University of Southern California

Viterbi School of Engineering

EE352

Computer Organization and Architecture

Memory Overview

References:

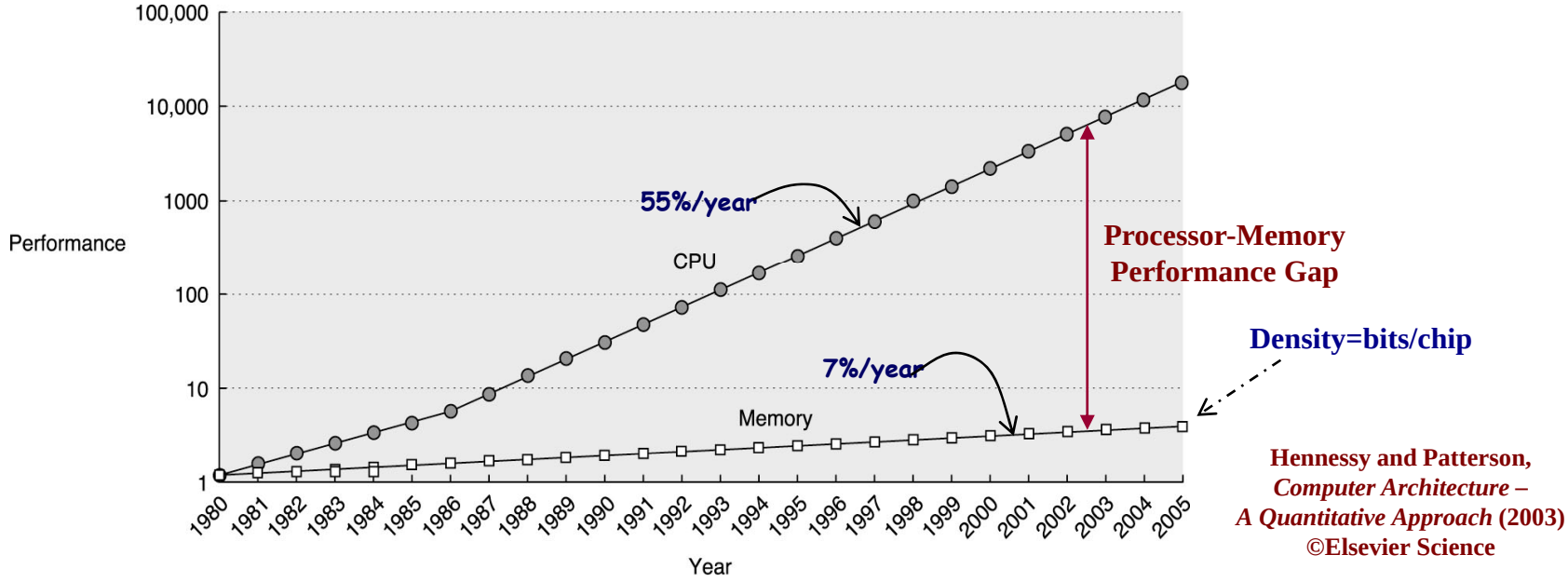
1) **Textbook**
2) **Mark Redekopp's slide series**

Shahin Nazarian

Spring 2010

The Memory Wall

- Problem: The Memory Wall
  - Processor speeds have been increasing much faster than memory access speeds (Memory technology targets density rather than speed)
  - Large memories yield large address decode and access times
  - Main memory is physically located on separate chips and inter-chip signals have much higher propagation delay than intra-chip signals



Hennessy and Patterson, *Computer Architecture – A Quantitative Approach* (2003)
©Elsevier Science

Improving Memory Performance

- Possibilities for improvement
  - Technology
    - We can improve our transistor-level design to create faster RAMs
    - We can integrate memories on the same chip as our processing logic
  - Architectural
    - We can organize memory in a more efficient manner (this is our focus)

[Optional] Introduction

- In general the number of transistors for data storage is much larger than that for logic operations and other purposes
- Maximum realizable data storage capacity doubles every two years
- The area efficiency, i.e., the number of stored data bits per unit area is one of the key design criteria that determine the overall storage capacity and hence the memory cost per bit
- Another key parameter, is the memory access time, i.e., the time required to store and/or retrieve a particular data bit in the memory. It specifies the memory speed
- Static and dynamic power consumption of memory is also a significant factor to be considered during design
- The semiconductor memory is generally classified according the type of data storage and data access, e.g., R/W memory must permit modification (writing) of data bits stored in memory array as well as their retrieval (reading)

[Optional] Introduction (Cont.)

- R/W memory is commonly called Random Access Memory (RAM) due to historical reasons
- RAMs are classified into Dynamic or Static RAMs
- DRAM consists of a capacitor to store binary 1 (high voltage) or 0 (low voltage) and a transistor to access the capacitor. Cell info is degraded due to junction leakage current at the storage node, so cell data must be read and rewritten periodically (refresh operation)
- SRAM consists of a latch, so the cell data is kept as long as power is turned on and refresh operation is not required
- Due to low cost and high density, DRAM is widely used for the main memory in personal and mainframe computers and workstations
- SRAM is mainly used for the cache memory in microprocessors, main frame computers, engineering workstations and memory in hand-held devices due to high speed and low power consumption

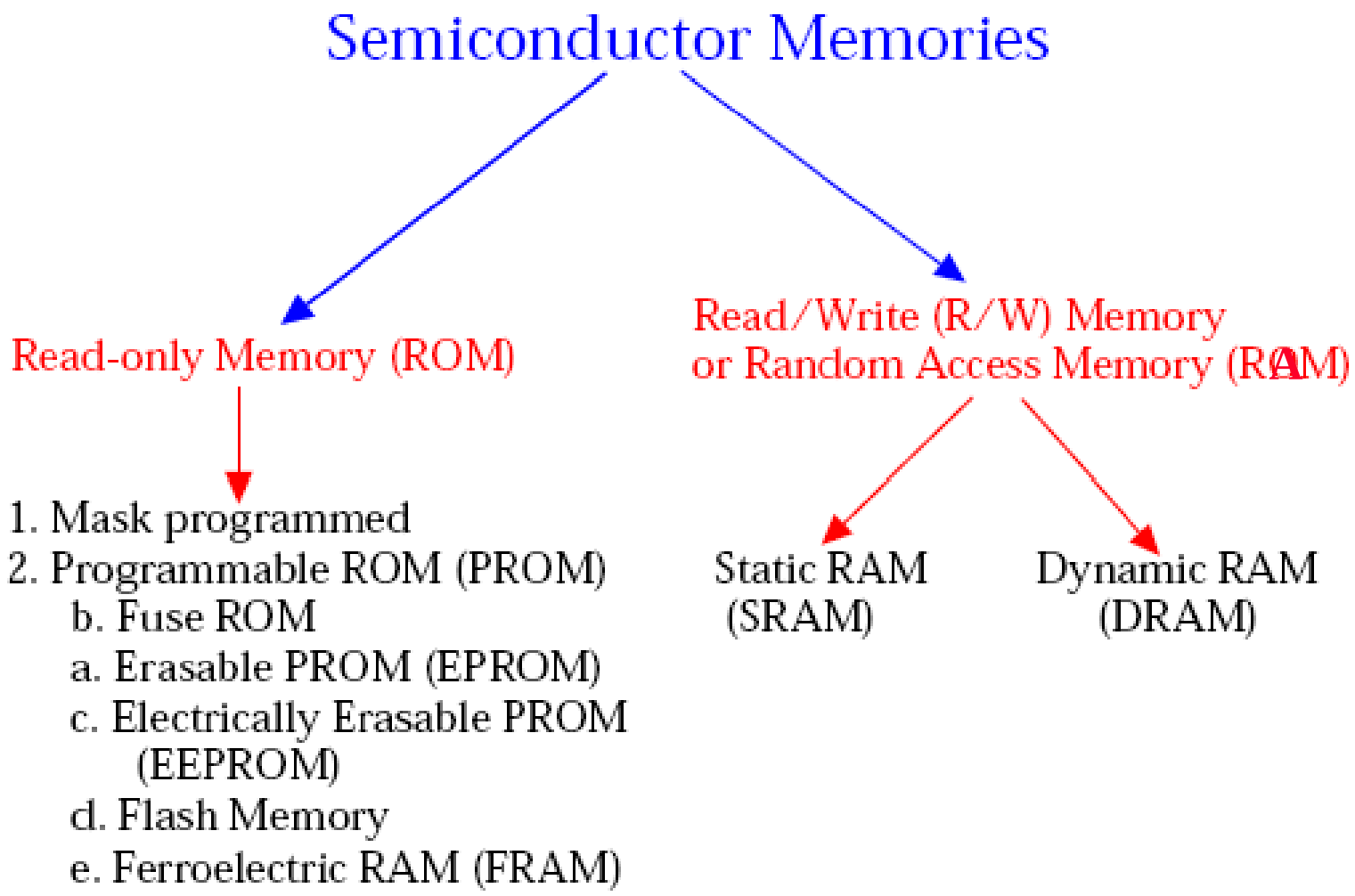
[Optional] Introduction (Cont.)

- ROM allows only retrieval of previously stored data. No modification is permitted. ROMs are nonvolatile memories, i.e., the stored data is not lost even when the power supply is off and refresh operation is not required
- ROM is classified to Mask ROM and PROM
  - In Mask ROM, data is written during chip manufacturing by using a photo mask
  - In PROM the data is written electronically after the chip is fabricated
- PROM is classified to Fuse ROM, EPROM, and EEPROM
  - Data written by blowing the fuse electrically cannot be erased and modified in Fuse ROM
  - Data in EPROM and EEPROM can be rewritten, but the number of subsequent re-writes is limited to 10^4-10^5

[Optional] Introduction (Cont.)

- In EPROM: ultraviolet rays that can penetrate through the crystal glass on the package are used to erase whole data in chip simultaneously
- In EEPROM high electrical voltage is used to erase data in 8 bit units
- Flash memory is similar to EEPROM where data in the block can be erased by using high electrical voltage
- EEPROM drawback: slower write speed, in order of microseconds
- Ferroelectric RAM (FRAM) utilizes the hysteresis characteristics of a ferroelectric capacitor to overcome the slow write operation of other EEPROMs
- ROMs are generally used for permanent (look-up) memory in printers, fax, game machines, and ID cards, due to lower cost than RAM

[Optional] Overview of Semiconductor Memory Types



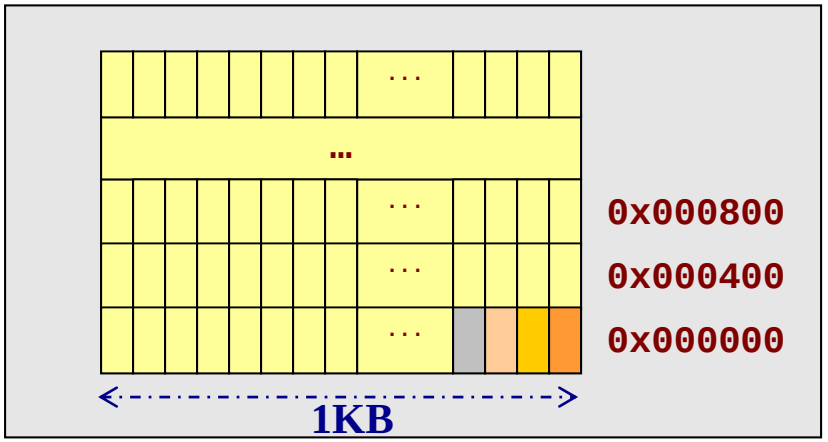
DRAM & SRAM

Memory Array

- Logical View = 1D array of rows (words)
  - Already this is 2D because each word is 32-bits (i.e. 32 columns)
- Physical View = 2D array of rows and columns
  - Each row may contain 1000s of columns (bits) though we have to access at least 8- (and often 16-, 32-, or 64-) bits at a time

C8004DB2	0x000c
89AB97CD	0x0008
AB4982FE	0x0004
123489AB	0x0000

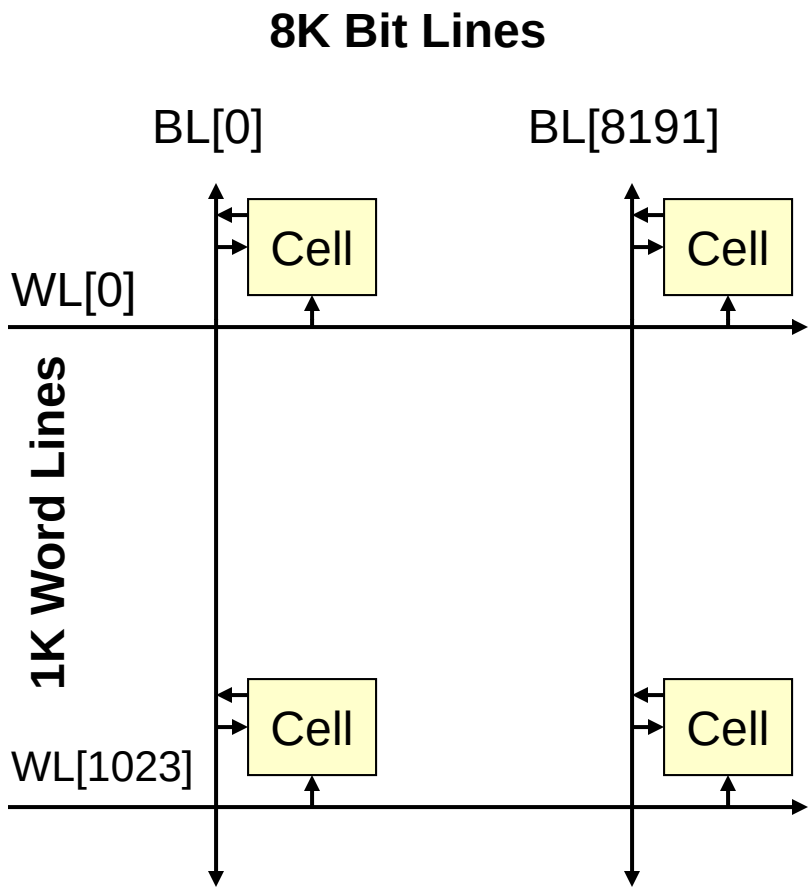
1D Logical View
(Each row is a single word = 32-bits)



