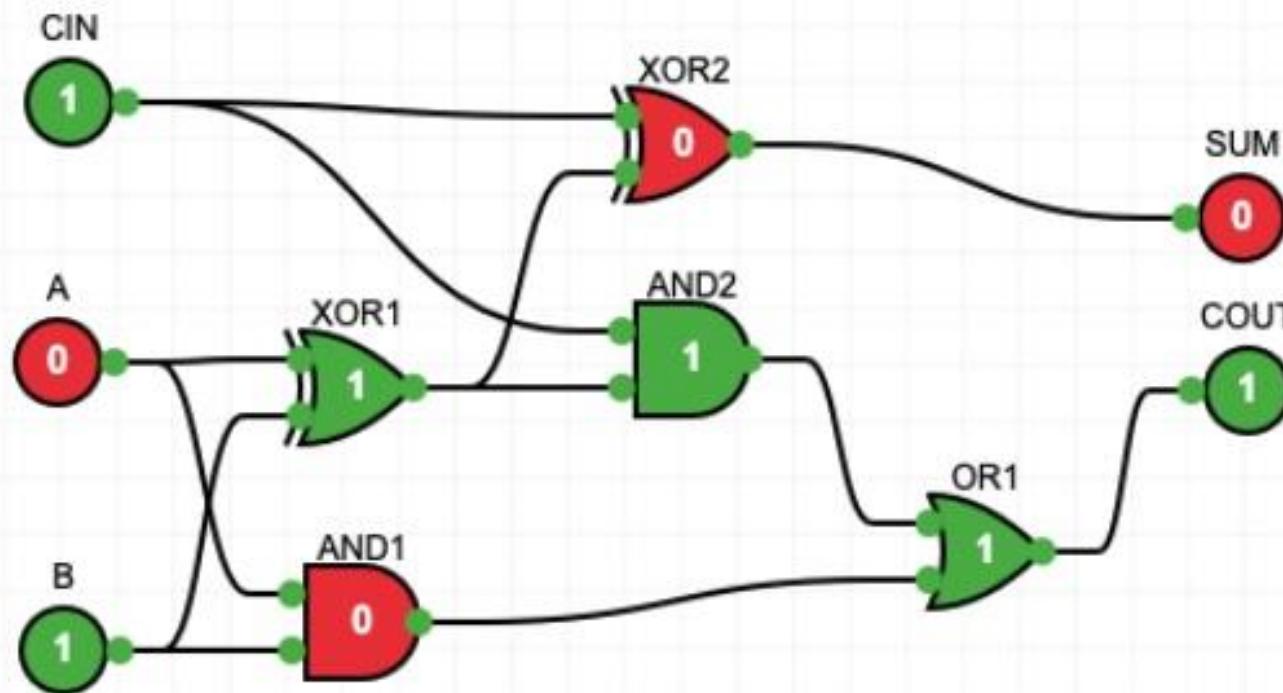
Full Adder

- Adds three 0/1 numbers for a sum/carry between 0 and 3



Full Adder

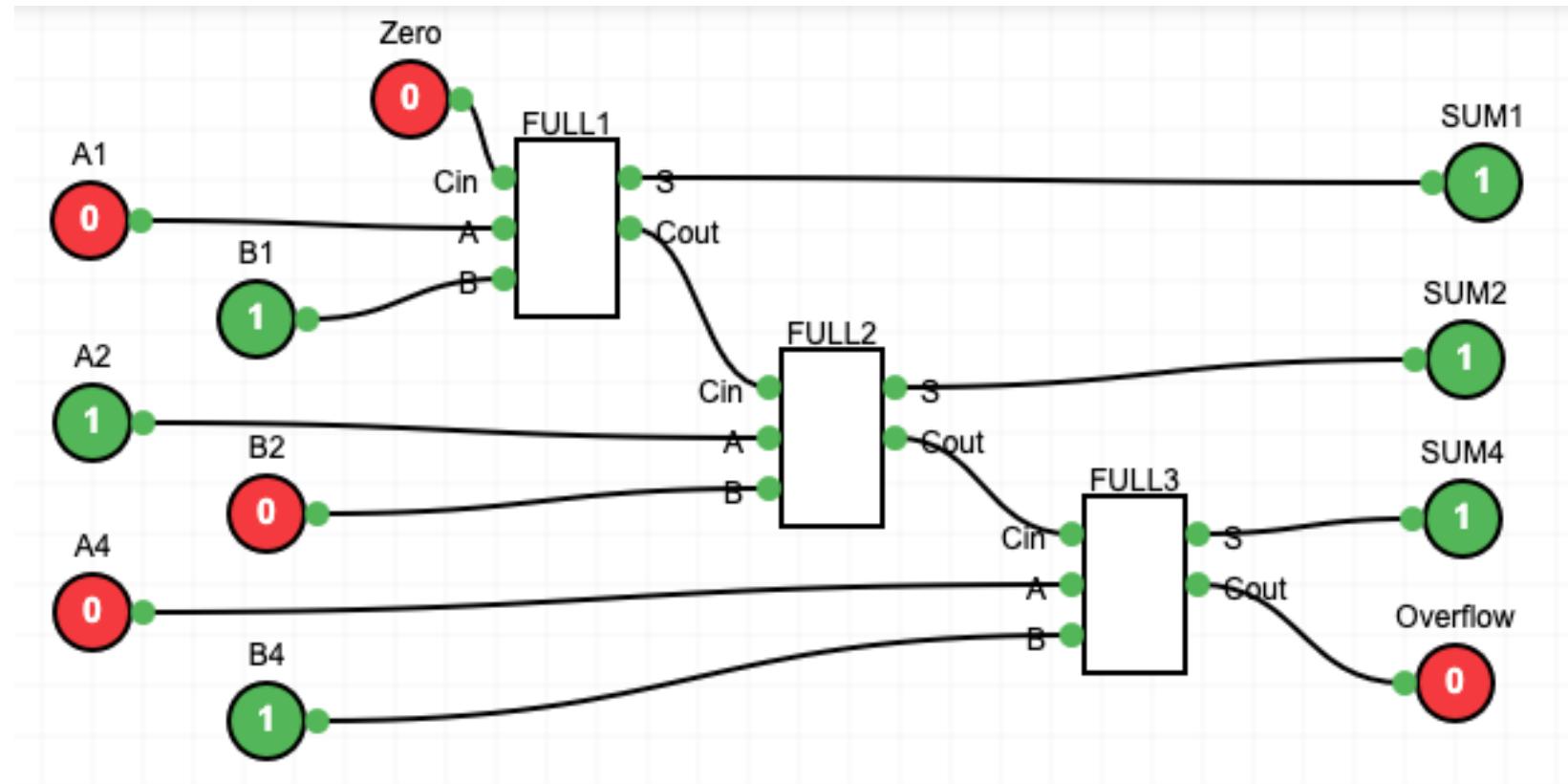
- Adds three 0/1 numbers for a sum/carry between 0 and 3



$$\begin{array}{r} & 1 \text{ (CIN)} \\ & 0 \text{ (A)} \\ + & 1 \text{ (B)} \\ \hline & 1 \text{ 0 (2)} \end{array}$$

