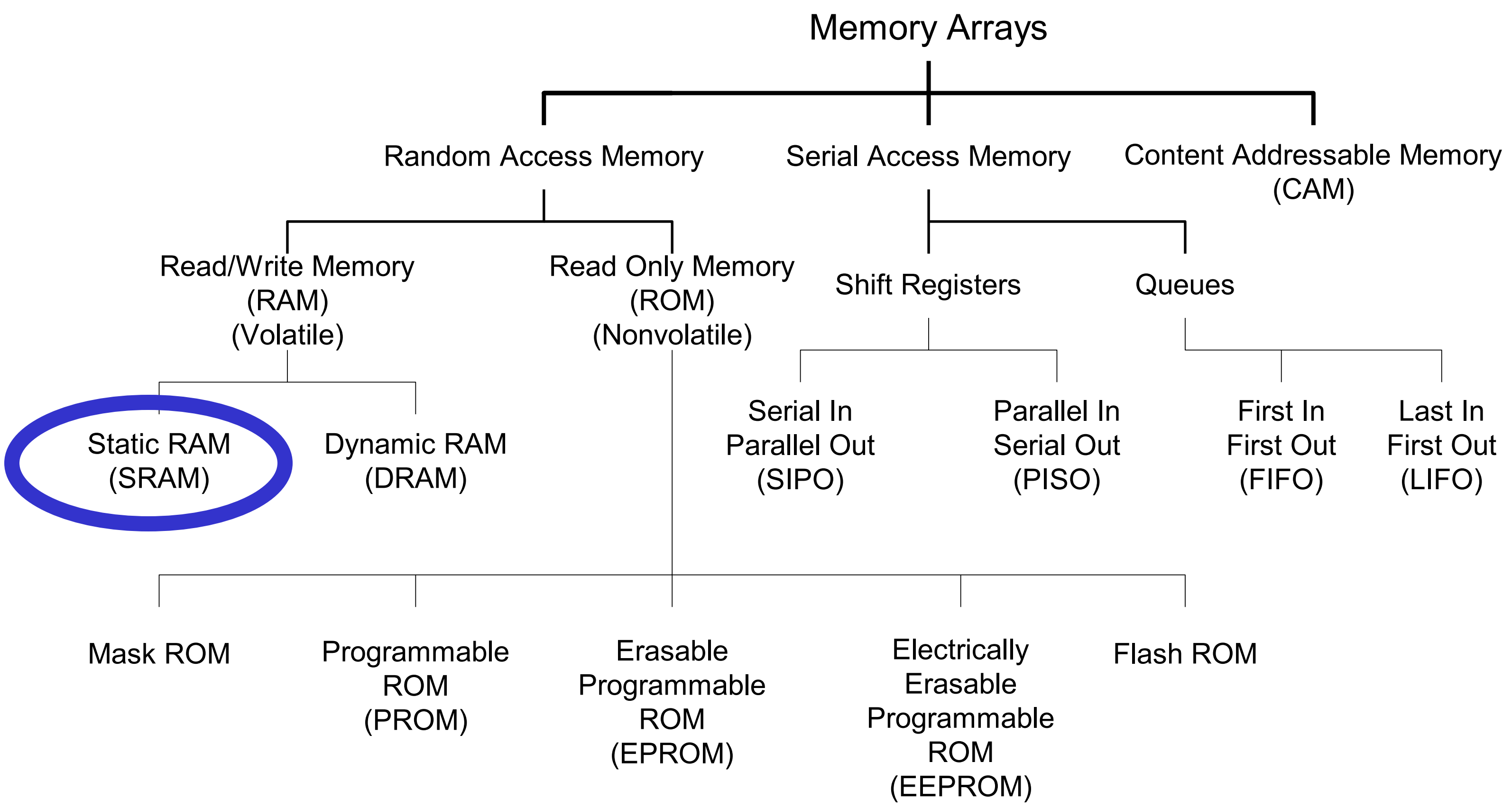
ECE1388 VLSI Design Methodology

Lecture 14

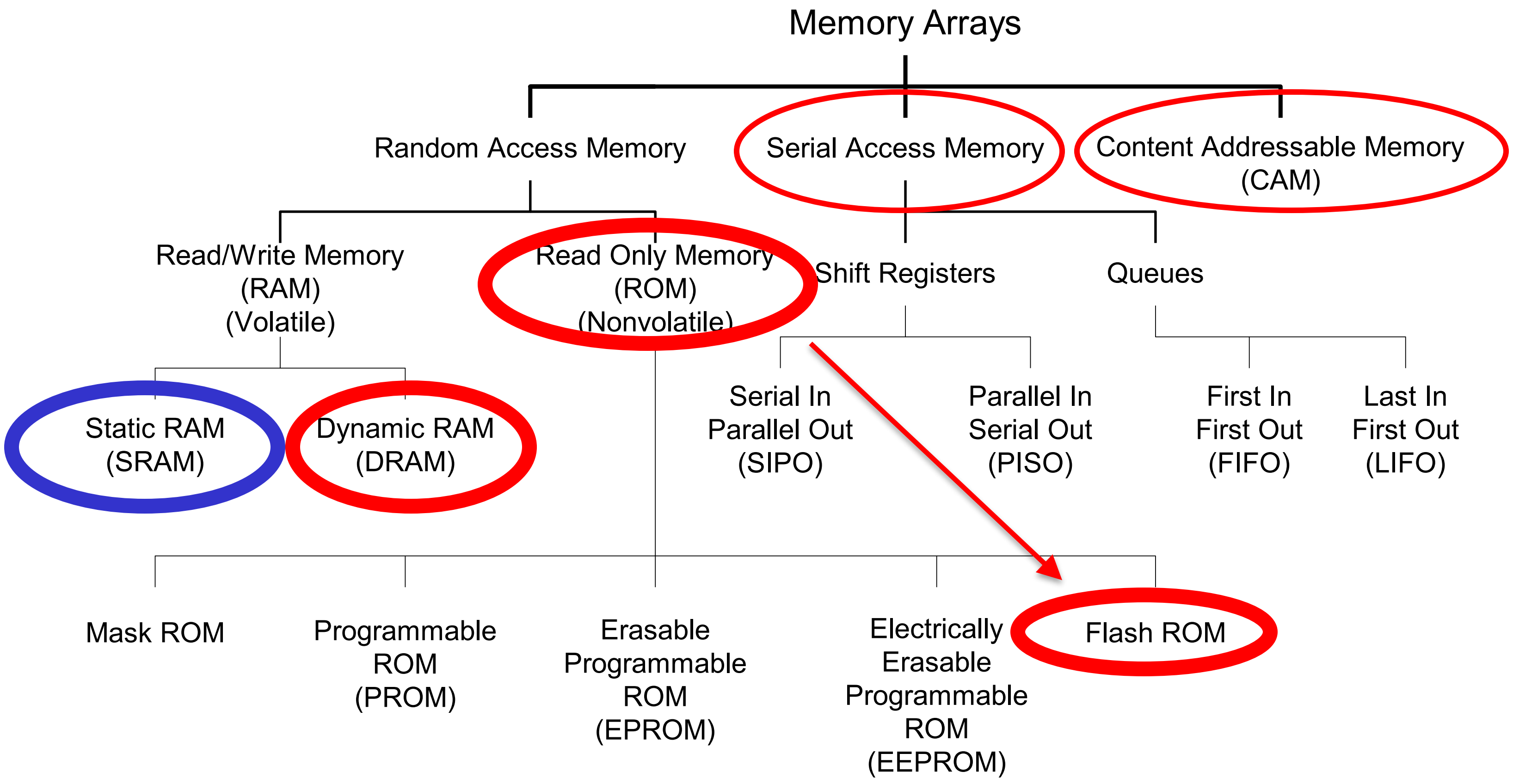
Memories II: DRAM, ROM, CAM

Adapted from Weste, Harris CMOS VLSI Design

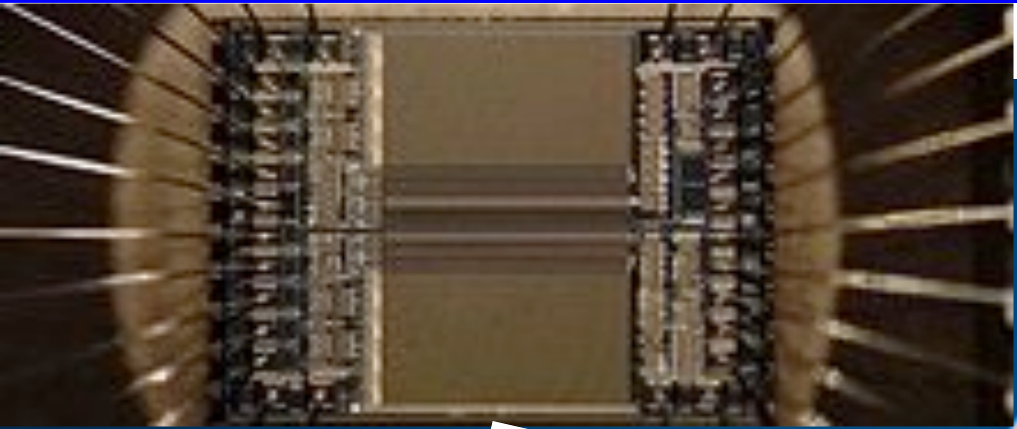
Memory Arrays


Memory Arrays
Random Access Memory
Serial Access Memory
Content Addressable Memory (CAM)
Read/Write Memory (RAM) (Volatile)
Read Only Memory (ROM) (Nonvolatile)
Shift Registers
Queues
Static RAM (SRAM)
Dynamic RAM (DRAM)
Serial In Parallel Out (SIPO)
Parallel In Serial Out (PISO)
First In First Out (FIFO)
Last In First Out (LIFO)
Mask ROM
Programmable ROM (PROM)
Erasable Programmable ROM (EPROM)
Electrically Erasable Programmable ROM (EEPROM)
