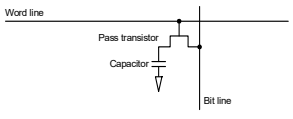
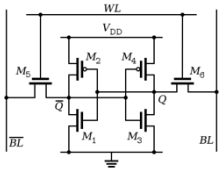
# Memory technology and optimizations (§ 2.3)



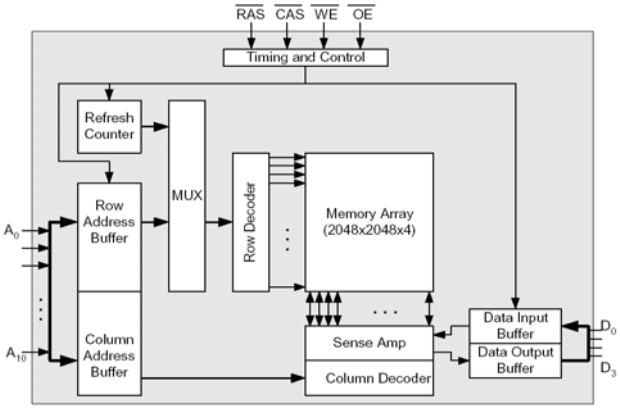
# Main Memory

- Performance of Main Memory:
  - Latency: affects Cache Miss Penalty
    - » *Access Time*: time between request and word arrival
    - » *Cycle Time*: minimum time between requests
  - Bandwidth: affects I/O & Large Block Miss Penalty (L2)
- Main Memory uses *DRAM*: Dynamic Random Access Memory
  - Needs to be refreshed periodically (one row at a time)
  - Addresses divided into 2 halves (Memory as a 2D matrix):
    - » *RAS* or *Row Access Strobe*
    - » *CAS* or *Column Access Strobe*
- Cache uses *SRAM*: Static RAM
  - No refresh (6 transistors/bit vs. 1)
    - »*Size*: DRAM/SRAM *4-8*,
    - »*Cost/Cycle time*: SRAM/DRAM *8-16*



## DRAM



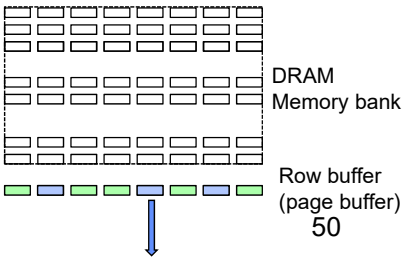

- SDRAM = DRAM with a clocked interface
- DDR SDRAM = double data rate, transfer data at both clock edges
  - DDR2 (1.8 V, 266-400 MHz)
  - DDR3 (1.5 V, 800 MHz)
  - DDR4 (1-1.2 V, 1600 MHz)

49

## DRAM operation

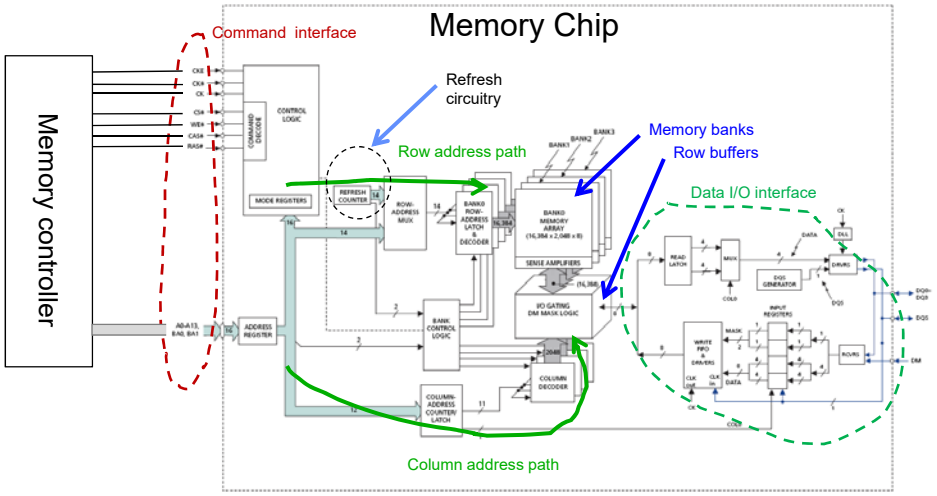
- Each memory bank has a "row buffer", which is non-volatile (SRAM registers)
- To read a byte (a similar process applies for writing):
  - The memory controller sends the row address of the byte
  - The entire row is read into the row buffer (the row is opened)
  - The memory controller sends the column address of the byte
  - The memory returns the byte to the controller (from the row buffer)
  - The memory controller sends a Pre-charge signal (close the open row)

- **Closed row/page policy**: close the row after you read the data
- **Open row/page policy**: close the row only when you want to open a new row – useful if you will read again from the same row.

