ESE370: Circuit-Level Modeling, Design, and Optimization for Digital Systems

Lec 27: November 12, 2021

Memory Periphery, Serial Access Memories

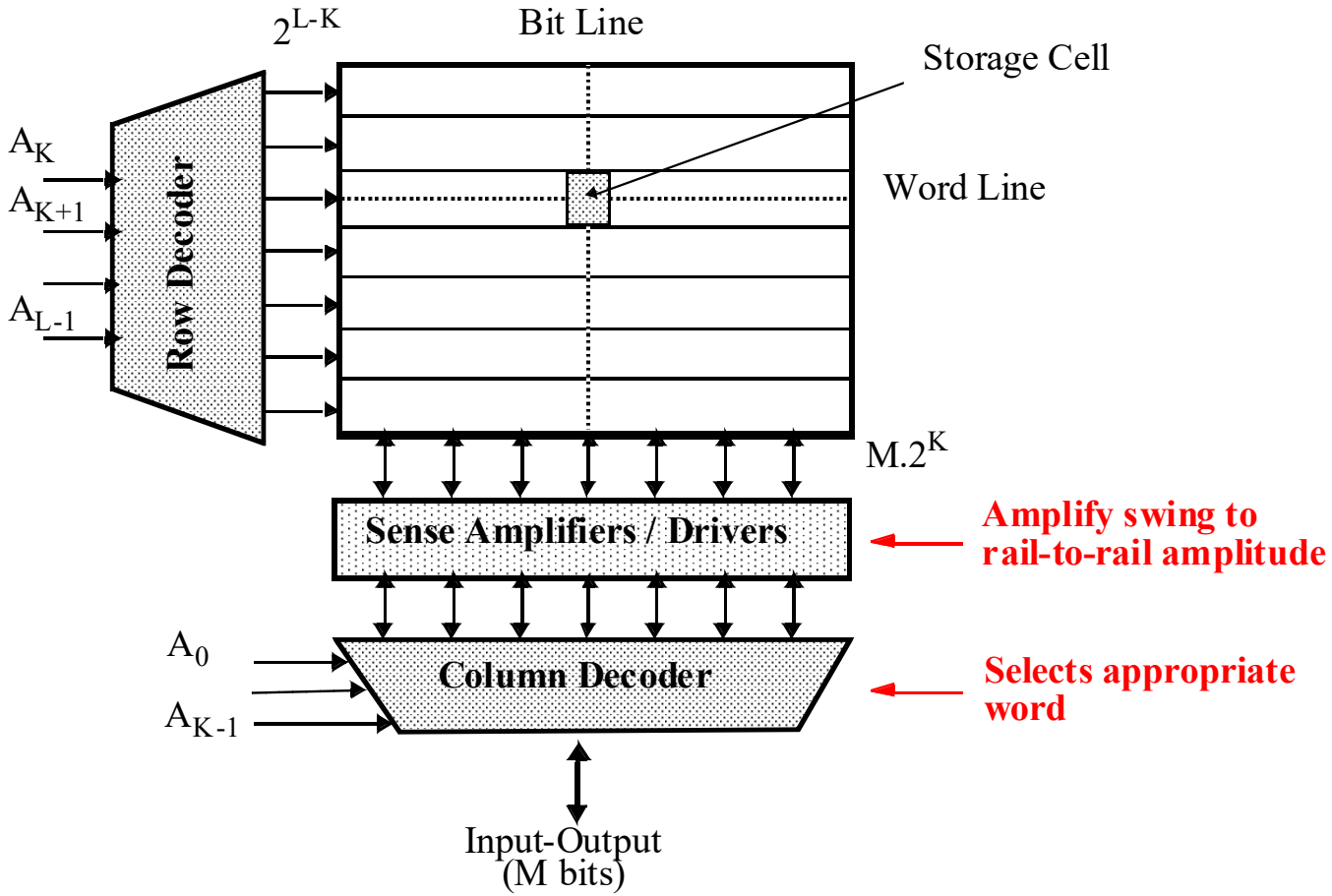
Today

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

- Project 2 is on this

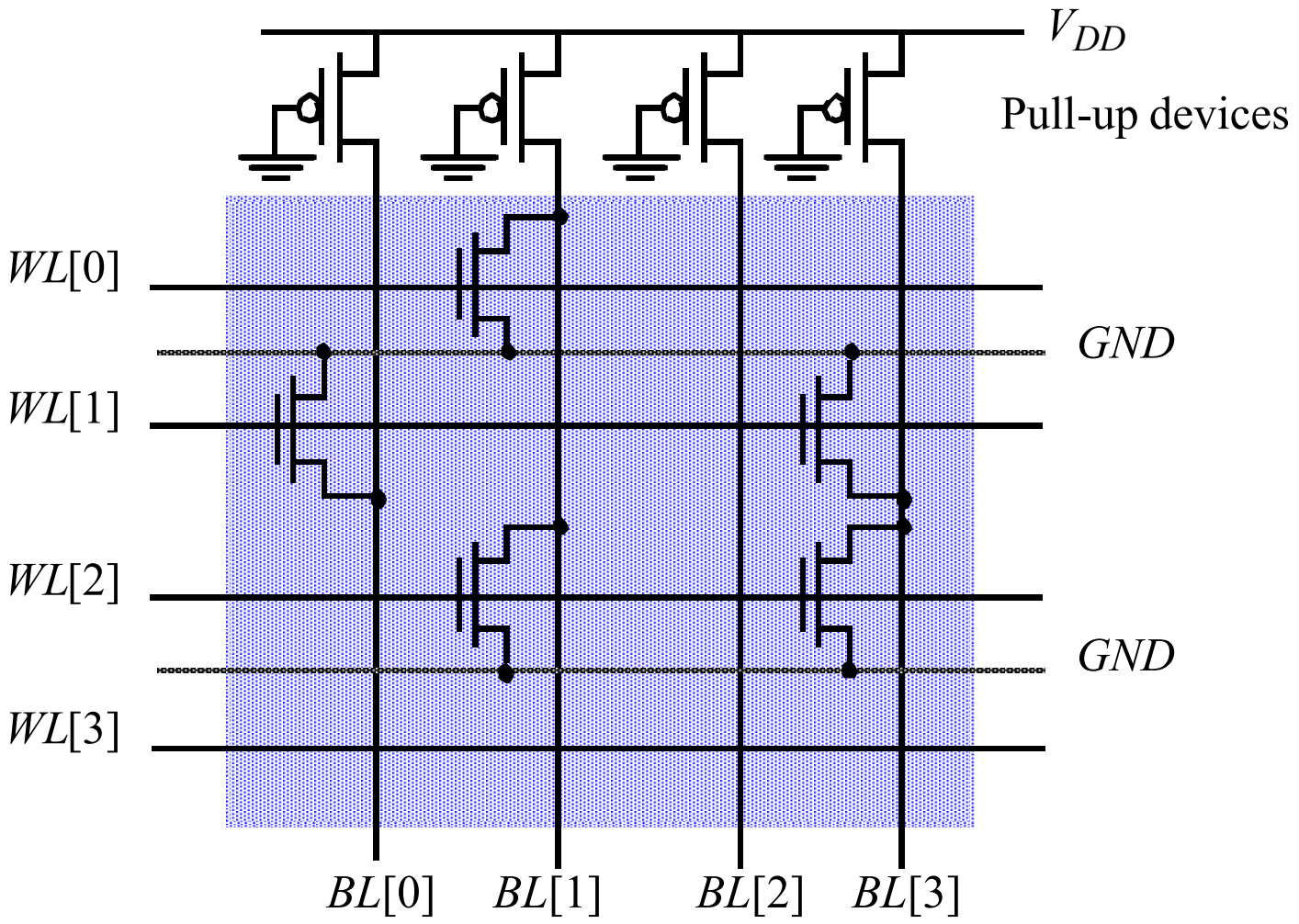
Array-Structured Memory Architecture

Problem: ASPECT RATIO or HEIGHT >> WIDTH