2D Physical View
(e.g. a row is 1KB = 8Kb)

1Mx8 Memory Array Layout

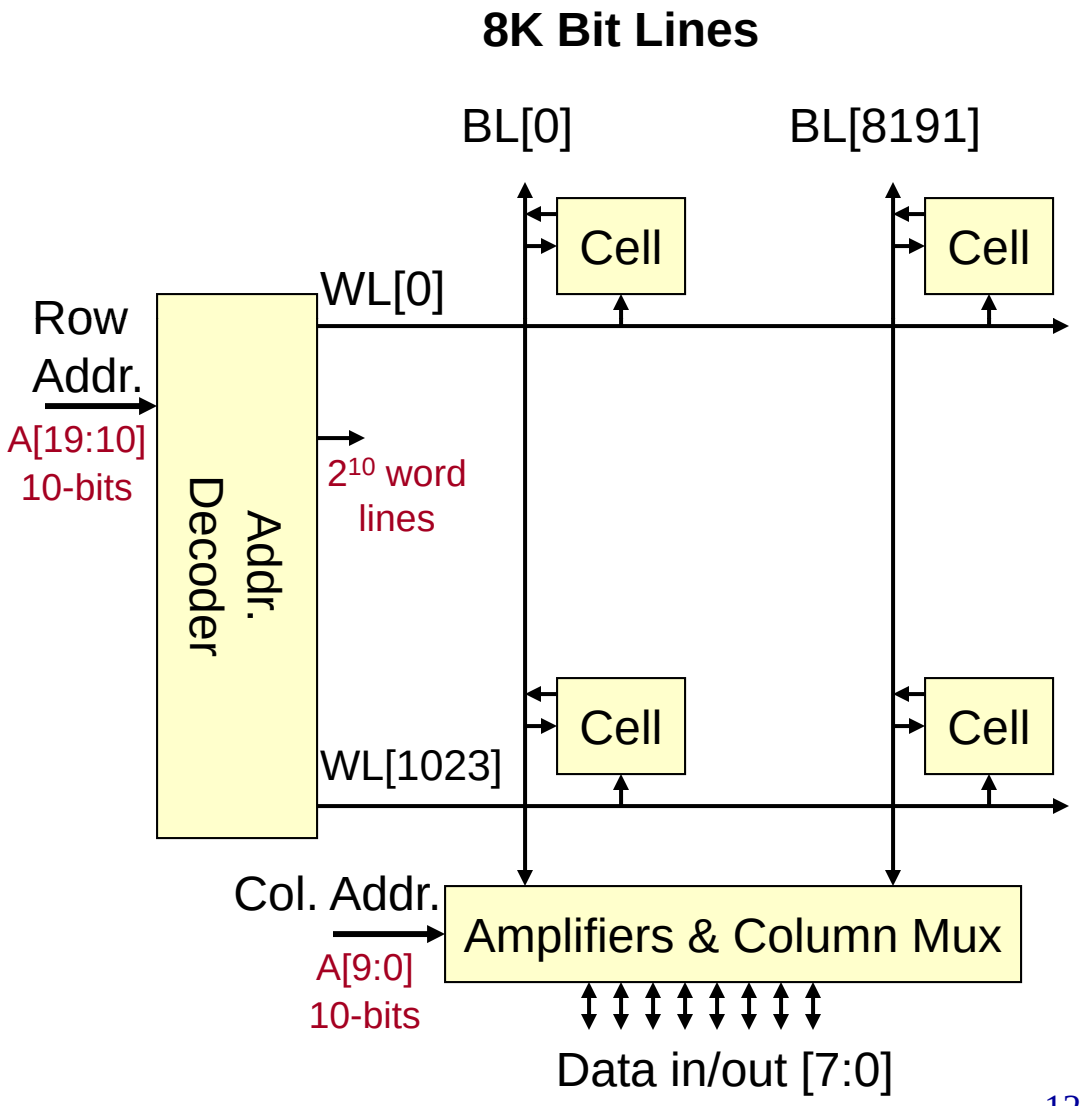
- Start with array of cells that can each store 1-bit
- 1MB = 8Mbits = 2^{23} total cells*
- This can be broken into a 2D array of cells (2^{10} x 2^{13})
- Each row connects to a WL = word line which selects that row
- Each column connects to a BL = bit line for read/write data



* B and b stand for Byte and bit respectively

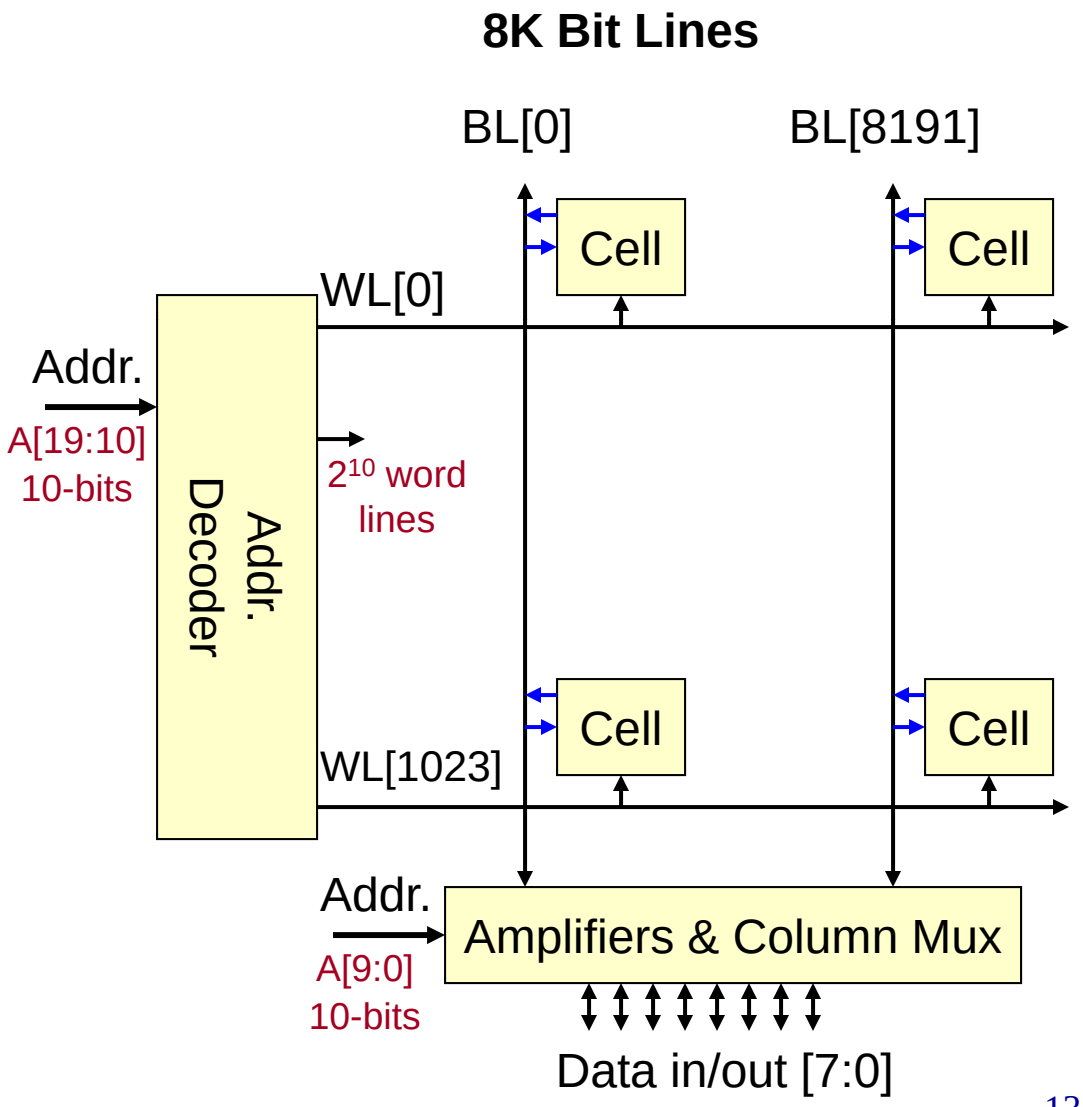
Row and Column Address

- For 1MB we need 20 address bits
- 10 upper address bits can select one of the 1024 rows
- Suppose we always want to read/write an 8-bits (byte), then we will group each set of 8-bits into 1024 byte columns (1024*8=8K)
- 10 lower address bits will select the column



Periphery Logic

- Address decoders selects one row based on the input address number
- Bit lines are bidirectional lines for reading (output) or writing (input)
- Column multiplexers use the address bits to select the right set of 8-bits from the 8K bit lines

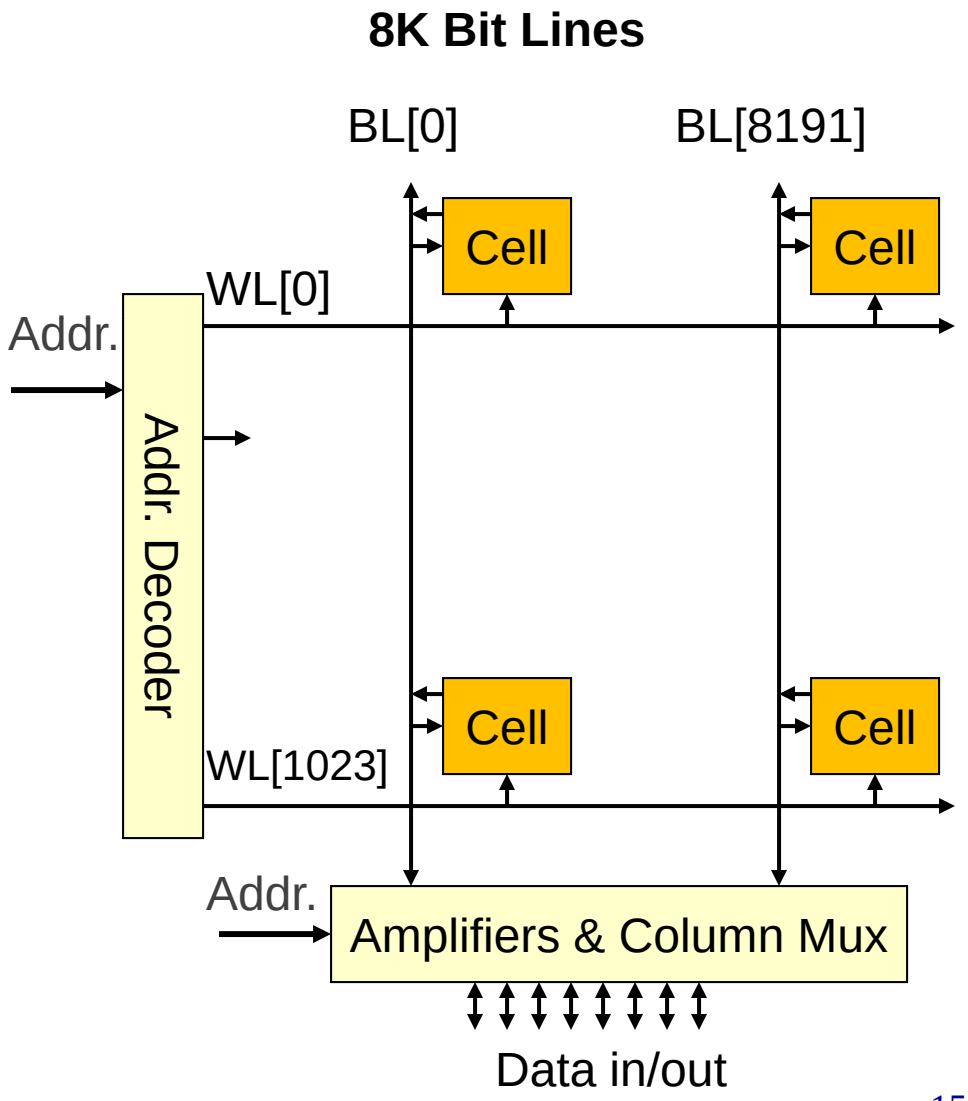


Memory Technologies

- Static RAM (SRAM)
  - Will retain values indefinitely (as long as power is on)
  - Stored bit is actively "remembered" (driven and regenerated by circuitry)
- Dynamic RAM (DRAM)
  - Will lose values if not refreshed periodically
  - Stored bit is passively "remembered" and needs to be regenerated by external circuitry

