Memory Design

張孟洲

國立彰化師範大學電子系教授

Outline

- **Memory Classification**
- Memory cells
    - Static memory cell
    - Dynamic memory cell
    - ROM cells
- Address decoders

Classification of Memories

RWMemory		NVRWM	ROM
Random Access	Non-Random Access	EPROM EEPROM FLASH	Mask Programmed PROM (Fuse Programmed)
SRAM (Static) DRAM (Dynamic)	FIFO (Queue) LIFO (Stack) SR (Shift Register) CAM (Content Addressable)		

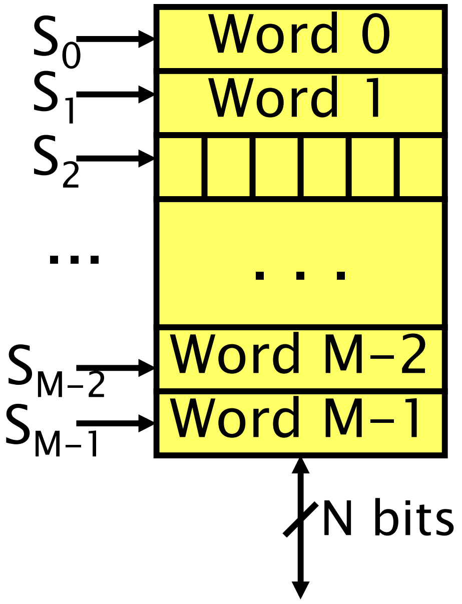
Memory Design (cont.)

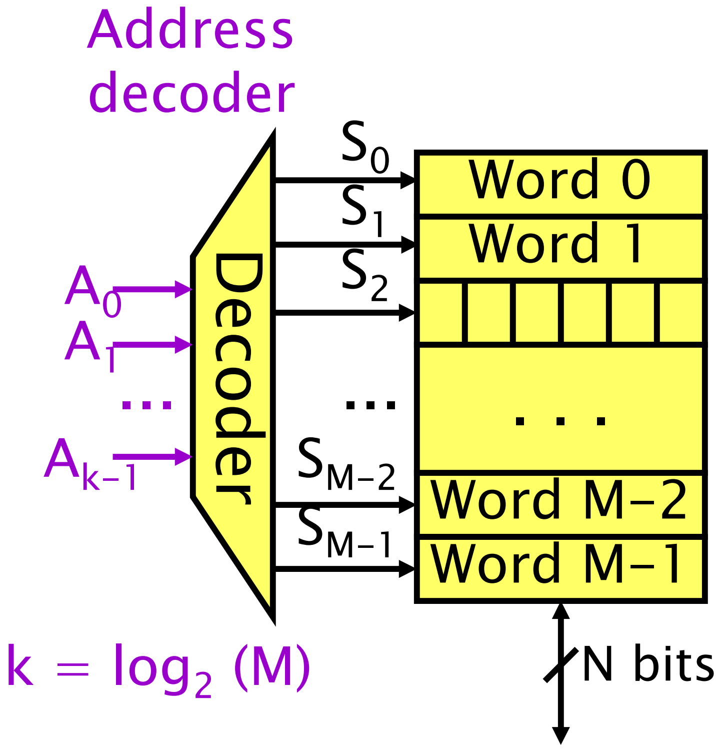
- Static vs. dynamic RAM
  - Dynamic needs refreshing
    - Refreshing: read, then write back to restore charge
    - Either periodically or after each read
- Static (SRAM)
  - Data stored as long as supply voltage is applied
  - Large (6 transistors/cell)
  - Fast
- Dynamic (DRAM)
  - Periodic refresh required
  - Small (1-3 transistors/cell)
  - Slower
  - Special fabrication process

[©Prentice Hall]

Memory Architecture: the Big Picture

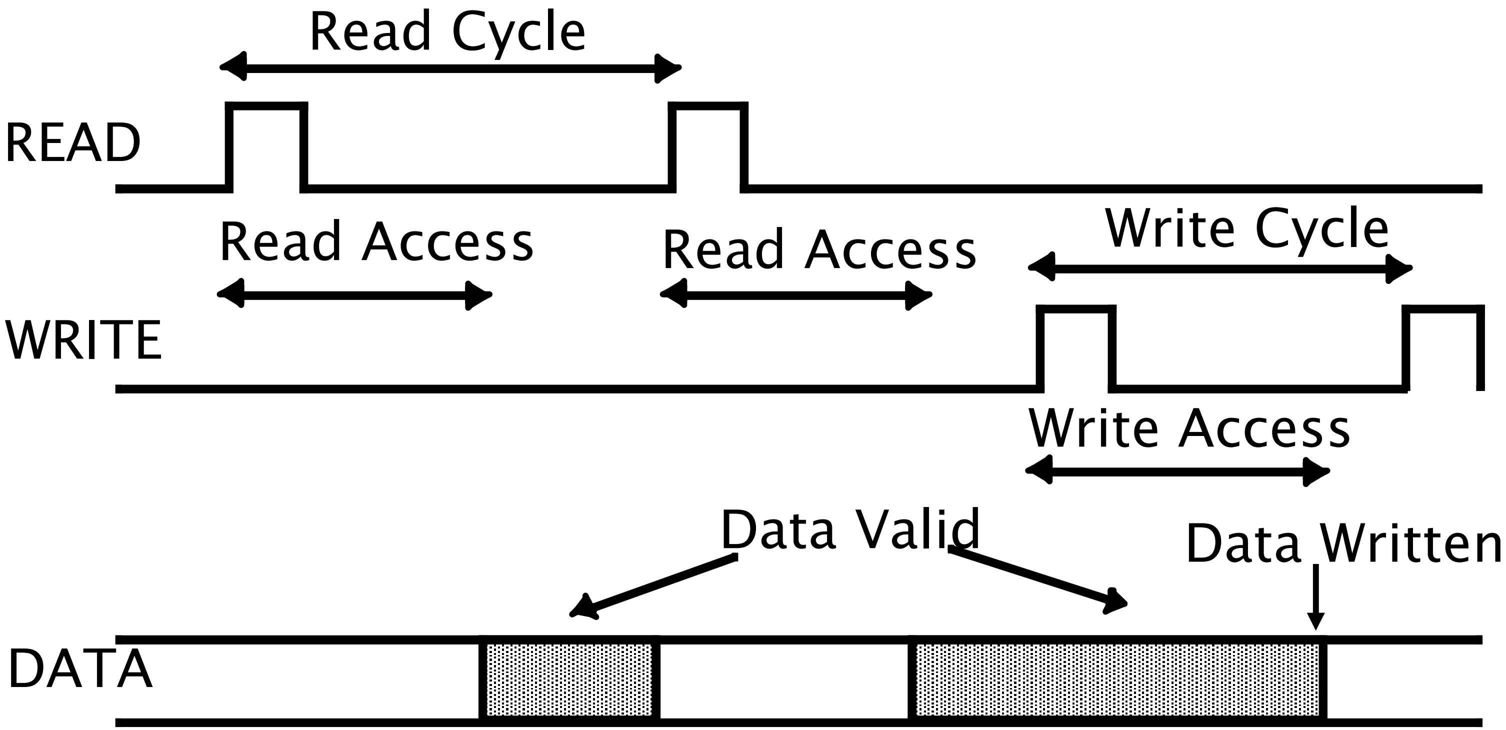
- Address: which one of the M words to access
- Data: the N bits of the word are read/written





Memory Access Timing: the Big Picture

- Timing:
  - Send address on the address lines, wait for the word line to become stable
  - Read/write data on the data lines



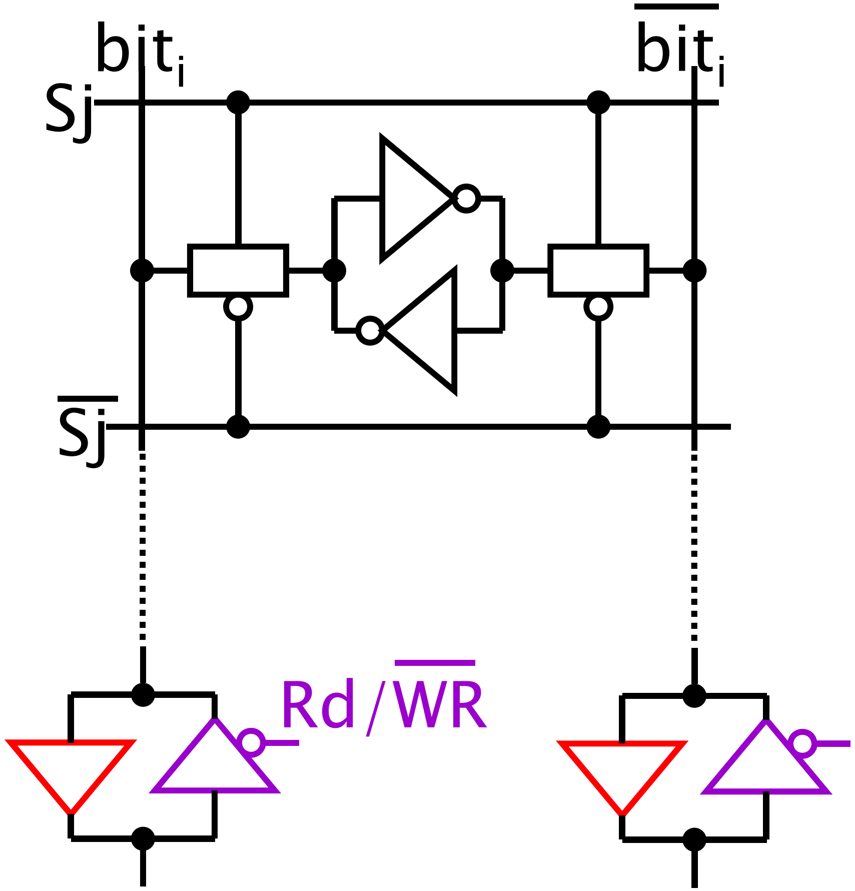
[©Prentice Hall]

Outline

- Registers (flip-flops), shift registers
- Memory interface
- Memory cells
  - **Static memory cell**
  - Dynamic memory cell
  - ROM cells
- Address decoders
- Content addressable memory (CAM)
- Non volatile memory cells

Memory Cell: Static RAM (8 transistors)

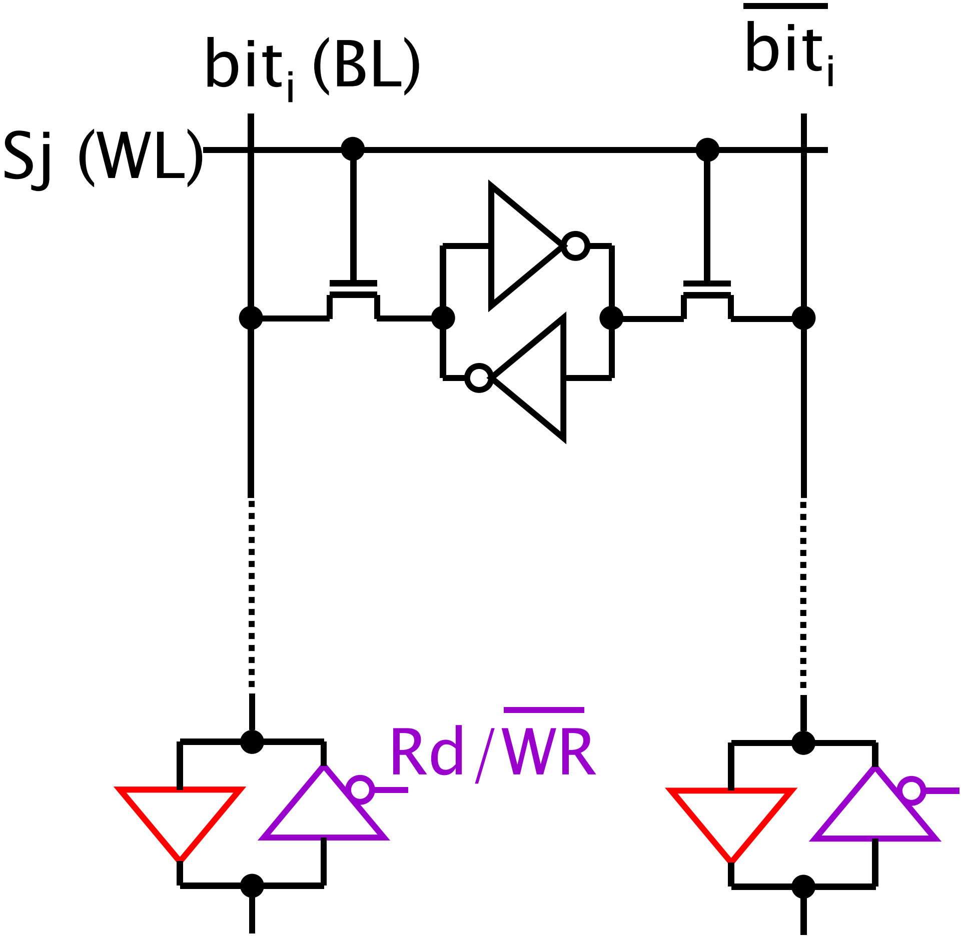
- 8-transistor cell
  - Bit_i is the data bus
  - Sj is the word line
  - Bit’ used to reduce delay
- Bus drivers
  - Sense Amplifier (inverter with high gain) used for fast switching
  - Make sure inverters in cell are weaker than the combination of “write buffer” and pass transistor



[©Hauck]

Memory Cell: Static RAM (6 transistors)

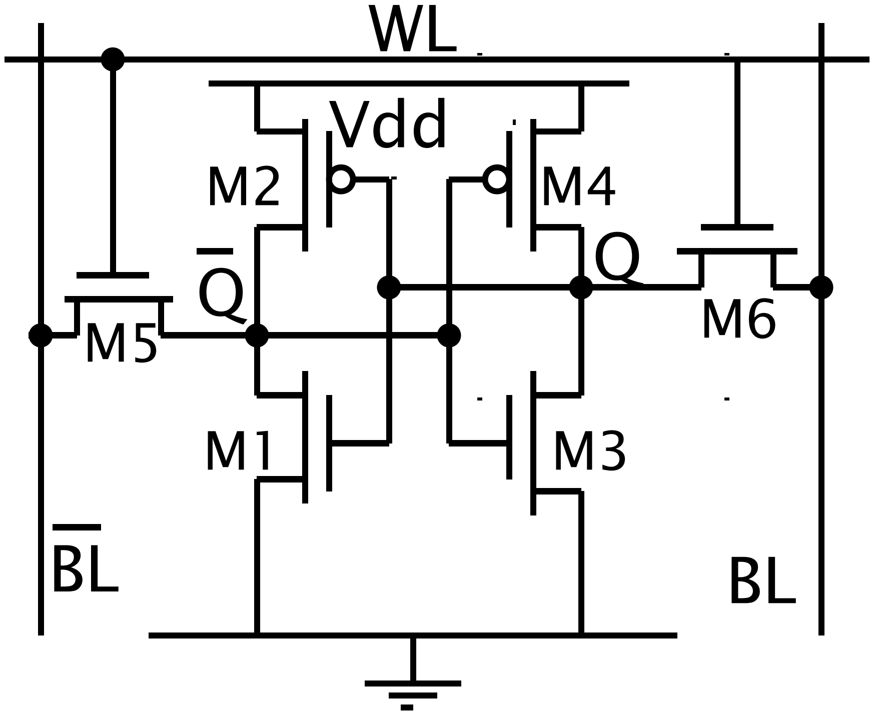
- 6-transistor cell
  - Must adjust inverters for input coming through n-type pass gate
- Bus drivers
  - Must adjust senseAmp for input coming through n-type pass gate
  - Harder to drive 1 than 0 through write buffer (high resistance via n-transistor)
  - One side is sending 0 anyway (bit or bit') ➔ written correctly

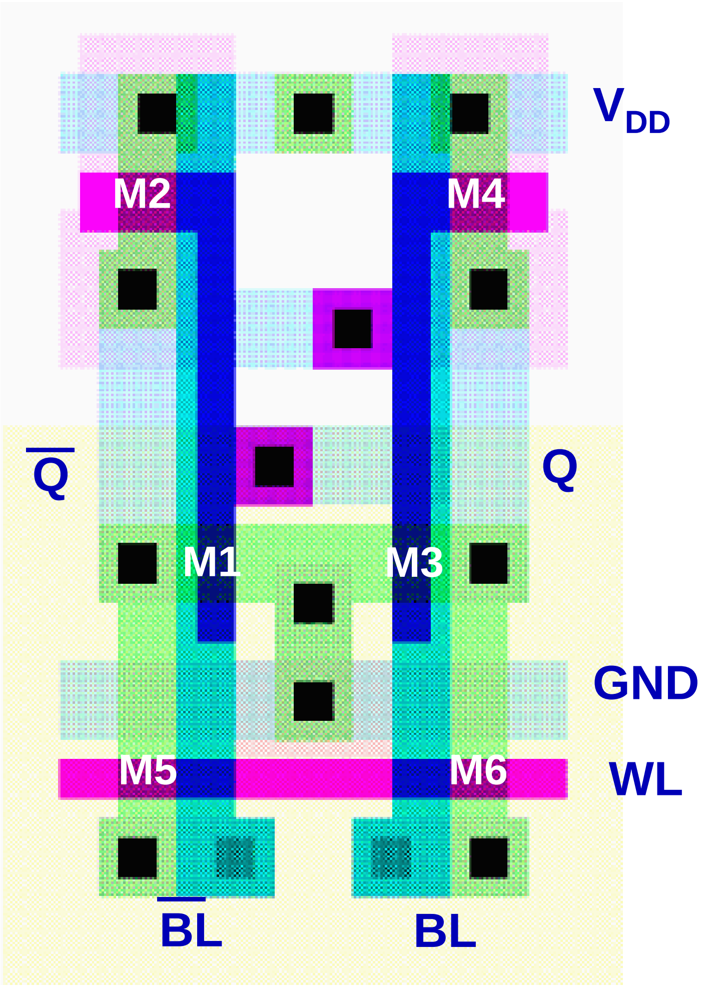


[©Hauck]

6-Transistor SRAM Cell: Layout

- WL is word line (select line Sj)
- BL is bit line (bit_i)

