Digital IC Design

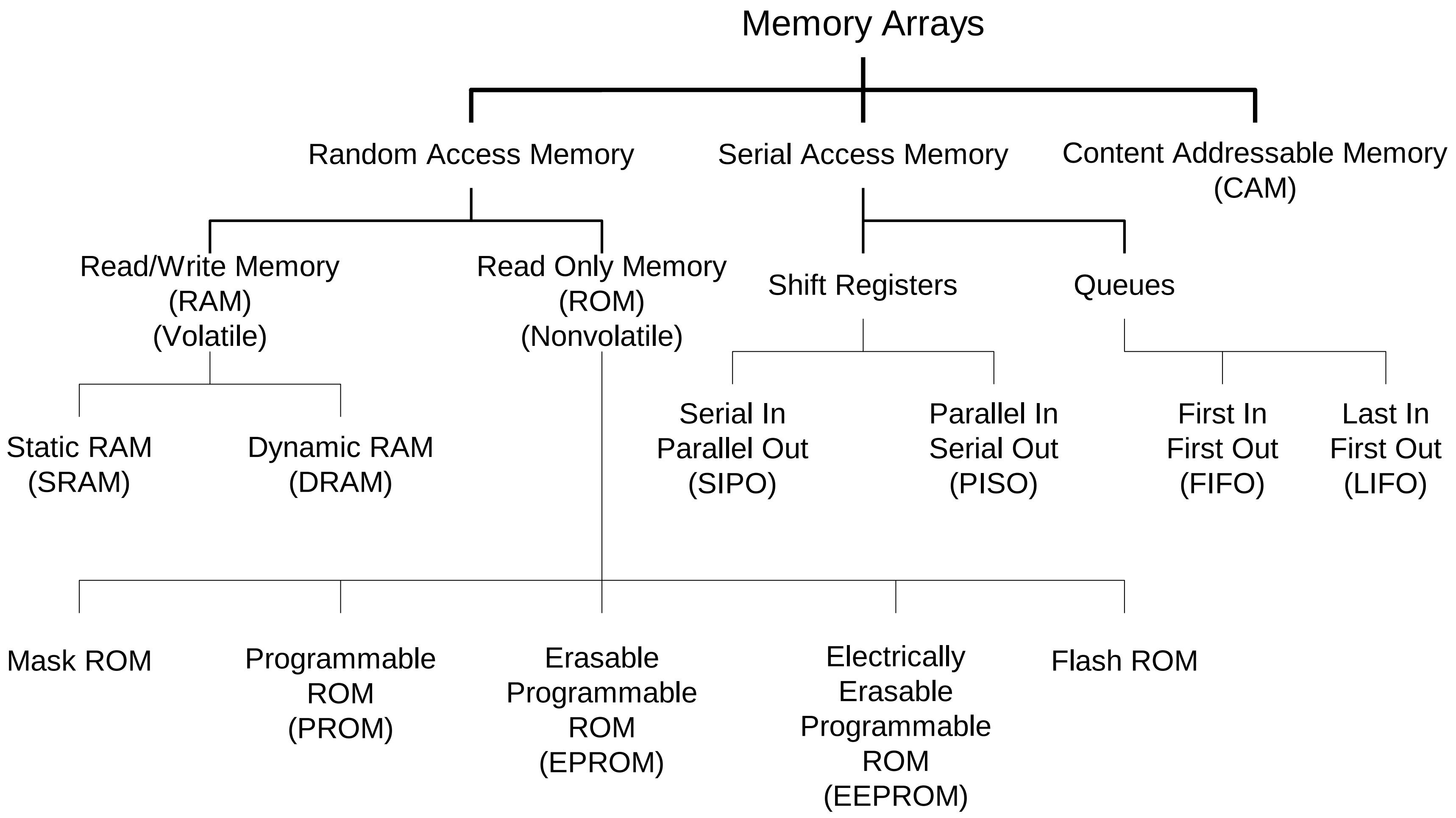
Lecture 8:
Embedded Memory

黃柏蒼 Po-Tsang (Bug) Huang
bughuang@nycu.edu.tw

International College of Semiconductor Technology
National Chiao Tung Yang Ming University

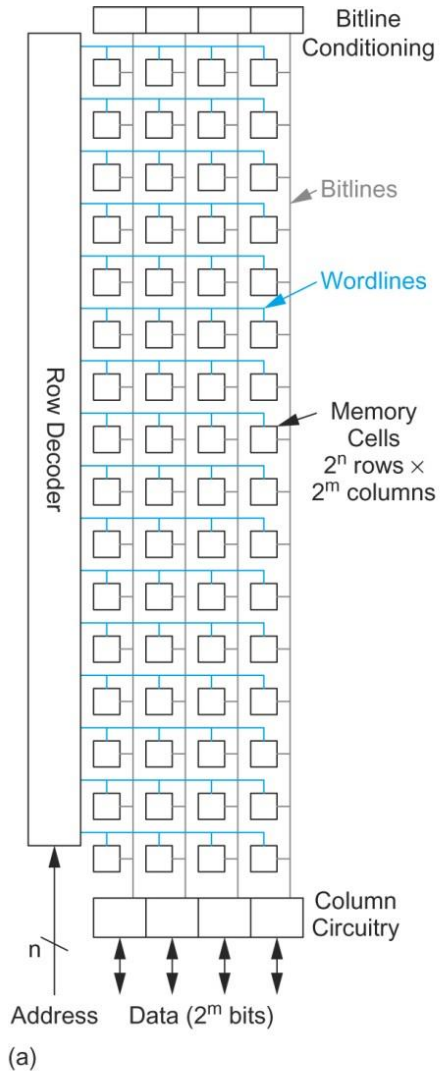
國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

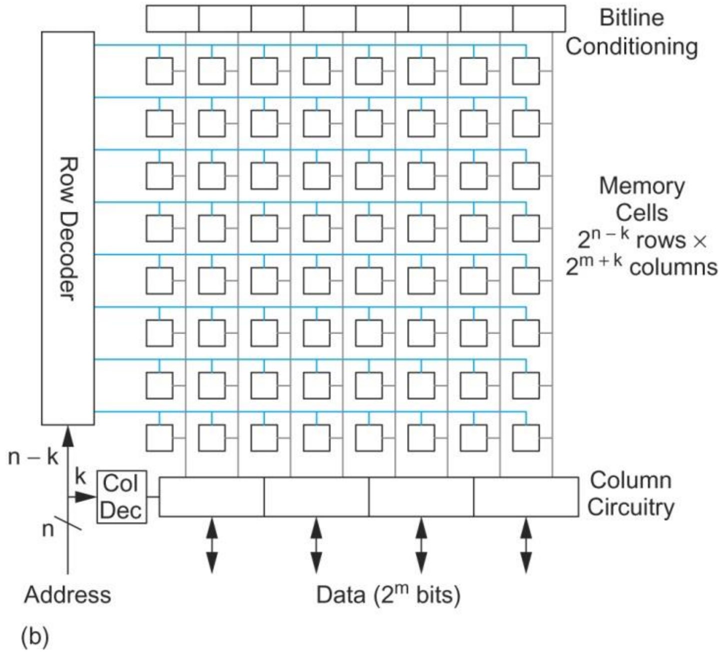
Memory Arrays


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

Memory Array

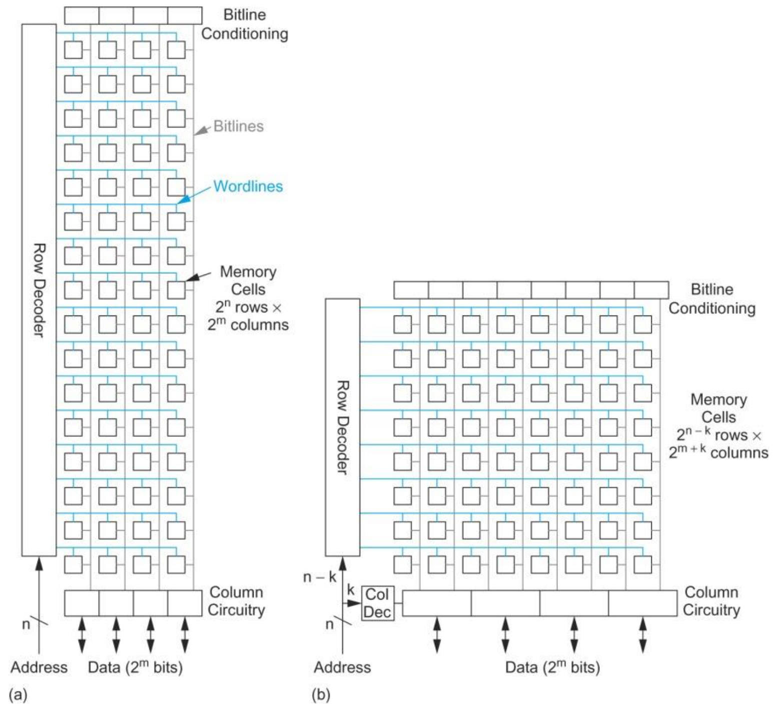
- Decode address to access the corresponding data



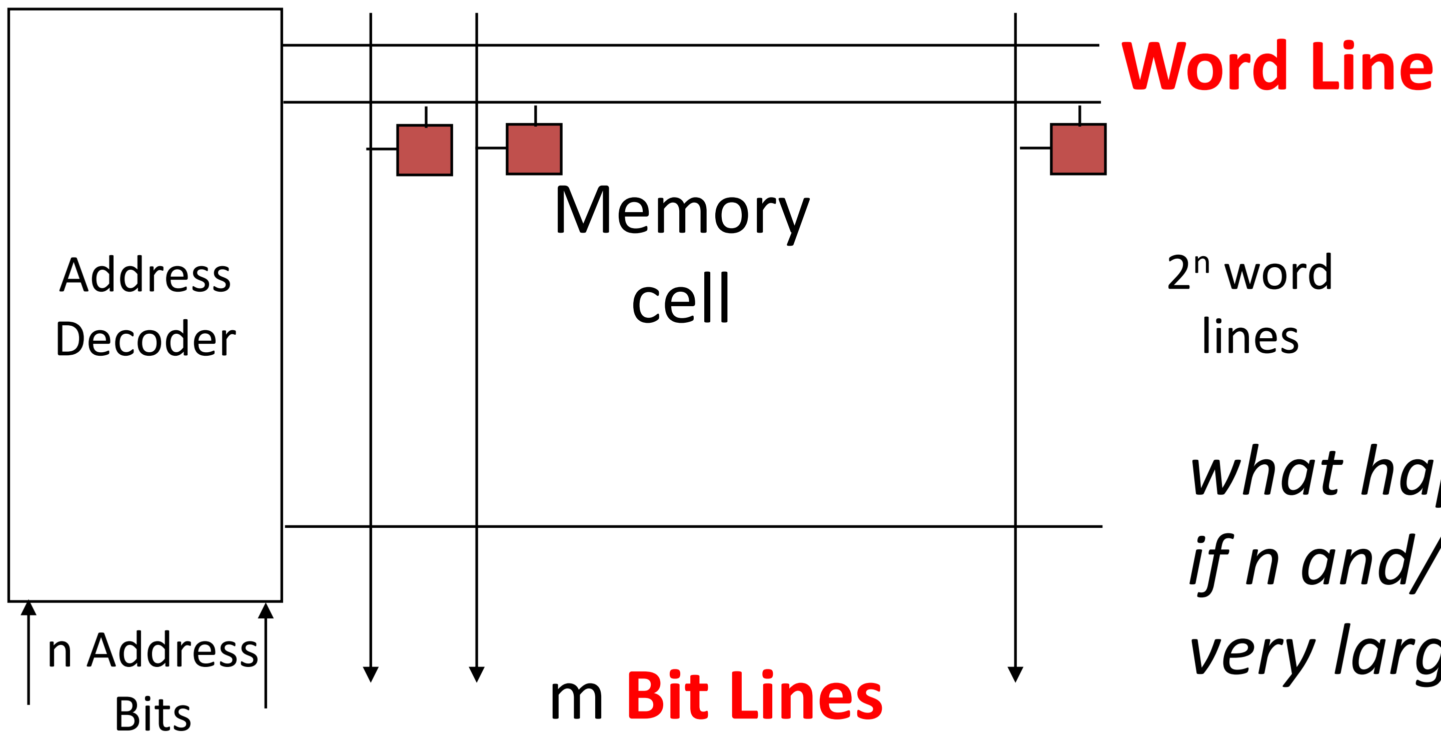


Memory Array Architecture

- 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



Basic Memory Subsystem Block Diagram

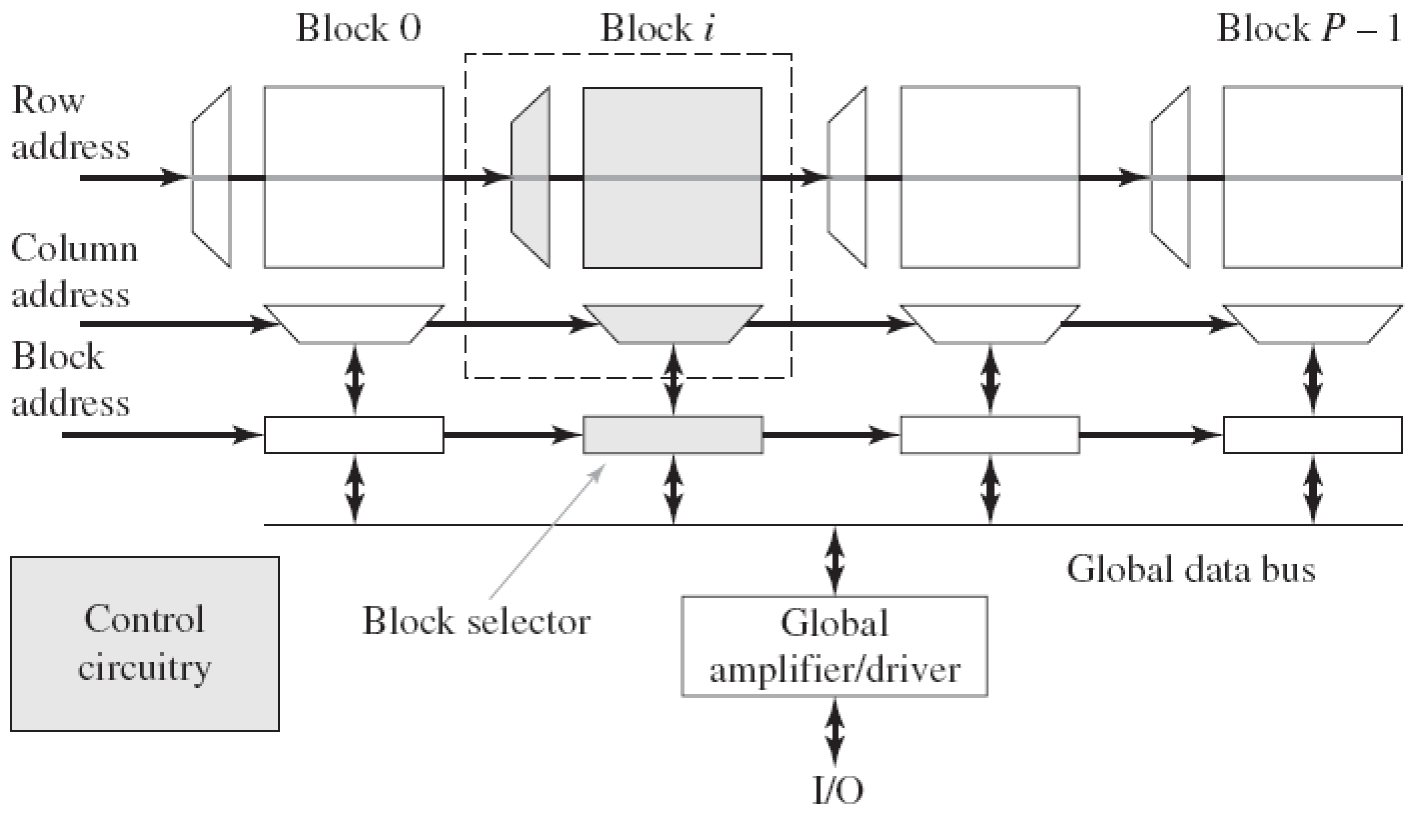


what happens if n and/or m is very large?

Construct large memory through modular design

Memory Structure

- Memory array
- Interconnection architecture (memory bus)

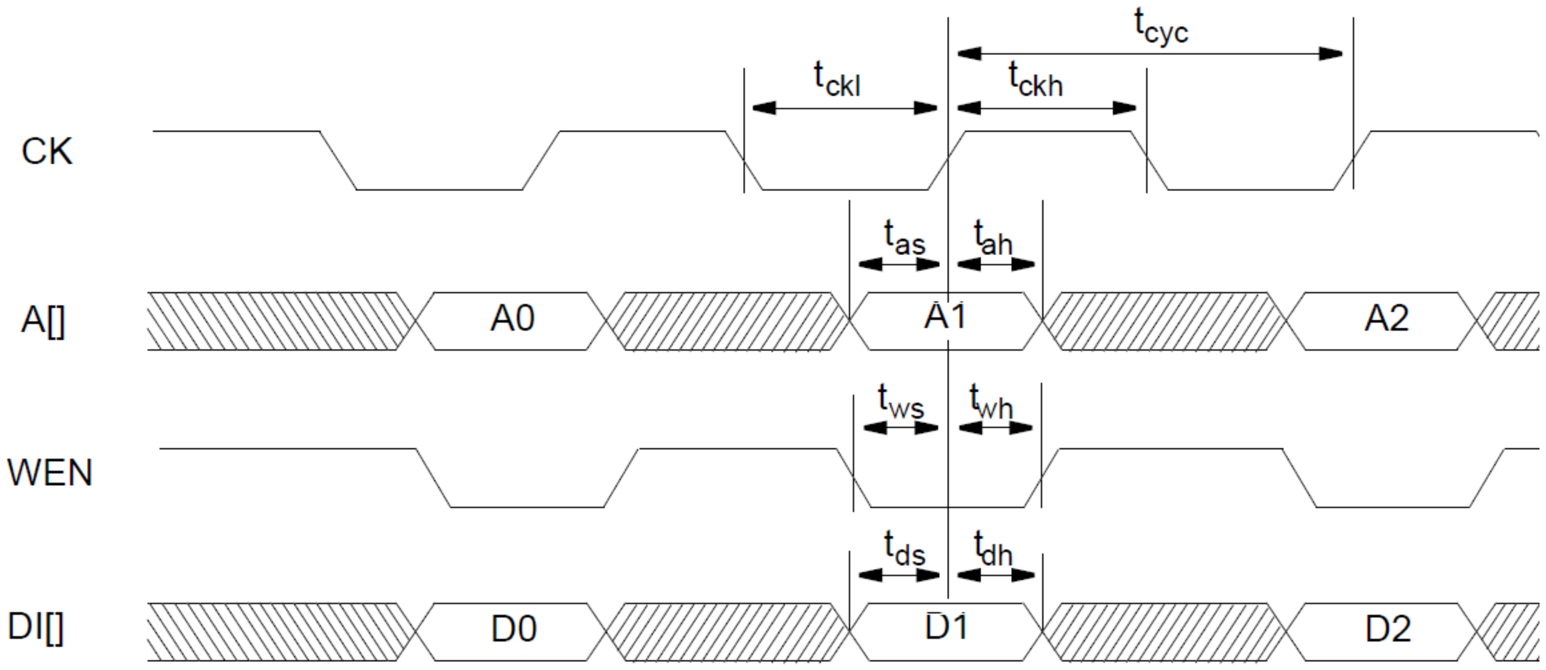


Memory Array Architecture

- Address decoder
  - 1-D address
  - 2-D address
    - Separated into row/column
- Memory cell array
  - Static: SRAM
  - embedded DRAM
  - embedded Nonvolatile memory
    - eflash, ReRAM, MRAM
- Read out
  - Sense amplifier