Memory Layout

- Memory technologies share the same layout but differ in their cell implementation
- SRAM
  - When read, the stored bit is actively driven onto the bit lines
- DRAM
  - When read, the stored bit passively pulls the bit line voltage up or down slightly and needs to be amplified



[Optional] Various Configurations of SRAM cell

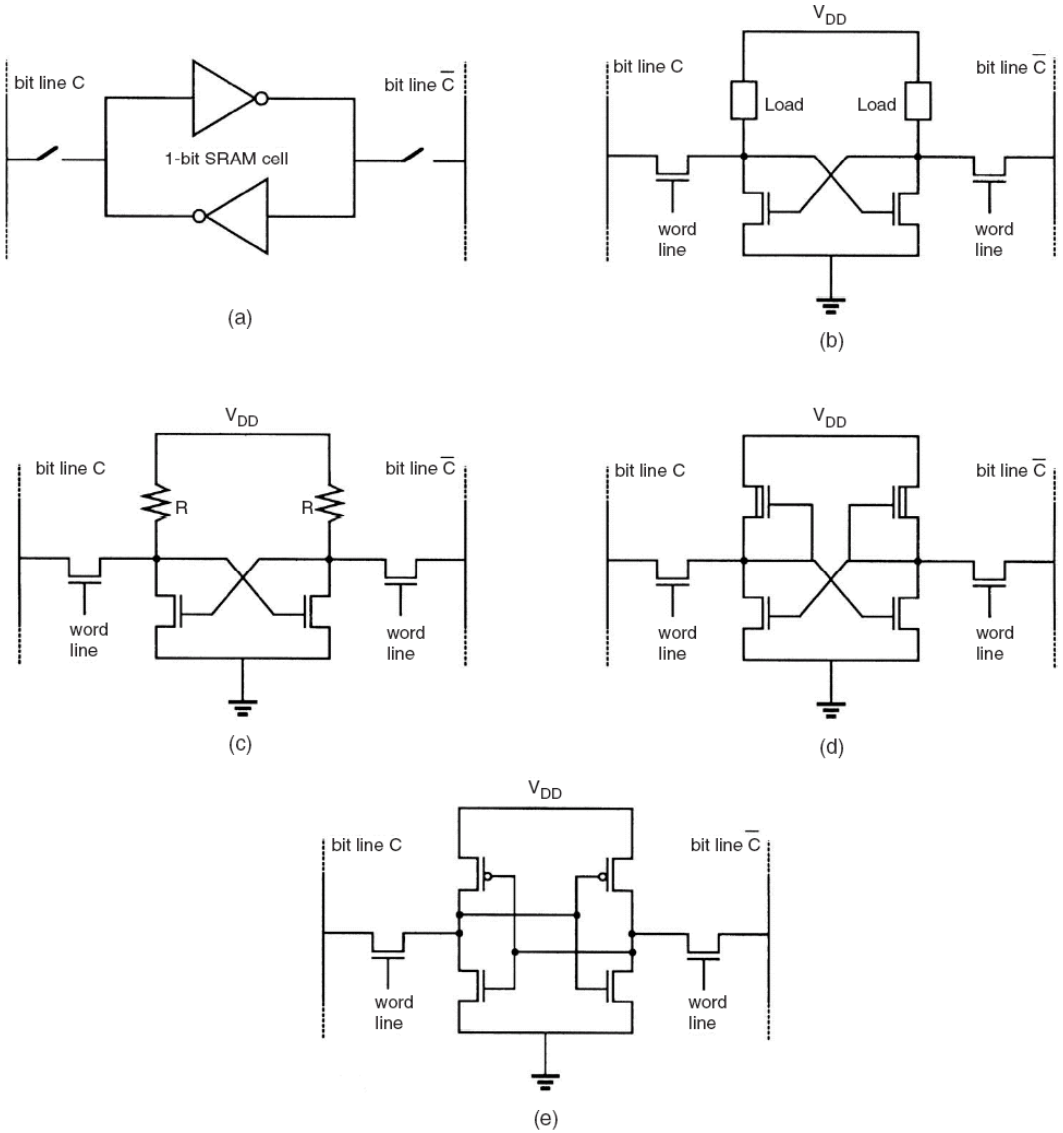
(a) Symbolic representation of the two-inverter latch circuit with access switches

(b) Generic circuit topology of the MOS static RAM (SRAM) cell

(c) Resistive-load SRAM cell

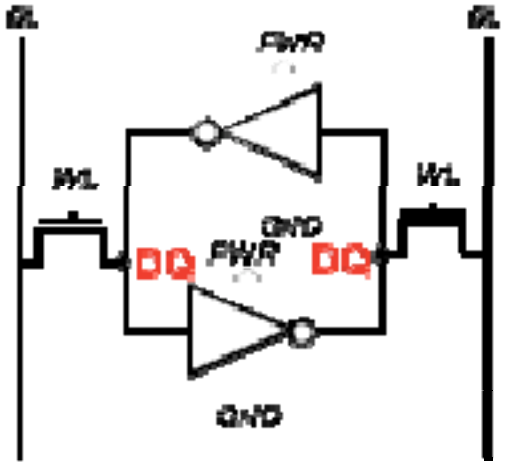
(d) Depletion-load NMOS SRAM cell

(e) Full CMOS SRAM cell

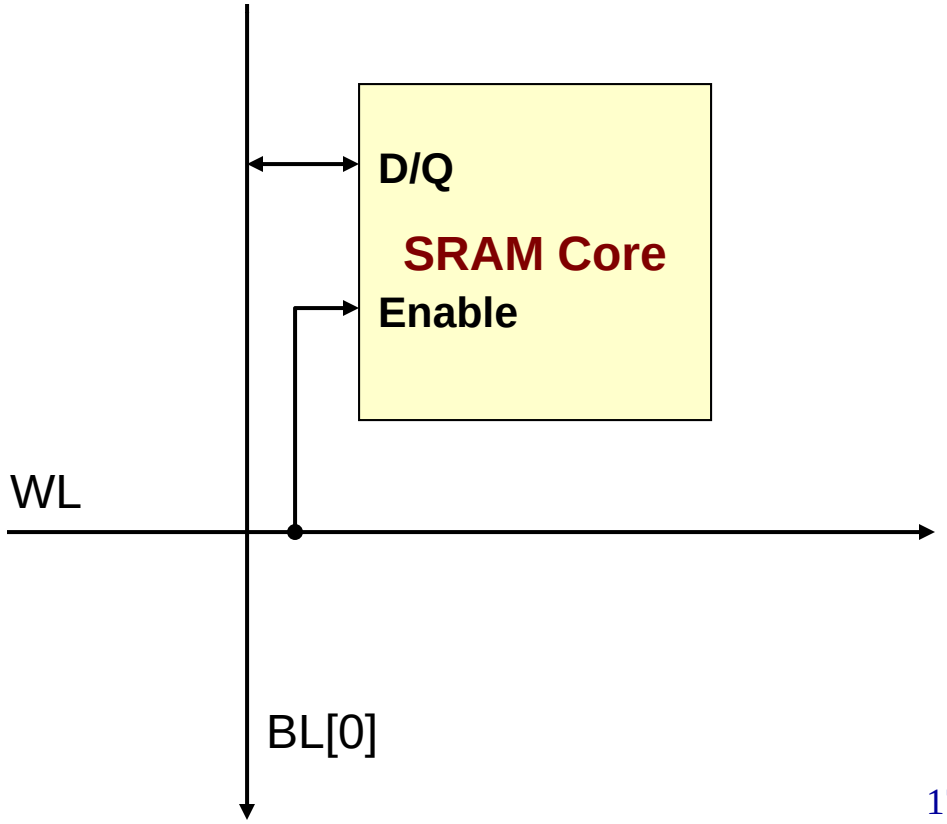


SRAM Cell

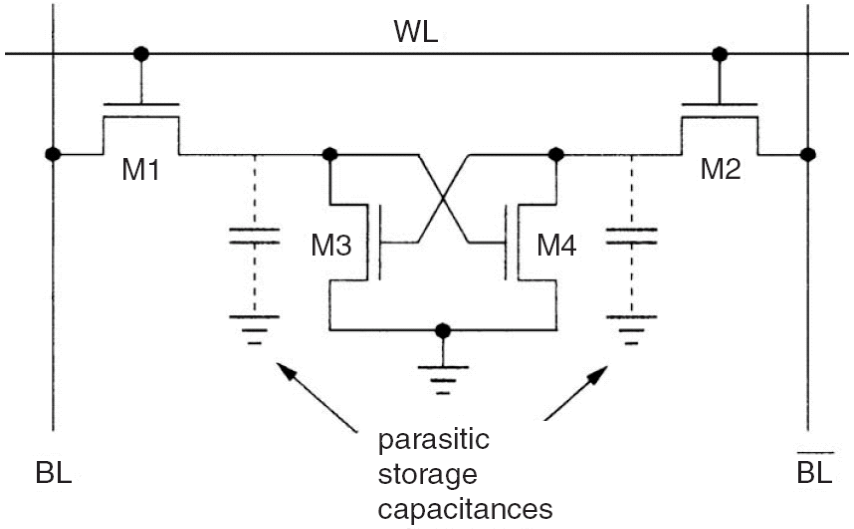
- Each memory cell requires 6 transistors
- Each cell consists of a D-Latch is made from cross connected inverters which have active connections to PWR and GND
  - Thus, the signal is *remembered* and *regenerated* as long as power is supplied



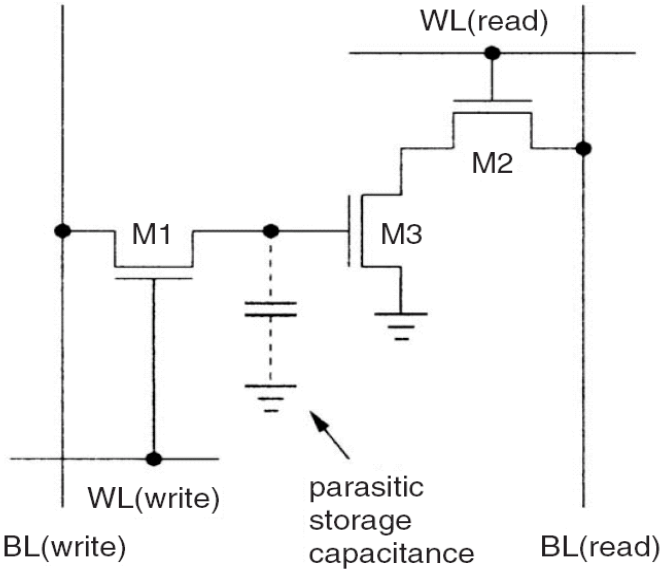
SRAM core implementation



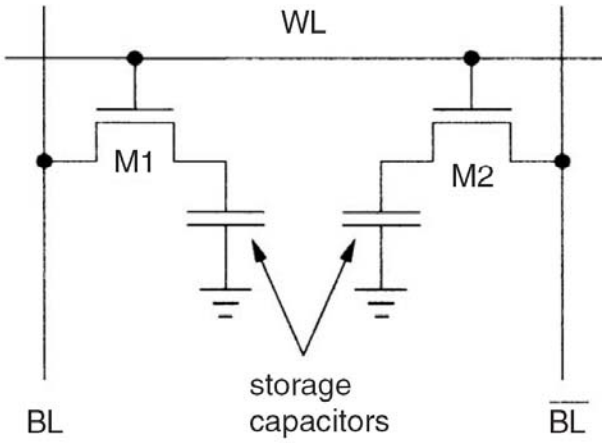
[Optional] Various Configurations of DRAM Cell



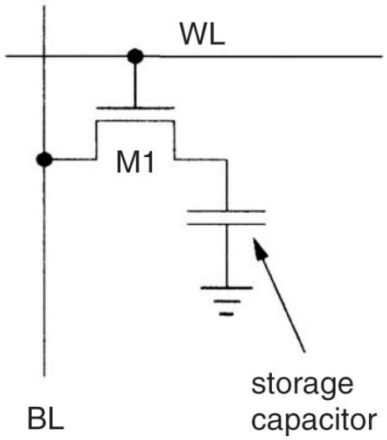
(a)



(b)



