inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

Lecture 23 – Combinational Logic Blocks

2004-10-22

Lecturer PSOE Dan Garcia

www.cs.berkeley.edu/~ddgarcia

Age of the Machines? ⇒

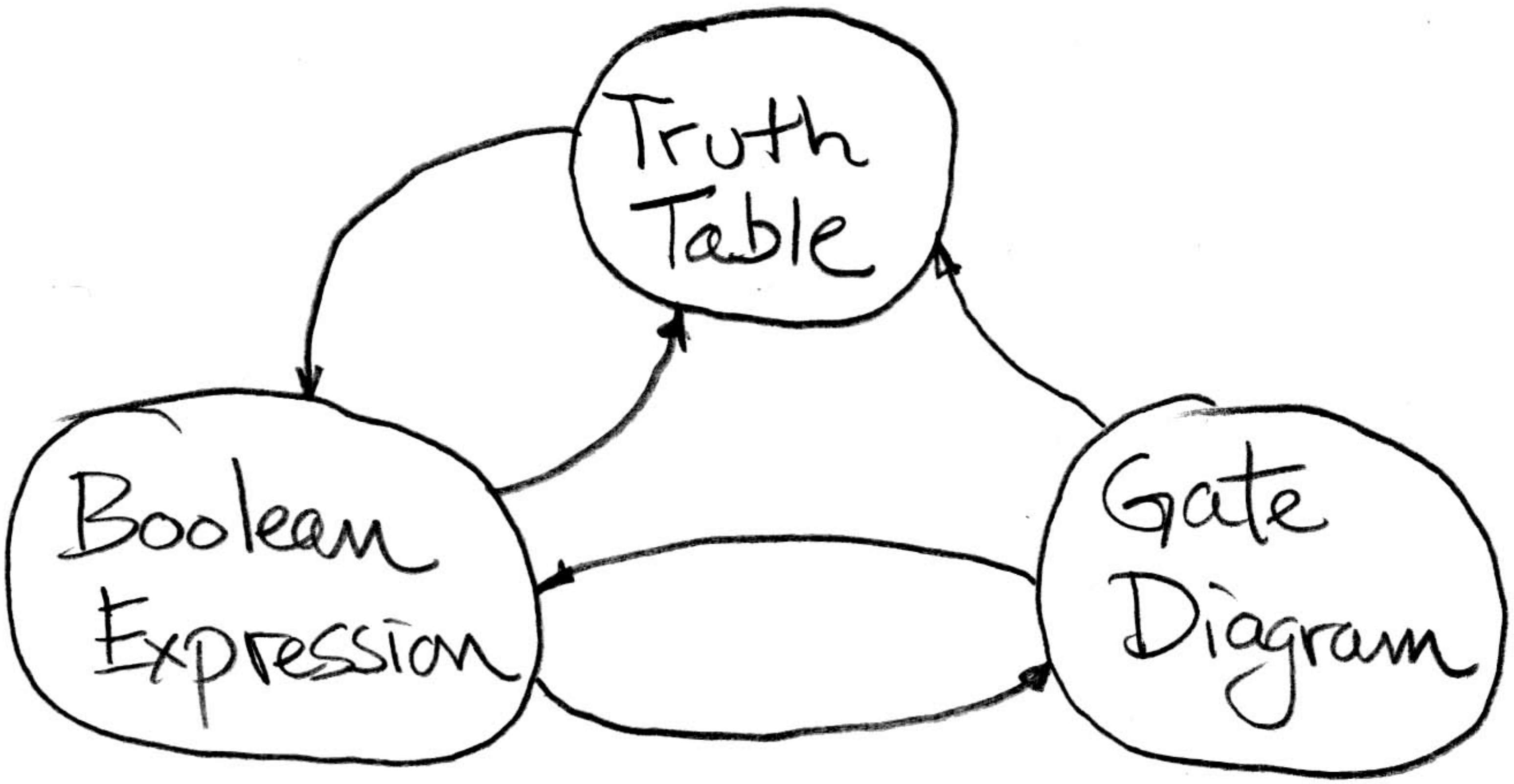
irobot.com

"The UN's annual World Robotics Survey for 2004 predicts that there will be a seven-fold surge in household robots by the end of 2007. Robots that mow your lawn, vacuum, wash windows, clean swimming pools, & entertainment robots such as the Aibo are vying for a place in our homes."

