# ESE3700: Circuit-Level Modeling, Design and Optimization for Digital Systems

Lec 19: April 15, 2024

RAM Core and Periphery

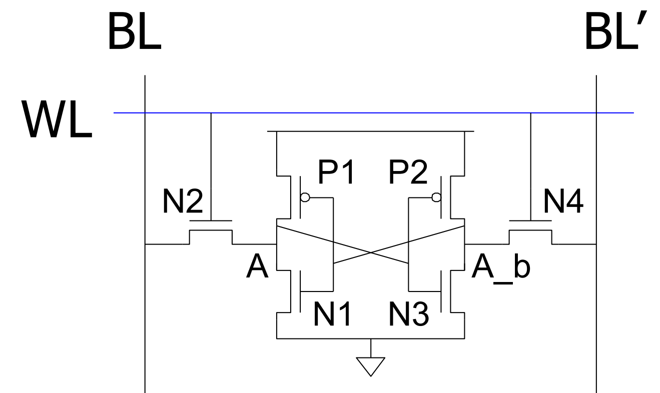
# Today

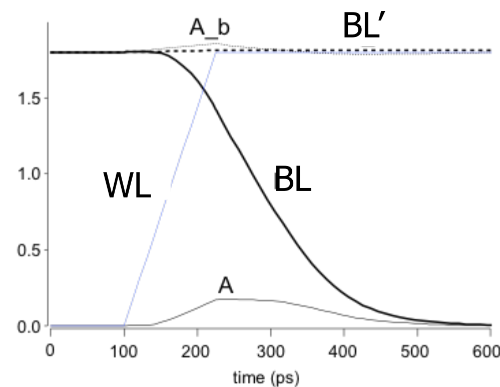
- Memory
  - Classification
  - Architecture
  - RAM Core
  - Periphery
  - Serial Access Memories

- Project 2 is on this

# SRAM Read

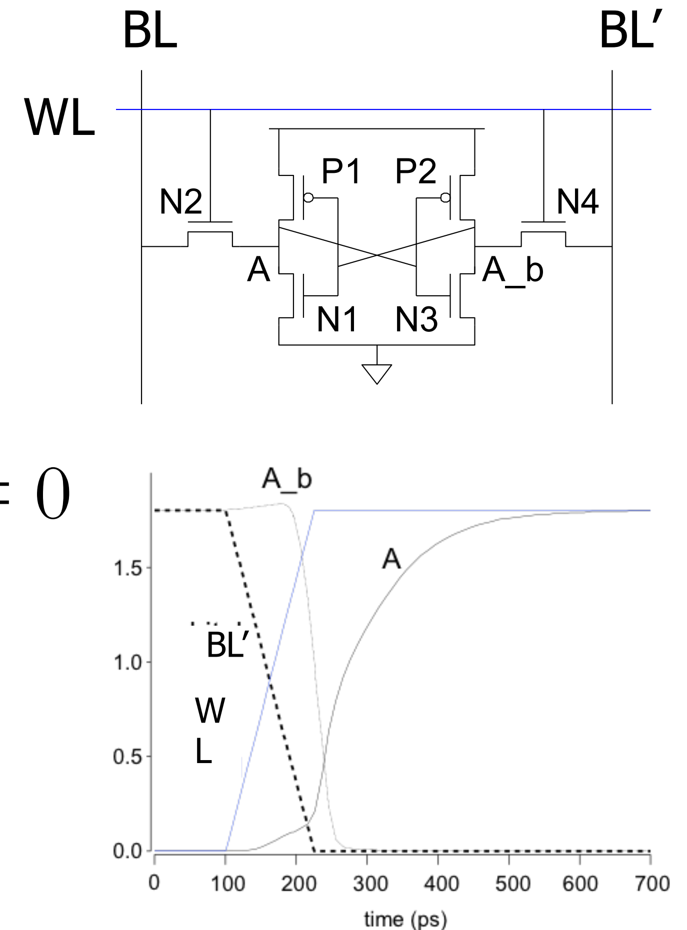
- Precharge both bitlines high
- Then turn on wordline, WL
- One of the two bitlines will be pulled down by the cell
- Ex: A = 0, A_b = 1
  - BL discharges, BL' stays high
  - But A bumps up slightly
- *Read stability*
  - A must not flip
  - N1 > N2

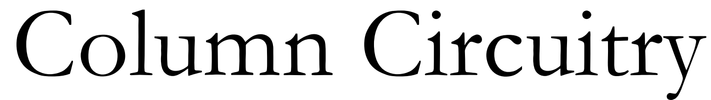


# SRAM Write

- Drive one bitline high, the other low
  - Depending on write data
- Then turn on wordline, WL
- Bitlines overpower cell with new value
- Ex: A = 0, A_b = 1, BL = 1, BL' = 0
  - Force A_b low, then A charges high
- *Writability*
  - Must overpower feedback inverter
  - N4 >> P2



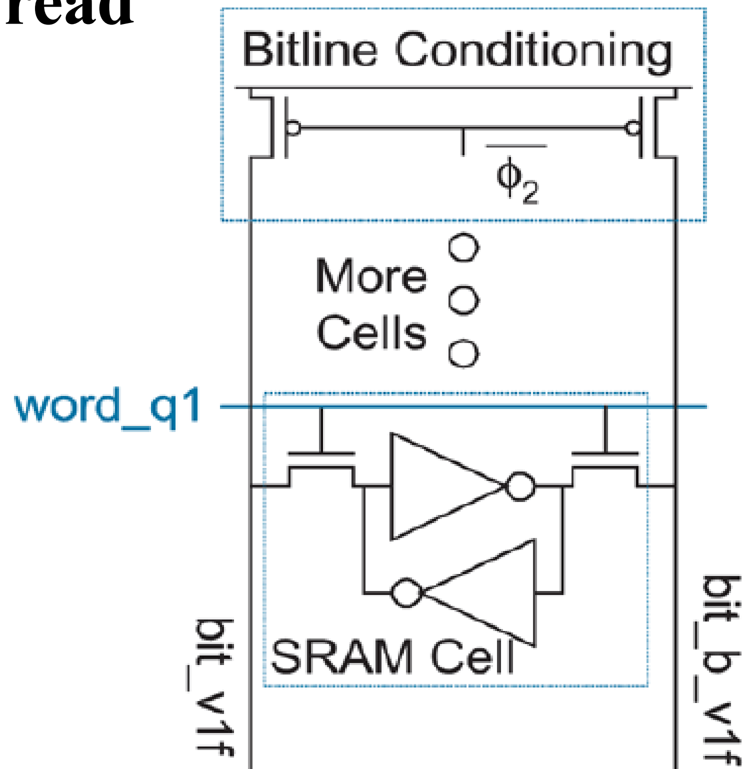


# Column Circuitry

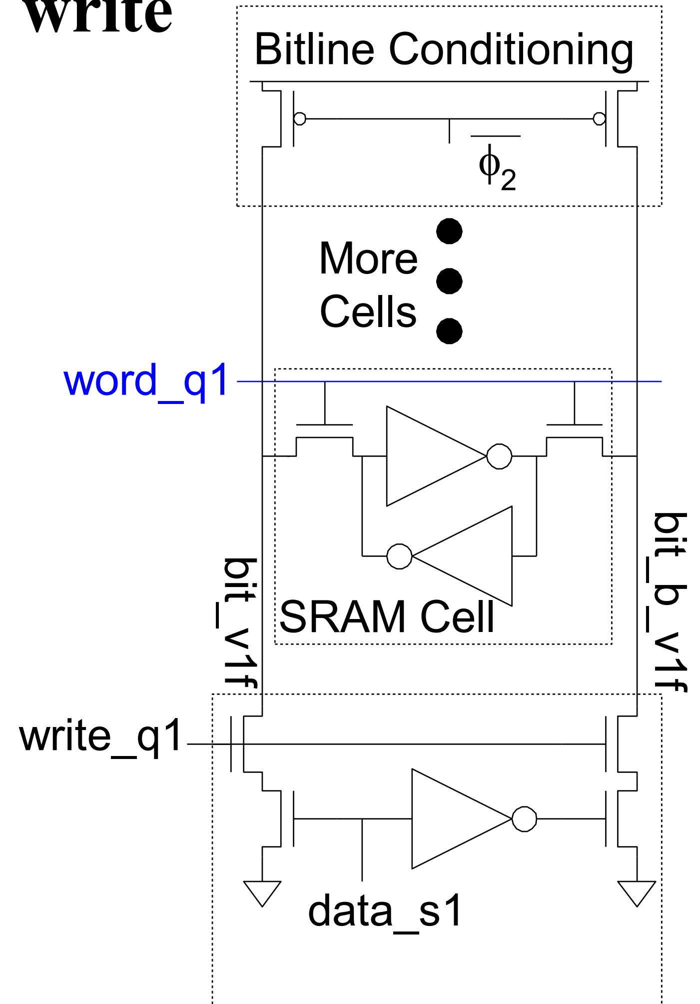
- Some circuitry is required for each column
  - Required: Bitline conditioning
    - Precharging
    - Driving input data to bitline
  - Increased speed: Sense amplifiers
  - Aspect ratio (square memory): Column multiplexing (AKA Column Decoders)

# SRAM Column Example

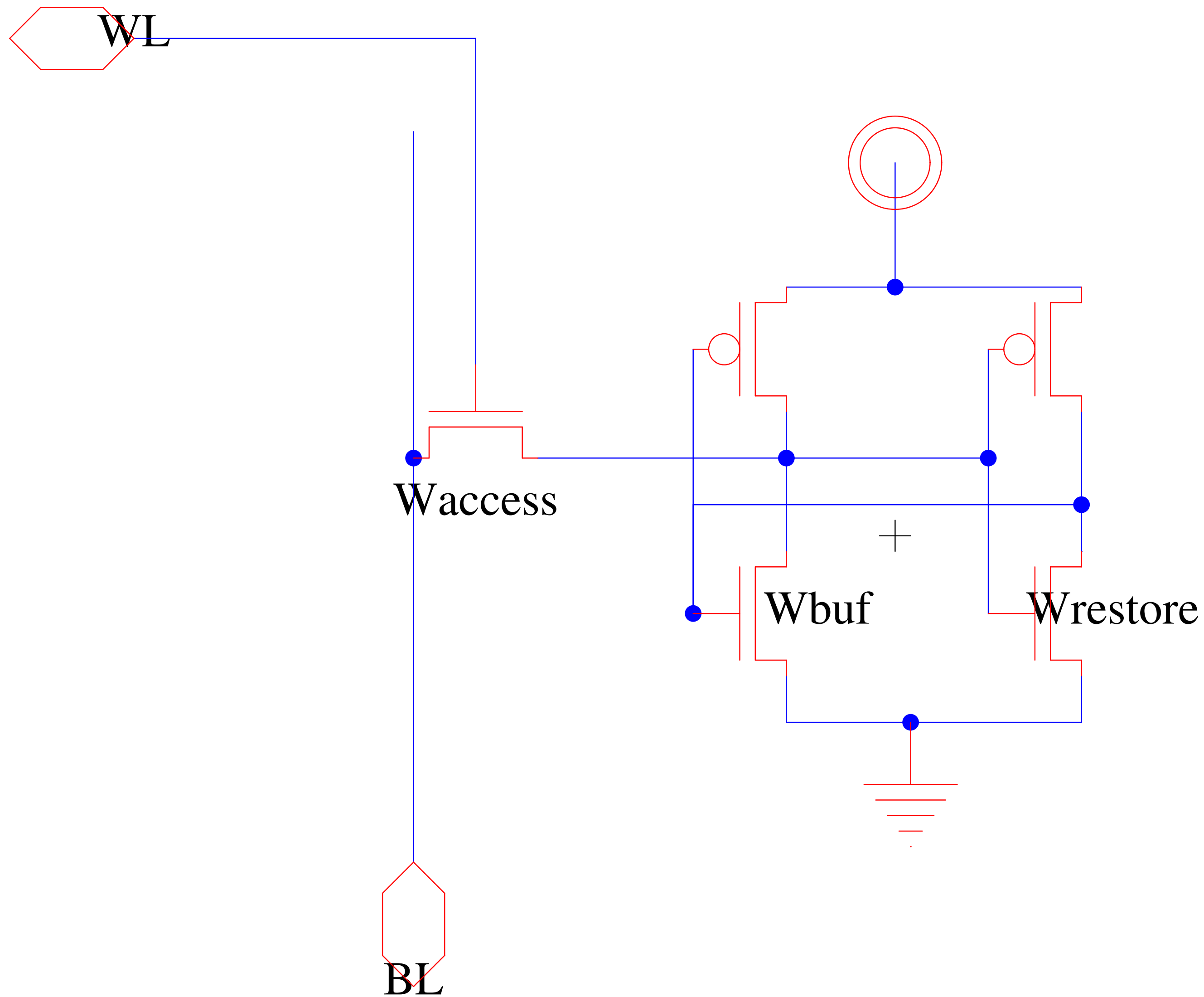
**read**




**write**



# 5T SRAM

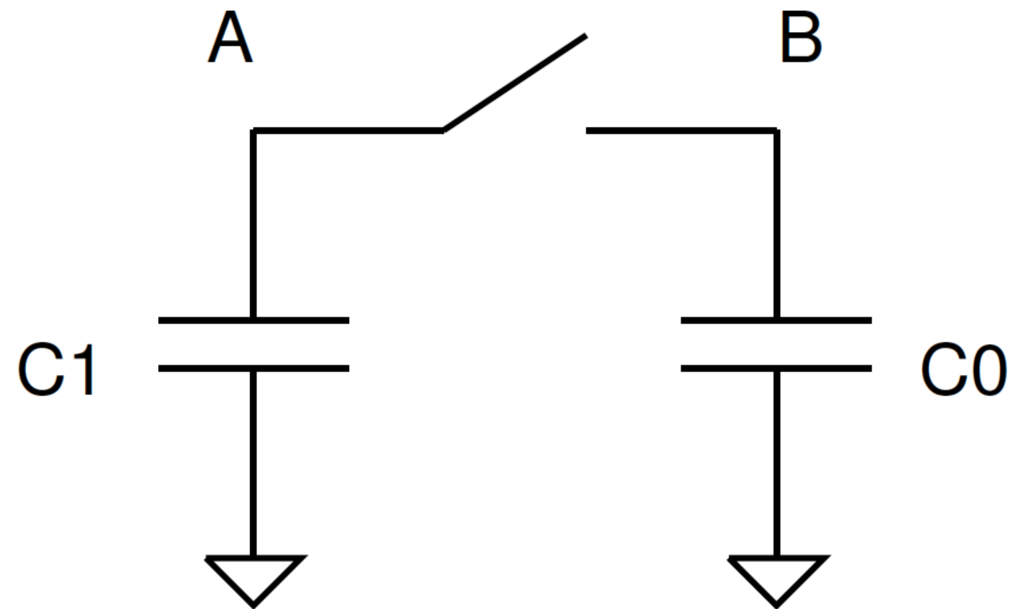


# Charge Sharing (Preclass 1)

Initially

- A @ 1V
- B @ 0V

- $Q_A$=1V*C1=C1



# Charge Sharing (Preclass 1)

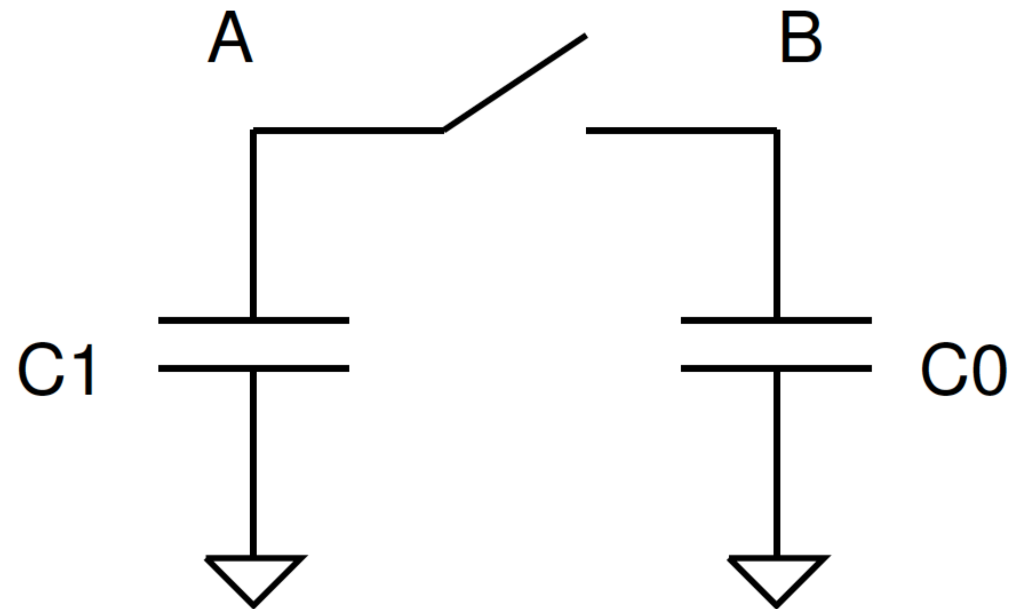
Initially

- A @ 1V
- B @ 0V

- $Q_A$=1V*C1=C1

Close switch

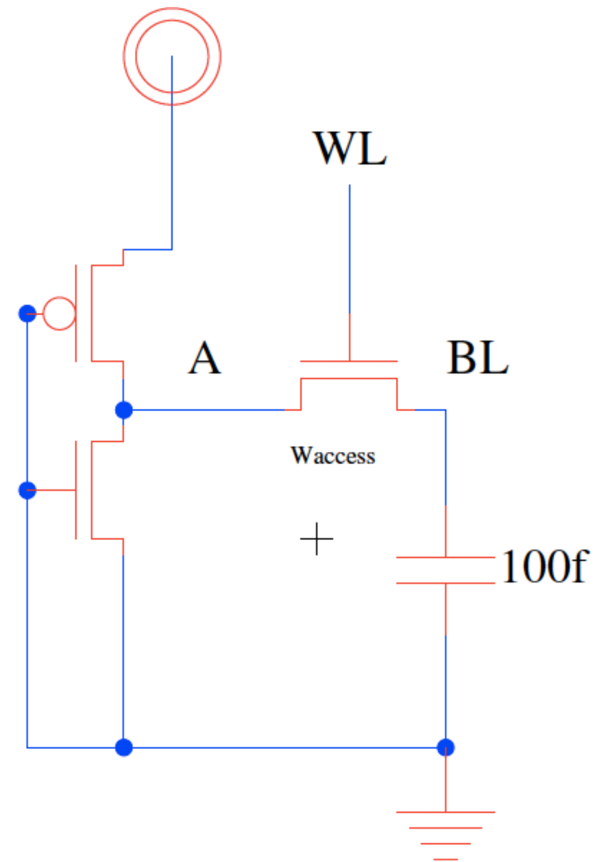
- $Q_{tot}=V_{final}*(C1+C0)$
- Charge conservation
  - $Q_A=Q_{tot}$
- $C1=V_{final}*(C1+C0)$




$$V_{final} = \frac{C1}{C1 + C0}$$

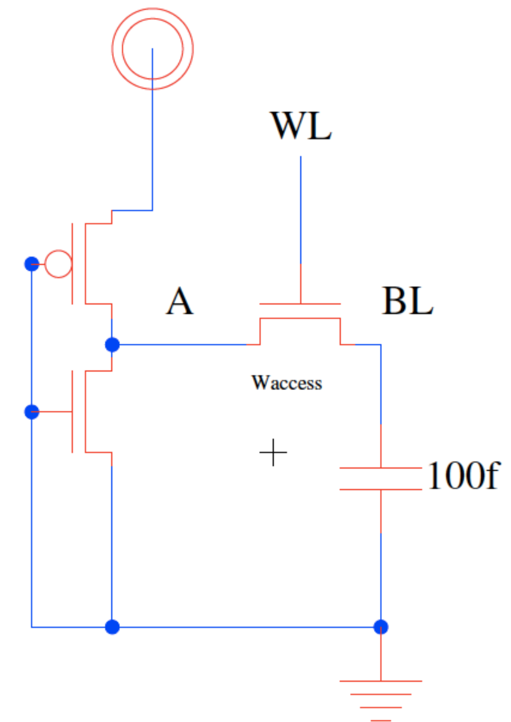
# Consider (preclass 2)

- Read: What happens to voltage at A when WL turns from 0→1?
  - Assume $W_{access}$ large
  - $W_{access} >> W_{pu}=1$
  - BL initially 0



