Array subsystems (RAM, ROM(flash), PLA,...)

Part I

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

Part II

- Content-Addressable Memories
- Read-Only Memories
- Programmable Logic Arrays

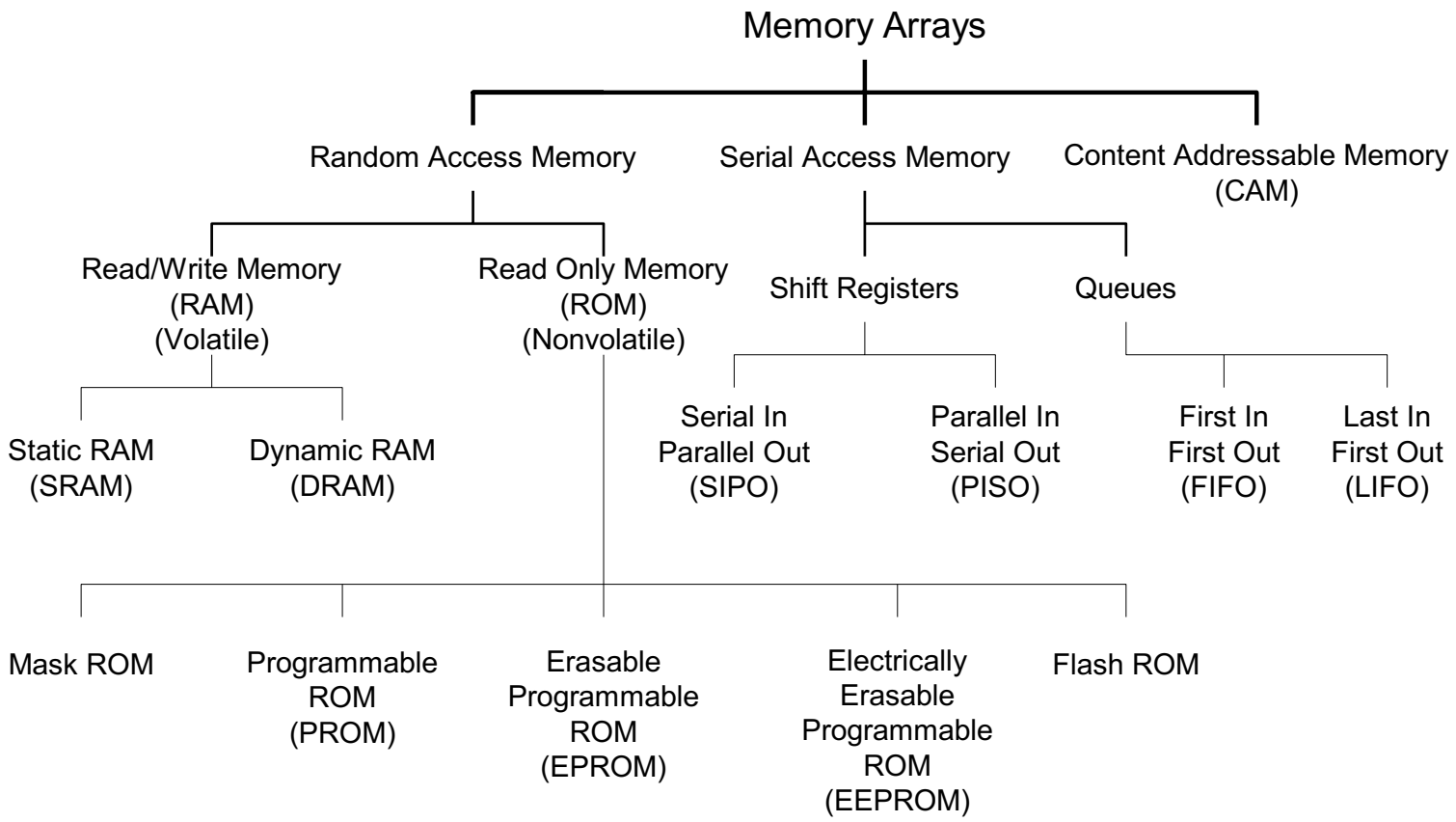
Part III

- Flash Memory

Part I

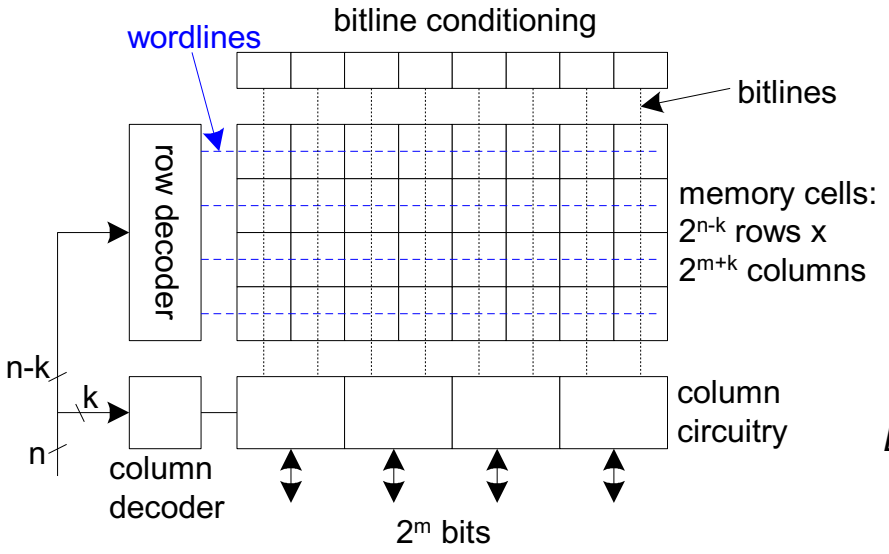
- Memory Arrays
- SRAM/DRAM Architecture
  - SRAM
  - DRAM
  - Decoders
  - Column Circuitry
  - Multiple Ports
- Serial Access Memories

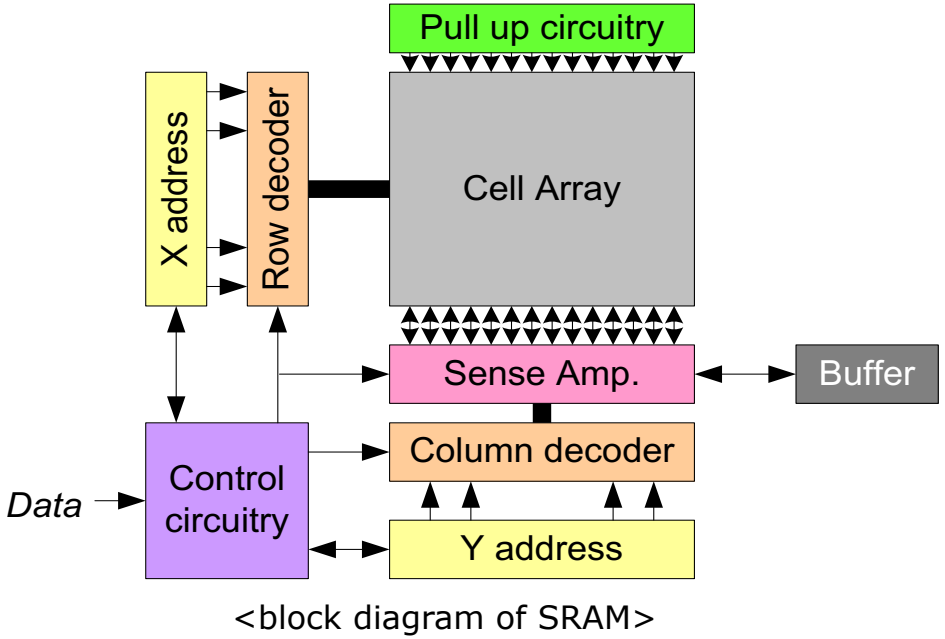
Memory Arrays



Array Architecture

- 2^n *words* of 2^m *bits* each
- If n >> m, fold by 2^k into fewer *rows* of more *columns*

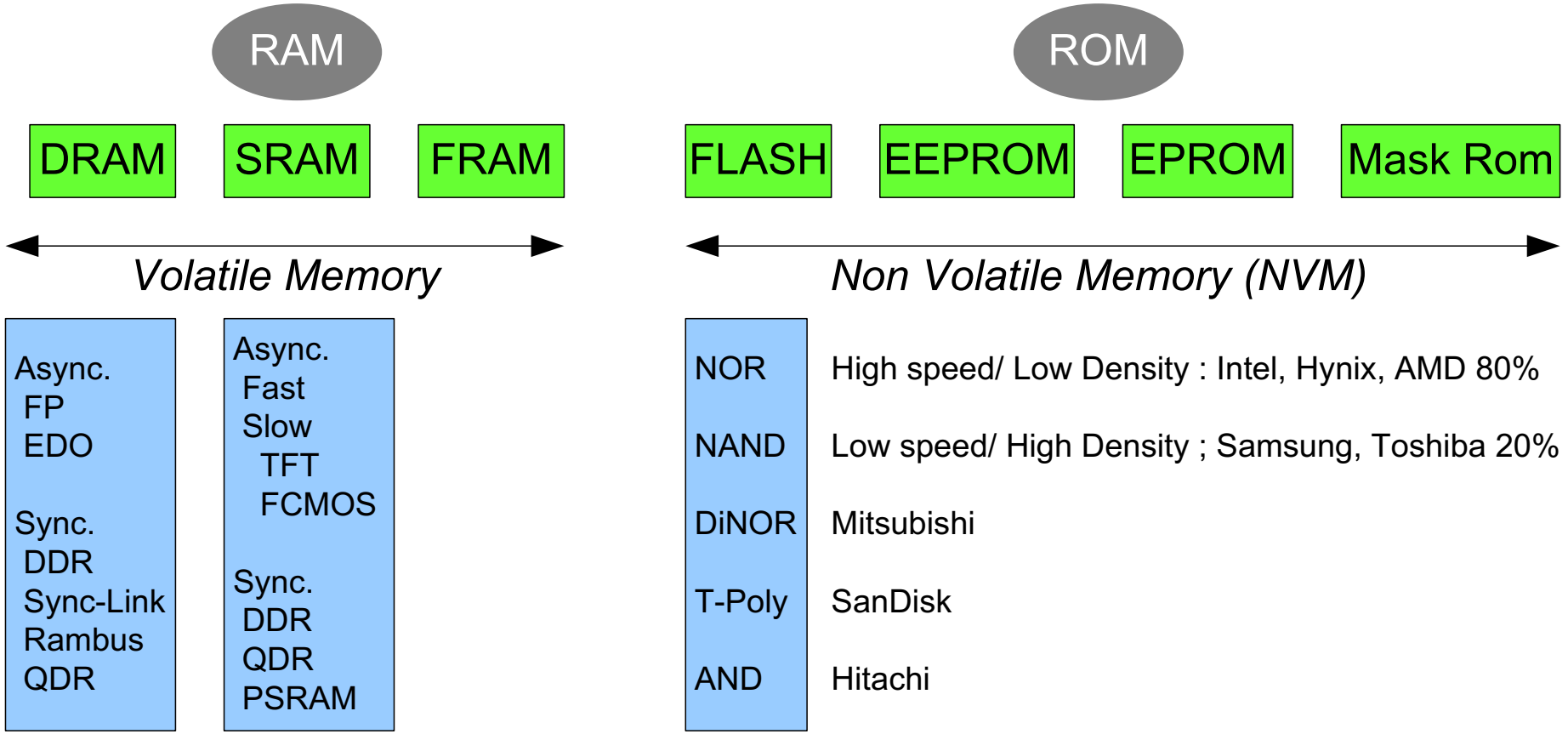




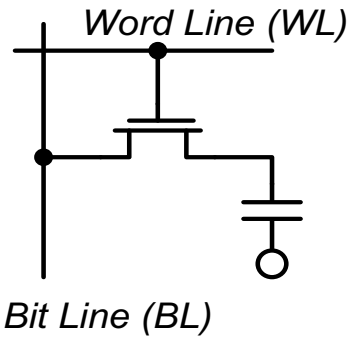
<block diagram of SRAM>

- Good regularity – easy to design
- Very high density if good cells are used

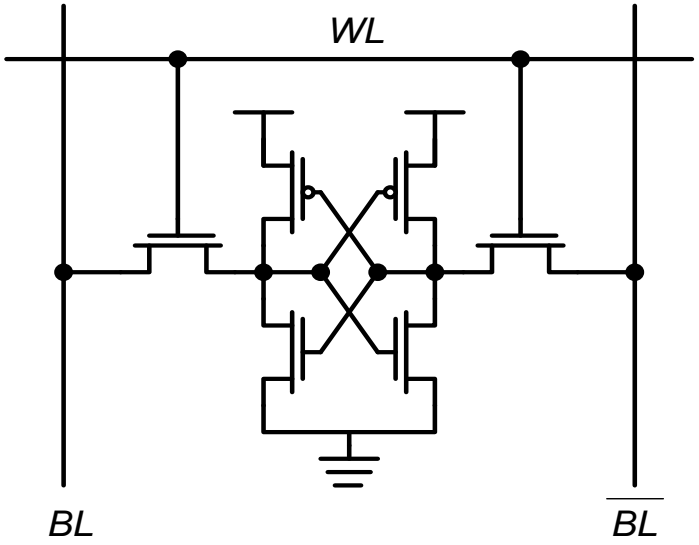
Memories and Their Applications



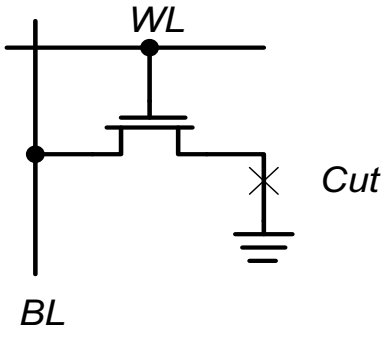
Equivalent Circuits of Memory Cells



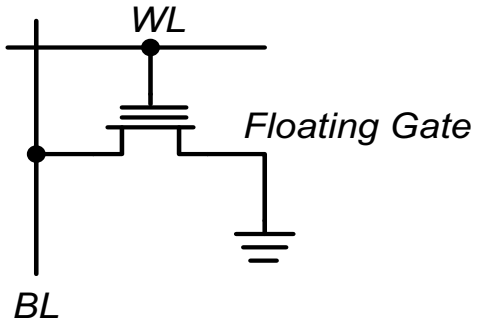
DRAM



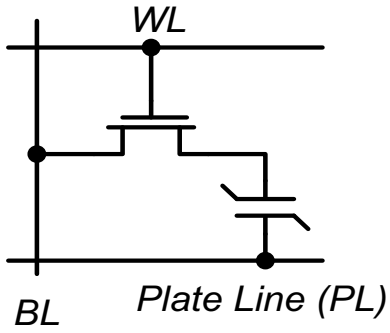
SRAM



Mask (Fuse) ROM

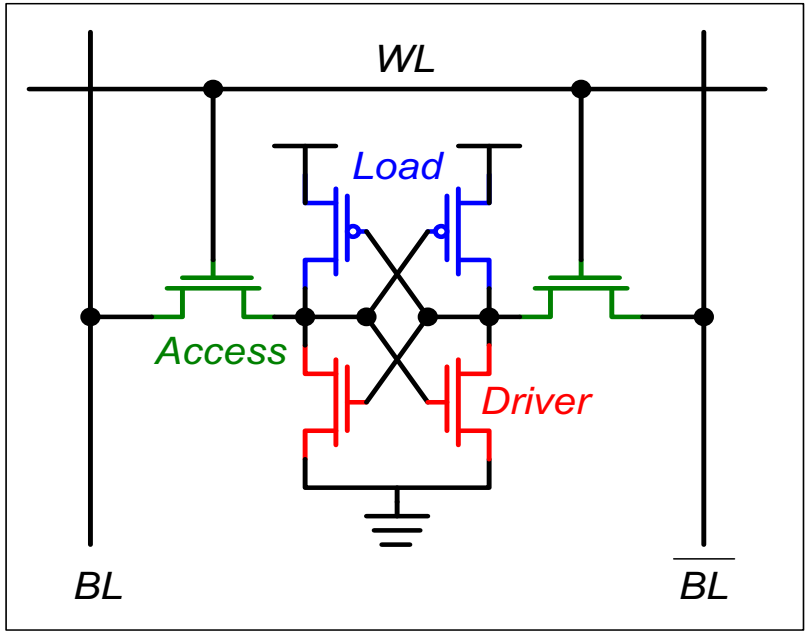
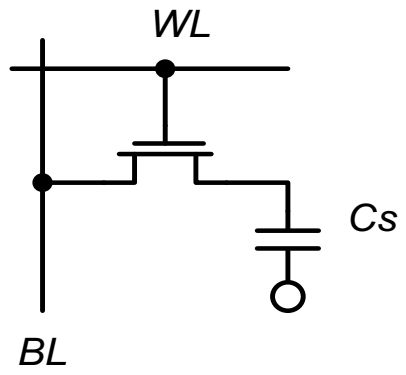


Erasable PROM (EPROM)

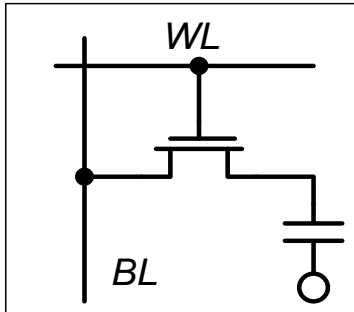
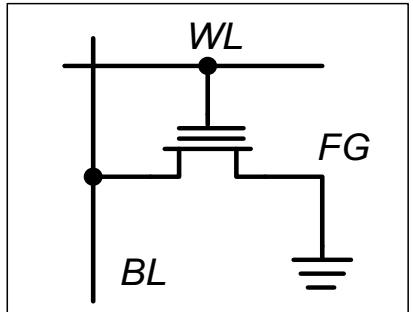
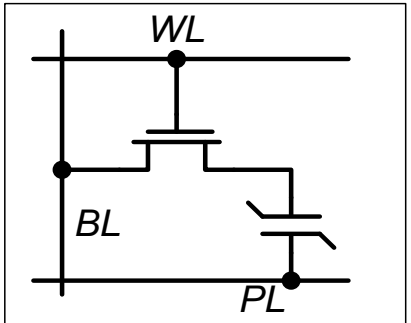
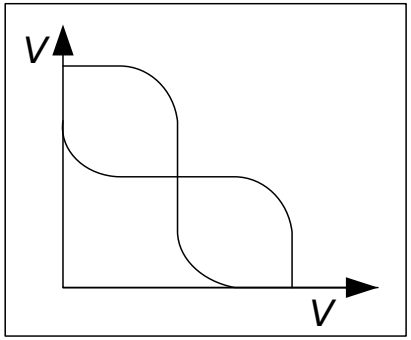
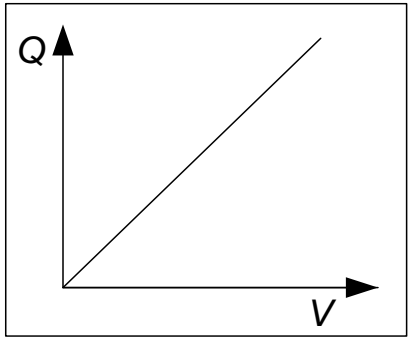
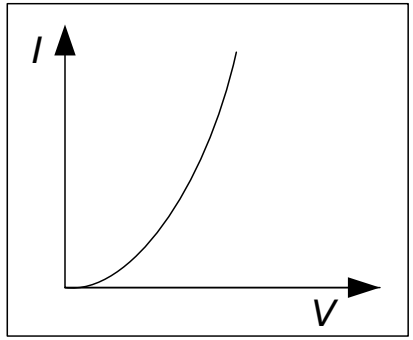
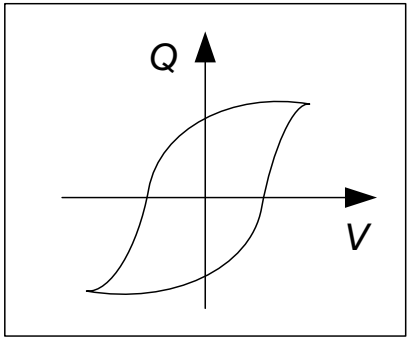


Ferroelectric RAM (FRAM)

Comparison of Memory cells

	SRAM CELL	DRAM CELL
Memory Cell Structure		
Features	- Static Random Access Memory (SRAM) - 4 Transistors + 2 Resistors (data latch) - Low density - Simple operation - Fast access time	- Dynamic Random Access Memory (DRAM) - 1 Transistor + 1 capacitor - High density - Complex operation (refresh…) - Slow access time → Fast access time (SDRAM, RDRAM…)

Comparison of Memory cells

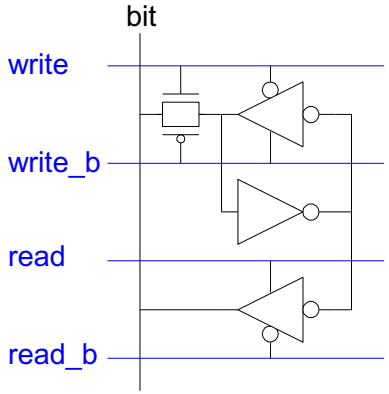
SRAM	DRAM	FLASH	FRAM
			
			
6 Tr. or 4 Tr. + 2 loads	1 Tr. + 1 Cap.	1 Tr.	1 Tr. + 1 Cap.
- No refresh - Volatile	- Refresh required - Volatile	- No Refresh - Non Volatile	- No Refresh - Non Volatile

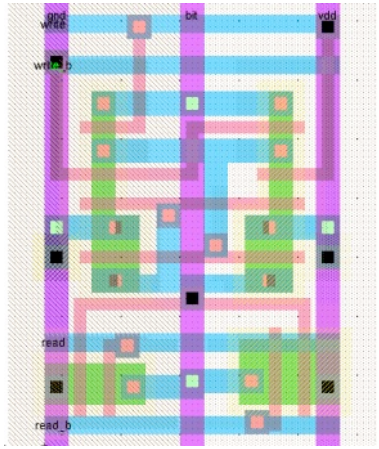
Comparison of Memory cells

	DRAM	SRAM	FLASH	FRAM
Data Storage	Charge & Discharge of the capacitor	Switching of Latch	Charge & Discharge of floating gate	Dipole switching of Dipole Ferro-Cap.
Read speed	~ 50 ns	~ 10/70 ns	~ 50 ns	~ 100 ns
Write speed	~ 40 ns	~ 5/40 ns	~ (10 us – 1 ms)	~ 100 ns
Write voltage	Low	Low	High	Middle
Time to erase	No need	No need	~ ms	No need
# of R/W cycles	Infinite (> 10^{15})	Infinite (> 10^{15})	$10^{15}/10^{6}$	> 10^{10}
Power	High	High/low	Low	High
Cost	Low	High	Low	Low

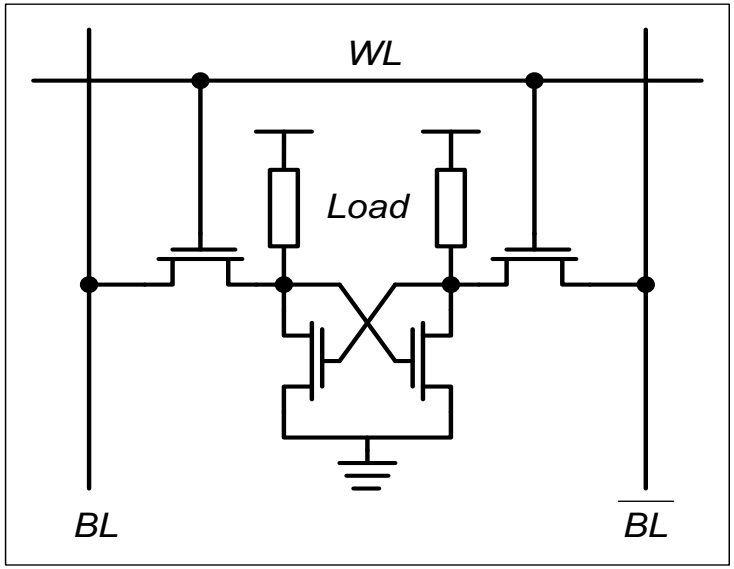
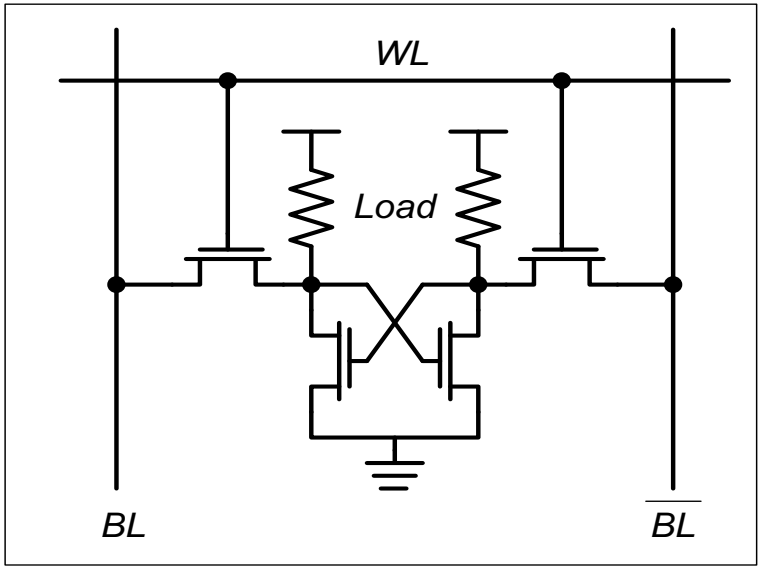
12T SRAM Cell

- 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

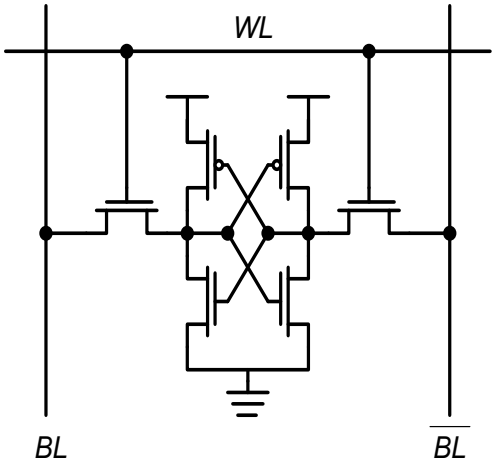




Various configurations of SRAM cell

	Generic circuit topology	*Resistive-load SRAM cell*
SRAM Cell Structure		
Features	- Load can be polysilicon resistors, depletion-type nMOS transistors, thin film transistors (TFT), pMOS transistors.	- R (polysilicon resistor) can be on top of Trs. → Relatively small area - High resistance → low power - Low resistance → wide noise margin, high speed

Full CMOS Cell



-At the beginning, Isolation (pWell, nWell) → Larger size

-Recently, Shallow Trench Isolation (STI)

-Dual Damascene

-Cell ration can be reduced → overcome size problem.