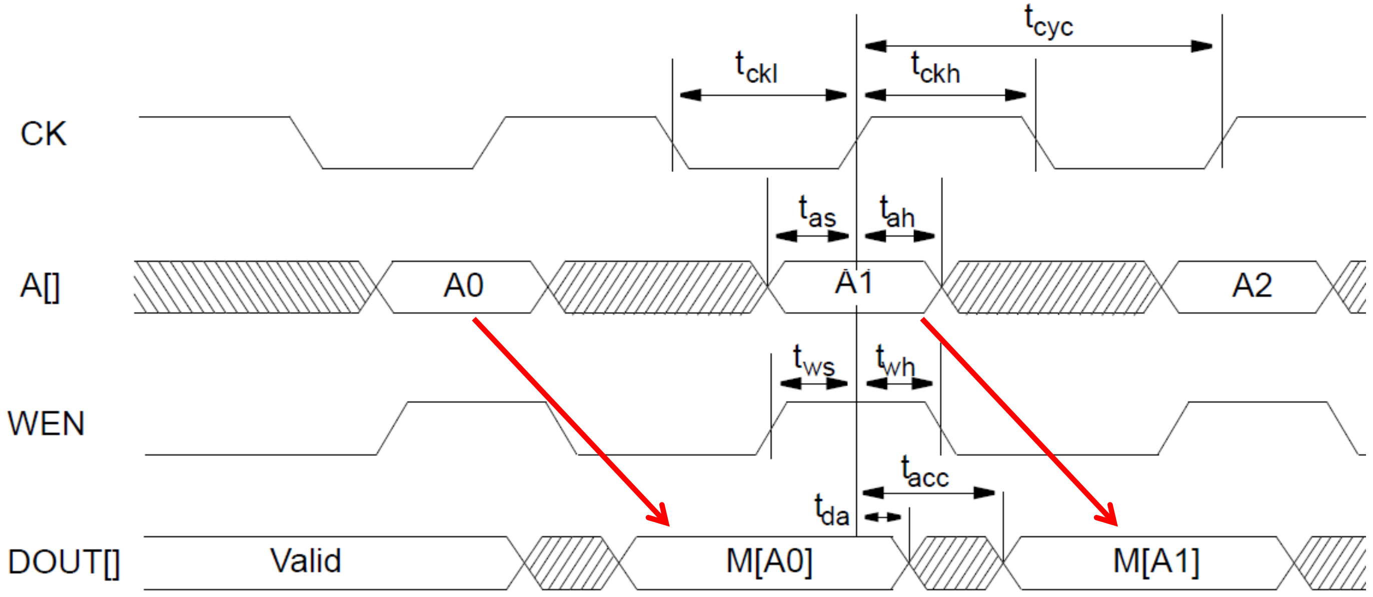
Operation of Memory

- Write
  - ◆ Send address, read/write control, *data to write*



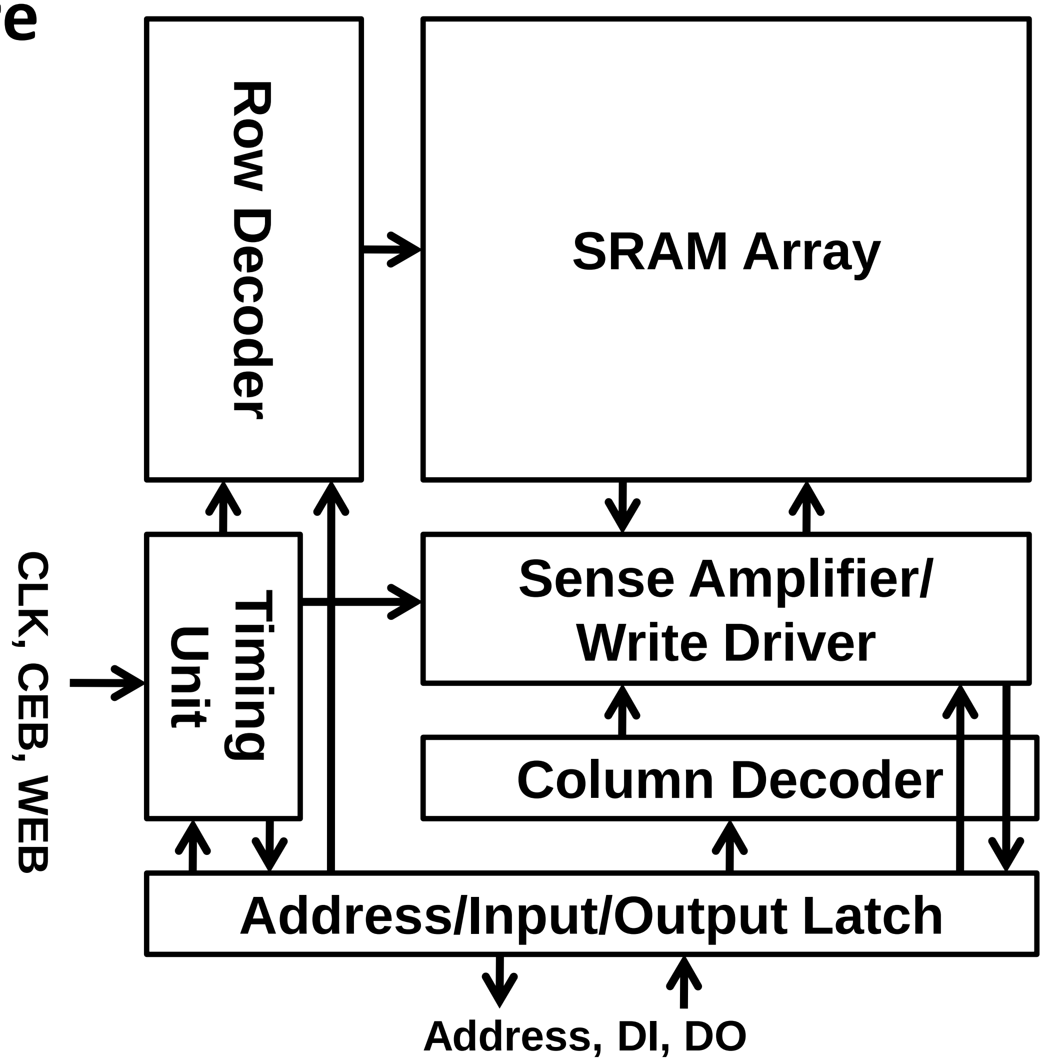
Operation of Memory

- Read
  - Send address, read/write control
  - Output data



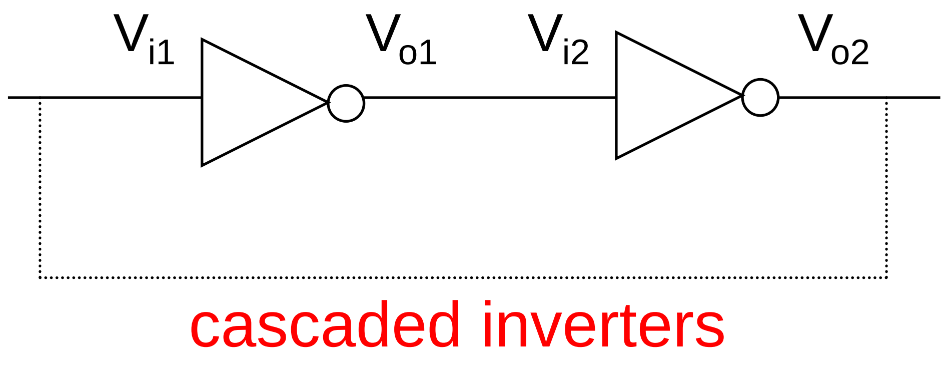
Basic Blocks of Memory

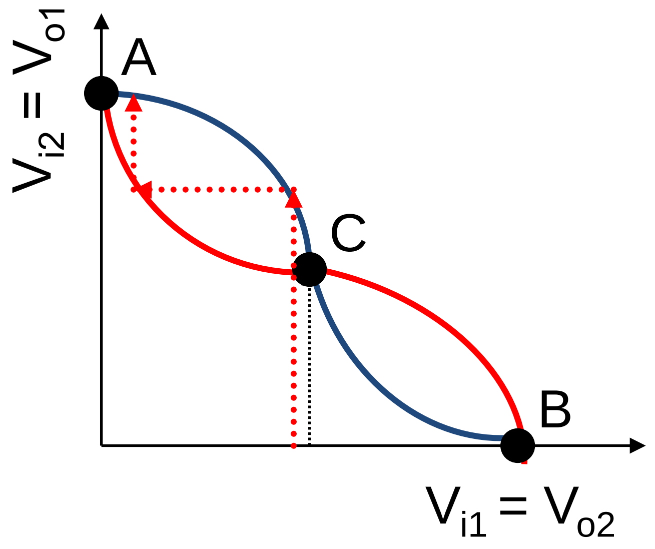
- **SRAM basic structure**
  - SRAM Array
  - Sense amplifier
  - Write Driver
  - Row decoder
  - Column decoder
  - Timing unit
  - Address Latch
  - Input latch
  - Output latch



Regenerative Property of Bistable Circuits

- The cross-coupling of two inverters results in a bistable circuit (a circuit with two stable states)
- If the gain in the transient region is larger than 1, only A and B are stable operation points. C is a metastable operation point.



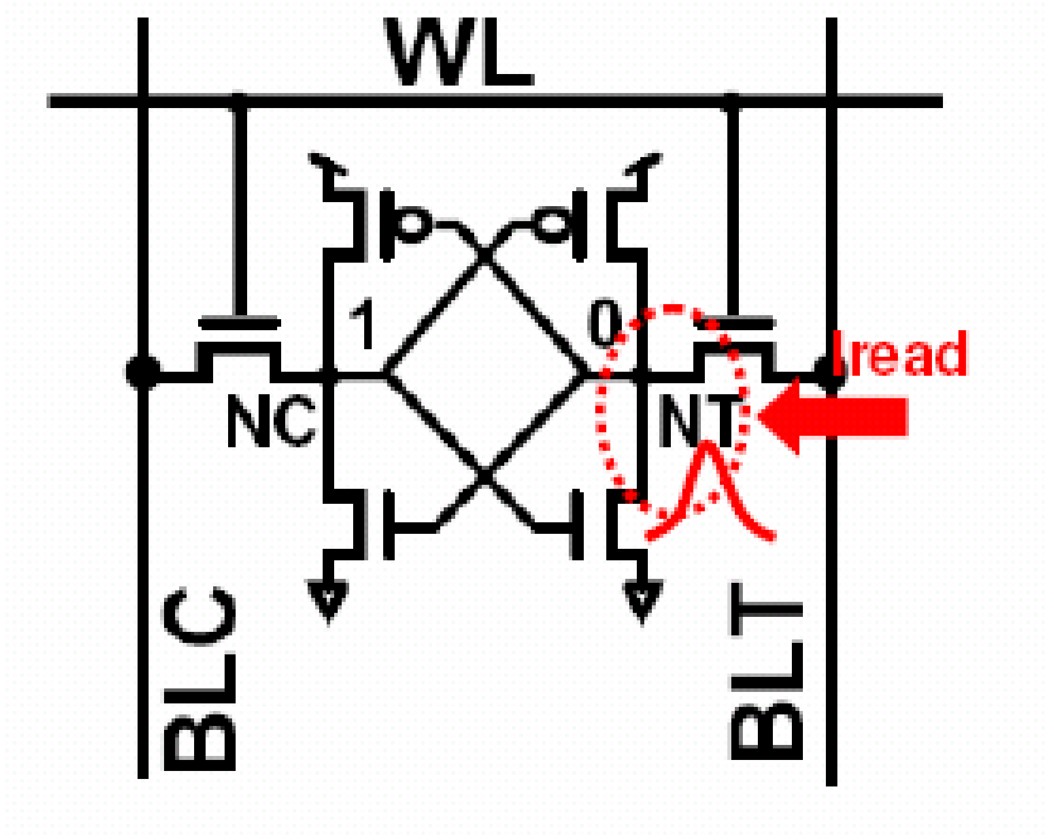


How to write(flip) a Bistable Memory

- Have to be able to change the stored value by making A (or B) temporarily unstable by increasing the loop gain to a value larger than 1
  - done by applying a trigger pulse at V_{i1} or V_{i2}
  - the width of the trigger pulse need be only a little larger than the total propagation delay around the loop circuit (twice the delay of an inverter)
- Two approaches for flipping data
  - cutting the feedback loop (latch/flip-flop)
  - overpowering the feedback loop (as used in SRAMs)