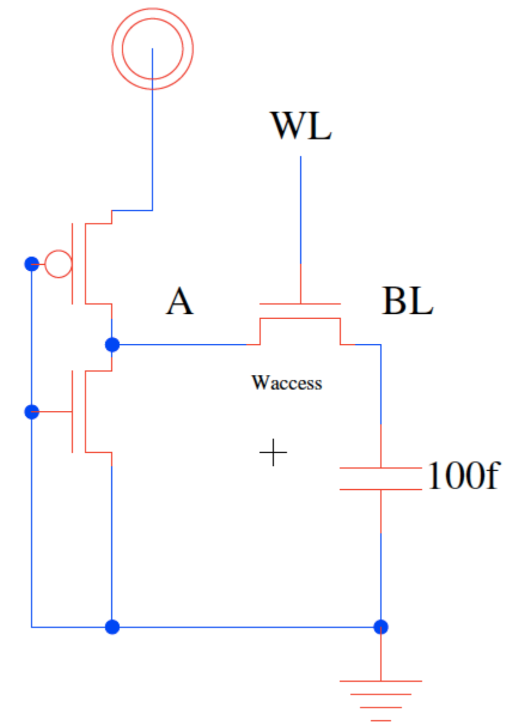
# Voltage After enable Word Line

- $Q_{BL} = 0$
- $Q_A = (1V)(\gamma 2C_0+\gamma W_{access}C_0)$
- $100fF=C_{BL}>>C_A=(\gamma(2+W_{access})C_0)$

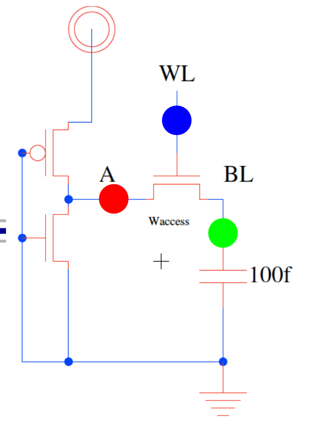


# Voltage After enable Word Line

- $Q_{BL} = 0$
- $Q_A = (1V)(\gamma 2C_0+\gamma W_{access}C_0)$
- $100fF=C_{BL}>>C_A=(\gamma(2+W_{access})C_0)$
- After enable $W_{access}$ ($W_{access}$ large)
  - Total charge $Q_{BL}+Q_A$ unchanged
    - Charge conservation
  - Distributed over larger capacitance $\sim=C_{BL}$
  - $V_A=V_{BL}\sim= C_A/C_{BL}$



# Simulation: $W_{access}$=100

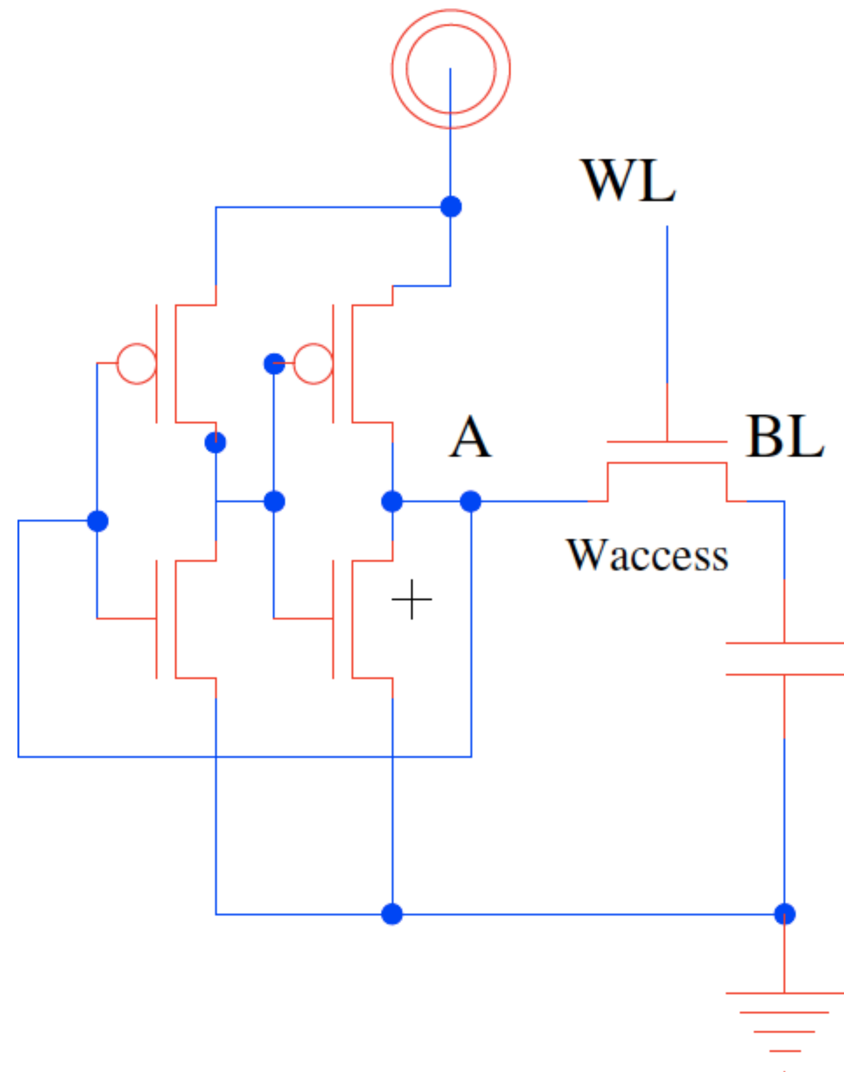



Transient Response

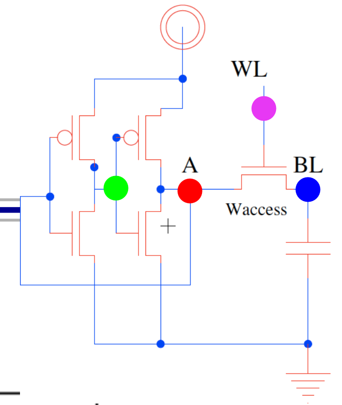
1
0.8
0.6
0.4
0.2
0
V
0
2e-10
4e-10
6e-10
8e-10
1e-09
s
A
BL
WL

# Consider (5T SRAM) (preclass 3)

- What happens to voltage at A when WL turns from 0→1?
  - Assume $W_{access}$ large
  - A initially 1
  - BL initially 0



# Simulation $W_{access}$=20




Transient Response

V

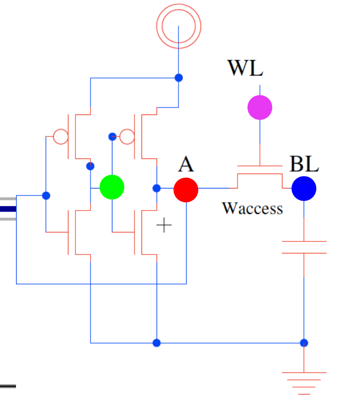
1
0.8
0.6
0.4
0.2
0

A
not A
BL
WL

1e-10 2e-10 3e-10 4e-10 5e-10 6e-10 7e-10 8e-10 9e-10 1e-09

s

# Simulation $W_{access}$=4




Transient Response

V

1
0.8
0.6
0.4
0.2
0

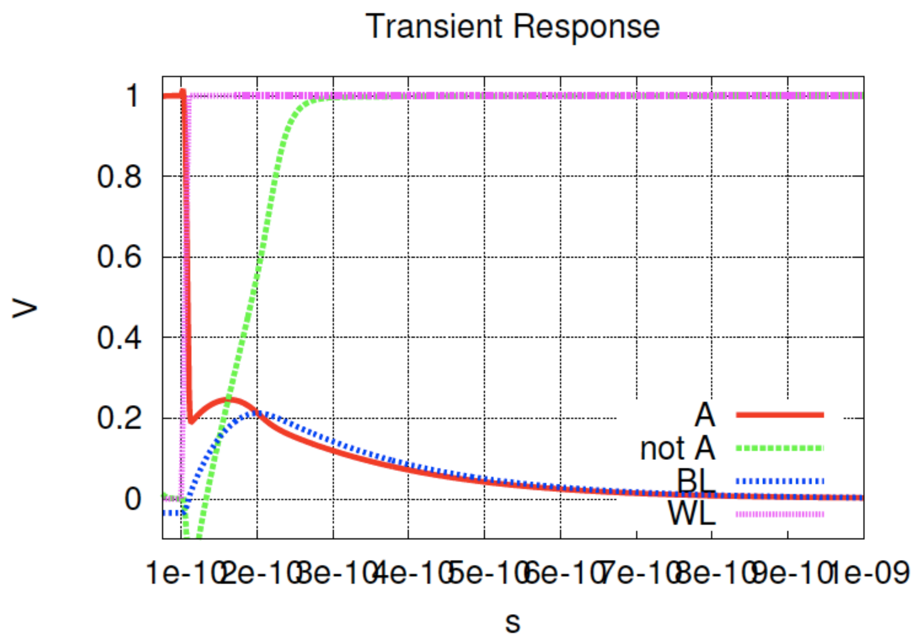
A
not A
BL
WL

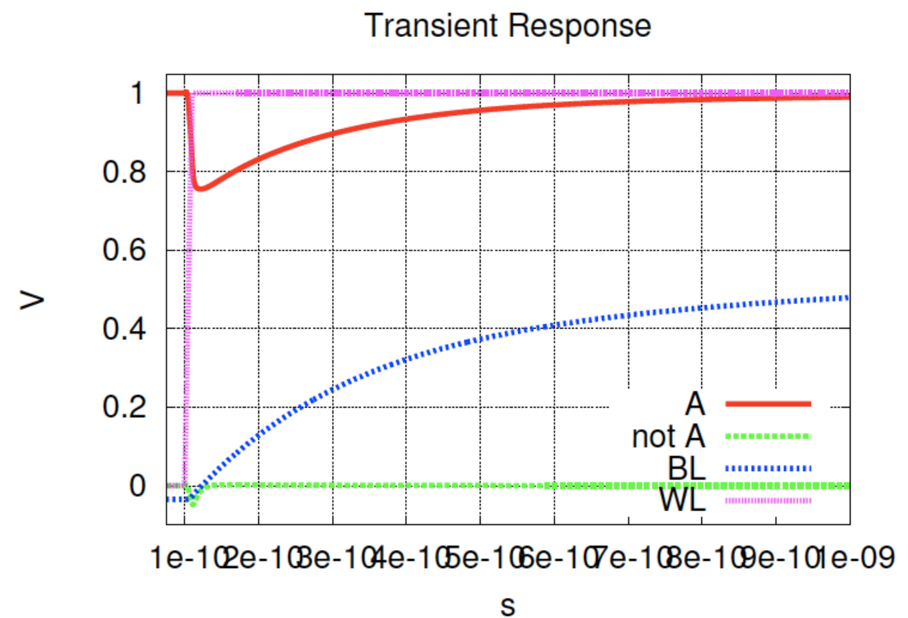
1e-10 2e-10 3e-10 4e-10 5e-10 6e-10 7e-10 8e-10 9e-10 1e-09

s

# Charge Sharing

- **Conclude:** charge sharing can lead to read upset
  - Charge redistribution/sharing adequate to flip state of bit



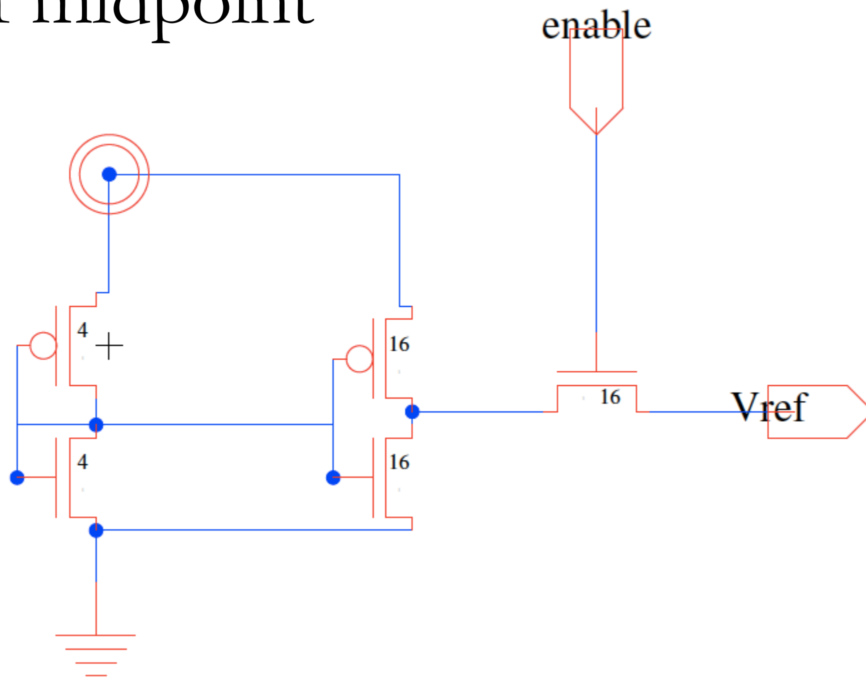



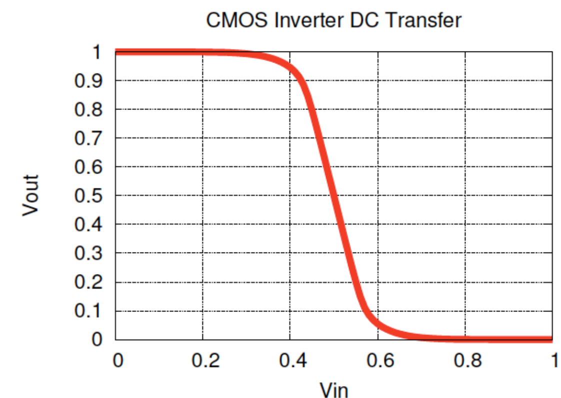

$W_{access}=20$

$W_{access}=4$

# Charge to middle Voltage

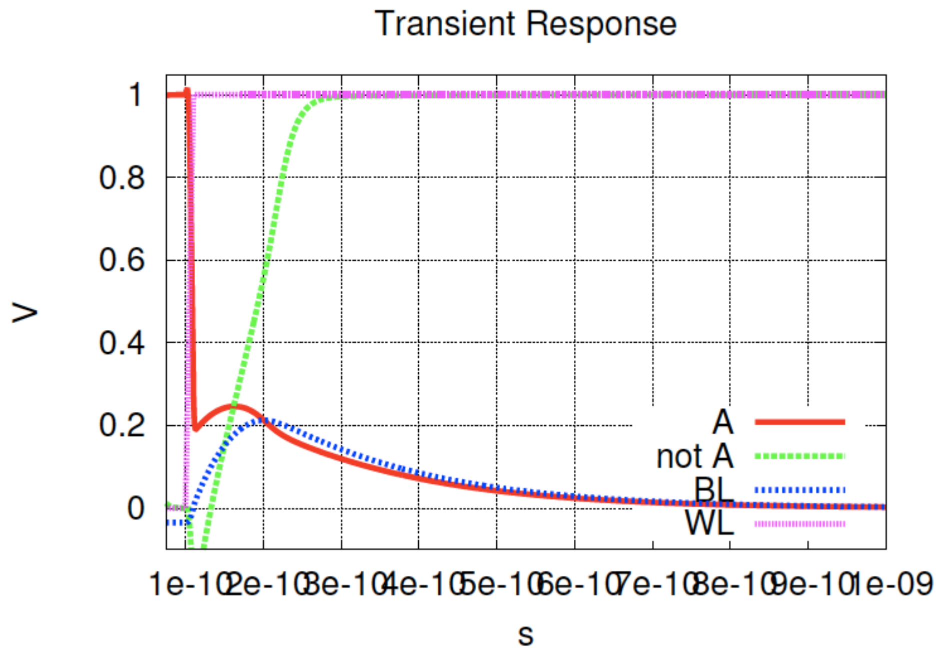
- Pre-charge bitlines to $V_{dd}/2$ before begin read operation
- Now charge sharing doesn't swing to opposite side of midpoint



# Compare

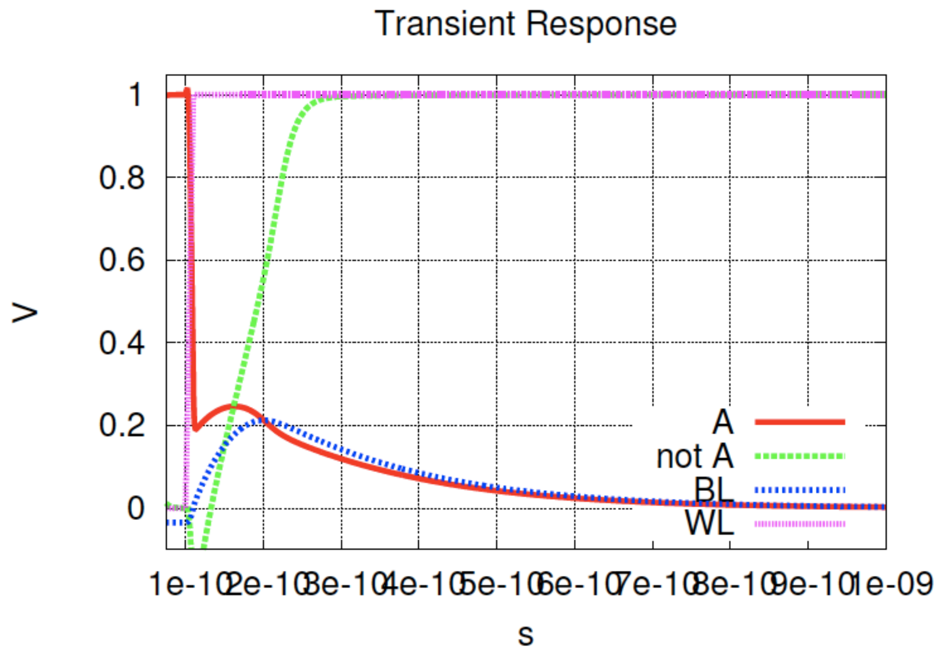
- Both $W_{access}$=20; vary BL precharge voltage



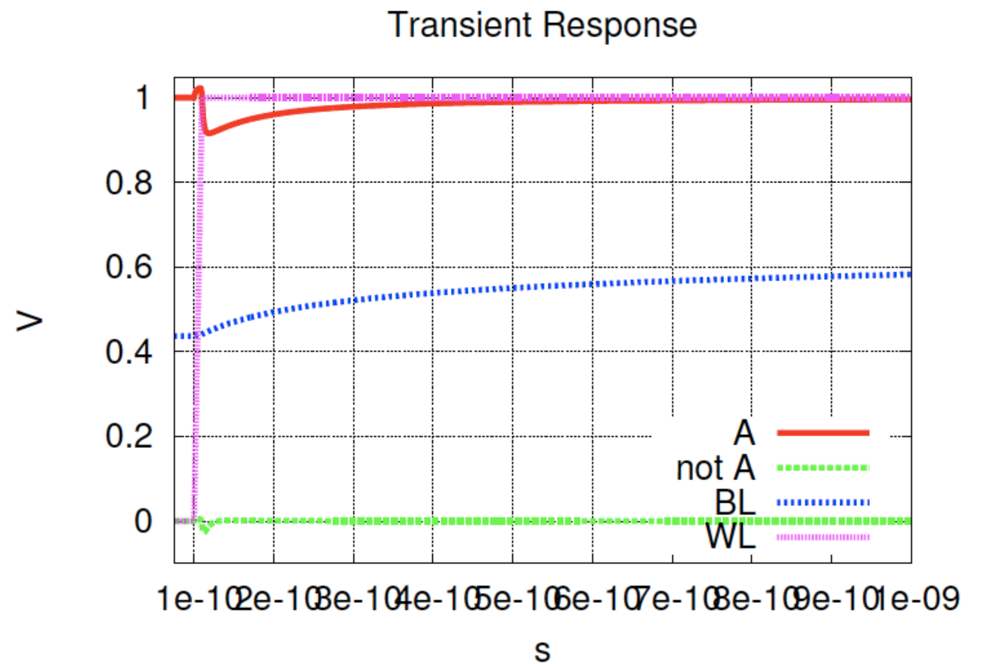

0 precharge

# Compare

- Both $W_{access}$=20; vary BL precharge voltage

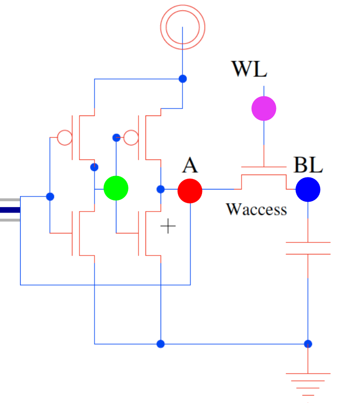



0 precharge




$V_{dd}$/2 precharge

# Simulation $W_{access}$=20 (precharge Vdd/2, reading 0)




Transient Response (reading 0)

