UC Berkeley
Teaching Professor
Dan Garcia

CS61C

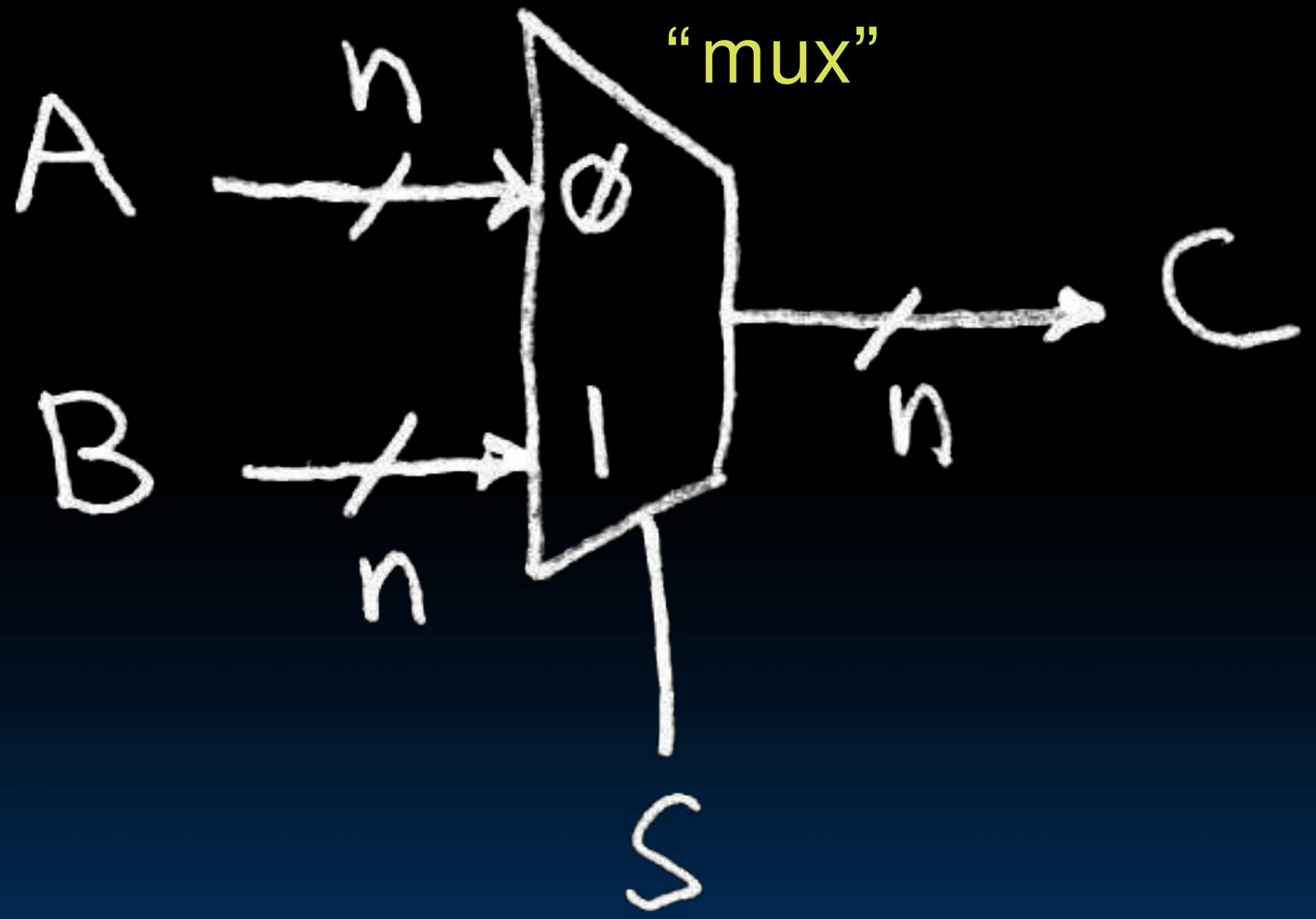
Great Ideas in Computer Architecture (a.k.a. Machine Structures)

UC Berkeley
Professor
Bora Nikolić

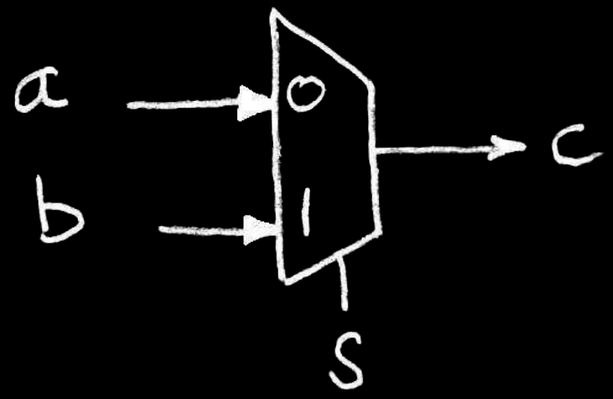
Combinational Logic Blocks

Data Multiplexors

Data Multiplexor (here 2-to-1, n-bit-wide)



N instances of 1-bit-wide mux



How many rows in TT?

