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

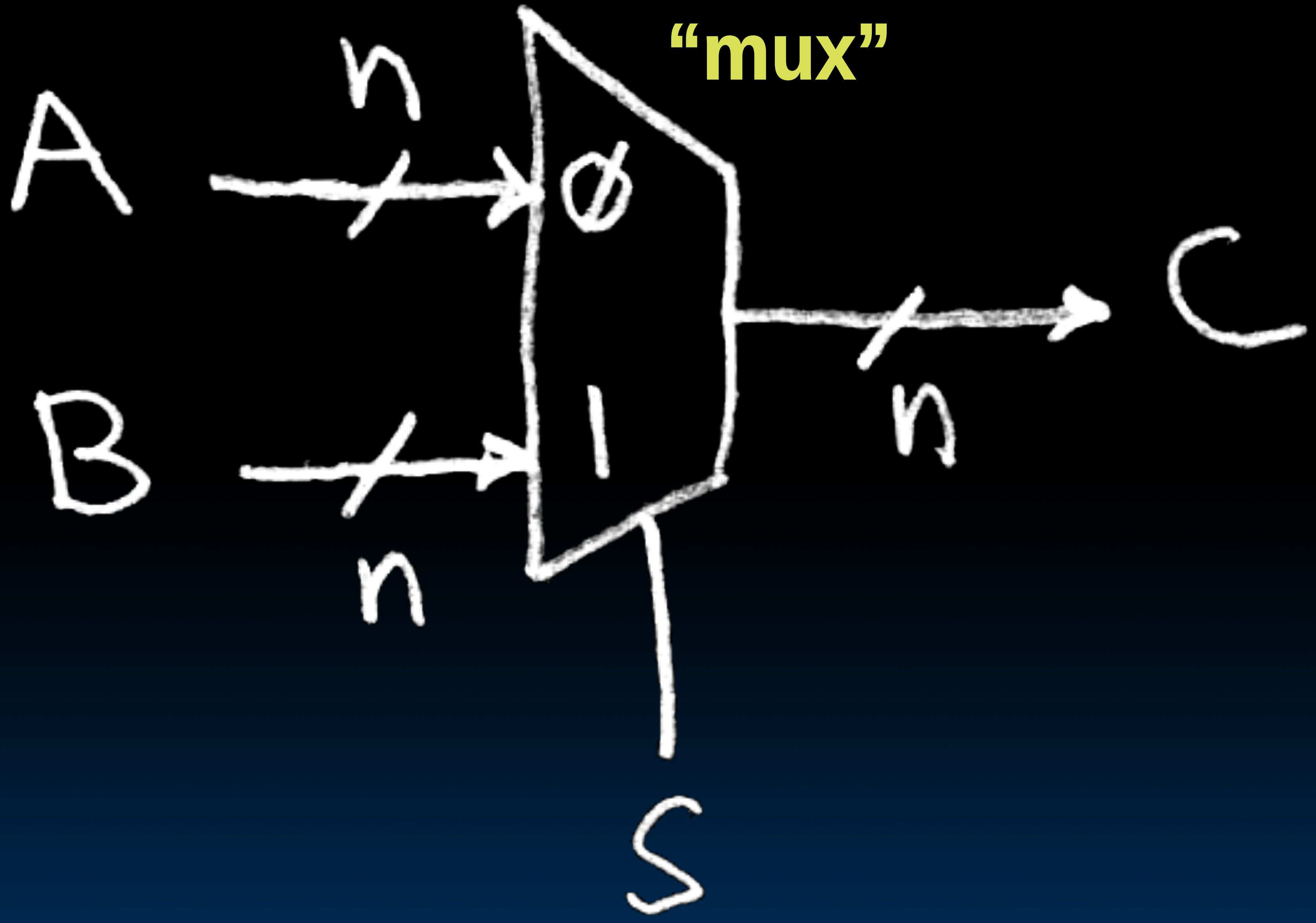
Berkeley
UNIVERSITY OF CALIFORNIA

cs61c.org

Data Multiplexors

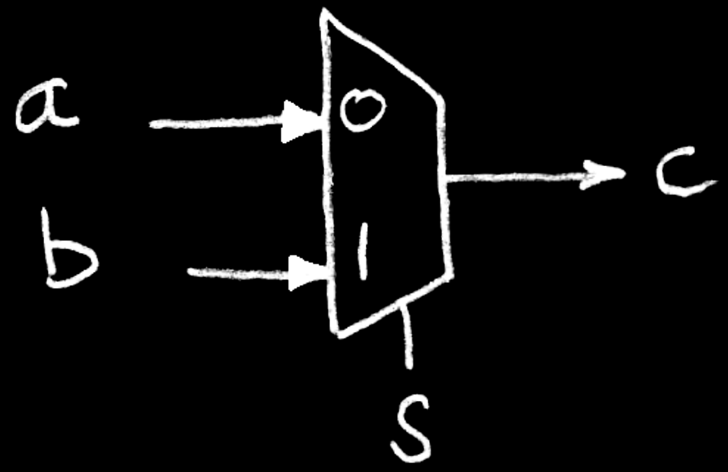
Data Multiplexor (here 2-to-1, n-bit-wide)

“mux”



Garcia, Nikolić

N instances of 1-bit-wide mux



How many rows in TT?