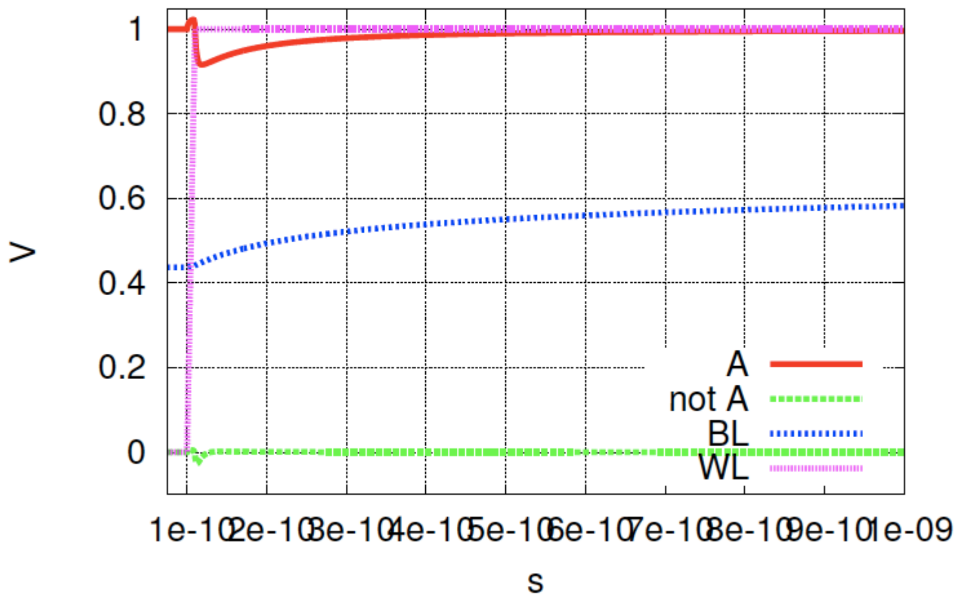
V

1
0.8
0.6
0.4
0.2
0

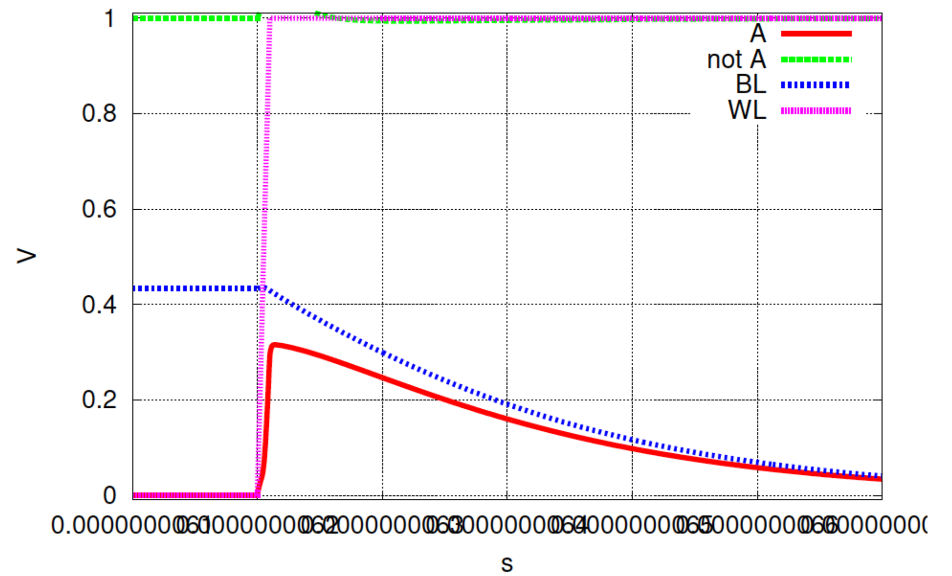
A
not A
BL
WL

s

# Simulation $W_{access}$=20 (with precharge Vdd/2)



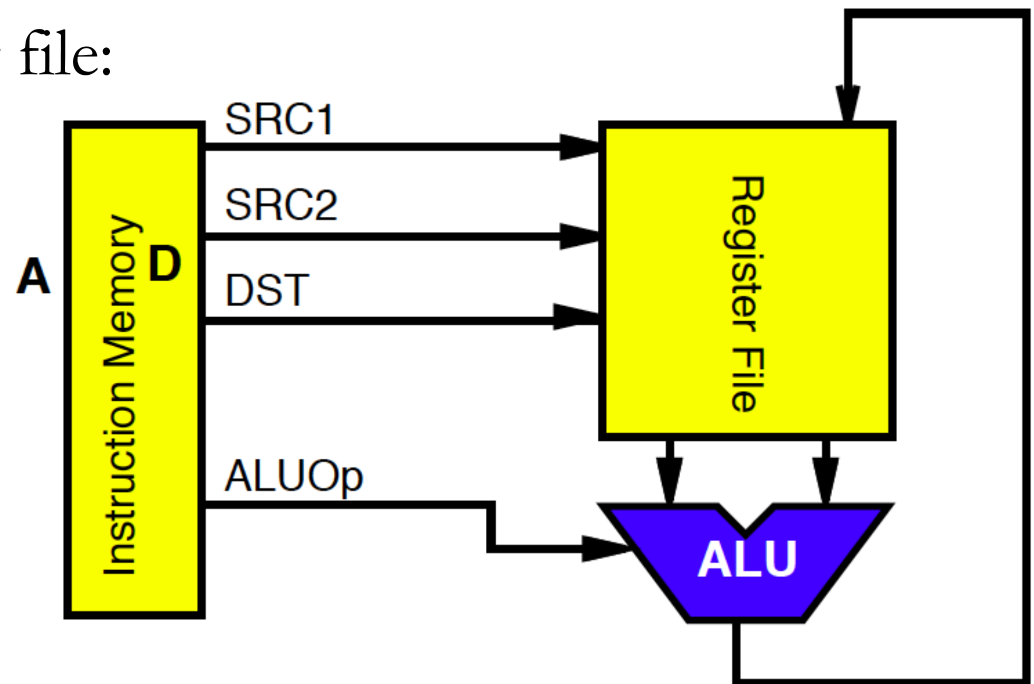

Read 1




Read 0

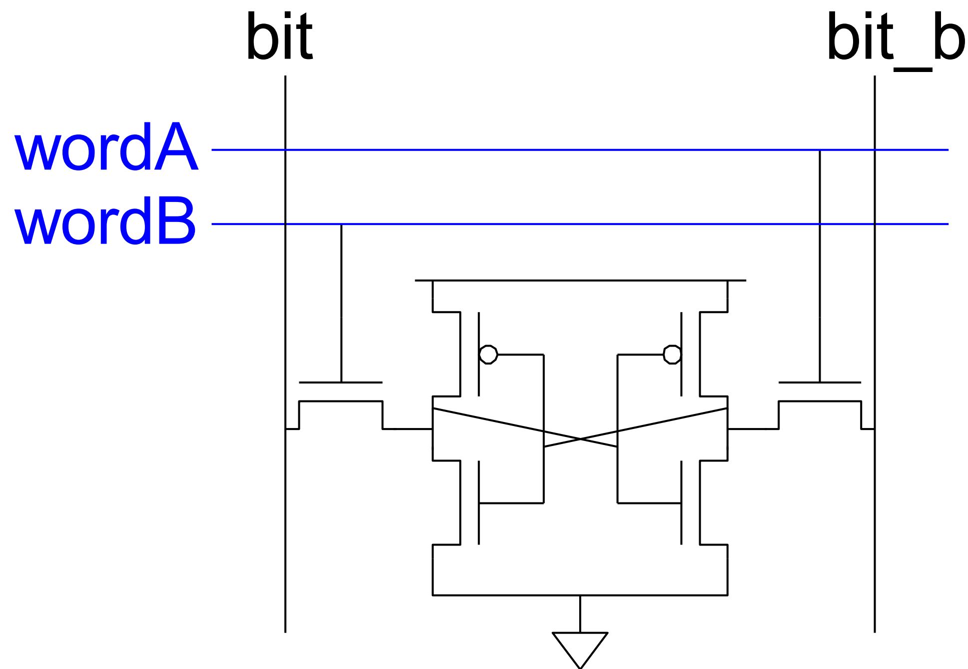
# Multiple Ports

- We have considered single-ported SRAM
  - One read or one write on each cycle
- *Multiported* SRAM are needed for register files
- Examples:
  - Pipelined ALU register file:
    - add r1,r2,r3
    - R3←R1+R2
    - Requires two reads and one write



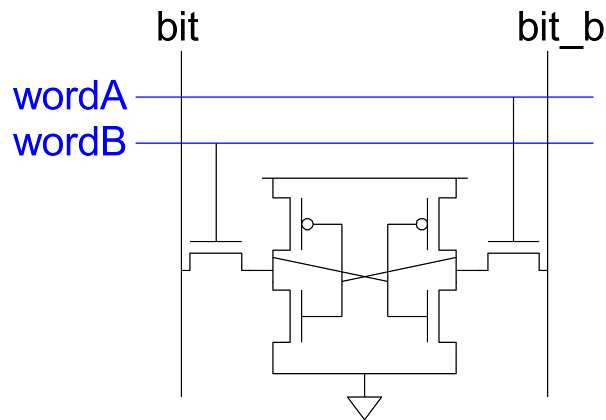
# Dual-Ported SRAM

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



# Dual-Ported SRAM

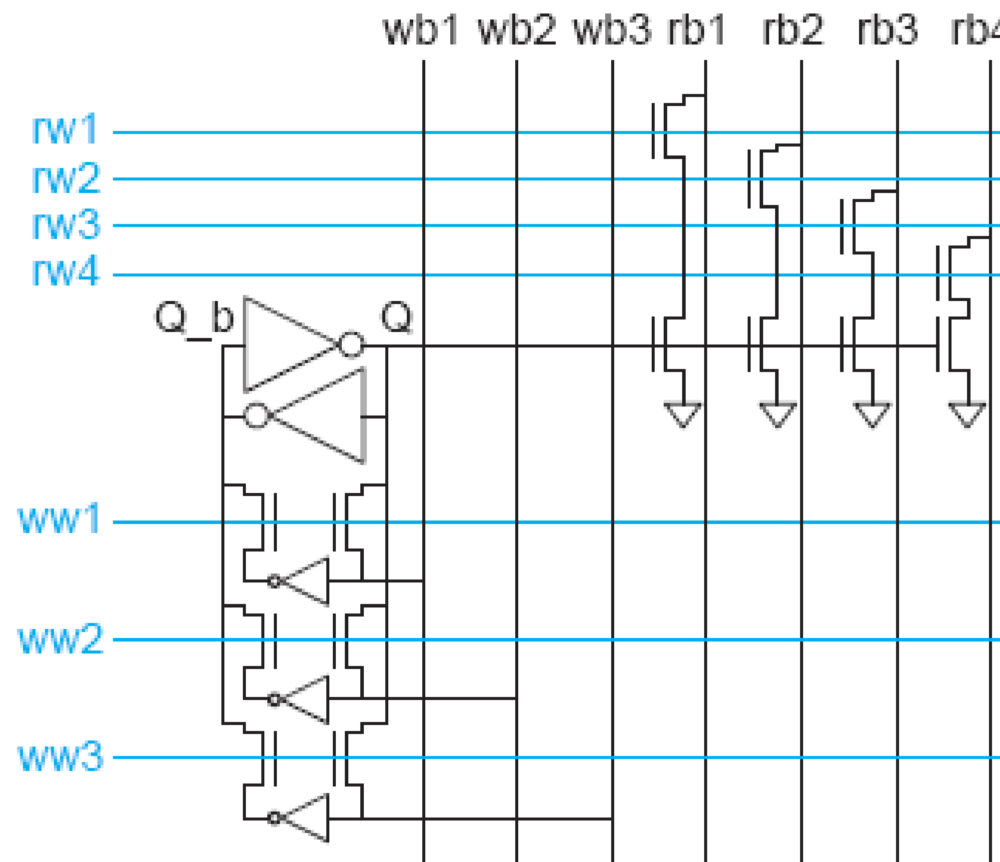
- Simple dual-ported SRAM
  - Two independent single-ended reads
  - Or one differential write




- 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 bitlines save area

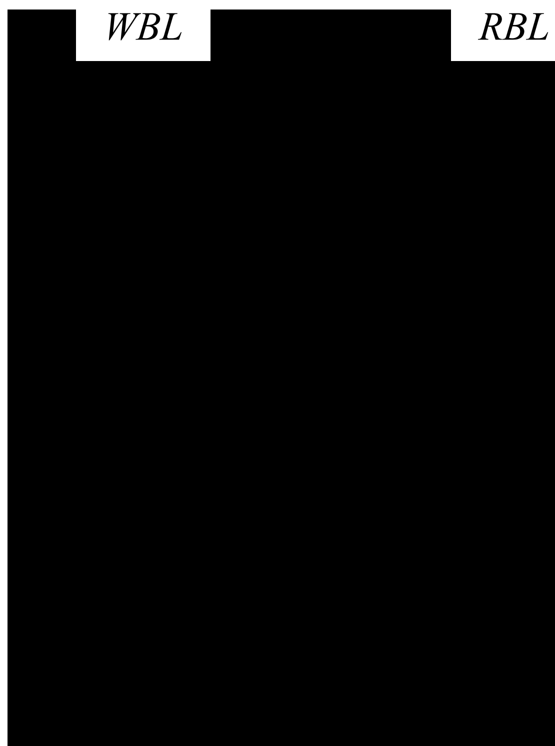


# DRAM

- Smaller than SRAM
- Require data refresh to compensate for leakage

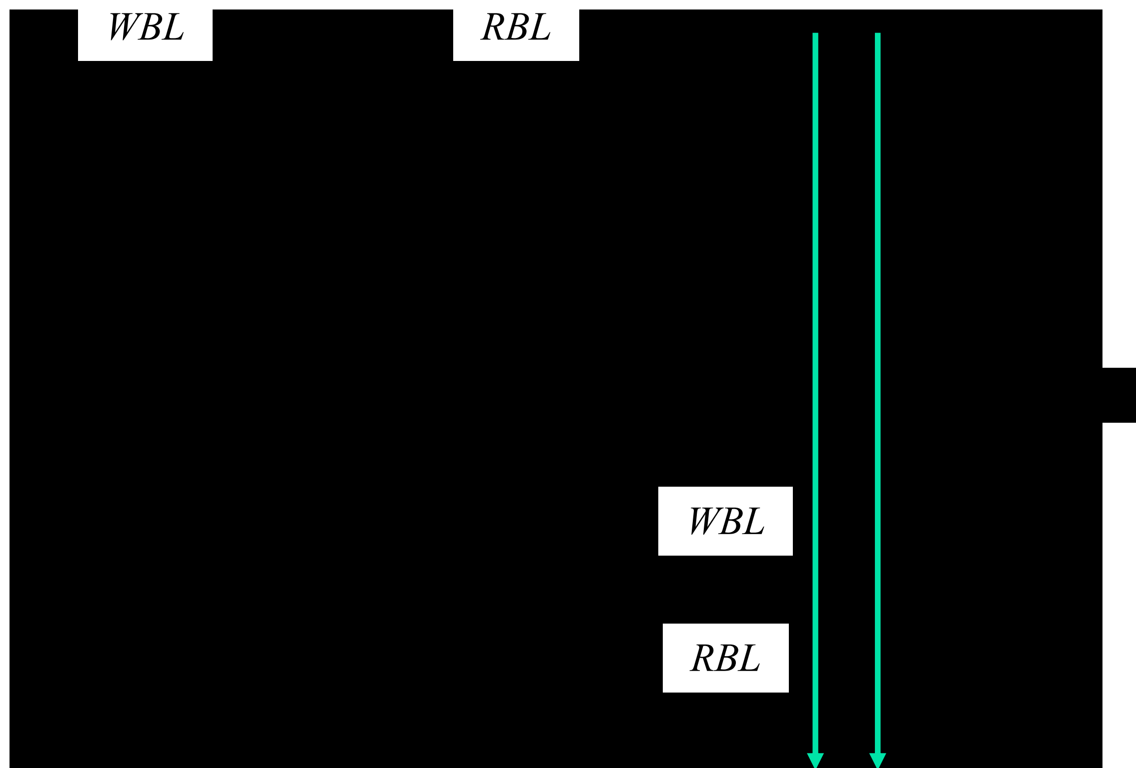
# 3-Transistor DRAM Cell (preclass 4)

