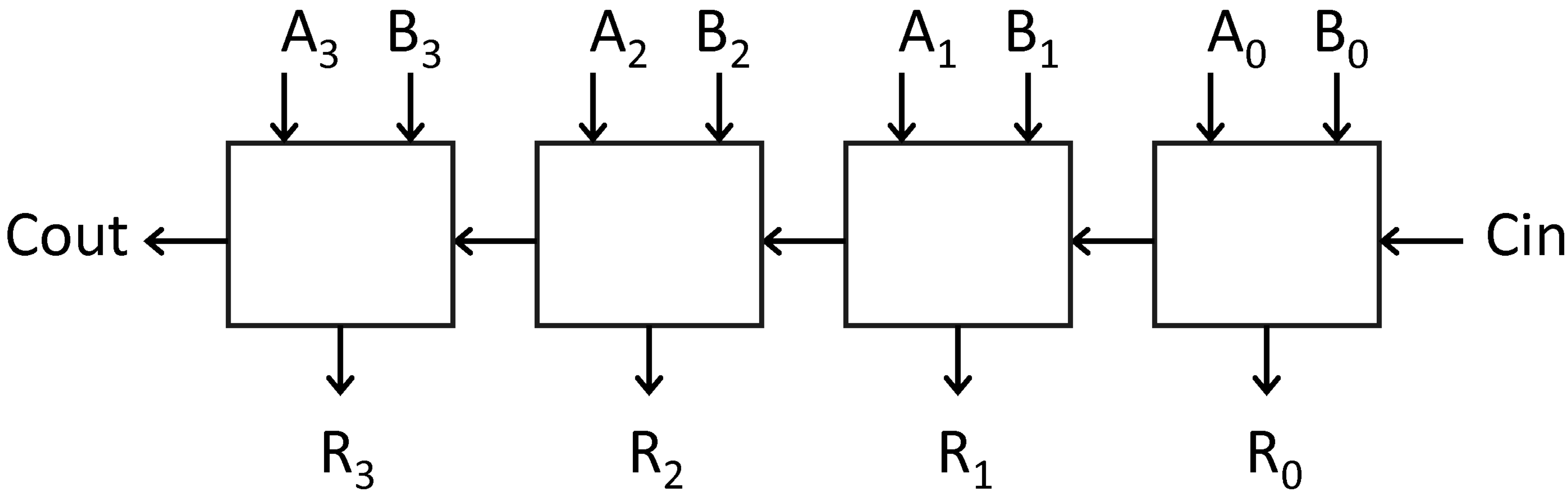
Arithmetic

Kevin Walsh
CS 3410, Spring 2010
Computer Science
Cornell University

See: P&H Chapter 3.1-3, C.5-6



- Adds two 4-bit numbers, along with carry-in
- Computes 4-bit result and carry out

Addition with negatives:

- pos + pos → add magnitudes, result positive
- neg + neg → add magnitudes, result negative
- pos + neg → subtract smaller magnitude
  keep sign of bigger magnitude

pos + pos → add magnitudes
result pos

neg + neg ⇒ add mag.
res. neg

pos + neg ⇒ larger minus smaller
sign of larger

First Attempt: Sign/Magnitude Representation

- 1 bit for sign (0=positive, 1=negative)
- N-1 bits for magnitude

0111 = +7

1111 = -7

Better: Two's Complement Representation

- Leading 1's for negative numbers
- To negate any number:
  - complement *all* the bits
  - then add 1 , ignore carry out.

+6 = 0110
~6 1001
+1
-6 = 1010

+20 = 00010100
11101011
+1
-20 = 11101100

Non-negatives (as usual):	Negatives (two's complement: flip then add 1):	
+0 = 0000	~0 = 1111	-0 = 0000
+1 = 0001	~1 = 1110	-1 = 1111
+2 = 0010	~2 = 1101	-2 = 1110
+3 = 0011	~3 = 1100	-3 = 1101
+4 = 0100	~4 = 1011	-4 = 1100
+5 = 0101	~5 = 1010	-5 = 1011
+6 = 0110	~3 = 1001	-6 = 1010
+7 = 0111	~7 = 1000	-7 = 1001
~~+8 = 1000~~	~8 = 0111	-8 = 1000