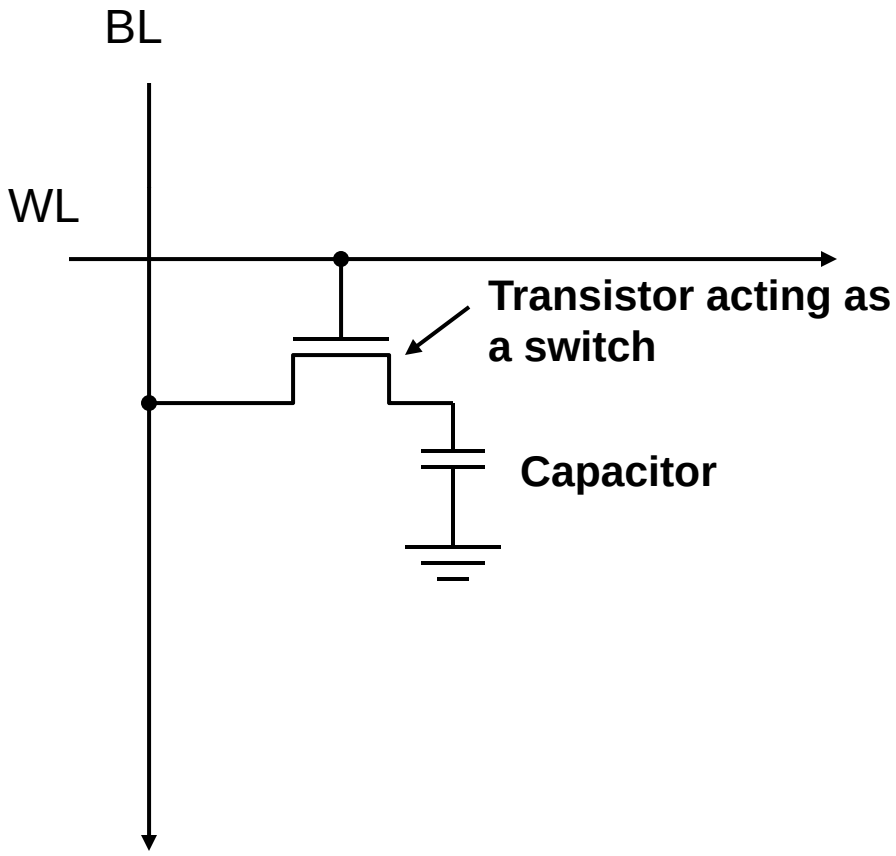
(c)



(d)

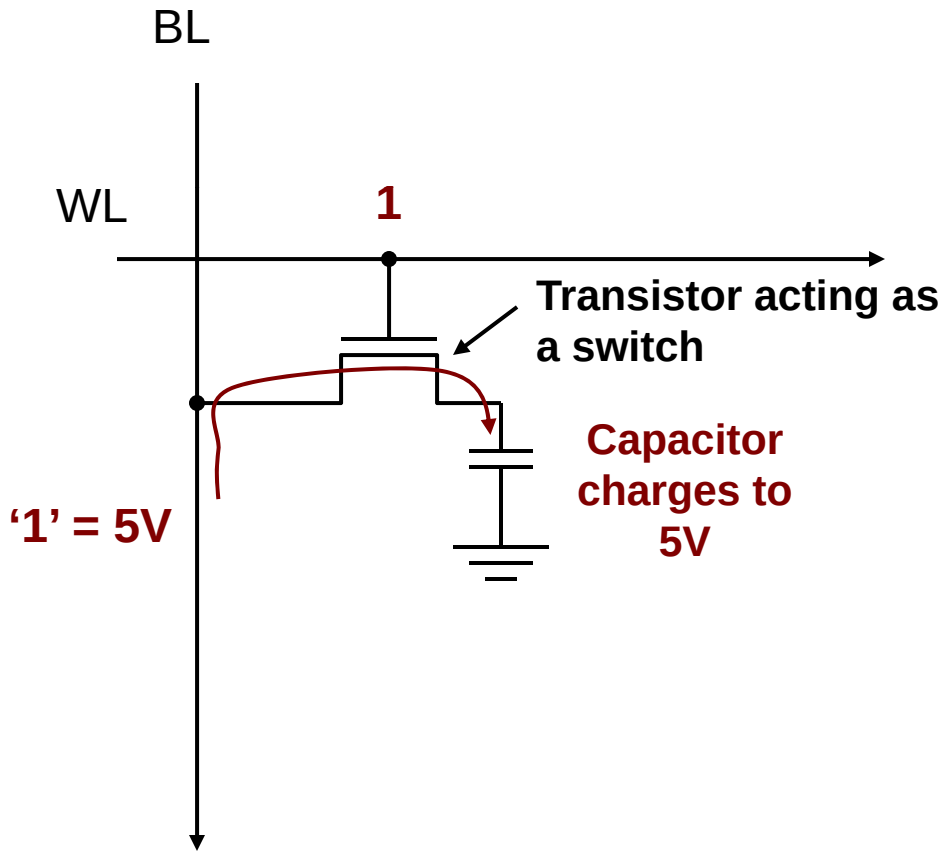
DRAM Cell

- Bit is stored on a capacitor and requires only 1 transistor and a capacitor



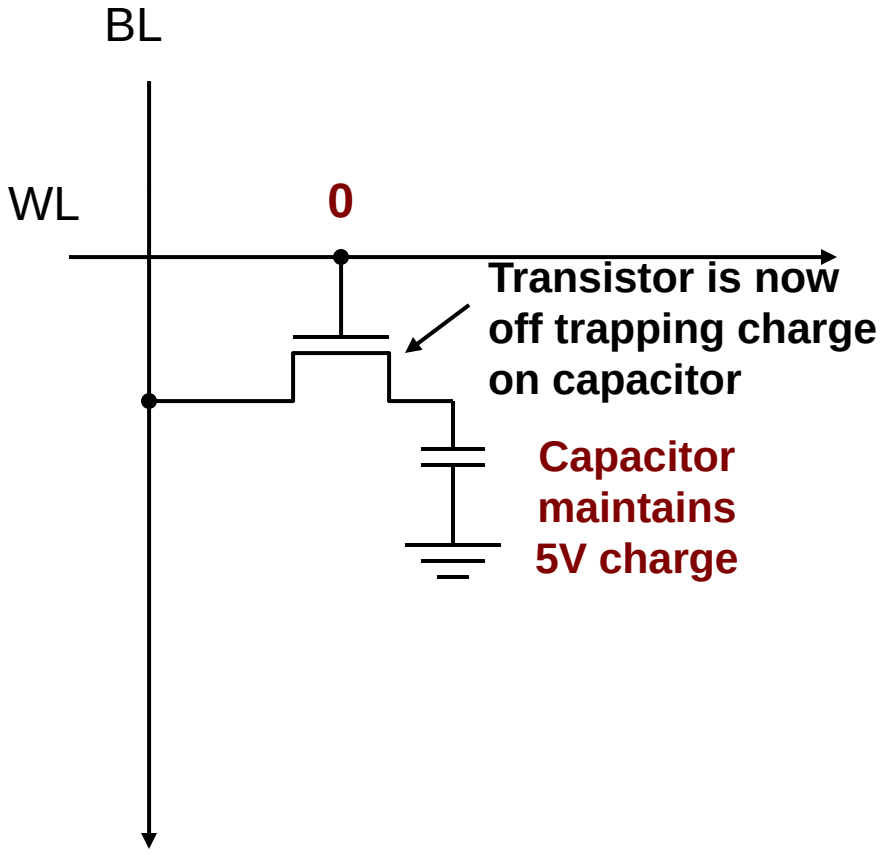
DRAM Cell

- Bit is stored on a capacitor and requires only 1 transistor
- **Write**
  - **WL=1 connects the capacitor to the BL which is driven to 1 or 0 and charges/discharges capacitor based on the BL value**



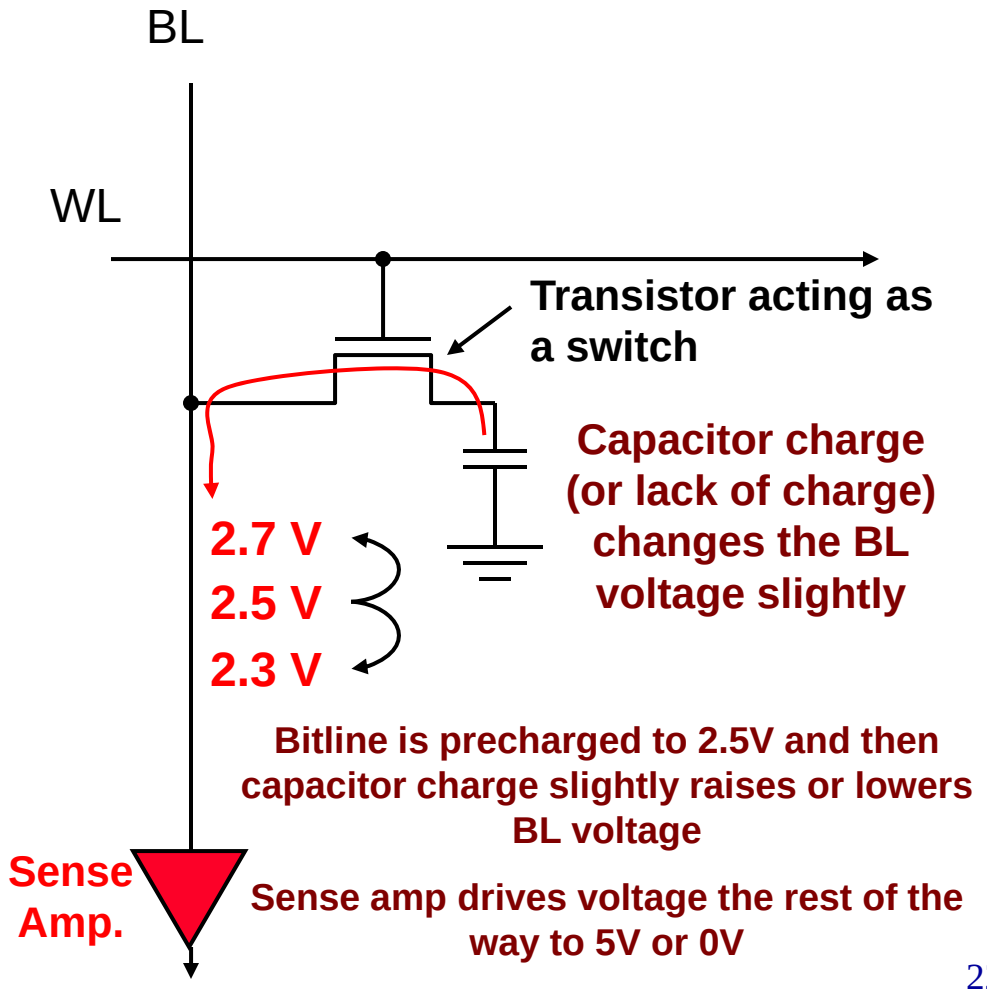
DRAM Cell

- Bit is stored on a capacitor and requires only 1 transistor
- Write
  - WL=1 connects the capacitor to the BL which is driven to 1 or 0 and charges/discharges capacitor based on the BL value
    - With WL=0 transistor is closed and value stored on the capacitor



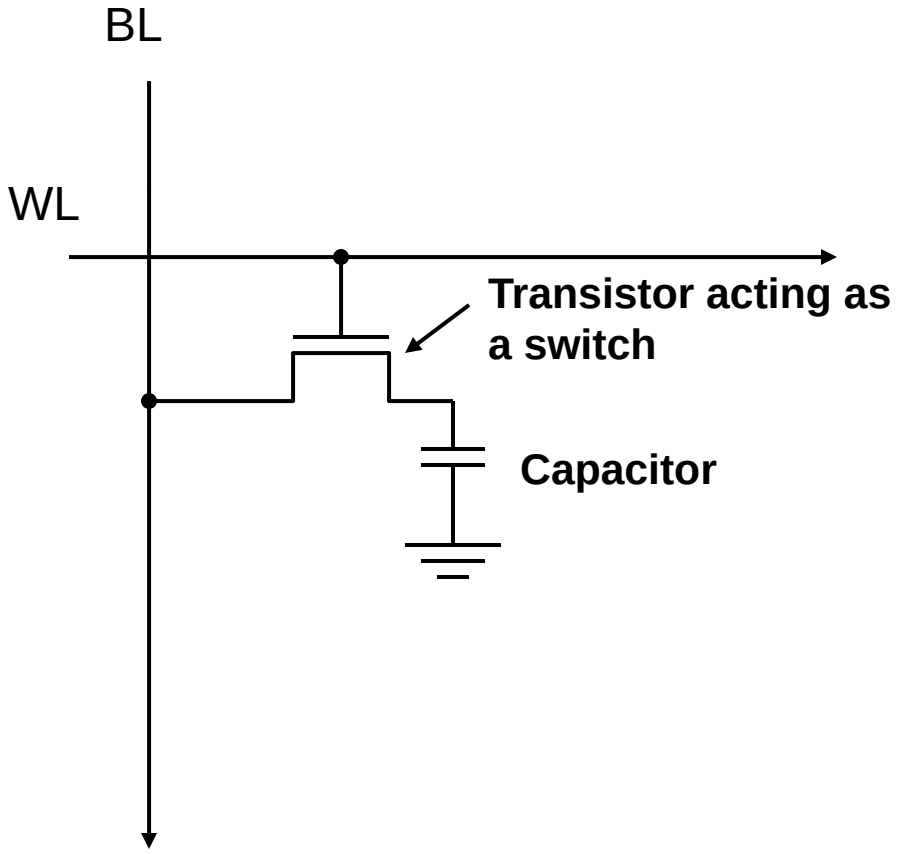
DRAM Cell

- Bit is stored on a capacitor and requires only 1 transistor
- Write
  - WL=1 connects the capacitor to the BL which is driven to 1 or 0 and charges/discharges capacitor based on the BL value
- **Read**
  - **BL is precharged to 2.5V**
  - **WL=1 connects the capacitor to the BL allowing charge on capacitor to change the voltage on the BL**