Flash ROM

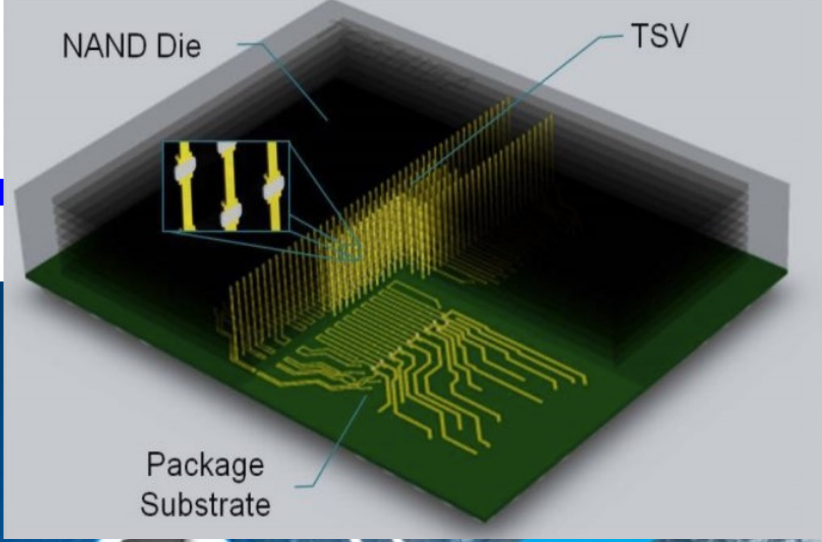
Memory Arrays

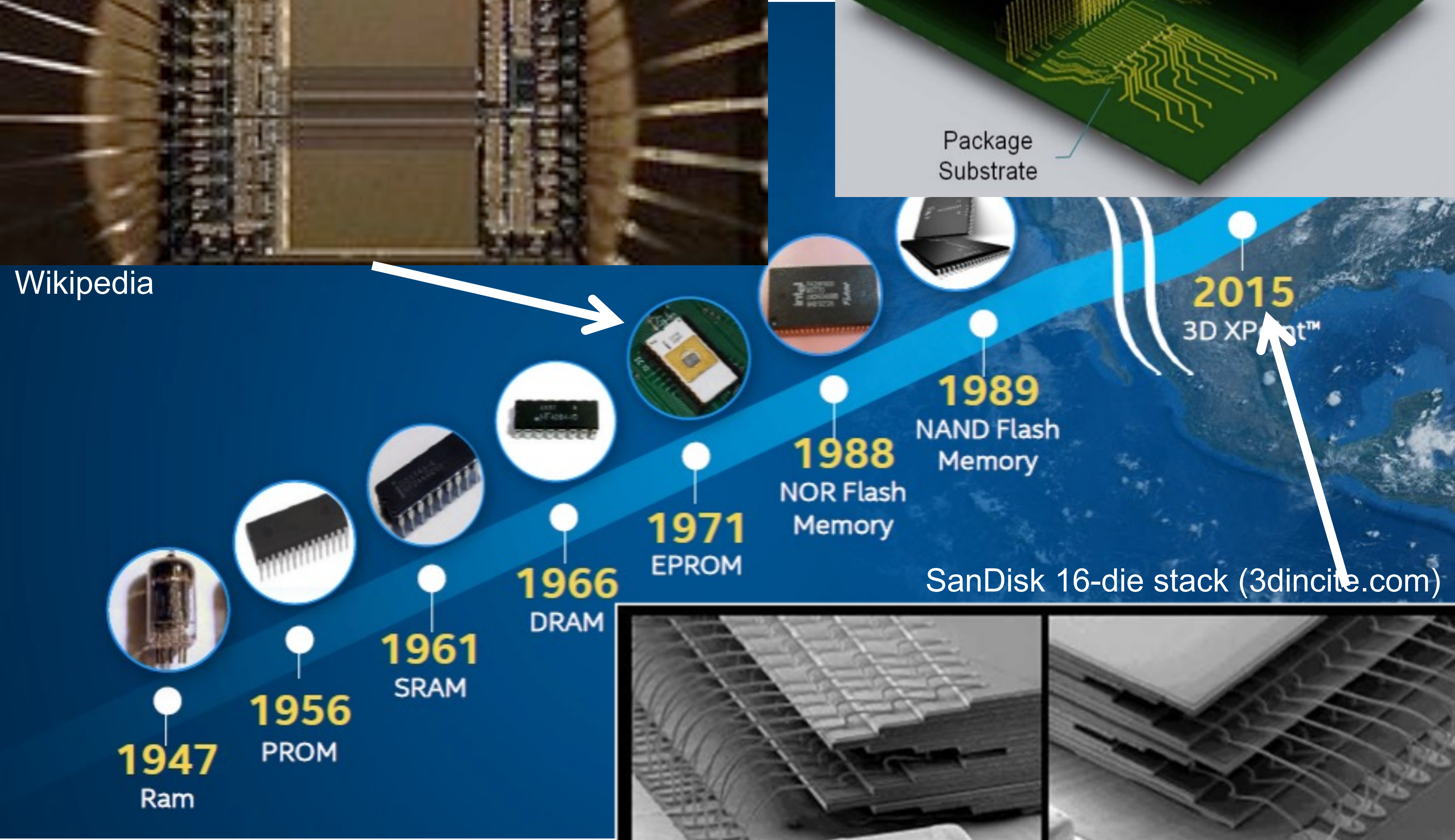

Memory Arrays
Random Access Memory
Serial Access Memory
Content Addressable Memory (CAM)
Read/Write Memory (RAM) (Volatile)
Read Only Memory (ROM) (Nonvolatile)
Shift Registers
Queues
Static RAM (SRAM)
Dynamic RAM (DRAM)
Serial In Parallel Out (SIPO)
Parallel In Serial Out (PISO)
First In First Out (FIFO)
Last In First Out (LIFO)
Mask ROM
Programmable ROM (PROM)
Erasable Programmable ROM (EPROM)
Electrically Erasable Programmable ROM (EEPROM)
Flash ROM

Semiconductor Memories

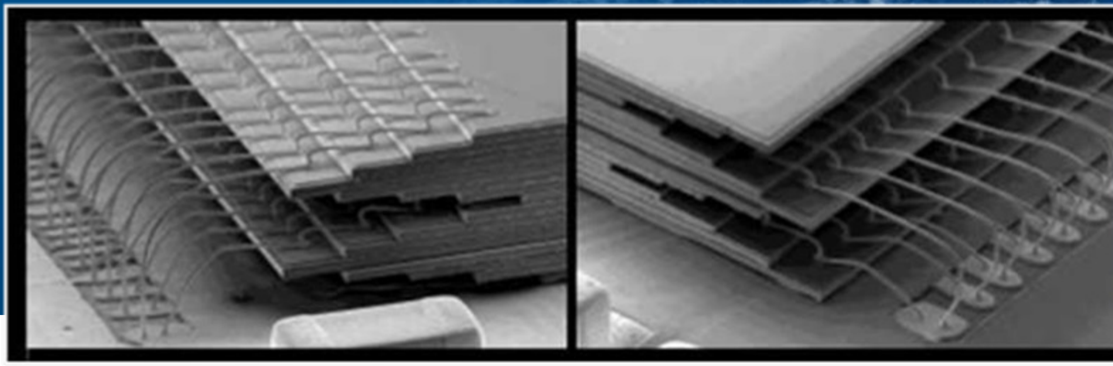


Wikipedia





SanDisk 16-die stack (3dincite.com)