Signed two's complement

- Negative numbers have leading 1's
- zero is unique: +0 = - 0
- wraps from largest positive to largest negative

N bits can be used to represent

- unsigned: $0 \ldots (2^N) - 1$
  - eg: 8 bits ⇒ $0 \ldots 2^8 - 1 = 255$
- signed (two's complement):
  - ex: 8 bits ⇒ $-128 \ldots 0 \ldots +127$

$$-\left(\frac{2^N}{2}\right) \ldots \left(\frac{2^N}{2}\right) - 1$$

Extending to larger size

0 1 1 0

$-1 = 1\,1\,1\,1$

$\underbrace{0\,0\,0\,0}\ \underbrace{0\,1\,1\,0}$

$\underbrace{1\,1\,1\,1}\ \underbrace{1\,1\,1\,1}$

Truncate to smaller size

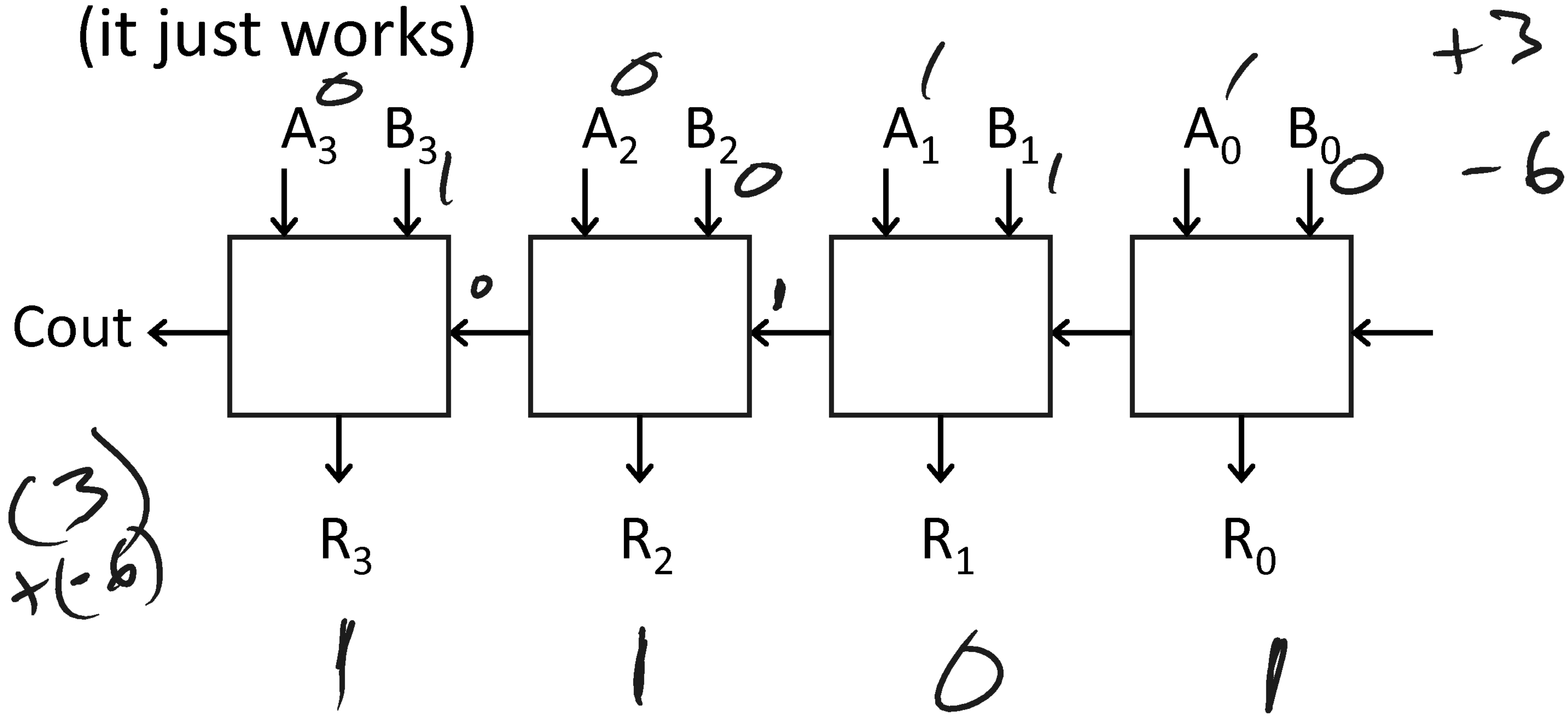
Drop leading sign bits, as long as sign doesn't change