slashdot.org/article.pl?sid=04/10/21/0214230&tid=126&tid=216

Review

- **Use this table and techniques we learned to transform from 1 to another**



Peer Instruction Correction

A. $(a+b) \cdot (\overline{a}+b) =?= b$

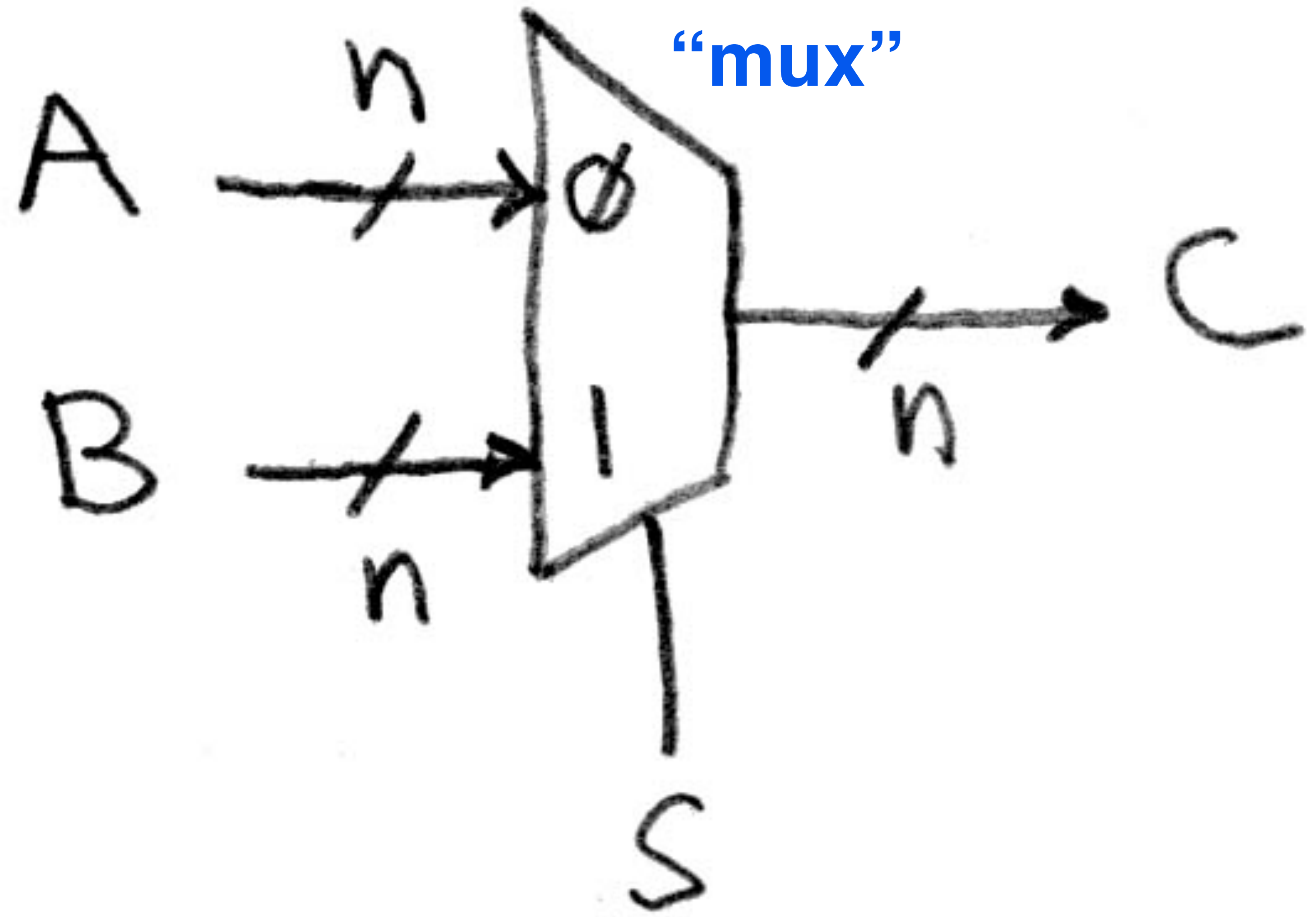
$(a+b)\cdot(\overline{a}+b)$	
$a\overline{a}+ab+b\overline{a}+bb$	*distribution*
$0+b(a+\overline{a})+b$	*complimentarity, commutativity, distribution, idempotent*
$b(1)+b$	*identity, complimentarity*
$b+b$	*identity*
b	*idempotent*

TRUE

Today

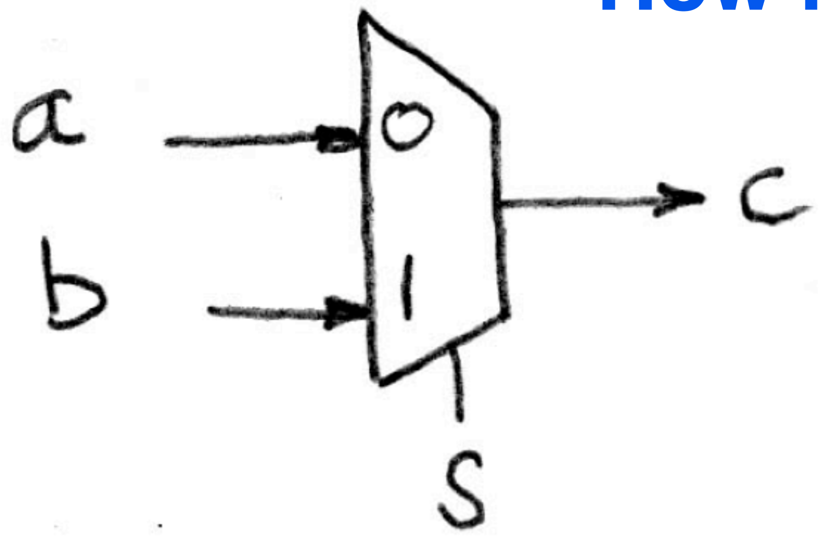
- Data Multiplexors
- Arithmetic and Logic Unit
- Adder/Subtractor