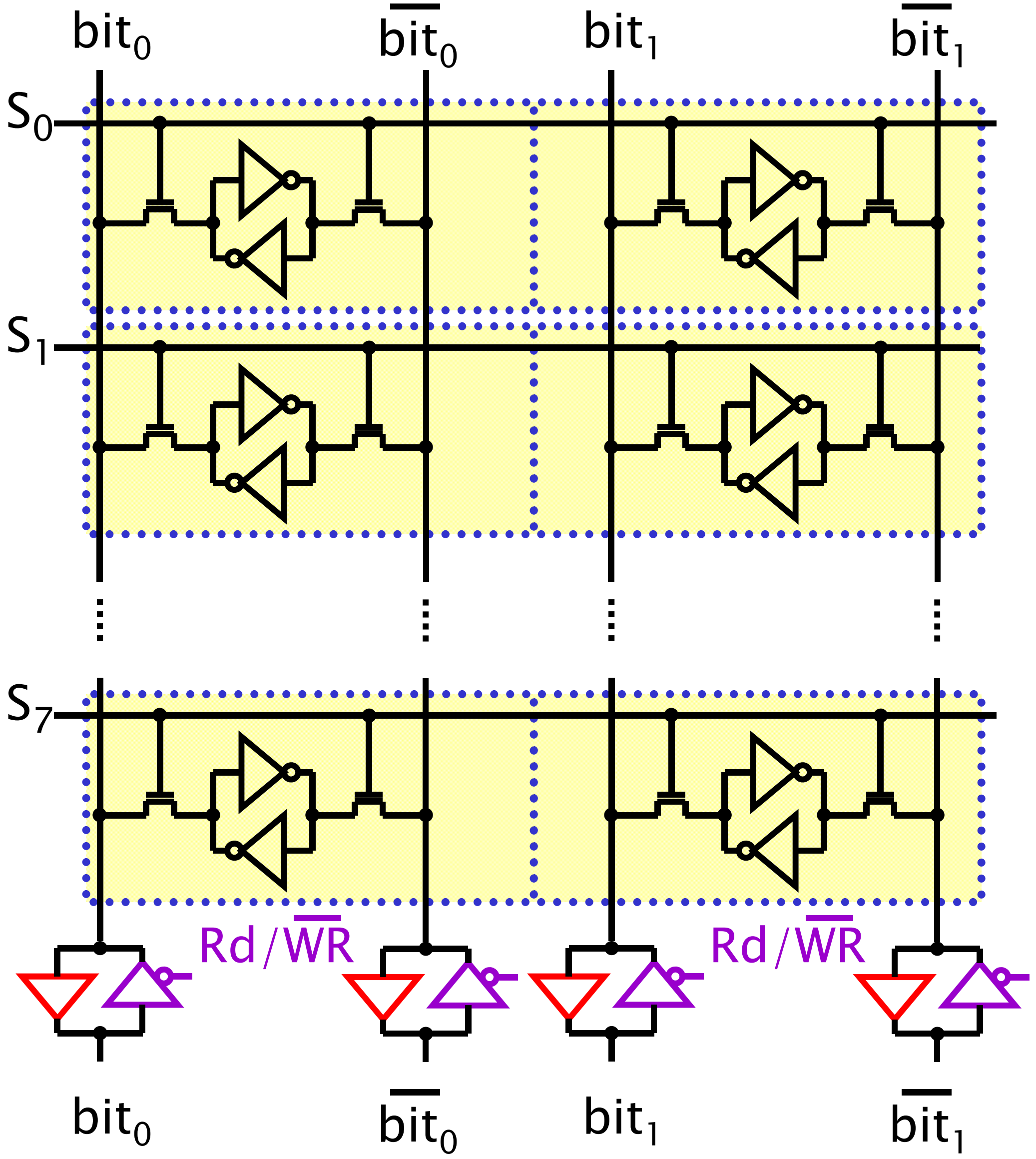




[Rab96] p.578
[©Prentice Hall]

6-Transistor Memory Array

- 8 words deep RAM, 2 bits wide words
- To write to word j:
  - Set S_j=1, all other S lines to 0
  - Send data on the global bit_0, bit_0', bit_1, bit_1'
- To read word k:
  - Set S_k=1, all other S lines to 0
  - Sense data on bit_0 and bit_1.

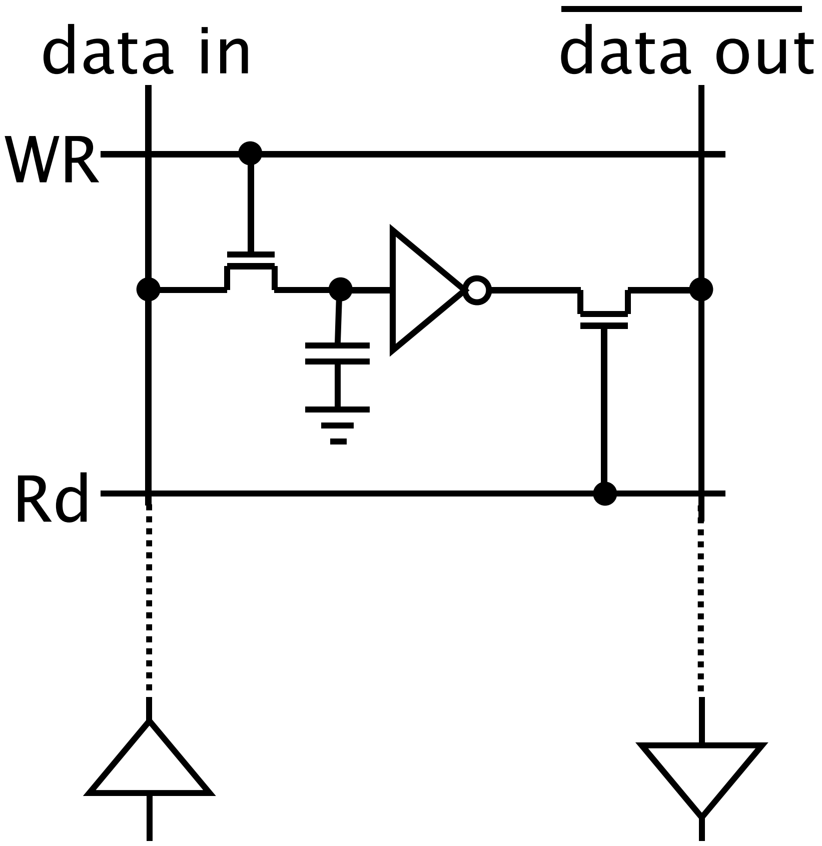


Outline

- Memory interface
- Memory cells
  - Static memory cell
  - **Dynamic memory cell**
  - ROM cells
- Address decoders

Dynamic RAM 4-Transistor Cell

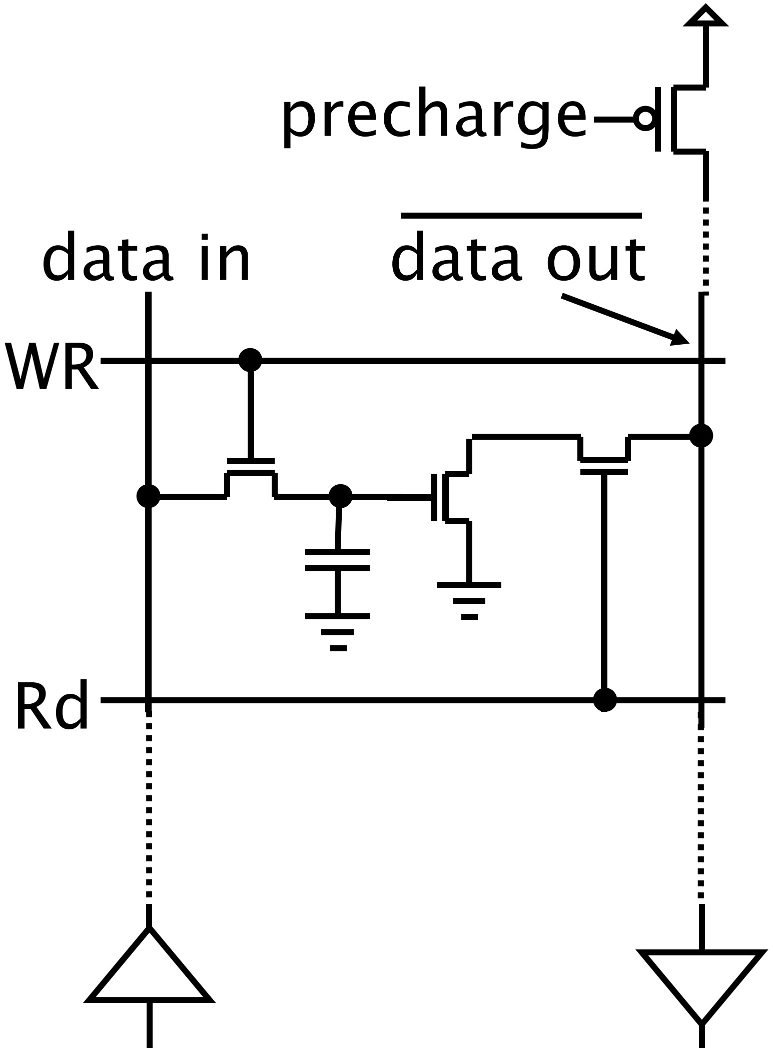
- 4-transistor cell
- Dynamic charge storage must be refreshed
- Dedicated busses for reading and writing



[©Hauck]

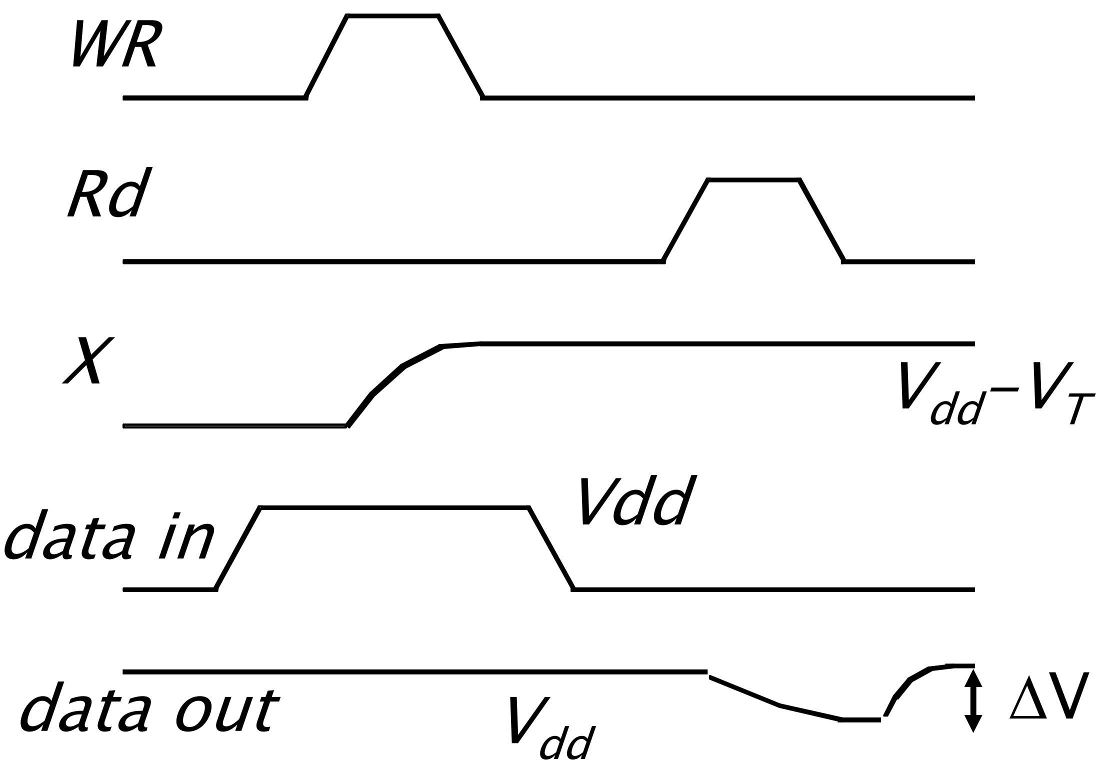
Dynamic RAM 3-Transistor Cell

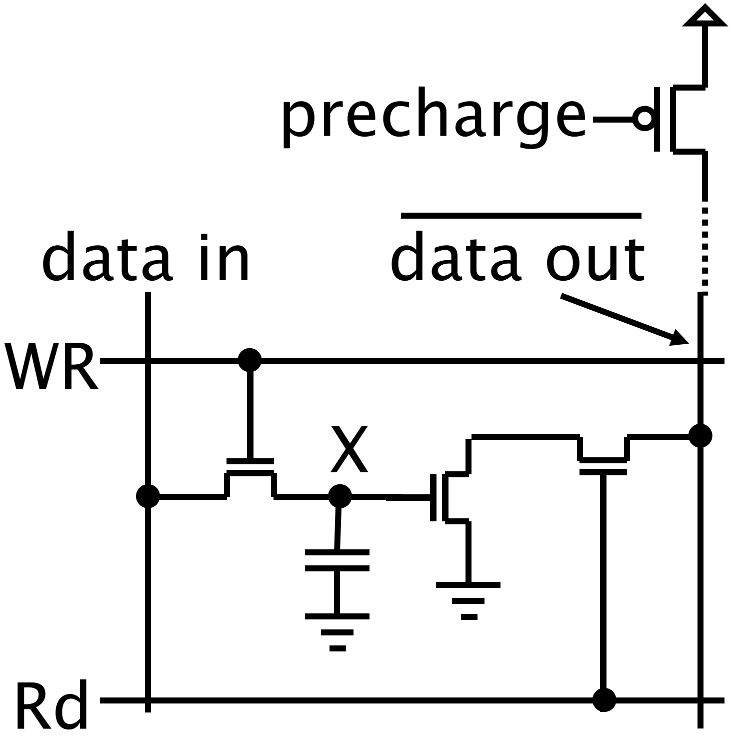
- 3-transistor cell
  - No p-type transistors yield a very compact layout for cell
  - No Vdd connection
  - Sense Amplifier must be able to quickly detect dropping voltage
  - Precharge data_out' to generate '1' outputs



[©Hauck]

Dynamic RAM 3-Transistor Cell: Timing

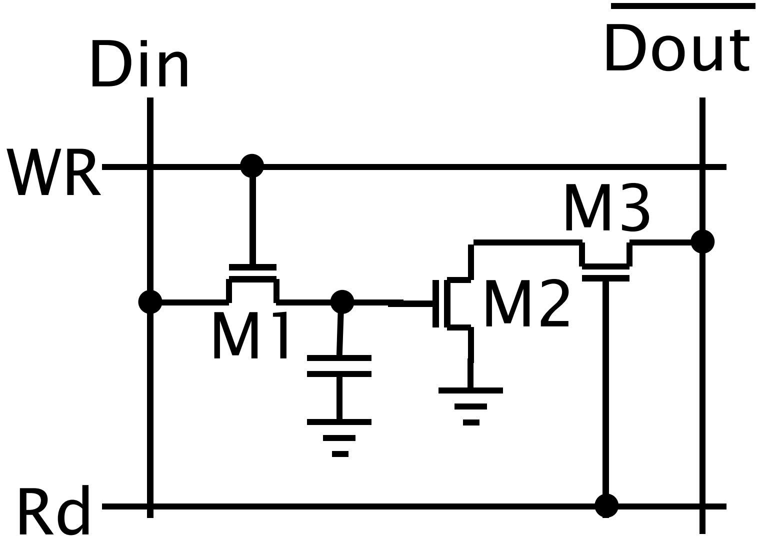


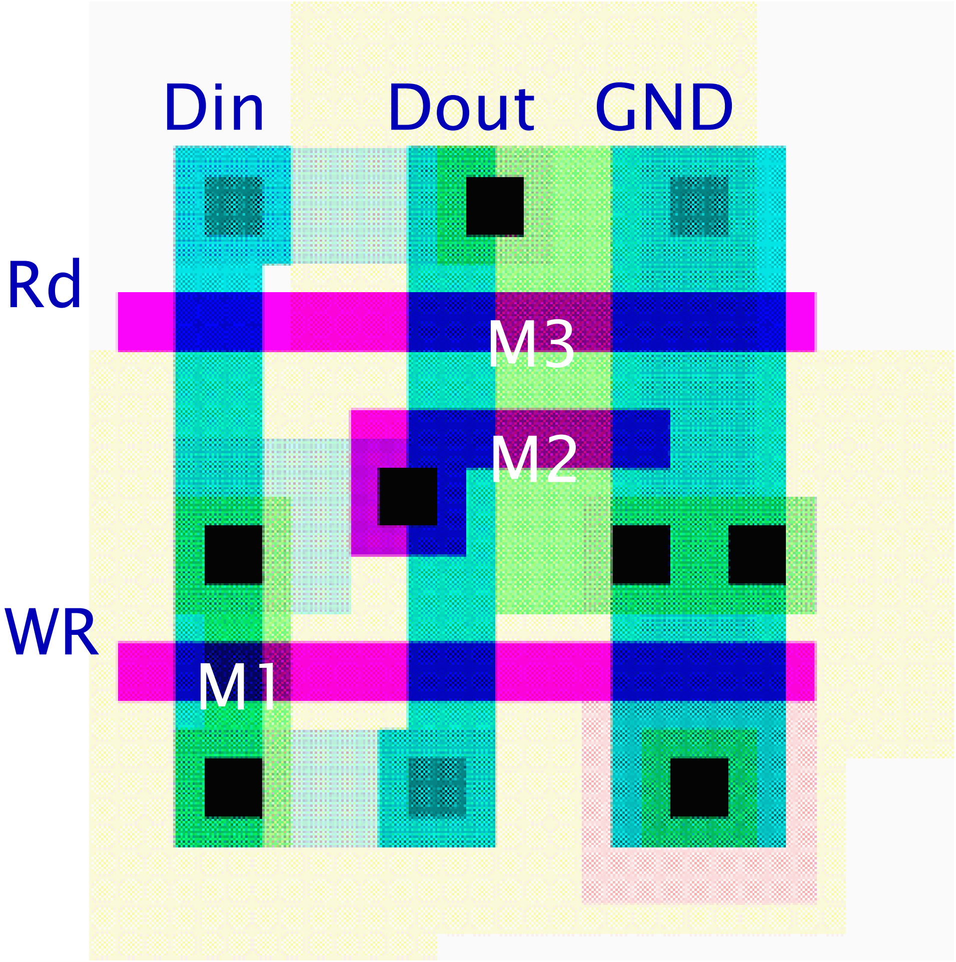


Value stored at node X when writing a “1”=V_{WR}-V_{Tn}

[Rab96] p.586

Dynamic RAM 3-Transistor Cell: Layout

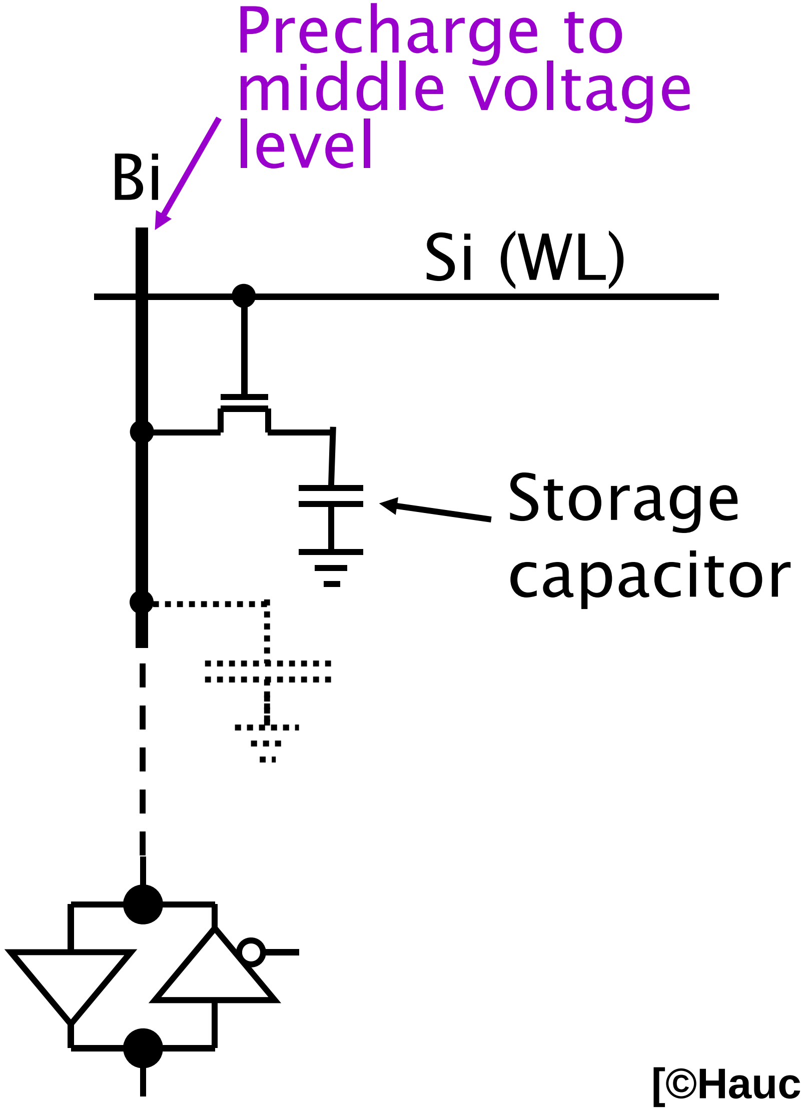




[Rab96] p.586
[©Prentice Hall]

