In a DRAM cell, reading the capacitor changes the capacitor's voltage level.

- value is wiped out
- after reading the value, the hardware has to write that value back to the capacitor.

Capacitors also "leak".

- they lose their charge over time.
- hardware has to periodically read every DRAM cell and write its value back to it.

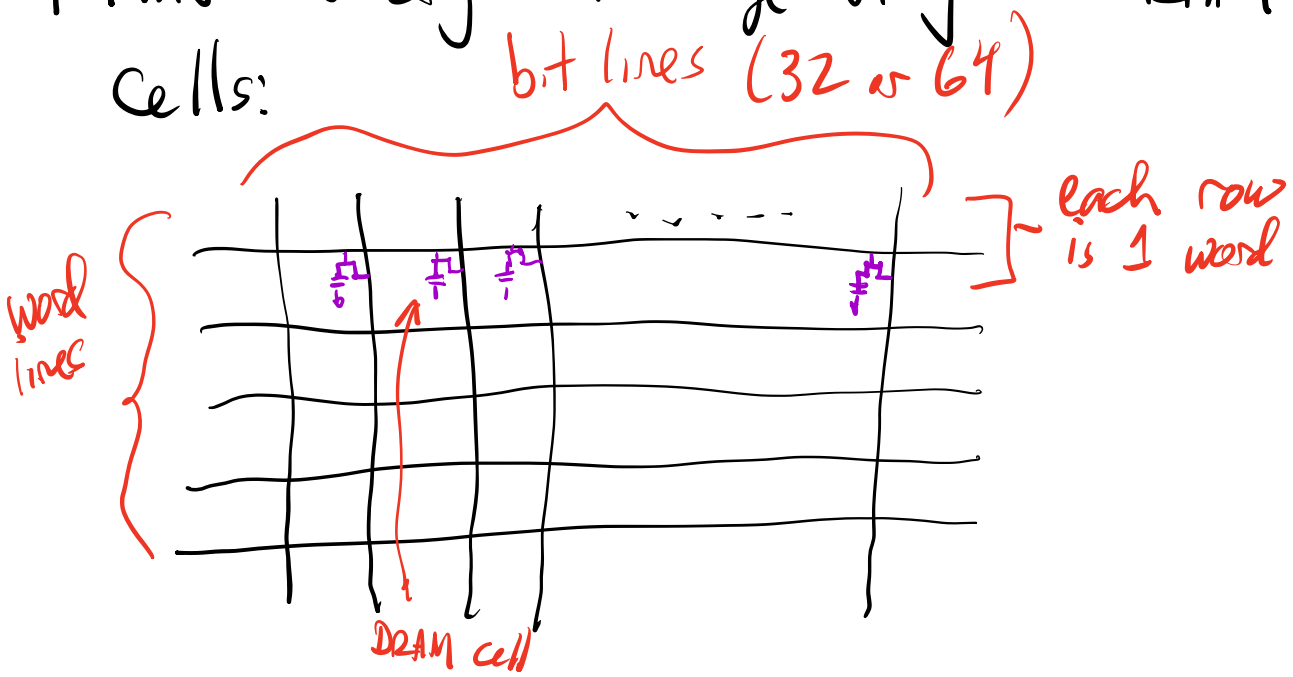
"refresh"

- happens every 64ms
  - roughly a 0.4% overhead.
- this dynamic refresh process is why DRAM is "dynamic" RAM.

- SRAM doesn't need refreshing

"static" RAM

Main memory is a huge array of DRAM cells:

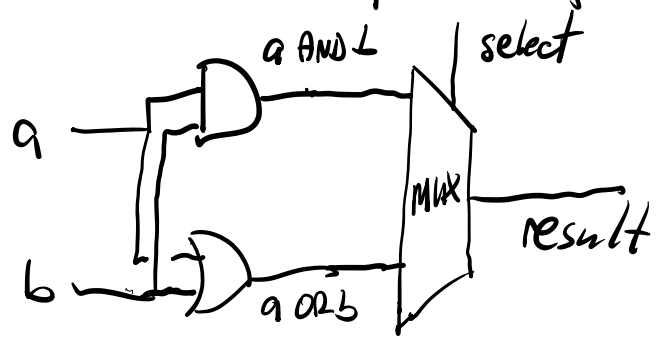


ALU - "arithmetic logic unit"

- performs the arithmetic and logic operations.
  - combinational circuit - performs computations using gates, doesn't store anything.

