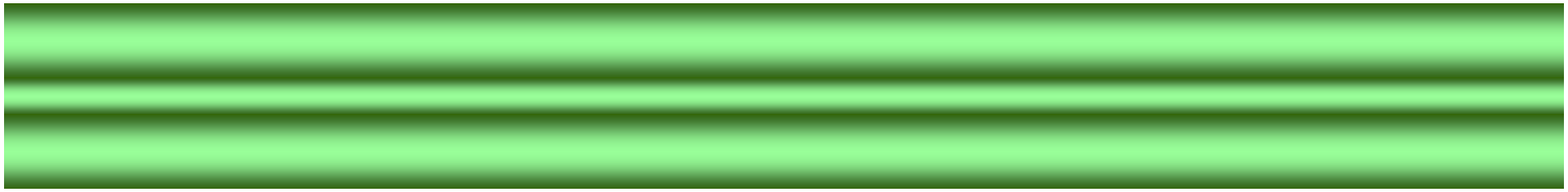
EE414 Embedded Systems

Ch 5. Memory

Part 2/2



Byung Kook Kim
School of Electrical Engineering
Korea Advanced Institute of Science and Technology

Overview

- 6.1 introduction
- 6.2 Memory Write Ability and Storage Performance
- 6.3 Common Memory Types
- 6.4 Composing Memory
- 6.5 Cache
- 6.6 Advanced RAM

5.4 Composing Memory

- Memory size needed often differs from
  size of readily available memories

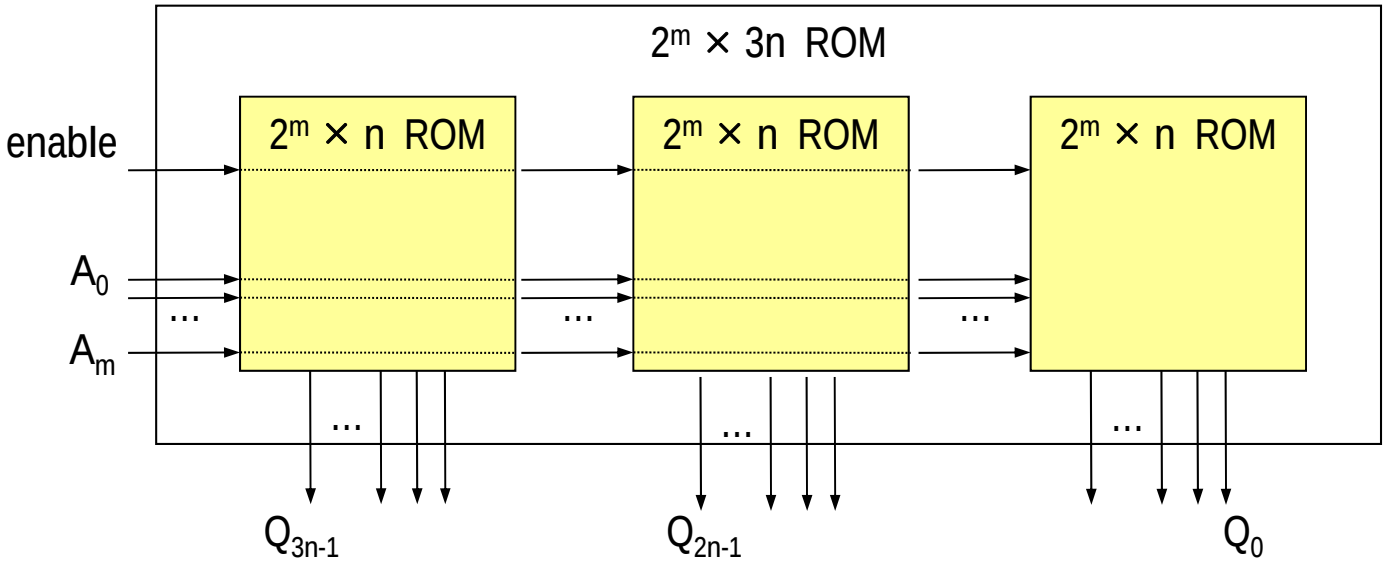
- When available memory is *larger*, simply ignore unneeded high-order address bits and higher data lines

- When available memory is *smaller*, compose several smaller memories into one larger memory:
  - A. To *increase width of words*
  - B. To *increase number of words*
  - C. To *increase number and width of words*

Composing Memory (II)

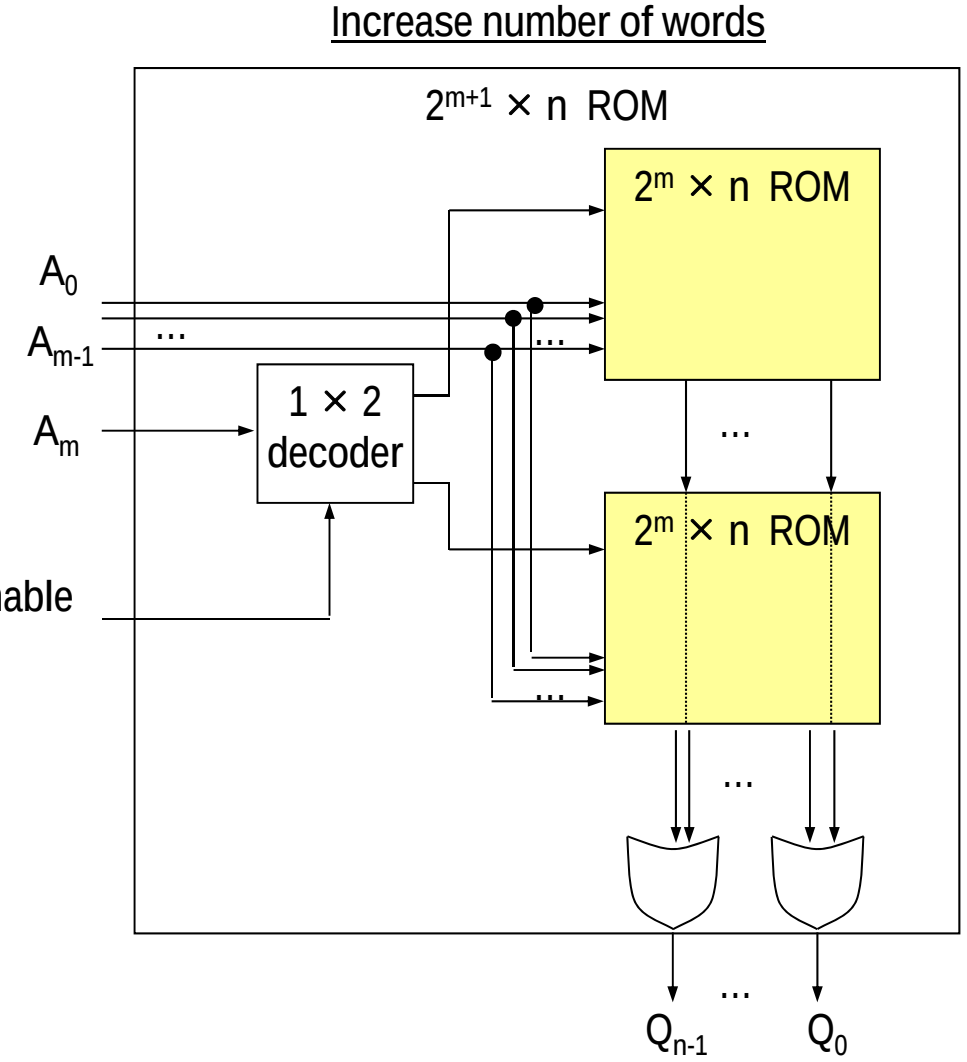
- **A. To *increase width of words***
  - Connect side-by-side to *increase width of words*