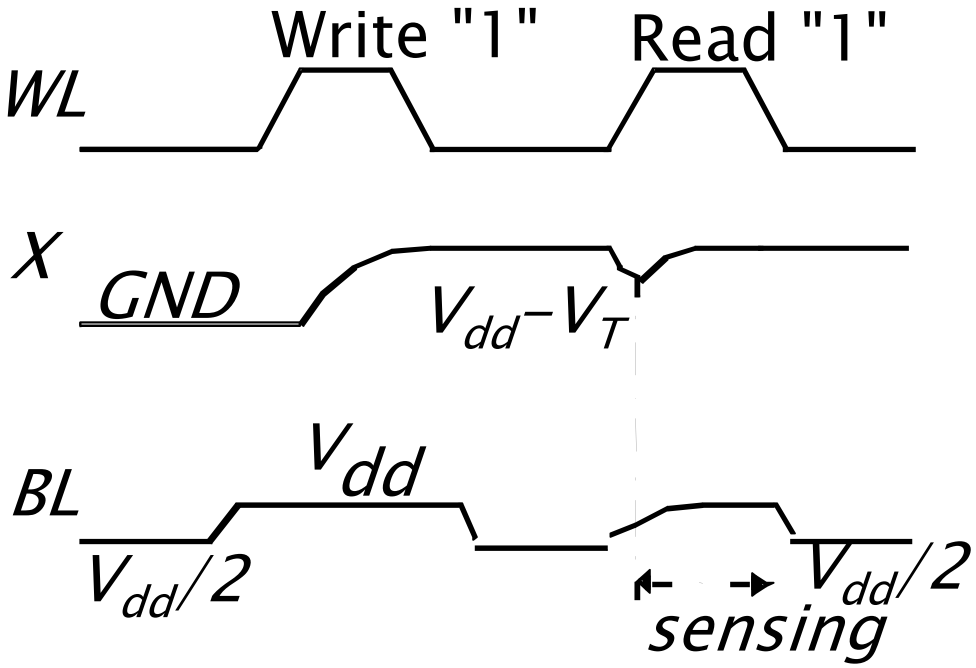
Dynamic RAM 1-Transistor Cell

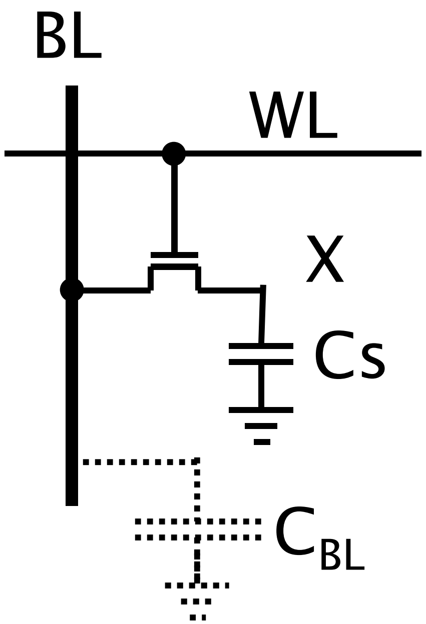
- 1-transistor cell
  - Storage capacitor is source of cell transistor
  - Special processing steps to make the storage capacitor large
  - Charge sharing with bus capacitance ($C_{cell} << C_{bus}$)
  - Extra demand on sense amplifier to detect small changes
  - Destructive read (must write immediately)



[©Hauck]

Dynamic RAM 1-Transistor Cell: Timing





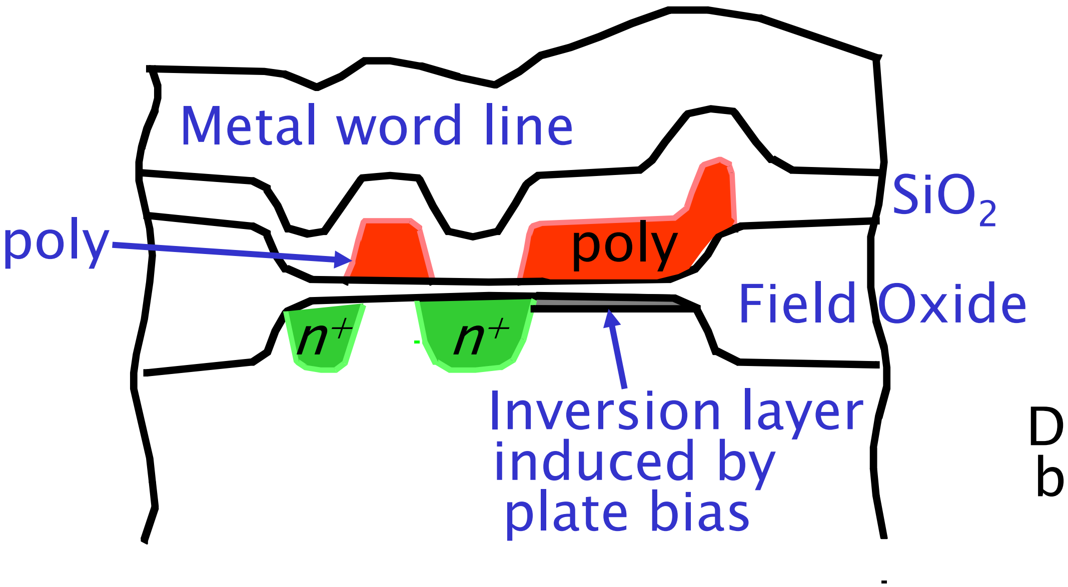
- Write: Cs is charged/discharged
- Read
  - Voltage swing is small (~250 mV)
  - $\Delta V = V_{BL} - V_{PRE} = (V_X - V_{PRE}) \cdot Cs / (Cs + C_{BL})$

[Rab96] p.587
[©Prentice Hall]

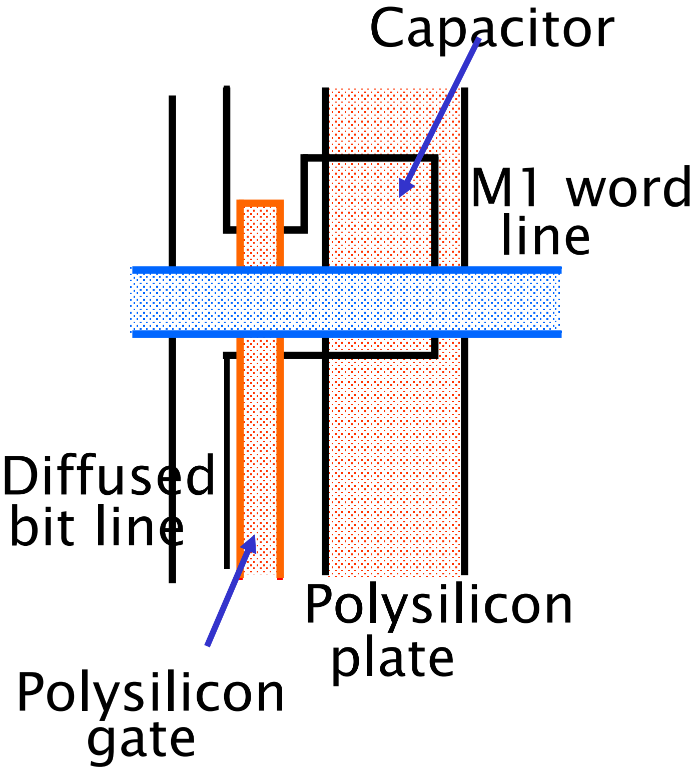
Dynamic RAM 1-Transistor Cell: Observations

- DRAM memory cell is single-ended
- Read operation is destructive
- Unlike 3T cell, 1T cell requires presence of an extra capacitance that must be explicitly included in the design
  - Polysilicon-diffusion plate capacitor
  - Trench or stacked capacitor
- When writing a “1” into a DRAM cell, a threshold voltage is lost
  - Set WL to a higher value than Vdd

Dynamic RAM 1-Transistor Cell: Layout



(a) Cross-section



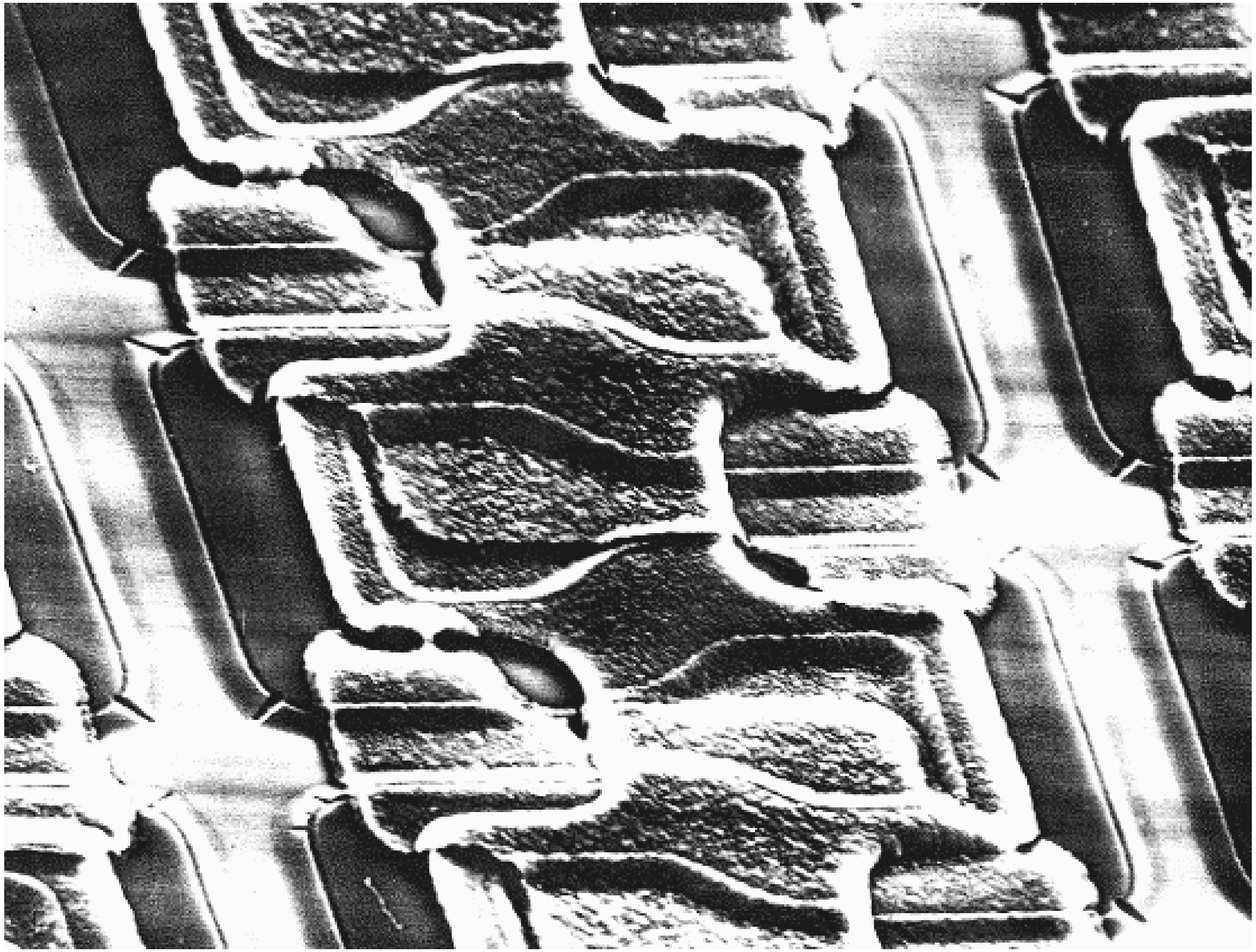
(b) Layout

Used Polysilicon-Diffusion Capacitance

Expensive in Area

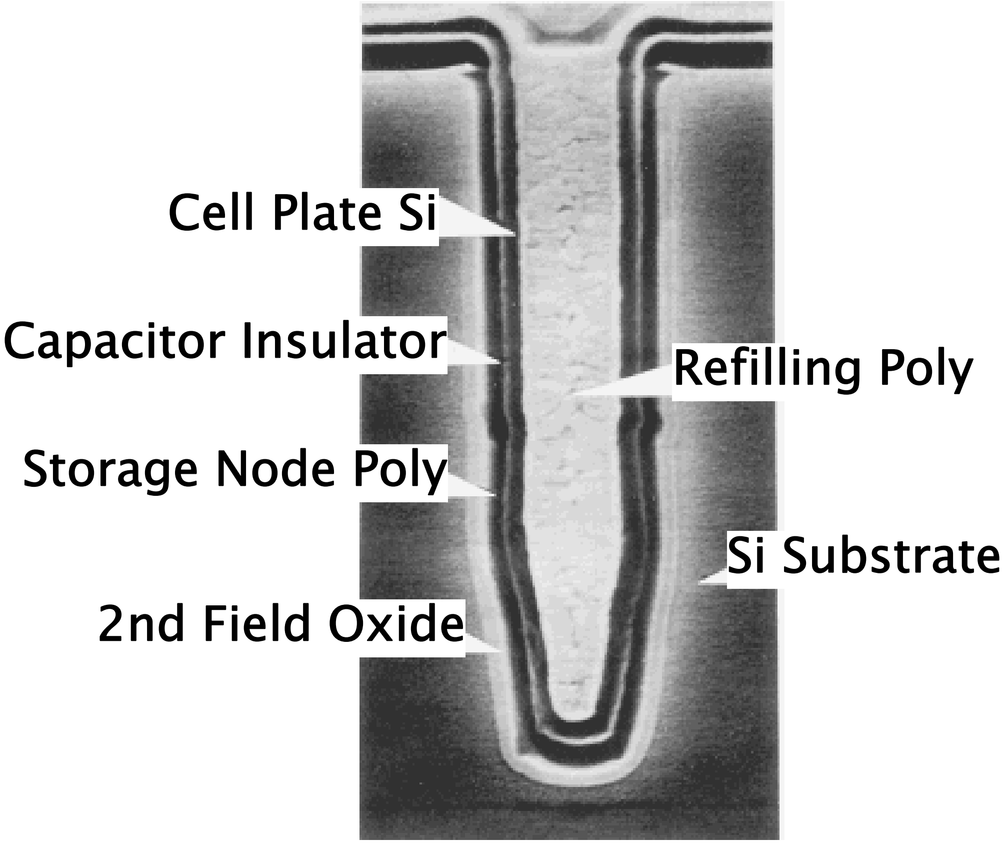
[©Prentice Hall]

Dynamic RAM 1-Transistor Cell: Layout

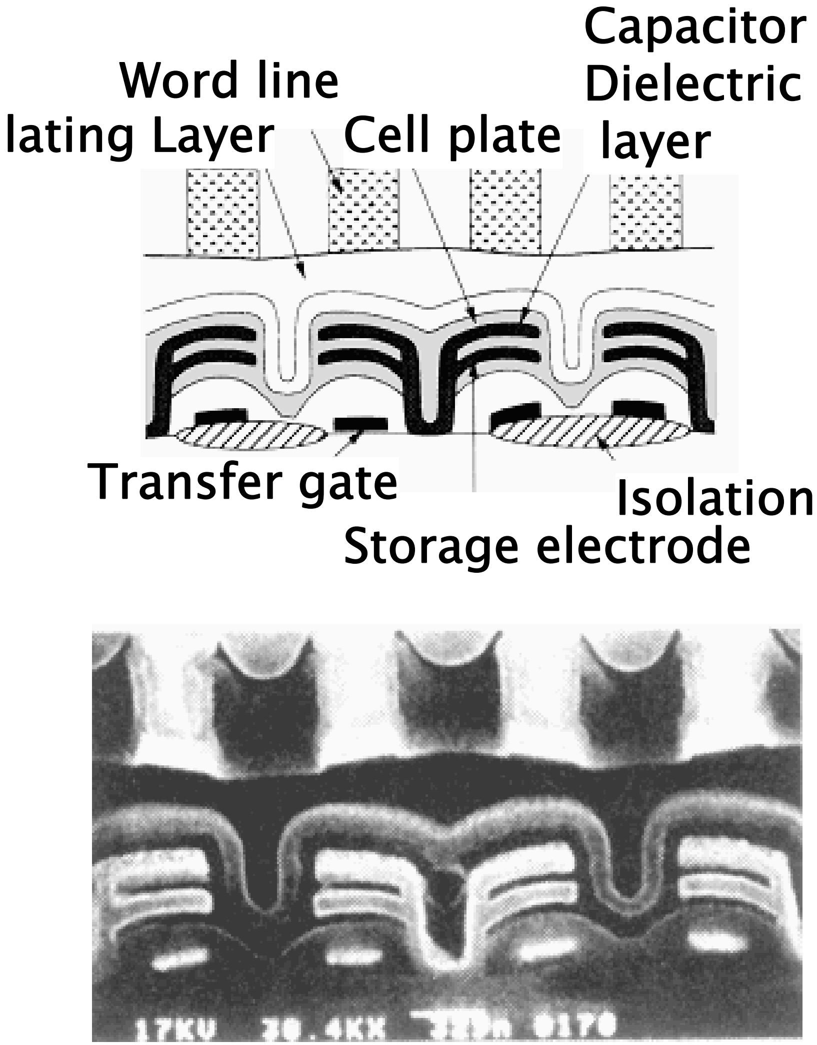


[©Prentice Hall]