- Cell is inverting on read operation



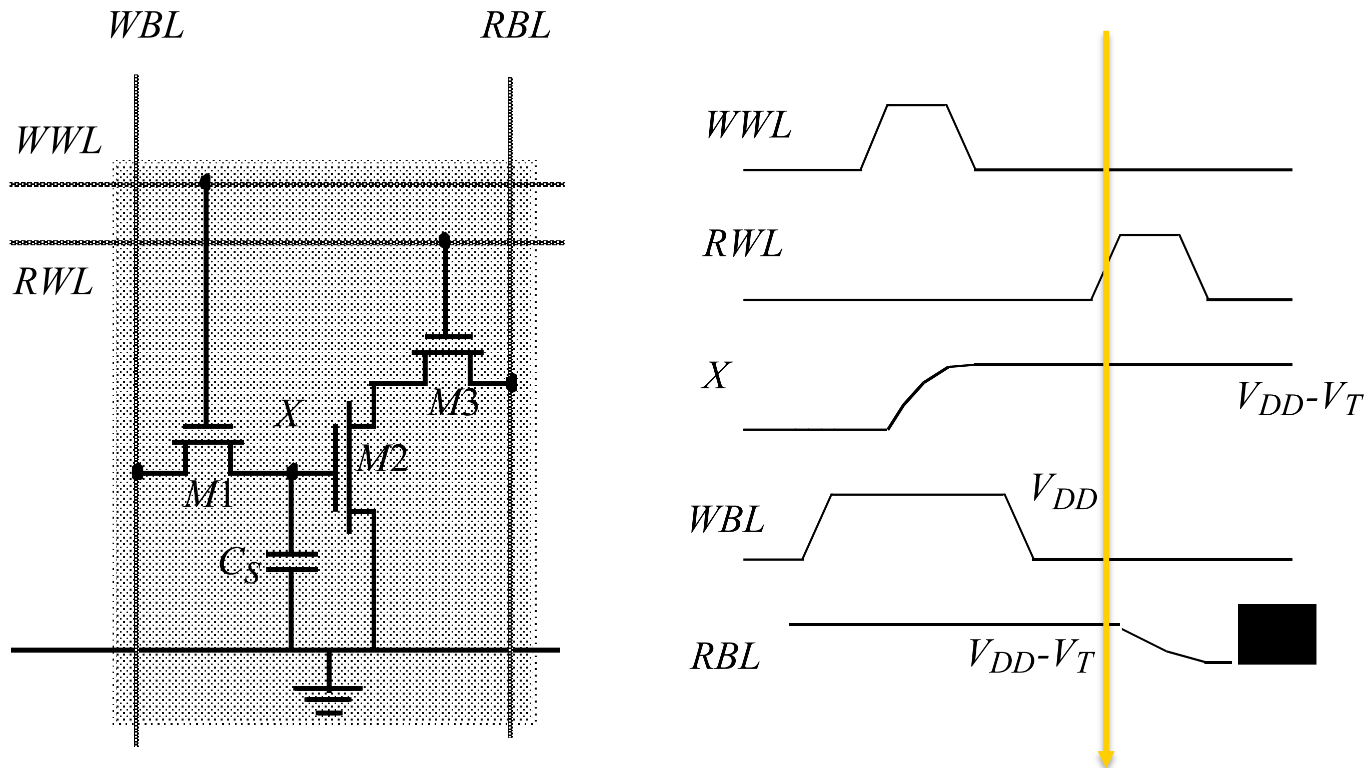
# 3-Transistor DRAM Cell

- Cell is inverting on read operation



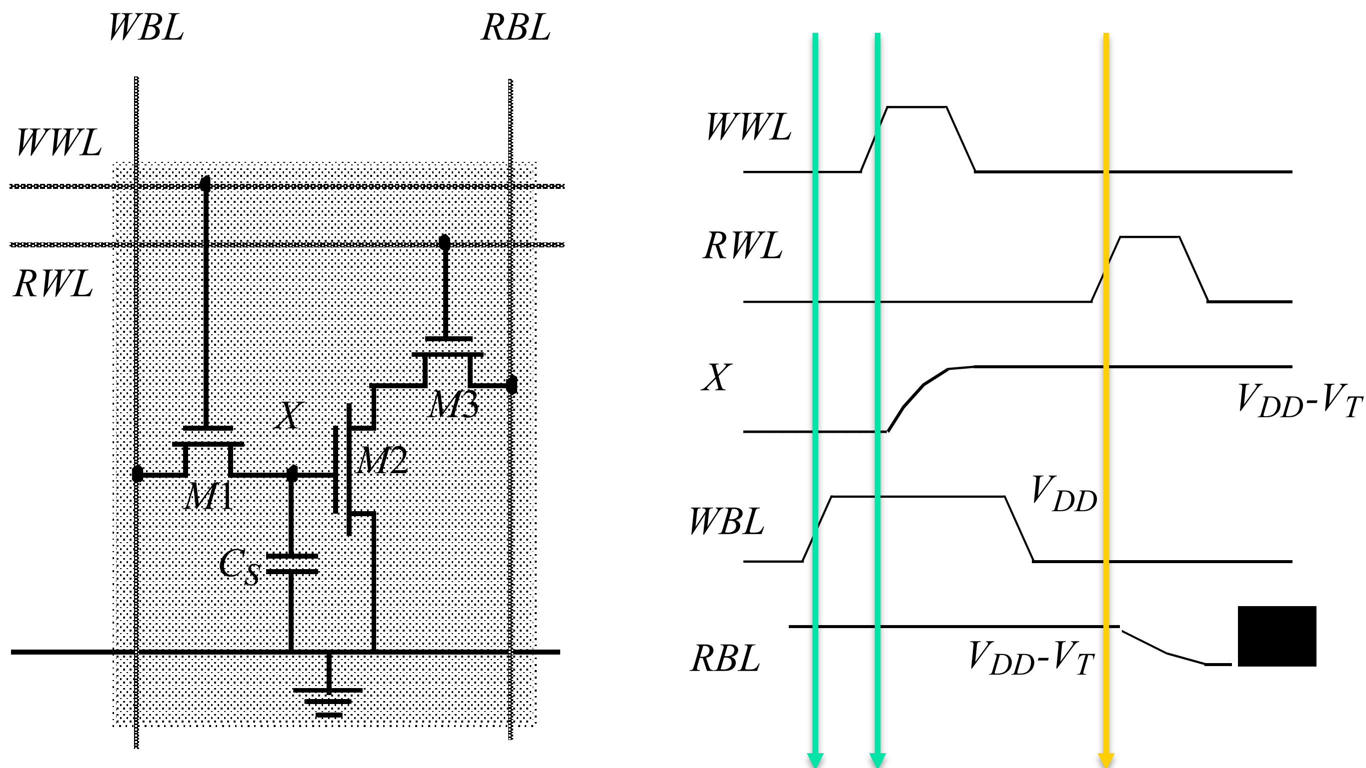
# 3-Transistor DRAM Cell

- Cell is inverting on read operation



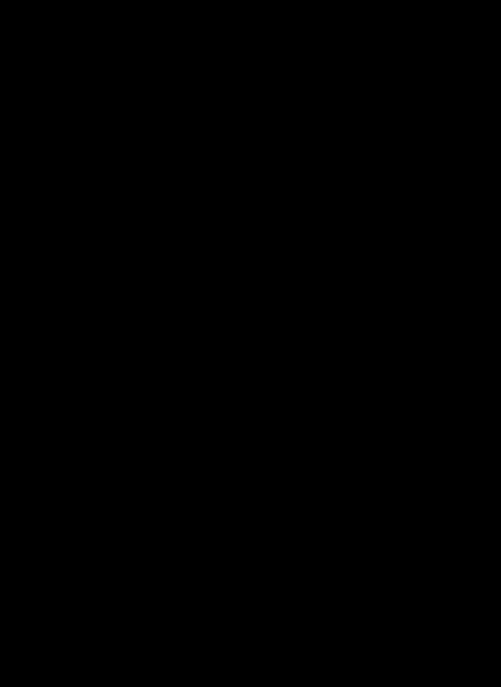
# 3-Transistor DRAM Cell

- Cell is inverting on read operation

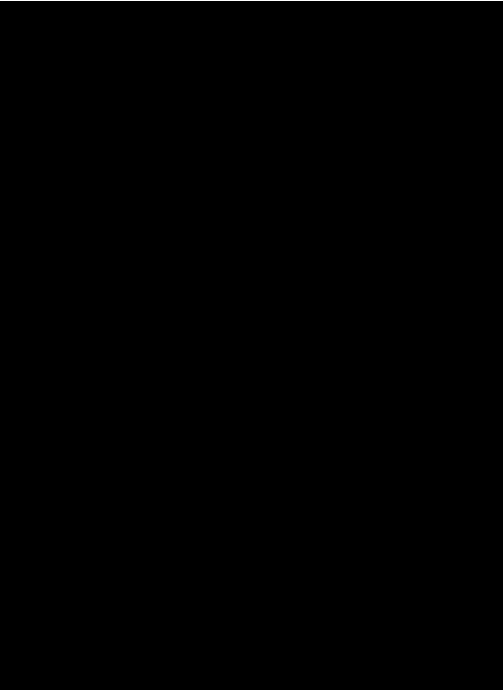


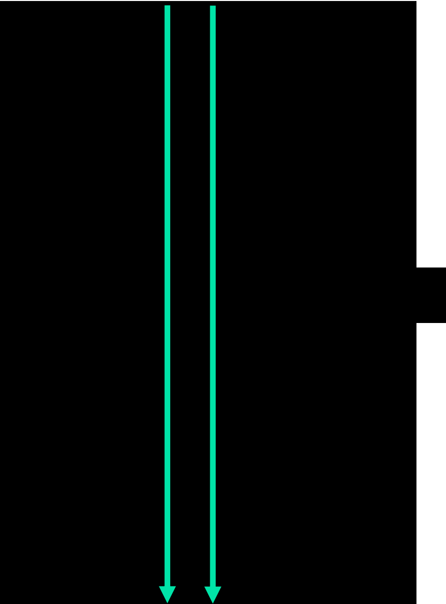

- No constraints on device ratios
- Reads are non-destructive
- Data stored has a $V_T$ drop
  - When storing a 1, value at X = $V_{WWL}$-$V_{Tn}$

# 1-Transistor DRAM Cell (preclass 4)

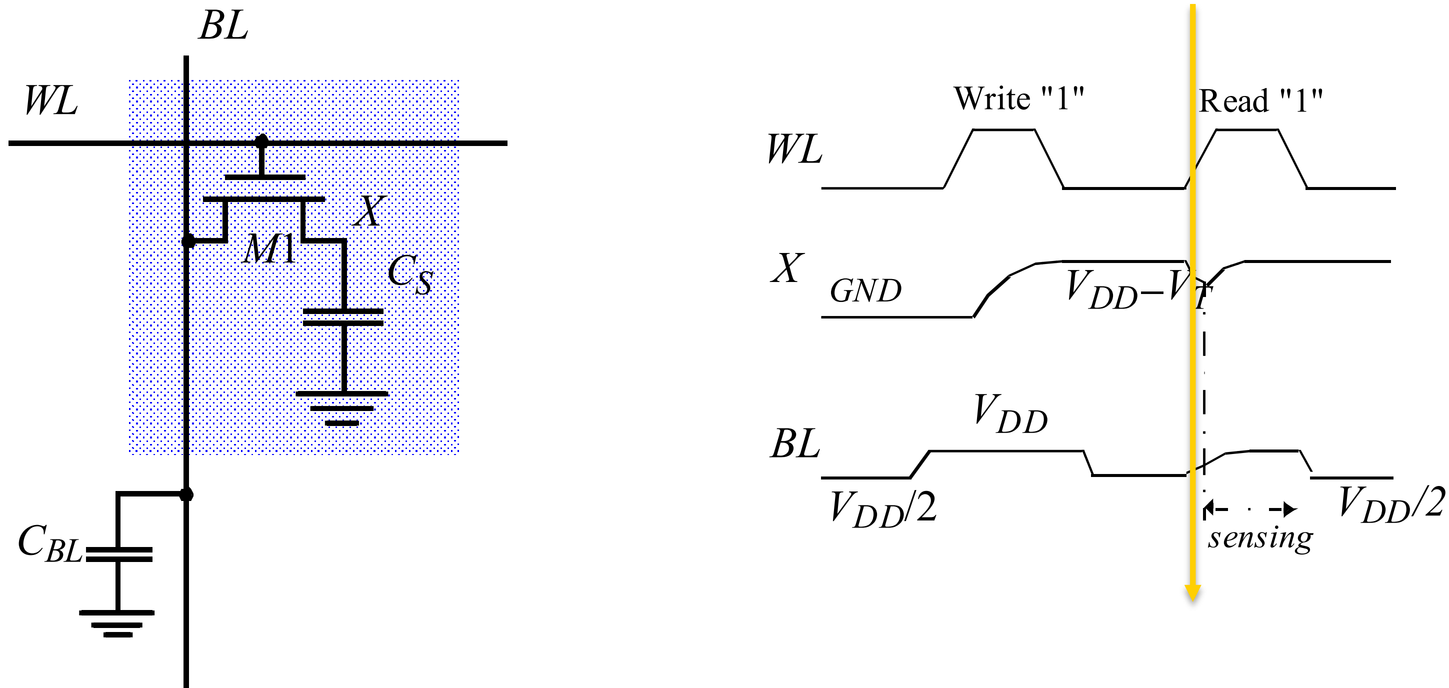


# 1-Transistor DRAM Cell





# 1-Transistor DRAM Cell




**Write: $C_S$ is charged or discharged by asserting WL and BL.**
**Read: Charge redistribution takes places between bit line and storage capacitance**

$$\Delta V = V_{BL} - V_{PRE} = (V_{BIT} - V_{PRE})\frac{C_S}{C_S + C_{BL}}$$

**Voltage swing is small; typically around 250 mV.**

# DRAM Cell Observations

- 1T DRAM requires a sense amplifier for each bit line, due to charge redistribution read-out
- DRAM memory cells are single ended in contrast to SRAM cells
- The read-out of the 1T DRAM cell is destructive; read and refresh operation are necessary for correct operation
- Unlike 3T cell, 1T cell requires presence of an extra capacitance that must be explicitly included in the design
- When writing a "1" into a DRAM cell, a threshold voltage is lost. This loss can be circumvented by bootstrapping the word lines to a higher value than $V_{DD}$.

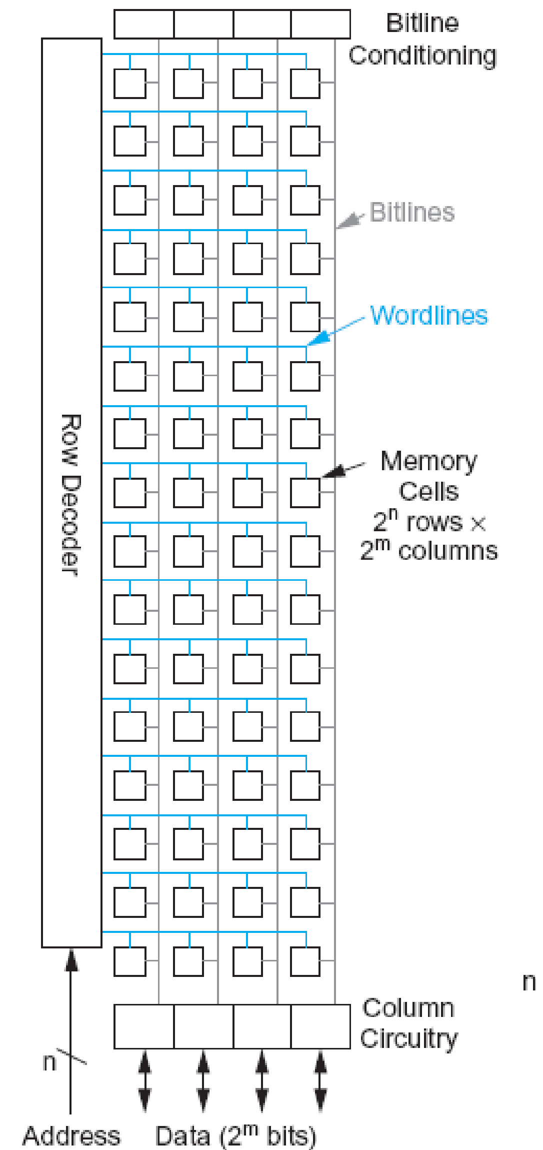
# Memory Periphery

# Periphery

- Decoders
- Column Circuitry
  - Bit-line Conditioning
  - Sense Amplifiers
  - Input/Output Buffers
- Control/Timing Circuitry

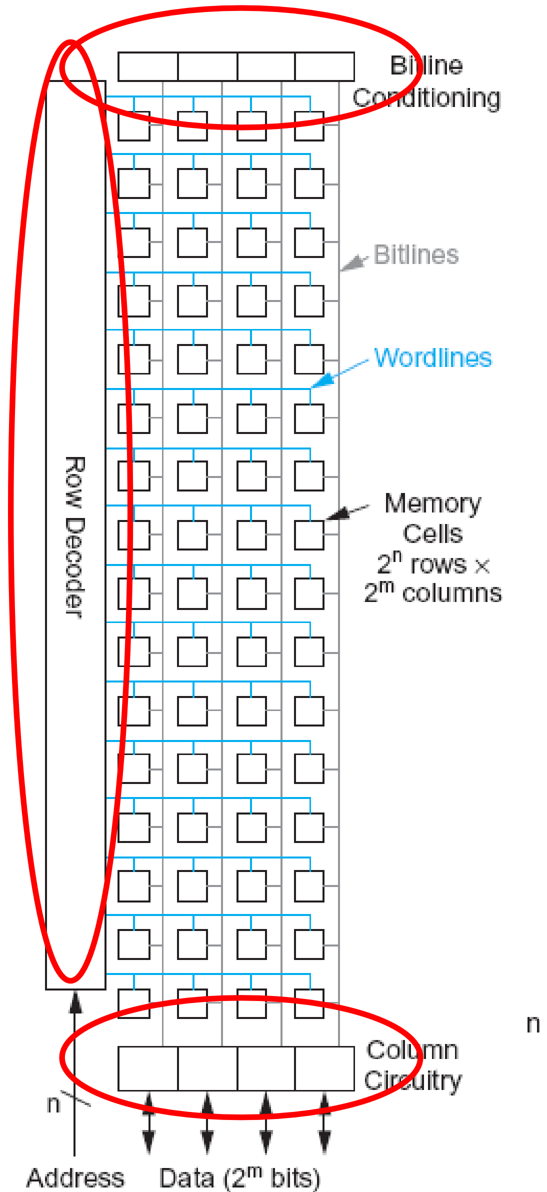
# Array Architecture

- $2^n$ *words* of $2^m$ *bits* each
- Good regularity – easy to design
- Very high density if good cells are used



# Array Architecture

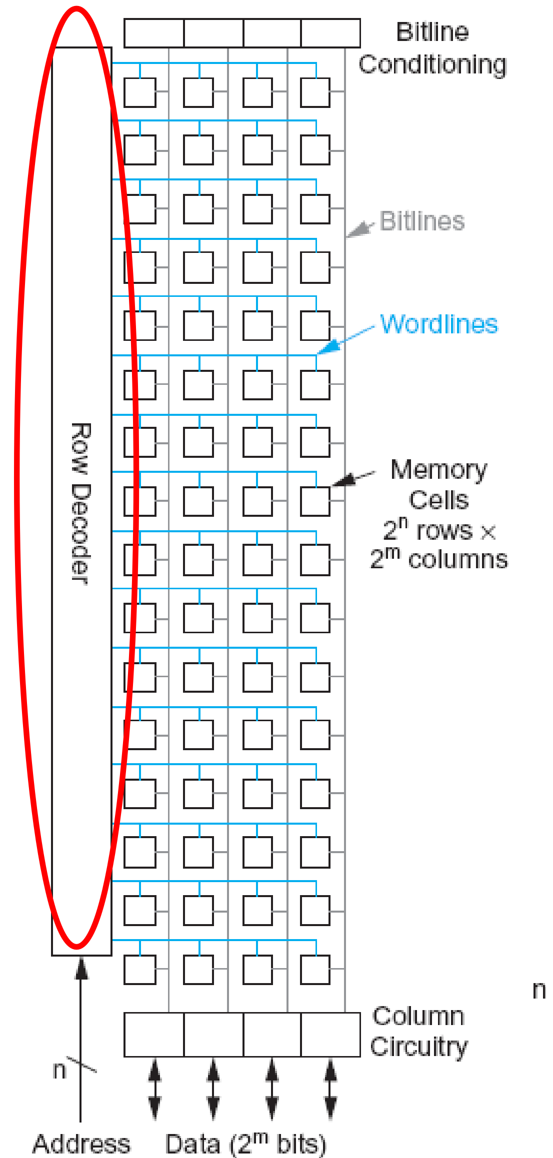
- $2^n$ *words* of $2^m$ *bits* each
- Good regularity – easy to design
- Very high density if good cells are used



# Decoders

# Array Architecture

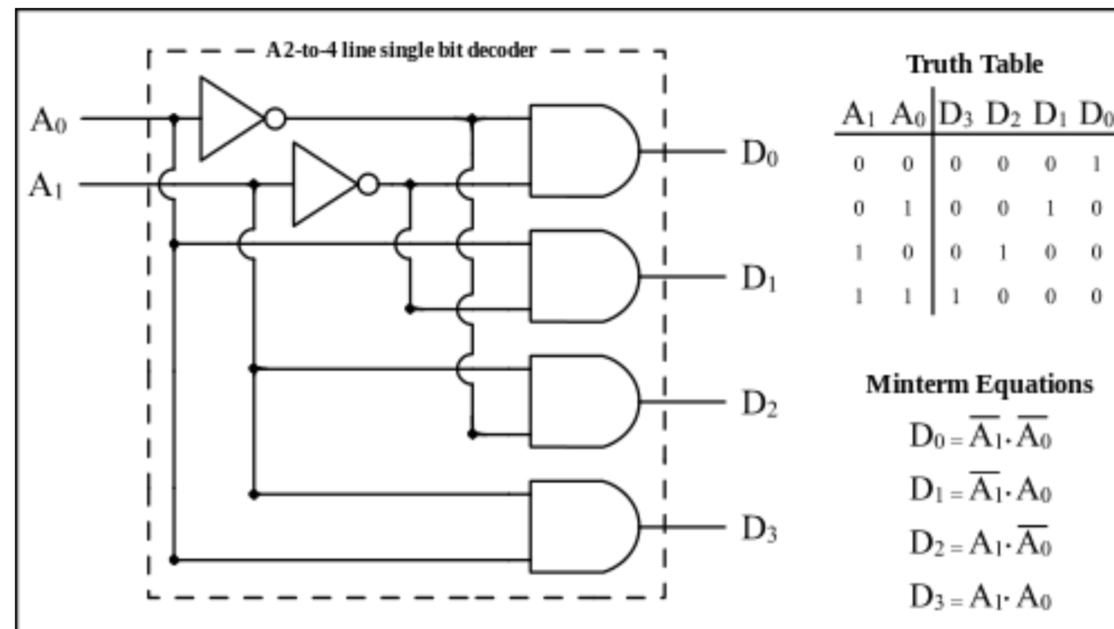
- $2^n$ *words* of $2^m$ *bits* each
- Good regularity – easy to design
- Very high density if good cells are used