DRAM Issues

- Destructive Read
  - Charge is lost when capacitor value is read
  - Need to write whatever is read back to the cap
- Leakage Current
  - Charge slowly leaks off the cap
  - Value must be refreshed periodically

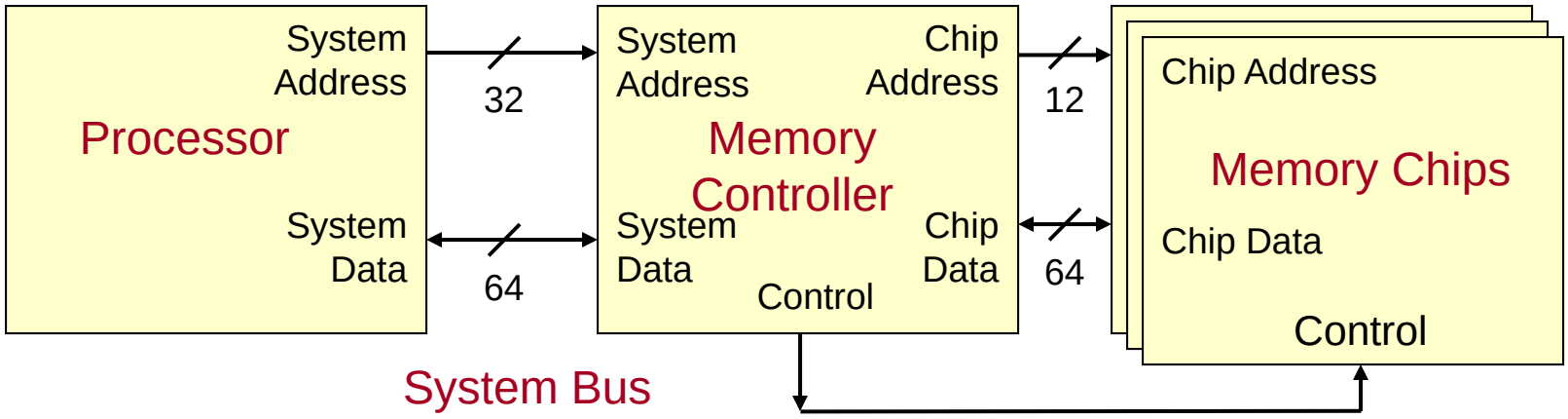


SRAM vs. DRAM Summary

- SRAM
  - Faster because bit lines are actively driven by the D-Latch
  - Faster, simpler interface due to lack of refresh
  - Larger Area for each cell which means less memory per chip
  - Used for cache memory where speed/latency is key
- DRAM
  - Slower because passive value (charge on cap) drives BL
  - Slower due to refresh cycles
  - Small area means much greater density of cells and thus large memories
  - Used for main memory where density is key

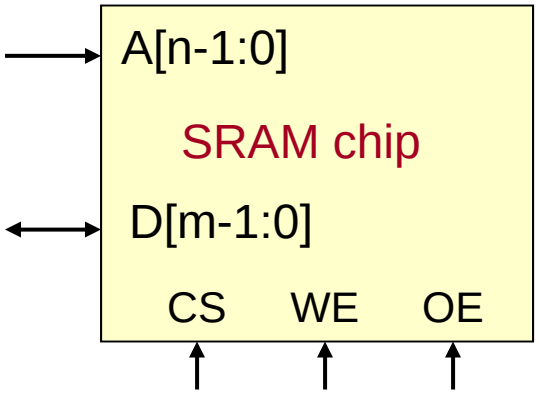
Architectural Block Diagram

- Memory controller interprets system bus transactions into appropriate chip address and control signals
  - System address and data bus sizes do not have to match chip address and data sizes

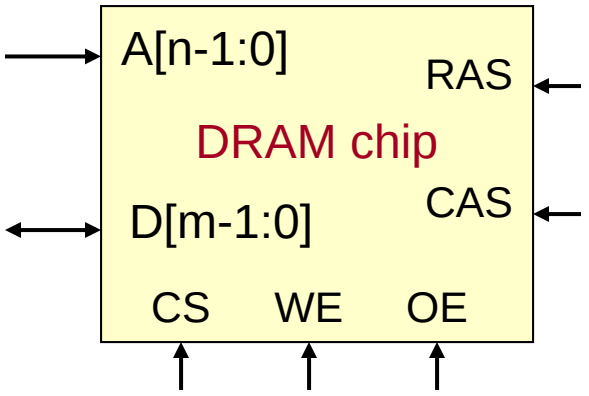


Memory Block Diagrams

- SRAM
  - Bidirectional, shared data bus (only 1 chip can drive bus at a time)
  - CS = Chip select (enables the chip…in the likely case that multiple chips connected to bus)
  - WE / OE = Write / Output Enable (write the data or read data to/from the given address)



- DRAM
  - Bidirectional, shared data bus (only 1 chip can drive bus at a time)
  - CS/WE/OE = Chip select & Write / Output Enable
  - RAS / CAS = Row / Column address strobe. Address broken into two groups of bits for the row and column in the 2D memory array (This allows address bus to be approx. half as wide as SRAM's)

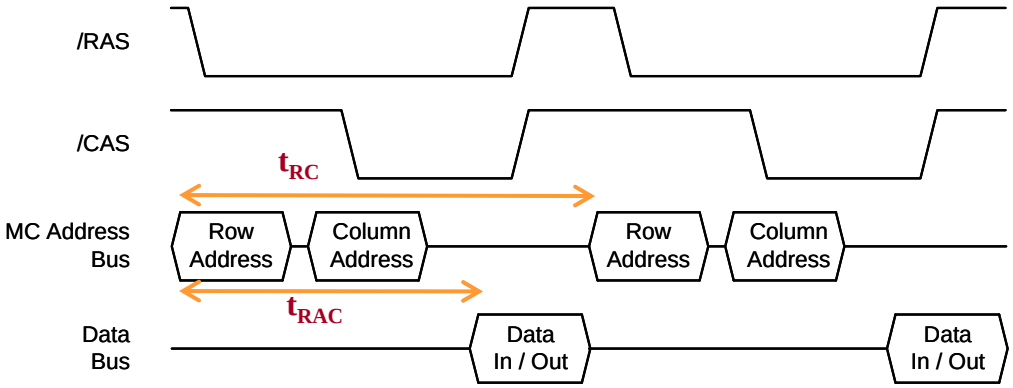


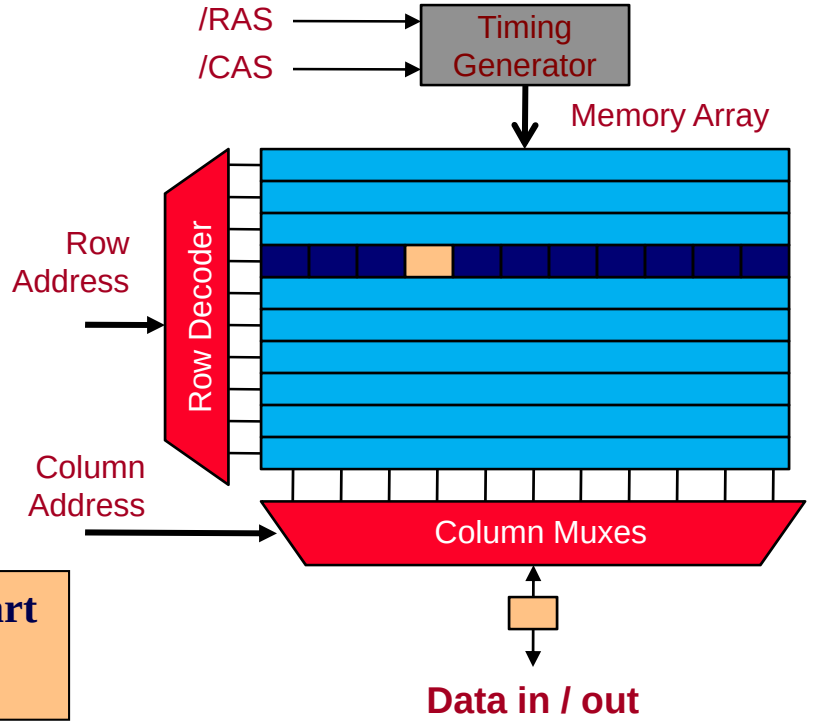
Implications of Memory Technology

- Memory latency of a single access using current DRAM technology will be slow
- We must improve bandwidth
  - Idea 1: Access more than just a single word at a time
    - Technology: EDO (Extended Data Out,) SDRAM (Synchronous DRAM,) etc.
  - Idea 2: Increase number of accesses serviced in parallel (in-flight accesses)
    - Technology: Banking

Legacy DRAM Timing

- Legacy DRAM must present new Row/Column address for each access. It can have only a single access "in-flight" at once
- Memory controller must send row and column address portions for each access
  - RAS active holds the row open for reading/writing
  - CAS active selects the column to read/write



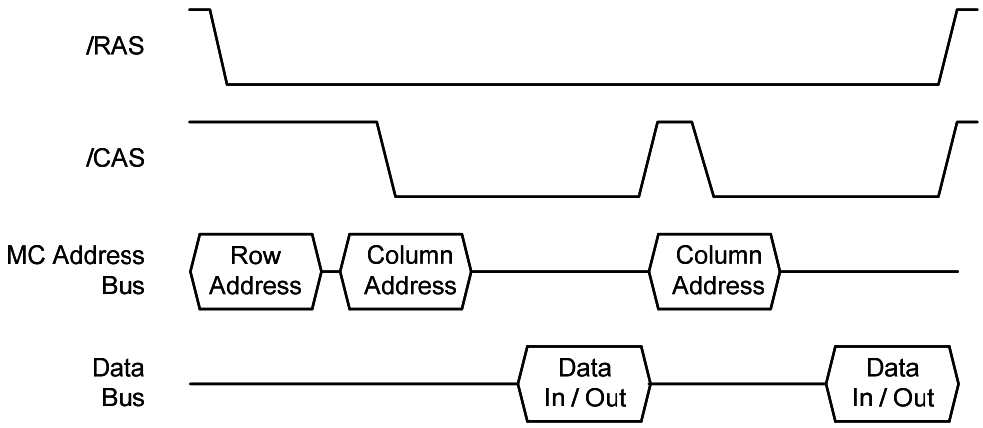


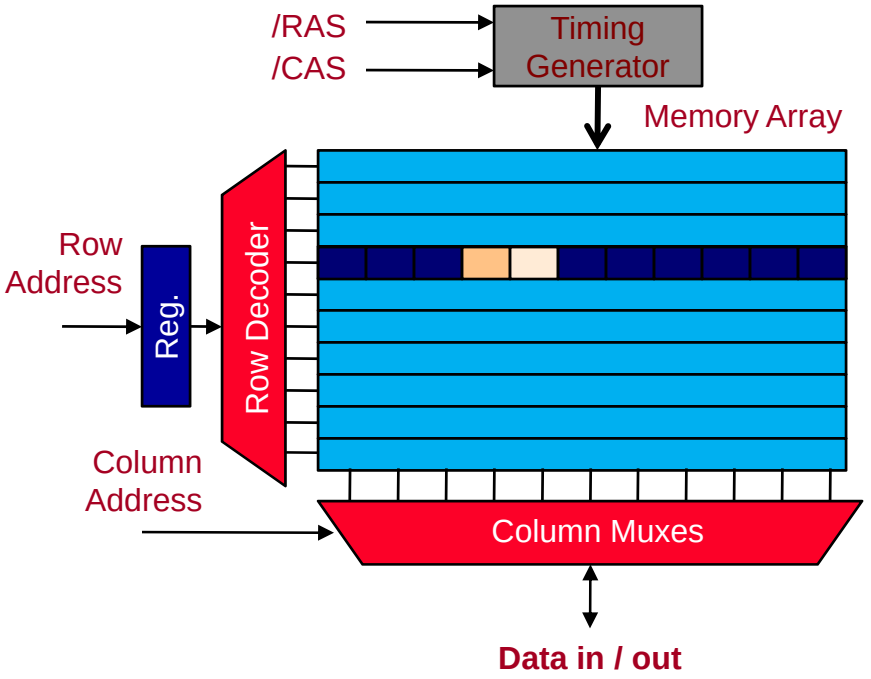
t_{RC}= Cycle Time (110ns) = Time before next access can start