Dynamic RAM 1-Transistor Cell: Layout



Trench Cell



Stacked-capacitor Cell

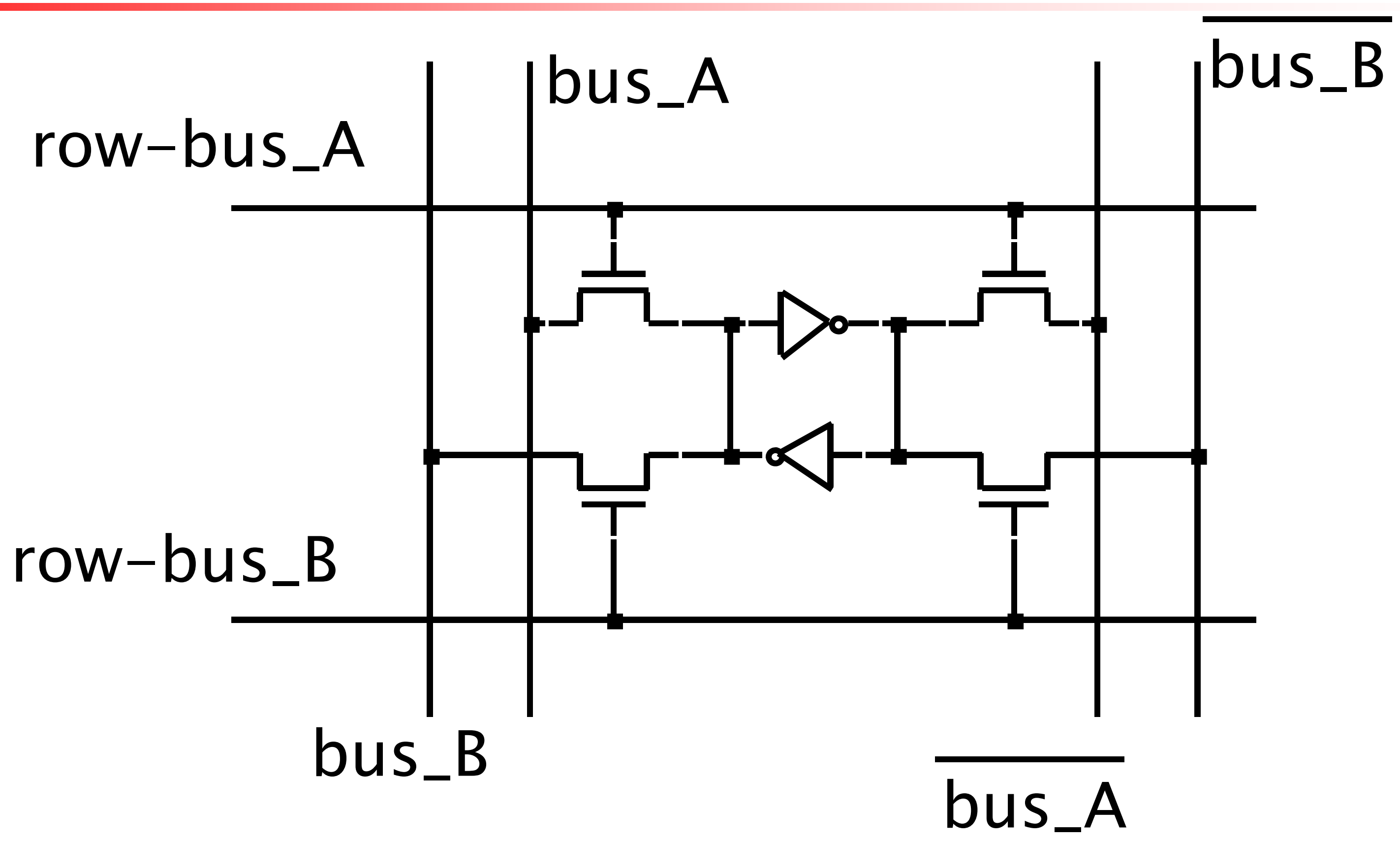
[©Prentice Hall]

RAM Cells: Summary

- Static
  - Fastest (no refresh)
  - Simple design
  - Right solution for small memory arrays such as register files
- Dynamic
  - Densest: 1T is best and is the way to go for large memory arrays
  - Built-in circuitry to step through cells and refresh (can do more than one word at a time)
  - Sense amplifier needed for fast read operation

[©Hauck]

Multi-Port RAM Cells

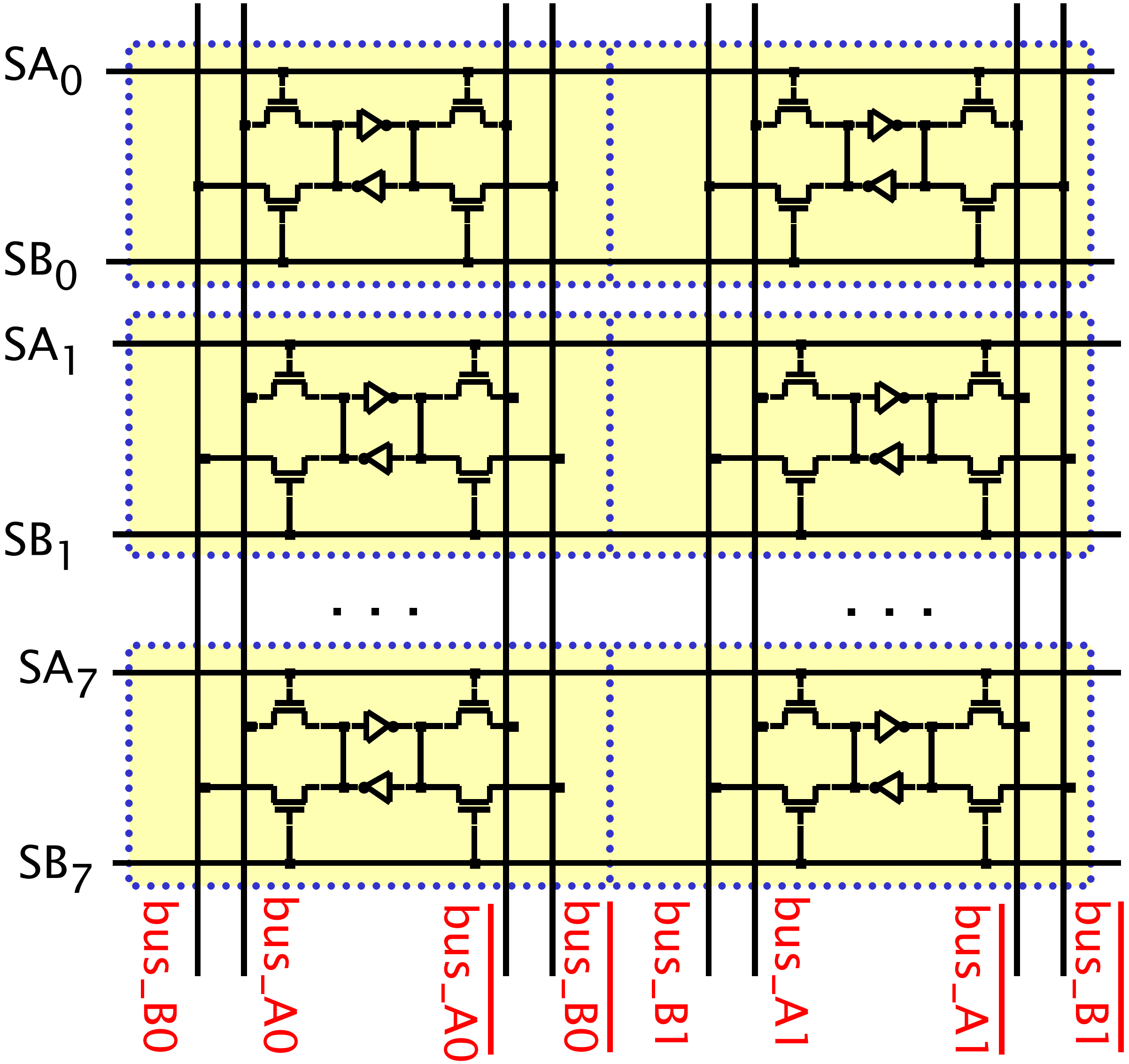


- Idea: add more input and output transistors
- Can be applied to all variants
  - Usually not done for 1T cells

[©Hauck]

Multi-Port RAM Cells Array

- 7 words deep,
  2 wide words,
  dual port mem
- To read from word j and write "d_1d_0" to word k simultaneously:
  - Set SA_j=1, and all other SA's=0
  - Set SB_k=1, and all other SB's=0
  - Sense the values on bus_A0 and bus_A1
  - Write d_1d_0 to bus_B0 and bus_B1


SA_0
SB_0
SA_1
SB_1
. . .
. . .
SA_7
SB_7
bus_B0
bus_A0
$\overline{\text{bus\_A0}}$
$\overline{\text{bus\_B0}}$
bus_B1
bus_A1
$\overline{\text{bus\_A1}}$
$\overline{\text{bus\_B1}}$

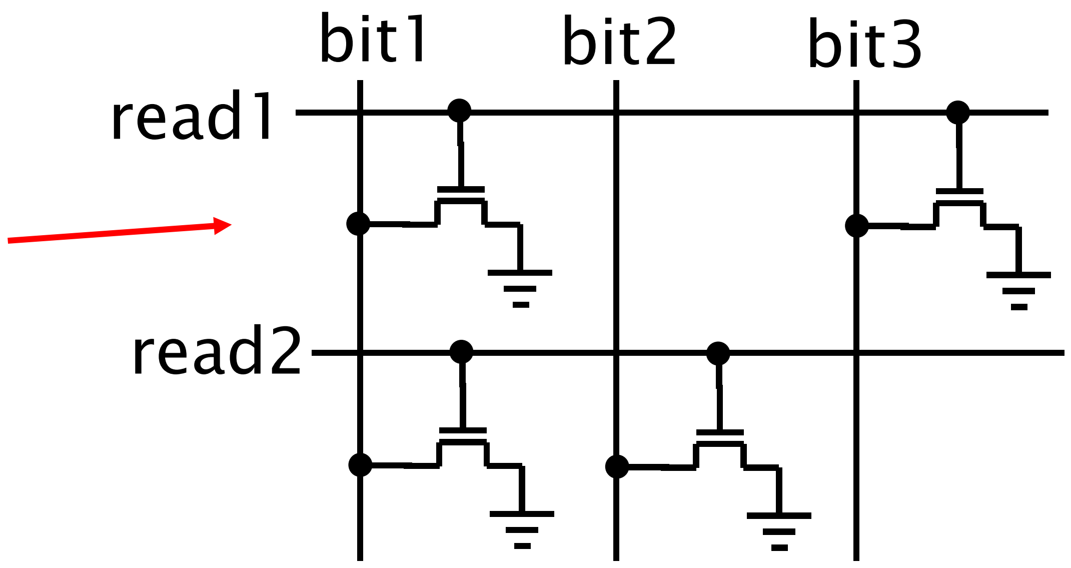
Outline

- Memory interface
- Memory cells
  - Static memory cell
  - Dynamic memory cell
  - **ROM cells**
- Address decoders

Read Only Memory (ROM) Cells: MOS NOR

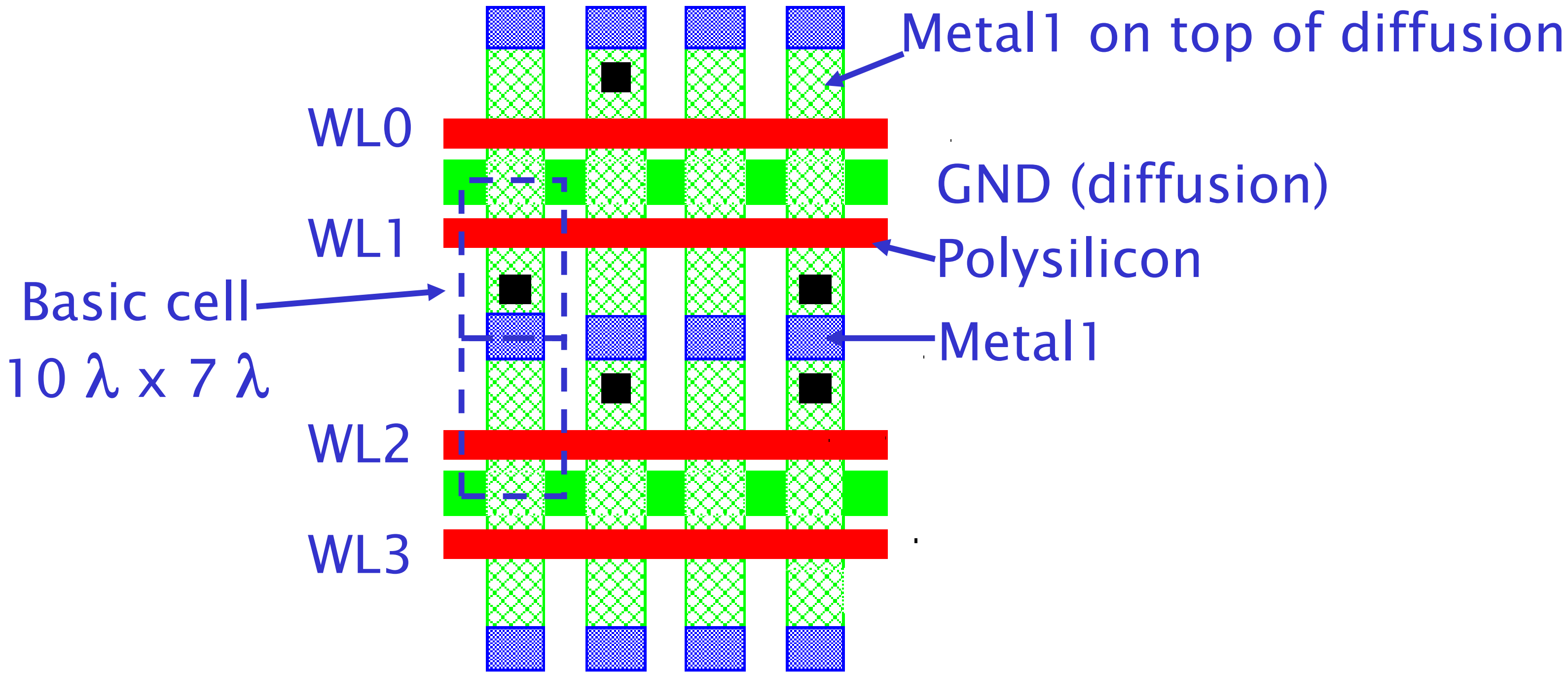
- To store constants data or invariant code
- Popular for control implementation
  - Store program or state machine
- Programmable logic array structure
- Can be precharged or pseudo-nMos

MOS NOR ROM



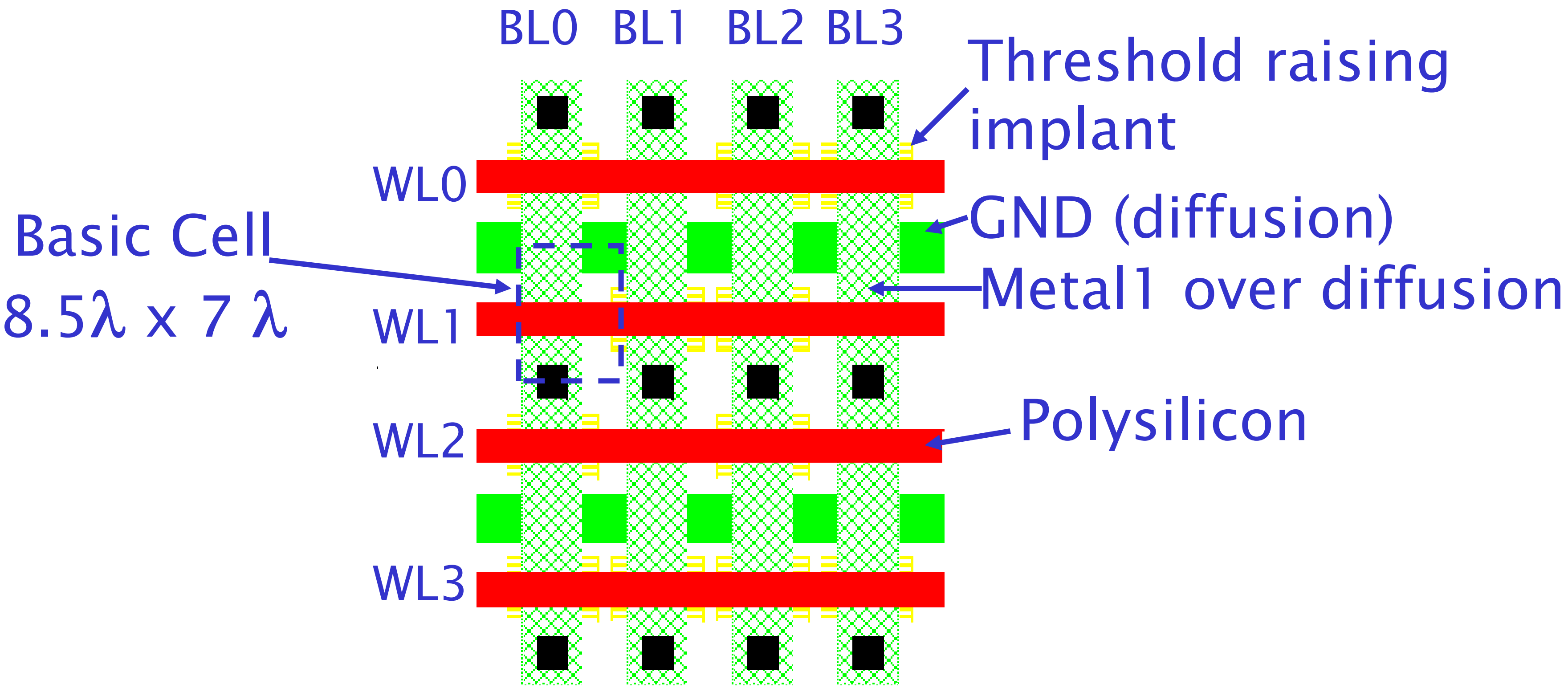
[©Hauck]