Constructing a simple ALU:

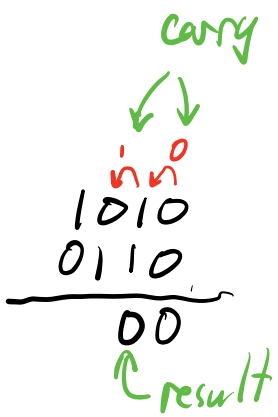
1-bit ALU performing AND and OR



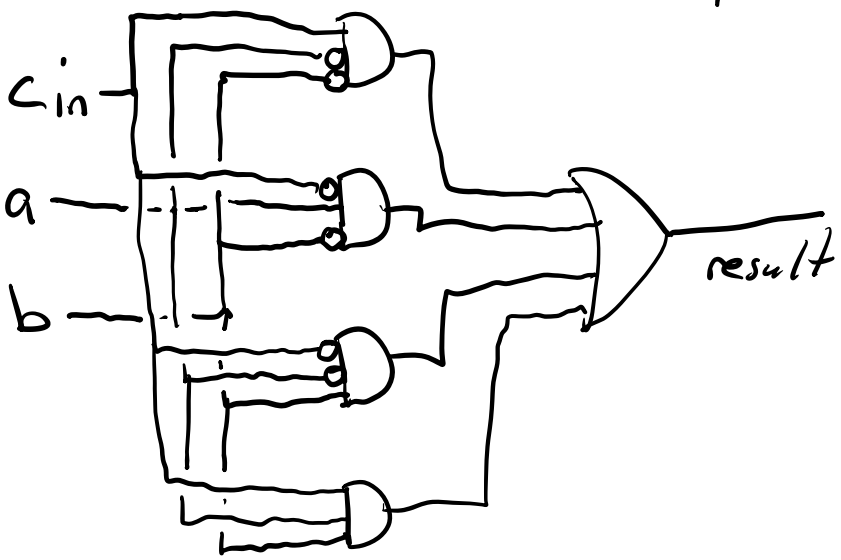
Performing addition:

1-bit adder:

- inputs: carry-in, a, b
- outs: result, carry-out

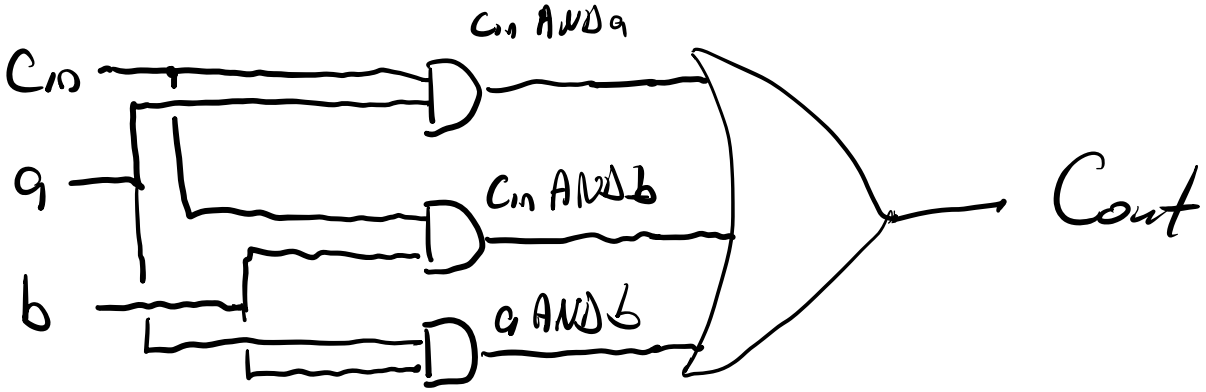


Result = 1 when exactly one input is 1 or all three inputs are 1.

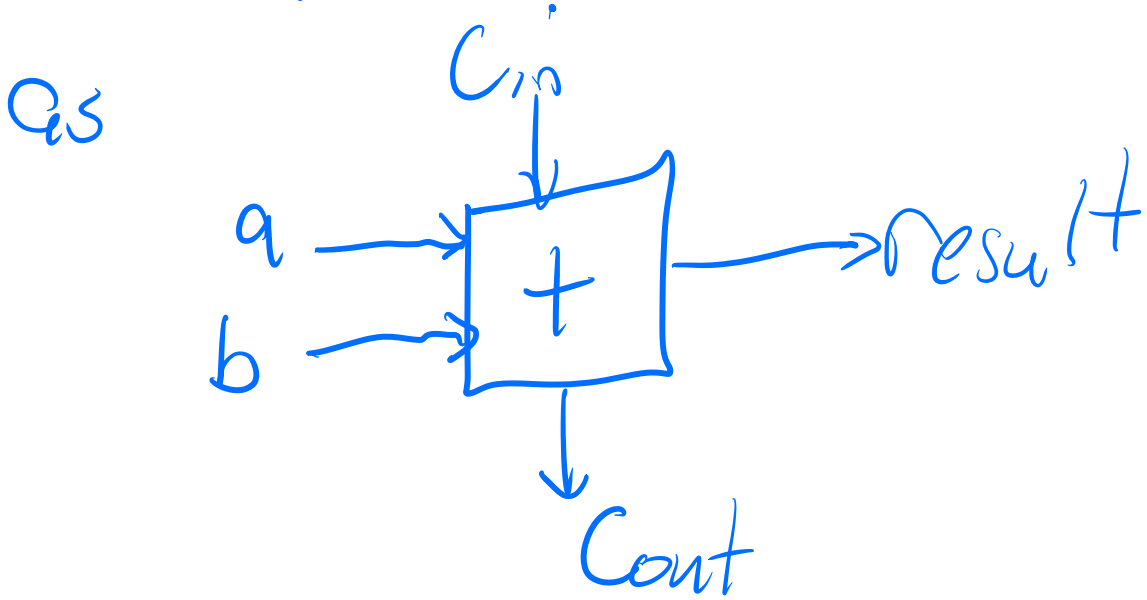


Carry out:

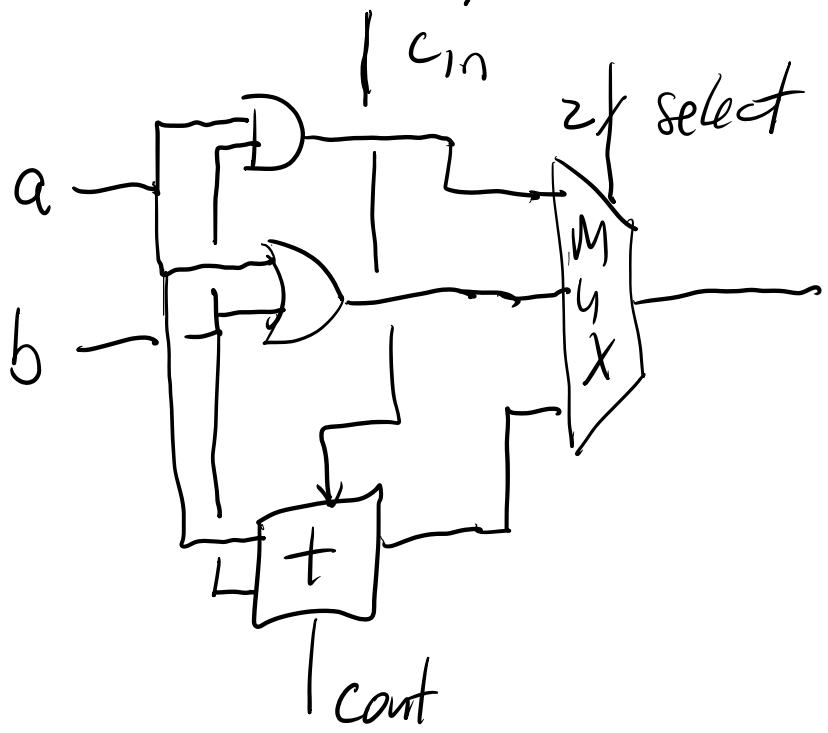
Carry out = 1 when at least two inputs are 1.



Draw the 1-bit adder



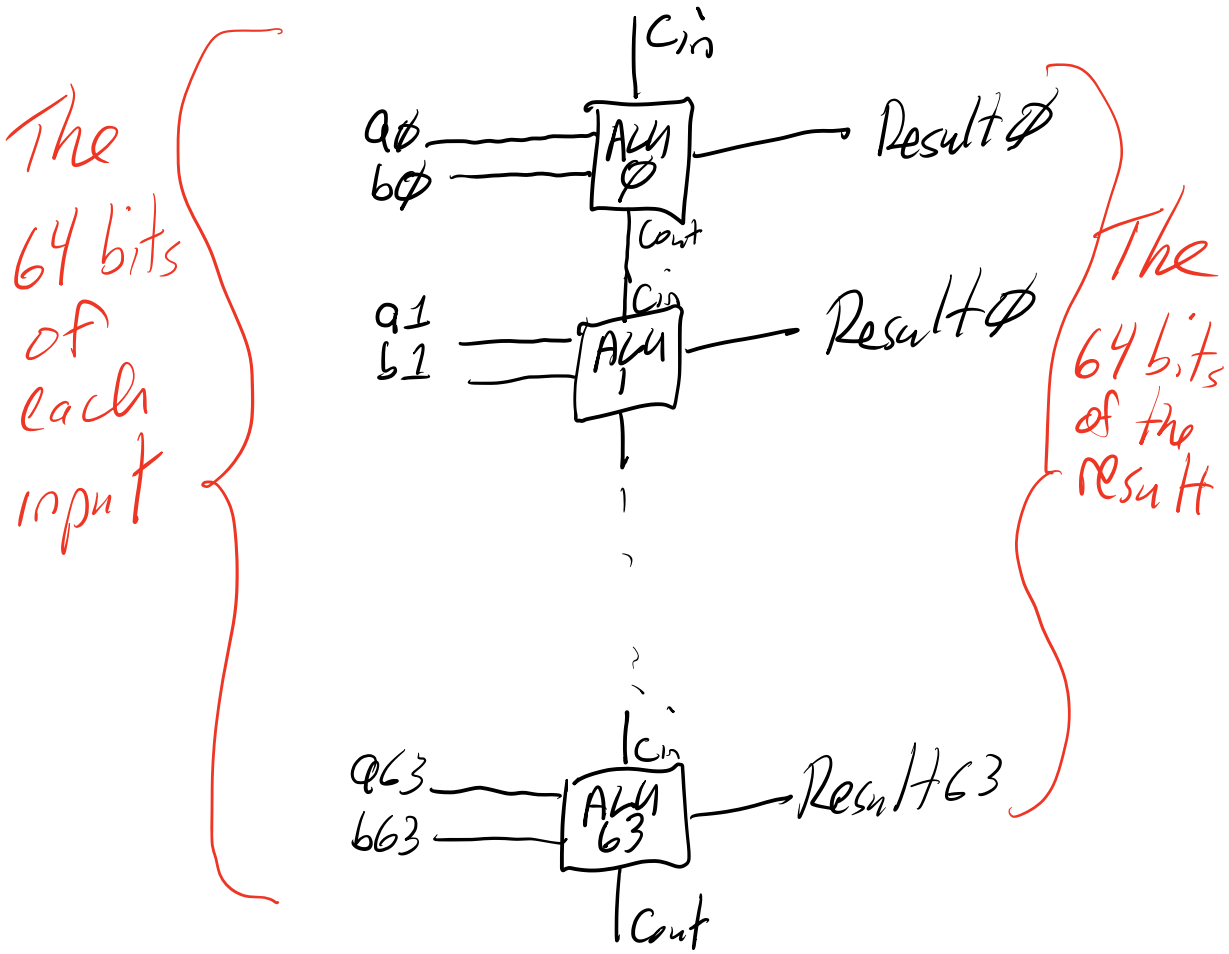
1-bit ALU w/ AND, OR, +



64-bit ALU

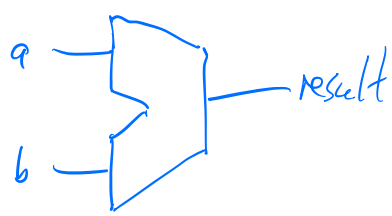
- bitwise AND, OR
- 64-bit addition

- just chain together 64 1-bit ALUs.