Increase width of words



Composing Memory (III)

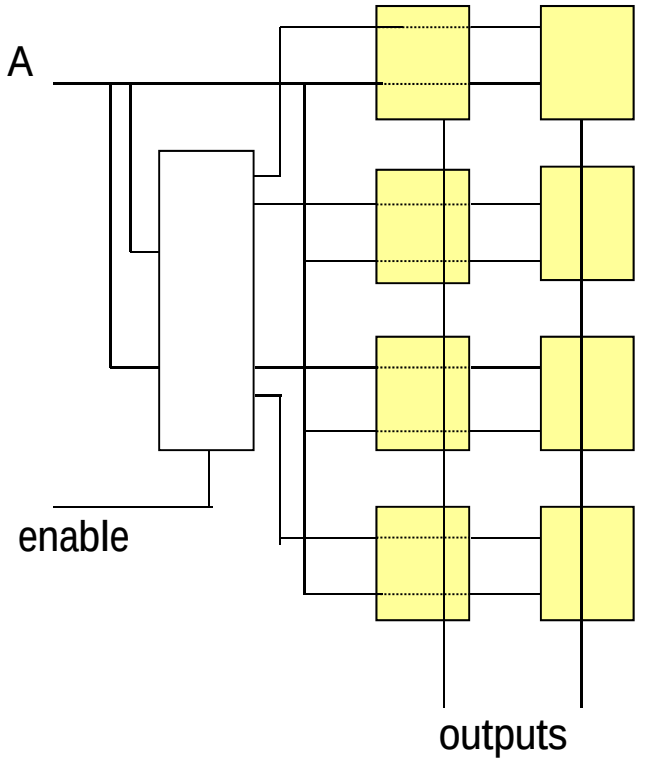
- **B. To *increase number of words***
  - Connect top to bottom to *increase number of words*
  - *Added high-order address line* selects smaller memory containing desired word using an *address decoder*
  - May use odd and even addresses: Ao to address decoder.



Composing Memory (IV)

- **C. To *increase number and width of words***
  - Combine techniques to *increase number and width of words*
  - *Address decoder*

Increase number and width of words



5.5 Cache

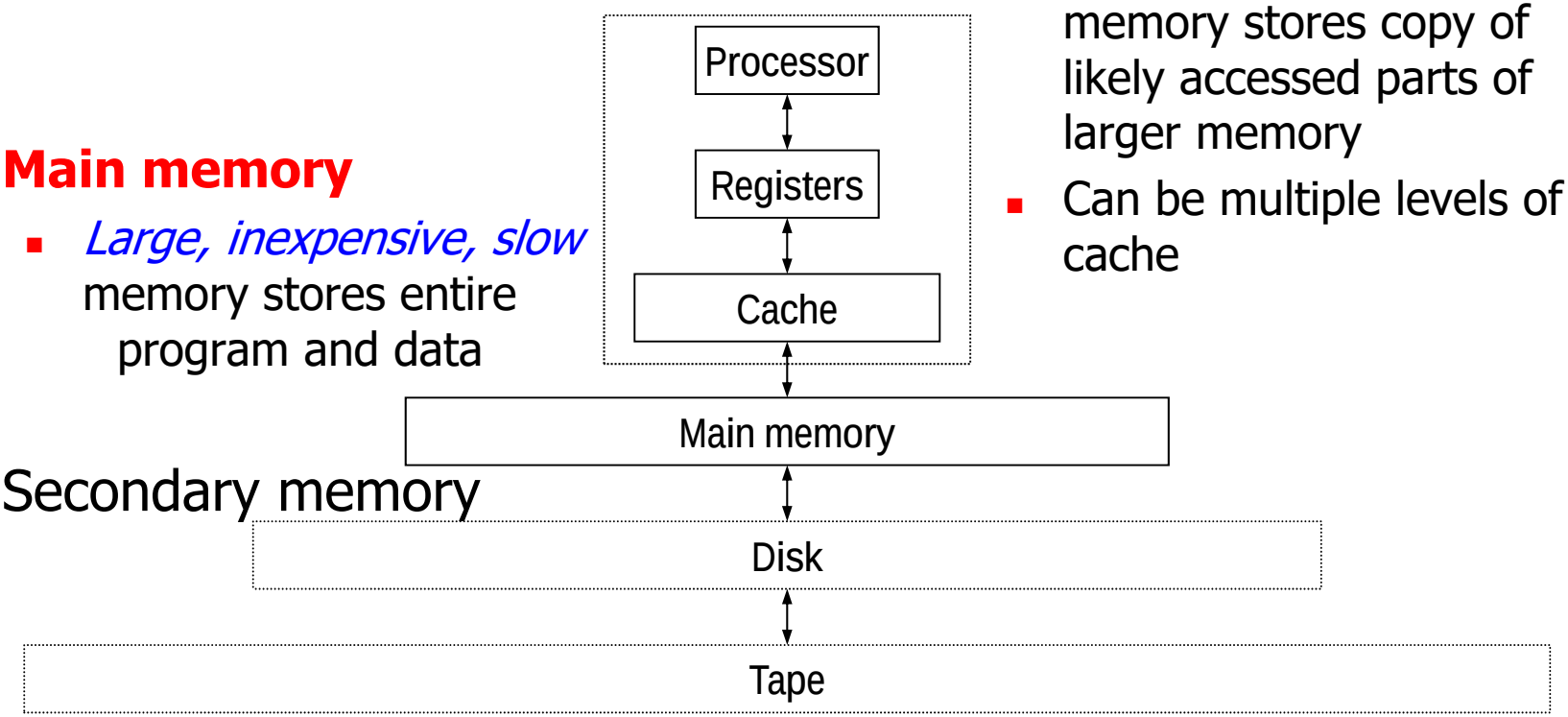
Memory Hierarchy

- Want inexpensive, fast memory

- **Cache**
  - *Small, expensive, fast* memory stores copy of likely accessed parts of larger memory
  - Can be multiple levels of cache

- **Main memory**
  - *Large, inexpensive, slow* memory stores entire program and data