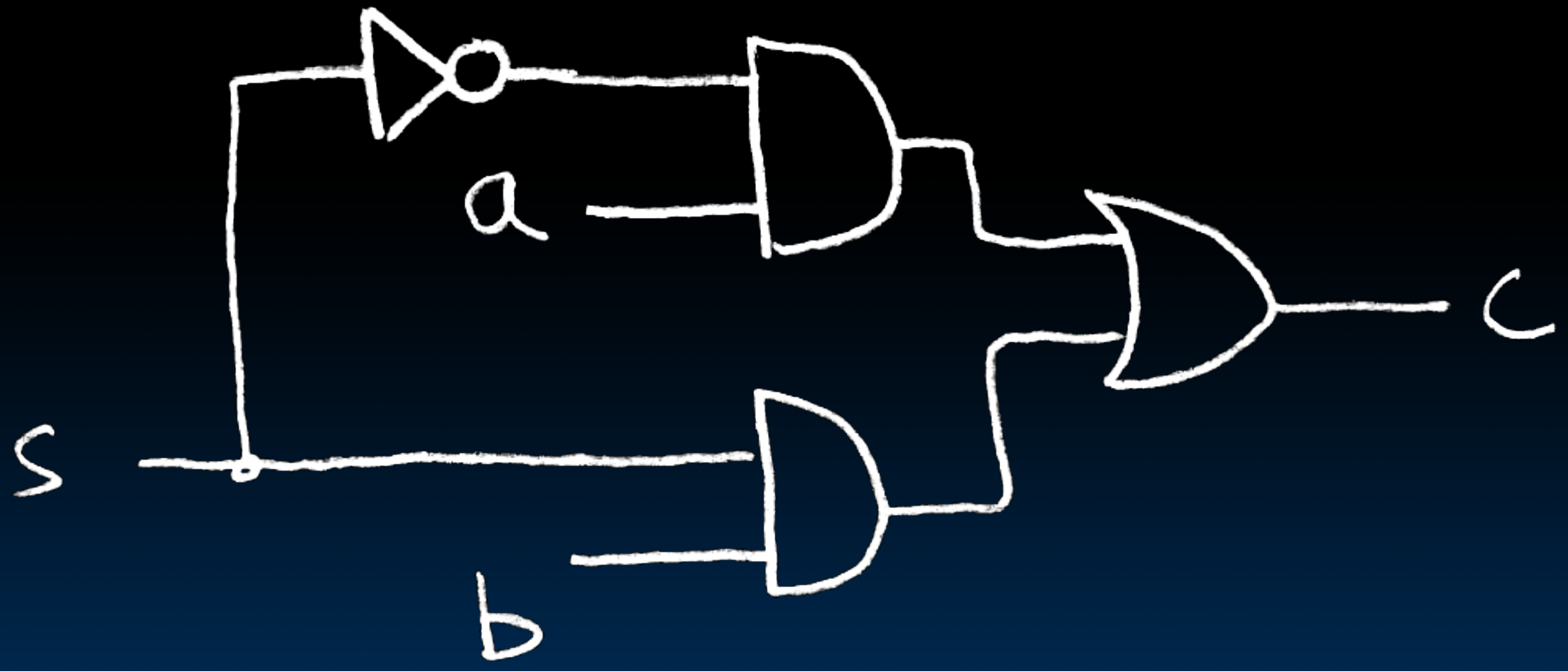
s	ab	c
0	00	0
0	01	0
0	10	1
0	11	1
1	00	0
1	01	1
1	10	0
1	11	1