0 0 0 0 1 1 1 1 = 15

0 1 1 1 = 7

Addition with two’s complement signed numbers

- Perform addition as usual, regardless of sign (it just works)



= -3

How does that work?

ten's complement

$$\boxed{-154}$$
$$+283$$

999
-154

1

283

845

1

283

$\boxed{846}$

283

Overflow

- adding a negative and a positive?

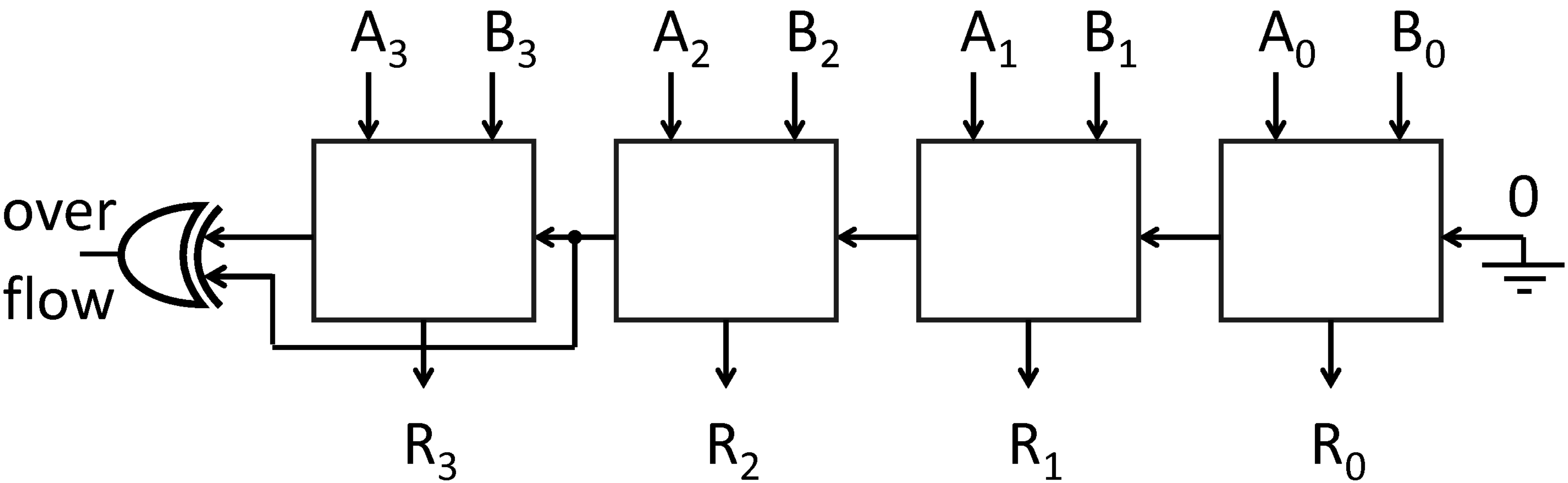
can't happen

- adding two positives?
- adding two negatives?

Rule of thumb:

Overflow happened iff
carry into msb != carry out of msb

Two's Complement Adder with overflow detection



0111
+ 1
1000

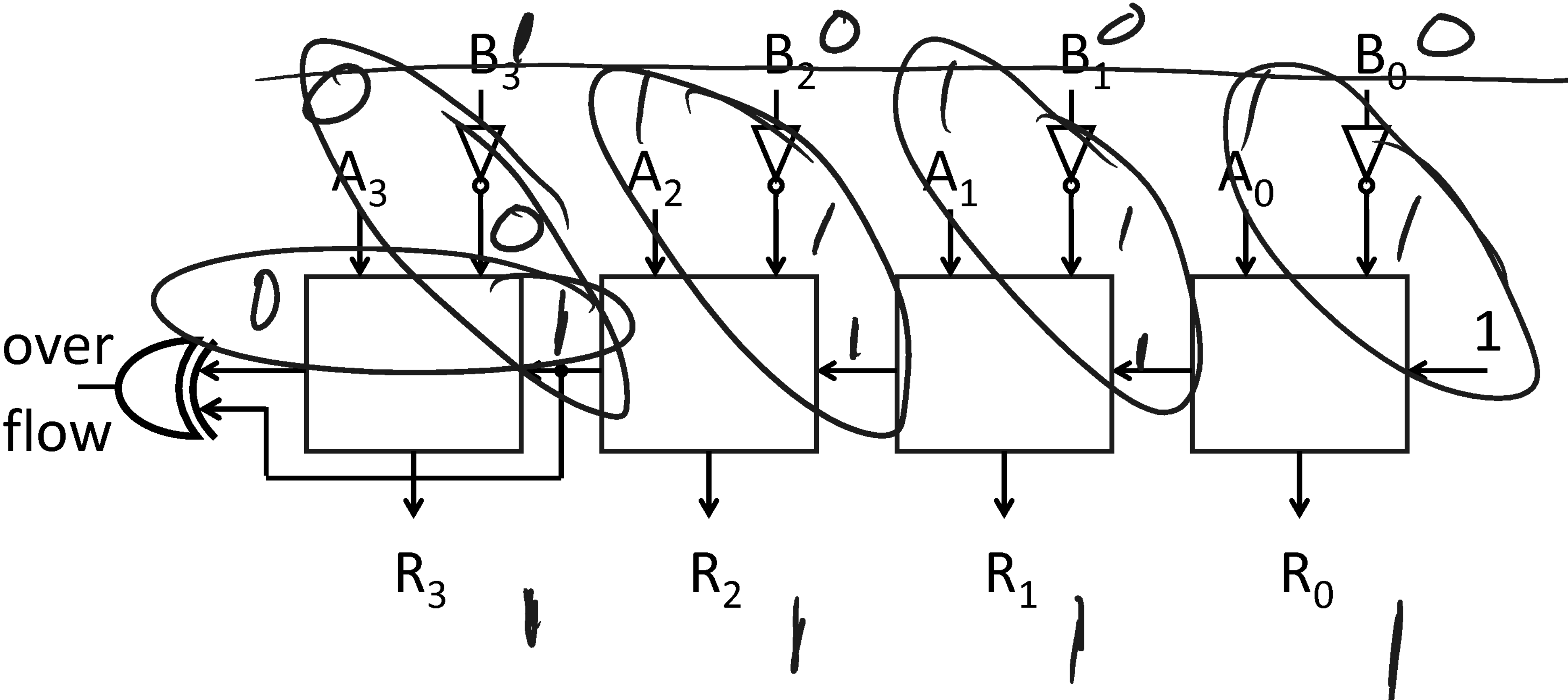
Two's Complement Subtraction

$B = -8 = \boxed{1000}$

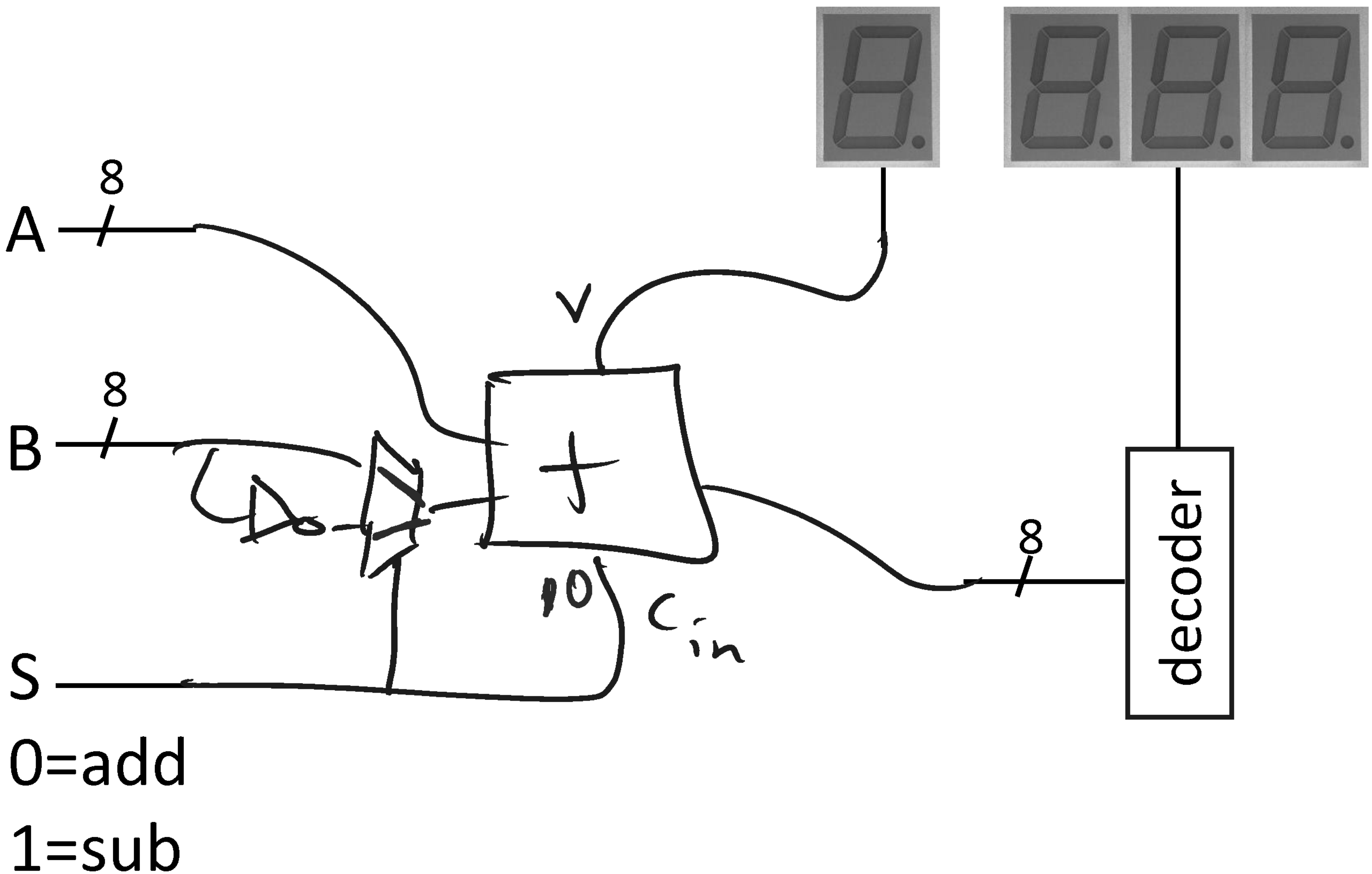
Lazy approach