- Secondary memory



Cache

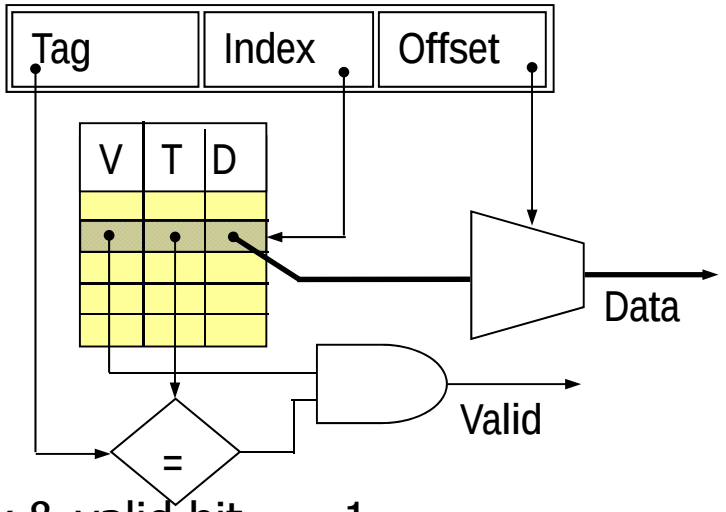
- Usually designed with *SRAM*
  - faster but more expensive than DRAM
- Usually on same chip as processor
  - space limited, so *much smaller* than off-chip main memory
  - *faster access* (1 cycle vs. several cycles for main memory)
- Cache operation:
  - Request for main memory access (read or write)
  - First, check cache for copy
    - **Cache hit**: copy is in cache, quick access
    - **Cache miss**: copy not in cache. *read* memory at the address and possibly its neighbors into cache
- Several cache design choices
  - Cache mapping, replacement policies, and write techniques…

Cache Mapping

- **Cache mapping**
  - Methods of *assigning main memory addresses* (M bits) to the far fewer number of available cache addresses (C bits)
  - Determine *hit or miss*: Whether a particular main memory address's contents are *in the cache.*
- Three basic techniques:
  - A. Direct mapping
  - B. Fully associative mapping
  - C. Set-associative mapping
- Cache line (or cache block)
  - Caches partitioned into indivisible blocks or lines of adjacent memory addresses
  - usually 4 or 8 memory addresses per cache line.
  - C bits = B blocks + O offset.

A. Direct Mapping

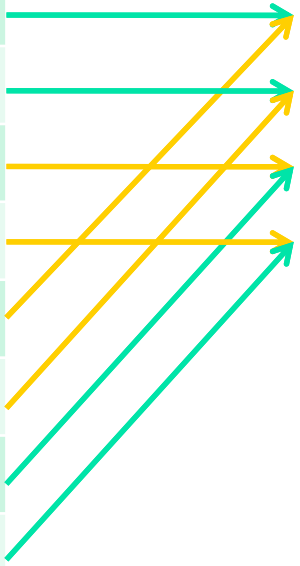
- Main memory address divided into 3 fields
  - Tag M - C
    - Compared with tag stored in cache block indicated by index B
  - Index B
    - Cache (block) address
    - Number of bits determined by cache size: B = $\log_2$(num. cache blocks)
  - Offset O
    - Used to find particular word in cache block
- Cache contains
  - Tag (to be compared)
  - Valid bit
    - Indicates whether data in slot has been loaded from memory.
    - Valid output = 1 if memory tag== cache tag at index & valid bit == 1.
  - Data for each cache block: Get with Offset.



Direct mapping example

- Memory: 64 bytes, Cache: 32 bytes, 8 bytes/cache block. M=6, C=5, O=3
  - 8 memory blocks. 4 cache blocks.
  - Memory address 0x0 to 0x3F = Tag 1 | Index 2 | Offset 3:

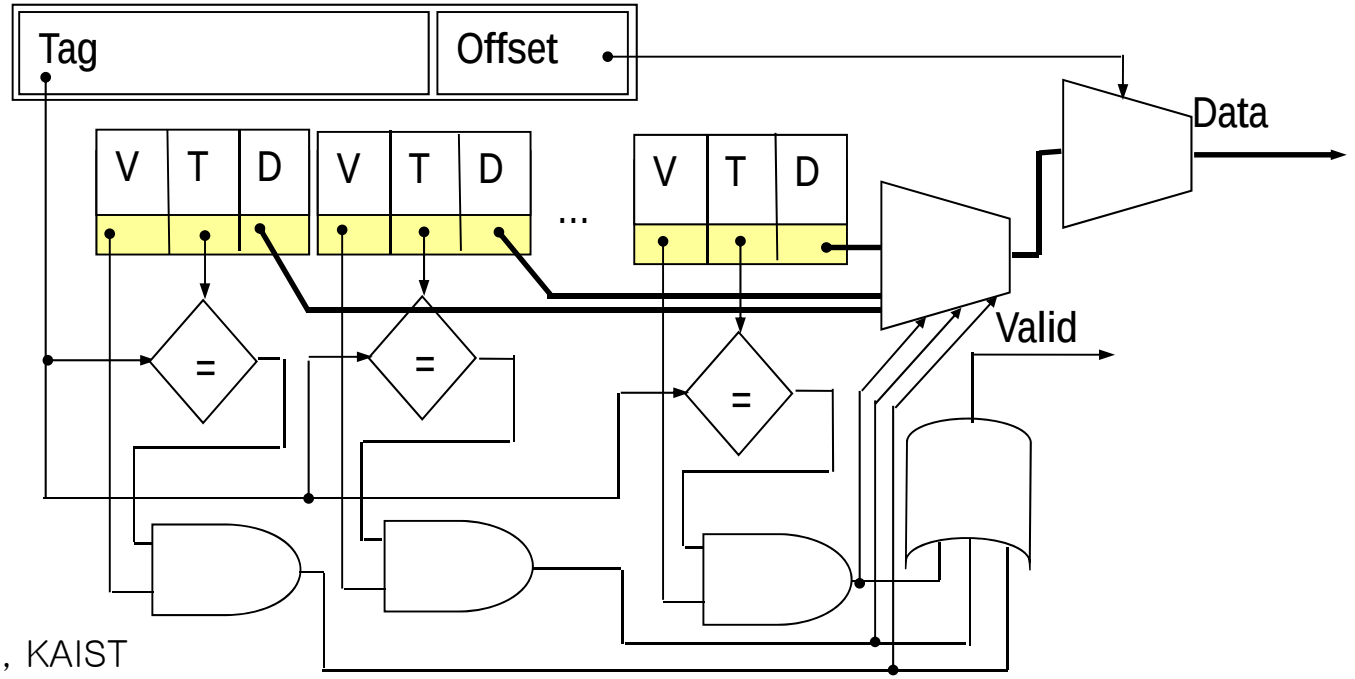
Tag	Index	Mem adr	Mem blocks
0	00	0x00	8 adr x 8 data
0	01	0x08	
0	10	0x10	Pgm 1/5
0	11	0x18	Pgm 2/5
1	00	0x20	Pgm 3/5
1	01	0x28	Pgm 4/5
1	10	0x30	Pgm 5/5 *
1	11	0x38	



Index	Valid	Tag	Cache blocks
00	1	1	Pgm 3/5
01	1	1	Pgm 4/5
10	0	0	Pgm 1/5
11	0	0	Pgm 2/5