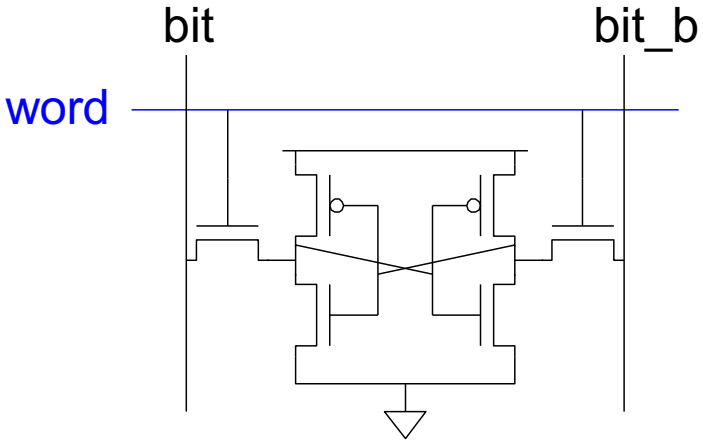
-Low power supply

-Logic Process Compatible

- Advantage
  - Low stand-by current
  - Low power supply
  - High noise margin
  - Good SER
  - high switching speed
- Disadvantages
  - Large cell size
  - Complexity of the CMOS process (Local interconnect with in a cell)
  - Latch-up phenomena

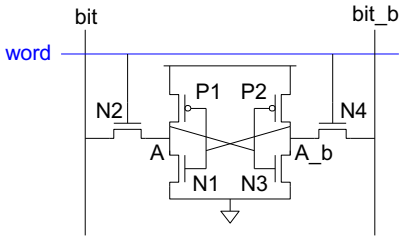
6T SRAM Cell

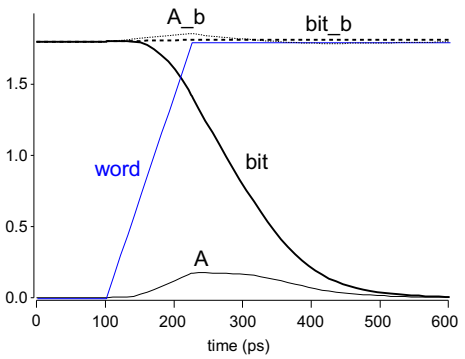
- Cell size accounts for most of array size
  - Reduce cell size at expense of complexity
- 6T SRAM Cell
  - Used in most commercial chips
  - Data stored in cross-coupled inverters
- Read:
  - Precharge bit, bit_b
  - Raise wordline
- Write:
  - Drive data onto bit, bit_b
  - Raise wordline



SRAM Read

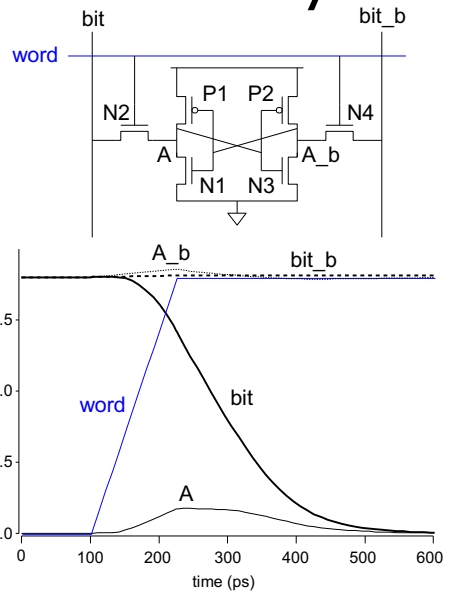
- Precharge both bitlines high
- Then turn on wordline
- One of the two bitlines will be pulled down by the cell
- Ex: A = 0, A_b = 1
  - bit discharges, bit_b stays high
  - But A bumps up slightly
- *Read stability*
  - A must not flip





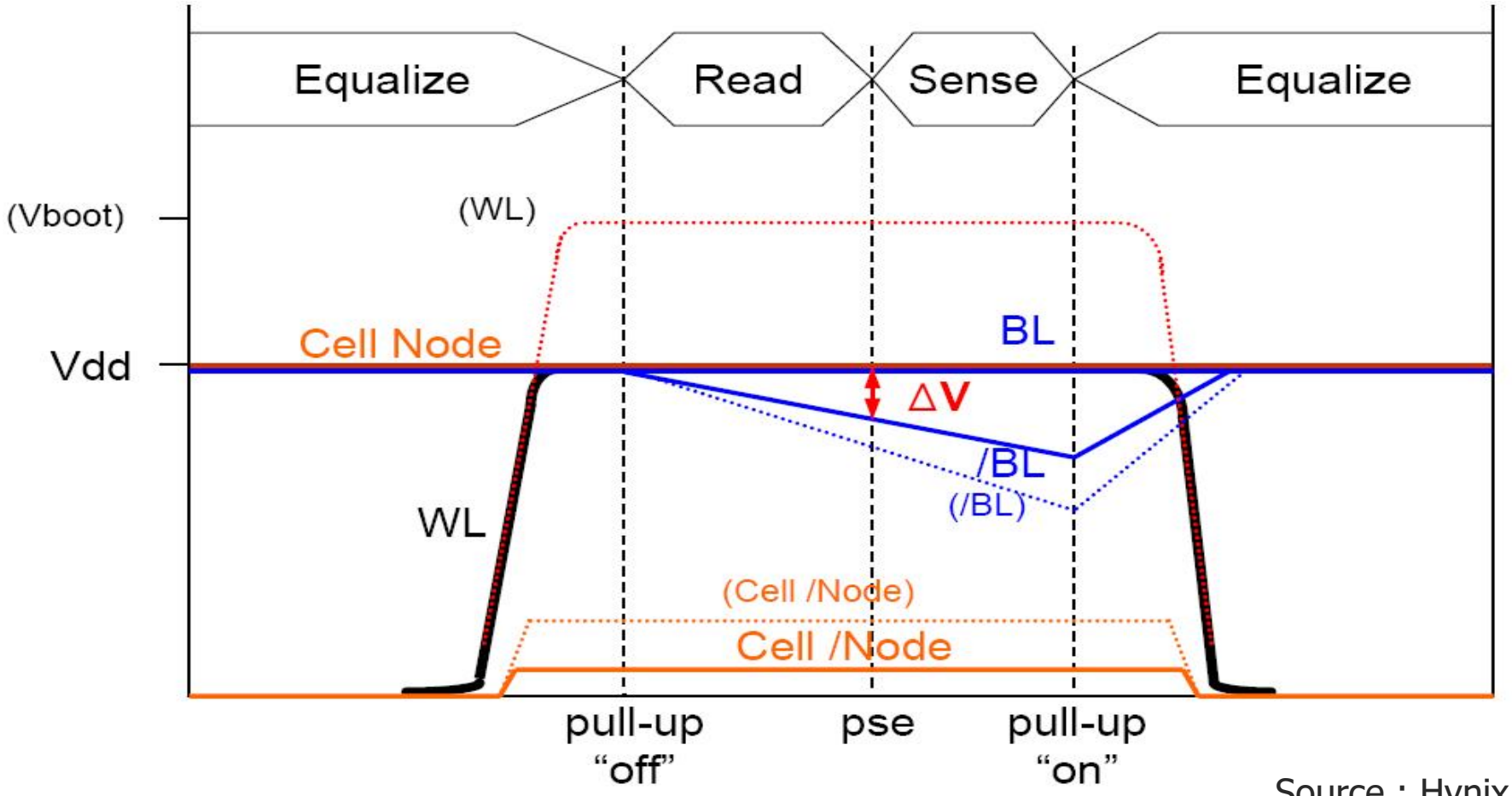
SRAM Read

- 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
    - driving capability of N1 should be stronger than that of N2
    - (R_{N1} << R_{N2} → small current of N2 and node A has about ground voltage)



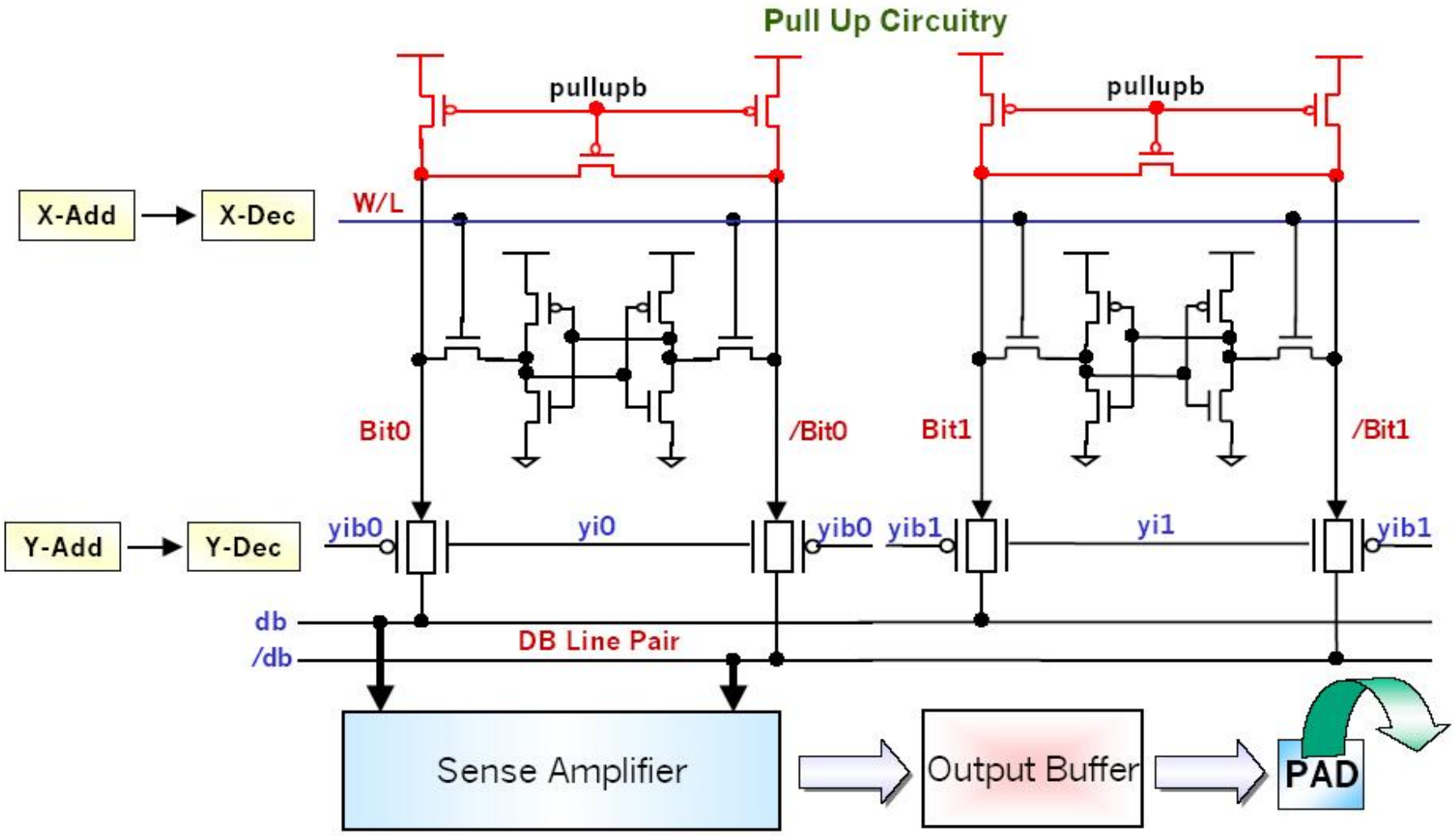
SRAM : Static Read Current (Discharge)

$$\Delta V = (I_{read} / C_{tot})\Delta t$$



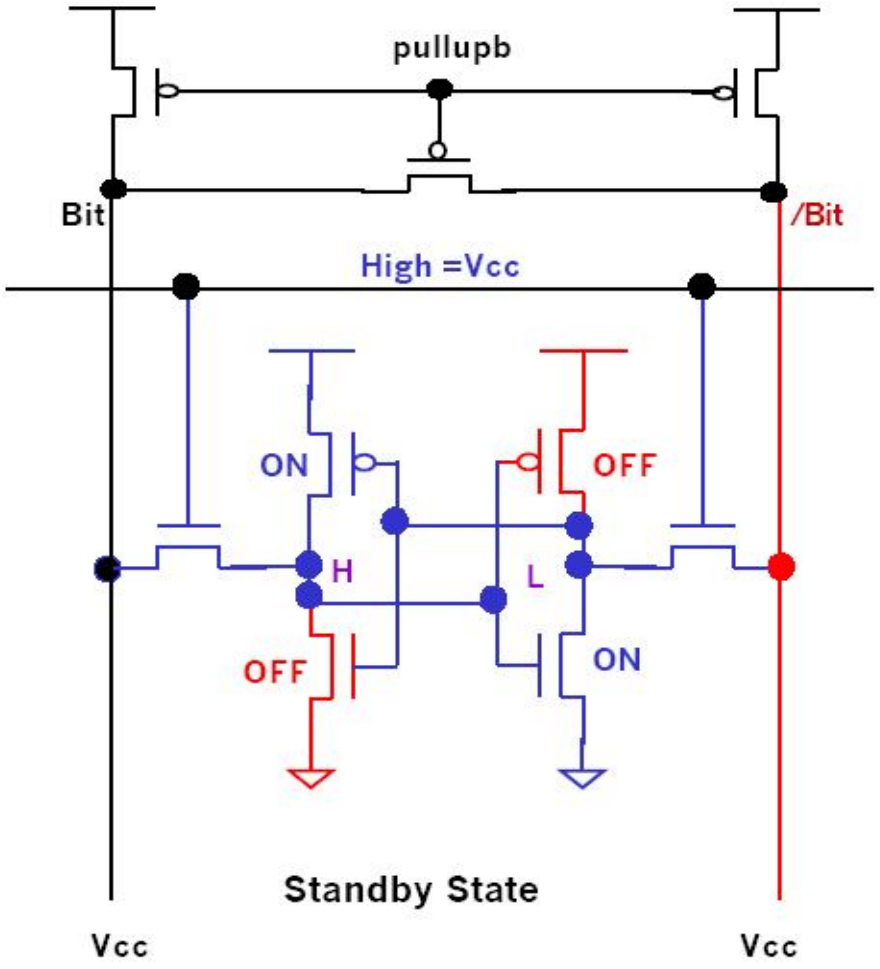
Source : Hynix

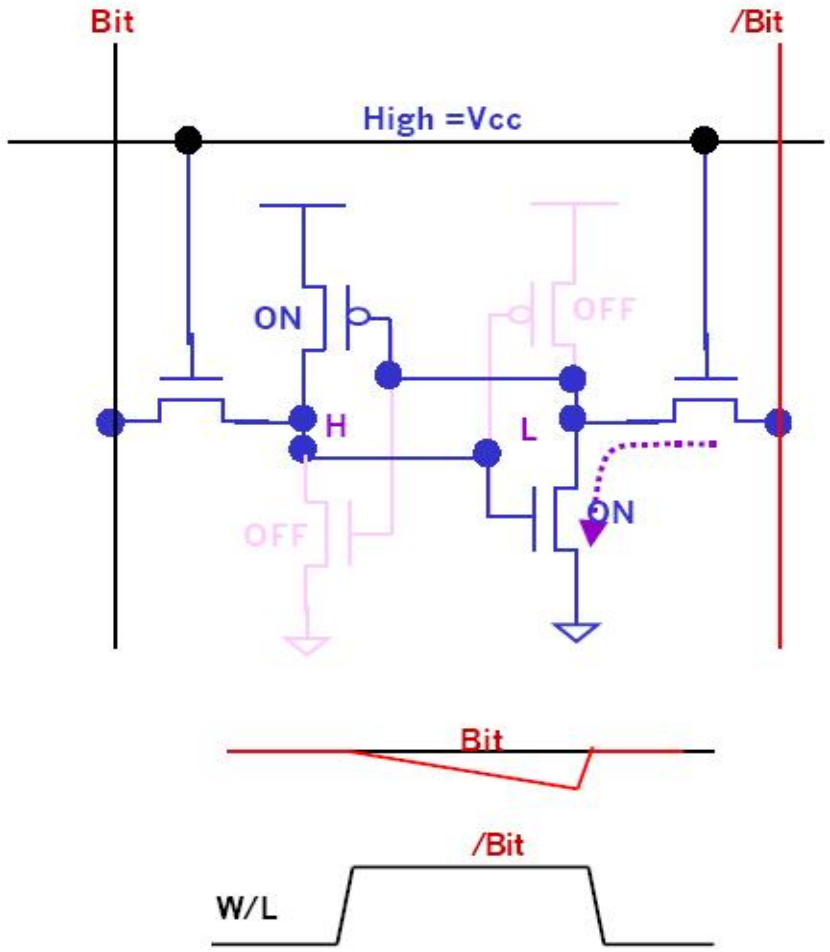
Read Operation : 1 Bit Read



Source : Hynix

Read Operation : Cell Operation

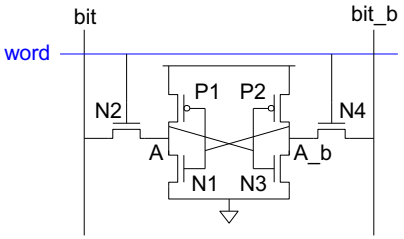


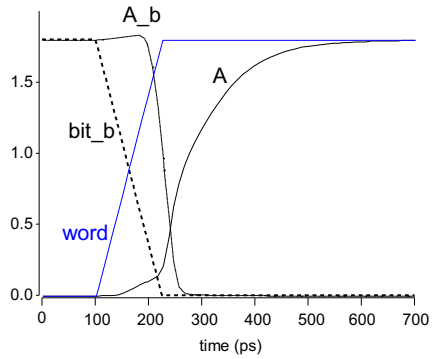


Source : Hynix

SRAM Write

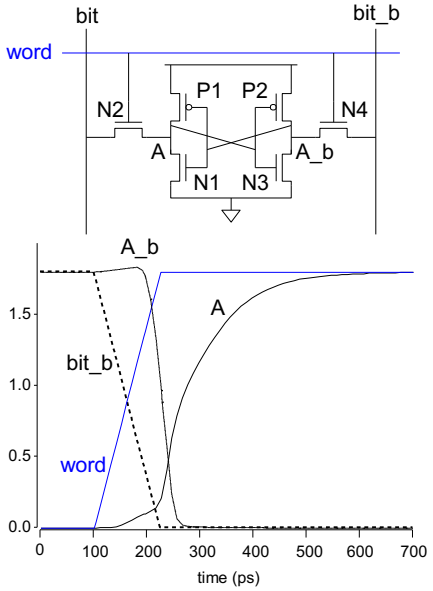
- Drive one bitline high, the other low
- Then turn on wordline
- Bitlines overpower cell with new value
- Ex: A = 0, A_b = 1, bit = 1, bit_b = 0
  - Force A_b low, then A rises high
- *Writability*
  - Must overpower feedback inverter



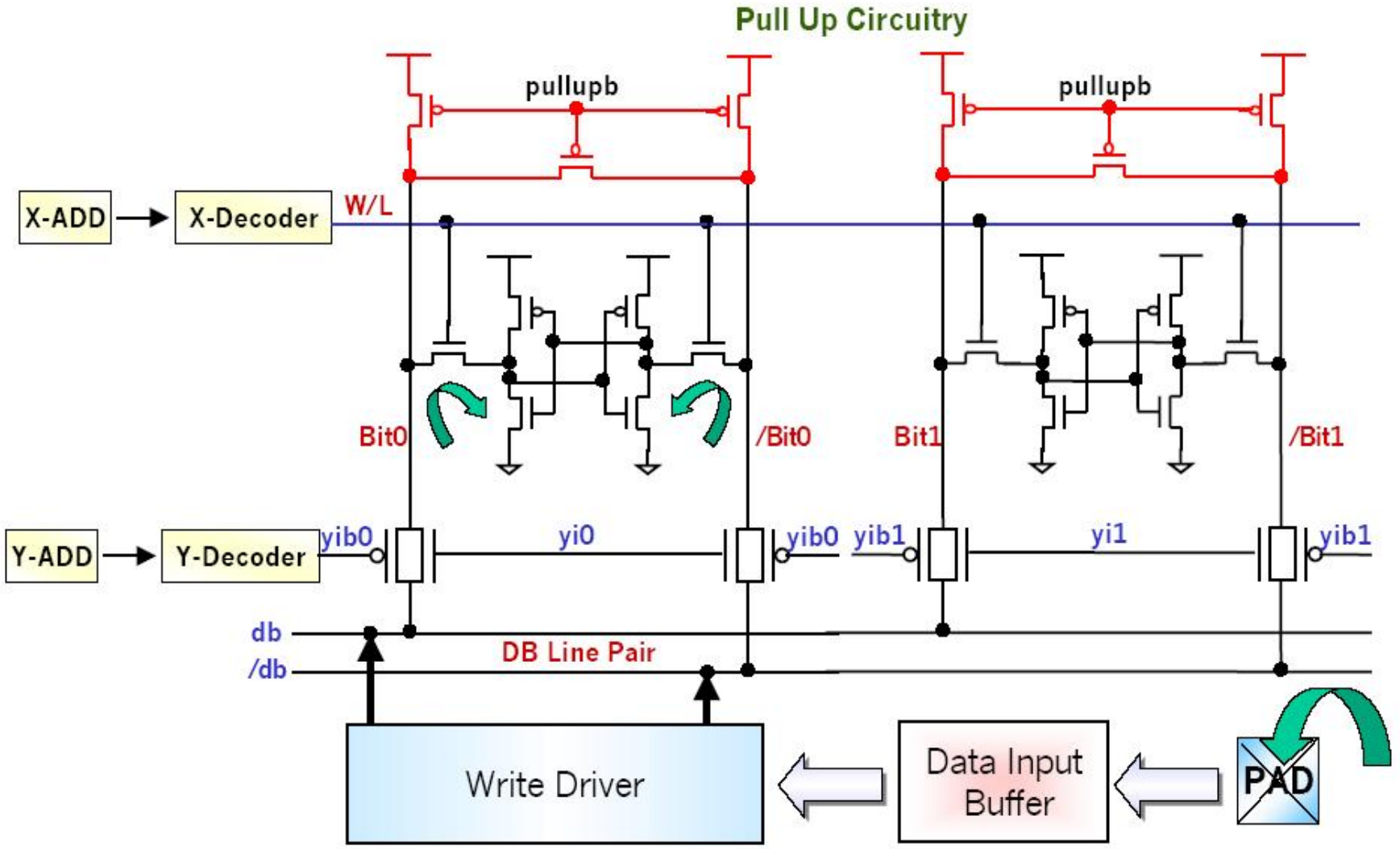


SRAM Write

- 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 (why?)
  - What about the case when A stores 1?
- *Writability*
  - N2 cannot drive N1
  - Forcing A_b low through N4
  - P2 opposes this operation.
  - Must overpower feedback inverter
  - N4(N2) >> P2(P1)
    - Since P2(P1) should not drive N4(N2)
    - If N4 >> P2 or N2 >> P1, then
    - $R_{p2} >> R_{n4}$ or $R_{p1} >> R_{n2} \rightarrow V_{A\_b} \sim 0$

