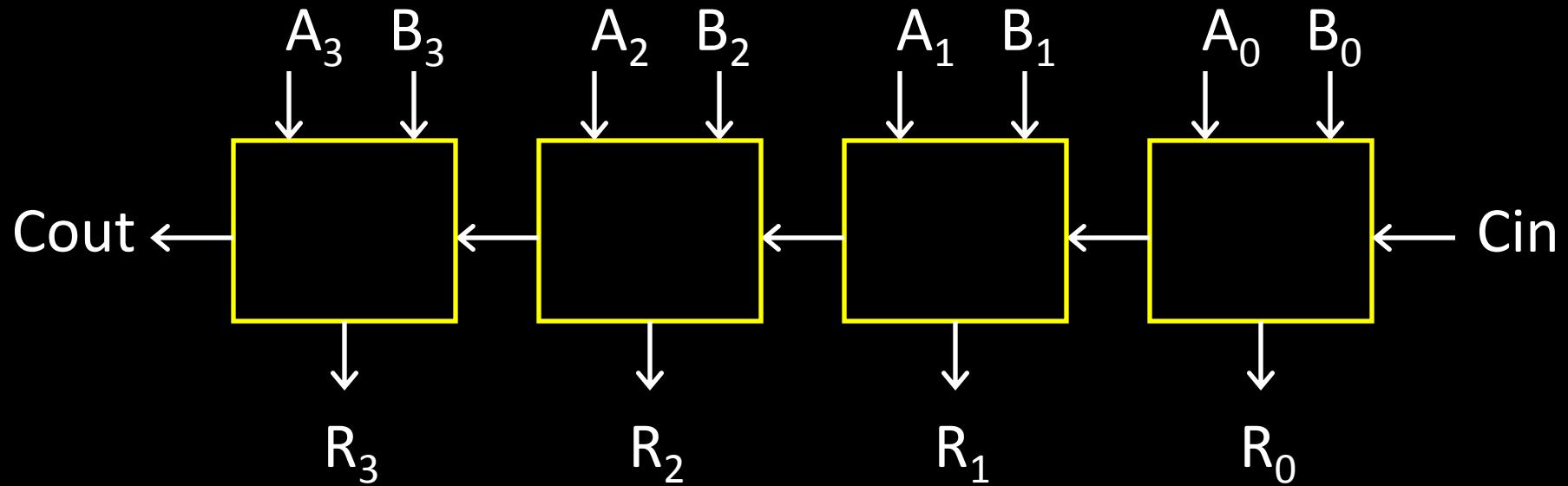


# Arithmetic

Kevin Walsh  
CS 3410, Spring 2010  
Computer Science  
Cornell University

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



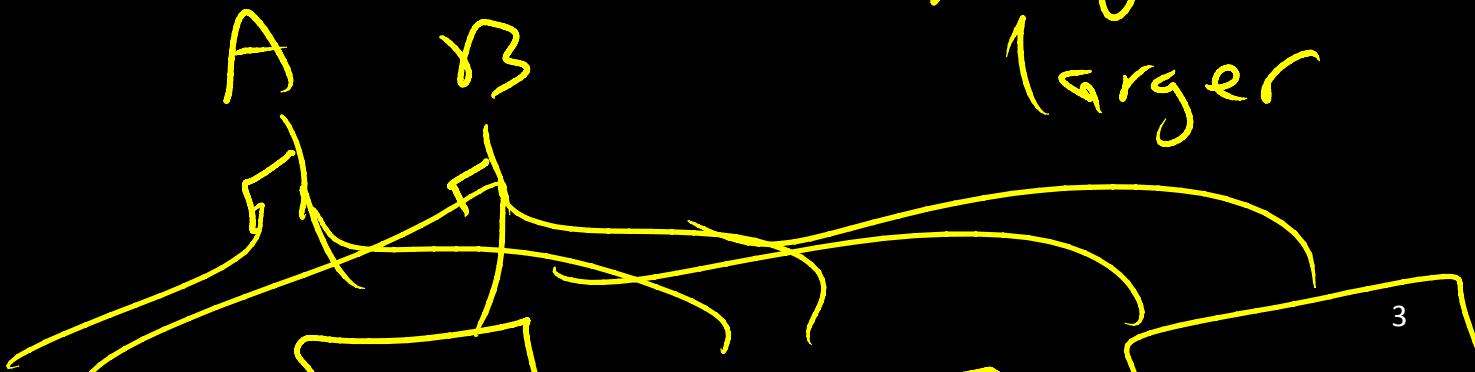
- Adds two 4-bit numbers, along with carry-in
- Computes 4-bit result and carry out