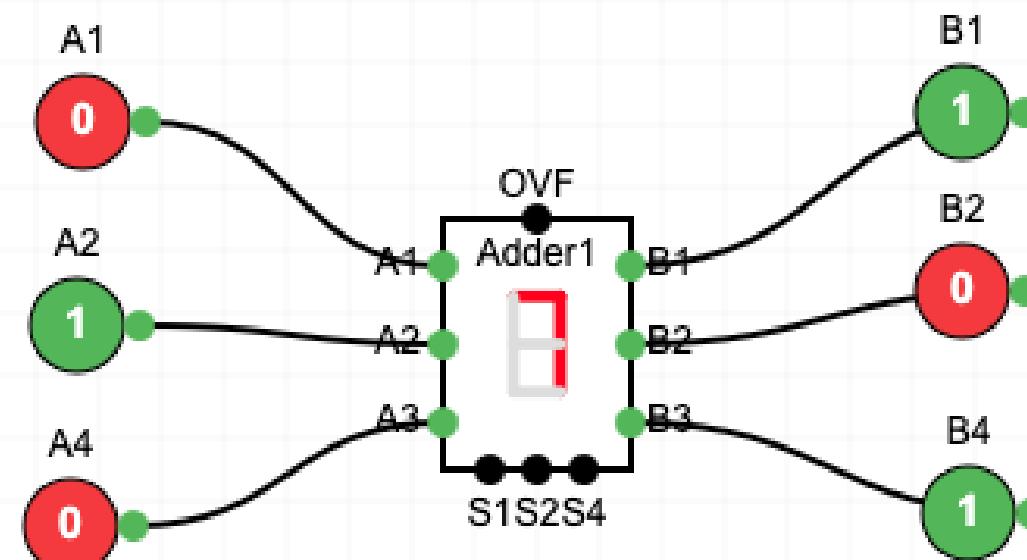
Chaining full adders

$$\begin{array}{r} 0 \ 1 \ 0 \ (2) \\ + 1 \ 0 \ 1 \ (5) \\ \hline 1 \ 1 \ 1 \ (7) \end{array}$$



Three Bit Adder Component

$$\begin{array}{r} 0 \ 1 \ 0 \ (2) \\ + \ 1 \ 0 \ 1 \ (5) \\ \hline 1 \ 1 \ 1 \ (7) \end{array}$$



- Once we know how to make something, add a component to our design system

Storing data with gates

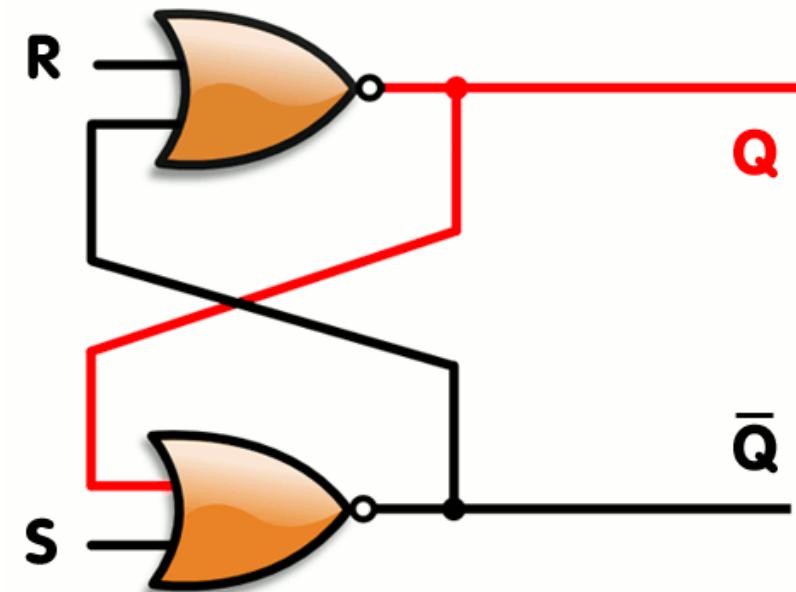
Latches and Flip-Flops

Storing Data Using Feedback Loops

- The simplest latch is a "Set-Reset" latch using two NOR gates
- The output of each NOR gate is connected to the one of the inputs of the other NOR gate

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

S	R	Q
0	0	Maintain
0	1	0
1	0	1
1	1	???

