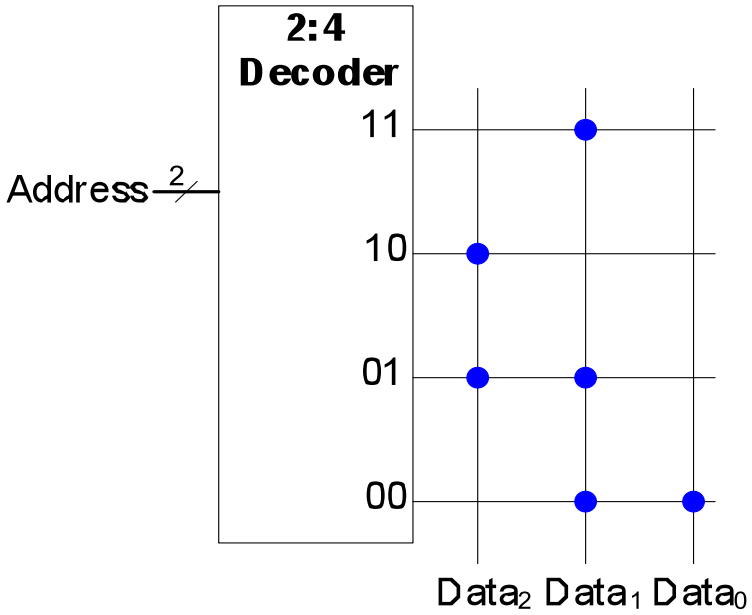
Chapter 5 :: Memory and Logic Arrays

Digital Design and Computer Architecture

David Money Harris and Sarah L. Harris

ROM Storage

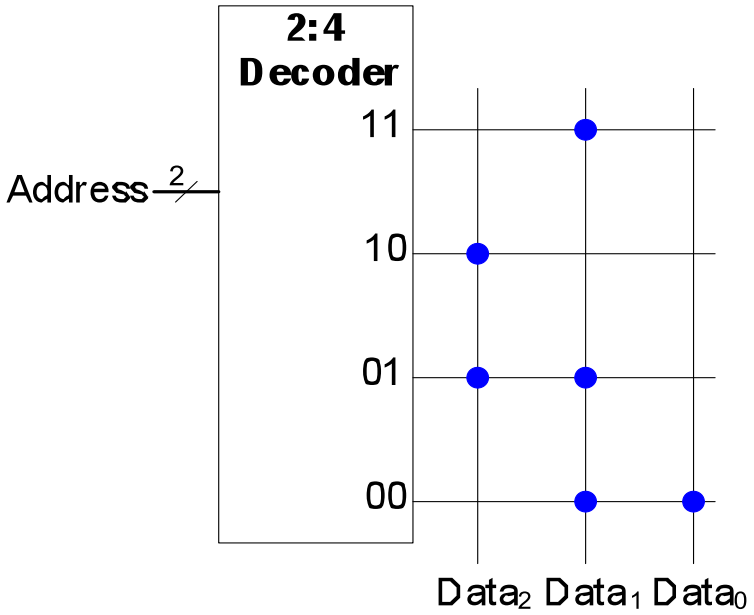


Address	Data		
11	0	1	0
10	1	0	0
01	1	1	0
00	0	1	1

depth

width

ROM Logic



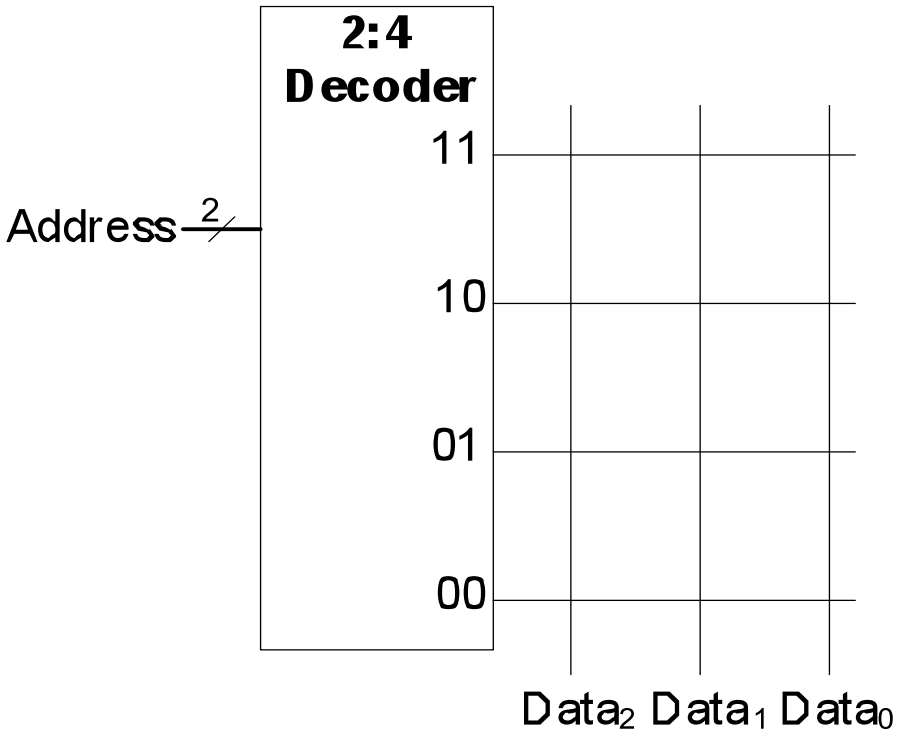
$$Data_2 = A_1 \oplus A_0$$

$$Data_1 = \overline{A_1} + A_0$$

$$Data_0 = \overline{A_1}\,\overline{A_0}$$

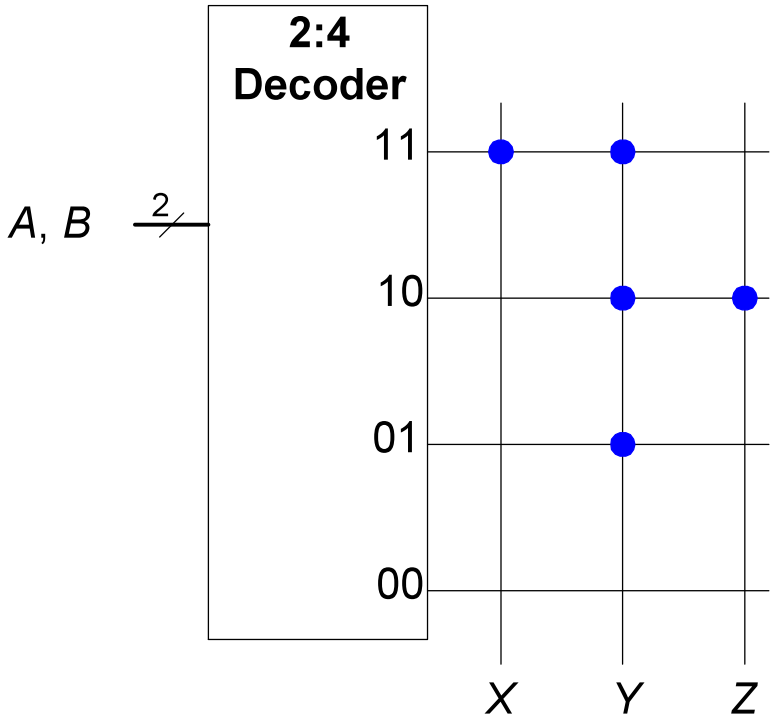
Example: Logic with ROMs

- Implement the following logic functions using a $2^2 \times 3$-bit ROM:
  - $X = AB$
  - $Y = A + B$
  - $Z = A\overline{B}$