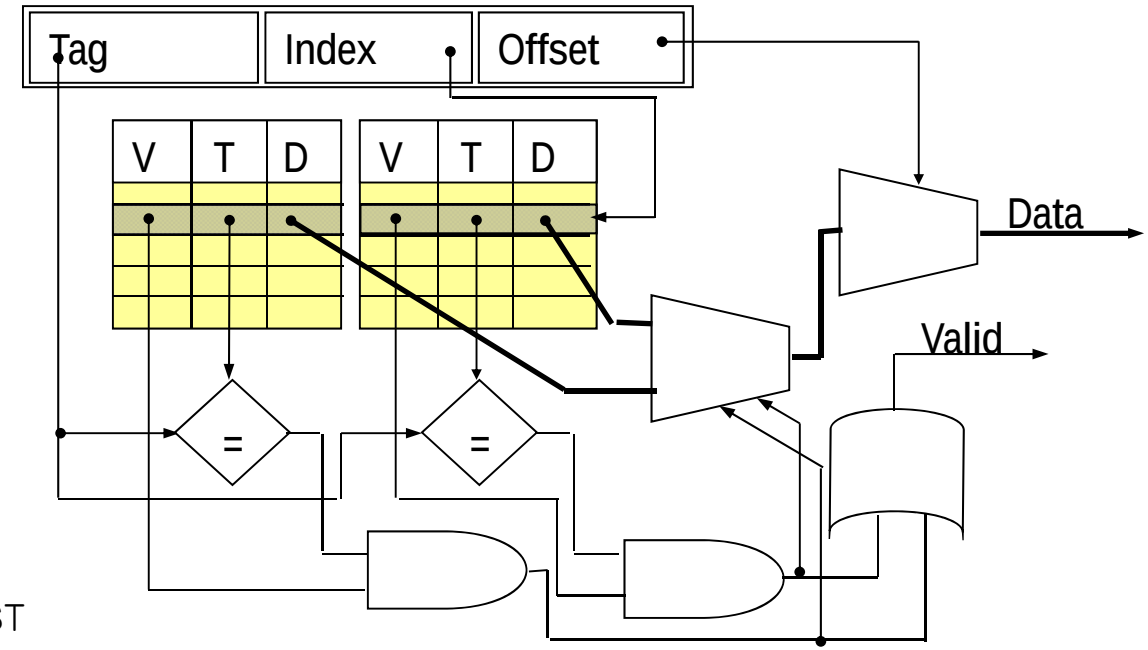
B. Fully Associative Mapping

- *Complete main memory address* (tag, excluding offset, no index: M-O bits) stored as cache tag in each cache block.
- All cache tags are *simultaneously compared* with the desired memory tag: Many comparators.
- Memory block mapped to any cache block.
- Valid outputs are OR-ed.



C. Set-Associative Mapping

- *Compromise* between direct mapping and fully associative mapping
- Index maps memory address to a cache address, but Cache has two or more set of entries: N-way.
- Each entry in cache contains *tag, valid bit, and cache block data.*
- Tags at the same index of each **set** *simultaneously (associatively) compared* .
- Cache with N entries is called N-way set-associative (N=2, 4, 8)
  - 2-way set-associative →



Comparison of Cache Mapping

Cache mapping techniques	Pros	Cons
Direct-mapped	Easy to implement	Numerous misses if several words with the same index are accessed frequently
Fully associative	Fast	The comparison logic is expensive to implement
Set-associative	Can reduce misses. Small number of comparators.	- !

Cache-Replacement Policy

- **Cache-replacement policy**
  - Technique for choosing which block to replace
    - when fully associative cache is full
    - when set-associative cache's line is full
  - Note: Direct mapped cache has no choice!

- Cache Replacement policies
  - **Random**
    - replace block chosen at random
  - **FIFO: first-in-first-out**
    - push block onto queue when accessed
    - choose block to replace by popping queue
  - **LRU: least-recently used**
    - replace block not accessed for longest time

Cache Write Techniques

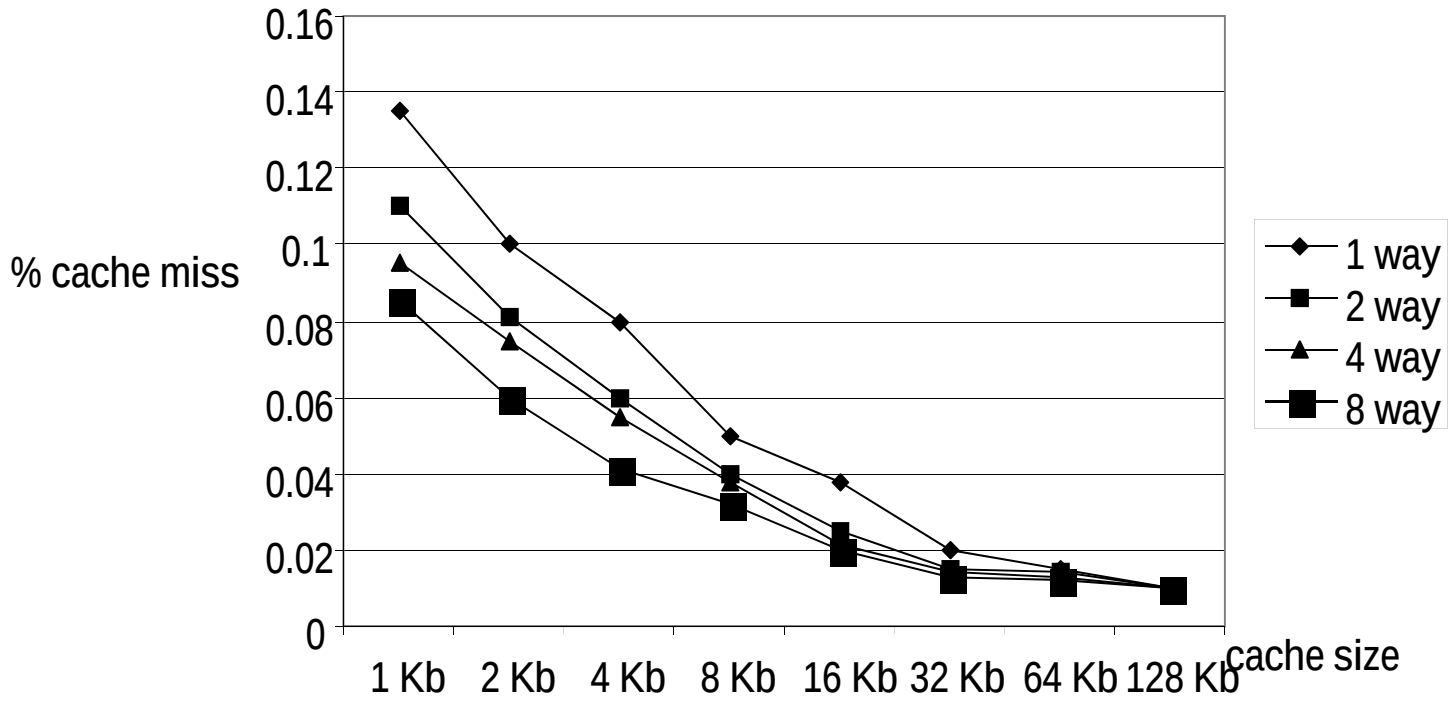
- When written, data cache must update the main memory
  - Note: Instruction cache is read-only.
- **Write-through**
  - Write to main memory whenever cache is written to
  - Pros: Easiest to implement
  - Cons: Processor must wait for slower main memory write
    - Potential for unnecessary writes
- **Write-back**
  - Main memory only written when "dirty" block replaced
  - *Extra dirty bit* for each block set when cache block written to
  - Pros: Reduces number of slow main memory writes