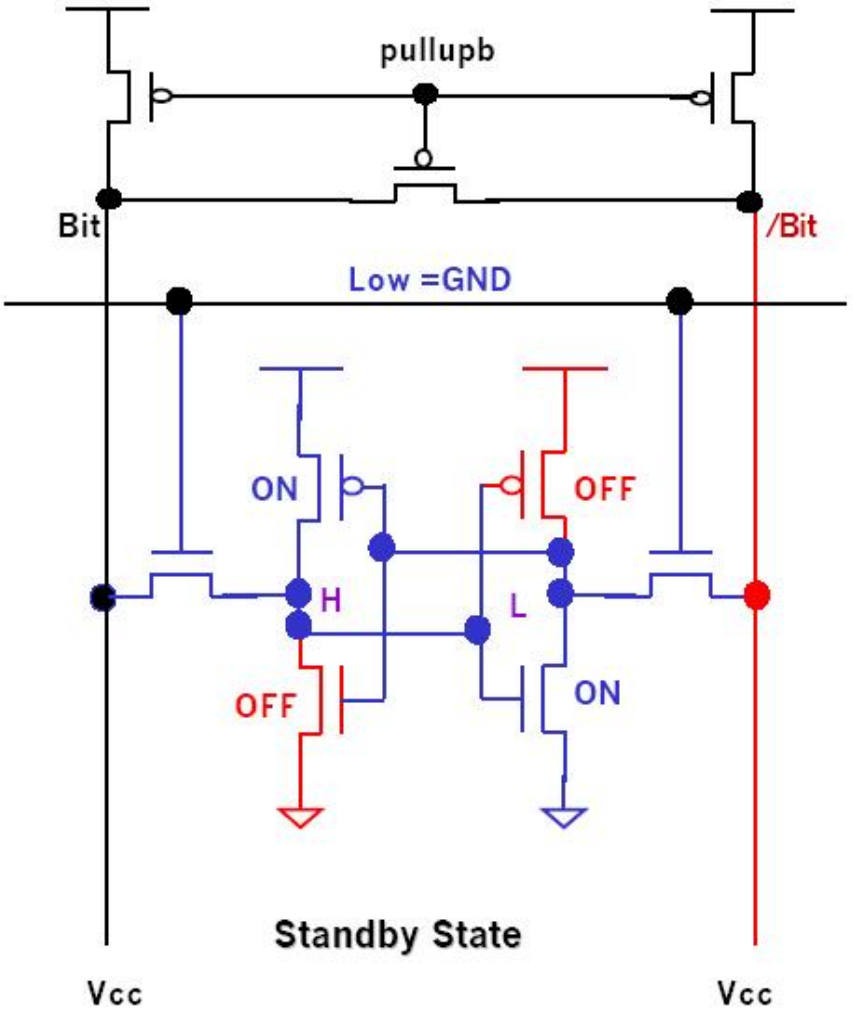


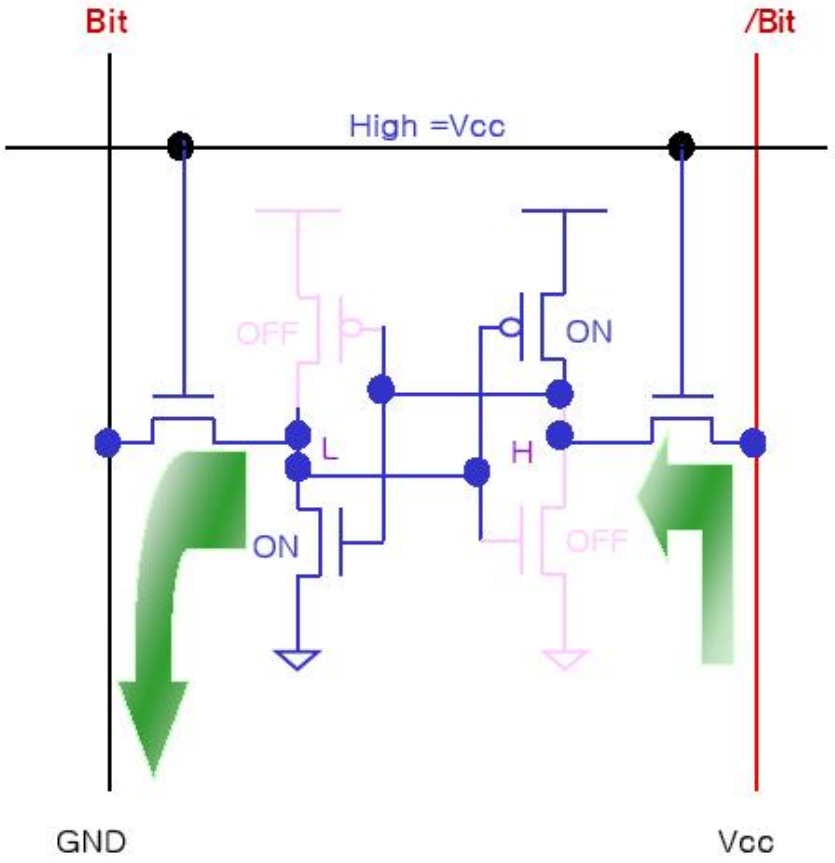
Write Operation : 1 Bit Write



Source : Hynix

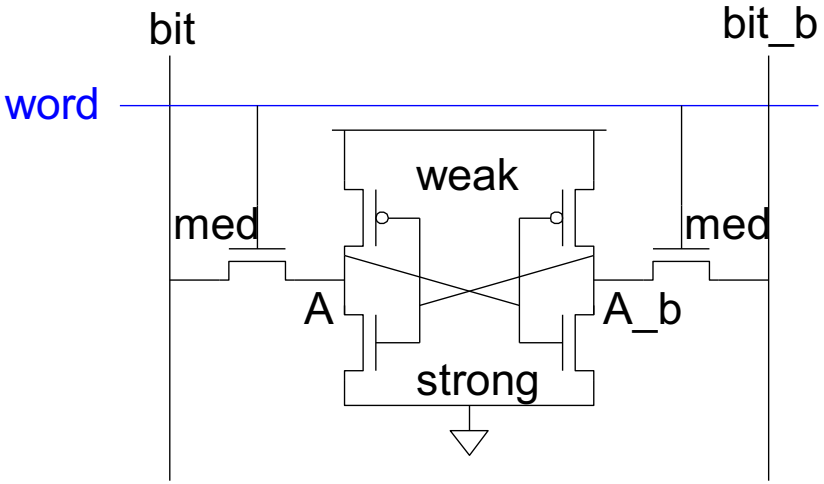
Write Operation : Cell Operation





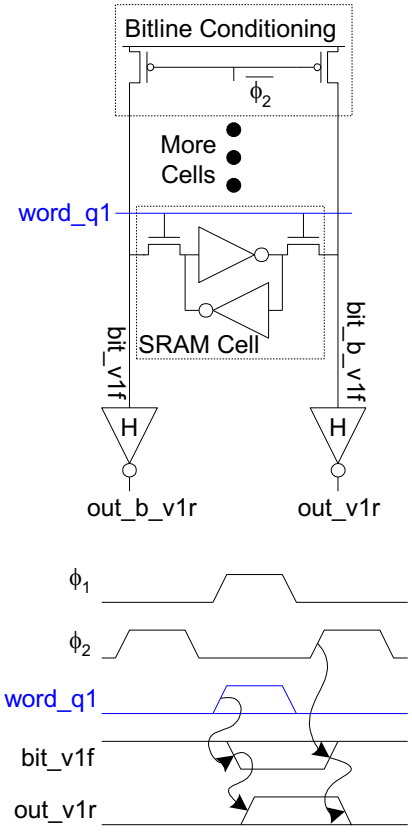
SRAM Sizing

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

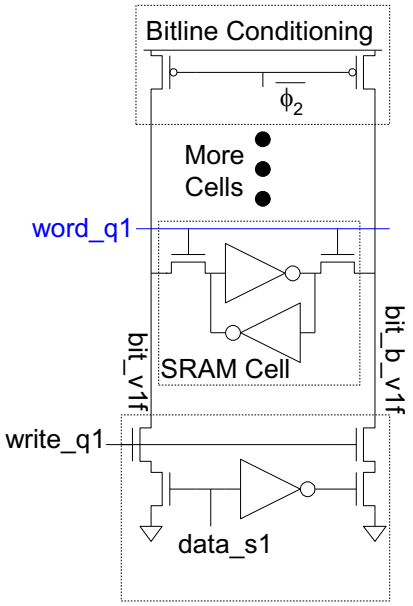


SRAM Full Column Example

Read

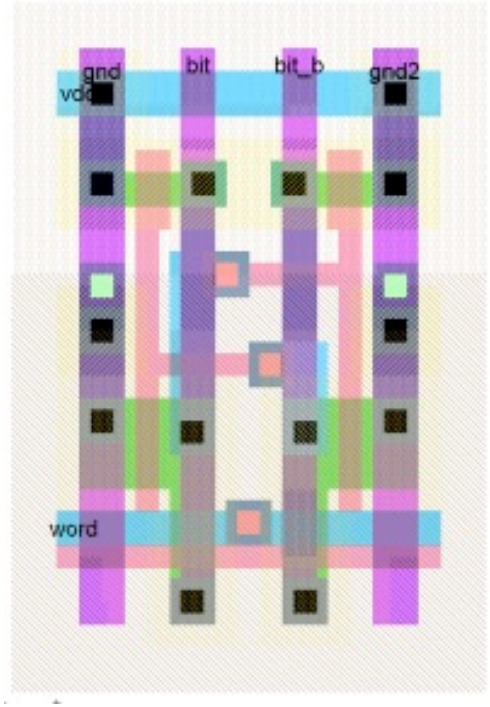


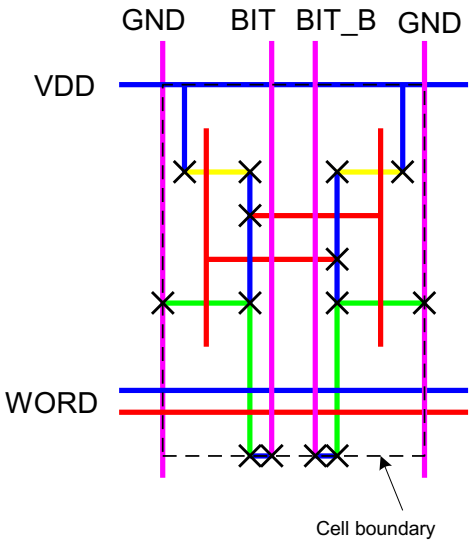
Write

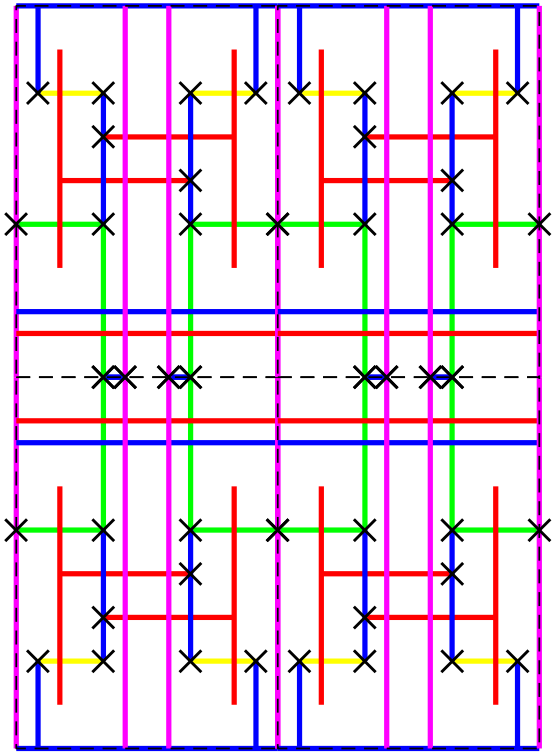


SRAM Layout

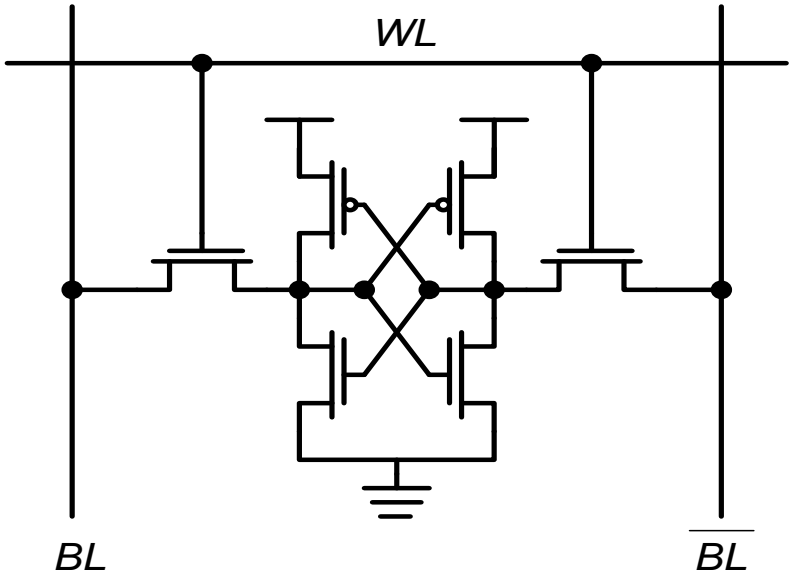
- Cell size is critical: 26 x 45 λ (even smaller in industry)
- Tile cells sharing V_{DD}, GND, bitline



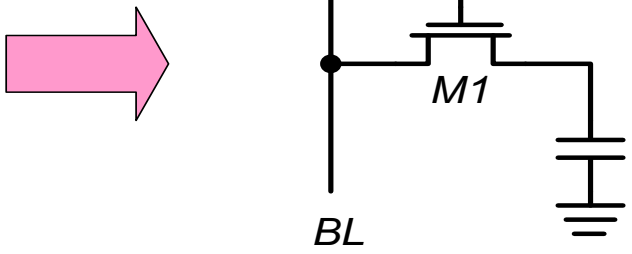




DRAM: why?



SRAM

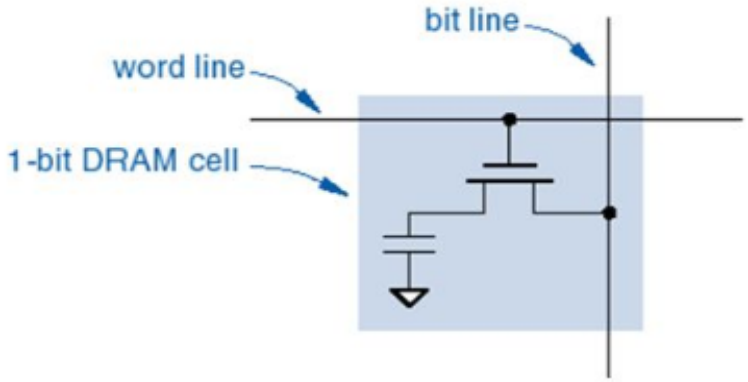


DRAM

- Static
- Data latch
- 4 or 6 Transistors
- Simple and fast

- Dynamic
- Refresh
- Charge sharing & S/A
- Small area → low cost

DRAM READ/WRITE



Read:

1. Precharge bit line to VDD/2
2. Take word line HIGH
3. Detect current flow of the cell(+-ΔV). Charge from capacitor fed via bit line to sense amplifier
4. Sense amplifiers compares capacitor voltage with reference value to determine 0 or 1

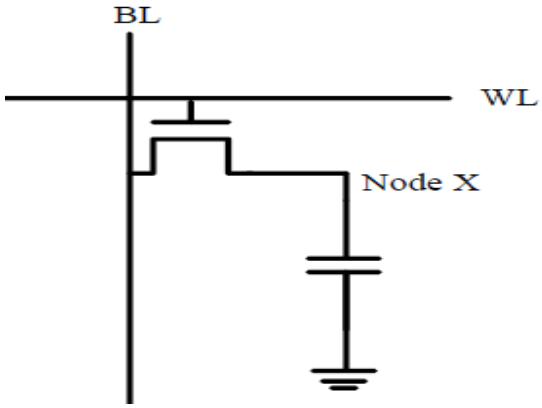
Note: Capacitor charge must be restored to complete the operation

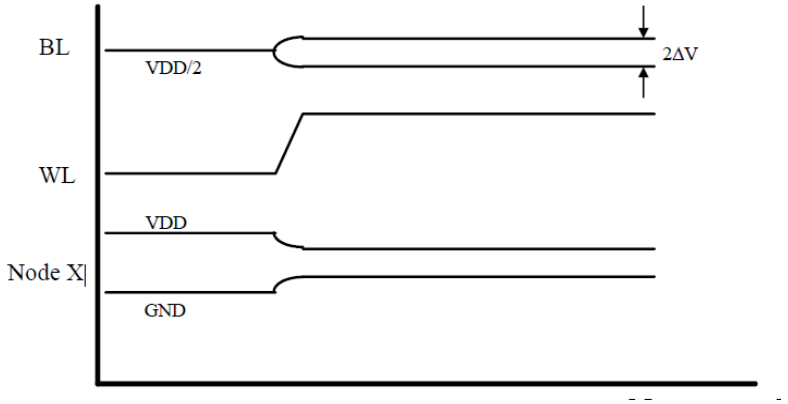
Write:

1. Take word line HIGH
2. Set the bit line LOW to store 0 or HIGH to store 1
3. Take word line LOW before changing bit line

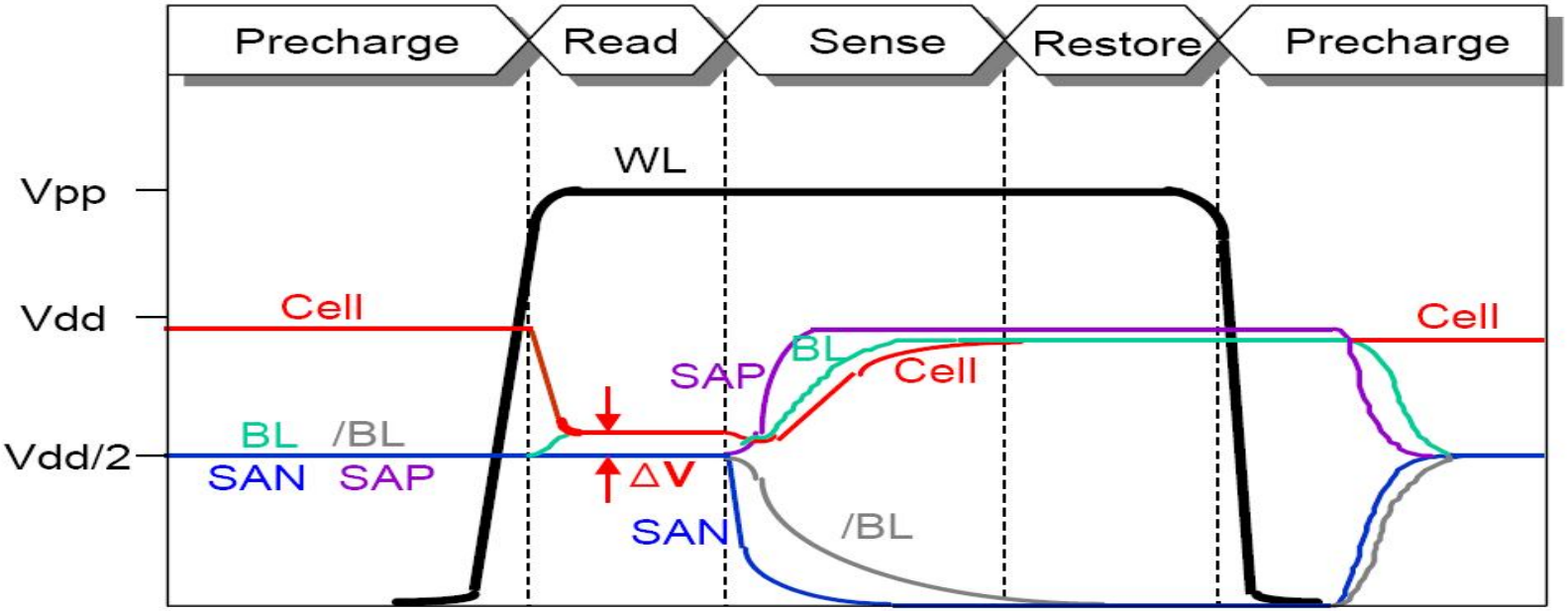
Note : The stored charge for a 1 will eventually leak off

DRAM : Charge Sharing

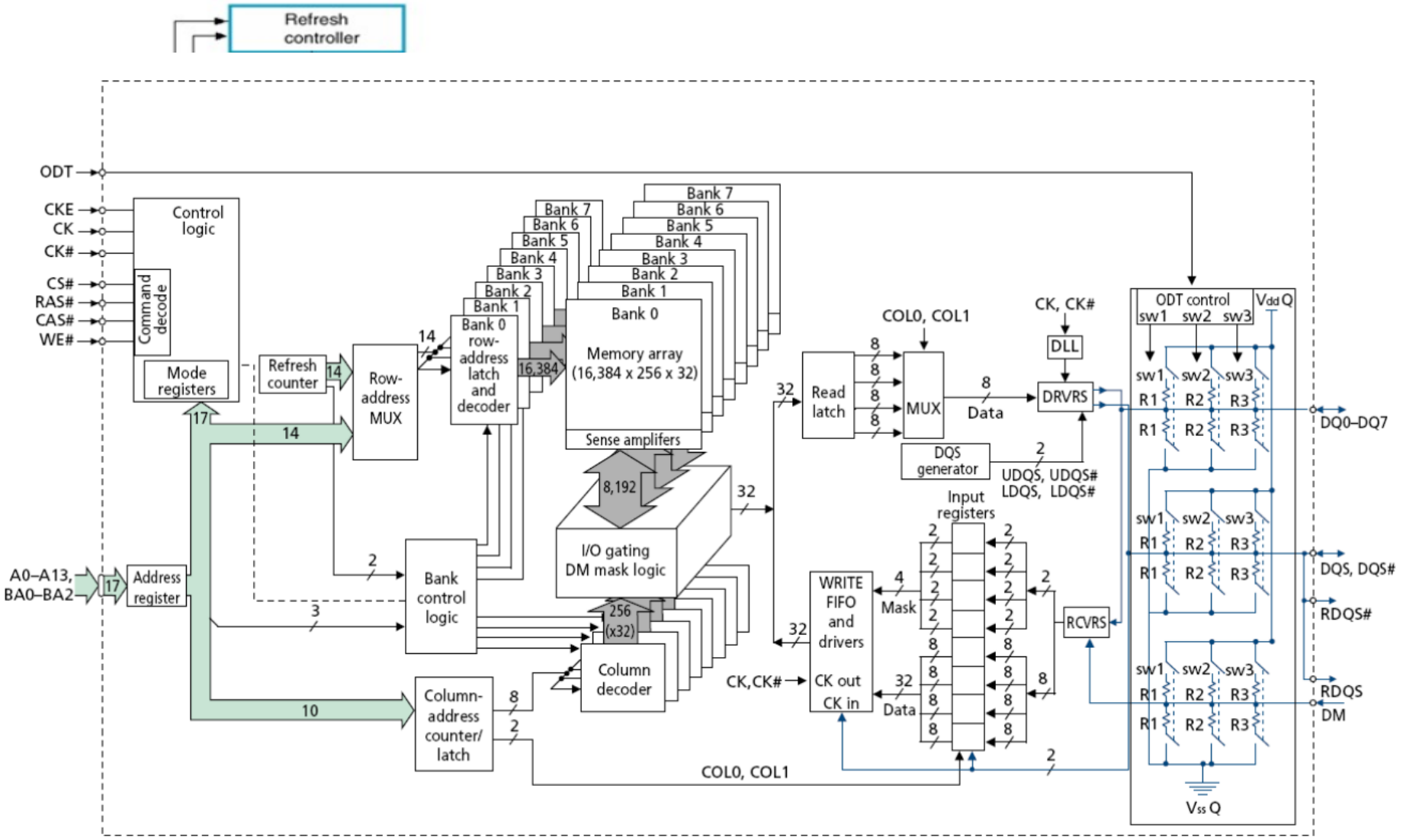




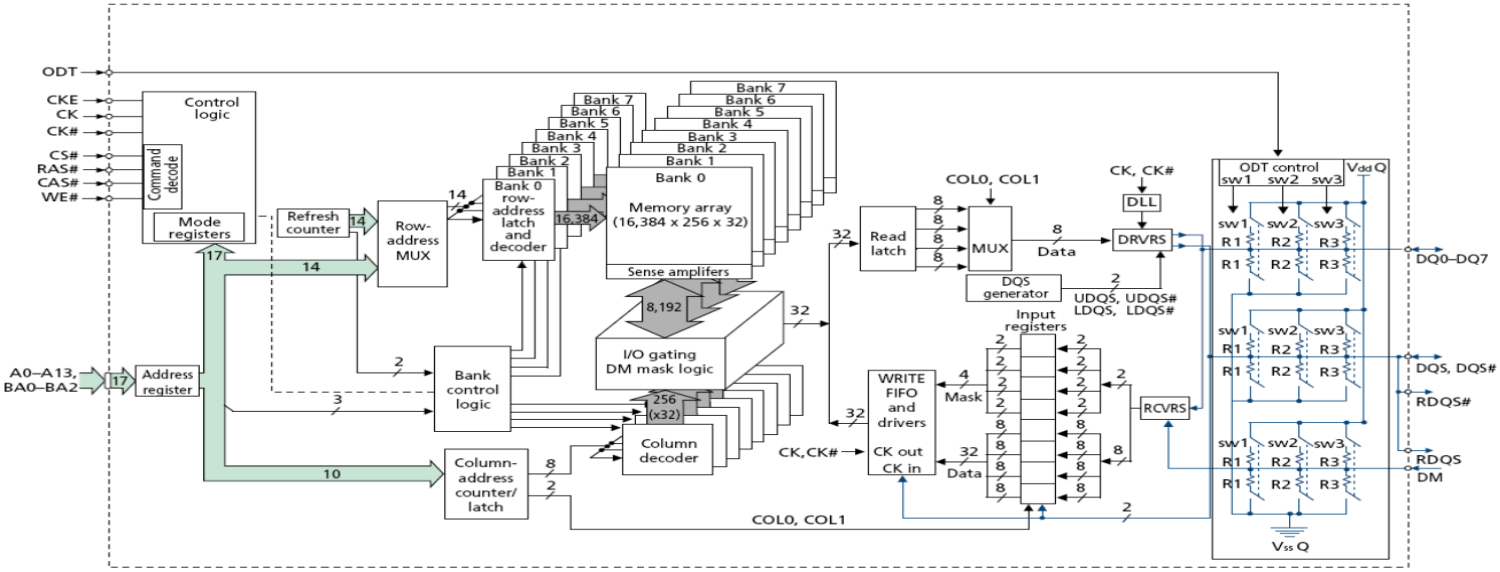
$$\Delta V = \frac{1}{2}\frac{V_{DD}}{1 + C_{BL} / C_S}$$