# 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

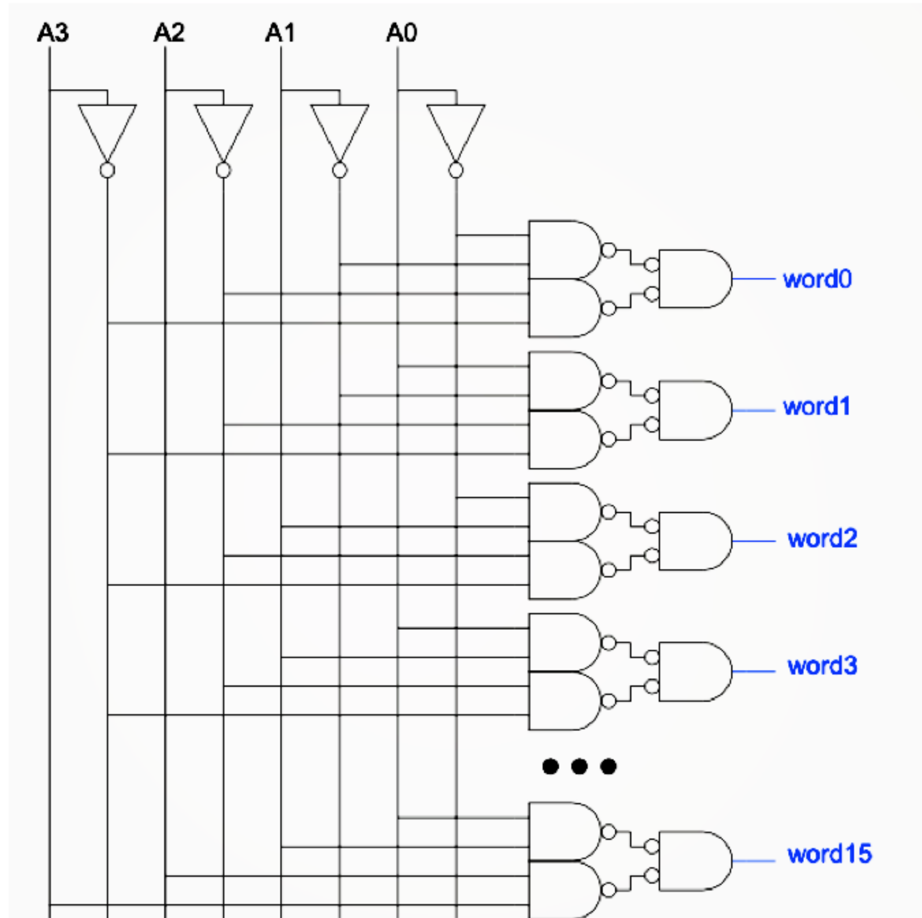
Static CMOS




| $A_1$ | $A_0$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

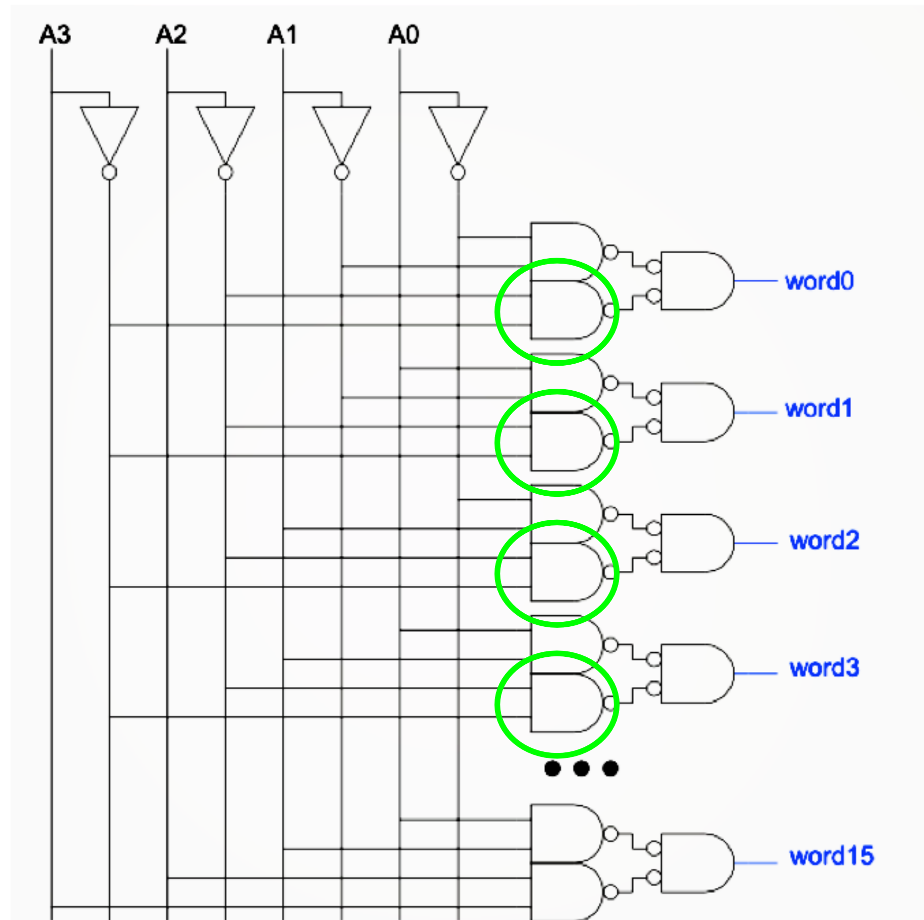
# Large Decoders

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



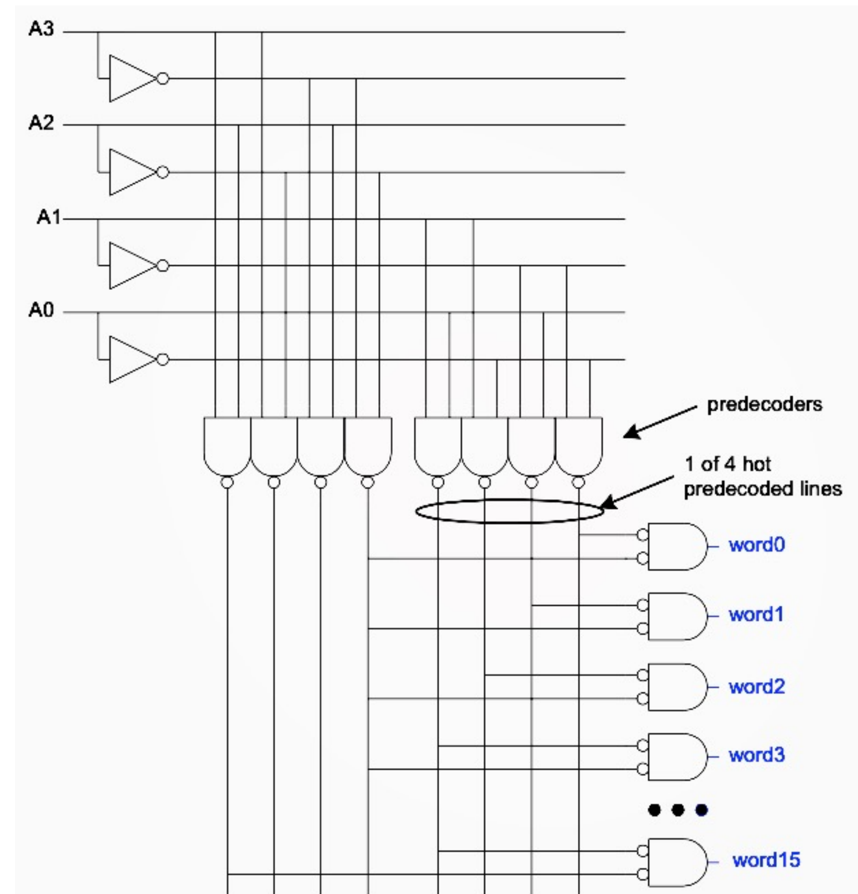
# Large Decoders

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

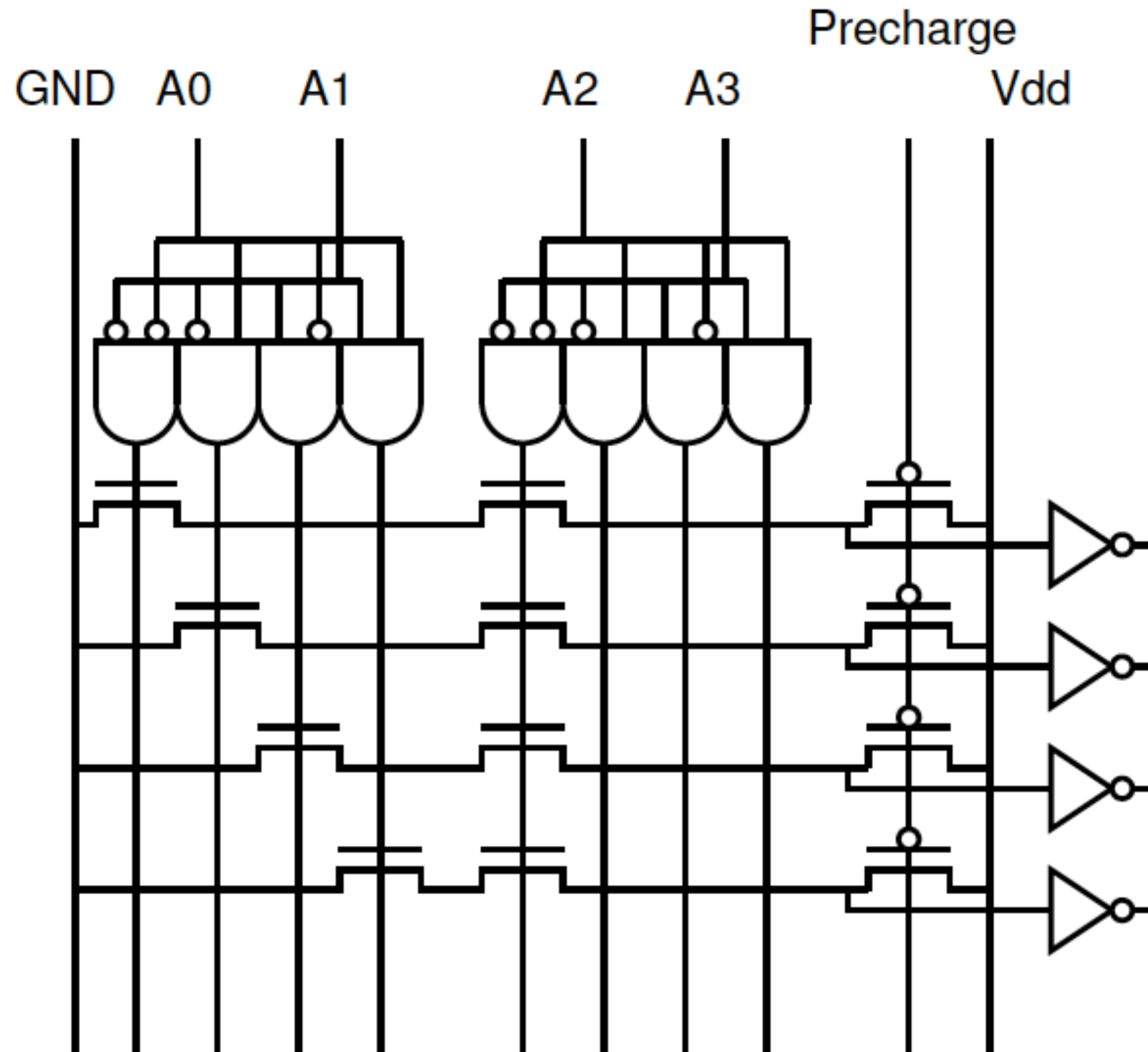


# Predecoding

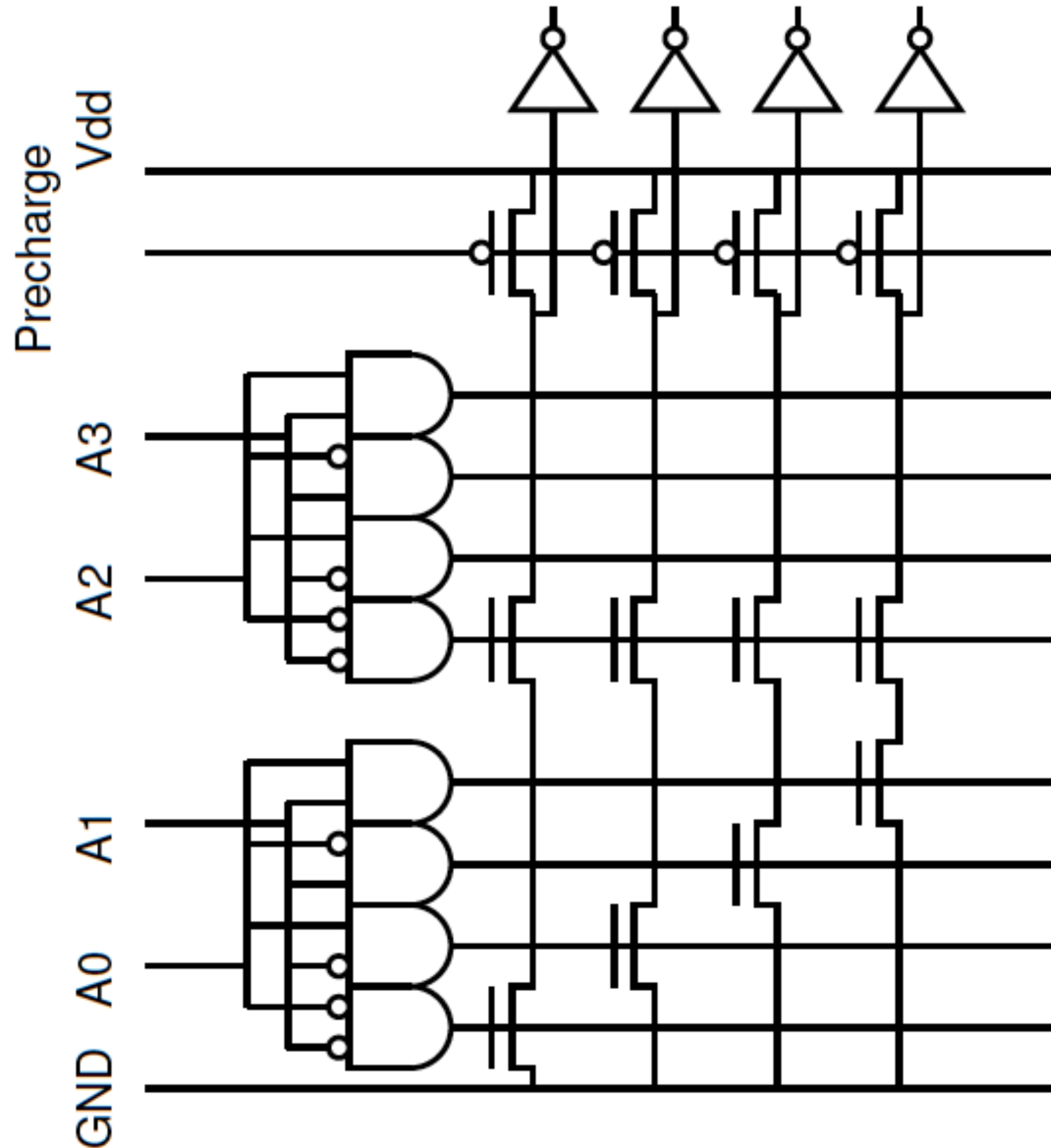
- Many of these gates are redundant
  - Factor out common gates into predecoder
  - Saves area
  - Same path effort



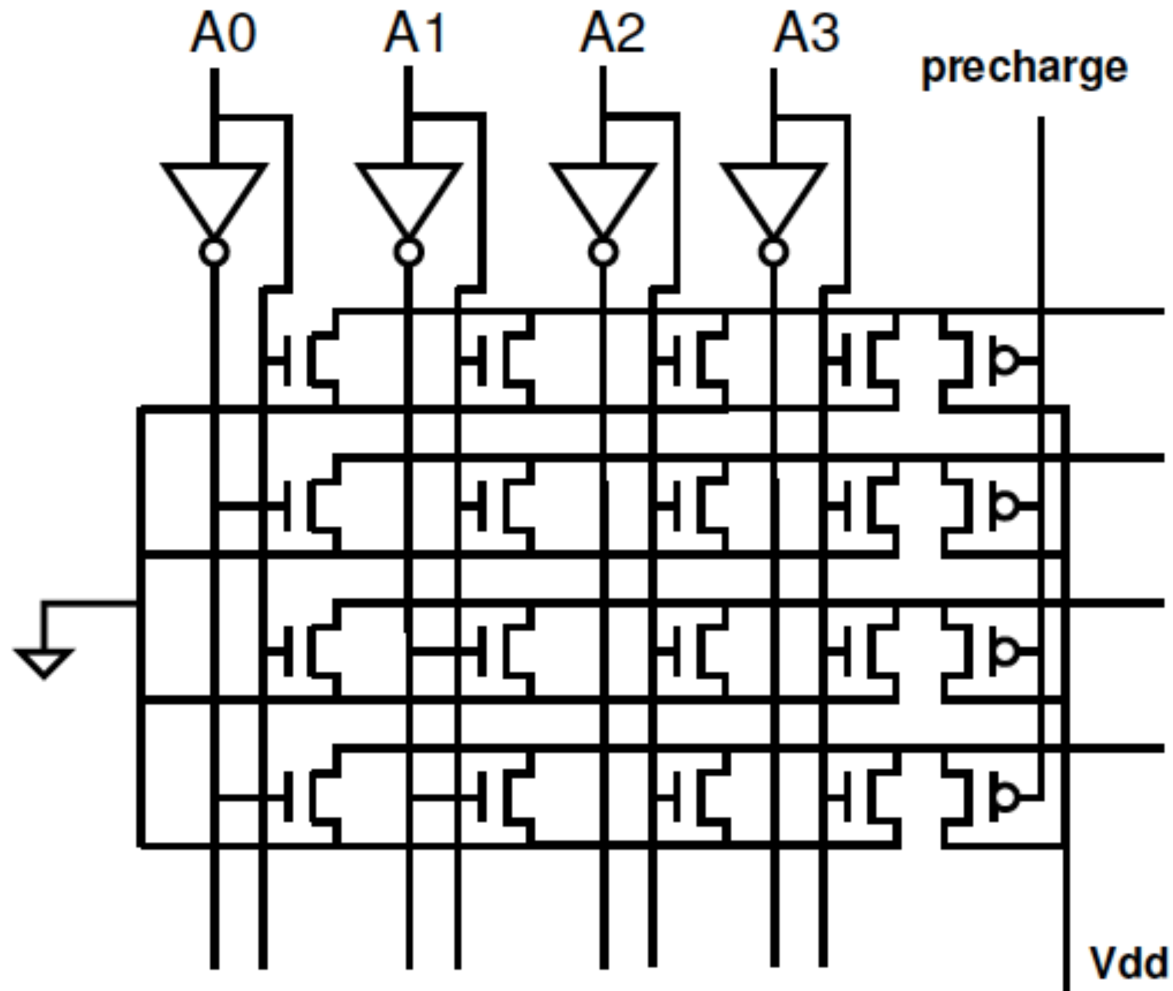
# Row Select: Precharge NAND



# Row Select: Precharge NAND



# Row Select: Precharge NOR

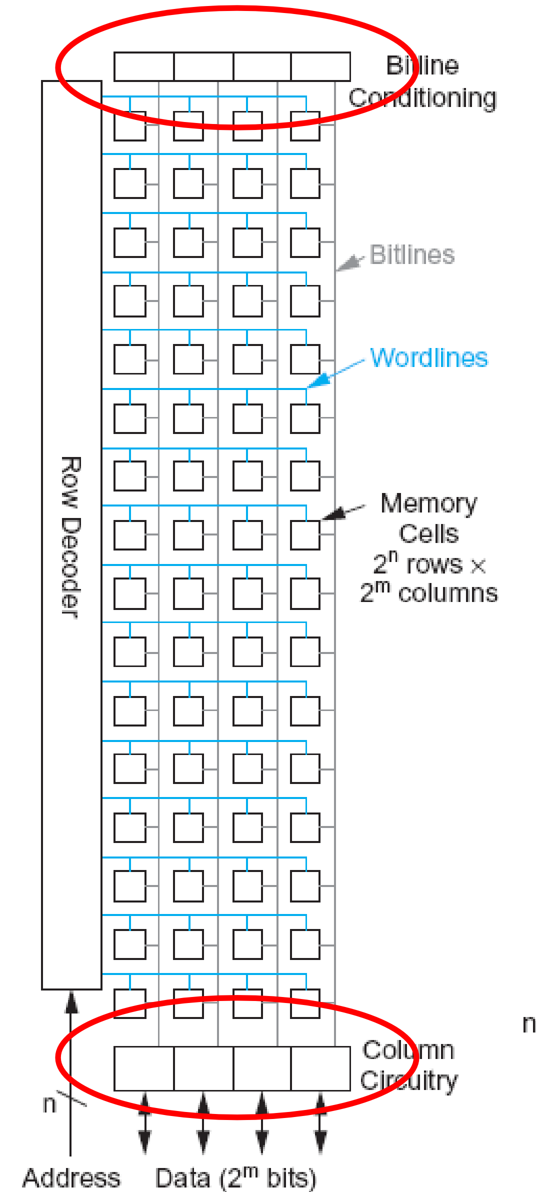


# Column Circuitry

## & Bit-line Conditioning

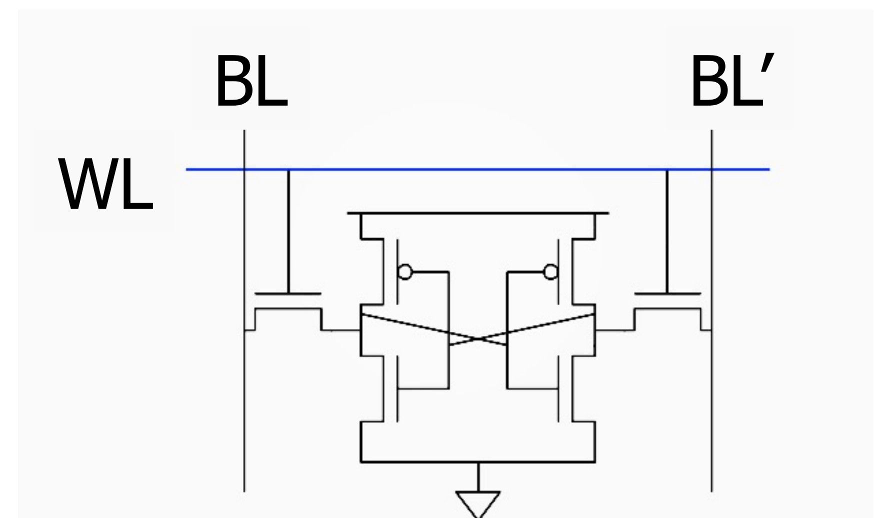
# Array Architecture

- $2^n$ *words* of $2^m$ *bits* each
- Good regularity – easy to design
- Very high density if good cells are used