# Addition with negatives:

- pos + pos → add magnitudes, result positive
- neg + neg → add magnitudes, result negative
- pos + neg → subtract smaller magnitude  
 neg + keep sign of bigger magnitude  
 $\Rightarrow$  res. neg

pos + neg  $\Rightarrow$  larger minus smaller

sign of larger



# First Attempt: ~~Sign/Magnitude~~ Representation

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

$$0 \mid \mid \mid = +7$$

$$1 \mid \mid \mid = -7$$

# 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.

$$\begin{array}{r}
 +20 = 0110 \\
 \sim 6 \quad 100' \\
 \hline
 -6 = \overbrace{1010}^{+1}
 \end{array}
 \qquad
 \begin{array}{r}
 +20 = 00010100 \\
 \quad \quad \quad | \\
 \quad \quad \quad 111010'1' \\
 \hline
 -20 = \overbrace{11101100}^{+1}
 \end{array}$$

ignore carry out.

# Non-negatives

(as usual):

$$+0 = 0000$$

$$+1 = 0001$$

$$+2 = 0010$$

$$+3 = 0011$$

$$+4 = 0100$$

$$+5 = 0101$$

$$+6 = 0110$$

$$+7 = 0111$$

$$\cancel{+8 = 1000}$$

# Negatives

(two's complement: flip then add 1):

$$\sim 0 = 1111 \quad 0 = -0 = 0000$$

$$\sim 1 = 1110 \quad -1 = 1111$$

$$\sim 2 = 1101 \quad -2 = 1110$$

$$\sim 3 = 1100 \quad -3 = 1101$$

$$\sim 4 = 1011 \quad -4 = 1100$$

$$\sim 5 = 1010 \quad -5 = 1011$$

$$\sim 6 = 1001 \quad -6 = 1010$$

$$\sim 7 = 1000 \quad -7 = 1001$$

$$\sim 8 = 1011 \quad -8 = 1000$$

# 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:  $0 \dots (2^N) - 1$ 
    - eg: 8 bits  $\Rightarrow 0 \dots 2^8 - 1 = 255$
  - signed (two's complement):
    - ex: 8 bits  $\Rightarrow -128 \dots 0 \dots +127$
- $$- \left( \frac{2^N}{2} \right) \dots \left( \frac{2^N}{2} \right) - 1$$