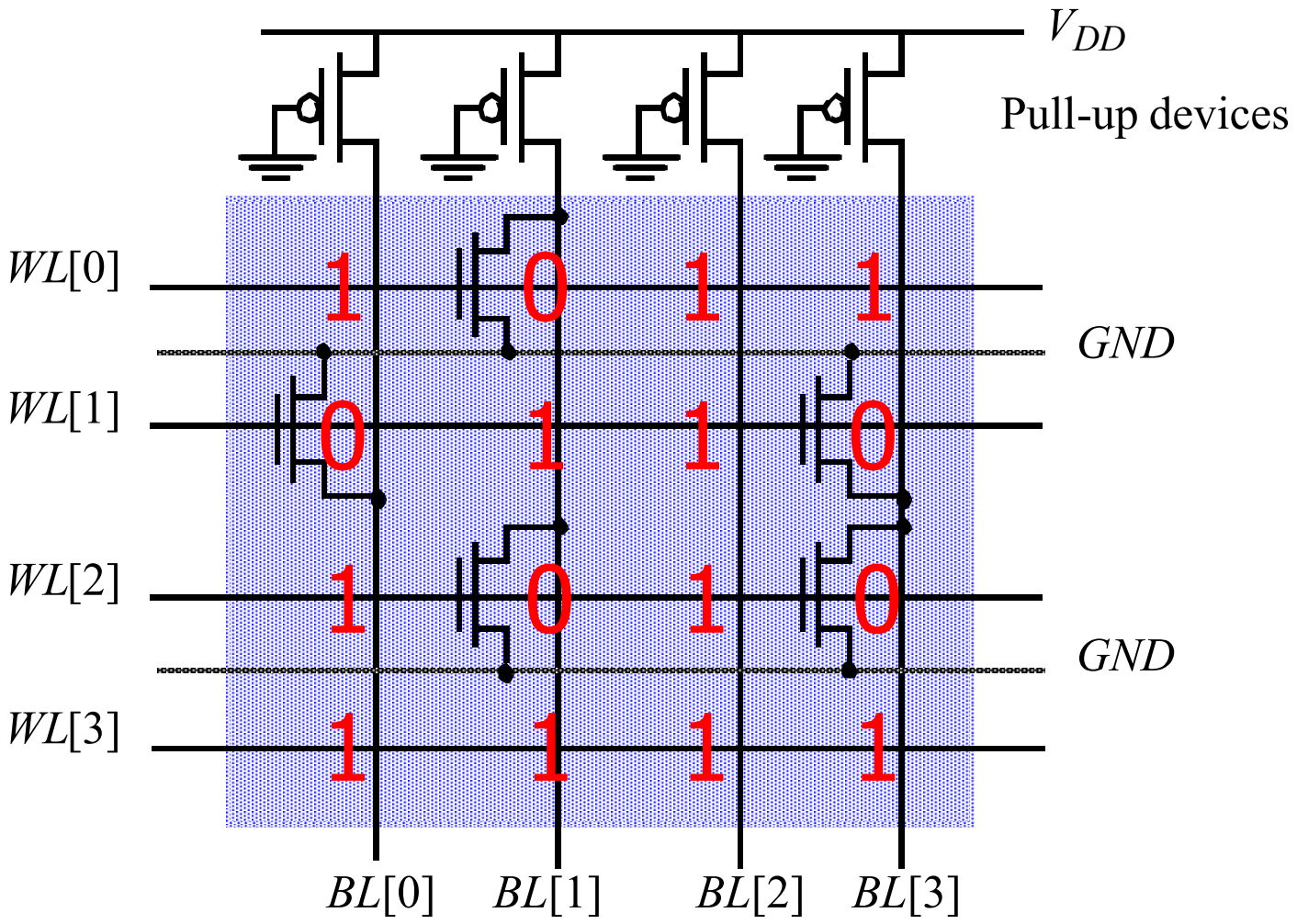


ROM Memories

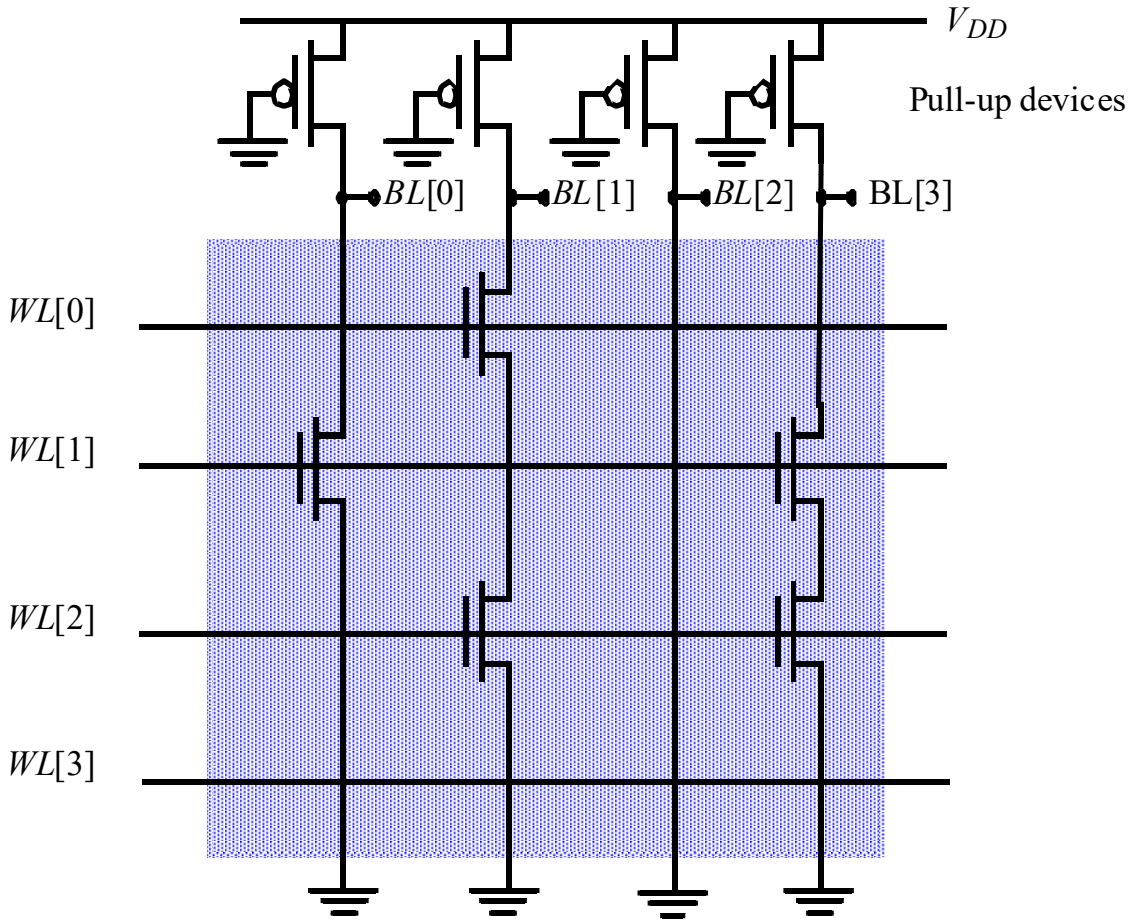
MOS NOR ROM



MOS NOR ROM

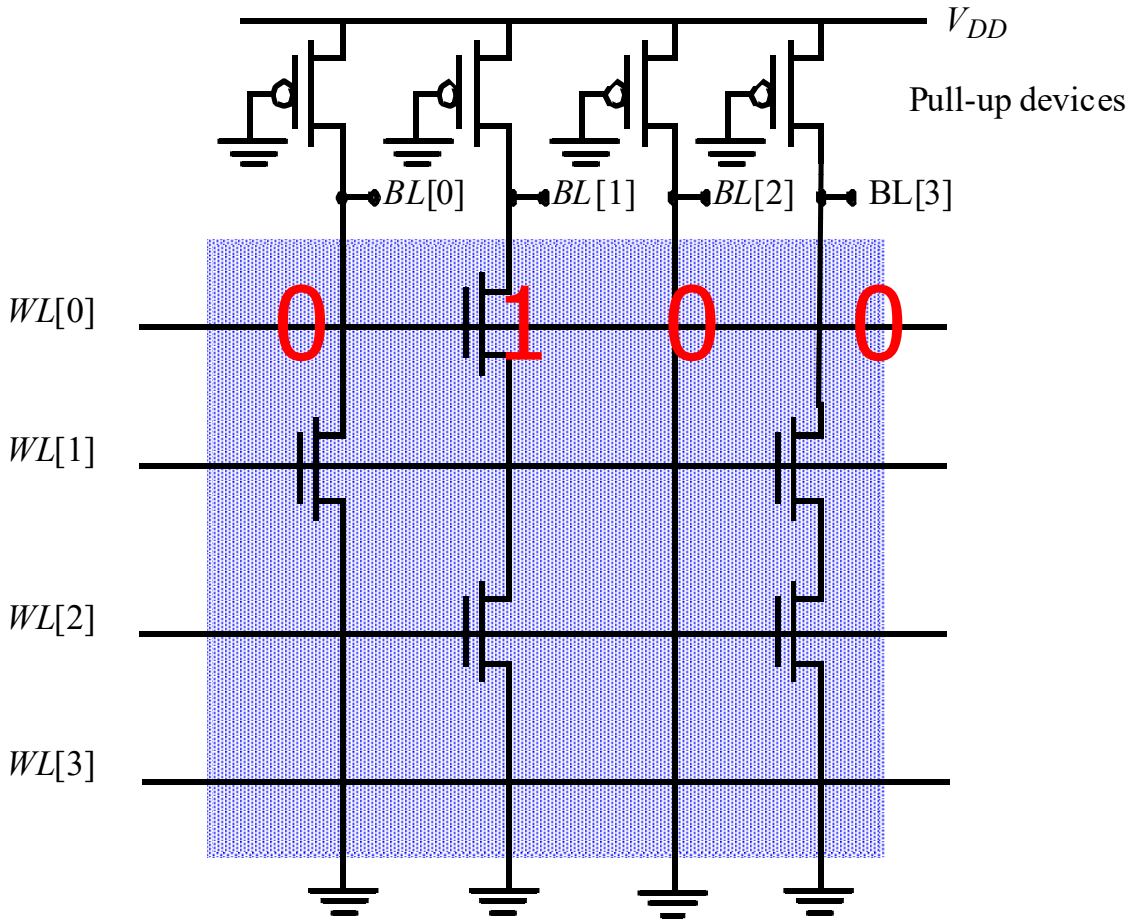

V_{DD}
Pull-up devices
WL[0]
WL[1]
WL[2]
WL[3]
GND
GND
1 0 1 1
0 1 1 0
1 0 1 0
1 1 1 1
BL[0]
BL[1]
BL[2]
BL[3]

MOS NAND ROM



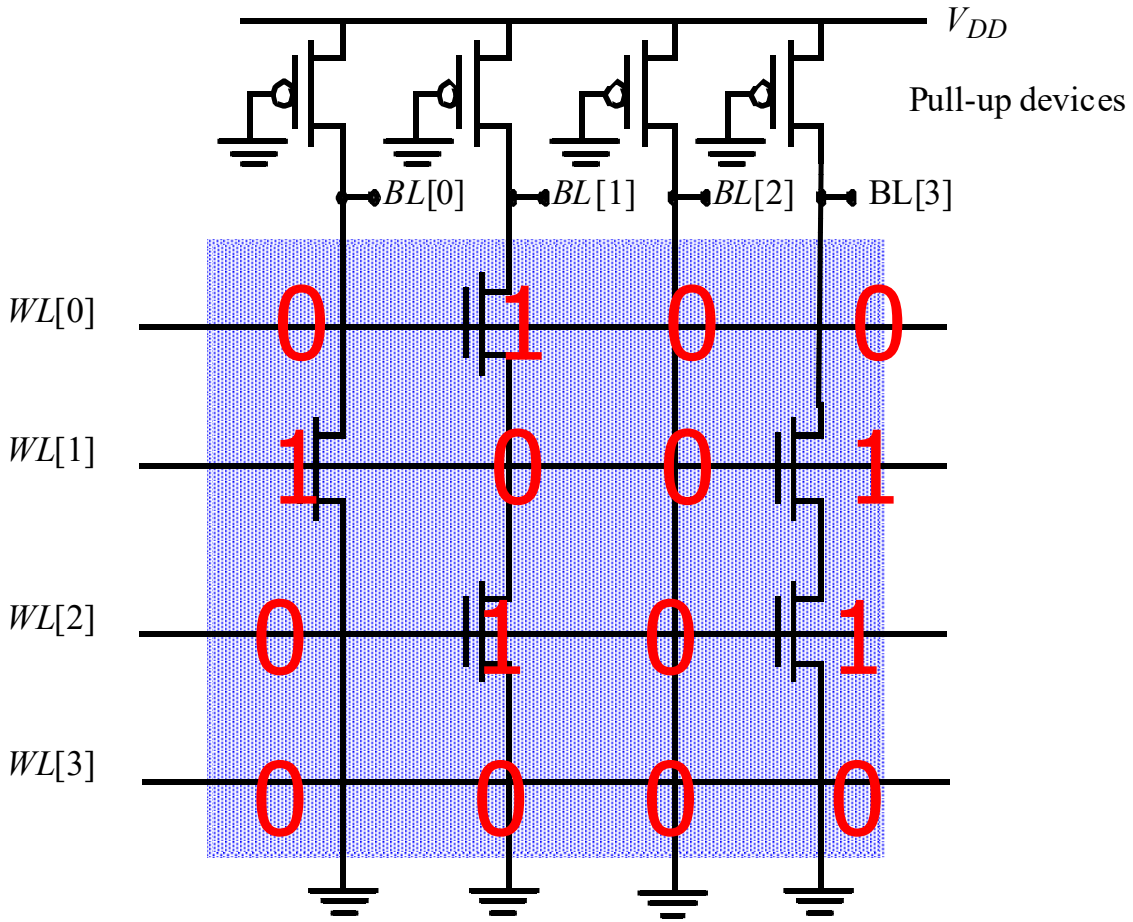
All word lines high by default with exception of selected row