ROM Cell: MOS NOR Layout



- Only 1 layer (metal-to-diffusion contact mask) is used to program memory array
- Programming of the memory can be delayed to one of last process steps

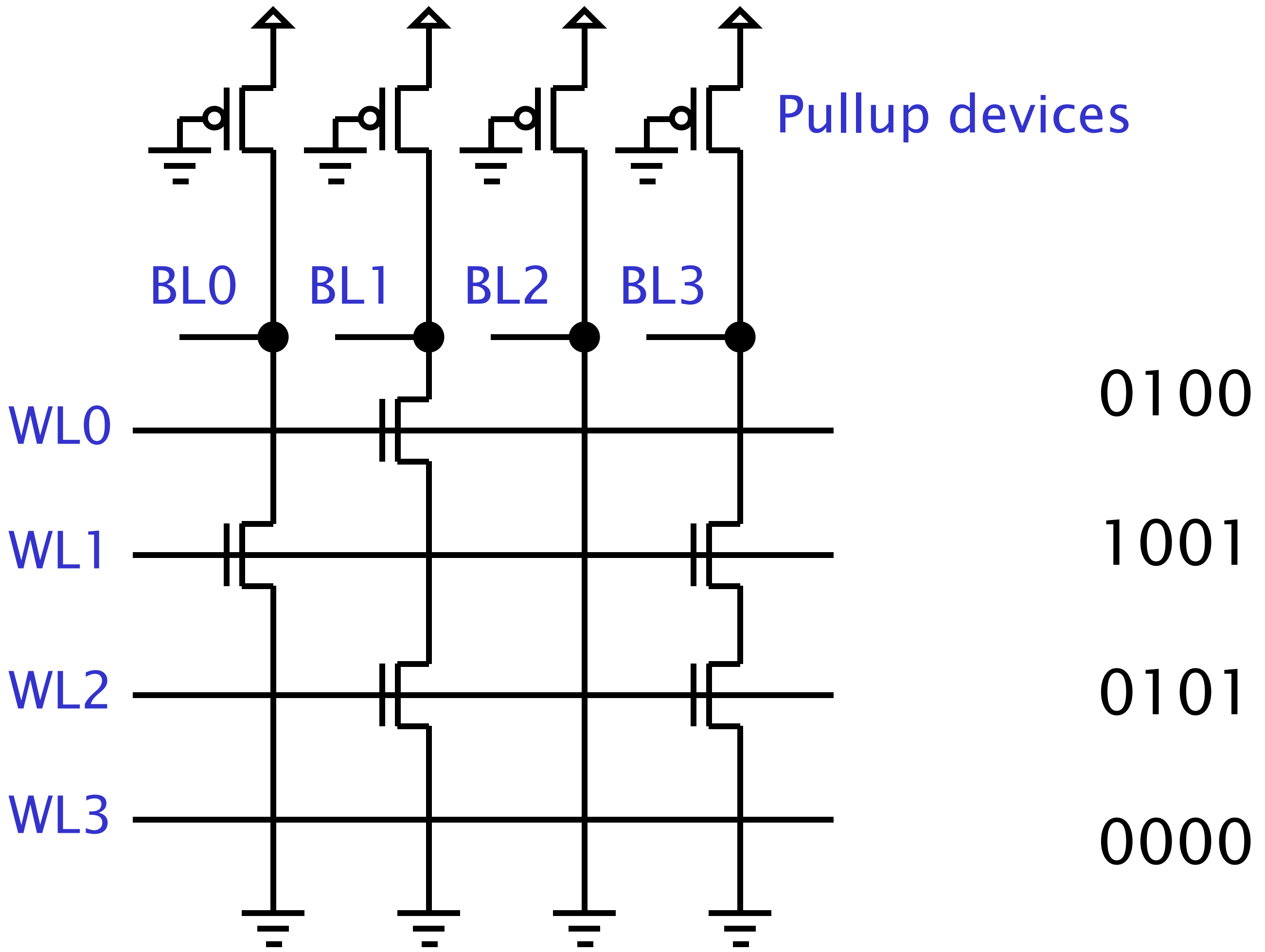
ROM Cell: MOS NOR Alternative Layout



- Threshold raising implants disable transistors

[©Prentice Hall]

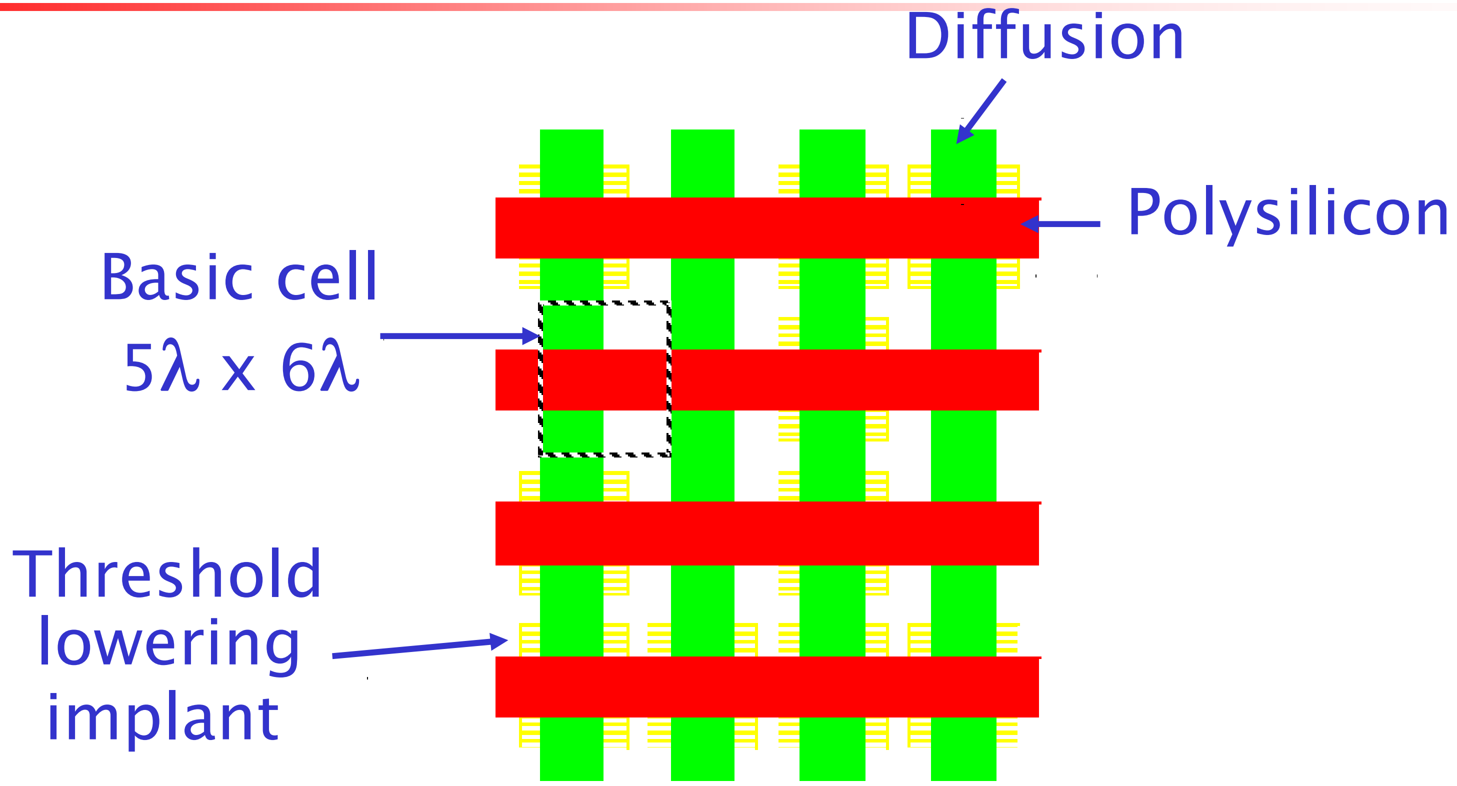
ROM Cell: MOS NAND



All word lines high by default with exception of selected row

[©Prentice Hall]

ROM Cell: MOS NAND: Layout



- No contact to Vdd or GND necessary ➔drastically reduced cell size
- Loss in performance compared to NOR ROM
  - Why?

[©Prentice Hall]

ROM Cells: Summary

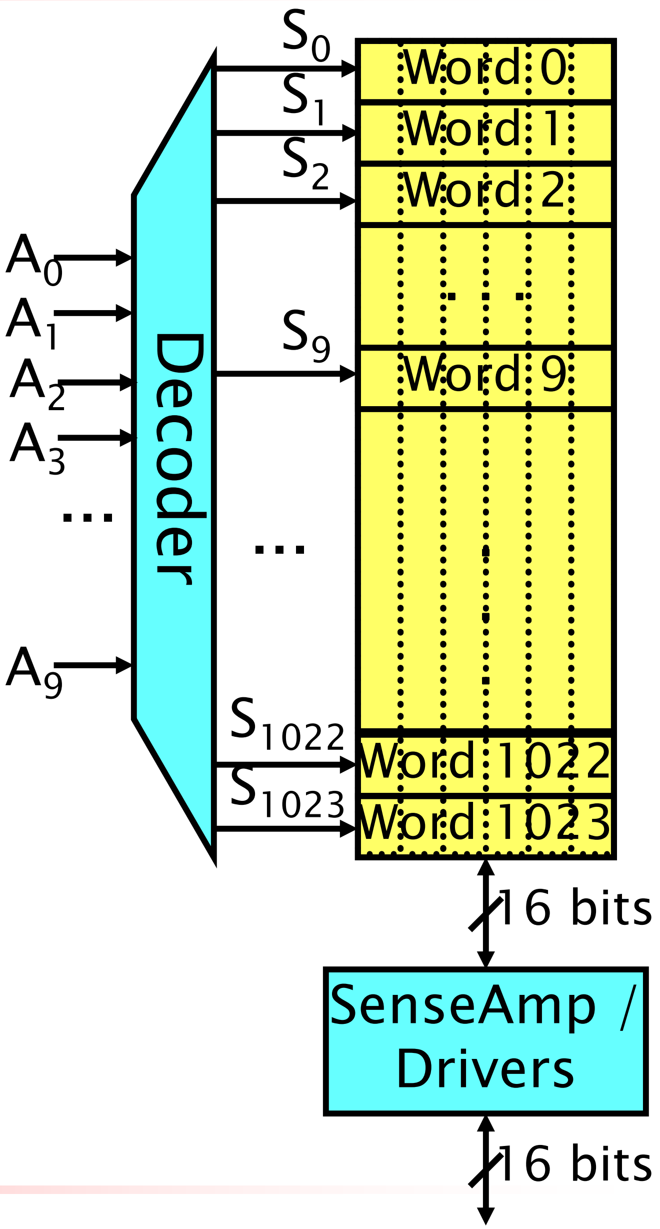
- Mask programmability
- Precharged vs. pseudo-nMos
- NAND cell, NOR cell
  - Area
  - Speed
- Other types: EEPROM, etc.

Outline

- Memory interface
- Memory cells
  - Static memory cell
  - Dynamic memory cell
  - ROM cells
- **Address decoders**

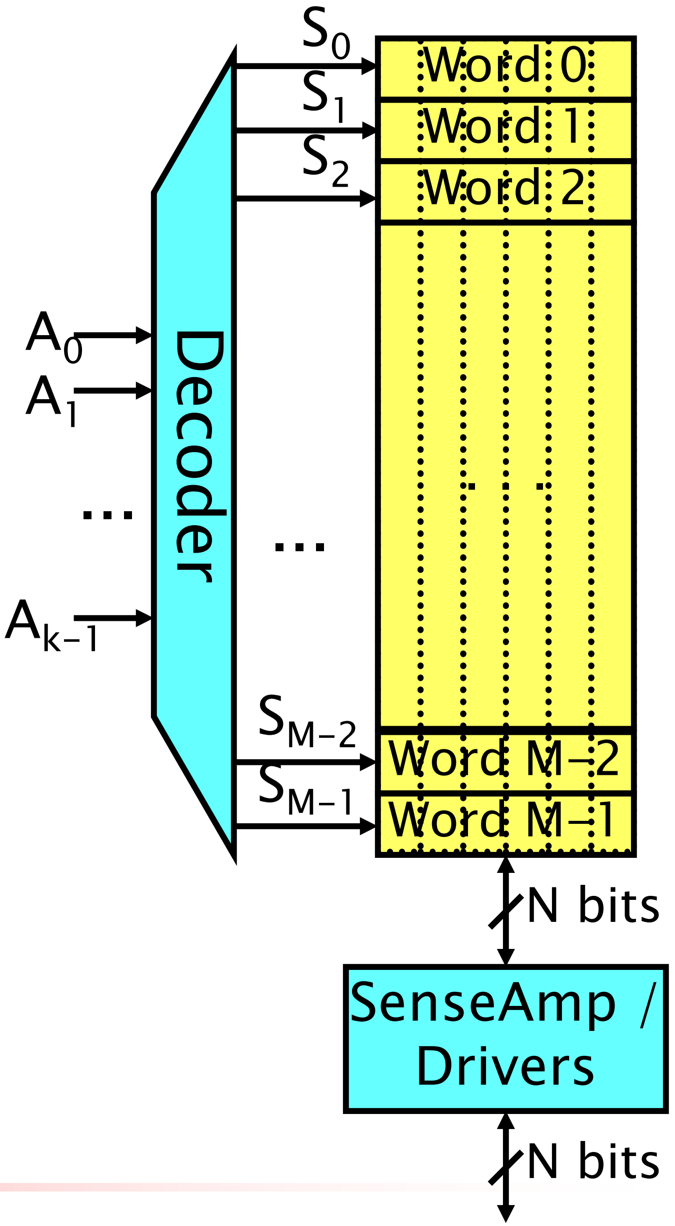
Memory Cell Array Interface: Example

- Memory parameters:
  - 16-bit wide
  - 1024-word deep
- Accessing word 9
  - Address = 0000001001_2



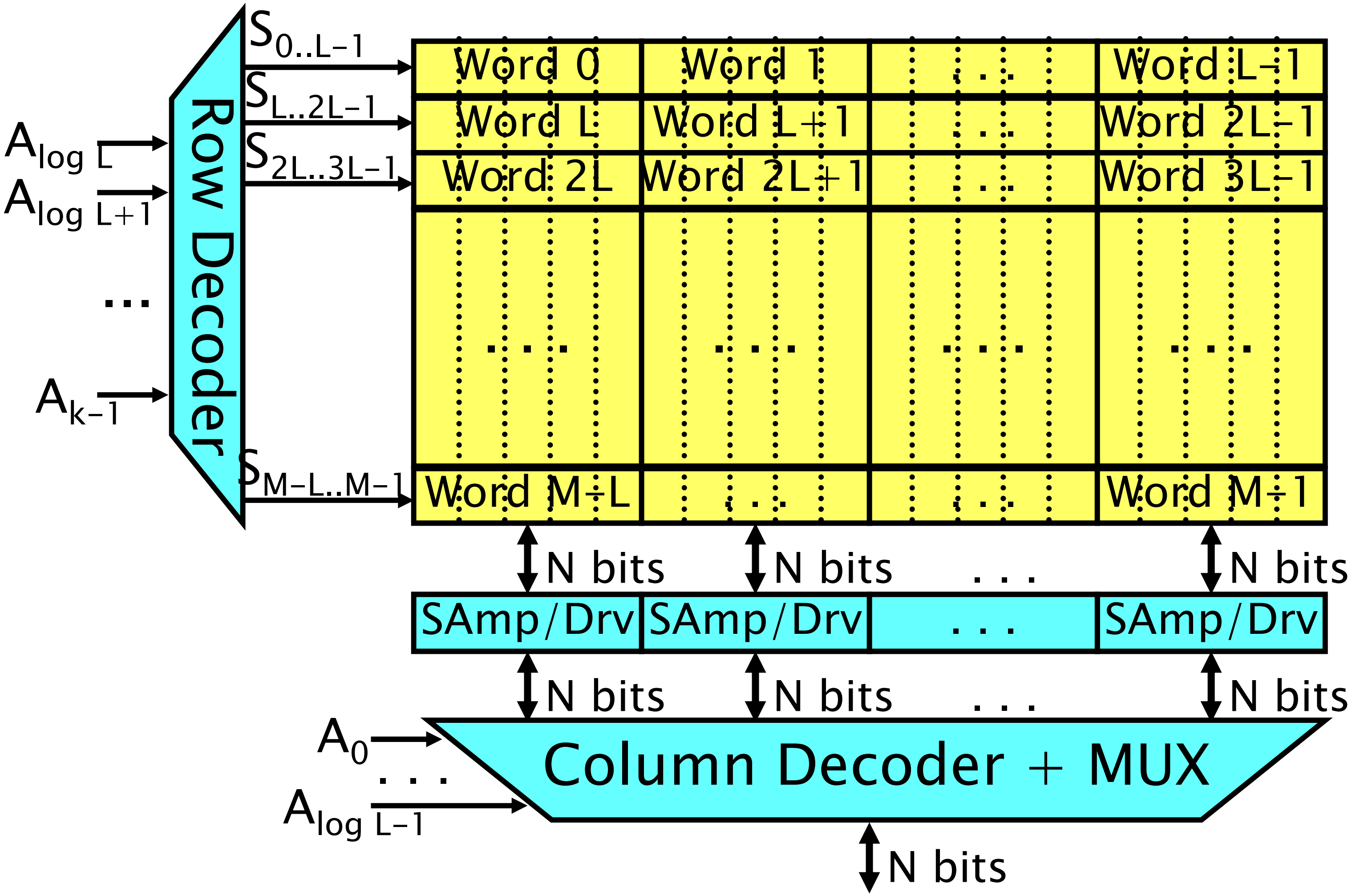
Memory Cell Array Layout

- Memory performance (speed)
  - Storage cell speed (read, write)
  - Data bus capacitance
  - Periphery: address decoders, sense amplifiers, buffers
- Memory area
  - Cell array layout
- How to layout the cells array?
  - Linear is bad:
    - Long data busses ➔ large capacity
    - A lot of cells connected to data bus
    - Decoder will have a lot of logic levels