## Extending to larger size

$$\begin{array}{r} 0110 \\ -1 = 1111 \end{array}$$



## Truncate to smaller size

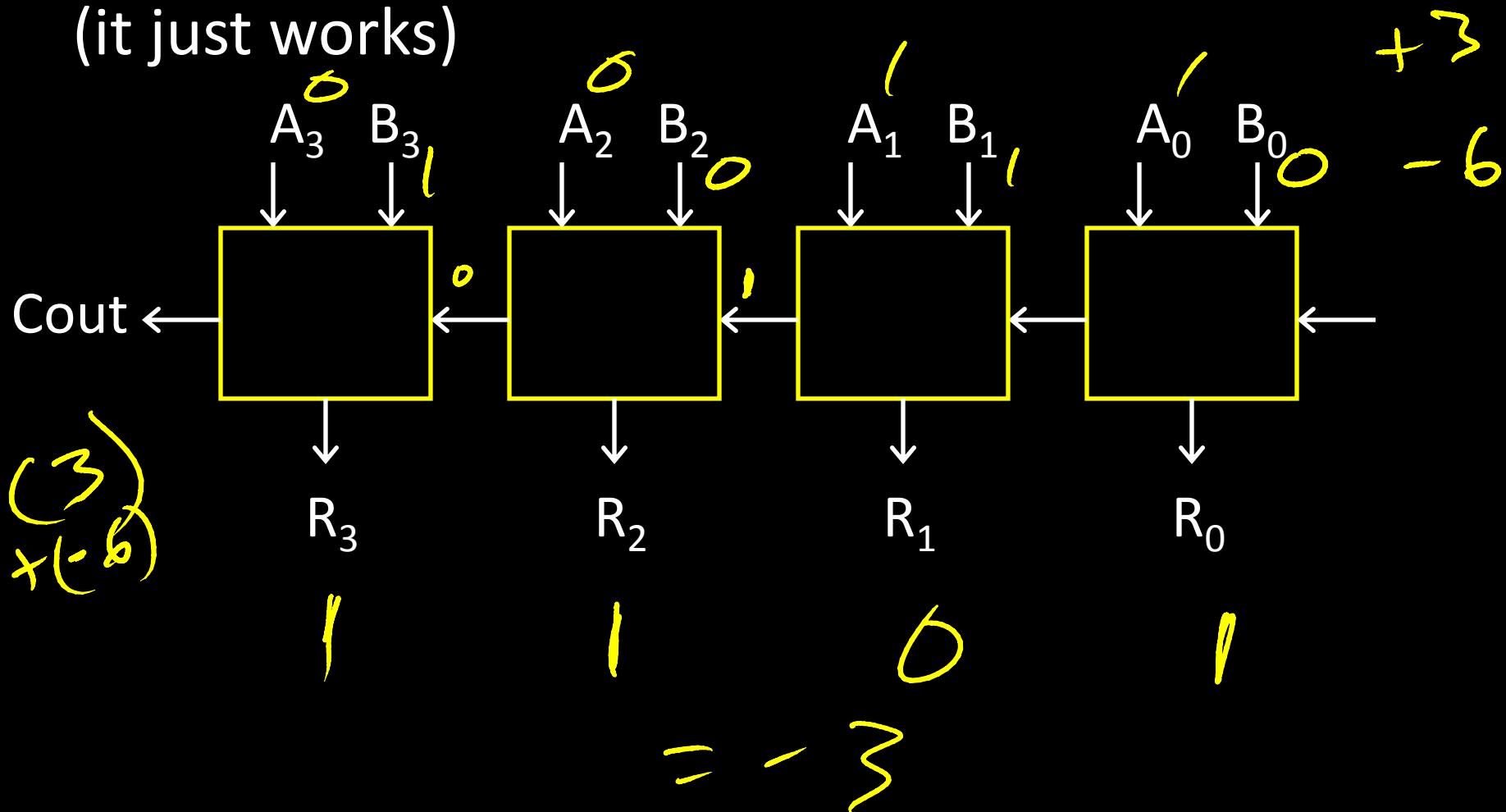
Drop leading sign bits, as long as sign doesn't change

~~0000 1111 - 15~~

~~0111 = 7~~

# Addition with two's complement signed numbers

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



How does that work?

