

Warm up!

Can I make an n -input AND
from 2-input AND gates?

Warm up!

What about XOR gates?

More bits, circuits, adders

CS 2130: Computer Systems and Organization 1

February 1, 2023

Announcements

- TA Office Hours start tonight!
 - Wednesdays, Rice 011
 - Thurs-Sun, Olsson 001
- Please join our Discord server
- Homework 1 due Monday

Quiz Review

Operations

So far, we have discussed:

- Addition: $x + y$
 - Can get multiplication
- Subtraction: $x - y$
 - Can get division, but more difficult
- Unary minus (negative): $-x$
 - Flip the bits and add 1

Operations (on Integers)

Bit vector: fixed-length sequence of bits (ex: bits in an integer)

- Manipulated by bitwise operations

Bitwise operations: operate over the bits in a bit vector

- Bitwise not: $\sim x$ - flips all bits (unary)
- Bitwise and: $x \ \& \ y$ - set bit to 1 if x, y have 1 in same bit
- Bitwise or: $x \ | \ y$ - set bit to 1 if either x or y have 1
- Bitwise xor: $x \ ^ \ y$ - set bit to 1 if x, y bit differs

Operations (on Integers)

- Logical not: $!x$
 - $!0 = 1$ and $!x = 0, \forall x \neq 0$
 - Useful in C, no booleans
 - Some languages name this one differently
- Left shift: $x \ll y$ - move bits to the left
 - Effectively multiply by powers of 2
- Right shift: $x \gg y$ - move bits to the right
 - Effectively divide by powers of 2
 - Signed (extend sign bit) vs unsigned (extend 0)

Floating Point Numbers

Four cases:

- **Normalized:** What we saw last time

$$s \ eeee \ ffff = \pm 1.ffff \times 2^{eeee - \text{bias}}$$

- **Denormalized:** Exponent bits all 0

$$s \ eeee \ ffff = \pm 0.ffff \times 2^{1 - \text{bias}}$$

- **Infinity:** Exponent bits all 1, fraction bits all 0 (i.e., $\pm\infty$)
- **Not a Number (NaN):** Exponent bits all 1, fraction bits not all 0

Our story so far

- Transistors
- Information modeled by voltage through wires (1 vs 0)
- Gates: $\&$ $|$ \sim \wedge

- Multi-bit values: representing integers
 - Signed and unsigned
 - Bitwise operators on bit vectors
- Floating point

How to do the *work* of multi-bit?

Multi-bit Mux

Our first multi-bit example: mux

Adder

Add 2 1-bit numbers: a, b

Adder

Can we use this in parallel to add multi-bit numbers?

Adder

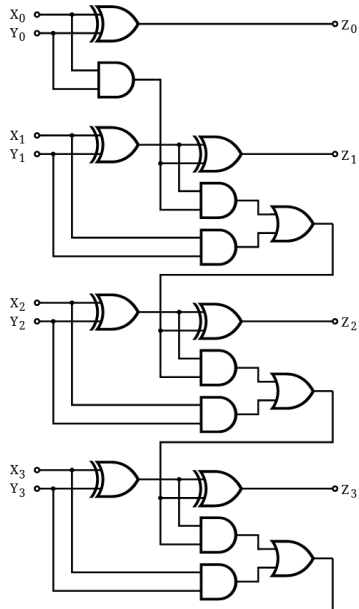
Can we use this in parallel to add multi-bit numbers? What is missing?
Consider:

$$\begin{array}{r} 11 \\ + 01 \\ \hline \end{array}$$

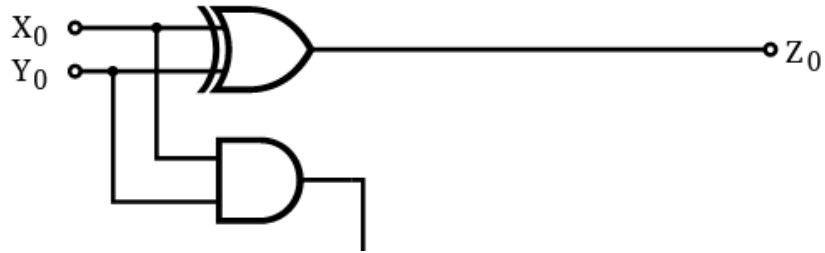
3-input Adder

Add 3 1-bit numbers: a, b, c

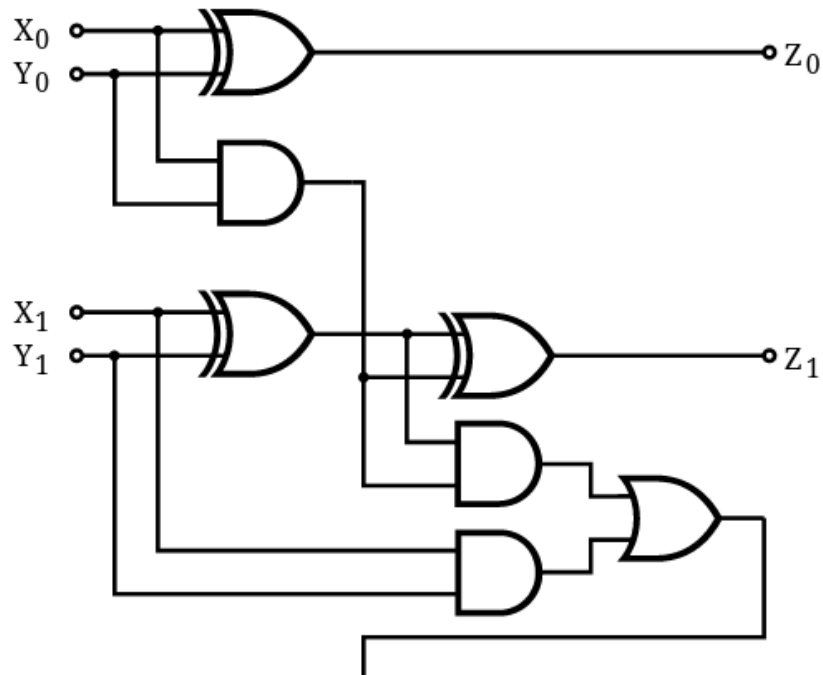
Ripple-Carry Adder



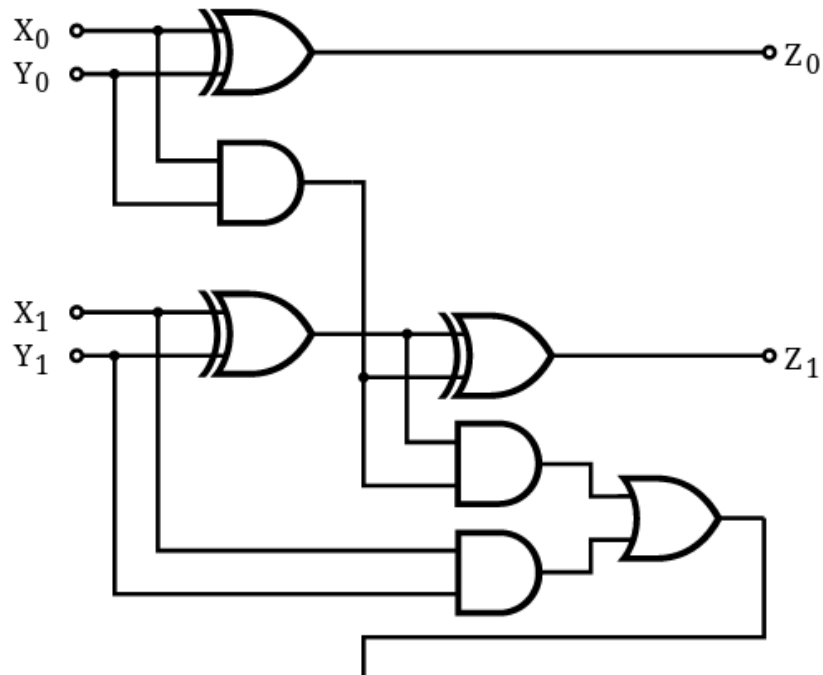
Ripple-Carry Adder: Lowest-order Bit



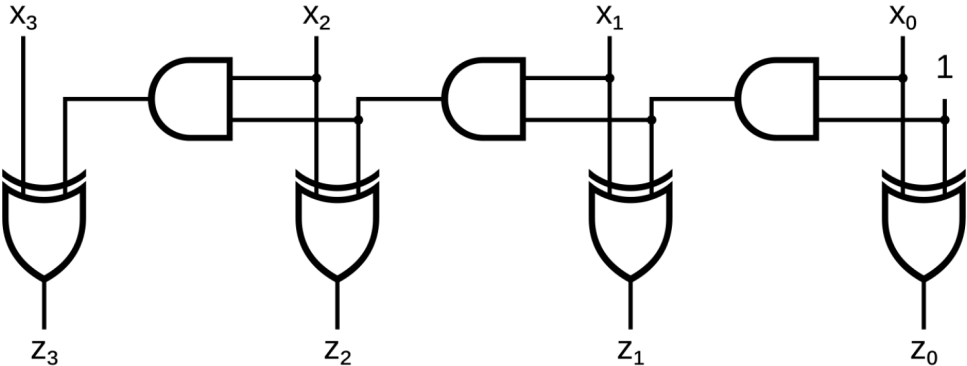
Ripple-Carry Adder: In General