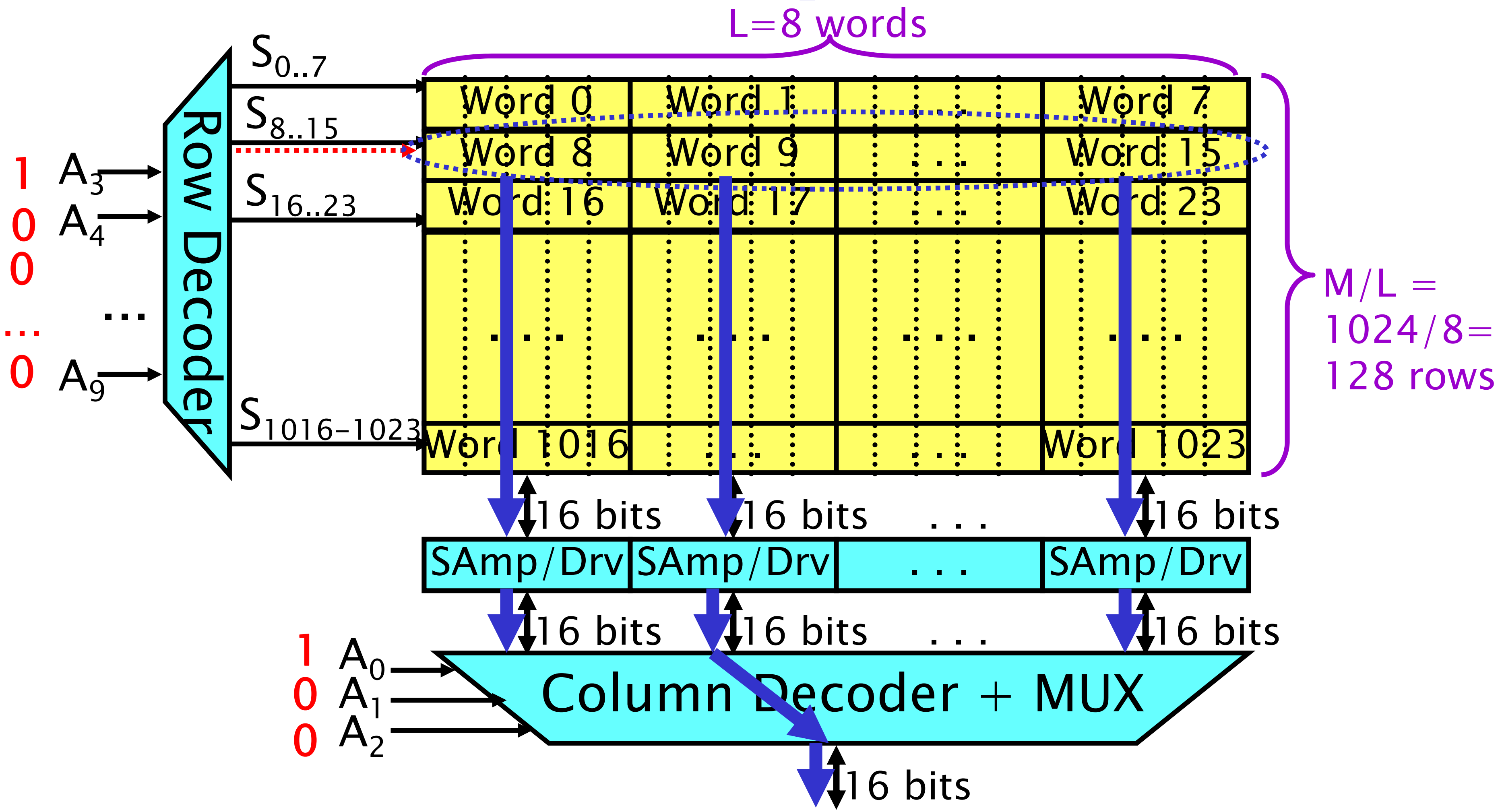
Memory Cell Array Layout (cont.)

- Group the M words into M/L rows, each containing L words
- Benefits?



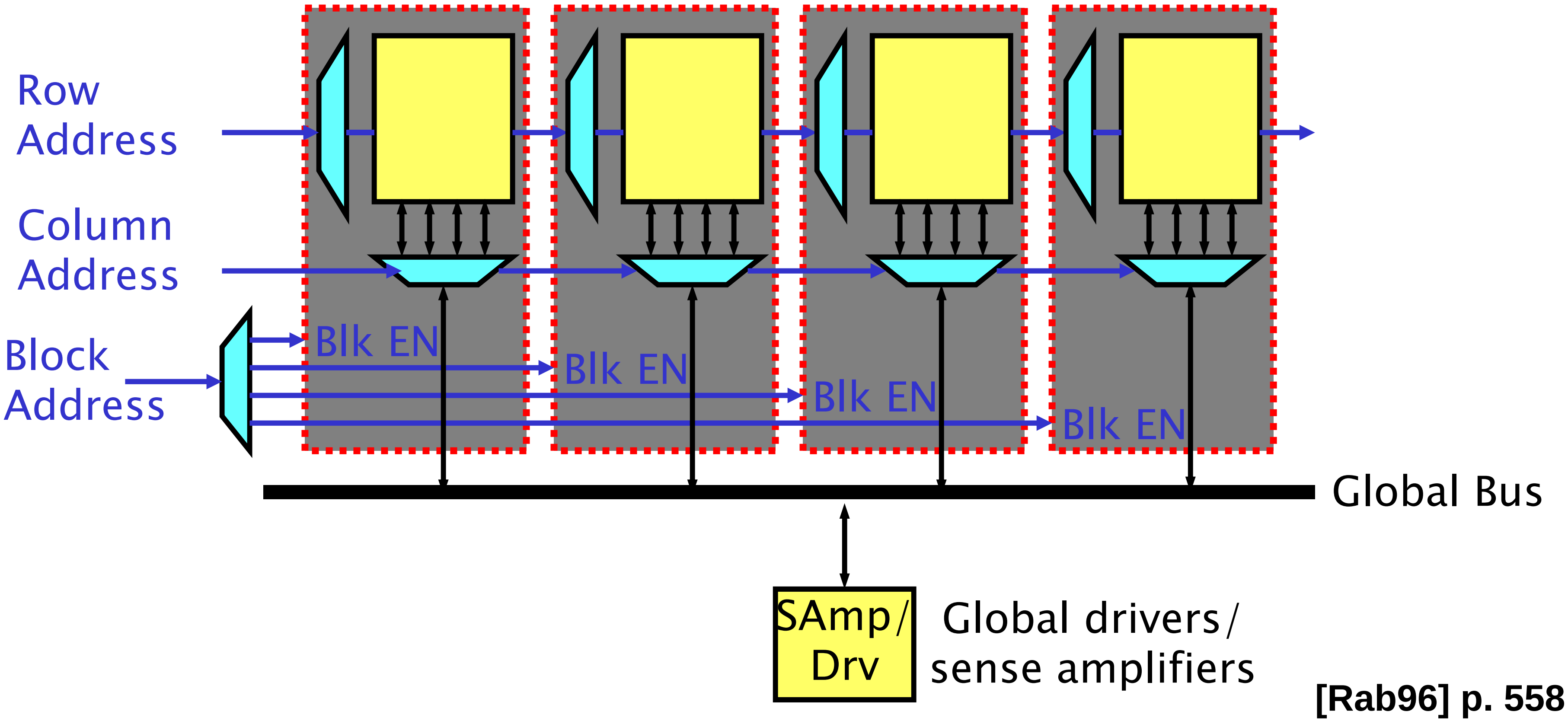
Memory Cell Array Access Example

- word=16-bit wide(N), row=8 words(L), address=10 bits (k)
- Accessing word 9= 0000001001_2



Hierarchical Memory Structure

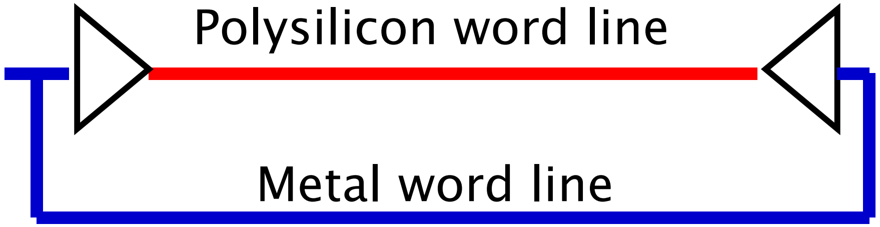
- Taking the idea one step further
  - Shorter wires within each block
  - Enable only one block addr decoder➔ power savings



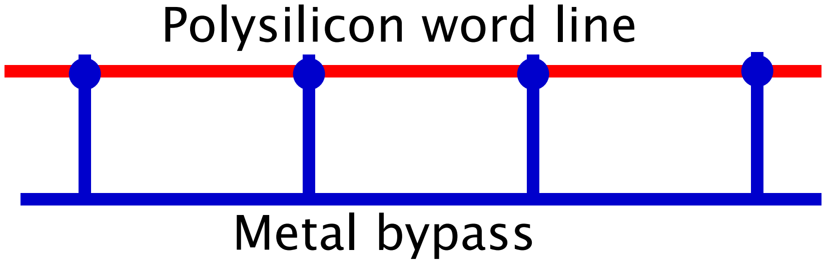
[Rab96] p. 558

Decreasing Word Line Delay

- Word line delay comes into play!
  - We used to have long busses, made 2D array ➔ shorter busses
  - But, longer word lines!
- How to decrease the delay on the word lines?
  - Break the word line by inserting buffers
  - Place the decoder in the middle



(a) Drive the word line from both sides

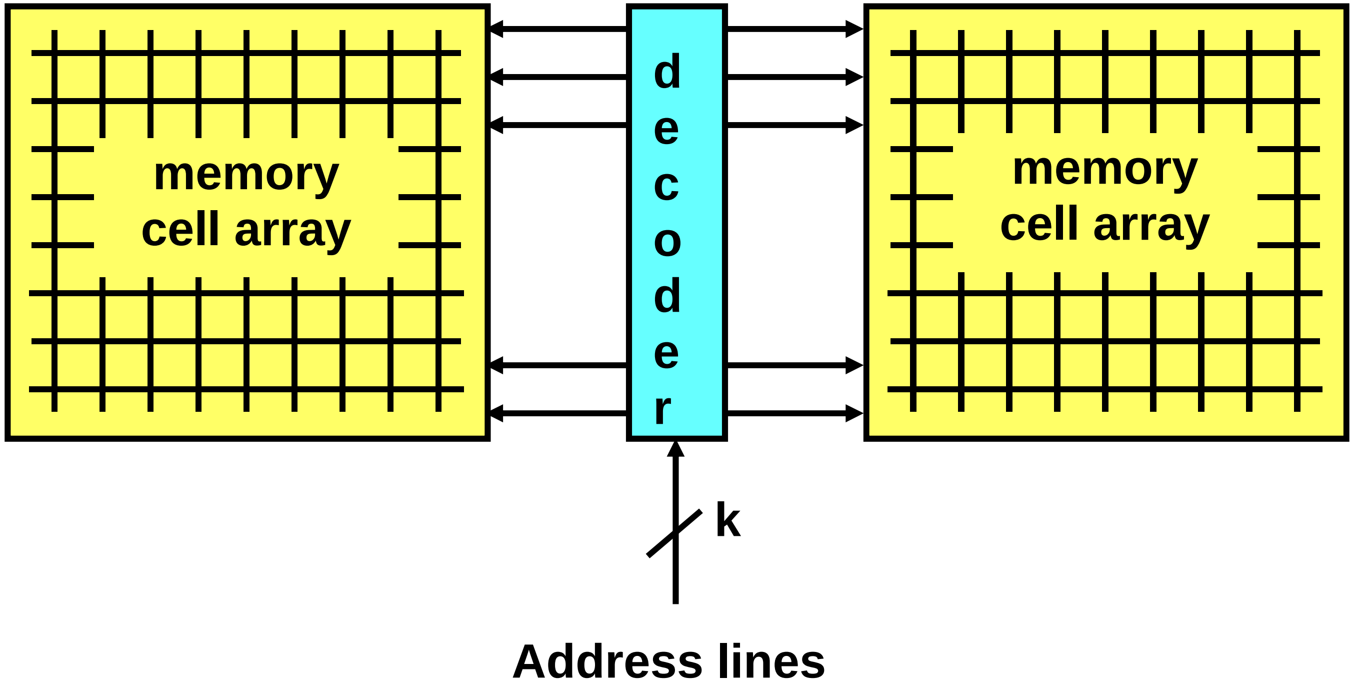


(b) Use metal bypass

[©Prentice Hall]

Decreasing Word Line Delay (cont.)

- Place the decoder in the middle
- Add buffers to outputs of decoder



[©Hauck]

Row Decoder Implementation

- Collection of 2^k high fan-in (k inputs) logic gates
- Regular and dense structure
- N(AND) decoder

$$WL_0 = \overline{A_0}.\overline{A_1}.\overline{A_2}.\overline{A_3}.\overline{A_4}.\overline{A_5}.\overline{A_6}.\overline{A_7}.\overline{A_8}.\overline{A_9}$$

$$WL_{511} = A_0.A_1.A_2.A_3.A_4.A_5.A_6.A_7.A_8.A_9$$

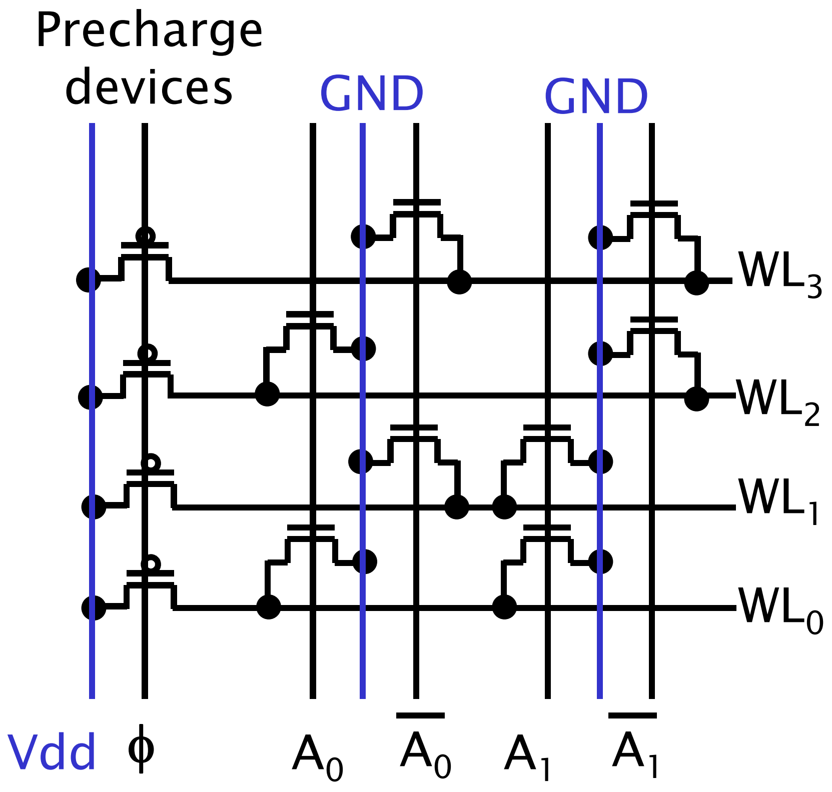
- NOR decoder

$$WL_0 = \overline{A_0+A_1+A_2+A_3+A_4+A_5+A_6+A_7+A_8+A_9}$$

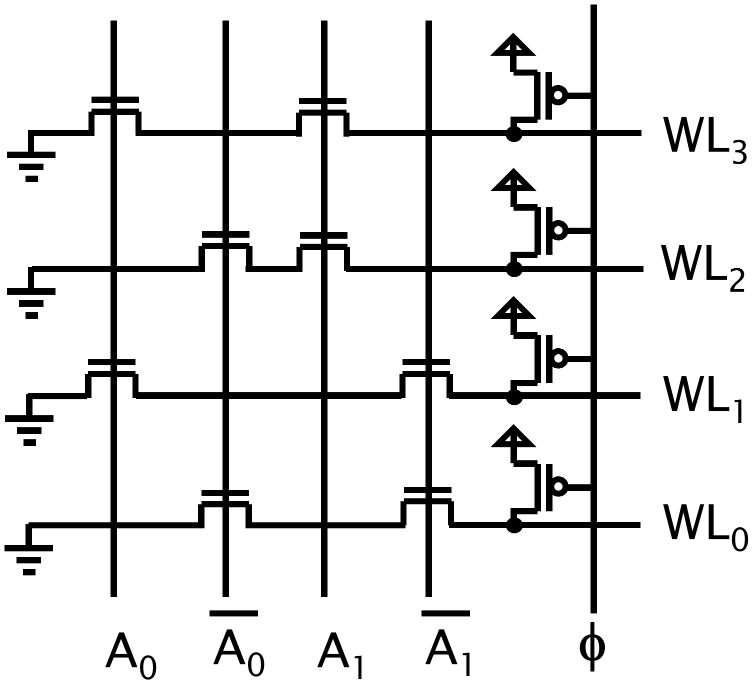
$$WL_{511} = \overline{\overline{A_0}+\overline{A_1}+\overline{A_2}+\overline{A_3}+\overline{A_4}+\overline{A_5}+\overline{A_6}+\overline{A_7}+\overline{A_8}+\overline{A_9}}$$

[©Prentice Hall]

Row Decoder Implementation (cont.)

