Memory Technology
and
System-level Memory Design

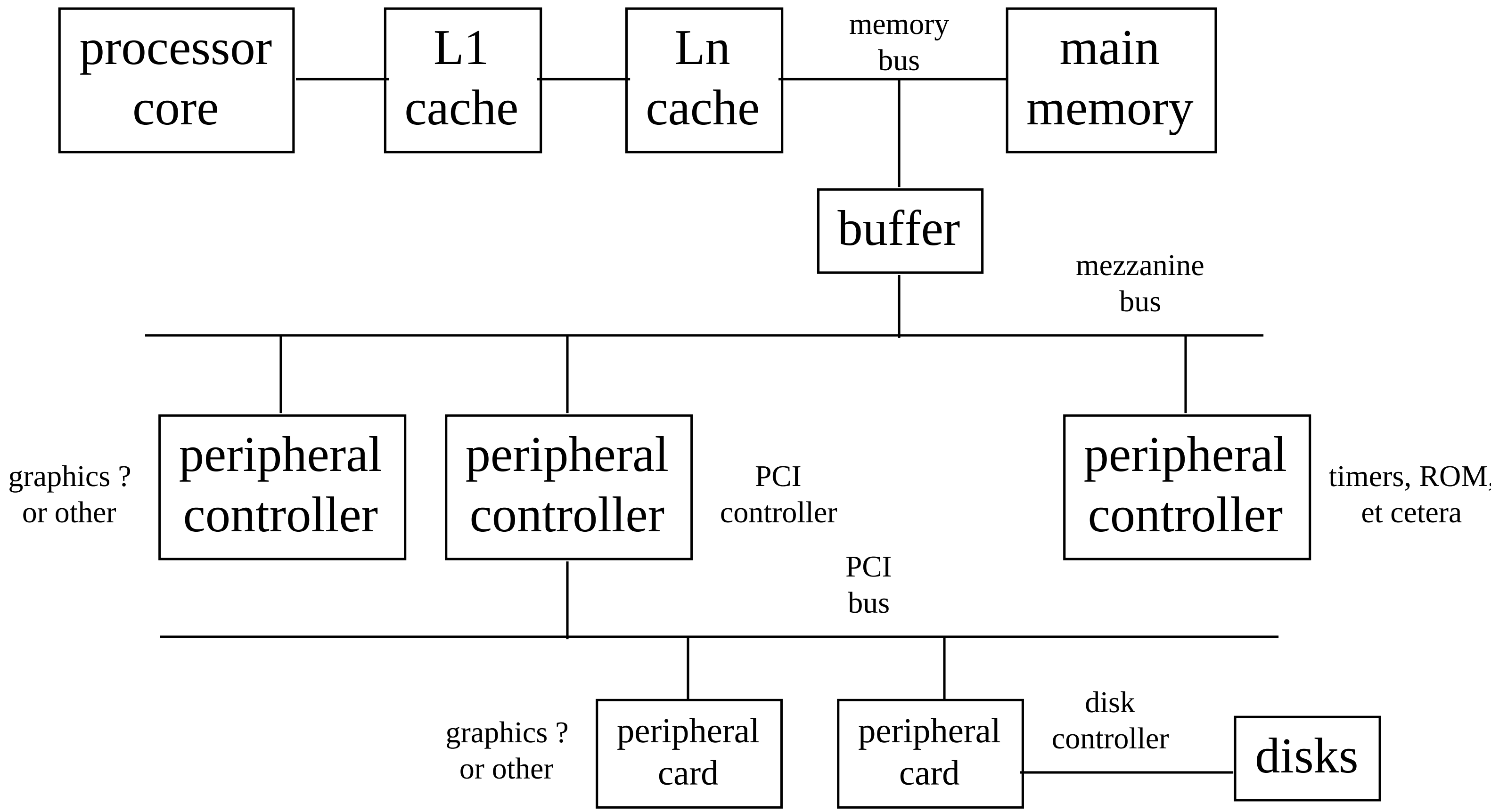
presented by **David Morano**
class professor **Professor David Kaeli**

class notes 03/09/29

objectives

- **review**
- **memory technology**
  - SRAM
  - DRAM
- **system-level design using memory**
  - SRAM
  - DRAM
    - refresh stragegies
    - burst read and write
    - error correction
- **questions**

posssibe computer organization



- **many, many variations**
- **more or less bus levels depending on overall computer size**

memory hierarchy

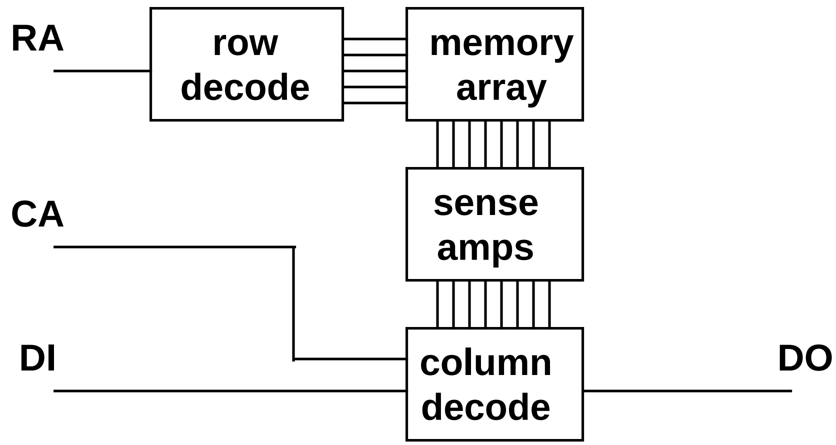
registers	64	flip-flops
L1 cache	32k	SRAM
L2 cache	1M	SRAM
Ln cache	4M	SRAM or DRAM
main memory	16G	DRAM
disk	1T	magnetic disk (w/ own cache)
storage archive	160T	optical disk, magnetic tape

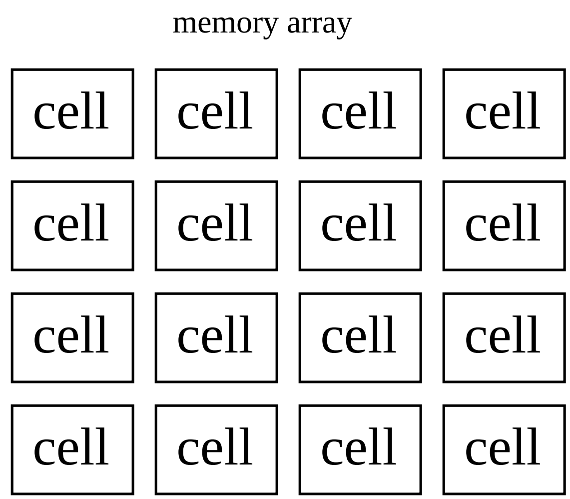
- **organized fastest to slowest**
- **highest cost per bit to lowest cost per bit**
- **smallest capacity components to largest capacity components**

memory technologies

- **SRAM and DRAM**
  - basic circuit
  - operation
  - features
  - operation and timing (for DRAM)
- **other**
  - not going to talk about them !