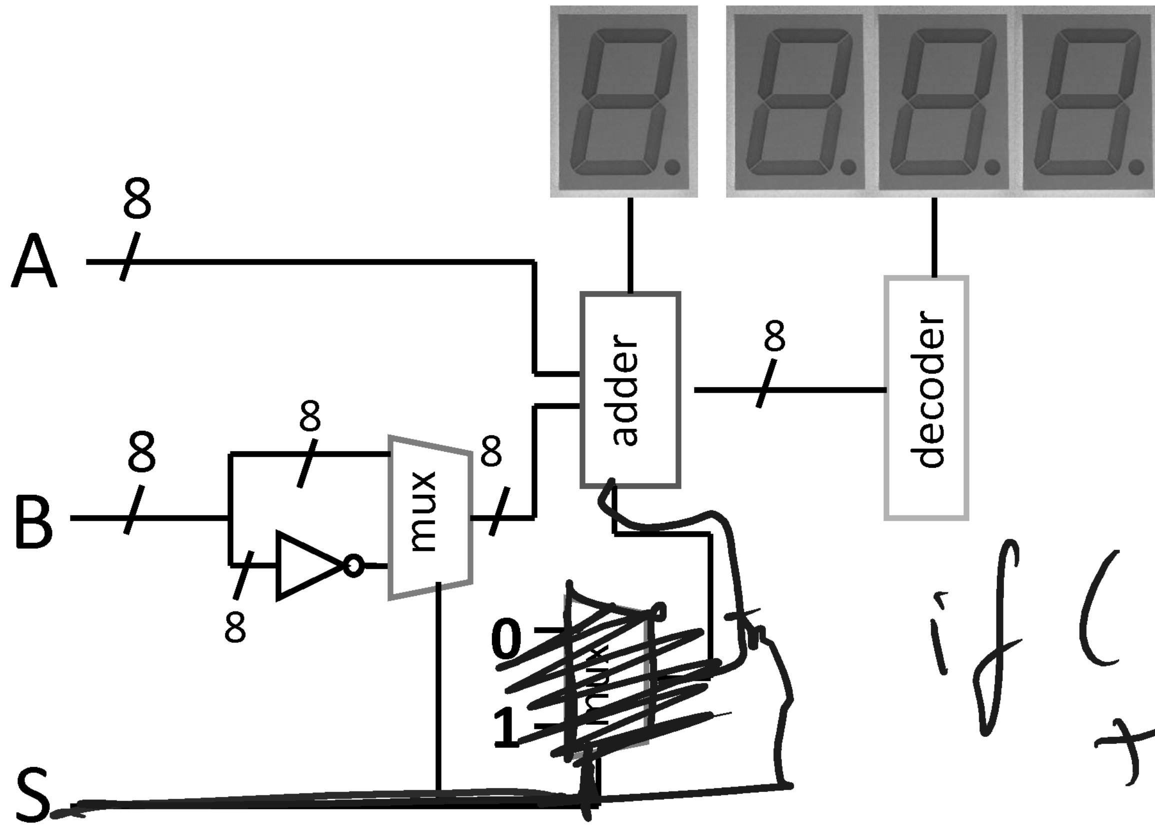
$$A - B = A + (-B) = A + (\overline{B} + 1)$$

$(7) - (-8)$



Q: What if (-B) overflows?