$$\begin{array}{r} -154 \\ +283 \\ \hline \end{array}$$

ten's complement

$$\begin{array}{r}
 999 \\
 -154 \\
 \hline
 845
 \end{array}
 \quad
 \begin{array}{r}
 845 \\
 -1 \\
 \hline
 844
 \end{array}
 \quad
 \begin{array}{r}
 844 \\
 -283 \\
 \hline
 561
 \end{array}$$

= 0

# Overflow

- adding a negative and a positive?

Can't happen

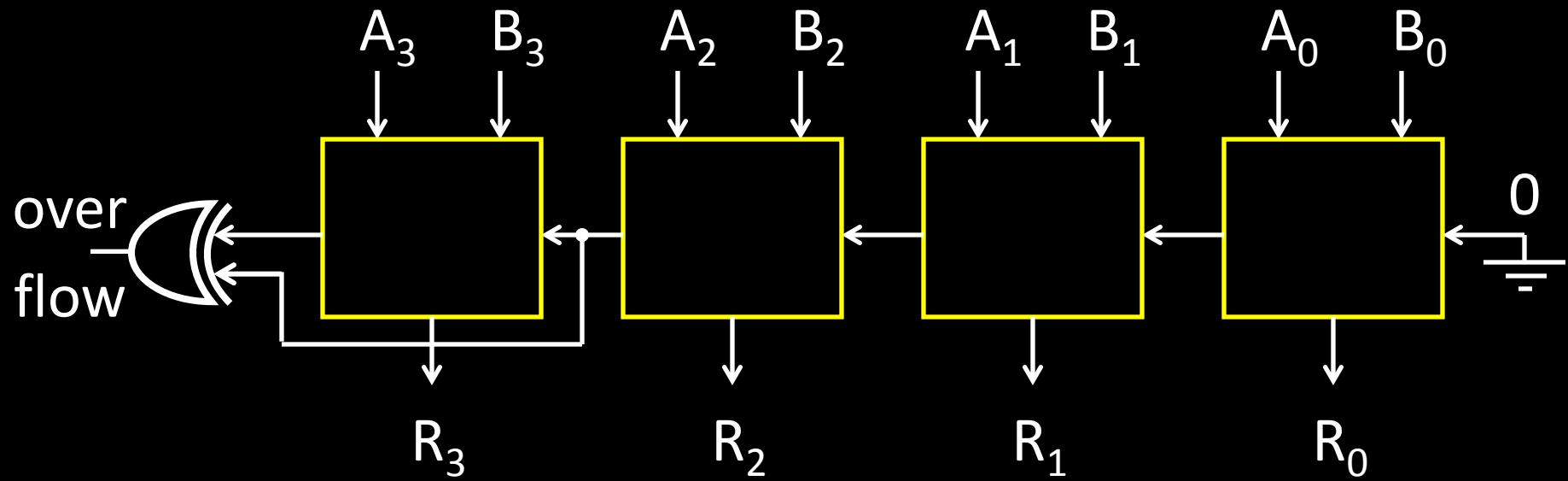
- adding two positives?