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

s	c
0	a
1	b

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

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

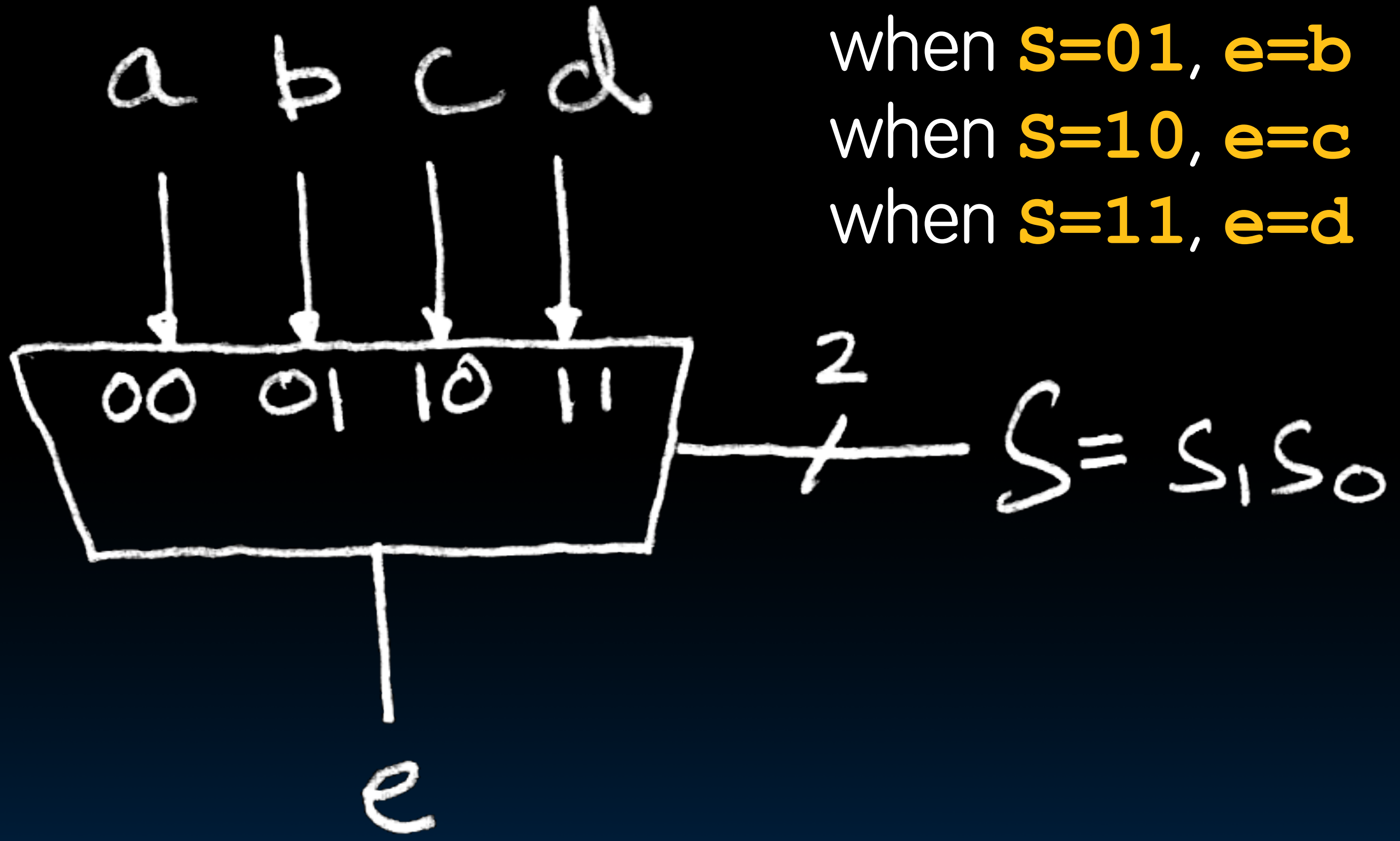


Garcia, Nikolić

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**