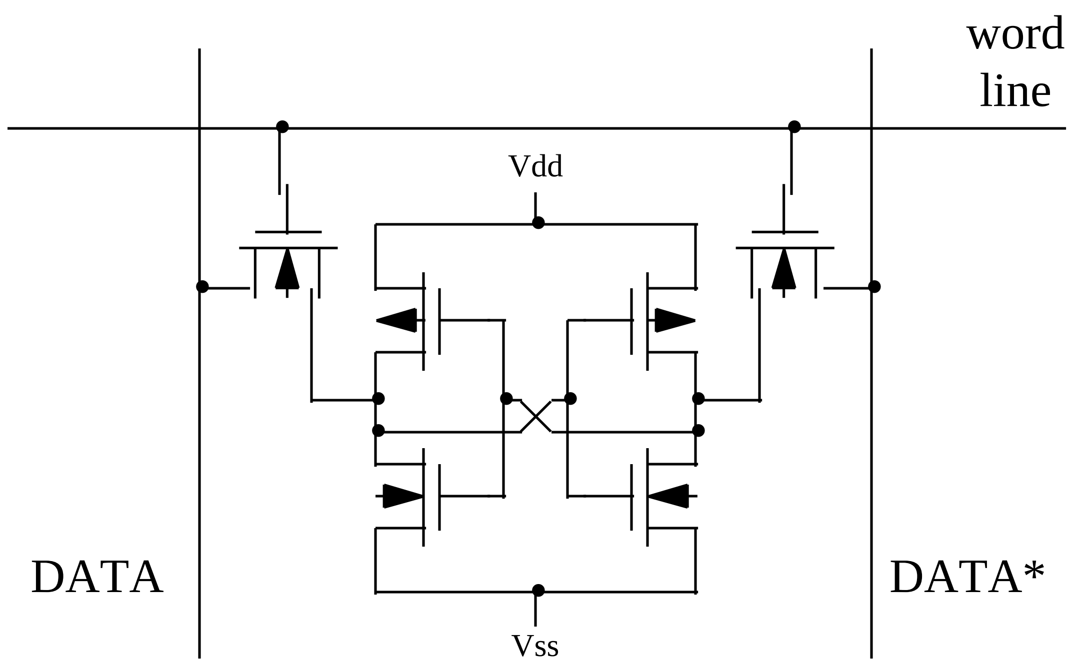
basic SRAM integrated circuit





- **RA Row Address**
- **CA Column Address**
- **DI   Data In**
- **DO Data Out**
- **only one column is selected at a time by the column decoder**

basic SRAM memory cell

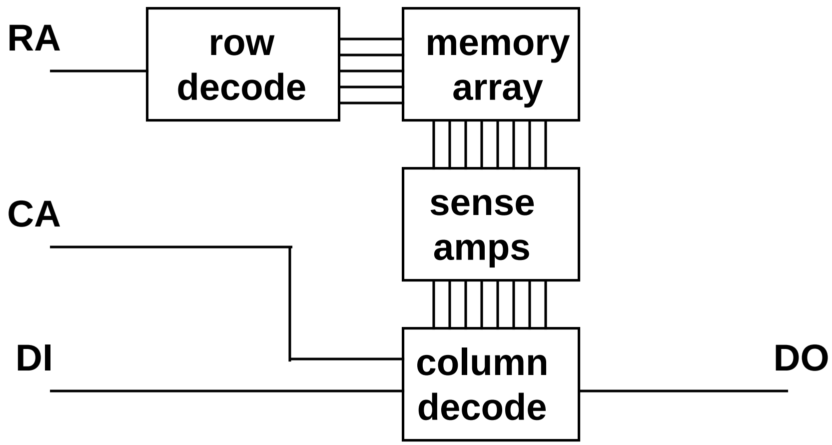


- **this example is CMOS -- six transistors, four N-channel devices, two P-channel**
- **two cross-coupled inverters**
- **needs two column lines per cell -- DATA and DATA***
- **word line selects a row of cells**
- **data (column) lines act on the selected cell**
- **data lines are generally pre-charged to Vdd / 2 before a read**

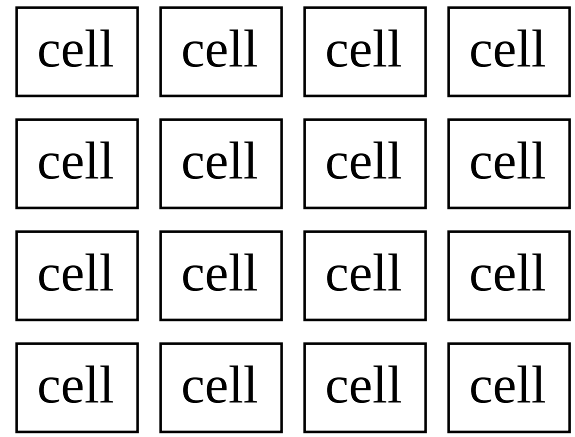
basic SRAM features

- **the cell bit is actively reinforced in the presence of noise by the inverters**
- **the state of the cell is changed by overpowering the outputs of the two inverters**
- **data is read out very quickly**
- **no power is needed to refresh the bit cell on a read**
- **control for reads and writes is very simple**
- **can use the same sort of IC manufacturing processing as used for the processor**
- **can be readily integrated on the same IC as the processor**