Cache Impact on System Performance

- Most important parameters in terms of performance:
  - Total size of cache
    - Total number of data bytes cache can hold
    - Tag, valid and other house keeping bits not included in total
  - Degree of associativity
  - Data block size

Cache Performance Trade-offs

- Improving cache hit rate without increasing size
  - Increase line size
  - Change set-associativity
- Effect of cache size and associativity →



5.6 Advanced RAM

DRAM

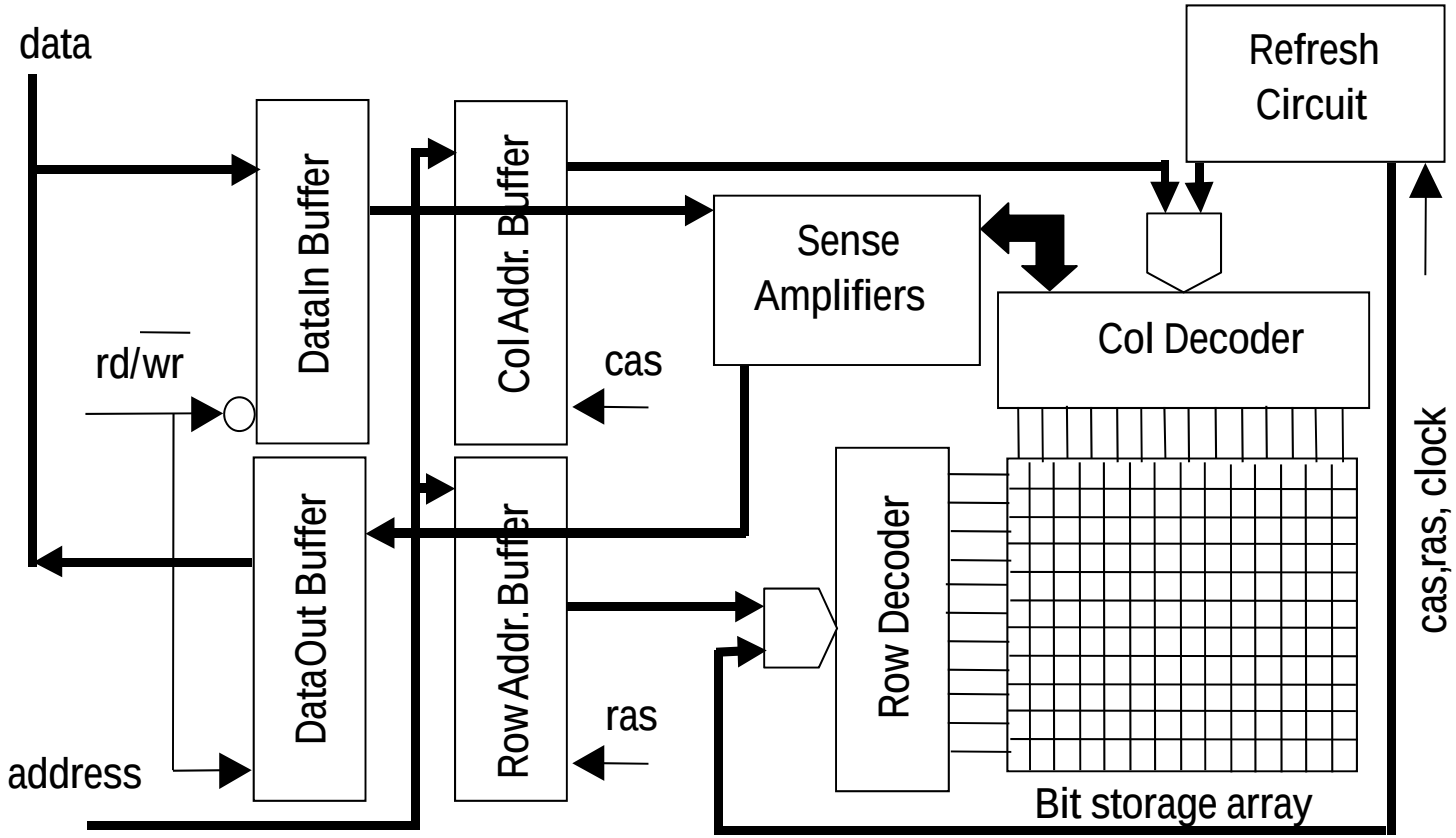
- DRAMs commonly used as main memory in processor based embedded systems
  - Single transistor/capacitor per bit
  - High capacity, low cost
  - Refresh circuit required!
- Many variations on DRAM…

A. Basic DRAM

- Address bus multiplexed between row and column components
  - DRAM memory address = RA | CA (Row Address | Column Address)
  - Row and column addresses are latched in, sequentially, by strobing *ras* and *cas* signals (RA Sel, CA Sel), respectively.
  - Reduce I/O pin requirements.
- Refresh circuitry can be external or internal to DRAM device
  - Strobes consecutive memory address periodically causing memory content to be refreshed
  - Note. Refresh circuitry disabled during read or write operation.