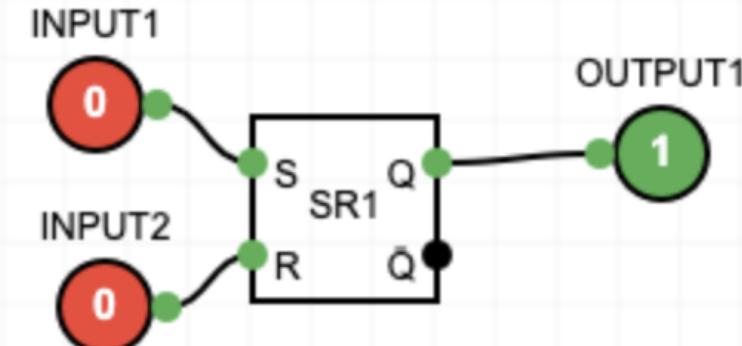


https://en.wikipedia.org/wiki/Flip-flop_%28electronics%29

SR Flip-Flop / Latch

- When S and R are zero, the latch maintains its value
- When S=0 and R=1 the internal value becomes 0
- When S=1 and R=0 the internal value becomes 0
- Setting both to 1 means output is undefined

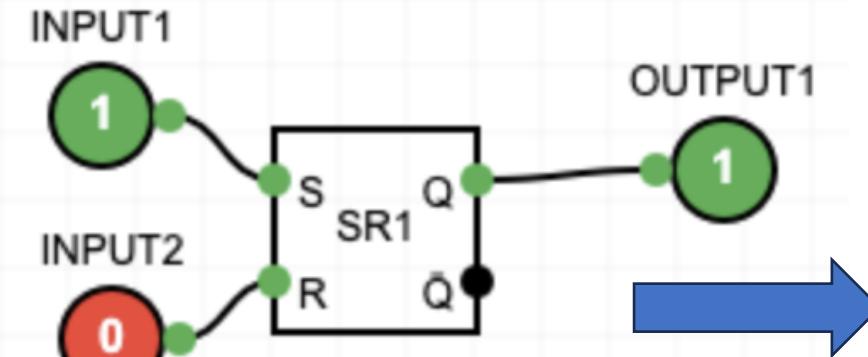
S	R	Q
0	0	Maintain
0	1	0
1	0	1
1	1	???



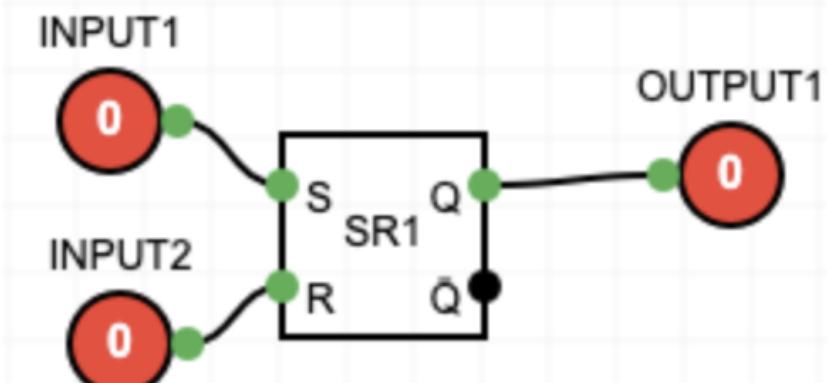
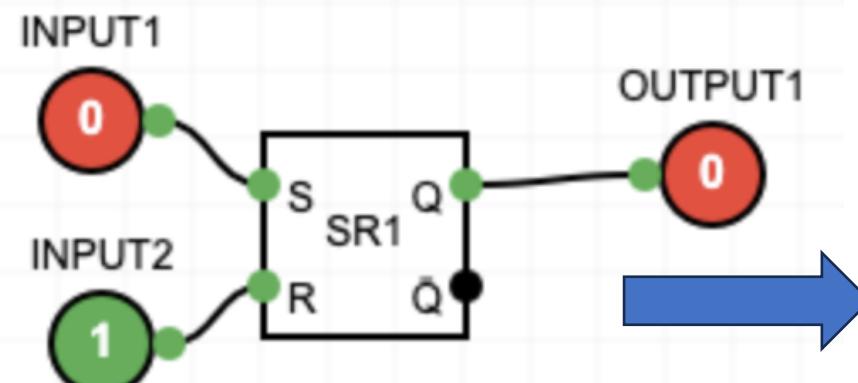
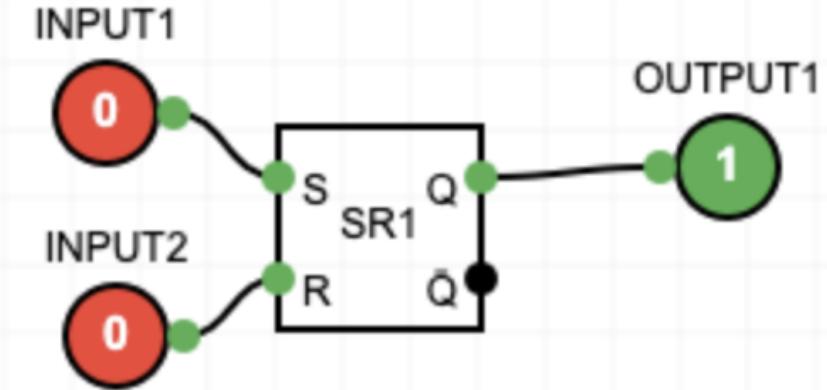
SR Flip-Flop / Latch