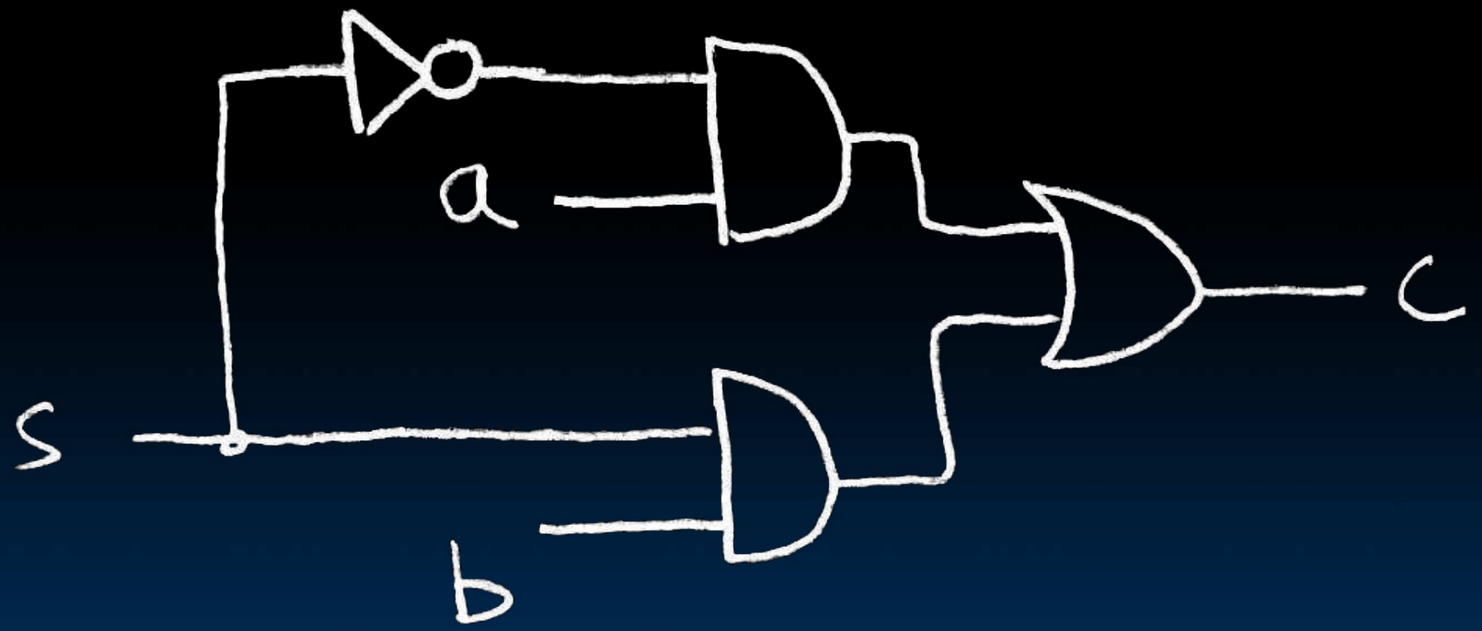
s	ab	c
0	00	0
0	01	0
0	10	1
0	11	1
1	00	0
1	01	1
1	10	0
1	11	1

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

s	c
0	a
1	b

How do we build a 1-bit-wide mux?