The Cout from an ALU is the Cin to the next ALU.
- "ripple adder"

We draw an ALU as:



What about subtraction?

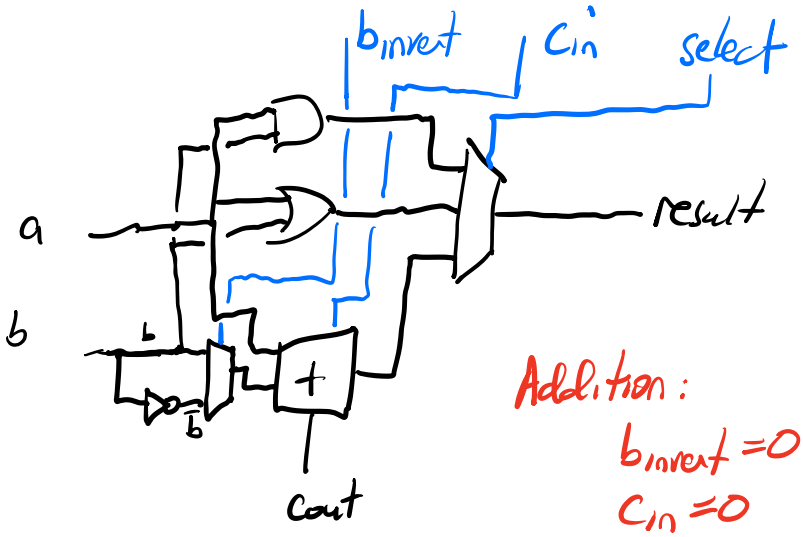
- really easy, thanks to 2's complement!

$$a - b = a + (-b) = a + (\bar{b} + 1)$$

$$= (a + \bar{b}) + 1$$

- use the adder
- flip bits of b
- set Carry In to 1.

1-bit ALU with addition, subtraction, and, or



Addition:

$b_{invert} = 0$

$C_{in} = 0$

Subtraction:

$b_{invert} = 1$

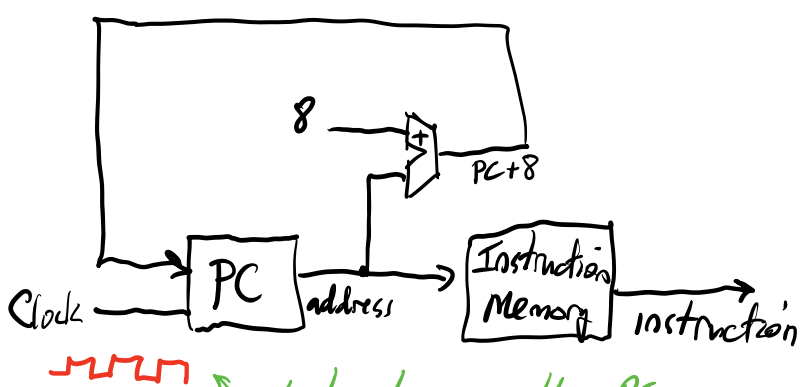
$C_{in} = 1$

The processor **datapath**
- how the data flows through the processor to support instructions.

The processor **control** specifies the values of multiplexer select lines, b invert, etc.
- not discussed here.

Portion of the datapath for fetching an instruction:

- uses a program counter (PC)
  - special register containing the address of the next instruction.
  - aka "instruction pointer"
    - %rip



clock makes sure the PC isn't overwritten prematurely.

Assume every instruction is 8 bytes (64-bits)

After each instruction,
PC ← PC+8
to point to the next instruction.