S	R	Q
0	0	Maintain
0	1	0
1	0	1
1	1	???

Set / Reset

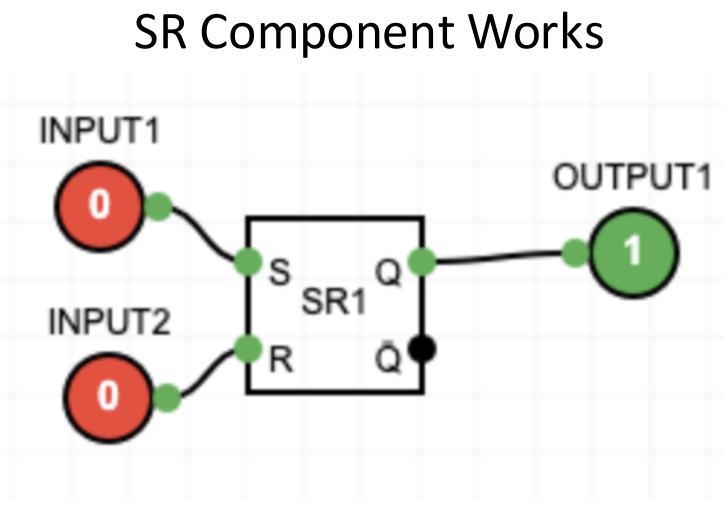
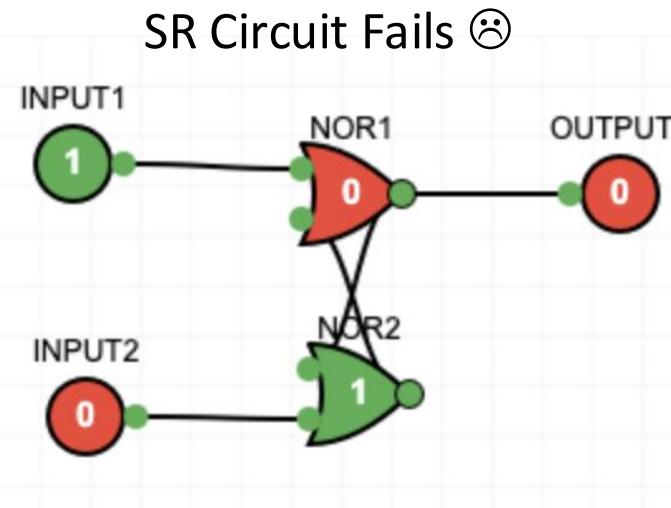
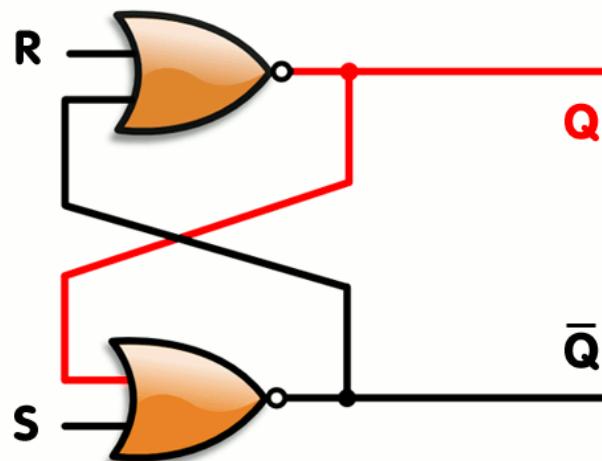