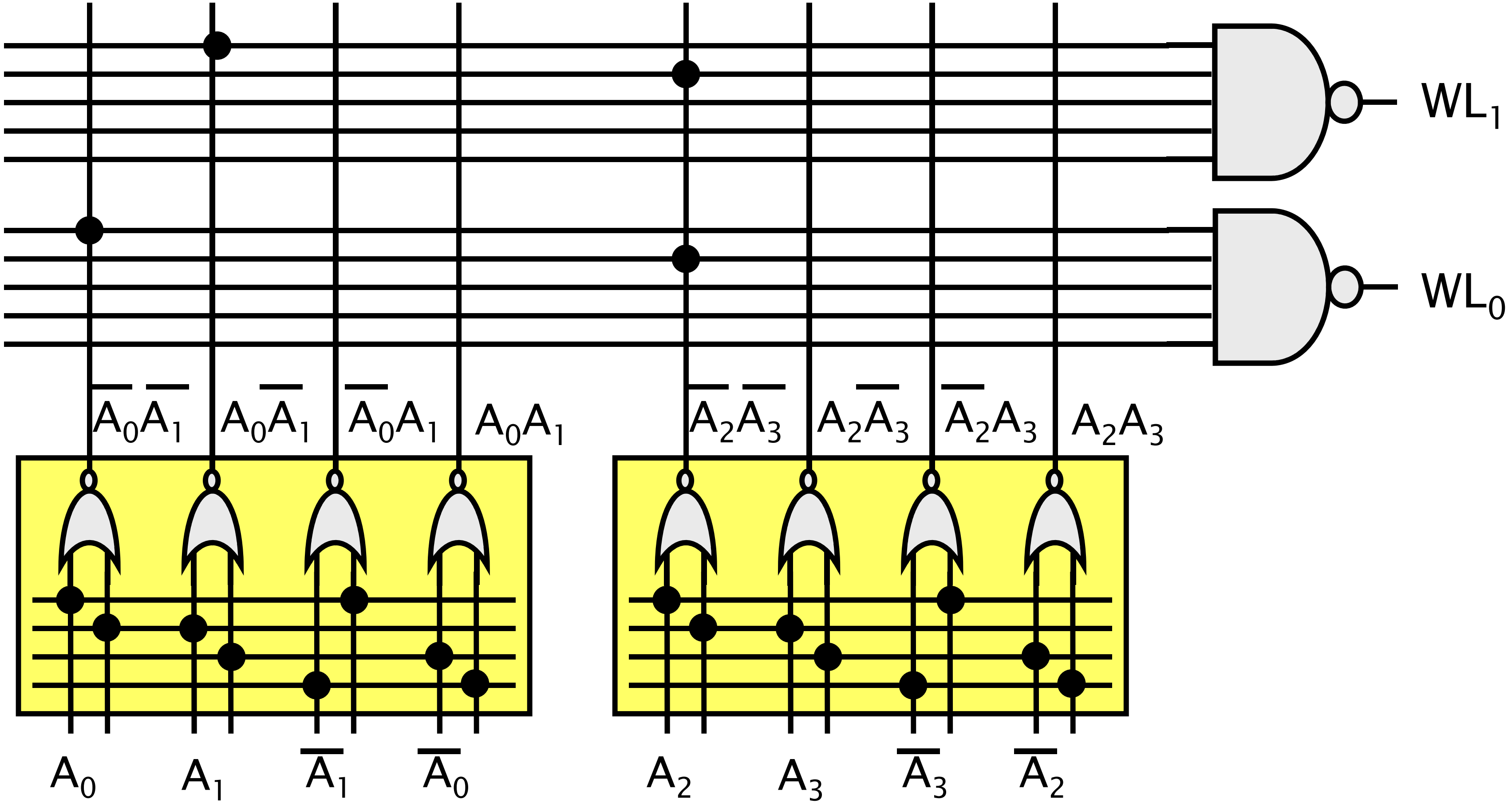


Dynamic 2-to-4 NOR Decoder



2-to-4 MOS Dynamic NAND Decoder

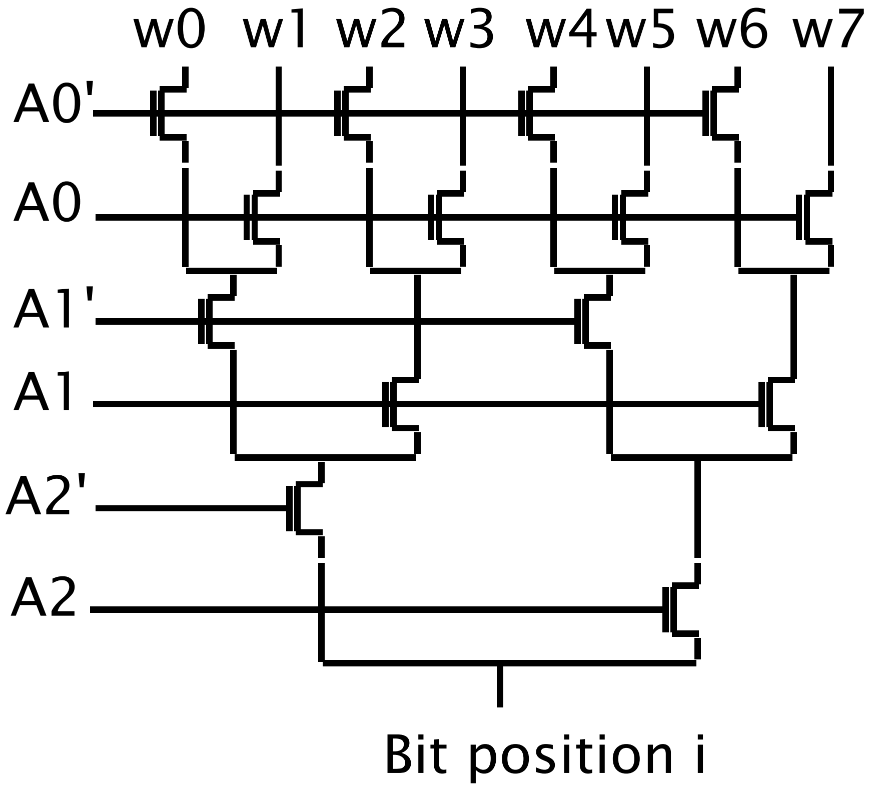
Row Decoder Implementation (cont.)



Splitting decoder into two or more logic layers produces a faster and cheaper implementation

Column Multiplexers: Tree-Based Decoder

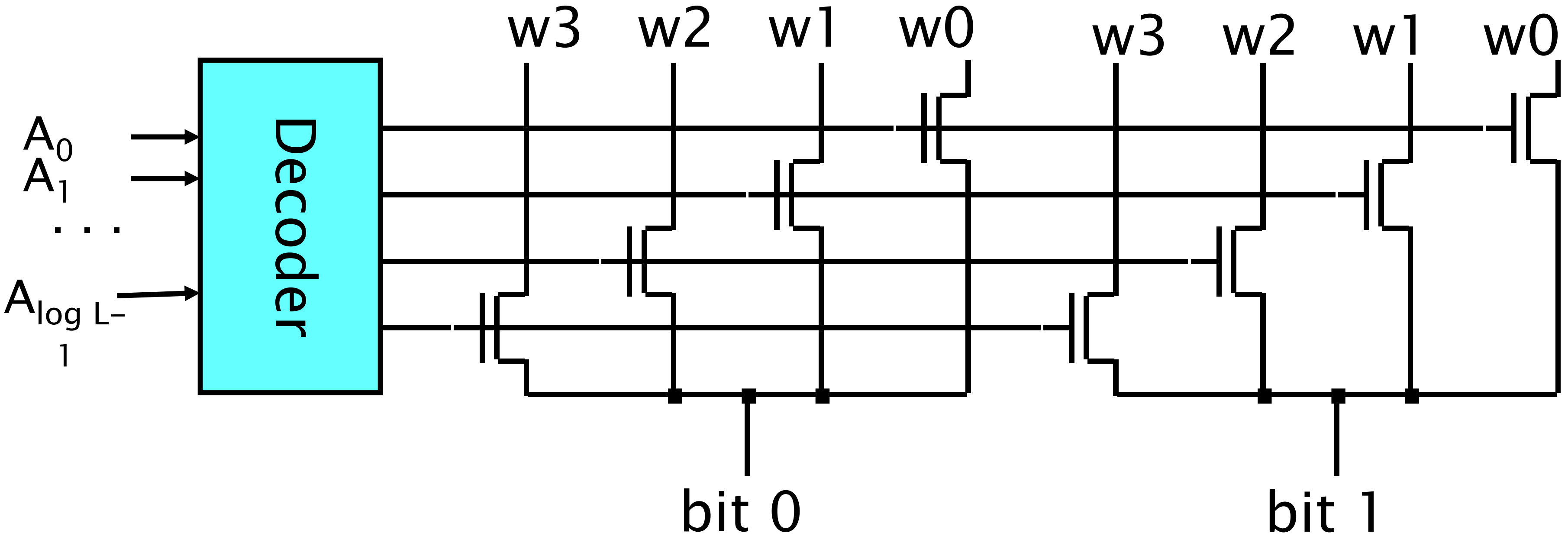
- Route many inputs to a single output
  - Inputs come from different words, same bit position
- Series transistors are slow
  - On the critical path too
- Area?
  - One-bit very small, but have to repeat the "decoding" for all bit positions.



[©Hauck]

Column Multiplexers: Faster Implementation

- Decode address into one-hot signals
- Each bit passes through single n-device or pass gate
- Column decoding done in parallel w/ row decoding



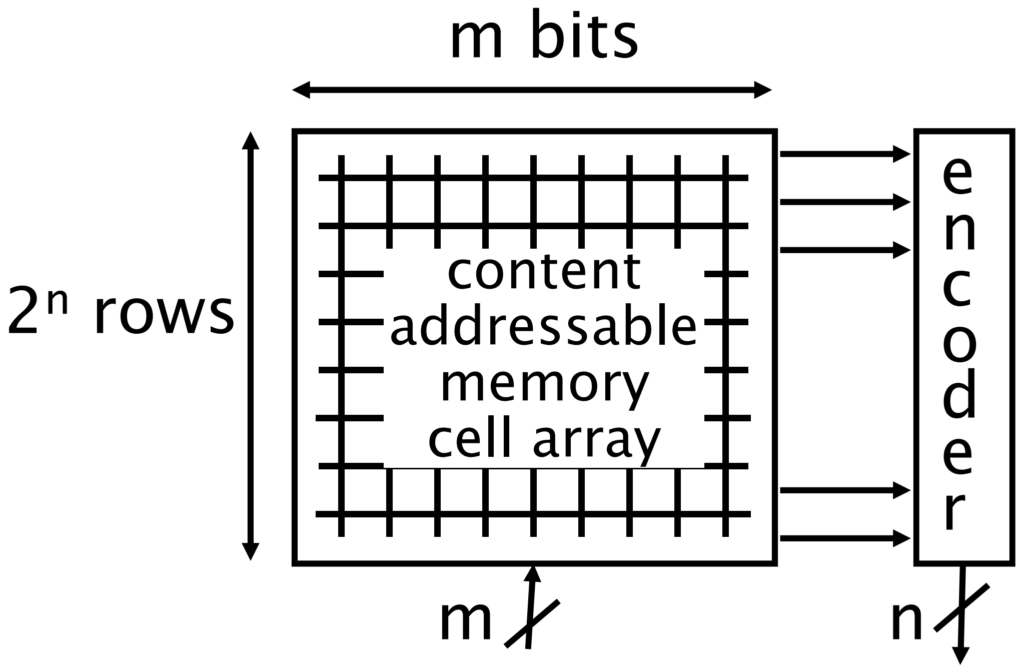
[©Hauck]

Outline

- Memory interface
- Memory cells
  - Static memory cell
  - Dynamic memory cell
  - ROM cells
- Address decoders
- **Content addressable memory (CAM)**
- Non volatile memory cells

Content Addressable Memory (CAM)

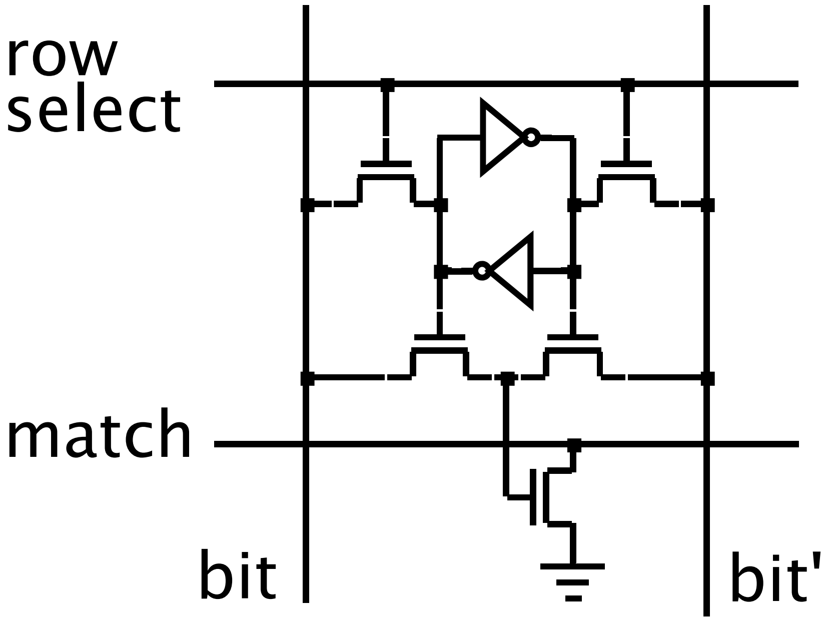
- Instead of address, provide data → find a match
  - Applications: cache, physical particle collider
- Needs “Encoder”:
  - Inverse function of decoder
  - Take a one-hot collection of signals and encode them



[©Hauck]

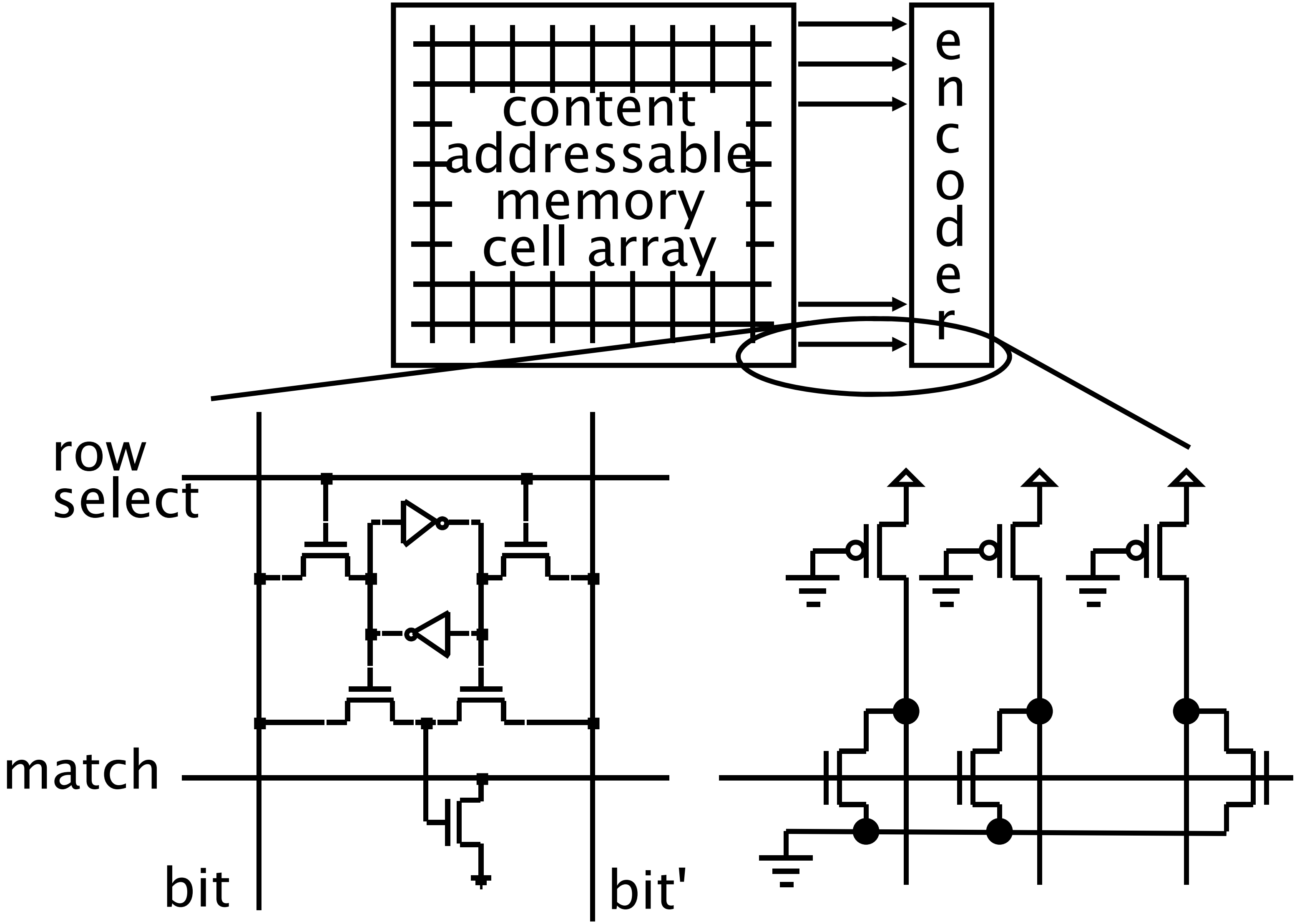
Content Addressable Memory Cell

- Read and write like normal 6T memory cell
- Match signal is precharged to 1, pulled to 0 if no match
  - Send data on bit’ and data’ on bit for matching
  - Match remains 1 iff all bits in word match



[©Hauck]

Encoders


content
addressable
memory
cell array
encoder
row
select
match
bit
bit'

Content Addressable Memory (CAM)

- Writing is done as normal SRAM
  - Address decoder needed
  - Drive row select
- *½ n log n* transistors on the address lines (in encoder)

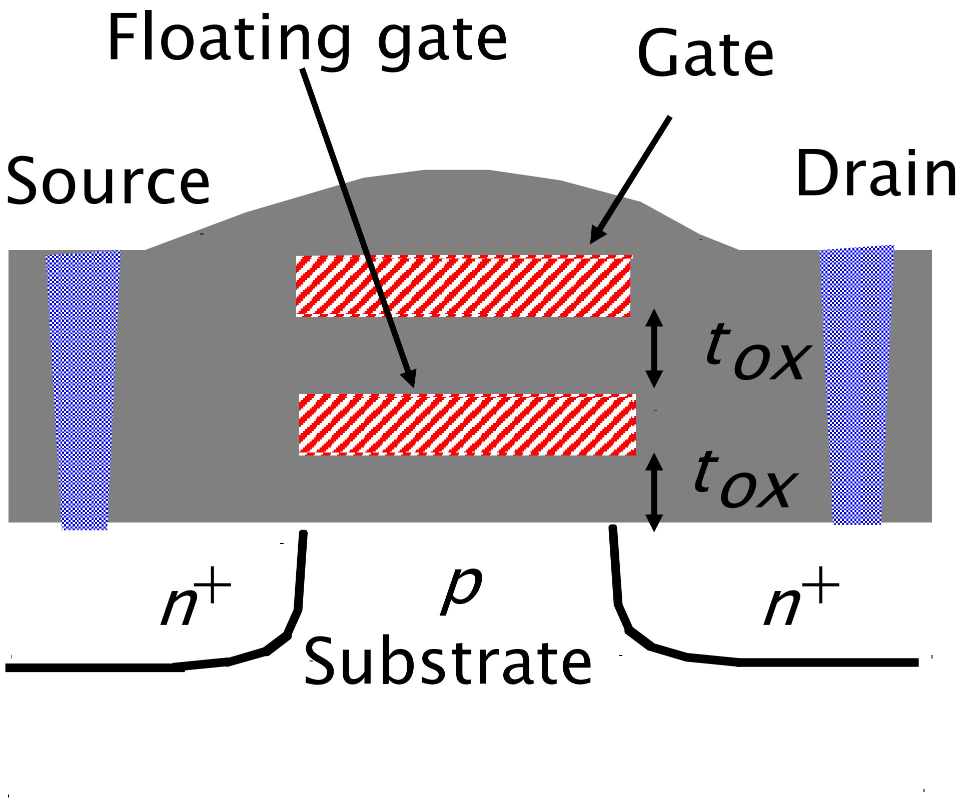
Outline

- Registers (flip-flops), shift registers
- Memory interface
- Memory cells
  - Static memory cell
  - Dynamic memory cell
  - ROM cells
- Address decoders
- Content addressable memory (CAM)
- **Non volatile memory cells**

