BIRZEIT UNIVERSITY

DEPARTMENT OF COMPUTER SYSTEM ENGINEERING

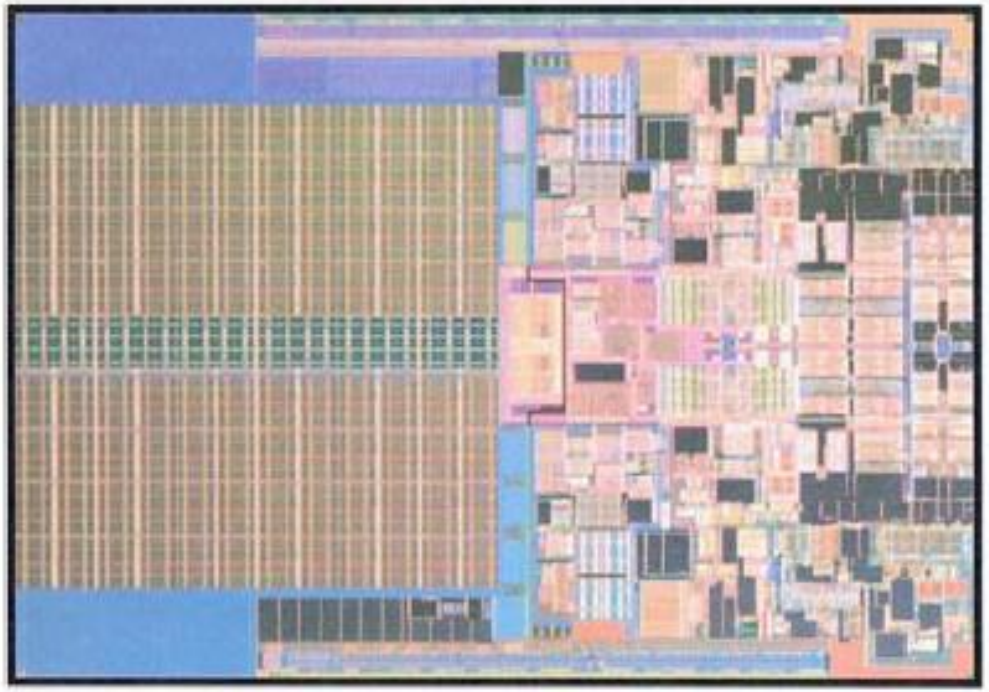
Digital Integrated Circuits - ENCS333

Dr. Khader Mohammad

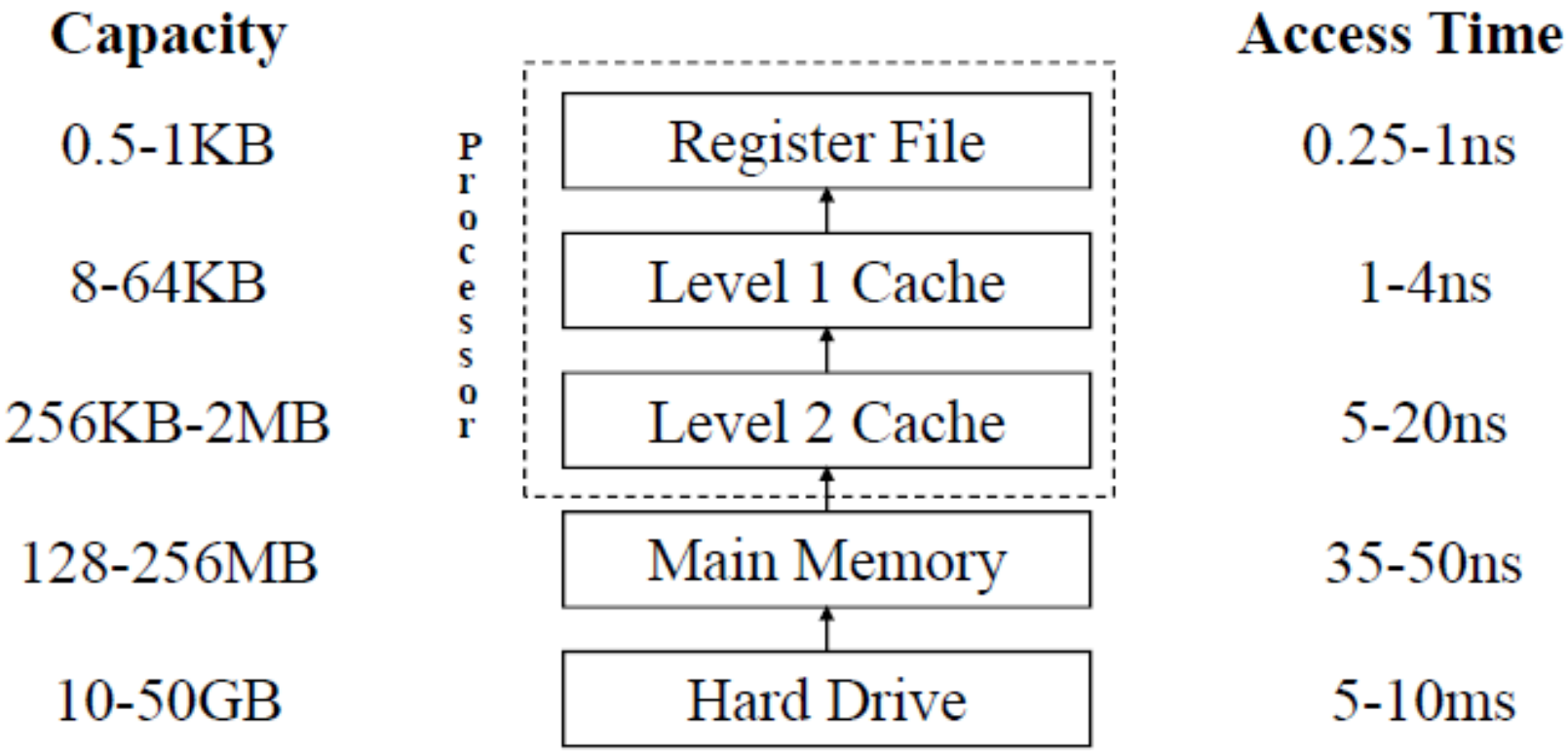
Lecture #15 Memory

Why Memory?

Intel 45nm Core 2

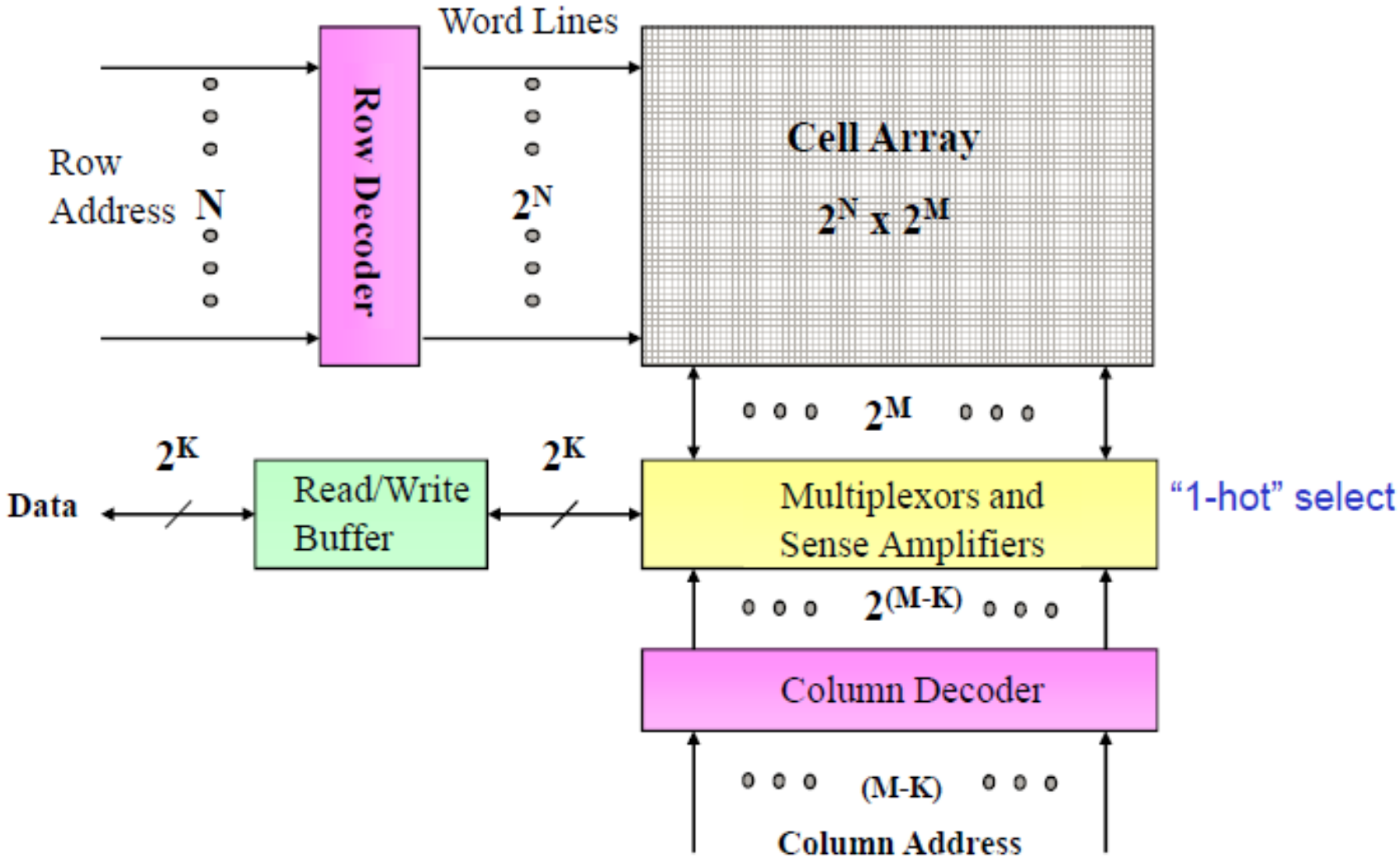


Memory Hierarchy



Memory hierarchy gives the appearance of large capacity and fast access time.

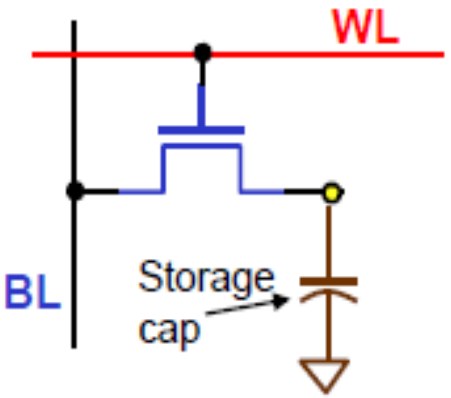
Functional Block Diagram



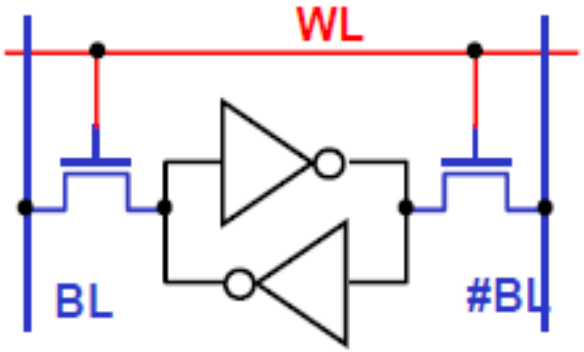
Memory Cell Types

- Schematic of 1-T DRAM cell, 6T dual ended SRAM cell

1-transistor DRAM



6-transistor SRAM



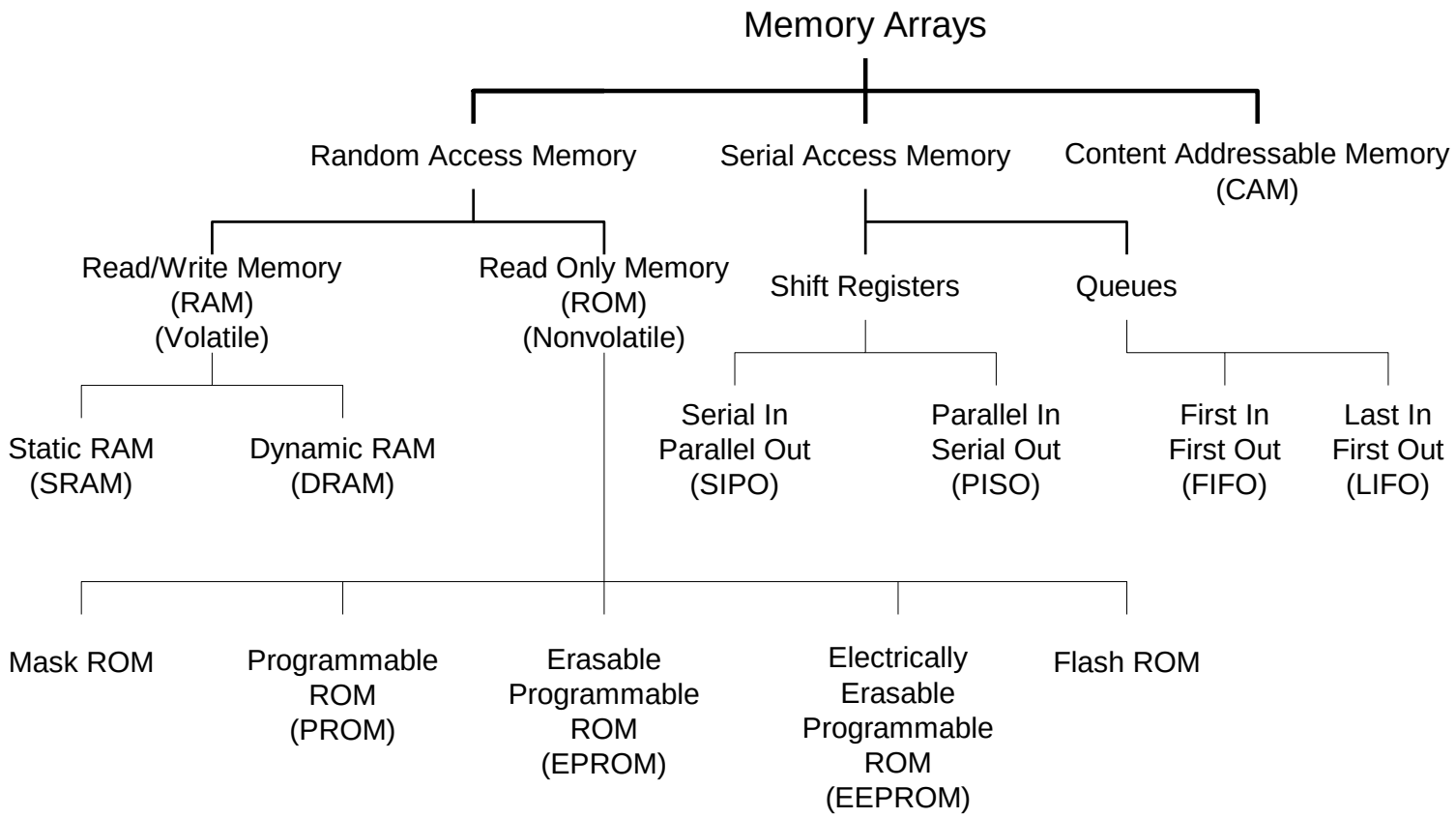
- Industry standard DRAM cell
- Smallest area per bit
- Explicit storage capacitor
- Destructive READ

- Industry standard SRAM cell
- Used for *FAST* static arrays
- Cross-coupled inverters
- Non-destructive READ with proper stability analysis

Semiconductor Memory Classification

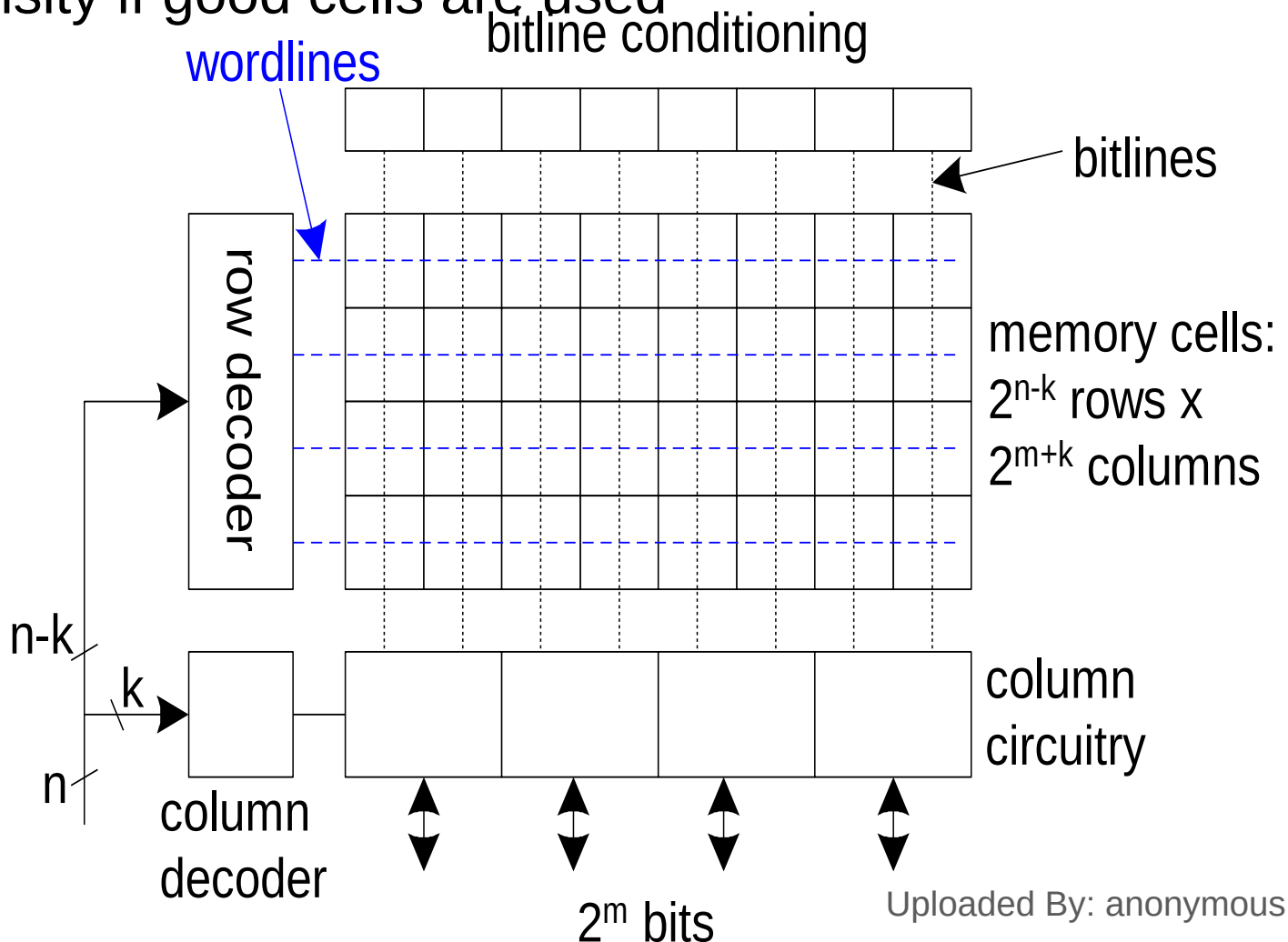
Read-Write Memory		Non-Volatile Read-Write Memory	Read-Only Memory
Random Access	**Non-Random Access**	EPROM	Mask-Programmed
SRAM	FIFO	E^2PROM	Programmable (PROM)
DRAM	LIFO	FLASH	
	Shift Register		
	CAM		

Memory Arrays



Array Architecture

- 2^n *words* of 2^m *bits* each
- If n >> m, fold by 2^k into fewer *rows* of more *columns*
- Good regularity – easy to design
- Very high density if good cells are used



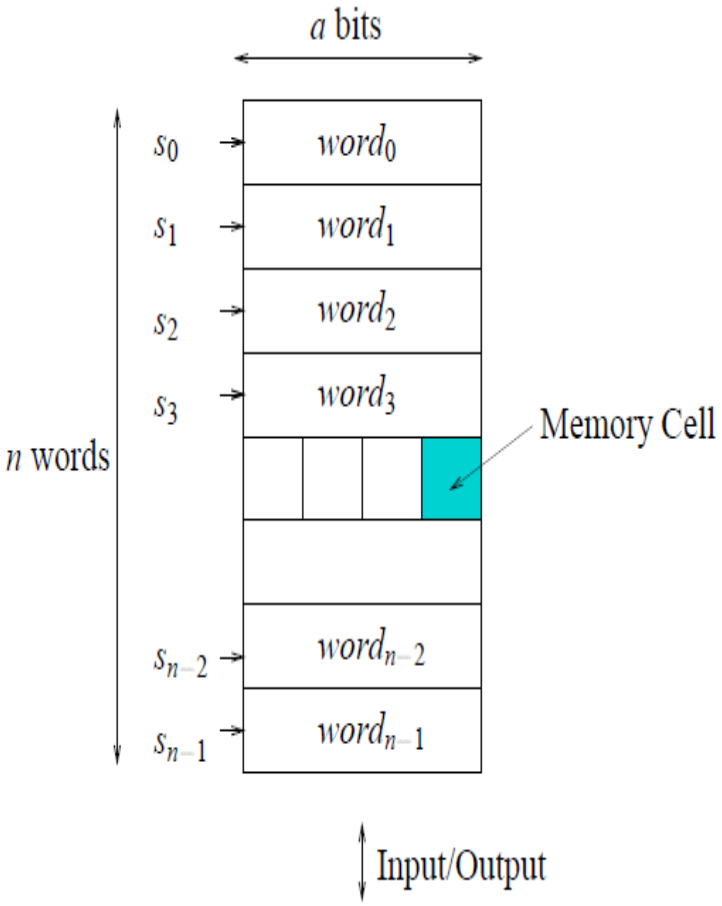
Relevant Parameter of Memory Cell

- **Read access time** - the time between when a word is requested and when it is available
- **Write access time** - the time between when a word is requested to be written and when it is actually written
- **Read/write access time** - the minimum time between successive read or write operations

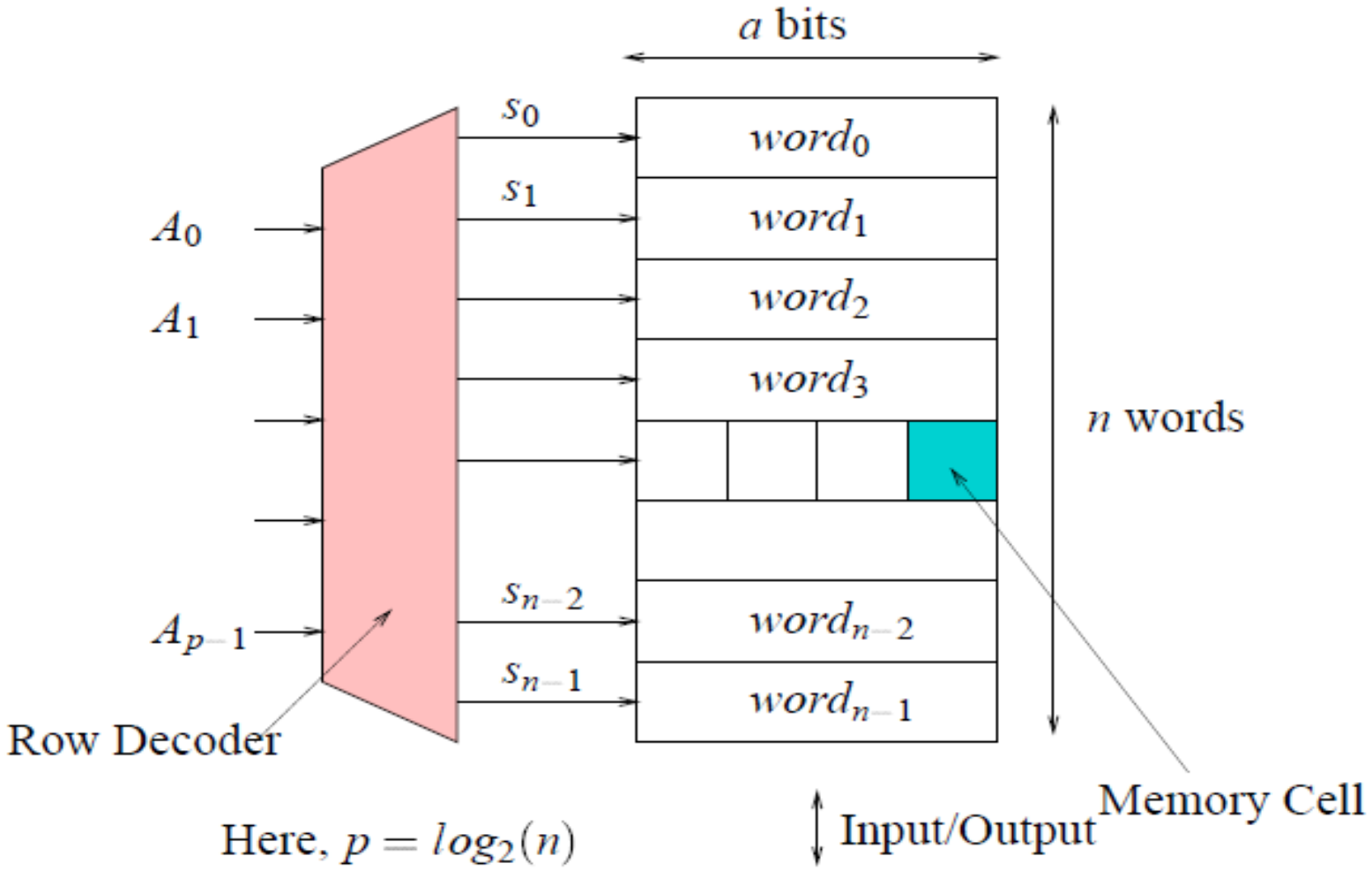
- Sometimes memories have multiple **ports**. In this case the same data can be accessed by several "customers" at once.
  - The read cycle may be staggered in time so that a read can occur just after a write. Such a multi-port memory is significantly challenging to design
  - Several ports may read the memory at once. Read access times drop with each additional port attempting to read a memory word
  - A single write is allowed to any given word at a time.

Relevant Parameter of Memory Cell

- Several writes **to different words** are allowed. This poses significant problems with bit and word-line cross-talk
- Multi-port memories are usually static
- It is hard to design multi-port memories witl more than 2 ports.
- Routing becomes harder as the number of port increases
- Multi-port memories are used in register file for RISC microprocessors, since very high band width access to registers is required in such de signs



- Memory is often organized in a **hierarchy**. So if a item of data is not found on level i, then it we search for it in level $i+1$ memory. Usually higher levels of memory are slower and larger. The last level of memory storage is usually a hard disk in such an arrangement
- What does the inside of the n word memory look like (where a word is a bits)?
  - Simple-minded solution - Arrange the words **linearly**

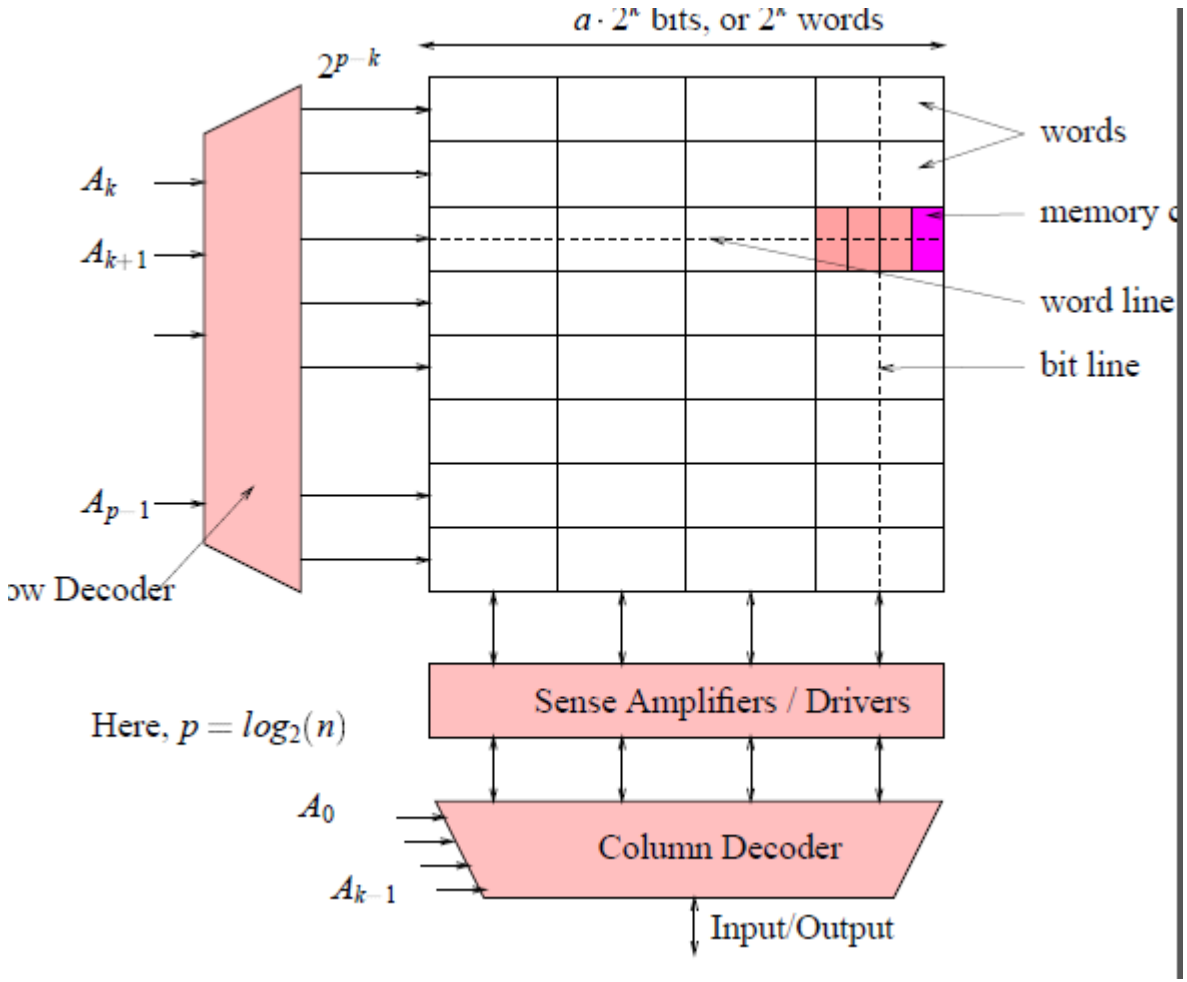


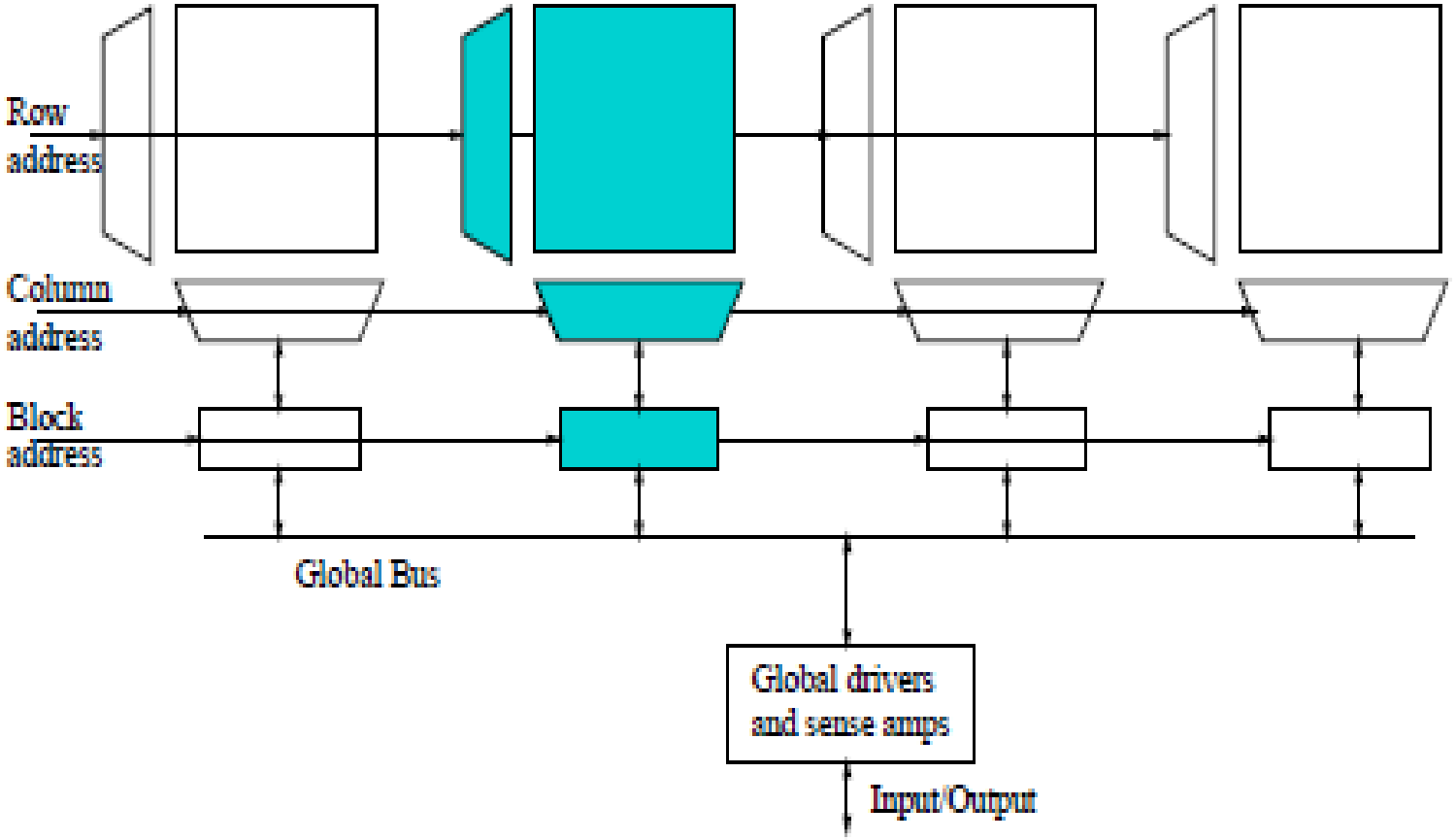
- But this requires n select lines, and n is large.
- This motivates the use of a **row decoder**

- So then the number of **address bits** is $p = log_2(n)$. Hence the number of input pins to the memory drops exponentially. The decoder ensures that only one of the select lines s_i is active at a time.
- However, with this approach, the memory is tall and thin. It is n words by a bits, with $n >> a$.
- For a memory with $n = 2^{20}$ and $a = 8$, the aspect ratio is roughly $2^{20}/2^3 = 2^{17} \simeq 128,000$.
- This is a problem for two reasons:
  * As a general rule, for the same circuit area, roughly square ICs result in best utilization of wafer area, are easy to manipulate and are faster since the longest wire length is smaller
  * The thin and tall memory cell has very long bit-lines, resulting in highly capacitive wires and large output delays.

So we want to make the memory more or less square.

To do this we use **row and column decoders** as shown below





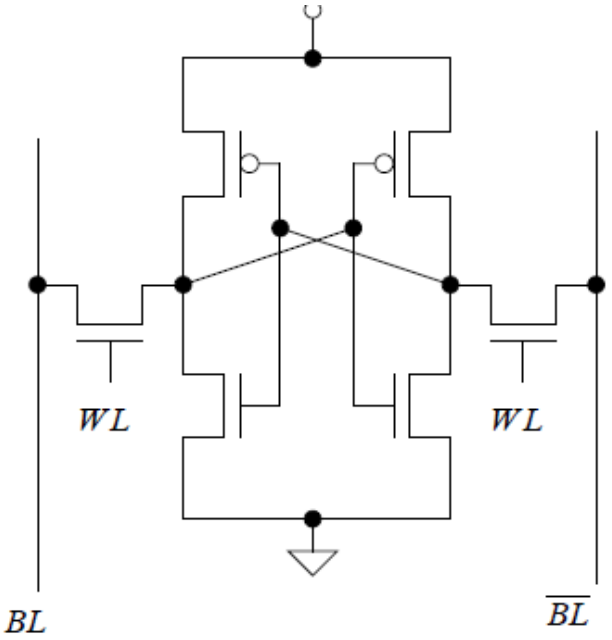
- This block-style design ensures that the longest wire in the design is smaller than in a monolithic memory. Hence it is faster.

Static RAM (SRAM)

- Data, once written, need not be refreshed
- Recall the definition of bit and word lines
- SRAM cells are among the larger memory cells
- A SRAM is utilized in applications where
  - Memory size is smaller but fast and clean signals are desired.
  - Typical application is a register file
- Generically, a SRAM cell looks like:

SRAM

- The SRAM cell operates as follows:
  - **Writing data:** Place the desired data (A) on the BL line, and $\overline{A}$ on $\overline{BL}$. Then assert WL. This causes the node a to attain the A value, and the node b attains the $\overline{A}$ value. WL is de-asserted now. The cell is in steady state.
  - **Reading data:** First we precharge both BL and $\overline{BL}$. Now we assert WL. Depending on the state of a and b, either BL or $\overline{BL}$ is pulled low, the other stays high.



- The load R can be implemented in several ways:
  - If it is implemented as a PMOS device as shown below, we get a 6-T SRAM cell (6-T stands for 6-transistor).

Random Access Memories (RAM)

- **STATIC (SRAM)**

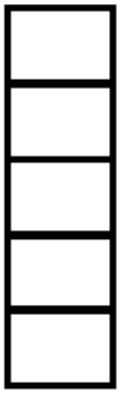
  Data stored as long as supply is applied
  Larger (6 transistors/cell)
  Fast
  Differential (usually)

- **DYNAMIC (DRAM)**

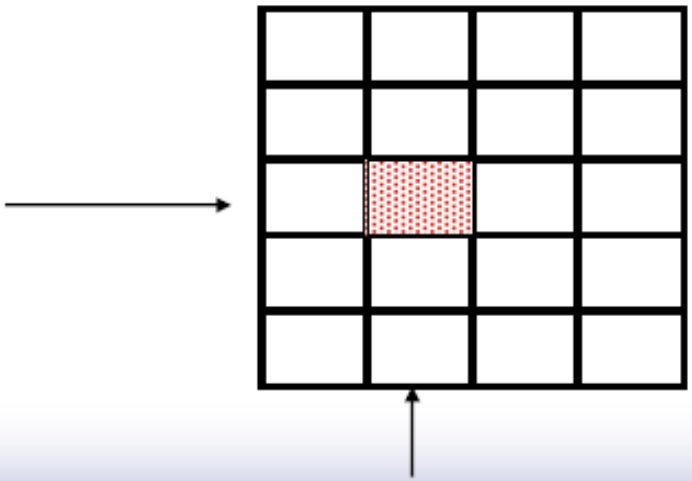
  Periodic refresh required
  Smaller (1-3 transistors/cell)
  Slower
  Single Ended

Random Access Chip Architecture

- Conceptual: linear array
  - Each box holds some data
  - But this does not lead to a nice layout shape
  - Too long and skinny



- Create a 2-D array
  - Decode Row and Column address to get data



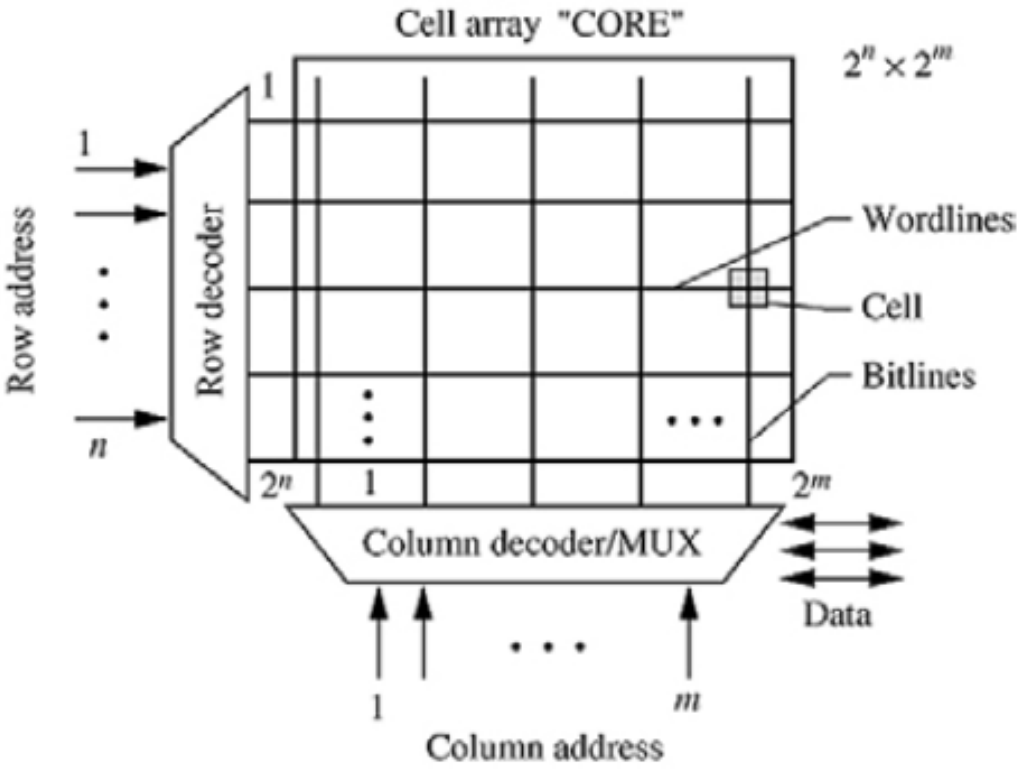
Basic Memory Array

CORE:
- keep square within a 2:1 ratio
- rows are **word lines**
- columns are **bit lines**
- data in and out on columns

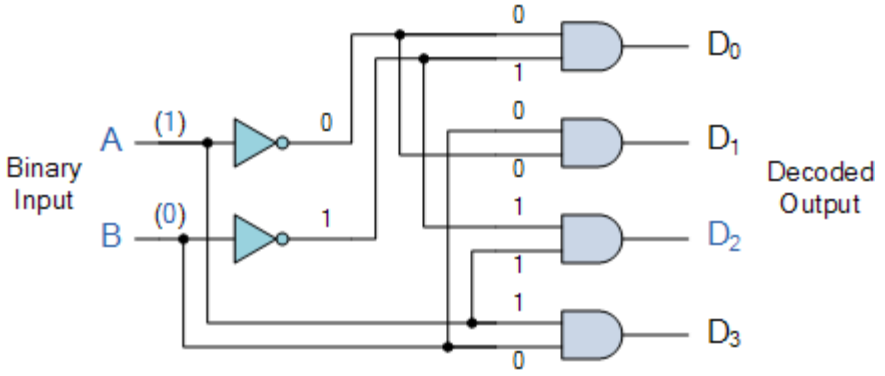
DECODERS:
- needed to reduce total number of pins; N+M address lines for 2^{N+M} bits of storage

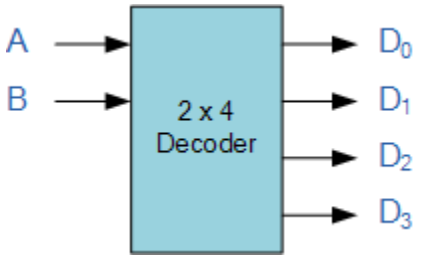
Ex: if N+M=20 → 2^{20}= 1Mb

MULTIPLEXING:
- used to select one or more columns for input or output of data



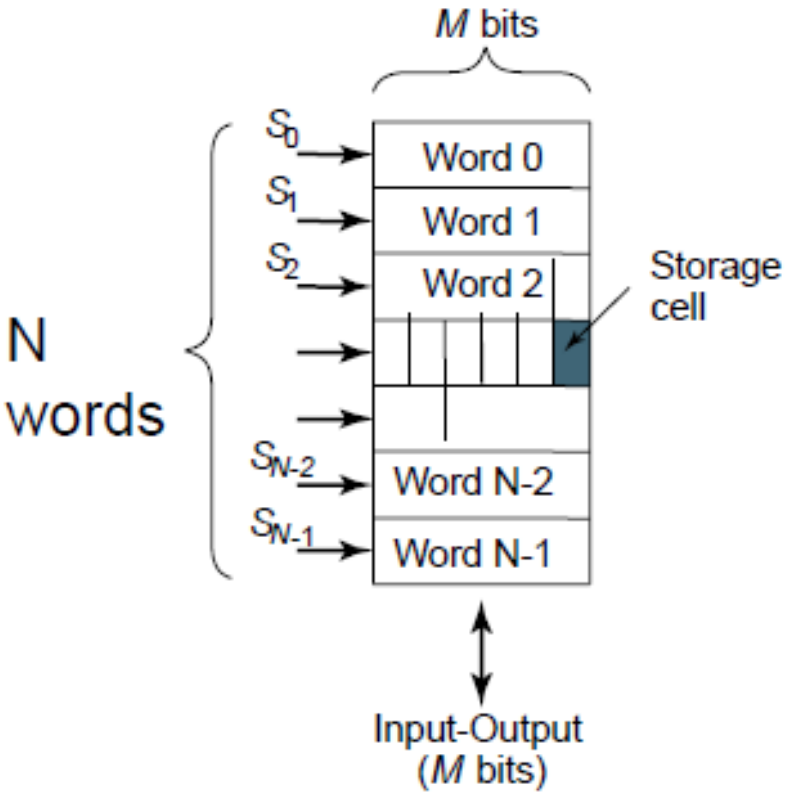
A 2-to-4 Binary Decoders



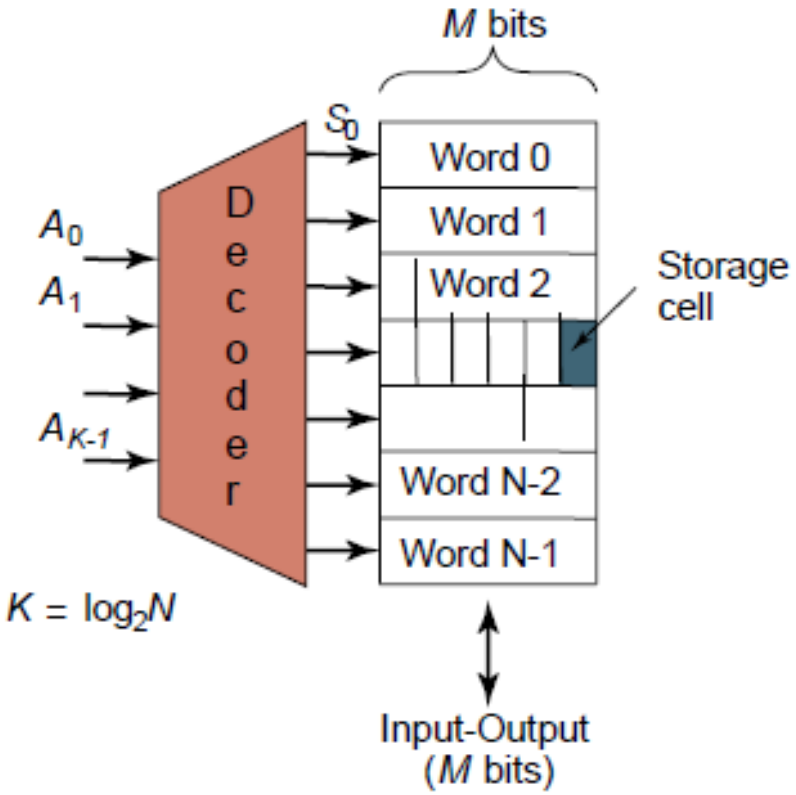


A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Memory Architecture: Decoders



Intuitive architecture for N x M memory
Too many select signals:
N words == N select signals



Decoder reduces the number of select signals

$K = log_2 N$

Row Decoders

Collection of 2^M complex logic gates
Organized in regular and dense fashion

(N)AND Decoder

$$WL_0 = \overline{A_0}\,\overline{A_1}\,\overline{A_2}\,\overline{A_3}\,\overline{A_4}\,\overline{A_5}\,\overline{A_6}\,\overline{A_7}\,\overline{A_8}\,\overline{A_9}$$

$$WL_{511} = A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 \overline{A_9}$$

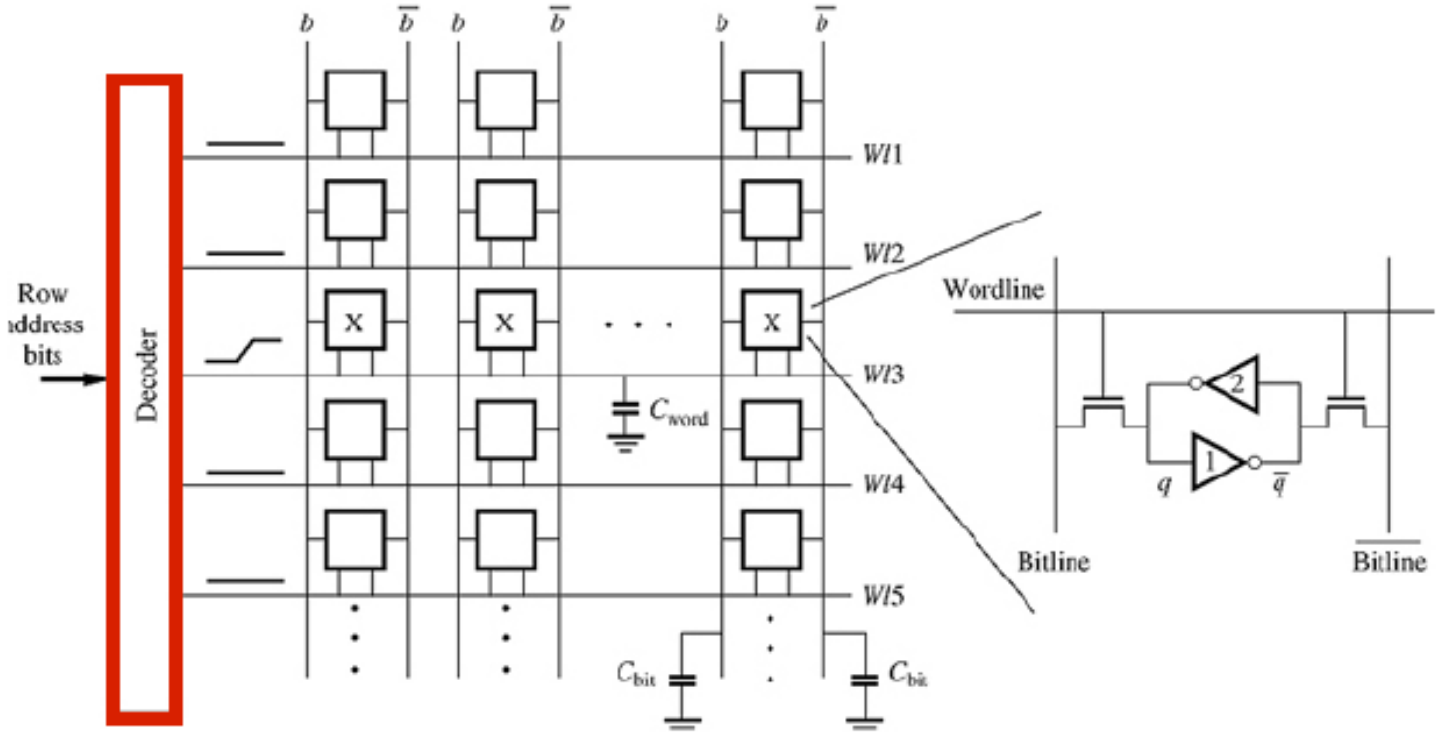
$WL_{16} =$

NOR Decoder

$$WL_0 = \overline{A_0 + A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8 + A_9}$$

$$WL_{511} = \overline{\overline{A_0} + \overline{A_1} + \overline{A_2} + \overline{A_3} + \overline{A_4} + \overline{A_5} + \overline{A_6} + \overline{A_7} + \overline{A_8} + A_9}$$

Decoder Design Example

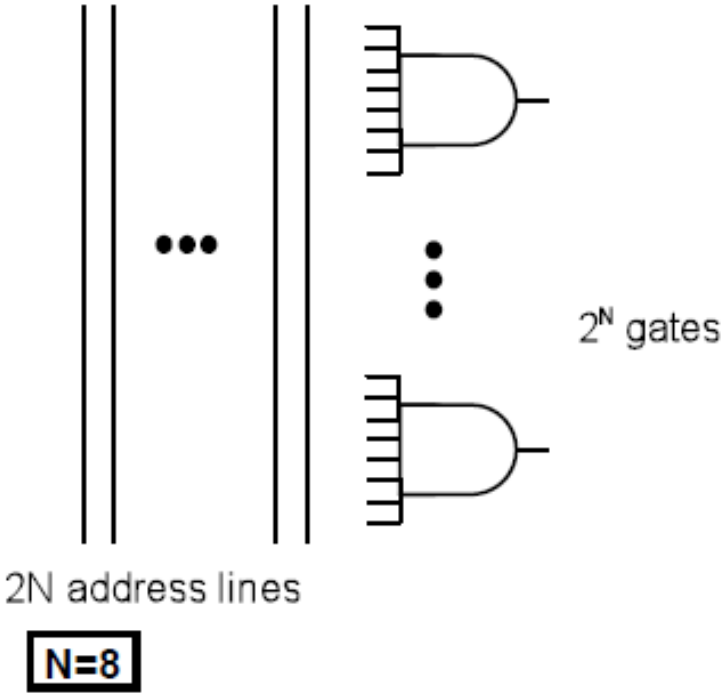


- Look at decoder for 256x256 memory block (8KBytes)

Problem Setup

- Goal: Build fastest possible decoder with static CMOS logic

- What we know
  - Basically need 256 AND gates, each one of them drives one word line



Problem Setup (1)

- Each word line has 256 cells connected to it
- Total output load is $256*C_{cell} + C_{wire}$
- Assume that decoder input capacitance is $C_{address}=4*C_{cell}$
- Each address drives $2^8/2$ AND gates
  - A0 drives ½ of the gates, A0_b the other ½ of the gates
- Neglecting C_{wire}, the fan-out on each one of the 16 address wires is:

$$F = \Pi B.\frac{C_{load}}{C_{in}} = 128.\frac{256C_{cell}}{4C_{cell}} = 2^{13}$$

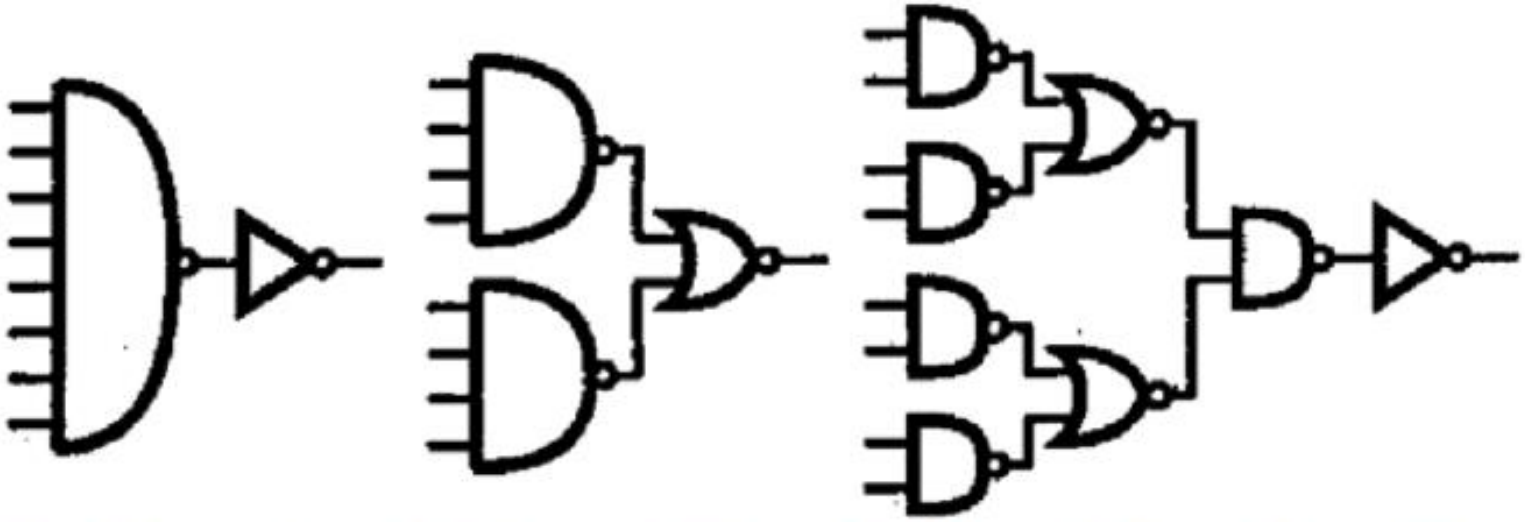
- Extra ..

Decoder Fan-Out

- ΠB.F at least 2^{13} means that we will want to use more than $\log_4(2^{13}) = 6.5$ stages to implement the AND8

- Need many stages anyways
  - So what is the best way to implement the AND gate?
  - Will see next that it's the one with the most stages and least complicated gates

8-Input AND

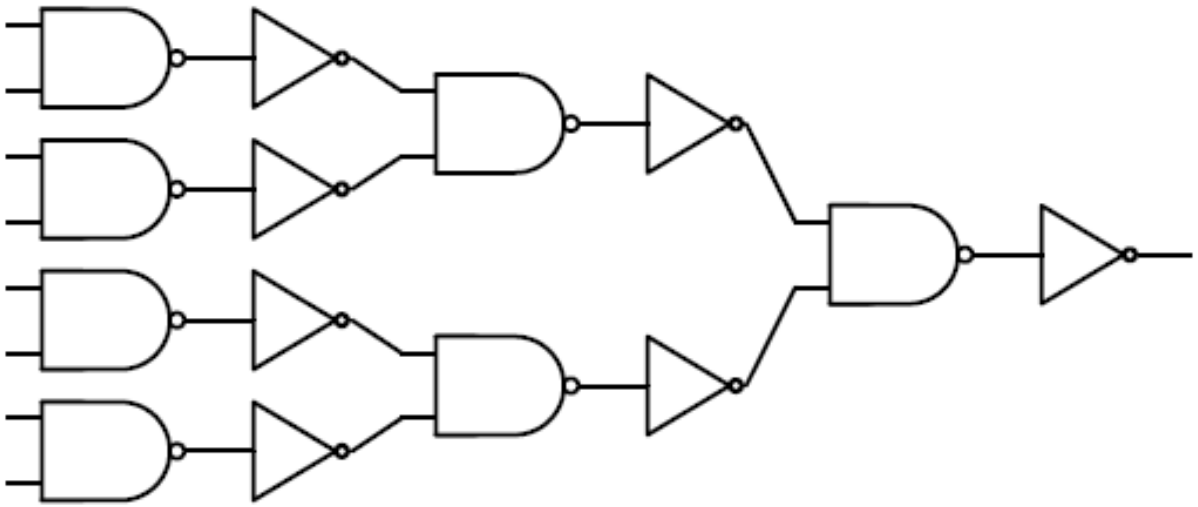


LE=10/3 1
ΠLE = 10/3
P = 8 + 1

LE=2 5/3
ΠLE = 10/3
P = 4 + 2

LE=4/3 5/3 4/3 1
ΠLE = 80/27
P = 2 + 2 + 2 + 1

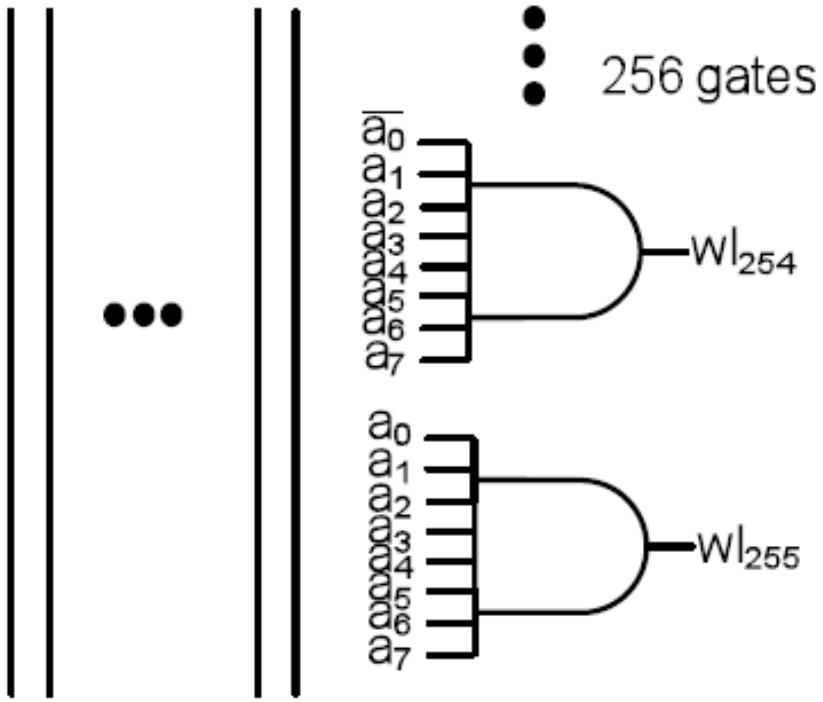
8-Input AND



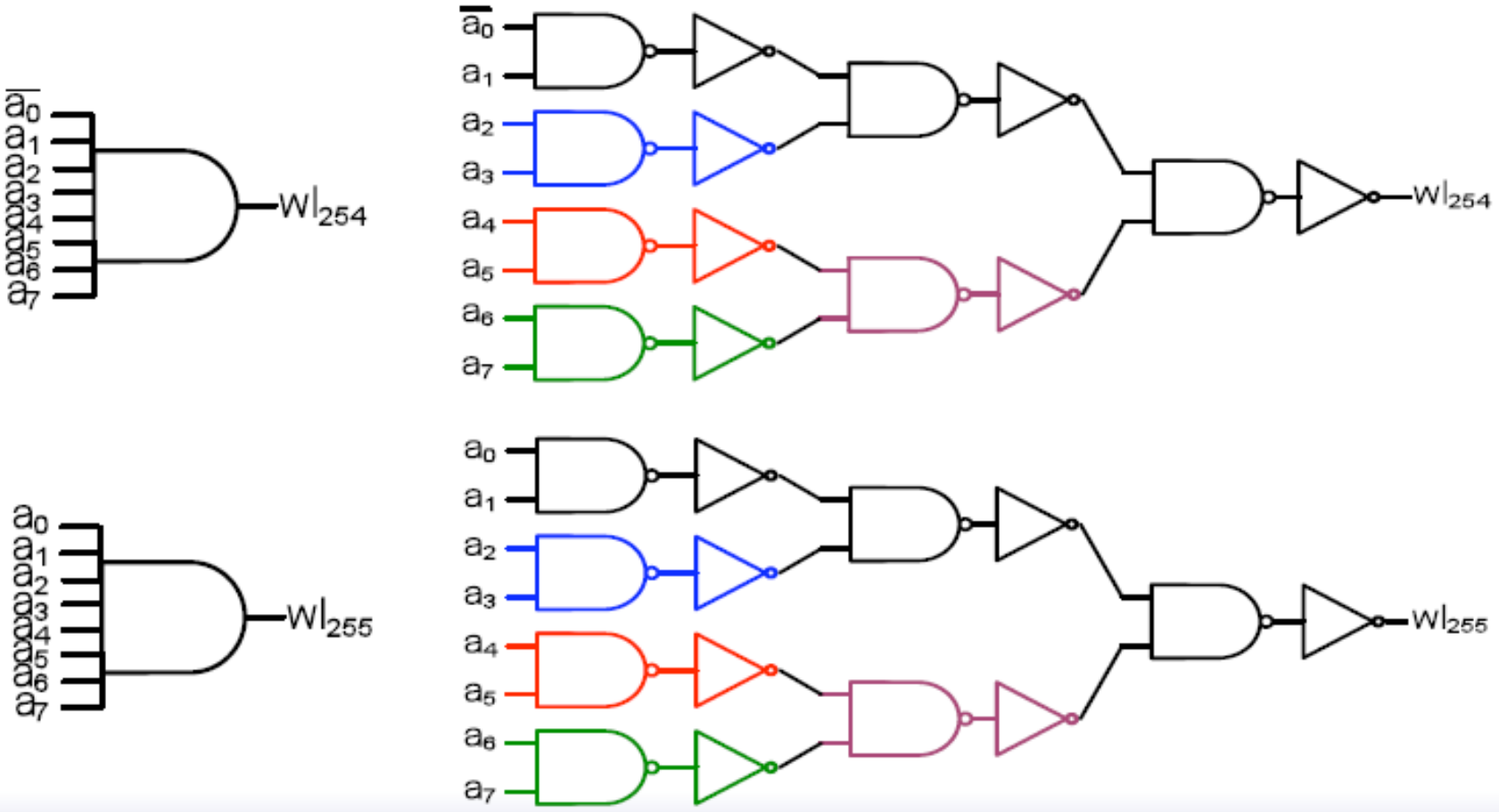
- Using 2-input NAND gates
  - 8-input gate takes 6 stages
- Total LE is $(4/3)^3 \approx 2.4$
- So PE is $2.4*2^{13}$ – optimal N of ~7.1

Decoder So Far

- 256 8-input AND gates
  - Each built out of tree of NAND gates and inverters
- Issue:
  - Every address line has to drive 128 gates (and wire) right away
  - Can't build gates small enough - Forces us to add buffers just to drive address inputs



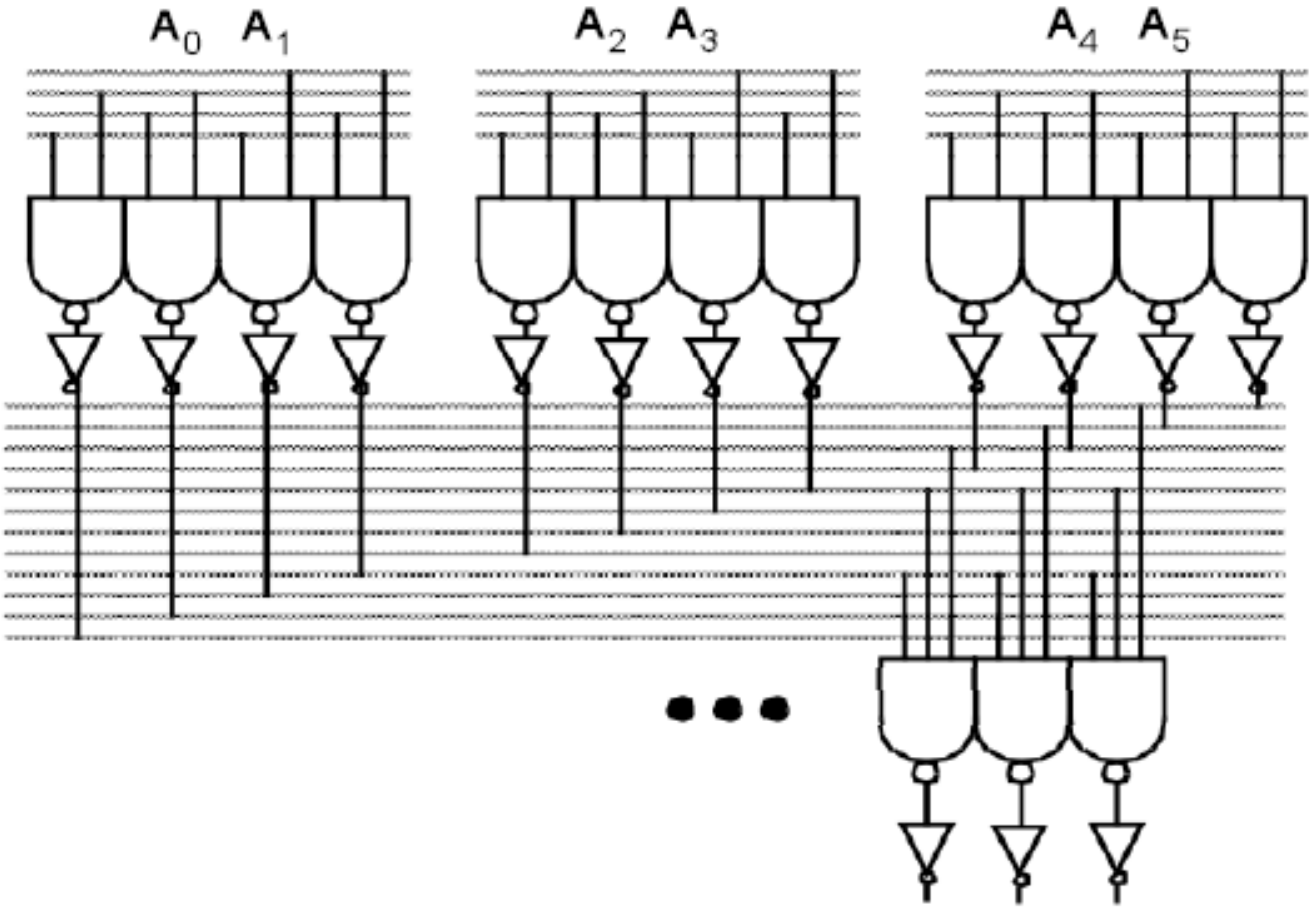
Look Inside Each AND8 Gate



Predecoders

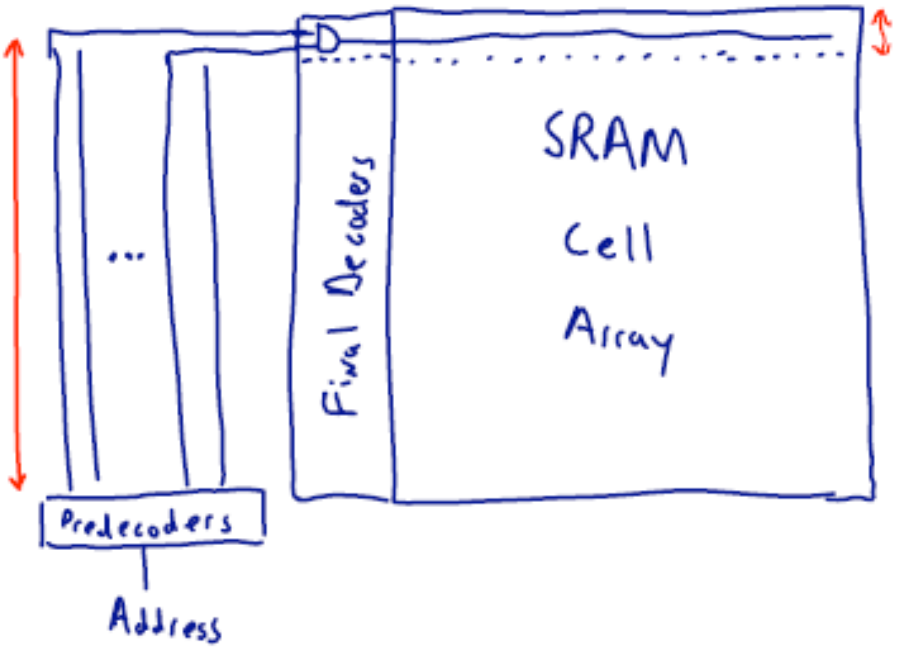
- Use a single gate for each of the shared terms
  - E.g., from A_0, $\overline{A_0}$, A_1, and $\overline{A_1}$, generate four signals: A_0A_1, $\overline{A_0}A_1$, $A_0\overline{A_1}$, $\overline{A_0}\overline{A_1}$
- In other words, we are decoding smaller groups of address bits first
  - And using the "predecoded" outputs to do the rest of the decoding

Predecoder and Decoder



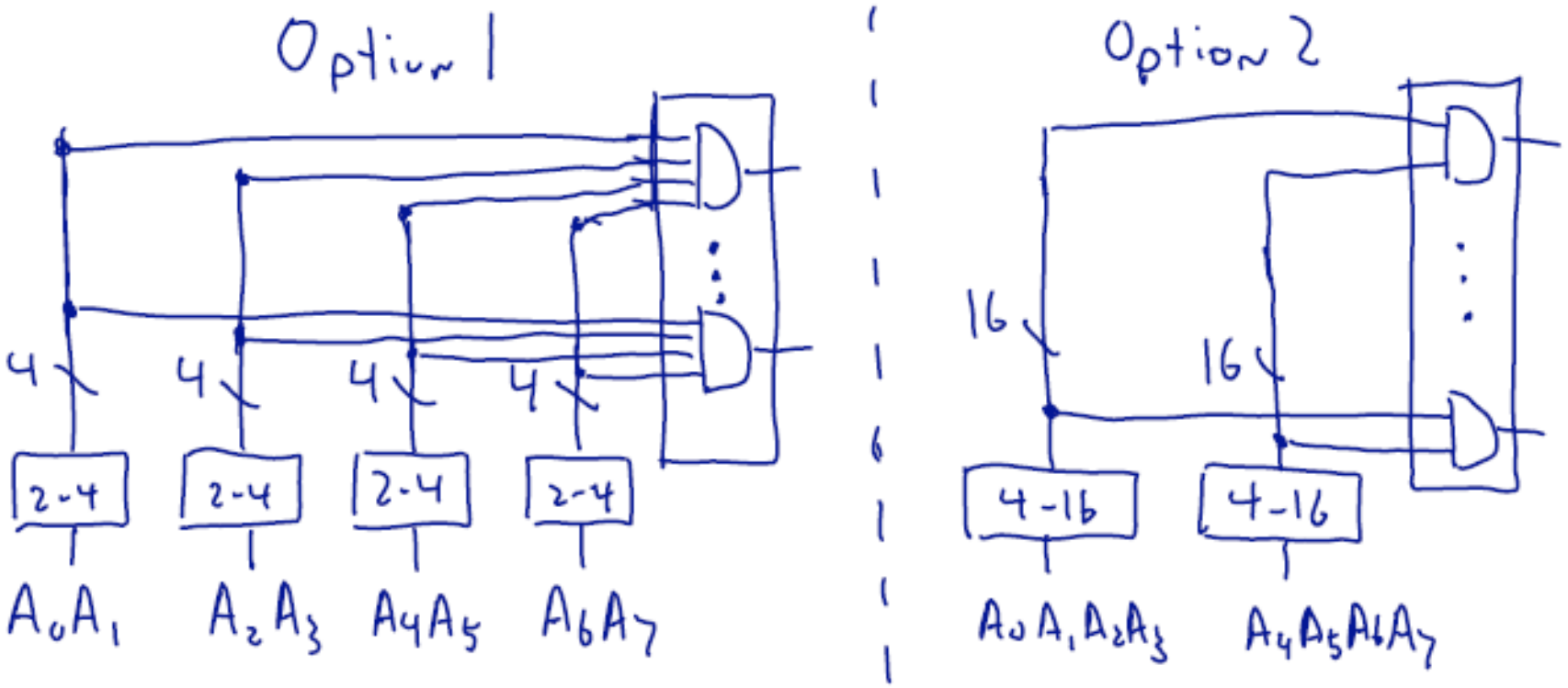
Predecoder/Decoder Layout

- Predecoder outputs run along height of the memory array.
- Decoder must match height of SRAM cell



Predecode Options

- Two options for predecoding:



Predecode Options (2)

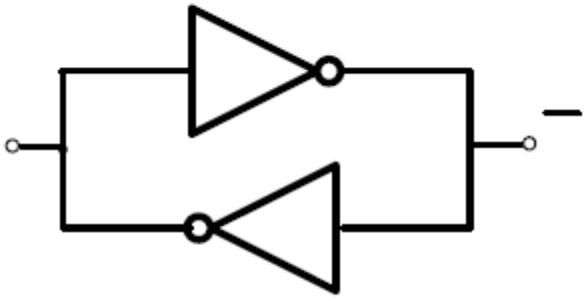
- Larger predecode usually better:
- More stages before the long wires
  - Decreases their effect on the circuit
- Fewer long wires switch
  - Lower power
- Easier to fit 2-input gate into cell pitch

What We Now Know

- Given decoder structure, input capacitance, final load
  - Can size the entire chain using LE for minimum delay

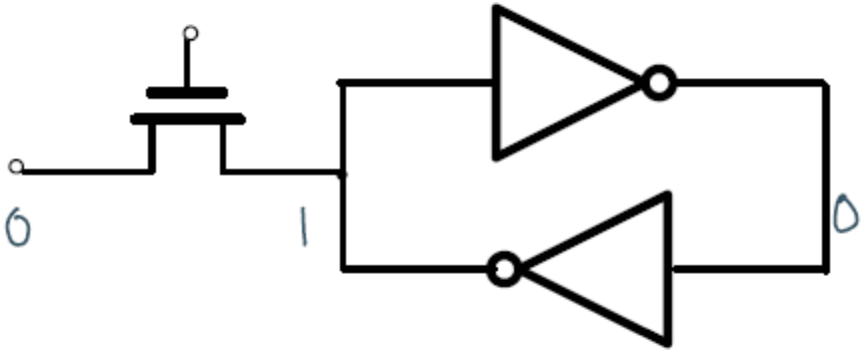
- Is this the "best" we can do in terms of power too?
  - Not necessarily – probably want to reduce sizes
    - (especially on final decoder inputs)
  - Is there anything else we can do to improve energy even further?

Basic Static Memory Element



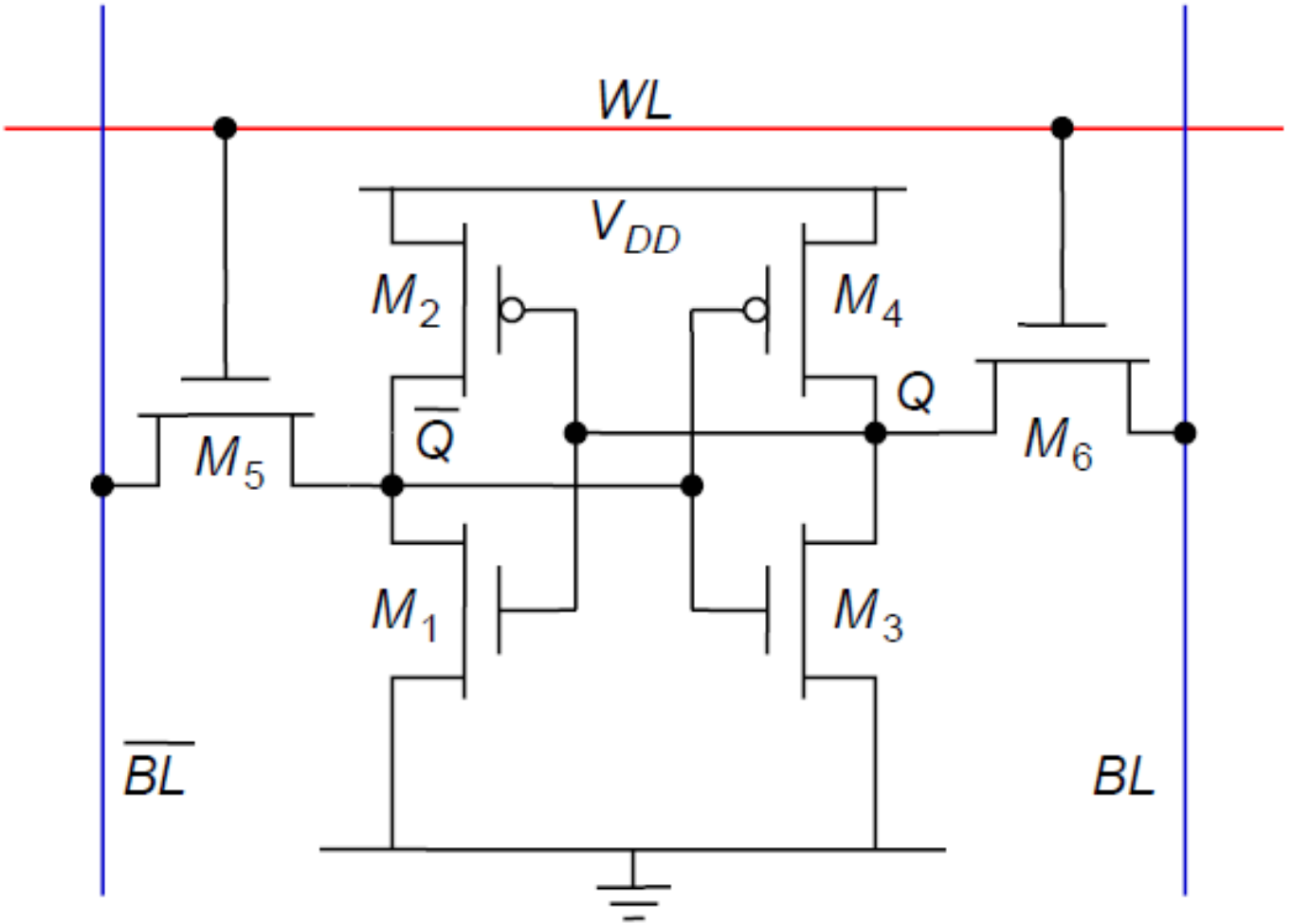
- If D is high, D_b will be driven low
  - Which makes D stay high
- Positive feedback

Writing into a Cross-Coupled Pair

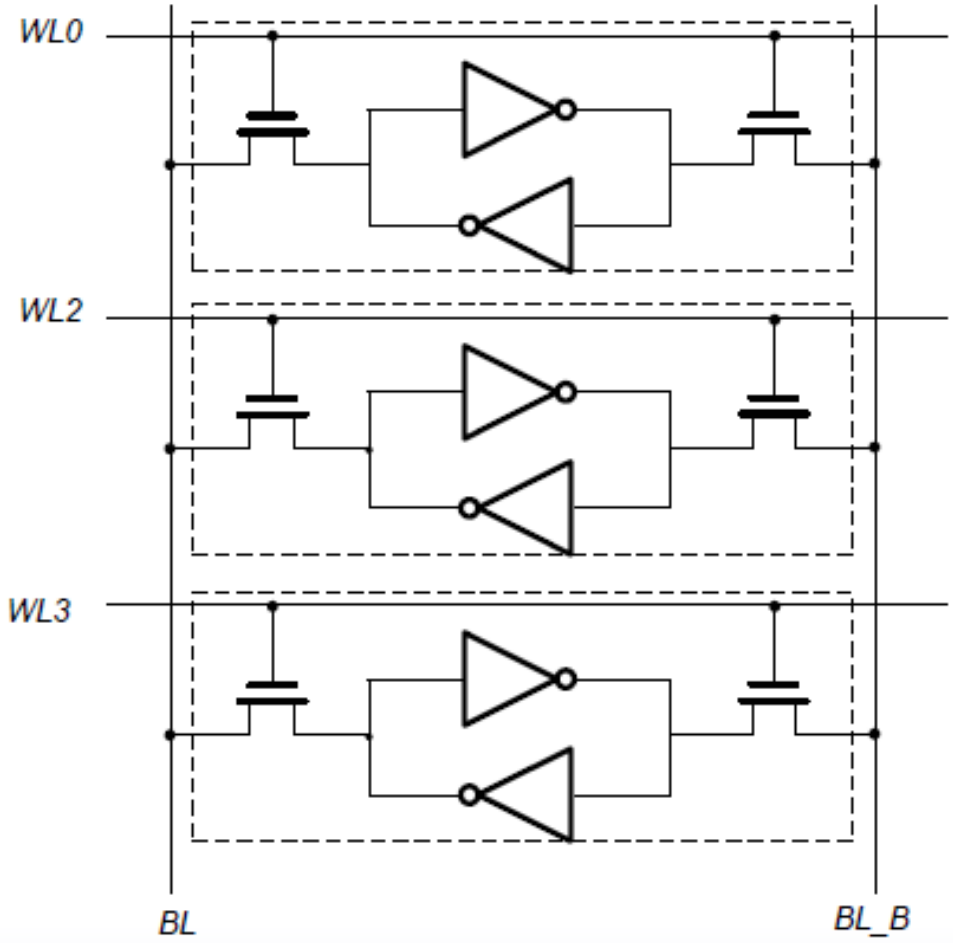


Access transistor must be able to overpower the feedback

6-transistor CMOS SRAM Cell

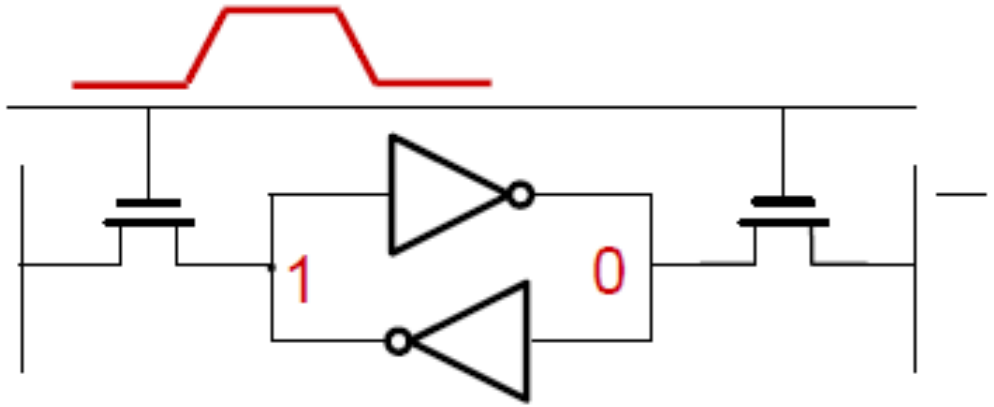


SRAM Column

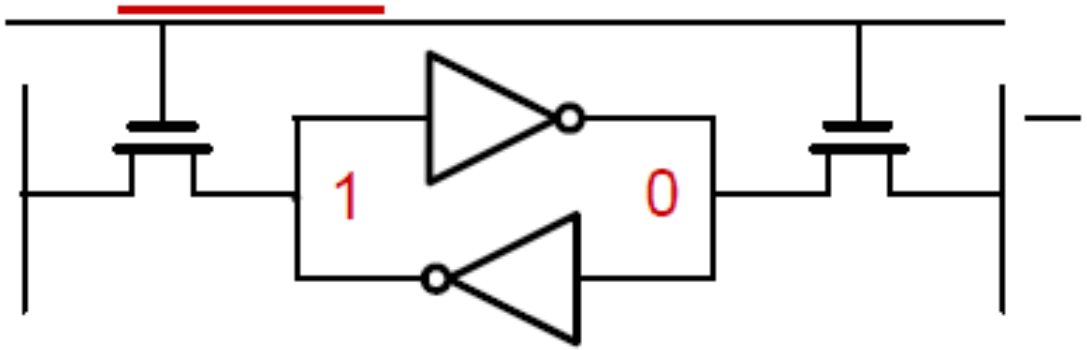


SRAM Operation

Write

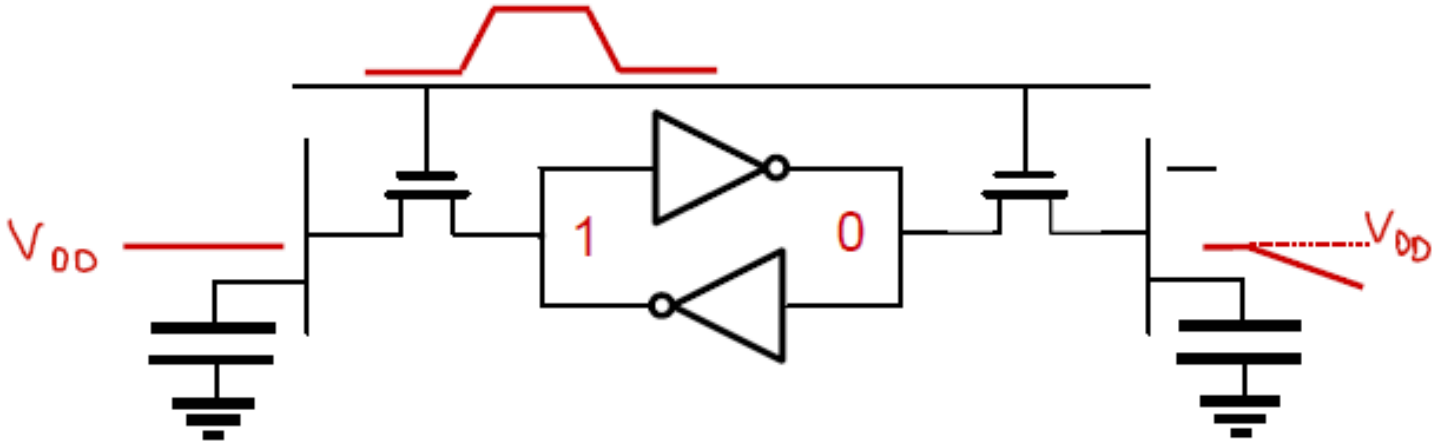


Hold



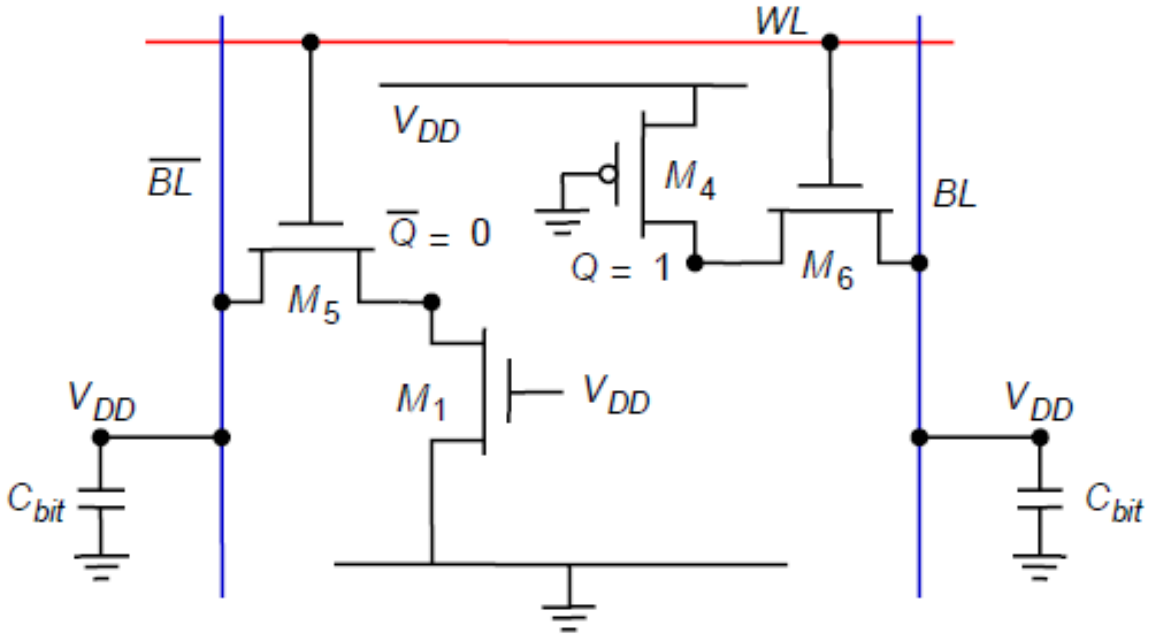
SRAM Operation

Read



- Q_b will get pulled up when WL first goes high
  - Reading the cell should not destroy the stored value

CMOS SRAM Analysis (Read)

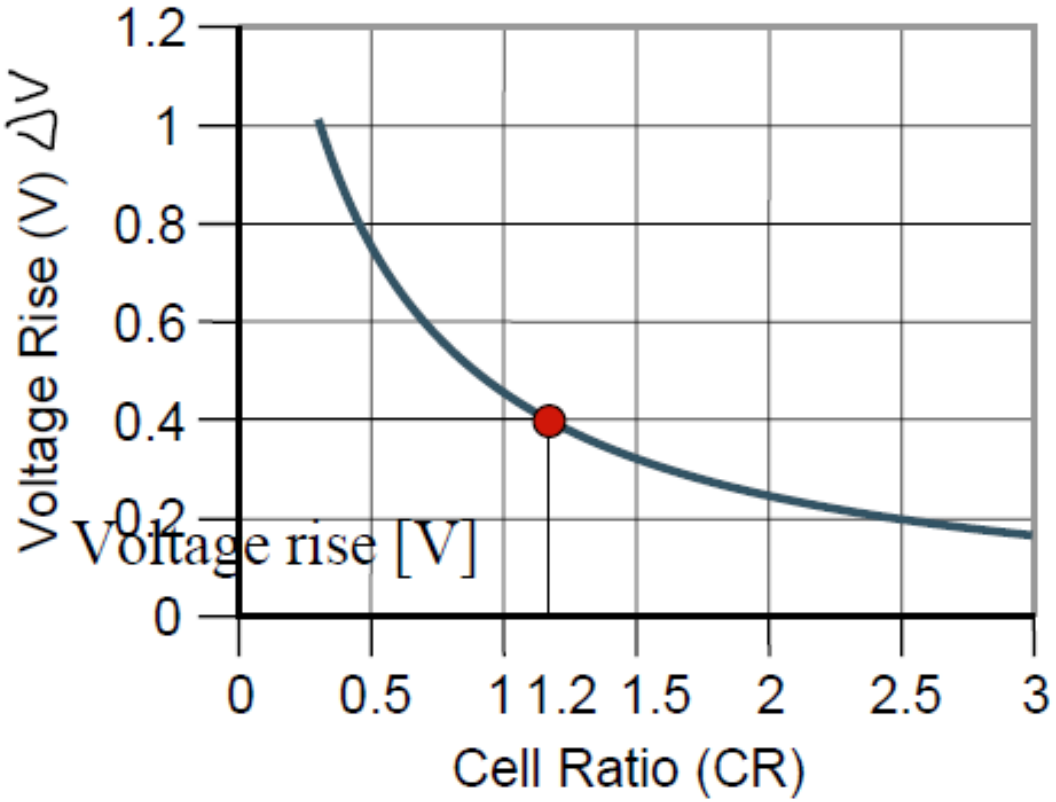


$$CR = \frac{W_1/L_1}{W_5/L_5}$$

$$k_{n,M5}\left((V_{DD} - \Delta V - V_{Tn})V_{DSATn} - \frac{V_{DSATn}^2}{2}\right) = k_{n,M1}\left((V_{DD} - V_{Tn})\Delta V - \frac{\Delta V^2}{2}\right)$$

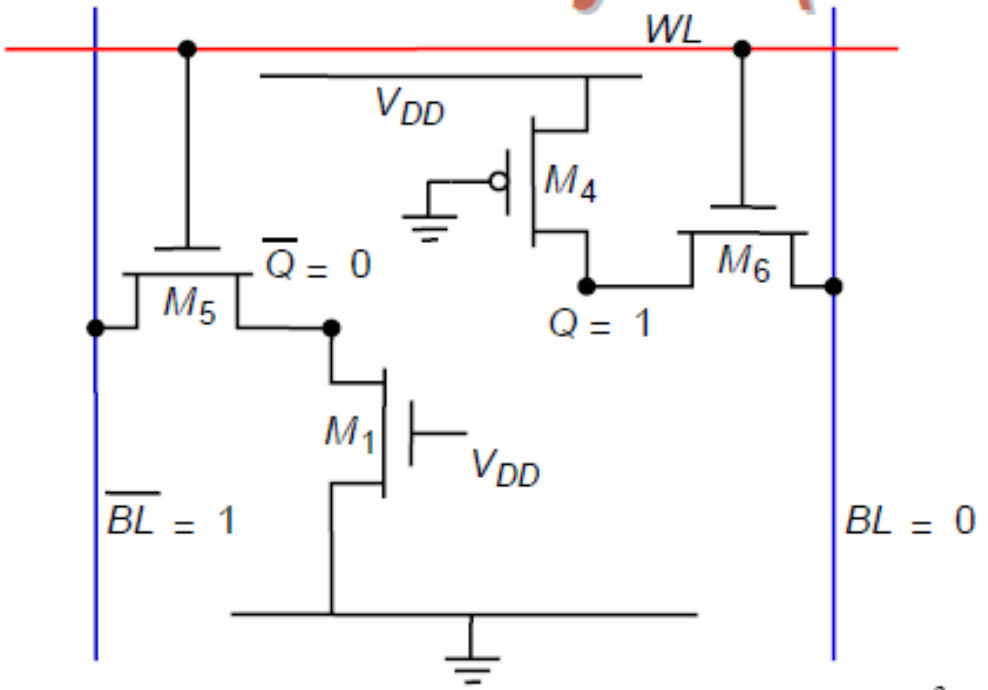
$$\Delta V = \frac{V_{DSATn} + CR(V_{DD} - V_{Tn}) - \sqrt{V_{DSATn}^2(1+CR) + CR^2(V_{DD} - V_{Tn})^2}}{CR}$$

CMOS SRAM Analysis (Read)



$$CR = \frac{W_1/L_1}{W_5/L_5}$$

CMOS SRAM Analysis (Write)

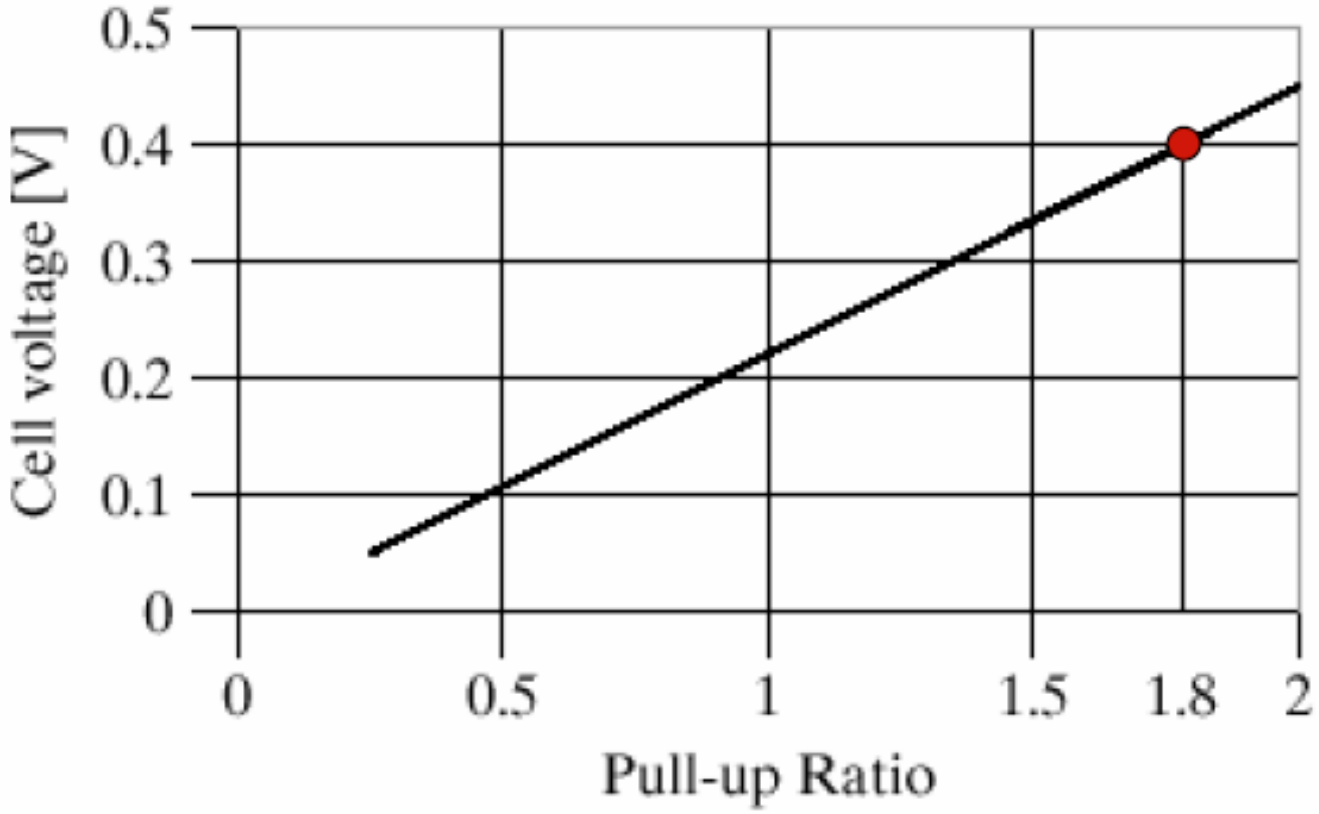


$$PR = \frac{(W/L)_4}{(W/L)_6}$$

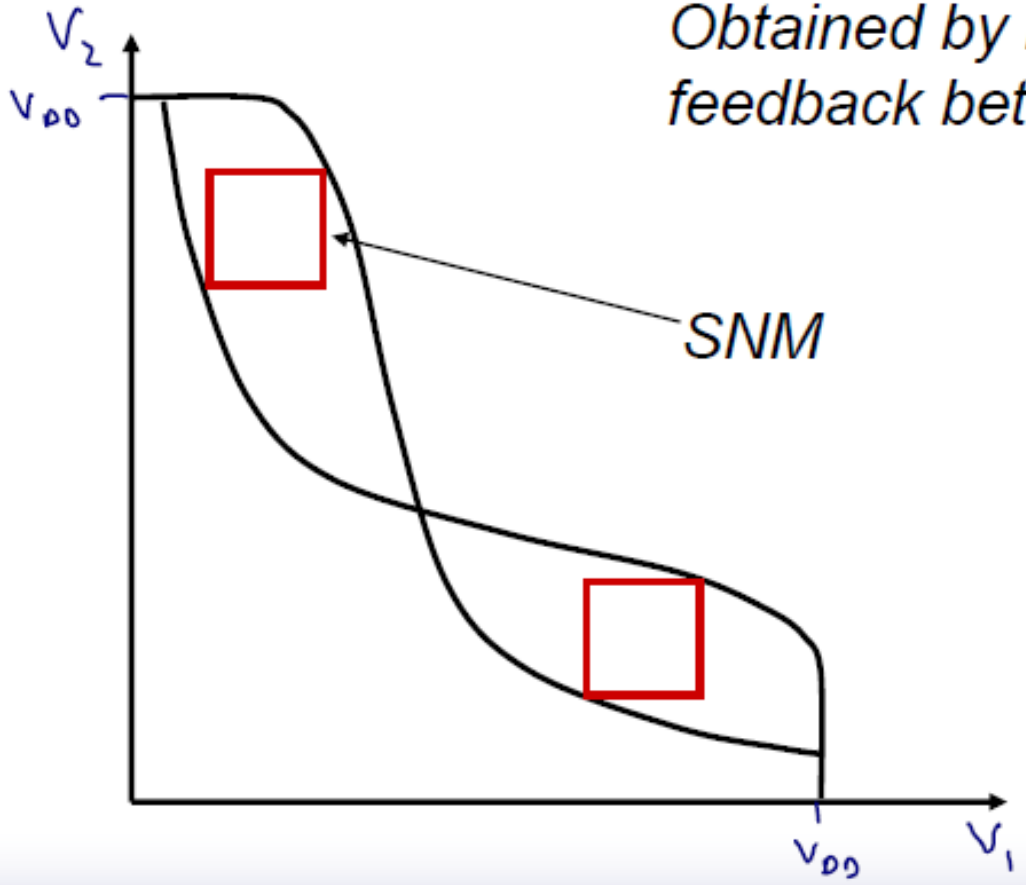
$$k_{n,M6}\left((V_{DD} - V_{Tn})V_Q - \frac{V_Q^2}{2}\right) = k_{p,M4}\left((V_{DD} - |V_{Tp}|)V_{\Psi SATp} - \frac{V_{\Psi SATp}^2}{2}\right)$$

$$V_Q = V_{DD} - V_{Tn} - \sqrt{(V_{DD} - V_{Tn})^2 - 2\frac{\mu_p}{\mu_n}PR\left((V_{DD} - |V_{Tp}|)V_{\Psi SATp} - \frac{V_{\Psi SATp}^2}{2}\right)},$$