Stored Value



My Simulators Can't Build an SR Latch :(

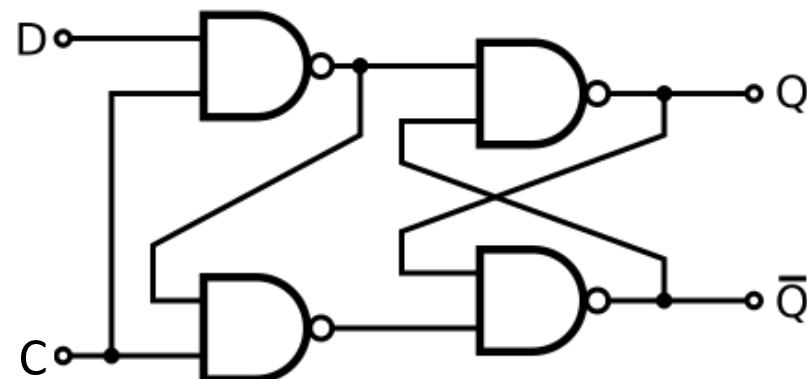
- My Digital Logic Builder does not handle feedback loops so you can't build a SR latch from two NOR gates (unless I fix it)

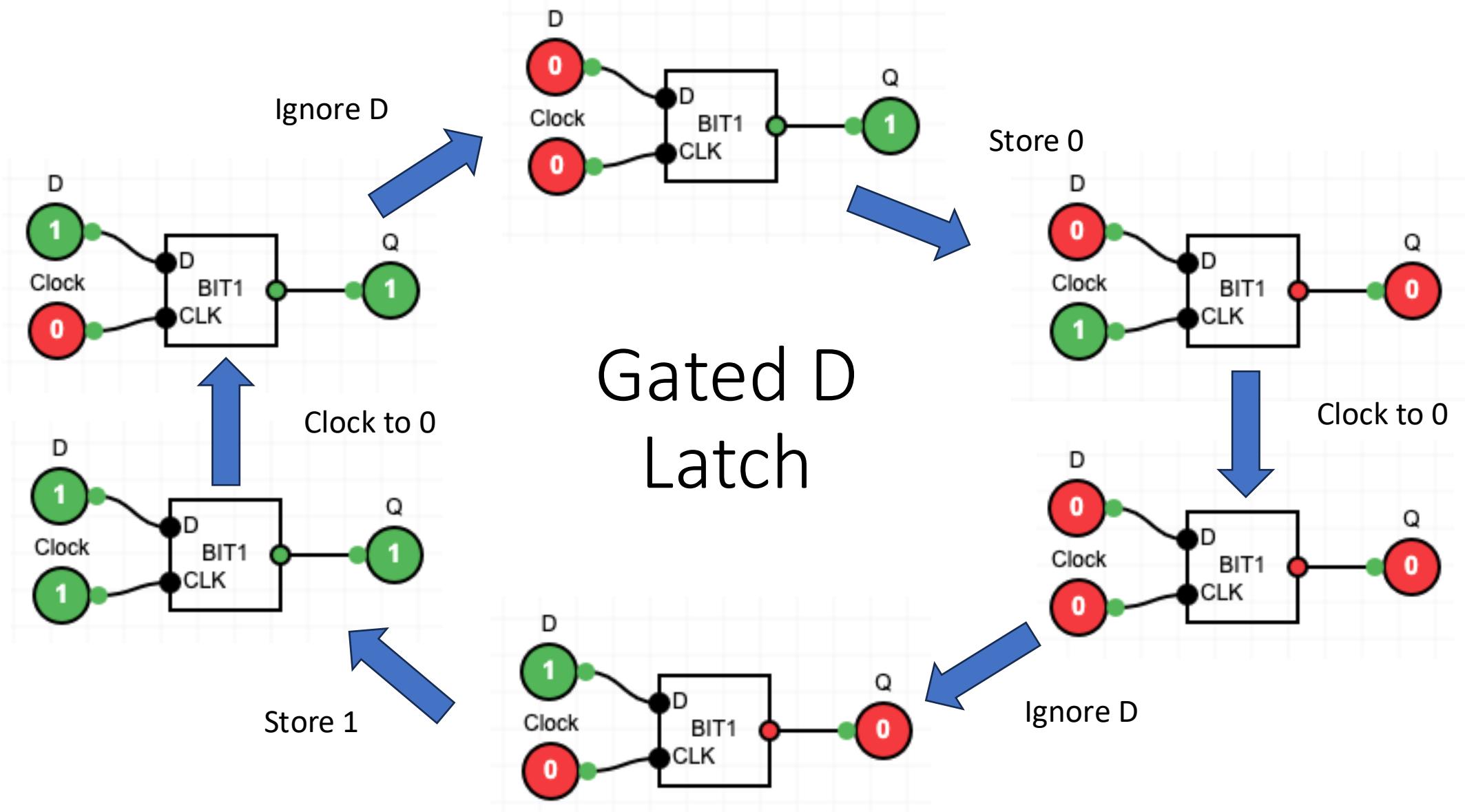


Gated D Latch

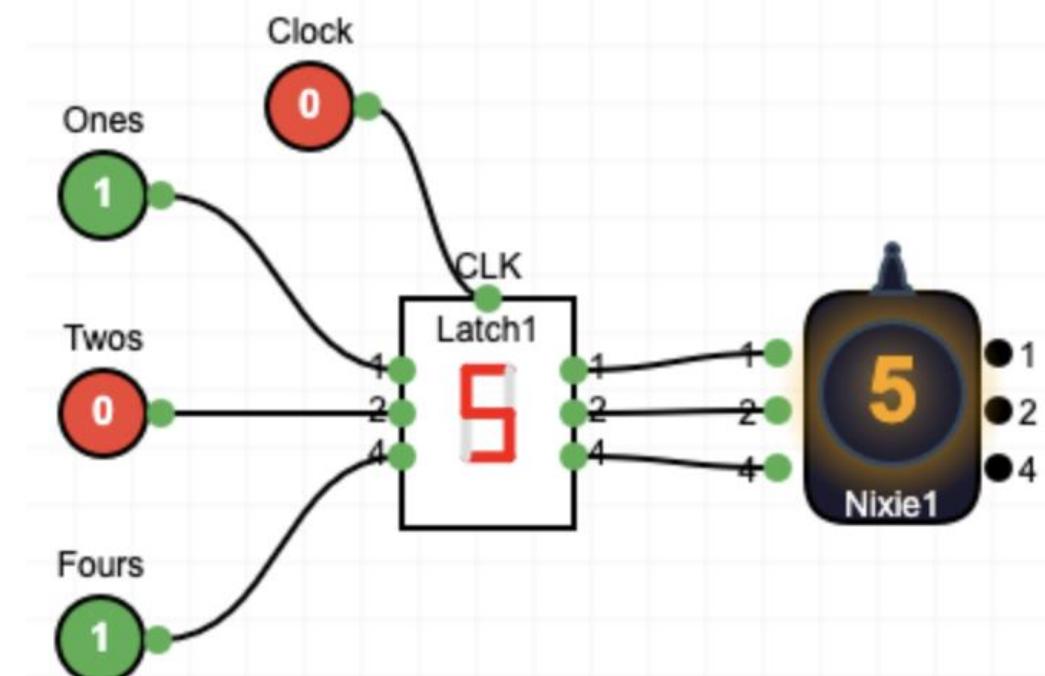
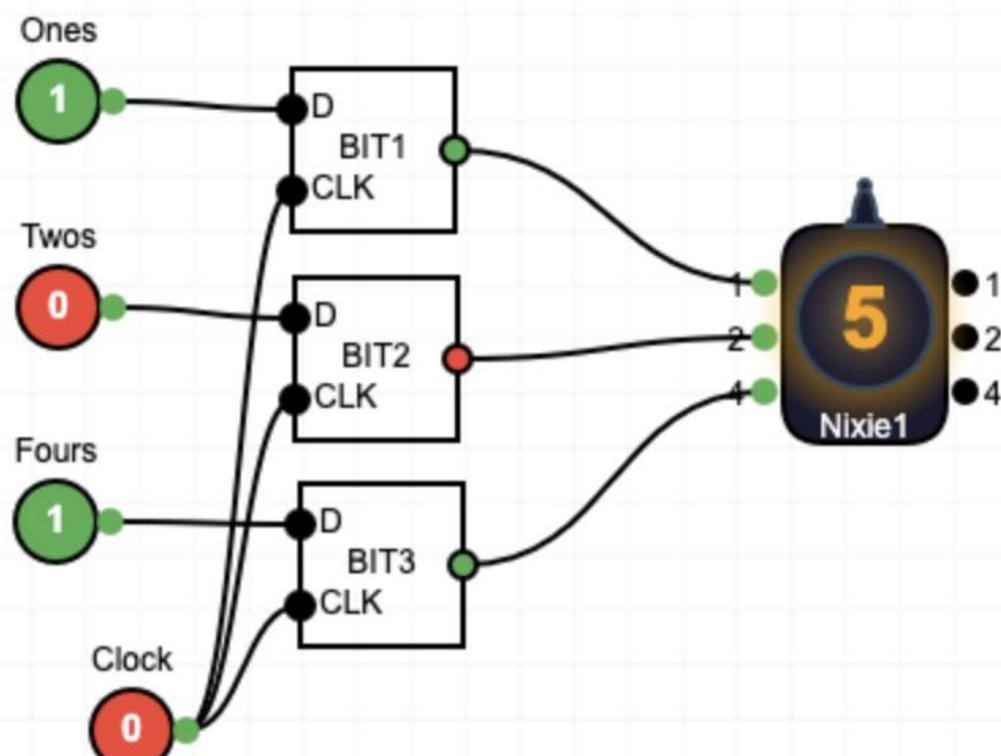
- The Gated D Latch has a D (data) and a C (clock) inputs.
- If C is zero, D is ignored, and the output is the most recent stored value
- If C is 1, D is copied into the internal state

C	D	Q
0	0	Maintain
0	1	Maintain
1	0	0
1	1	1





Stack D Latches to Make a 3-bit Register



Summary

- In this lecture we started with simple gates like AND and OR
- We built latches / registers for storage

Acknowledgements / Contributions

These slides are Copyright 2025- Charles R. Severance (online.dr-chuck.com) as part of www.ca4e.com and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

[Continue new Contributors and Translators here](#)

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates