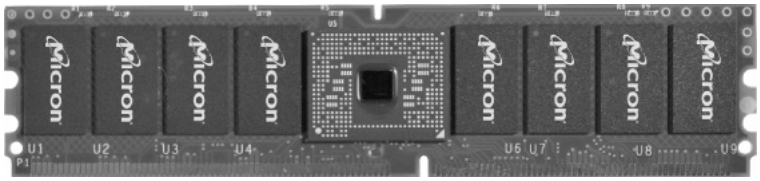
DRAM
Memory Access Protocols
develop generic model for thinking about timing

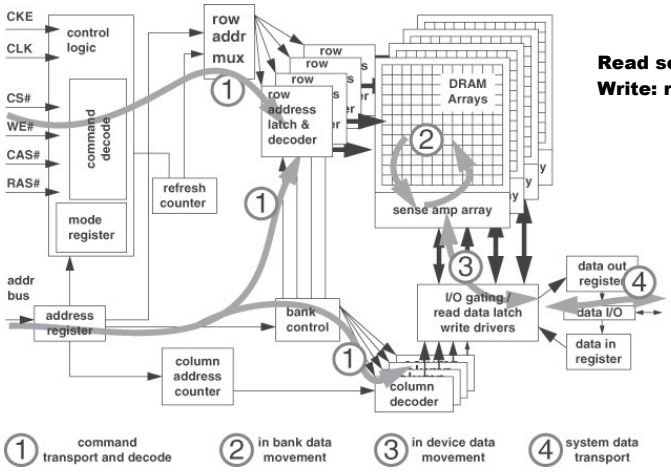


Reference: "Memory Systems: Cache, DRAM, Disk" & Micron website

Bruce Jacob, Spencer Ng, & David Wang

Today's material & any uncredited diagram came from chapter 11

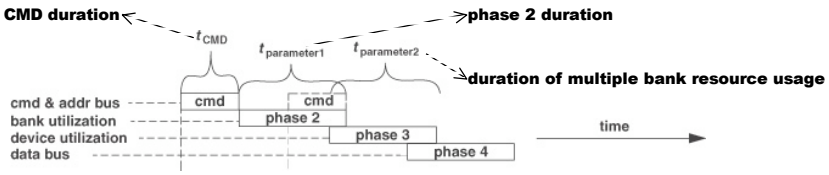
Generic Structure



Read sequence
Write: reverse 2,3,4

Abstract Command Structure

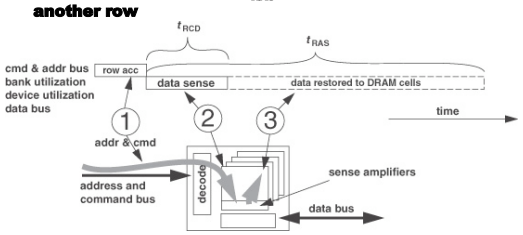
- **Reality**
  - **huge variety of command sequences possible**
    - » **all with heavily constrained timing issues**
      - **2 roles of timing**
        - **1) physical: latency, set-up and hold, signal integrity, lane retiming**
        - **2) power: limit concurrency to stay under thermal/power ceiling**
- **Start simple**
  - **command & phase overlap**



Note other overlaps - also specified by timing parameters

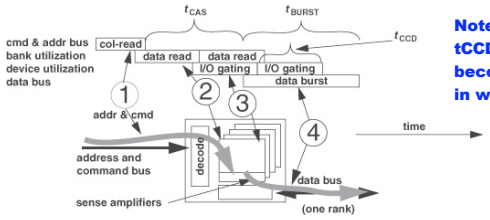
Row Access Command

- **Row activation**
  - **move data from the mats to sense amps and restore the mats**
    - » **controlled by 2 timing parameters**
      - **t_{RCD} - row command delay**
        - **time to move the data from the mats to the sense amps**
        - **after a RAS command + t_{RCD}: column reads or writes can commence**
      - **t_{RAS} - interval between a RAS command and row restore**
        - **after a RAS command + t_{RAS} sense amps can be precharged to activate another row**



Column Read Command

- **Bank specific**
  - **move data from sense amps through I/O's to the Mem_Ctlr**
    - » **3 timing parameters**
      - **t_{CAS} (or t_{CL}) - column address strobe**
        - **time between col-rd (CAS) command and data valid on the data bus**
        - **DDRx devices do this in short continuous bursts**
      - **t_{CCD} - minimum column to column command delay due to burst I/O gating**
        - **1 cycle for DDR, 2 cycles for DDR2, 4 cycles for DDR3, etc.**
      - **t_{BURST} - duration of the data burst on the bus**

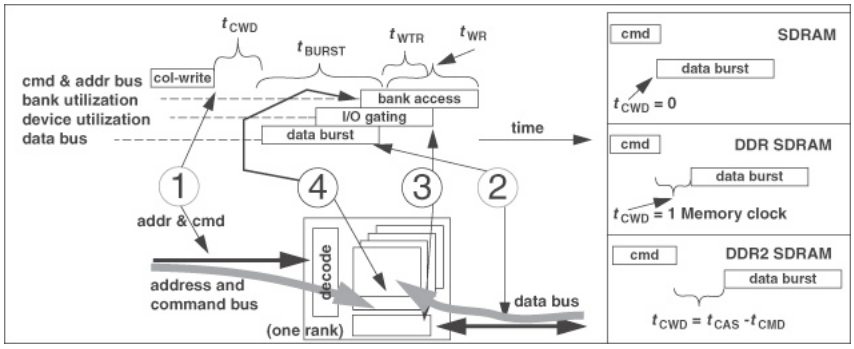


Note: some devices have tCCD>tBurst where tCCD becomes the limiting factor in what can happen next