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



N instances of 1-bit-wide mux

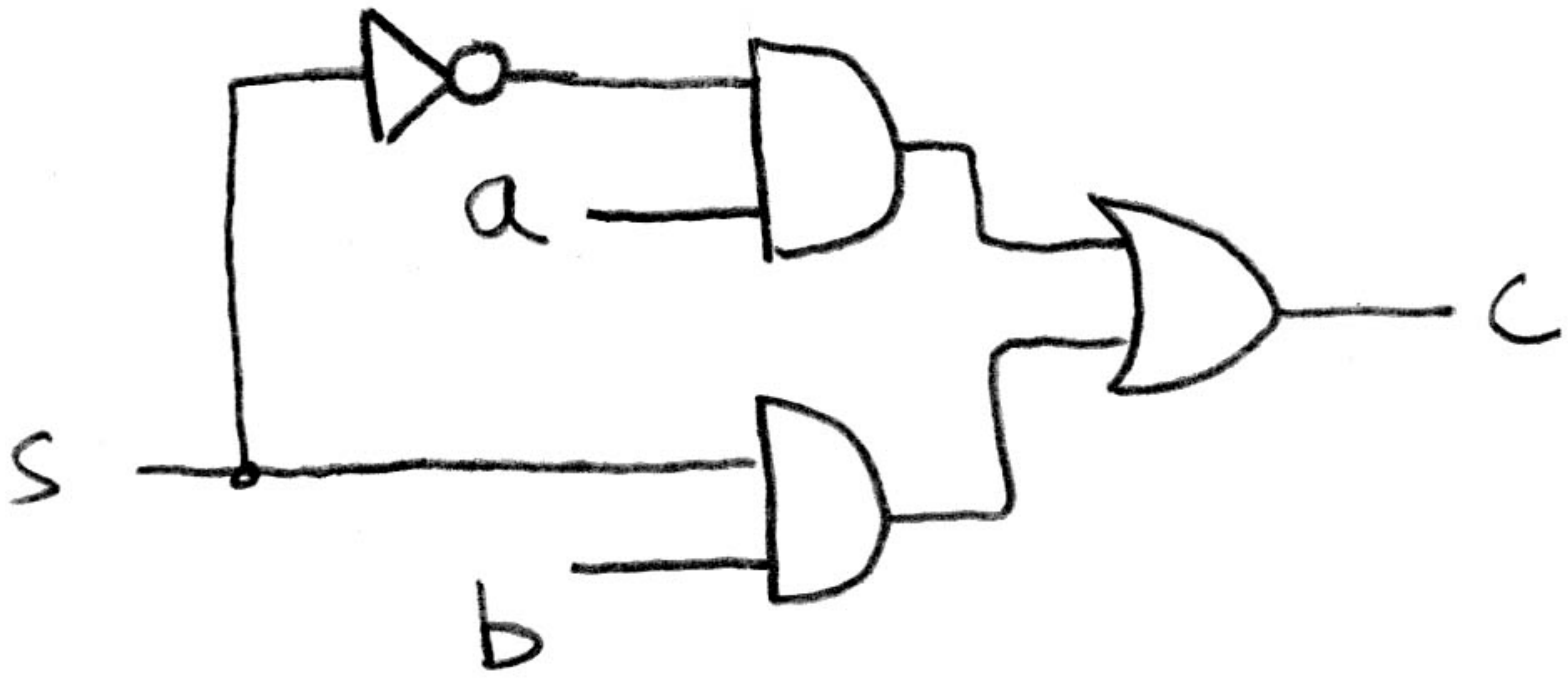
How many rows in TT?