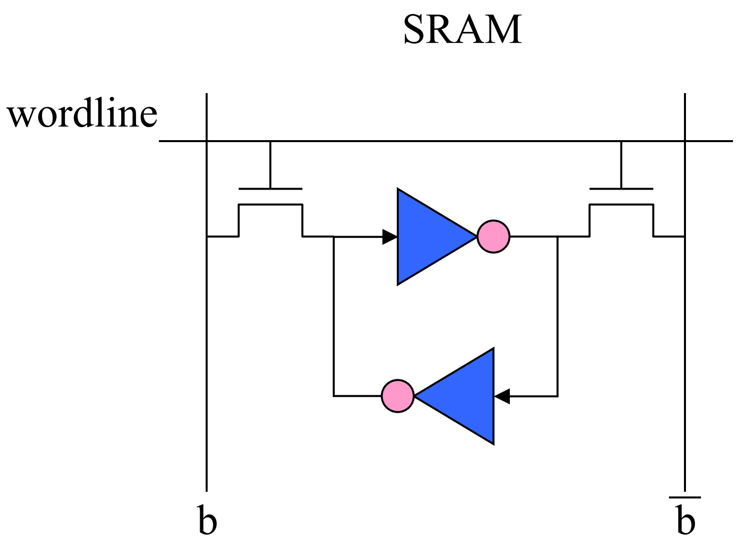


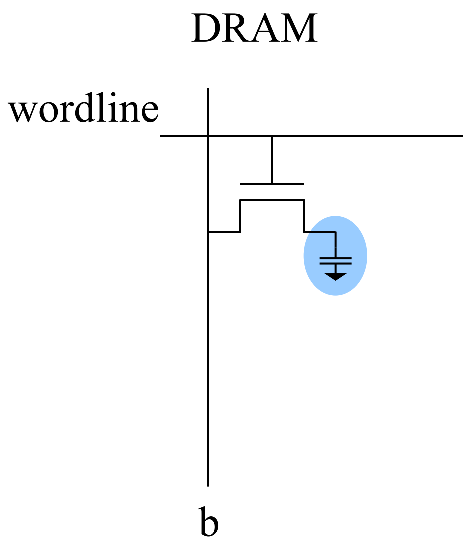
themFemoryguy.com

Outline

- DRAM
- Read-Only Memories (ROM), including Flash
- Serial Access Memories
- Content-Addressable Memories (CAM)

SRAM vs. DRAM

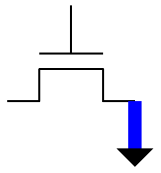




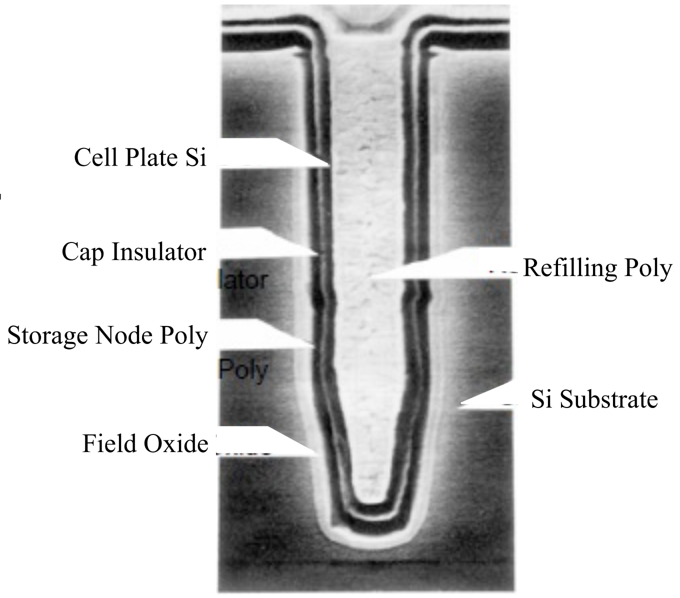
- SRAM: 6T per bit
  - built with standard high-speed CMOS technology
- DRAM: 1T per bit
  - built with special DRAM process optimized for density

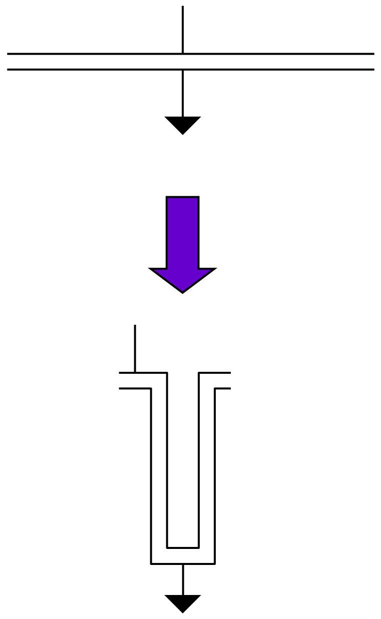
Adapted from Prof. M. Prvulovic CS 4290/6290 notes are Georgia Tech

DRAM Capacitor



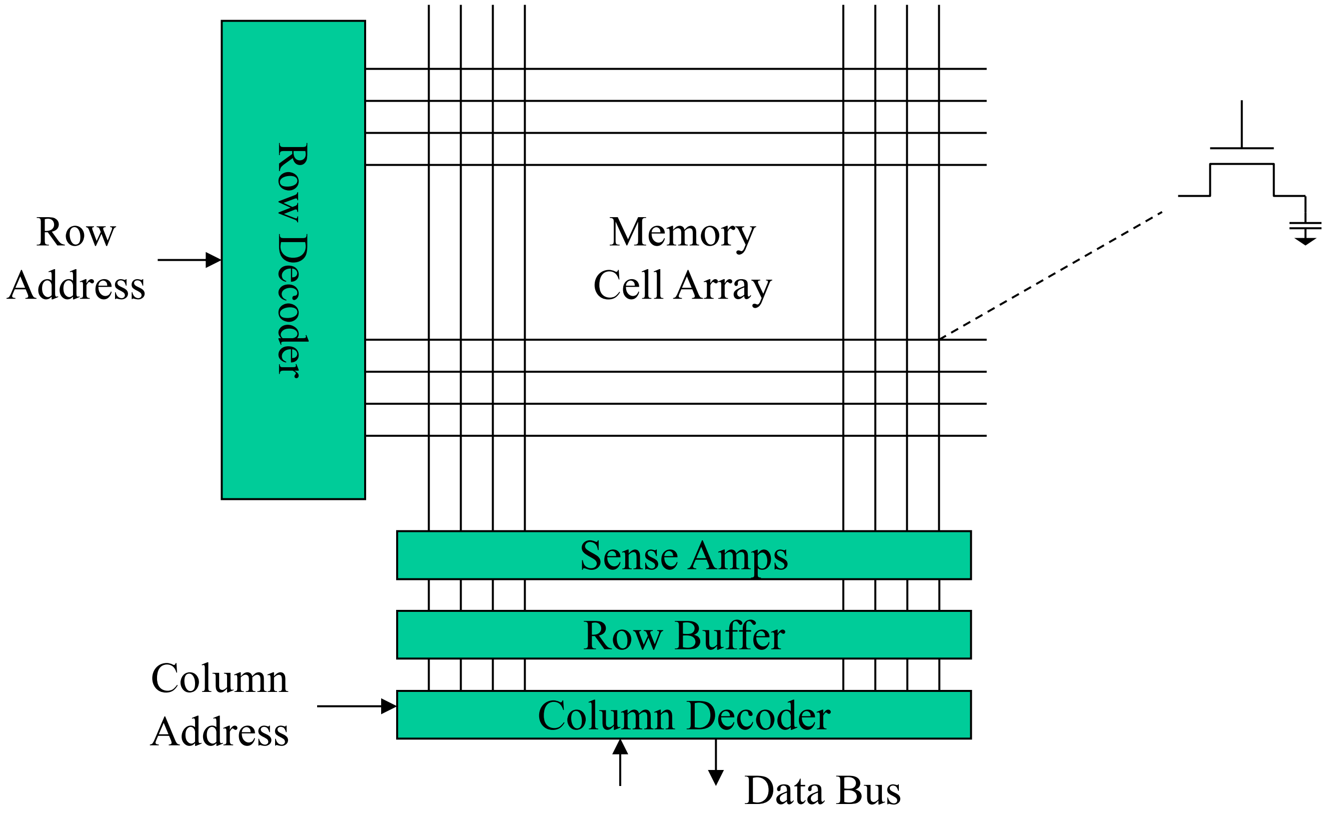
“Trench Cell”