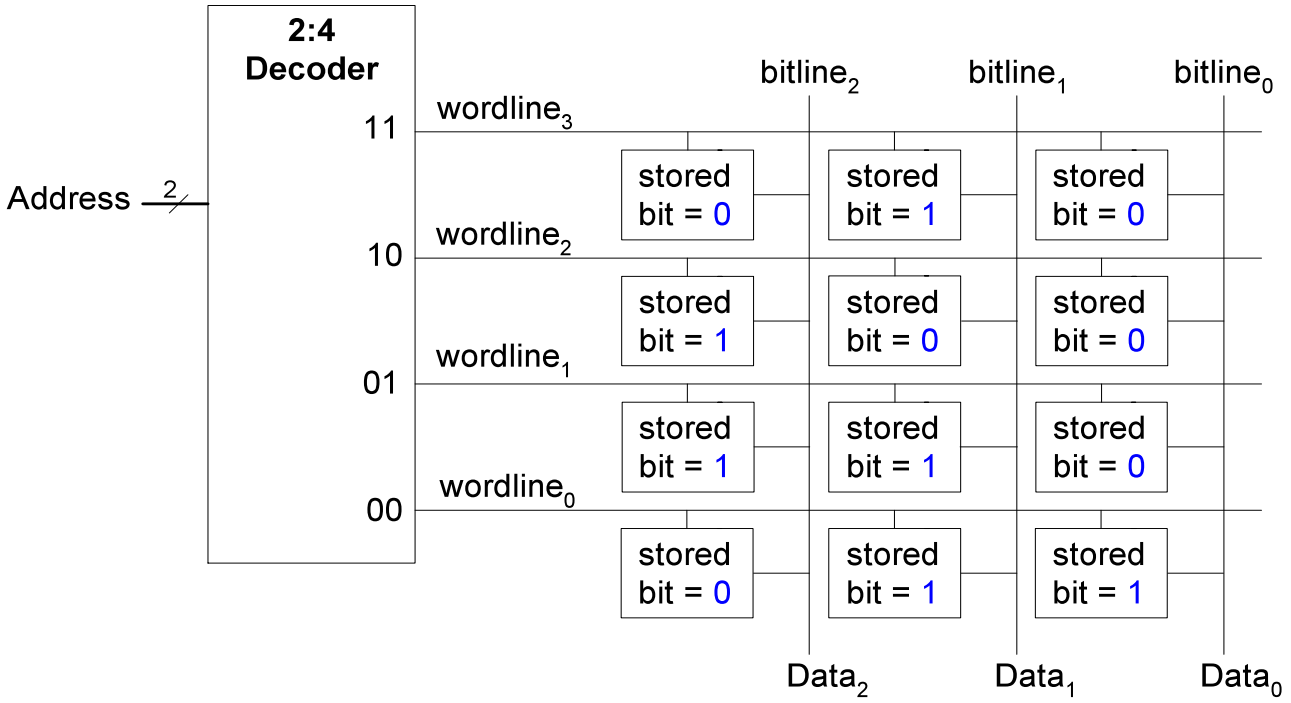


Example: Logic with ROMs

- Implement the following logic functions using a $2^2 \times 3$-bit ROM:
  - $X = AB$
  - $Y = A + B$
  - $Z = A\,\overline{B}$



Logic with Any Memory Array



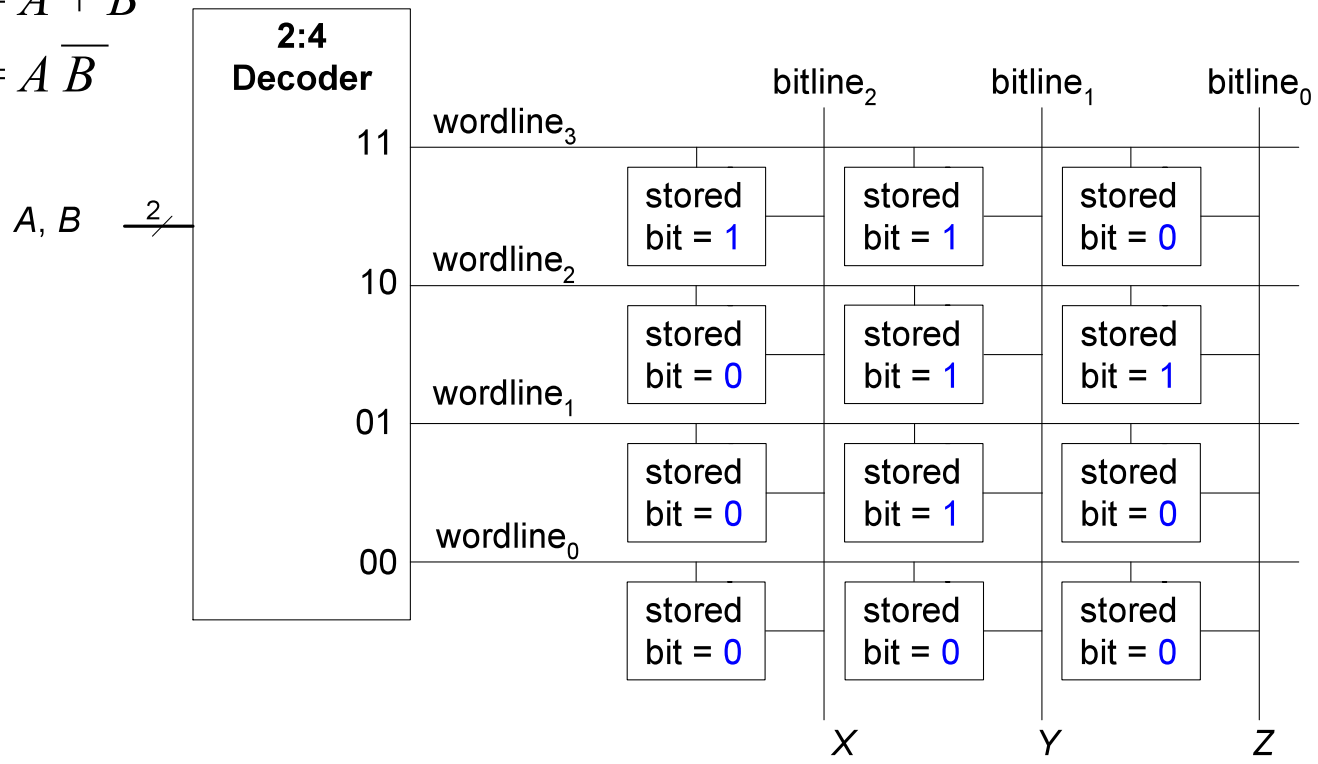
$$Data_2 = A_1 \oplus A_0$$

$$Data_1 = \overline{A_1} + A_0$$

$$Data_0 = \overline{A_1}\,\overline{A_0}$$

Logic with Memory Arrays

- Implement the following logic functions using a $2^2 \times 3$-bit memory array:
  - $X = AB$
  - $Y = A + B$
  - $Z = A\,\overline{B}$

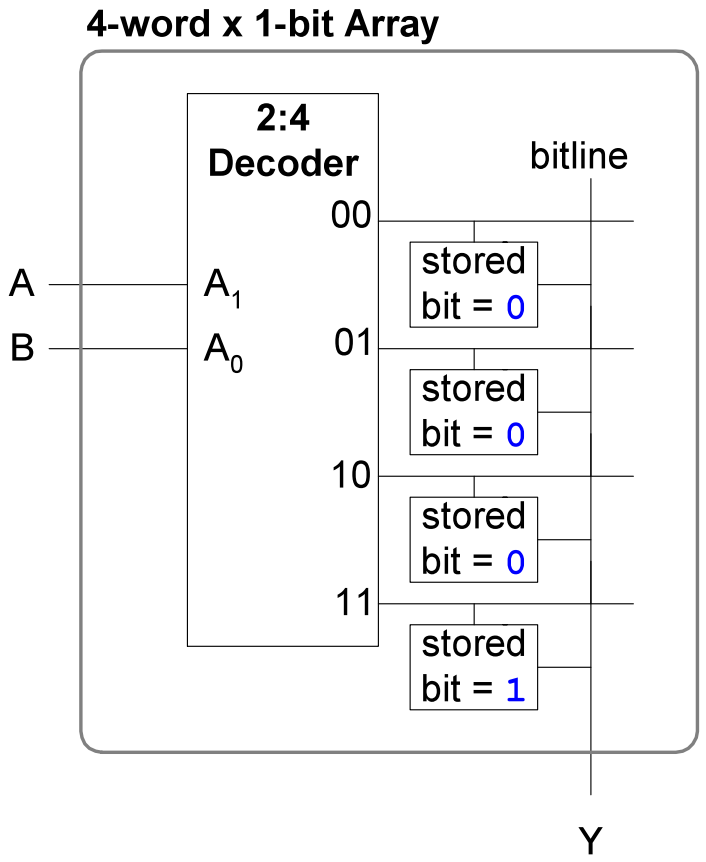


Logic with Memory Arrays

- Called *lookup tables* (LUTs): look up output at each input combination (address)

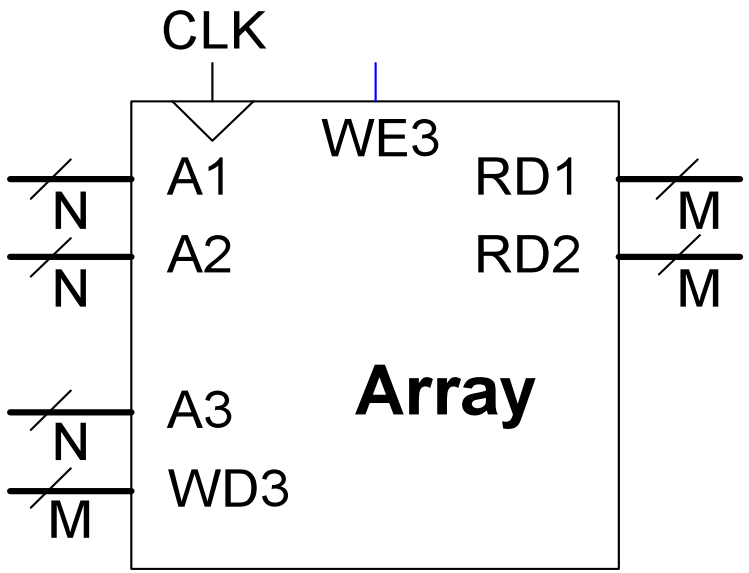
Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