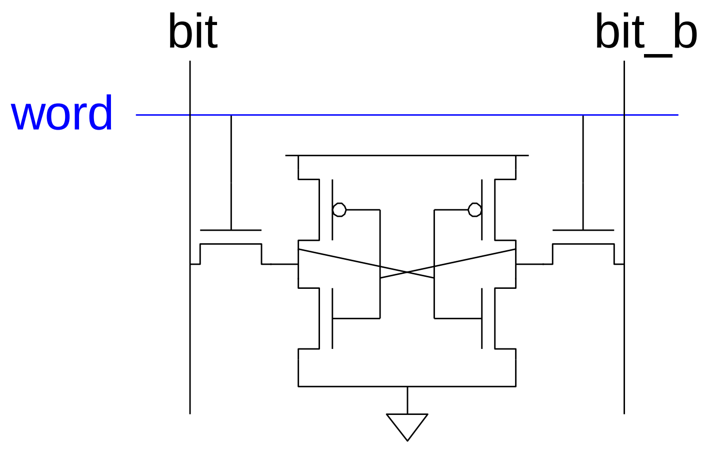
6T SRAM Cell

- Cross-coupled inverter to latch/store data
- Bi-directional access transistor

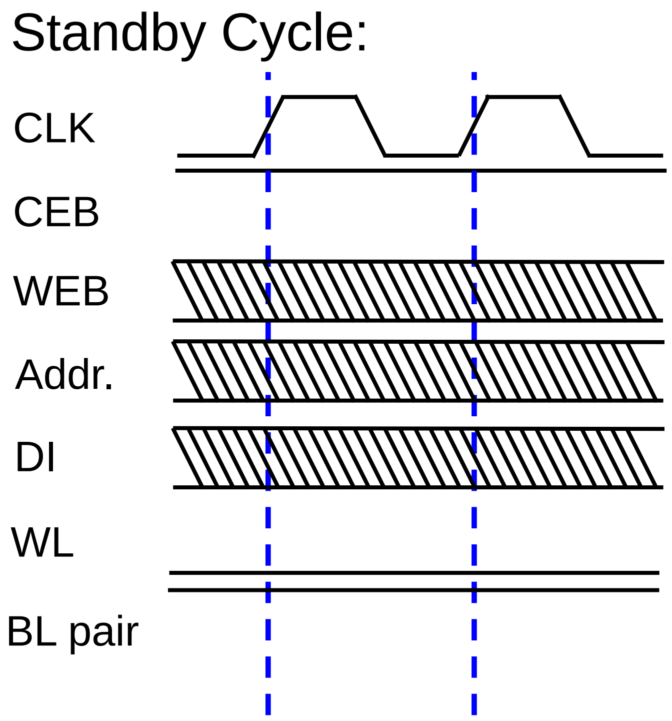


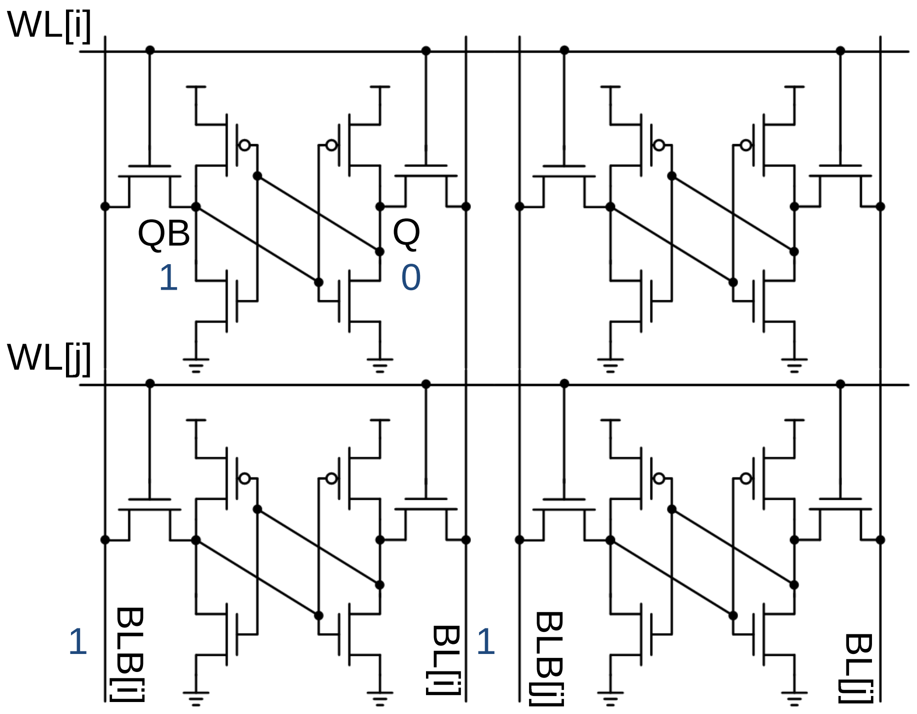
6T SRAM Cell

- 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

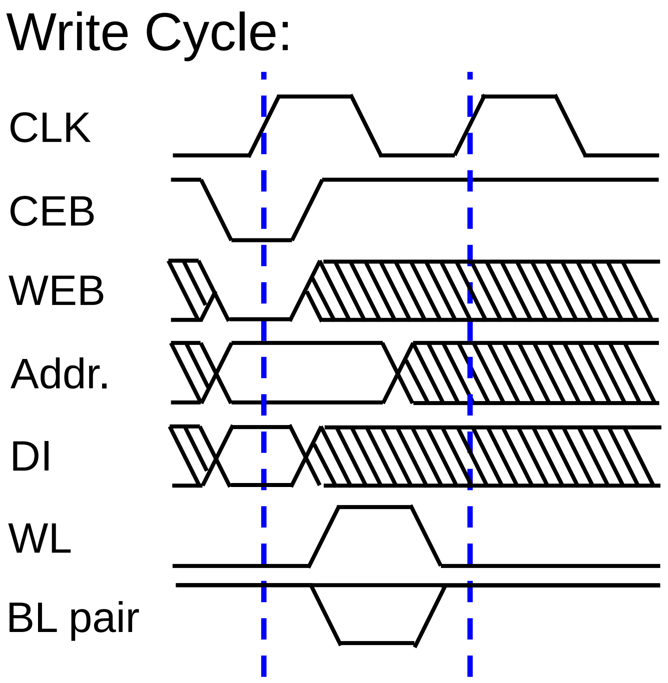


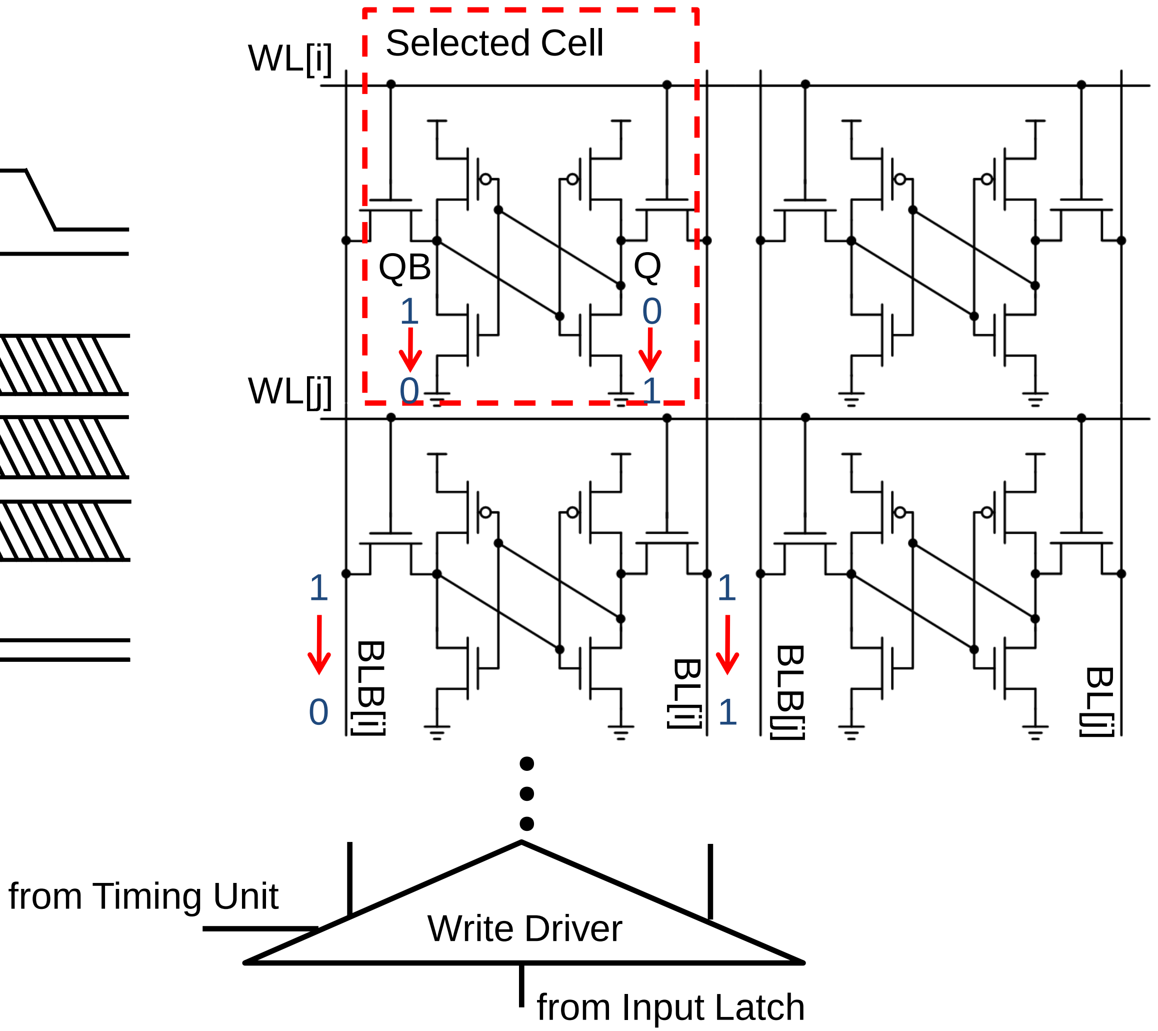
Standby Mode





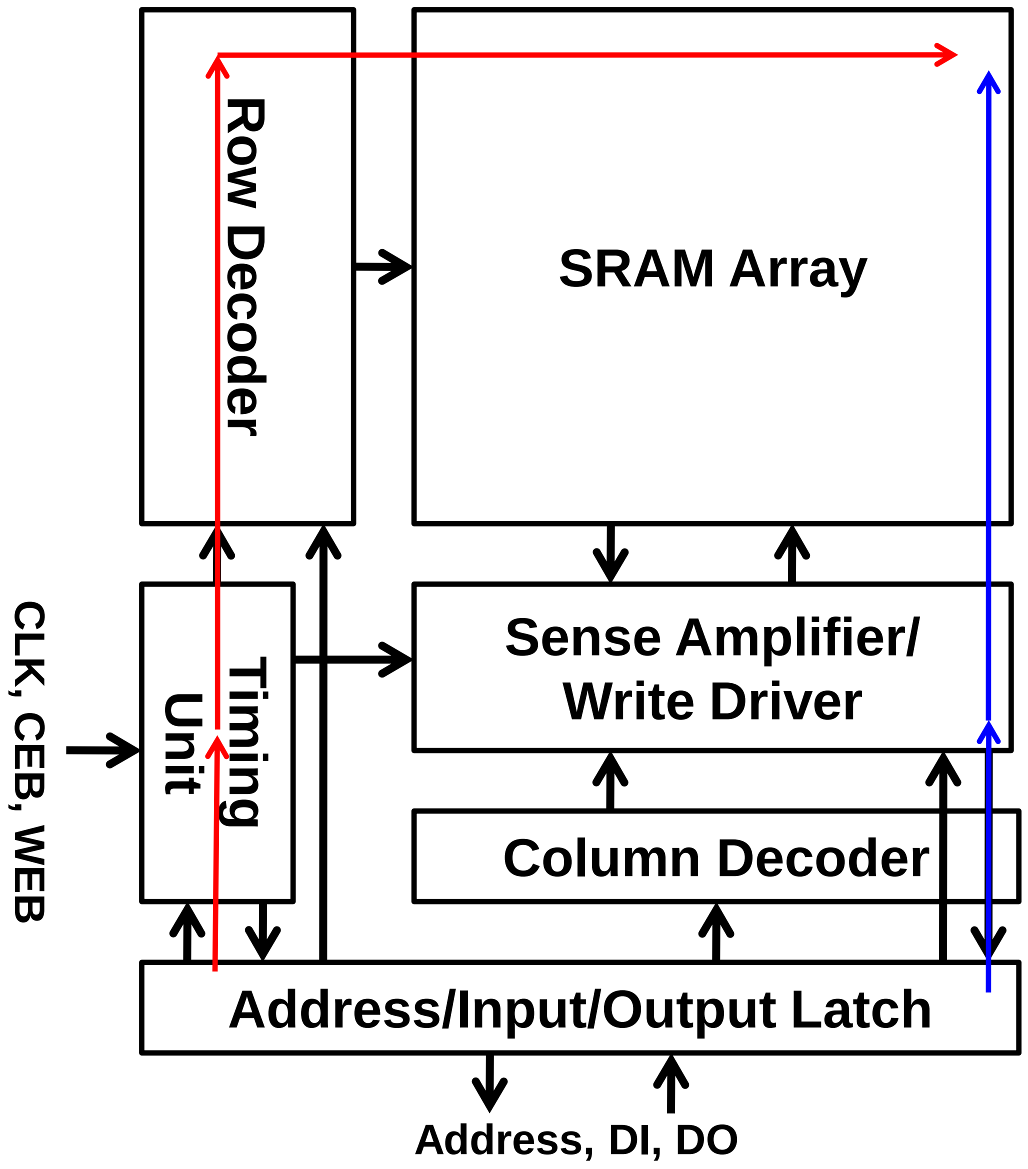
Write Operation in Local Circuits





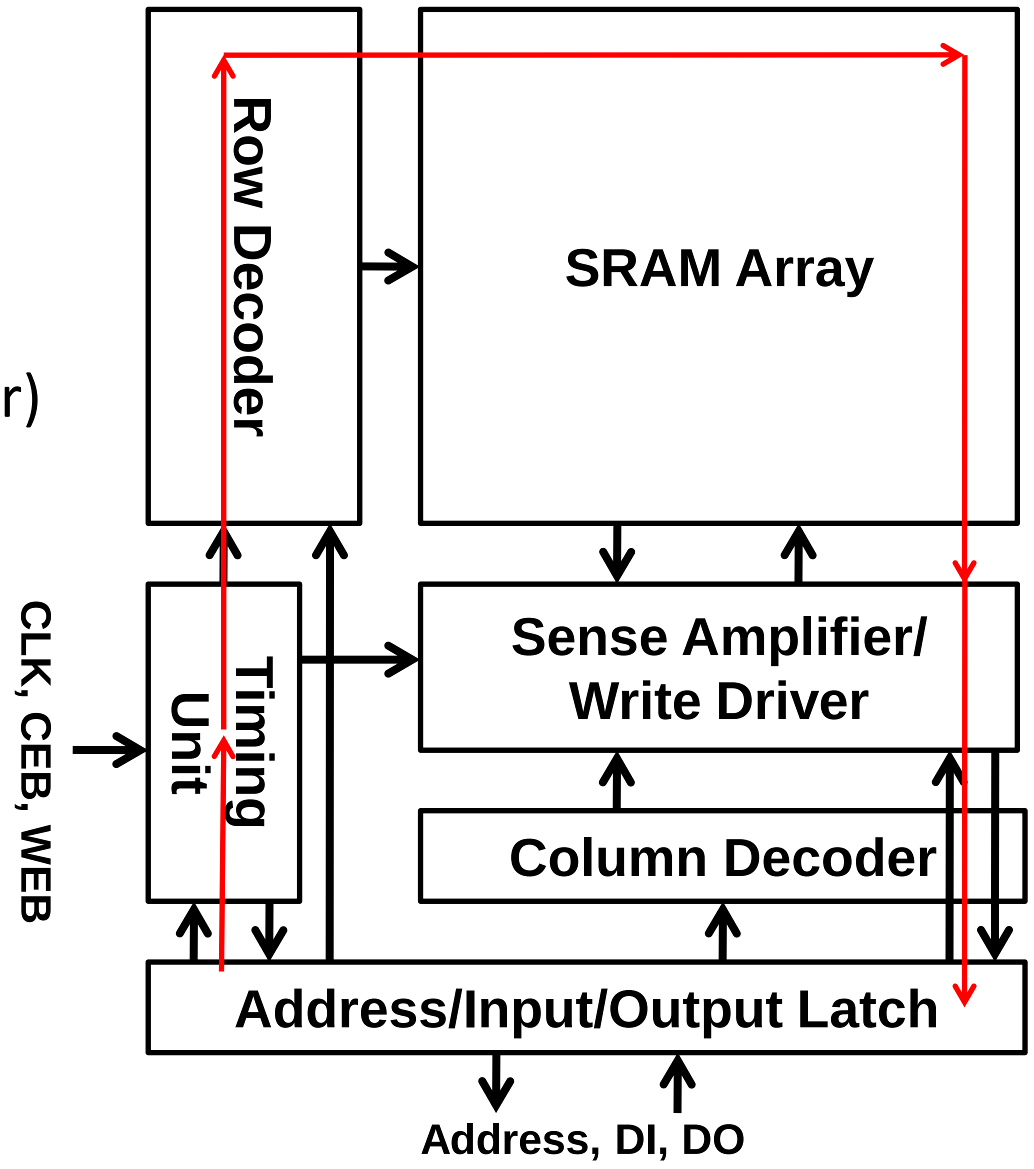
Write Operation in Global Circuits

- Data input path
  - Input Latch
  - Write Driver
  - SRAM Array (Selected BL pairs)
- Control path
  - Address latch
  - Timing Unit
  - Row Decoder
  - SRAM Array (Selected WL)



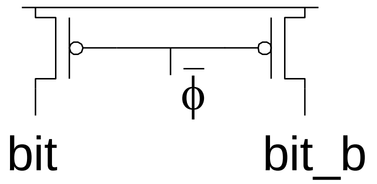
Read Operation in Global Circuits

- **Read critical path**
  - Address latch
  - Timing Unit
  - Row Decoder (WL driver)
  - SRAM Array
    - Selected WL
    - Selected BL pairs
  - Sense Amplifier
  - Output latch

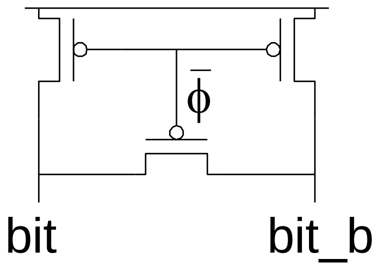


Bitline Conditioning (Precharge)

- Two purposes
- Precharge bitlines high before reads



- Equalize bitlines to minimize voltage difference when using sense amplifiers

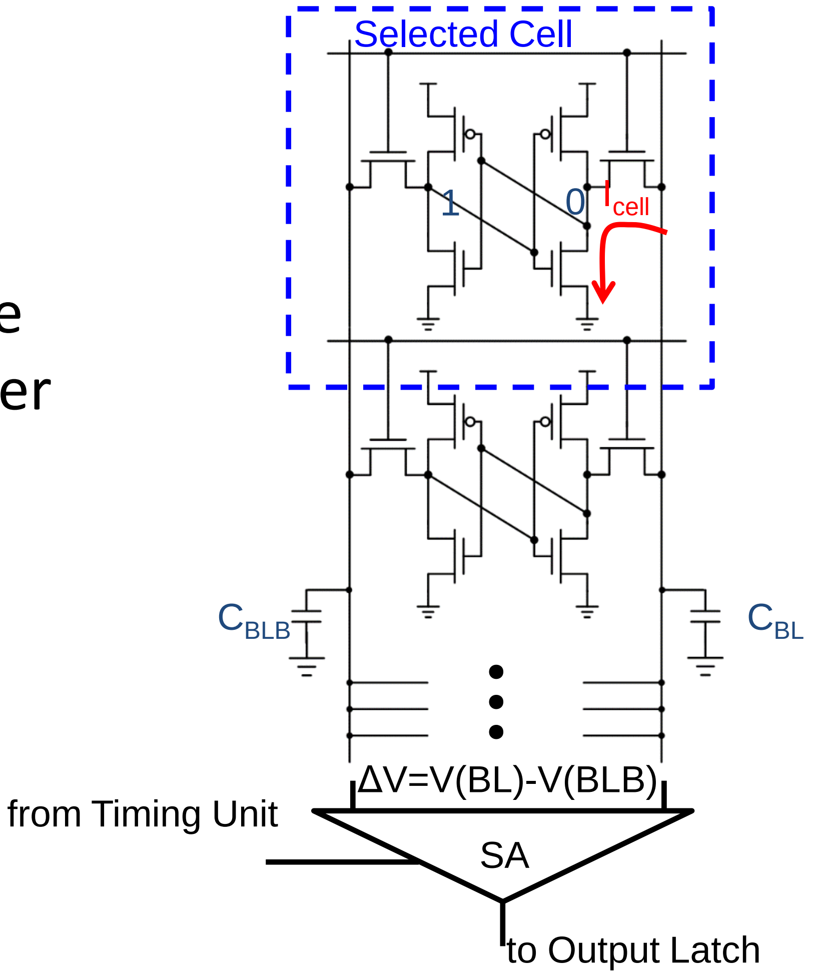


SRAM Read Performance

- In general, the latency of SRAM Read operation is longer than Write operation

$$T_{BL} = \frac{C_{BL}\Delta V}{I_{Cell}}$$

- ΔV: The difference voltage between BL and BLB, larger than the offset voltage (V_{offset}) of SA
- I_{cell}: Cell Read current



SRAM Cell

- What we care
  - Read/write operation
  - Stability of hold/read/write
  - Cell size

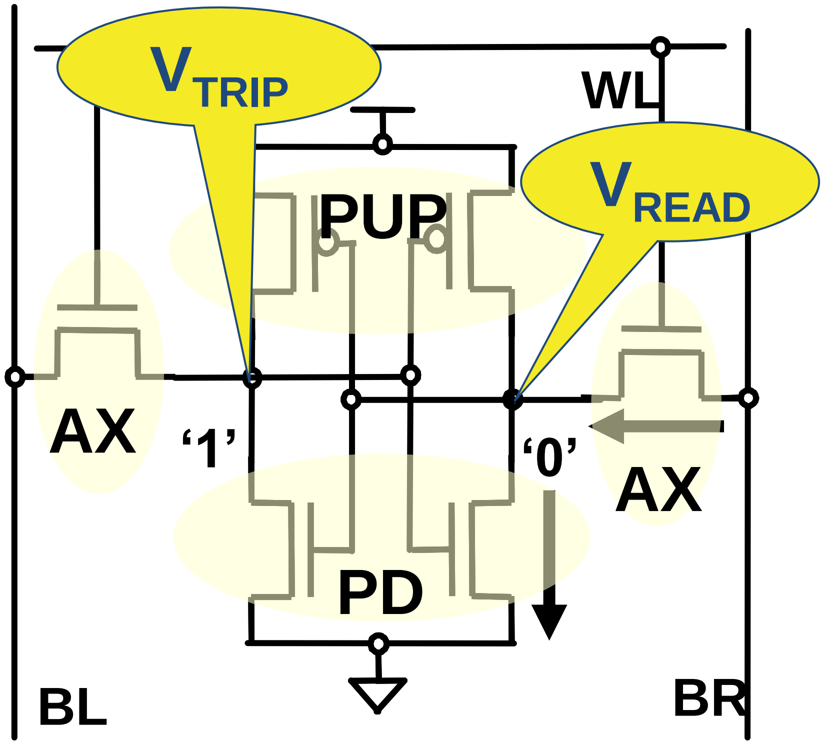
Conflicting Read/Write Requirements

- To facilitate Read and minimize Read-disturb (V_{READ})
  - Strong PD NMOS and weak AX NMOS (small $_2$)
- To improve Write-ability (Write margin)
  - Strong AX NMOS and weak PUP PMOS (large $_3$)

$$\frac{I_{PUP}}{I_{PD}} \sim \beta_1 = \frac{\mu_p \boxed{(W/L)_{PUP}}}{\mu_n \boxed{(W/L)_{PD}}} \Rightarrow V_{TRIP}$$

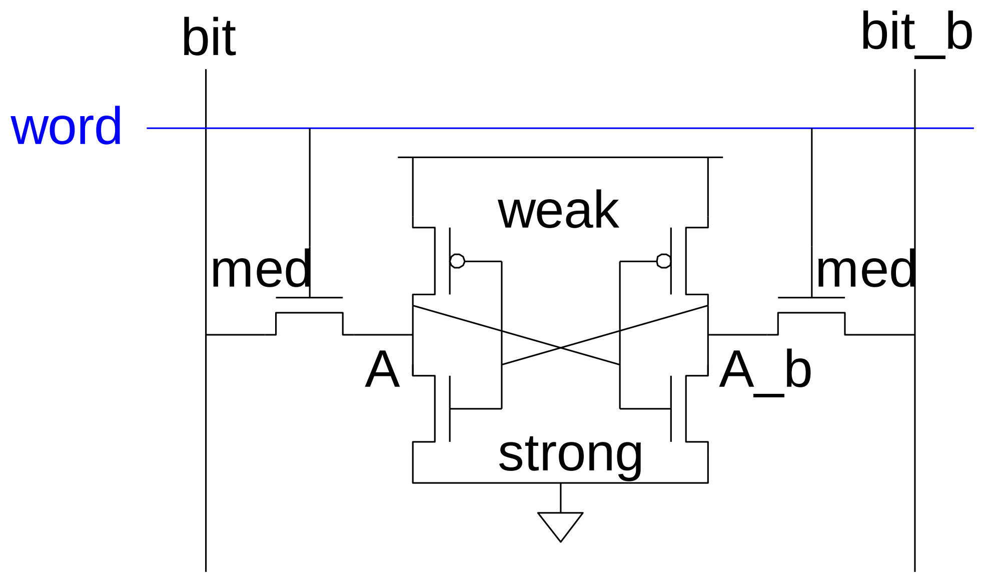
$$\frac{I_{AX}}{I_{PD}} \sim \beta_2 = \frac{\mu_n \boxed{(W/L)_{AX}}}{\mu_n \boxed{(W/L)_{PD}}} \Rightarrow V_{READ}$$

$$\frac{I_{AX}}{I_{PUP}} \sim \beta_3 = \frac{\mu_n \boxed{(W/L)_{AX}}}{\mu_p \boxed{(W/L)_{PUP}}} \Rightarrow T_{WRITE}$$