Adapted from Prof. Nikolic EECS141/2003 notes from UC-Berkeley

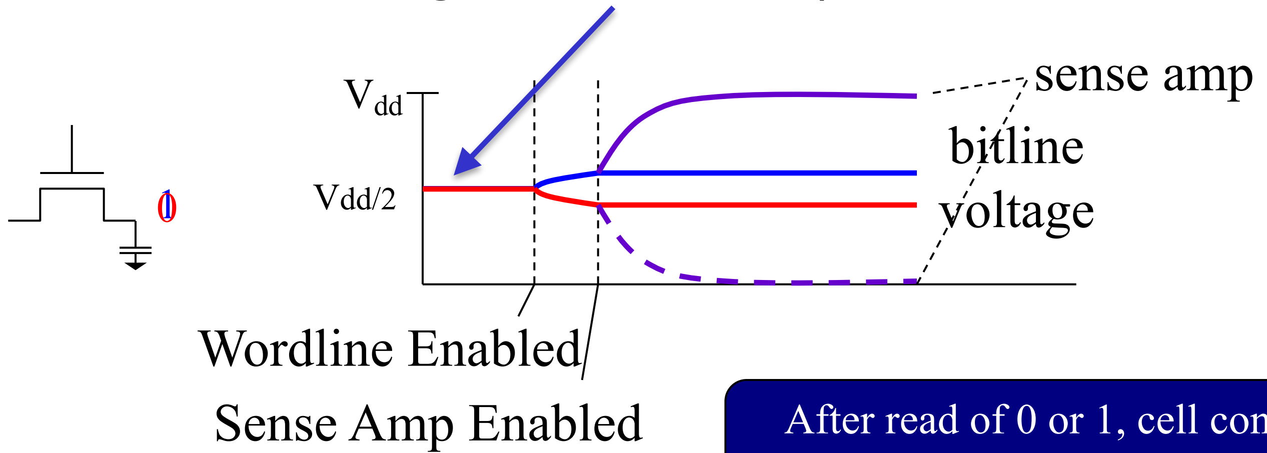
DRAM Bank



Adapted from Prof. M. Prvulovic CS 4290/6290 notes at Georgia Tech

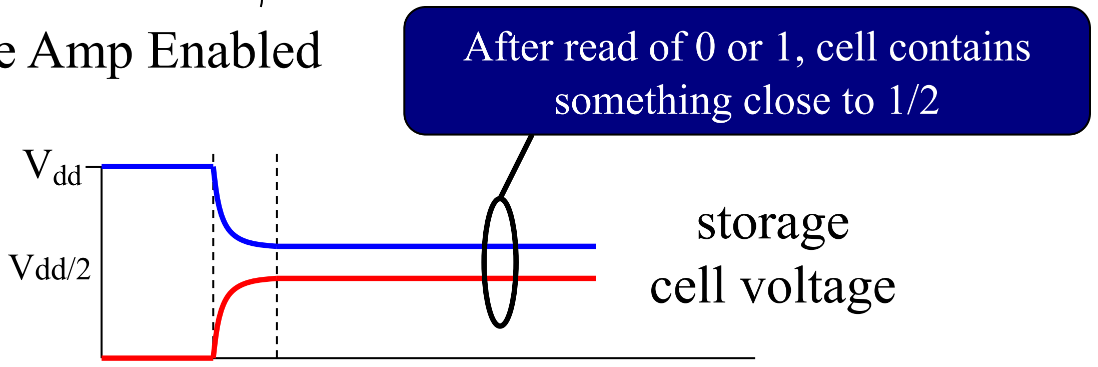
Destructive Read

- Bitline is precharged to Vdd/2 (versus Vdd in SRAM)



Charge sharing between cell capacitance and BL capacitance:

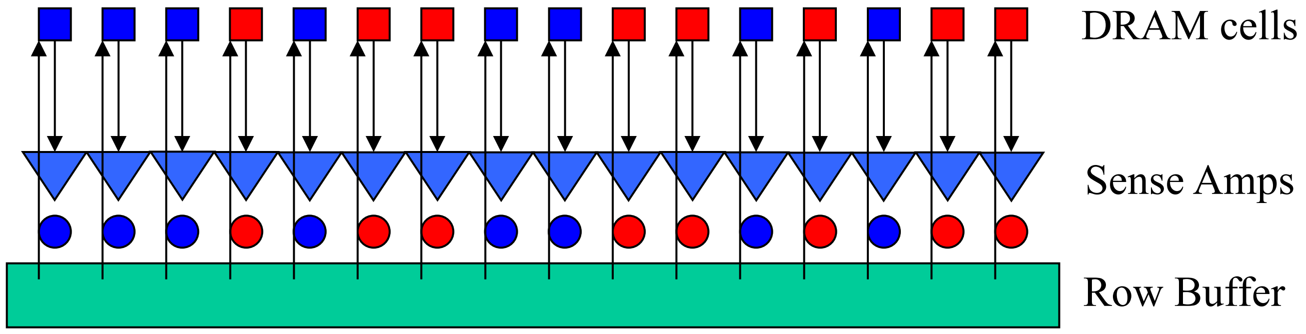
enables the readout but destroys the stored value



Adapted from Prof. M. Prvulovic CS 4290/6290 notes at Georgia Tech

Refresh / Write

- So after a read, the contents of the DRAM cell are gone
- The values are stored in the row buffer
- **Refresh**: write them back into the cells for the next read in the future
- **Write**: modify some bits in the row butter and write back



Adapted from Prof. M. Prvulovic CS 4290/6290 notes at Georgia Tech

DRAM vs. SRAM (2)

- Differences vs SRAM
    - reads are *destructive*: contents are erased during reading
- – row buffer
    - read lots of bits all at once, and then parcel them out based on different column addresses
        - – similar to reading a full cache line, but only accessing one word at a time
    - "Fast-Page Mode" FPM DRAM organizes the DRAM row to contain bits for a complete page
        - – row address held constant, and then fast read from different locations from the same page

Adapted from Prof. M. Prvulovic CS 4290/6290 notes at Georgia Tech

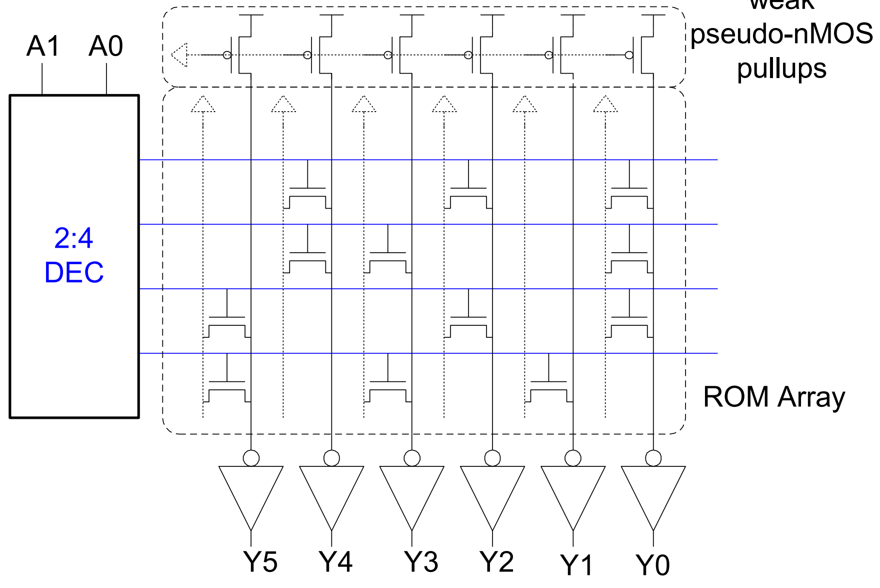
Read-Only Memories

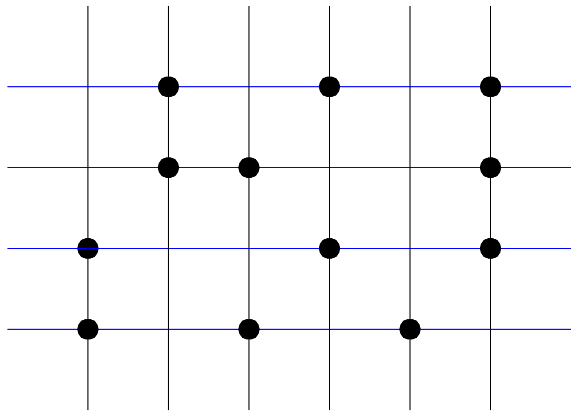
- Read-Only Memories are nonvolatile
  - Retain their contents when power is removed
- Mask-programmed ROMs use one transistor per bit
  - Presence or absence determines 1 or 0
- Two key architectures
  - NOR ROM
  - NAND ROM