Multi-ported Memories

- **Port:** address/data pair
- 3-ported memory
  - 2 read ports (A1/RD1, A2/RD2)
  - 1 write port (A3/WD3, WE3 enables writing)
- Small multi-ported memories are called *register files*



Verilog Memory Arrays

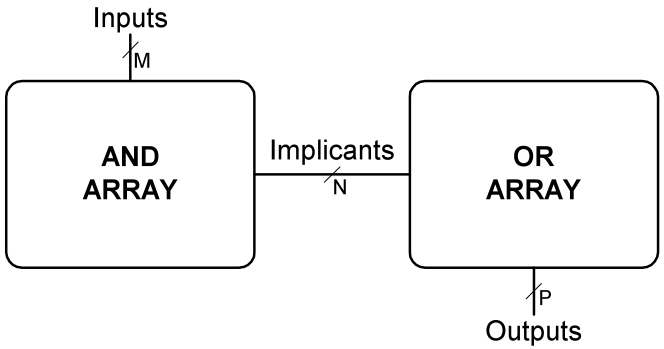
```
// 256 x 3 memory module with one read/write port
module dmem( input            clk, we,
             input  [7:0]     a
             input  [2:0]     wd,
             output [2:0]     rd);

  reg  [2:0] RAM[255:0];

  assign rd = RAM[a];

  always @(posedge clk)
      if (we)
          RAM[a] <= wd;
endmodule
```

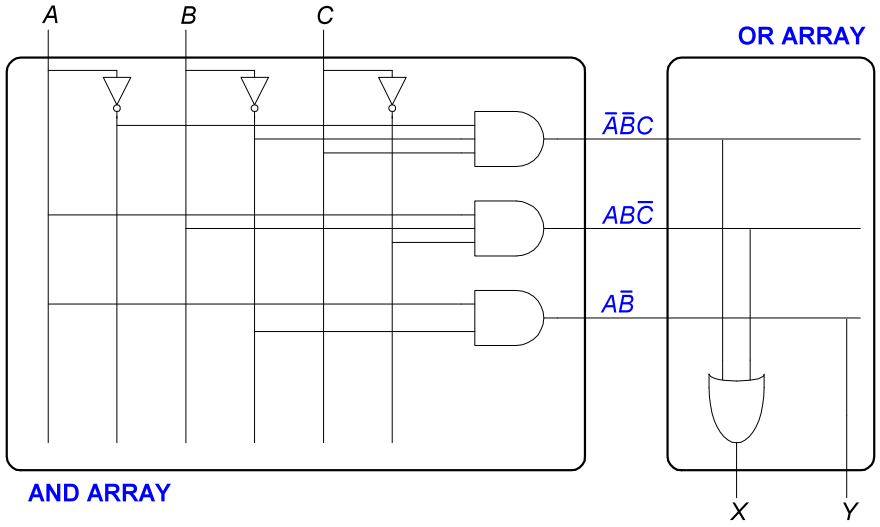
Logic Arrays

- Programmable logic arrays (PLAs)
  - AND array followed by OR array
  - Perform combinational logic only
  - Fixed internal connections
- Field programmable gate arrays (FPGAs)
  - Array of configurable logic blocks (CLBs)
  - Perform combinational and sequential logic
  - Programmable internal connections

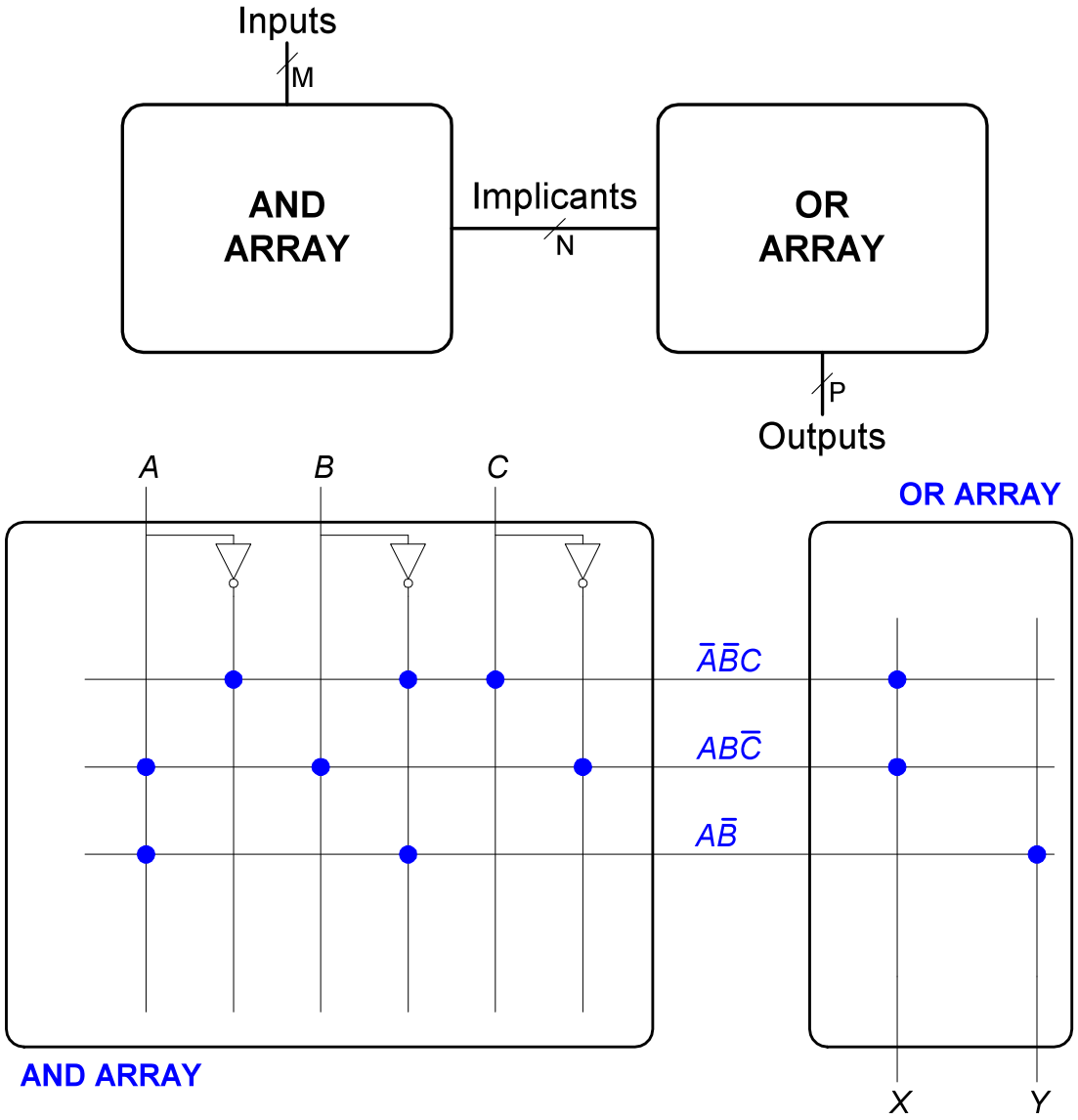
PLAs

- $X = \bar{A}\bar{B}C + AB\bar{C}$
- $Y = A\bar{B}$



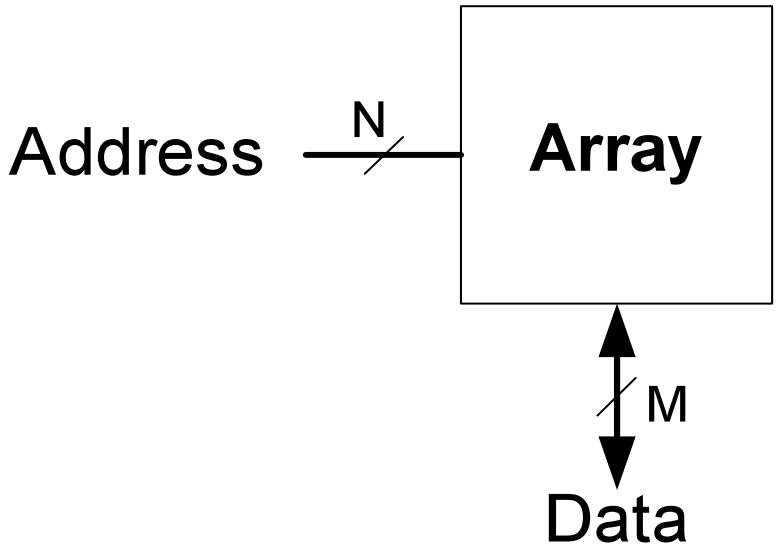


PLAs: Dot Notation



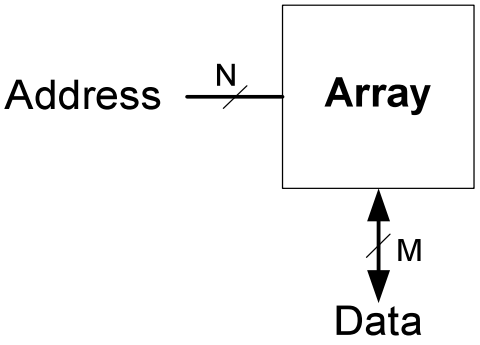
Memory Arrays

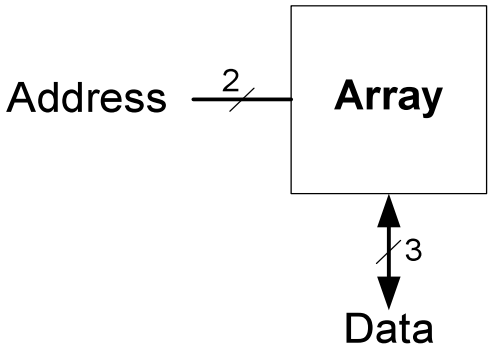
- Efficiently store large amounts of data
- Three common types:
  - Dynamic random access memory (DRAM)
  - Static random access memory (SRAM)
  - Read only memory (ROM)
- An *M*-bit data value can be read or written at each unique *N*-bit address.