- adding two negatives?

Rule of thumb:

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

# Two's Complement Adder with overflow detection



$$\begin{array}{r} 0111 \\ + 1 \\ \hline 1000 \end{array}$$

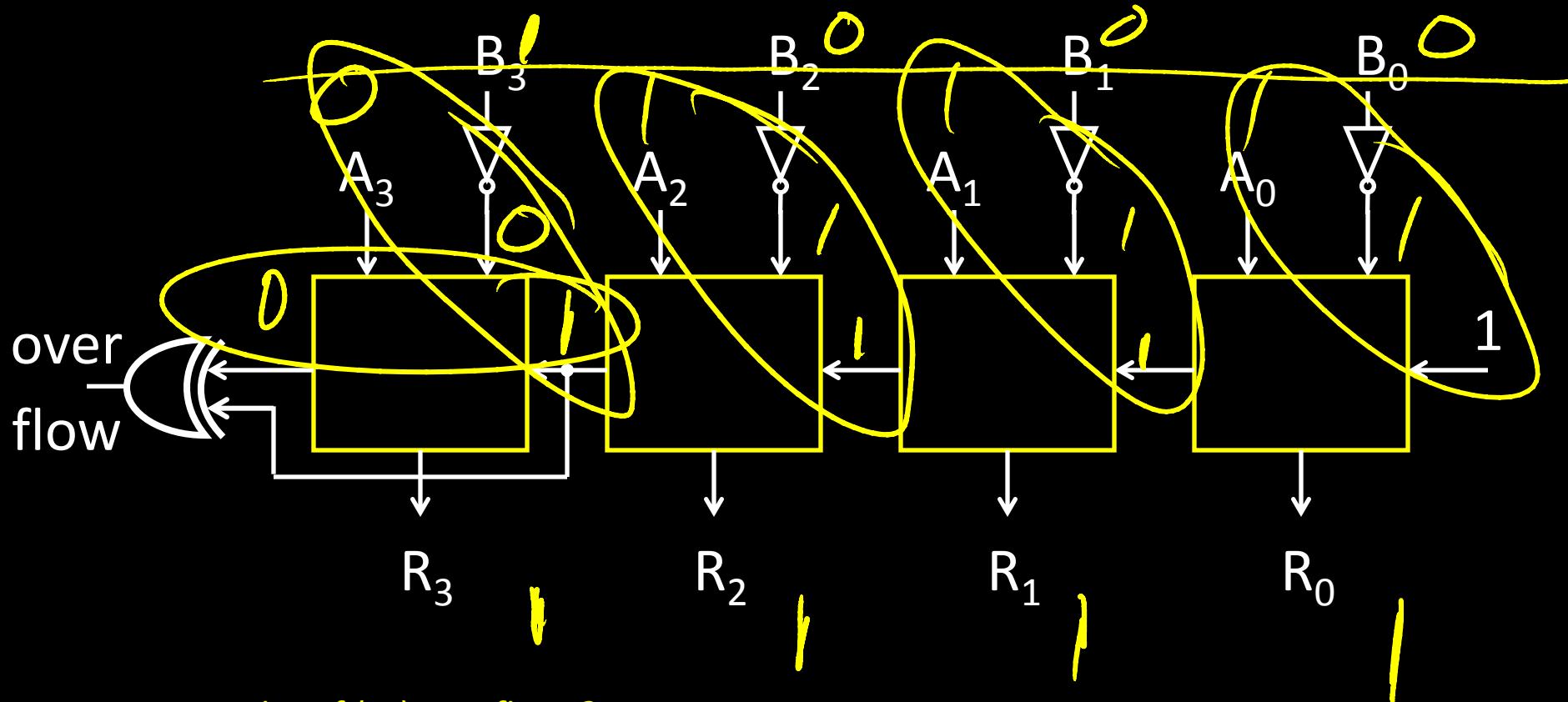
# Two's Complement Subtraction

Lazy approach

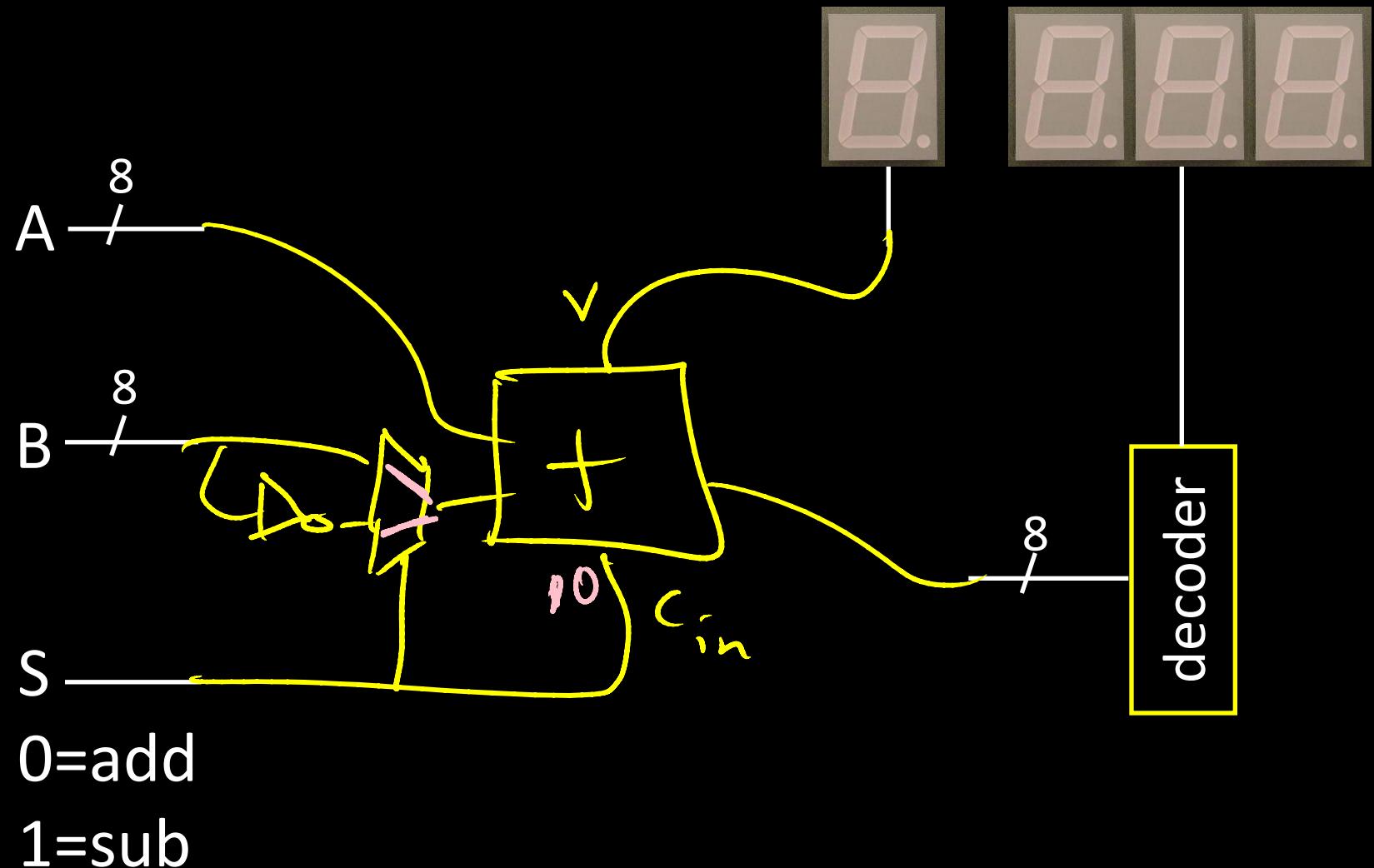
$$A - B = A + (-B) = A + (\bar{B} + 1)$$