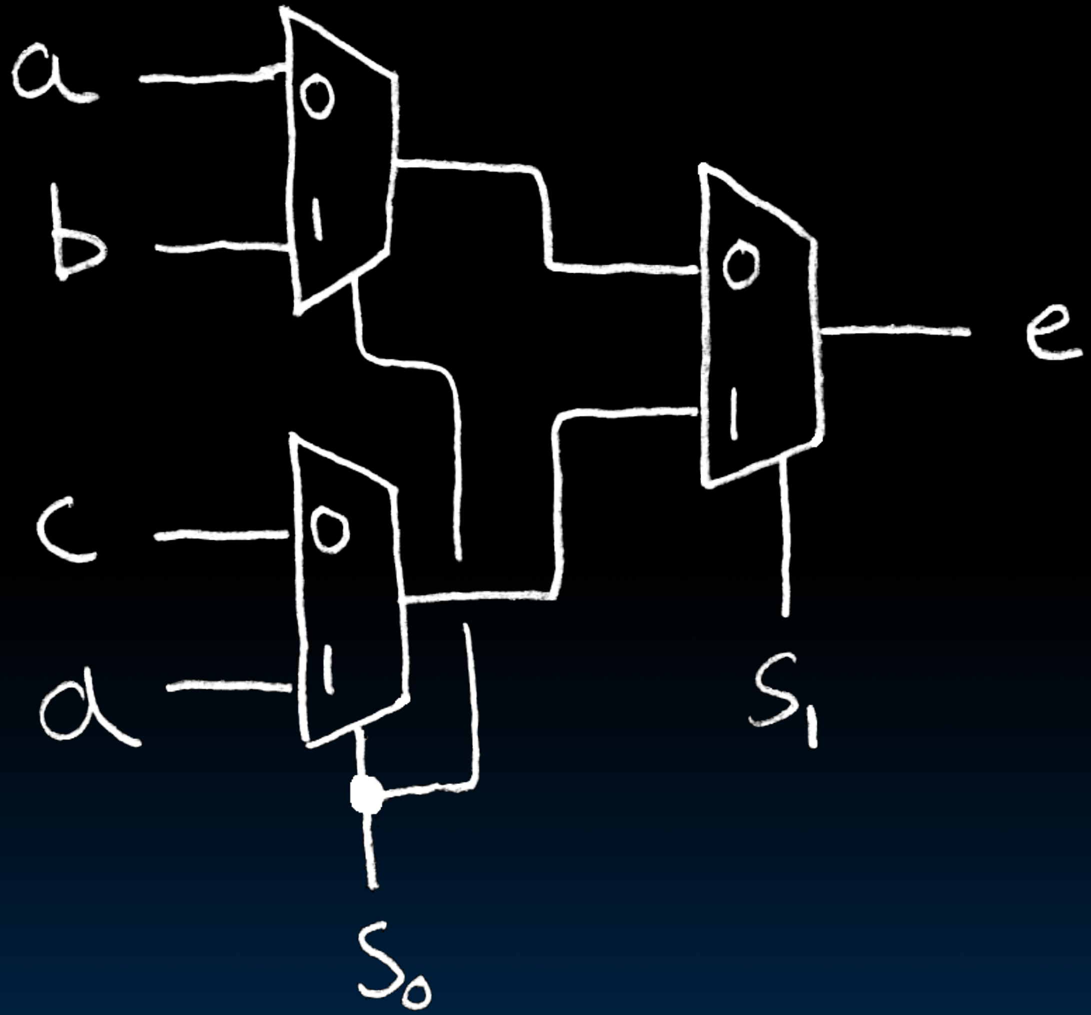
$s_1 s_0$	c
0 0	a
0 1	b
1 0	c
1 1	d

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

Berkeley
UNIVERSITY OF CALIFORNIA

Mux: is there any other way to do it?

Hint: NCAA tourney!

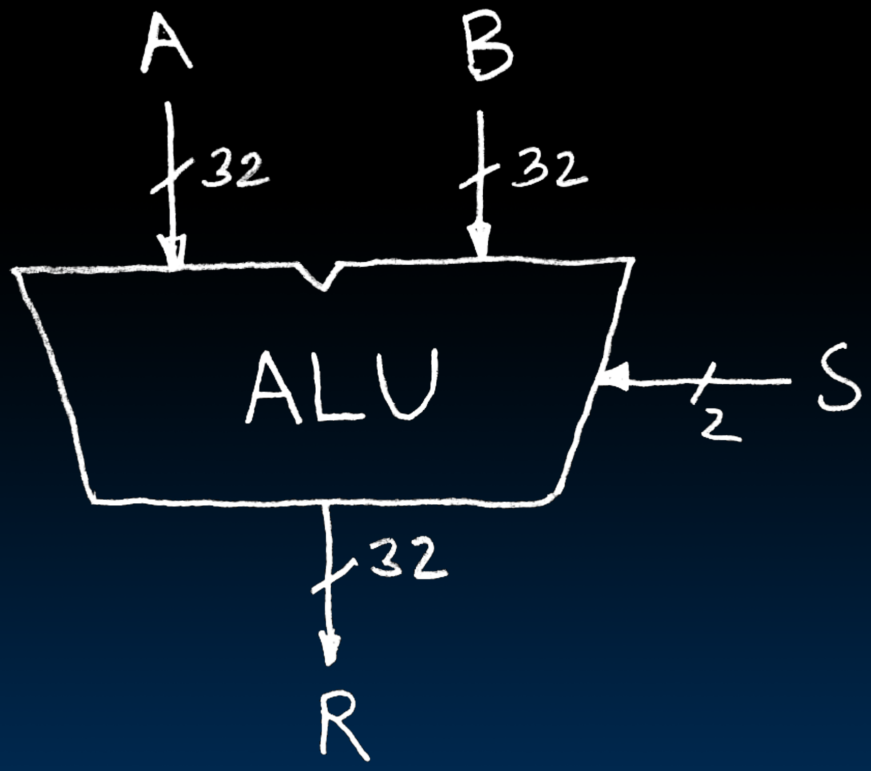


Ans: Hierarchically!