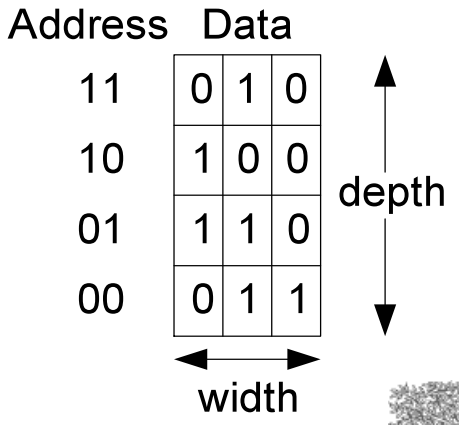


Memory Arrays

- Two-dimensional array of bit cells
- Each bit cell stores one bit
- An array with N address bits and M data bits:
  - 2^N rows and M columns
  - **Depth:** number of rows (number of words)
  - **Width:** number of columns (size of word)
  - **Array size:** depth × width = $2^N \times M$



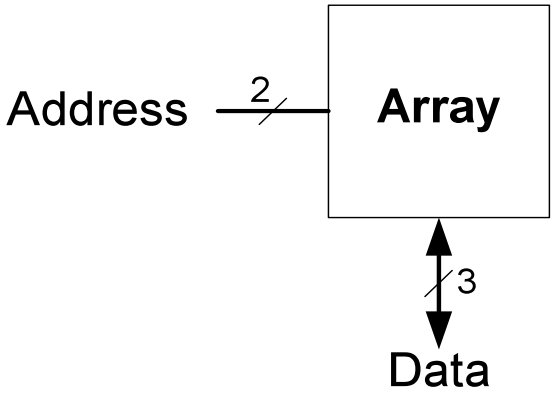


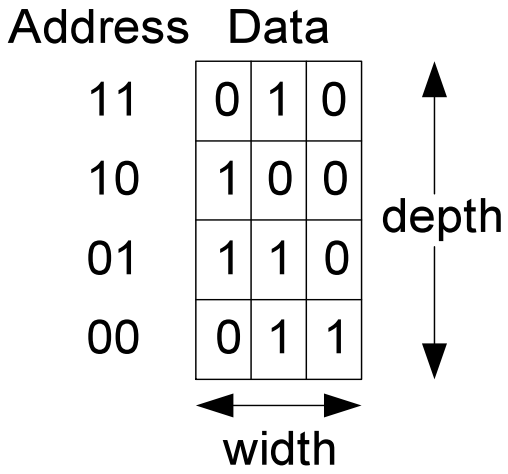


Memory Array: Example

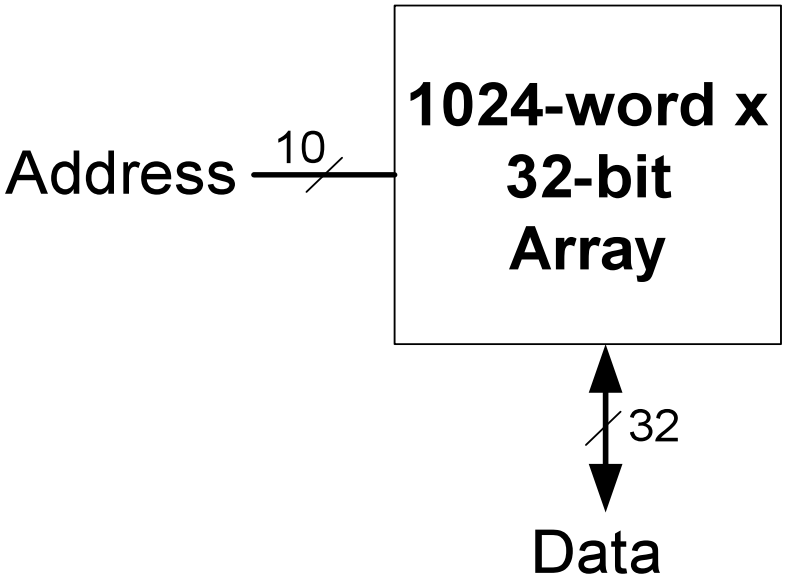
- $2^2 \times 3$-bit array
- Number of words: 4
- Word size: 3-bits
- For example, the 3-bit word stored at address 10 is 100

Example:

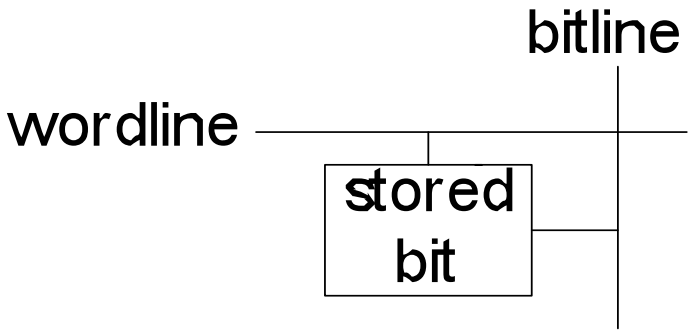




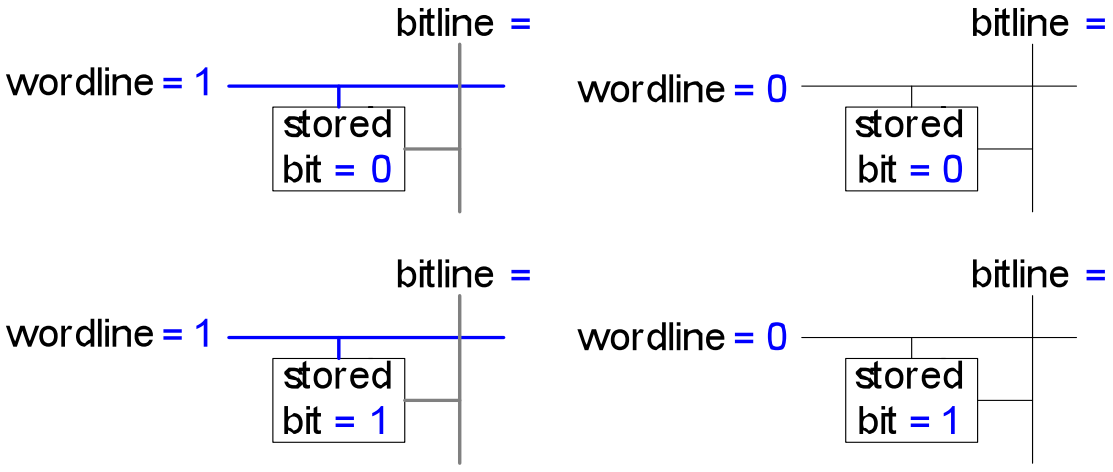
Memory Arrays



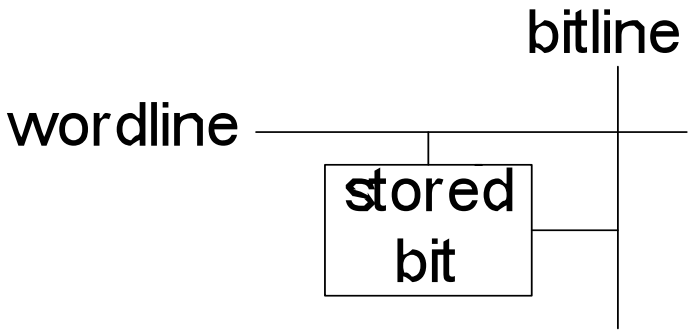
Memory Array Bit Cells



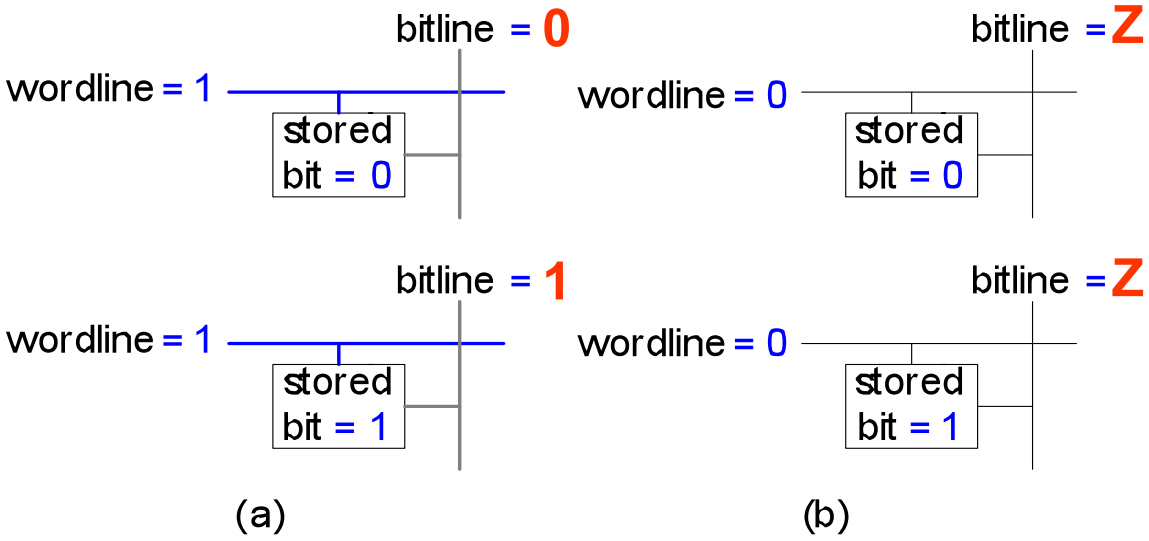
Example:



Memory Array Bit Cells

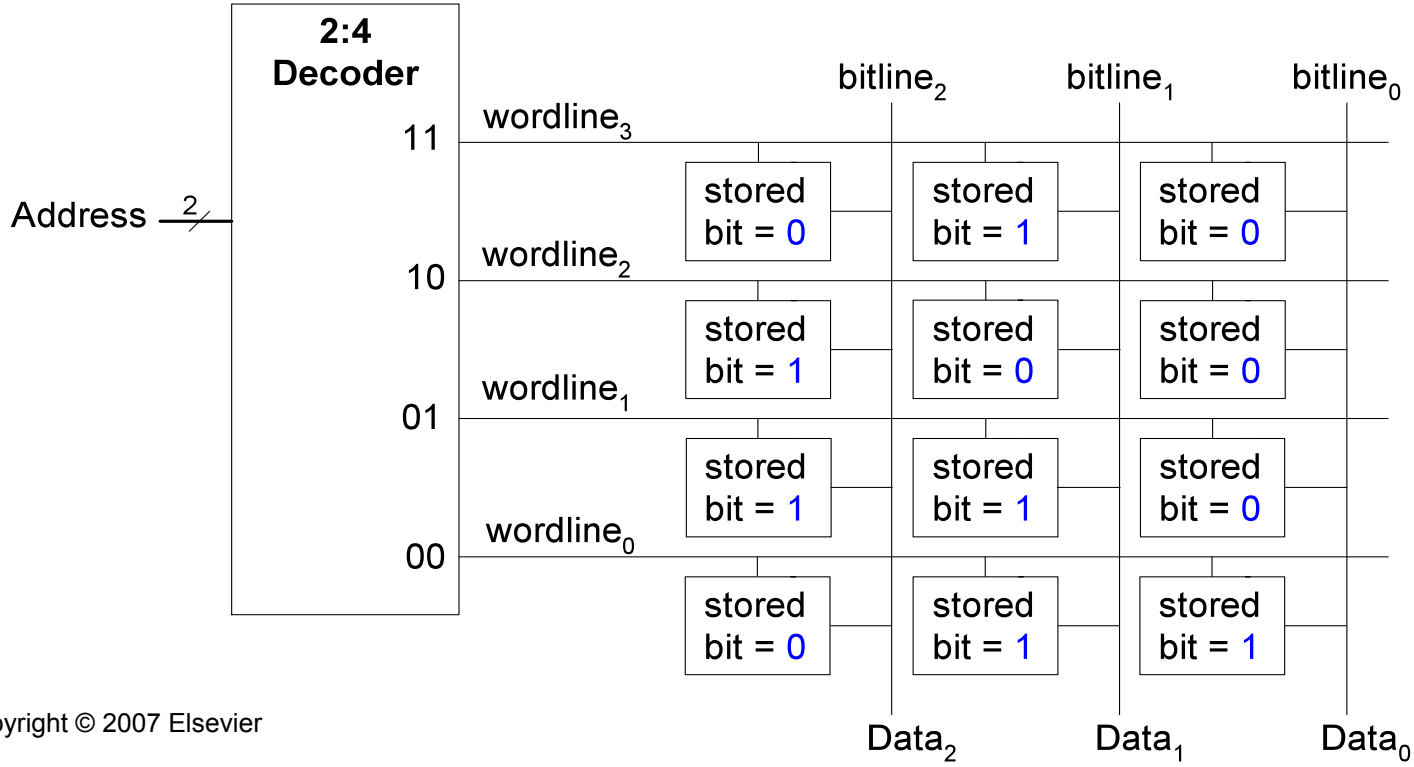


Example:



Memory Array

- **Wordline:**
  - similar to an enable
  - allows a single row in the memory array to be read or written
  - corresponds to a unique address
  - only one wordline is HIGH at any given time



Types of Memory

- Random access memory (RAM): volatile
- Read only memory (ROM): nonvolatile

RAM: Random Access Memory

- **Volatile:** loses its data when the power is turned off
- Read and written quickly
- Main memory in your computer is RAM (DRAM)

Historically called *random* access memory because any data word can be accessed as easily as any other (in contrast to sequential access memories such as a tape recorder)

ROM: Read Only Memory

- **Nonvolatile:** retains data when power is turned off
- Read quickly, but writing is impossible or slow
- Flash memory in cameras, thumb drives, and digital cameras are all ROMs

Historically called *read only* memory because ROMs were written at manufacturing time or by burning fuses. Once ROM was configured, it could not be written again. This is no longer the case for Flash memory and other types of ROMs.

Fujio Masuoka, 1944-

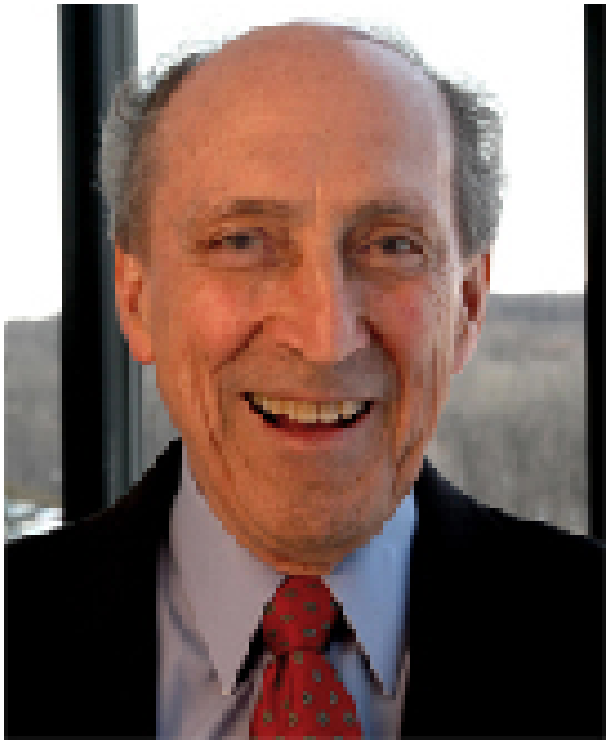
- Developed memories and high speed circuits at Toshiba from 1971-1994.
- Invented Flash memory as an unauthorized project pursued during nights and weekends in the late 1970's.
- The process of erasing the memory reminded him of the flash of a camera
- Toshiba slow to commercialize the idea; Intel was first to market in 1988
- Flash has grown into a $25 billion per year market.