NOR 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





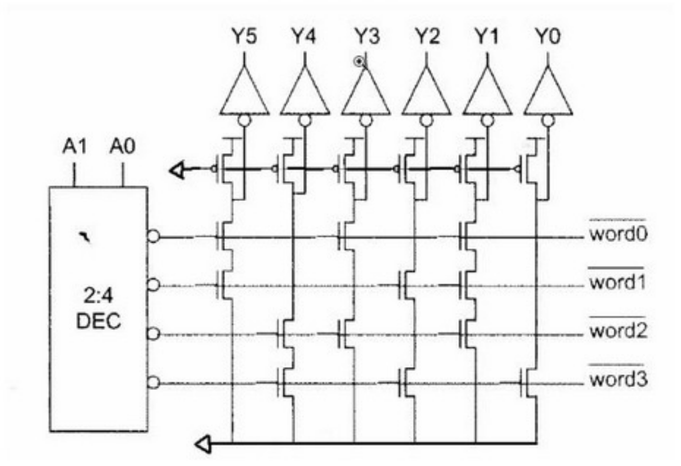
Looks like 6 4-input pseudo-nMOS NORs

NAND ROM Example

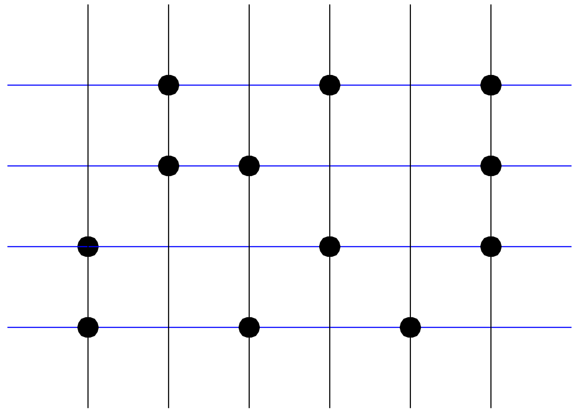
- 4-word x 6-bit ROM
  - Represented with dot diagram
  - Dots indicate 1's in ROM

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

NAND ROM



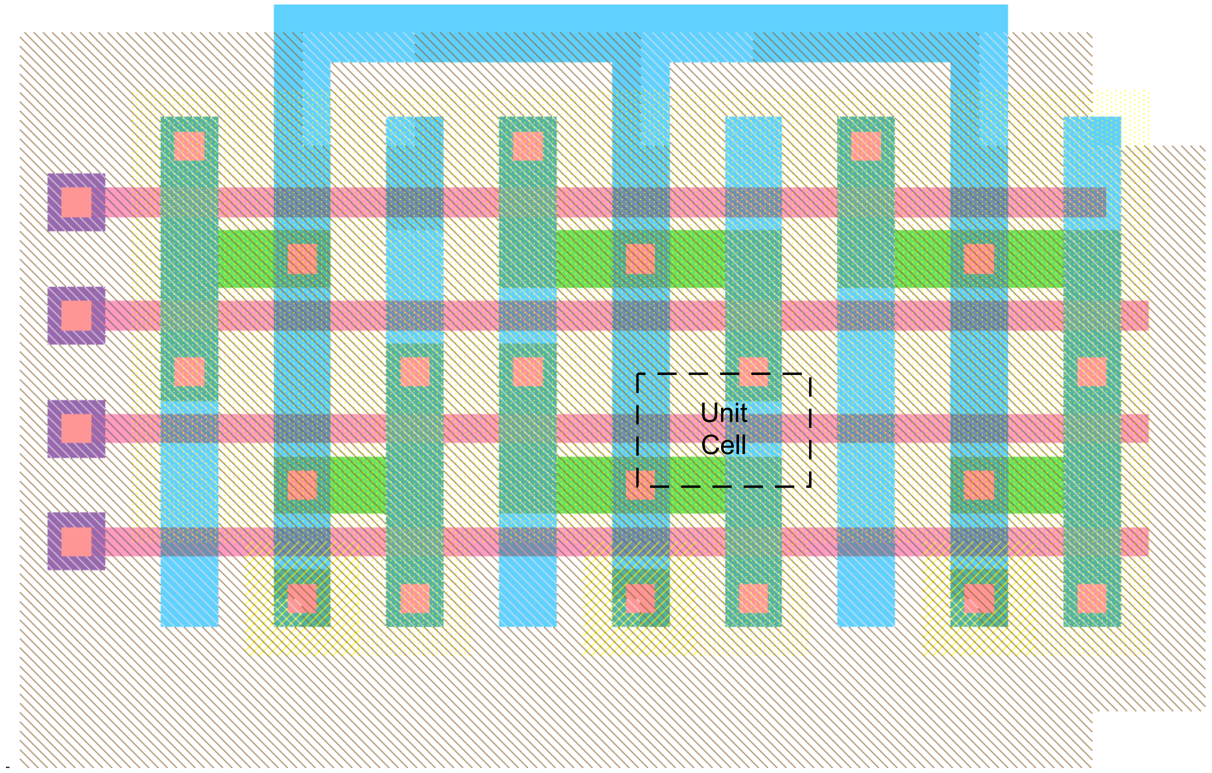
Weste/Harris F11.41



Looks like 6 4-input pseudo-nMOS NANDs

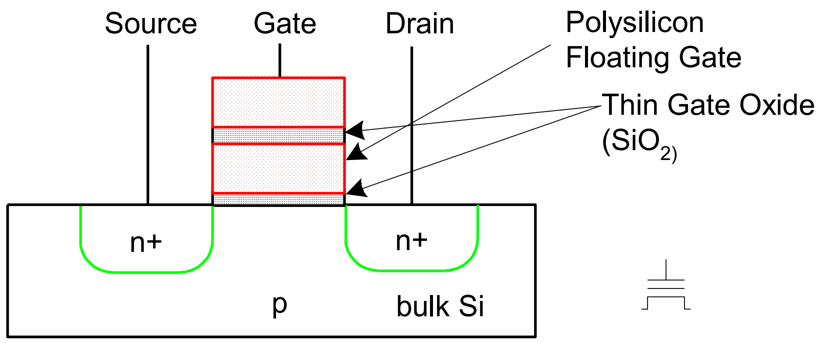
ROM Array Layout (NOR)