MOS NAND ROM



All word lines high by default with exception of selected row

MOS NAND ROM



All word lines high by default with exception of selected row

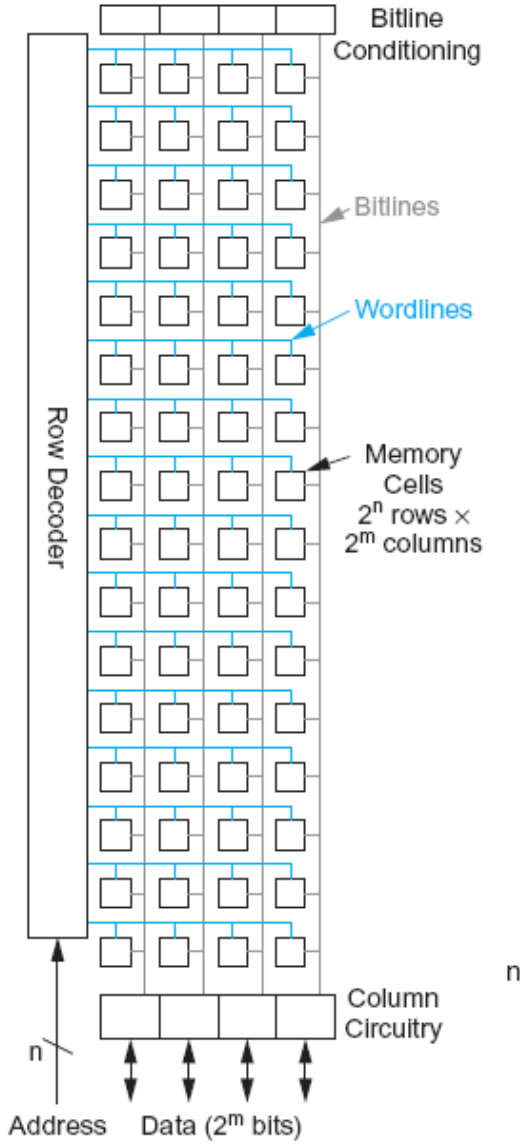
Memory Periphery

Periphery

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

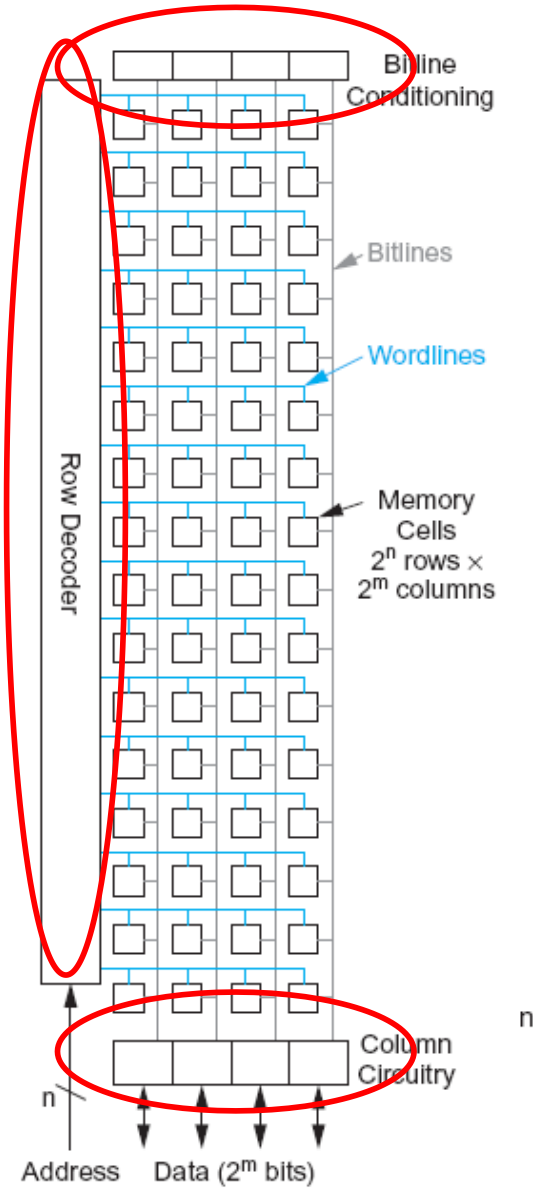
Array Architecture

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



Array Architecture

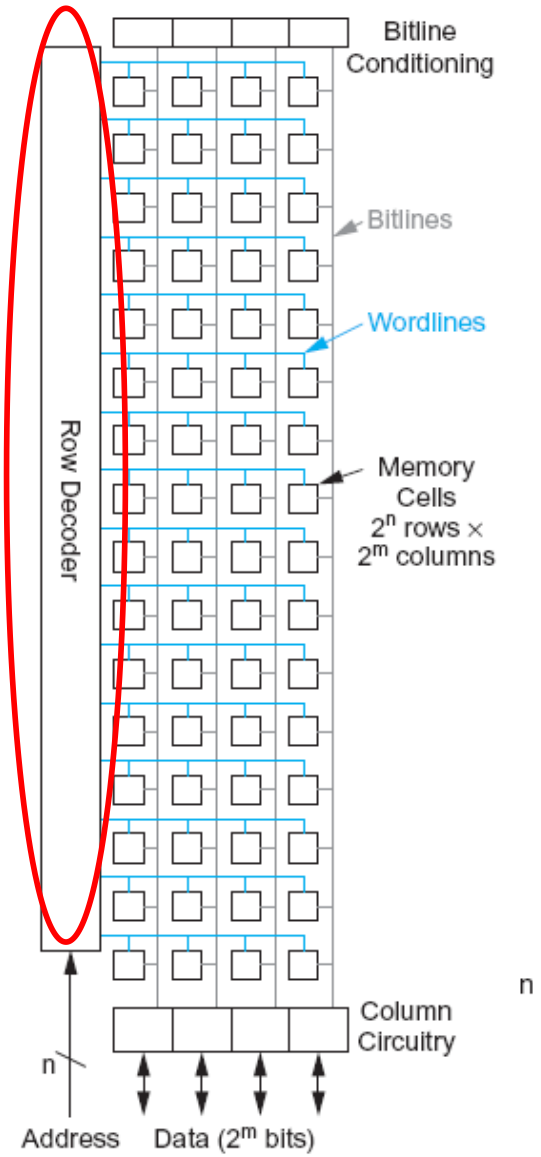
- 2^n *words* of 2^m *bits* each
- Good regularity – easy to design
- Very high density if good cells are used



Decoders

Array Architecture

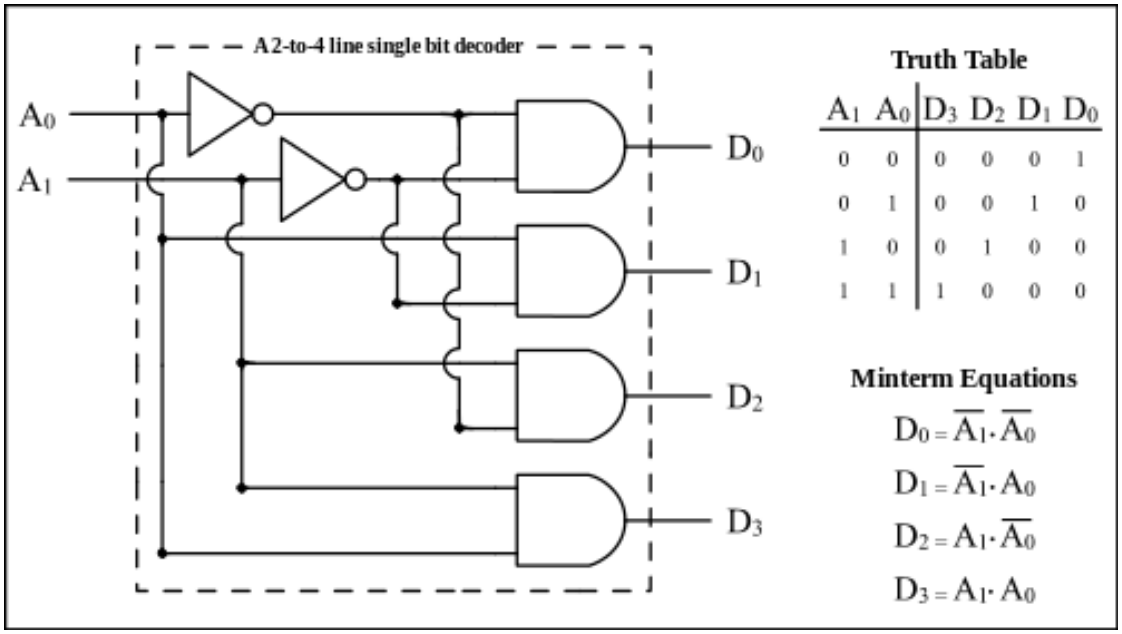
- 2^n *words* of 2^m *bits* each
- Good regularity – easy to design
- Very high density if good cells are used



Decoders

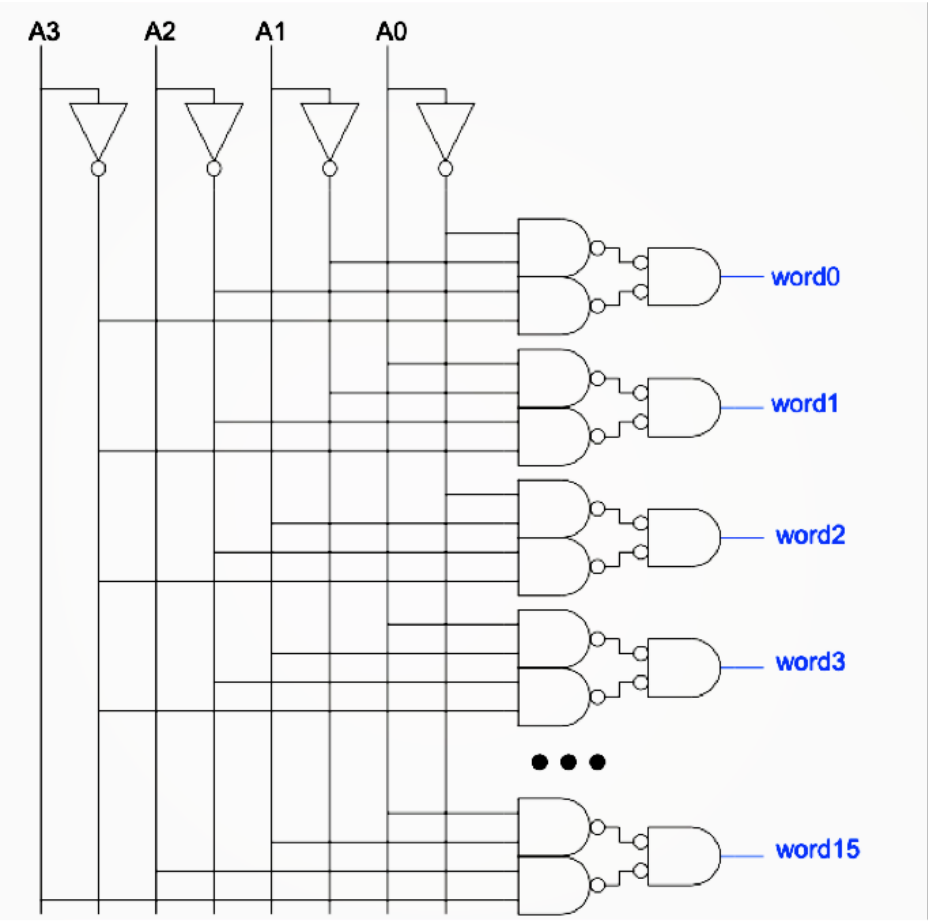
- n:2^n decoder consists of 2^n n-input AND gates
  - One needed for each row of memory
  - Build AND from NAND or NOR gates