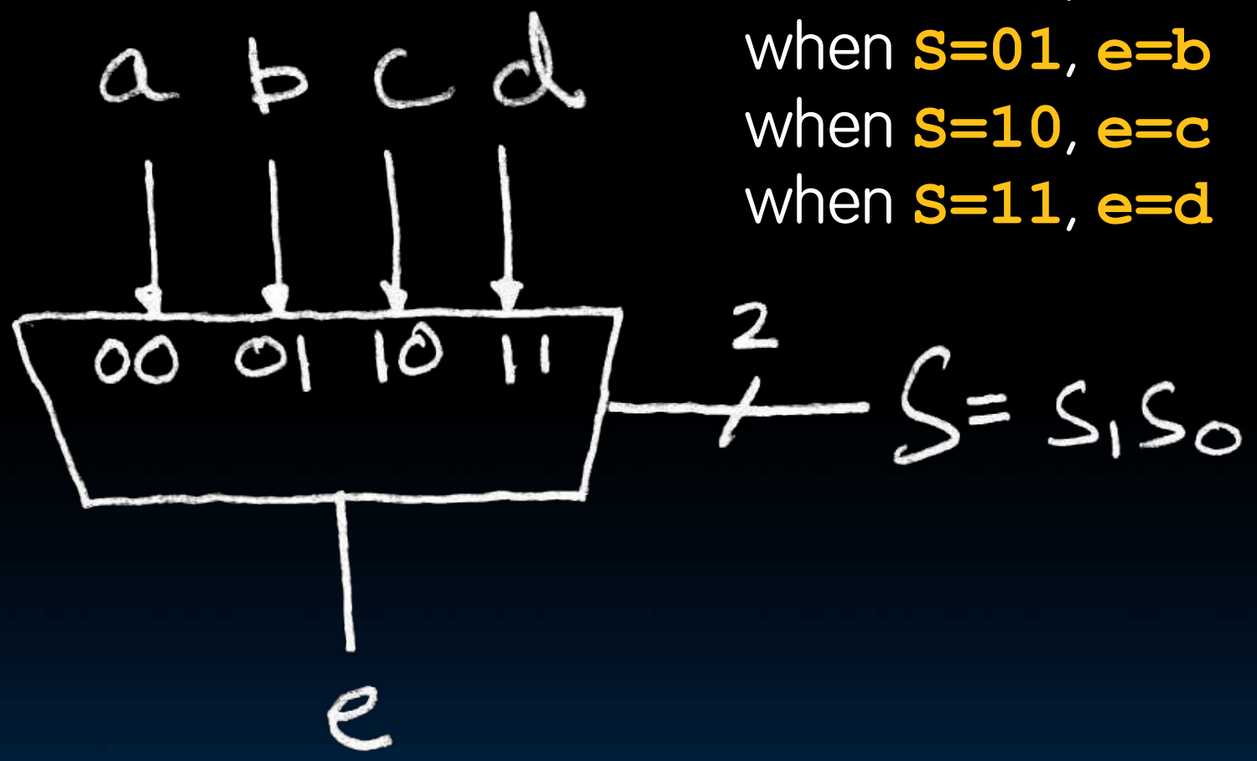
$$\overline{s}a + sb$$



4-to-1 Multiplexor?

- How many rows in the Truth Table?

when **S=00**, **e=a**
when **S=01**, **e=b**
when **S=10**, **e=c**
when **S=11**, **e=d**