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



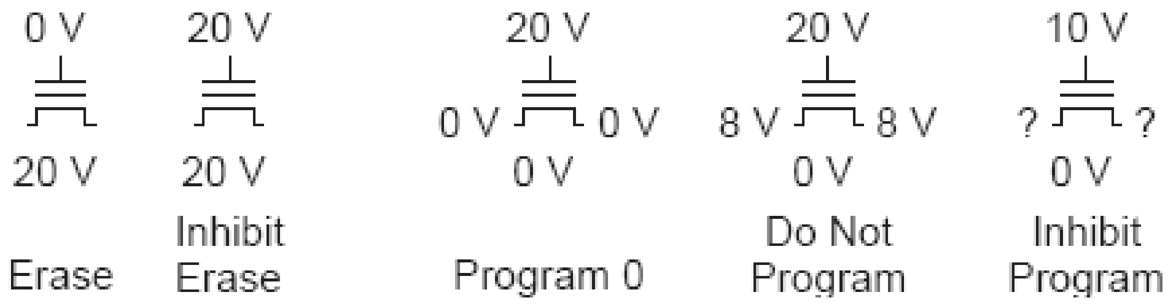
PROMs and EPROMs

- 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

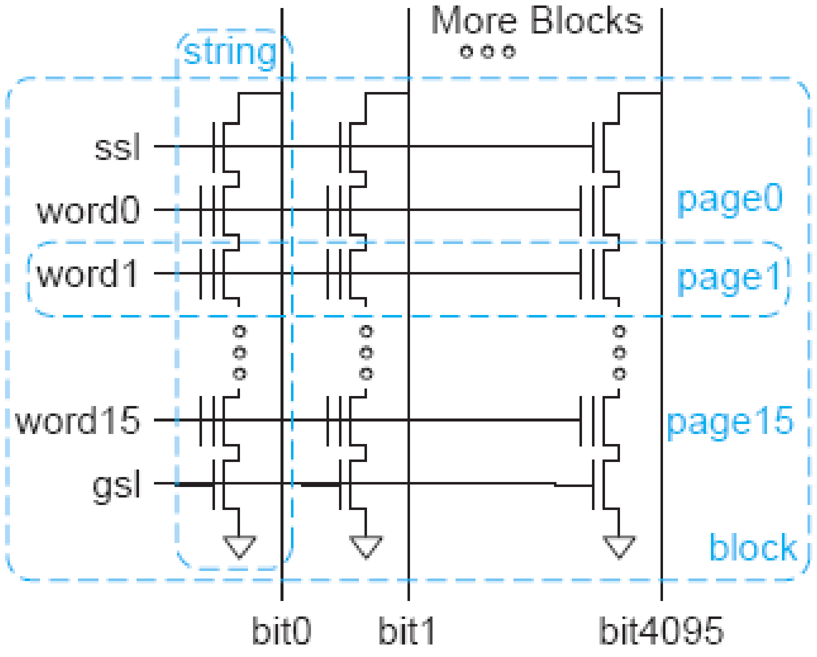


Flash Programming

- Charge on floating gate determines V_t
- Logic 1: negative V_t
- Logic 0: positive V_t
- Cells erased to 1 by applying a high body voltage so that electrons tunnel off floating gate into substrate
- Programmed to 0 by applying high gate voltage



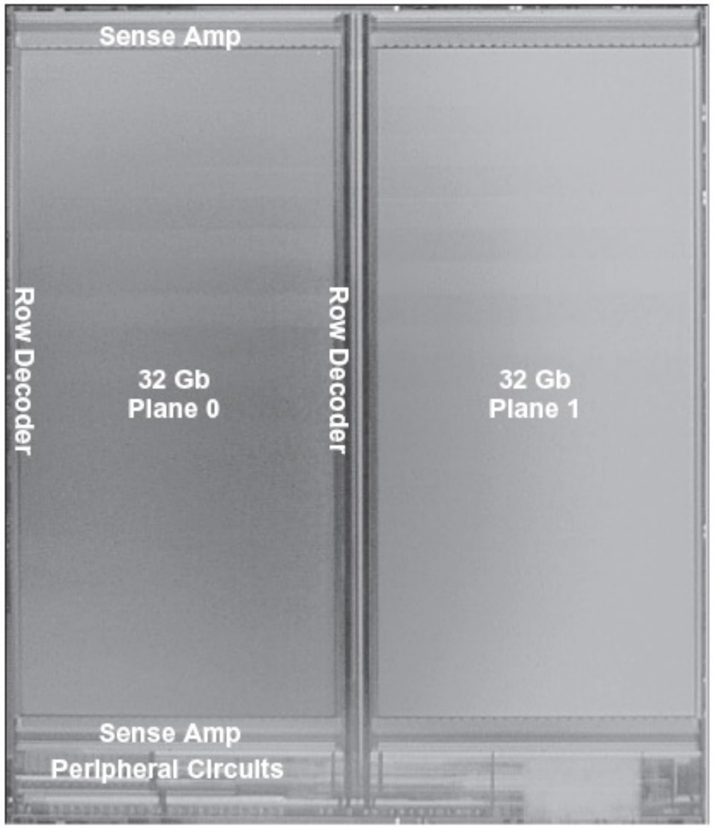
NAND Flash



- High density, low cost / bit
  - Programmed one page at a time
  - Erased one block at a time
- Example:
  - 4096-bit pages
  - 16 pages / 8 KB block
  - Many blocks / memory

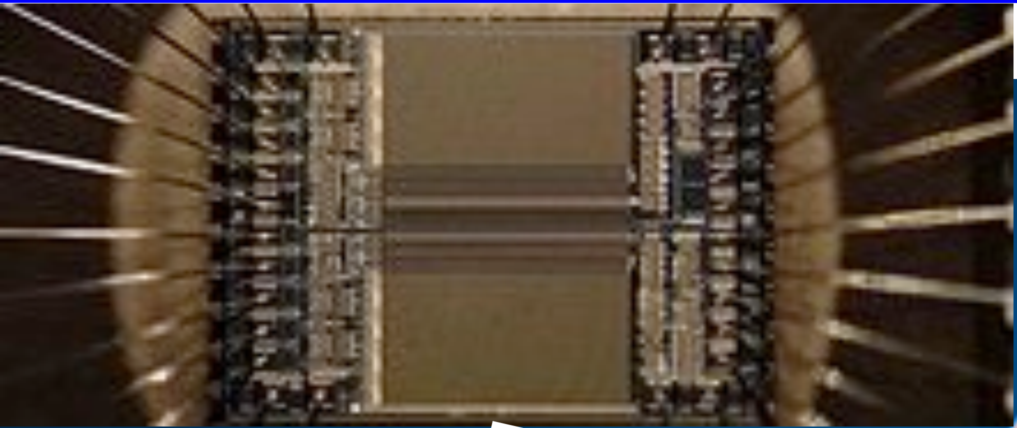
64 Gb NAND Flash

- 64K cells / page
- 4 bits / cell (multiple V_t)
- 64 cells / string
  - 256 pages / block
- 2K blocks / plane
- 2 planes