Static CMOS



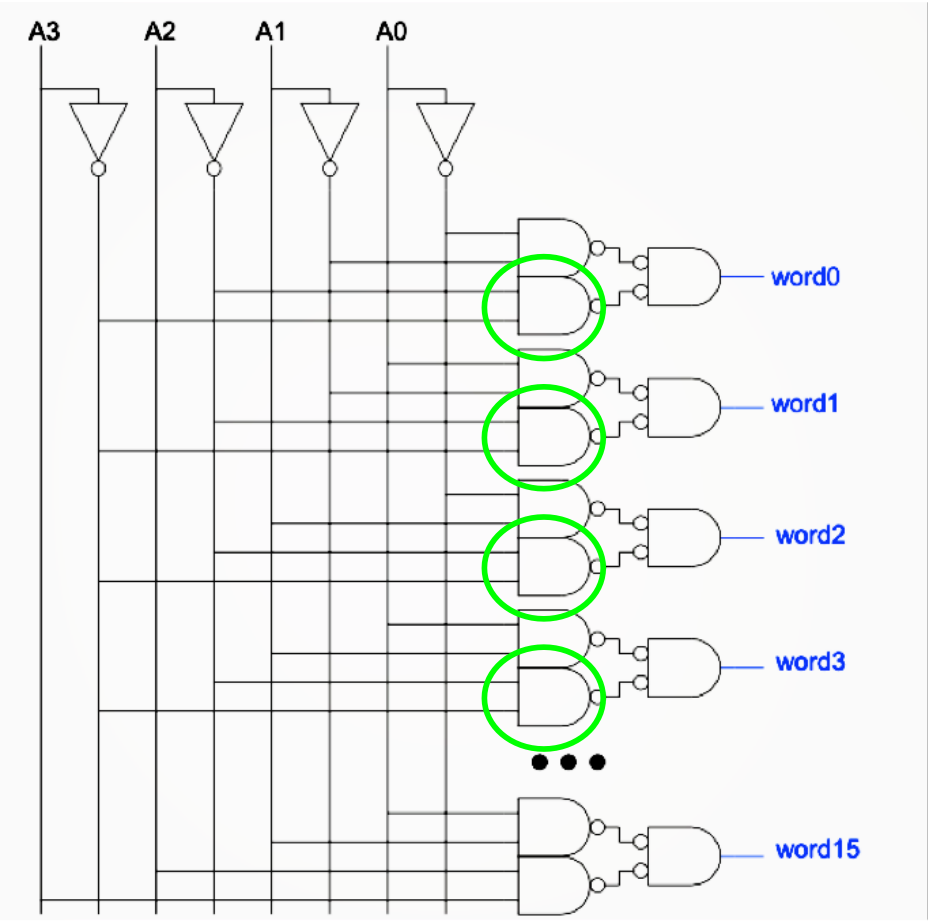
Large Decoders

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



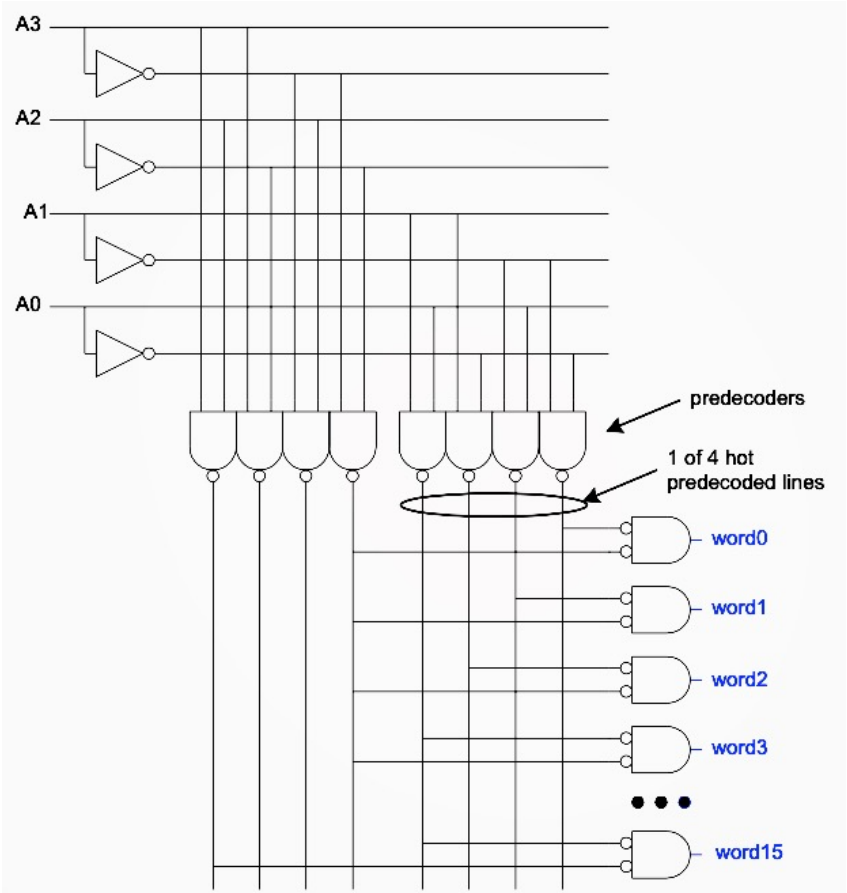
Large Decoders

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

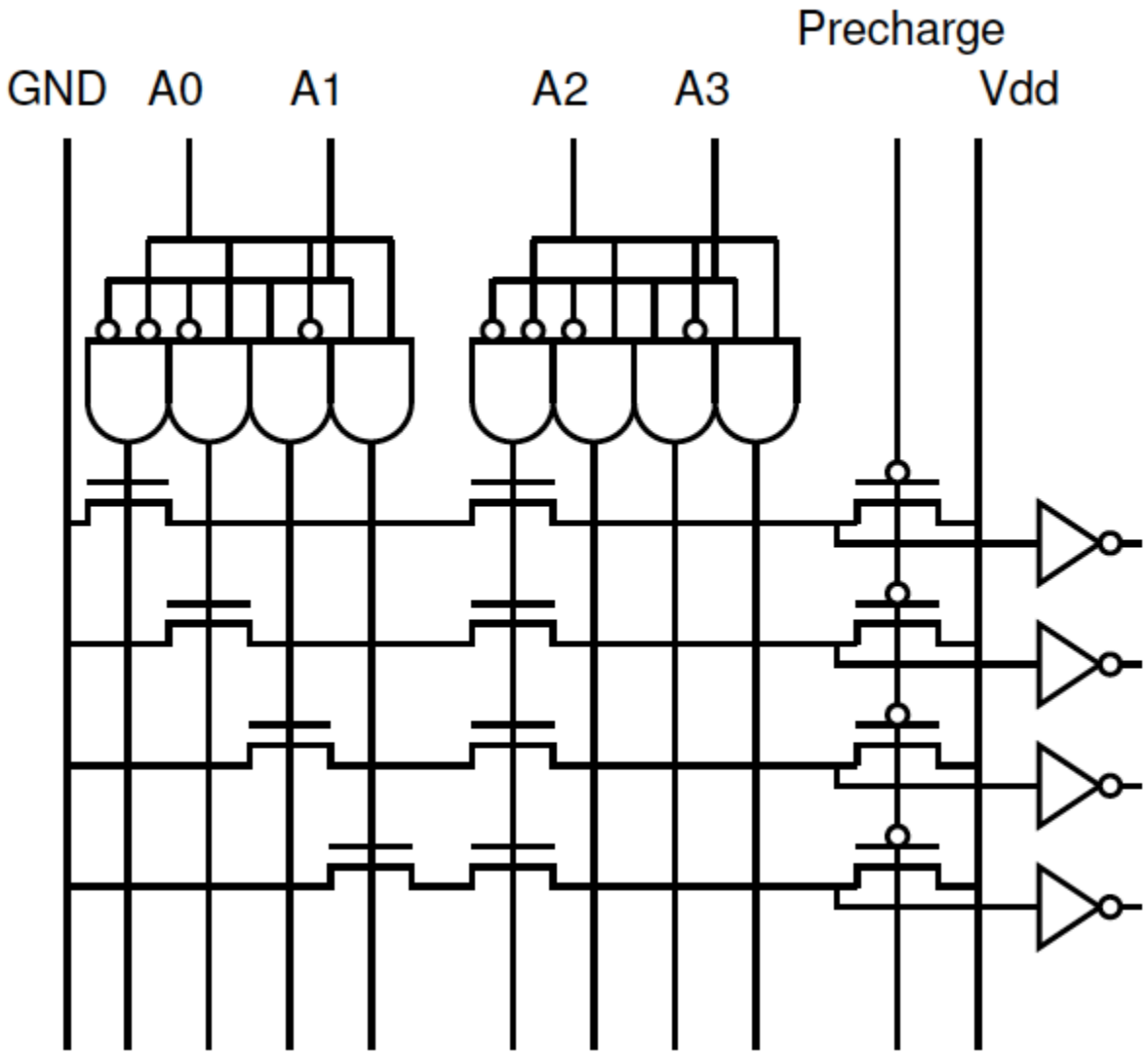


Predecoding

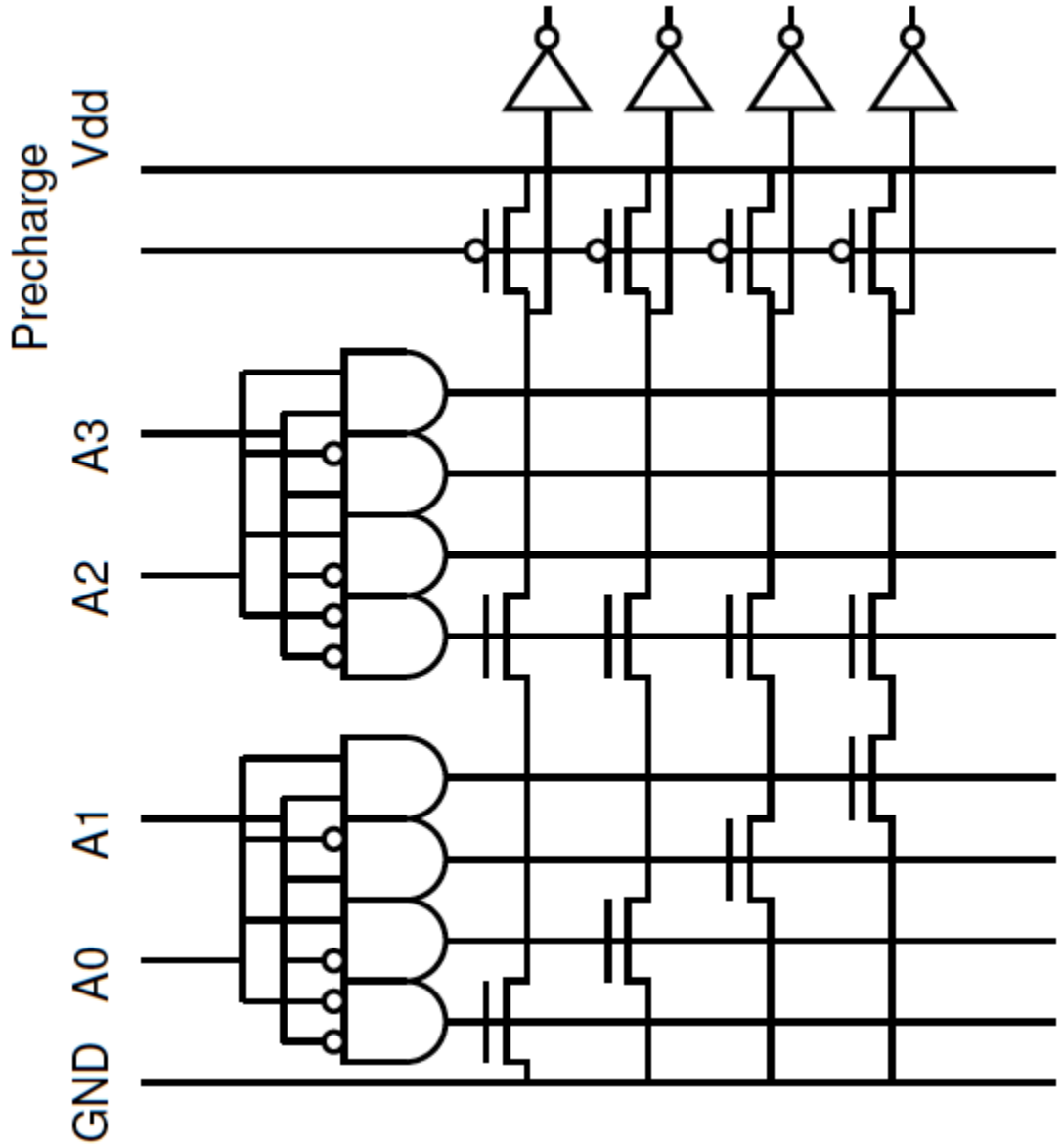
- Many of these gates are redundant
  - Factor out common gates into predecoder
  - Saves area
  - Same path effort