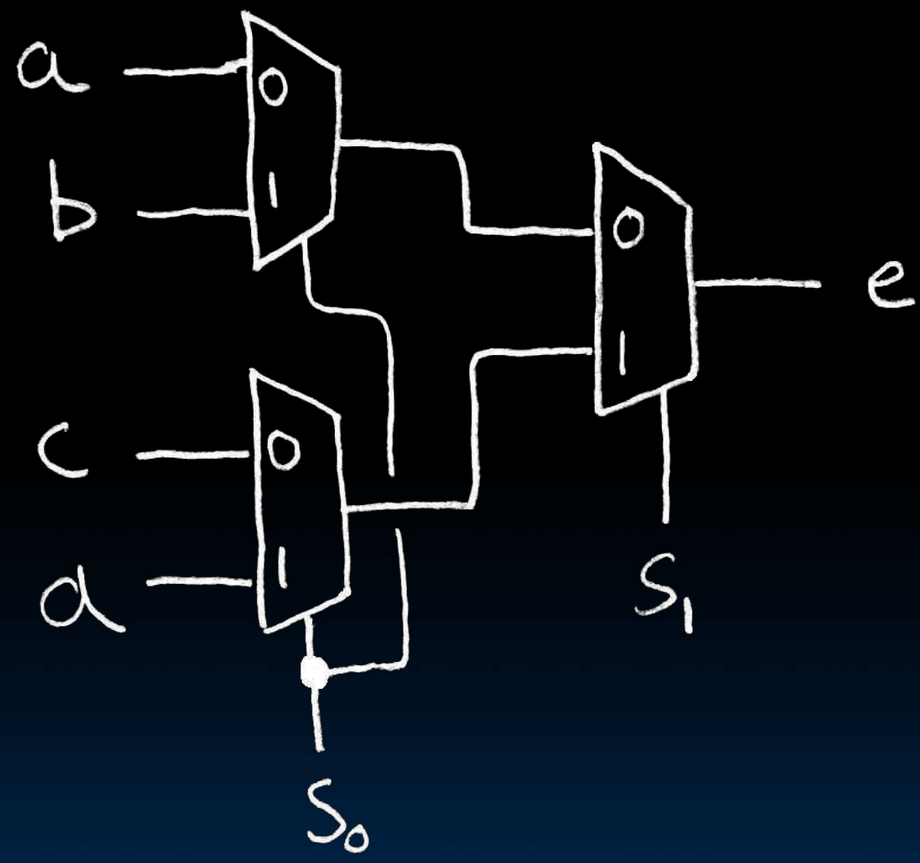


$s_1 s_0$	c
0 0	a
0 1	b
1 0	c
1 1	d

$$e = \overline{s_1} \cdot \overline{s_0} a + \overline{s_1} s_0 b + s_1 \overline{s_0} c + s_1 s_0 d$$

Mux: is there any other way to do it?

Hint: NCAA tourney!

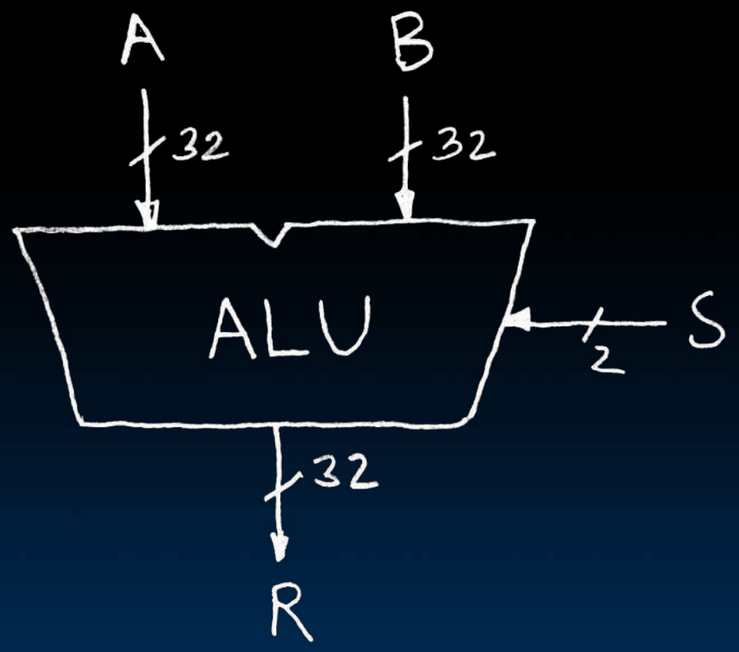


Ans: Hierarchically!

Arithmetic Logic Unit (ALU)