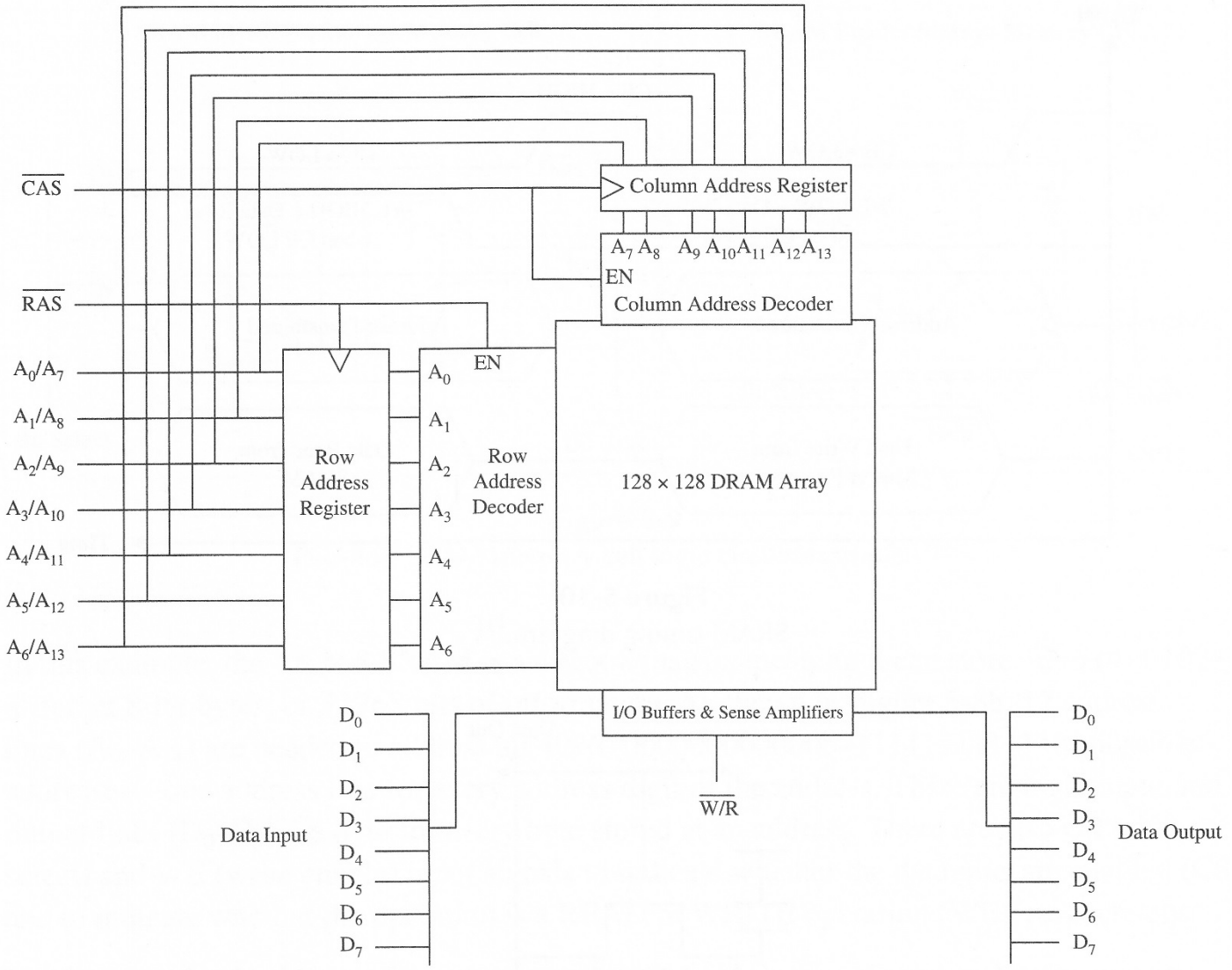
Basic DRAM (II)