DRAM Architecture

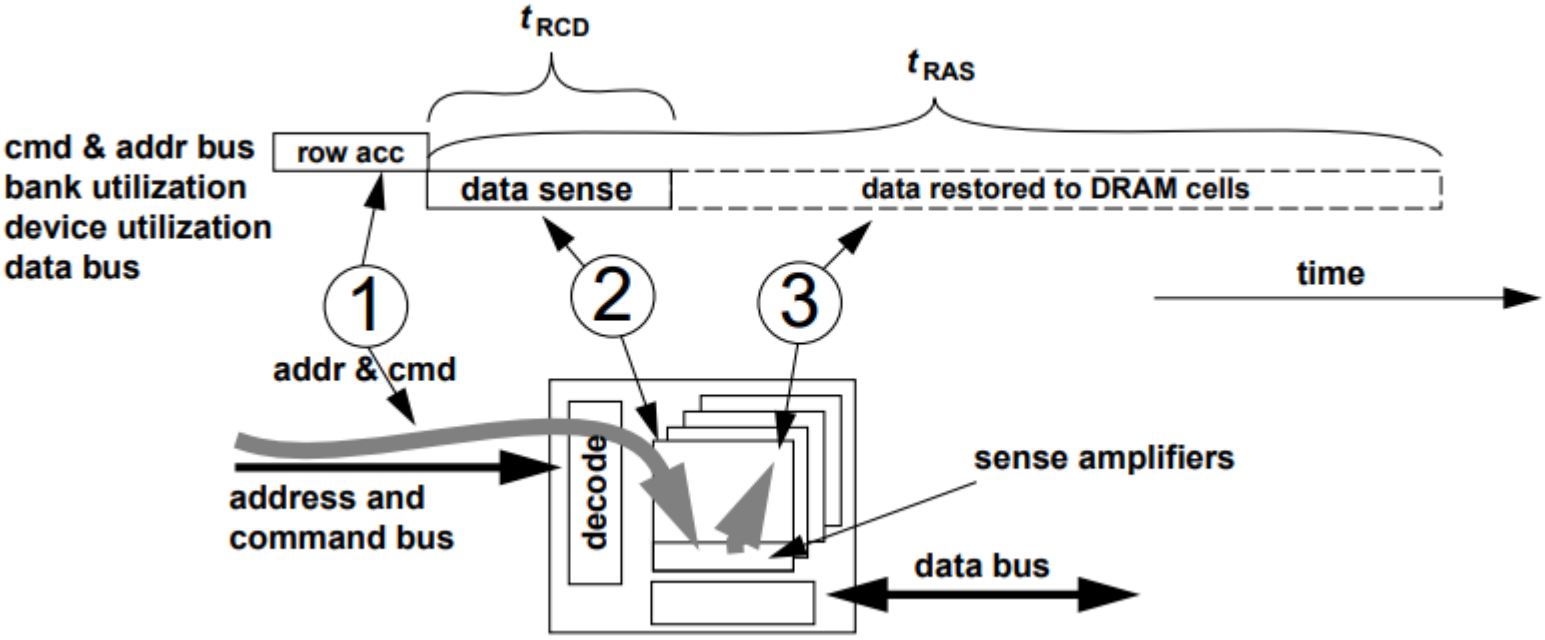


DRAM Support Circuitry



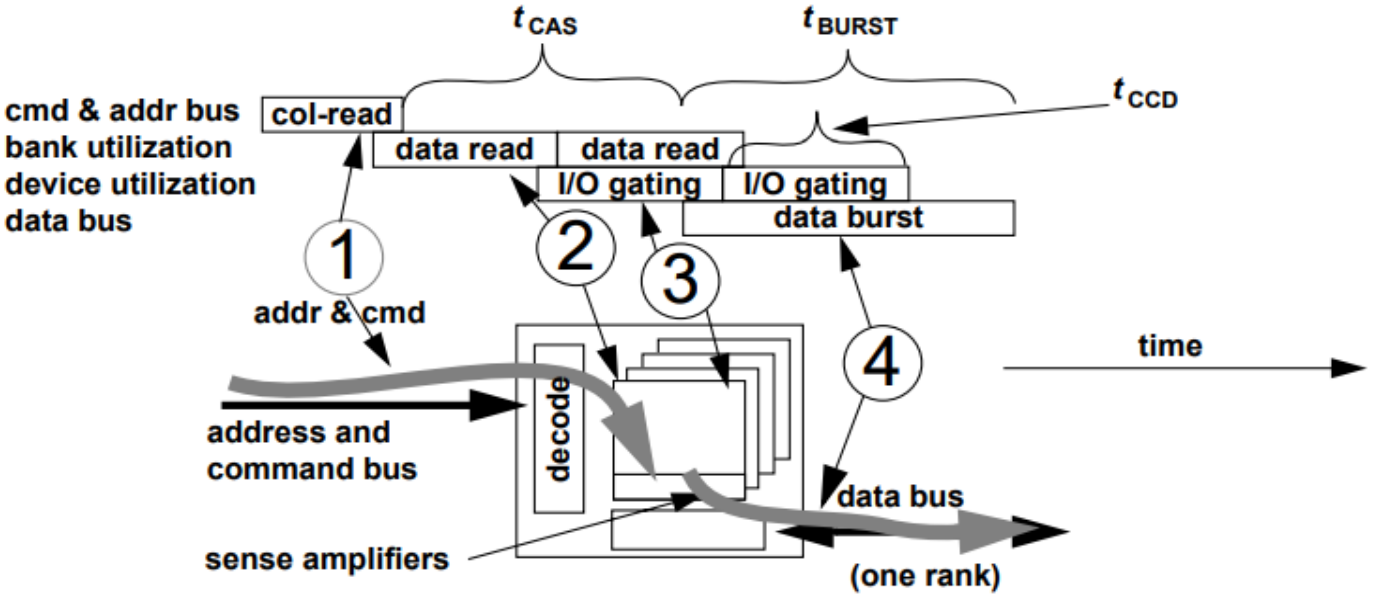
- **Sense amplifiers** to amplify the signal or charge detected on a memory cell.
- **Address logic** to select rows and columns.
- **Row Address Select ($\overline{RAS}$) and Column Address Select ($\overline{CAS}$) logic** to latch and resolve the row and column addresses and to initiate and terminate read and write operations.
- **Read and write circuitry** to store information in the memory's cells or read that which is stored there.
- **Internal counters or registers** to keep track of the refresh sequence, or to initiate refresh cycles as needed.
- **Output Enable logic** to prevent data from appearing at the outputs unless specifically desired

DRAM Access Steps



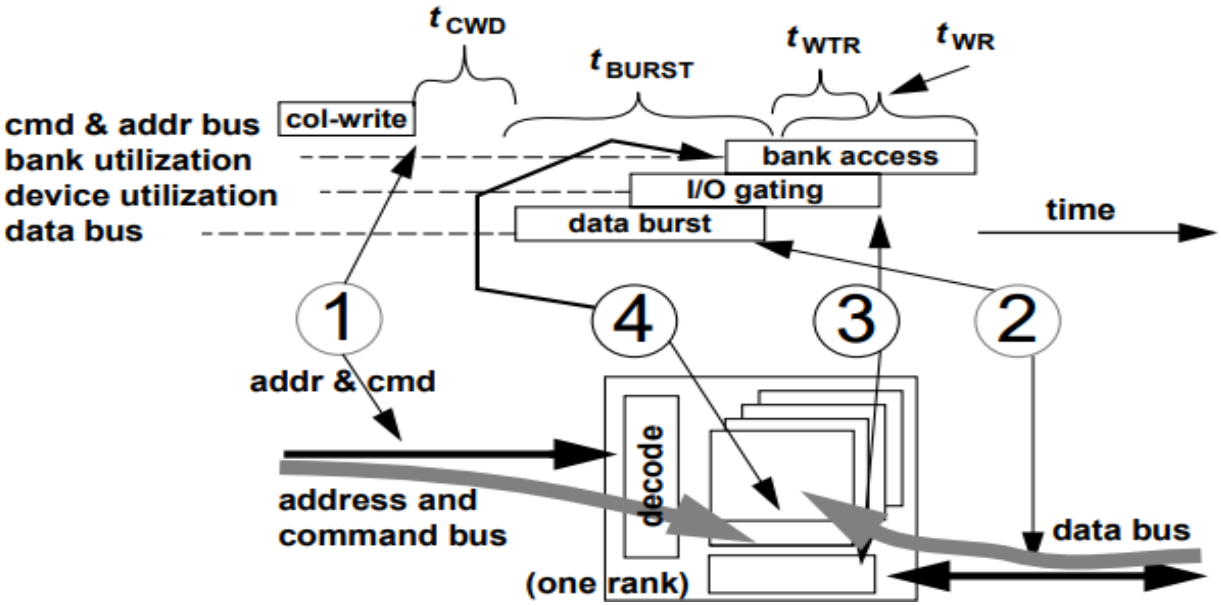
- **Row access command and timing**
  - $\boldsymbol{t_{RAS}}$: Row Access Strobe. The time interval between row access command and data restoration in a DRAM array. DRAM bank cannot be precharged until at least t_{RAS} time after the previous bank activation.
  - $\boldsymbol{t_{RCD}}$: Row to Column command Delay. The time interval between row access and data ready at sense amplifiers.

DRAM Access Steps



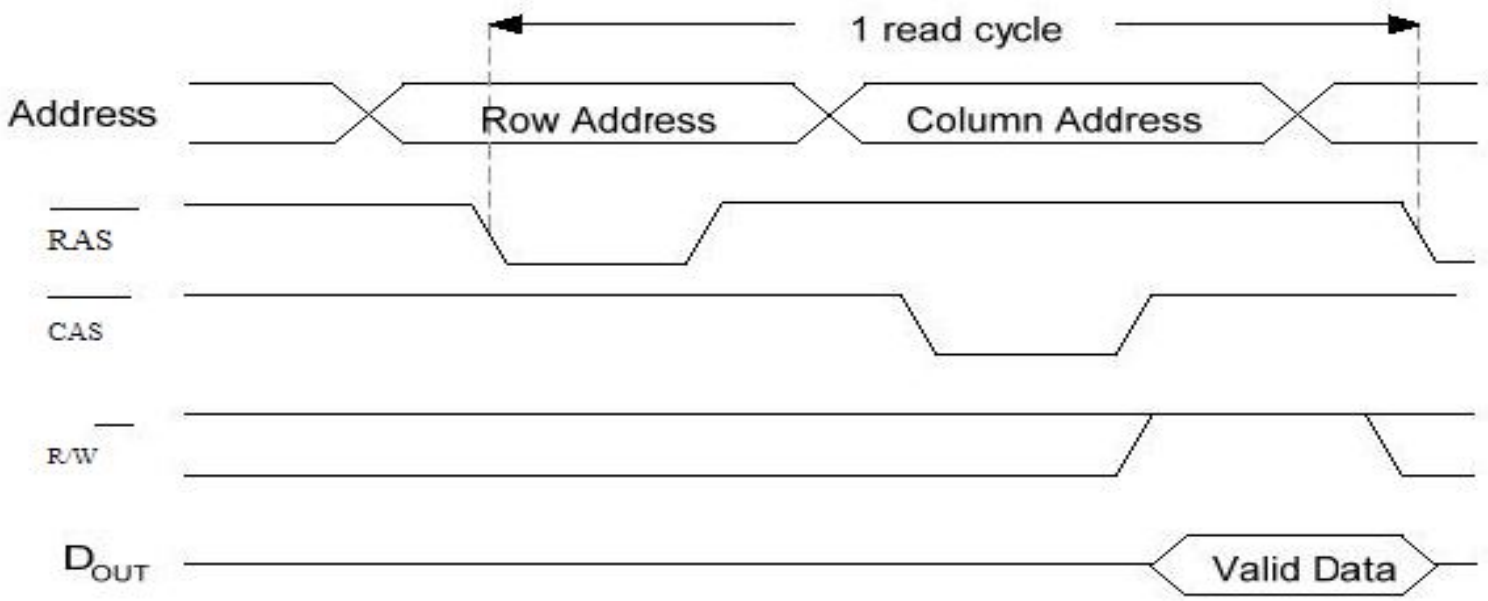
- **Column-read command and timing**
  - t_{CAS}: Column Access Strobe latency. The time interval between column access command and the start of data return by the DRAM device.
  - t_{BURST}: Data burst duration. The time period that data burst occupies on the data bus. Typically 4 or 8 beats of data. In DDR SDRAM, 4 beats of data occupy 2 full clock cycles.
  - t_{CCD}: Column-to-Column Delay. The minimum column command timing, determined by internal burst length.

DRAM Access Steps



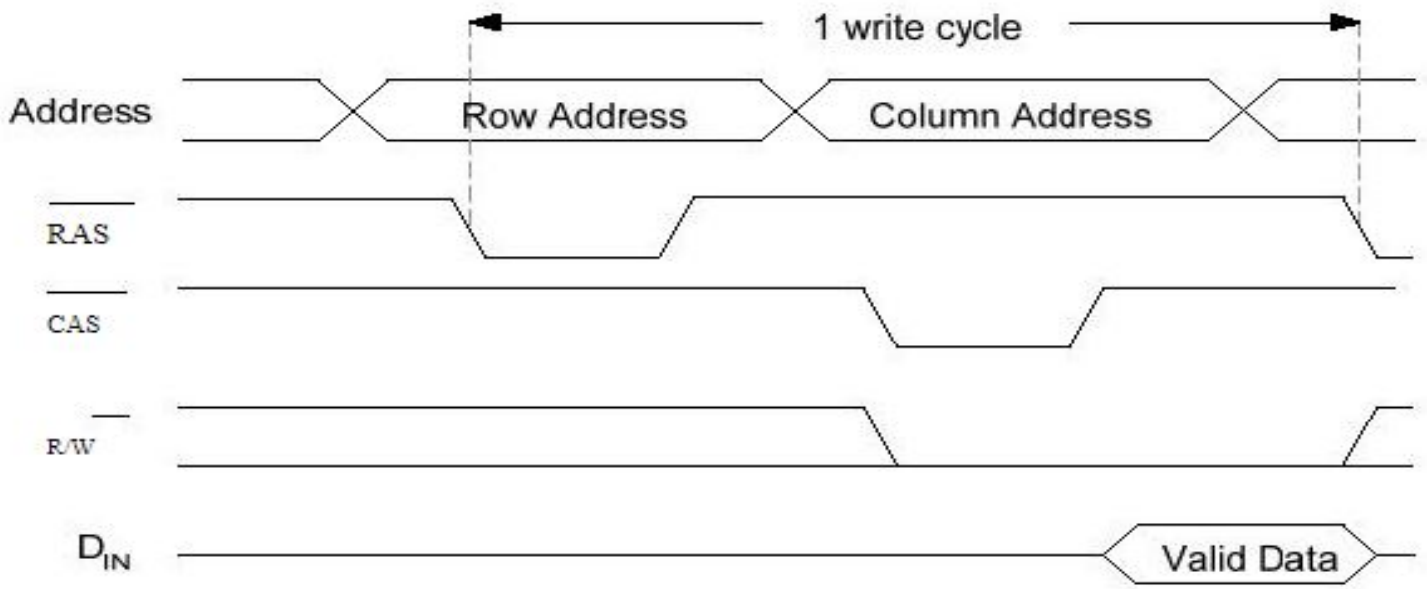
- **Column-write command and timing**
  - t_{CWD}: Column Write Delay. The time interval between issuance of the column-write command and placement of data on the data bus by the DRAM controller.
  - $\mathbf{t_{WR}}$**:** Write Recovery time. The minimum time interval between the end of a write data burst and the start of a precharge command. Allows sense amplifiers to restore data to cells.
  - t_{WTR}**:** Write To Read delay time. The minimum time interval between the end of a write data burst and the start of a column-read command. Allow I/O gating to overdrive sense amplifiers before read command starts.

Read Control Signals



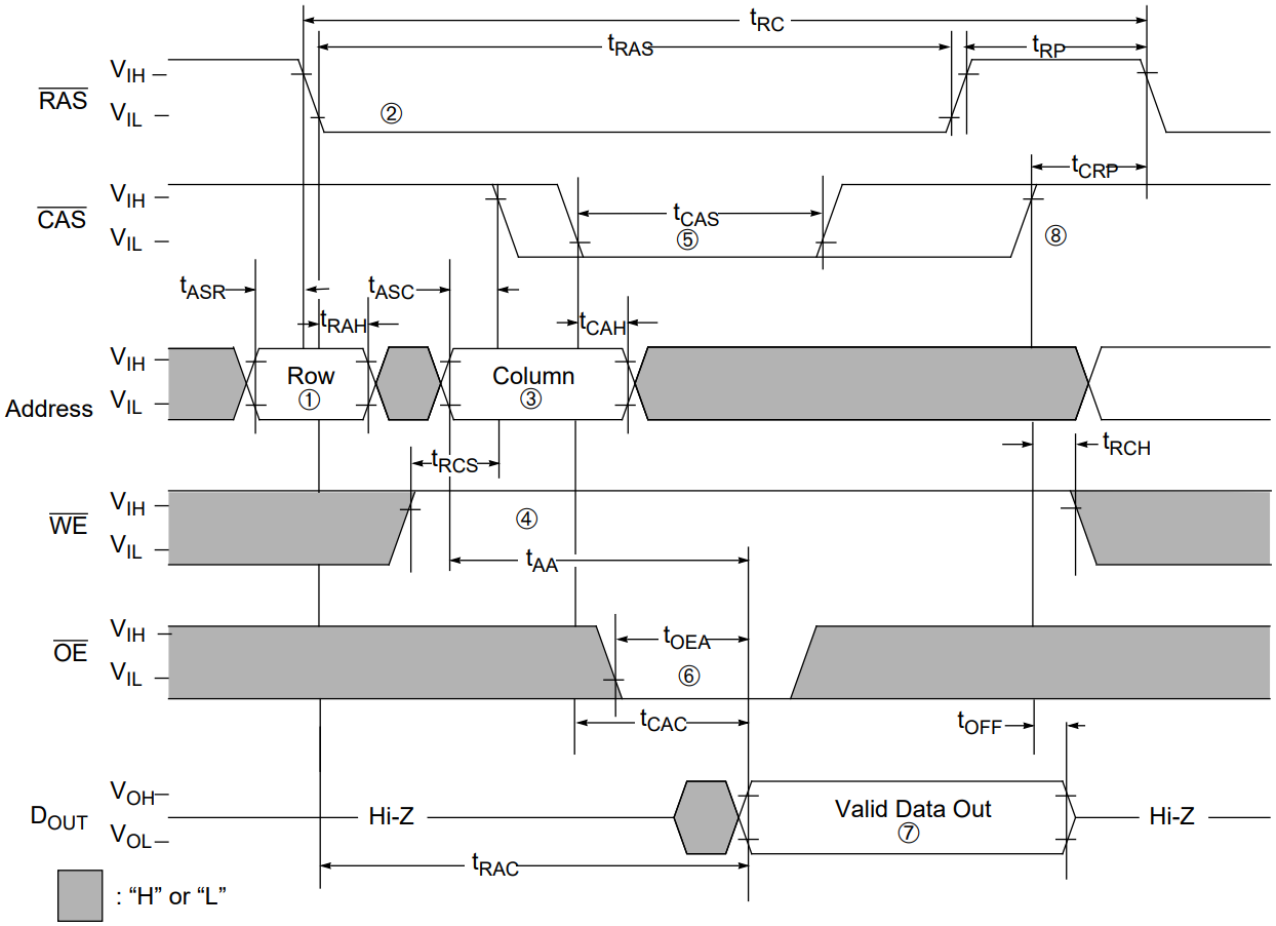
- RAS and CAS signals are active one after the other
- The multiplexed row and column addresses are applied at the multiplexed address input line
- The R/W signal activated to read data

Write Control Signals



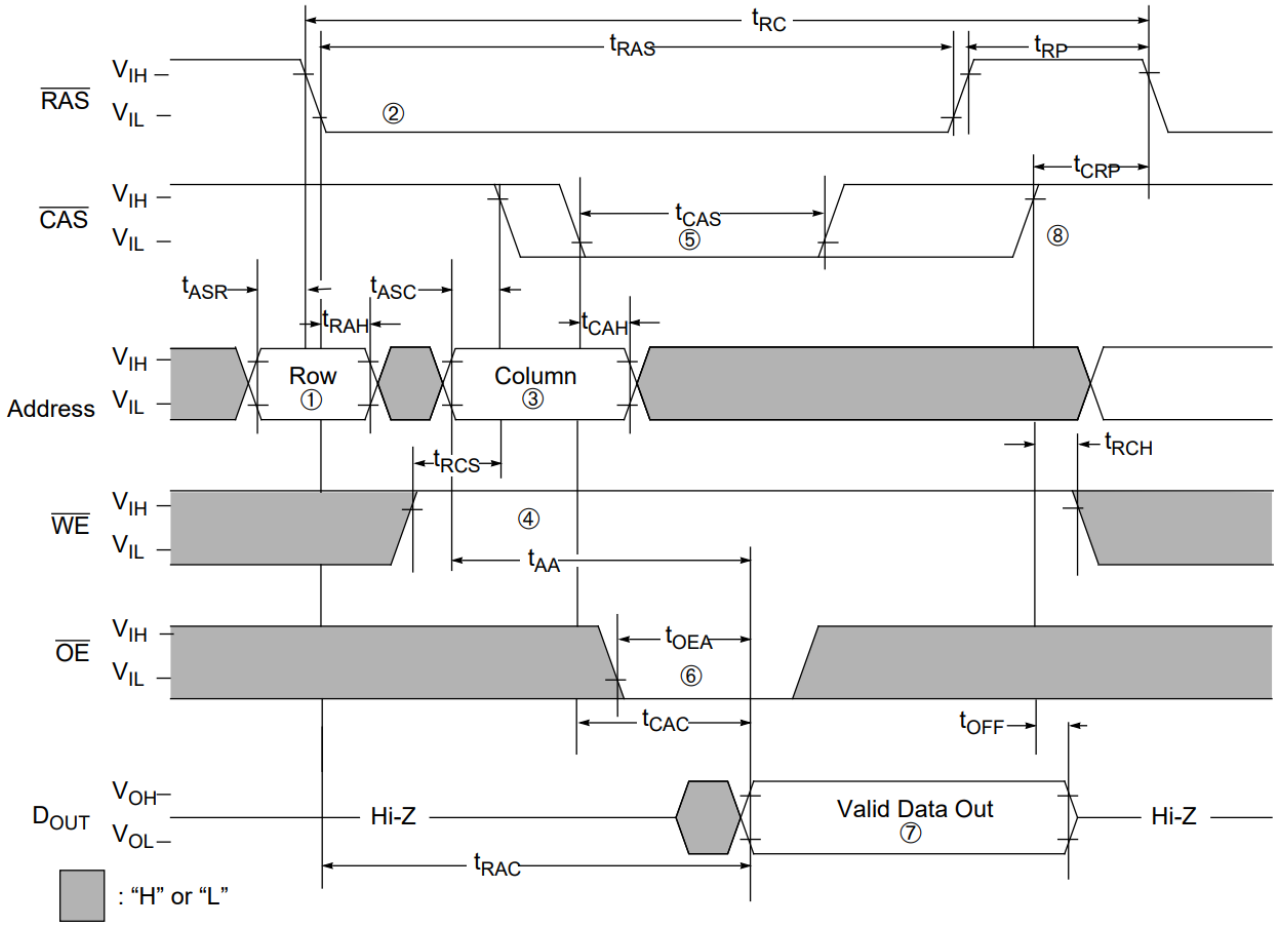
- Write cycle is similar to the read cycle:
- RAS and CAS signals are active one after the other
- The multiplexed row and column addresses are applied at the multiplexed address input line
- The write signal is activated allowing data placed at the DIN data line to be stored in the selected memory cell

Read Timing Waveforms



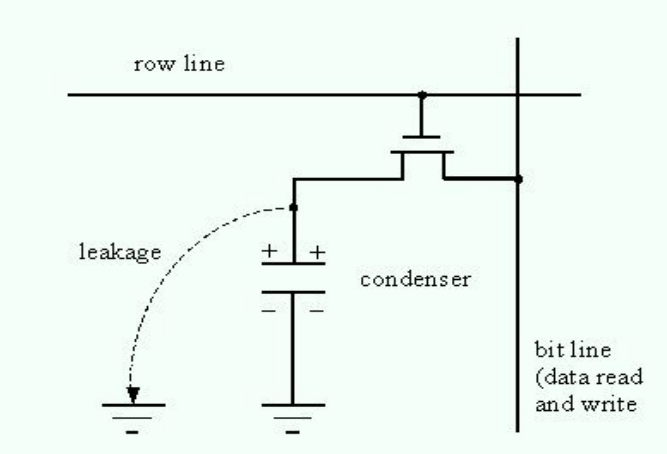
1. The row address must be applied to the address input pins on the memory device for the prescribed amount of time before $\overline{RAS}$ goes low (t_{ASR}) and held (t_{RAH}) after $\overline{RAS}$ goes low.
2. $\overline{RAS}$ must go from high to low and remain low (t_{RAS}).
3. A column address must be applied to the address input pins on the memory device for the prescribed amount of time (t_{ASC}) and held (t_{CAH}) after $\overline{CAS}$ goes low.
4. WE must be set high for a read operation to occur prior (t_{RCS}) to the transition of $\overline{CAS}$, and remain high (t_{RCH}) after the transition of $\overline{CAS}$.
5. $\overline{CAS}$ must switch from high to low and remain low (t_{CAS}).
6. OE goes low within the prescribed window of time. Cycling OE is optional; it may be tied low, if desired.
7. Data appears at the data output pins of the memory device. The time at which the data appears depends on when $\overline{RAS}$ (t_{RAC}), $\overline{CAS}$ (t_{CAC}), and OE (t_{OEA}) went low, and when the address is supplied (t_{AA})
8. Before the read cycle can be considered complete, $\overline{CAS}$ and $\overline{RAS}$ must return to their inactive states (t_{CRF}, t_{RP}).