basic DRAM integrated circuit

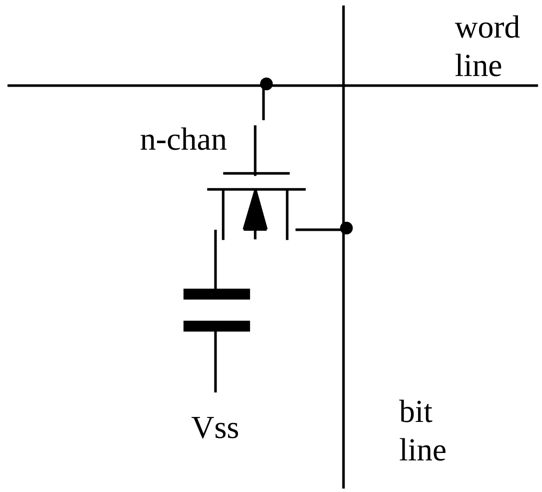


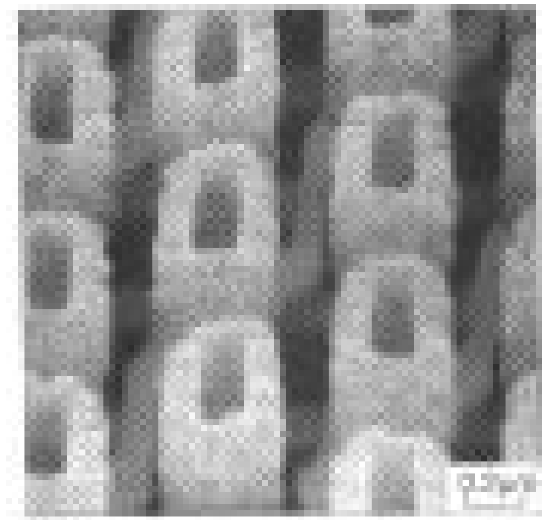
memory array



- **RA Row Address**
- **CA Column Address**
- **DI Data In**
- **DO Data Out**
- **all columns are active on a row access (read out and refreshed)**
- **sense amplifiers double as drivers for writes and drivers for refresh on reads**
- **all reads refresh the entire row they are in**

basic DRAM memory cell

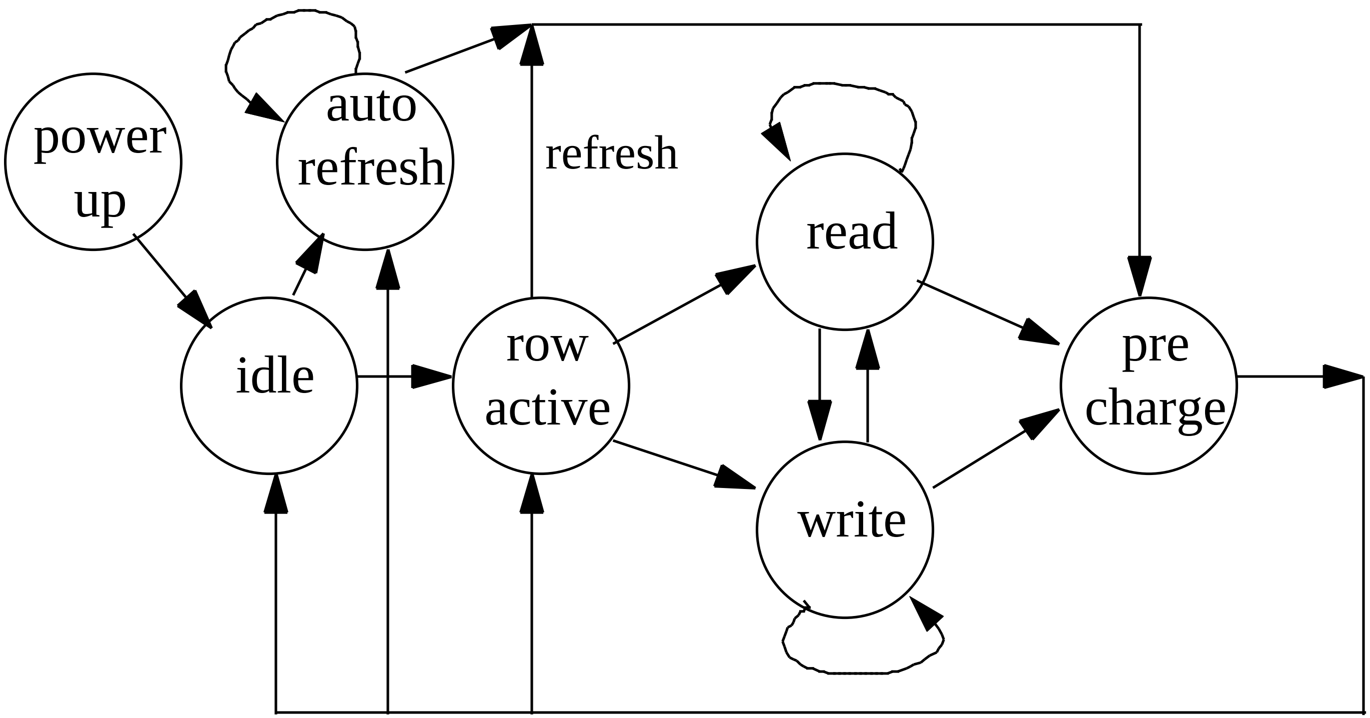




1 Gbit

- **this is a "one transister cell" (standard now -- there were 3 originally)**
- **one transister and one capacitor is used**
- **the word line selects a whole row of cells to work on (either read or write)**
- **the data line reads and writes the capacitor on a READ (regenerative read), and writes it on a WRITE**
- **the problem is in carefully reading the fragile voltage on the capacitor**
- **the bit line is pre-charged to Vdd / 2**
- **another reference capacitor is normally used for output voltage comparison**

simplified basic DRAM operation



- all DRAMs share a similar operational philosophy
- a DRAM "row" is also called a "page"
- "row active" is entered by providing a "row address" (RA)
- read and write is entered by providing a "column address" (CA)

basic DRAM control (DDR-SDRAM)

control signals

- **CLK -- Clock**
- **CKE -- Clock Enable** — **extended command operations**
- **CS* -- Chip Select**
- **RAS* -- Row Address Strobe**
- **CAS* -- Column Address Strobe**
- **WE* -- Write Enable**
- **DM -- Data Mask** — **masking out bits on WRITEs**
- **DQS -- Data Strobe** — **strobing data on WRITEs and READs**

forms basic commands

usage notes

- **all commands are synchronous with respect to the clock**
- **data is roughly synchronized to the clock and strobed by DQS**
- **the data rate is twice the clock rate (Double Data Rate -- DDR)**
- **minimum READ or WRITE burst length is 2 !**
- **all reads and writes are bursts of 2, 4, or 8 transfers**

basic DRAM features (1)

- **higher density than comparable generation of SRAM (always about 4 times higher)**
- **since cost tracks transistors, DRAMs always have a much better cost-per-bit than comparable generation SRAMs**
- **much slower than SRAM -- reading a capacitor is fairly difficult and slow in comparison to the reading of the differential output of two inverters, as with SRAM !**
- **the required timings and necessary control logic is fairly complex as compared with SRAM**
- **needs refreshing in order to maintain data storage integrity -- the charge on the capacitors leaks off ! -- typically 8k rows every 64 msec**
- **the memory bit cell is passive (not actively regenerated as with SRAM)**
- **has a higher soft bit error rate than SRAM -- things happen that blow away the charge on the capacitors !**
- **although it can be integrated along with a processor on the same IC, this is generally NOT done since it would compromise the performance of either the processor, the DRAM, or both !!**