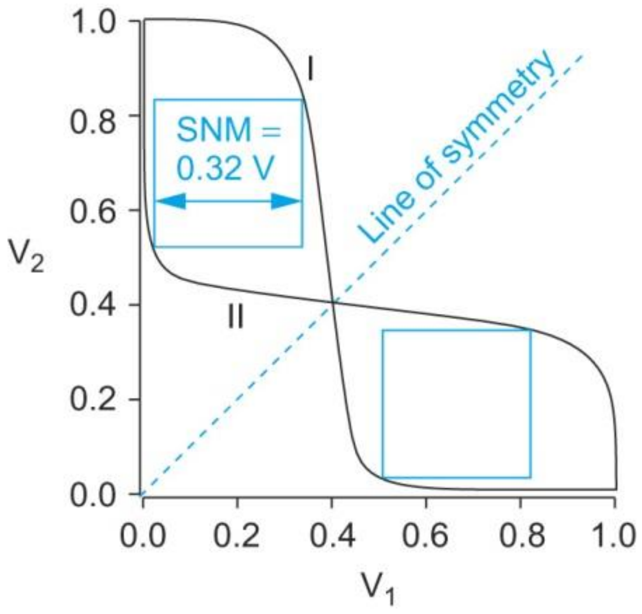
SRAM Sizing

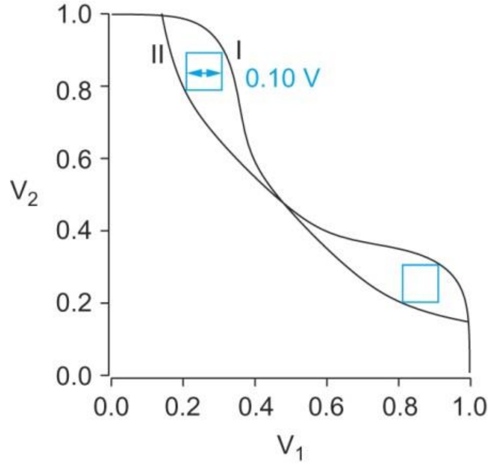
- High bitlines must not overpower inverters during reads
- But low bitlines must write new value into cell

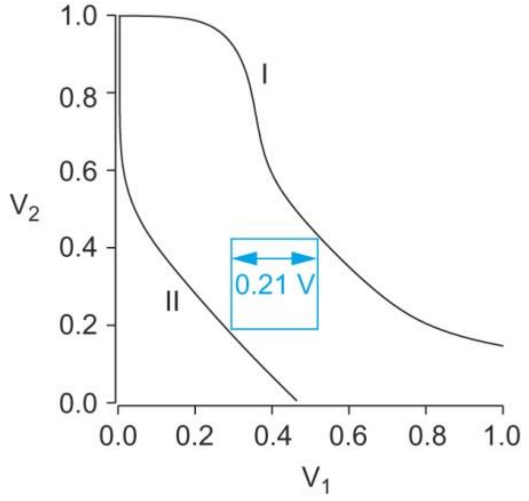


Cell Stability

- Hold margin
- Read margin
- Write margin

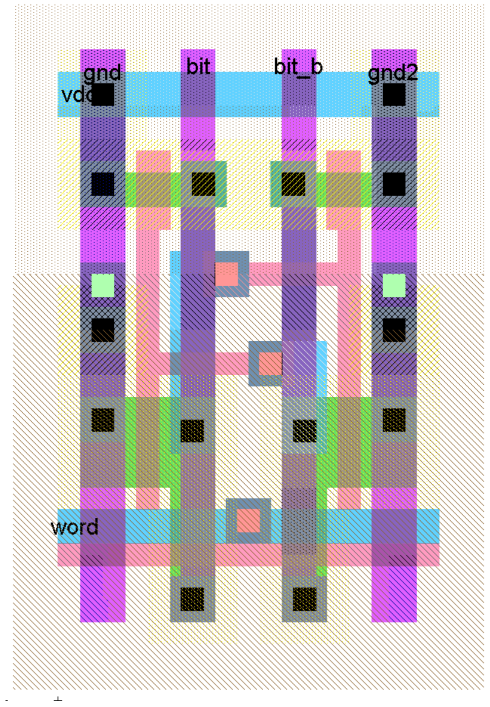


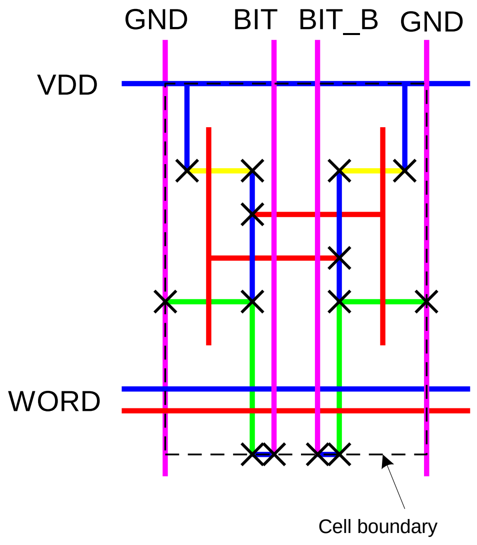


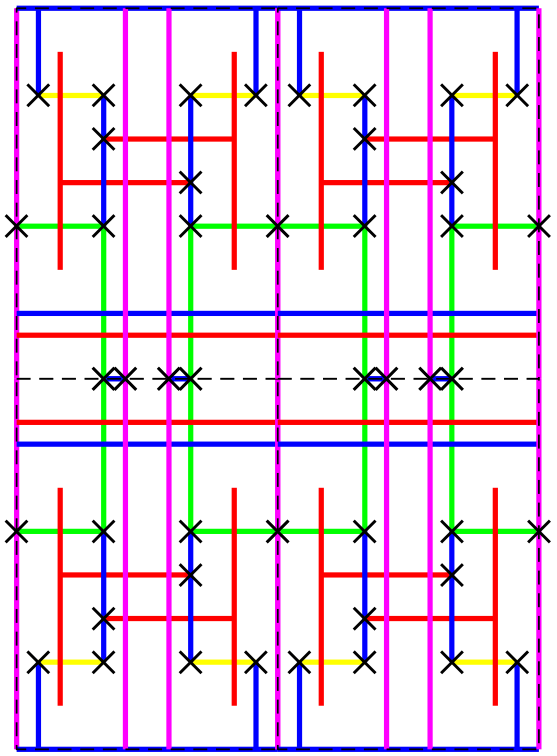


SRAM Layout

- Tile cells sharing V_{DD}, GND, bitline contacts

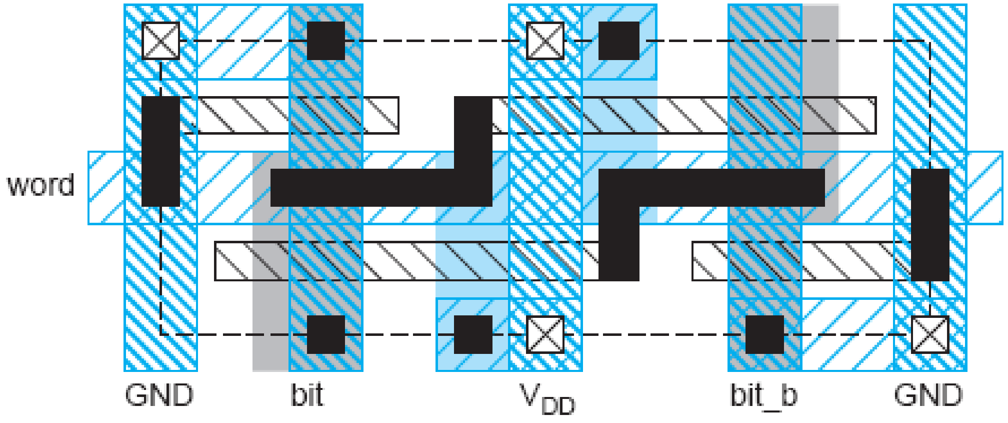






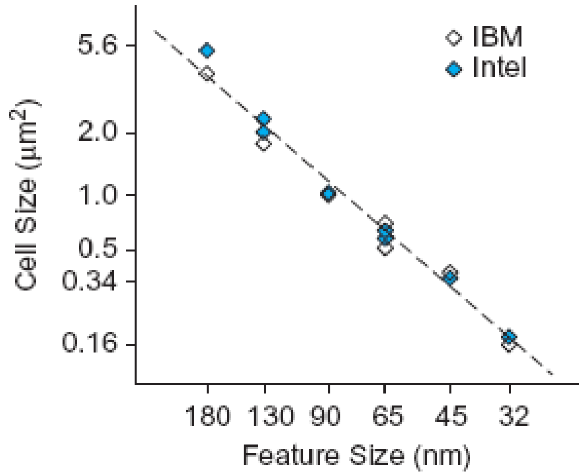
Thin Cell

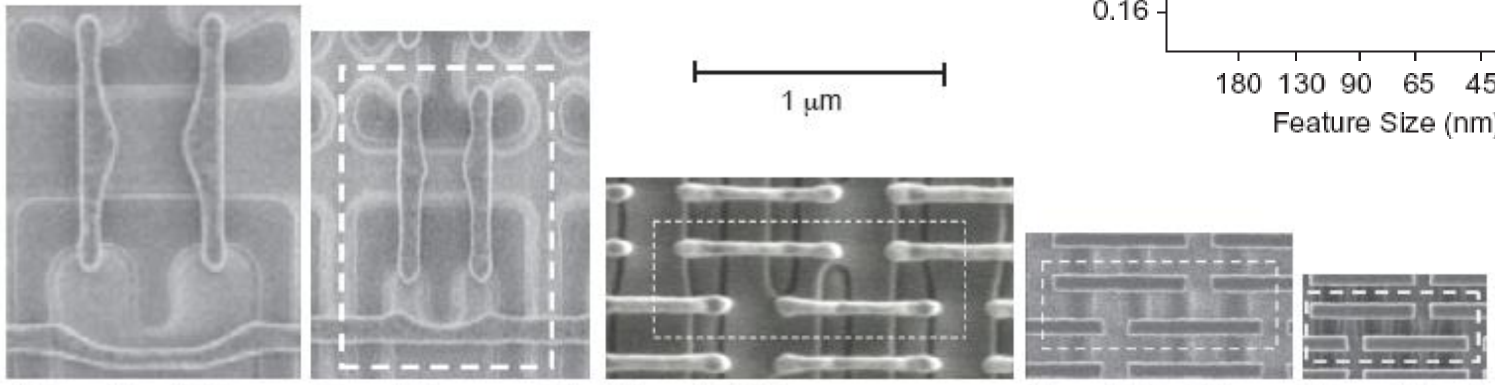
- In nanometer CMOS
  - Avoid bends in polysilicon and diffusion
  - Orient all transistors in one direction
- *Lithographically friendly* or *thin cell* layout fixes this
  - Also reduces length and capacitance of bitlines
    - Diffusion run vertically, poly run horizontally



Commercial SRAMs

- Five generations of Intel SRAM cell micrographs
  - Transition to thin cell at 65 nm
  - Steady scaling of cell area

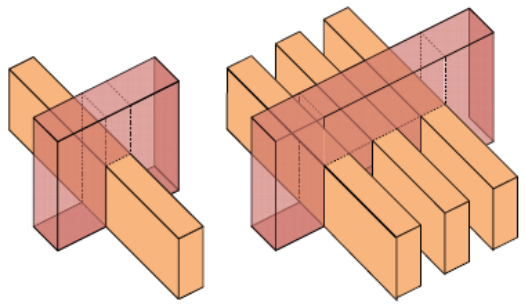


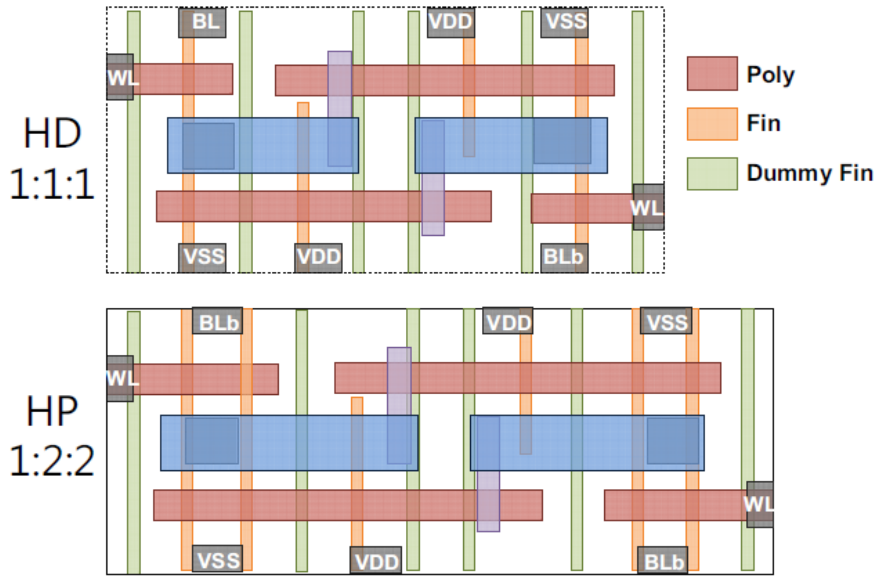


130 nm [Tyagi00] 90 nm [Thompson02] 65 nm [Bai04] 45 nm [Mistry07] 32 nm [Natarajan08]

FinFET SRAM

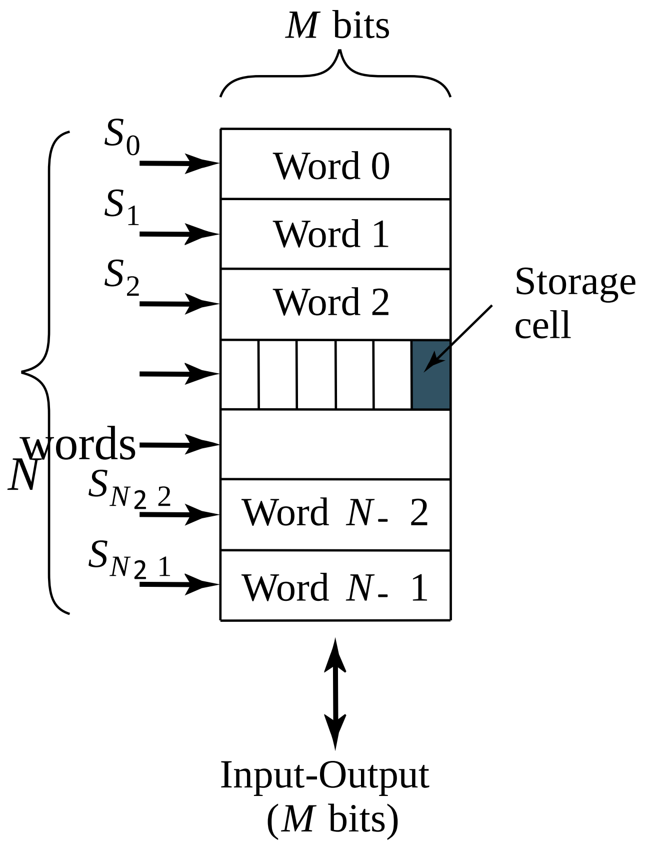
- In FinFET, the width which is given by $W = (T_{FIN} + 2H_{FIN}) \times N_{FIN}$ is quantized.
- For a high-density FinFET 6T-SRAM cell and a high-performance FinFET 6T-SRAM cell, the width ratio of Pull-Up PMOS, pass-gate NMOS and Pull-down NMOS are 1:1:1 and 1:2:2, respectively

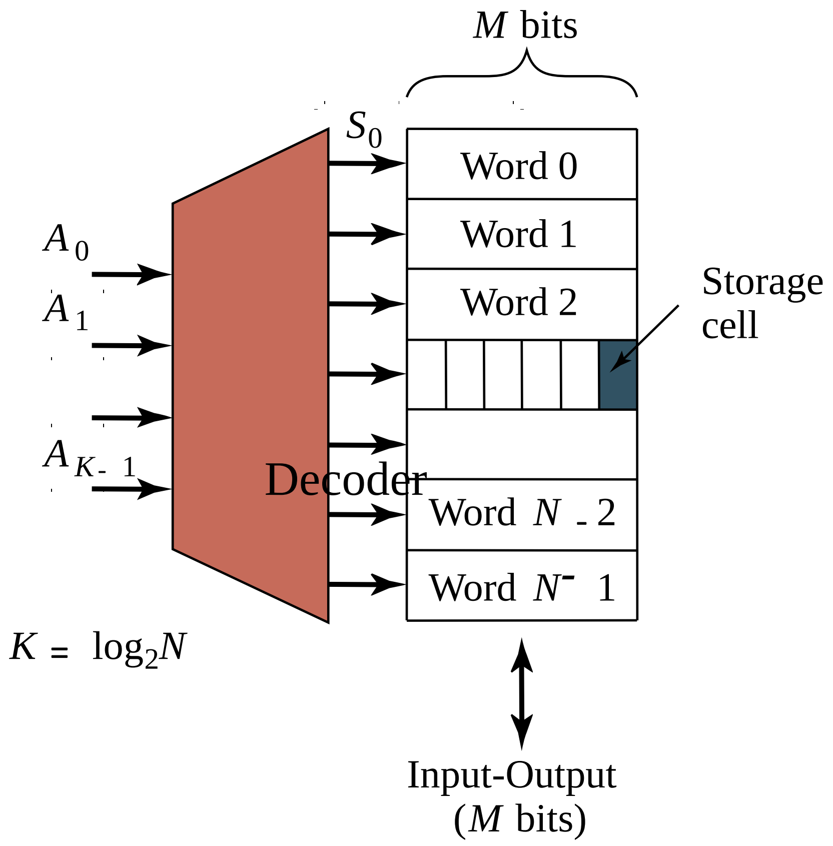




Fin-based SRAM bitcell design

Memory Architecture: Decoders



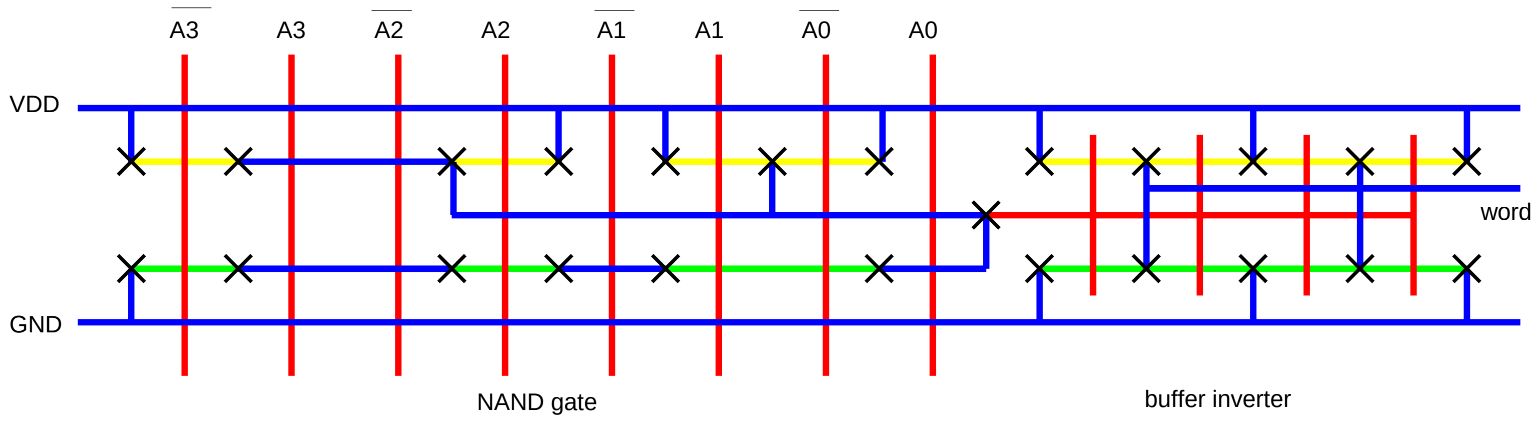


Intuitive architecture for N x M memory
Too many select signals:
N words == N select signals