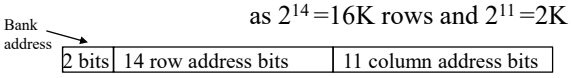



50

# DRAM internals (not in textbook)




EX: a 128 Mbytes chip ($2^{27}$) → 4 banks, each having 32 Mbytes organized as $2^{14}$ =16K rows and $2^{11}$ =2K columns




51

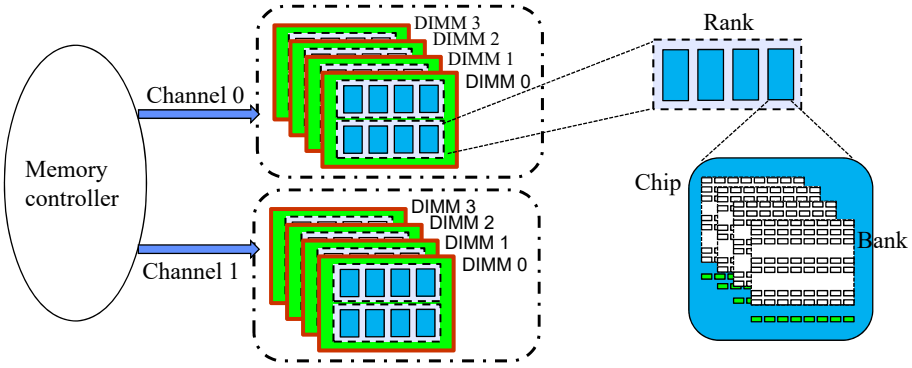
# DRAM Timing Parameters (An example)

- $t_{RAC}$: minimum time from RAS line falling to the valid data output.
  - Quoted as the speed of a DRAM when buy
- $t_{RC}$: minimum time from start of one row access to start of the next.
  - Roughly twice as long as $t_{RAC}$
- $t_{CAC}$: minimum time from CAS line falling to valid data output.
  - Roughly 25% of $t_{RAC}$
- $t_{PC}$: minimum time from start of one column access to start of the next.
  - Roughly 50% larger than $t_{CAC}$

- Examnple: A 60 ns ($t_{RAC}$) DRAM can
  - perform a row access only every 110 ns ($t_{RC}$)
  - perform column access ($t_{CAC}$) in 15 ns, but time between column accesses is at least 35 ns ($t_{PC}$) + any external delay.
- Multiple CAS accesses: sometimes called *open page mode*

52

# DRAM Organization

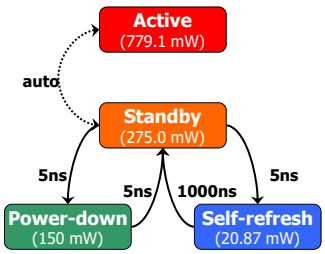


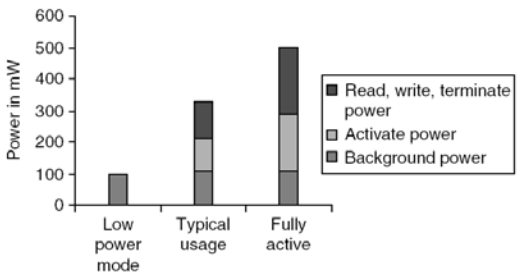

- A memory chip is organized internally as a number of banks (1-8 usually).
- Multiple banks can execute different commands at the same time
- A Rank consists of multiple (parallel) chips contributing to the same transaction. For example, each of 4 chips can provide a byte for a total of 32 data bits (read or written).
- A DIMM (Dual Inline Memory Module) consists of 1 – 4 ranks (2 in the figure) mounted on a single printed-circuit board.
- A Channel supports multiple DIMMS (4 in the above figure)

53

# Reducing memory power consumption

- Lower voltage
- Low power mode (ignores clock, continues to refresh)

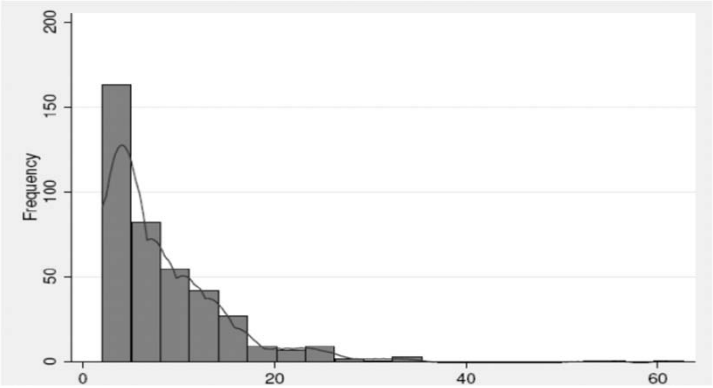



54

## OS assisted Memory Power Management?

- keep a histogram for patterns of bank accesses and idle time distributions.
- Use machine learning techniques to select the optimal “threashold” to switch to a lower power mode.




55

## Example of compiler assisted Memory Power Management?

```
....
Load x
....
Store x
....
Load z
....
Load y
....
Store z
....
Store y
....
....
```

Compiler transformation

```
Load x
Load y
Load z
....
....
....
....
....
....
Store y
....
Store z
```

Code transformations to increase the memory idle time (the time between memory accesses).