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

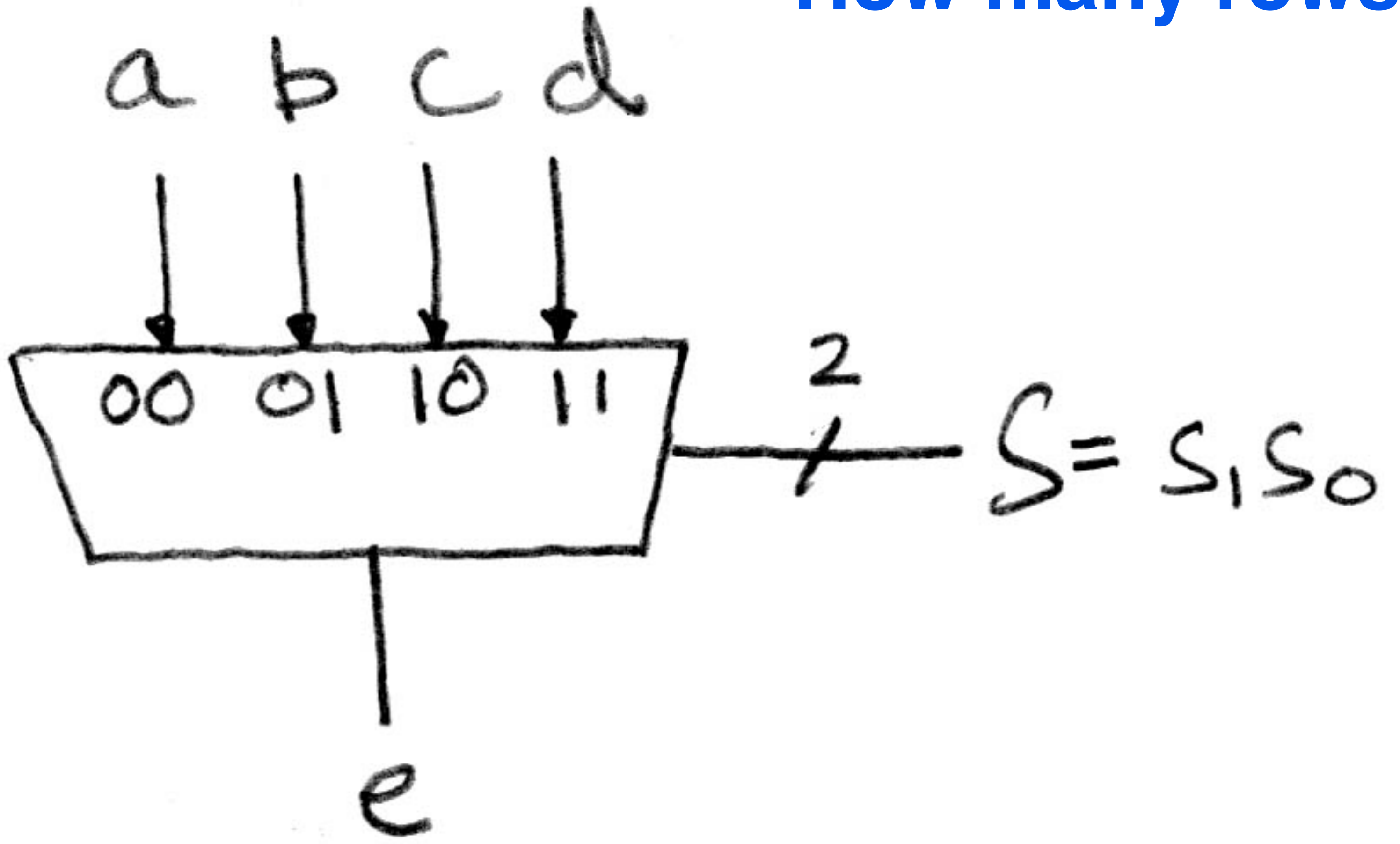
How do we build a 1-bit-wide mux?

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



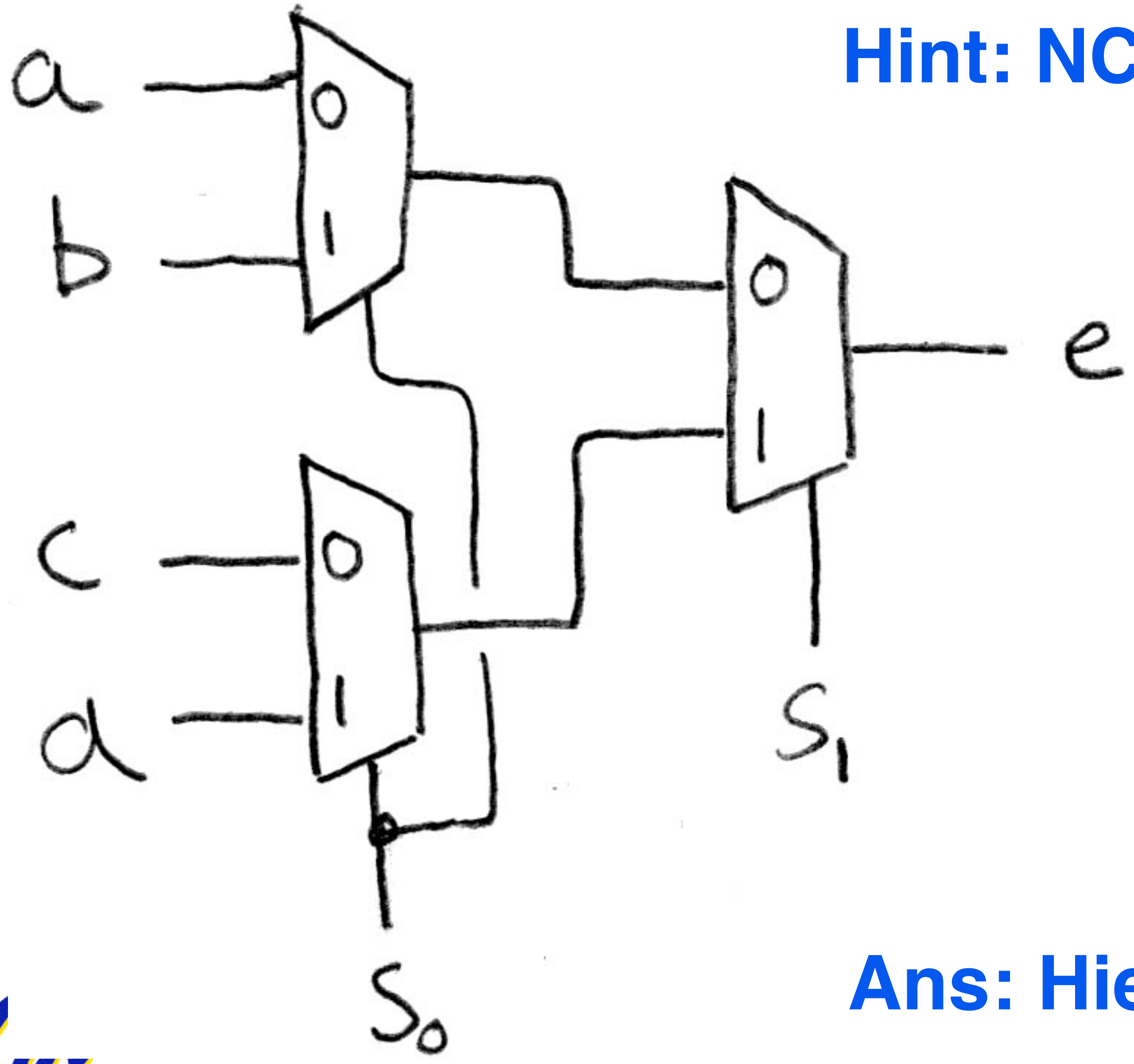
4-to-1 Multiplexor?