Decoder reduces the number of select signals
$K = \log_2 N$

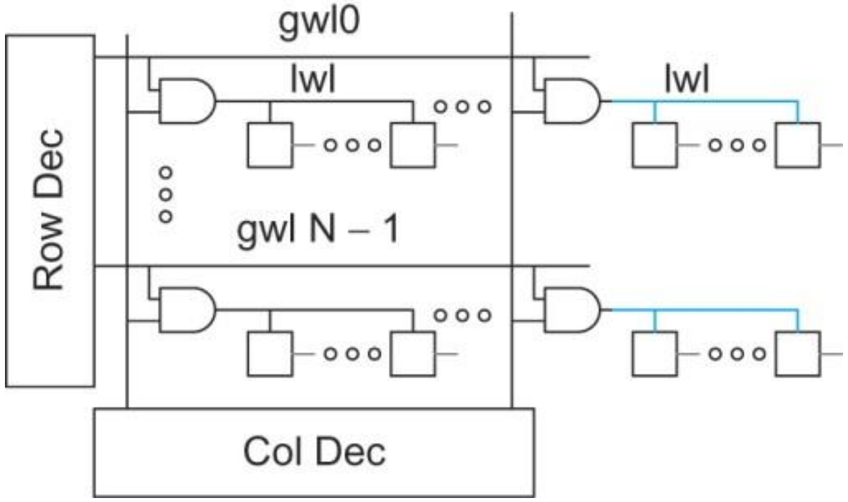
Decoder Layout

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



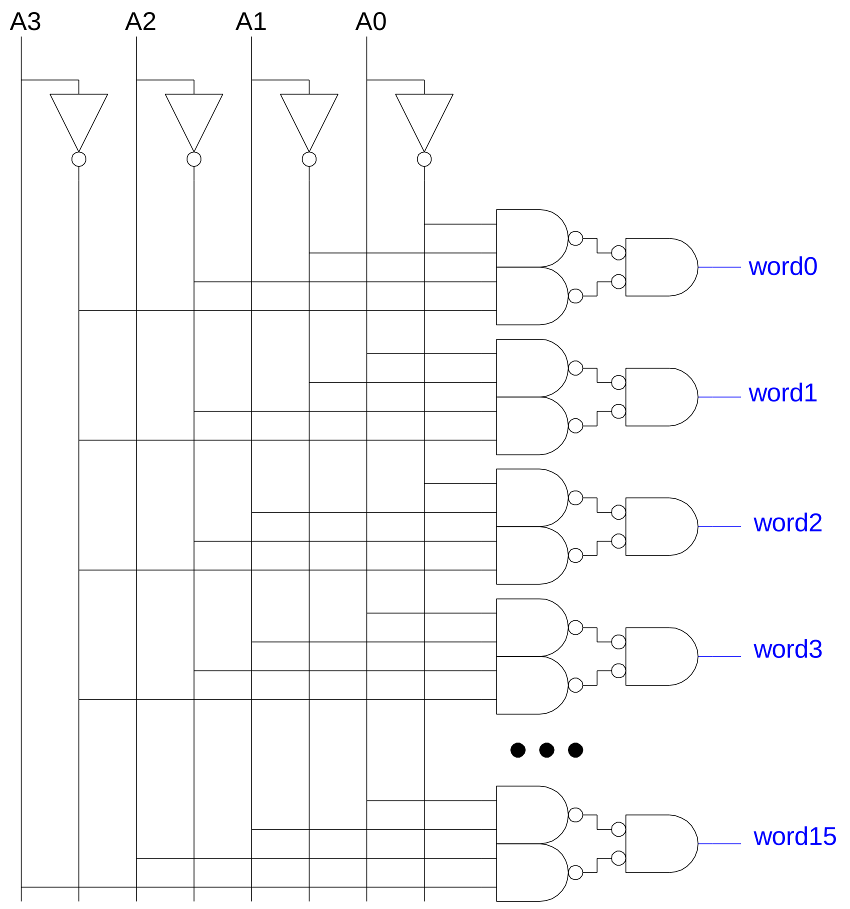
Address Decoders

- What we care
  - Layout match with SRAM cells
  - How to handle large address
    - Predecoding
    - Row/Column address
      - See large SRAM
    - Hierarchical word lines



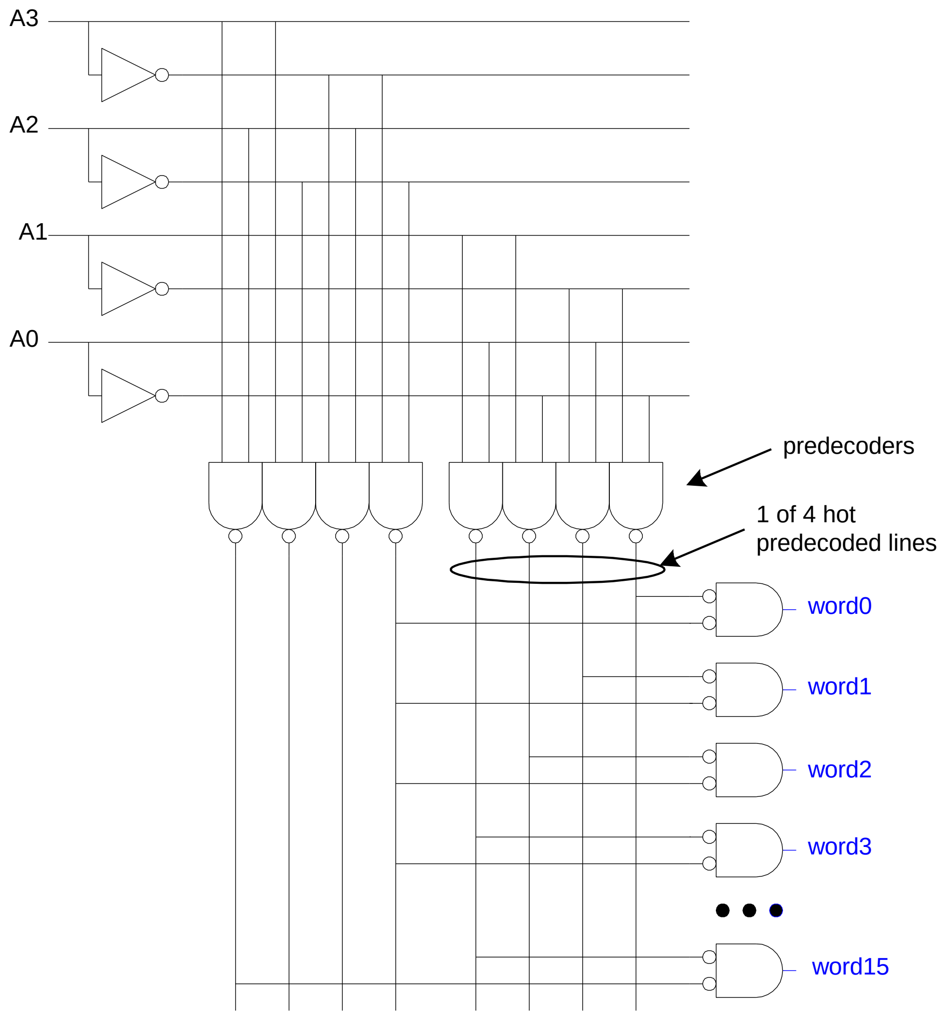
Large Decoders

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



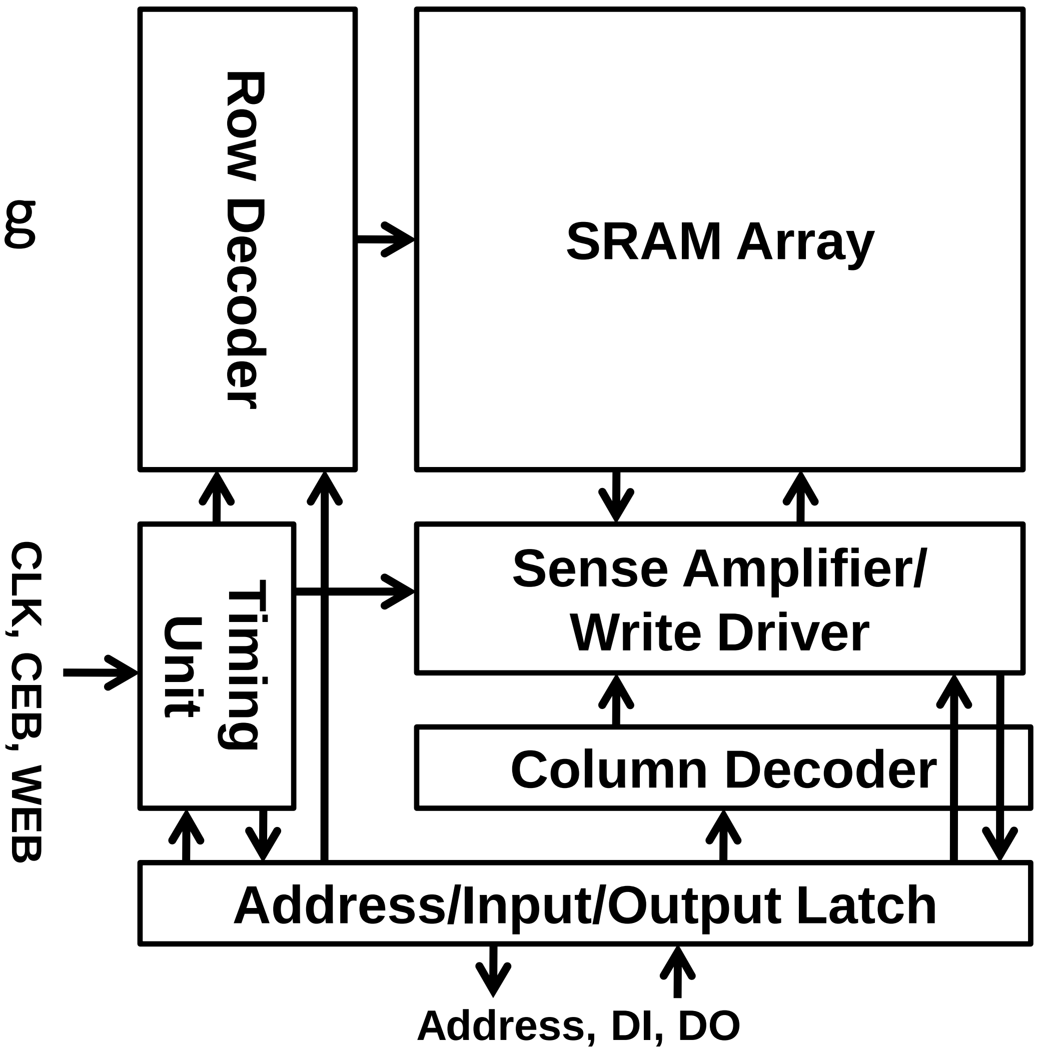
Predecoding

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



Column Circuitry

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



Sense Amplifiers

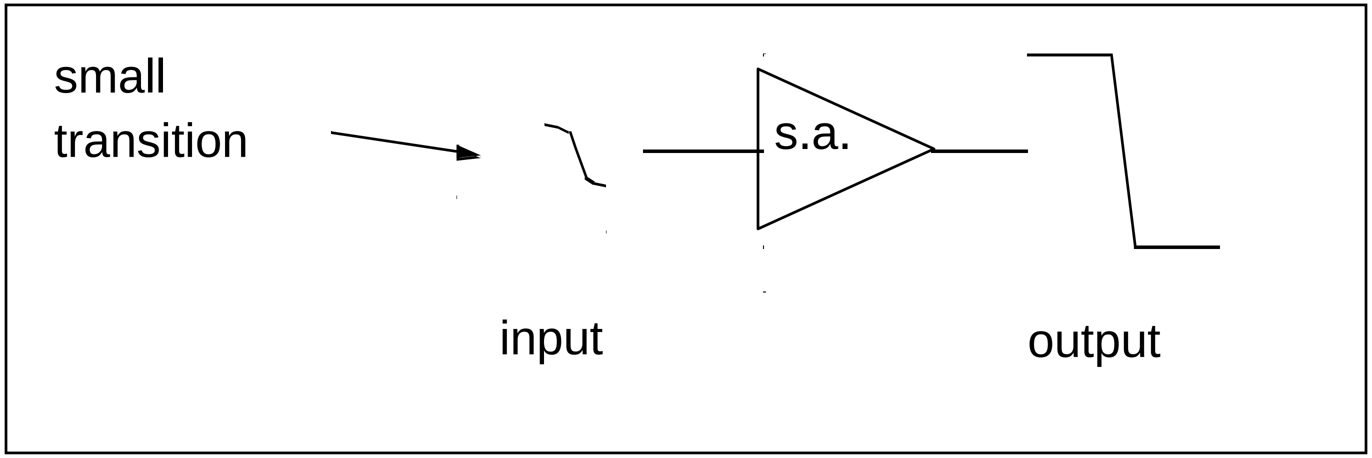
$$t_p = \frac{C \times \Delta V}{I_{av}}$$

make ΔV as small as possible

large

small

Idea: Use Sense Amplifer



Sense Amplifiers

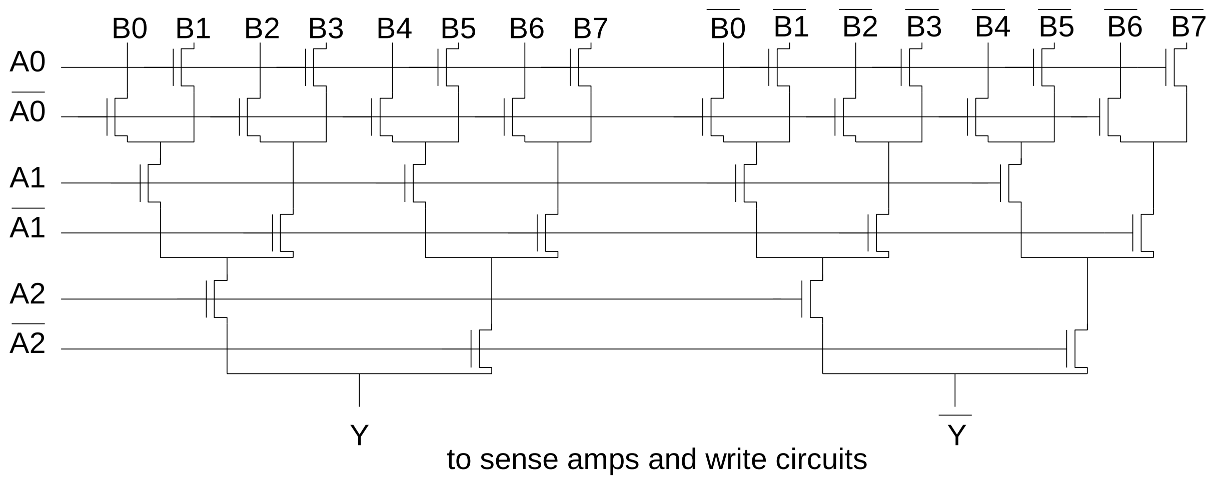
- Bitlines have many cells attached
  - Ex: 32-kbit SRAM has 128 rows x 256 cols
  - 128 cells on each bitline
- $t_{pd} \propto (C/I)\,\Delta V$
  - Even with shared diffusion contacts, 64C of diffusion capacitance (big C)
  - Discharged slowly through small transistors (small I)
- *Sense amplifiers*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