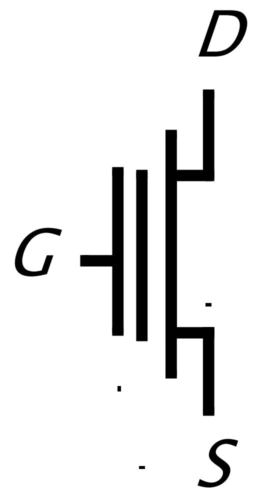
Non-Volatile Memory Cells

- Programmable after fabrication
- Keep their configuration even after the supply voltage is disconnected
- Basic idea:
  - Use a floating strip of polysilicon between the substrate and the gate
  - Put charges on the floating gate
  - Increase threshold voltage ➔ disable the device
- Different types based on the erasure method

Floating-Gate Transistor (FAMOS)



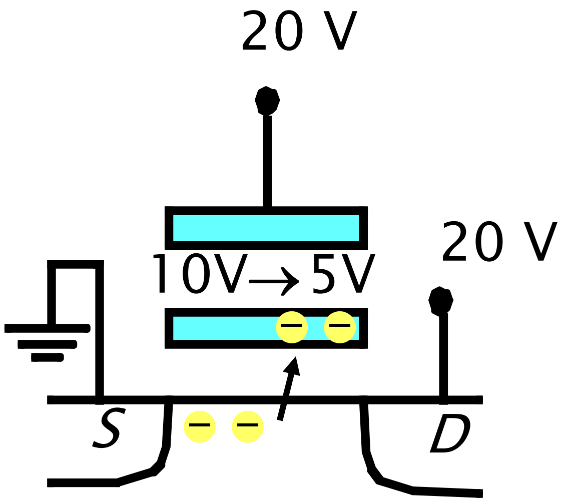
(a) Device cross-section



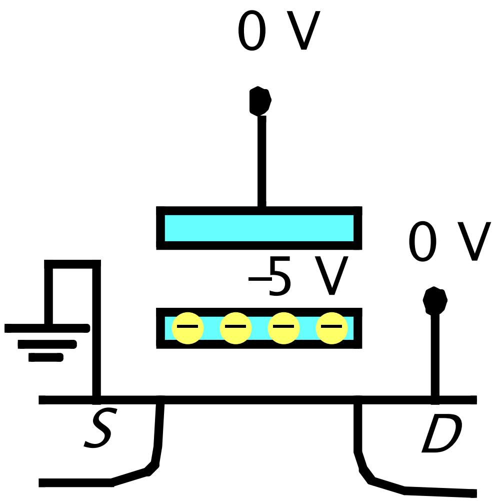
(b) Schematic symbol

[©Prentice Hall]

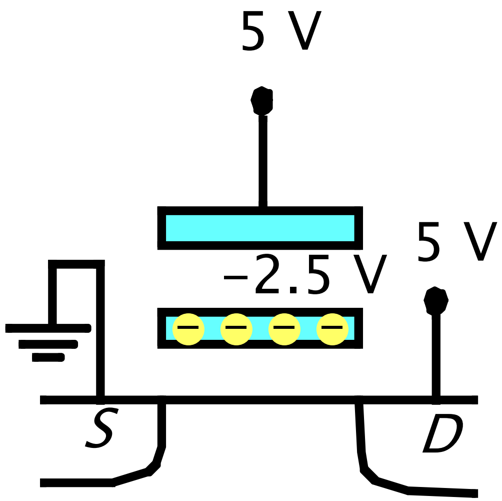
Floating-Gate Transistor: Programming



Avalanche injection.



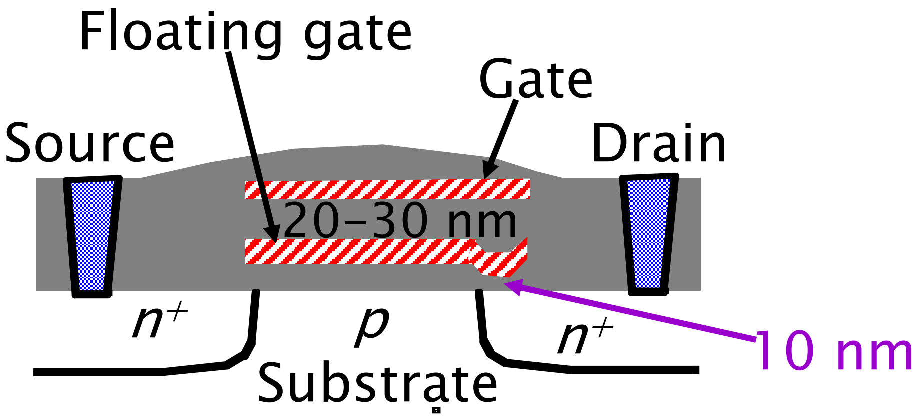
Removing programming voltage leaves charges trapped



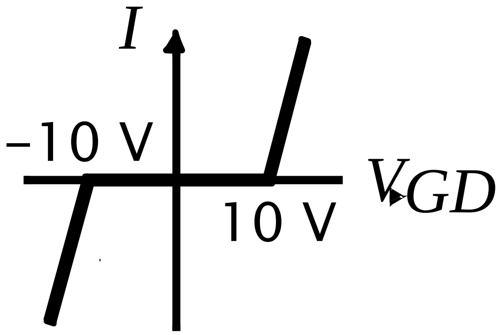
Programming results in higher V_T

[©Prentice Hall]

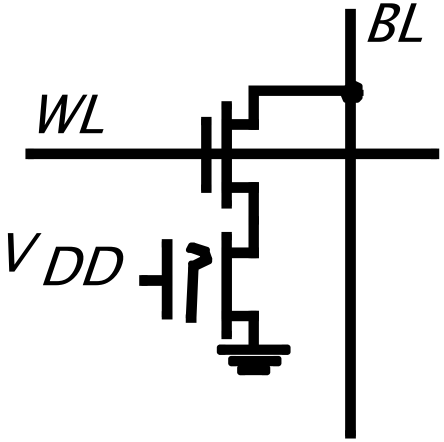
FLOTOX EEPROM



(a) Flotox transistor



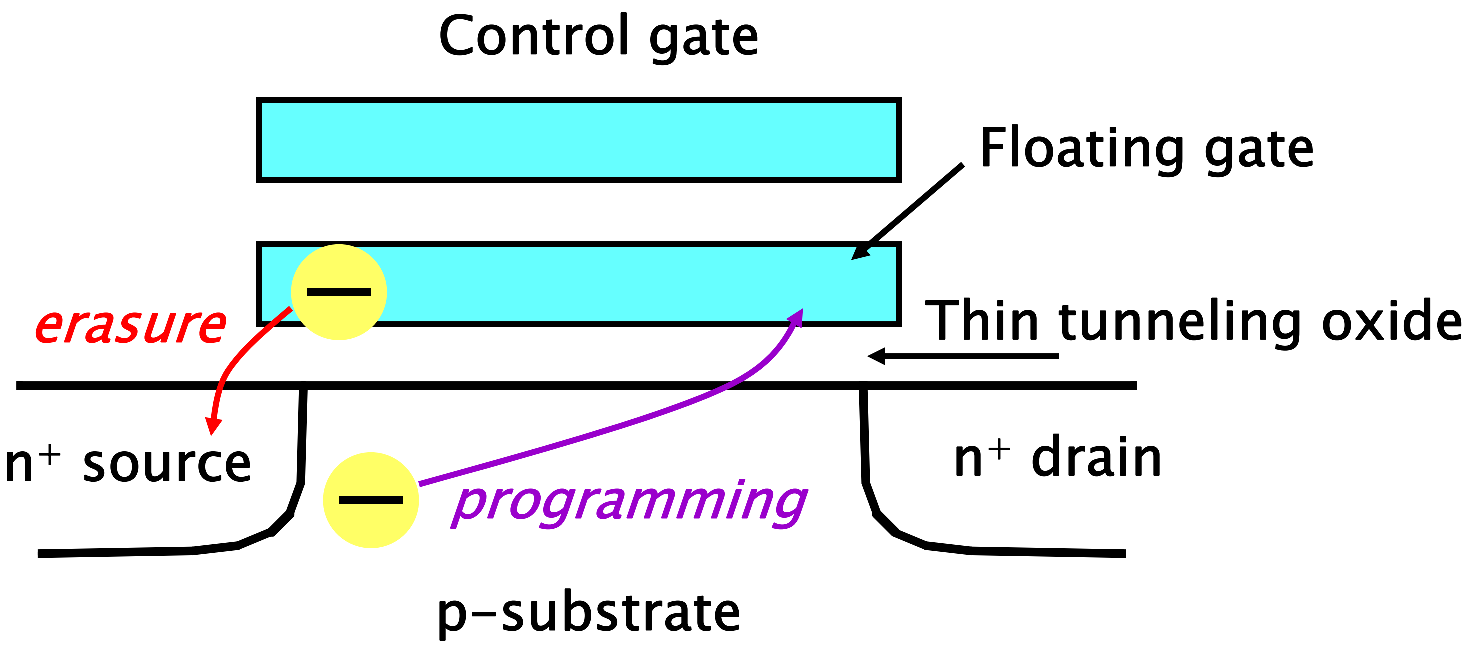
(b) Fowler–Nordheim
I–V characteristics



(c) EEPROM cell during a read operation

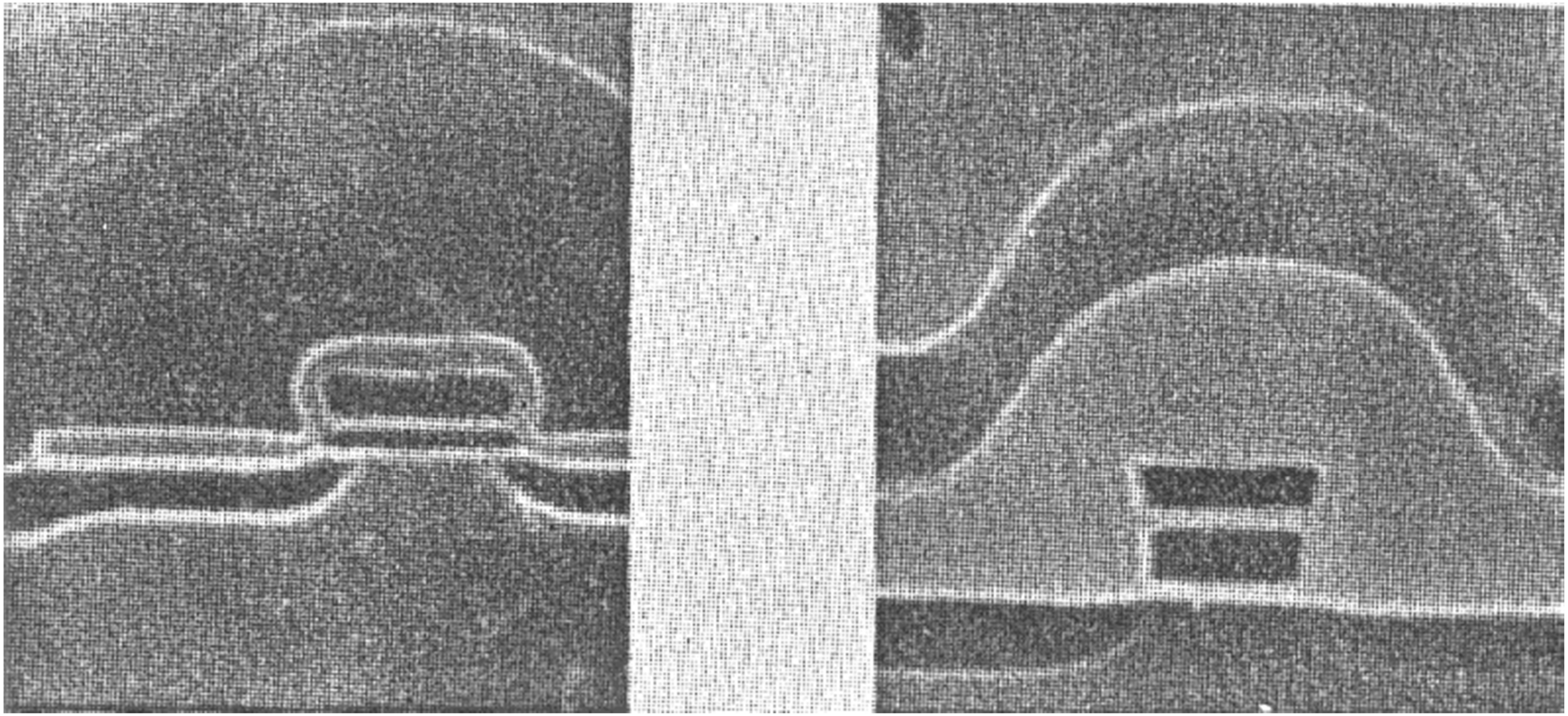
[©Prentice Hall]

FLASH EEPROM



[©Prentice Hall]

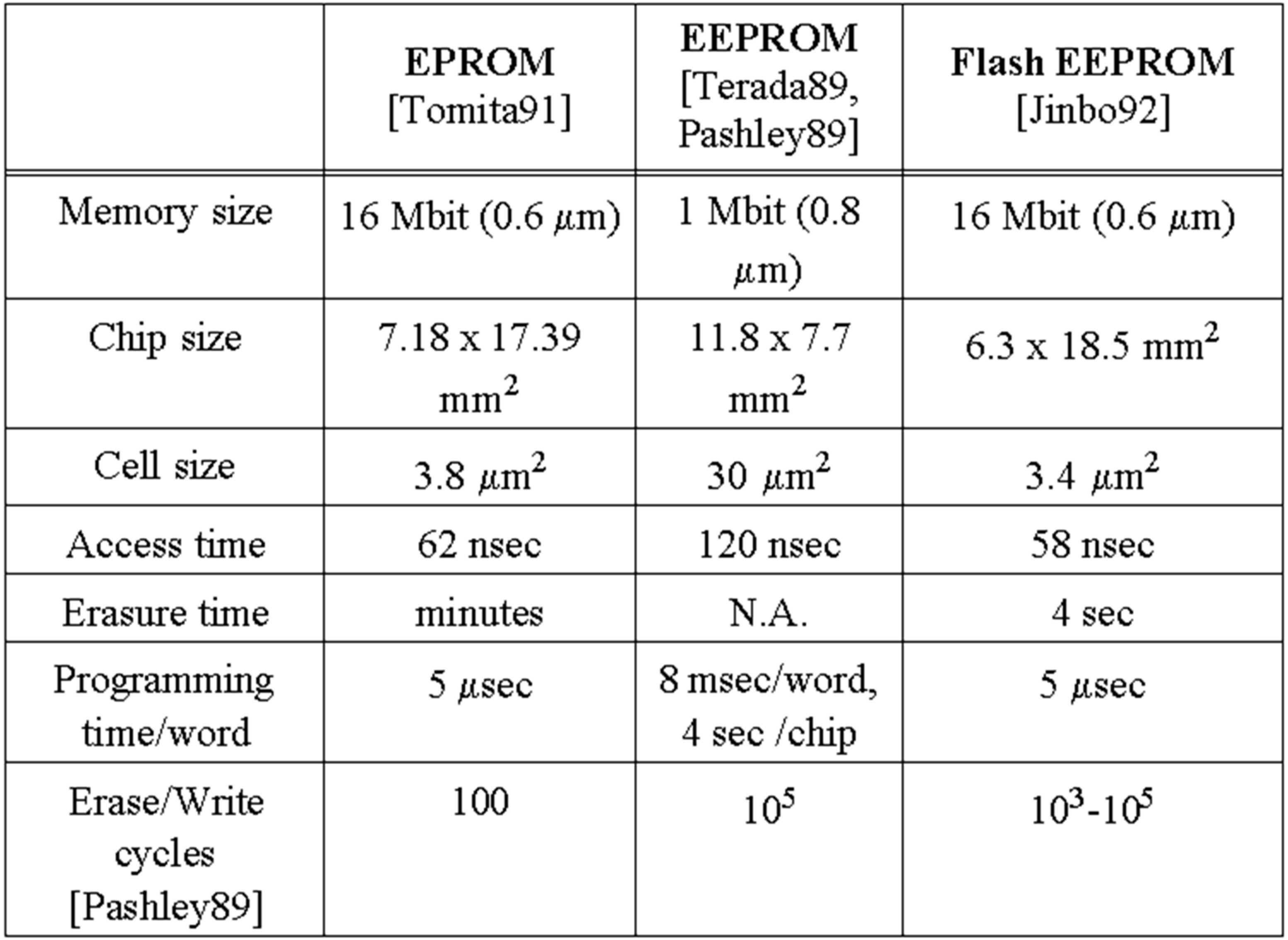
Cross-Section of NVM Cells



Courtesy Intel

[©Prentice Hall]

Characteristics of Some NVM Cells



	EPROM [Tomita91]	**EEPROM** [Terada89, Pashley89]	**Flash EEPROM** [Jinbo92]
Memory size	16 Mbit (0.6 μm)	1 Mbit (0.8 μm)	16 Mbit (0.6 μm)
Chip size	7.18 x 17.39 mm^2	11.8 x 7.7 mm^2	6.3 x 18.5 mm^2
Cell size	3.8 μm^2	30 μm^2	3.4 μm^2
Access time	62 nsec	120 nsec	58 nsec
Erasure time	minutes	N.A.	4 sec
Programming time/word	5 μsec	8 msec/word, 4 sec /chip	5 μsec
Erase/Write cycles [Pashley89]	100	10^5	10^3-10^5