How many rows in TT?



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

Is there any other way to do it?



Hint: NCAA tourney!

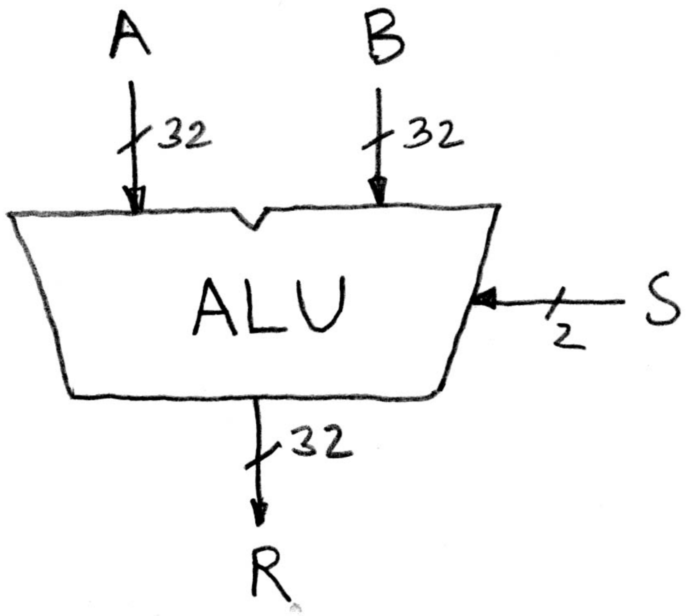
Ans: Hierarchically!

Administrivia : Midterm…

- You spoke and we heard
  - The final can be in pen OR pencil
    - We should have given the choice of pen or pencil, and if you choose pencil, no regrade
  - More scratch space…will be there (trees be damned)
- Want a regrade?
  - Return your exam and a stapled paragraph explaining which question(s) needed regrading AND WHY and we'll take a look at the next TA mtg
  - Your grade MAY go down, no complaints