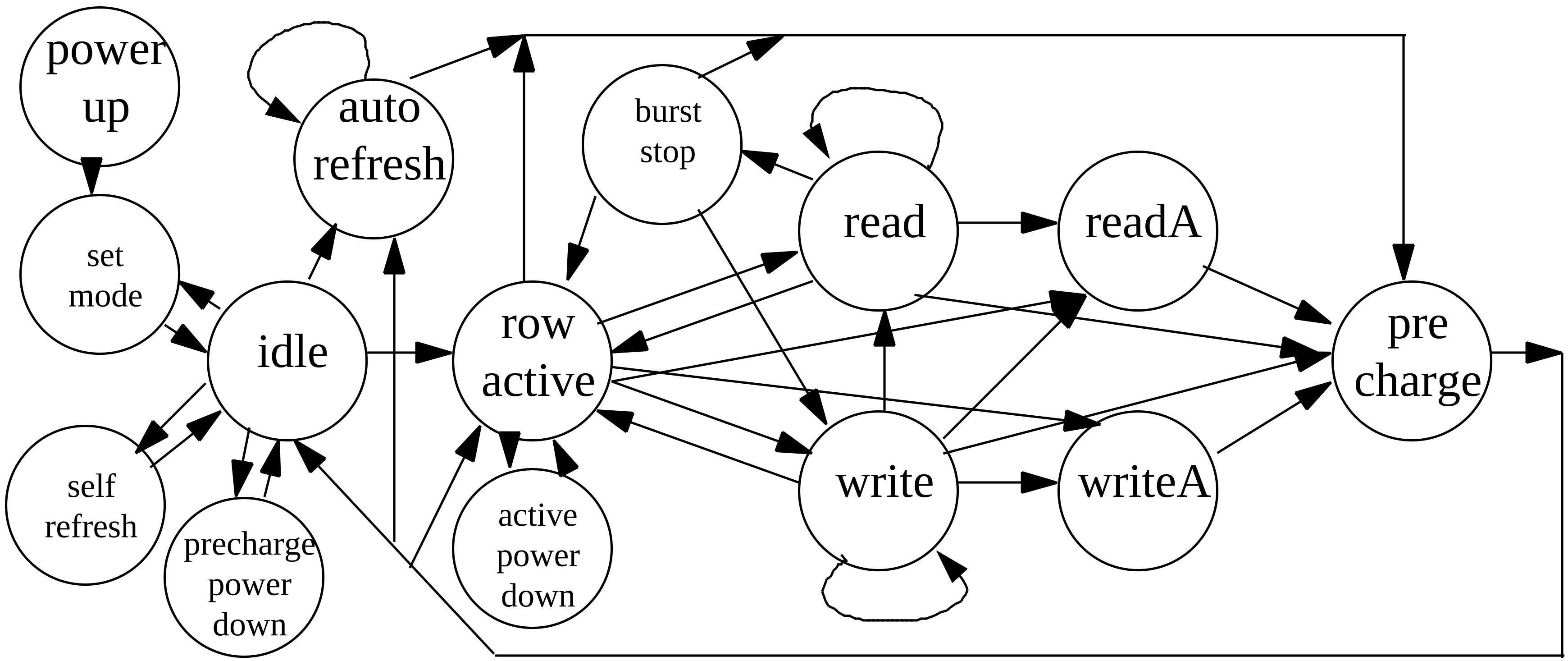
basic DRAM features (2)

- **cycle times (time to start a new "row" access) are long !!**
- **burst transfers are available (practically mandatory) and increase the access bandwidth**
- **for DDR SDRAMs, all reads and writes are bursts of 2, 4, or 8 transfers**
- **individual bursts can be prematurely terminated ("stopped")**
- **reads and writes can follow each other making long trains of burst transfers**
- **all memory accesses are made first to a "page" (row) and then to the bits within that page**
- **there is a required delay before going to another "page" -- "pre-charge"**
- **new DRAMs have multiple independent banks within them (generally 2 or 4 banks -- so far)**

basic DRAM features (3)

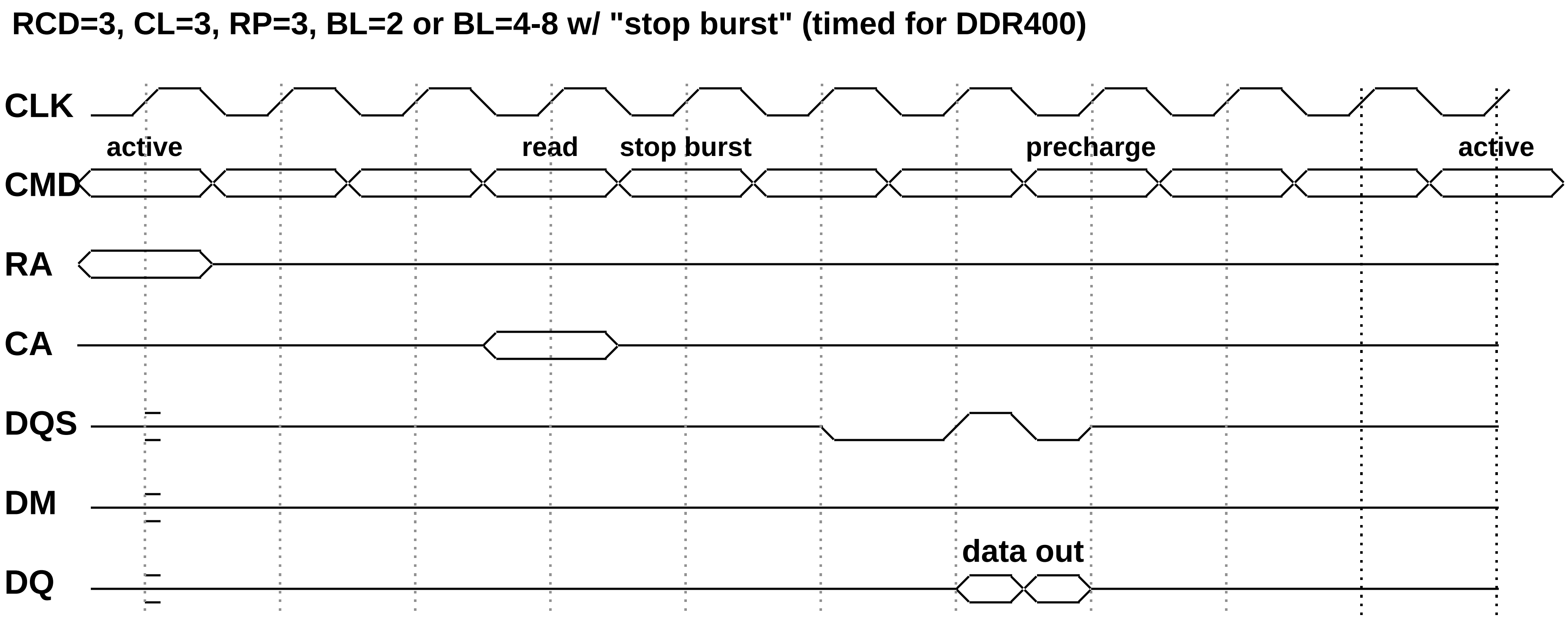
- **example speeds :**
  - **DDR333**
    - **167 MHz clock, CL=2.5, RCD=3, RP=3**
    - **earliest read data 33 nsec, single cyle time 60 nsec, BW 16.7 Mtps**
    - **burst read of 32, BW 281 Mtps**
  - **DDR400**
    - **200 MHz clock, CL=3, RCD=3, RP=3**
    - **earliest read data 30 nsec, single cyle time 50 nsec, BW 20.0 Mtps**
    - **burst read of 32, BW 337 Mtps**

