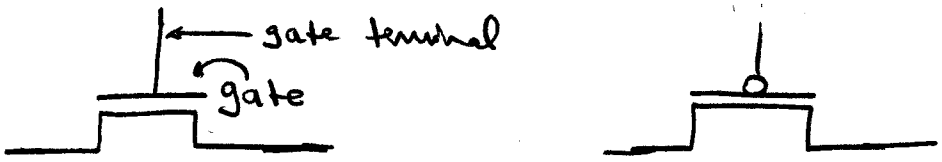
LECTURE #13

(A) CMOS Technology:

transistor: a basic switch.

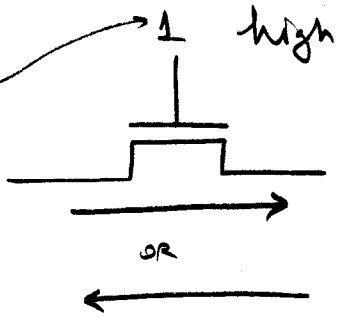
two types in CMOS: [complementary MOS] of transistors



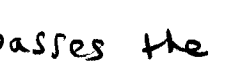
N-MOS P-MOS

Basic idea:

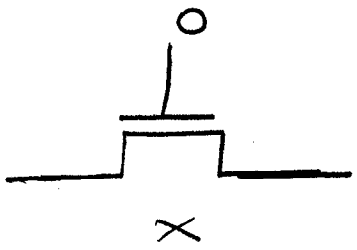
N-MOS: Apply 1 high voltage at gate



OR



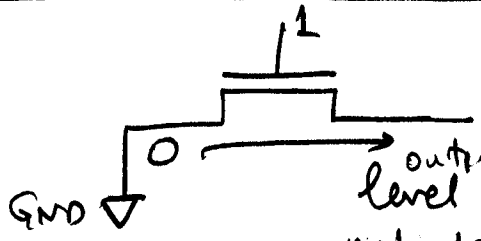
passes the signal (current) through.

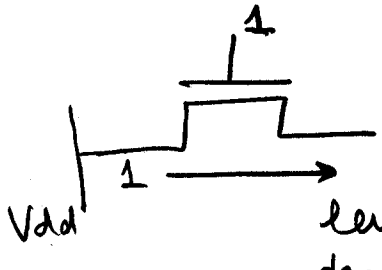


signal blocked

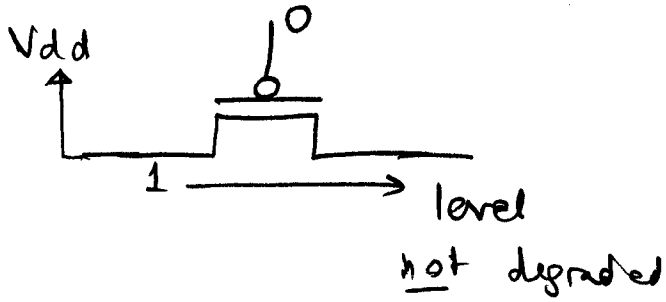
||| equivalent to:

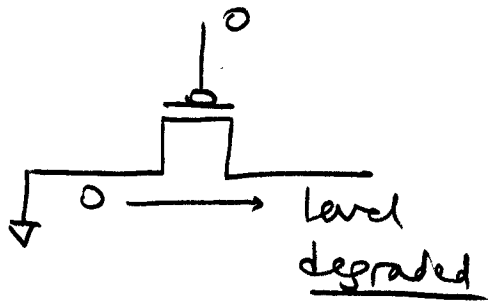
- N-MOS passes "0" well, (low voltage)





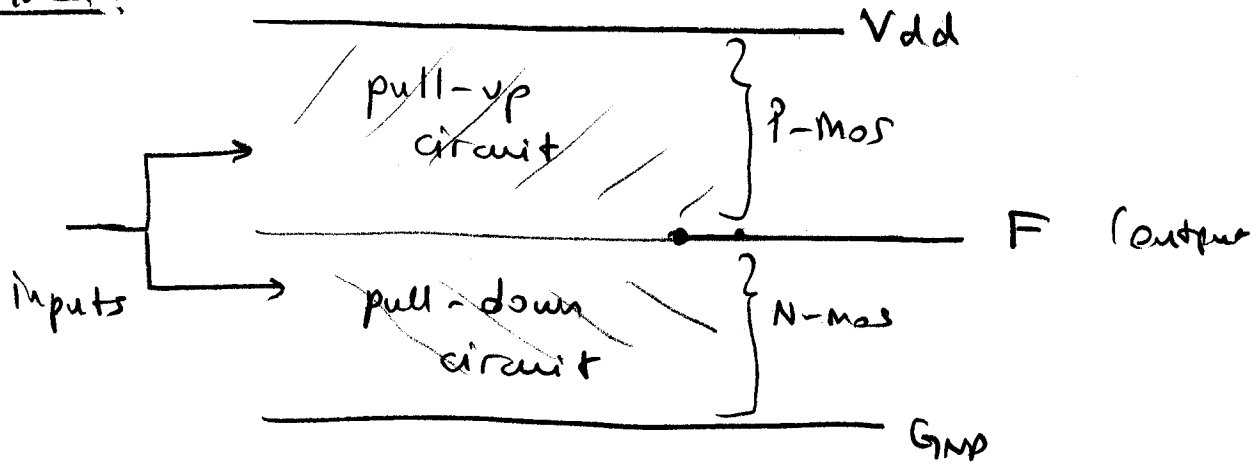
• P-MOS passes "1" well.





— Boolean functions can be implemented in a technology (CMOS) that uses P-MOS & N-MOS transistors.

Main idea:



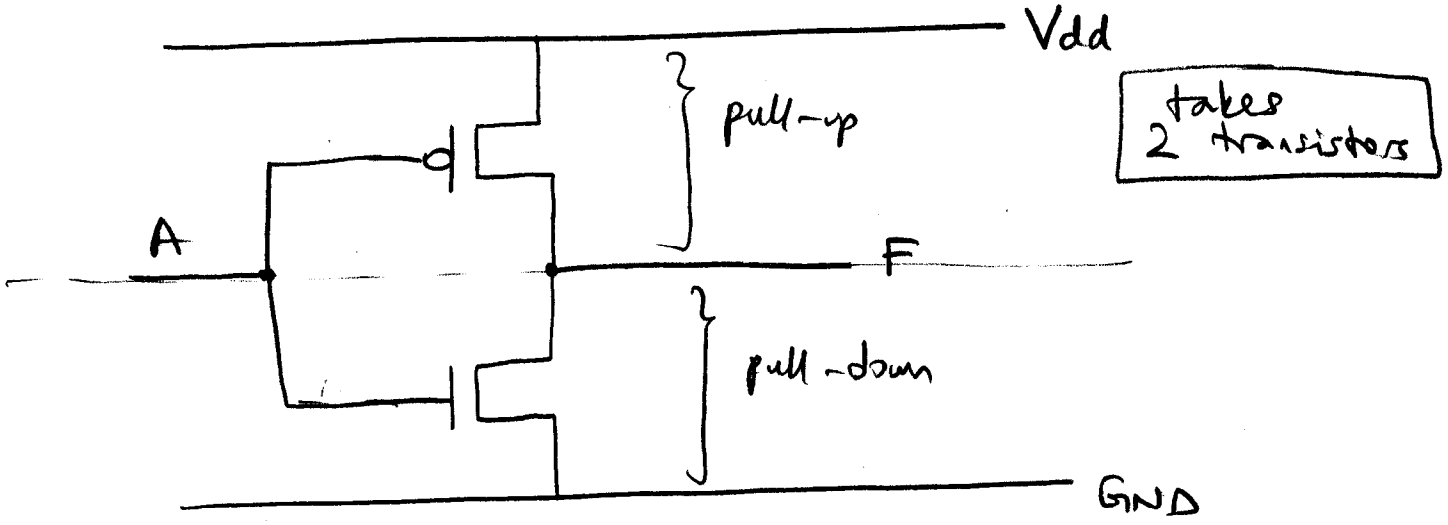
• Pull-up circuit implements the "1"s of F (the "1"s, i.e. ON-SETS or K-map of F.)

• Pull-down circuit implements the "0"s of F.

Example 1: Implement an inverter in CMOS.

Solution: $F = A'$ ⟶ implement in P-mos

$F' = A$ ⟶ implement in N-mos

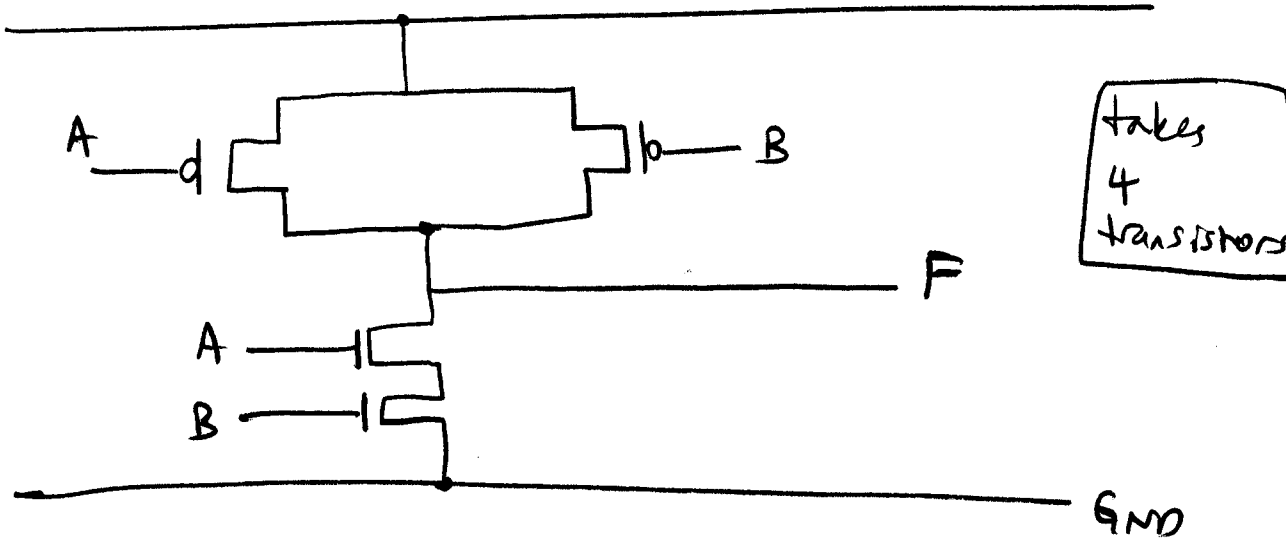


Example 2: Implement a NAND gate in CMOS.

Solution: $F = (A \cdot B)' = A' + B'$ ⟶ OR: Implement as a parallel network of switches

$F' = A \cdot B$

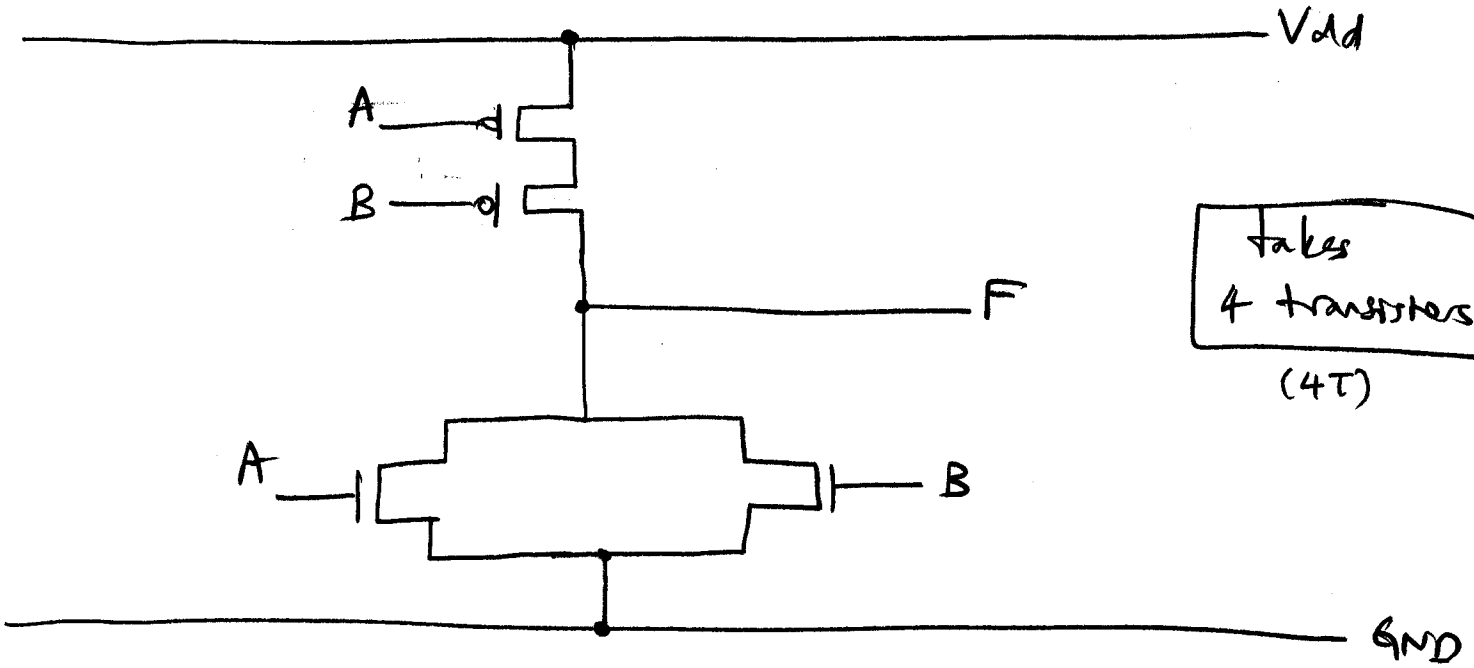
↑ AND: series network of switches



Example 3: Implement a NOR gate in CMOS.