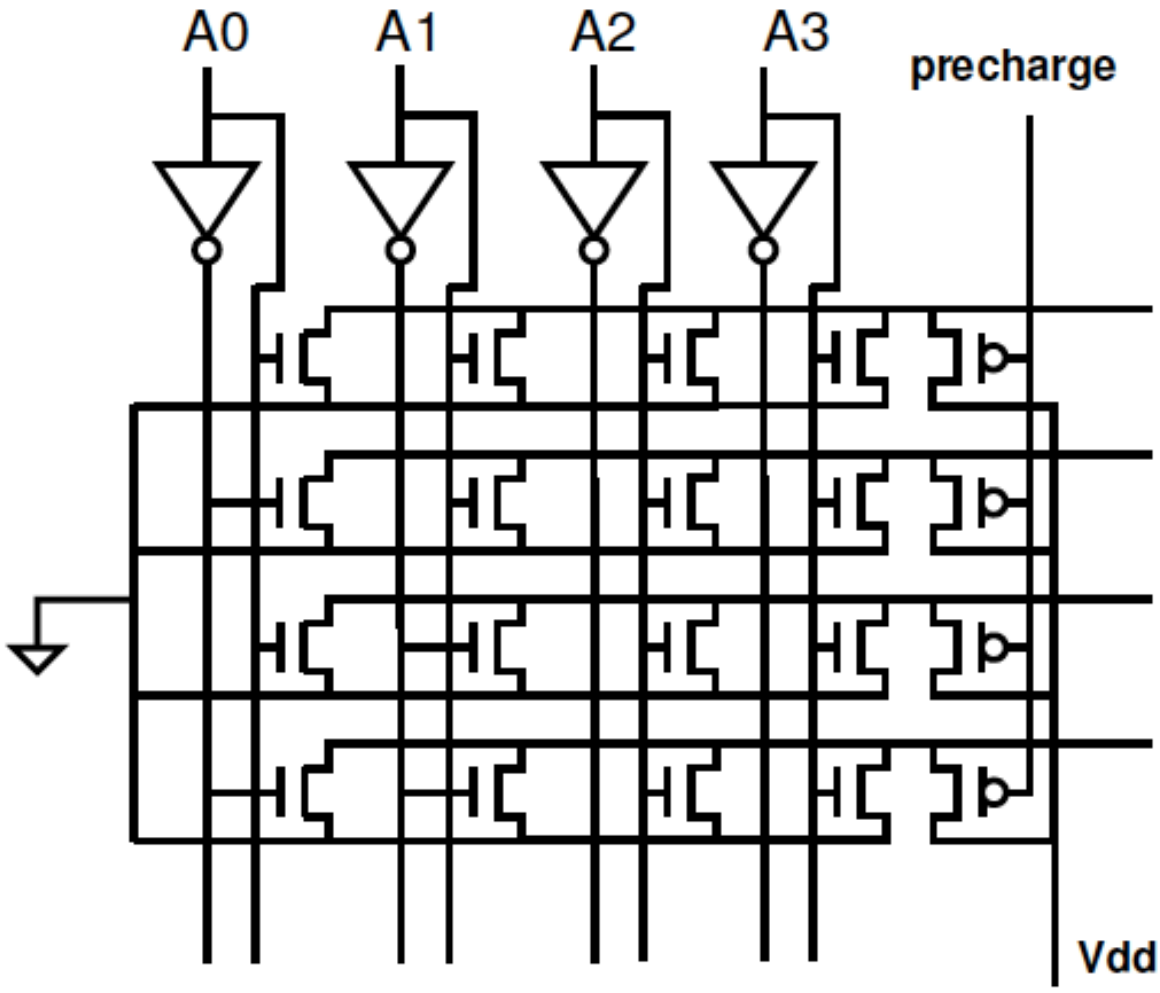
Row Select: Precharge NAND



Row Select: Precharge NAND



Row Select: Precharge NOR

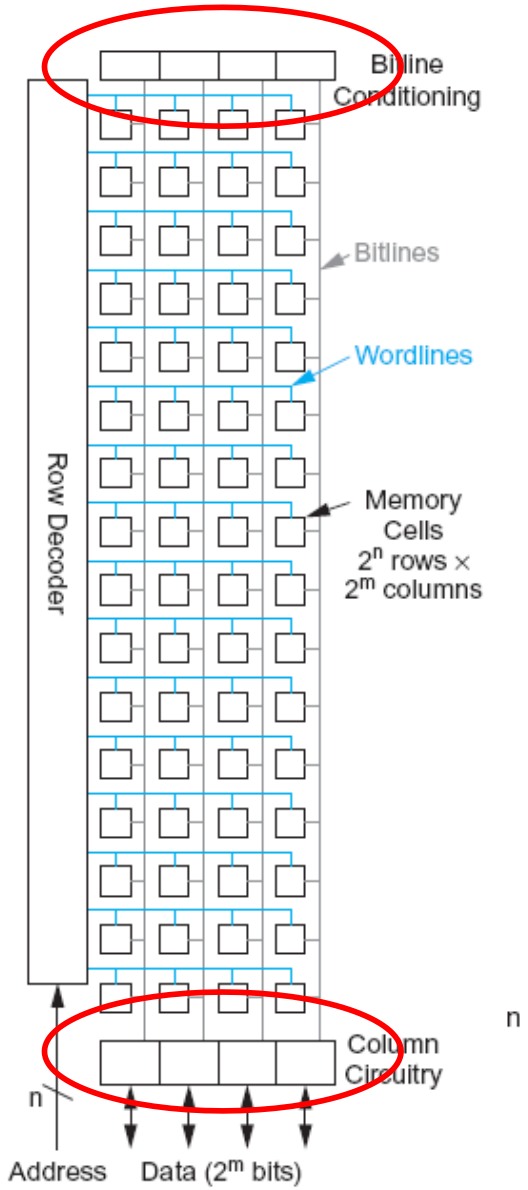


Column Circuitry

& Bit-line Conditioning

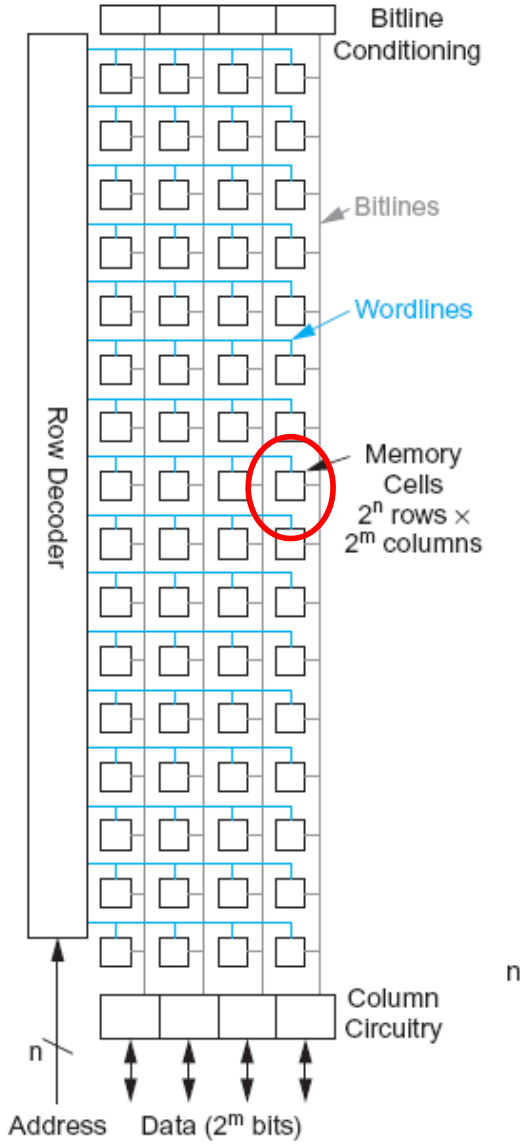
Array Architecture

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



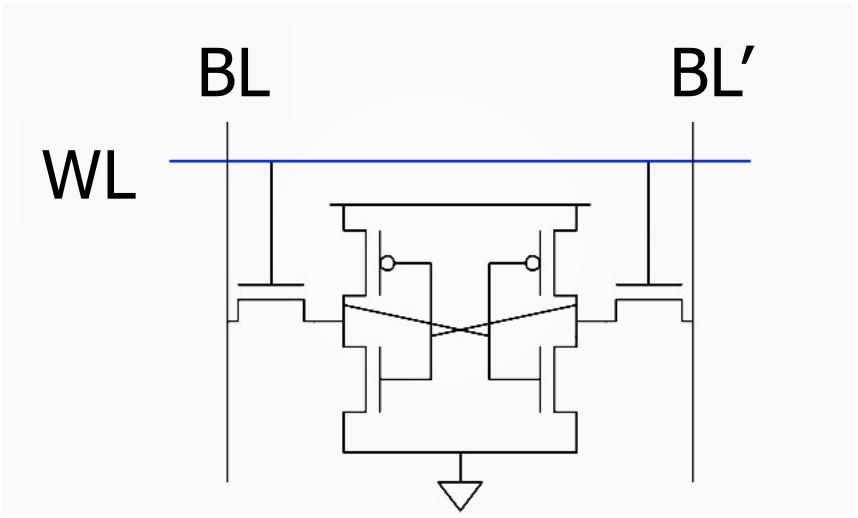
Array Architecture

- 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

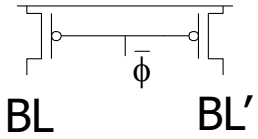


Column Circuitry

- 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)

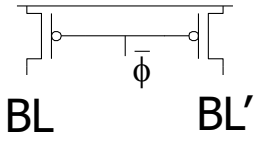
Bitline Conditioning

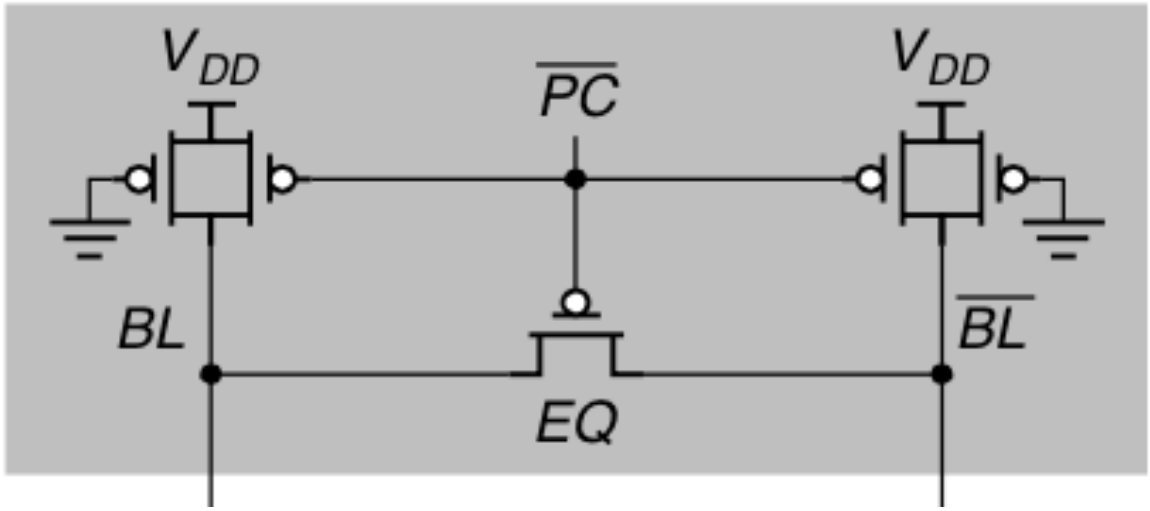
- Precharge bitlines high before read operations



Bitline Conditioning

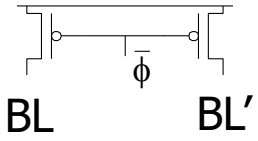
- Precharge bitlines high before reads

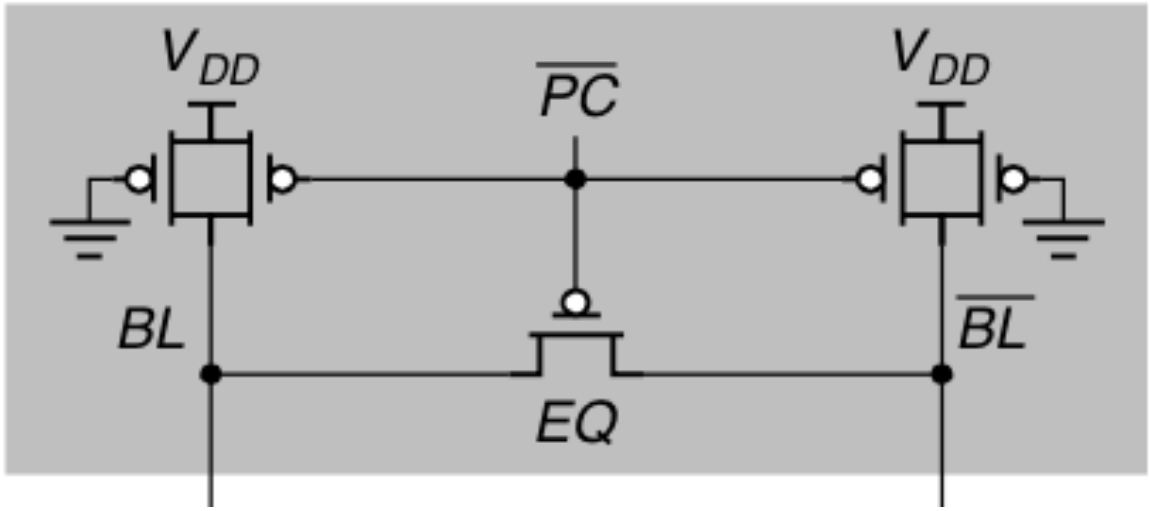




Bitline Conditioning

- Precharge bitlines high before reads

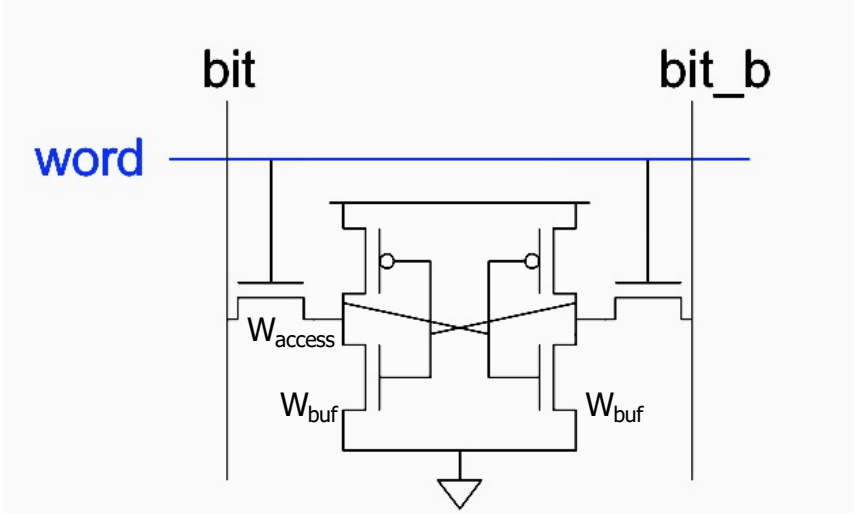


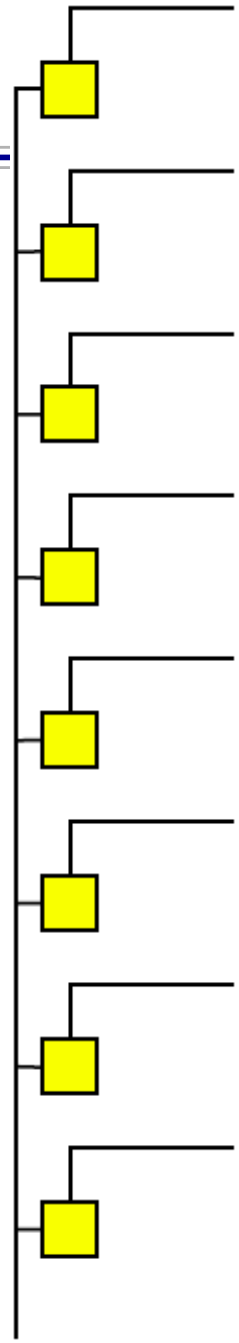


- What if pre-charged to Vdd/2?
  - Pros: reduces read-upset
  - Challenge: generate Vdd/2 voltage on chip

