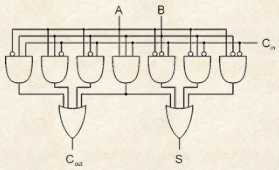
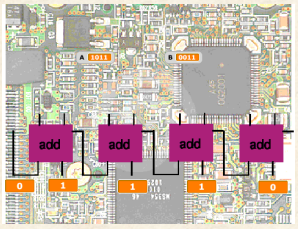
UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 202
Introduction to Computation

Professor Andrea Arpaci-Dusseau
Fall 2010

Lecture 11: How does a computer… do arithmetic?





Previous Lecture: Boolean Logic

Boolean logic: Operates on True (1) and False (0)
- Operators: AND, OR, NOT

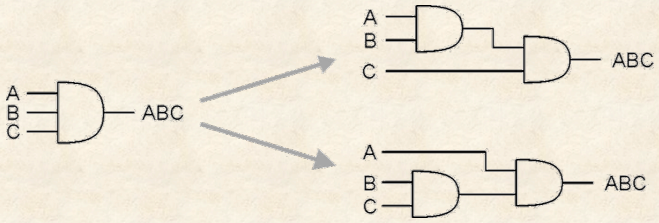
Boolean expression, truth table, combination circuit
- All equivalent

Truth table: Give output for all input combinations
- K inputs → ?? Rows needed for all input combinations? 2^k rows

Combinational circuit
- Output is function of current inputs
- No feedback or cycles in circuit

Shorthand in Logic Gates

Can draw AND and OR gates with more than two inputs



AND gate: Output is 1 if and only if all inputs are 1

OR gate: Output is 1 if one or more inputs are 1

Review: Sum of Products

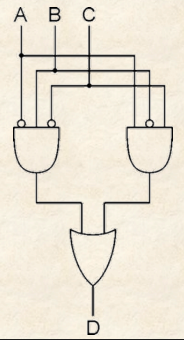
Can implement ANY truth table with AND, OR, NOT

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

What is the algorithm?

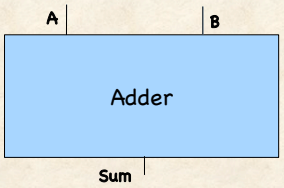
1. AND combinations that yield a "1" in the truth table.
2. OR the results of the AND gates.

What does circuit look like?



Today's Challenge

Can we perform addition using only AND, OR, and NOT gates?



Can compute logic circuit for any function

"Sum = A plus B" is a function of only inputs

Therefore, can create circuit for addition!

Approach #1: Sum of Products

Inputs: Two binary numbers A and B

View each bit of number as an input; 2 bits each

- A = a_1a_0
- B = b_1b_0

Output is a three-bit binary number

- Sum = $s_2s_1s_0$

Construct truth table of all input combinations

- 4 bits of input
- 2^4 = 16 rows of table

Use sum-of-products algorithm for three outputs

- s2, s1, and s0

a_1	a_0	b_1	b_0	s_2	s_1	s_0
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	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			

How to construct addition circuit?

1) Set up Truth Table; Enumerate all 16 input combinations for a1, a0, b1, b0

2) Determine decimal number corresponding to each input

3) Calculate decimal sum as output

4) Translate decimal number to binary

5) Create 3 circuits (Sum-of-products) for s2, s1, s0

Approach #1: Sum of Products

What if 32-bit integers instead of 2-bits??

Number of inputs to circuit?

- 2*32

How many rows in truth table?

- 2^{64}

Implication:

Need a fundamentally different approach for any real architecture!

Approach #2: Modular Design

Modular Design
- Library of small number of basic components
- Combine together to achieve desired functionality
- Basic principle of modern industrial design

Requires some insight to design component

What algorithm do you use for decimal addition?

```
  6925
+ 8729
------
 15654
```

Informal:
- Memorize facts for adding numbers 0..9 to 0..9 + 1 (carry)
- Apply facts to ones position; record units; carry tens
- Repeat for each position (tens, hundred, thousands) w/ carry
- Record final carry out

Algorithm for binary addition?

```
      1 1 0 0 1
 25     11001      16 + 8 + 1 = 25
+29     11101      16 + 8 + 4 + 1 = 29
------------
 54    110110      32+ 16 + 4 + 2 = 54
```

We know these facts:

```
0 + 0 + 0 = 00
1 + 0 + 0 = 01
1 + 1 + 0 = 10 (two)
1 + 1 + 1 = 11 (three)
```

Modular Design for Addition

Notation:

Carry bits

$$\begin{array}{rcccccc} & c_{N-1} & c_{N-2} & \cdots & c_1 & c_0 & \\ & a_{N-1} & a_{N-2} & \cdots & a_1 & a_0 & \\ + & b_{N-1} & b_{N-2} & \cdots & b_1 & b_0 & \\ \hline s_N & s_{N-1} & s_{N-2} & \cdots & s_1 & s_0 & \end{array}$$