Types of RAM

- Two main types of RAM:
  - Dynamic random access memory (DRAM)
  - Static random access memory (SRAM)
- Differ in how they store data:
  - DRAM uses a capacitor
  - SRAM uses cross-coupled inverters

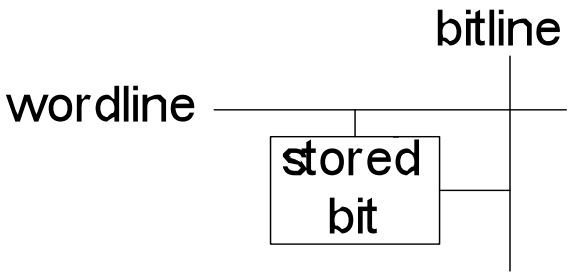
Robert Dennard, 1932 -

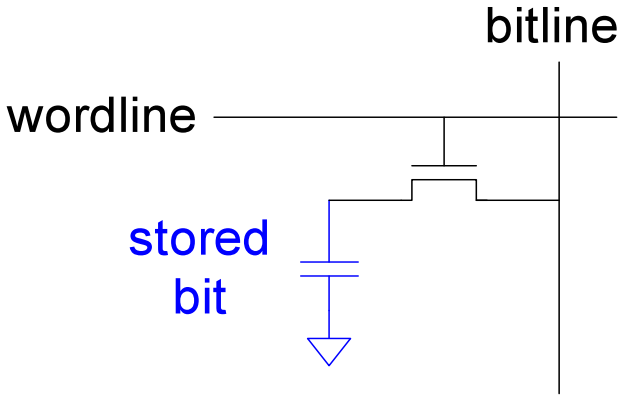
- Invented DRAM in 1966 at IBM
- Others were skeptical that the idea would work
- By the mid-1970's DRAM was in virtually all computers



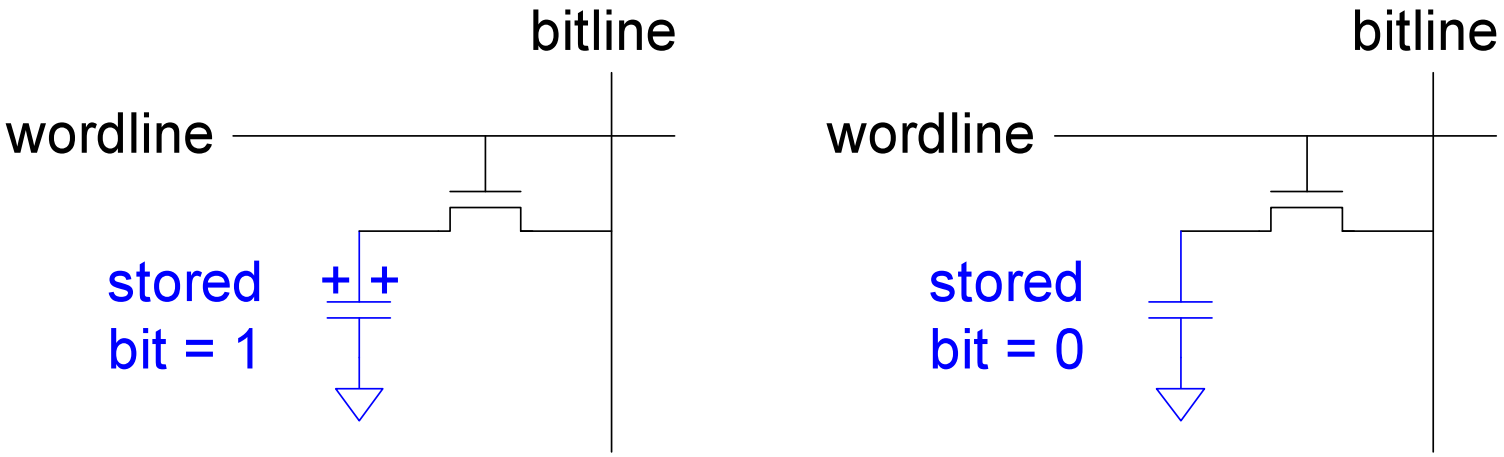
DRAM

- Data bits stored on a capacitor
- Called *dynamic* because the value needs to be refreshed (rewritten) periodically and after being read:
  - Charge leakage from the capacitor degrades the value
  - Reading destroys the stored value

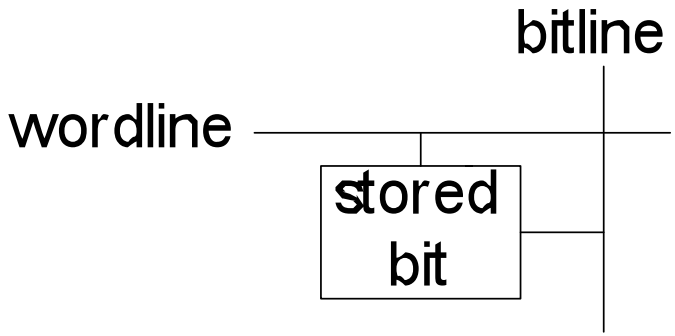


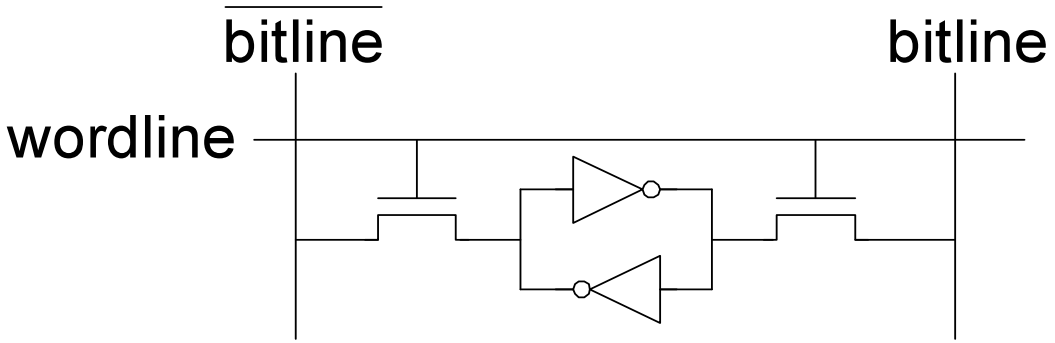


DRAM

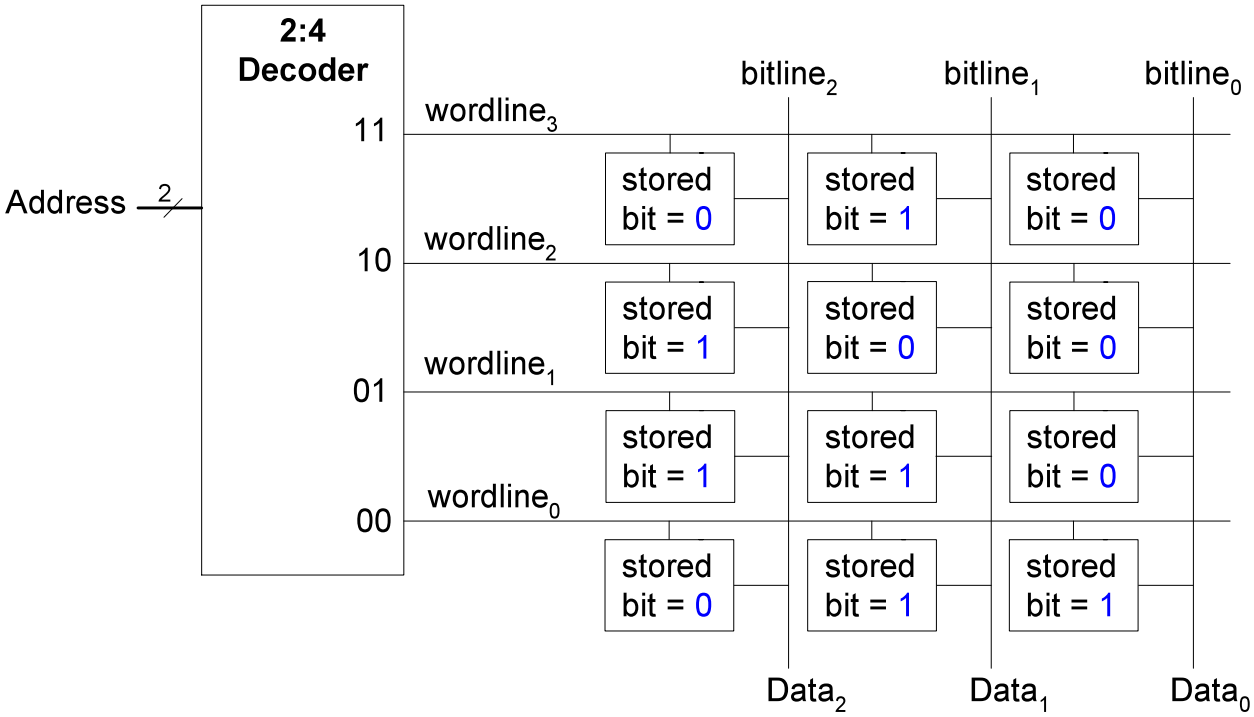


SRAM

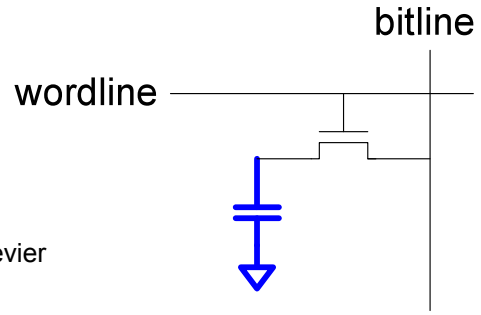




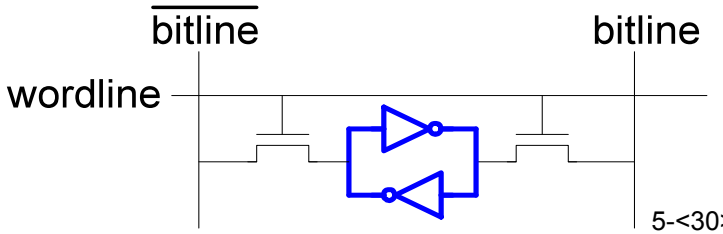
Memory Arrays



DRAM bit cell:

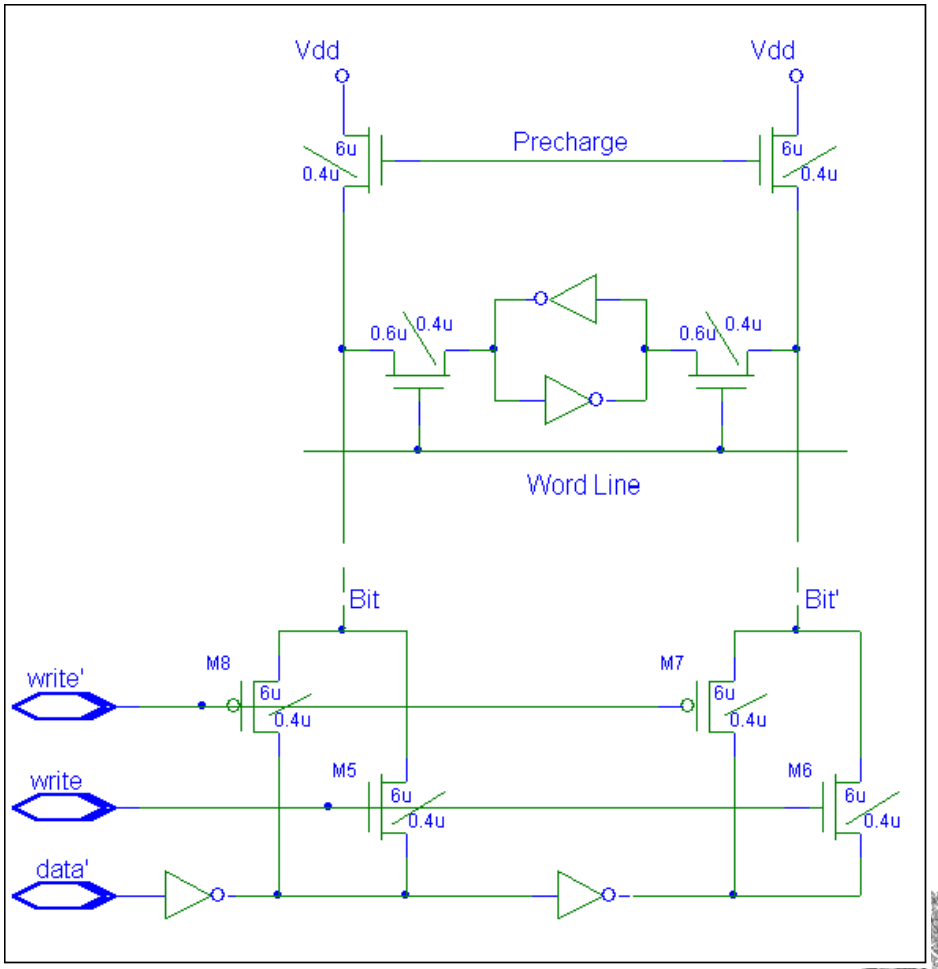


SRAM bit cell:



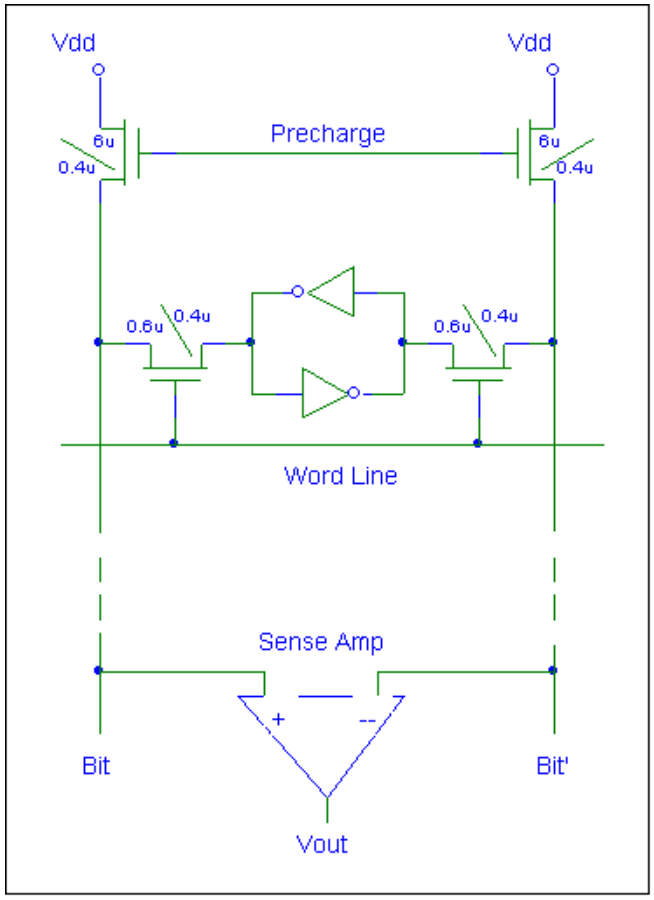
Writing an SRAM cell

- Writing is easy
  - Enable the word line
  - Overwrite the cell's contents



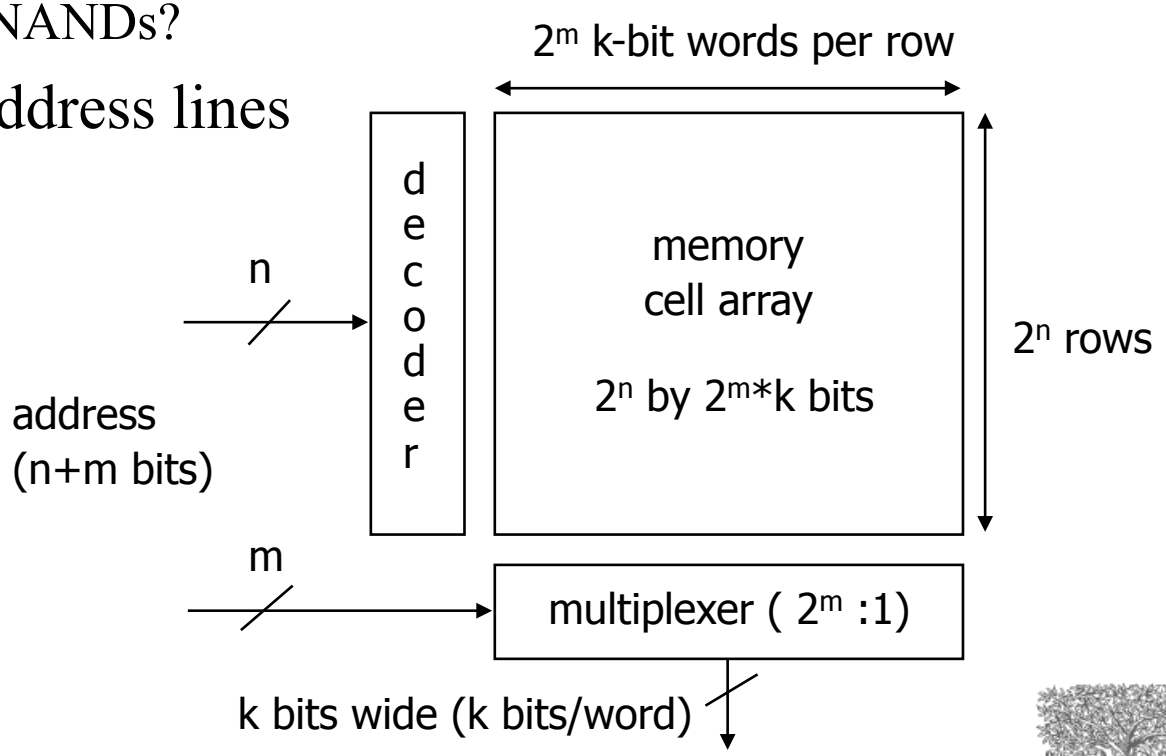
Reading an SRAM cell

- Reading is hard
  - Bit line capacitance is huge (CBL > 1pF)
    - SRAM cell can't slew bit-line quickly
    - Can lose the cell's contents
- Design peripheral circuitry to read reliably
  - Precharge bit lines lines to Vdd/2
  - Then release the precharge
    - Reduces chance of erroneous switching
  - Use sense amps for differential sensing
    - Detect small voltage change
    - Small swings ⇒ low power



