CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Muxes, Adders, and ALUs

Instructors:
Randy H. Katz
David A. Patterson
http://inst.eecs.Berkeley.edu/~cs61c/fa10

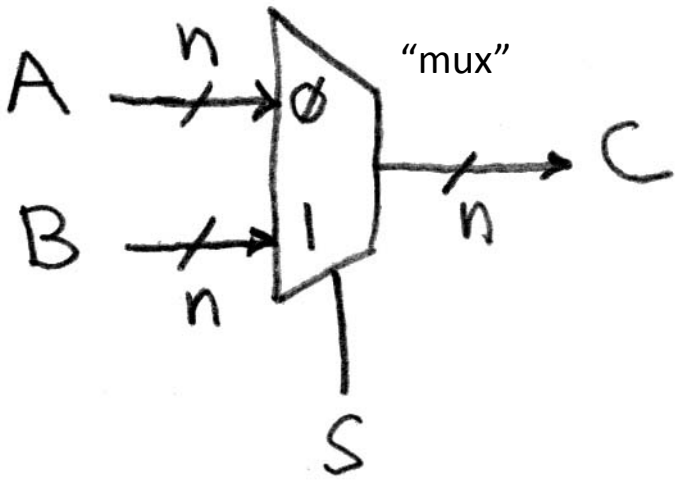
Agenda

- Multiplexer
- Administrivia
- Technology Break
- ALU Design

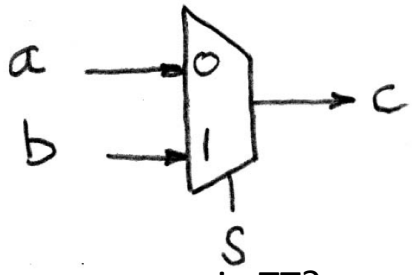
Agenda

- Multiplexer
- Administrivia
- Technology Break
- ALU Design

Data Multiplexer (e.g., 2-to-1 x n-bit-wide)



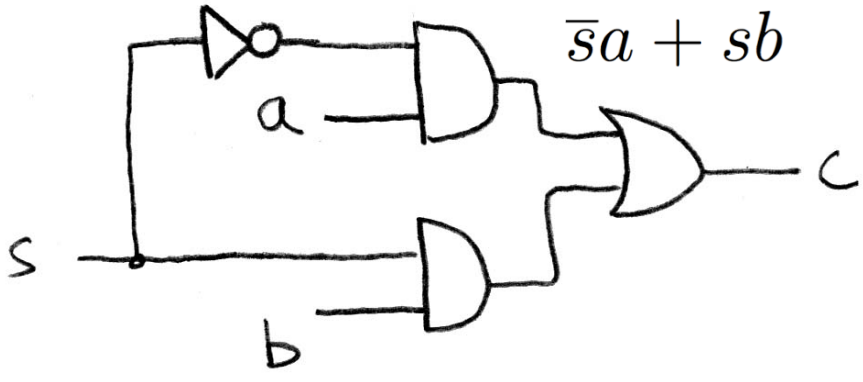
N Instances of 1-bit-Wide Mux



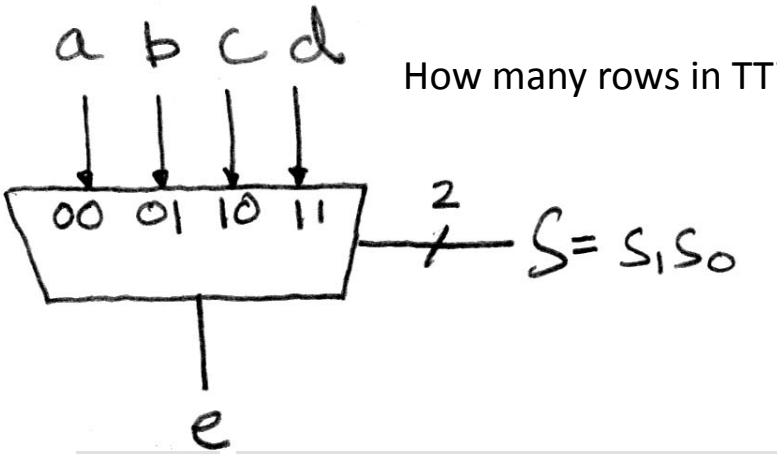
How many rows in TT?