t_{RAC}=Access Time (60ns) = Time until data is valid

FPM (Fast Page Mode) DRAM Timing

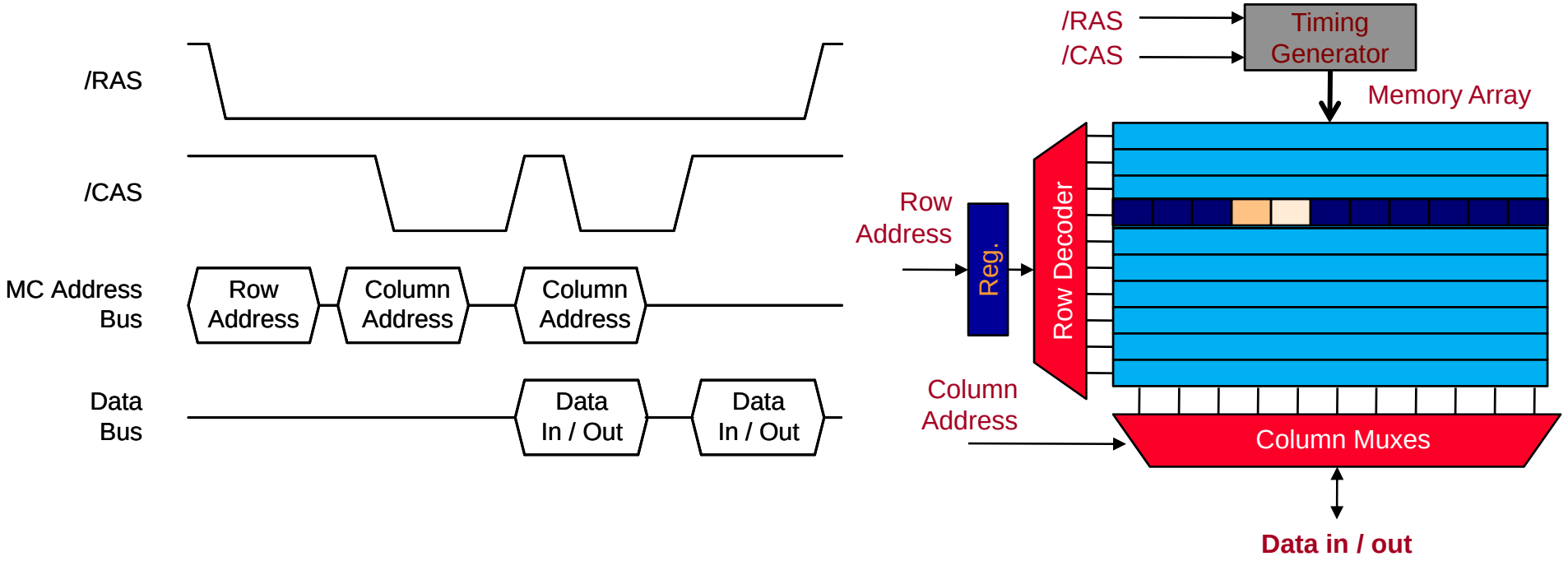
- In case of FPM, future addresses that fall in same row can pull data from the latched row
- FPM can provide multiple column addresses with only one row address





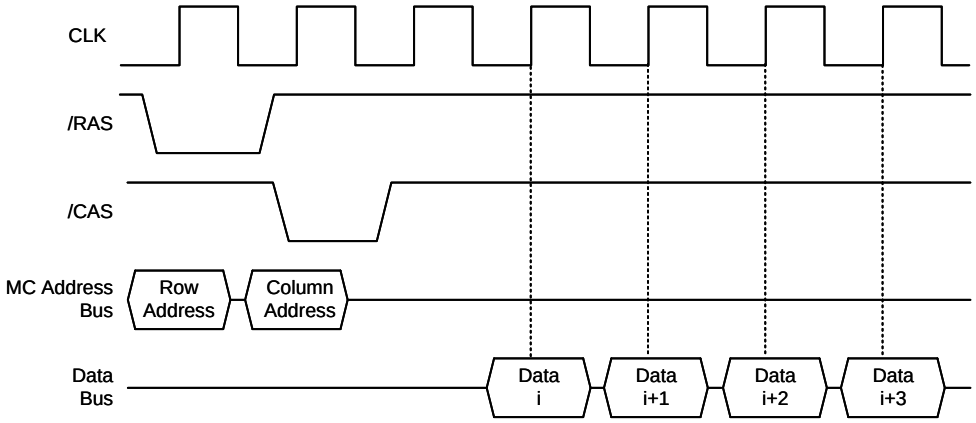
EDO (Extended Data Out) DRAM Timing

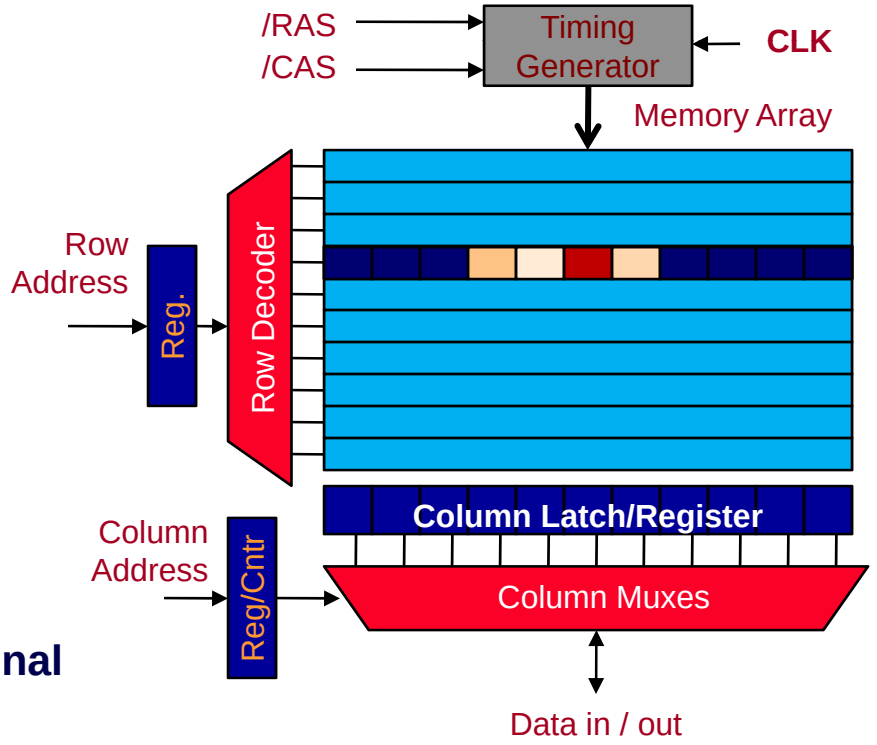
- In EDO, column address i+1 is sent while data i is being transferred on the bus
- Similar to FPM but overlaps data i with column address i+1



SDRAM (Synchronous DRAM) Timing

- SDRAM registers the column address and automatically increments it, accessing *n* sequential data words in *n* successive clocks called bursts… n=4 or 8 usually)
  - Will get up to 'n' consecutive words in the next 'n' clocks after column address is sent

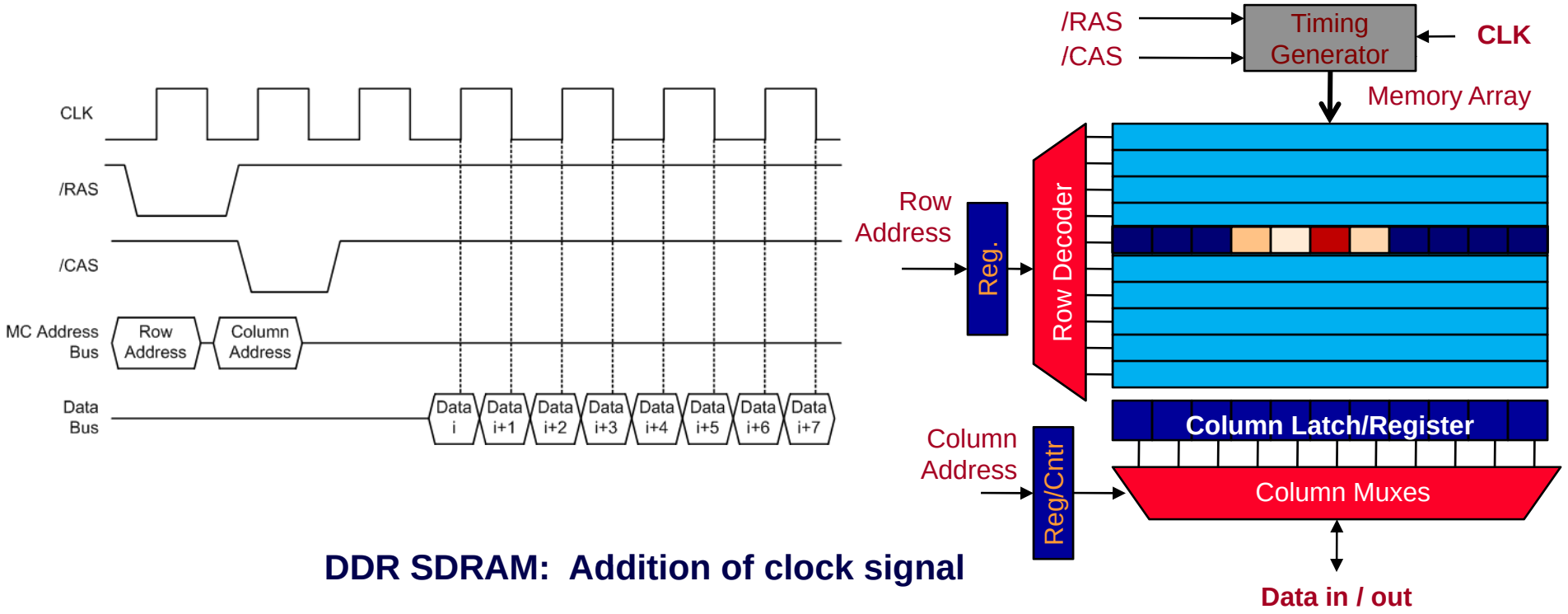




SDRAM: Addition of clock signal

DDR (Double Data Rate) SDRAM Timing

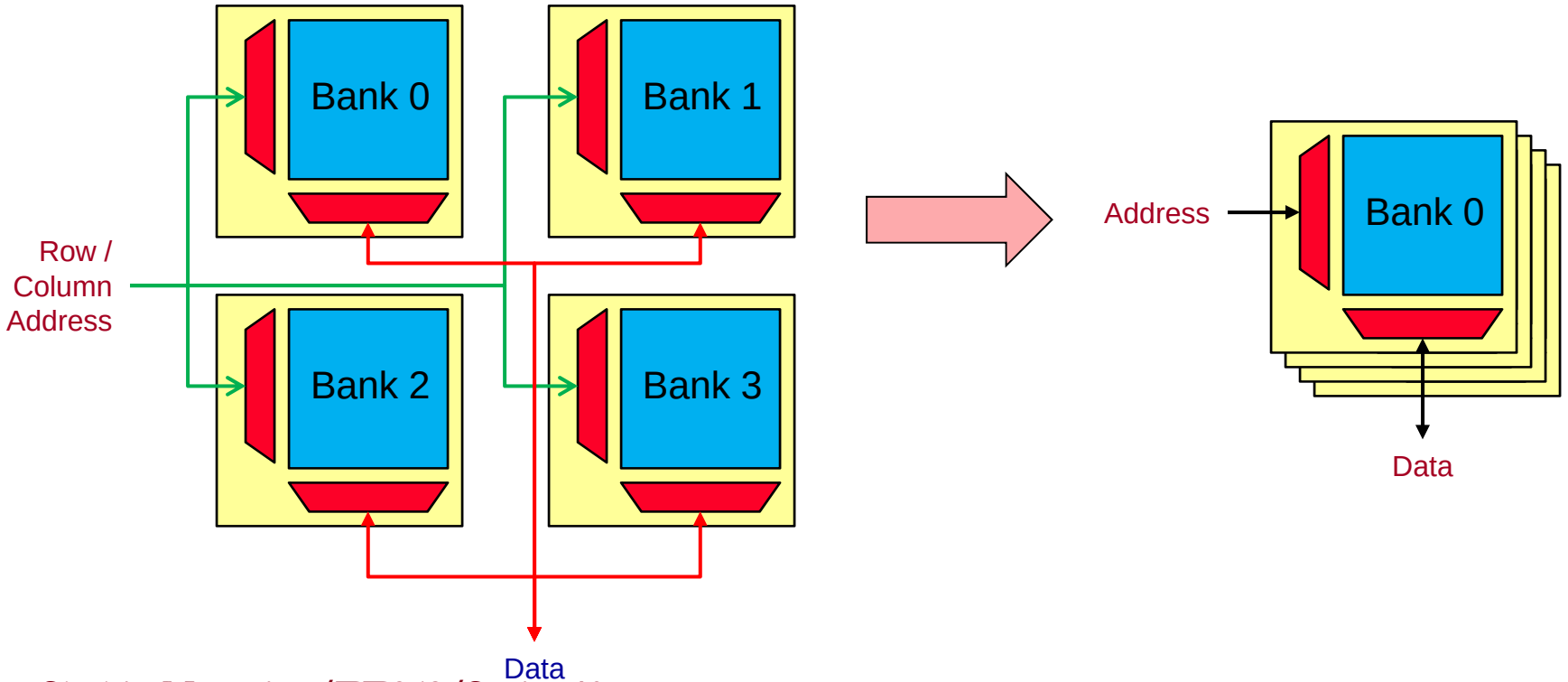
- In DDR the data is accessed every half clock cycle
  - Will get up to '2n' consecutive words in the next 'n' clocks after column address is sent