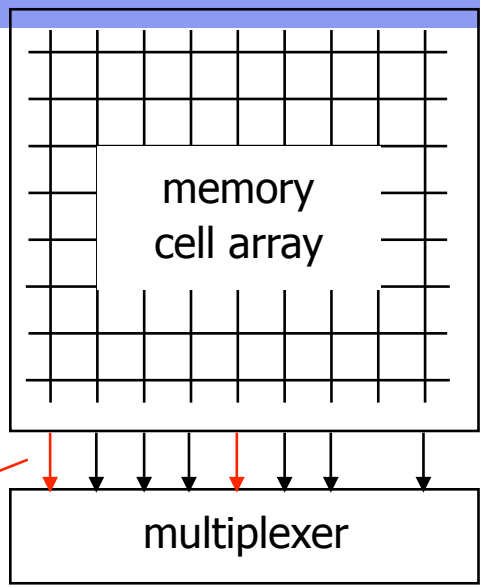
Addressing a memory

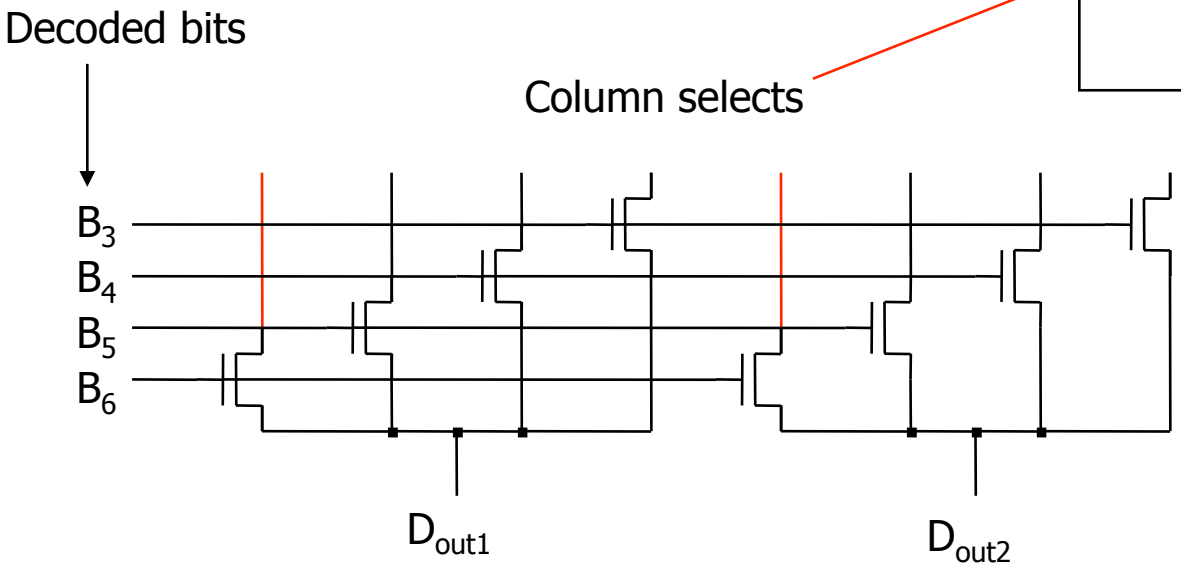
- Want square memory array
- Want simple decoding logic
  - Problem: A 1Meg×1 RAM uses 1,048,576 20-input NANDs?
- Want short data & address lines



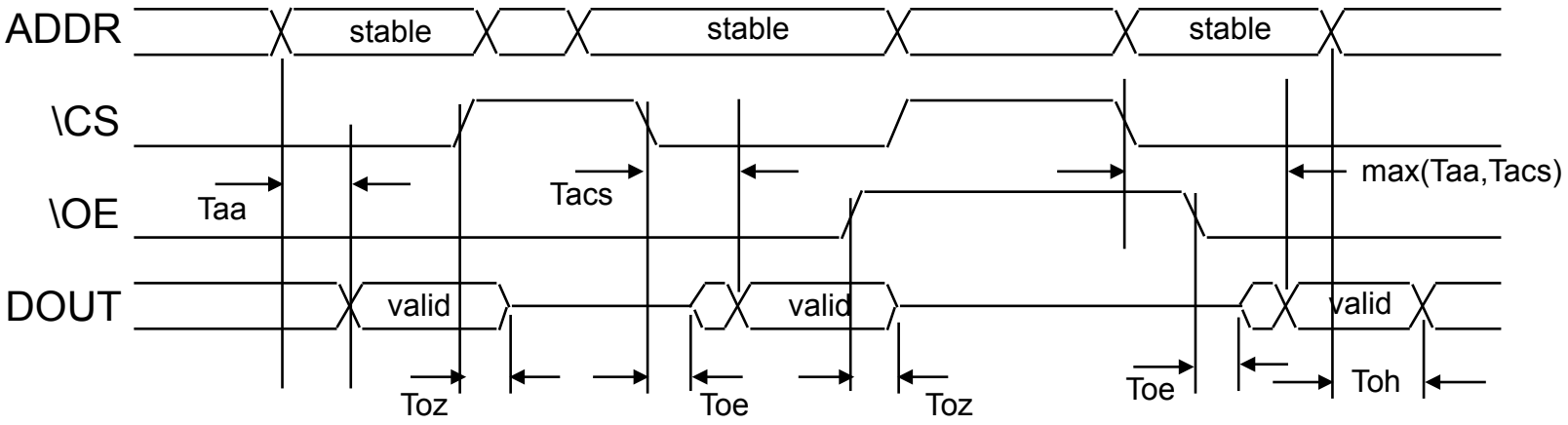
Multiplexer

- Selects a column (bit or byte)
  - Column decoding needed at end of read or write





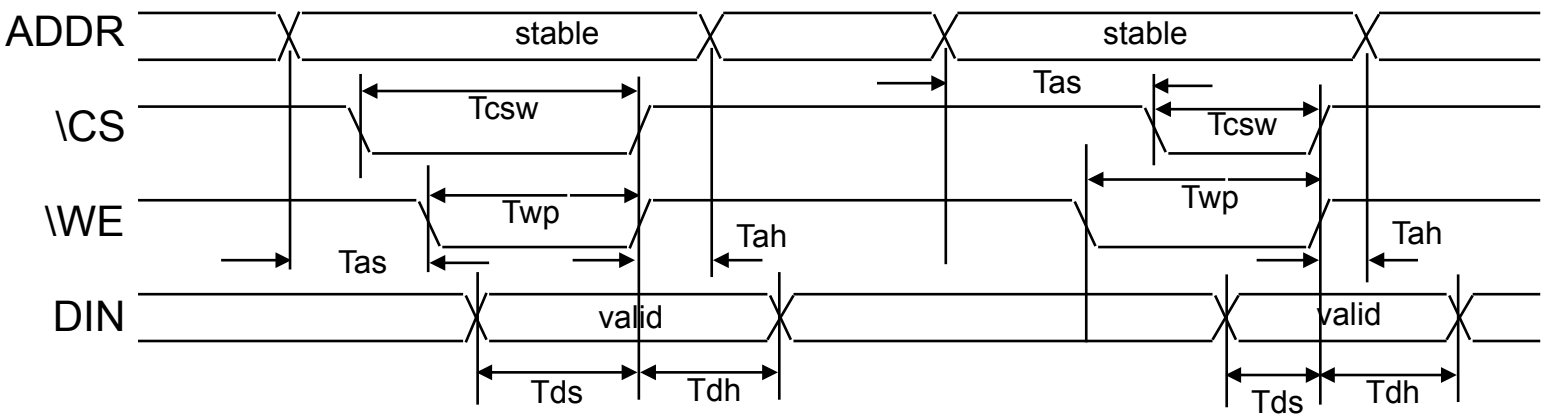
SRAM read timing



Taa: access time from address
Tacs: access time from chip select
Toz: output disable time
Toe: output enable time
Toh: output hold time

ELSEVIER

SRAM write timing



Tas: address setup time before write
Tah: address hold time after write
Tcsw: chip select time before write
Twp: write pulse minimum width
Tds: data setup time
Tdh: data hold time

Address must be stable before WE', else invalid data may glitch other cells

DRAM notes

- Definitions
  - RAS ≡ row-address strobe
  - CAS ≡ column-address strobe
- DRAMs share address pins
  - Row address and column address use same pins
    - RAS and CAS load row and column addresses
  - Example: 1MB DRAM has 10 address pins
    - RAS loads 10 row addresses
    - CAS loads 10 column addresses
- Older DRAMs are asynchronous
  - Timing depends on rising and falling edges of RAS and CAS

DRAM notes (con't)

- Most DRAMs have built-in refresh counters
  - Counter cycles through rows
  - You supply \CAS pulse (or \CAS-before-\RAS)
    - Internal logic reads and rewrites a row
- Most DRAMs allow fast page-mode reads/writes
  - Use when reading multiple words from same row
  - Supply RAS once but CAS many times
- Modern DRAMs are synchronous (SDRAM)
  - Read/write on a clock edge
  - Pipelining to internal memory banks
  - Complex controller handles hidden refresh
  - DDR + RamBus

DRAM refresh

- Assert RAS' to refresh row
  - Reads data and writes it back
  - Takes ~1% of DRAM cycles