- Basic DRAM architecture



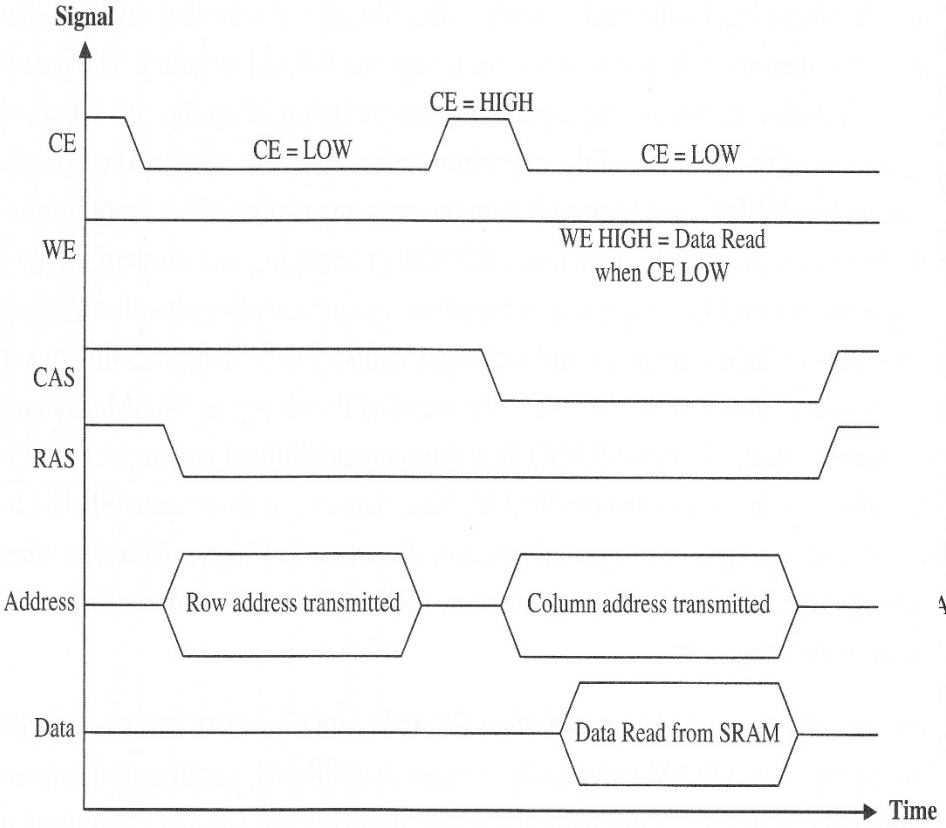
DRAM Logic Circuit

- 16K x 8 DRAM

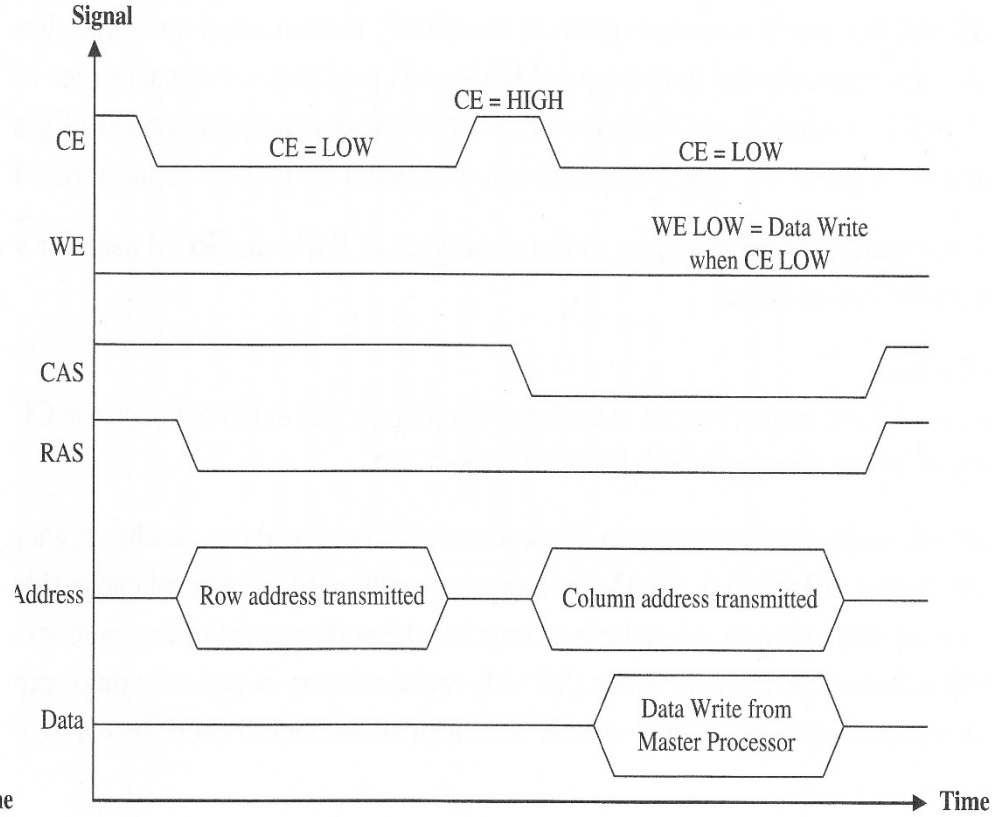


DRAM Timing Diagram

- Read

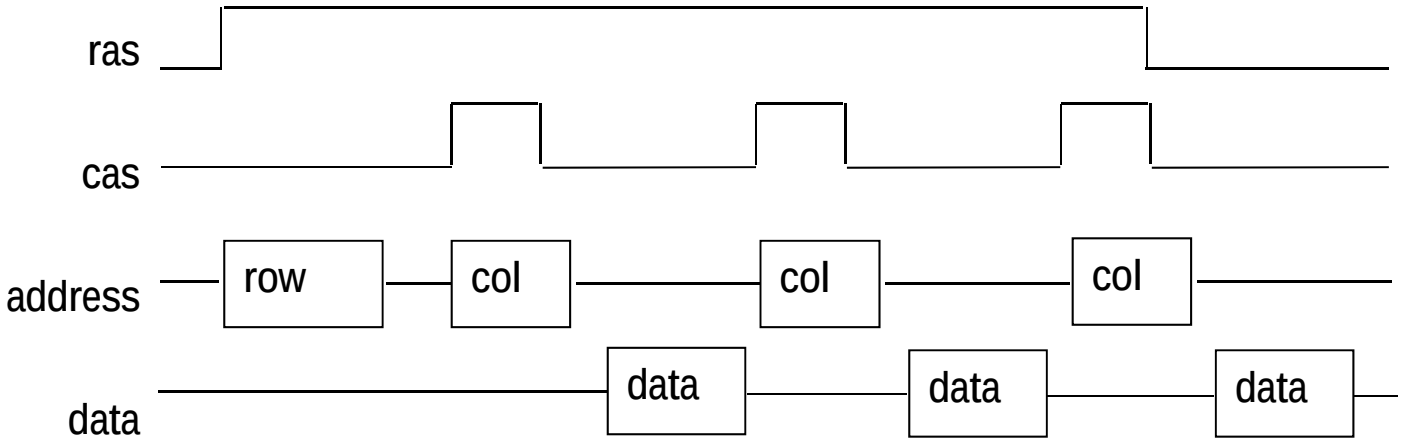


- Write



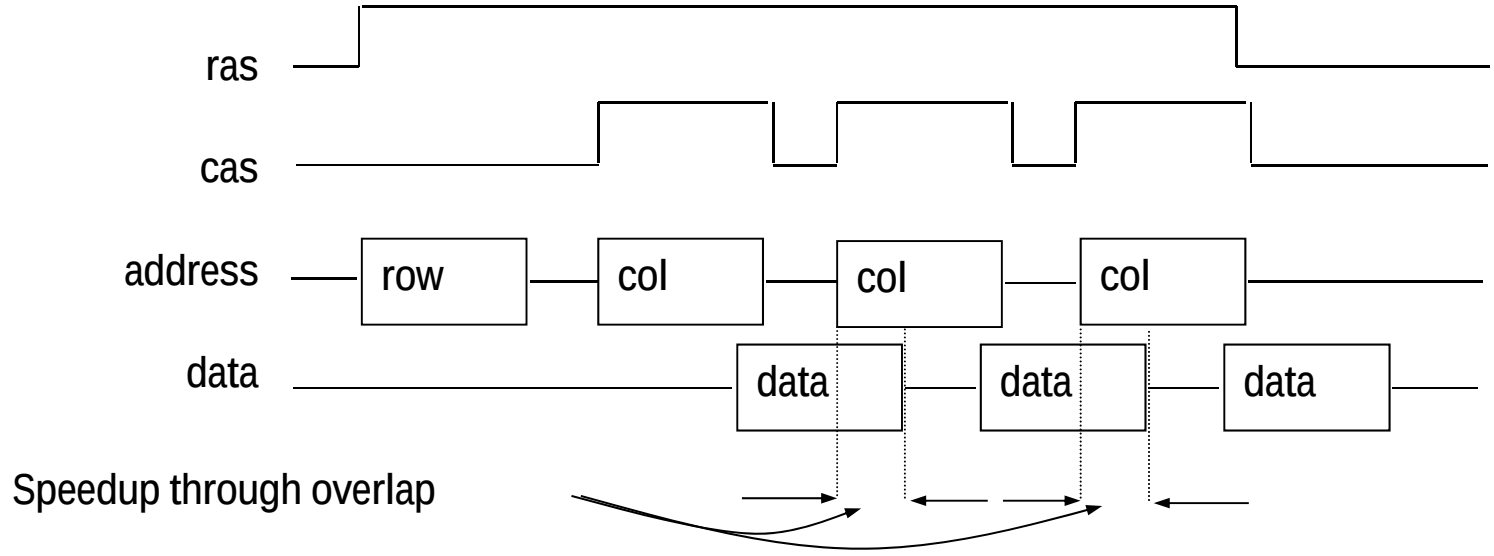
B. Fast Page Mode DRAM (FPM DRAM)

- An improvement on DRAM architecture
  - Each row of memory bit array is viewed as a page
  - A page contains multiple words
  - Individual words addressed by column address only!
- Timing diagram →
  - Row (page) address sent
  - 3 words read consecutively by sending column address for each
- Extra cycle eliminated on each read/write of words from the same page.



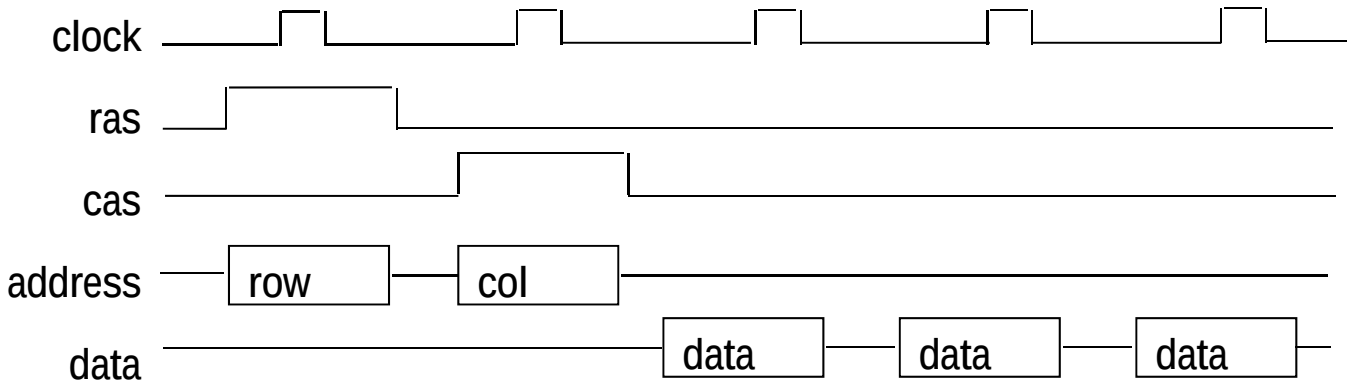
C. Extended Data Out DRAM (EDO DRAM)