Column Capacitance Consequence

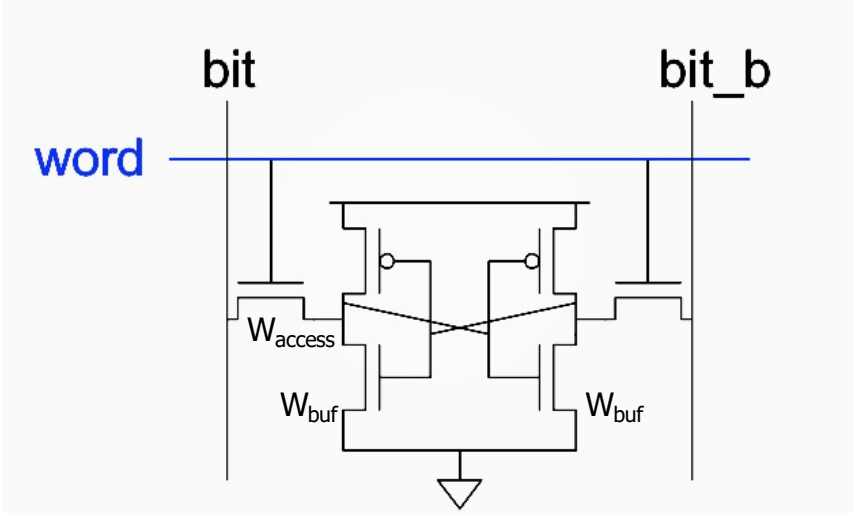
- Preclass1: What is capacitance of a bitline?
  - W_{access} (pass transistor size), d rows, $\gamma = C_{diff0}/C_0$

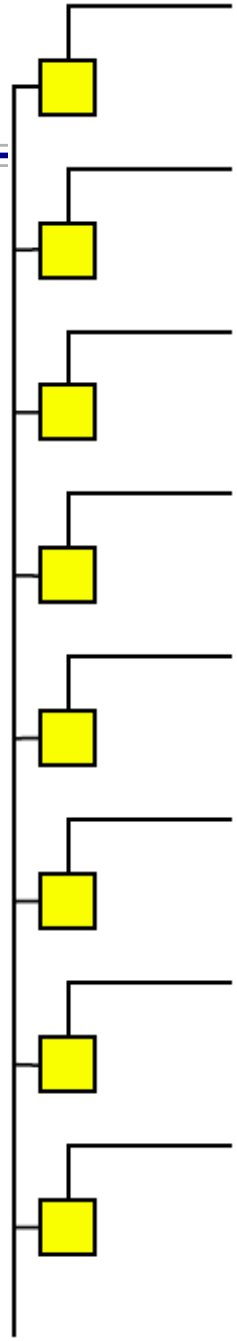




Column Capacitance Consequence

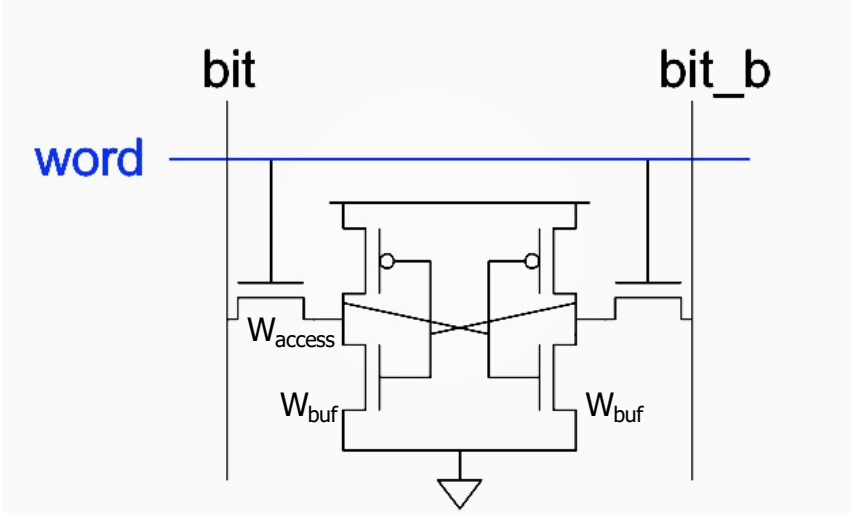
- Preclass1: What is capacitance of a bitline?
  - W_{access} (pass transistor size), d rows, $\gamma = C_{diff0}/C_0$
- Preclass2: What is the delay for the cell to drive the bitline during a read?
  - W_{buf} (inverter size in cell), R_0





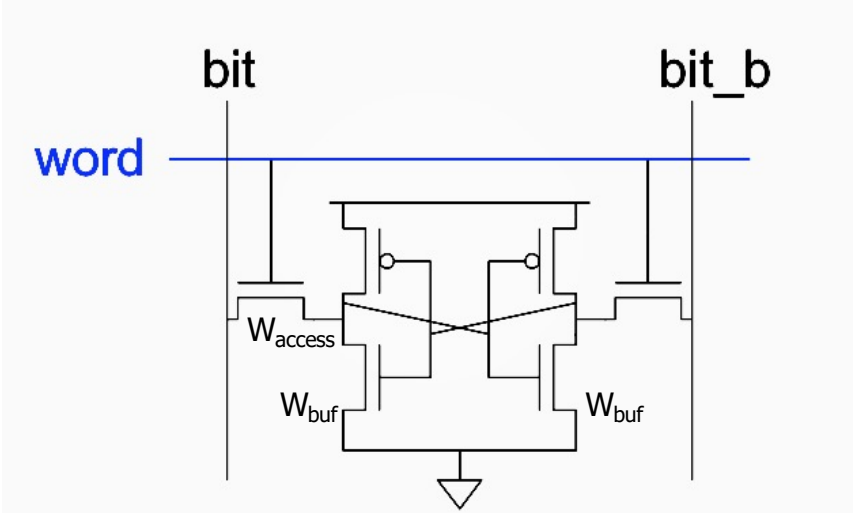
Column Capacitance Consequence

- Preclass1: What is capacitance of a bitline?
  - W_{access} (pass transistor size), d rows, $\gamma = C_{diff0}/C_0$
- Preclass2: What is the delay for the cell to drive the bitline during a read?
  - W_{buf} (inverter size in cell), R_0
- Preclass3: Waccess=Wbuf=1, $\gamma = 1/2$
  - Delay for d=32, 512?



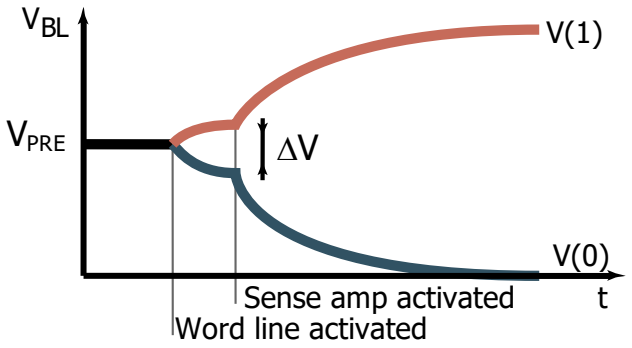
Column Capacitance Consequence

- Preclass1: What is capacitance of a bitline?
  - W_{access} (pass transistor size), d rows, $\gamma = C_{diff0}/C_0$
- Preclass2: 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



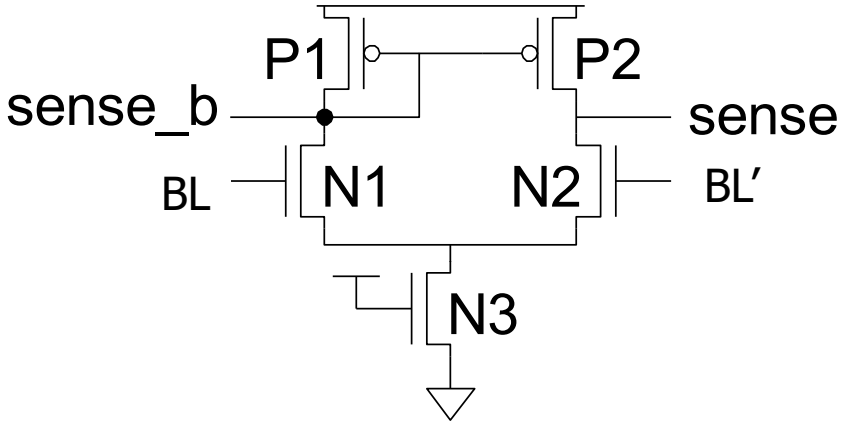
Sense Amplifiers

- 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 (ΔV)



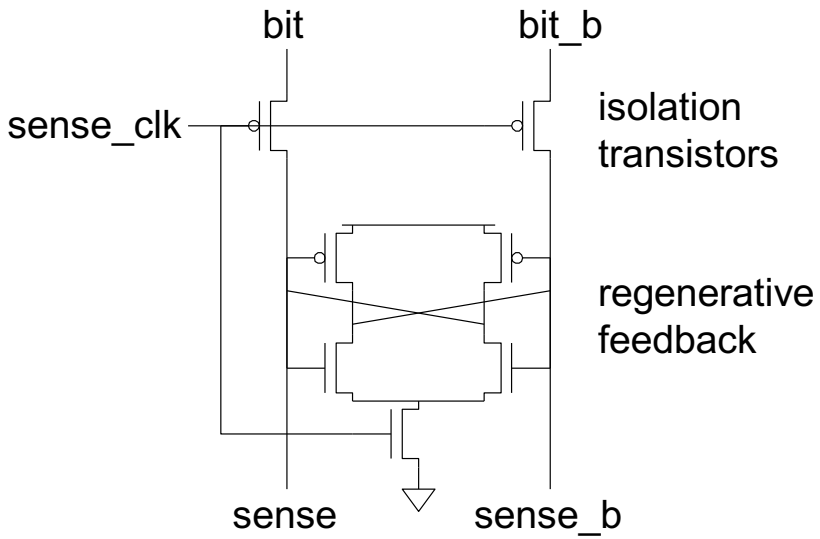
Differential Pair Amp

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



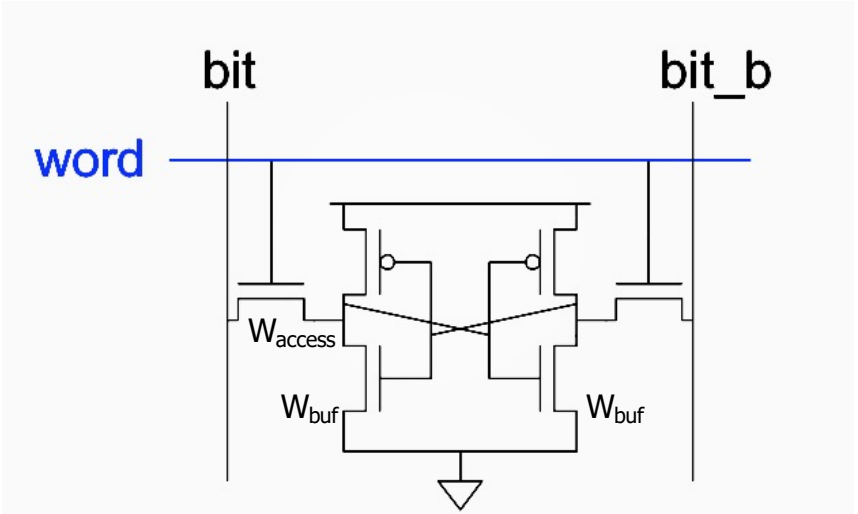
Clocked Sense Amp

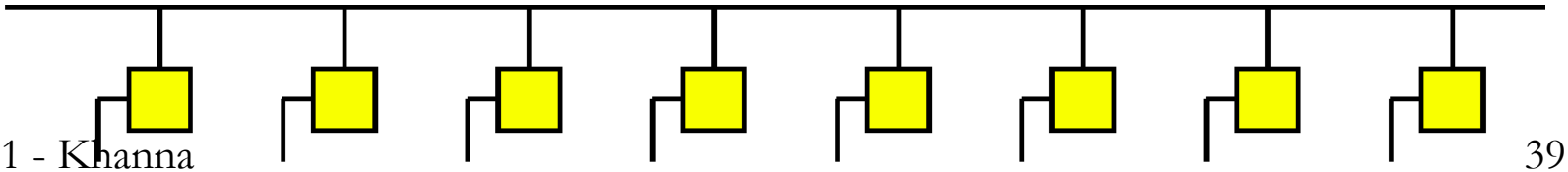
- Clocked sense amp saves power
- Requires sense_clk after enough bitline swing
- Isolation transistors cut off large bitline capacitance