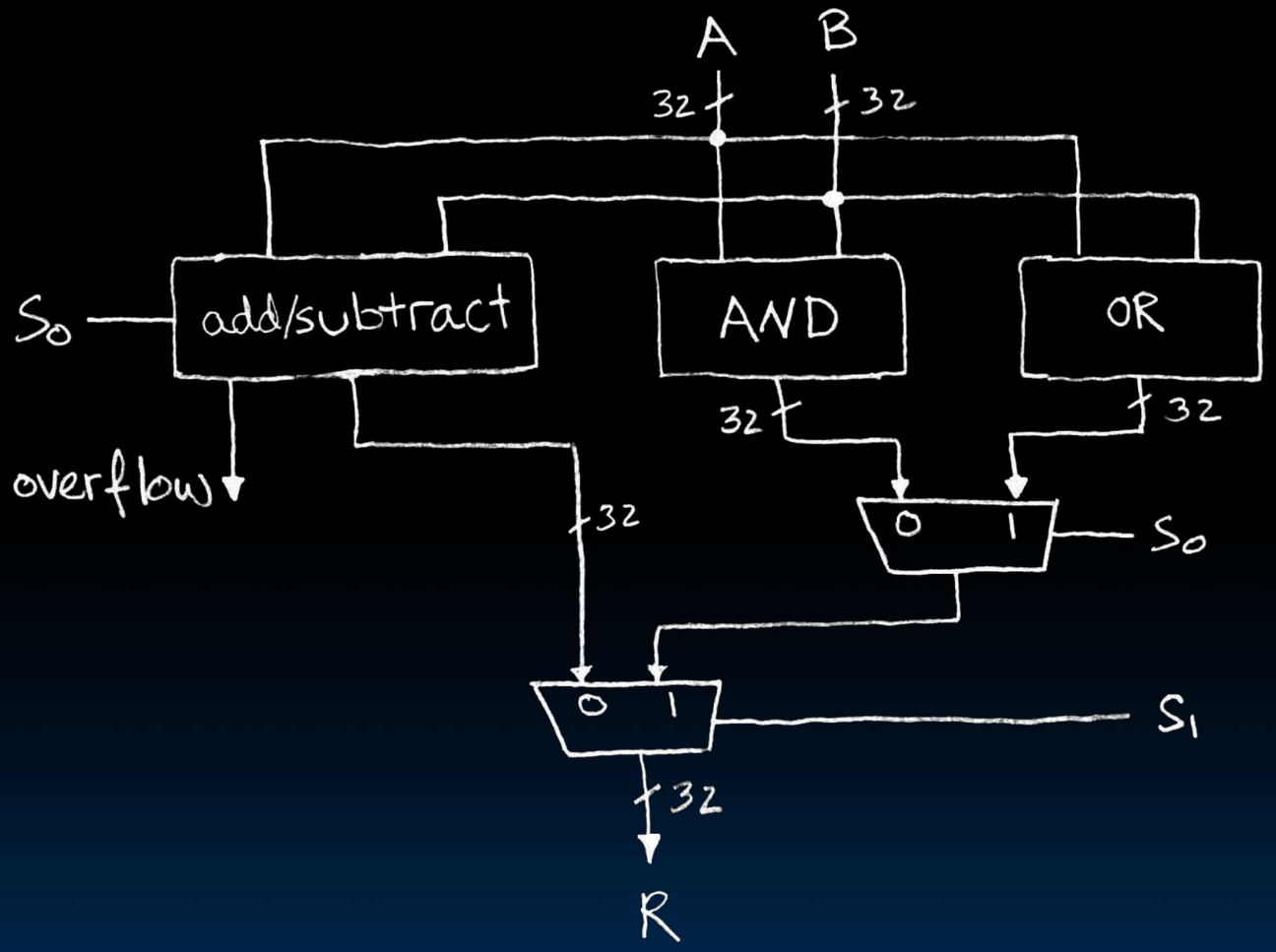
Arithmetic and Logic Unit

- 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&B`**
when **`S=11`**, **`R=A|B`**

Our simple ALU



Adder / Subtractor

Adder / Subtracter Design – how?

- Truth-table, then determine canonical form, then minimize and implement as we've seen before
- Look at breaking the problem down into smaller pieces that we can cascade or hierarchically layer

Adder / Subtractor – One-bit adder LSB…

$$
\begin{array}{cccc|c}
 & a_3 & a_2 & a_1 & a_0 \\
+ & b_3 & b_2 & b_1 & b_0 \\
\hline
 & s_3 & s_2 & s_1 & s_0
\end{array}
$$

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

$$s_0 = a_0 \text{ XOR } b_0$$
$$c_1 = a_0 \text{ AND } b_0$$

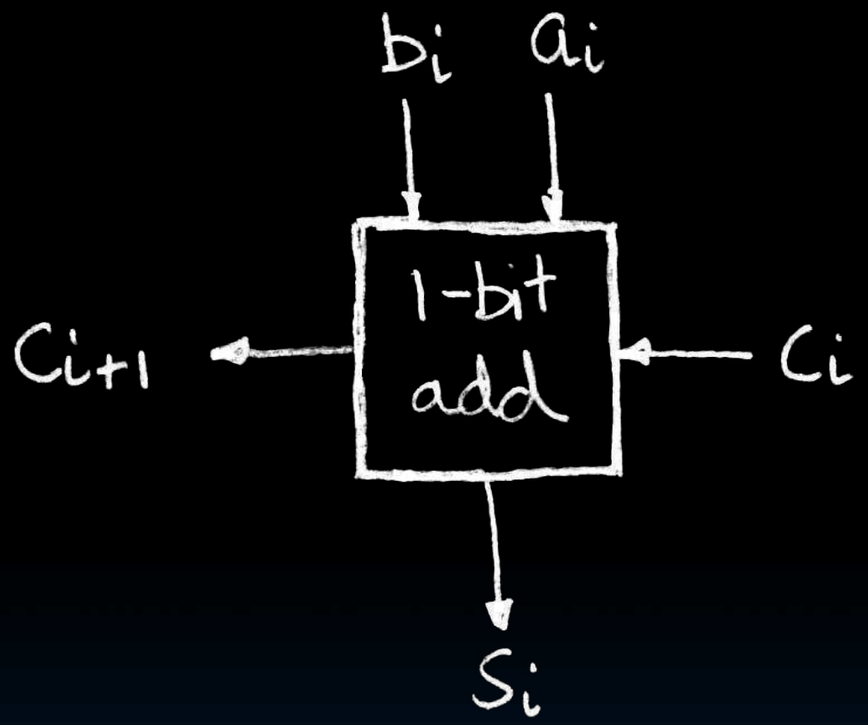
Adder / Subtractor – One-bit adder (1/2)…