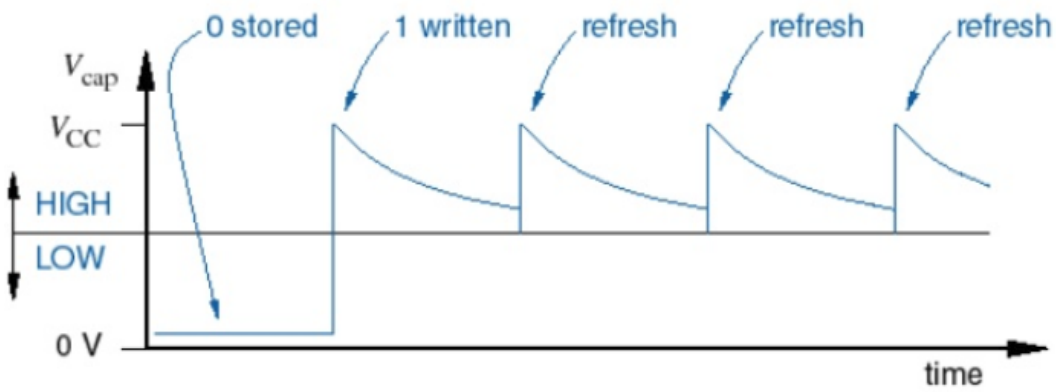
Write Timing Waveforms



1. The row address must be applied to the address input pins on the memory device for the prescribed amount of time before $\overline{RAS}$ goes low (t_{ASR}) and held (t_{RAH}) after $\overline{RAS}$ goes low.
2. $\overline{RAS}$ must go from high to low and remain low (t_{RAS}).
3. A column address must be applied to the address input pins on the memory device for the prescribed amount of time (t_{ASC}) and held (t_{CAH}) after $\overline{CAS}$ goes low.
4. WE must be set high for a read operation to occur prior (t_{RCS}) to the transition of $\overline{CAS}$, and remain high (t_{RCH}) after the transition of $\overline{CAS}$.
5. $\overline{CAS}$ must switch from high to low and remain low (t_{CAS}).
6. OE goes low within the prescribed window of time. Cycling OE is optional; it may be tied low, if desired.
7. Data appears at the data output pins of the memory device. The time at which the data appears depends on when $\overline{RAS}$ (t_{RAC}), $\overline{CAS}$ (t_{CAC}), and OE (t_{OEA}) went low, and when the address is supplied (t_{AA})
8. Before the read cycle can be considered complete, $\overline{CAS}$ and $\overline{RAS}$ must return to their inactive states (t_{CRF}, t_{RP}).

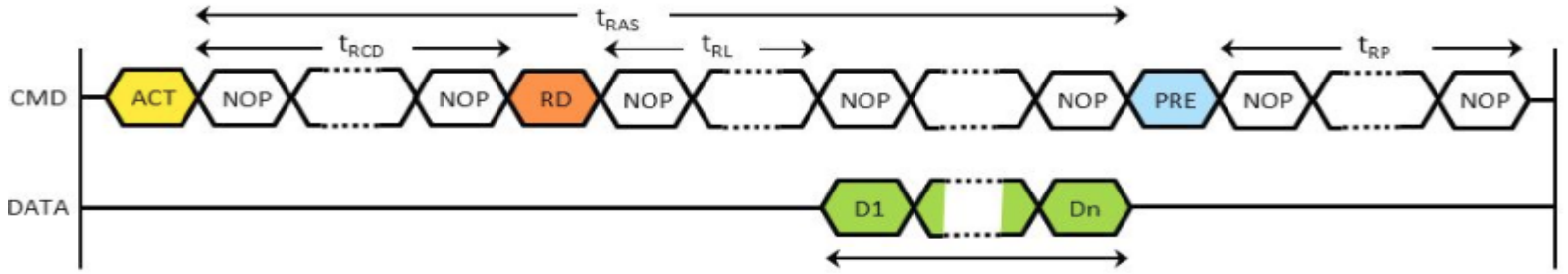
DRAM Refreshing





- Refresh is required in DRAM due to the leakage current
  - ✓ junction leakage exponential with temp
  - ✓ Decreases noise margin, destroys information
- Retention time changes by cells capacitance, temperature and etc. these factors must been considered setting Refresh time.
- Usually DRAM refresh work by row, so refresh time could considered as row cycle time.
- All columns in a selected row are refreshed when read
  - Count through all row addresses once per 3 msec (for example). (no write possible then)
- Requires additional refresh counter and I/O control

DRAM Timing Contraints



t_{CCD}= Time between column commands

t_{WTR}= Write to read delay (bus turn around time)

t_{CAS}= Time between column command and data out

t_{WR}= Time from end of last write to PRECHARGE

t_{FAW}= Four ACTIVATE window (limits current surge)

Maximum number of ACTIVATEs in this window is limited to four

t_{RC}= **t_{RAS}**+ **t_{RP}**= Row "cycle" time

Minimum time between accesses to different rows

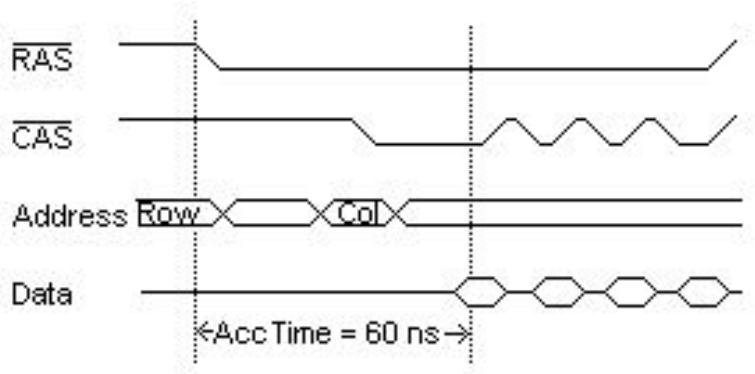
t_{RCD}= Row to Column command delay

Time taken by the charge stored in the capacitor cells to reach the sense amps

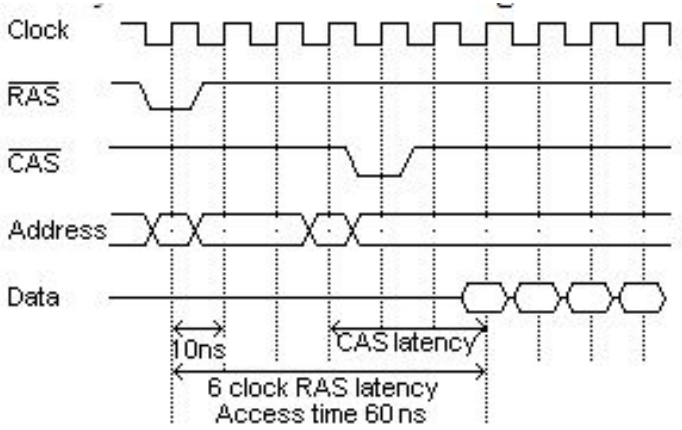
t_{RAS}= Time between RAS and data restoration in DRAM array (minimum time a row must be open)

t_{RP}= Time to precharge DRAM array

Asynchronous and Synchronous DRAMs



Asynchronous DRAM timing



Synchronous DRAM timing

- Asynchronous DRAMs
  - Read and write operations are triggered by a rising or falling signal
  - Can occur at any time
- Synchronous DRAMs
  - Read and write operations are synchronized to a clock signal
  - The operation begins at expected times

SDR and DDR

- **SDR SDRAM**
  - ✓ SDRAM operates only at rising or falling edge. Operates once in one clock cycle.

- **DDR SDRAM (Double Data Rate)**
  - ✓☐SDRAM operates at rising and falling edge both. Operates double in on clock cycle.
  - ✓☐Faster twice than SDR in same clock speed.
  - ✓☐Nowadays, commonly used.

DDR Generations

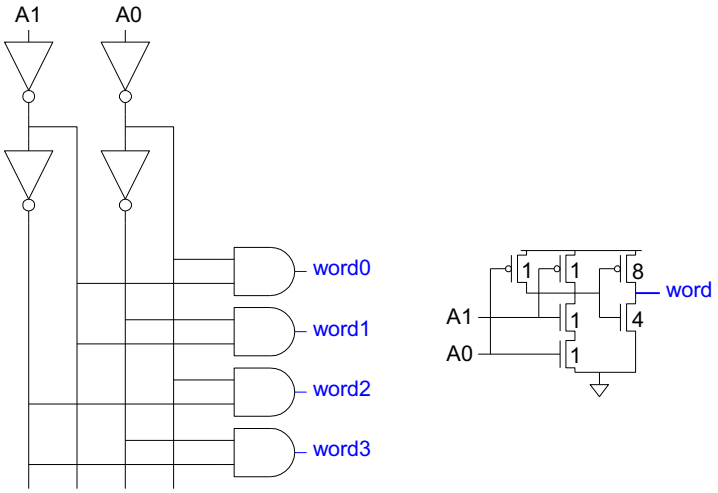
Type of DD RAM	Launched year	Voltage (V)	I/O bus clock (MHz)	Memory clock (MHz)	Data rate (MT/s)	Peak transfer rate (MB/s)
DDR1- 400	2000	2.5	200	200	400	3200
DDR3-800	2003	1.5/1.35	400	100	800	6400
DDR3-2133	2007	1.5/1.35	1066.6	266.6	2133.33	17000
DDR4-1600	2013	1.2	800	200	1600	12800
DDR4-3200	2013	1.2	1600	400	3200	25600

- **DDR1**: transfer data on the rising and falling edges of the clock signal
- **DDR2**: same internal clock speed, but external data bus is twice as fast as DDR, which is achieved by improved bus signal. Double in size of prefetch buffer(compare to DDR1).
- **DDR3**: reduces 40% power consumption by lowering operating voltage. Double in size of prefetch buffer(compare to DDR2). Automatic Self-Refresh(ASR) and Self-Refresh Temperature(SRT) are added.
- **DDR4**: lower operating voltage (1.2V) compares to DDR3. DDR4 can process 4 data in one clock cycle, due to four new Bank Groups technology. DBI (Data Bus Inversion), CRC (Cyclic Redundancy Check) and CA parity function are added
- **DDR5**: reduced power consumption, double the memory bandwidth and capacity. ~2020.

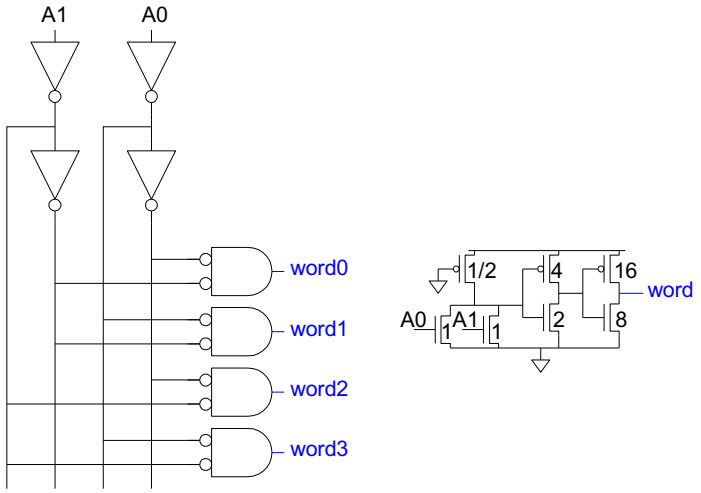
Decoders

- 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

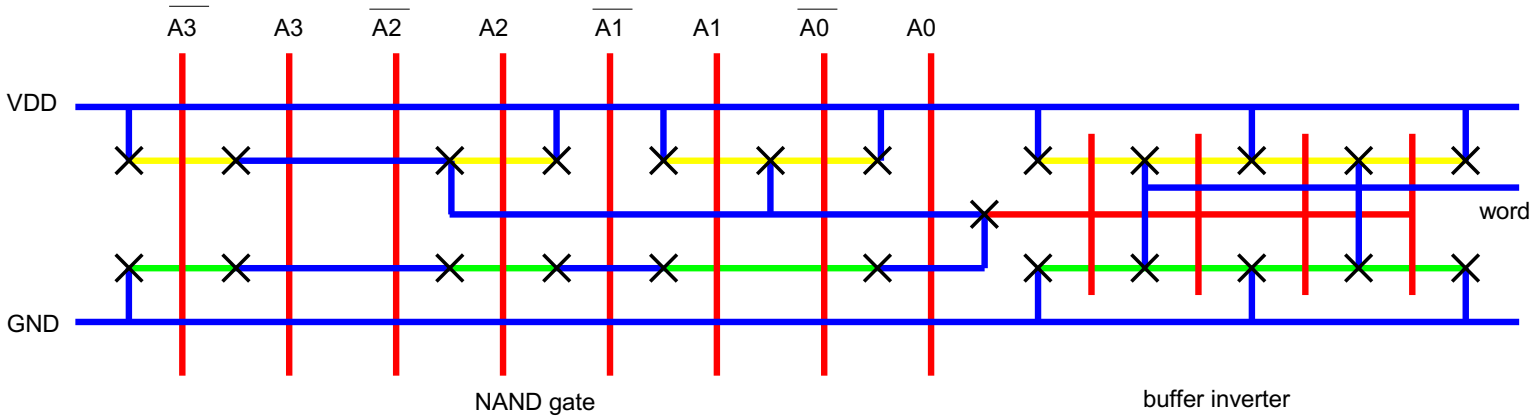


Pseudo-nMOS



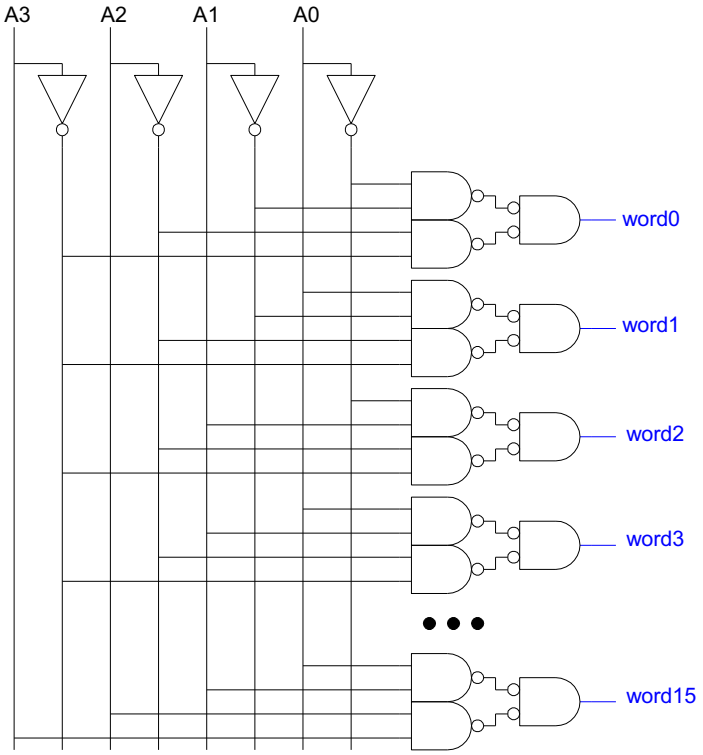
Decoder Layout

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



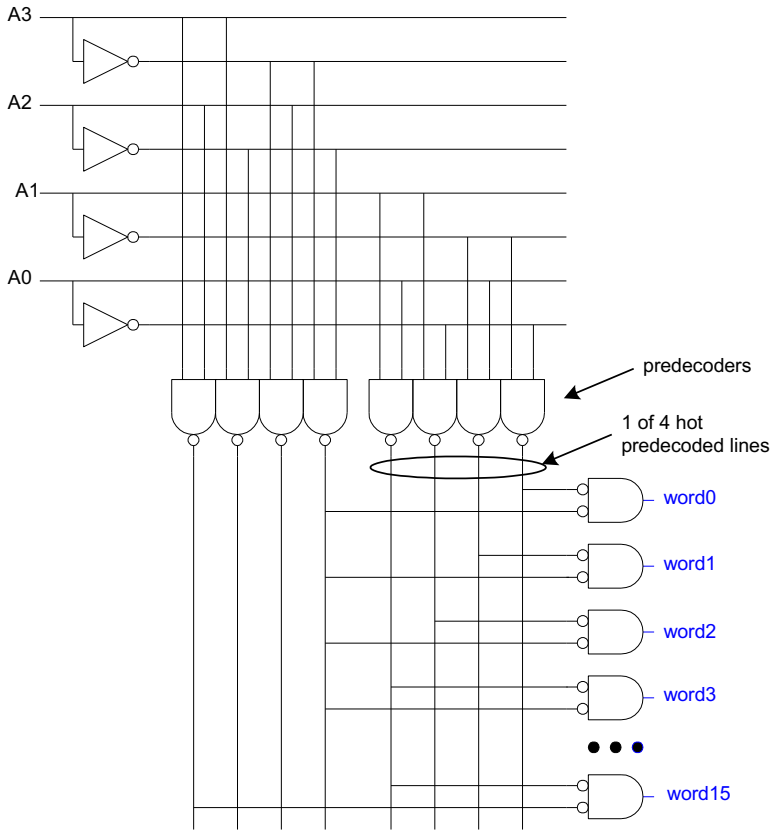
Large Decoders

- 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

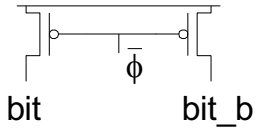


Column Circuitry

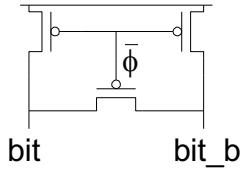
- Some circuitry is required for each column
  - Bitline conditioning
  - Sense amplifiers
  - Column multiplexing

Bitline Conditioning

- Precharge bitlines high before reads



- Equalize bitlines to minimize voltage difference when using sense amplifiers

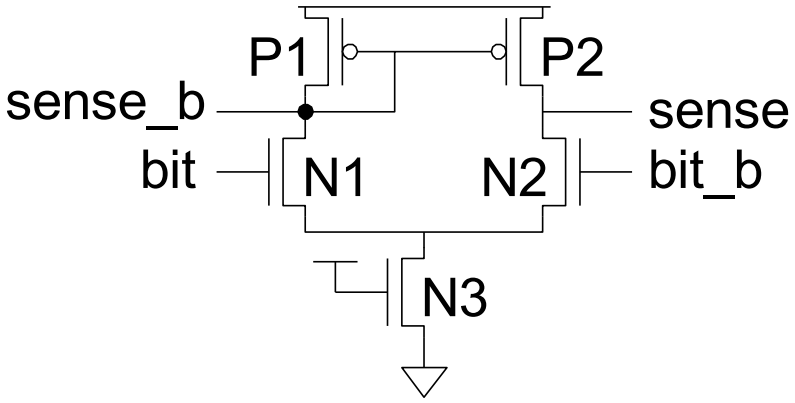


Sense Amplifiers

- 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)