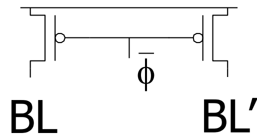
# 6T SRAM Cell

- 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 BL, BL'
  - Raise WL
- Write:
  - Drive data onto BL, BL'
  - Raise WL



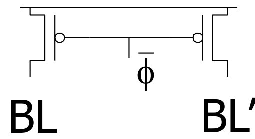
# Bitline Conditioning

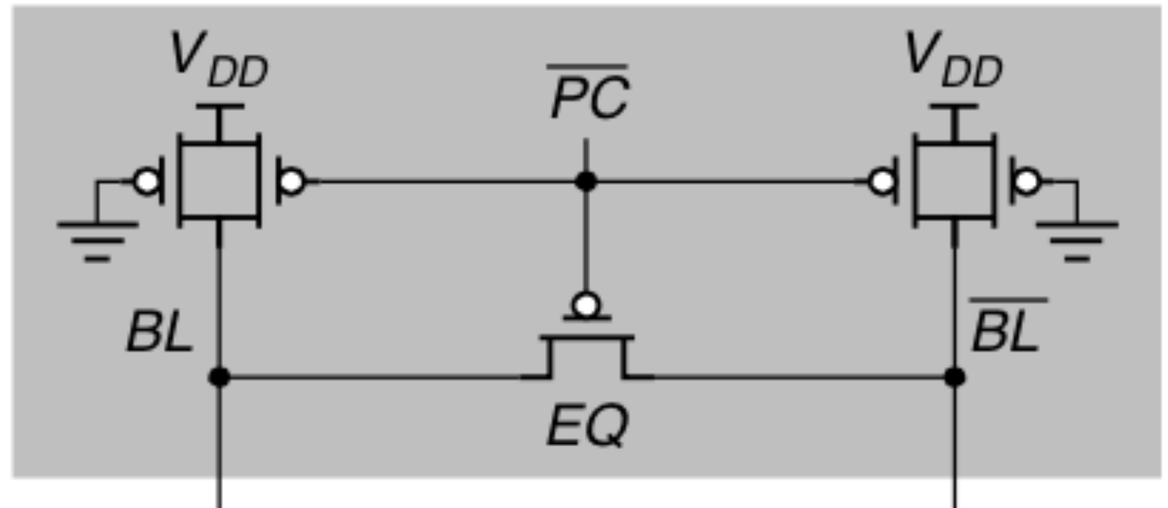
- Precharge bitlines high before read operations



# Bitline Conditioning

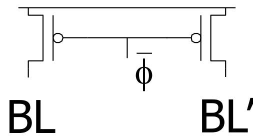
- Precharge bitlines high before reads

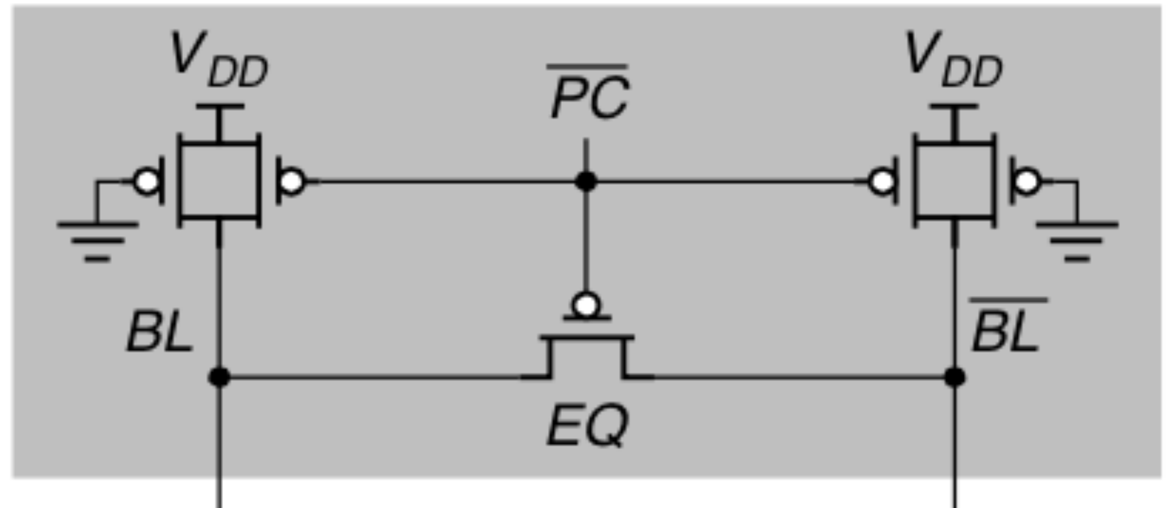


# Bitline Conditioning

- Precharge bitlines high before reads

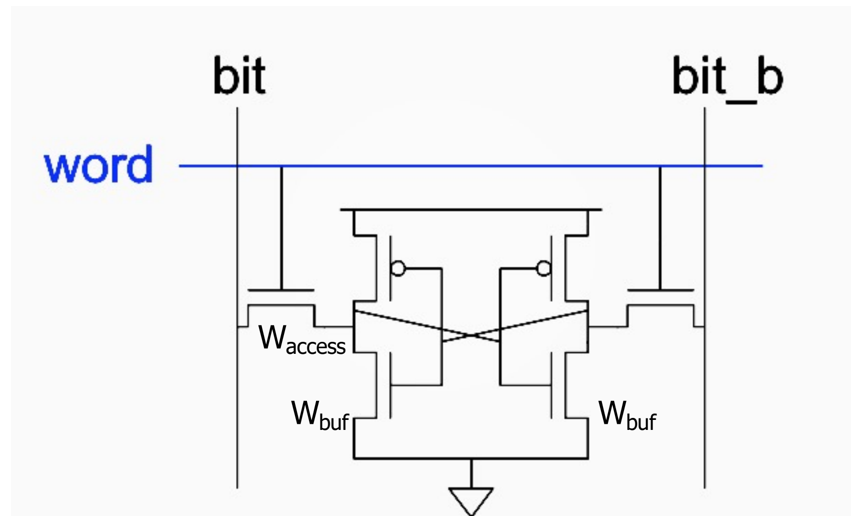


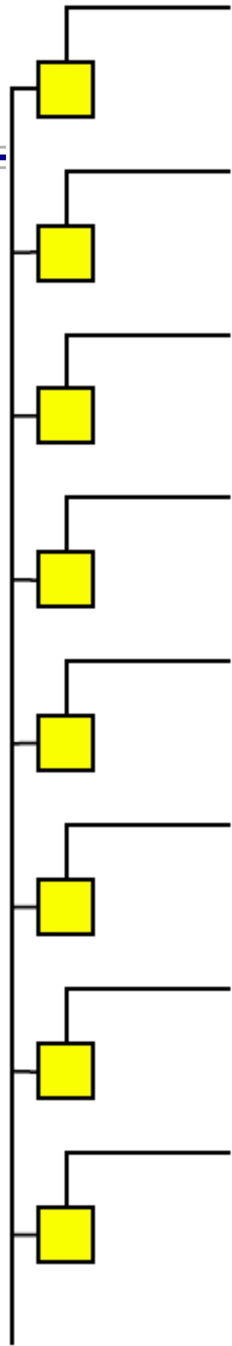




- What if pre-charged to Vdd/2?
  - Pros: reduces read-upset
  - Challenge: generate Vdd/2 voltage on chip

# Column Capacitance Consequence

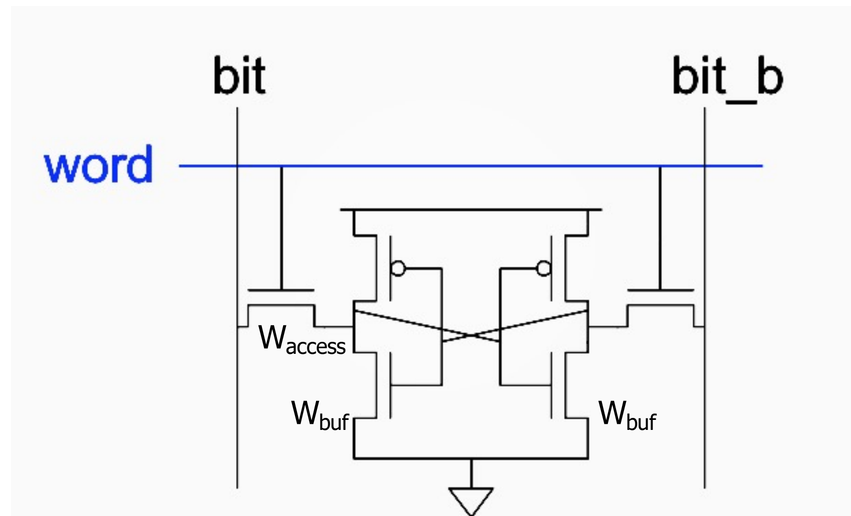
- Preclass 5: What is capacitance of a bitline?
  - $W_{access}$ (pass transistor size), d rows, $\gamma = C_{diff0}/C_0$

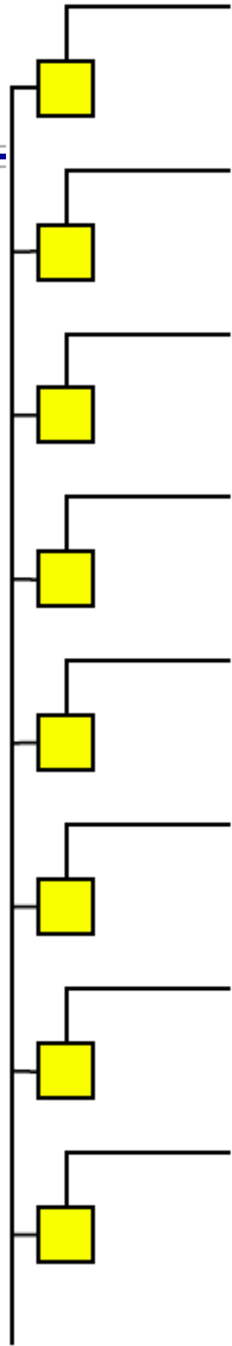


# Column Capacitance Consequence

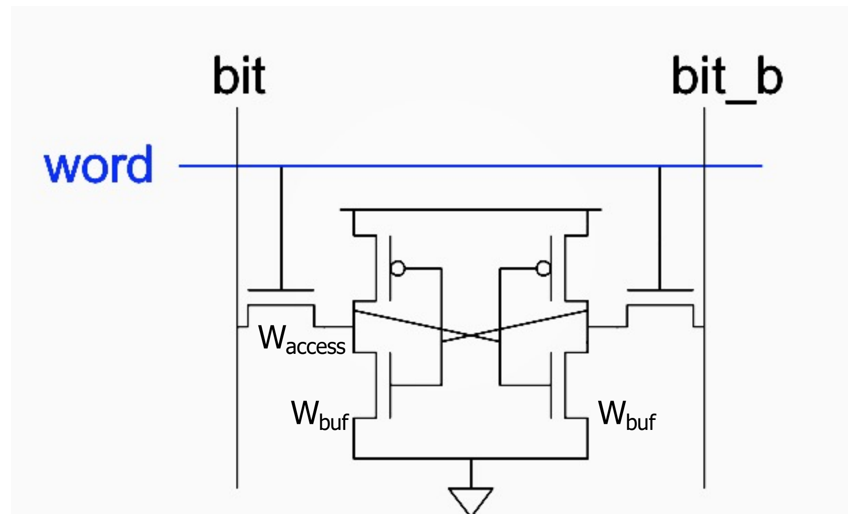
- Preclass 5: What is capacitance of a bitline?
  - $W_{access}$ (pass transistor size), d rows, $\gamma = C_{diff0}/C_0$
- Preclass 6: What is the delay for the cell to drive the bitline during a read?
  - $W_{buf}$ (inverter size in cell), $R_0$

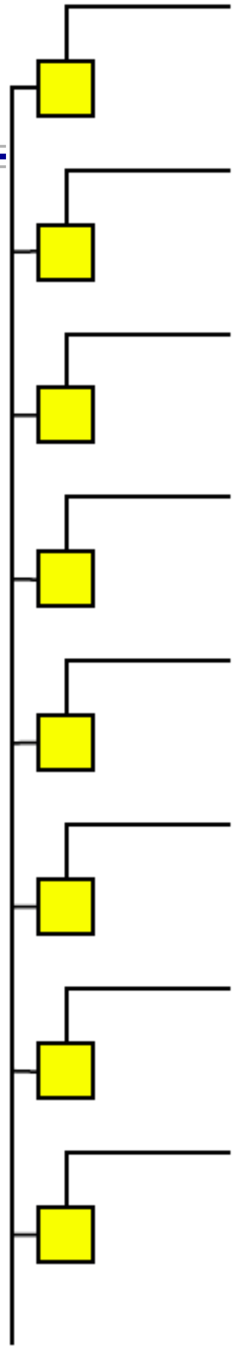


# Column Capacitance Consequence

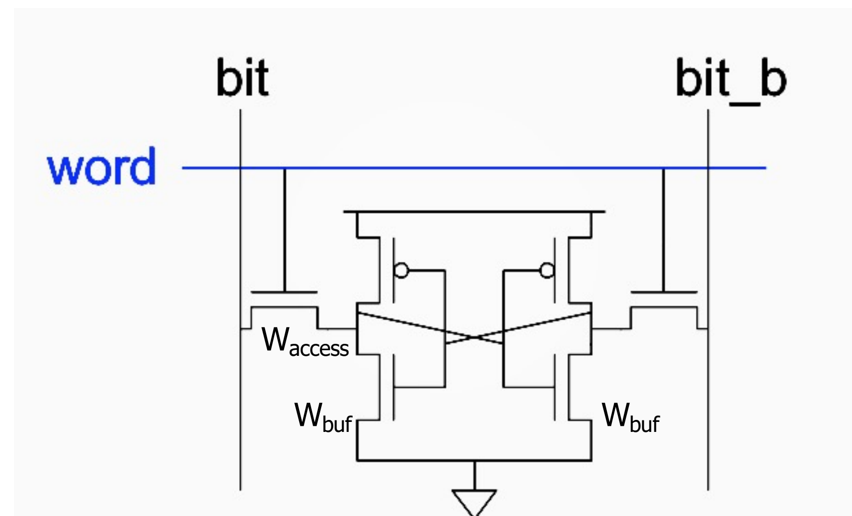
- Preclass 5: What is capacitance of a bitline?
  - $W_{access}$ (pass transistor size), d rows, $\gamma=C_{diff0}/C_0$
- Preclass 6: What is the delay for the cell to drive the bitline during a read?
  - $W_{buf}$ (inverter size in cell), $R_0$
- Preclass 7: Waccess=Wbuf=1, $\gamma=1/2$
  - Delay for d=32, 512?

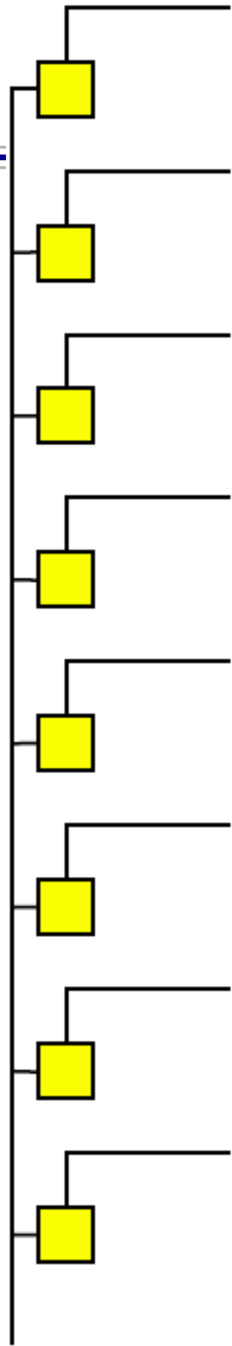


# Column Capacitance Consequence

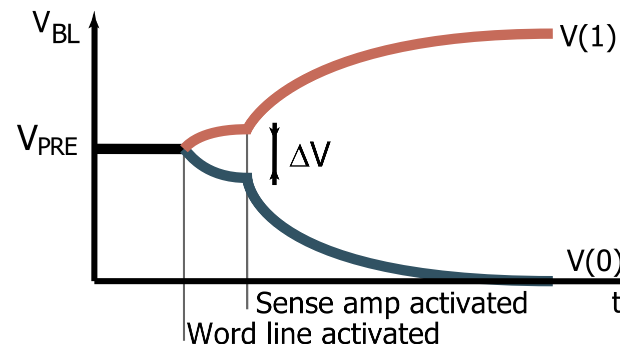
- Preclass 5: What is capacitance of a bitline?
  - $W_{access}$ (pass transistor size), d rows, $\gamma = C_{diff0}/C_0$
- Preclass 6: What is the delay for the cell to drive the bitline during a read?
  - $W_{buf}$ (inverter size in cell), $R_0$
- **Conclude:** Can't size up cell→driving bitline will be slow



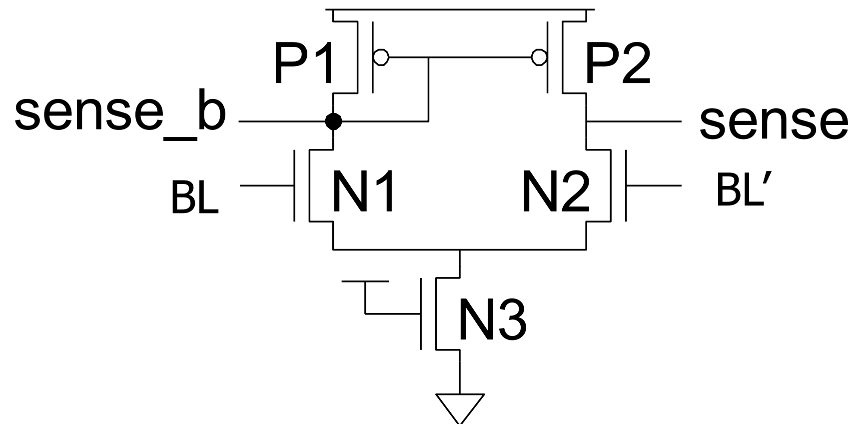
# Sense Amplifiers

- 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 in each memory cell (small I)
- *Sense amplifiers* are triggered on small voltage swing ($\Delta V$)