Solution: $F = (A+B)' = A' \cdot B'$

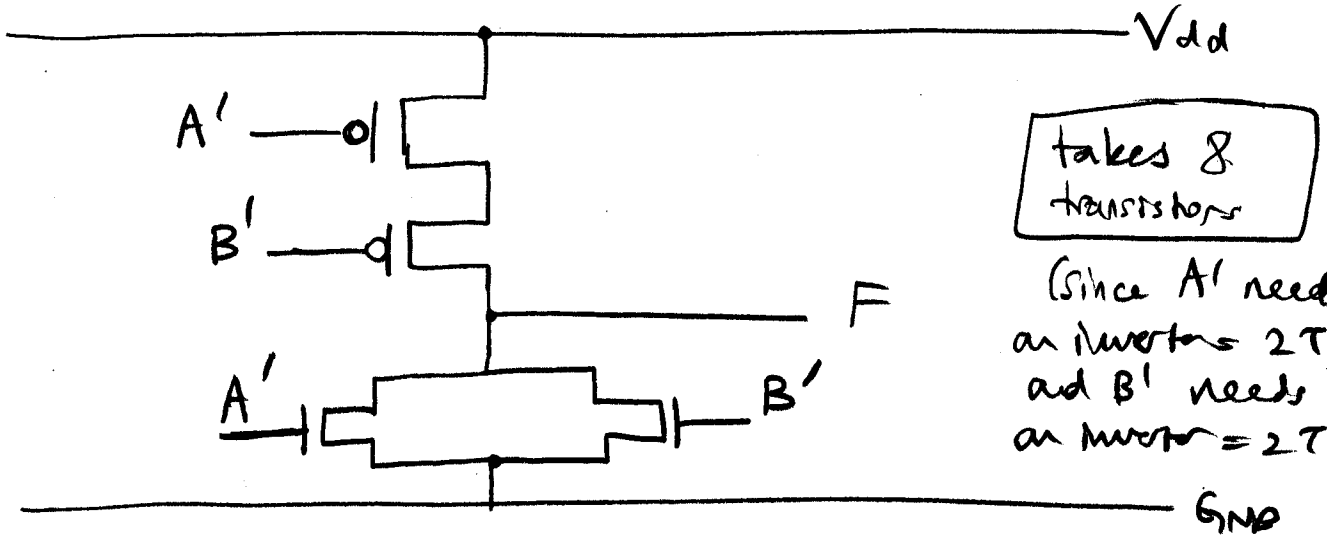
$F' = A+B$.



Example 4: Implement an AND gate in CMOS

Solution: $F = A \cdot B$

$F' = (A \cdot B)' = A' + B'$

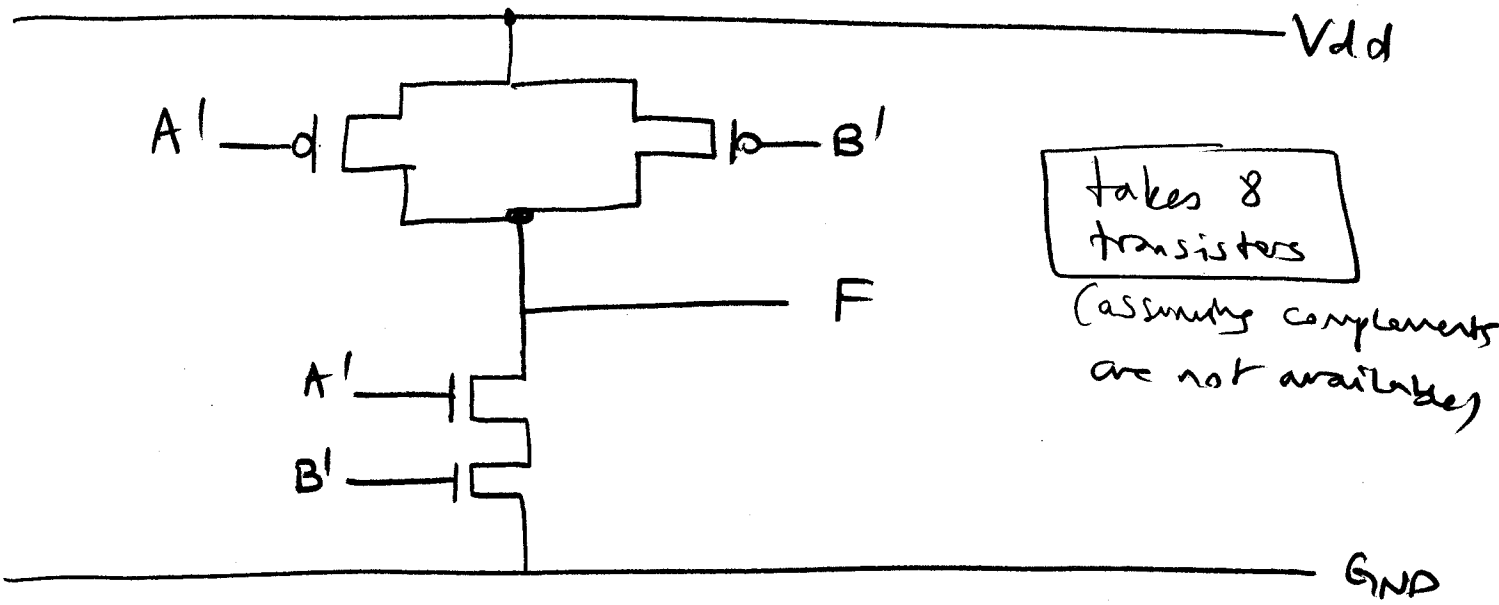


– The above is one of the reasons that NAND gates are more frequently used in CMOS implementations (i.e. NAND gate does not require any complements of inputs)

Example 5 : OR gate in CMOS

Solution : $F = A + B$

$F' = (A+B)' = A' \cdot B'$



Example 6 : Implement $F = A \cdot B + C$ in CMOS.