simplified DRAM operation (DDR)



- "readA" -- read followed by "auto-precharge" ; similar w/ "writeA"
- each read or write is generally a burst operation with programmed length
- due to bus turn-around, a read transitions to write through "burst stop"
- new DRAMs now have "power down" modes
- DRAM operational mode is programmed after power up

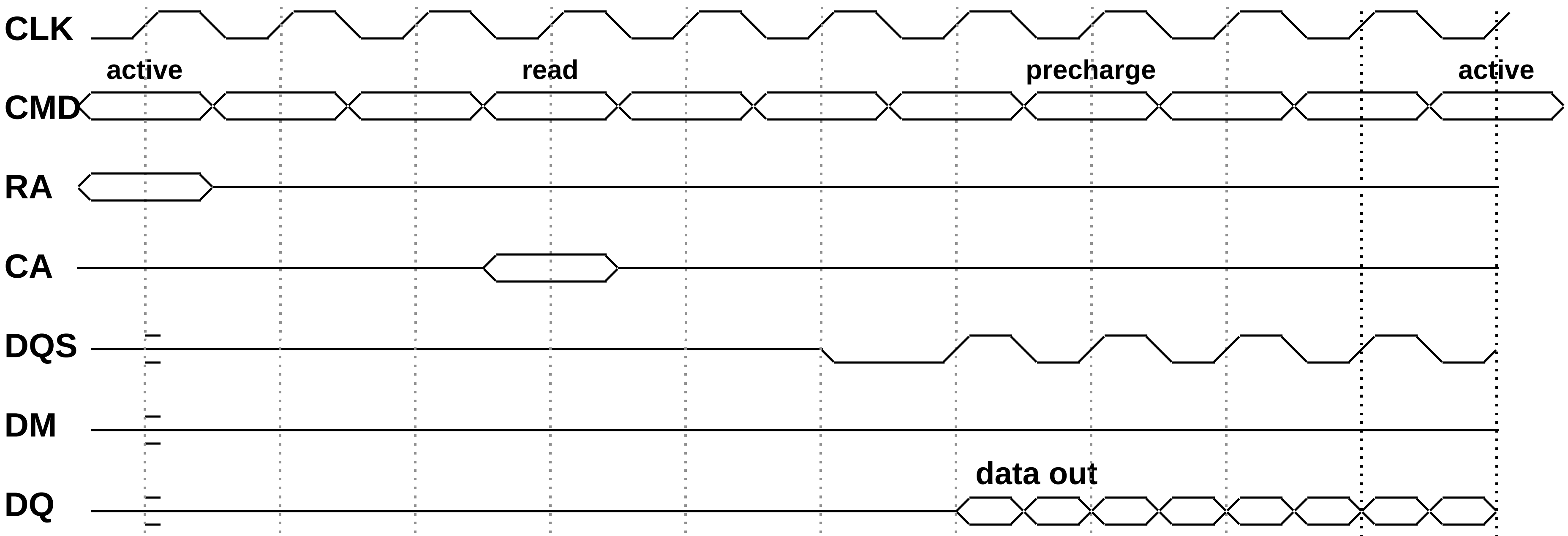
basic single read



- **minimum of 2 data transfers (due to DDR) -- ignore any not wanted !**
- **the long cycle time limits the bandwidth when doing short transfers !**

basic burst read

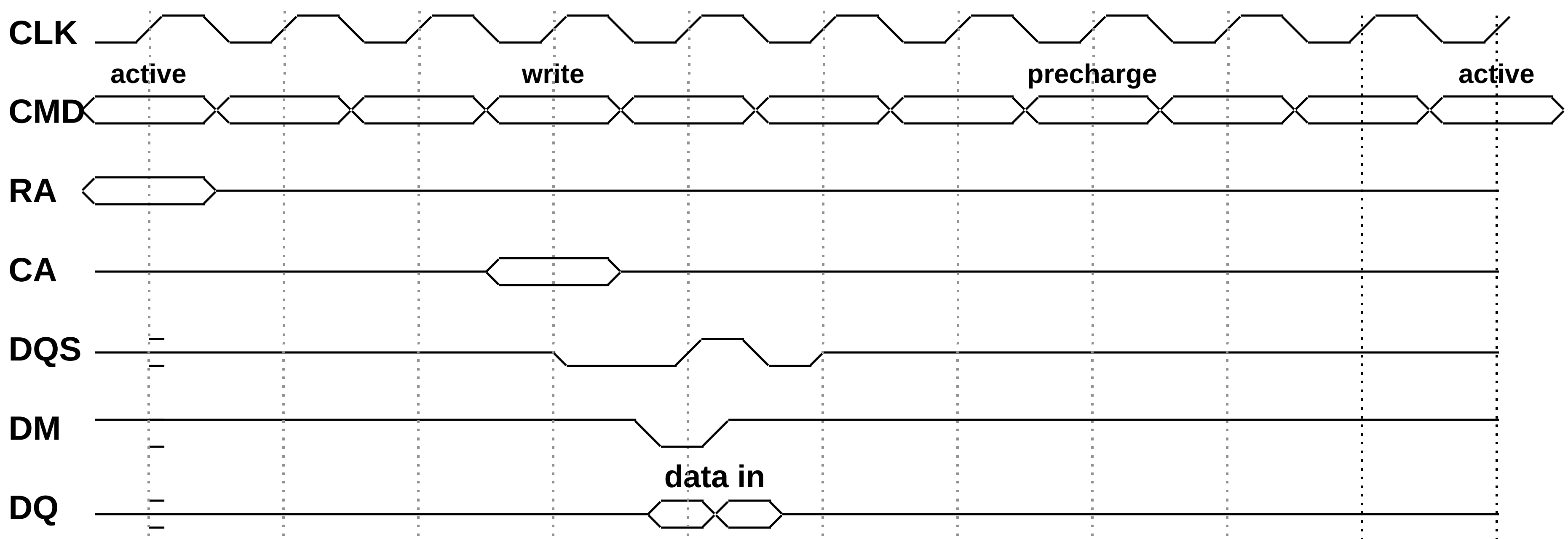
RCD=3, CL=3, RP=3, BL=8 (timed for DDR400)



