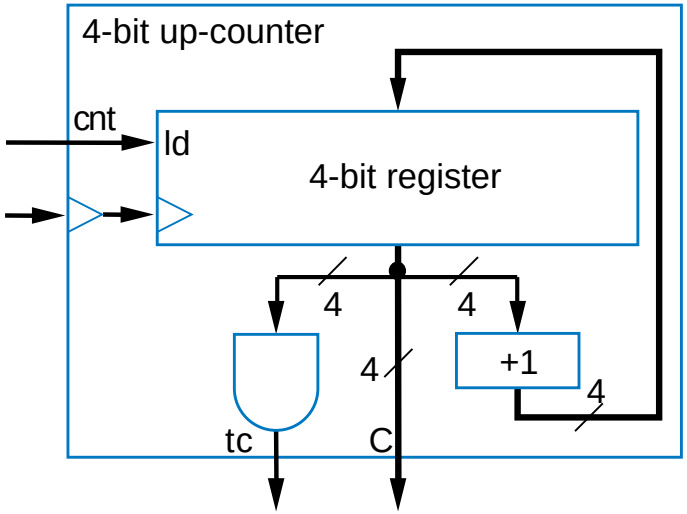
B38DB: Digital Design and Programming
Datapath Components – Subtractor

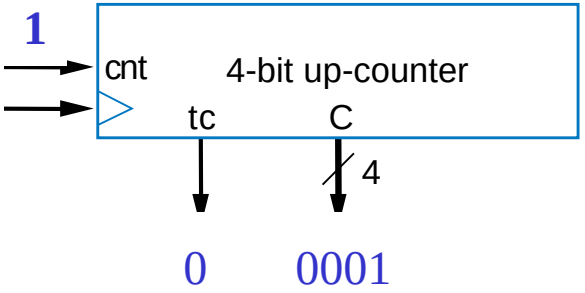
Mustafa Suphi Erden

Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering
Room: EM 2.01
Phone: 0131-4514159
E-mail: m.s.erden@hw.ac.uk

Counters

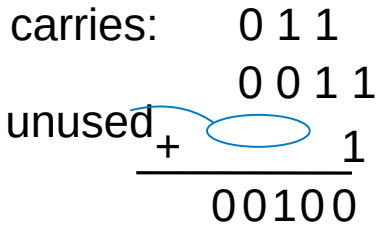
- ***N-bit up-counter***: N-bit register that can increment (add 1) to its own value on each clock cycle
  - 0000, 0001, 0010, 0011, ..., 1110, 1111, 0000
  - Note how count "rolls over" from 1111 to 0000
    - Terminal (last) count, tc, equals1 during value just before rollover
- Internal design
  - Register, incrementer, N-input AND gate

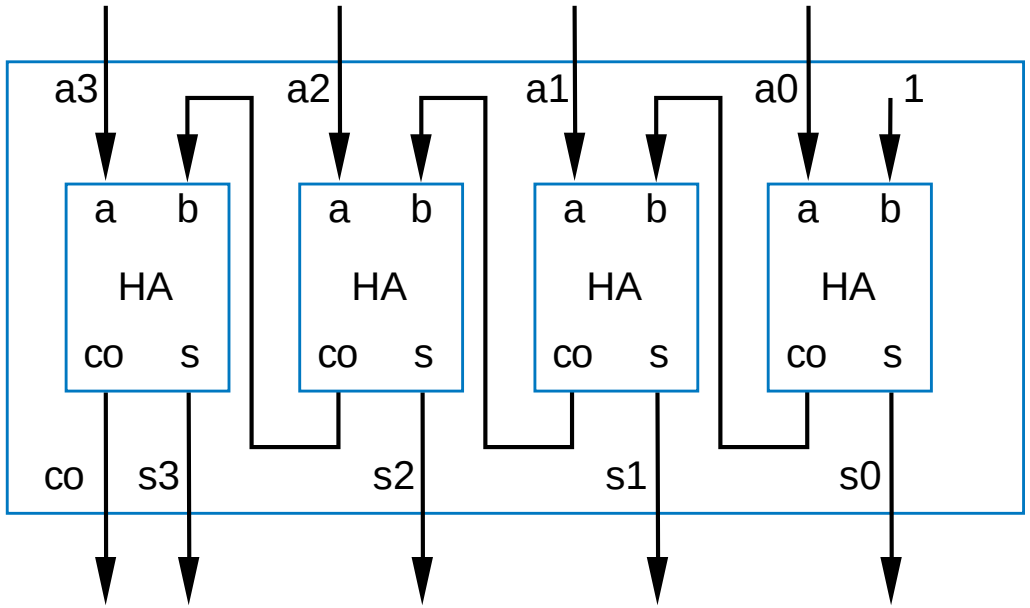


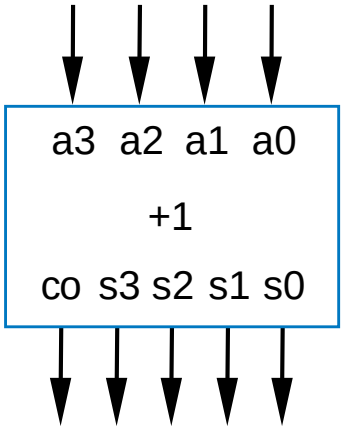


Incrementer

- We could use carry-ripple adder with B input set to 00...001
  - But when adding 00...001 to another number, the leading 0's obviously don't need to be considered -- so just two bits being added per column
- Use half-adders (adds two bits) rather than full-adders (adds three bits)



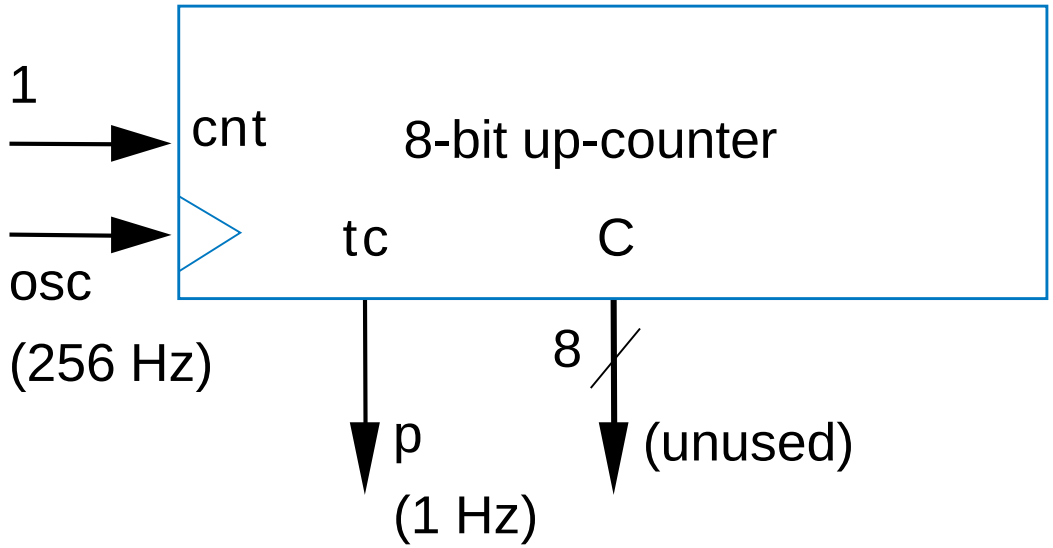




Similarly, implement a decrementer with a Half-Subtractor!

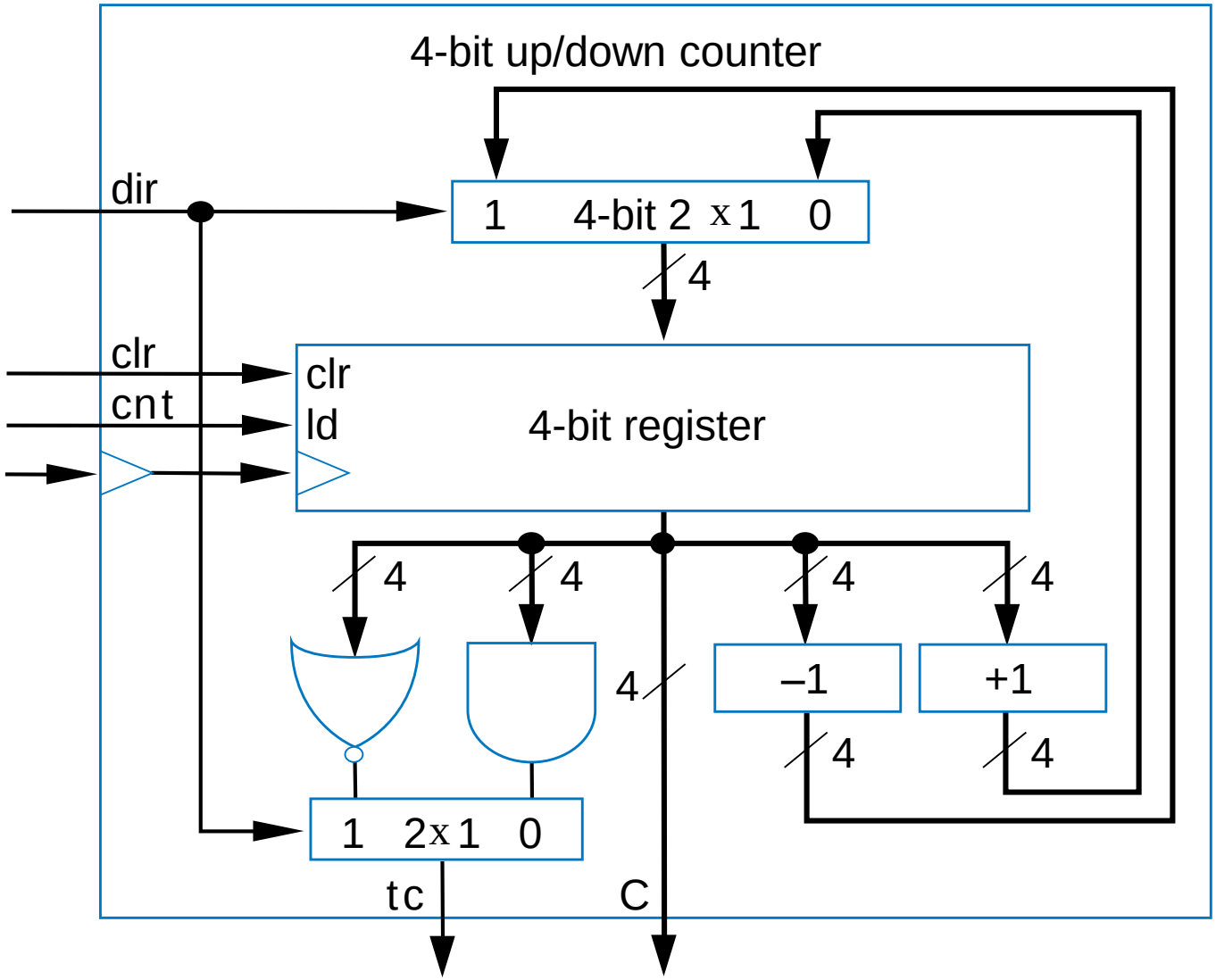
Counter Example:
1 Hz Pulse Generator Using 256 Hz Oscillator

- Suppose we have **256 Hz** oscillator, but we want **1 Hz** pulses
  - 1 Hz is 1 pulse per second



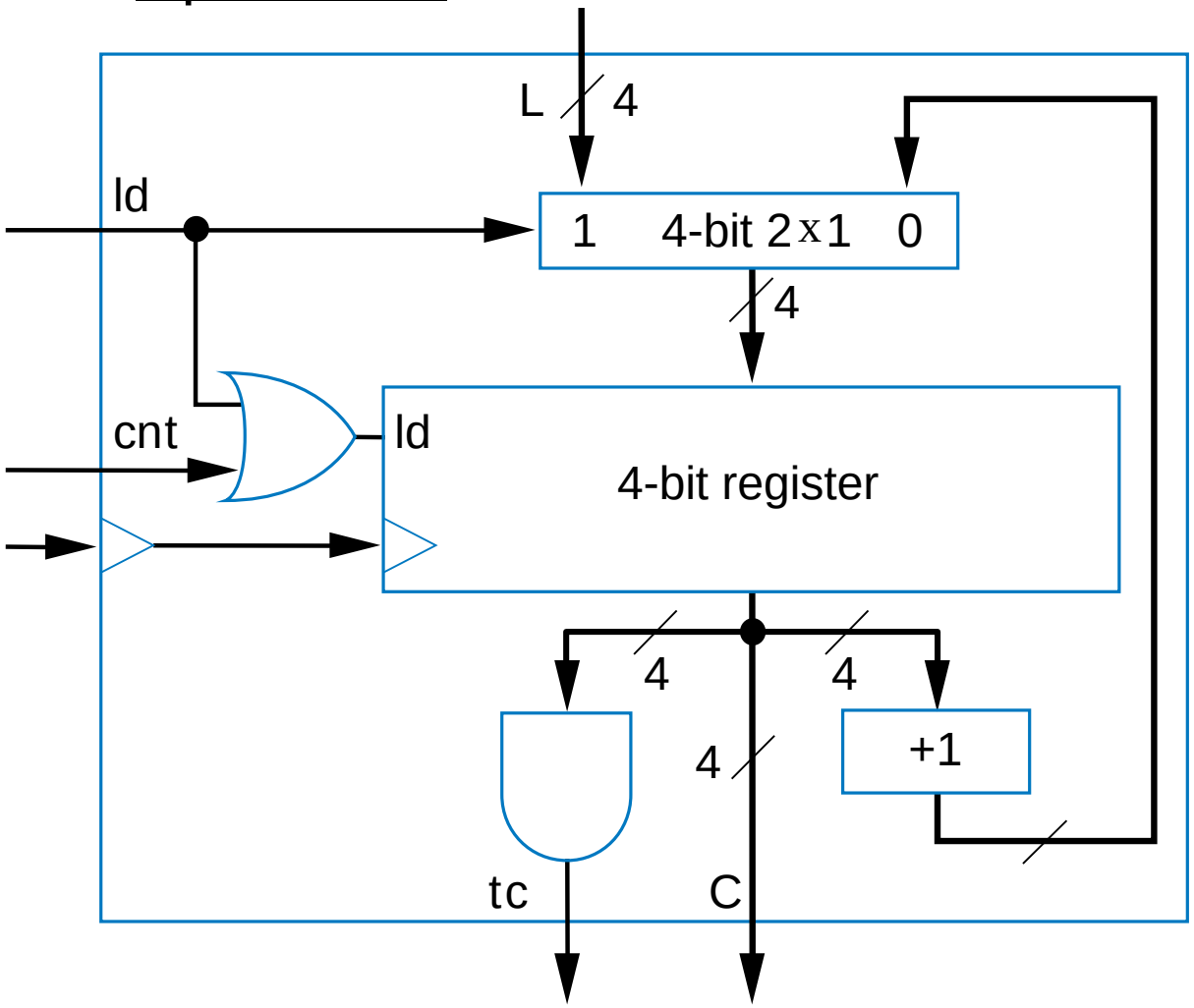
- 8-bit up-counter, uses tc output as pulse
  - Counts from 0 to 255 (256 counts), so pulses tc every 256 cycles

Up/Down-Counter

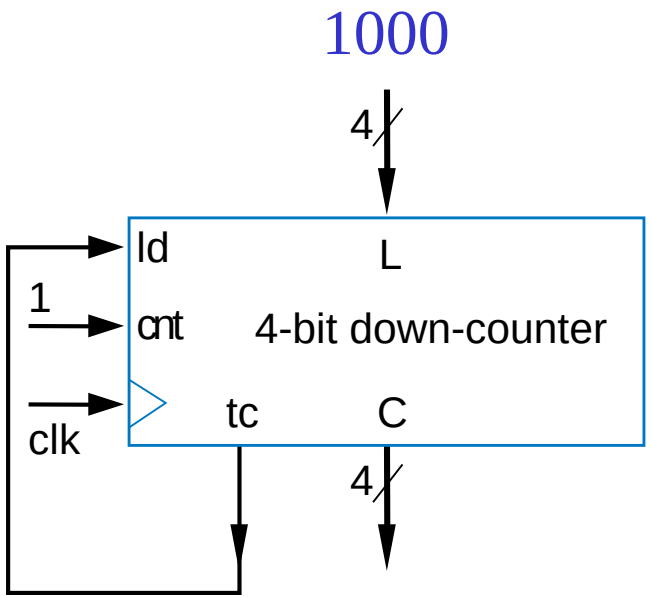


Counter with Parallel Load

- Up-counter that can be loaded with external value

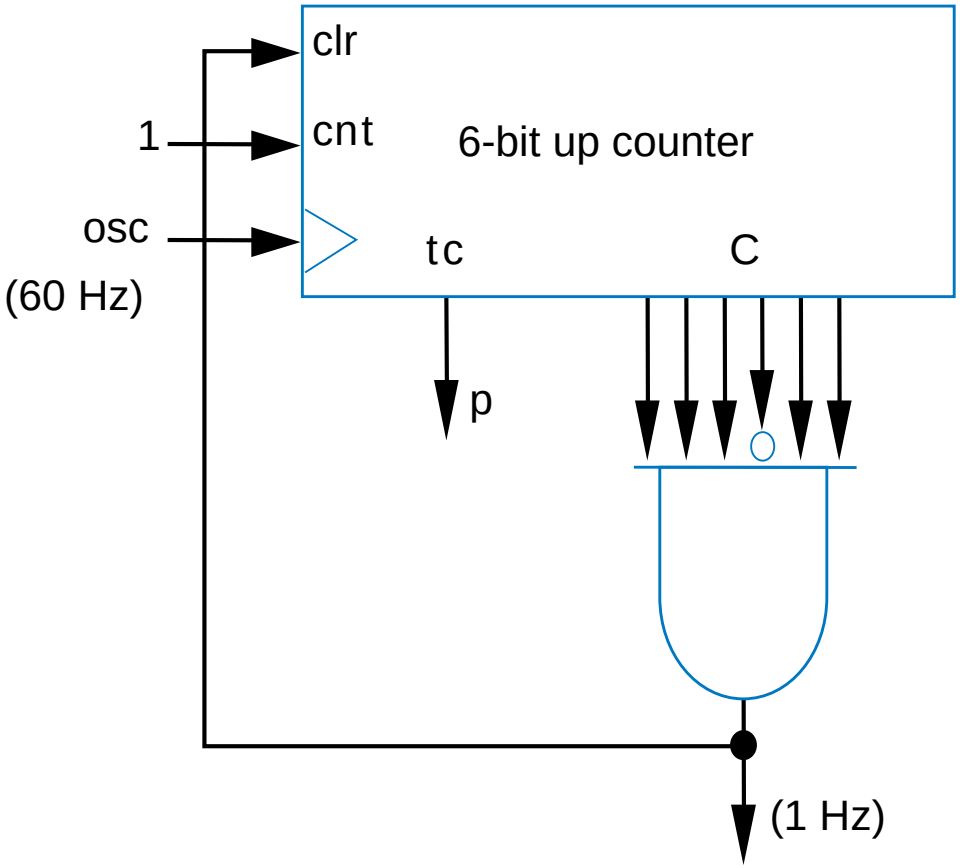


- Down-counter that can be loaded with external value



Counter Example: 1 Hz Pulse Generator from 60 Hz Clock

- U.S. electricity standard uses 60 Hz signal
- Use 6-bit up-counter
  - Can count from 0 to 63
  - Create simple logic to detect 59 (for 60 counts)
    - Use to clear the counter back to 0 (or to load 0)



Representing Negative Numbers: Two's Complement

- We can build a subtractor as we built a carry-ripple adder; but we do not need to!
- How to represent negative numbers in binary?
- Signed-magnitude
  - Use leftmost bit for sign bit
    - So -5 would be:

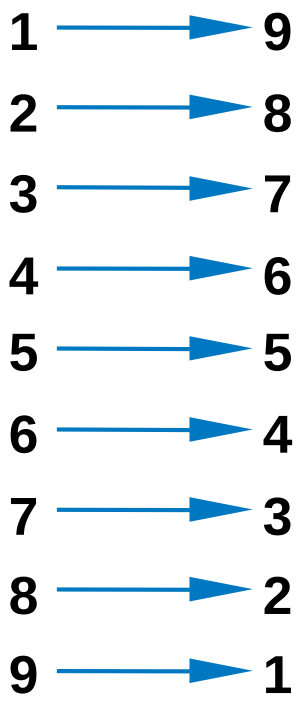
      1101 using four bits

      10000101 using eight bits
- Better way: **Two's complement**
  - Big advantage: Allows us to perform **subtraction using addition**
  - Thus, **no need for a separate subtractor component**!

Ten's Complement (1/2)

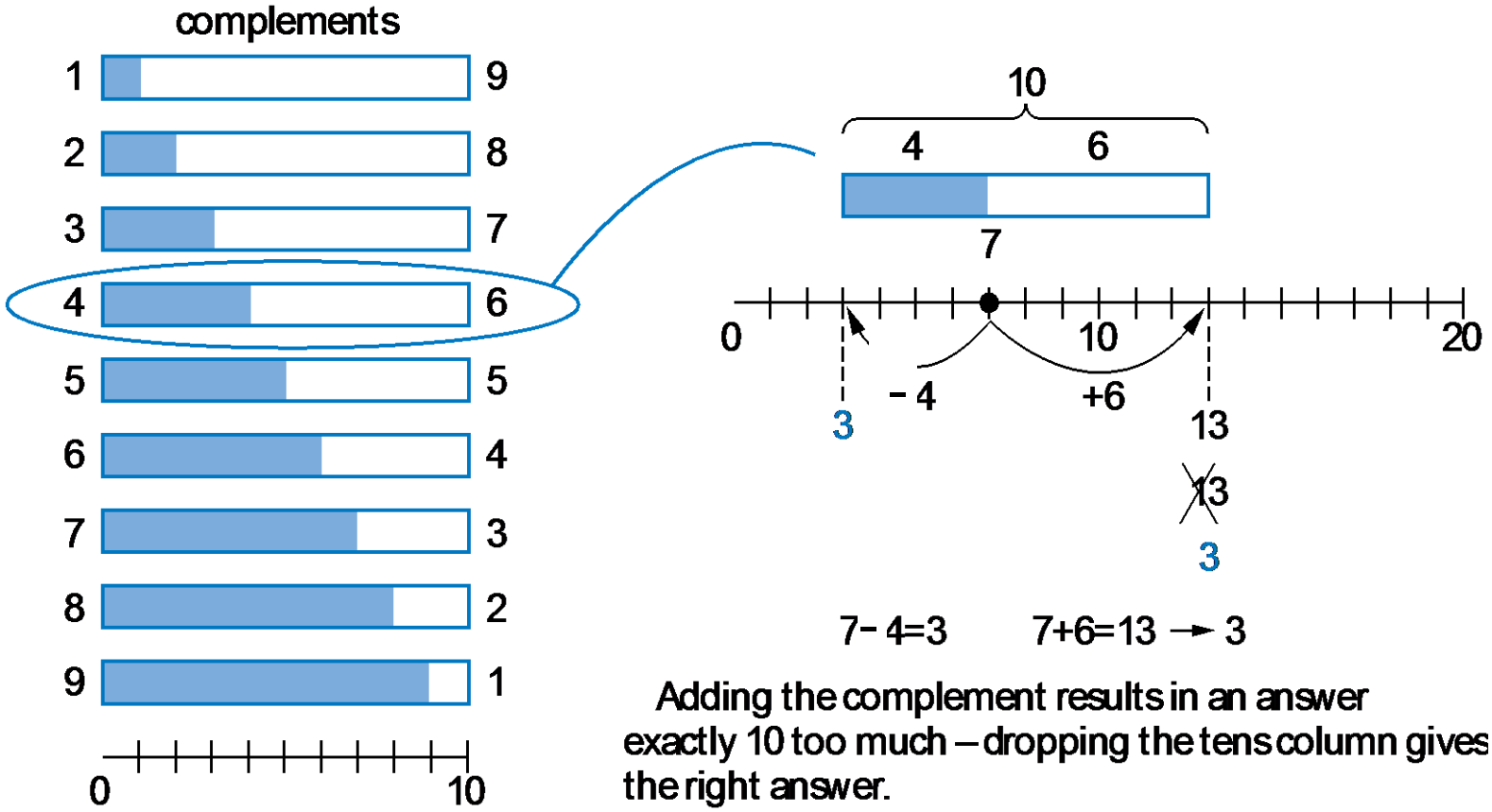
- Before introducing two's complement, let's consider ten's complement
  - But, be aware that computers DO NOT USE TEN'S COMPLEMENT. Introduced for intuition only.
  - Complements of each base ten number are shown on the right

– Complement is the number that when added it gives the result 10



Ten's Complement (2/2)

- Instead of subtracting a number, adding its complement results in an answer with exactly 10 larger than what it should be
- So just drop the 1 → results in subtracting using addition only



Two's Complement is Easy to Compute: Just Invert Bits and Add 1

- Two's complement of 011 is 101, because 011 + 101 is 1000
- Could compute complement of 011 as 1000 – 011 = 101
- **Easier method: Just invert all the bits, and add 1**
- The complement of 011 is 100+1 = 101 – it works!

Q: What is the two's complement of 0101? A: 1010+1=1011
(check: 0101+1011=10000)

Q: What is the two's complement of 0011? A: 1100+1=1101

Two's Complement Subtractor Built with an Adder

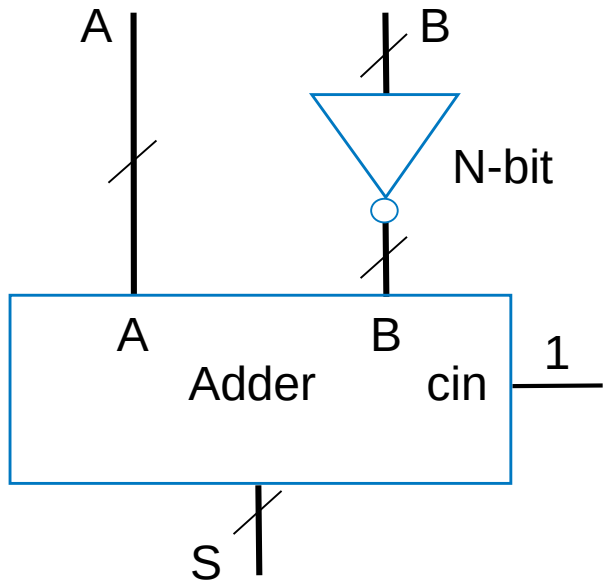
- Using two's complement

$$A - B = A + (-B)$$

$$= A + (\text{two's complement of } B)$$

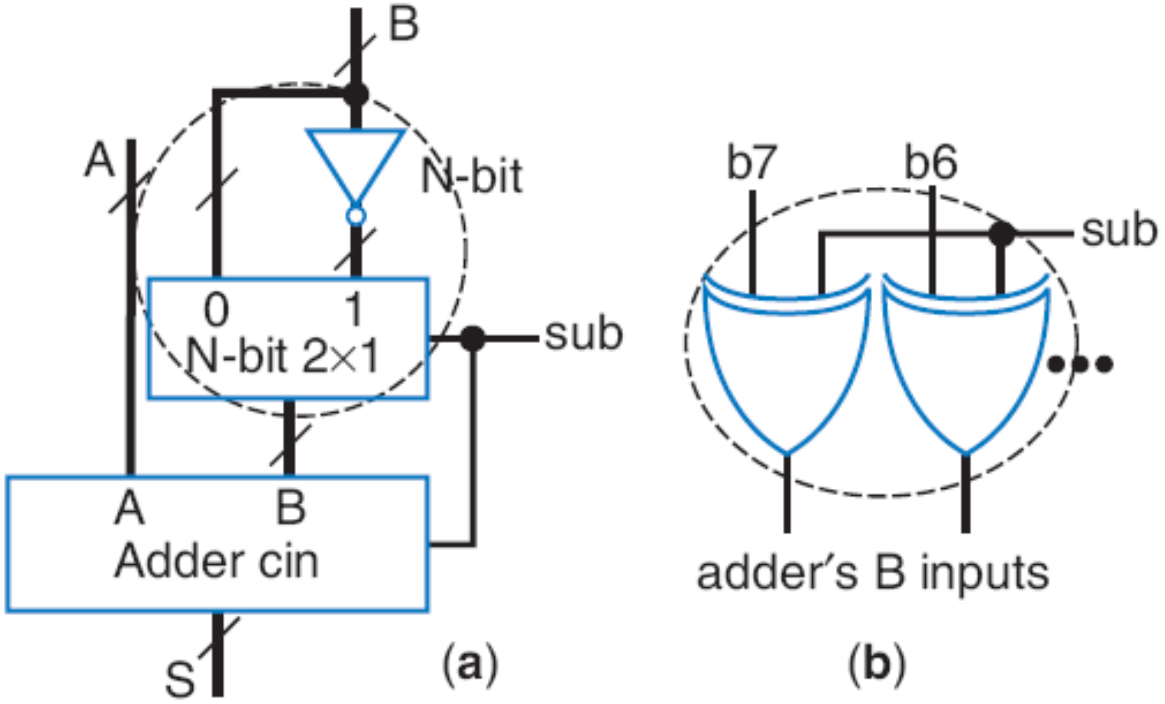
$$= A + \text{invert\_bits}(B) + 1$$

- Build a subtractor using an adder, inverting B's bits, and setting carry-in to 1:



Adder/Subtractor

- Adder/subtractor: control input determines whether add or subtract
  - Can use 2x1 mux – sub input passes either B or inverted B
  - Alternatively, can use XOR gates – if sub input is 0, B's bits pass through; if sub input is 1, XORs invert B's bits

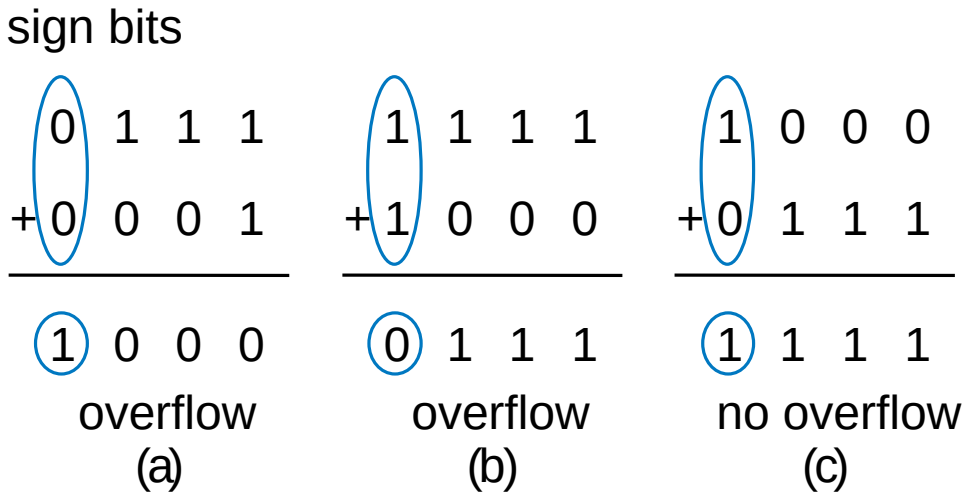


Extra information: Overflow

- Sometimes the result can not be represented with given number of bits:
  - Too large magnitude of either positive or negative value
  - e.g., 4-bit two's complement addition of 0111+0001 (7+1=8). But 4-bit two's complement can't represent number >7
    - 0111+0001 = 1000 WRONG answer, 1000 in two's complement is -8, not +8
  - Adder/subtractor should indicate when overflow has occurred, so that the result can be discarded.

Detecting Overflow

- Assuming 4-bit two's complement numbers, can detect overflow by detecting when the two numbers' sign bits are the same but are different from the result's sign bit
  - If the two numbers' sign bits are different, overflow is impossible
    - Adding a positive and negative can't exceed largest magnitude positive or negative
- Simple circuit
  - overflow = a3'b3's3 + a3b3s3'
  - Include "overflow" output bit on adder/subtractor



If the numbers' sign bits have the same value, which differs from the result's sign bit, overflow has occurred.