DDR SDRAM: Addition of clock signal

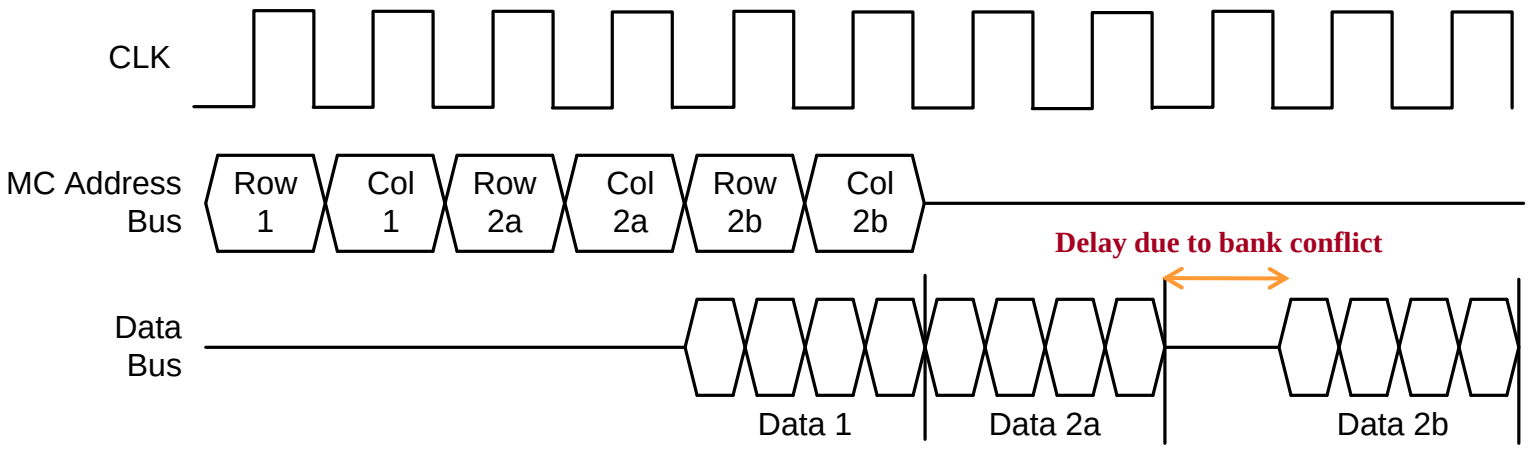
Banking

- Divide memory into "banks" duplicating row/column decoder and other peripheral logic to create independent memory arrays that can access data in parallel
  - Uses a portion of the address to determines which bank to access



Bank Access Timing

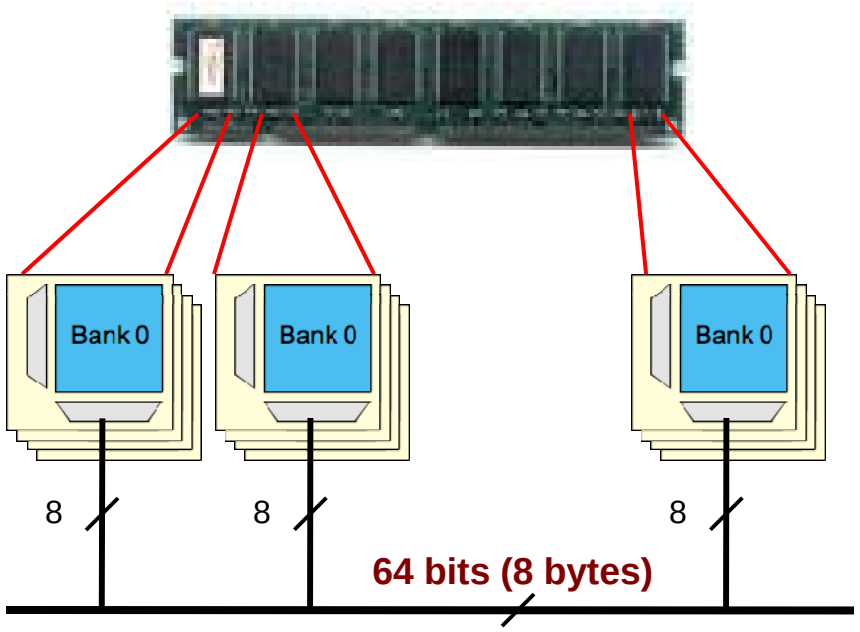
- Consecutive accesses to different banks can be overlapped to hide the time to access the row and select the column
- Consecutive accesses within a bank (to different rows) still exposes the access latency



Access 1 maps to bank 1 while access 2a maps to bank 2 allowing parallel access. However, access 2b immediately follows and maps to bank 2 causing a delay

Memory Modules

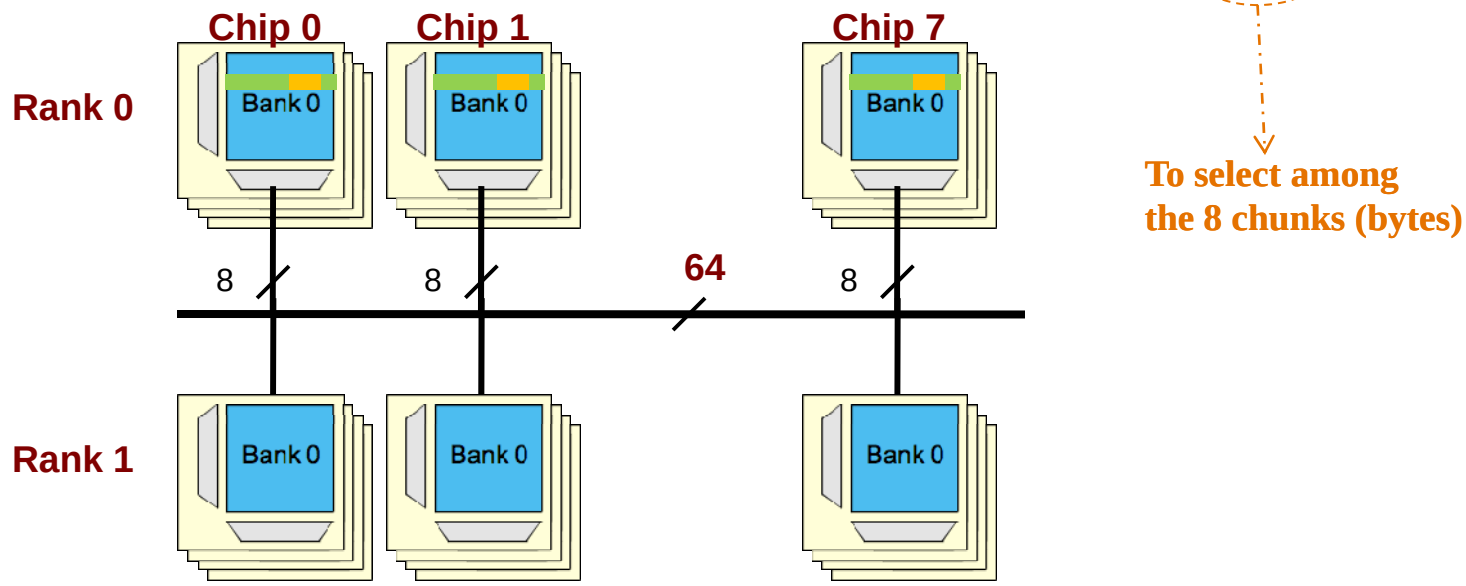
- Integrate several chips on a module
  - SIMM (Single In-line Memory Module) = single sided
  - DIMM (Dual In-line Memory Module) = double sided
- Each side is termed a rank
  - SIMM = 1 rank
  - DIMM = 2 ranks
- Example
  - Eight 1Mx8 chips each output 8 bits to form a 64-bit data bus



Address Breakdown

- Assume a 1GB memory system made of 2 ranks of eight 64Mx8-bit chips connected to a 64-bit (8-byte) wide data bus
  - Each chip consists of 4 banks x 16K rows x 1K cols x 8 bits, i.e, 64Mx8 bits

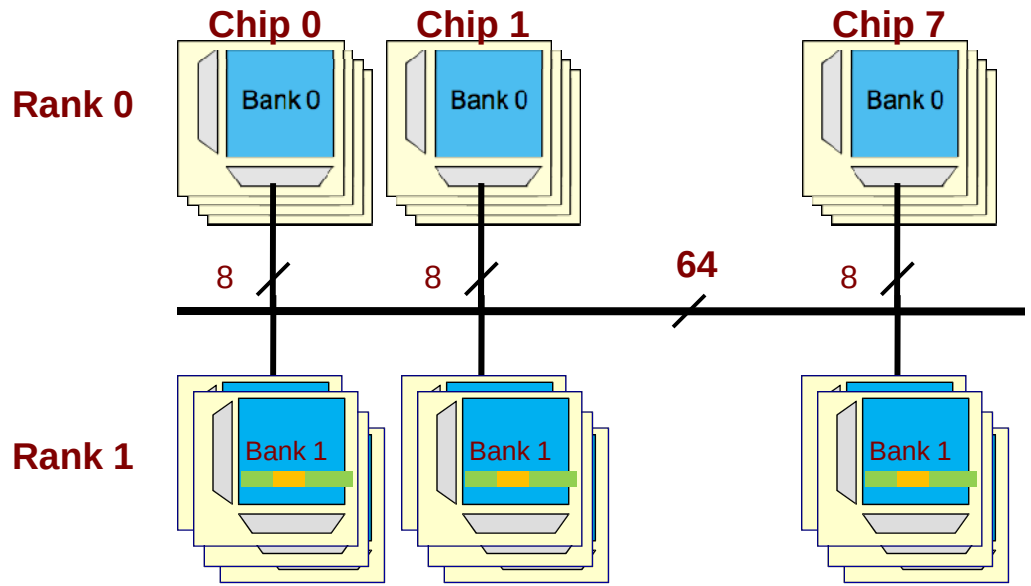
	Unused	Rank	Row	Bank	Col.	Unused
	A31,A30	A29	A28…A15	A14,A13	A12…A3	A2..A0
0x004f1ad8	00	0	00 0000 1001 1110	00	1101011011	000



Address Breakdown

- Assume a 1GB memory system made of 2 ranks of eight 64Mx8-bit chips (each has 4 x 16K x 1K x 8 bits) connected to a 64-bit (8-byte) wide data bus

	Unused	Rank	Row	Bank	Col.	Unused
	A31,A30	A29	A28…A15	A14,A13	A12…A3	A2..A0
0xff05aac0	11	1	11 1110 0000 1011	01	0101011000	000



Address Breakdown

- Assume 2 GB of memory on a single SIMM module with eight 256MB chips each broken into 8 banks and 8K rows and 4K columns, i.e, 8 x 8K x 4K x 8 bits

Unused	Rank	Row	Bank	Col.	Unused
A31	-	A30-A18	A17-A15	A14-A3	A2-A0

Memory Summary

- Opportunities for speedup / parallelism
  - Sequential accesses lower latency (due to bursts of data via FPM, EDO, SDRAM techniques)
  - Ensure accesses map to different banks and ranks to hide latency (parallel decoding and operation)

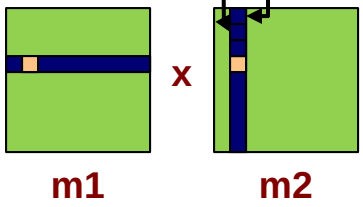
Programming Considerations

- For memory configuration given earlier, accesses to the same bank but different row occurs on an 32KB boundary
- Now consider a matrix multiply of 8K x 8K integer matrices (i.e. 32KB x 32KB)
- In code below…m2[0][0] @ 0x10010000 while m2[1][0] @ 0x10018000

Unused	Rank	Row	Bank	Col.	Unused
A31,A30	A29	A28…A15	A14,A13	A12…A3	A2..A0
00	0	1 0000 0000 0001 0	00	0000000000	000
00	0	1 0000 0000 0001 1	00	0000000000	000