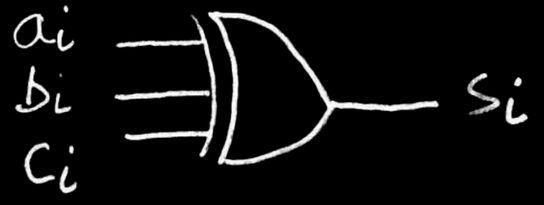
$$\begin{array}{cccccc} & a_3 & a_2 & \boxed{a_1} & a_0 \\ + & b_3 & b_2 & \boxed{b_1} & b_0 \\ \hline & s_3 & s_2 & \boxed{s_1} & s_0 \end{array}$$

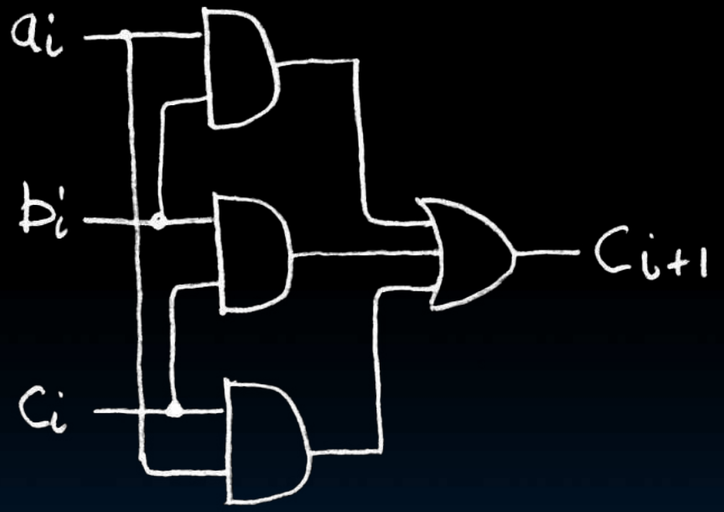
a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\begin{aligned} s_i &= \mathrm{XOR}(a_i, b_i, c_i) \\ c_{i+1} &= \mathrm{MAJ}(a_i, b_i, c_i) = a_i b_i + a_i c_i + b_i c_i \end{aligned}$$