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



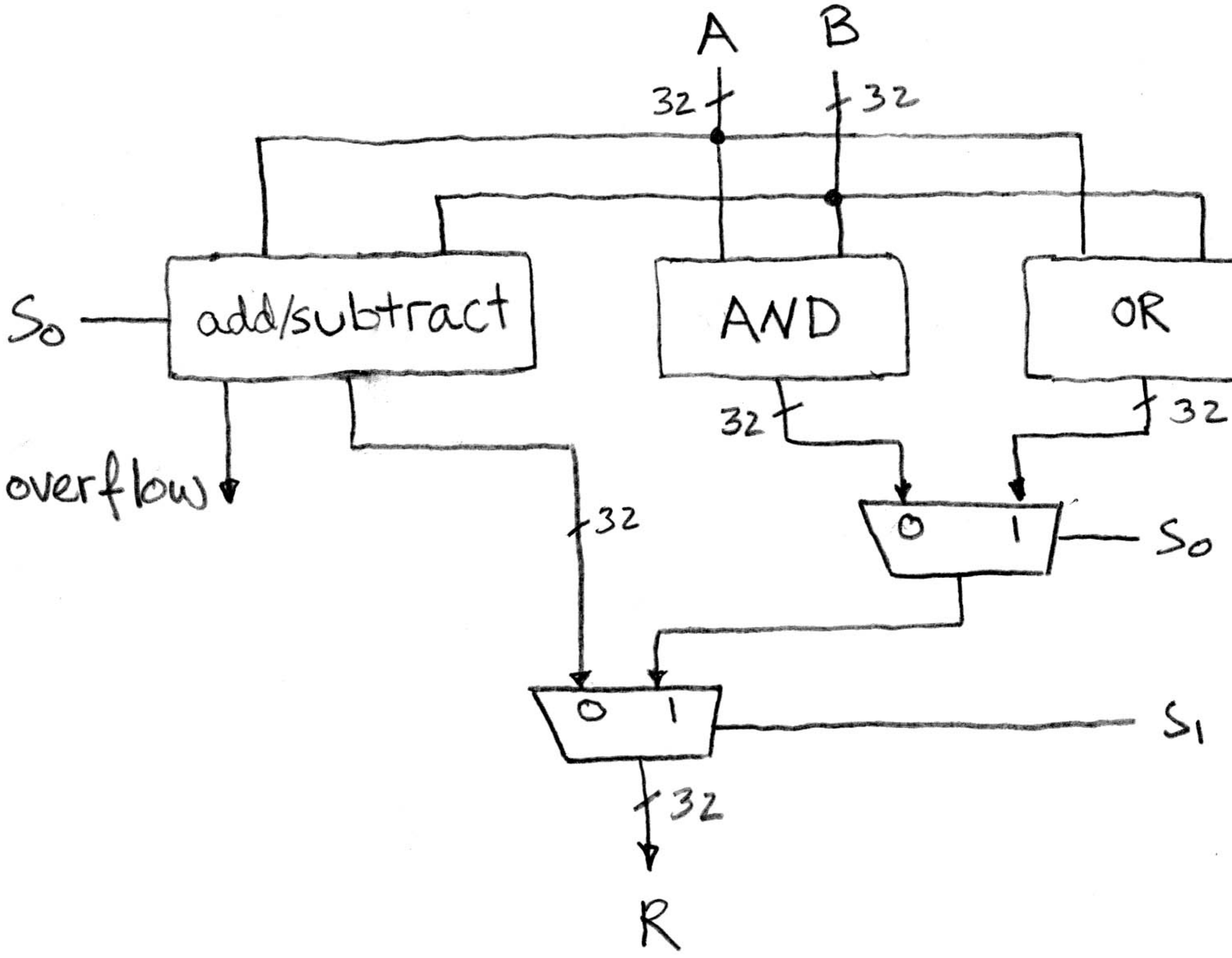
when S=00, R=A+B

when S=01, R=A-B

when S=10, R=A AND B

when S=11, R=A OR B

Our simple ALU



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/Subtracter – One-bit adder LSB…

$$\begin{array}{cccc|c|} \cline{5-5} & a_3 & a_2 & a_1 & a_0 \\ + & b_3 & b_2 & b_1 & b_0 \\ \hline & s_3 & s_2 & s_1 & s_0 \\ \cline{5-5} \end{array}$$

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

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

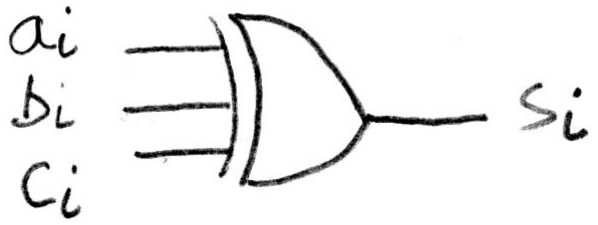
Adder/Subtracter – One-bit adder (1/2)…

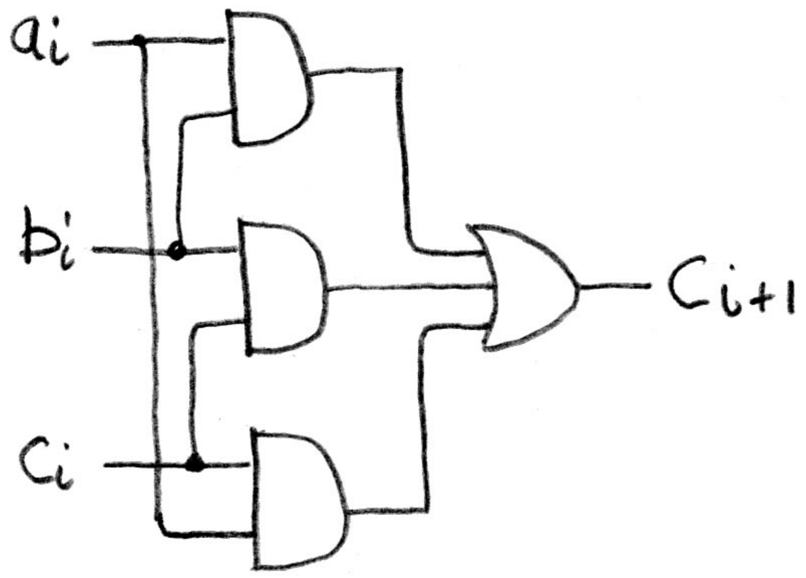
$$\begin{array}{ccc|c|c}
 & a_3 & a_2 & a_1 & a_0 \\
+ & b_3 & b_2 & b_1 & b_0 \\
\hline
 & s_3 & s_2 & s_1 & s_0
\end{array}$$