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 (|)



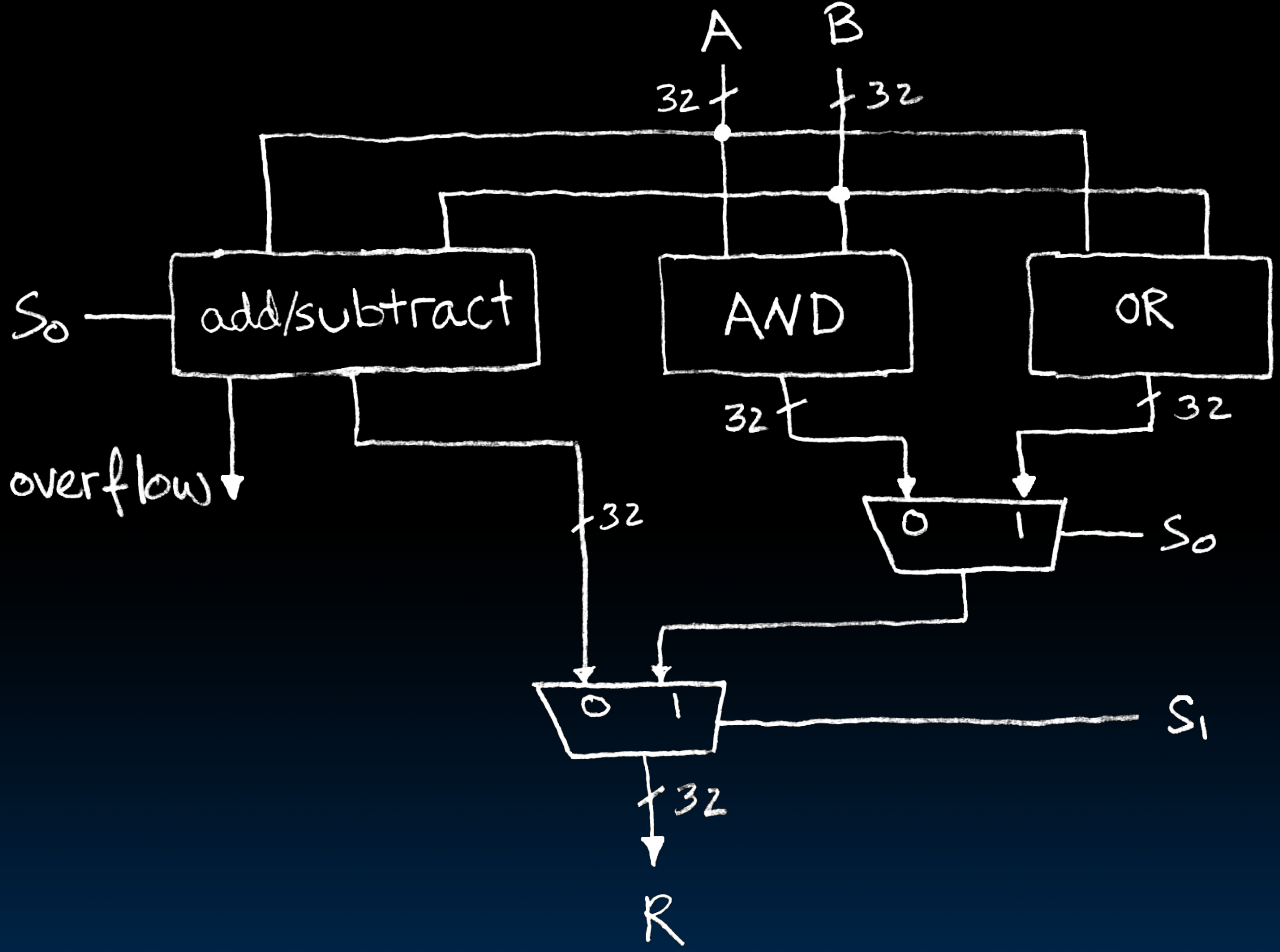
when S=00, R=A+B

when S=01, R=A-B

when S=10, R=A&B

when S=11, R=A|B

Our simple ALU



Garcia, Nikolić

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…

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

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

$$\begin{aligned} s_0 &= a_0 \text{ XOR } b_0 \\ c_1 &= a_0 \text{ AND } b_0 \end{aligned}$$