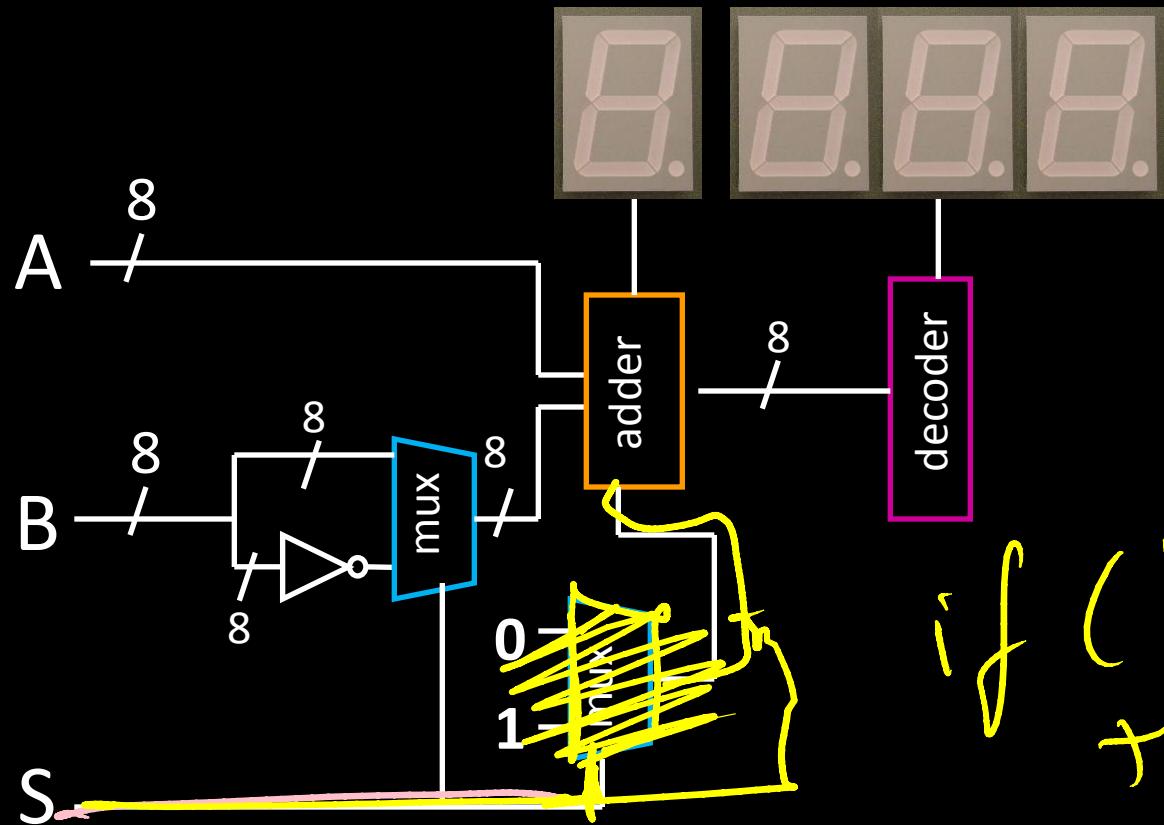
$$B \equiv -8 = \boxed{1000}$$

$$(7) - (-8)$$



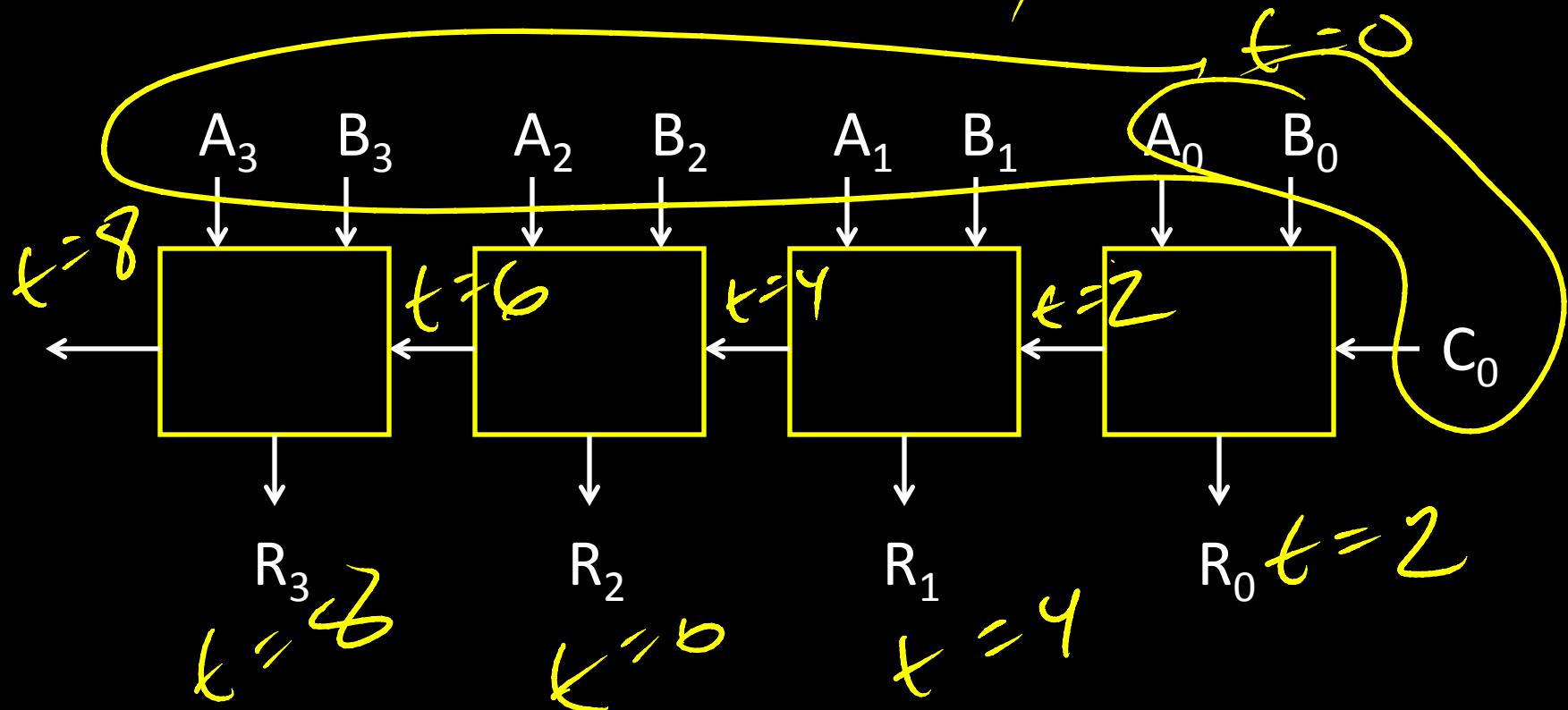
Q: What if  $(-B)$  overflows?