- Improvement of FPM DRAM
  - Extra latch before output buffer
  - Allows strobing of *cas* before data read operation completed
- Reduces read/write latency by additional cycle



D. (S)ynchronous and Enhanced Synchronous (ES) DRAM

- SDRAM latches data on active edge of clock
  - Eliminates time to detect *ras/cas* and *rd/wr* signals
  - A counter is initialized to column address then incremented on active edge of clock to access consecutive memory locations
- ESDRAM improves SDRAM
  - Added buffers enable overlapping of column addressing
  - Faster clocking and lower read/write latency possible



E. Rambus DRAM (RDRAM)

- More of a bus interface architecture than DRAM architecture
- Data is latched on both rising and falling edge of clock
- Broken into 4 banks each with own row decoder
  - can have 4 pages open at a time
- Capable of very high throughput
  - 600 M cycles possible.

DIMM [Wikipedia]

- A **DIMM** or **dual in-line memory module** comprises a series of dynamic random-access memory integrated circuits. These modules are mounted on a printed circuit board and designed for use in personal computers, workstations and servers.
- DIMMs began to replace SIMMs (single in-line memory modules) as the predominant type of memory module as Intel P5-based Pentium processors began to gain market share.
- DDR, DDR2, DDR3 and DDR4 all have different pin counts, and different notch positions. As of August, 2014, DDR4 SDRAM is a modern emerging type of dynamic random access memory (DRAM) with a high-bandwidth ("double data rate") interface, and has been in use since 2013.

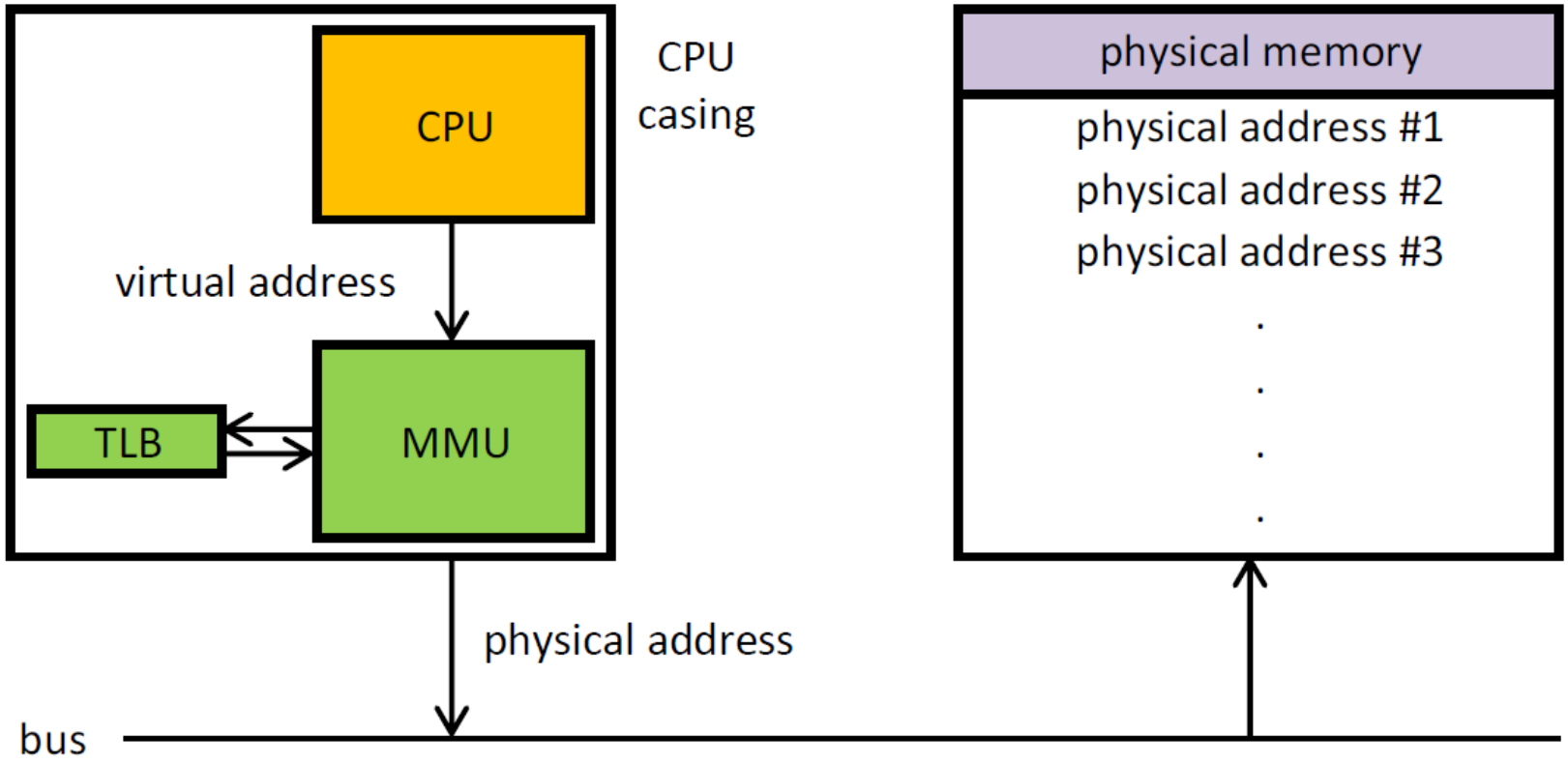
DRAM Integration Problem

- SRAM easily integrated on same chip as processor
- DRAM more difficult
  - Different chip making process between DRAM and conventional logic
  - Goal of conventional logic (IC) designers:
    - Minimize parasitic capacitance to reduce signal propagation delays and power consumption
  - Goal of DRAM designers:
    - Create capacitor cells to retain stored information
  - Integration processes beginning to appear.

Memory Management Unit (MMU)

- Duties of MMU
  - Handles DRAM refresh, bus interface and arbitration
  - Takes care of memory sharing among multiple processors
  - Translates *virtual memory addresses* from processor to *physical memory addresses* of DRAM
- Modern CPUs often come with MMU built-in.
- Single-purpose processors can be used.

Memory Management Unit [Wikipedia]



CPU: Central Processing Unit
MMU: Memory Management Unit
TLB: Translation lookaside buffer

References

- [1] Frank Vahid, “Embedded system design: A unified hardware/software introduction”, John Wiley & Sons, 2002.