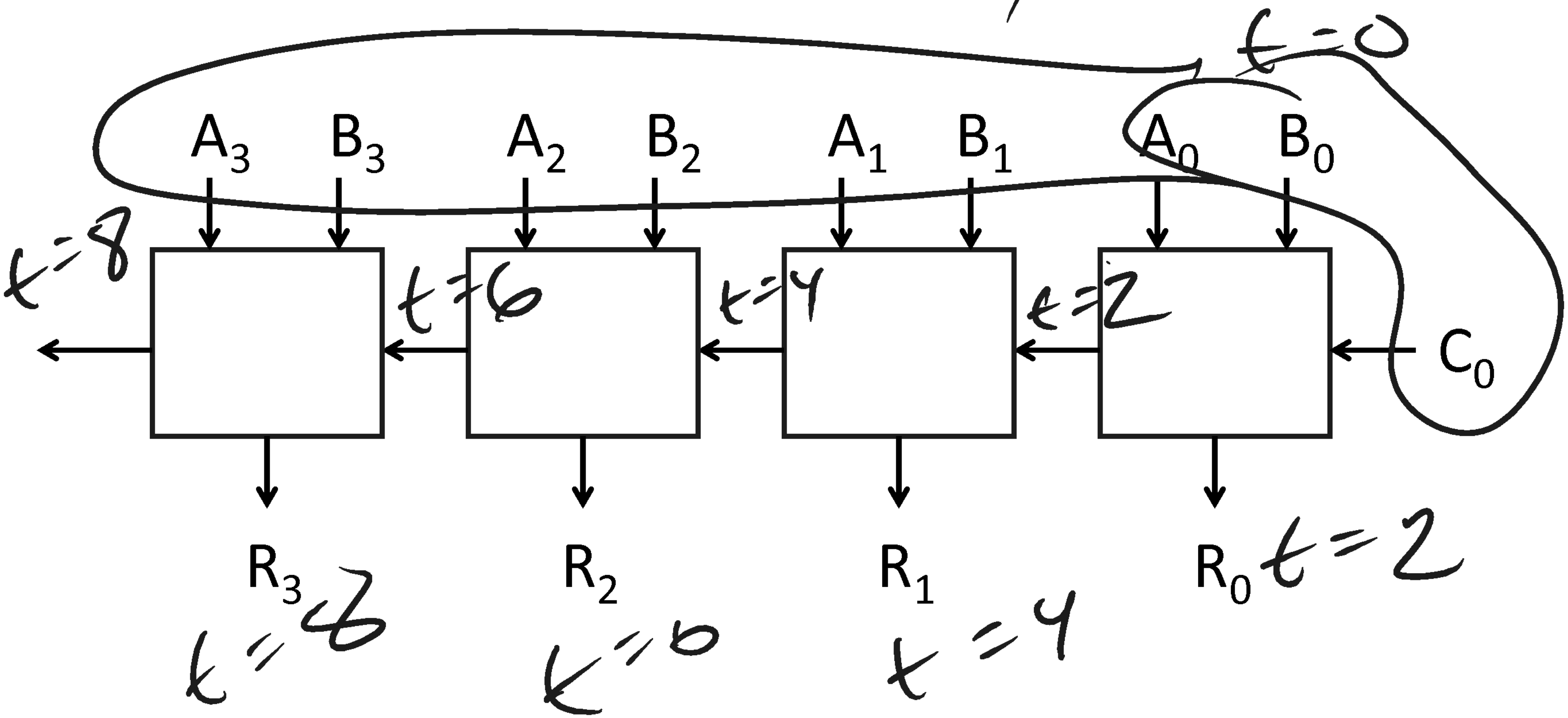

A
8
B
8
S
0=add
1=sub
+
V
0
C_{in}
8
decoder



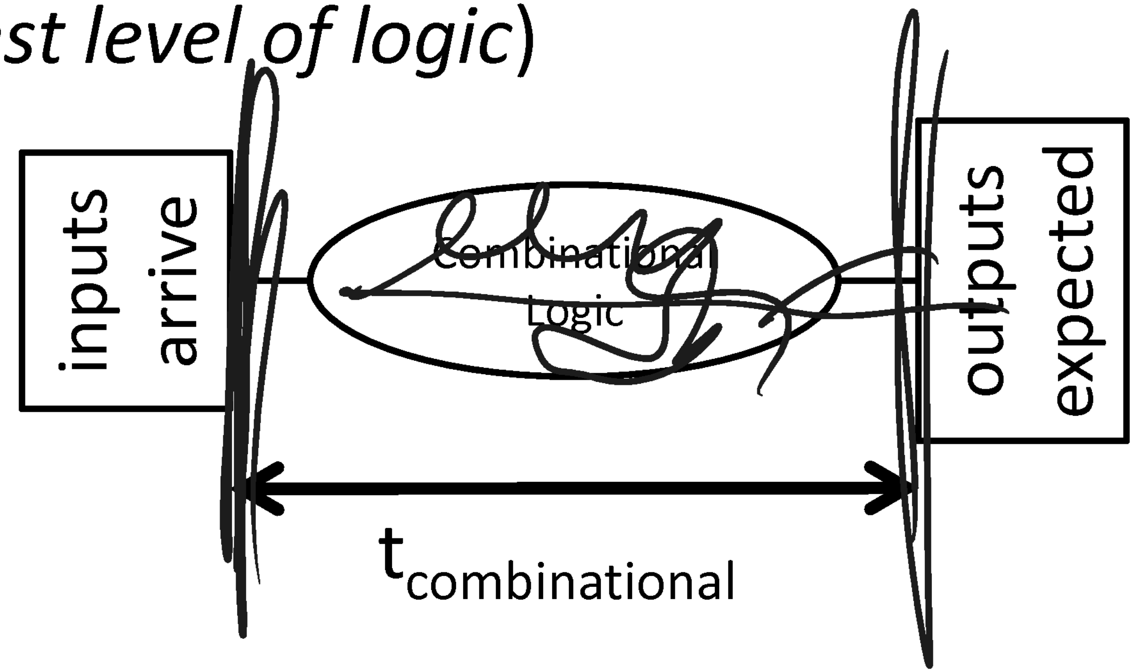
if (b ~~true~~)
then ~

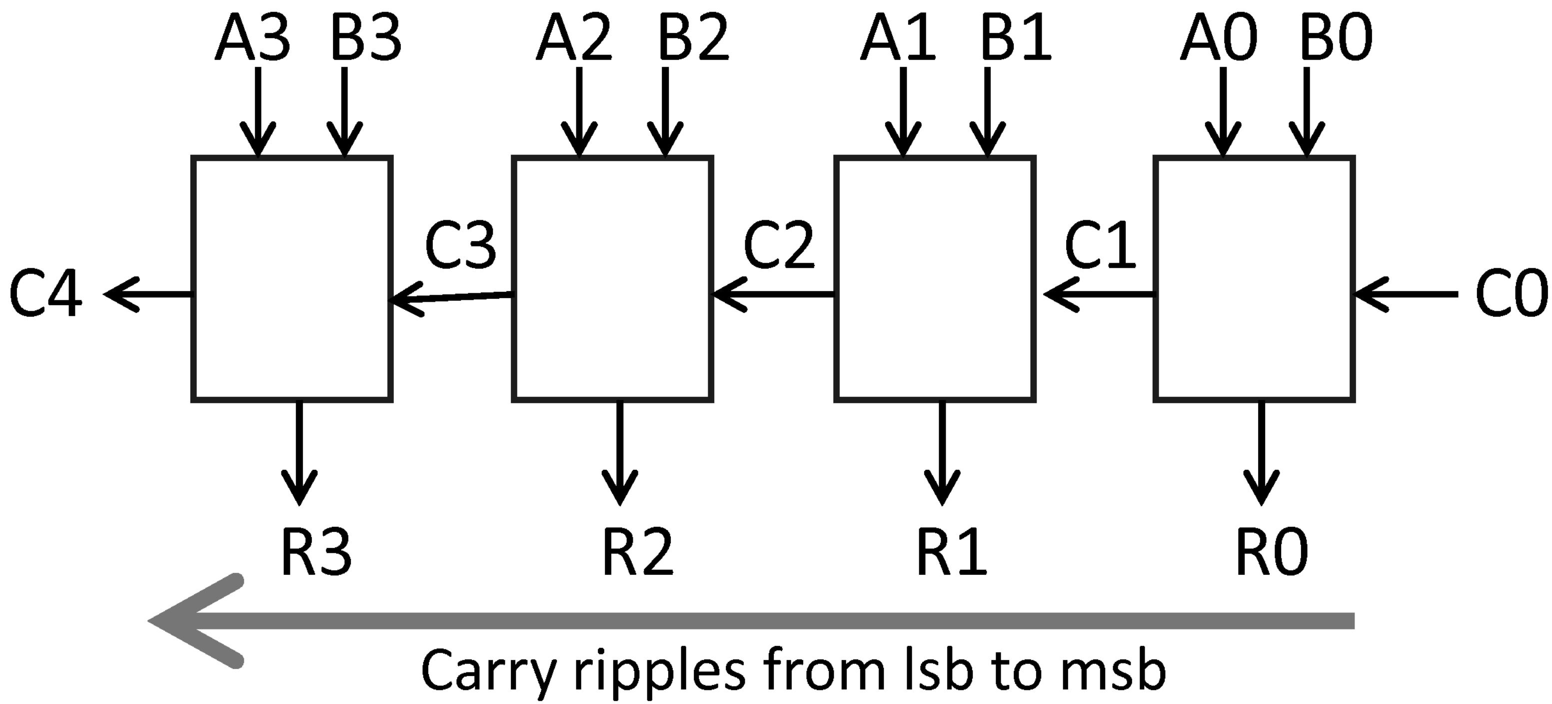
- Is this design fast enough?
- Can we generalize to 32 bits? 64? more?

delay = 8



Speed of a circuit is affected by the number of gates in series (on the *critical path* or the *deepest level of logic*)





- First full adder, 2 gate delay
- Second full adder, 2 gate delay
- …