- **burst transfer of 8**
- **but bandwidth is still low due to the long cycle time !**

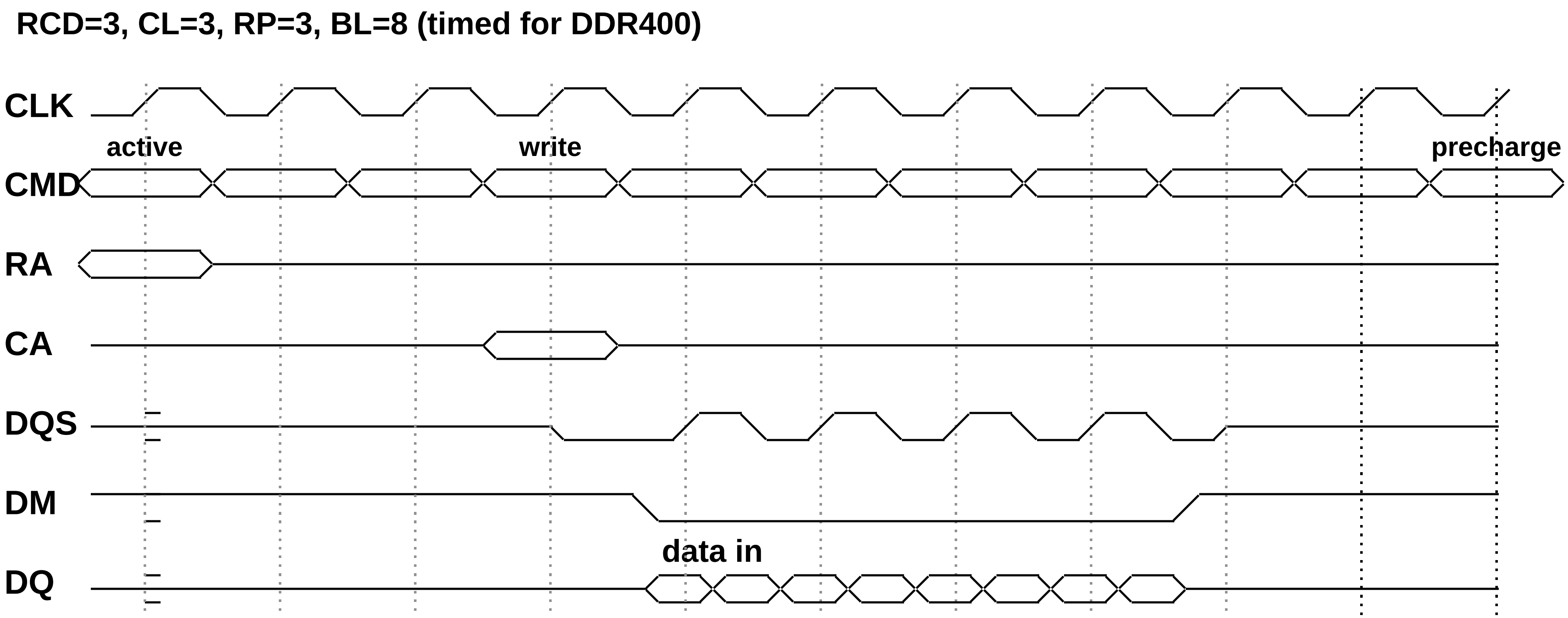
basic single write

RCD=3, CL=3, RP=3 (timed for DDR400)



- **minimum of 2 data transfers (due to DDR)**
- **second data transfer is "masked" in this example using DM signal**
- **the 2 clocks after the write data is supplied forms "write recovery" time**
- **the long cycle time limits the bandwidth when doing short transfers !**

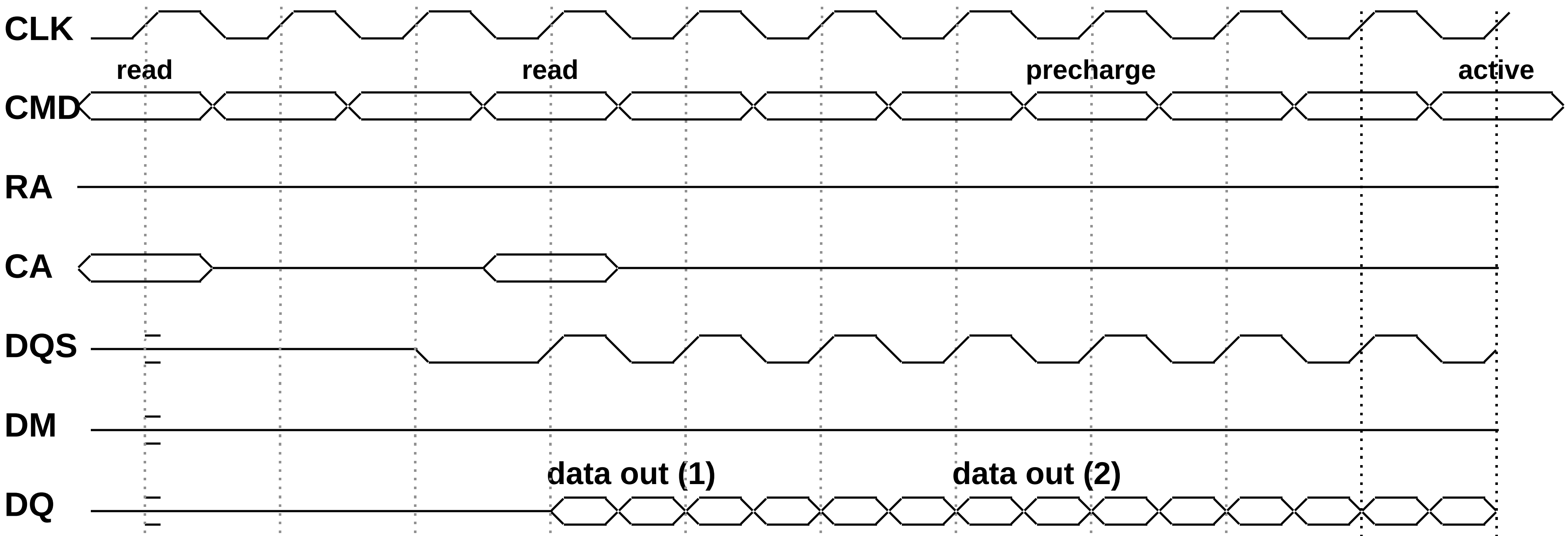
basic burst write



- **burst transfer of 8**
- **the whole cycle is not even shown -- need 3 clocks more !**
- **even w/ the burst the long cycle time of 13 clocks still limits the bandwidth**

read interrupted by read

RCD=3, CL=3, RP=3, BL=8 (timed for DDR400)