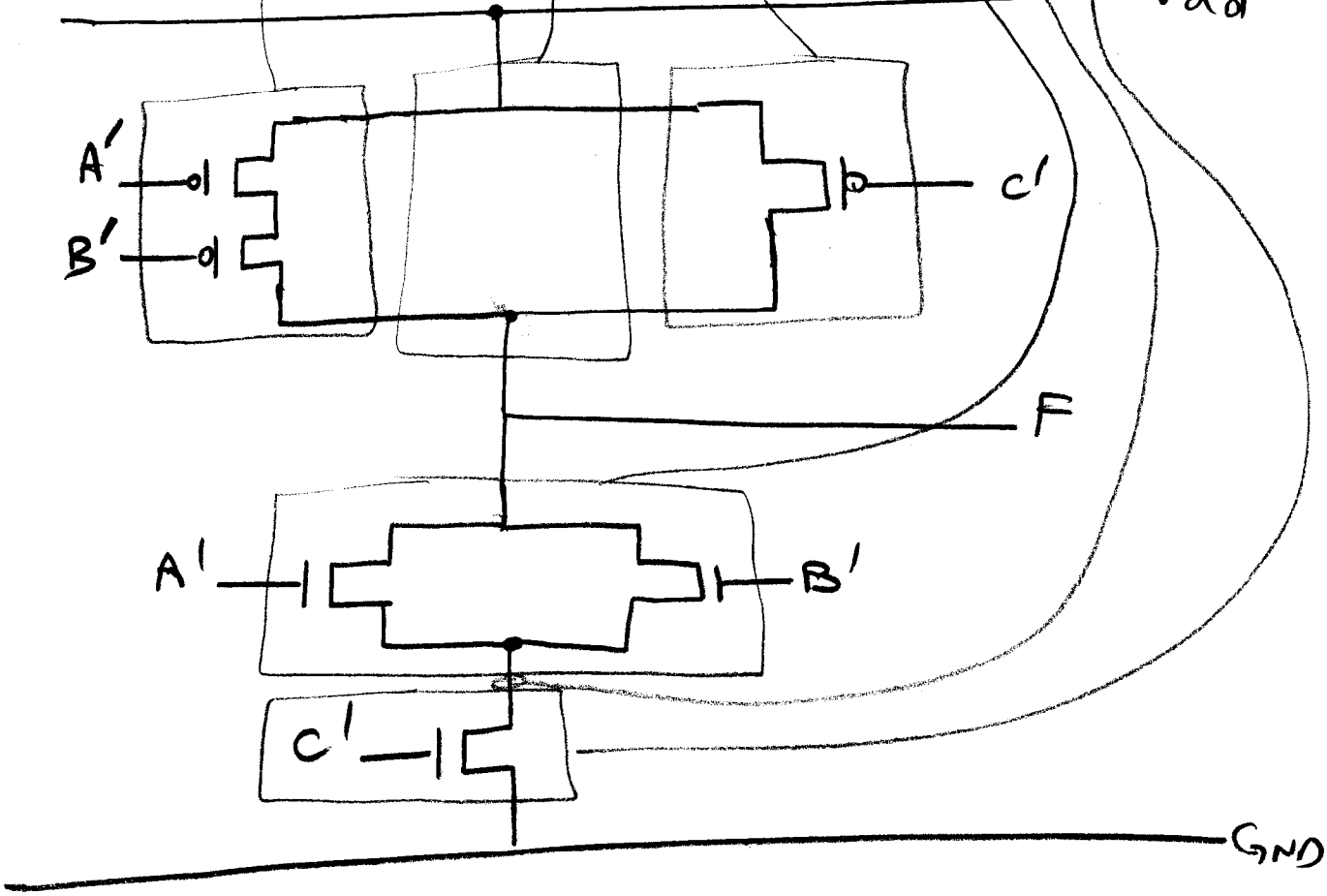
[This shows the general procedure to implement any Boolean function.]

Solution:

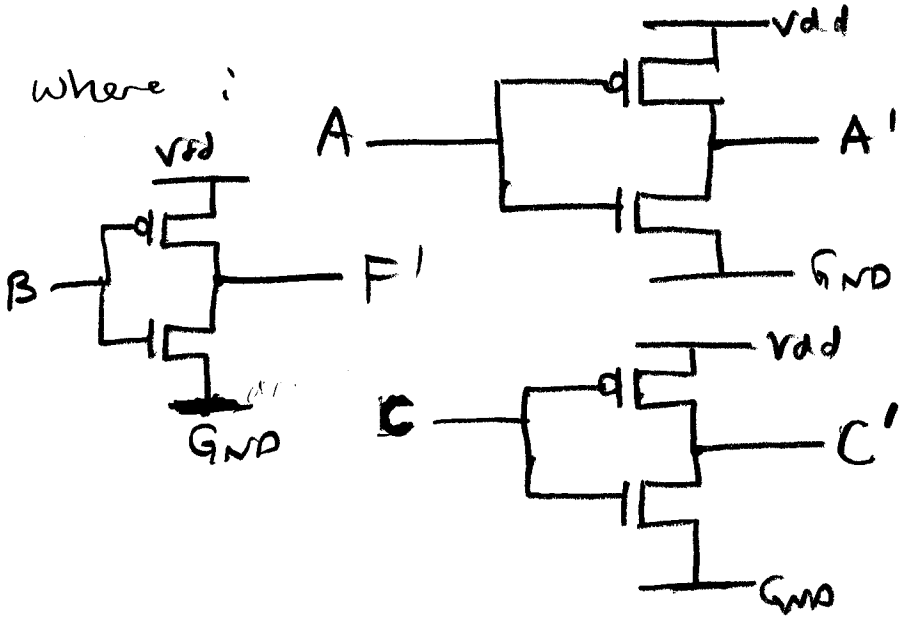
$F = A \cdot B + C$] pull-up

$F' = (A \cdot B + C)' = (A \cdot B)' \cdot C'$

$= (A' + B') \cdot C'$] pull-down



where :

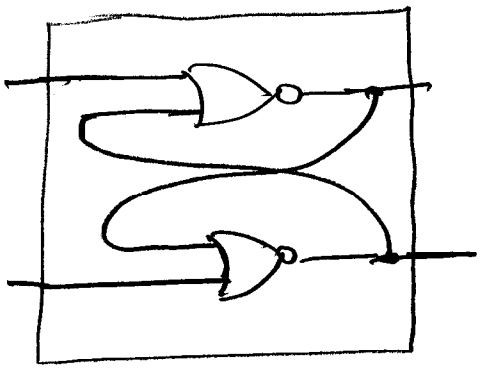


A total of 12 transistors needed for this F

So far: combinational logic. Now:

- Memory elements:

  - cross-coupled NOR gates: (basic latch):



- How many transistors used if implemented in CMOS?