Berkeley
UNIVERSITY OF CALIFORNIA

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

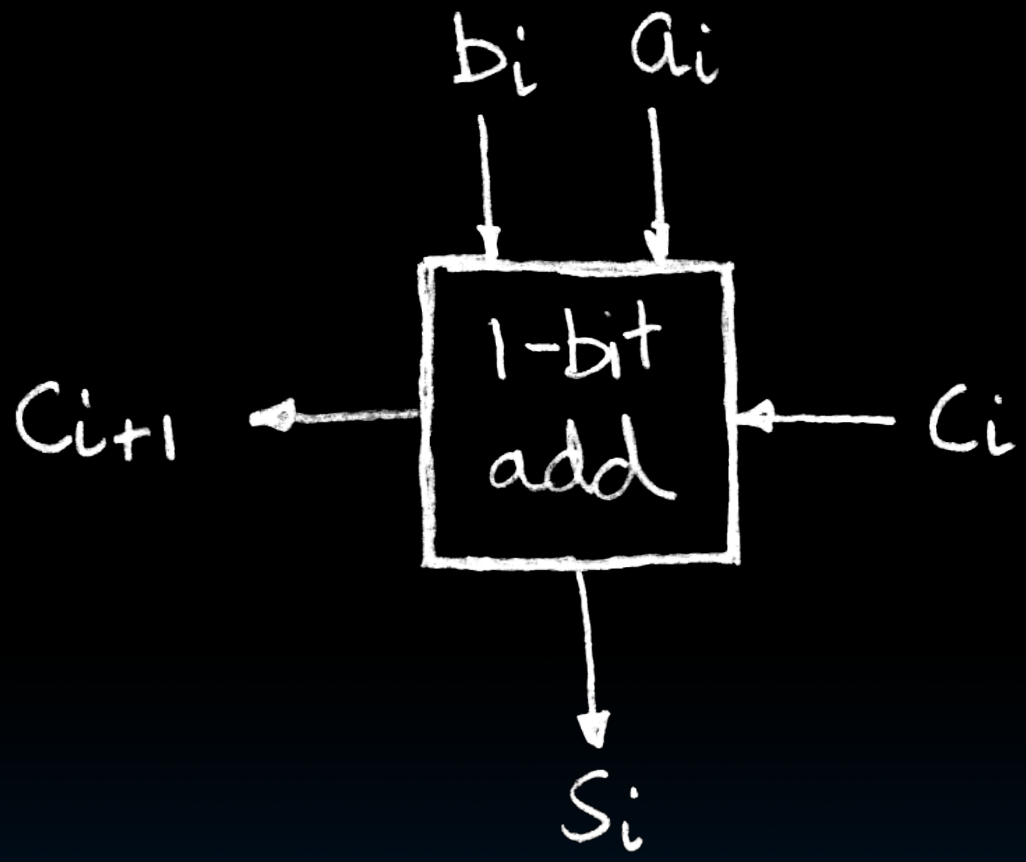
$$\begin{array}{cccc|c|c} & a_3 & a_2 & a_1 & a_0 \\ + & b_3 & b_2 & b_1 & b_0 \\ \hline & s_3 & s_2 & s_1 & s_0 \end{array}$$

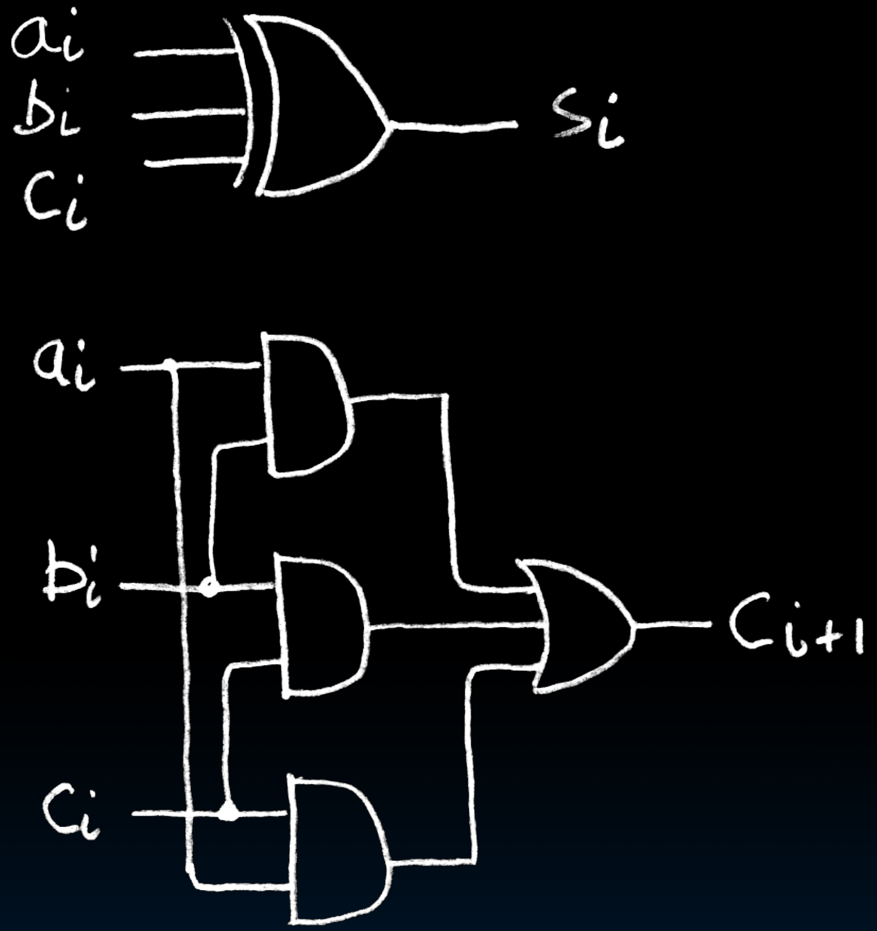
a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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

