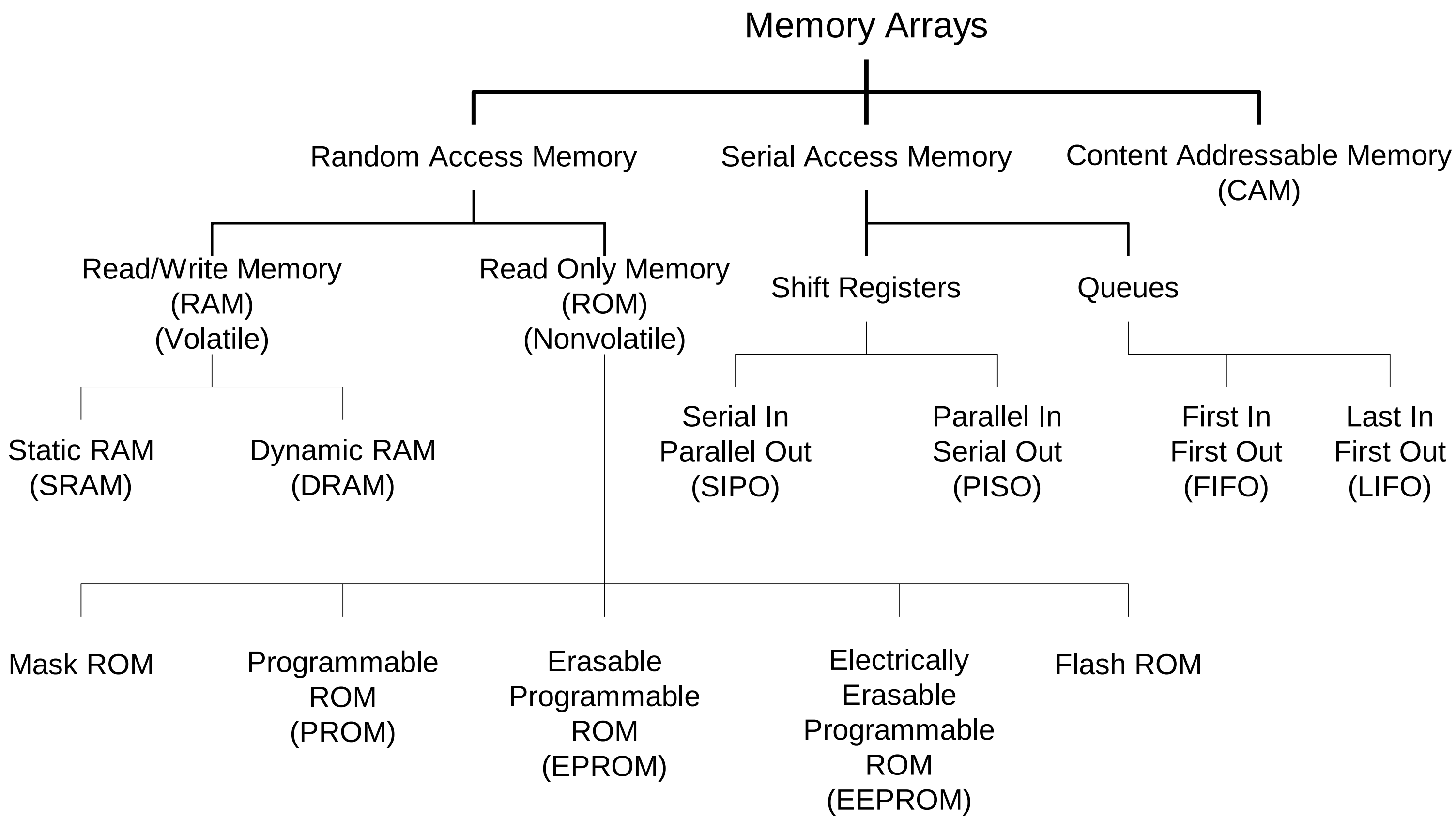
Introduction to CMOS VLSI Design

Lecture 13: SRAM

Outline

- Memory Arrays
- SRAM Architecture
  - SRAM Cell
  - Decoders
  - Column Circuitry
  - Multiple Ports
- Serial Access Memories

Memory Arrays



Memory Arrays

Random Access Memory

Serial Access Memory

Content Addressable Memory (CAM)

Read/Write Memory (RAM) (Volatile)

Read Only Memory (ROM) (Nonvolatile)

Shift Registers

Queues

Static RAM (SRAM)

Dynamic RAM (DRAM)

Serial In Parallel Out (SIPO)

Parallel In Serial Out (PISO)

First In First Out (FIFO)

Last In First Out (LIFO)

Mask ROM

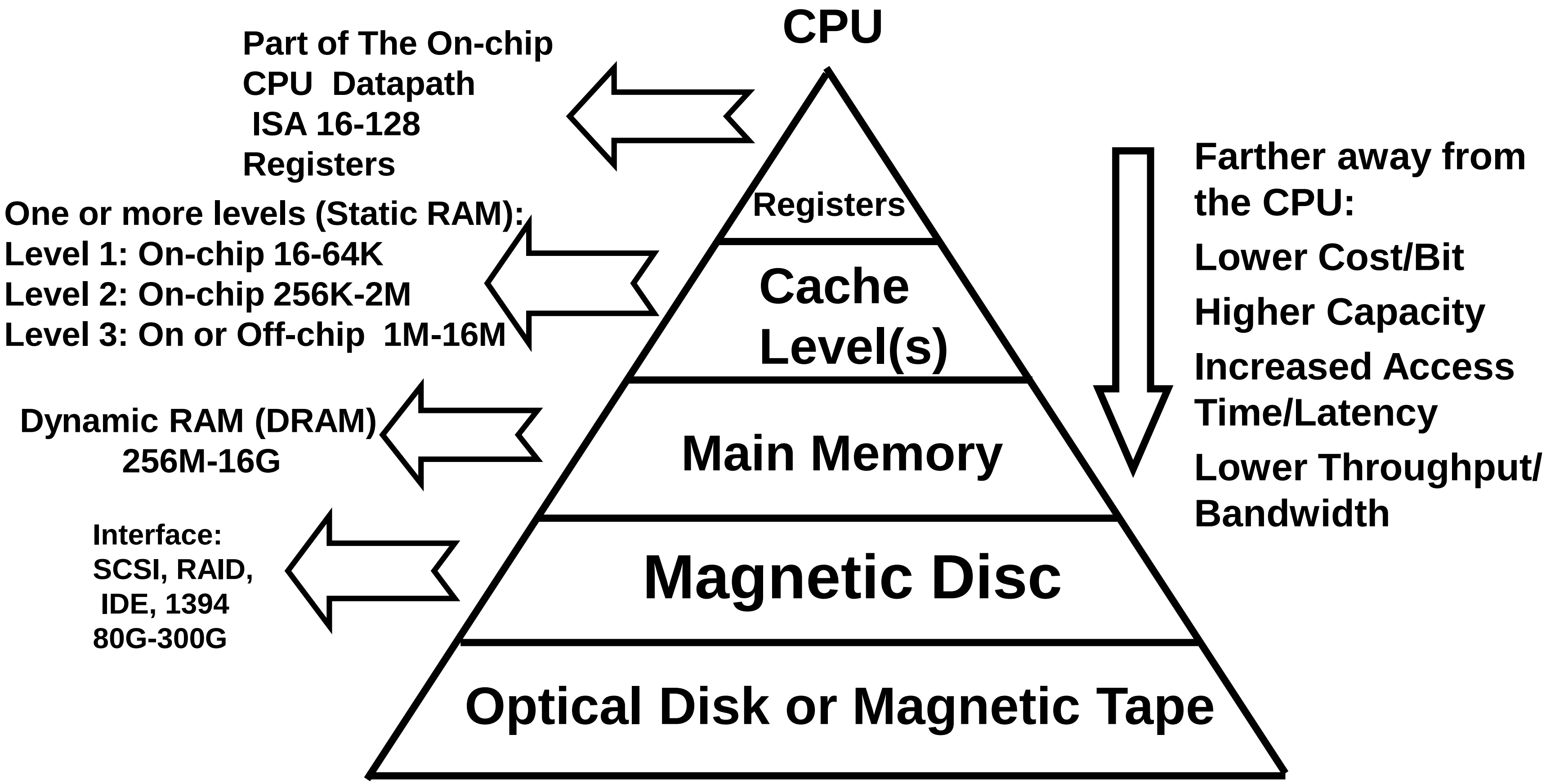
Programmable ROM (PROM)

Erasable Programmable ROM (EPROM)

Electrically Erasable Programmable ROM (EEPROM)

Flash ROM

Levels of the Memory Hierarchy



CPU

Part of The On-chip CPU Datapath ISA 16-128 Registers

One or more levels (Static RAM):
Level 1: On-chip 16-64K
Level 2: On-chip 256K-2M
Level 3: On or Off-chip 1M-16M

Dynamic RAM (DRAM)
256M-16G

Interface:
SCSI, RAID, IDE, 1394
80G-300G

Registers

Cache Level(s)

Main Memory

Magnetic Disc

Optical Disk or Magnetic Tape

Farther away from the CPU:

Lower Cost/Bit

Higher Capacity

Increased Access Time/Latency

Lower Throughput/ Bandwidth

Memory Hierarchy Comparisons

Capacity
Access Time
Cost

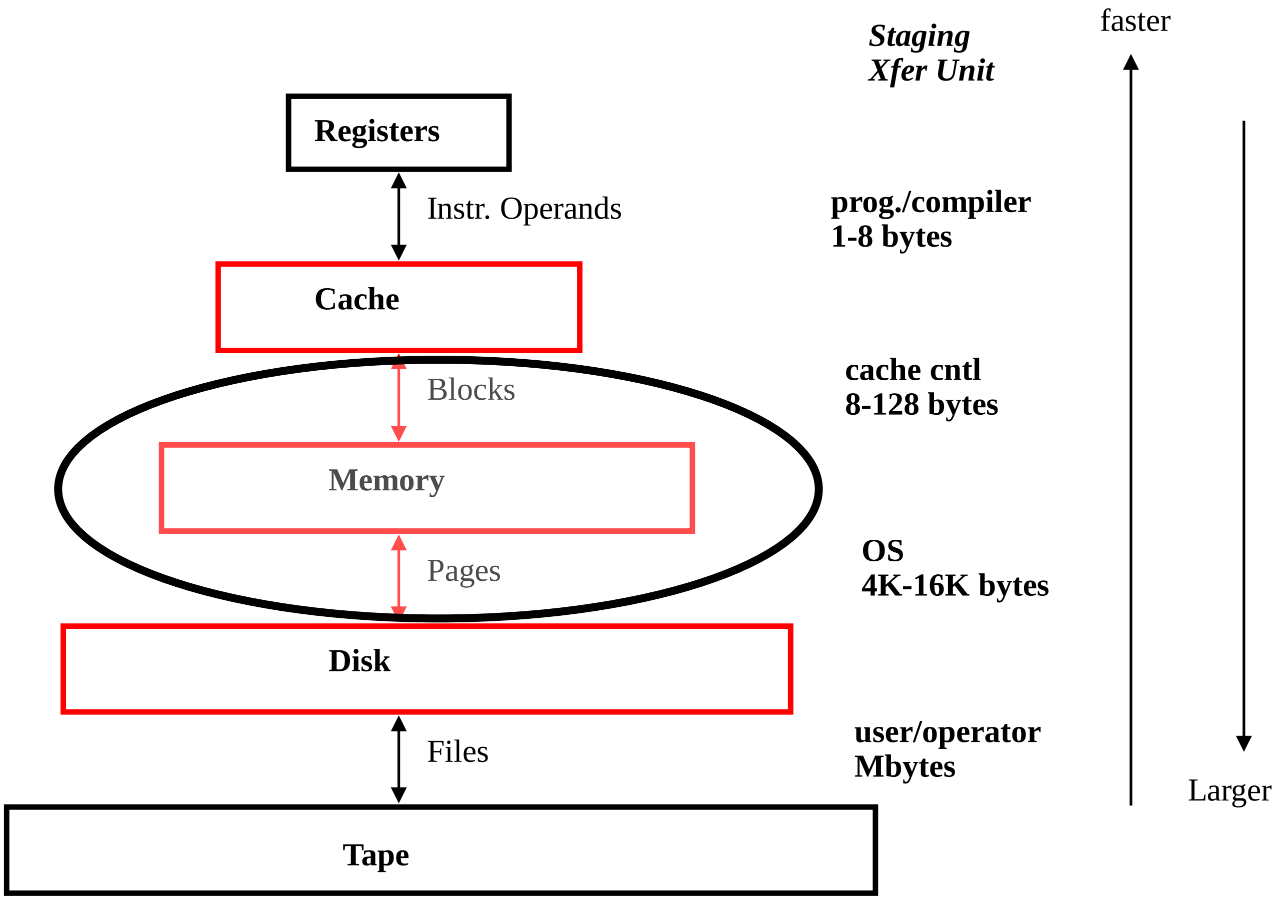
CPU Registers
100s Bytes
<10s ns

Cache
K Bytes
10-100 ns
1-0.1 cents/bit

Main Memory
M Bytes
200ns- 500ns
$.0001-.00001 cents /bit

Disk
G Bytes, 10 ms
(10,000,000 ns)
$10^{-5} - 10^{-6}$ cents/bit

Tape
infinite
sec-min
10^{-8}



Staging
Xfer Unit

faster

Registers

Instr. Operands

prog./compiler
1-8 bytes

Cache

Blocks

cache cntl
8-128 bytes

Memory

Pages

OS
4K-16K bytes

Disk

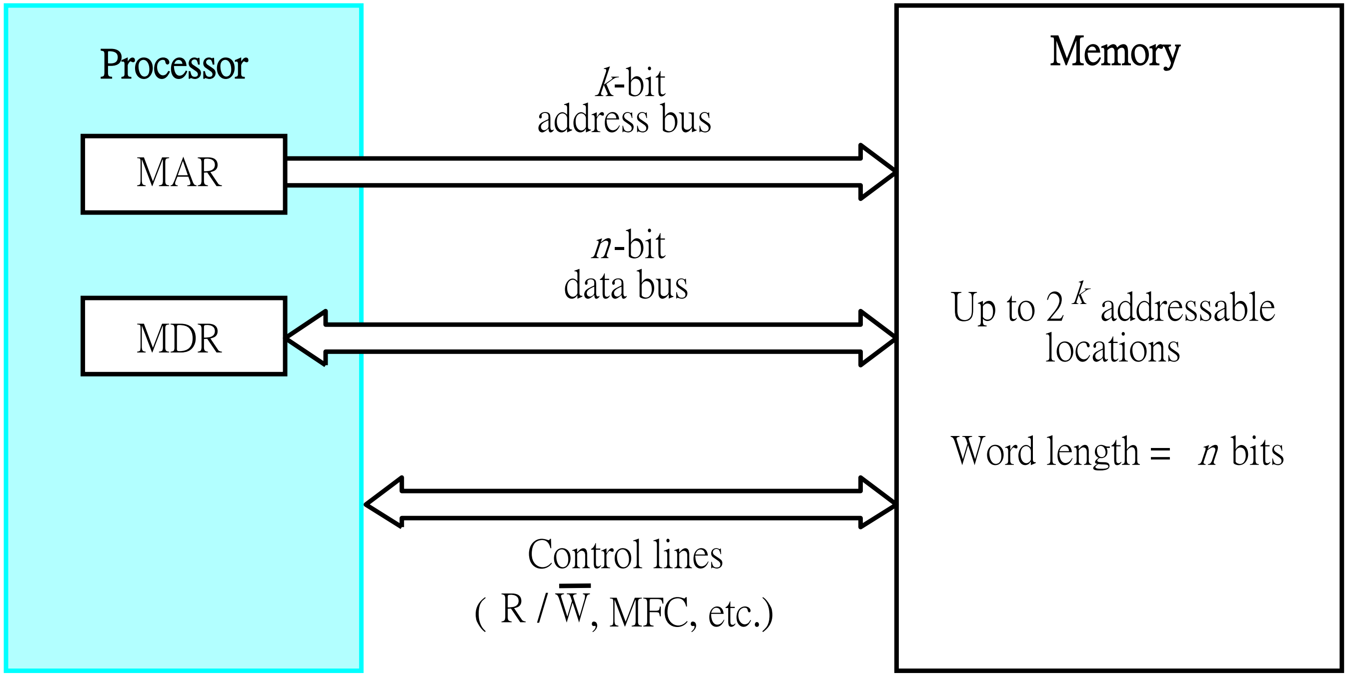
Files

user/operator
Mbytes

Larger

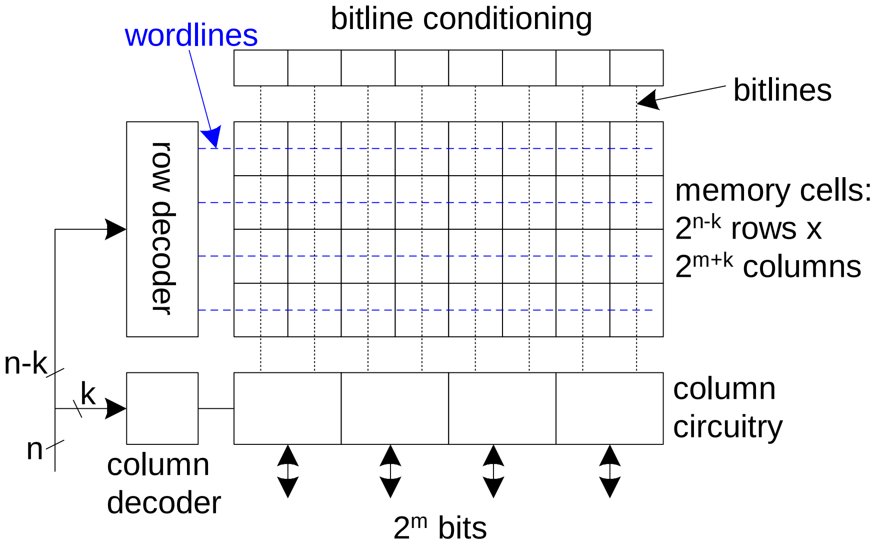
Tape

Connecting Memory



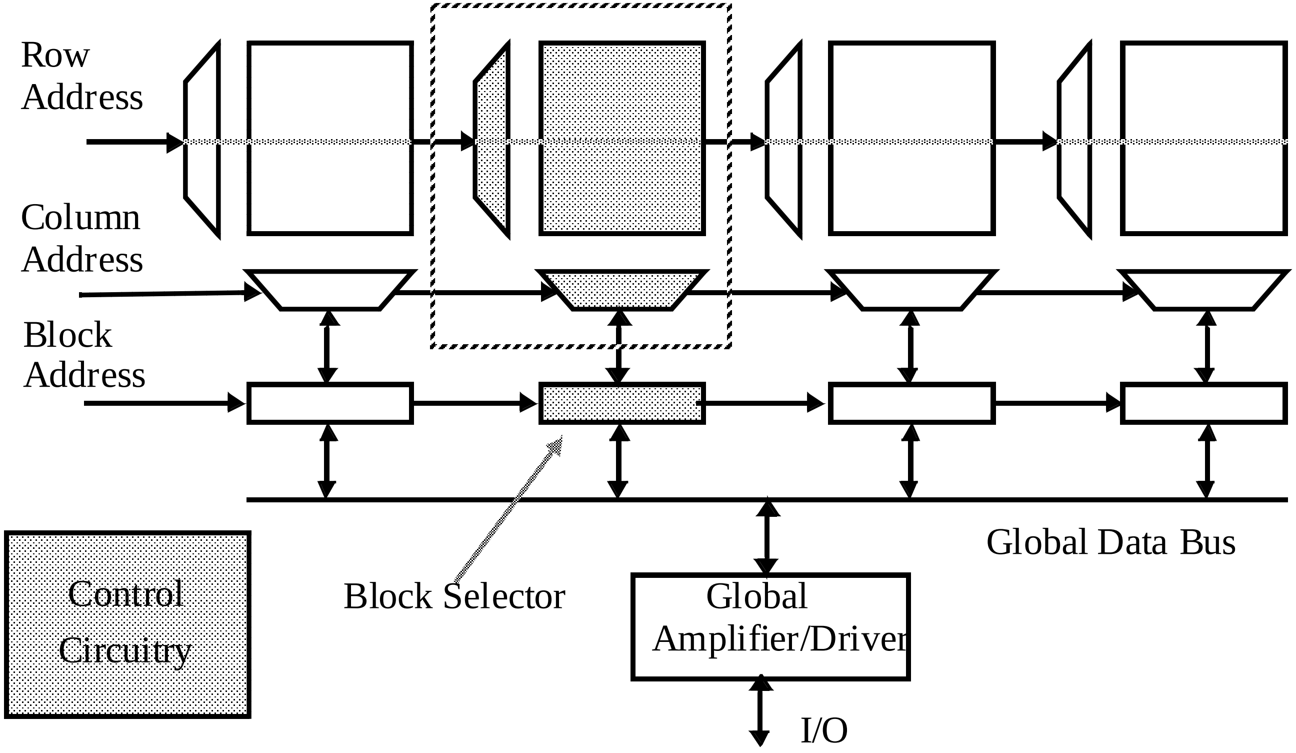
Array Architecture

- ❑ 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

Hierarchical Memory Architecture



Advantages:
1. Shorter wires within blocks
2. Block address activates only 1 block => power savings

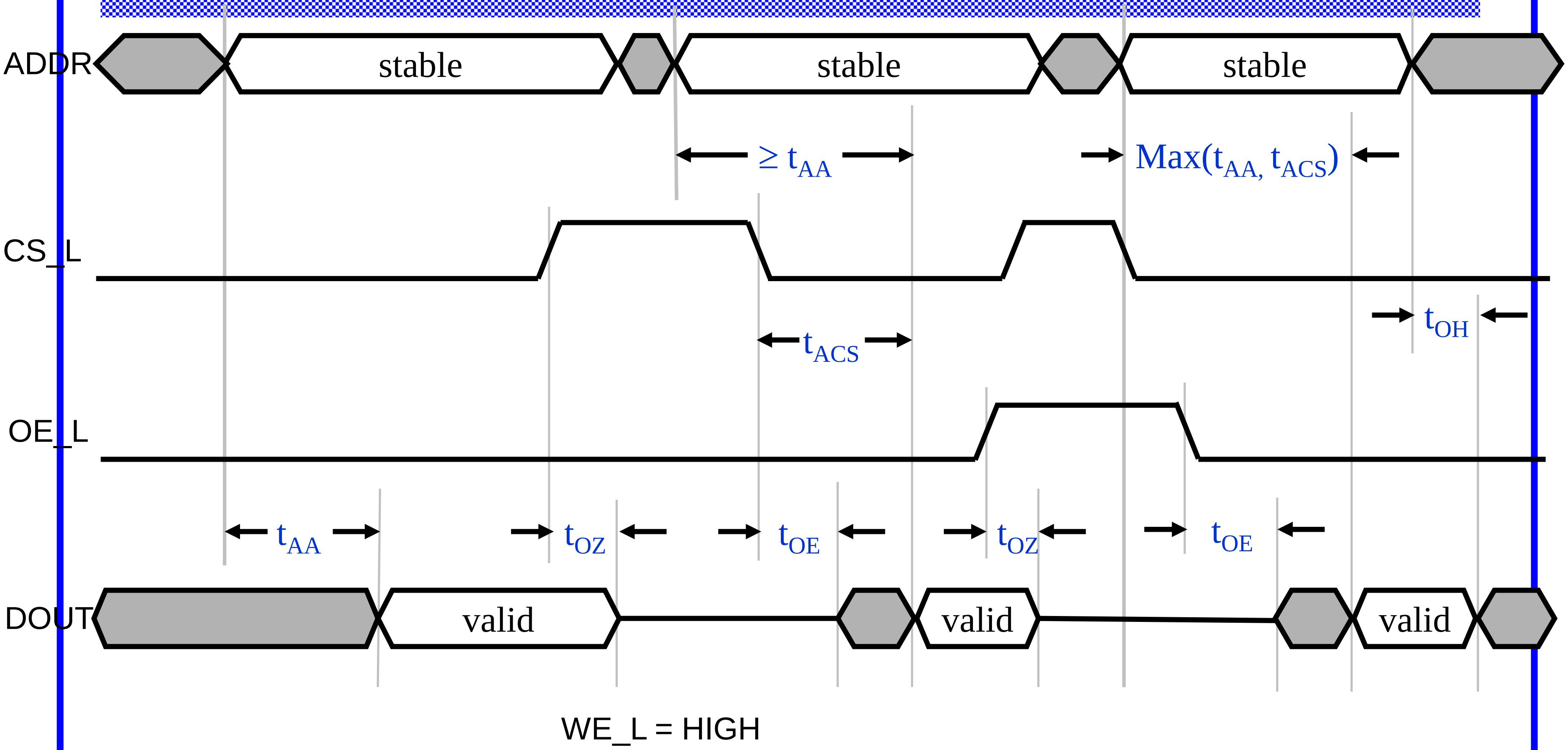
Array Organization Design Issues

- ❑ aspect ratio should be relative square
  - – Row / Column organization (matrix)
  - – R = $\log_2$(N_rows); C = $\log_2$(N_columns)
  - – R + C = N (N_address_bits)
- ❑ number of rows should be power of 2
  - – number of bits in a row need not be…
- ❑ sense amplifiers to speed voltage swing
- ❑ 1 -> 2^R row decoder
- ❑ 1 -> 2^C column decoder
  - – M column decoders (M bits, one per bit)
    - • M = output word width

SRAM Read Timing (typical)

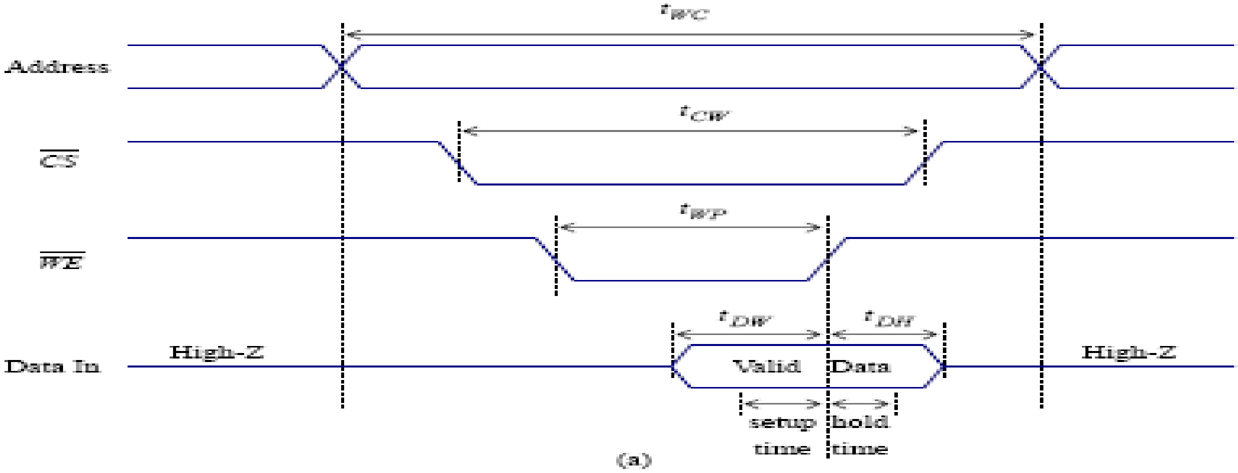
- ❑ t_{AA} (access time for address): time for stable output after a change in address.
- ❑ t_{ACS} (access time for chip select): time for stable output after CS is asserted.
- ❑ t_{OE} (output enable time): time for low impedance when OE and CS are both asserted.
- ❑ t_{OZ} (output-disable time): time to high-impedance state when OE or CS are negated.
- ❑ t_{OH} (output-hold time): time data remains valid after a change to the address inputs.

SRAM Read Timing (typical)

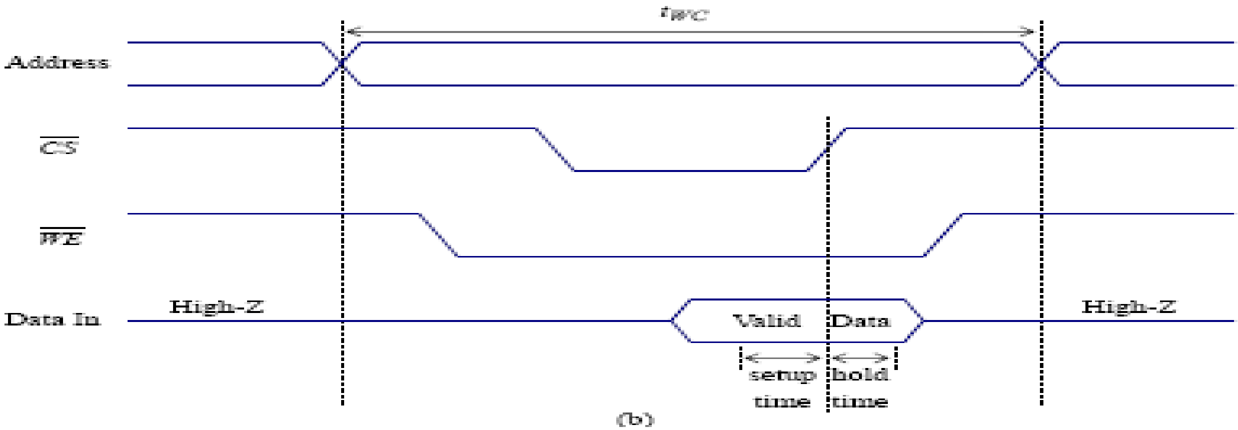


SRAM write cycle timing

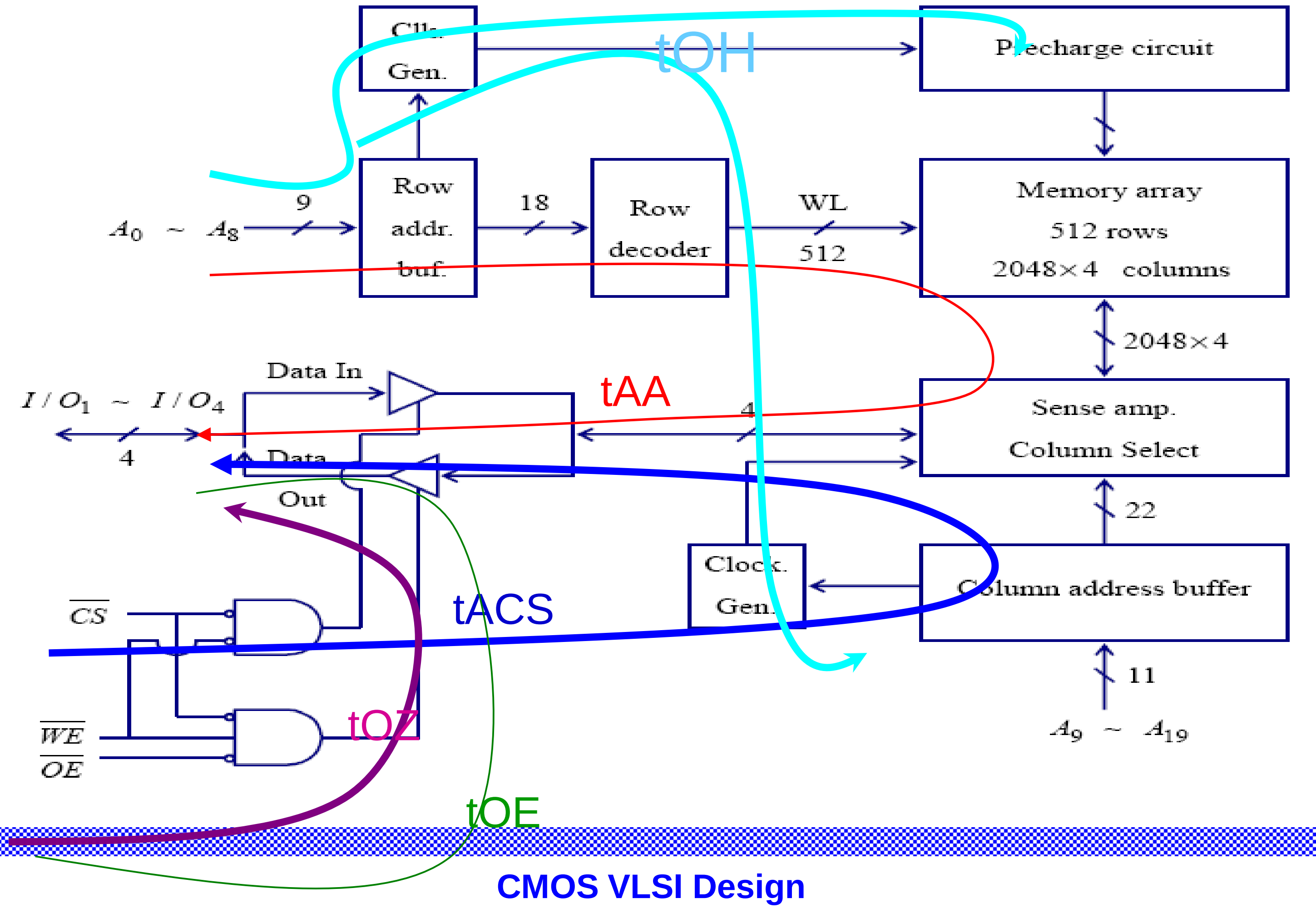
~WE controlled



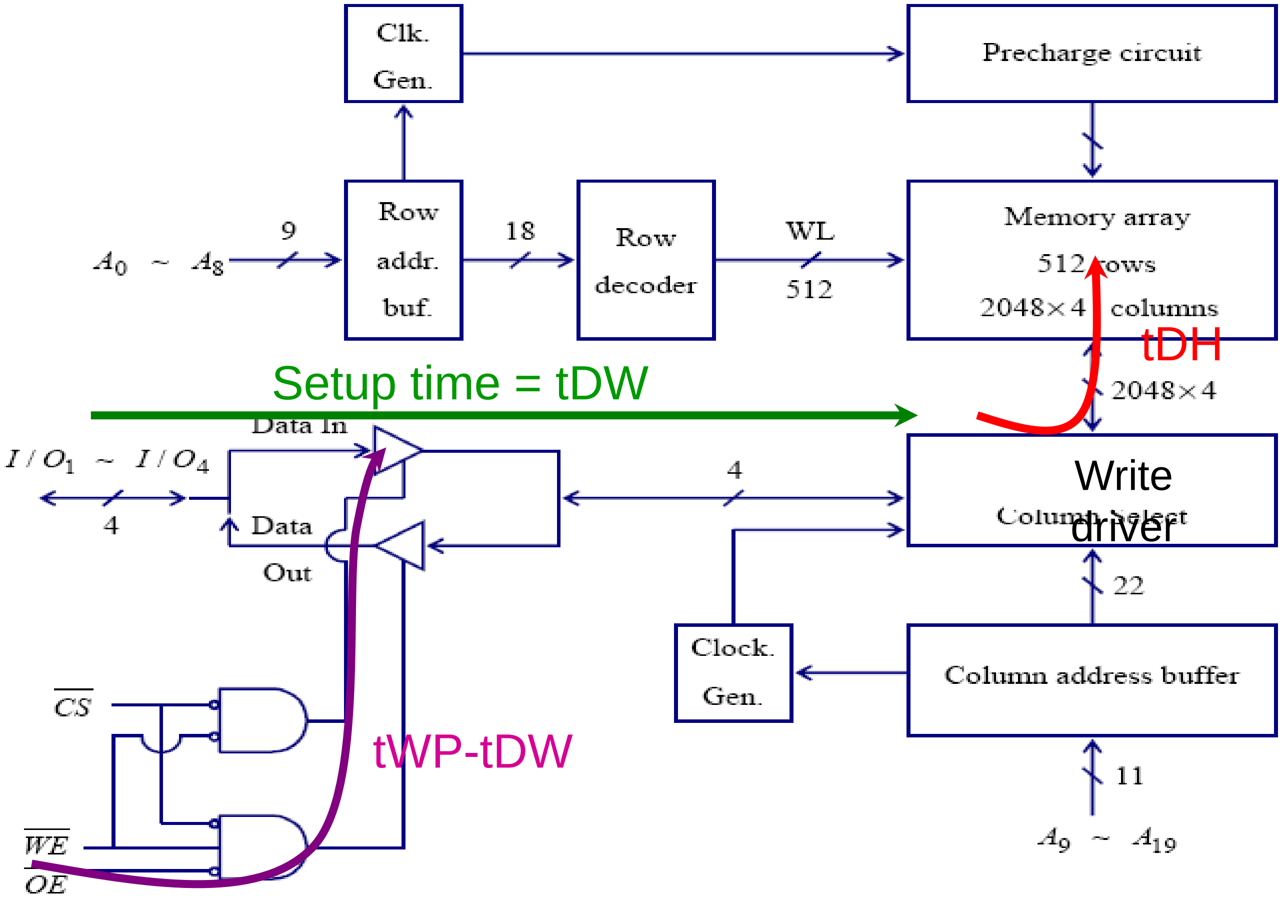
~CS controlled



SRAM Architecture and Read Timings



SRAM Architecture and Write Timings

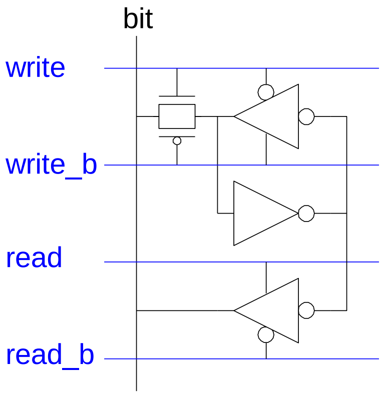


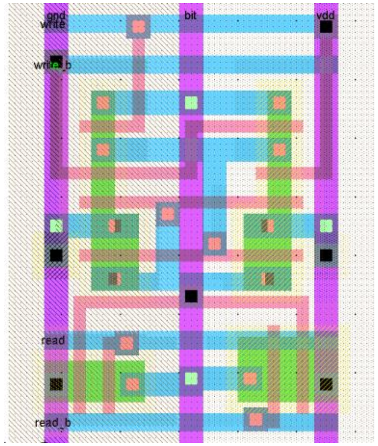
nMOS I-V Summary

$$I_{ds} = \begin{cases} 0 & V_{gs} < V_t & \text{cutoff} \\ \beta\left(V_{gs} - V_t - {V_{ds}}/{2}\right)V_{ds} & V_{ds} < V_{dsat} & \text{linear} \\ \dfrac{\beta}{2}\left(V_{gs} - V_t\right)^2 & V_{ds} > V_{dsat} & \text{saturation} \end{cases}$$

12T SRAM Cell

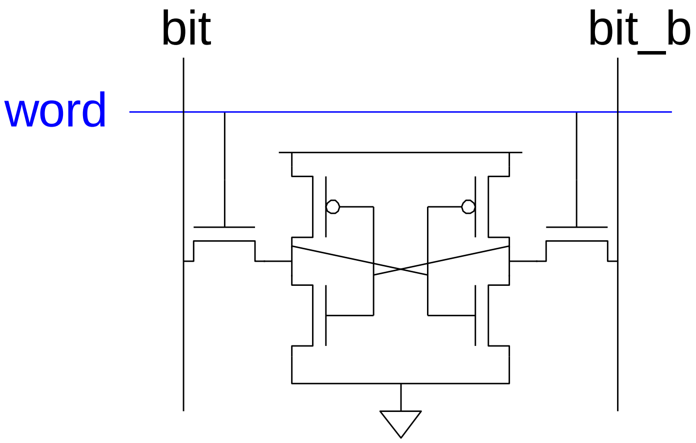
- ❑ Basic building block: SRAM Cell
  - – Holds one bit of information, like a latch
  - – Must be read and written
- ❑ 12-transistor (12T) SRAM cell
  - – Use a simple latch connected to bitline
  - – 46 x 75 λ unit cell



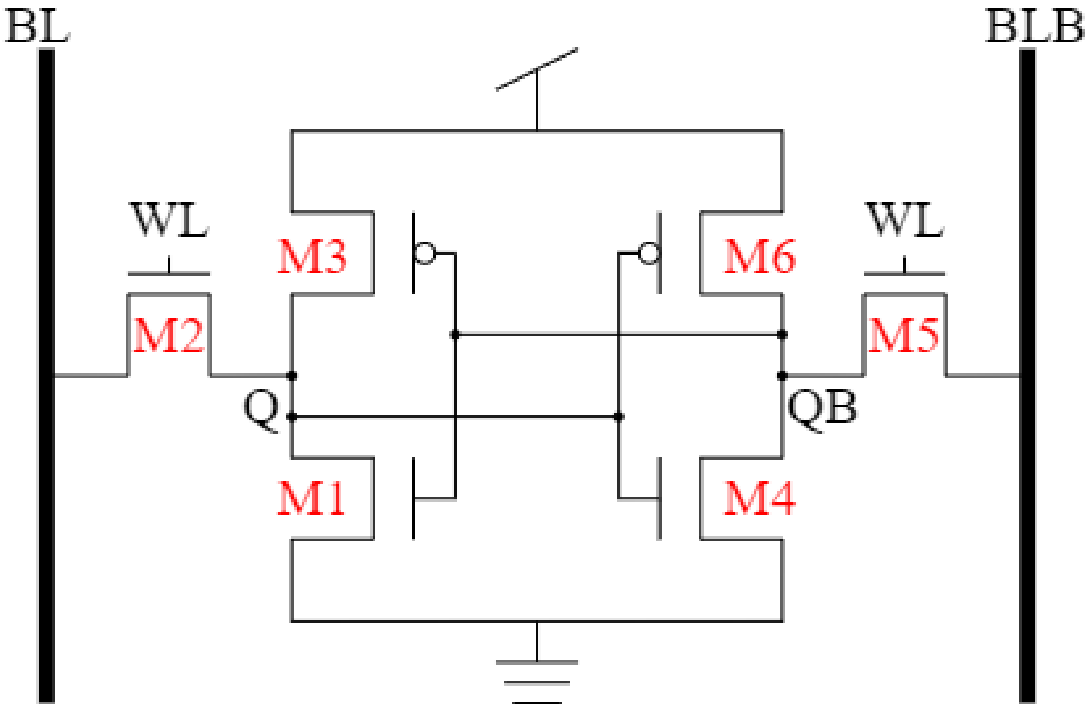


6T SRAM Cell

- ❑ 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

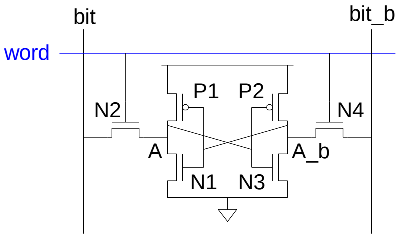


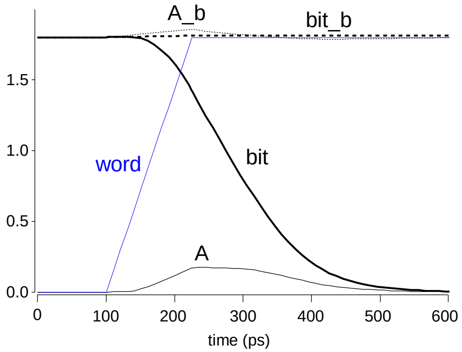
6T SRAM Cell



SRAM Read

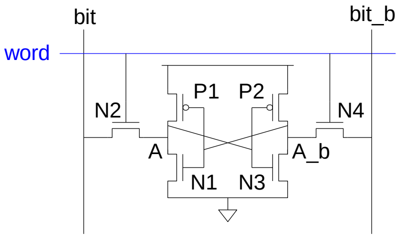
- ❑ Precharge both bitlines high
- ❑ Then turn on wordline
- ❑ One of the two bitlines will be pulled down by the cell
- ❑ Ex: A = 0, A_b = 1
  - – bit discharges, bit_b stays high
  - – But A bumps up slightly
- ❑ *Read stability*
  - – A must not flip

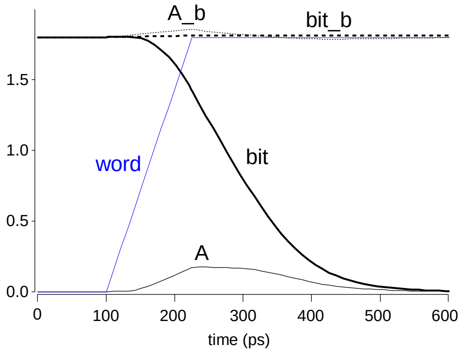




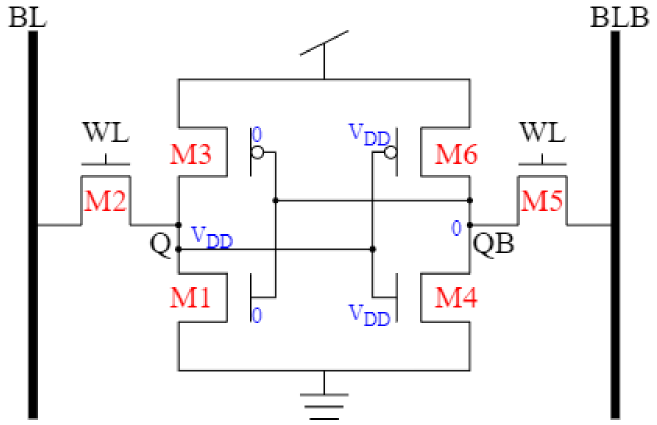
SRAM Read

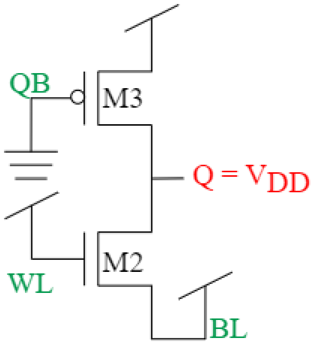
- ❑ Precharge both bitlines high
- ❑ Then turn on wordline
- ❑ One of the two bitlines will be pulled down by the cell
- ❑ Ex: A = 0, A_b = 1
  - – bit discharges, bit_b stays high
  - – But A bumps up slightly
- ❑ *Read stability*
  - – A must not flip
  - – N1 >> N2





SRAM Read



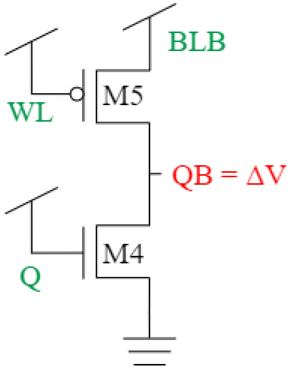


Left Side:

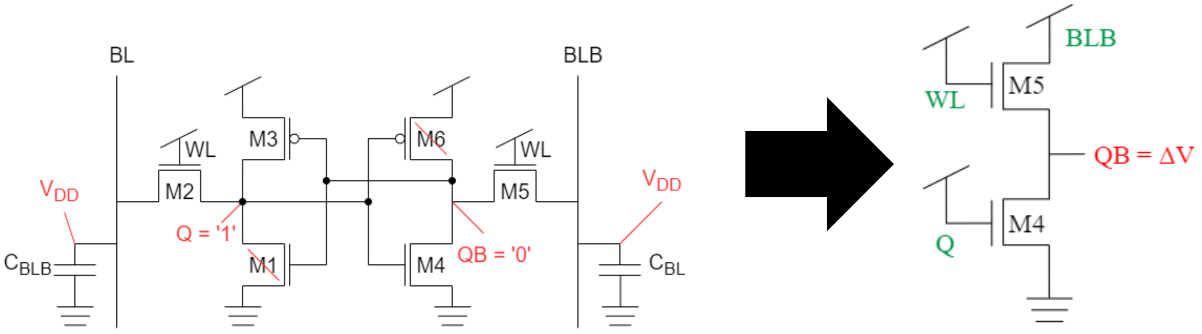
Nothing Changes

Right Side:

“nMOS” inverter – QB voltage rises



SRAM Read



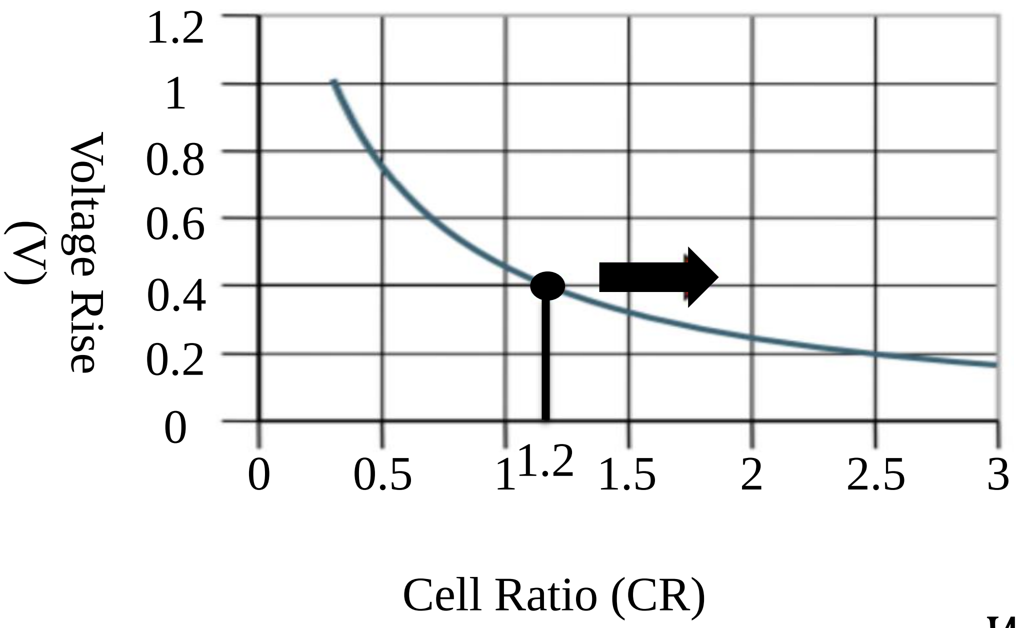
Cell Ratio:

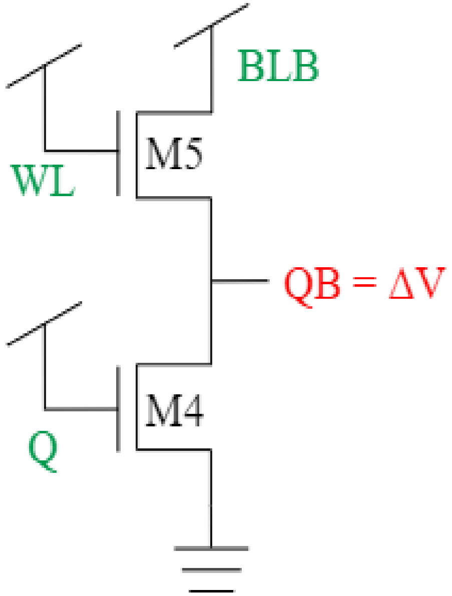
$$CR \equiv \frac{W_4/L_4}{W_5/L_5}$$

$$k_{M5}\left[\left(V_{DD} - \Delta V - V_{T,n}\right)V_{DSat,n} - \frac{V^2{}_{DSat,n}}{2}\right] = k_{M4}\left[\left(\mathrm{V}_{\mathrm{DD}} - \mathrm{V}_{\mathrm{T},n}\right)\Delta V - \frac{\Delta V^2}{2}\right]$$

$$\Delta V = \frac{V_{DSat,n} + CR\left(V_{DD} - V_{T,n}\right) - \sqrt{V^2{}_{DSat,n}(1 + CR) + CR^2\left(V_{DD} - V_{T,n}\right)^2}}{CR}$$

SRAM Read

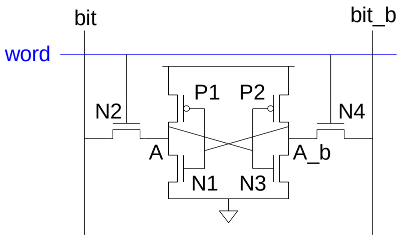


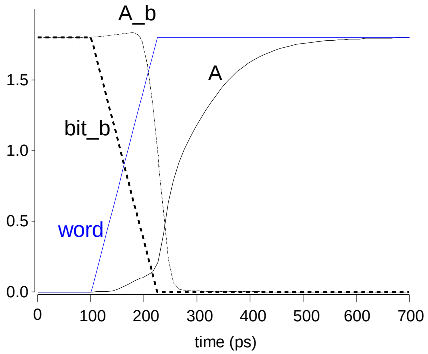


$$CR \equiv \frac{W_4/L_4}{W_5/L_5}$$

SRAM Write

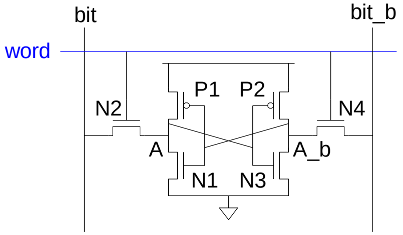
- ❑ Drive one bitline high, the other low
- ❑ Then turn on wordline
- ❑ Bitlines overpower cell with new value
- ❑ Ex: A = 0, A_b = 1, bit = 1, bit_b = 0
  - – Force A_b low, then A rises high
- ❑ *Writability*
  - – Must overpower feedback inverter

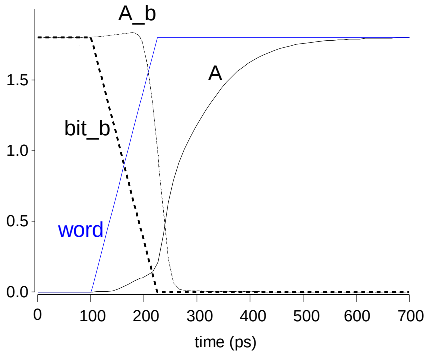




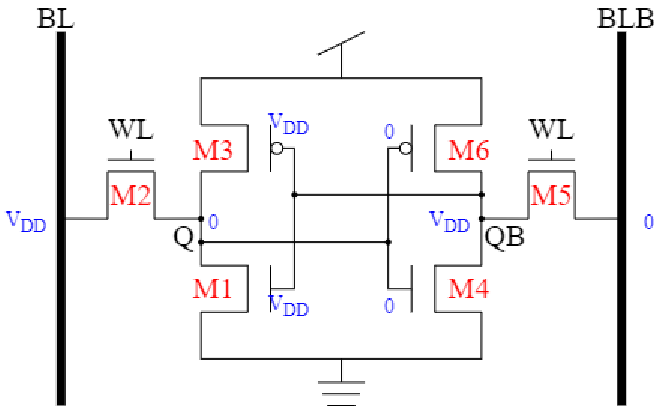
SRAM Write

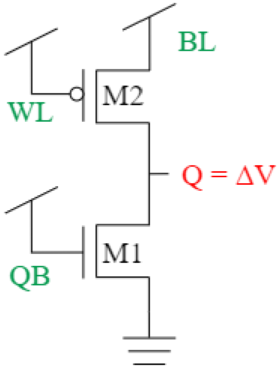
- Drive one bitline high, the other low
- Then turn on wordline
- Bitlines overpower cell with new value
- Ex: A = 0, A_b = 1, bit = 1, bit_b = 0
  - Force A_b low, then A rises high
- *Writability*
  - Must overpower feedback inverter
  - N2 >> P1





SRAM Write



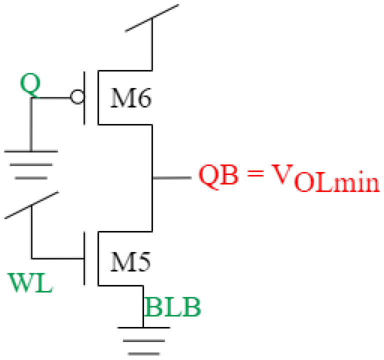


Left Side:

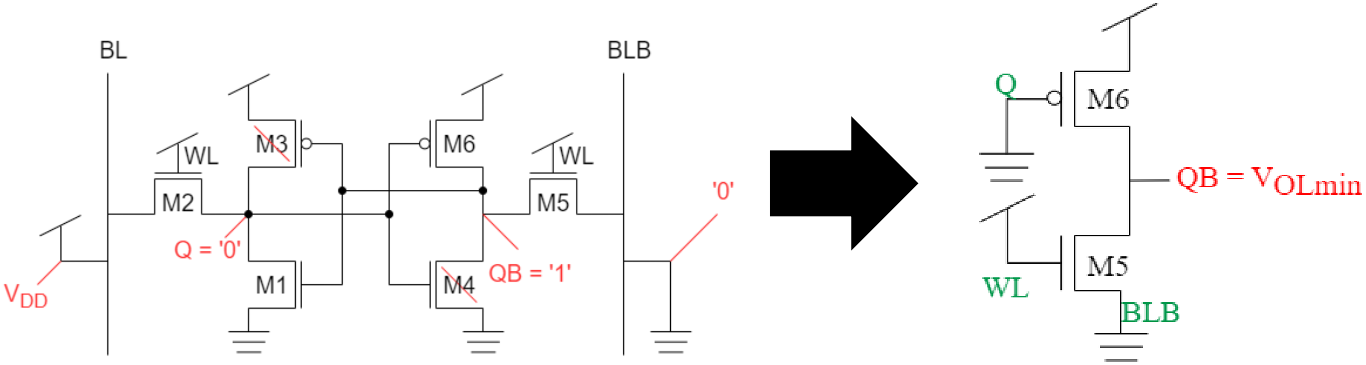
Same as doing read – designed so $\Delta V < V_M$

Right Side:

Pseudo nMOS inverter!



SRAM Write

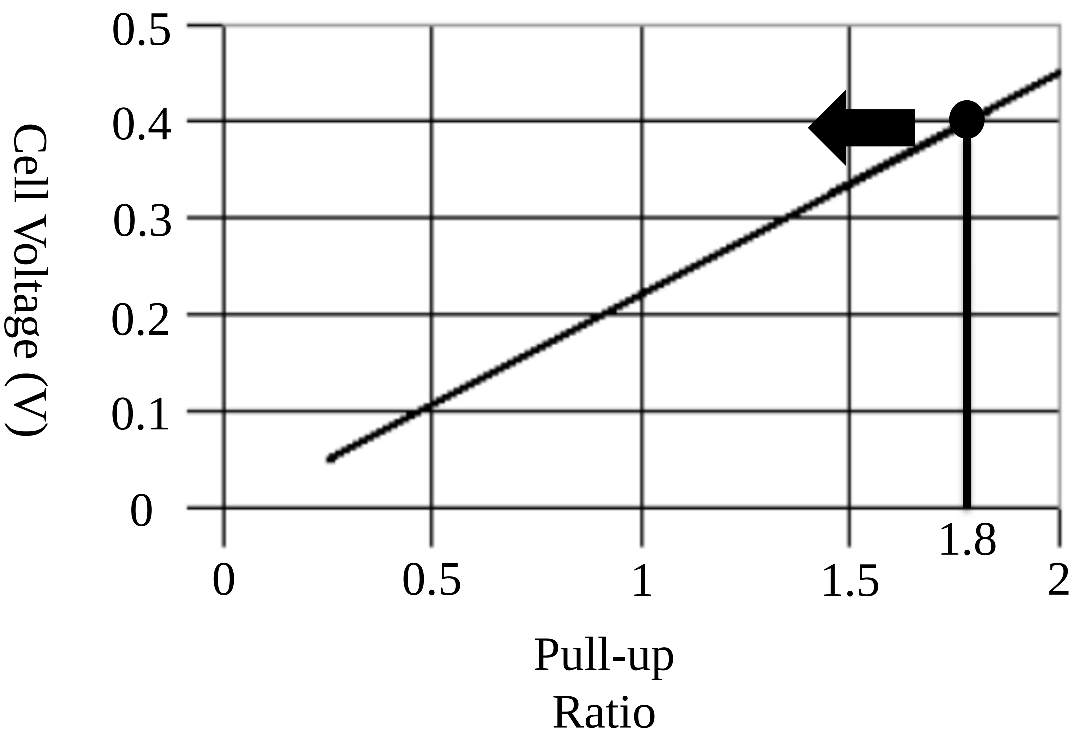


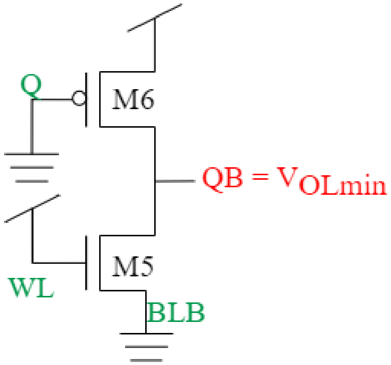
$$PR \equiv \frac{W_6/L_6}{W_5/L_5}$$

$$k_{M6}\left[(V_{DD} - |V_{T,p}|)V_{DSat,p} - \frac{V^2{}_{DSat,p}}{2}\right] = k_{M5}\left[(\mathrm{V_{DD}} - \mathrm{V}_{\mathrm{T},n})\Delta V_{QB} - \frac{\Delta V^2{}_{QB}}{2}\right]$$

$$V_{QB} = V_{DD} - V_{T,n} - \sqrt{(V_{DD} - V_{T,n})^2 - 2\frac{\mu_p}{\mu_n}PR\left[(V_{DD} - |V_{T,p}|)V_{DSat,p} - \frac{V^2{}_{DSat,p}}{2}\right]}$$

SRAM Write

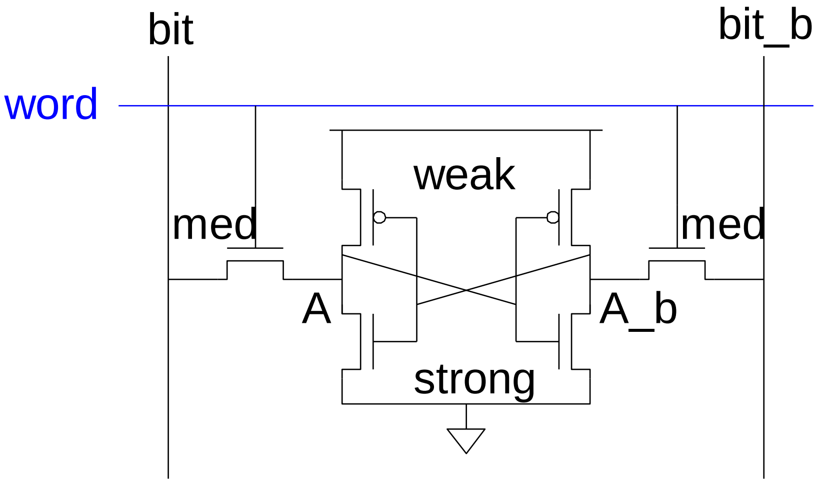




$$PR \equiv \frac{W_6/L_6}{W_5/L_5}$$

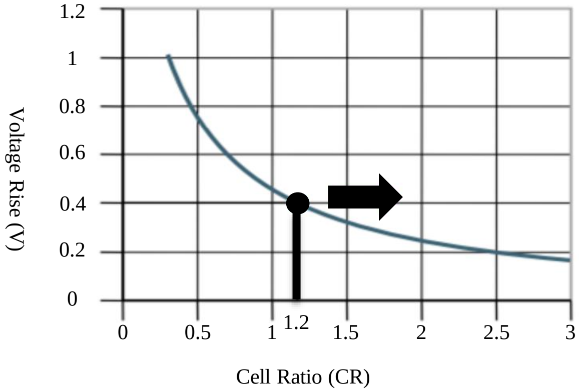
SRAM Sizing

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

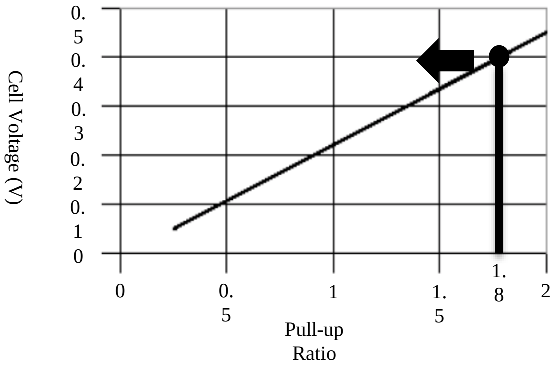


SRAM Sizing

Read Constraint



Write Constraint



$$CR \equiv \frac{W_1/L_1}{W_2/L_2} = \frac{W_4/L_4}{W_5/L_5} = \frac{PDN}{access}$$

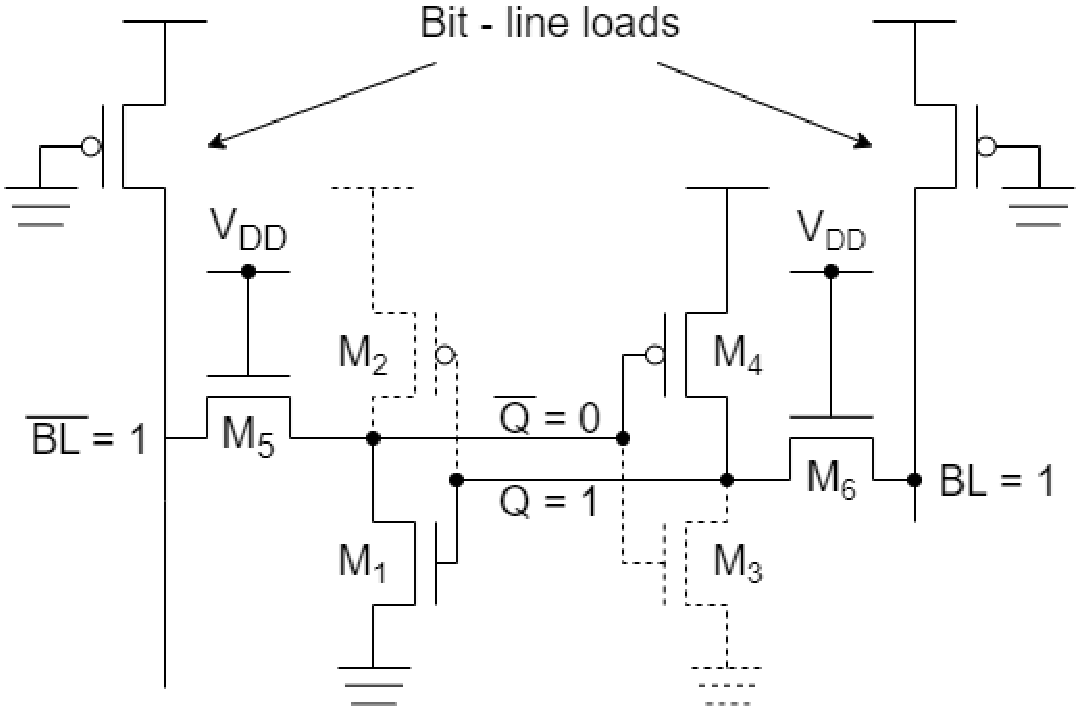
$$K_{PDN} > K_{access}$$

$$\boxed{K_{PDN} > K_{access} > K_{PUN}}$$

$$K_{access} > K_{PUN}$$

$$PR \equiv \frac{W_3/L_3}{W_2/L_2} = \frac{W_6/L_6}{W_5/L_5} = \frac{PUN}{access}$$

Simplified CMOS SRAM Analysis (Read)



Simplified CMOS SRAM Analysis (Read)

$\overline{\mathrm{Q}}$ node voltage can not over the transition voltage of M3 and M4 inverter i.e. $V_{\overline{Q}}$ < transition voltage (assume V_{DD} / 2) or < $V_{TN0.}$ So, the marginal condition is:

$$\frac{K_5}{2}\left(V_{DD} - V_{\overline{Q}} - V_{TN}\,|_{|V_{BS}| = V_{\overline{Q}}}\right)^2 = K_1\left[(V_{DD} - V_{TN0})V_{\overline{Q}} - \frac{{V_{\overline{Q}}}^2}{2}\right]$$

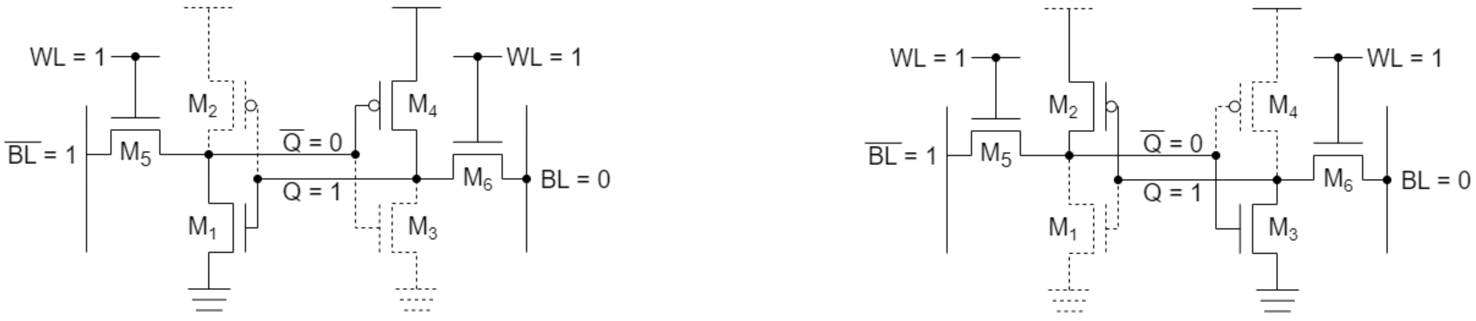
,where $V_{\overline{Q}} < V_t \;\; or \;\; V_{\overline{Q}} < V_{TN0}$

$Assume\ that\ V_{DD} = 3.3V\ and\ V_{TN0} = 0.59V$

$$V_{TN}|_{V_{BS} = 0.5V_{DD}} = 0.93V \;\; and \;\; V_{TN}|_{V_{BS} = V_{\mathrm{TN0}}} = 0.75V$$

$We\ obtain\ \mathrm{K}_5 < 12\mathrm{K}_1$ or $\mathrm{K}_5 < 0.8\mathrm{K}_1$ $,so\ we\ select\ \mathrm{W}_5 < 0.8\mathrm{W}_1$

Simplified CMOS SRAM Analysis (Write)



(a)

(b)

$$K_4\left[(\mathrm{V_{DD}} - ||\mathrm{V_{TP0}}||)(\mathrm{V_{DD}} - \mathrm{V_Q}) - \frac{(\mathrm{V_{DD}} - \mathrm{V_Q})^2}{2}\right] = K_6\left[(V_{DD} - V_{TN0})V_Q - \frac{{V_Q}^2}{2}\right]$$

$$\mathrm{V_Q} < \mathrm{V}_t$$

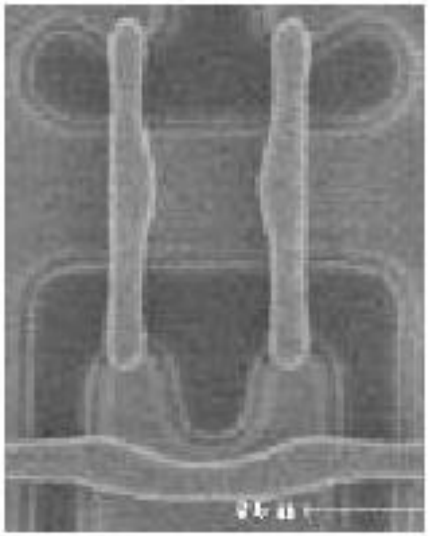
$$\frac{K_5}{2}\left(V_{DD} - V_{TN0} - V_{TN}\,|_{V_{\mathrm{BS}}=V_{TN0}}\right)^2 = K_1\left[(V_{DD} - V_{TN0})V_{\overline{Q}} - \frac{{V_{\overline{Q}}}^2}{2}\right]$$

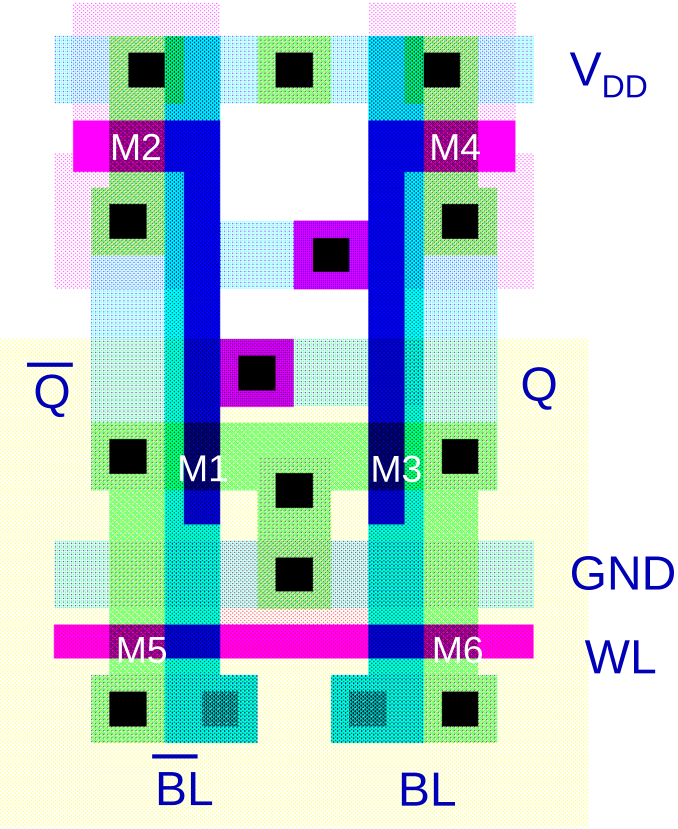
$$\mathrm{V}_{DD} = 3.3V\text{、}V_{TN0} = 0.59V \qquad \mathrm{V}_{TP0} = -0.72V \qquad V_Q < V_t \qquad K_4 \le 0.73K_6$$

$$\mu_p C_{ox}\left(\frac{W}{L}\right)_4 \le 0.73V\mu_n C_{ox}\left(\frac{W}{L}\right)_6 \quad \left(\frac{W}{L}\right)_4 \le 0.73\frac{\mu_n}{\mu_p}\left(\frac{W}{L}\right)_6 \quad or \quad \left(\frac{W}{L}\right)_4 \le 1.8\left(\frac{W}{L}\right)_6$$

6T-SRAM — Layout

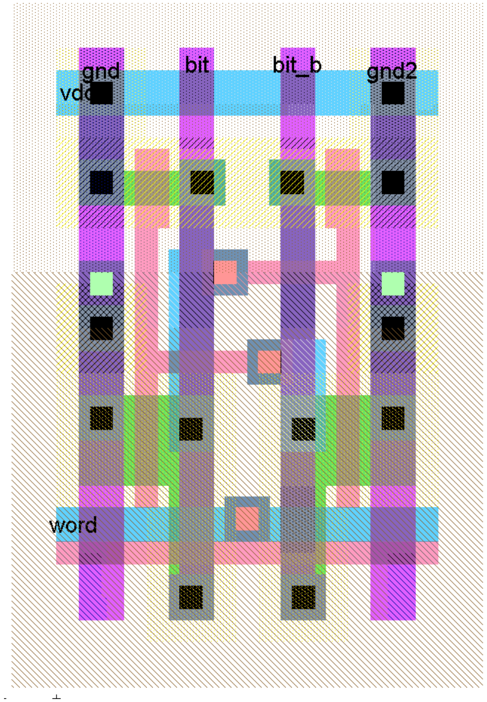
- Extremely dense
- Modern processes can fit a 6T SRAM cell in ~1.3μm^2

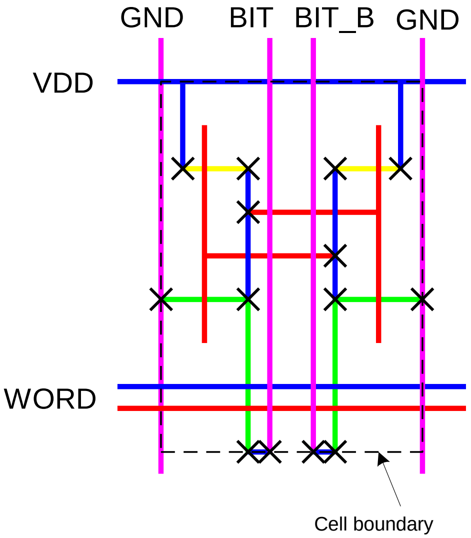


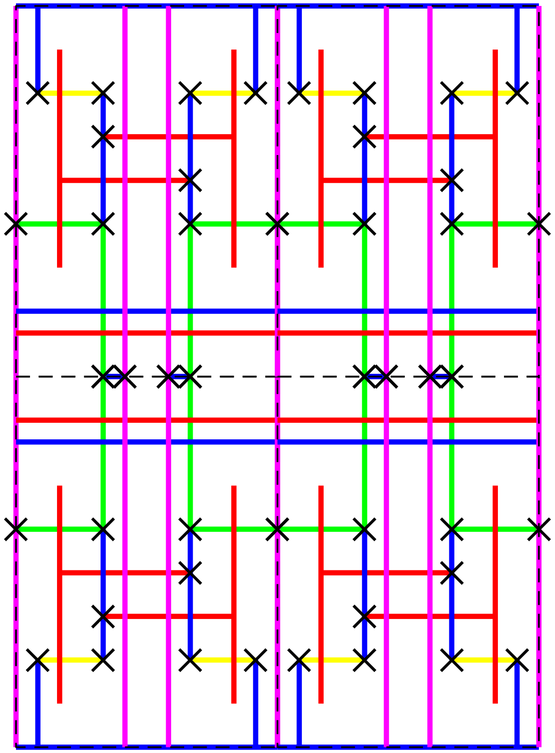


SRAM Layout

- Cell size is critical: 26 x 45 λ (even smaller in industry)
- Tile cells sharing V_{DD}, GND, bitline contacts



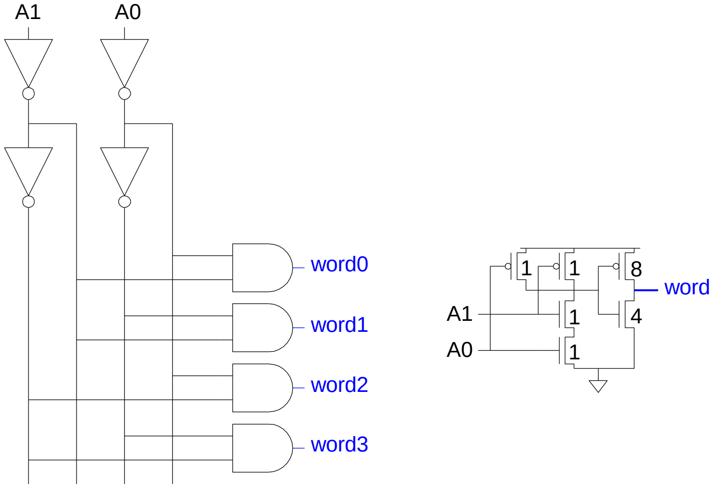




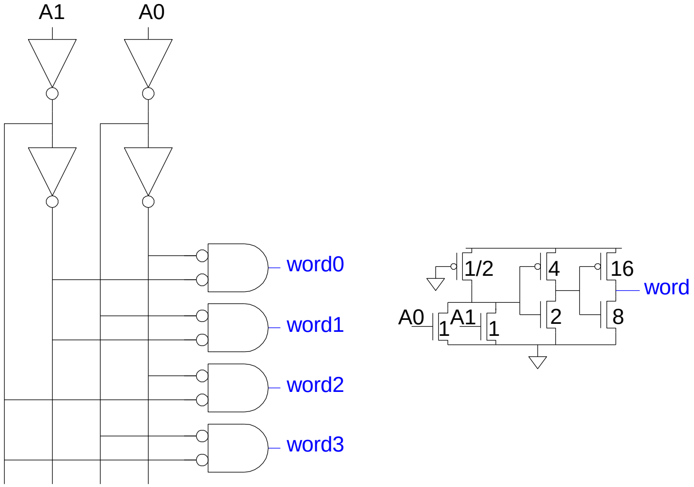
Decoders

- 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



ROW Decoder

- Standard Decoder Design
  - Each output row is driven by an AND gate has a unique combination of address inputs
  - For example, an 8-bit row address has 256 8-input AND gates

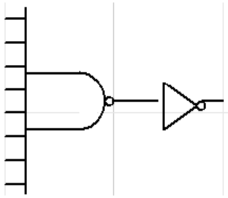
  WL0=/A7/A6/A5/A4/A3/A2/A1/A0

  WL255=A7A6A5A4A3A2A1A0
- NOR decoder
  - DeMorgan will provide a nor decoder

  WL0=/(A7+A6+A5+A4+A3+A2+A1+A0)

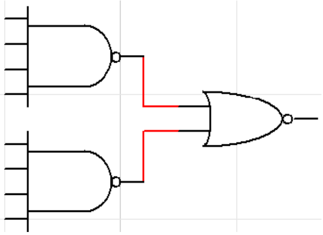
How Should We Build It

- Let’s build a row decoder for a 256x256 SRAM Array
  - We need 256 8-input AND gates
  - Each gate drives 256 bit cells
- Various options:



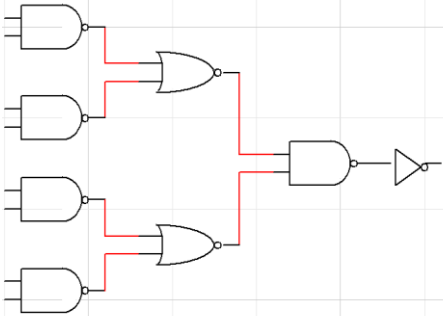
$\text{LE} = \frac{10}{3} \cdot 1 = \frac{10}{3}$

$P = 8 + 1 = 9$



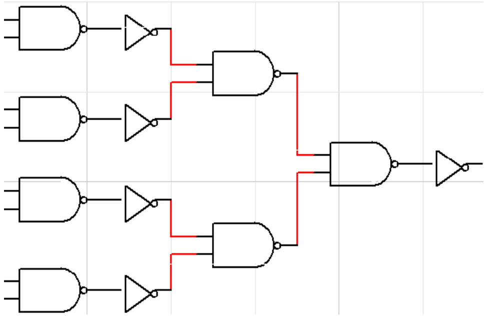
$\text{LE} = \frac{5}{3} \cdot 2 = \frac{10}{3}$

$P = 4 + 2 = 6$



$\text{LE} = \frac{4}{3} \cdot \frac{5}{3} \cdot \frac{4}{3} = \frac{80}{27}$

$P = 2 + 2 + 2 + 1$
$= 7$



$\text{LE} = (\frac{4}{3})^3 = 2.37$

$P = 2 \times 3 + 1 \times 3$
$= 9$

- Which one is best?

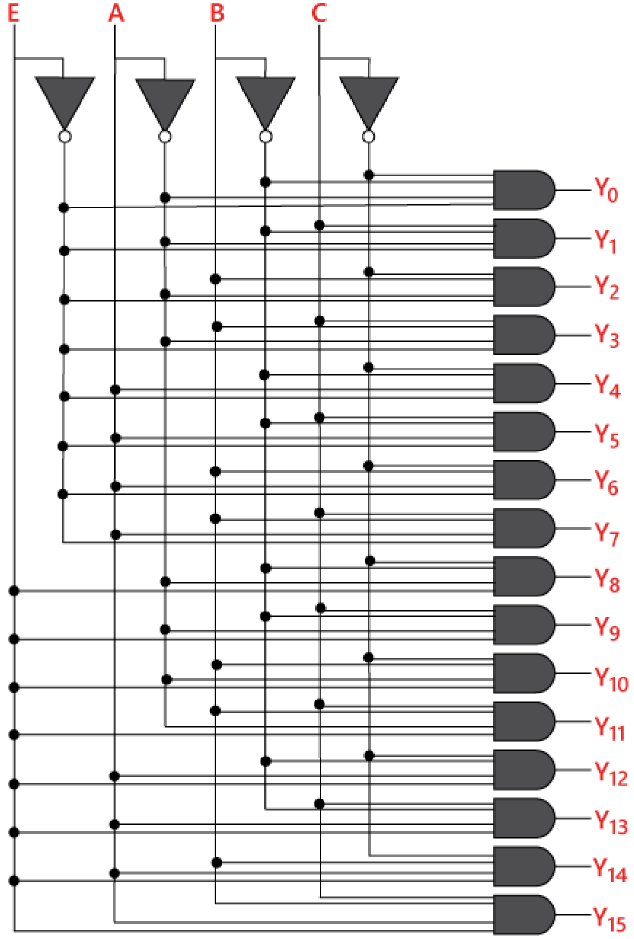
Discussion

- What is the Branching Effort?
  - Let’s take a look at the Boolean expression :

  WL0=/A7/A6/A5/A4/A3/A2/A1/A0
  WL255=A7A6A5A4A3A2A1A0

  - Each address driver drives 128 8-input AND gates

Discussion



Number of Stages

- The path effort: PE=LE*B*F=LE*128*(256/4)=LE*2^{13}

- The best case logical effort: LE=1

- The minimum number of stages for the optimal delay: N=LE*$log_{3.6} 2^{13}$ =7

Which Implementation

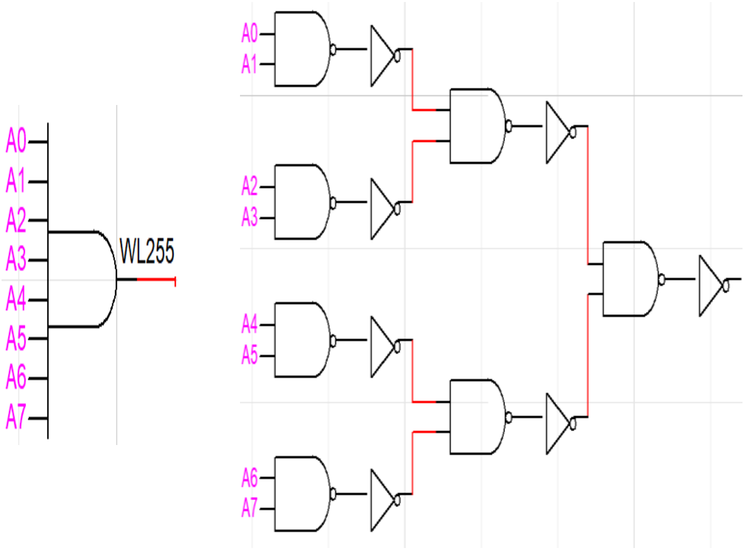
- The one with the minimum Logical Effort:

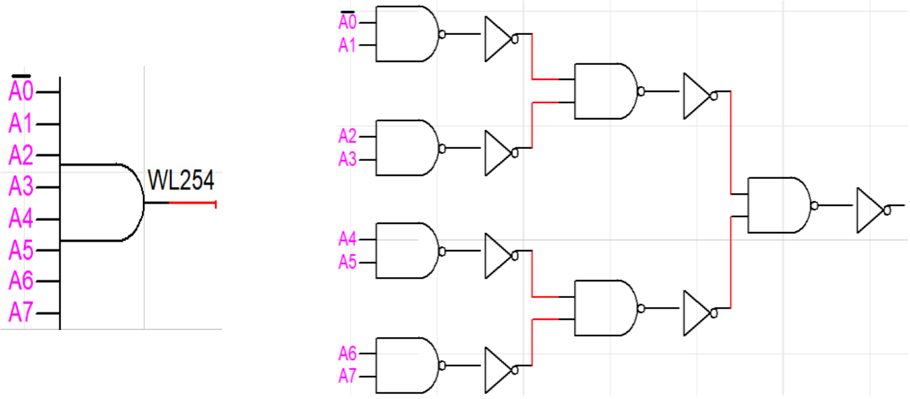
  PE=2.37* 2^{13}=19.418K
  N=$log_{3.6}$19.418K=7.7

- So now we can calculate the actual path effort
- We could add another inverter to get closer to the the optimal number of stages

Predecoding Concept

- Let’s look at two decoder paths:WL254 WL255





- We see there are many shared gate .
  So why not share them?

Predecoding Method

- Look at the final Boolean expression with combinations of groups of inputs.
- We actually create a small decoder by grouping together a few inputs.
- Then we just AND the outputs of all predecoder For example Two 4:16 predecoders

$\mathrm{D} = \mathrm{dec}(A_0，A_1，A_2，A_3)$ 、$\mathrm{E} = \mathrm{dec}(A_4，A_5，A_6，A_7)$

$\mathrm{WL}_0 = D_0 * E_0$；$\mathrm{WL}_{255} = D_{15} * E_{15}$；$\mathrm{WL}_{254} = D_{14} * E_{15}$；

Predecoding Example

- Look at the example:

- What is the new branching effort?
  - Each address drives half the lines of the small decoder
  - Each predecoder output drives 256/16 post-decoder gates
  - In summary, the branching effort is:

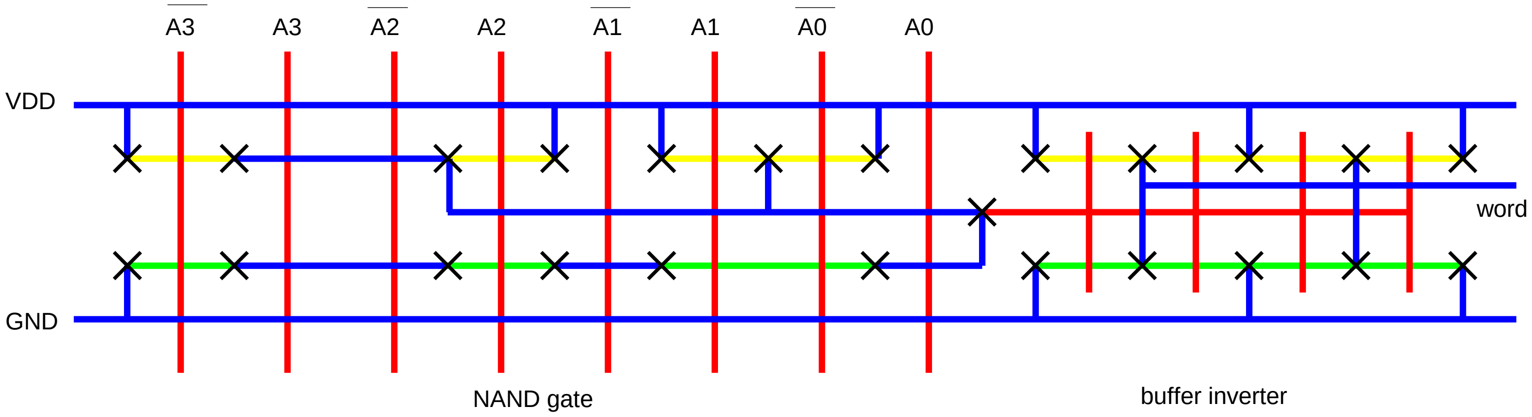
    B=baddr driver .bpredecoder=16/2 256/16=128
- Same as before

Precoding Example

- We can try using four 2-input predecoders:
  - This will require us to use 256 4-input NAND gates

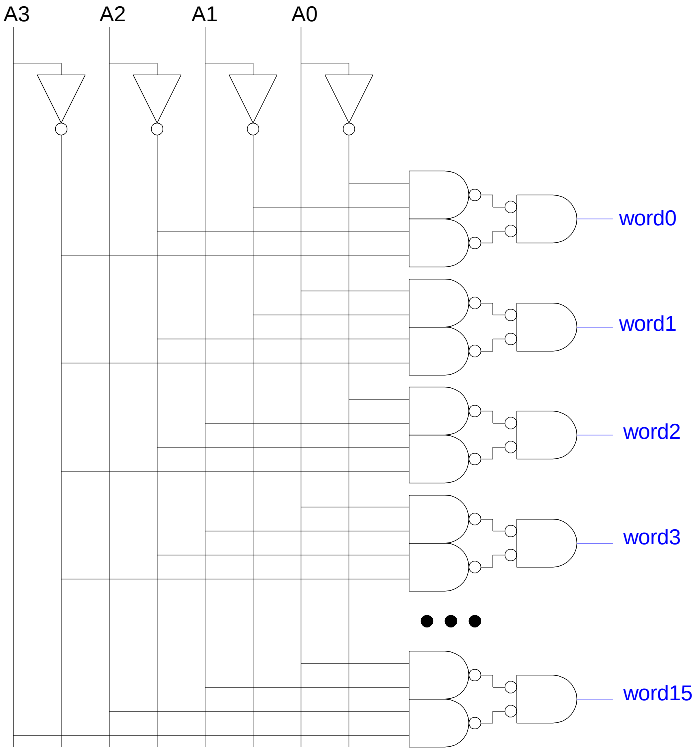
Decoder Layout

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



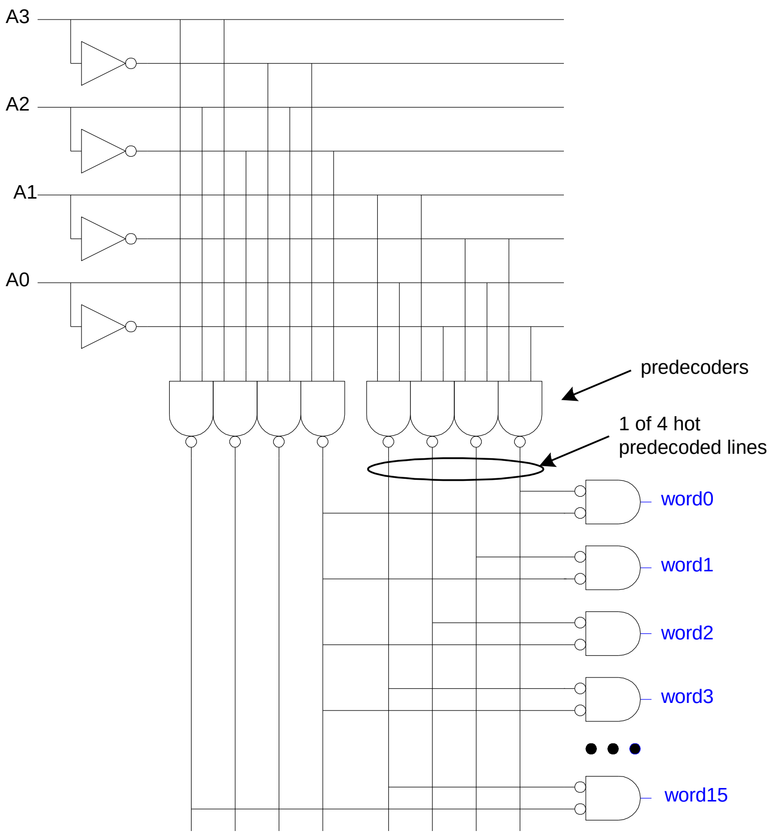
Large Decoders

- ❑ For n > 4, NAND gates become slow
  - – Break large gates into multiple smaller gates



Predecoding

- Many of these gates are redundant
  - Factor out common gates into predecoder
  - Saves area
  - Same path effort

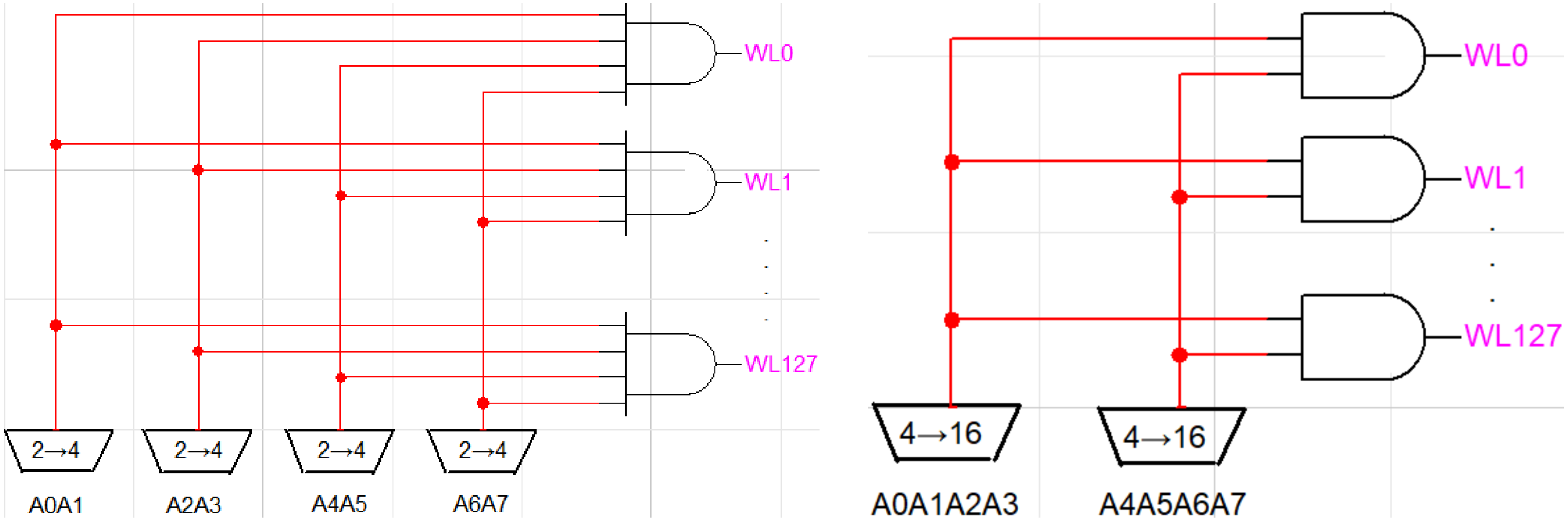


How to Choose A Configuration

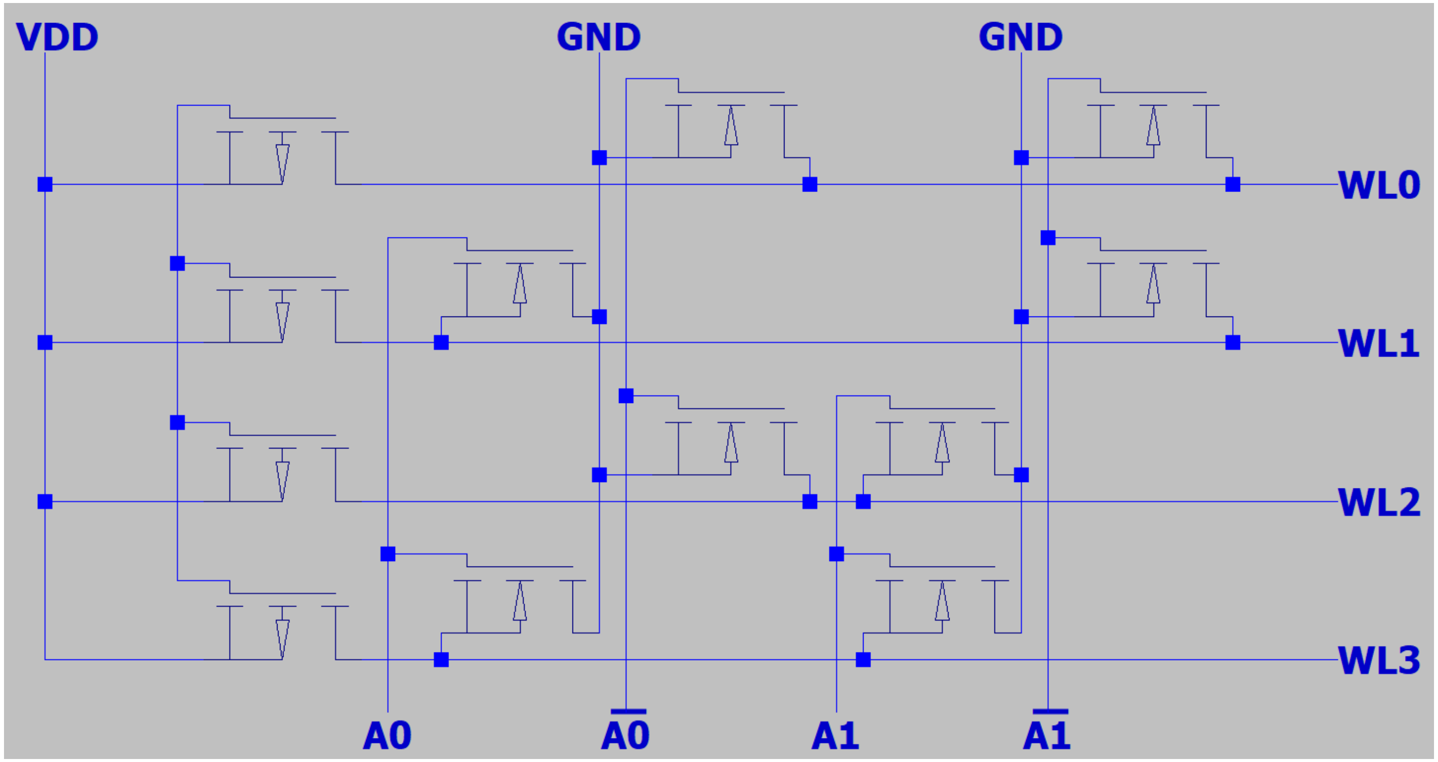
There are several factors to be considered-

- Pitch Fitting
- Switching Capacitance:
  How many switch at each transition?
- Stage before the large capacitor:
  Distribution of the load along the delay
- We usually do as much predecoding as possible

How to Choose A Configuration



Alternative Circuit Dynamic Decoder

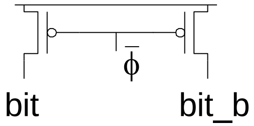


Column Circuitry

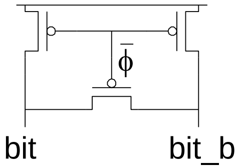
- Some circuitry is required for each column
  - Bitline conditioning
  - Sense amplifiers
  - Column multiplexing

Bitline Conditioning

- Precharge bitlines high before reads



- Equalize bitlines to minimize voltage difference when using sense amplifiers



Sense Amplifier: Why?

- □ Bit line cap significant for large array
  - – If each cell contributes 2fF,
    - • for 256 cells, 512fF plus wire cap
  - – Pull-down resistance is about 15K
  - – RC = 7.5ns! (assuming $\Delta V = V_{dd}$)
- □ Cannot easily change R, C, or V_{dd}, but can change ΔV i.e. smallest sensed voltage
  - – Can reliably sense ΔV as small as <50mV

Cell pull down resistance

$$\tau = \frac{RC\Delta V}{V_{dd}}$$

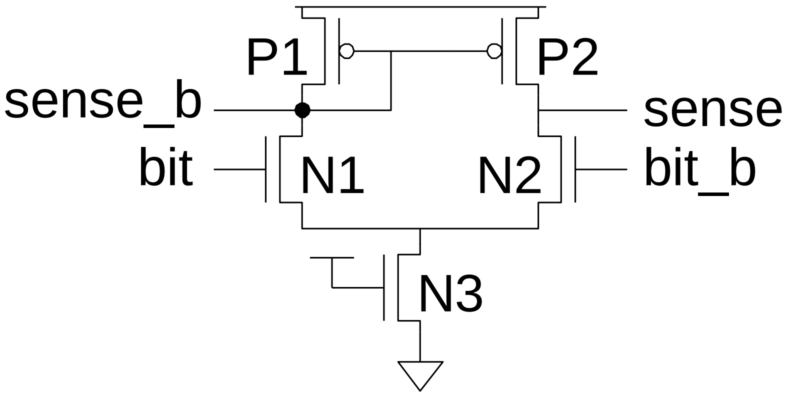
Cell current

Sense Amplifiers

- ❑ Bitlines have many cells attached
  - – Ex: 32-kbit SRAM has 256 rows x 128 cols
  - – 128 cells on each bitline
- ❑ $t_{pd} \propto (C/I)\ \Delta V$
  - – Even with shared diffusion contacts, 64C of diffusion capacitance (big C)
  - – Discharged slowly through small transistors (small I)
- ❑ *Sense amplifiers* are triggered on small voltage swing (reduce ΔV)

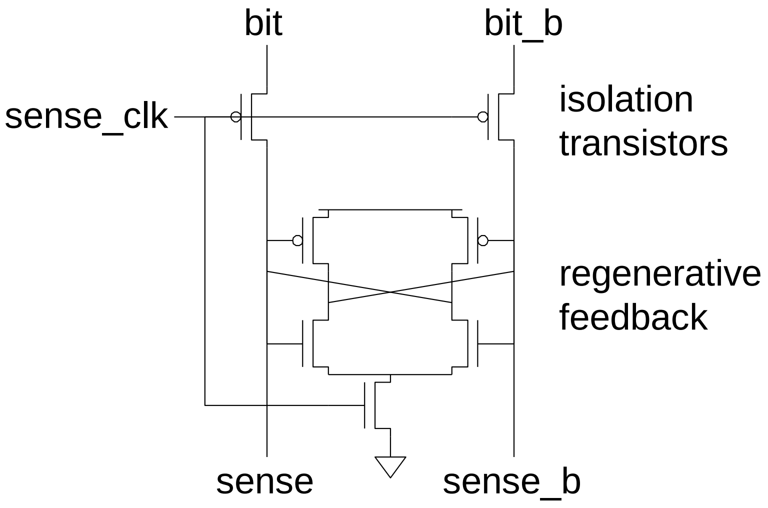
Differential Pair Amp

- ❑ Differential pair requires no clock
- ❑ But always dissipates static power



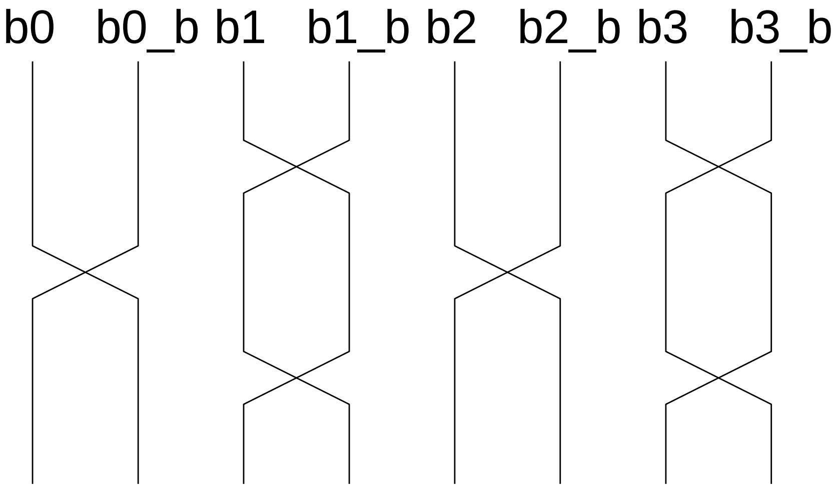
Clocked Sense Amp

- ❑ Clocked sense amp saves power
- ❑ Requires sense_clk after enough bitline swing
- ❑ Isolation transistors cut off large bitline capacitance



Twisted Bitlines

- ❑ Sense amplifiers also amplify noise
  - – Coupling noise is severe in modern processes
  - – Try to couple equally onto bit and bit_b
  - – Done by *twisting* bitlines

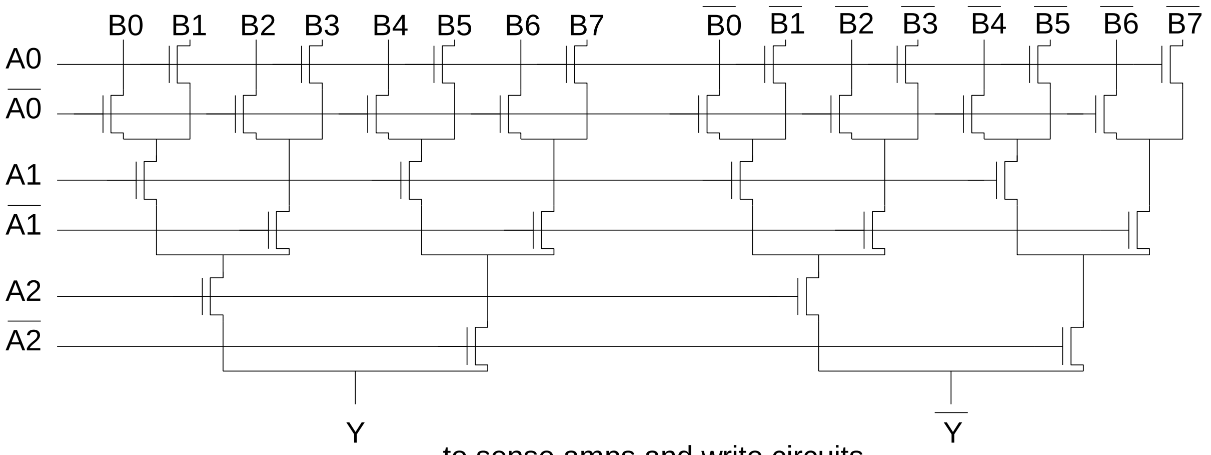


Column Multiplexing

- Recall that array may be folded for good aspect ratio
- Ex: 2 kword x 16 folded into 256 rows x 128 columns
  - Must select 16 output bits from the 128 columns
  - Requires 16 8:1 column multiplexers

Tree Decoder Mux

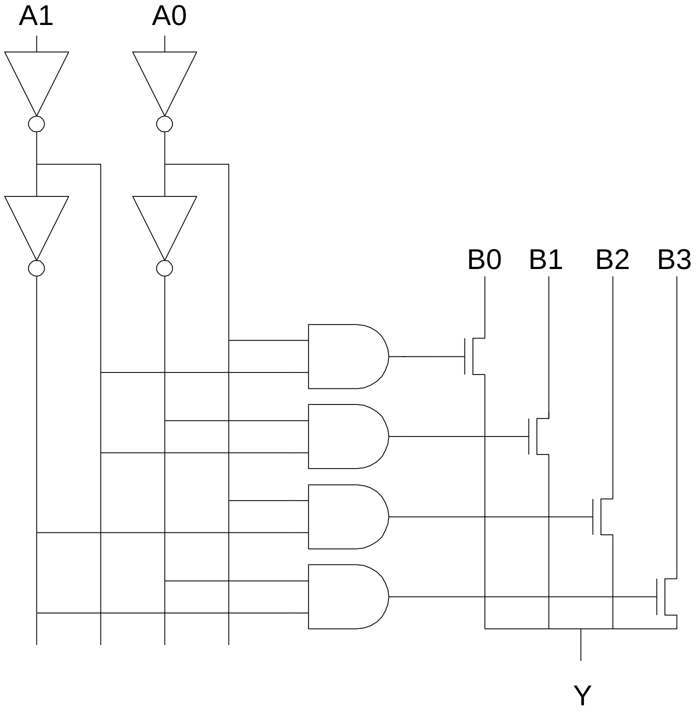
- Column mux can use pass transistors
  - Use nMOS only, precharge outputs
- One design is to use k series transistors for 2^k:1 mux
  - No external decoder logic needed (big area reduction)



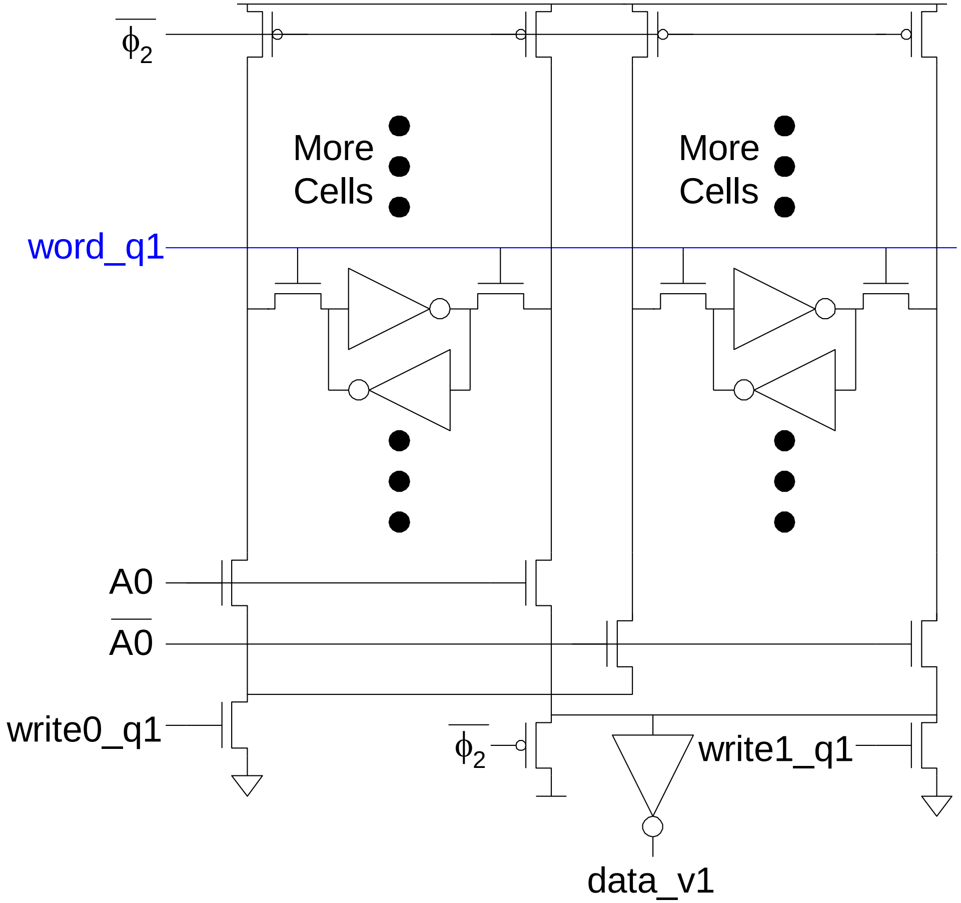
to sense amps and write circuits

Single Pass-Gate Mux

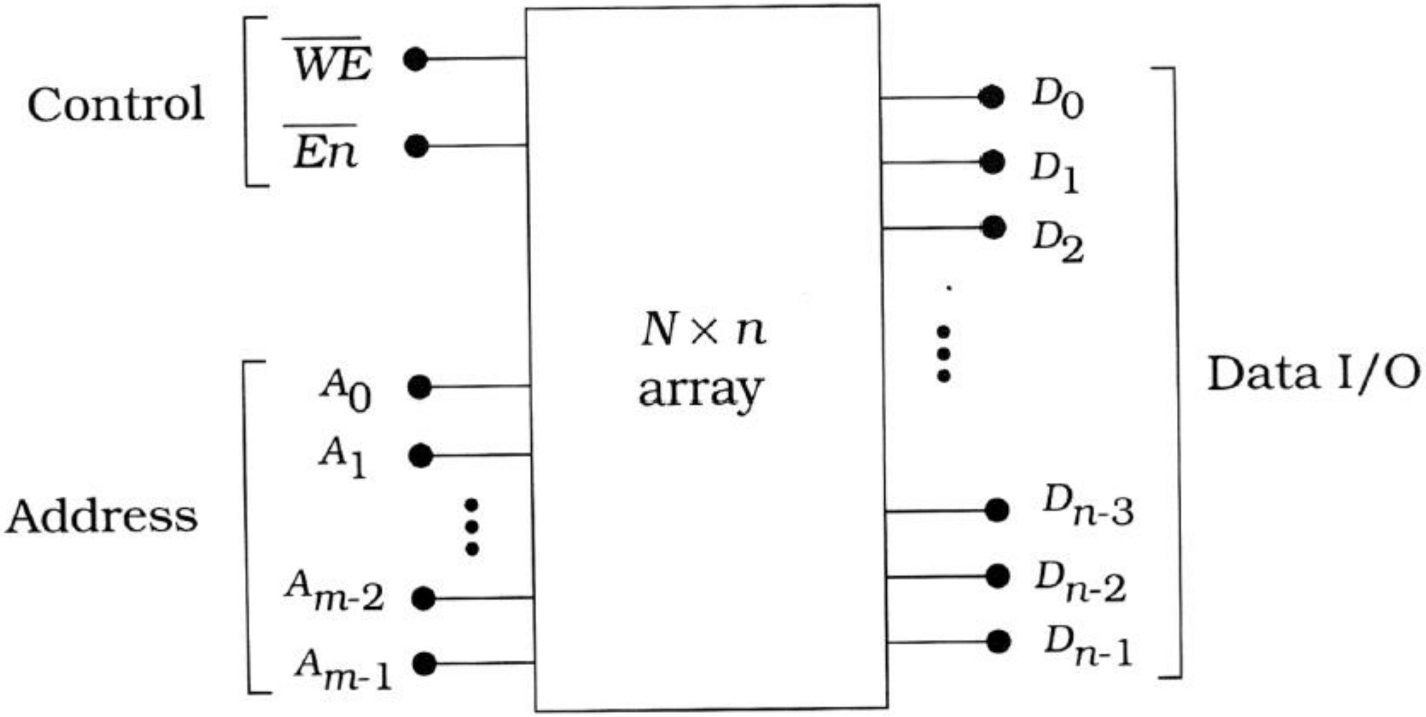
- ❑ Or eliminate series transistors with separate decoder



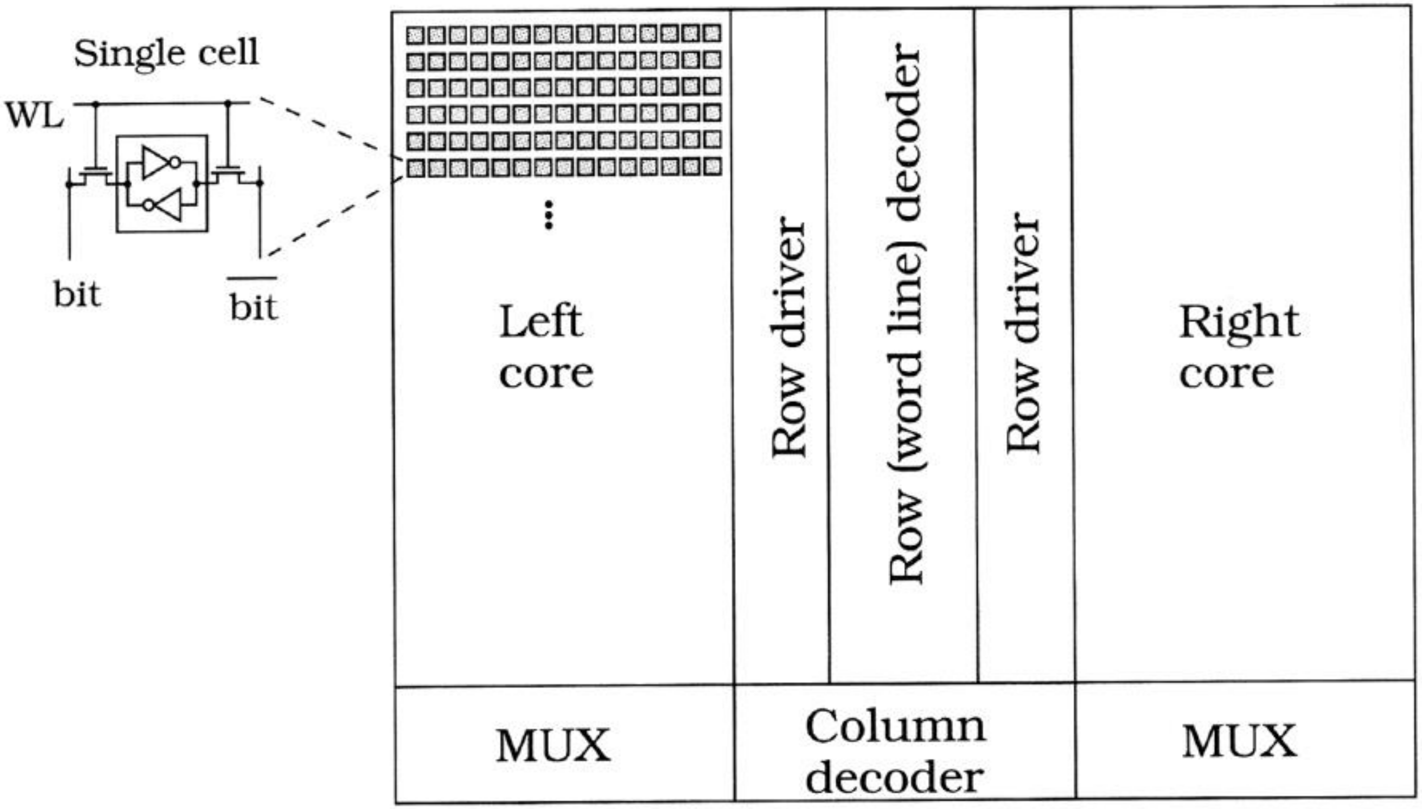
Ex: 2-way Muxed SRAM



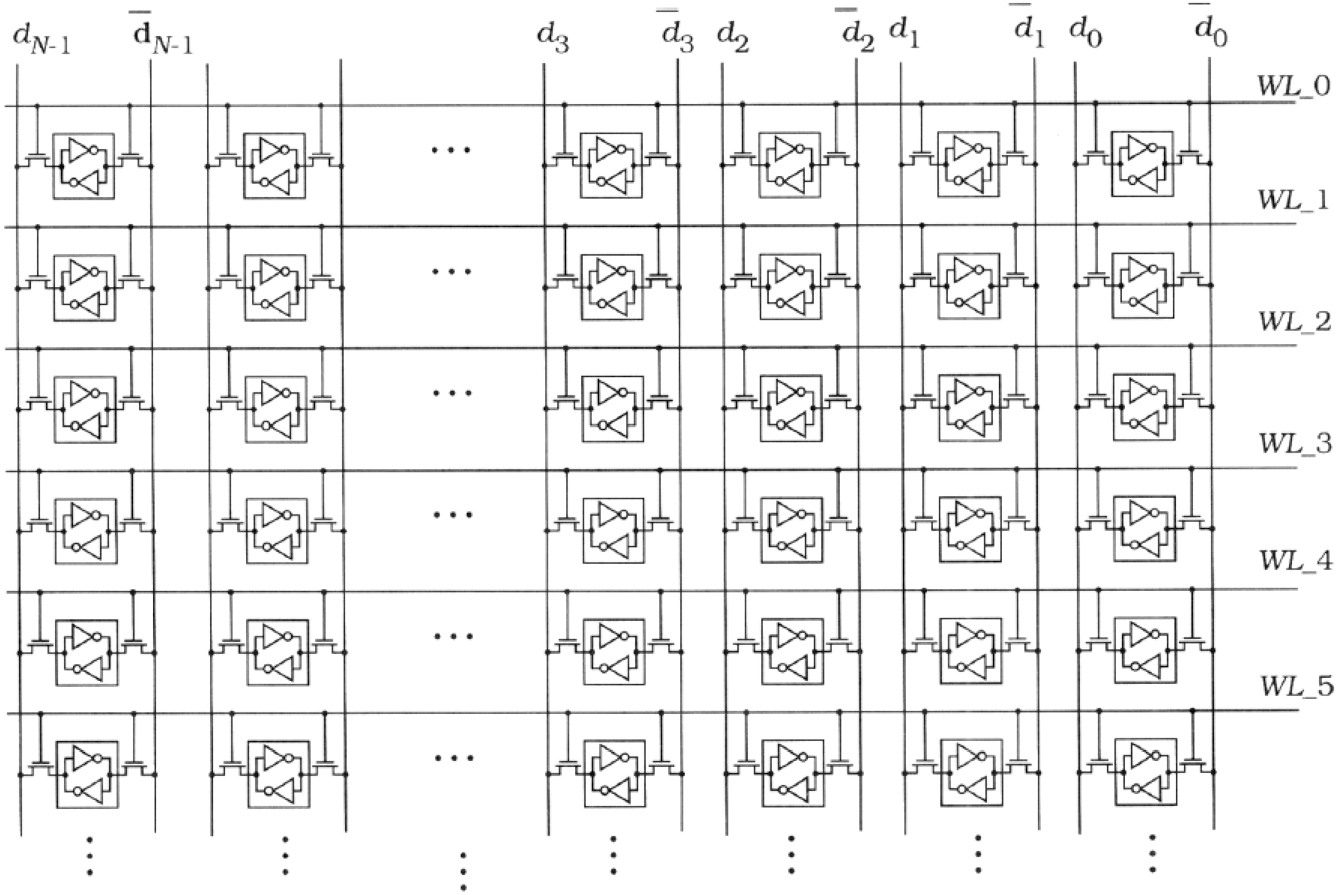
High-level view of an SRAM



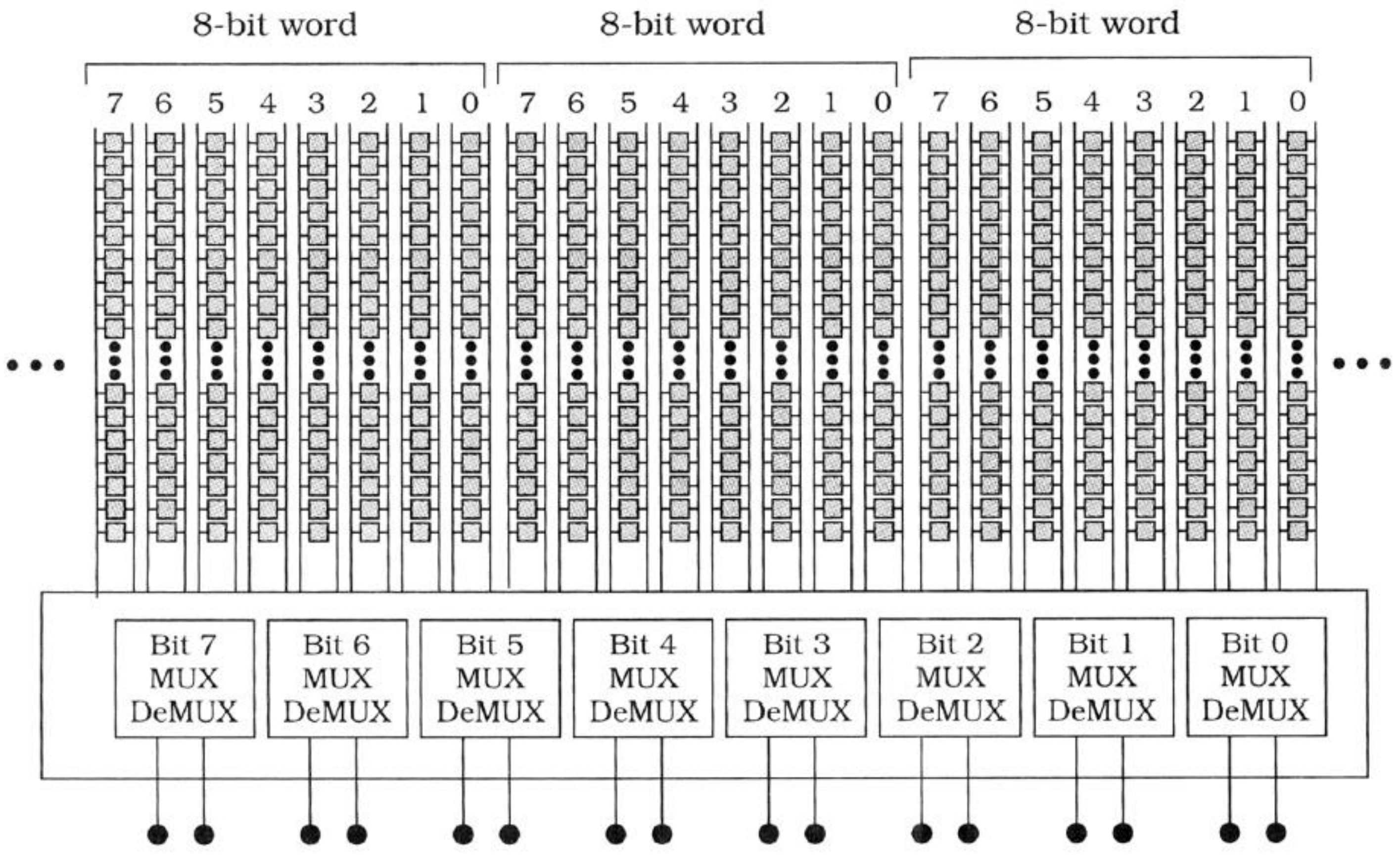
Central SRAM Block Architecture



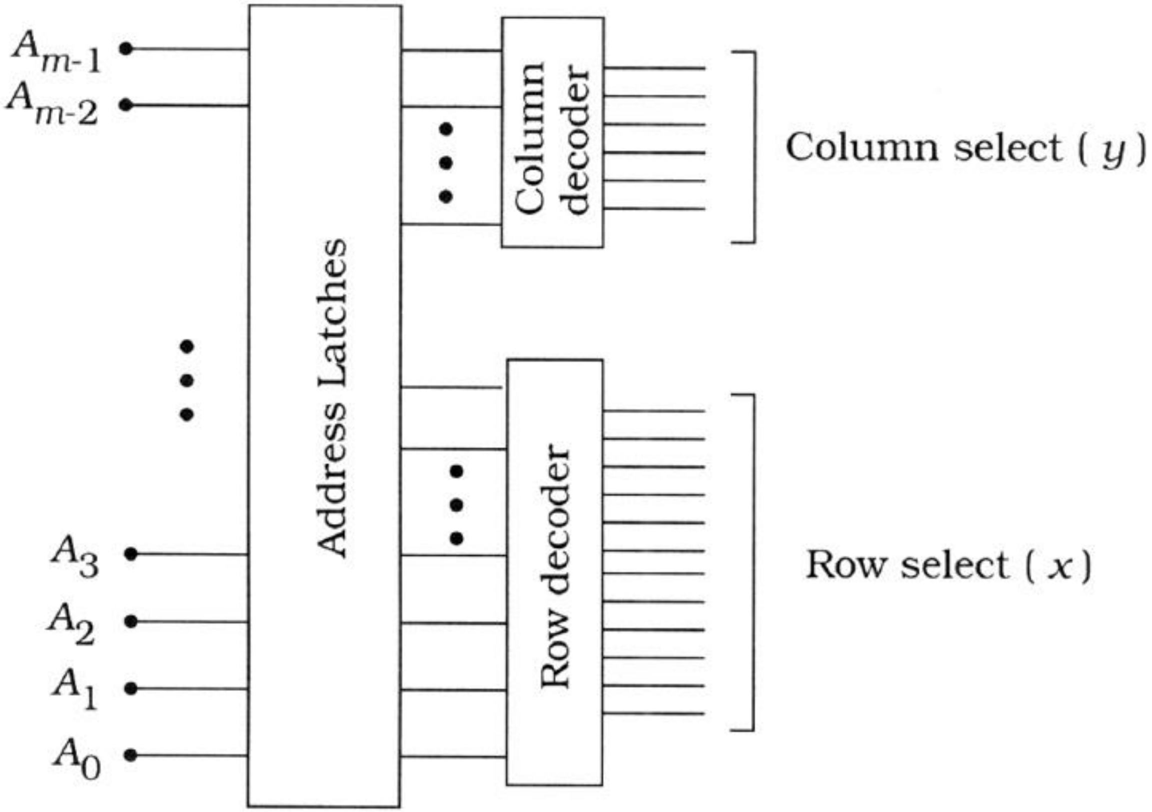
Cell Arrangement in a Core Region



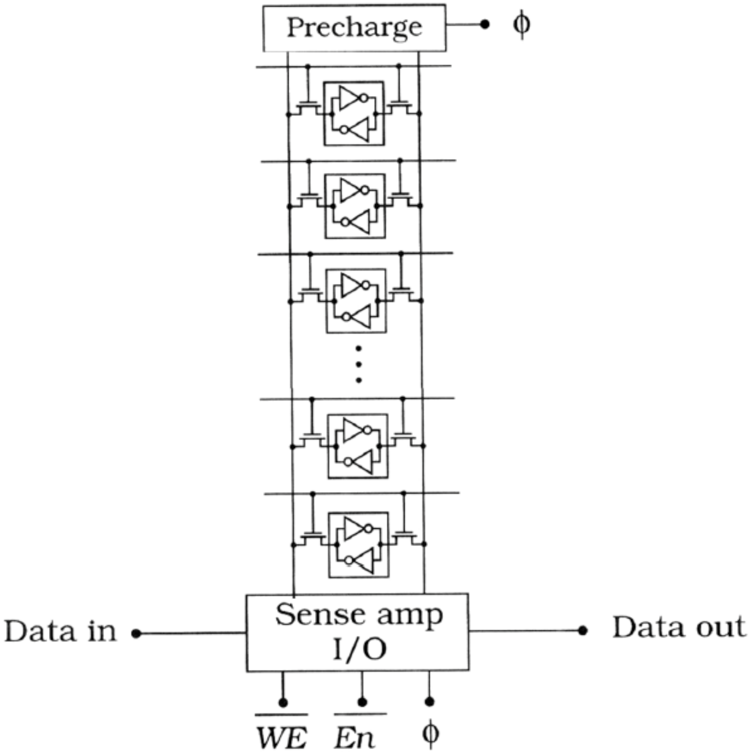
Column MUX/DeMUX network for 8-bit Words



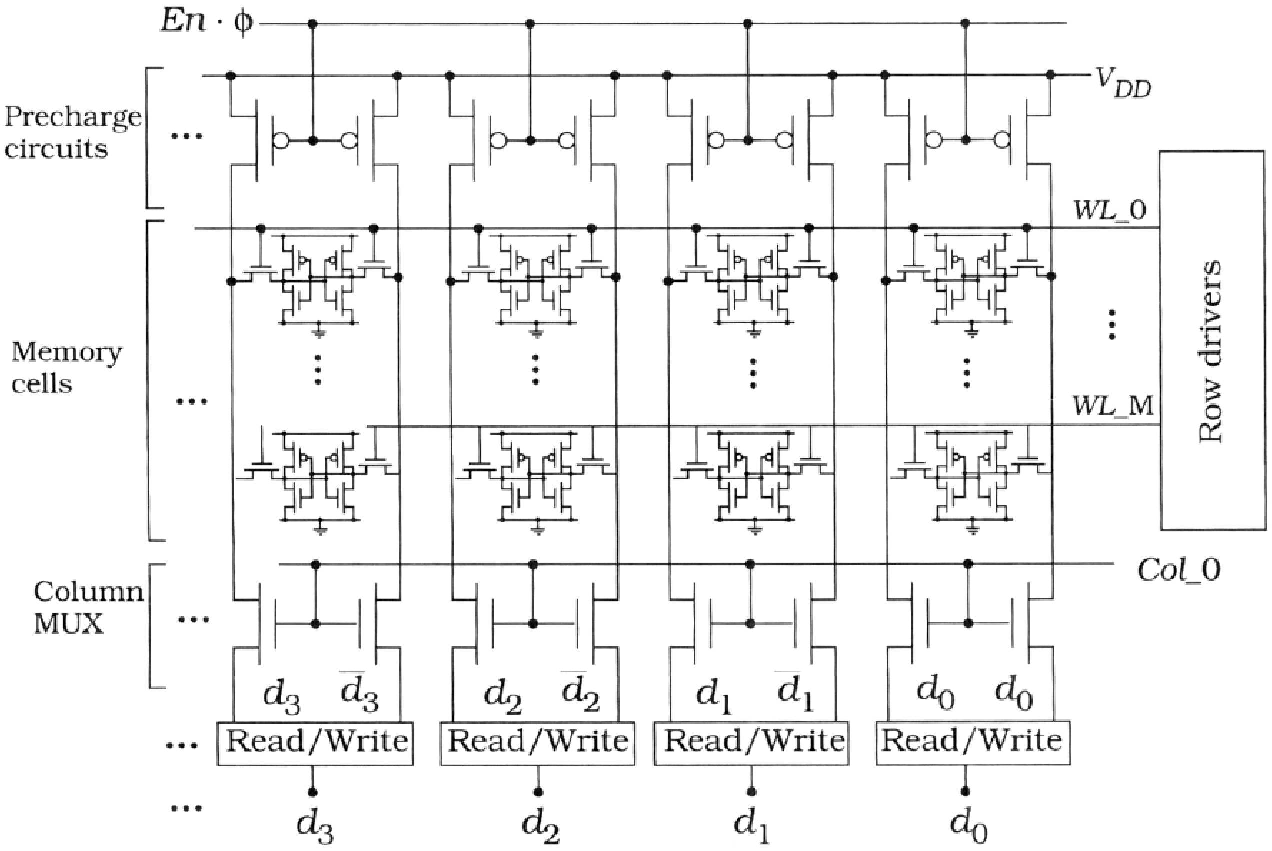
Basic Address Scheme



Precharge and I/O Circuits for a Single Column



Example View of Column Circuitry

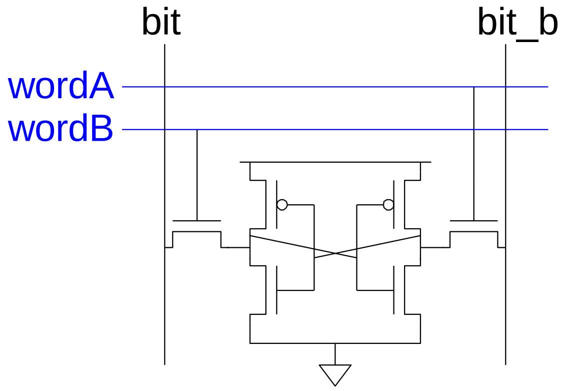


Multiple Ports

- ❑ We have considered single-ported SRAM
  - – One read or one write on each cycle
- ❑ *Multiported* SRAM are needed for register files
- ❑ Examples:
  - – Multicycle MIPS must read two sources or write a result on some cycles
  - – Pipelined MIPS must read two sources and write a third result each cycle
  - – Superscalar MIPS must read and write many sources and results each cycle

Dual-Ported SRAM

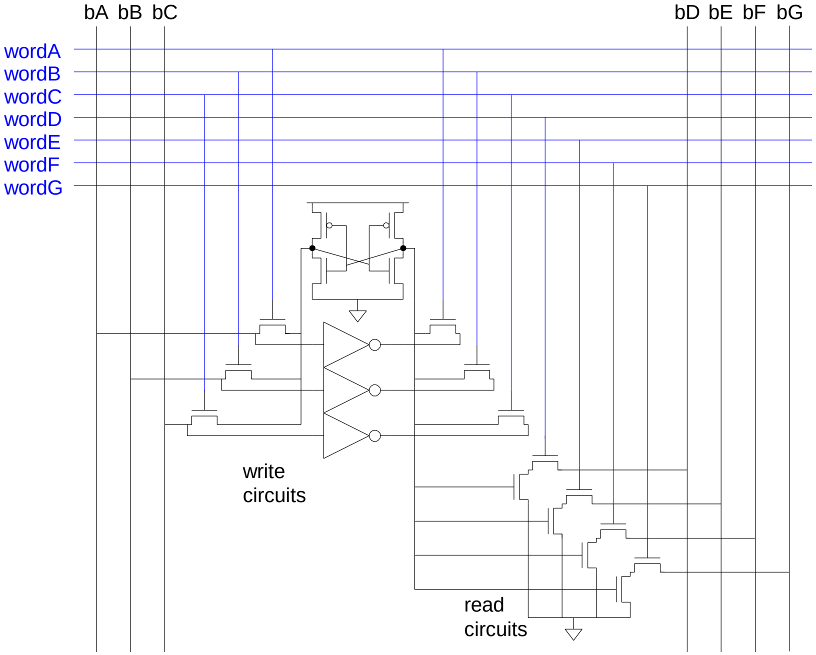
- ❑ Simple dual-ported SRAM
  - – Two independent single-ended reads
  - – Or one differential write



- ❑ Do two reads and one write by time multiplexing
  - – Read during ph1, write during ph2

Multi-Ported SRAM

- ❑ Adding more access transistors hurts read stability
- ❑ Multiported SRAM isolates reads from state node
- ❑ Single-ended design minimizes number of bitlines

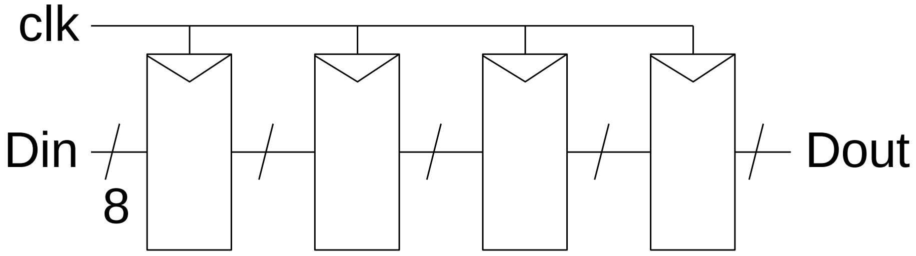


Serial Access Memories

- ❑ Serial access memories do not use an address
  - – Shift Registers
  - – Tapped Delay Lines
  - – Serial In Parallel Out (SIPO)
  - – Parallel In Serial Out (PISO)
  - – Queues (FIFO, LIFO)

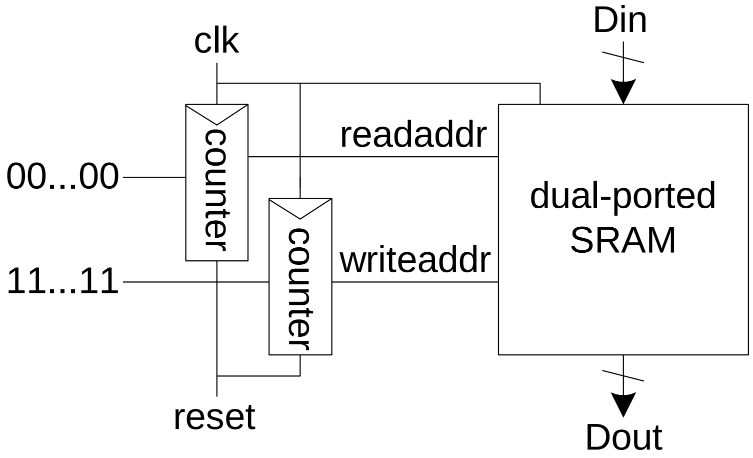
Shift Register

- *Shift registers* store and delay data
- Simple design: cascade of registers
  - – Watch your hold times!



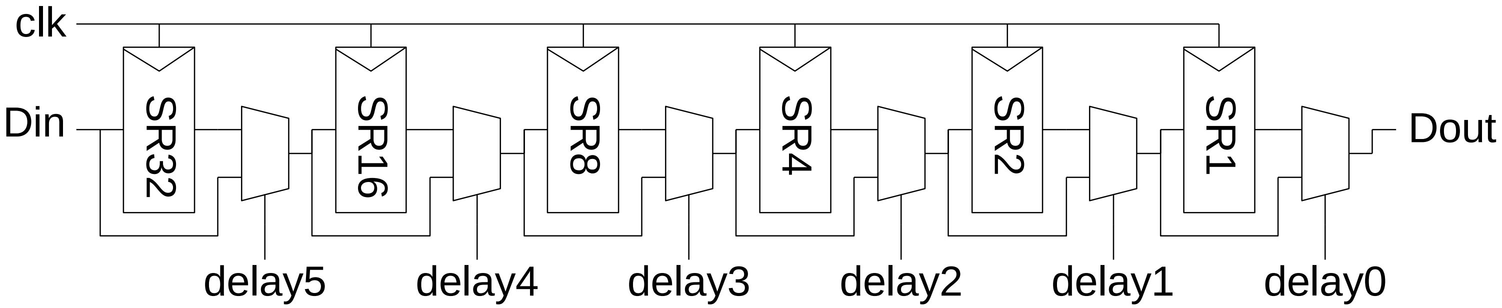
Denser Shift Registers

- ❑ Flip-flops aren’ t very area-efficient
- ❑ For large shift registers, keep data in SRAM instead
- ❑ Move read/write pointers to RAM rather than data
  - – Initialize read address to first entry, write to last
  - – Increment address on each cycle



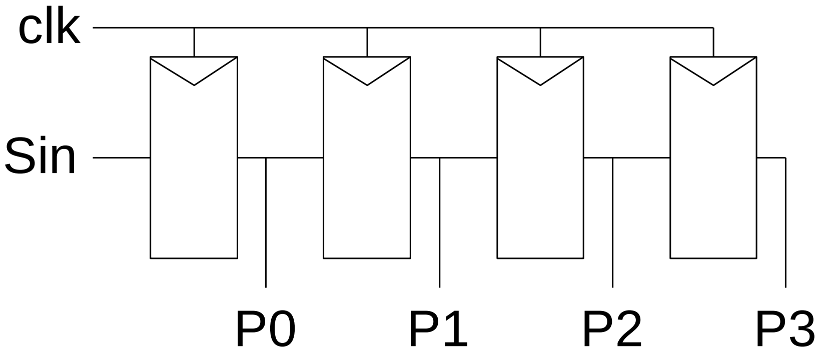
Tapped Delay Line

- ❑ A *tapped delay line* is a shift register with a programmable number of stages
- ❑ Set number of stages with delay controls to mux
  - – Ex: 0 – 63 stages of delay



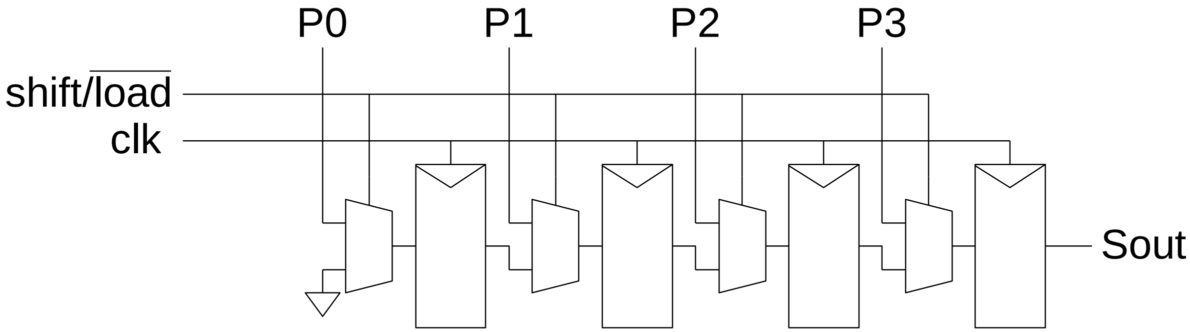
Serial In Parallel Out

- 1-bit shift register reads in serial data
  - After N steps, presents N-bit parallel output



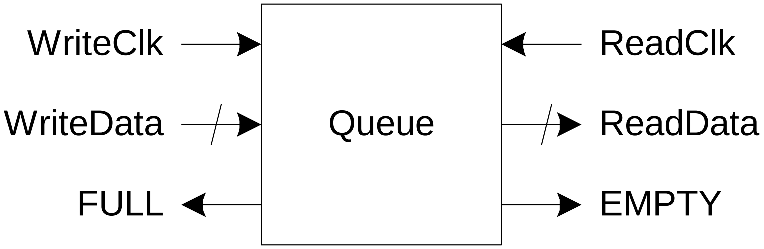
Parallel In Serial Out

- Load all N bits in parallel when shift = 0
  - Then shift one bit out per cycle



Queues

- *Queues* allow data to be read and written at different rates.
- Read and write each use their own clock, data
- Queue indicates whether it is full or empty
- Build with SRAM and read/write counters (pointers)



FIFO, LIFO Queues

- *First In First Out* (FIFO)
  - Initialize read and write pointers to first element
  - Queue is EMPTY
  - On write, increment write pointer
  - If write almost catches read, Queue is FULL
  - On read, increment read pointer
- *Last In First Out* (LIFO)
  - Also called a *stack*
  - Use a single *stack pointer* for read and write