CMOS SRAM Analysis (Write)

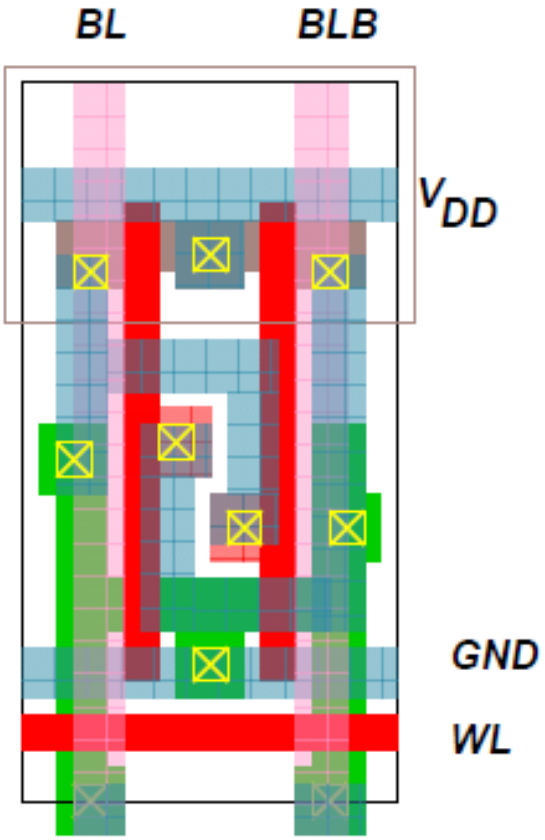


Read Static Noise Margin



Obtained by breaking the feedback between the inverters

6T-SRAM — Layout

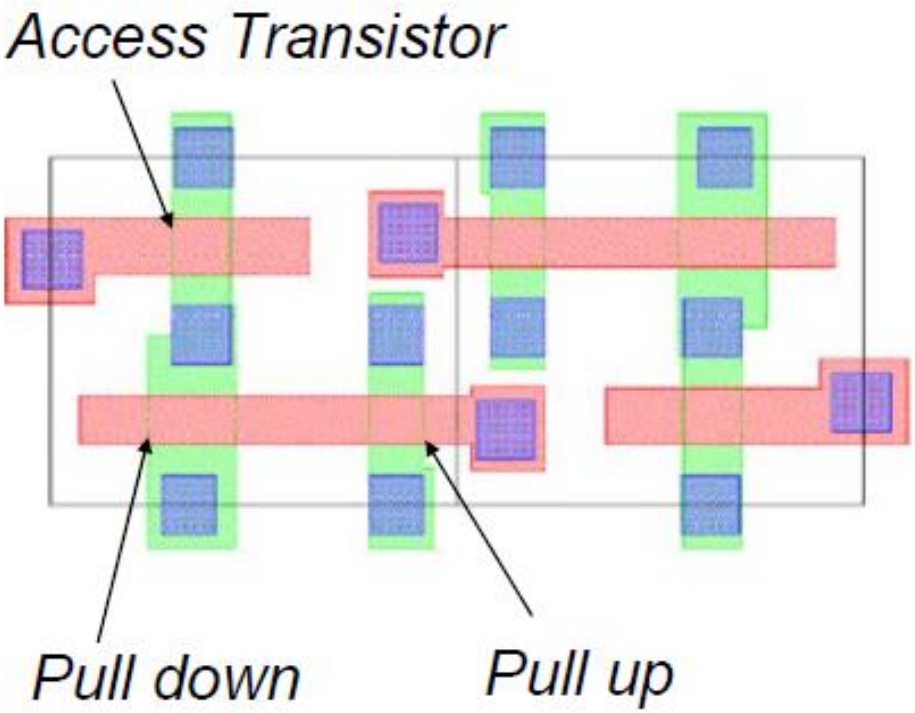


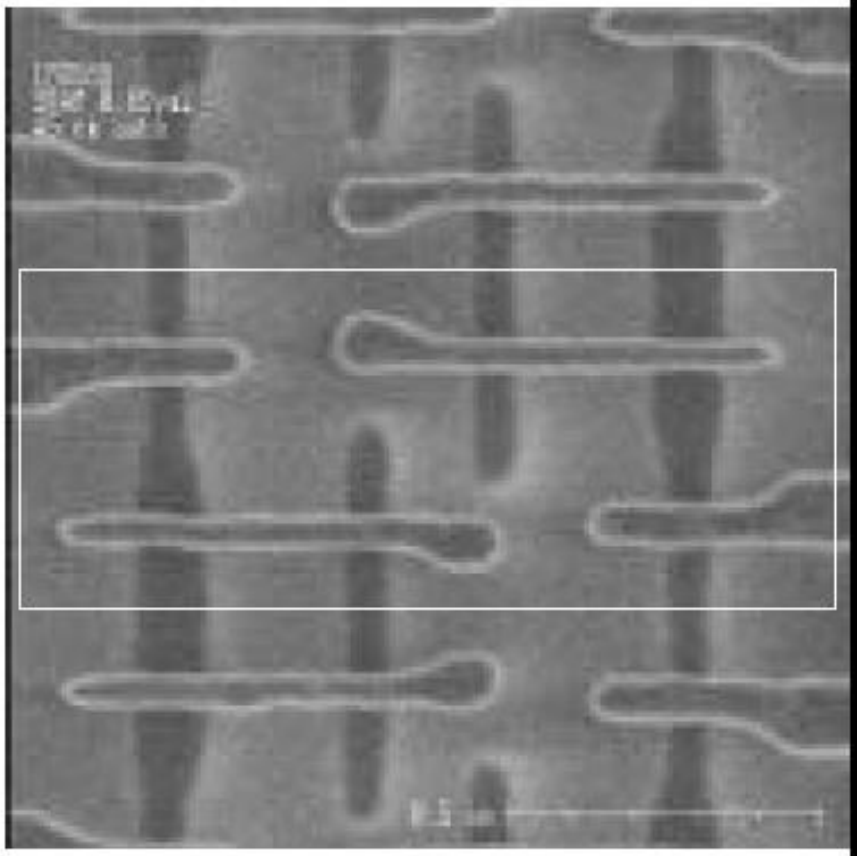
Compact cell

- Bitlines: M2
- Wordline: bootstrapped in M3

65nm SRAM

- ST/Philips/Motorola



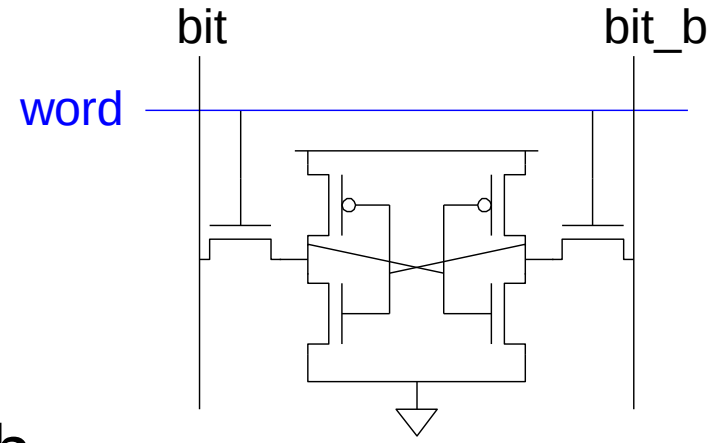


What does the future hold?

- Scaling will go on for some time
- But will slow down …
- The good news: Design rules!

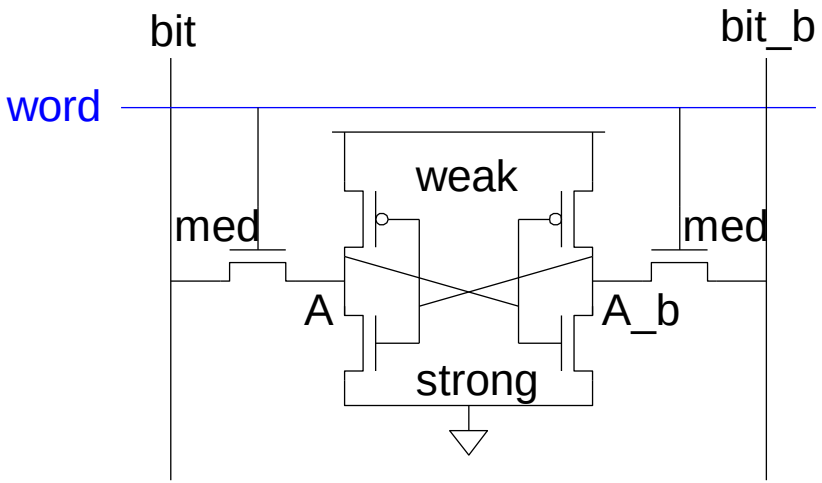
6T SRAM Cell

- Cell size accounts for most of array size
  - Reduce cell size at expense of complexity
- 6T SRAM Cell
  - Used in most commercial chips
  - Data stored in cross-coupled inverters
- Read:
  - Precharge bit, bit_b
  - Raise wordline
- Write:
  - Drive data onto bit, bit_b
  - Raise wordline



SRAM Sizing

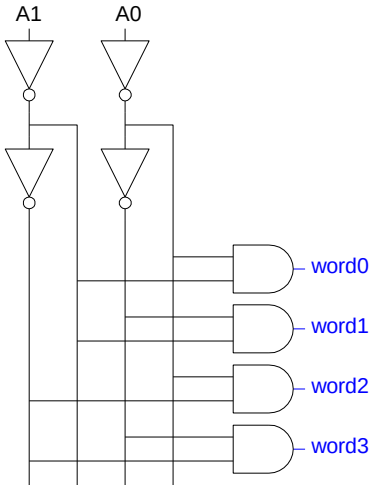
- High bitlines must not overpower inverters during reads
- But low bitlines must write new value into cell

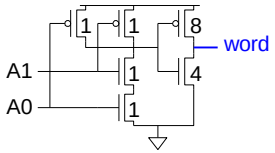


Decoders

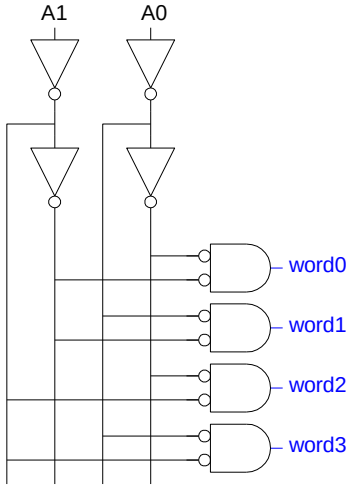
- n:2^n decoder consists of 2^n n-input AND gates
  - One needed for each row of memory
  - Build AND from NAND or NOR gates

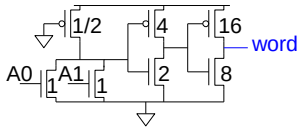
Static CMOS





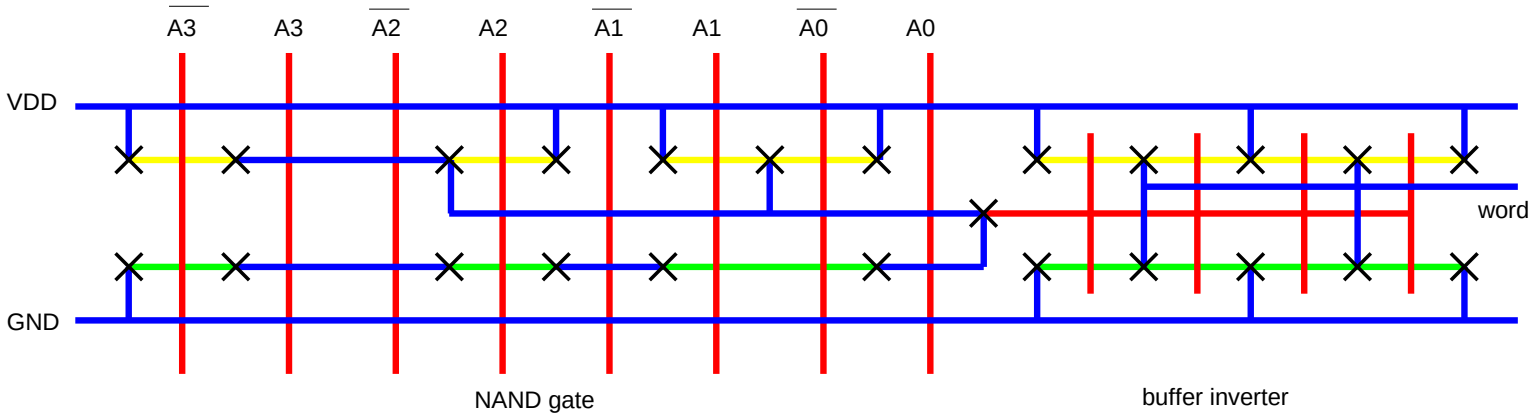
Pseudo-nMOS





Decoder Layout

- Decoders must be pitch-matched to SRAM cell
  - Requires very skinny gates

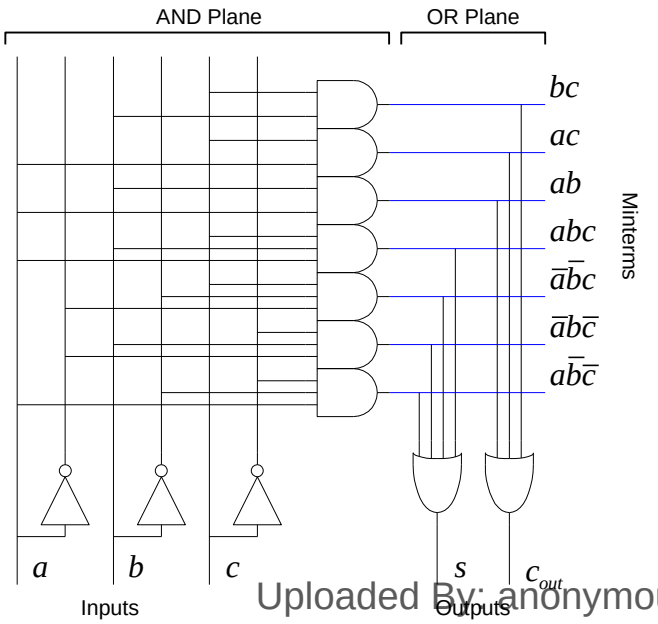


PLAs

- A *Programmable Logic Array* performs any function in sum-of-products form.
- *Literals*: inputs & complements
- *Products / Minterms*: AND of literals
- *Outputs*: OR of Minterms
- Example: Full Adder

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc$$

$$c_{\text{out}} = ab + bc + ac$$



PLA Schematic & Layout