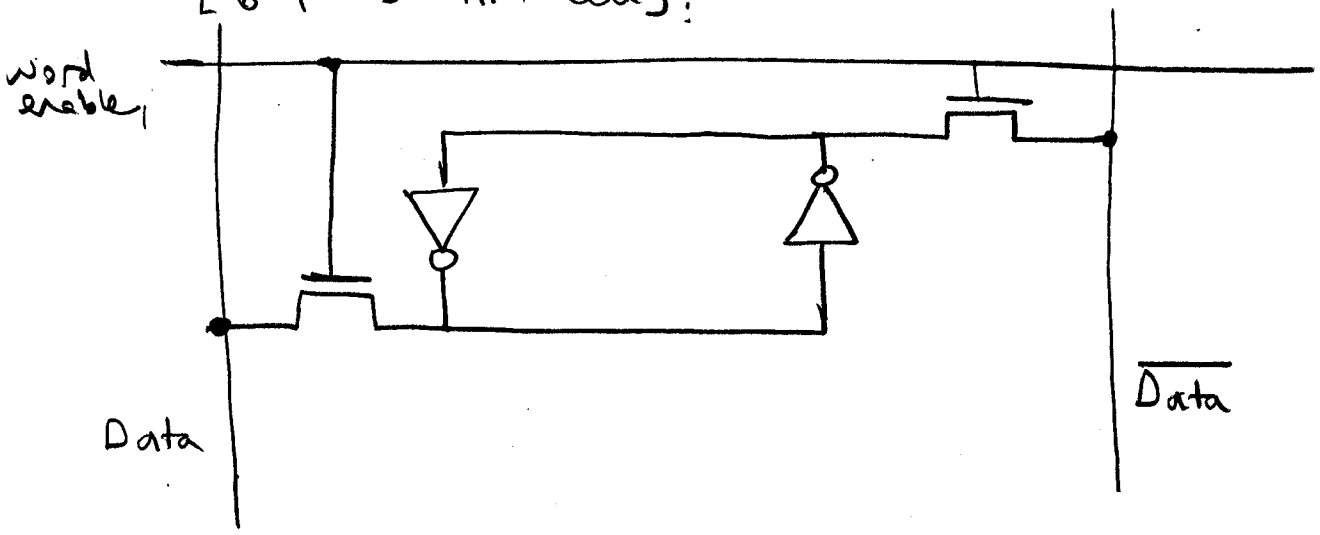
$2 \times (4) = \boxed{8\ T}$ (8 transistors)

↑ per NOR gate

- So far, we used memory elements in discrete components
- For, Integrated memories, need high density (i.e., a high # bits stored per area). - So, reduce the area of a "cell" (a storage unit to hold one bit).

↑ (dependent on the # transistors & the wiring complexity)

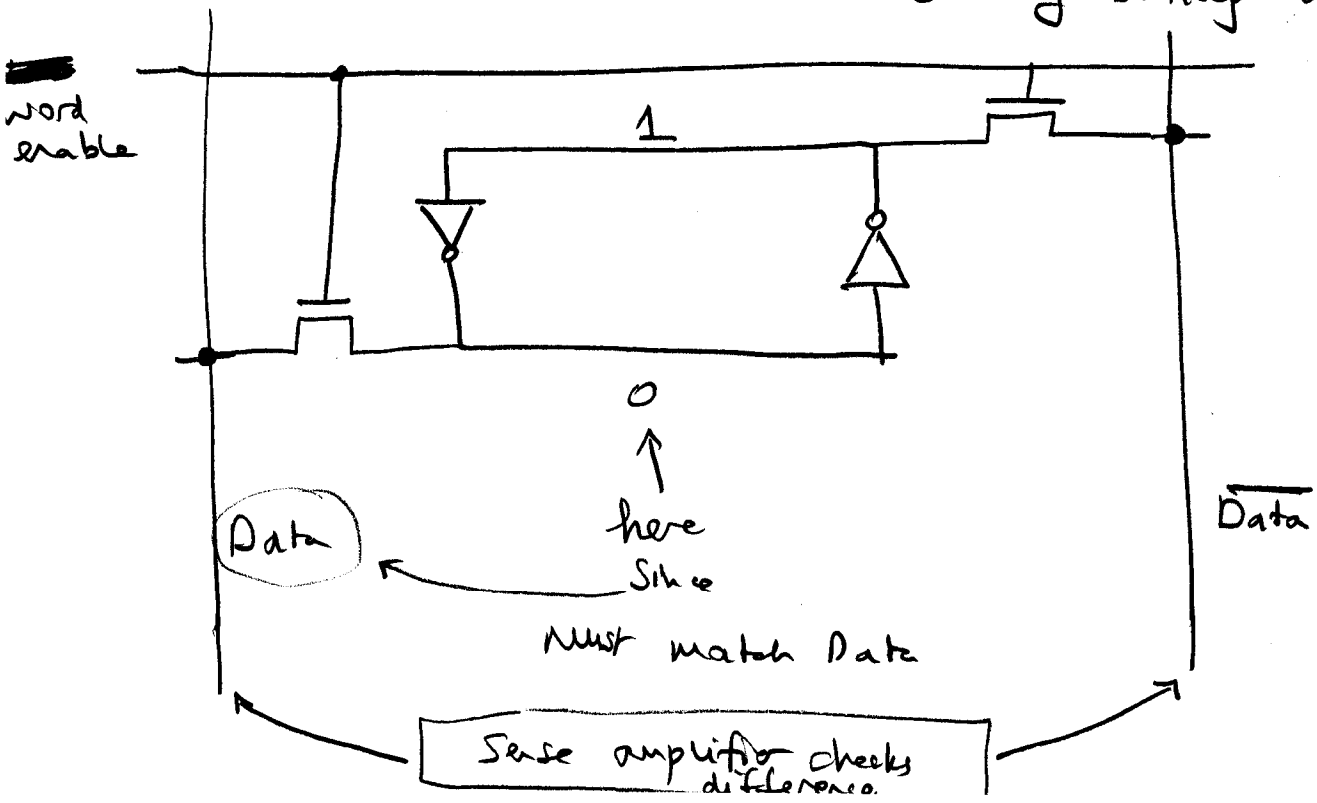
- A "memory cell" that uses 6T (6 transistors) [6T S-RAM cell]:



- This cell holds 1 bit.
- Cross-coupled inverters.
- (2T/inverter x 2 inverters) + 2 (n-mos T for access) = **6T**.

Operation:

Assume that the cell is originally storing a "0".



READ:

- Now to read from the cell,
  1. Both Data and $\overline{\text{Data}}$ are pre-charged to 1.
  2. Word Enable is put to high.
     - The "0" in the cell pulls the Data line to 0 (since N-MOS passes 0 well.)
  3. The sense amplifier senses the difference between Data and $\overline{\text{Data}}$.
     - If Data has a voltage (sufficiently less than) $<$ $\overline{\text{Data}}$, then a value of "0" is declared.
     - otherwise "1" is declared.