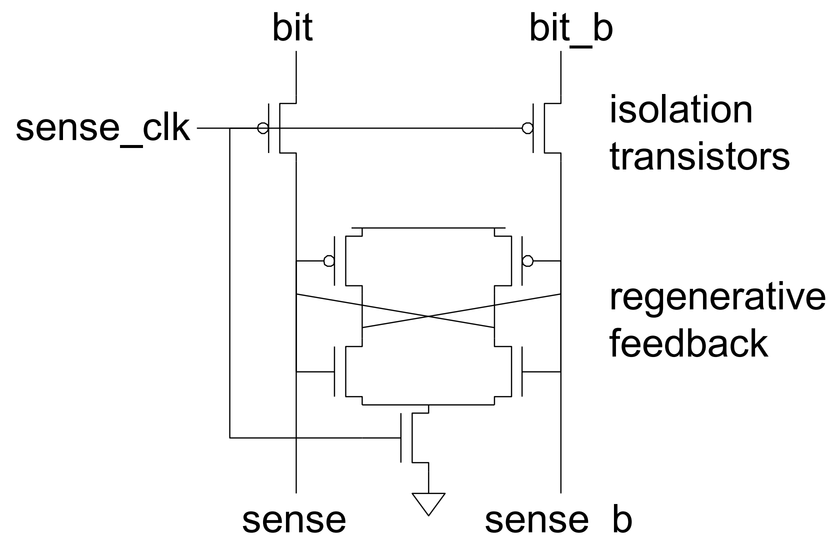
# Differential Pair Amp

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



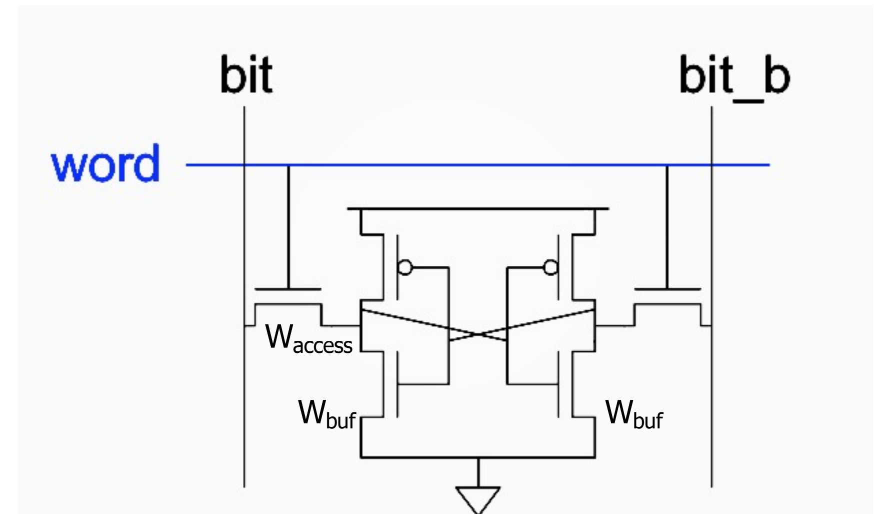
# Clocked Sense Amp

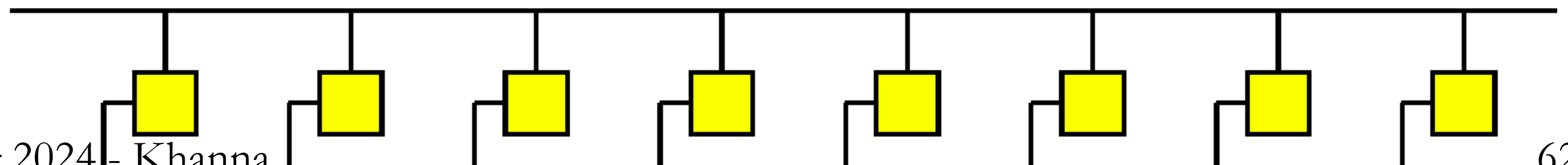
- Clocked sense amp saves power
- Requires sense_clk after enough bitline swing
- Isolation transistors cut off large bitline capacitance



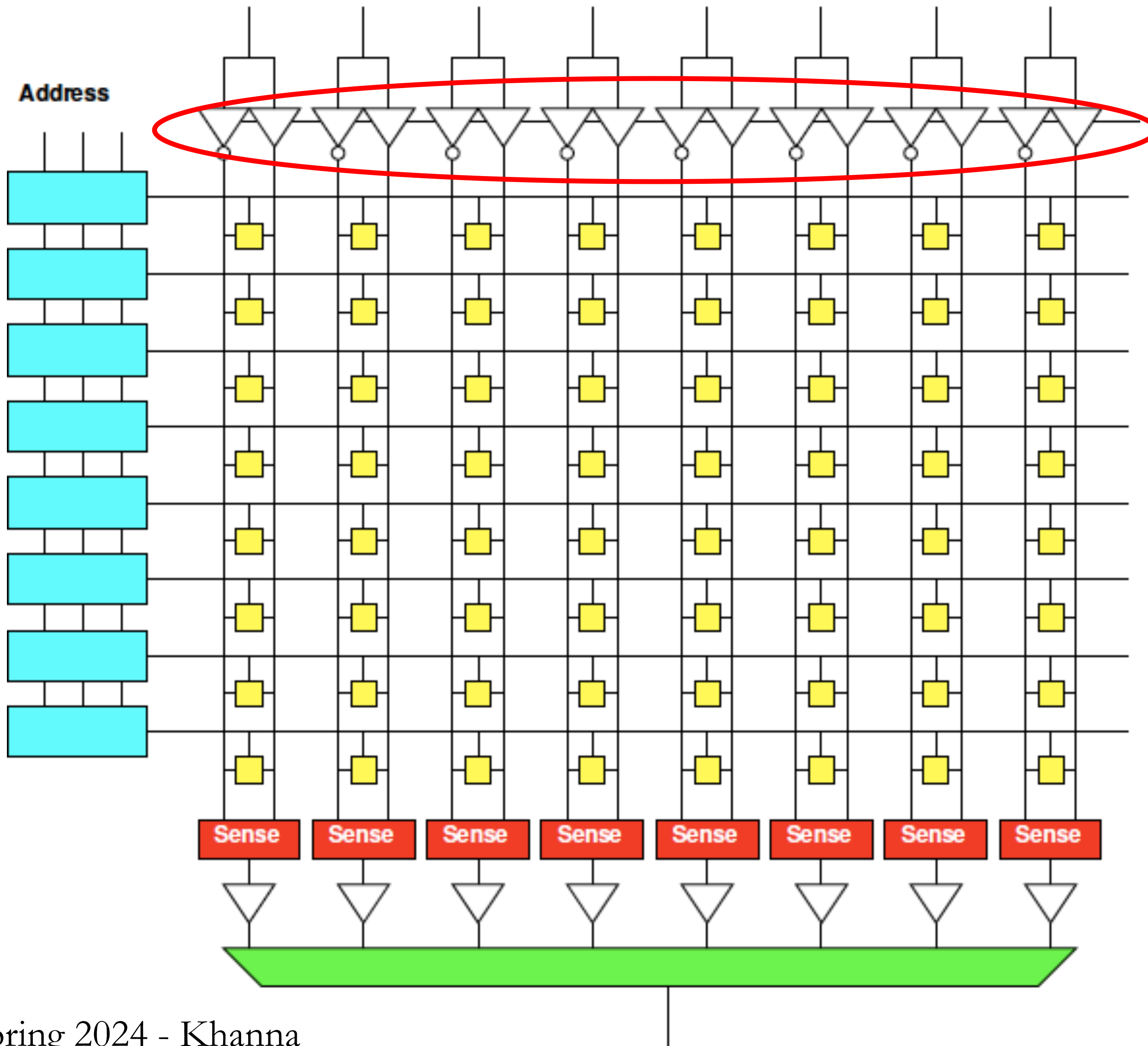
# Word Line Capacitance

- Preclass 8: What is capacitance of word line (row)?
  - $W_{access}$ – transistor width of column device
  - w columns
  - $\gamma = C_{diff0}/C_0$
- Preclass 9: Delay driving word line?
  - $W_{wldrive}$ Drive inverter



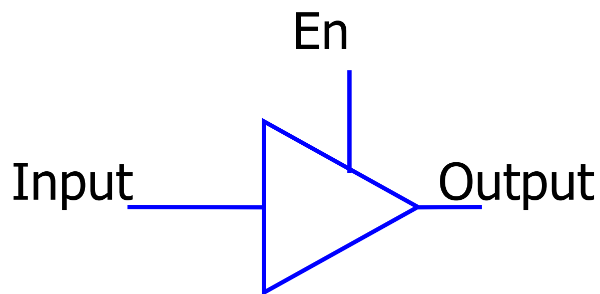



# Column Drivers: Memory Bank



# Tristate Buffer

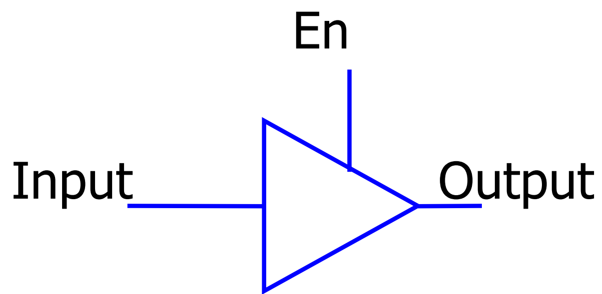
- Typically used for signal traveling, e.g. bus
- Ideally all devices connected to a bus should be disconnected except for active device reading or writing to bus
- Use high-impedance state to simulate disconnecting

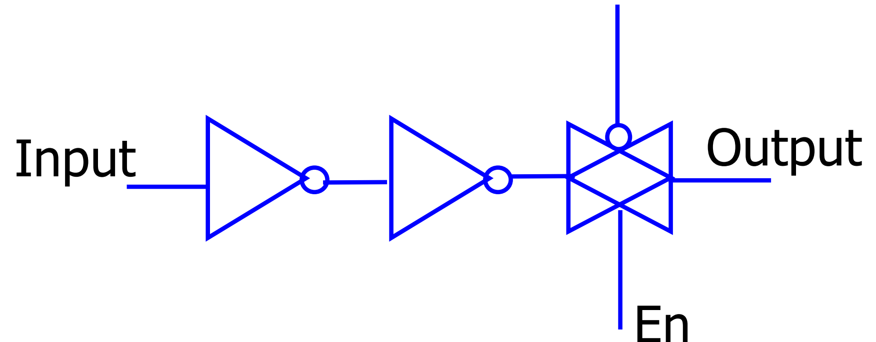


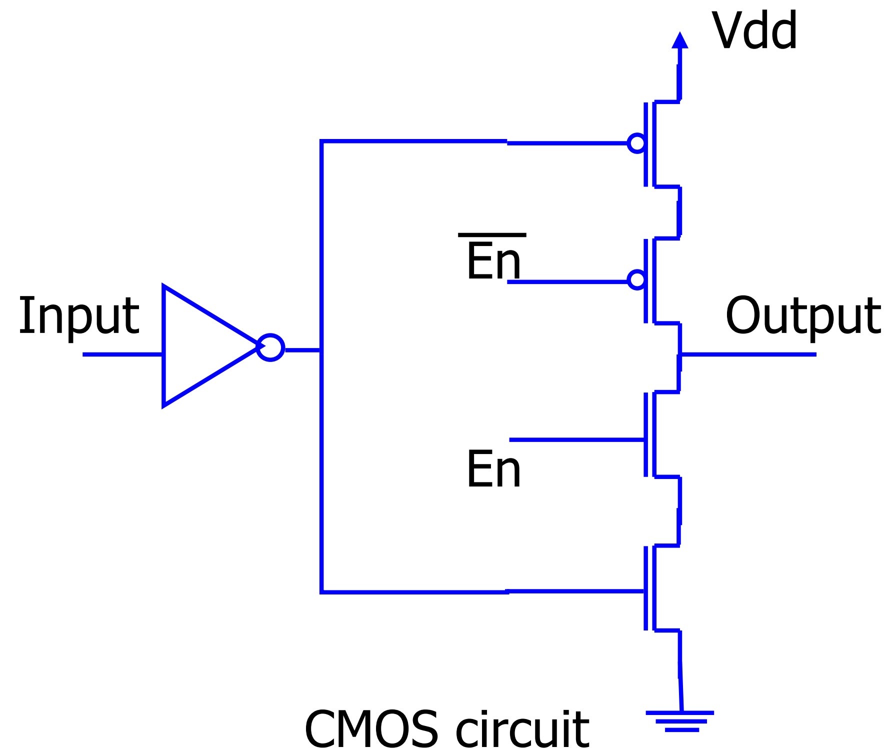

Active-high buffer

| Input | En | Ouptut |
|---|---|---|
| 0 | 0 | Z |
| 1 | 0 | Z |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