$$
\begin{aligned}
c &= \overline{s}a\overline{b} + \overline{s}ab + s\overline{a}b + sab \\
&= \overline{s}(a\overline{b} + ab) + s(\overline{a}b + ab) \\
&= \overline{s}(a(\overline{b} + b)) + s((\overline{a} + a)b) \\
&= \overline{s}(a(1) + s((1)b) \\
&= \overline{s}a + sb
\end{aligned}
$$

How Do We Build a 1-bit-Wide Mux?



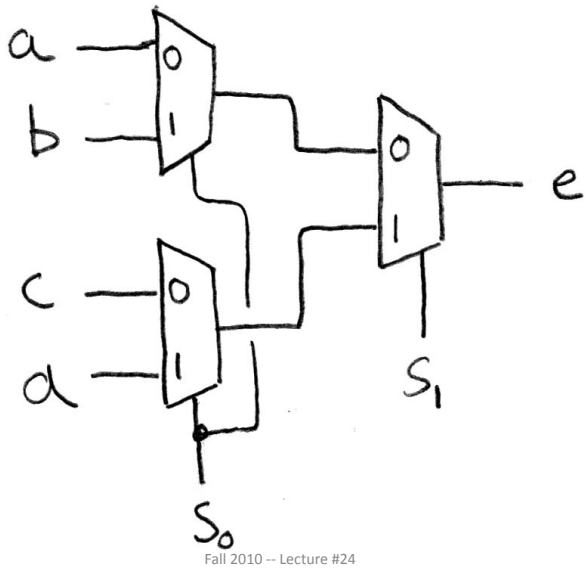
4-to-1 Multiplexer



How many rows in TT?

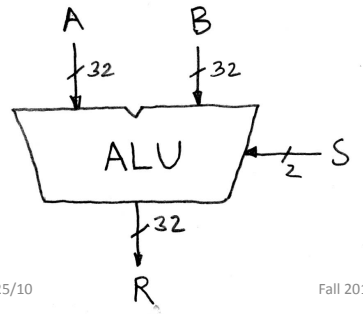
$$e = \overline{s_1}\,\overline{s_0}a + \overline{s_1}s_0b + s_1\overline{s_0}c + s_1s_0d$$

Alternative Hierarchical Approach



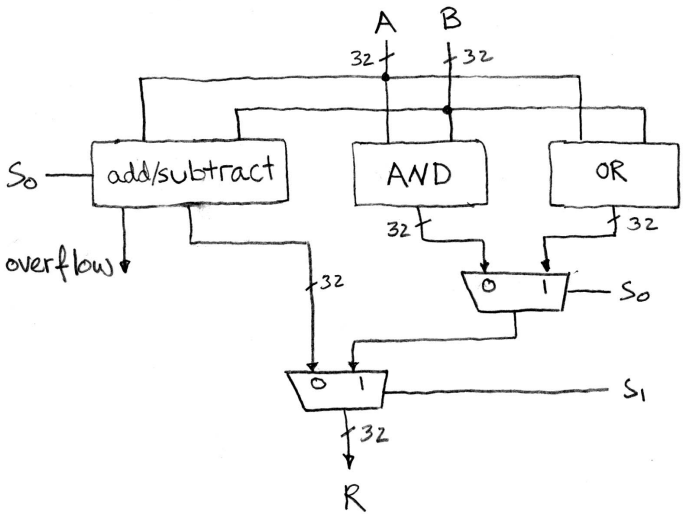
Arithmetic and Logic Unit

- Most processors contain a special logic block called "Arithmetic and Logic Unit" (ALU)
- We'll show you an easy one that does ADD, SUB, bitwise AND, bitwise OR