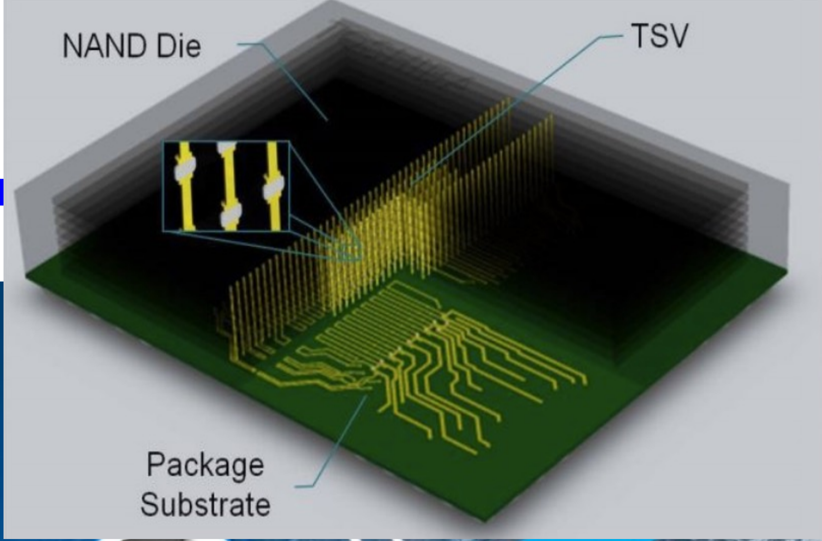


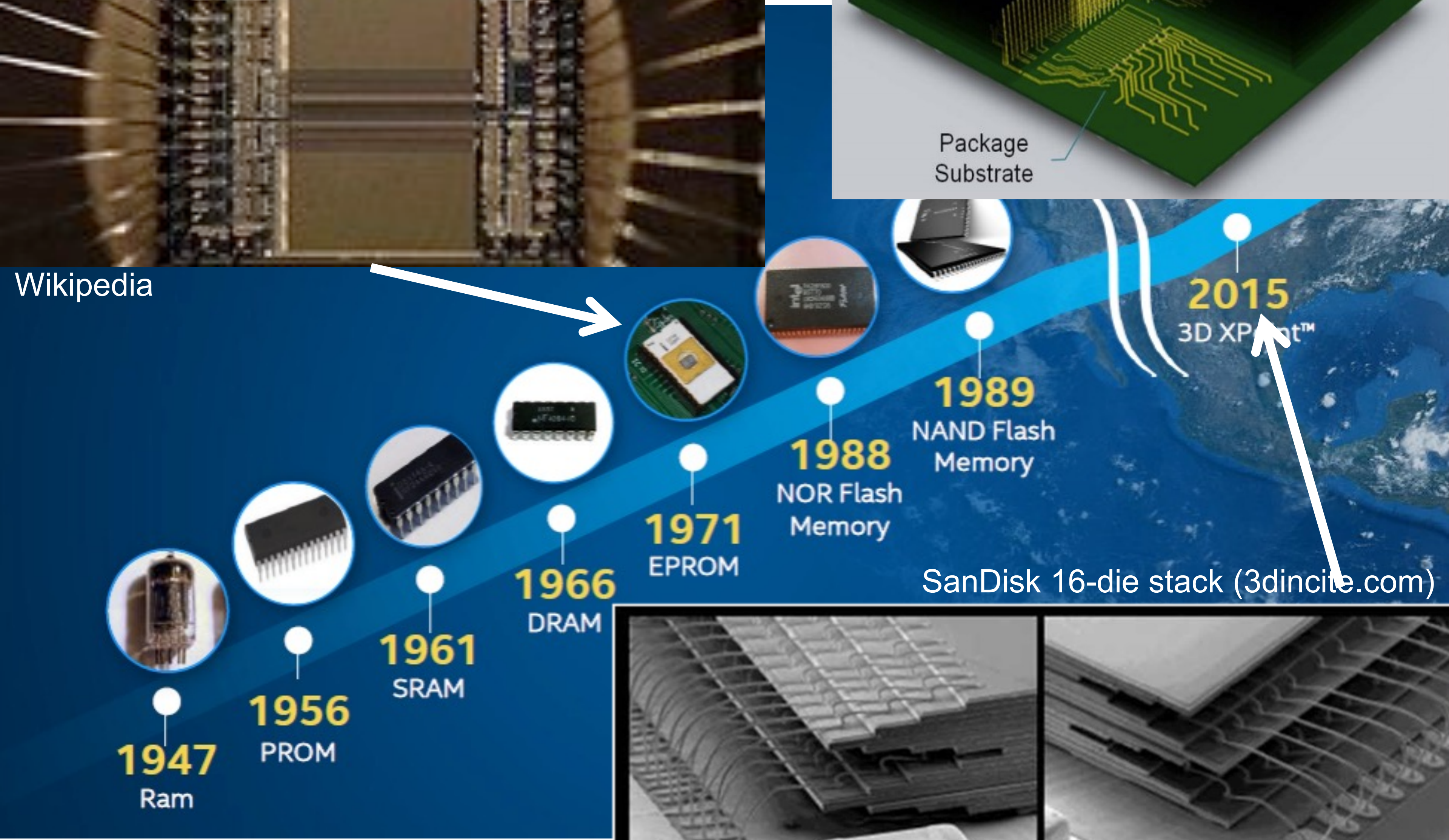
[Trinh09]

Semiconductor Memories

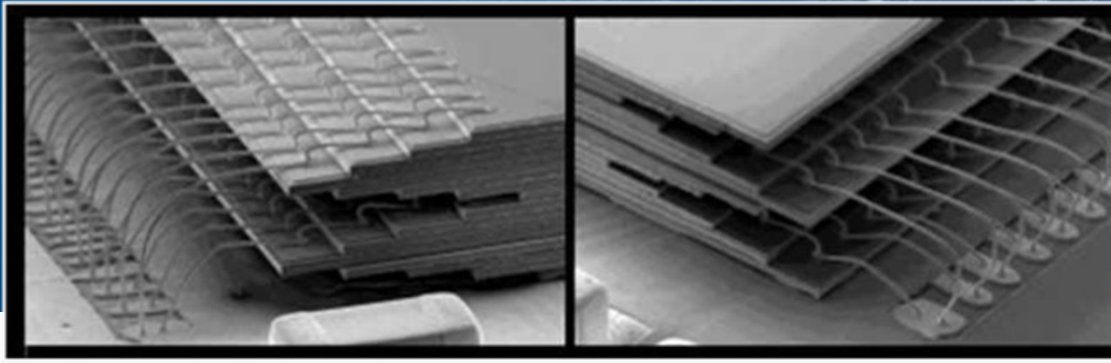


Wikipedia





SanDisk 16-die stack (3dincite.com)



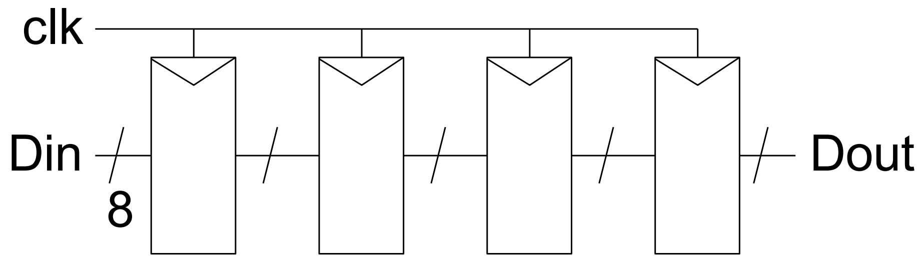
themermoryguy.com

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)