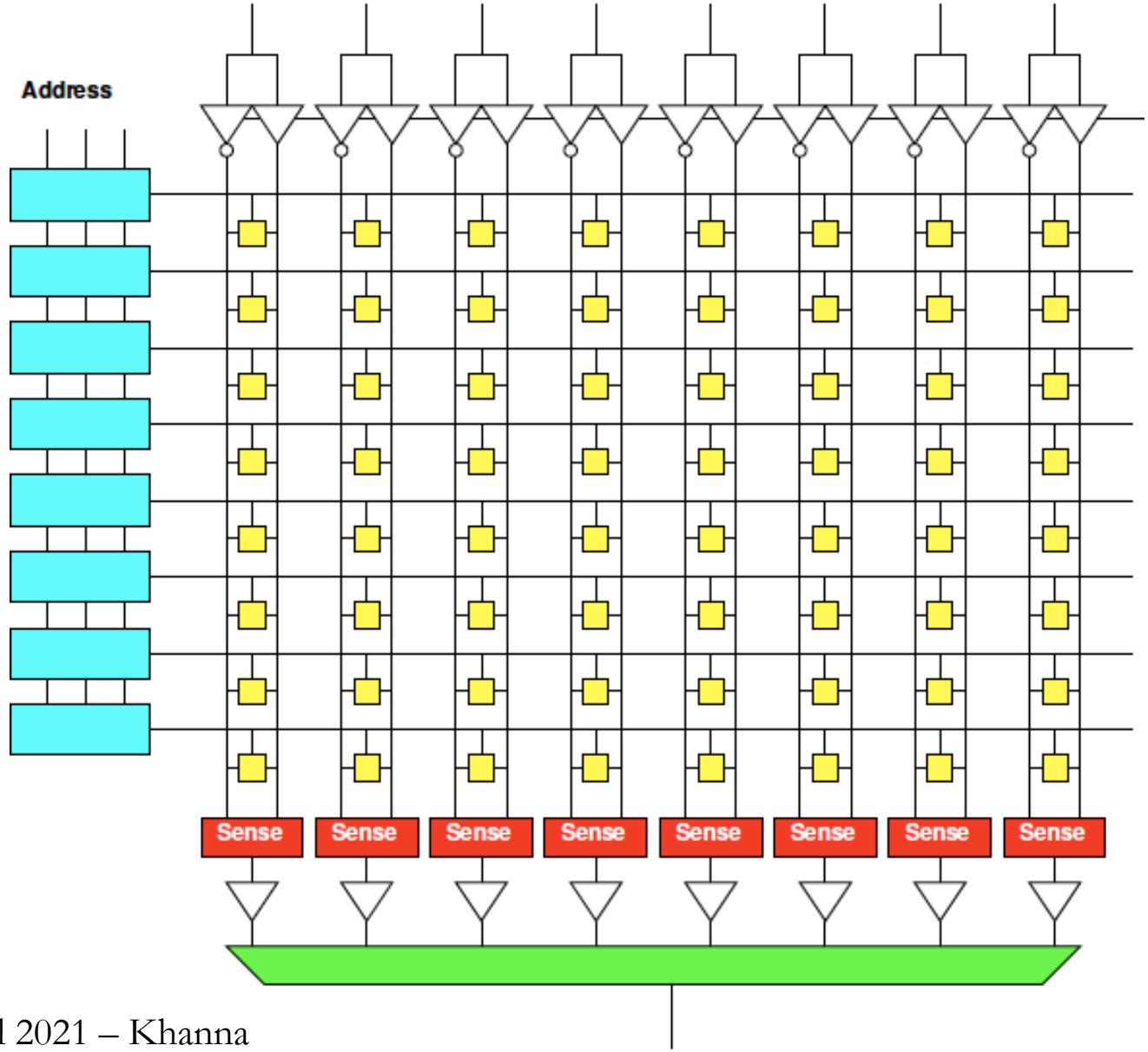
Word Line Capacitance

- Preclass4: What is capacitance of word line (row)?
  - W_{access}– transistor width of column device
  - w columns
  - $\gamma = C_{diff0}/C_0$
- Preclass5: Delay driving word line?
  - $W_{wldrive}$ Drive inverter

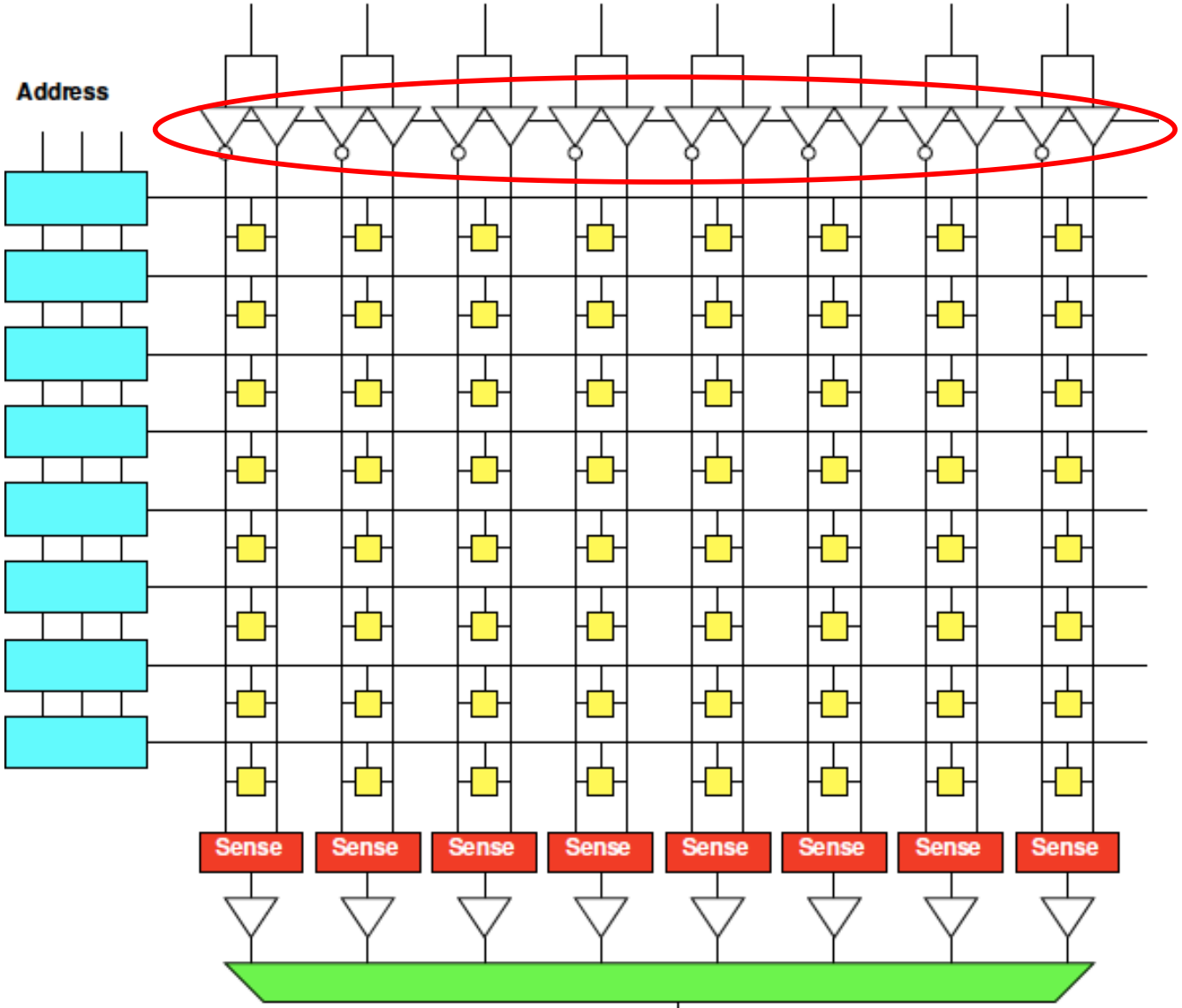




Column Drivers: Memory Bank

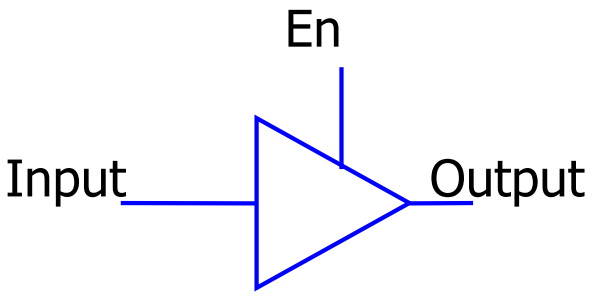


Column Drivers: Memory Bank



Tristate Buffer

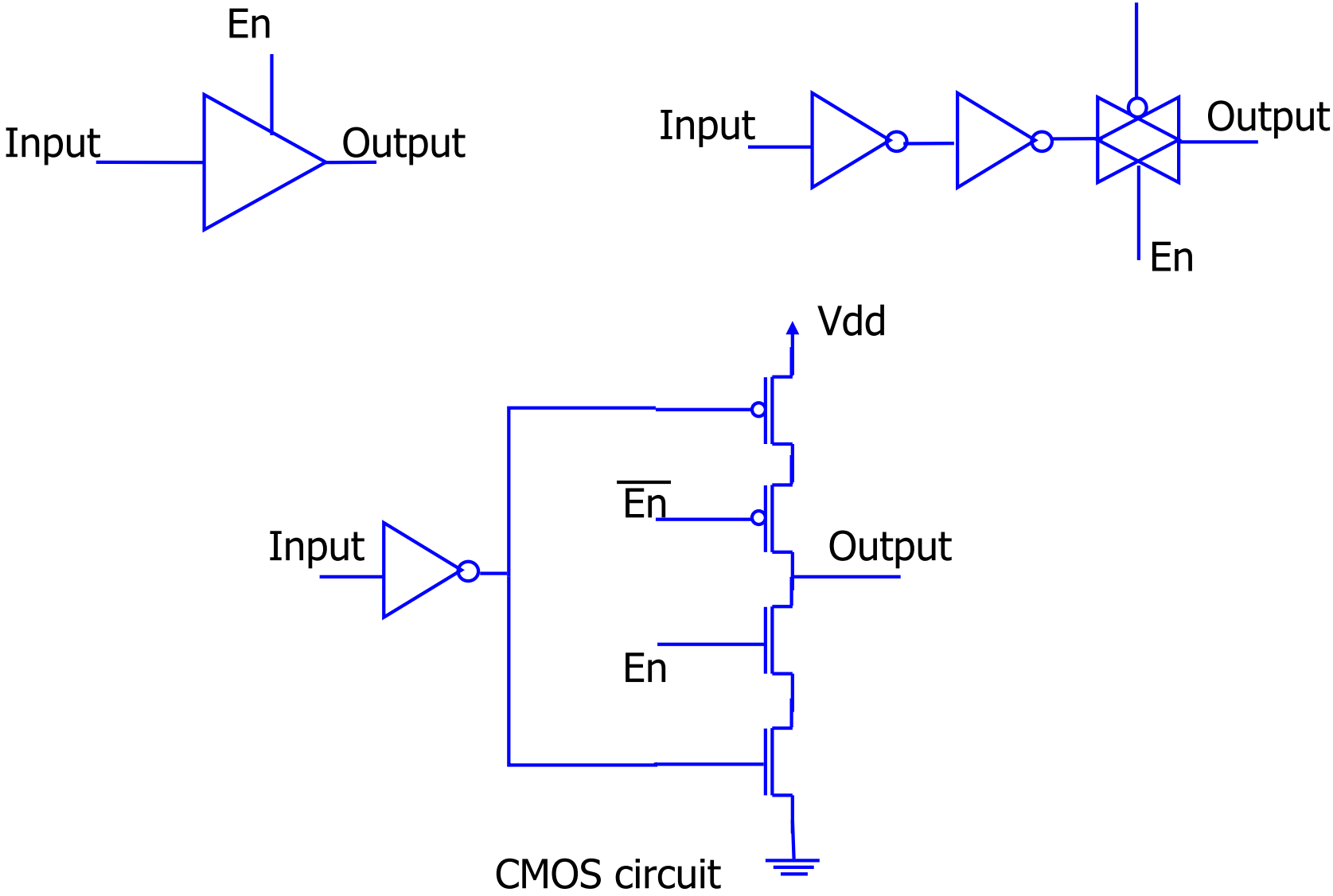
- Typically used for signal traveling, e.g. bus
- Ideally all devices connected to a bus should be disconnected except for active device reading or writing to bus
- Use high-impedance state to simulate disconnecting