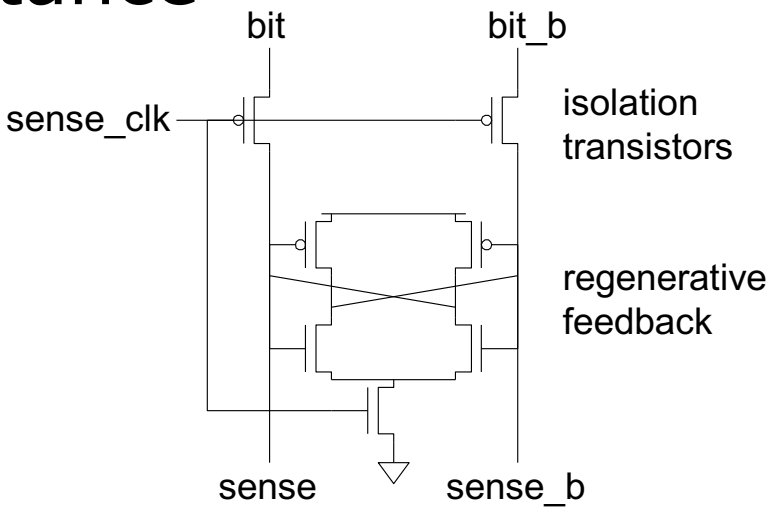
Differential Pair Amp

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



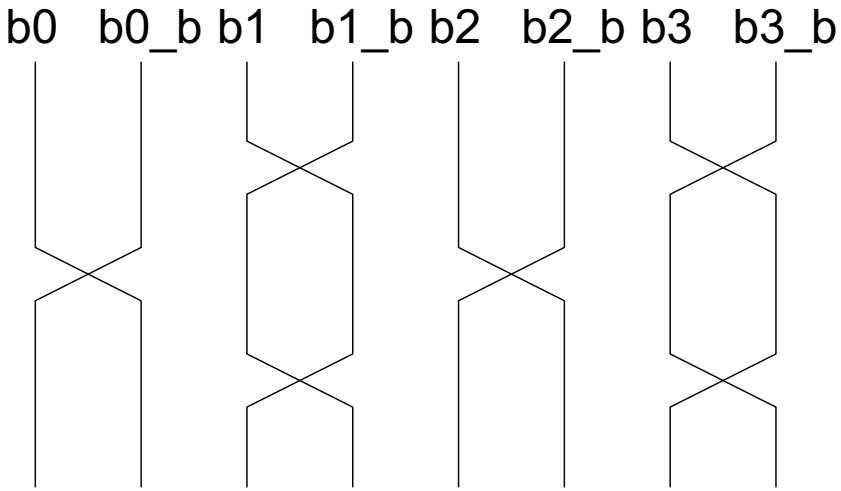
Clocked Sense Amp

- 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

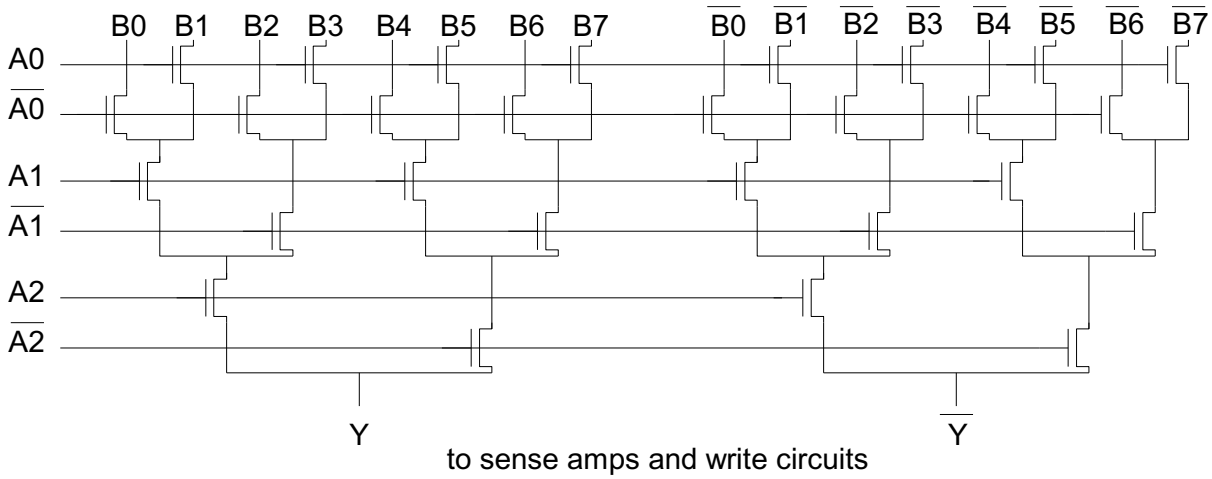


Column Multiplexing

- 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

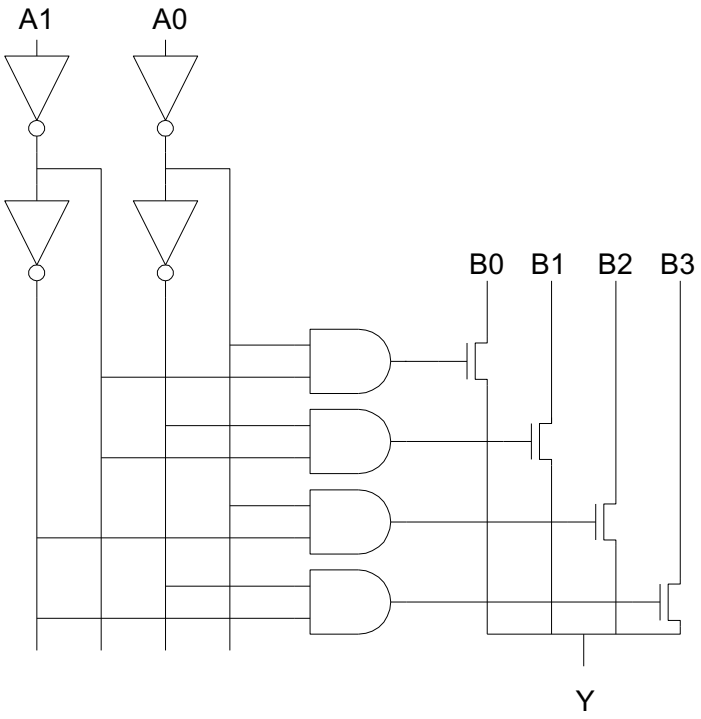
Tree Decoder Mux

- 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

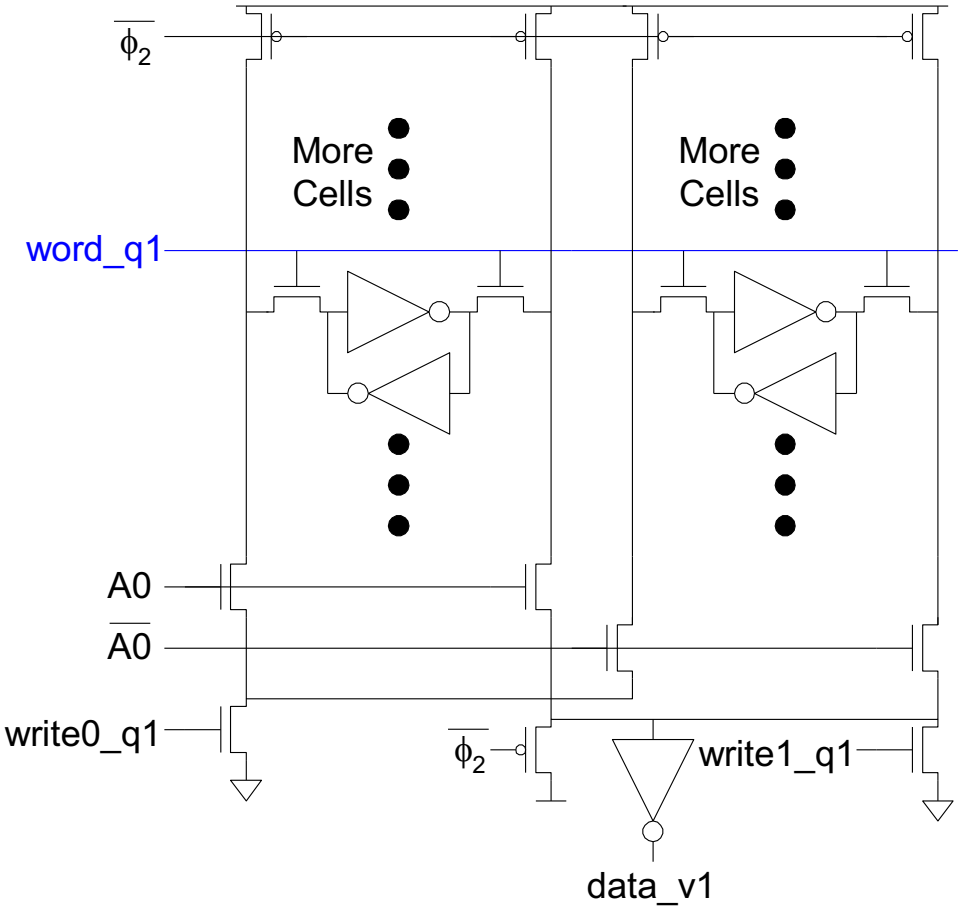


Single Pass-Gate Mux

- Or eliminate series transistors with separate decoder



Ex: 2-way Muxed SRAM

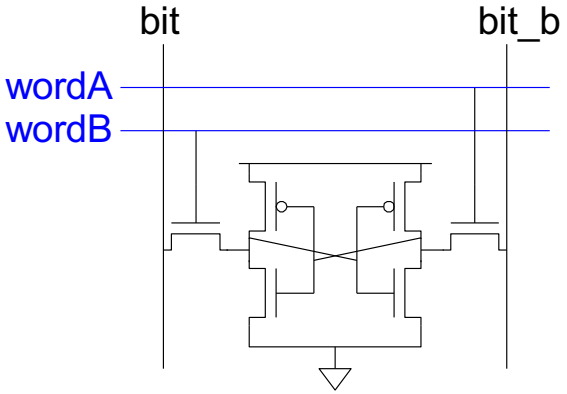


Multiple Ports

- 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

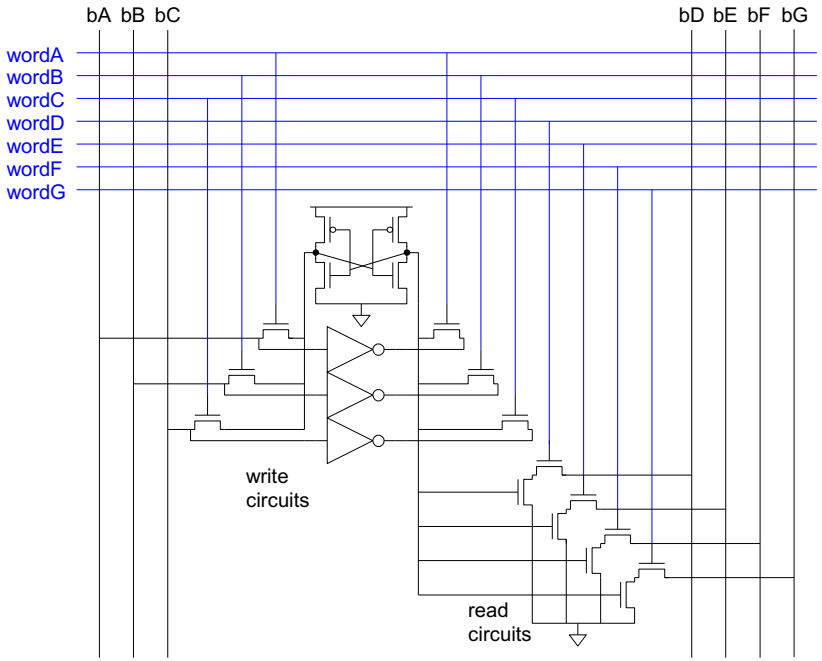
- Simple dual-ported SRAM
  - Two independent single-ended reads
  - Or one differential write



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

Multi-Ported SRAM

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

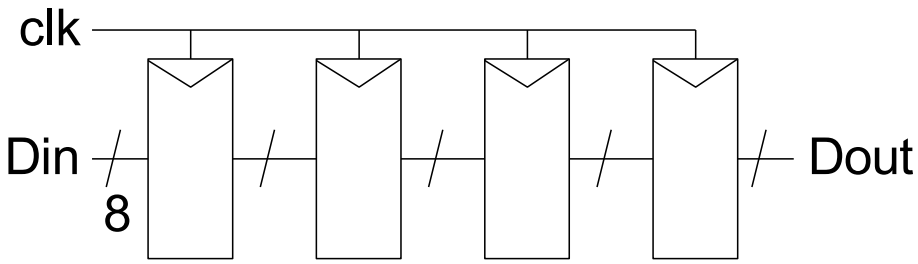


Serial Access Memories

- 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)

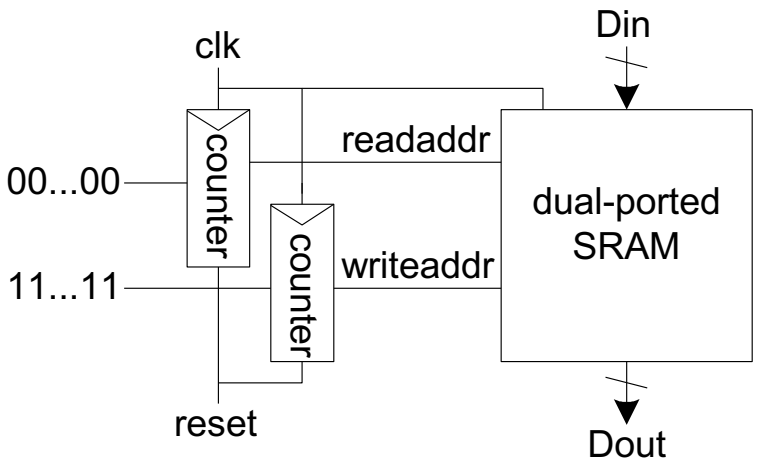
Shift Register

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



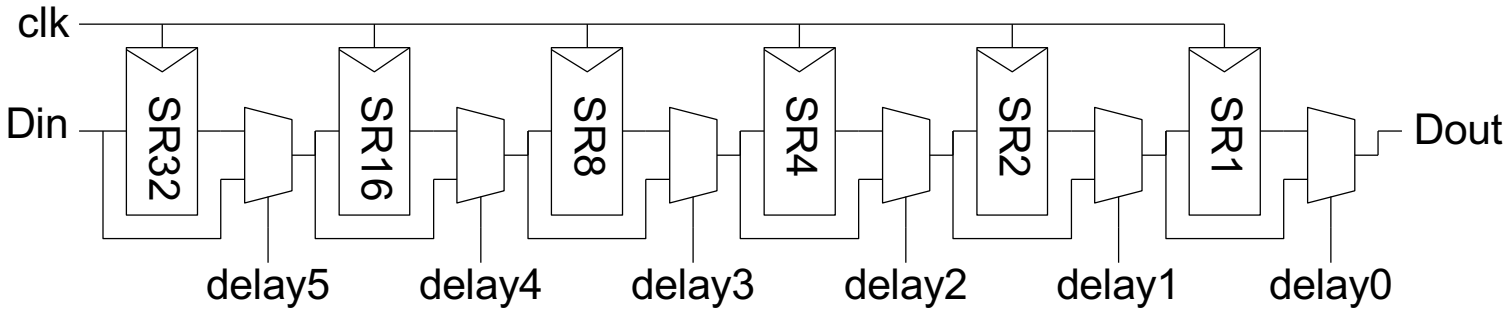
Denser Shift Registers

- 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



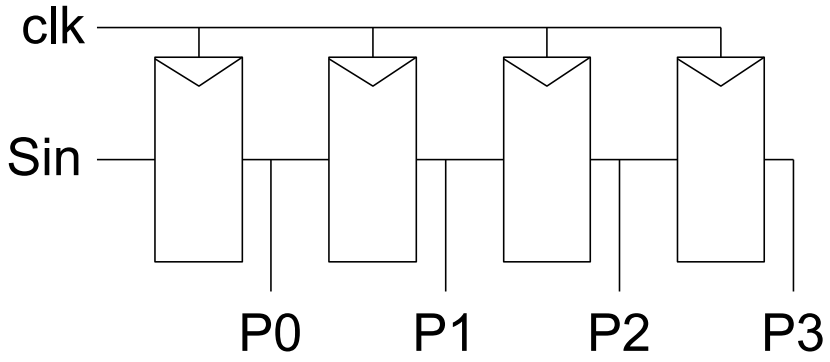
Tapped Delay Line

- 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



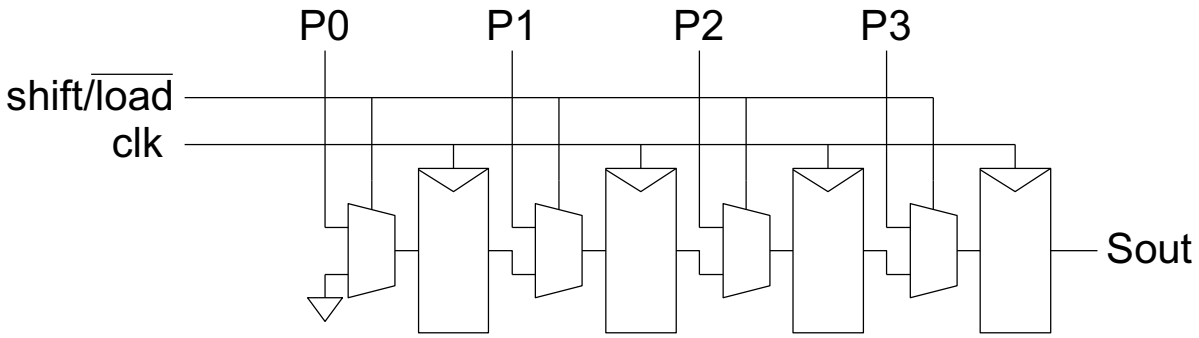
Serial In Parallel Out

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



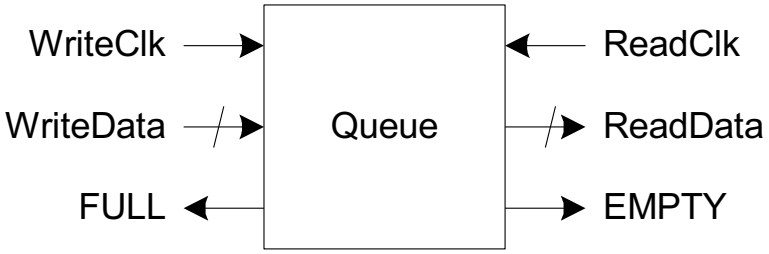
Parallel In Serial Out

- 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

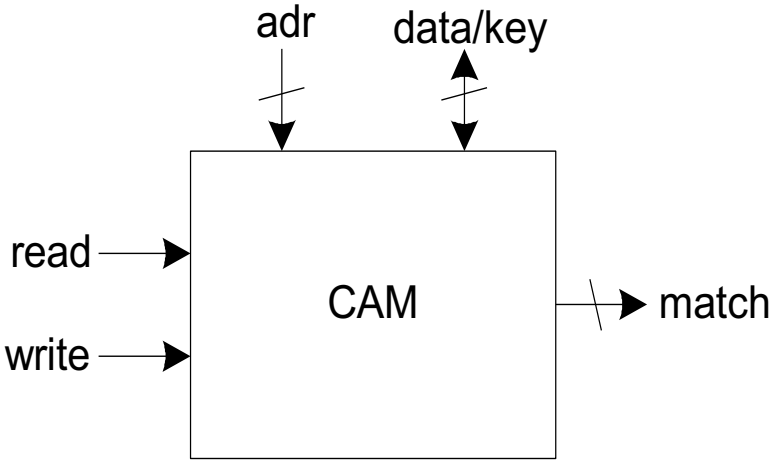
- *First In First Out* (FIFO)
  - Initialize read and write pointers to first element
  - Queue is EMPTY
  - On write, increment write pointer
  - If write almost catches read, Queue is FULL
  - On read, increment read pointer
- *Last In First Out* (LIFO)
  - Also called a *stack*
  - Use a single *stack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

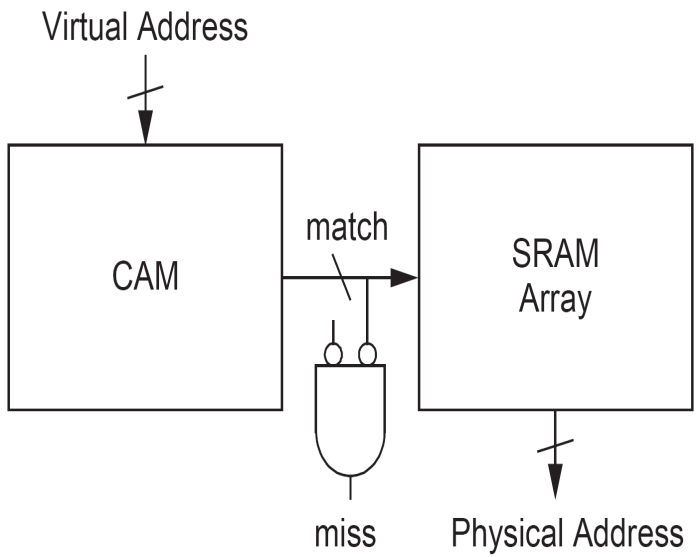
Part II

- Content-Addressable Memories
- Read-Only Memories
- Programmable Logic Arrays

CAMs

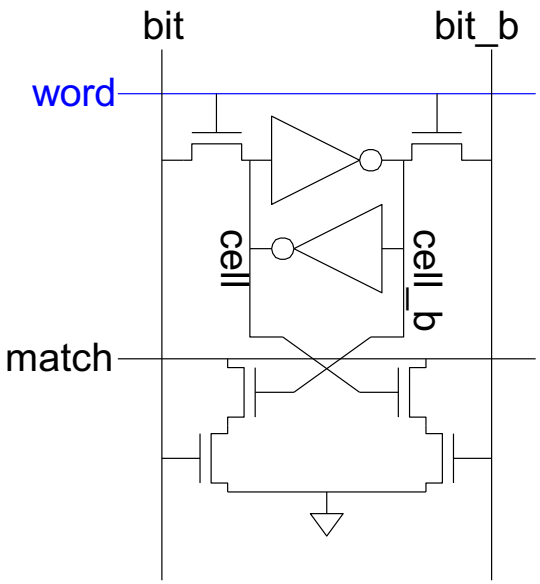
- Extension of ordinary memory (e.g. SRAM)
  - Read and write memory as usual
  - Also *match* to see which words contain a *key*
- An application example: Virtual memory(TLB)

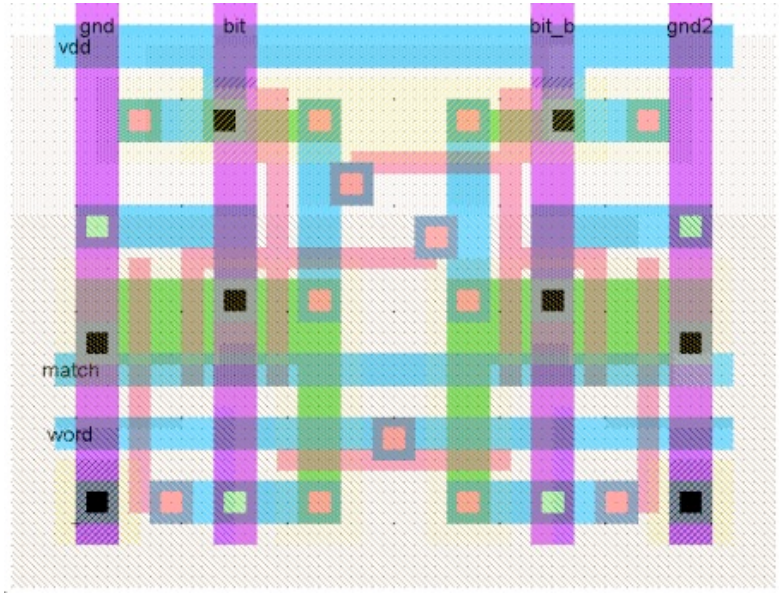




10T CAM Cell

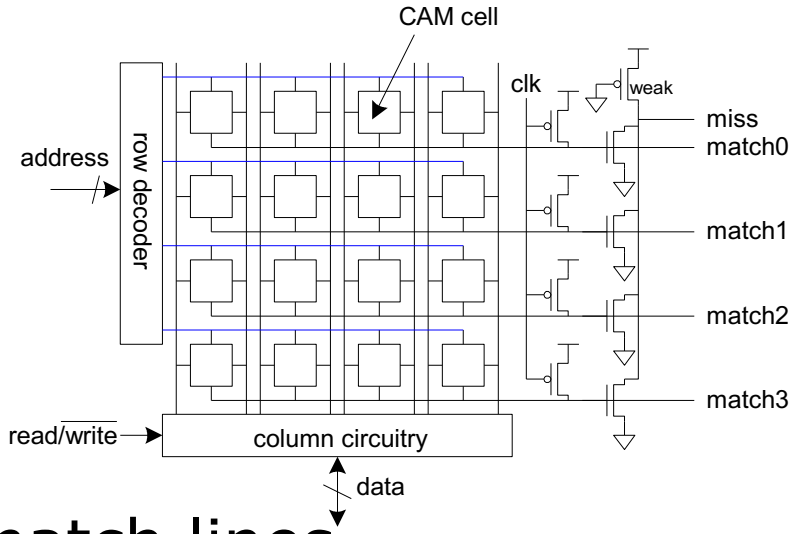
- Add four match transistors to 6T SRAM
  - 56 x 43 λ unit cell





CAM Cell Operation

- Read and write like ordinary SRAM
- For matching:
  - Leave wordline low
  - Precharge matchlines
  - Place key on bitlines
  - Matchlines evaluate
- Miss line
  - Pseudo-nMOS NOR of match lines
  - Goes high if no words match



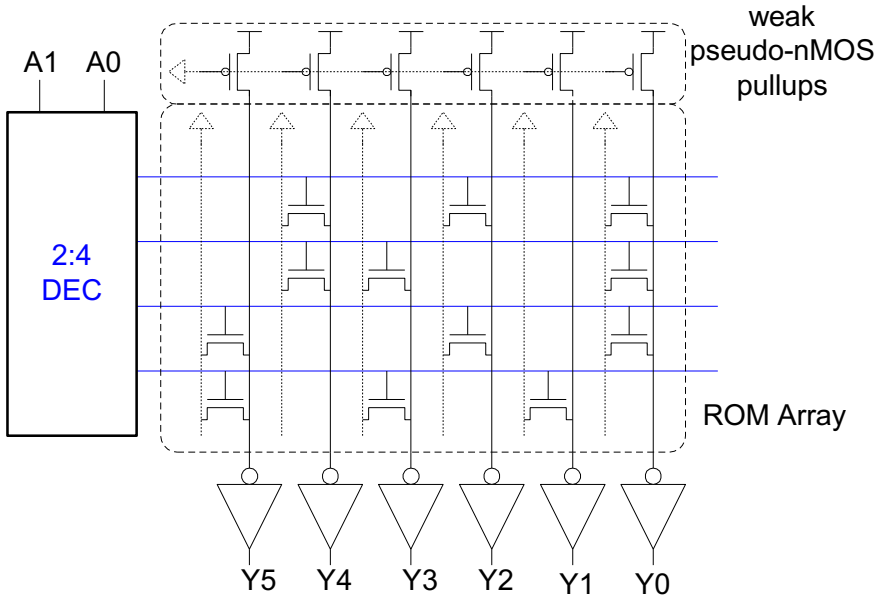
Read-Only Memories

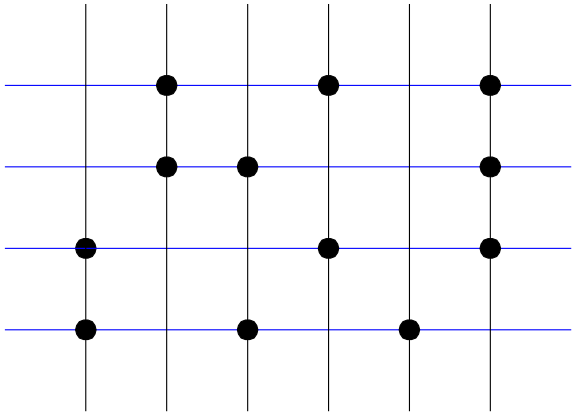
- Read-Only Memories are nonvolatile
  - Retain their contents when power is removed
- Mask-programmed ROMs use one transistor per bit
  - Presence or absence determines 1 or 0

ROM Example

- 4-word x 6-bit ROM
  - Represented with dot diagram
  - Dots indicate 1's in ROM

```
Word 0: 010101
Word 1: 011001
Word 2: 100101
Word 3: 101010
```