Repeatedly (N times) do 1-bit full add:
Take cin, a, b as input
Compute cout, s as output

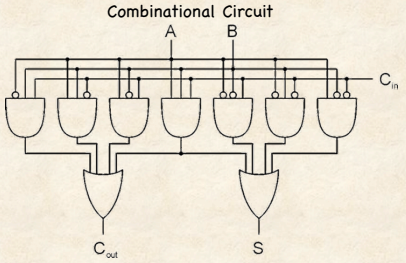
Module: 1-bit Full Adder

Implement using just AND, OR, NOT

- Add two bits (A, B) and carry-in (C_{in}), for one-bit sum (S) and carry-out (C_{out})

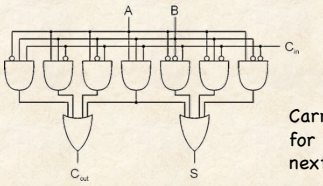
Truth Table

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

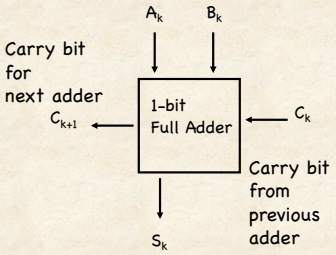


Abstraction: 1-bit Full Adder

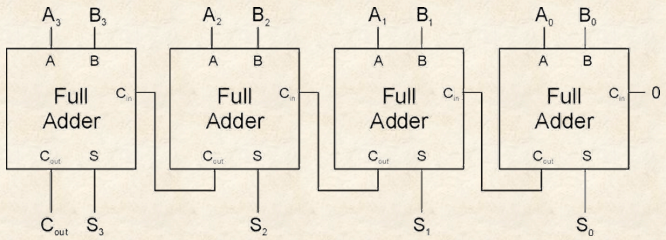
Represent this circuit:



With this diagram:



Four-bit Adder

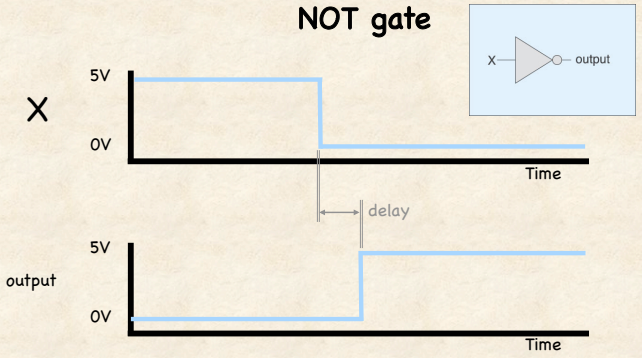


Do you find anything strange or disturbing about this adder?

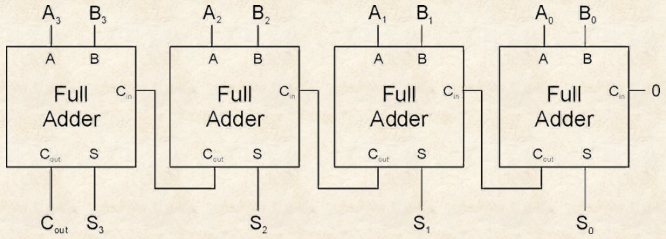
To compute sum of higher bits need results from lower bits

"Ripple-carry" adder

Timing Diagram



Four-bit Adder



How many gate delays until output settles?

Each 1-bit adder requires 2 gate delays (AND + OR gates)

4 adders * 2 gate delays/adder = 8 gate delays

Adder: Example

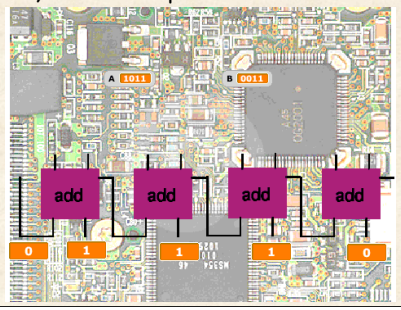
	1 1 0 0 1 0
25	11001
+29	11101
54	110110

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

Create Addition Circuit in Scratch!

Add two 4-bit numbers to produce 1 5-bit number

- Only use AND, OR, and NOT!
- 4 nearly identical Sprites



Scripts for 1 Full-bit Adder

Scripts identical across 4 Adder Sprites

Each has private variable for a, b, cin, cout, s

To connect adders, simply set cin to cout of "previous" Sprite

Summary

Today's Topics

- Combinational circuits: Output computed from inputs
- We can do addition with just AND, OR, and NOT!

Reading:

- Chapter 4.5–4.6 (pp 152–183) in Invitation to CS

Announcements

- Homework 4 Due Friday: Binary and Variables