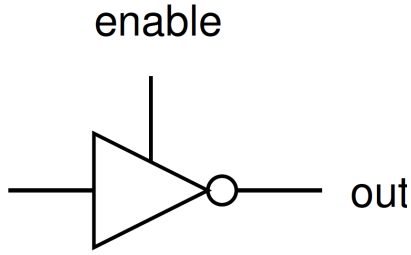
Active-high buffer

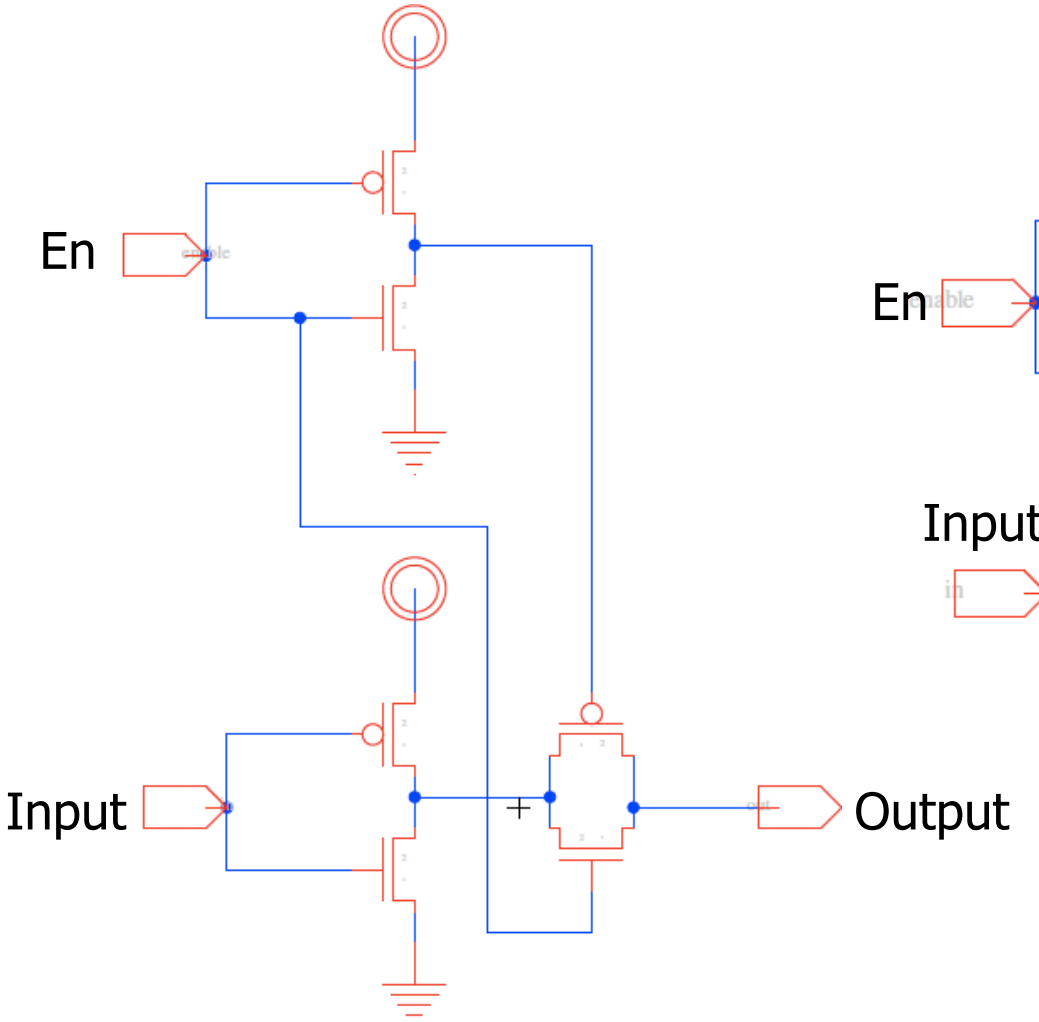
Input	En	Ouptut
0	0	Z
1	0	Z
0	1	0
1	1	1

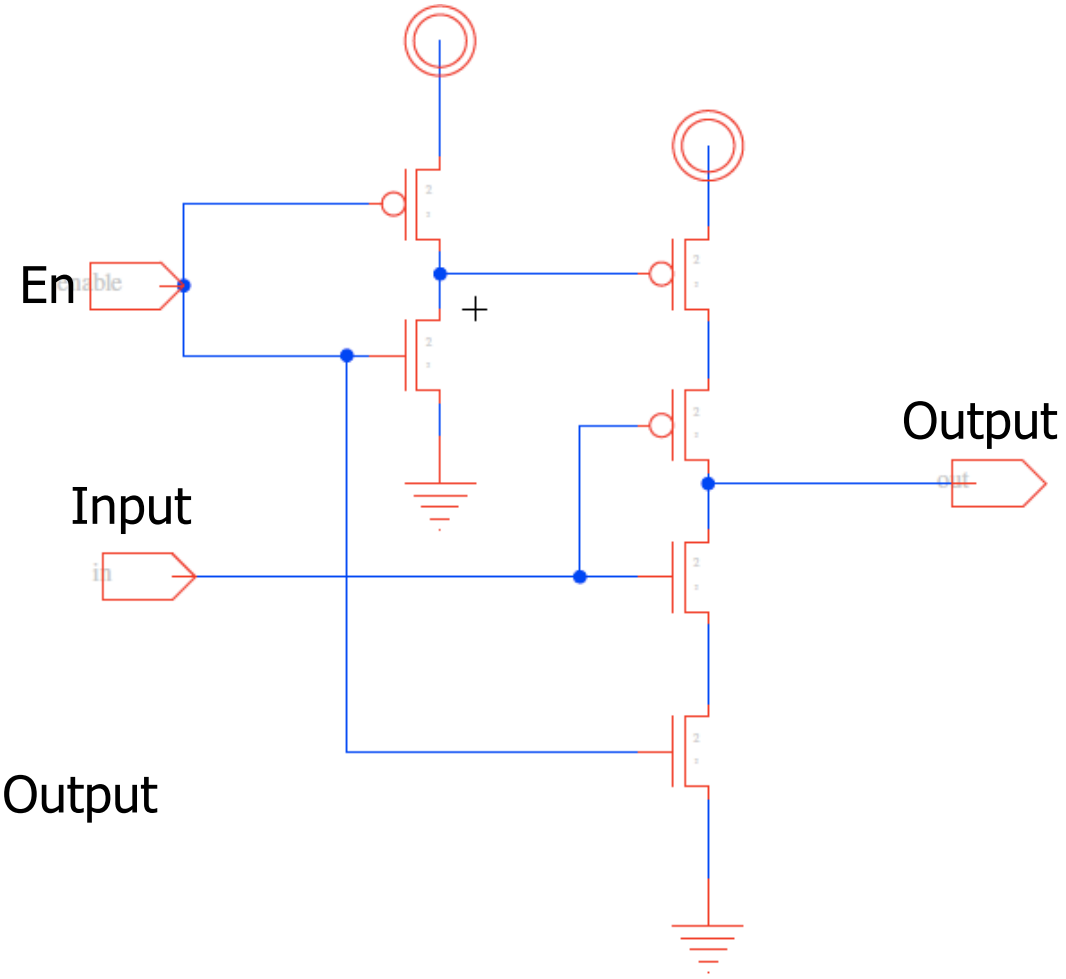
Tristate Buffer



Tristate Inverters





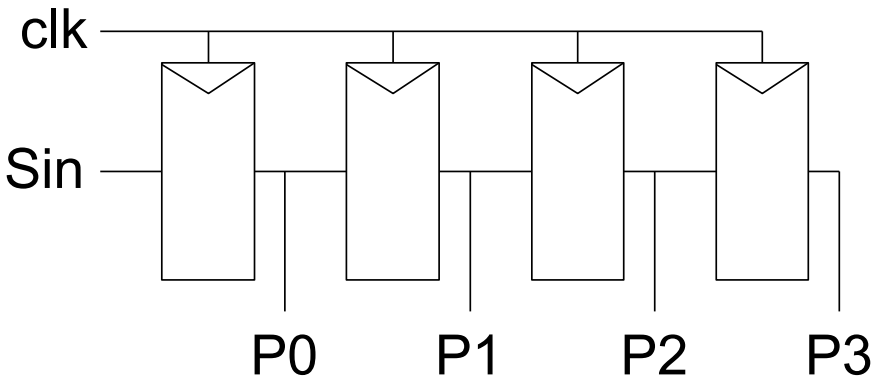


Serial Access Memories

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

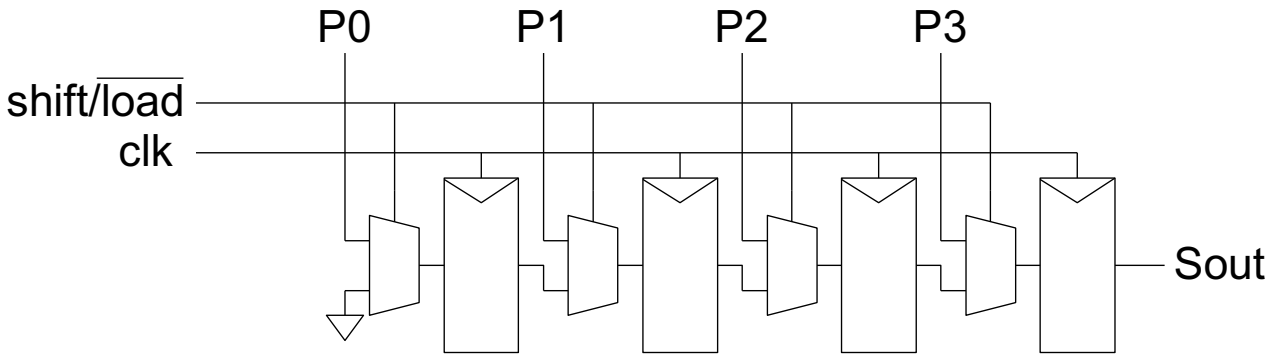
Serial In Parallel Out

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



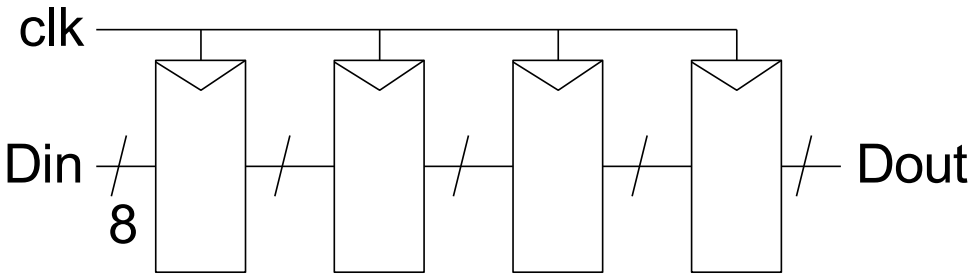
Parallel In Serial Out

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



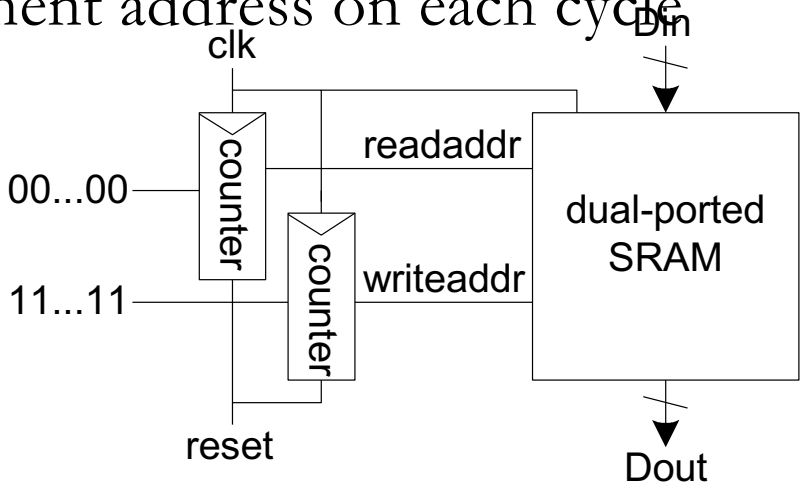
Shift Register

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



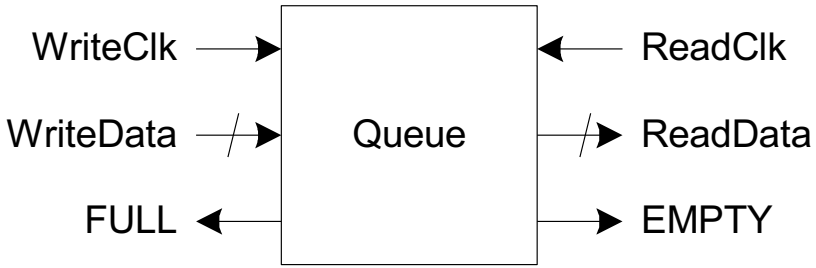
Denser Shift Registers

- 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
- Share circuitry across many bits
  - Minimize area per bit → maximize density
- Aggressively use:
  - Pass transistors, Ratioing
  - Precharge, Amplifiers to keep area down

Admin

- Homework 6 due tonight
- Project 2 out Monday 11/15
  - Work in teams of up to two
  - Milestone due Monday 11/22
  - I will give feedback by Wednesday (night) 11/24
  - Final report due 12/3