Berkeley
UNIVERSITY OF CALIFORNIA

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

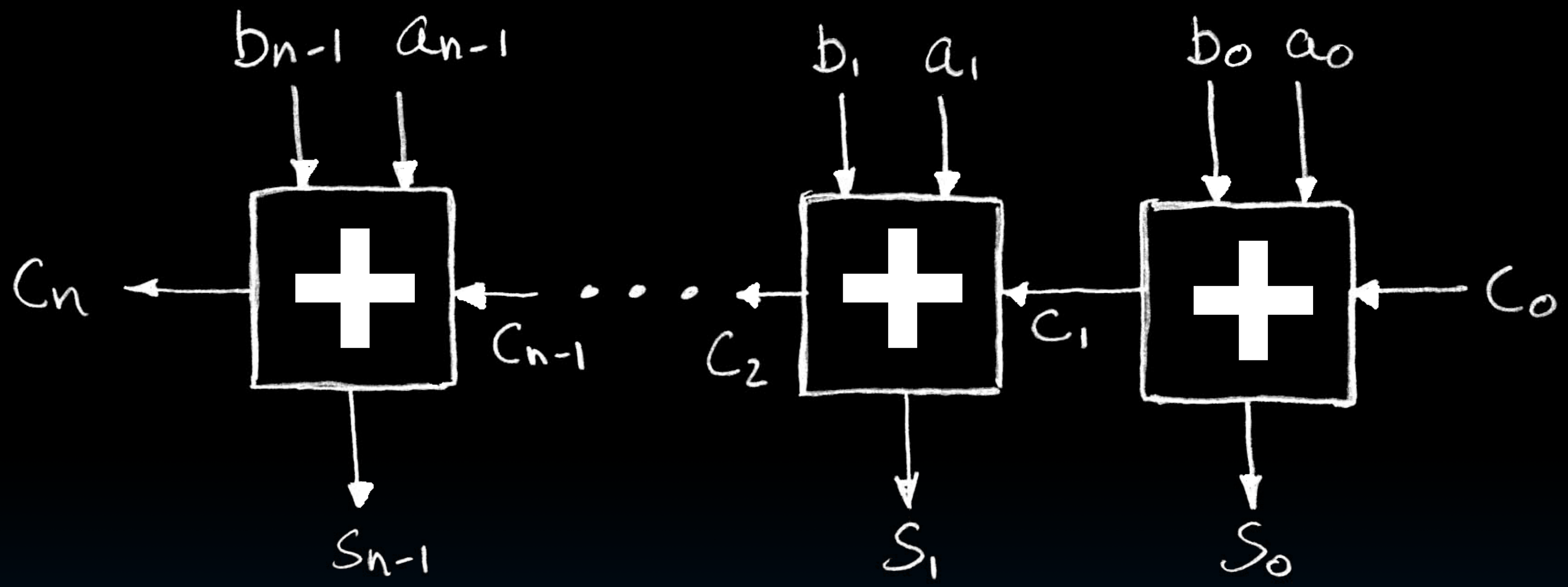




$$
\begin{aligned}
s_i &= \mathrm{XOR}(a_i, b_i, c_i) \\
c_{i+1} &= \mathrm{MAJ}(a_i, b_i, c_i) = a_i b_i + a_i c_i + b_i c_i
\end{aligned}
$$

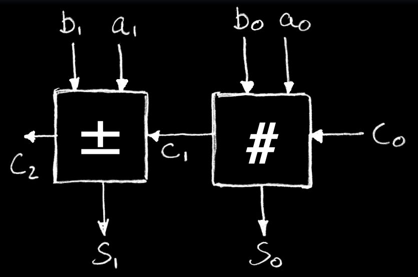
Berkeley UNIVERSITY OF CALIFORNIA

N 1-bit adders → 1 N-bit adder



What about overflow?
Overflow = c_n?