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

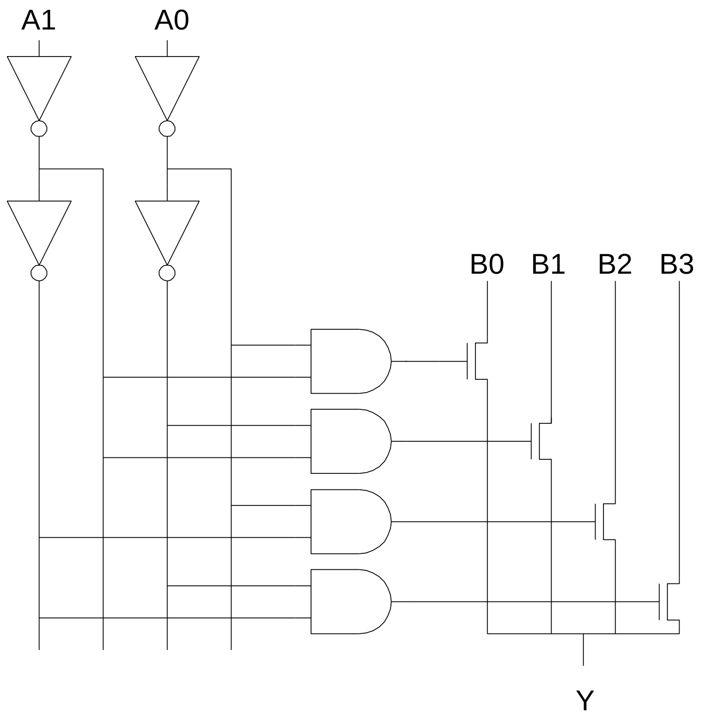
Tree Decoder Mux

- 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

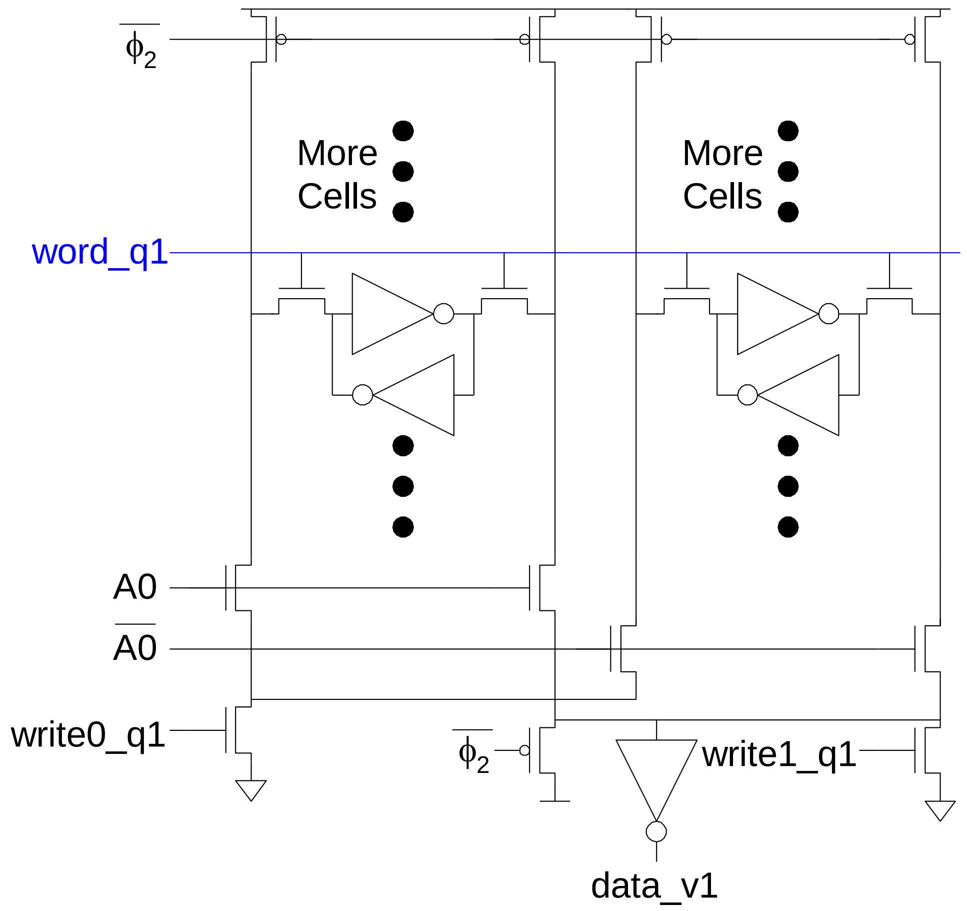


Single Pass-Gate Mux

- Or eliminate series transistors with separate decoder



Ex: 2-way Muxed SRAM

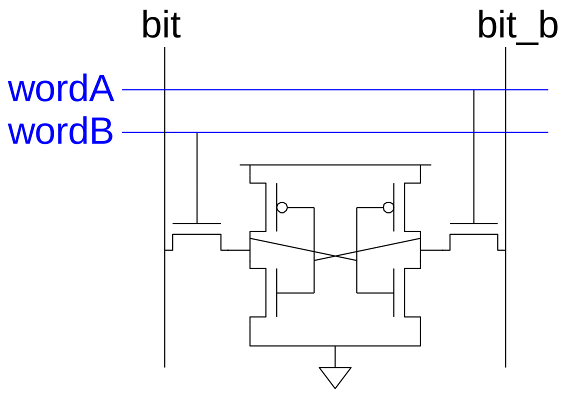


Multiple Ports

- 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

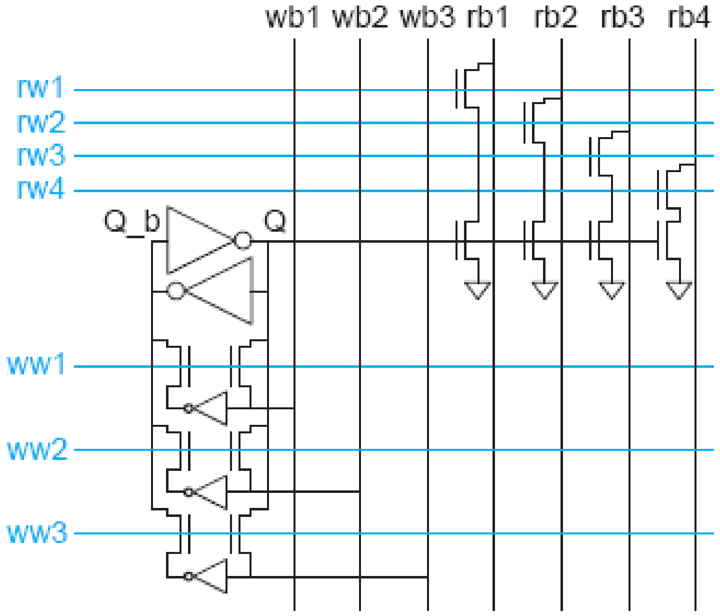
- Simple dual-ported SRAM
  - Two independent single-ended reads
  - Or one differential write



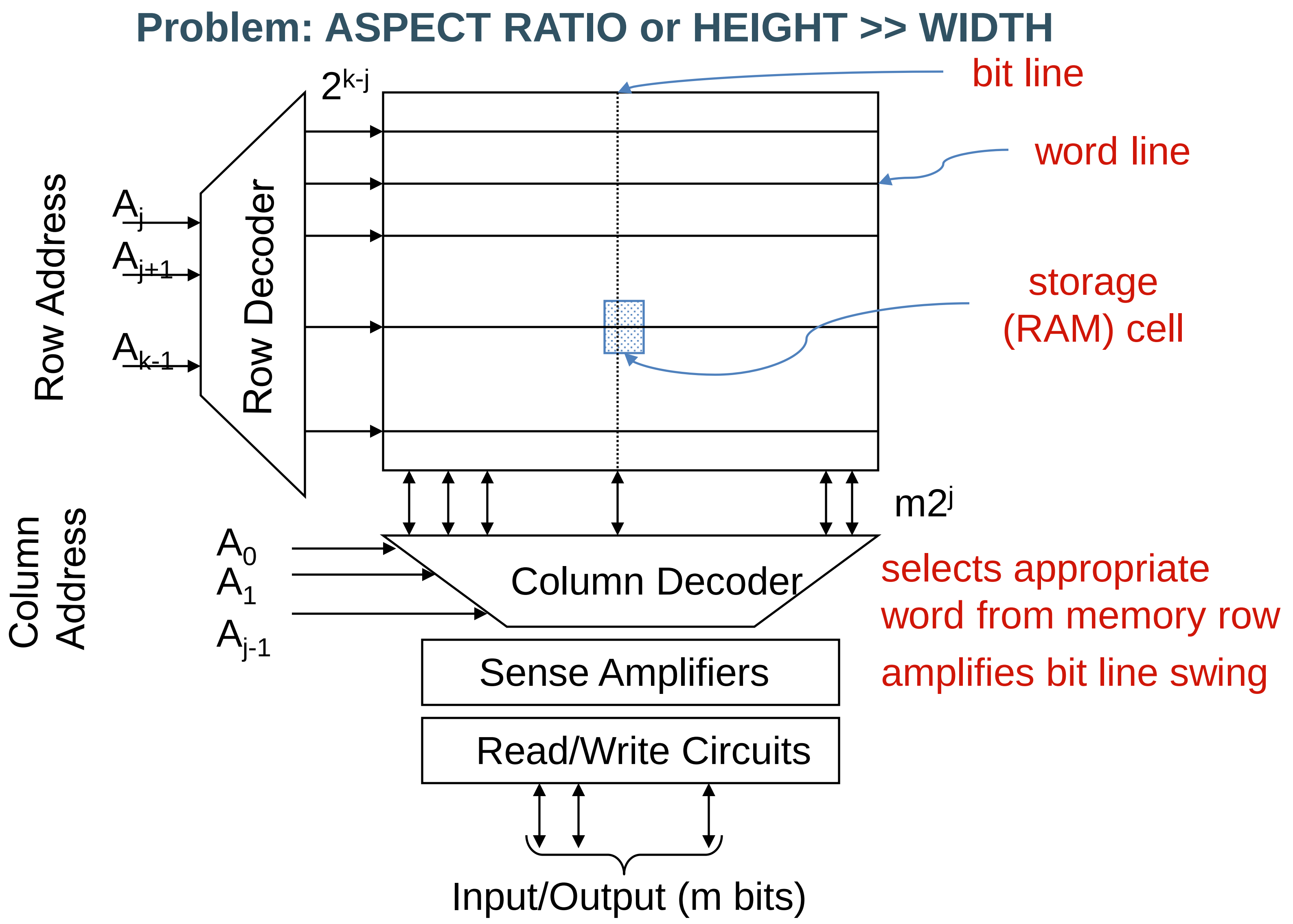
- Do two reads and one write by time multiplexing
  - Read during ph1, write during ph2

Multi-Ported SRAM (Register File)

- Adding more access transistors hurts read stability
- Multiported SRAM isolates reads from state node
- Single-ended bitlines save area

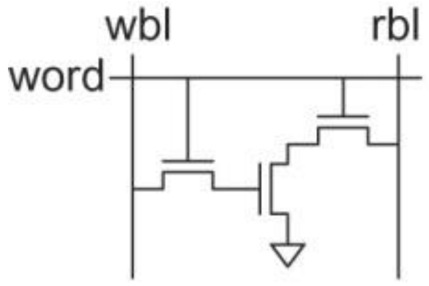


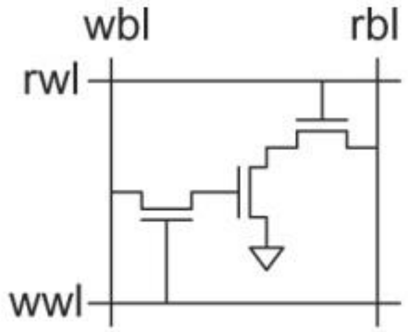
Array-Structured Memory Architecture

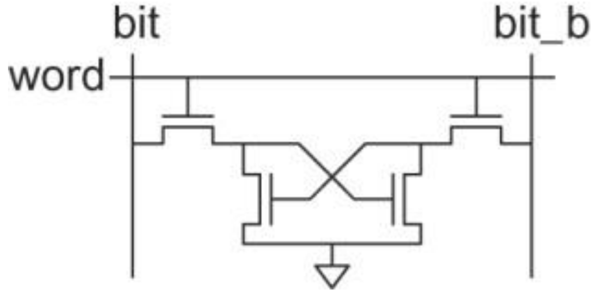


Embedded DRAM

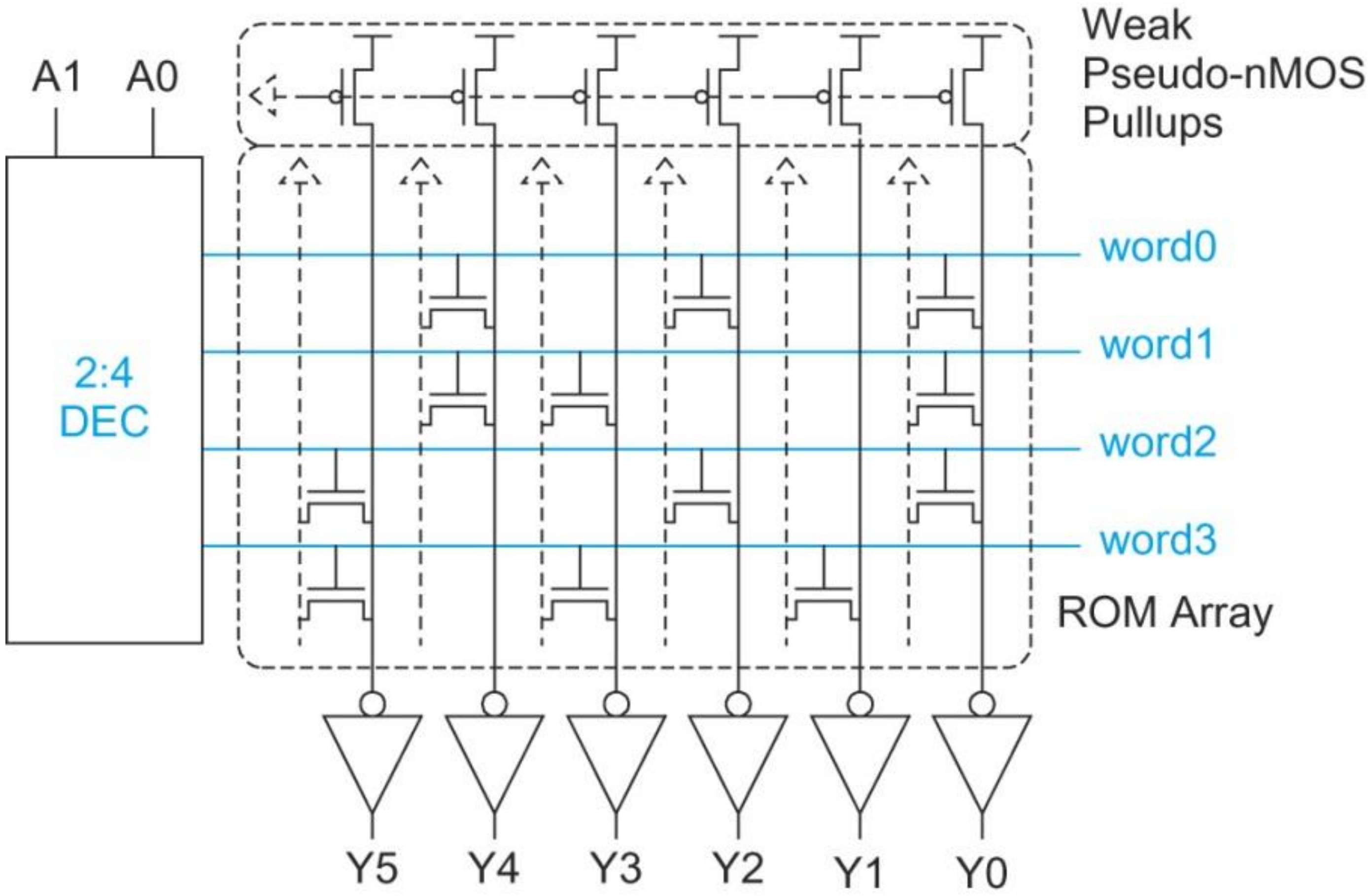
- Logic process
- 3T or 4T cells





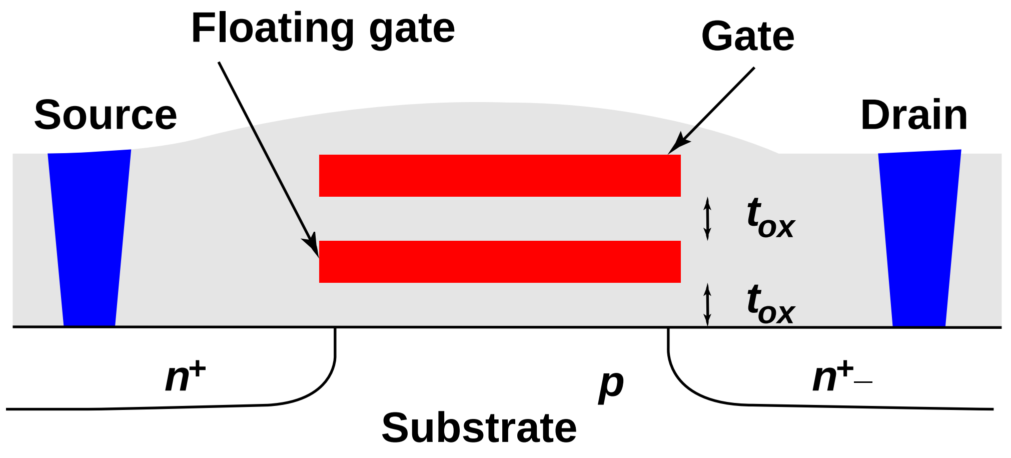


Read-Only Memory (ROM)

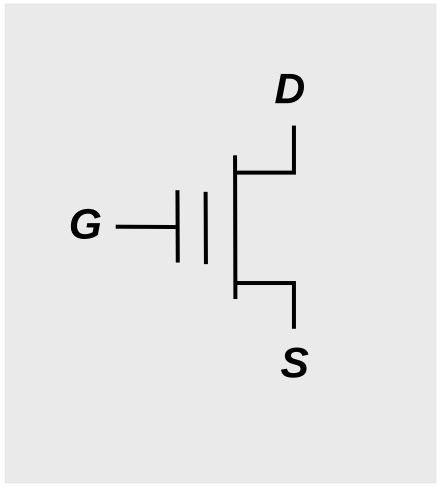


Non-Volatile Memories

- Flash - floating-gate transistor

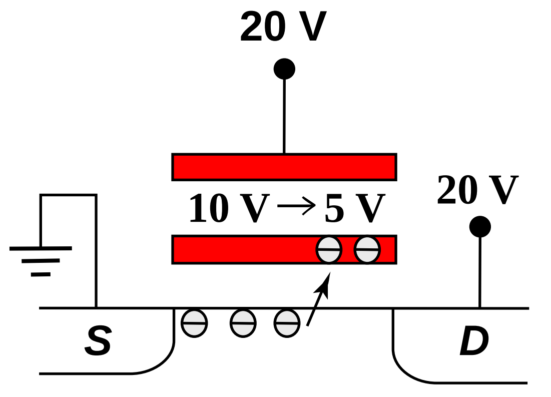


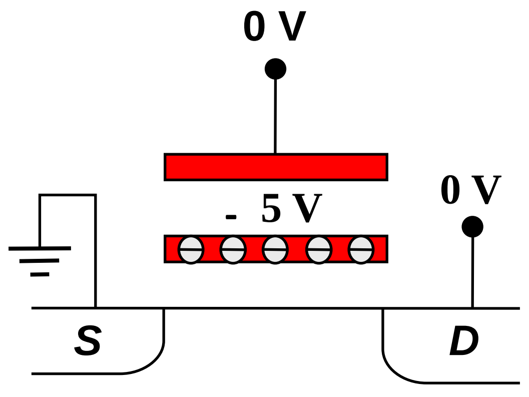
Device cross-section

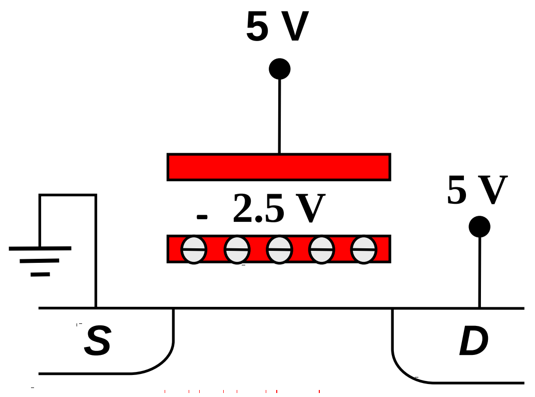


Schematic symbol

Floating-Gate Transistor Programming







Avalanche injection

Removing programming voltage leaves charge trapped

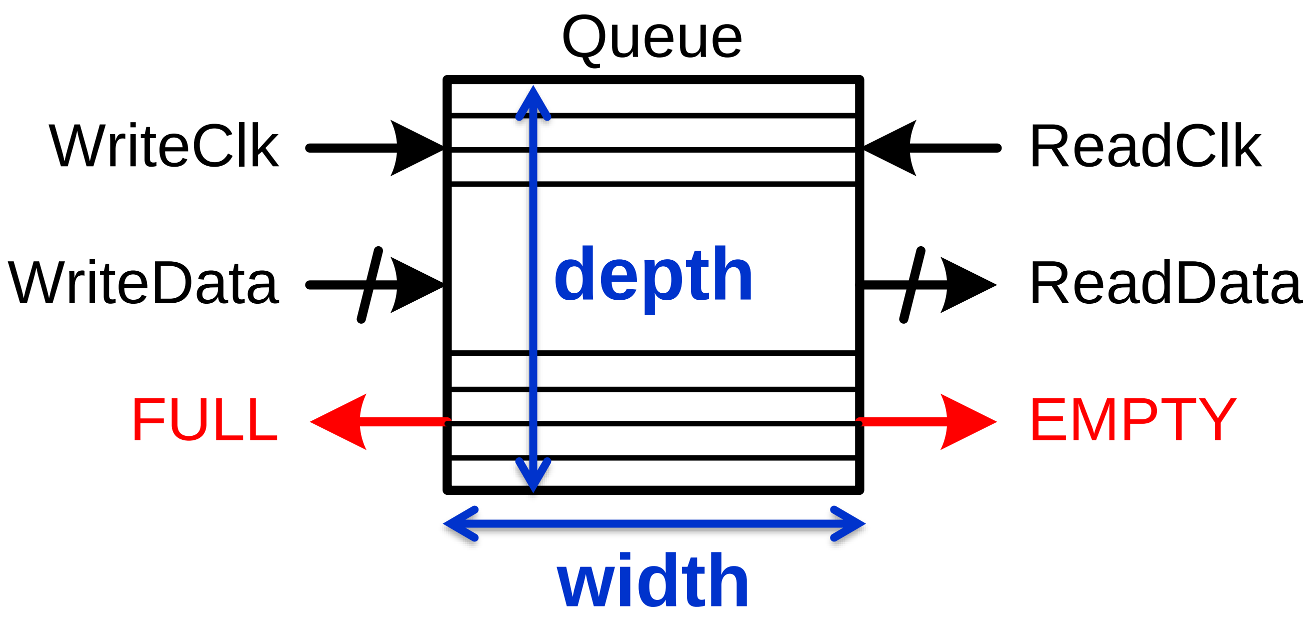
Programming results in higher V_T.