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

$$\begin{aligned} s_i &= \\ c_{i+1} &= \end{aligned}$$

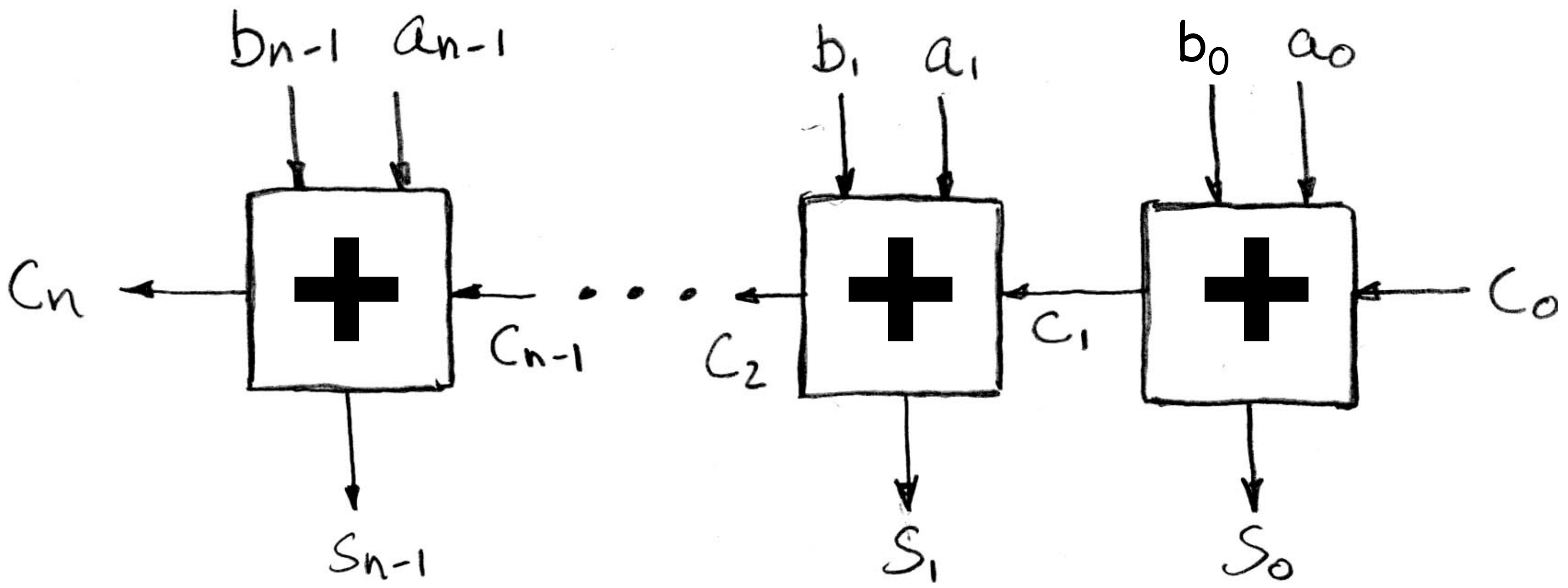
Adder/Subtracter – One-bit adder (2/2)…





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

N 1-bit adders ⇒ 1 N-bit adder

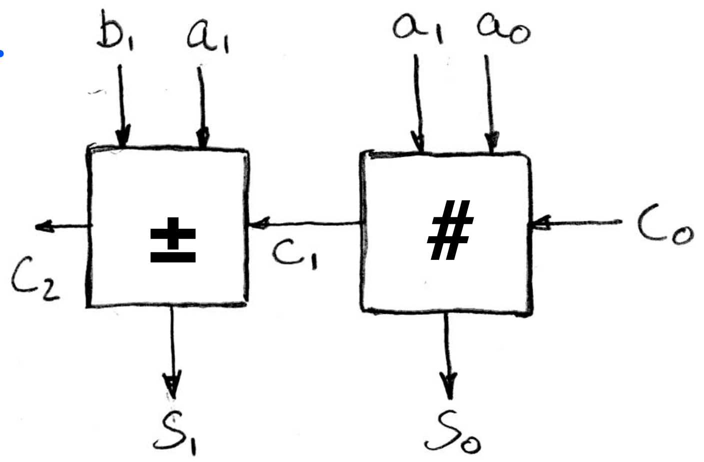


What about overflow?

Overflow = c_n?

What about overflow?

- Consider a 2-bit signed # & overflow:
  - `10 = -2 + -2 or -1`
  - `11 = -1 + -2 only`
  - `00 =  0 NOTHING!`
  - `01 =  1 + 1 only`



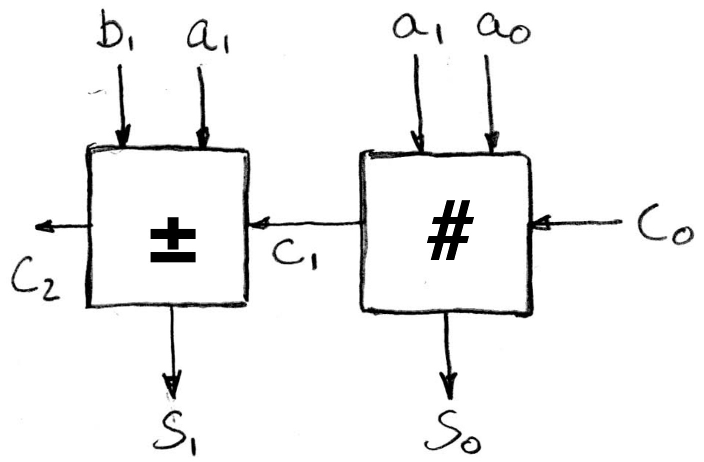
- Highest adder
  - C_1 = Carry-in = C_{in}, C_2 = Carry-out = C_{out}
  - No C_{out} or C_{in} $\Rightarrow$ NO overflow!
  - C_{in}, and C_{out} $\Rightarrow$ NO overflow!
  - C_{in}, but no C_{out} $\Rightarrow$ A,B both > 0, overflow!
  - C_{out}, but no C_{in} $\Rightarrow$ A,B both < 0, overflow!