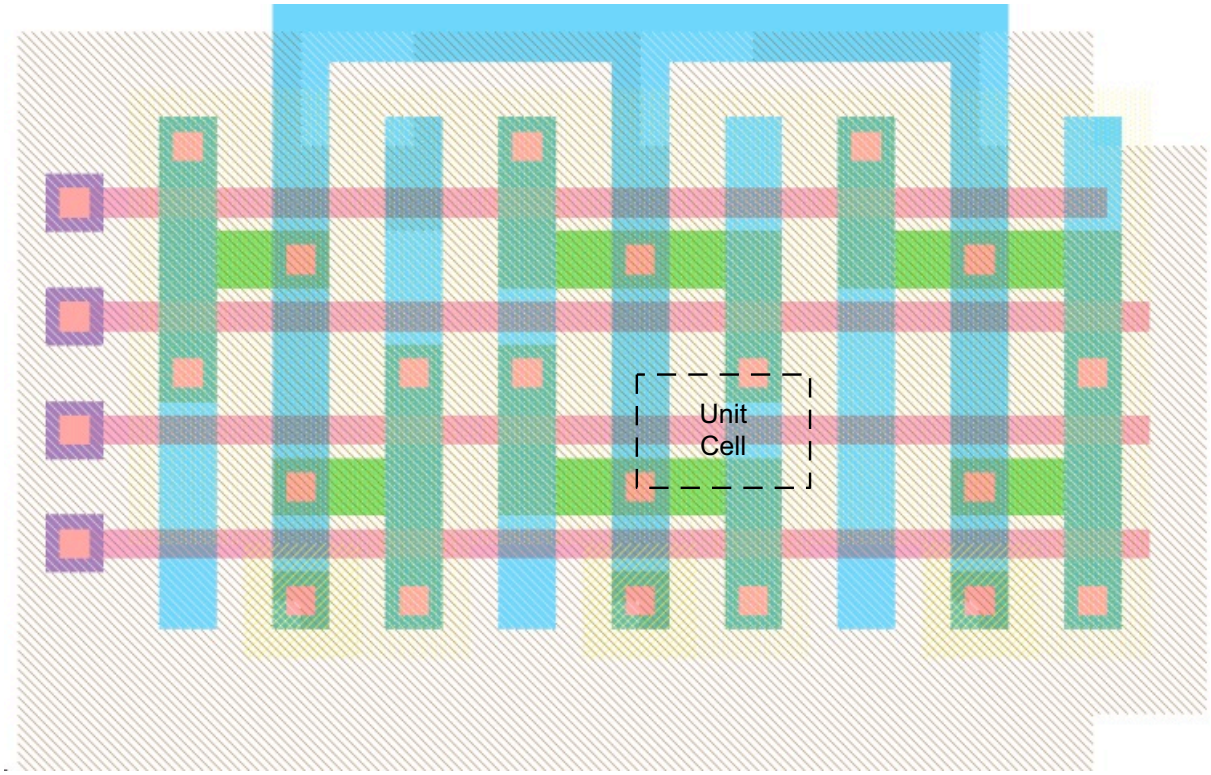




Looks like 6 4-input pseudo-nMOS NORs

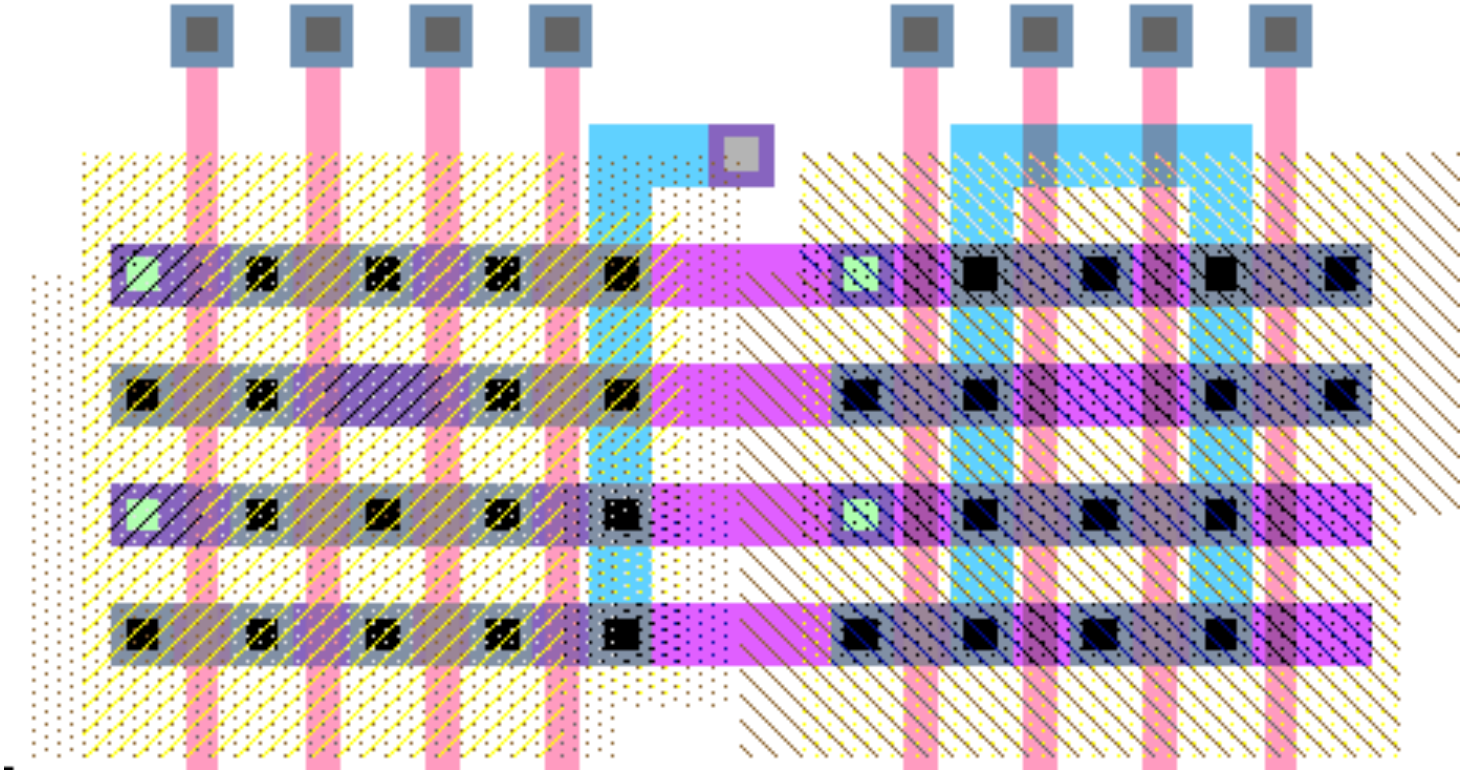
ROM Array Layout

- Unit cell is 12 x 8 λ (about 1/10 size of SRAM)

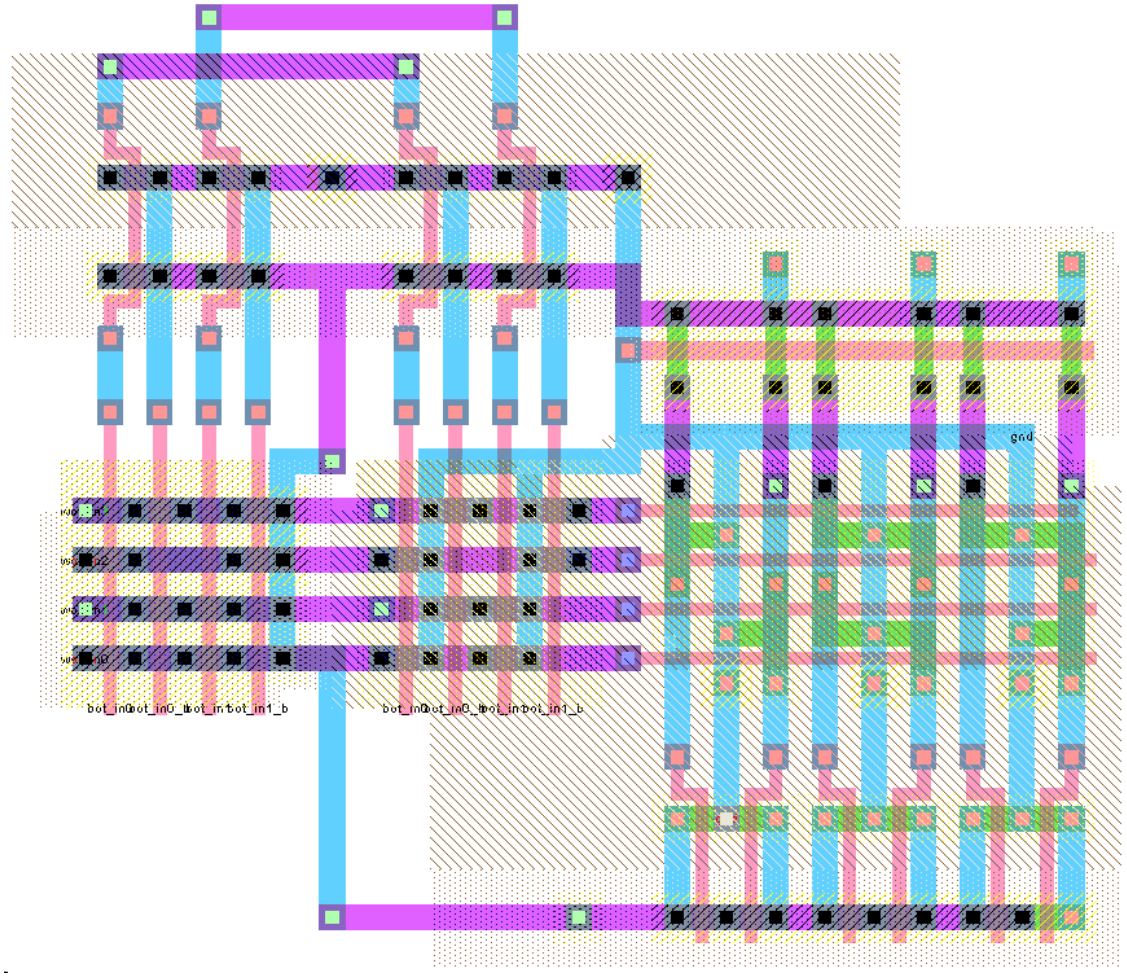


Row Decoders

- ROM row decoders must pitch-match with ROM
  - Only a single track per word!

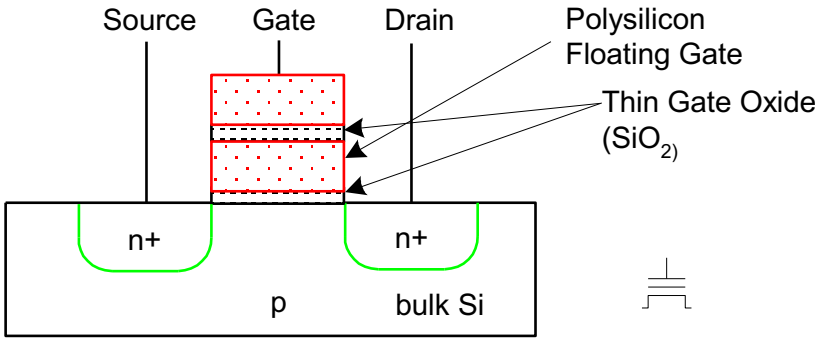


Complete ROM Layout



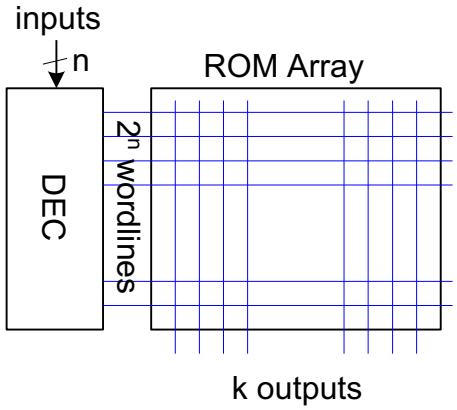
PROMs and EPROMs

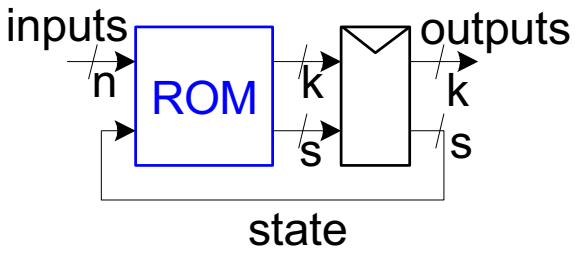
- Programmable ROMs
  - Build array with transistors at every site
  - Burn out fuses to disable unwanted transistors
- Electrically Programmable ROMs
  - Use floating gate to turn off unwanted transistors
  - EPROM, EEPROM, Flash



Building Logic with ROMs

- Use ROM as lookup table containing truth table
  - n inputs, k outputs requires 2^n words x k bits
  - Changing function is easy – reprogram ROM
- Finite State Machine
  - n inputs, k outputs, s bits of state
  - Build with 2^{n+s} x (k+s) bit ROM and (k+s) bit reg

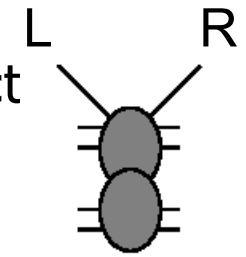




Example: RoboAnt

Let's build an Ant

Sensors: Antennae
(L,R) – 1 when in contact



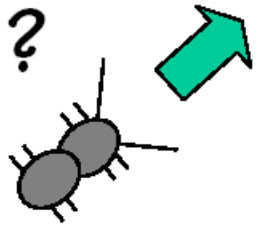
Actuators: Legs
Forward step F
Ten degree turns TL, TR

Goal: make our ant smart enough to get out of a maze

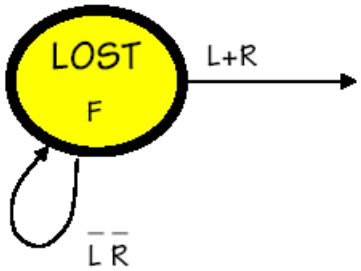
Strategy: keep right antenna on wall

(RoboAnt adapted from MIT 6.004 2002 OpenCourseWare by Ward and Terman)

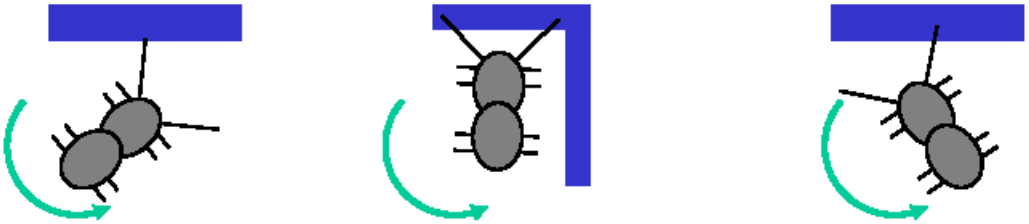
Lost in space



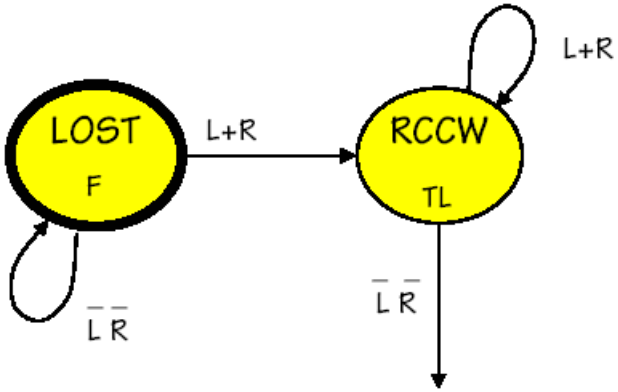
- Action: go forward until we hit something
  - Initial state



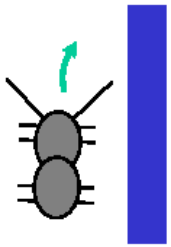
Bonk!!!



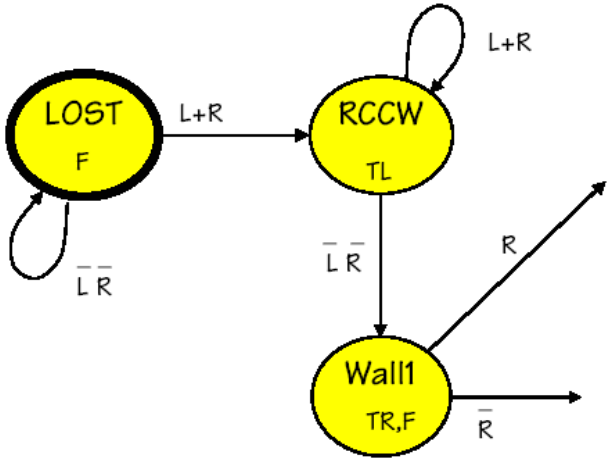
- Action: turn left (rotate counterclockwise)
  - Until we don't touch anymore



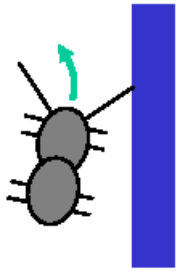
A little to the right



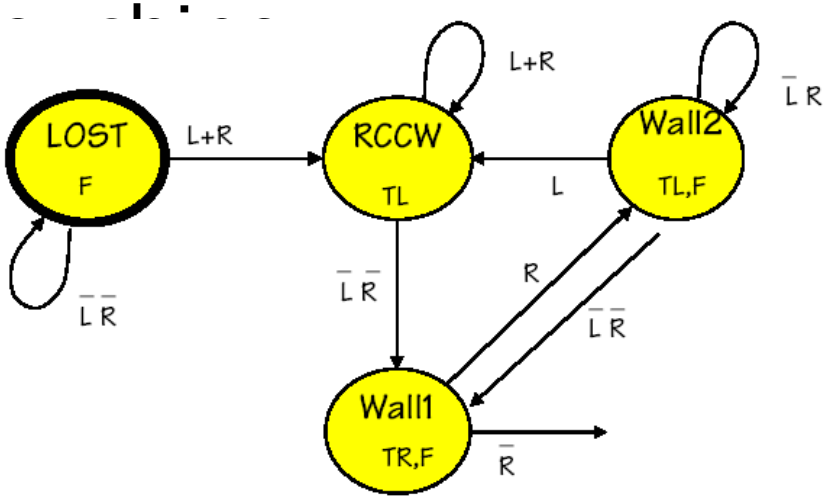
- Action: step forward and turn right a little
  - Looking for wall



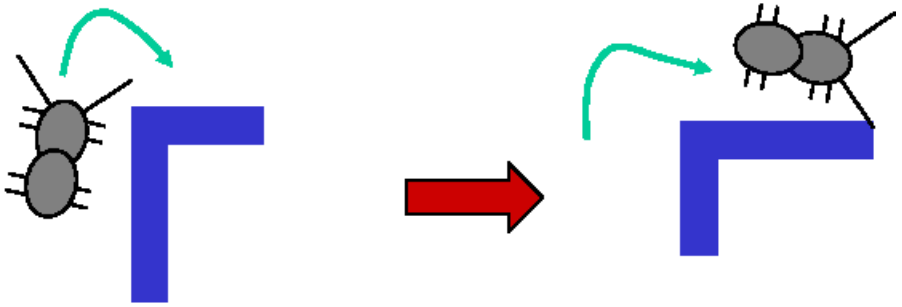
Then a little to the right



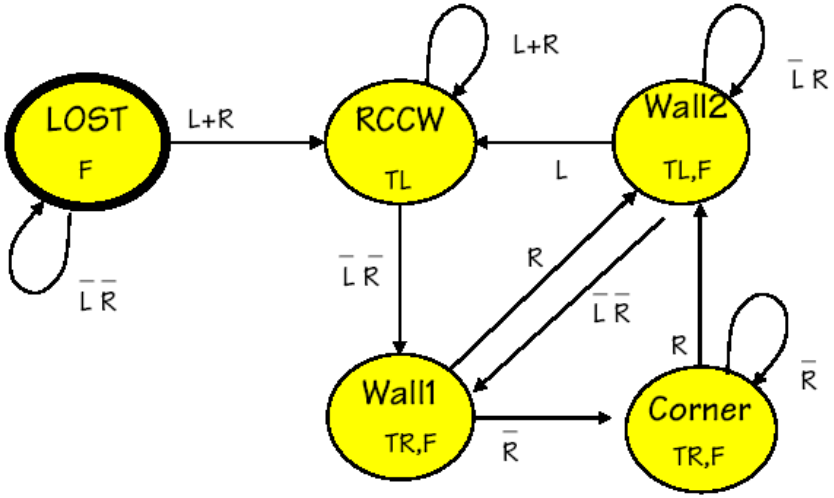
- Action: step and turn left a little, until not



Whoops – a corner!

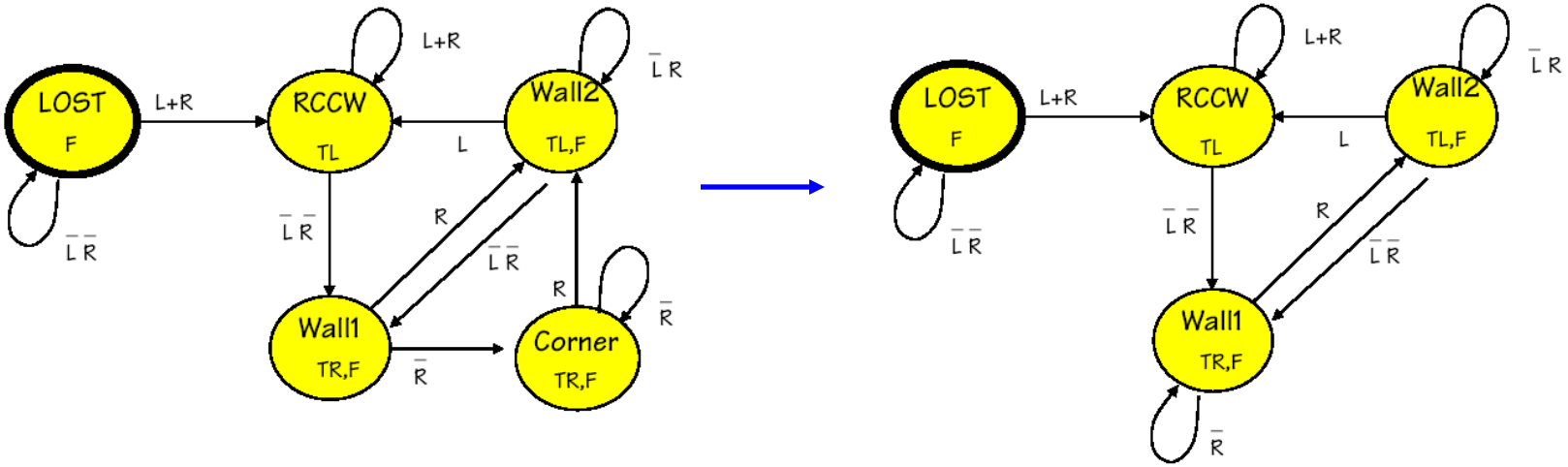


- Action: step and turn right until hitting next \



Simplification

- Merge equivalent states where possible

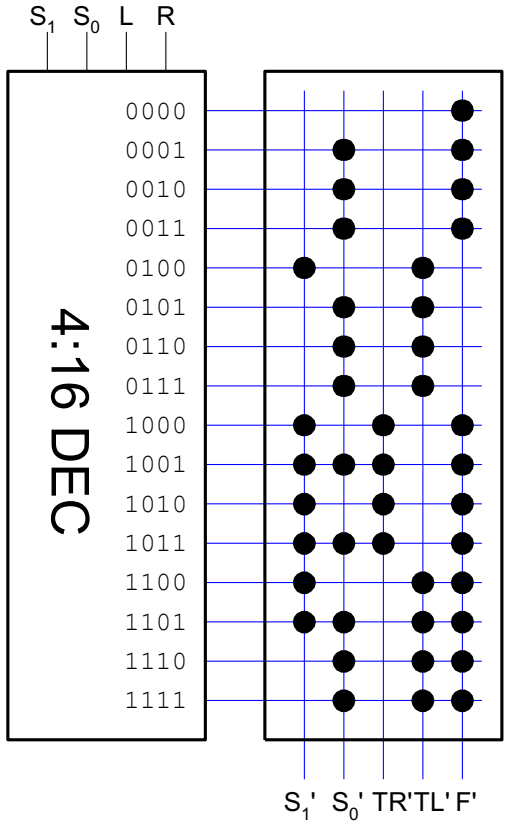


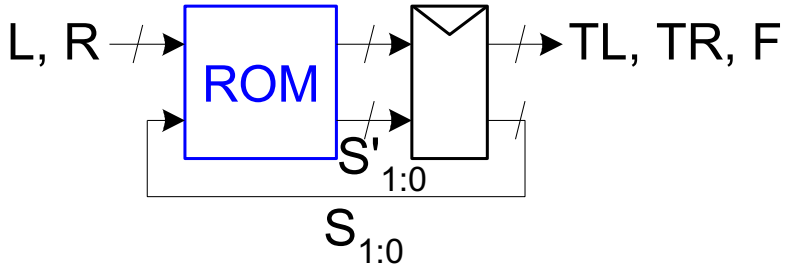
State Transition Table

	$S_{1:0}$	L	R	$S_{1:0}'$	TR	TL	F
Lost	00	0	0	00	0	0	1
	00	1	X	01	0	0	1
	00	0	1	01	0	0	1
RCCW	01	1	X	01	0	1	0
	01	0	1	01	0	1	0
	01	0	0	10	0	1	0
Wall1	10	X	0	10	1	0	1
	10	X	1	11	1	0	1
Wall2	11	1	X	01	0	1	1
	11	0	0	10	0	1	1
	11	0	1	11	0	1	1