- **first read of burst transfer of 8 was interrupted and only output 6**
- **the second read was able to put out 8**
- **different reads can have different column addresses (but still in same row)**

some other DRAM operations

- **read followed by read**
- **write followed by write**
- **burst read burst stopped**
- **burst read interrupted by stop & write**
- **burst write interrupted by read**
- **burst write interrupted by write**
- **burst read interrupted by precharge (need to close page)**
- **burst write interrupted by precharge (need to close page)**
- **all multi-bank variations of all of these so far !**
- **many variations of refresh**
- **more**

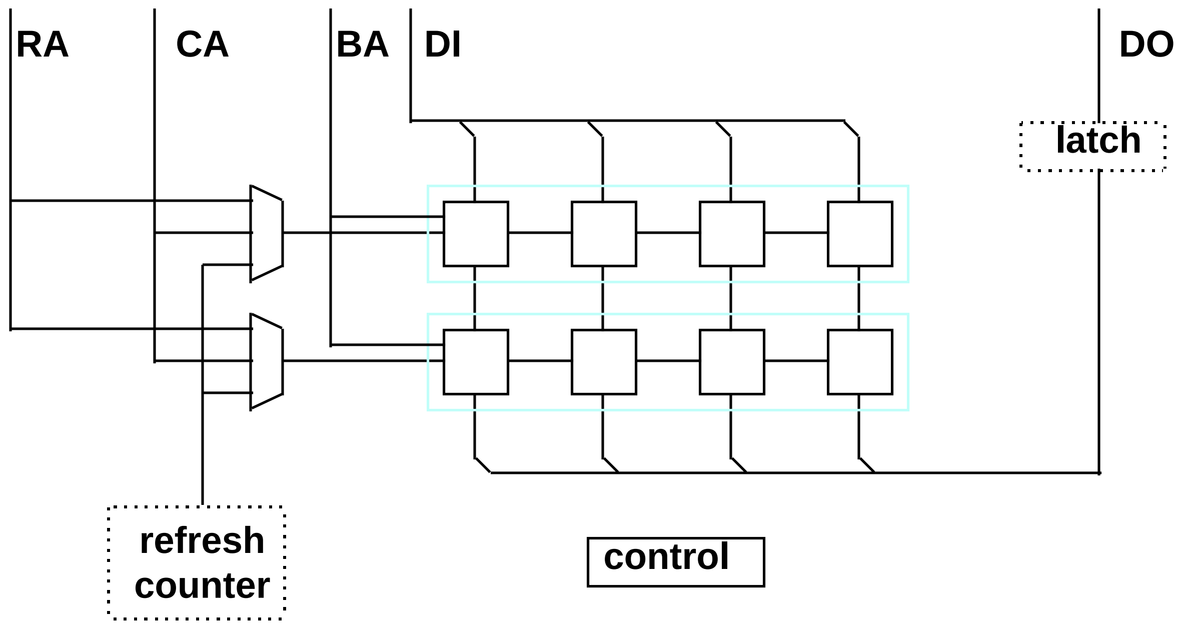
system-level memory design

- **what is SRAM good for in a computer**
- **representative DRAM IC layout**
- **basic DRAM refresh strategies**
- **delaying RAS precharge**
- **using multiple independent banks within the DRAM ICs (or otherwise)**
- **error detection and correction**
- **interaction between the cache and main memory**

using SRAM

- **since SRAM is :**
  - — fast
  - — tracks the processor clock rate fairly closely
  - — uses similar IC fabrication processing as the processor IC
  - — has relatively simple control logic requirements
- **use it for the caches**
  - — L1
  - — L2
  - — often L3
- **due to its high cost and lower bit density, it is not a good fit for :**
  - — main memory
  - — long-term storage

basic system DRAM circuit



- **BA Bank Address (for DRAMs with multiple banks)**
- **RA Row Address**
- **CA Column Address**
- **DI Data In**
- **DO Data Out**
- **controls for read, write, refresh, et cetera**

basic refresh strategies

- **refresh all rows in a single burst during each refresh period**
- **refresh one row at a time**
- **refresh a few rows in a burst periodically**
- **software does the refreshing**
- **try to hide the refreshes when the higher level cache doesn’t need access**
- **let something like video hardware do it for you**
- **let a DRAM controller do what it wants (or program it also)**
- **try to do a refresh when you think the processor is not about to request a transfer**
- **other -- use your imagination !**

delaying RAS precharge

- **use the current "open" page that is on the sense amplifiers as a cache**
- **we've seen this already with burst transfers to different column addresses**
- **what about more random non-burst transfers ?**
  - **put high processor addresses on DRAM row and low processor addresses on DRAM column**
  - **wait to see where the next access is addressed to**
  - **must time-out on maximum RAS "on" time**
  - **close the page after the last access to the page (if known)**
    - **last address in the page (like if they have been sequential)**
    - **last access to this page because another page is likely needed next (like if the processor is changing state of some kind)**
  - **time-out on waiting for the next access then close the current page**

using multiple independent banks

- **newer SDRAMs have multiple independent banks within each IC or DIMM**
- **this means that we can have one page open for each bank at any given time (very nice)**
- **of course, each bank can be independently active with their own read or write operations in progress**
- **we can "close" pages in some banks that we think we don't need while keeping pages open in the other banks that may still be needed for likely caching advantages**
- **with multiple banks, we don't always need to close an open page to go to another page that is needed now**
- **use your imagination !**
- **and ... old-style external memory banks can still also be used**
- **note: memory banks are not needed for address memory interleaving (like in the old days) since DRAMs perform interleaved burst transfers (sequential and non-sequential) already**

error detection and/or correction (1)