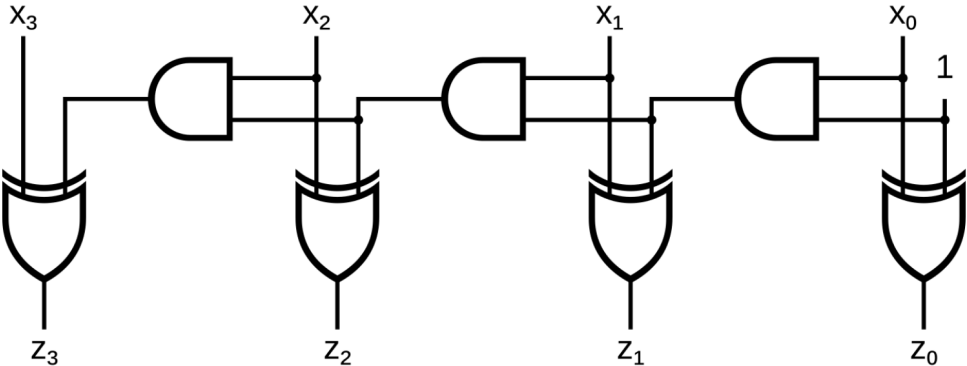
Ripple-Carry Adder: In General



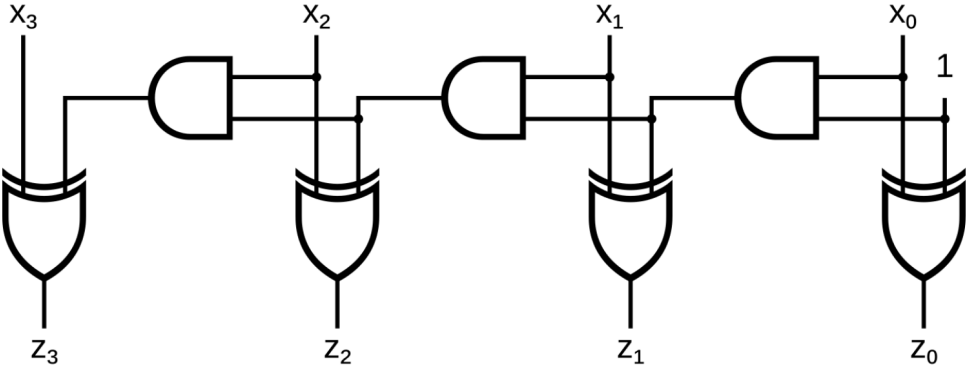
What does this circuit do?



What does this circuit do?

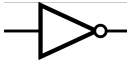


Increment Circuit



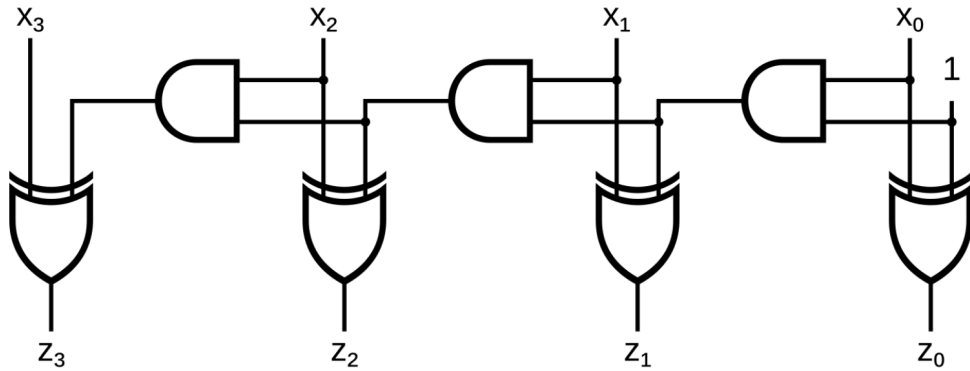
Gate Delay

What happens when I change my input?