when S=00, R=A+B
when S=01, R=A-B
when S=10, R=A AND B
when S=11, R=A OR B

Simple ALU



Agenda

- Mutiplexers
- Administrivia
- Technology Break
- ALU

Agenda

- Mux + Adder Design
- Administrivia
- Technology Break
- ALU Design

Agenda

- Multiplexer
- Administrivia
- Technology Break
- ALU Design

Adder/Subtractor: One-bit adder Least Significant Bit

	a_3	a_2	a_1	a_0
+	b_3	b_2	b_1	b_0
	s_3	s_2	s_1	s_0

a_0	b_0	s_0	c_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$s_0 = a_0$$
$$c_1 = a_0$$

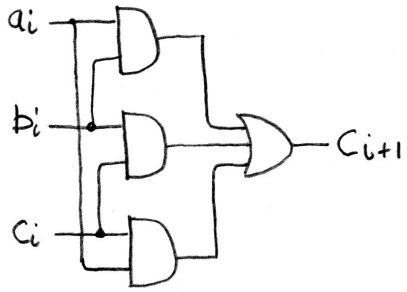
Adder/Subtractor: One-bit adder (1/2) …

	a_3	a_2	a_1	a_0
+	b_3	b_2	b_1	b_0
	s_3	s_2	s_1	s_0

a_i	b_i	c_i	s_i	c_{i+1}
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

$$s_i =$$
$$c_{i+1} =$$

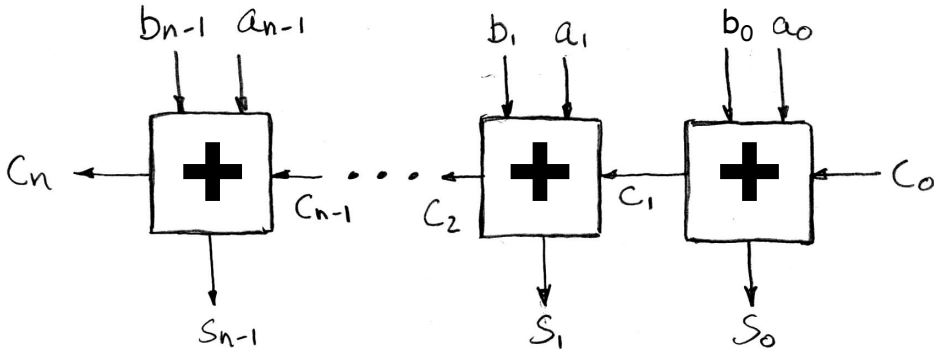
Adder/Subtractor: One-bit Adder (2/2) …



$$s_i = \text{XOR}(a_i, b_i, c_i)$$
$$c_{i+1} = \text{MAJ}(a_i, b_i, c_i) = a_i b_i + a_i c_i + b_i c_i$$

N x 1-bit Adders ⇒ 1 N-bit Adder

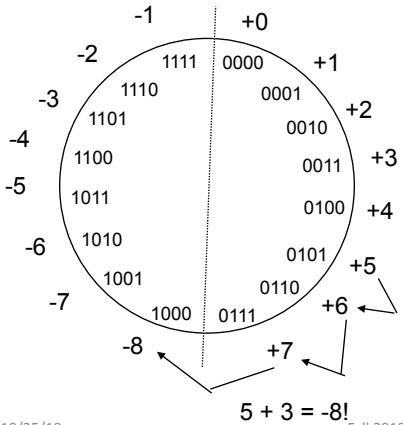
Connect Carry Out i-1 to Carry in i:



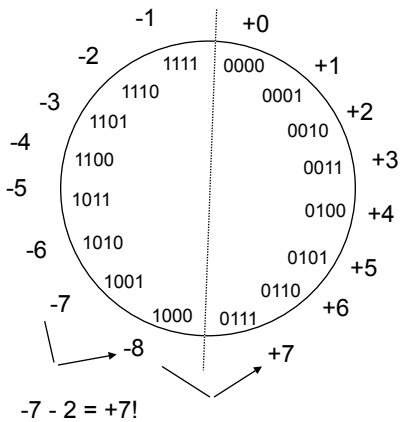
Overflow Conditions

Add two positive numbers to get a negative number

or two negative numbers to get a positive number



5 + 3 = -8!