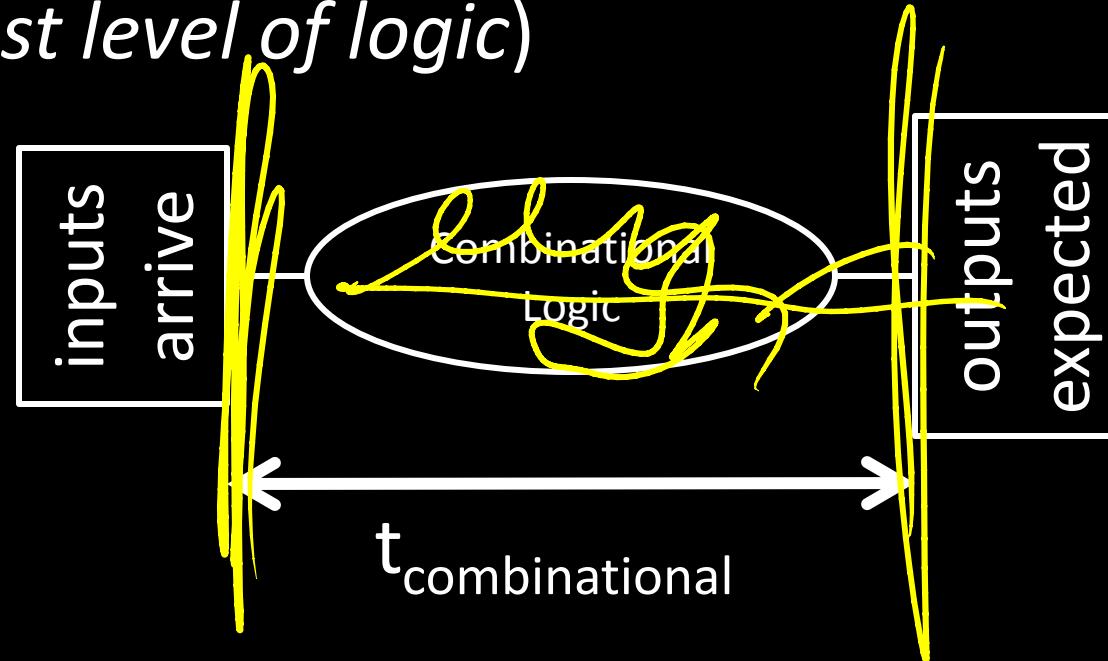


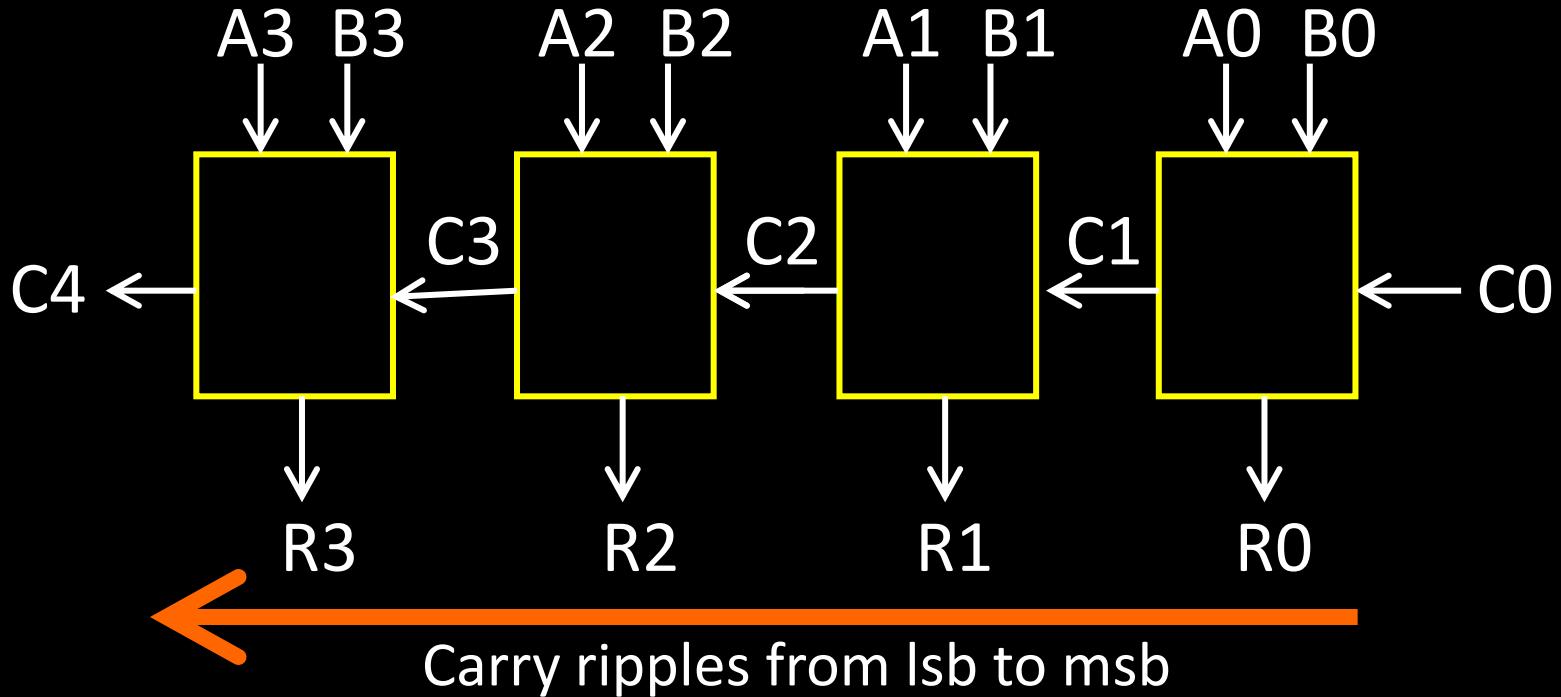
- Is this design fast enough?
- Can we generalize to 32 bits? 64? more?

*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
- ...