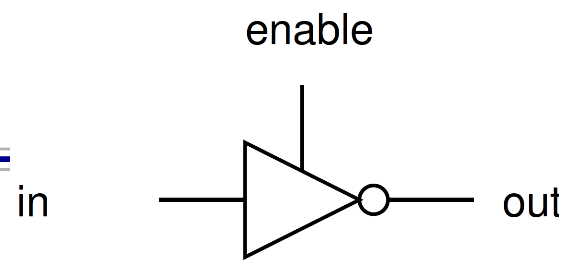
# Tristate Buffer

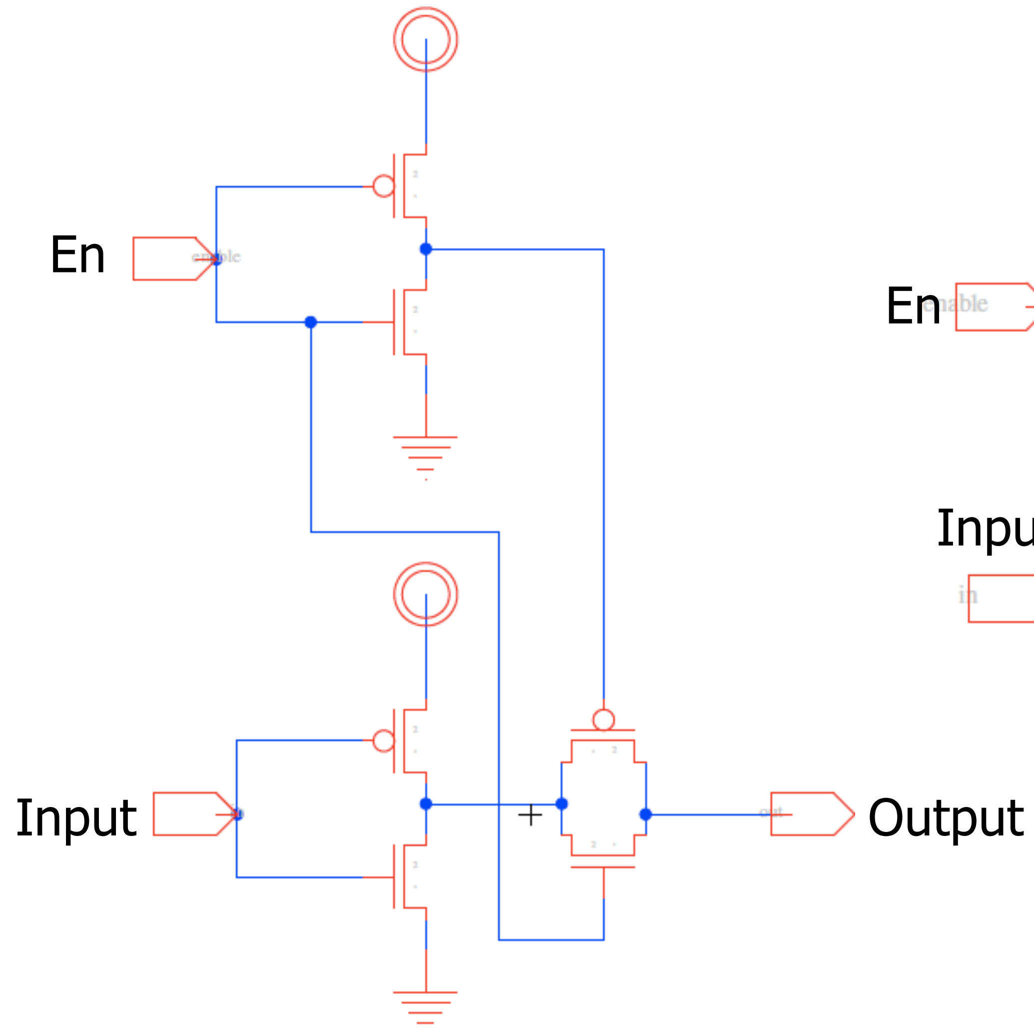


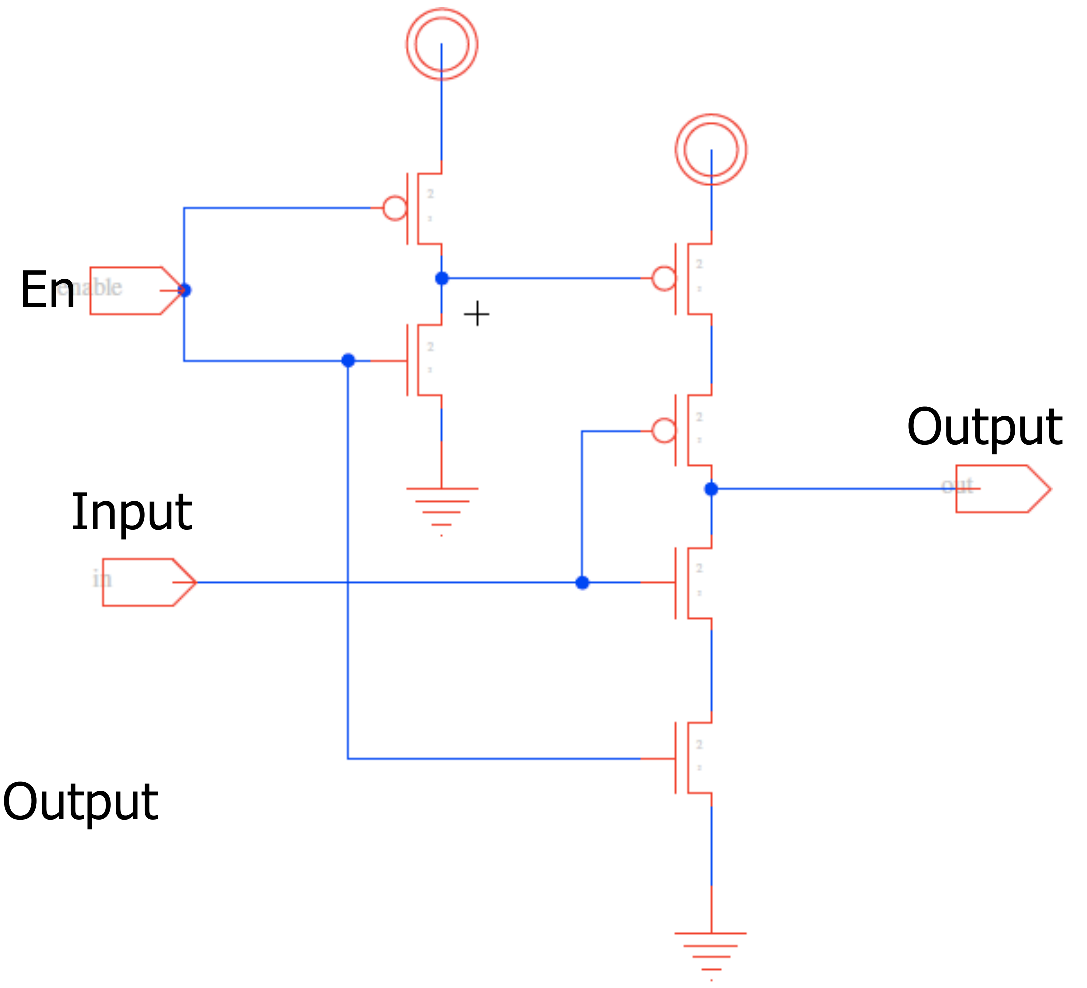



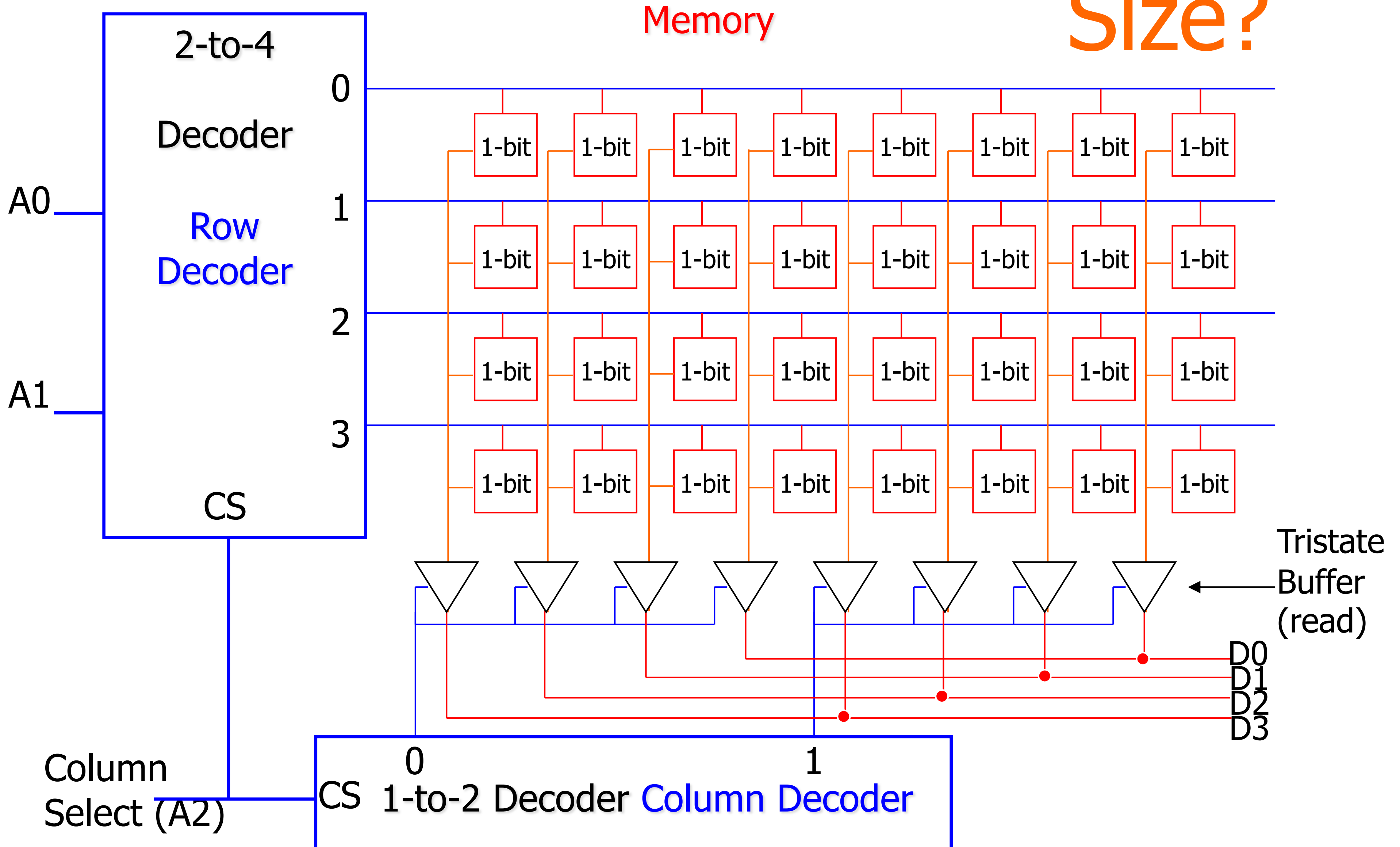
# Tristate Inverters



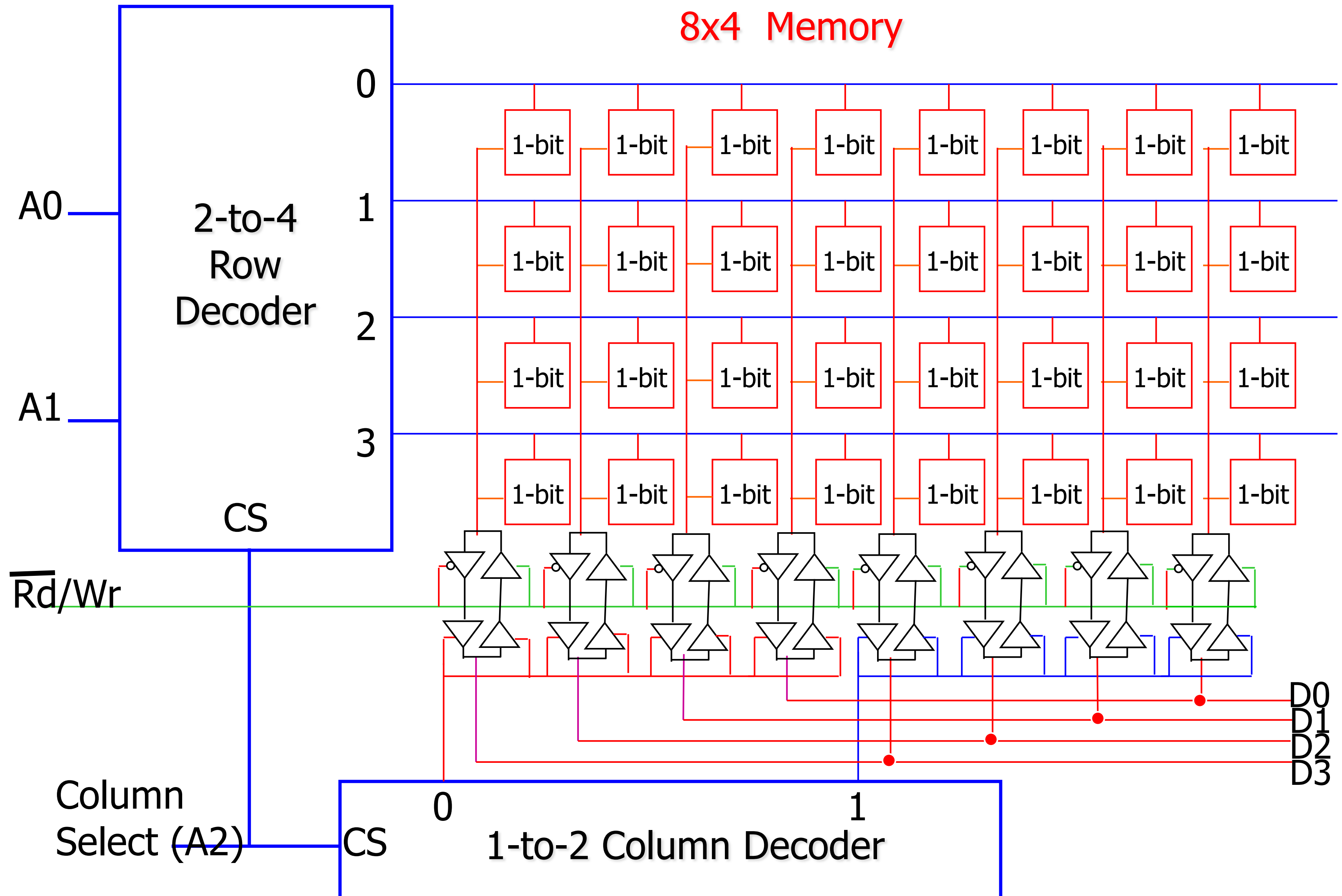



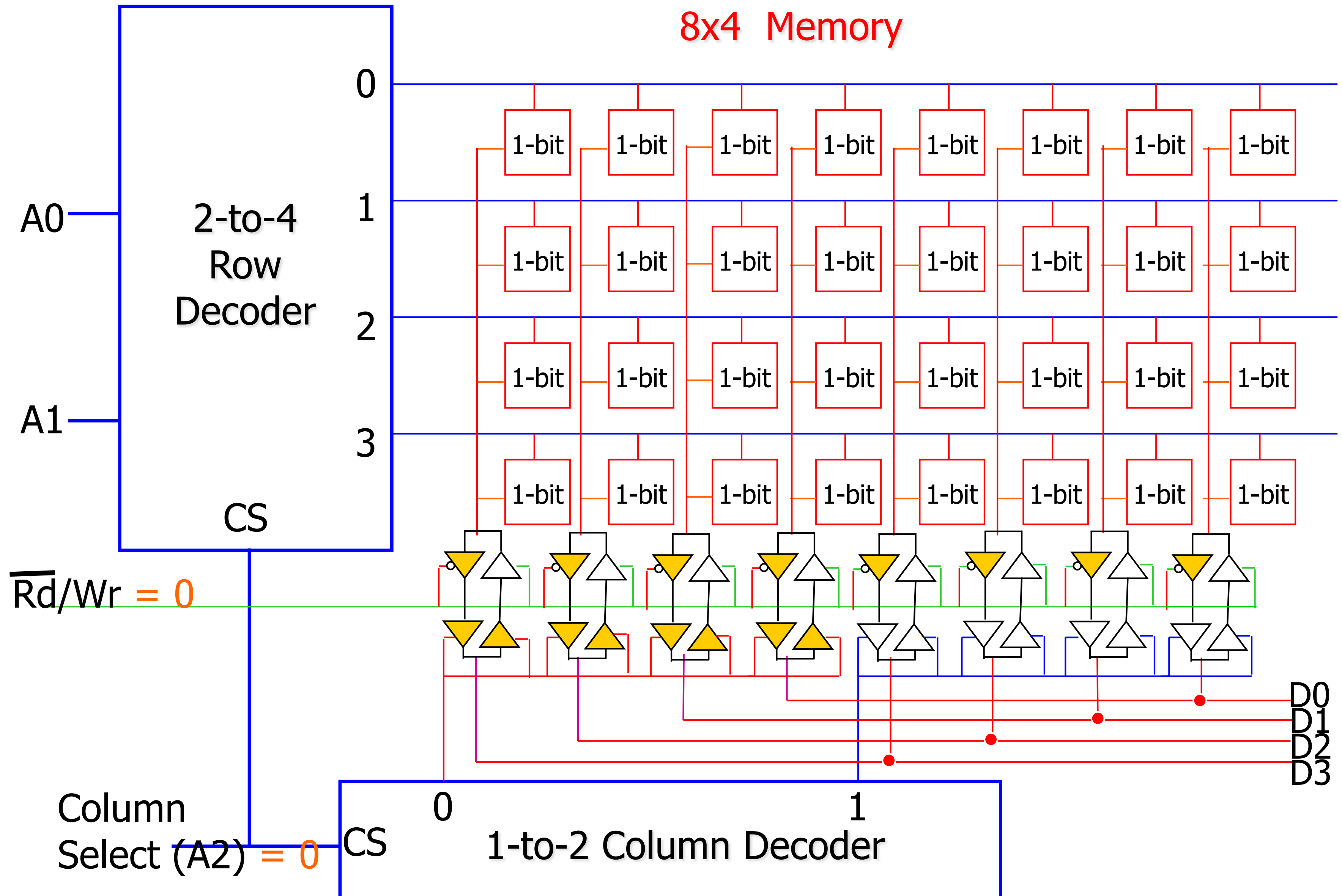
# Memory with column decoder



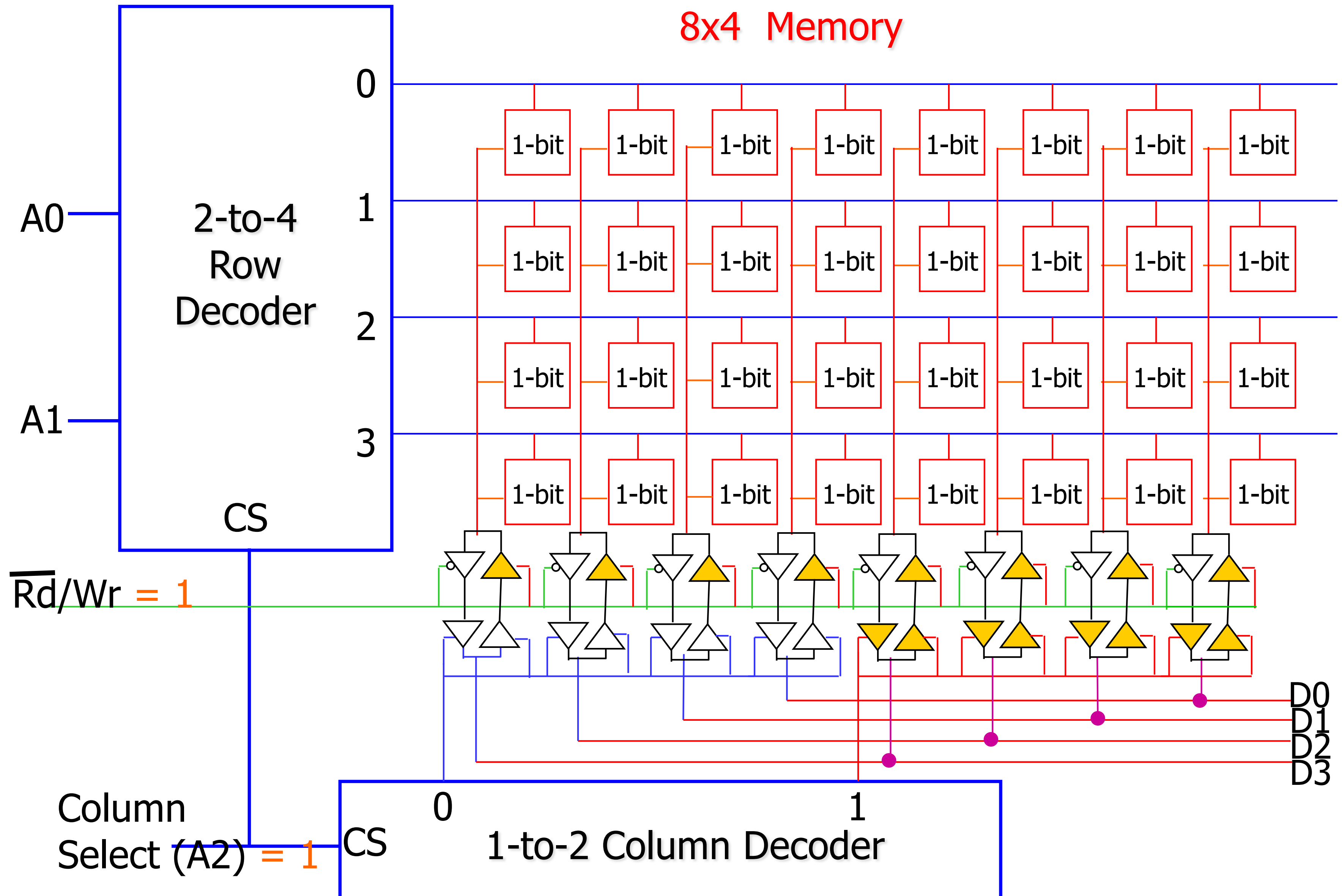
# Read/Write Memory



# Read/Write Memory



# Read/Write Memory

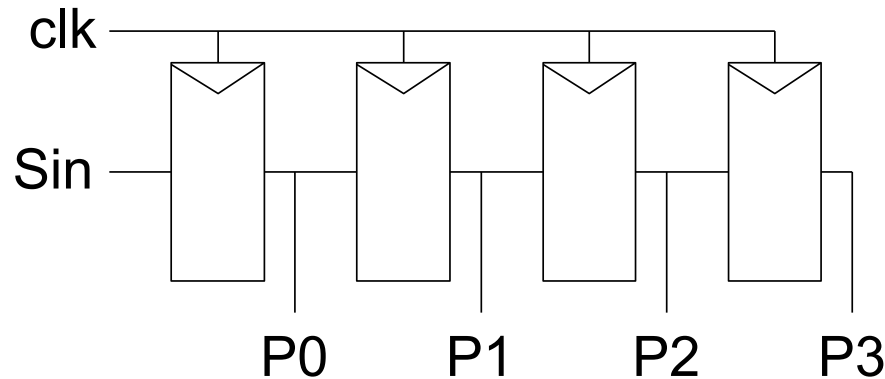


# Serial Access Memories

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

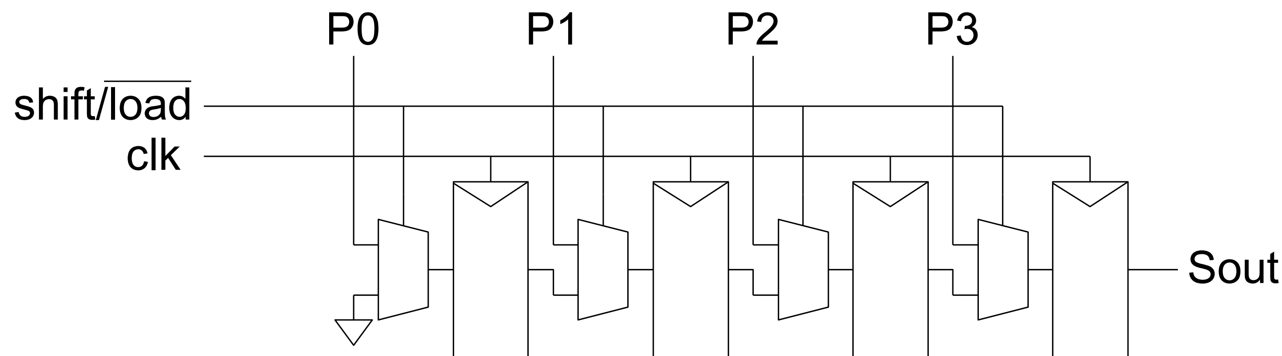
# Serial In Parallel Out

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



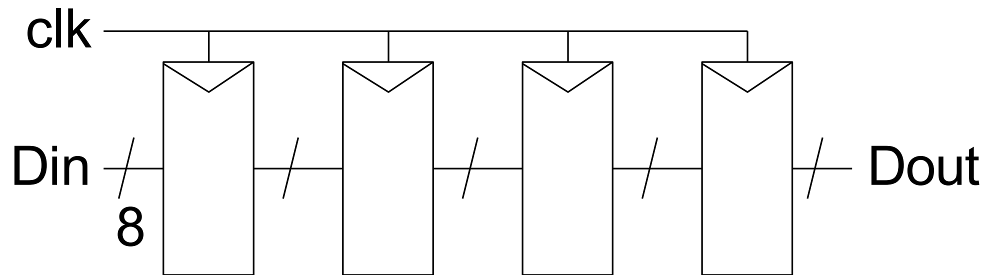
# Parallel In Serial Out

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



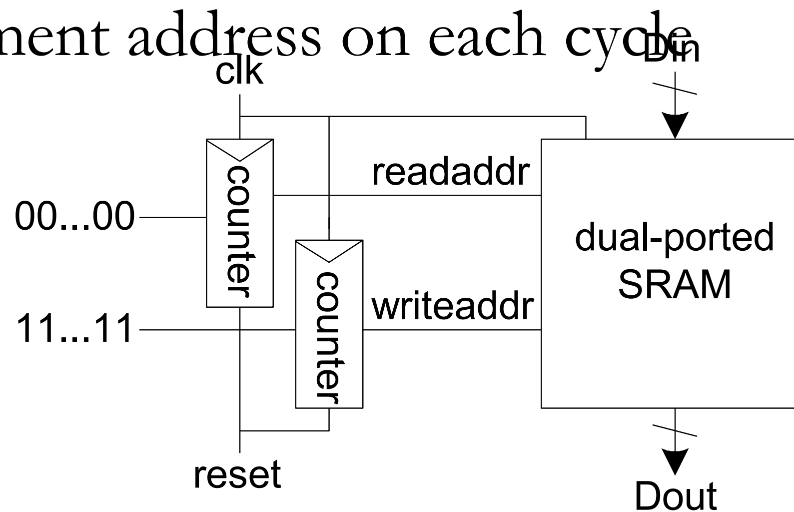
# Shift Register

- *Shift registers* store and delay data
- Simple design: cascade of registers



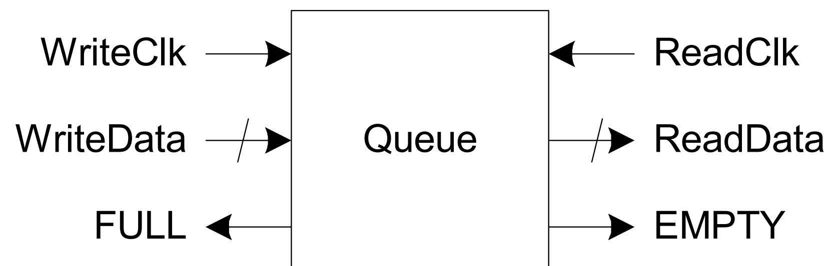
# Denser Shift Registers

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



# 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
  - If read catches write, Queue is EMPTY
- *Last In First Out* (LIFO)
  - Also called a *stack*
  - Use a single *stack pointer* for read and write

# Idea

- Memory for compact state storage
- Minimize area of repeated cell
  - 6T/5T SRAM
  - Mulitport trade off area for function
  - 1T/3T DRAM helps but slower
- Share circuitry across many bits
  - Minimize area per bit → maximize density
- Aggressively use:
  - Pass transistors, Ratioing
  - Precharge, Amplifiers to keep area down

# Admin

- Homework 7
  - Come to lab 4/22 and turn in worksheet for credit
- Project 2 out now
  - Design SRAM array
  - Work in teams of up to two
  - Milestone due F 4/19
    - Will get feedback from me by M 4/22
  - Final report due Wednesday 5/1

# Acknowledgement


- Prof. André DeHon (University of Pennsylvania)
- Prof. Jing Li (University of Pennsylvania)