What op?

What about overflow?

- Consider a 2-bit signed # & overflow:

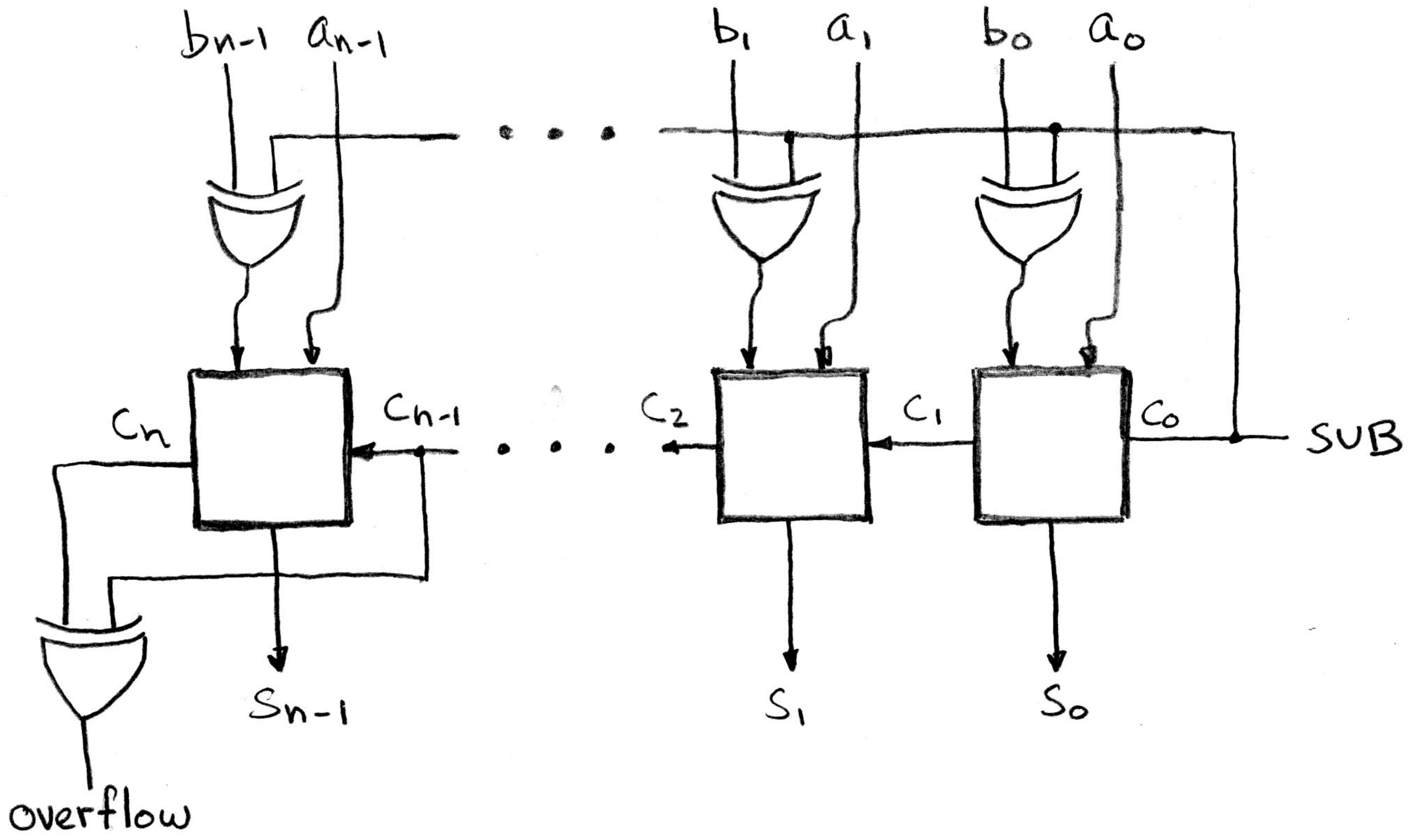
```
10 = -2
11 = -1
00 =  0
01 =  1
```



- Overflows when…
  - C_{in}, but no C_{out} ⇒ A,B both > 0, overflow!
  - C_{out}, but no C_{in} ⇒ A,B both < 0, overflow!

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

Extremely Clever Subtractor



Peer Instruction

A. Truth table for mux with 4-bits of signals is 2^4 rows long

B. We could cascade N 1-bit shifters to make 1 N-bit shifter for sll, srl

C. If 1-bit adder delay is T, the N-bit adder delay would also be T

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TTF
8:	TTT

Peer Instruction Answer

"And In conclusion…"

- Use muxes to select among input
  - S input bits selects 2S inputs
  - Each input can be n-bits wide, indep of S
- 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