0x10010000

0x10018000

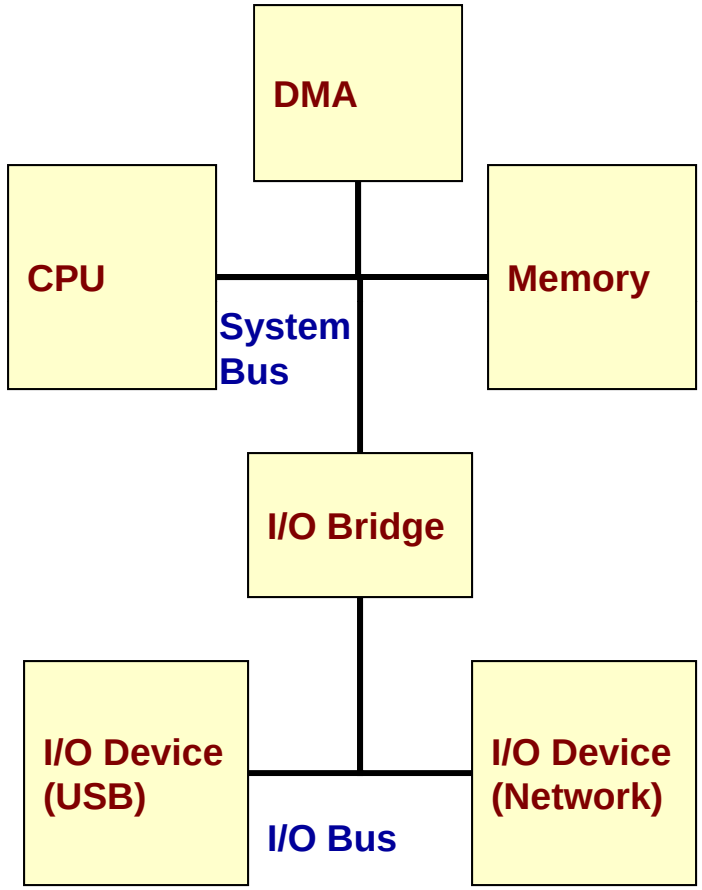



```
int m1[8192][8192], m2[8192][8192], result[8192][8192];
int i,j,k;
...
for(i=0; i < 8192; i++){
  for(j=0; j < 8192; j++){
    result[i][j]=0;
    for(k=0; k < 8192; k++){
      result[i][j] += matrix1[i][k] * matrix2[k][j];
} } }
```

DMA & ENDIAN-NESS

Direct Memory Access (DMA)

- Large buffers of data often need to be copied between:
  - Memory and I/O (video data, network traffic, etc.)
  - Memory and Memory (OS space to user app. space)
- DMA devices are small hardware devices that copy data from a source to destination freeing the processor to do "real" work

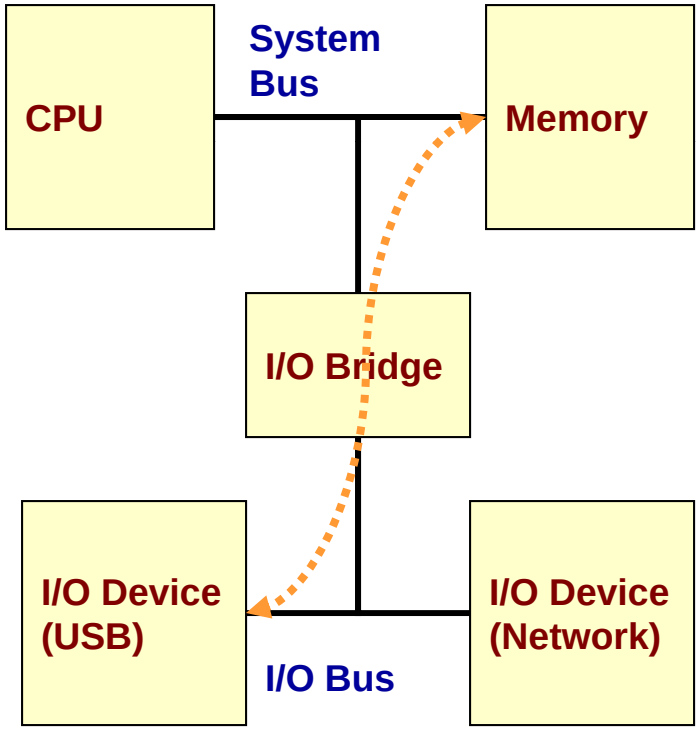


Data Transfer w/o DMA

- Without DMA, processor would have to move data using a loop
- Move 16Kwords pointed to by ($s1) to ($s2)

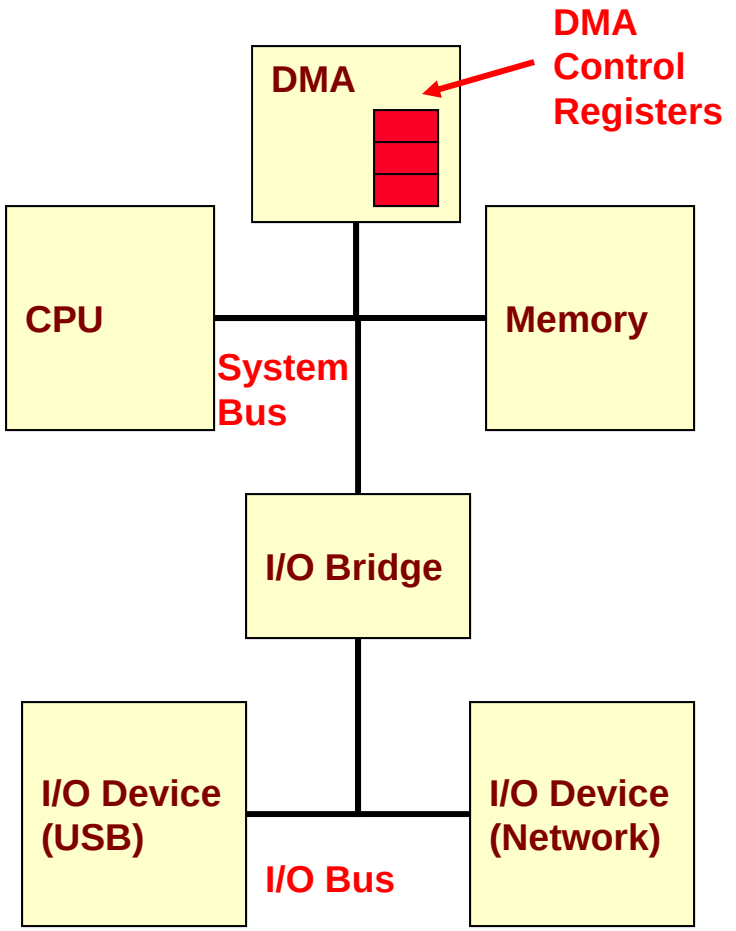
```
        li    $t0,16384
AGAIN:  lw    $t1,0($s1)
        sw    $t1,0($s2)
        addi  $s1,$s1,4
        addi  $s2,$s2,4
        subi  $t0,$t0,1
        bne   $t0,$zero,AGAIN
```

- Processor would waste valuable execution time moving data



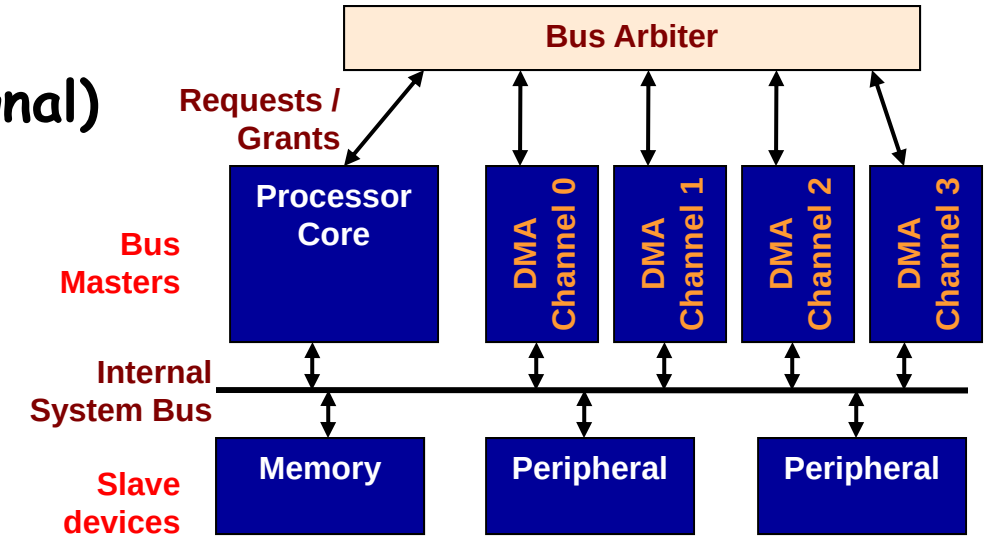
Data Transfer w/ DMA

- Processor sets values in DMA control registers
  - Source Start Address
  - Destination Start Address
  - Byte Count
  - Control & Status (Start, Stop, Interrupt on Completion, etc.)
- DMA becomes "bus-master" (controls system bus to generate reads and writes) while processor is free to execute other code
  - Small problem: Bus will be busy
  - Hopefully, data & code needed by the CPU will reside in the processor's cache



DMA Engines

- Systems usually have multiple DMA engines/channels
- Each can be configured to be started/controlled by the processor or by certain I/O peripherals
  - Network or other peripherals can initiate DMAs on their behalf
- Bus arbiter assigns control of the bus
  - Usually winning requestor has control of the bus until it relinquishes it (turns off its request signal)



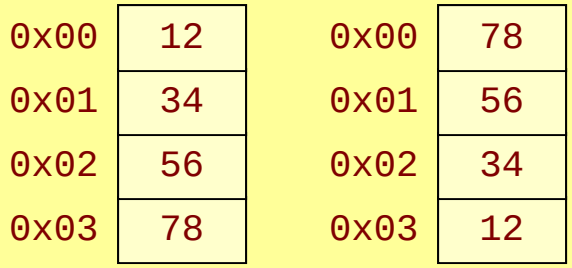
Endian-ness

- Endian-ness refers to the two alternate methods of ordering the bytes in a larger unit (word, long, etc.)
  - Big-Endian
    - PPC, Sparc
    - *MS byte* is put at the starting address
  - Little-Endian
    - Used by Intel processors / PCI bus
    - *LS byte* is put at the starting address

The longword value:

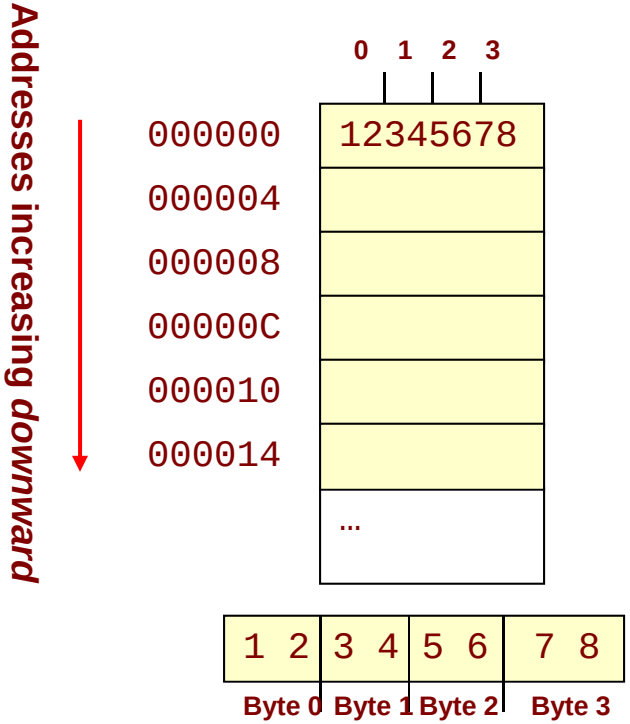
0 x 1 2 3 4 5 6 7 8

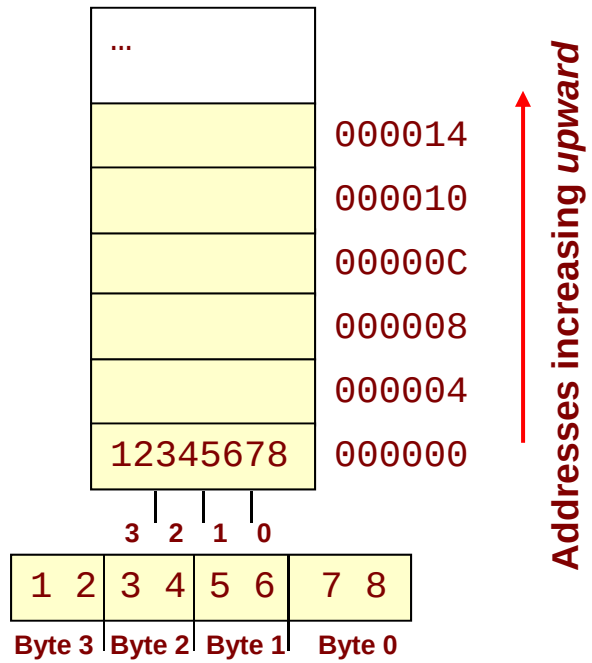
can be stored differently



Big-endian vs. Little-endian

- Big-endian
  - Makes sense if you view your memory as starting at the top-left and addresses increasing as you go down
- Little-endian
  - Makes sense if you view your memory as starting at the bottom-right and addresses increasing as you go up





Big-endian vs. Little-endian

- Issues arise when transferring data between different systems
  - Byte-wise copy of data from big-endian system to little-endian system
  - Major issue in networks (little-endian computer => big-endian computer) and even within a single computer (System memory => Peripheral (PCI device) [PCI≡ Peripheral Component Interconnect]