Adder / Subtractor – One-bit adder (2/2)…



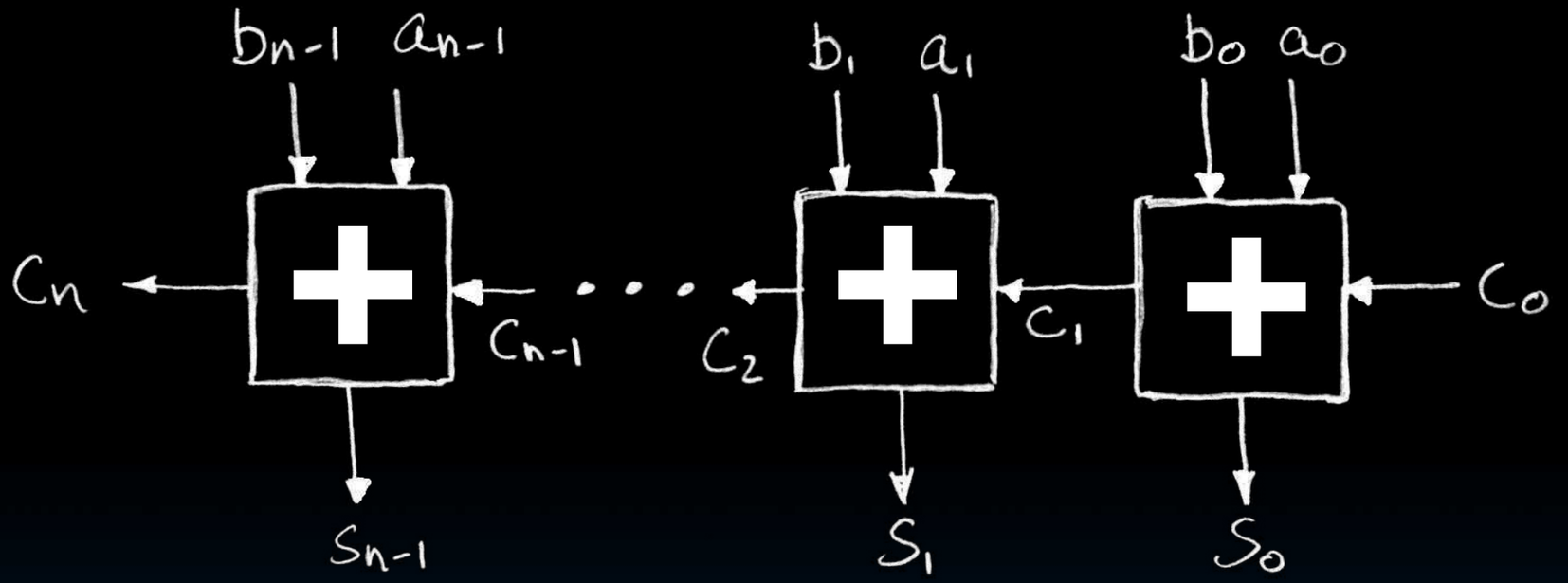




$$s_i = \mathrm{XOR}(a_i, b_i, c_i)$$

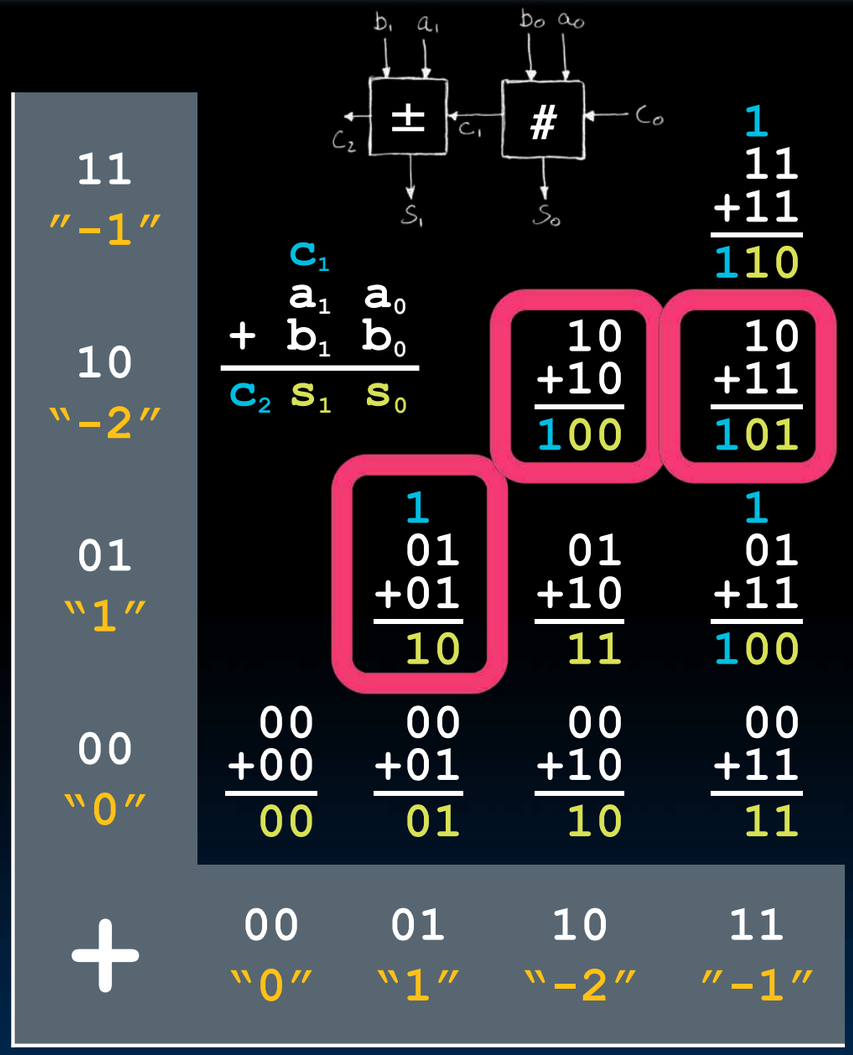
$$c_{i+1} = \mathrm{MAJ}(a_i, b_i, c_i) = a_i b_i + a_i c_i + b_i c_i$$

N 1-bit adders → 1 N-bit adder



What about overflow?
Overflow = c_n?