WRITE:

- To write to the cell: (e.g. write a "0".)
  1. Data = 0 and $\overline{\text{Data}}$ = 1 by external circuitry
  2. Word Enable = 1.
     - The external circuitry drives these values into the cell (0 to lower part of cell).

- How is the cell used?

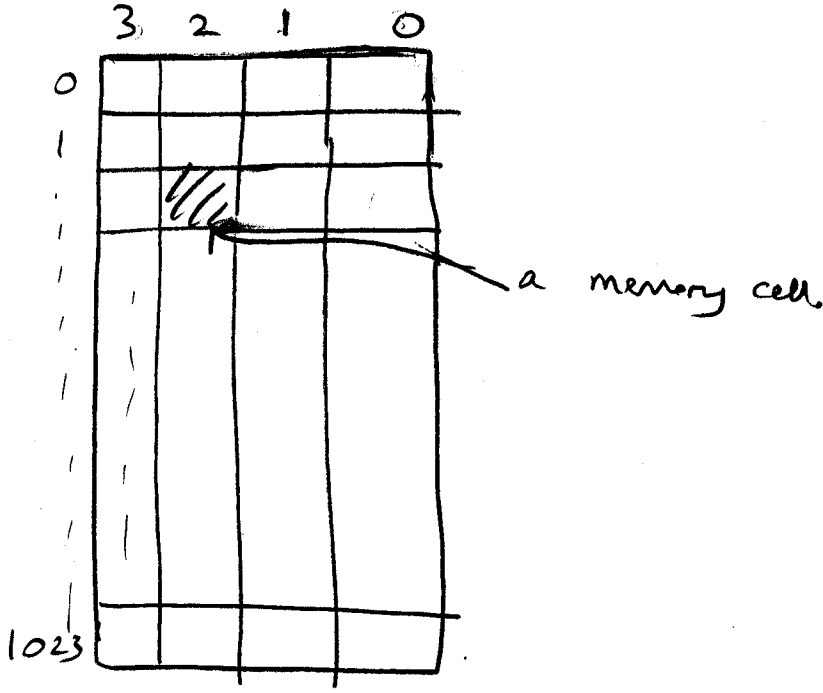
S-RAM : static RAM (× dynamic RAM — requires refresh / to leakage)

Means: will hold the contents as long as power is ON, with no refresh needed.

Ex)

1024 × 4 S-RAM.

1024: # of words; 4: # of bits per word.

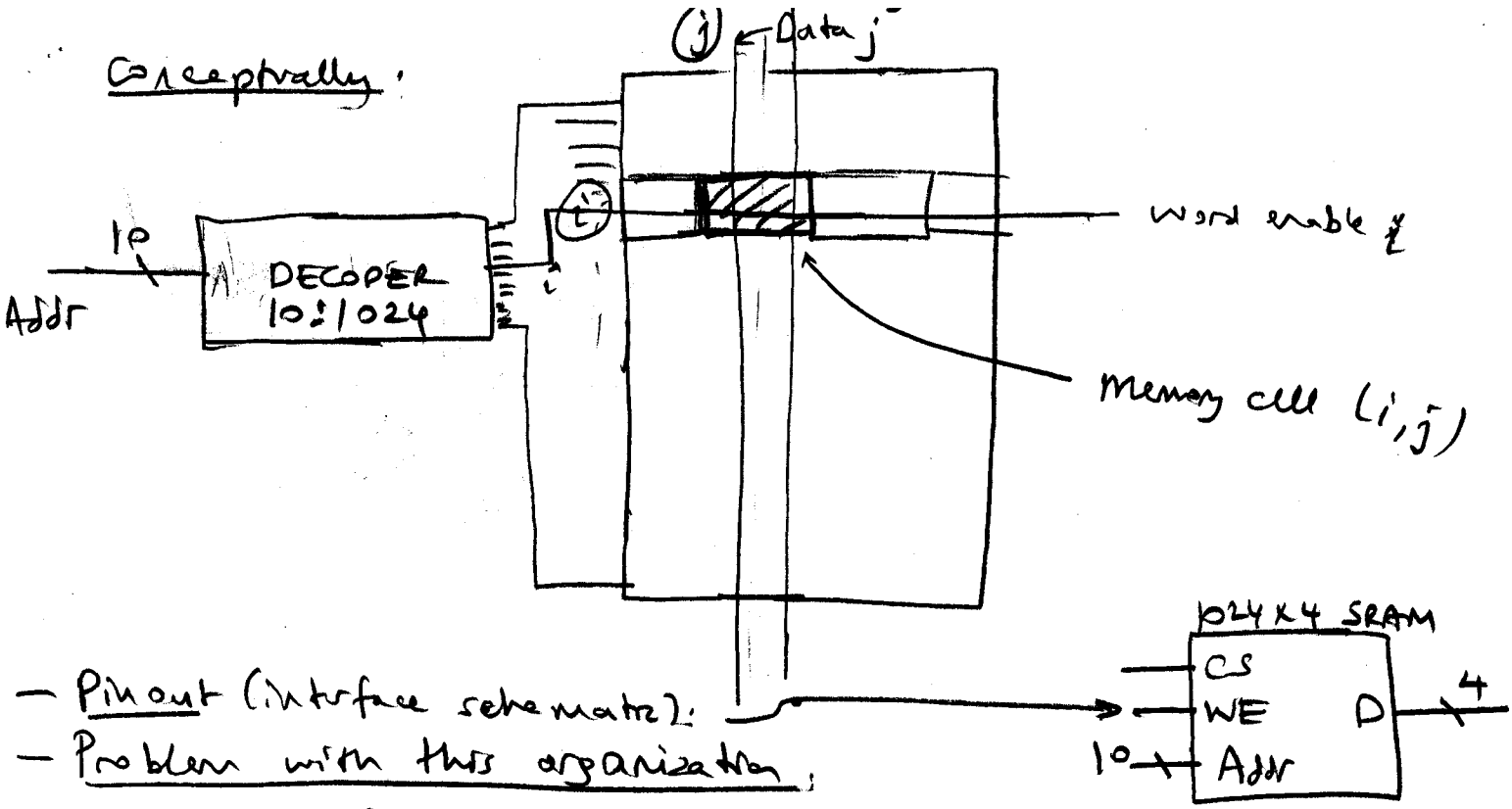


Conceptual organization:

($2^{10} \cdot 2^2 = \boxed{2^{12}}$ memory cells required here.)

- Idea: Address one of the 1024 (by a 10-bit address) and read out (or write in) an entire word.