56

## Example of compiler assisted Memory Power Management?

Declare A[], B[], C[], D[]
….
Access A
….
Access D
….
Access B
….
Access C
….
Access B
….
….

Memory allocation

A[], B[] C[], D[]

OR

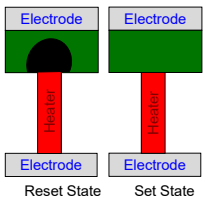
A[], D[] C[], B[]

Algorithms that use the access pattern to allocate memory to banks in a way that maximizes bank idle times

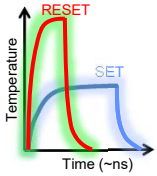
57

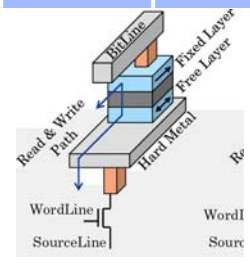
## Alternative memory technology

| | Read Speed | Write Speed | Cell Area | Endurance | Addressability |
|---|---|---|---|---|---|
| DRAM | 20~50ns | 20~50ns | $6F^2$ | $10^{15}$ | Yes |
| SRAM | ~2ns | ~2ns | $146F^2$ | $10^{15}$~$10^{16}$ | Yes |
| NAND Flash | 25us | 500us | $5F^2$ | $10^4$~$10^5$ | No |
| STT-RAM | 2ns | 10ns | $37$~$40F^2$ | $10^{12}$ | Yes |
| PCM | 30~50ns | ~1us | $5$~$8F^2$ | $10^7$~$10^8$ | Yes |
| Domain wall memory | | | | | |




PCM cells




STT_RAM cells

58

## Memory dependability

- Memory is susceptible to cosmic rays
- *Soft errors*: dynamic/transient errors
  - Detected and fixed by error correcting codes (ECC)
- *Hard errors*: permanent errors
  - Use sparse rows to replace defective rows
- Chip-level errors – (Chipkill: a RAID-like error recovery technique)
- *Stuck-at errors*
  - May use data-dependent sparing
- Endurance problems
- Cross-talk (bit-line and word-line)
- Read disturbance
- Write disturbance
- Safely reducing refresh rate of DRAM
  (Example: Refresh-now-and-then)

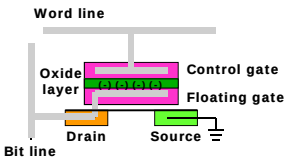
59

## Flash Storage

- Nonvolatile semiconductor storage
  - 100× – 1000× faster than disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)

- Flash bits wears out after 1000's of accesses
  - Not suitable for direct RAM replacement
  - Wear leveling: remap data to less used blocks

NOR flash:
Lower density
Random access
More reliable
Slower erase
Faster random read
Used for instruction memory

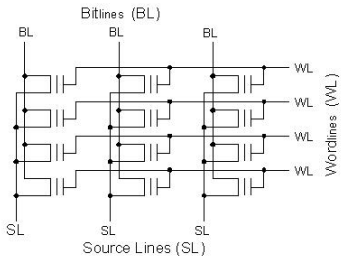



NAND flash:
Higher density
Page access
Less reliable (needs ECC)
Faster erase
Faster streaming read
Used for streamed data

60

# Block erasure

- Blocks of cells are organized as pages (rows)
- A whole block is erased at once
- Cannot erase only part of a block
- A page can be written only once before the whole block is erased




Flash Translation Layer (FTL):

- Maps virtual pages to physical pages
- Remaps a page on a write
  - invalidate the corresponding page
- Erase blocks with many obsolete pages
  - consolidate valid pages into new blocks

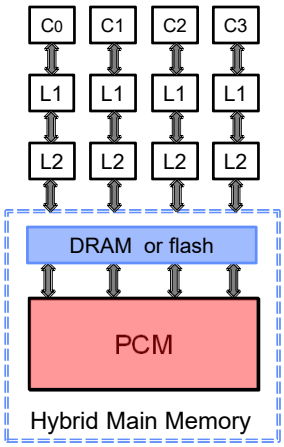
Block

| |
|---|
| Page 1: valid |
| Page 2: obsolete |
| Page 1: obsolete |
| Page 1: valid |
| …….. |
| Page 64: valid |

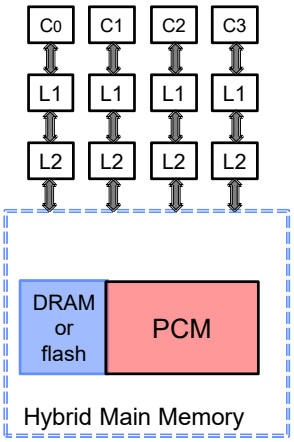
61

# Hybrid main memory

- To deal with asymmetric read and write
  - Write is slower, more time consuming and causes wear.




Vertical organization




Horizontal organization

62