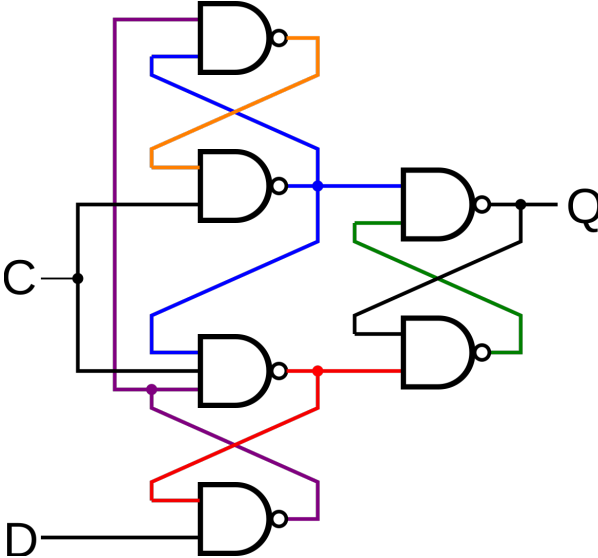
Building a Counter

Building a Counter

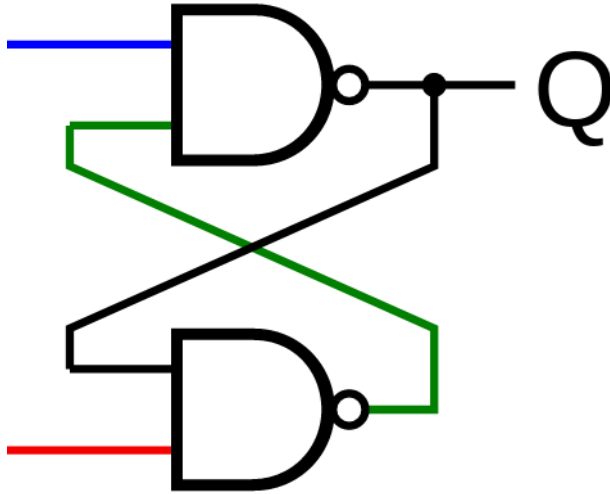


Building a Counter - Waiting

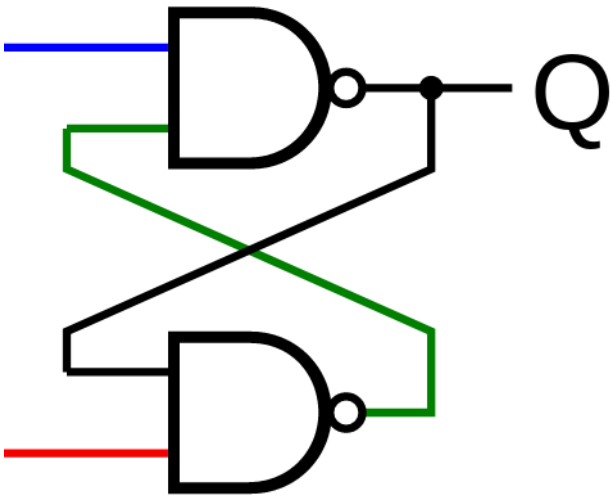
1-bit Register Circuit



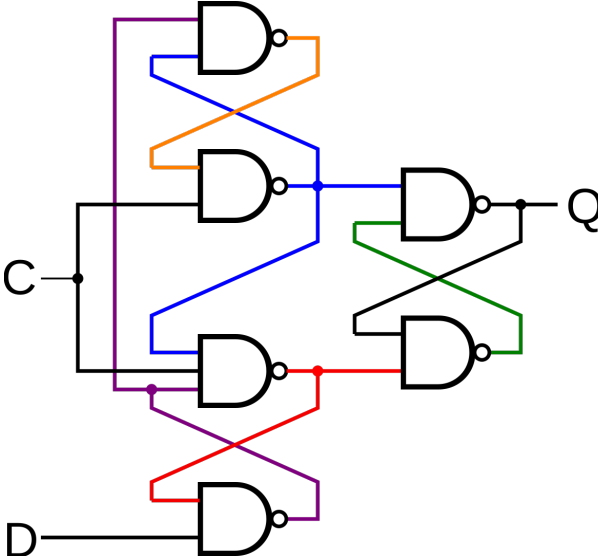
1-bit Register Circuit



1-bit Register Circuit



1-bit Register Circuit



Building a Counter