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)

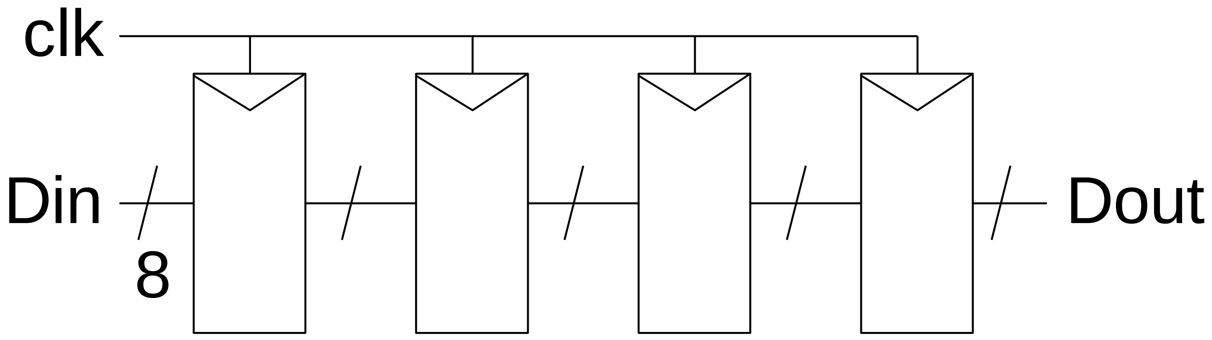
Queues

- 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
- Synchronous, semi-synchronous, asynchronous



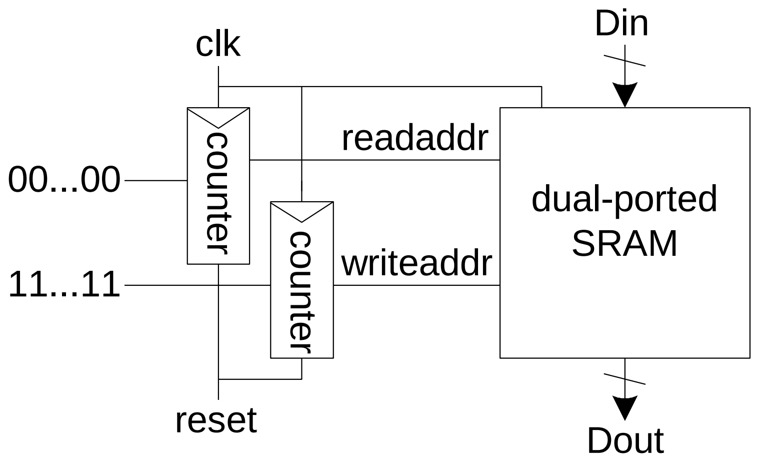
Shift Register

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



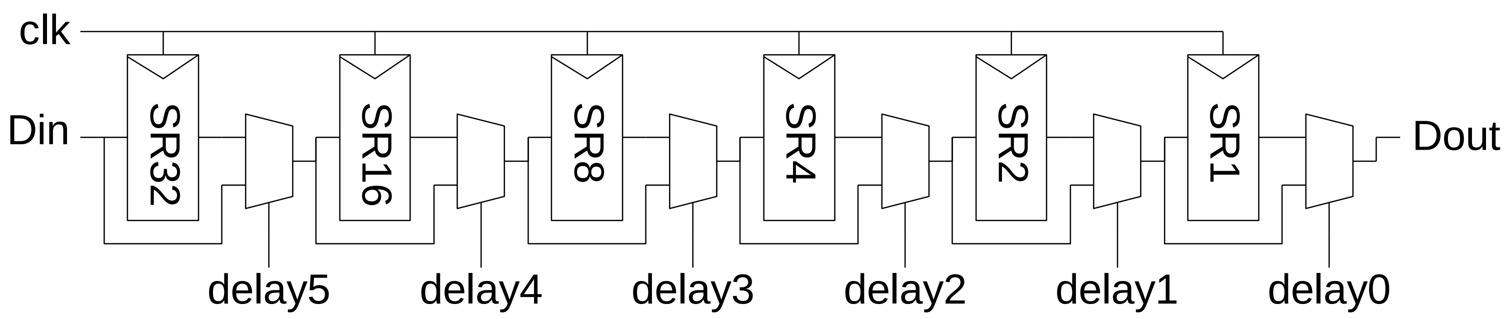
Denser Shift Registers

- 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



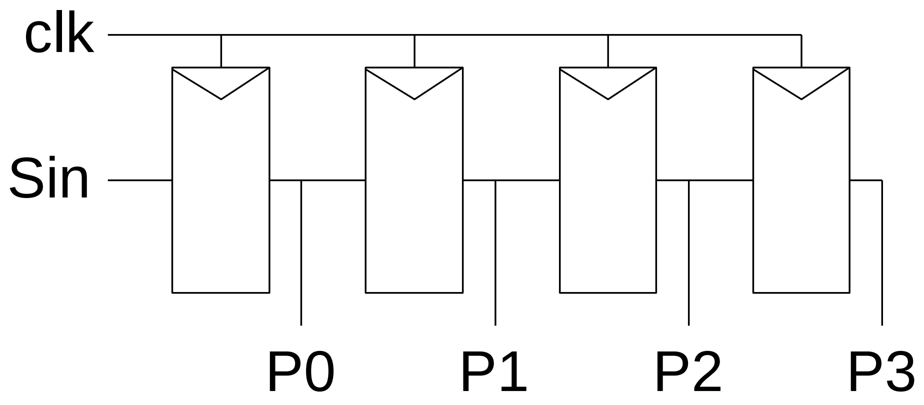
Tapped Delay Line

- 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



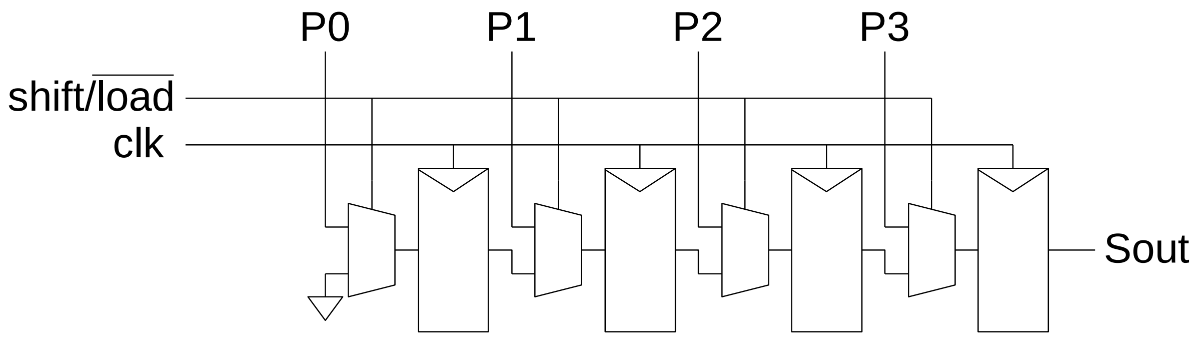
Serial In Parallel Out

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



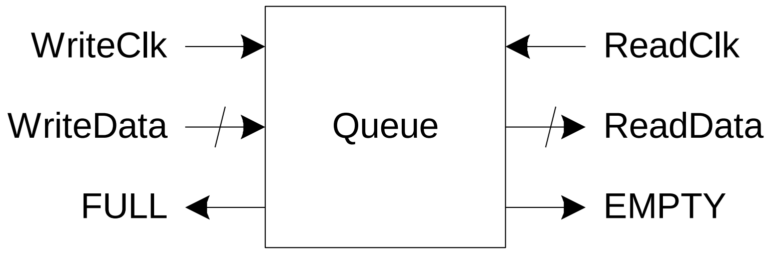
Parallel In Serial Out

- 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)

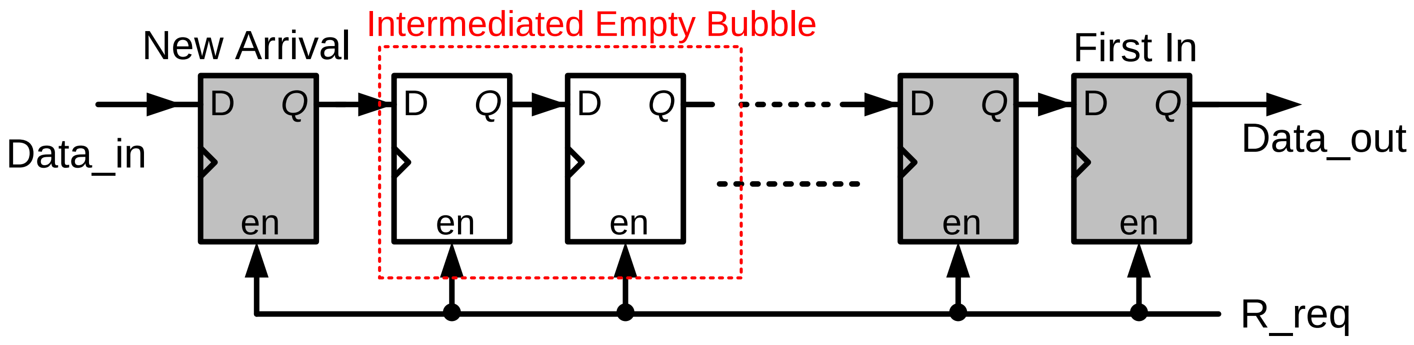


FIFO, LIFO Queues

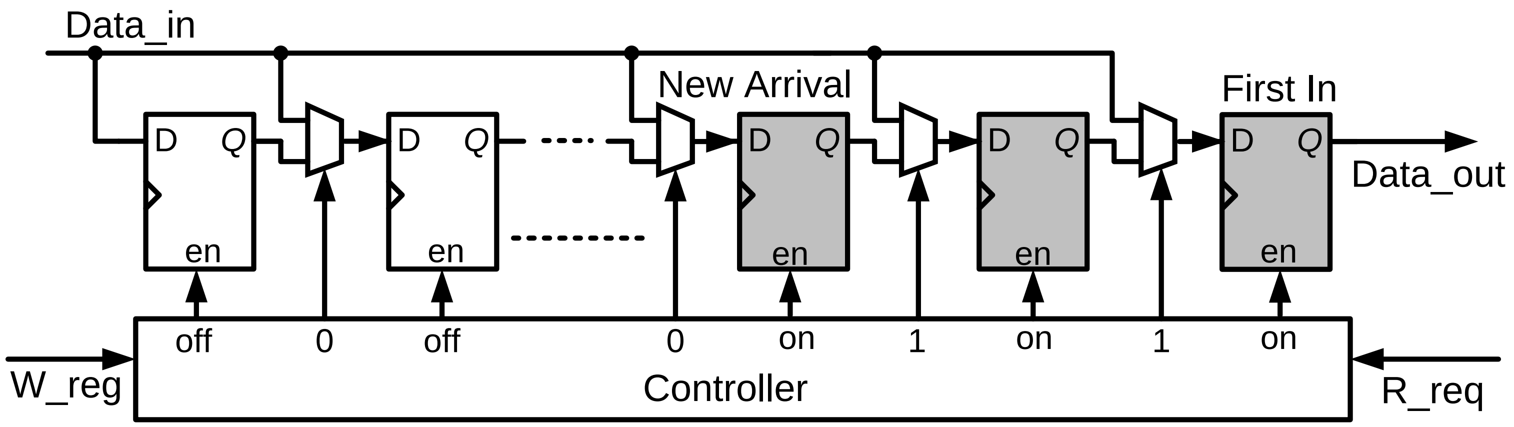
- *First In First Out* (FIFO)
  - Initialize read and write pointers to first element
  - Queue is EMPTY
  - On write, increment write pointer
  - If write almost catches read, Queue is FULL
  - On read, increment read pointer
- *Last In First Out* (LIFO)
  - Also called a *stack*
  - Use a single *stack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