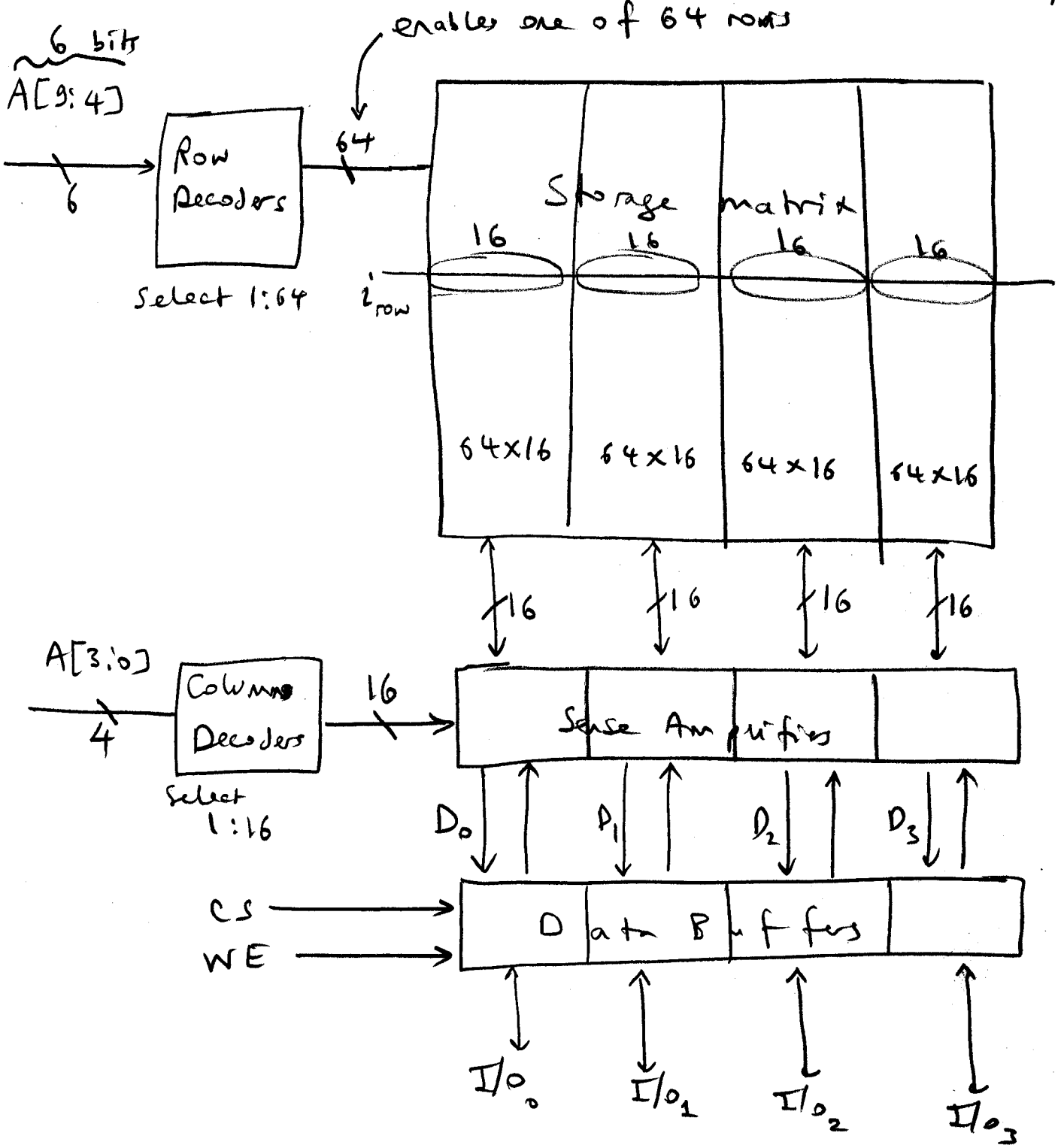
Conceptually:



- Pinout (interface schematic):
- Problem with this organization:
  - Long Data (and $\overline{Data}$) wires:
    - Increase delay (due to wire length)

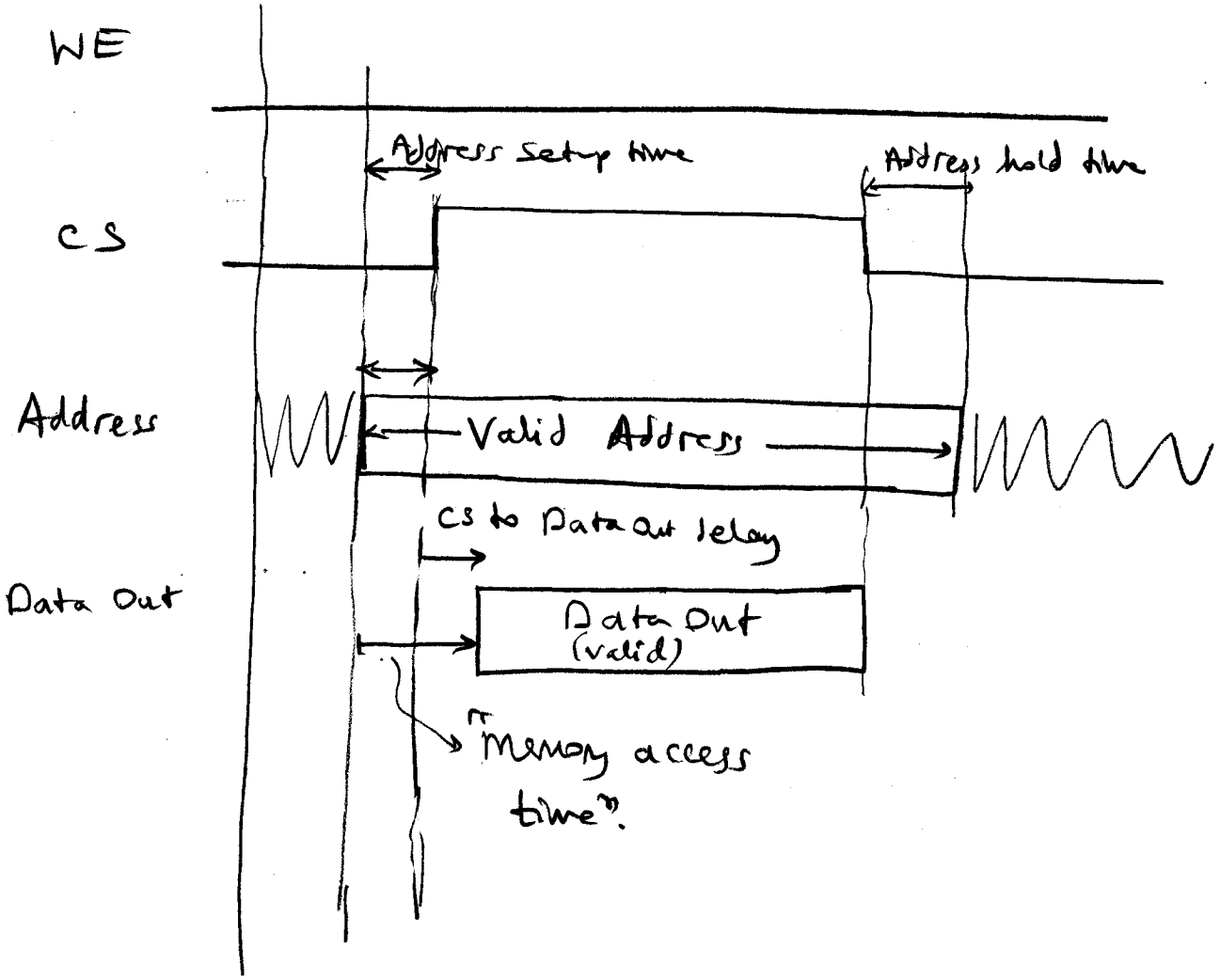
Better internal organization uses a "storage matrix" that is square: $1024 \times 4 = 64 \times 64$