-7 - 2 = +7!

Overflow Conditions

5	0 1 1 1 0 1 0 1	
3	0 0 1 1	
-8	1 0 0 0	

Overflow

-7	1 0 0 0 1 0 0 1	
-2	1 1 0 0	
7	1 0 1 1 1	

Overflow

5	0 0 0 0 0 1 0 1	
2	0 0 1 0	
7	0 1 1 1	

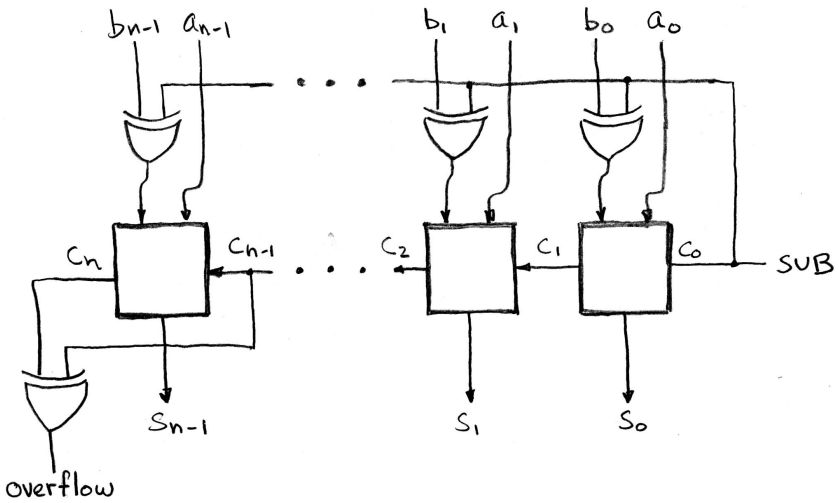
No overflow

-3	1 1 1 1 1 1 0 1	
-5	1 0 1 1	
-8	1 1 0 0 0	

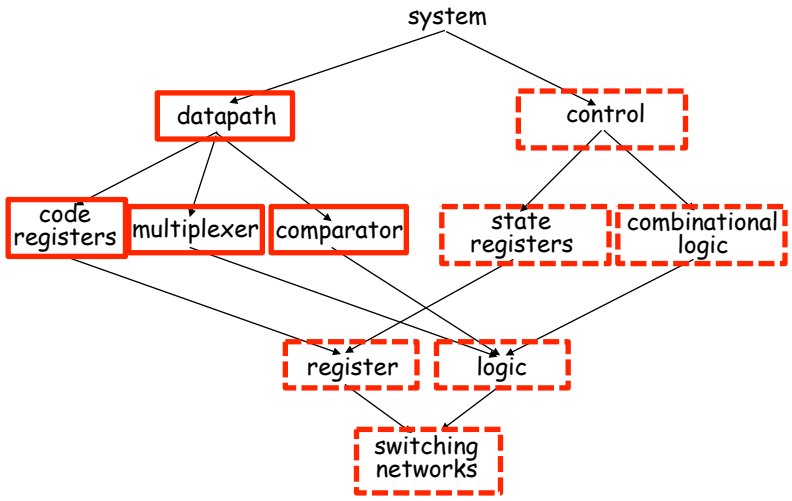
No overflow

Overflow when carry in to sign does not equal carry out: C_n xor C_{n-1}

Twos Complement Adder/ Subtractor



Design Hierarchy



Summary

- Use muxes to select among input
  - S input bits selects 2S inputs
  - Each input can be n-bits wide, indep of S
- Can implement muxes hierarchically
- ALU can be implemented using a mux
  - Coupled with basic block elements
- N-bit adder-subtractor done using N 1-bit adders with XOR gates on input
  - XOR serves as conditional inverter