ROM Implementation

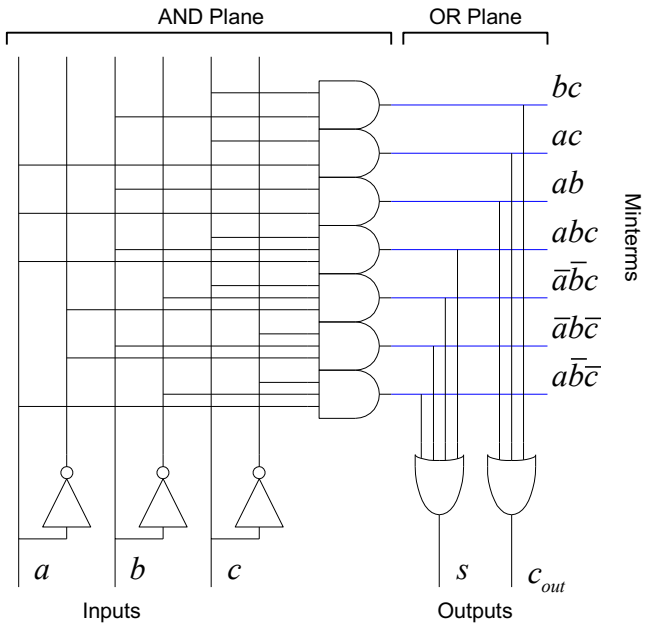
- 16-word x 5 bit ROM





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



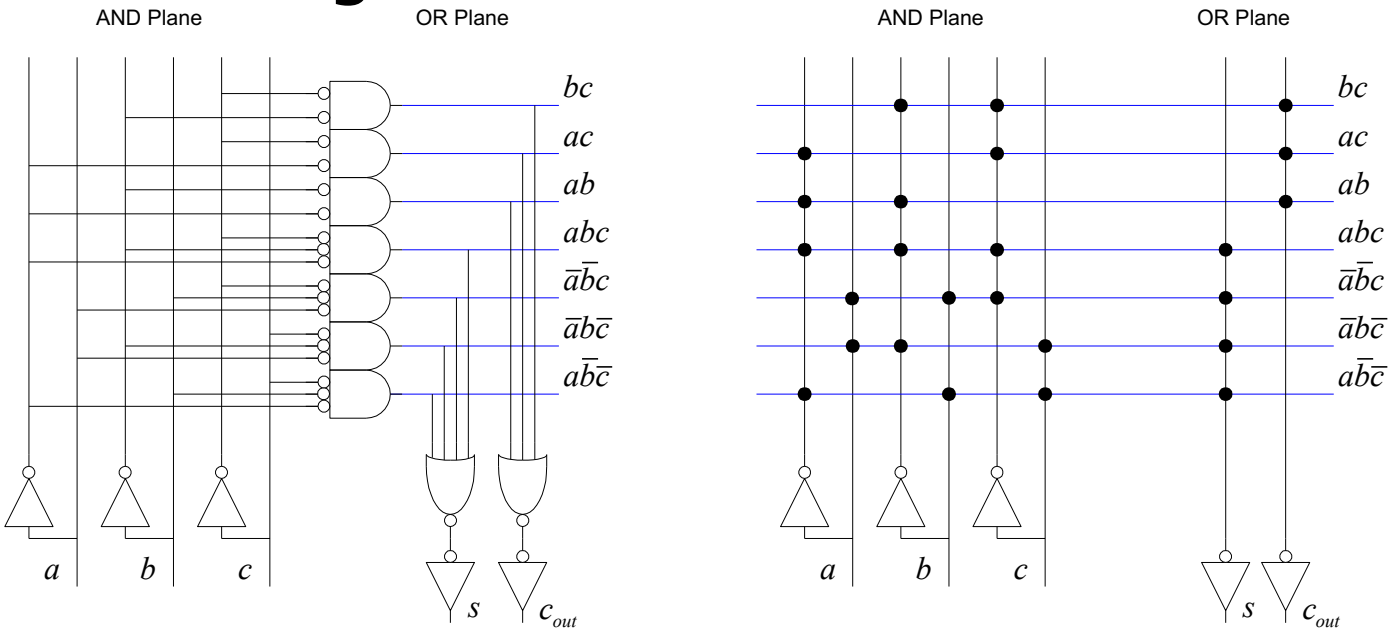
- Example: Full Adder

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

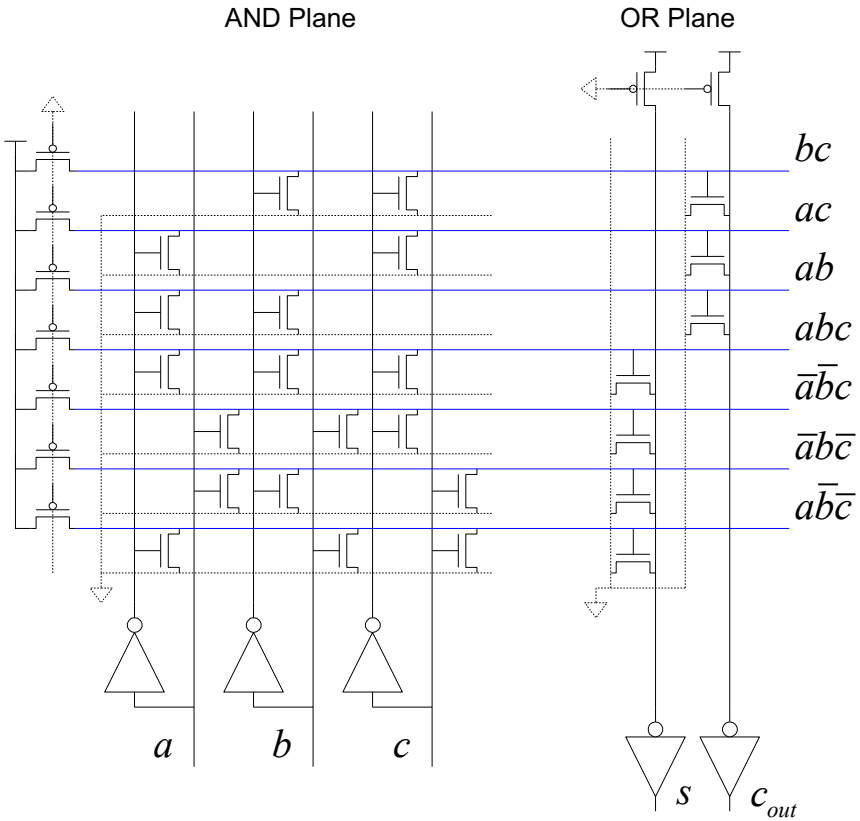
$$c_{\text{out}} = ab + bc + ac$$

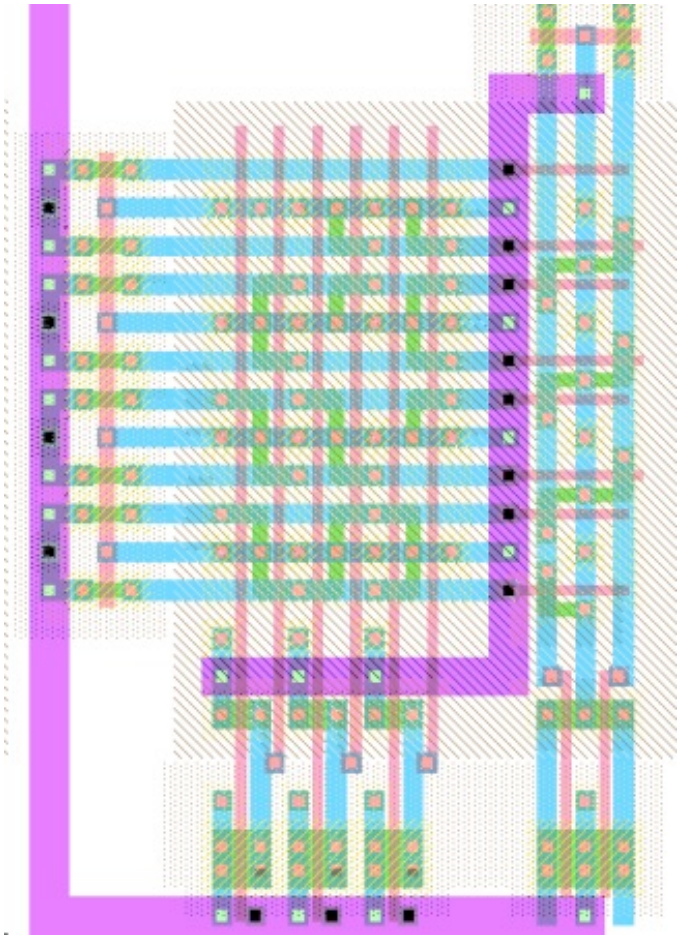
NOR-NOR PLAs

- ANDs and ORs are not very efficient in CMOS
- Dynamic or Pseudo-nMOS NORs are very efficient
- Use DeMorgan's Law to convert to all NORs



PLA Schematic & Layout





PLAs vs. ROMs

- The OR plane of the PLA is like the ROM array
- The AND plane of the PLA is like the ROM decoder
- PLAs are more flexible than ROMs
  - No need to have 2^n rows for n inputs
  - Only generate the minterms that are needed
  - Take advantage of logic simplification

Example: RoboAnt PLA

- Convert state transition table to logic equations

$S_{1:0}$	L	R	$S_{1:0}'$	TR	TL	F
00	0	0	00	0	0	1
00	1	X	01	0	0	1
00	0	1	01	0	0	1
01	1	X	01	0	1	0
01	0	1	01	0	1	0
01	0	0	10	0	1	0
10	X	0	10	1	0	1
10	X	1	11	1	0	1
11	1	X	01	0	1	1
11	0	0	10	0	1	1
11	0	1	11	0	1	1

S1′

LR \ S_1S_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	1
10	0	0	0	1

$$S_1' = S_1\overline{S_0} + \overline{L}S_1 + \overline{LR}S_0$$

S0′

LR \ S_1S_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	0

$$S_0' = R + L\overline{S_1} + LS_0$$

$$TR = S_1\overline{S_0}$$

$$TL = S_0$$

$$F = S_1 + \overline{S_0}$$

RoboAnt Dot Diagram

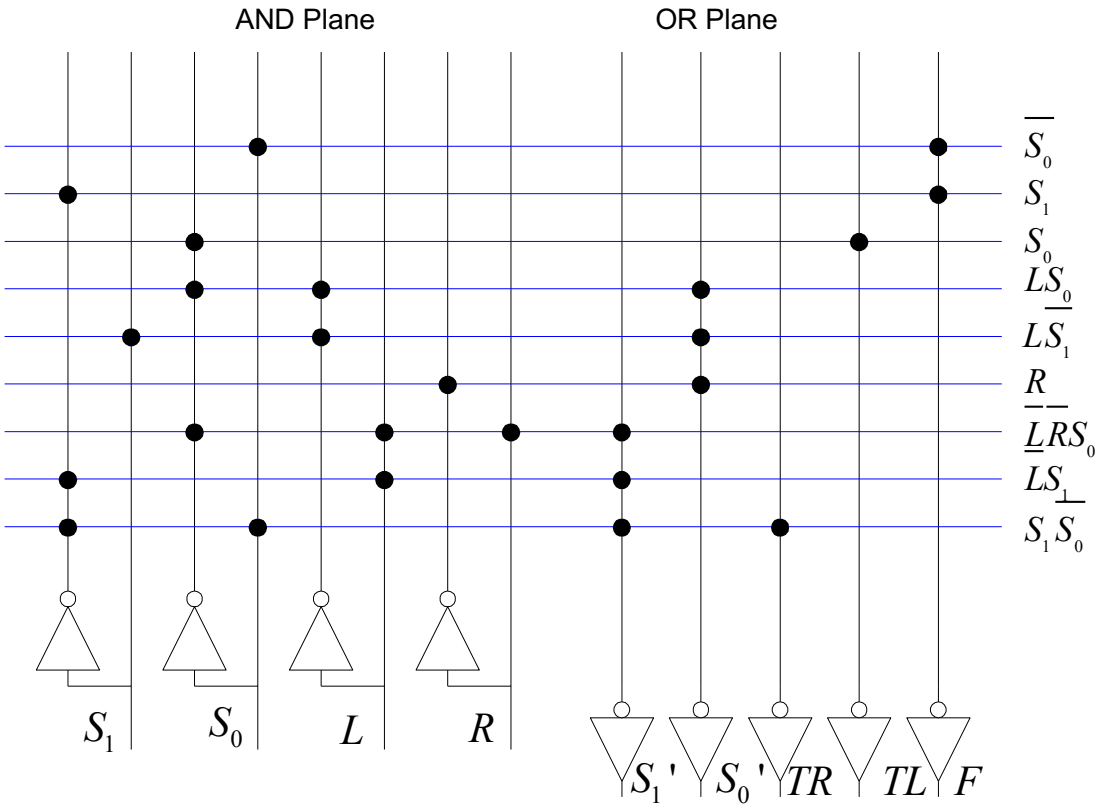
$$S1' = S_1\overline{S_0} + \overline{L}S_1 + \overline{L}\,\overline{R}S_0$$

$$S0' = R + L\overline{S_1} + LS_0$$

$$TR = S_1\overline{S_0}$$

$$TL = S_0$$

$$F = S_1 + \overline{S_0}$$



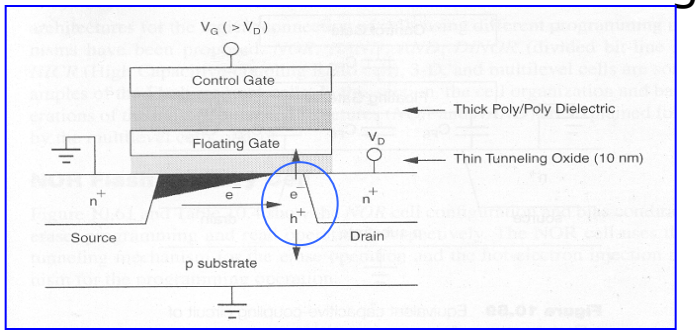
Part III: Flash Memory

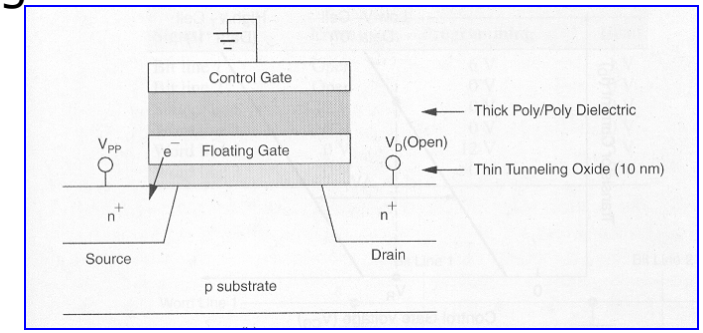
Introduction

- Flash memory: a later form of EEPROM
  - EEPROM
    - Programmed and erased multiple times electrically
    - write/erase cycle lifetimes: 100,000 to 10,000,000
    - Read from unlimited number of times
    - Bit-wise writable memory
    - Needs both a read and a write transistors
  - Flash memory
    - block-wise writable memory → faster
    - Needs only one transistor with floating gate
    - storing and eject electrons through hot-electron injection and tunneling mechanism

Flash Memory Cell

- One transistor with a floating gate





- 교과서에는 1과 0의 로직이 반대로 설명되었음. 확인요
- Threshold voltage can be changed (programmed) by applying electric field to its gate and terminals
  - Two states corresponding to charges (electrons) at the floating gate
    - Electrons at the floating gate → high V_{th} → logic "1"
      - Bit line precharged level is maintained because memory cell is not turned on (see slide 71)
    - Removing electrons at the floating gate → lower V_{th} → logic "0"
      - Bit line discharges to ground
- How to inject or remove electrons?
  - Injection: hot-electron ← high voltage V_D (6V) and V_{CG} ($V_{CG} > V_D$) (12V)
    - High energy → ionize atoms near drain→ self-reinforcing →rapidly increase hole + electron pairs → high voltage on CG attracts electrons at FG but not CG through thick dielectric ← but not destructive
  - Removal: Fowler-Nordheim tunneling mechanism
    - high voltage on source (source:12V or 20V, CG:0V) → attract electrons at FG through thin oxide tunnel

Equivalent Capacitive-Coupling Circuit and Threshold Voltages

- When V_{CG} and V_D are applied
  - By capacitive coupling (how to get V_{FG}?)

$$C_{total} = C_{FC} + C_{FS} + C_{FB} + C_{FD}$$

$$V_{FG} = \frac{Q_{FC}}{C_{total}} + \frac{C_{FC}}{C_{total}} V_{CG} + \frac{C_{FD}}{C_{total}} V_D$$

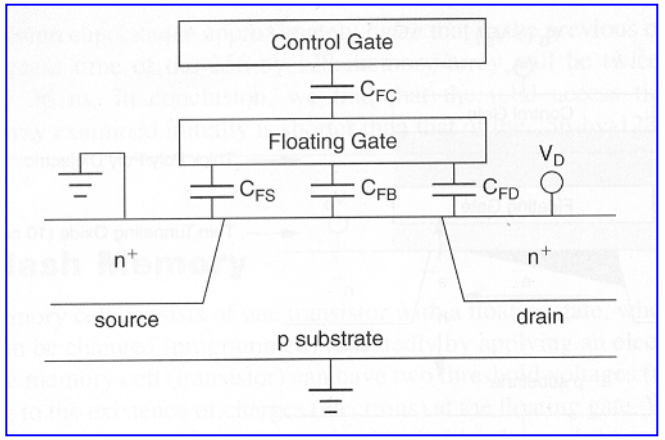
- Minimum control gate voltage
  - By substituting V_T(FG)(threshold voltage to turn on the floating gate transistor) into V_{FG} and arranging

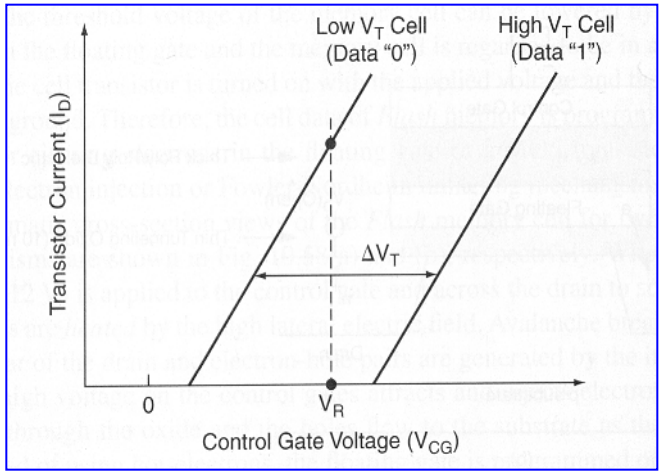
$$V_T(CG) = \frac{C_{total}}{C_{FC}} V_T(FG) - \frac{Q_{FG}}{C_{FC}} - \frac{C_{FD}}{C_{FC}} V_D$$

- The difference of threshold voltages between two memory data states "0" and "1"

$$\Delta V_T(\text{CG}) = -\frac{\Delta Q_{FG}}{C_{FC}}$$

- I-V characteristic curves of flash memory
  - When V_{CG} is V_R
    - Low V_T: ON, High V_T: OFF
    - Cell read operation
- Various types of the Flash memory cell structures
  - NOR and NAND structures are popular

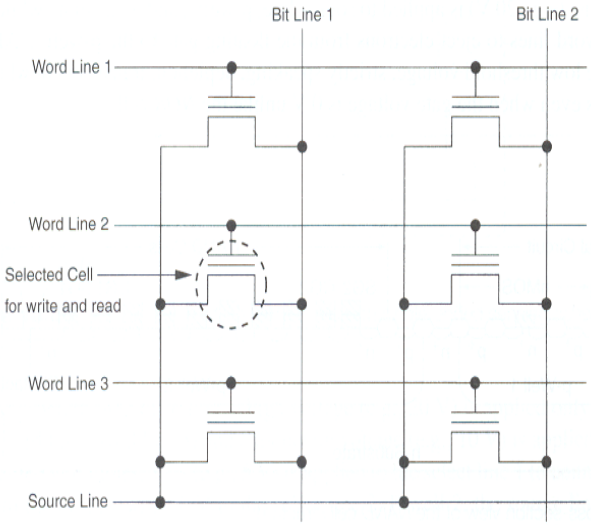




NOR Flash Memory Cell

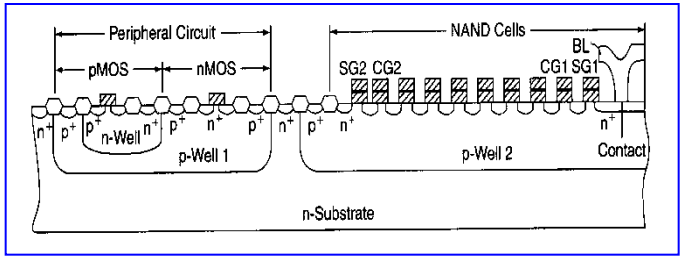
- NOR cell configuration and bias conditions for erase, programming and read operations
- Erase operation
  - F-N tunneling mechanism
- Programming operation
  - hot-electron injection mechanism
- Read operation
  - By applying moderate voltage (V_{DD}) to the control gate and a small voltage (1V) to drain to avoid generating hot-electrons
  - When the cell data is '0' (no electrons at FG), current flows
  - Detects this current flow → amplify the signal difference → read out cell data
- Allows random access to any location
  - By a full address/data interface
  - Suitable for storage of program code

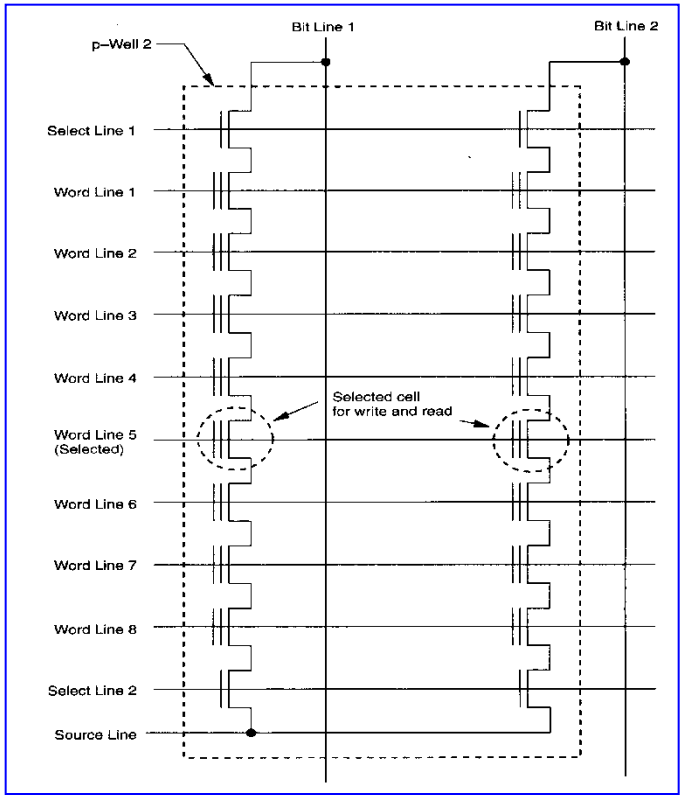
	Operation		
Signal	Erase	Programming	Read
Bit line 1	Open	6 V	1 V
Bit line 2	Open	0 V	0 V
Source line	12 V	0 V	0 V
Word line 1	0 V	0 V	0 V
Word line 2	0 V	12 V	5 V
Word line 3	0 V	0 V	0 V



NAND Flash Memory Cell

- By placing eight or sixteen cells in series,
  - Reduced area by eliminating of contacts!
- Cross-sectional view of NAND cell structure (top right figure)
- Bias condition for erase, programming and read operations (see table)
  - Erase: F-N tunneling mechanism
    - High voltage (e.g. 20V) to P-well 2 and n-substrate





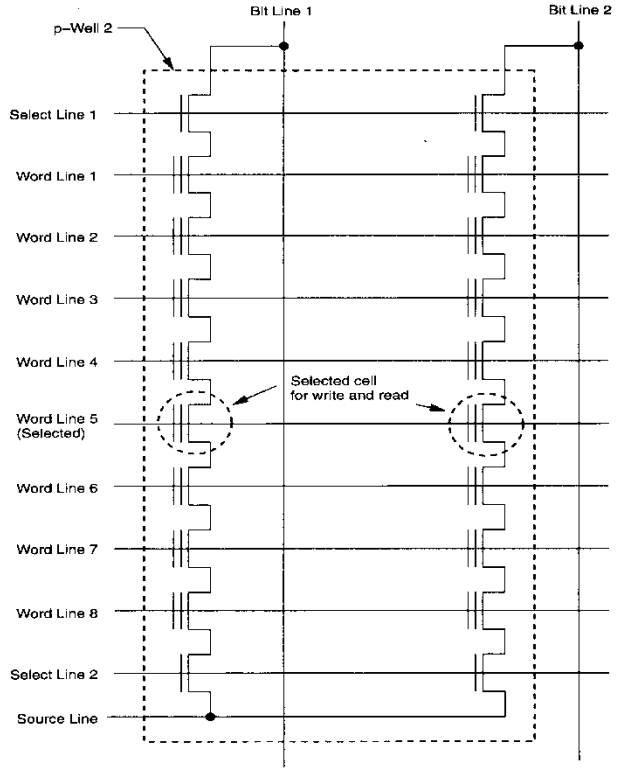
Signal	Operation		
	Erase	Programming	Read
Bit line 1	Open	0 V	1 V
Bit line 2	Open	0 V	1 V
Select line 1	Open	5 V	5 V
Word line 1	0 V	10 V	5 V
Word line 2	0 V	10 V	5 V
Word line 3	0 V	10 V	5 V
Word line 4	0 V	10 V	5 V
Word line 5	0 V	20 V	0 V
Word line 6	0 V	10 V	5 V
Word line 7	0 V	10 V	5 V
Word line 8	0 V	10 V	5 V
Select line 2	Open	0 V	5 V
Source line	Open	0 V	0 V
p-well 2	20 V	0 V	0 V
n-sub	20 V	0 V	0 V

NAND Flash Memory Cell

- Programming: F-N tunneling mechanism
  - High voltage only to selected word line
  - Moderate voltage to all the unselected WLs
  - TR at only the selected WL is enough to be programmed, electrons are attracted from the channel (substrate)

- Read operation
  - 0 voltage is applied to only selected WL
  - When '1' is stored, the cell transistor is turned off → precharge level of BL (e.g. 1V) is maintained
  - When '0' is stored, the current path is formed from the bit line to GND ← cell TR is normally turned on with the 0 V word line (usually negative threshold)

Signal	Operation		
	Erase	Programming	Read
Bit line 1	Open	0 V	1 V
Bit line 2	Open	0 V	1 V
Select line 1	Open	5 V	5 V
Word line 1	0 V	10 V	5 V
Word line 2	0 V	10 V	5 V
Word line 3	0 V	10 V	5 V
Word line 4	0 V	10 V	5 V
Word line 5	0 V	20 V	0 V
Word line 6	0 V	10 V	5 V
Word line 7	0 V	10 V	5 V
Word line 8	0 V	10 V	5 V
Select line 2	Open	0 V	5 V
Source line	Open	0 V	0 V
p-well 2	20 V	0 V	0 V
n-sub	20 V	0 V	0 V



Characteristic Comparison between the NOR and NAND Cells

	NOR	**NAND**
Erase method	Fowler-Nordheim tunneling	F-N tunneling
Programming method	Hot electron injection	F-N tunneling
Erase speed	Slow	Fast
Program speed	Fast	Slow
Read speed	Fast	Slow
Cell size	Large	Small
Scalability	Difficult	Easy
Applications	Embedded systems	Mass storage

- Why? $C_{total} = C_{FC} + C_{FS} + C_{FB} + C_{FD}$

$$V_{FG} = \frac{Q_{FC}}{C_{total}} + \frac{C_{FC}}{C_{total}} V_{CG} + \frac{C_{FD}}{C_{total}} V_D$$

- By applying superposition principle
  1. No VCG and no VD
  2. Only VCG
  3. Only VD

  VCG = result of (1+2+3)