- **due to their relatively high soft bit error rate, DRAMs generally have either error detection or error detection with some correction**
- **error detection of 1 or 2 bits per word is common**
- **1 bit error detection is done easily using parity**
- **2 bit detection and 1 bit correction ("SECDED") requires a Hamming distance of at least 4 between each valid code word**
- **SECDED is done using an Error Correcting Code (ECC) that commonly uses E extra bits per memory word, where E is roughly computed :**

$$E = S + 1$$
$$S = \log_2 (W + E)$$

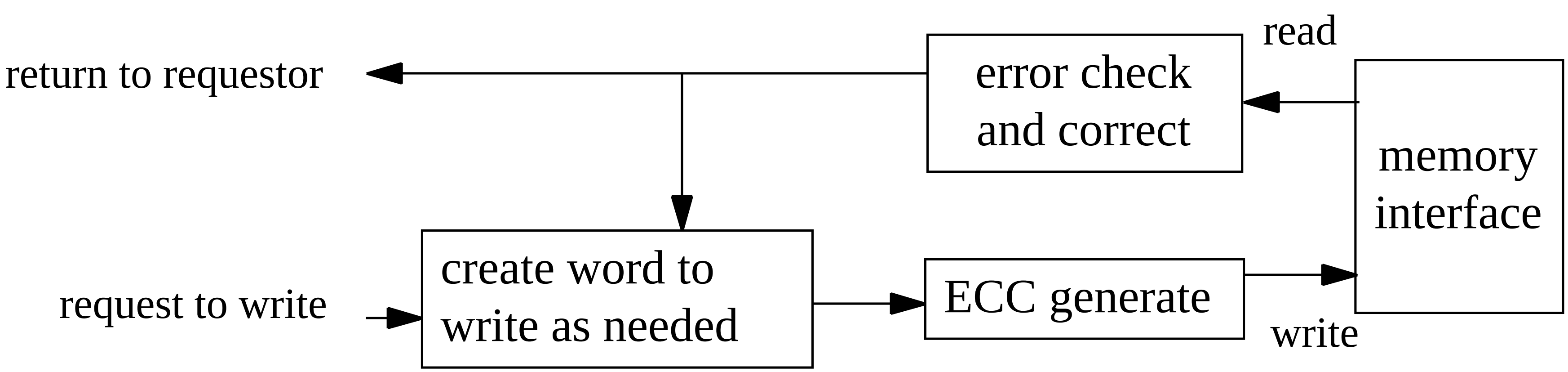
- **solved iteratively and rounded up to an integer**
- **W is the number of architected bits in the memory word**
- **S bits are used to form the "syndrome" to find the single bit in error**
- **the extra single bit (E - S) differentiates a single from a double bit error**
- **other codes, and codes that correct up to 2 errors are also used**

error detection and/or correction (2)

- **extra bits needed for given memory word lengths to implement SECDED :**
  - **16 6**
  - **32 7**
  - **64 8**
  - **128 9**
  - **256 10**
- **memory reads of either the full or partial memory word :**
  - **read the whole memory word (with the ECC bits) and check for error**
- **on a double bit error, signal the error back to the processor**
  - **will cause either the process that initiated the error to get killed**
  - **or cause the whole OS to panic if the error was in the OS code or data**
- **on a single bit error (correctable), we correct the error (flip the bit that is in error) and then write the corrected value back to memory**
  - **this would be a write after a read**

error detection and/or correction (3)

- **memory writes of the full memory word width need to :**
  - **— computer the ECC code for the architected data being written**
  - **— write the architected and the ECC bits to the memory word**
- **memory writes of less than the full memory word width need to :**
  - **— read the memory word, replace the part of it with what is being written**
  - **— proceed as above**
  - **— this operation is a read followed by a write to the same memory word**



error detection and/or correction (4)

- **what if a memory word is rarely read ? a single bit error (correctable) can get compounded by an additional error causing a double bit error (uncorrectable) !! this is a bad thing !**
- **solution: periodically, we will read all of memory and check it for any errors**
  - **$ cat /dev/mem > /dev/null**
    - (don't laugh -- this was actually done on many UNIX based systems)
  - **the memory system hardware can do the periodic reads (and make any associated corrections)**
  - **this is called "memory scrubbing" (we are periodically "scrubbing" the memory clean) !**
- **more recent OSes may have paging algorithms that guarantee that all pages are periodically read by some means or another**
- **note that "scrubbing" corrects soft errors in the memory and can be applied to either DRAM or SRAM**
- **in contrast "refresh" is the process of retaining the memory bits in DRAM**

cache - main memory interaction

- **the processor (executing your programs) likes to perform many single memory reads and single memory writes**
- **but DRAM has a very high cost (in terms of a very long cycle time) for doing single reads and writes**
- **however, the bandwidth of accessing DRAM can dramtically increase if we can do burst reads and burst writes !!**
- **solution: have the cache that is just above the DRAM in the memory heirarchy convert single reads and writes from the processor to burst reads and writes to the DRAM**
- **for reads :**
  - the cache will fill whole cache blocks (4, 8, 16, 32, ? words) on a single read miss using a burst read to the DRAM
  - arrange for the access that was the "miss" to be returned first
- **for writes :**
  - the cache will use a write strategy called "write back" to accumulate several single processor writes into a single burst write to the DRAM

some key points

- — SRAM is faster and more costly per bit than DRAM
- — SRAM is easier to control and can be integrated with the processor without substantial performance compromise
- — SRAM is generally used for cache memory
- — DRAM is slower but higher density than SRAM (about 4 times higher)
- — DRAM is used for main memory
- — DRAM needs to be refreshed
- — DRAM has a higher soft bit error rate than SRAM
- — design with DRAM often uses an Error Correcting Code (ECC)
- — DRAM has a long cycle time for doing single or short transfers
- — we try to use burst transfers with DRAM to get higher bandwidth
- — we try to create burst transfers from the way we manage the cache

review questions (1)

- **memory technology**
  — how does the bit density compare between SRAM and DRAM ?
  — which technology (SRAM or DRAM) is faster at any given time ?
  — how do they compare on cost-per-bit ?
  — which is better for integration on the same IC as the processor ?
  — what does "dynamic" mean in DRAM ? what does "static" mean ?
  — which is easier to control ?
  — which technology has higher bit error rates ?

review questions (2)

- **system design**
  - — why is a memory hierarchy used in computer design ?
  - — where do each technology (SRAM or DRAM) better fit into an overall memory hierarchy ? (which might be higher or lower in the hierarchy)
  - — what are some ways to create burst read sequences from single reads from the core processor ?
  - — what are some ways to create burst write sequences from single write requests from the core processor ?
  - — what sort of maintenance does each memory technology (SRAM or DRAM) require, if any ?
  - — what is a refresh operation ?
  - — what is an Error Correcting Code (ECC) ?
  - — what is a scrub operation ?
  - — what is the difference between refreshing and scrubbing ?