- This organization is shown on the next page.



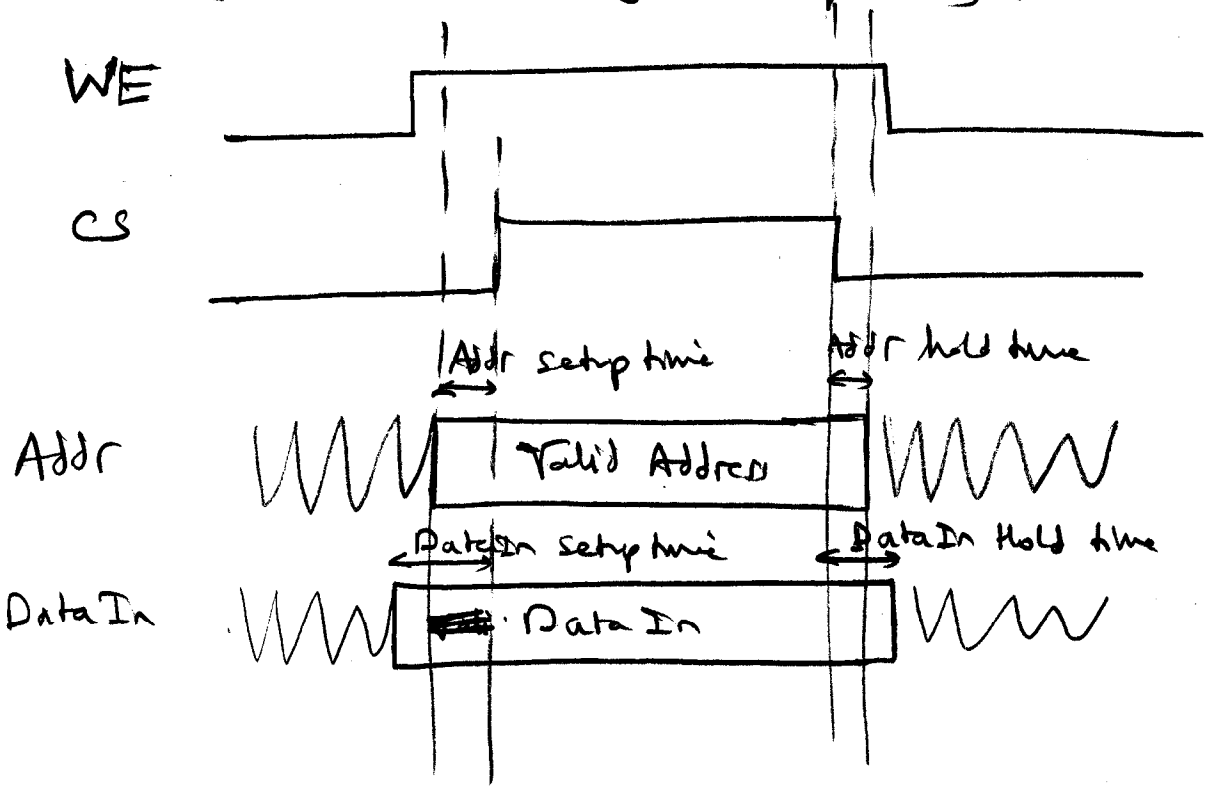
- Note that the 4 bits in a RAM word do not sit next to each other in the storage matrix. (i.e. the physical storage location is different from that in the conceptual organization).

SRAM Read cycle sequencing



- Keys: the address has to settle to the correct values a setup time before the CS (or WE) goes high. (otherwise will read out junk.)
- WE must be kept low the entire time CS is high. (for reading)

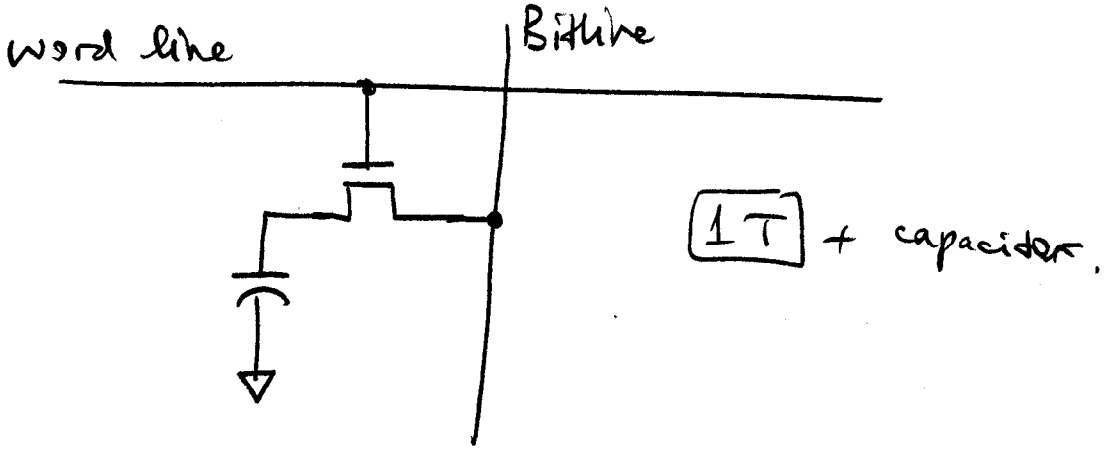
SRAM write cycle sequencing:



Keys:
- Both the Addr and the DataIn have to have settled to their correct values, respective setup times, before the CS goes high.
- Note that write cycle begins when both WE and CS are high.

Dynamic RAM:

- Smaller space per cell ⇒ higher integration density
- slower than SRAM.
- needs refreshing (≈ 1 - 4 ms or so.)



- capacitor leaks the charge over time (≈ milliseconds) So need to read out contents & write them back in. ("Refresh operation")