Implementation for Short-Depth Queues

- Register-based FIFO for short-depth queues
  - Shift-in shift-out FIFO

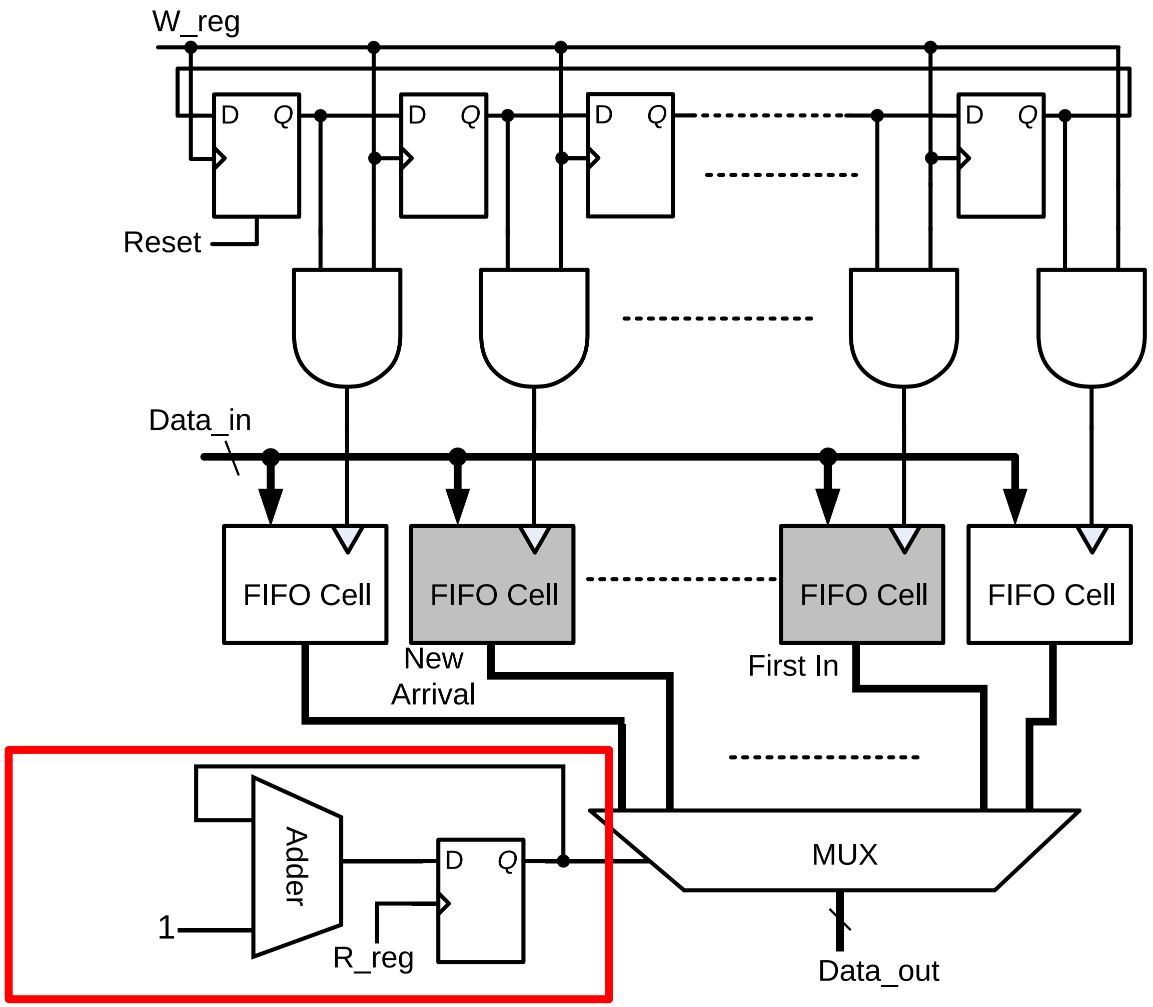


  - Bus-in shift-out FIFO



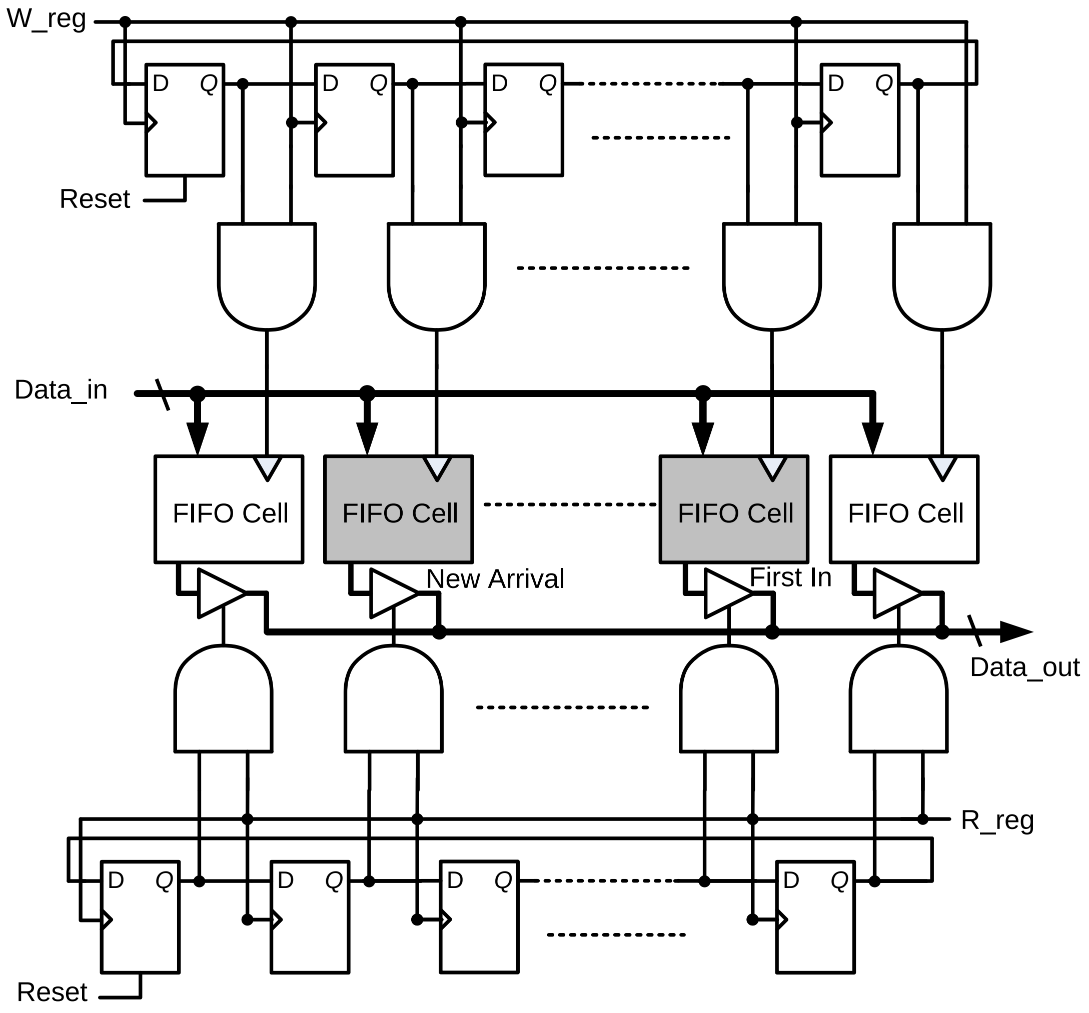
Bus-In Mux-Out Register-Based FIFO

- Read pointer for read token
- Complex wire routing



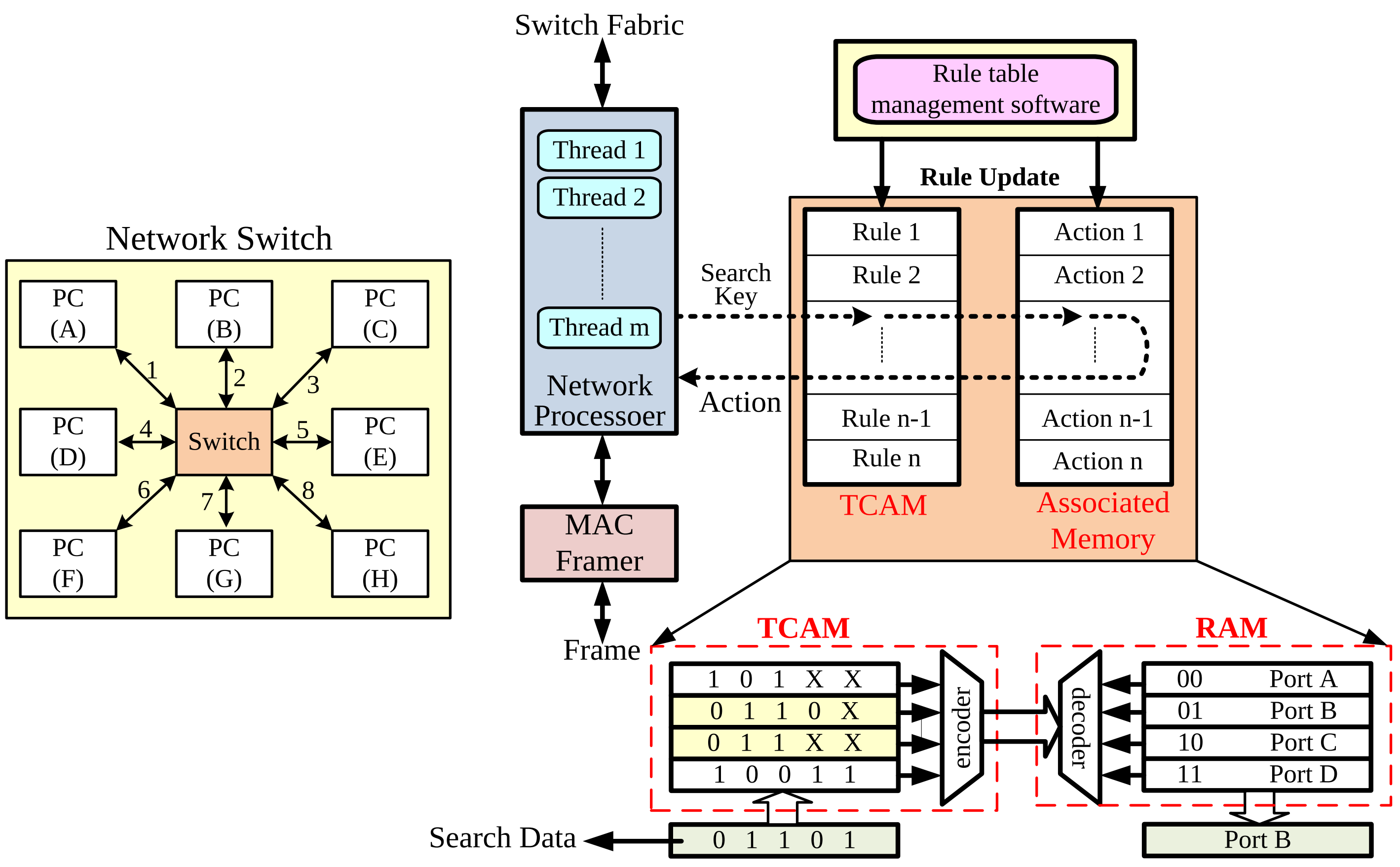
Bus-In Bus-Out Register-Based FIFO

- Registers for Read/write tokens
- Large capacitance on I/Os



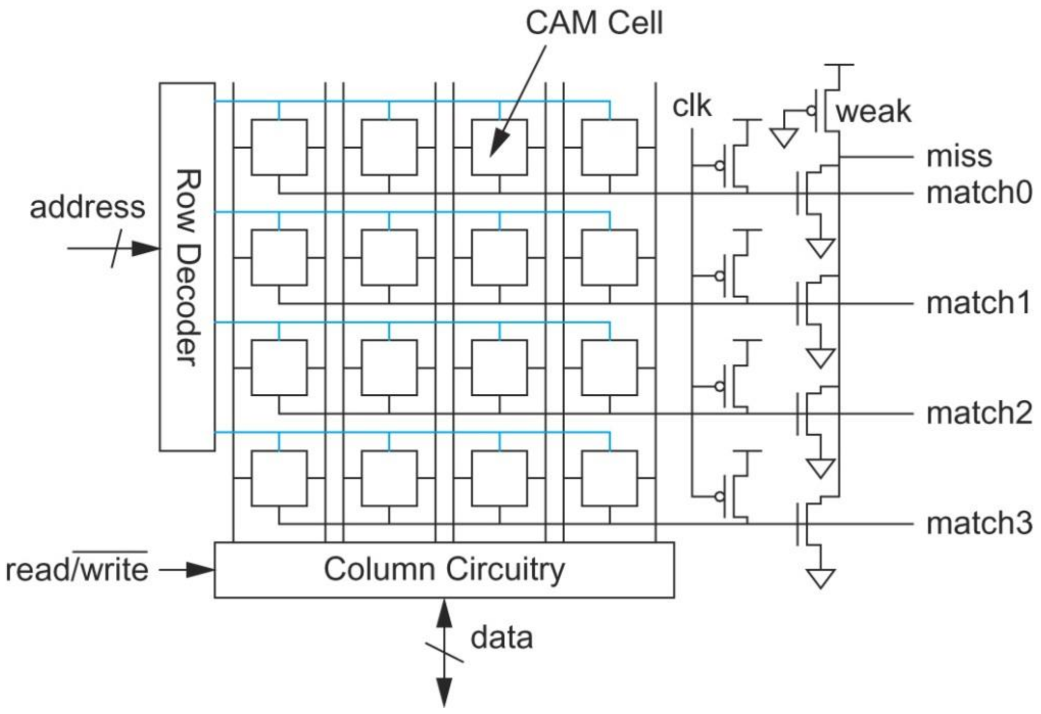
Introduction to CAM/TCAM

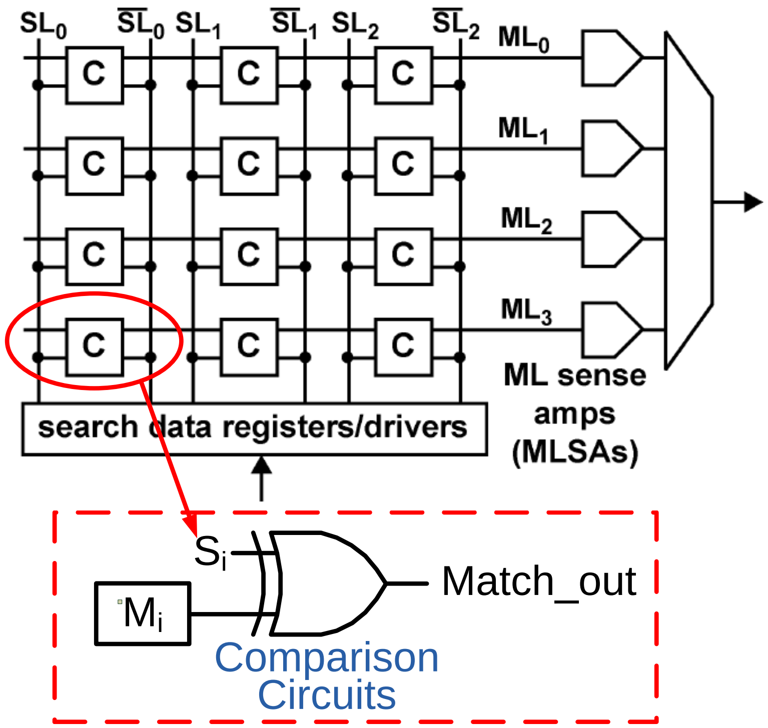
- Applications: lookup table
  - Great amounts of XORs
- Binary CAM and Ternary CAM (TCAM)



CAM Array

- Vertical lines: search-lines, bit-lines
- Horizontal lines: word-lines, match-lines
- Operation: Write, read, search

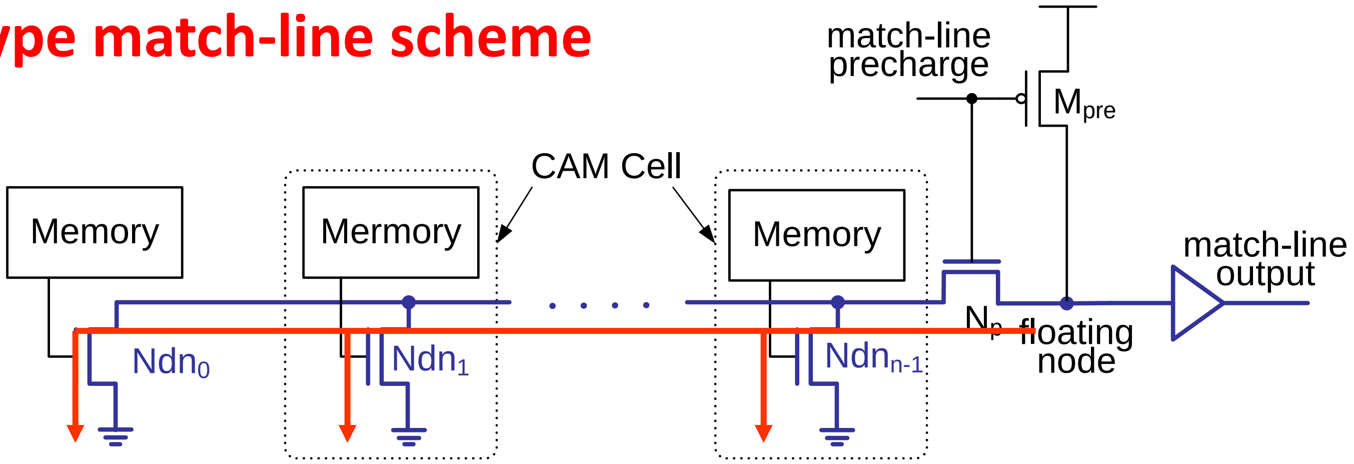




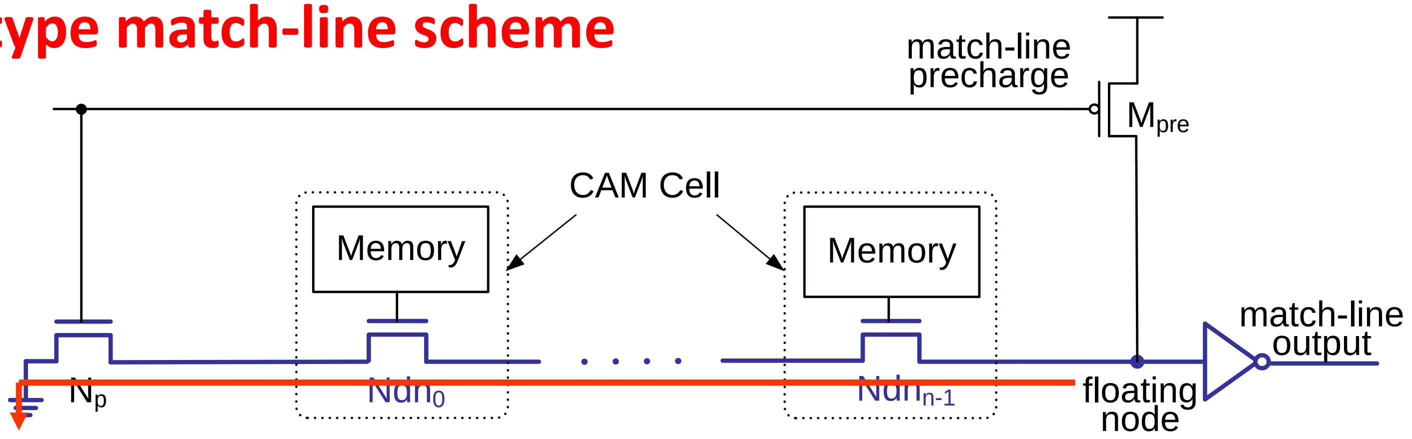
Match-line schemes for CAM

- Match-line: match/mismatch collection

NOR-type match-line scheme



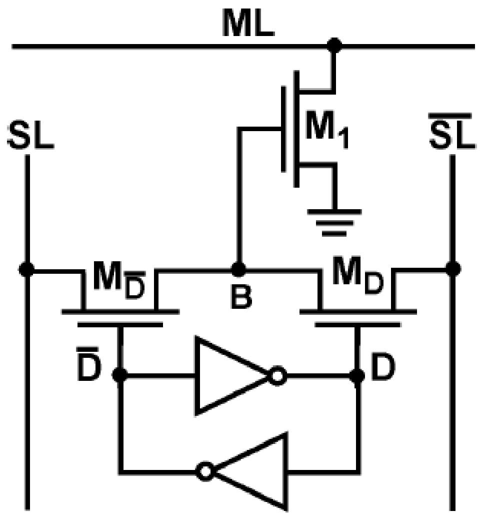
AND-type match-line scheme



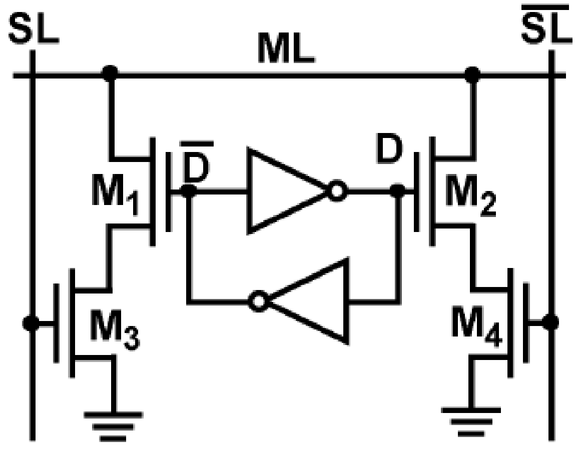
Operation of Binary CAM cell

◆ NOR-type CAM

State	Q_i	SL	ML
Zero (0)	0	0	floating
	0	1	0
One (1)	1	0	0
	1	1	floating



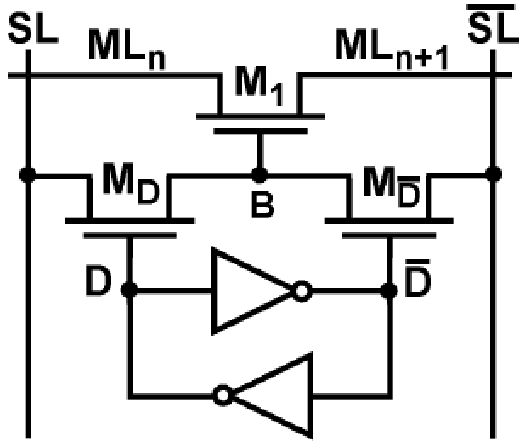
9T NOR



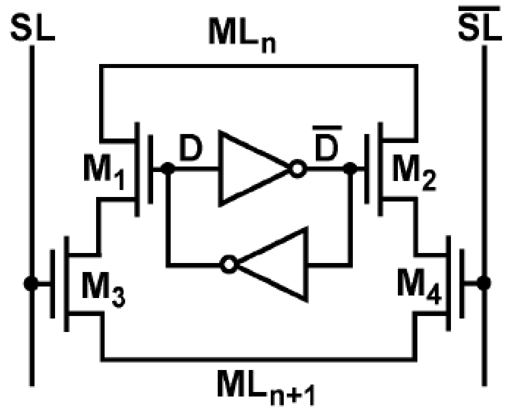
10T NOR

◆AND-type CAM

State	Q_i	SL	ML
Zero (0)	0	0	0
	0	1	floating
One (1)	1	0	floating
	1	1	0



9T NAND

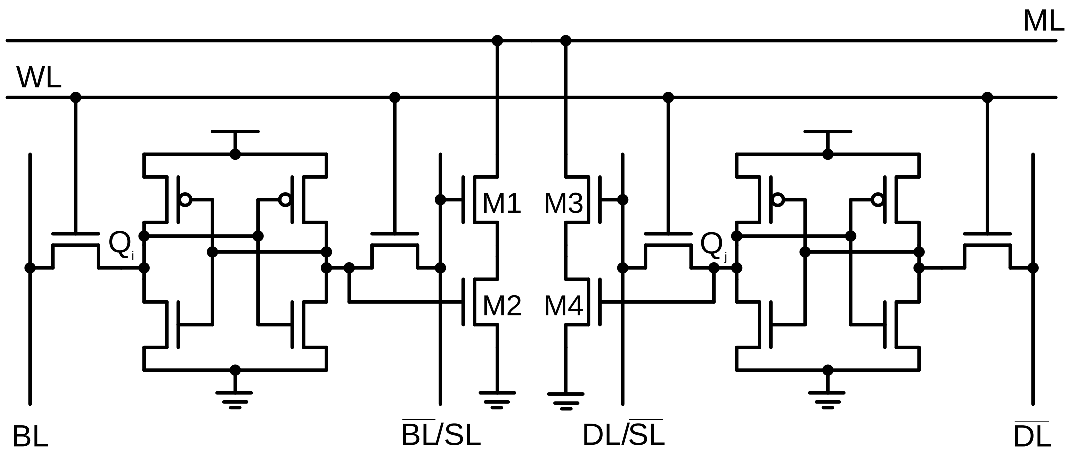


10T NAND

Operations of Binary CAM cell

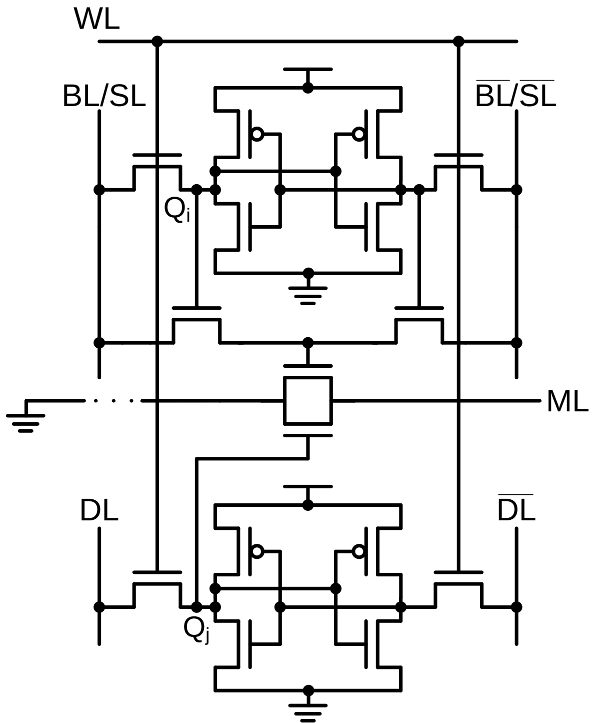
◆ NOR-type TCAM

State	Q_i	Q_j	SL	ML
Zero (0)	0	1	0	floating
	0	1	1	0
One (1)	1	0	0	0
	1	0	1	floating
Don’t Care (X)	0	0	0	floating
	0	0	1	floating
Not Allowed	1	1	0	—
	1	1	1	—



◆ AND-type TCAM

State	Q_i	Q_j	SL	ML
Zero (0)	0	0	0	0
	0	0	1	floating
One (1)	1	0	0	floating
	1	0	1	0
Don’t Care (X)	0	1	0	0
	0	1	1	0
	1	1	0	0
	1	1	1	0



WL
BL/SL
$\overline{BL}/\overline{SL}$
Q_i
ML
DL
$\overline{DL}$
Q_j