Sum of two 2-bit numbers…



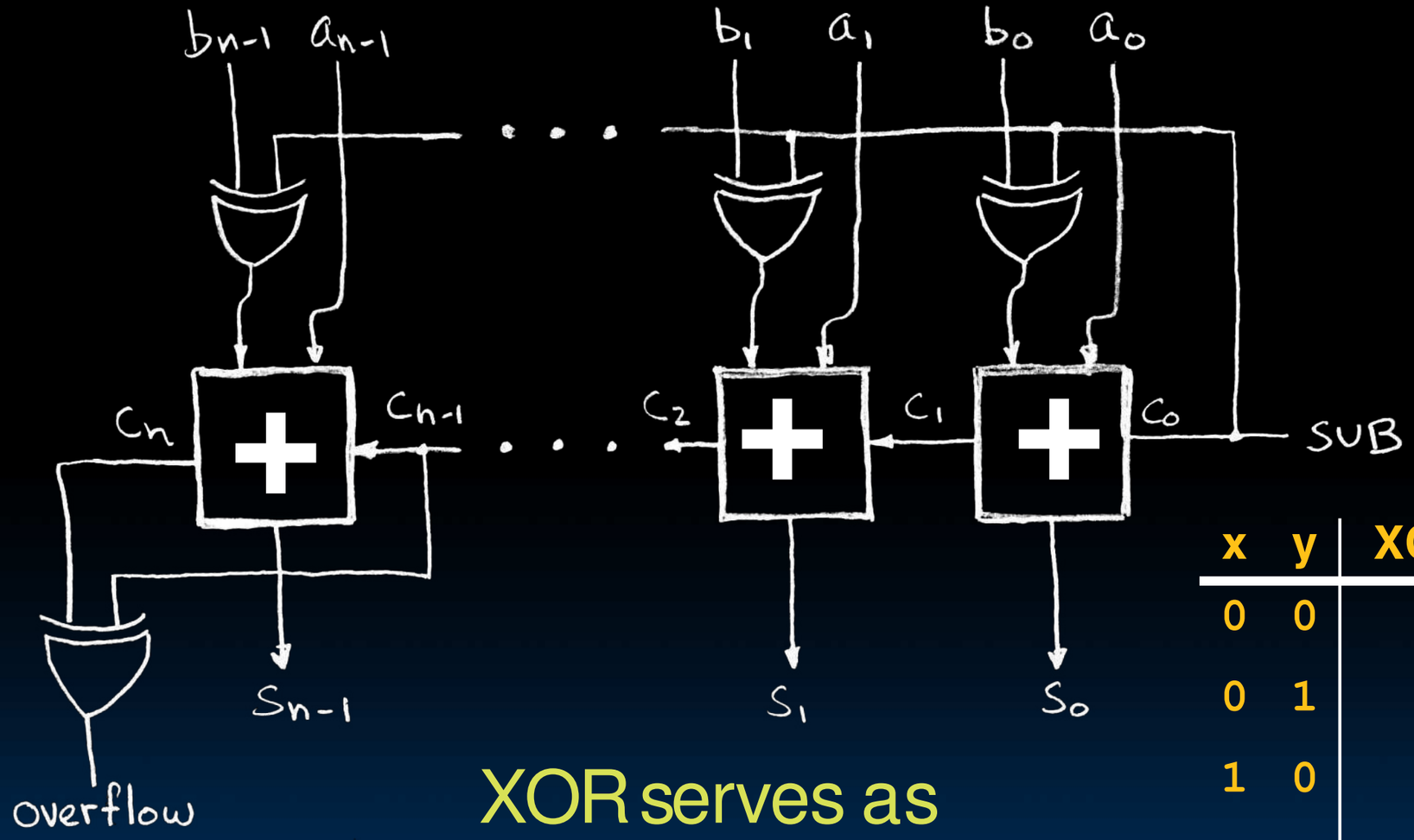
- Let's add
  - First unsigned
  - Then signed (Two's Complement)
    - When do the lowest 2 bits of sum not represent correct sum?
    - Is there a pattern of when this happens? Hint: Check out the carry-bit and the sum-4s-column-bit
- Highest adder
  - C_{in} = Carry-in = c_1, C_{out} = Carry-out = c_2
  - No C_{out} or C_{in} → NO overflow!
  - C_{in} and C_{out} → NO overflow!
  - C_{in}, but no C_{out} → A,B both > 0, overflow!
  - C_{out}, but no C_{in} → A or B are -2, overflow!

What operation is this?

$$\text{overflow} = c_n \text{ XOR } c_{n-1}$$

Subtractor Design

Extremely Clever Subtractor: A-B = A + (-B)



x	y	XOR(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

XOR serves as conditional inverter!

"And In conclusion…"

- Use muxes to select among input
  - S input bits selects 2^S 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