Sum of two 2-bit numbers…



$$\begin{array}{r} c_1 \quad \\ a_1 \; a_0 \\ +\; b_1 \; b_0 \\ \hline c_2 \; s_1 \; s_0 \end{array}$$

11 "-1"				1 11 +11 110
10 "-2"			10 +10 100	10 +11 101
01 "1"		1 01 +01 10	01 +10 11	1 01 +11 100
00 "0"	00 +00 00	00 +01 01	00 +10 10	00 +11 11
+	00 "0"	01 "1"	10 "-2"	11 "-1"

- **Let's add**
  - First unsigned
  - Then signed (Two's Complement)
    - When do the lowest 2 bits of sum <u>not</u> 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?

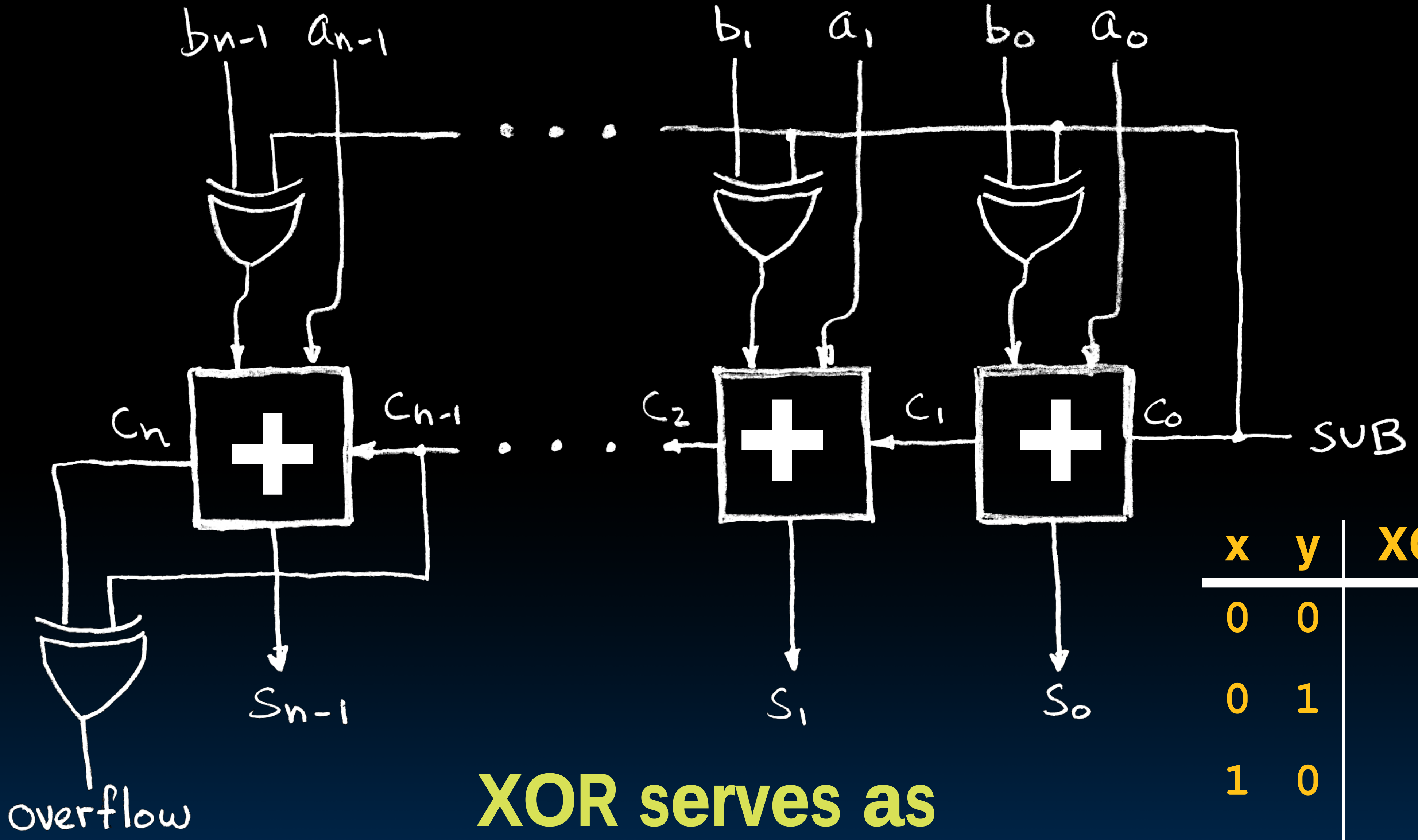
$$\text{overflow} = c_n \text{ XOR } c_{n\text{-}1}$$

☑

Berkeley UNIVERSITY OF CALIFORNIA

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

☑