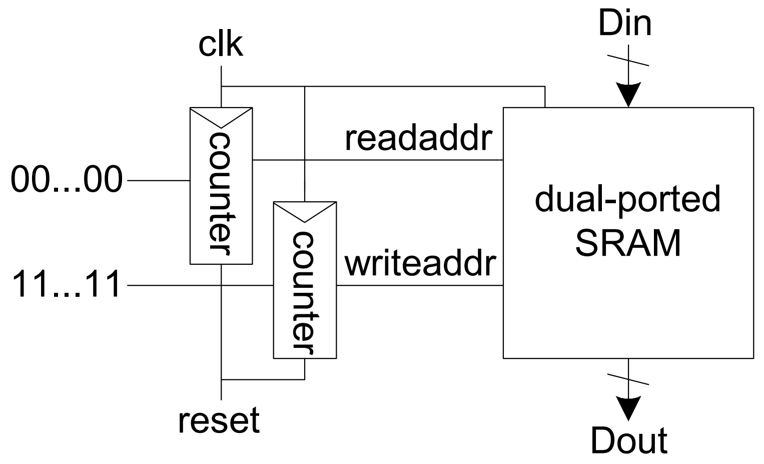
Shift Register

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



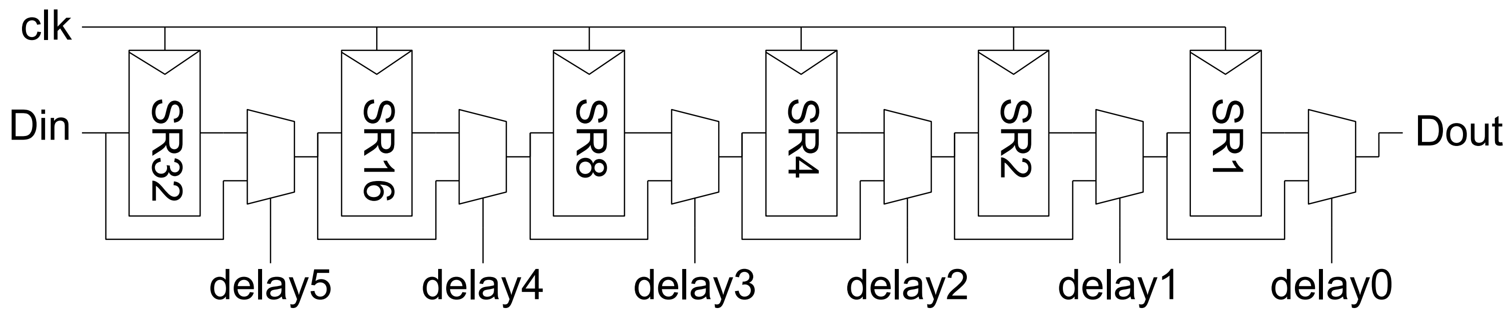
Denser Shift Registers

- 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



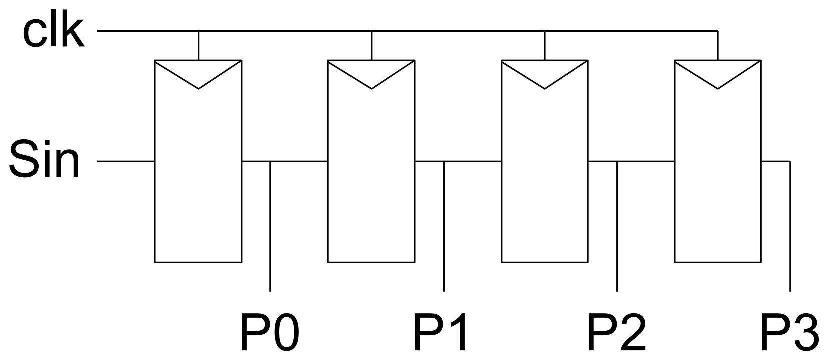
Tapped Delay Line

- 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



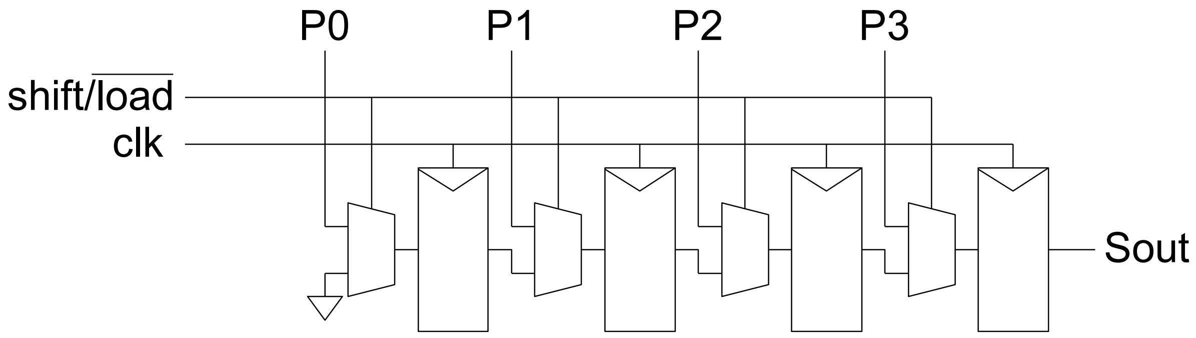
Serial In Parallel Out

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



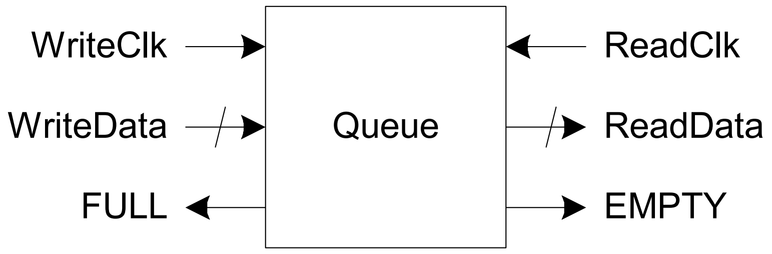
Parallel In Serial Out

- 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)

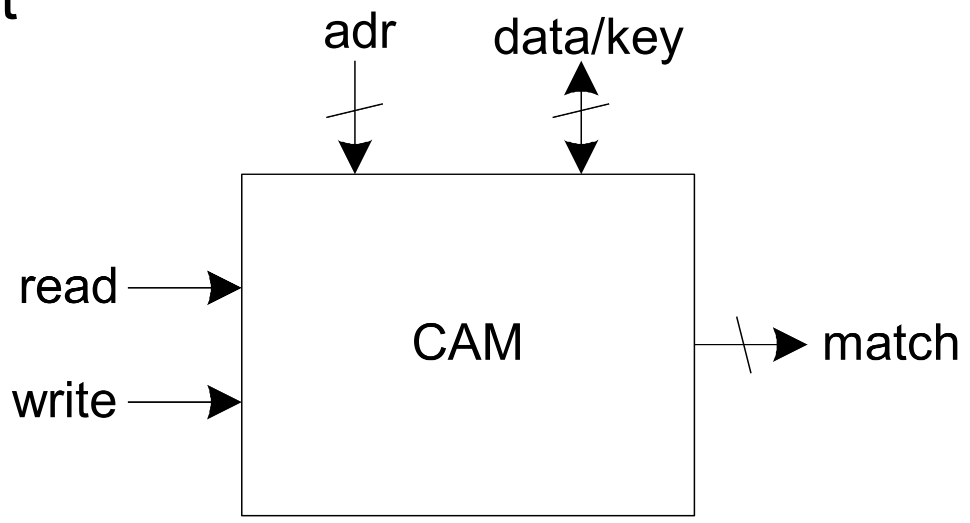


FIFO, LIFO Queues

- *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

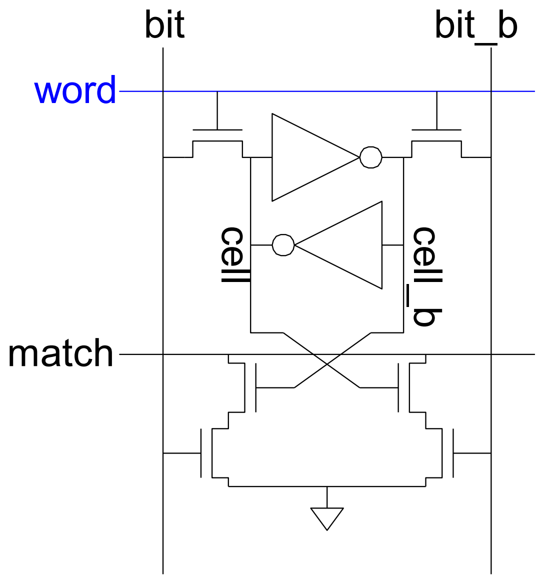
CAMs

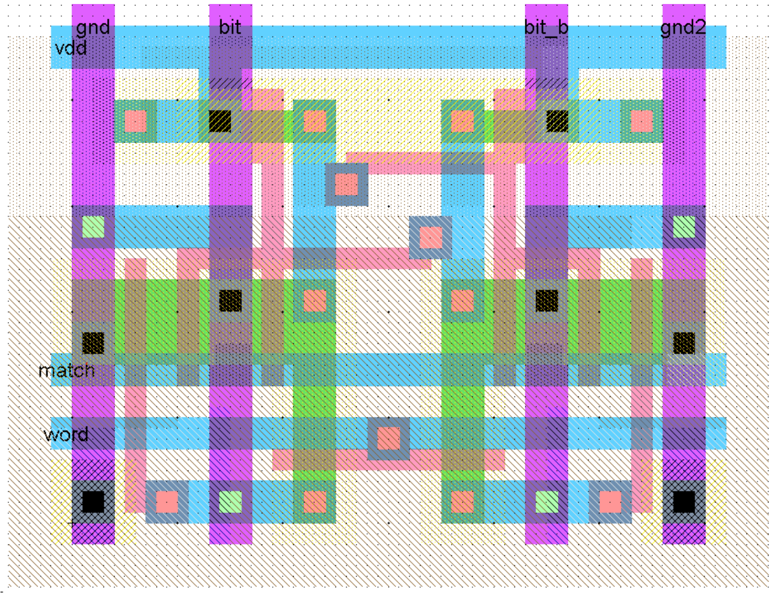
- Extension of ordinary memory (e.g. SRAM)
  - Read and write memory as usual
  - Also *match* to see which words contain a *key*
  - Used in internet routers: maps MAC address to a port



10T CAM Cell

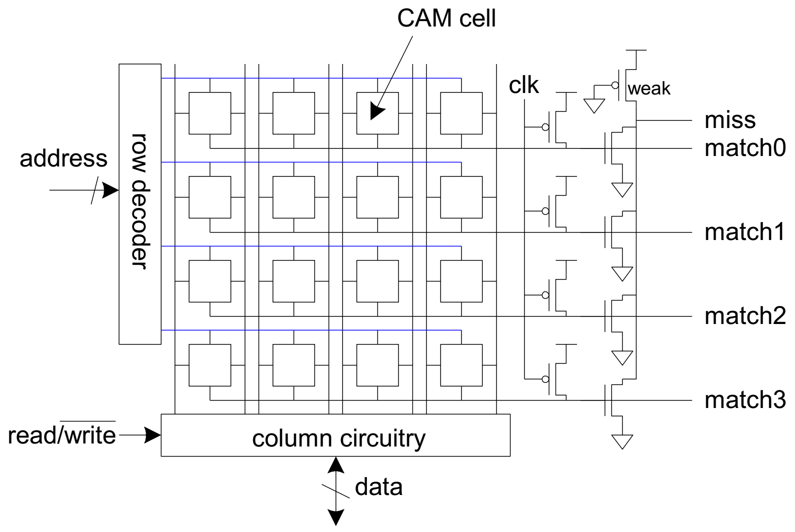
- Add four match transistors to 6T SRAM
  - 56 x 43 λ unit cell





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


CAM cell
clk
weak
miss
match0
address
row decoder
match1
match2
match3
read/$\overline{\text{write}}$
column circuitry
data