Column Write Command

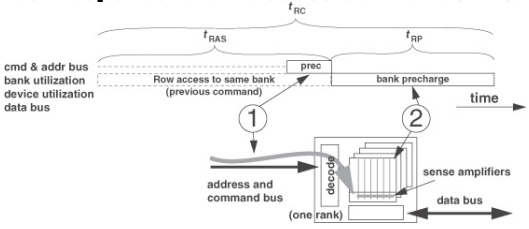
- **Move data from mem_ctrl to sense amps**
  - **timing parameters**
    - » **t_{CWD} - delay between col-write and data valid on bus from mem-ctrlr**
      - **some per device differences differences**
        - **SDRAM: t_{CWD} is typically 0**
        - **DDR - typically 1 memory clock cycle**
        - **DDR2 - t_{CAS} - 1 cycle**
        - **DDR3 - t_{CWD} is programmable**
    - » **Other parameters control a subsequent command's timing**
      - **t_{WTR} - write to read delay**
        - **end of write data burst to column read command delay**
      - **t_{WR} - write recovery delay**
        - **min. interval between end of a write data burst and start of a precharge command**
        - **I/O gating allowed to overdrive sense amps prior to col-rd-cmdn (mat restore)**
      - **t_{CMD} - time command occupies command bus**

Column Write Overview



Precharge Command

- **Basic sequence**
  - **precharge --> RAS --> (CAS R/W)* -- precharge ....**
- **Timing constraints**
  - **t_{RP} - row precharge delay**
    - » **time delay between precharge and row access command**
  - **t_{RC} - row cycle time**
    - » **$t_{RC} = t_{RAS}+t_{RP}$**
    - » **limits independent row access commands in same bank**

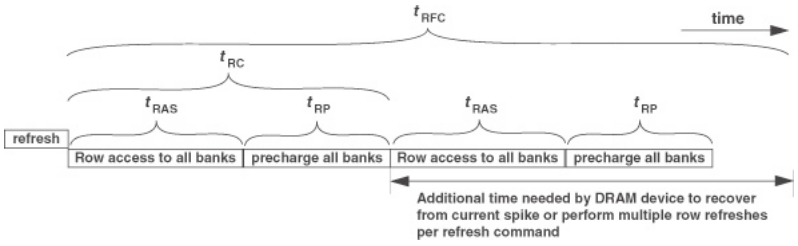


Refresh

- Necessary evil of 1T1C DRAM density advantage
  - +: density improves $/bit
    - » but the T is not a perfect switch due to leakage
  - -: parasitic
    - » power, bandwidth, and resource availability
- Refresh approach varies
  - options exist to reduce 1 of the parasitic effects
    - » total refresh power will be constant
      - reduced peak power of the device has some options
  - typical
    - » concurrent row precharge in all of the device's banks
      - mem_ctlr issues periodic refresh commands
      - most devices contain row precharge address counter
        - holds addr. of last precharged row
      - t_{RFC} - refresh cycle time
        - duration between refresh commands and an activation (RAS) command

Refresh Overview

- Typical refresh model is block refresh
  - refresh entire device all at once
    - » avoids trying to be smart & associated control complexity
    - » refresh counter wraps to 0 to indicate done



Refresh Trends

- t_{RFC} is going up
  - decreases availability ==> slower system memory
  - vendor choice
    - » keep inside the 64 ms refresh period
      - even though the number of rows goes up

Family	Vdd	Device Capacity Mb	# Banks	# Rows	Row Size kB	Refresh Count	t_{RC} ns	t_{RFC} ns
DDR	2.5V	256	4	8192	1	8192	60	67
		512	4	8192	2	8192	55	70
DDR2	1.8V	256	4	8192	1	8192	55	75
		512	4	16384	1	8192	55	105
		1024	8	16384	1	8192	54	127.5
		2048	8	32768	1	8192	~	197.5
		4096	8	65536	1	8192	~	327.5

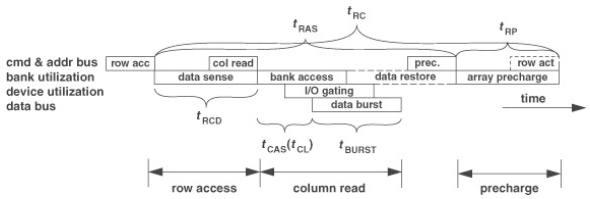
Other Refresh Options

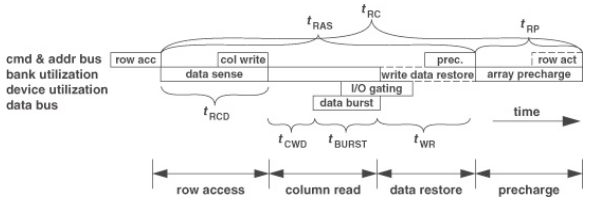
- All have control overhead
  - usually pushed to memory controller
    - » since device vendors need to minimize $/bit
      - device could do it
        - – classic cost-performance dilemma
- Separate bank refresh
  - allow a bank to be refreshed
    - » while other bank accesses are still allowed
      - bandwidth win since memory bus can still be active
      - peak power win since 1 RAS on command bus at a time
      - mem_ctlr schedule gets harder
  - next step
    - » only refresh what is going to expire
      - huge scheduling problem - probably too hard

Effects of Variable Command Sequences

- **Significant performance variation**
- **Best case**
  - **read everything in a row and move to next row**
    - » **1-2 kB in a row - lots of energy expended**
      - **pass 64-128 B cache-lines to the mem_ctlr**
      - **access all 8-32 cache lines before opening another row in same bank**
        - **low probability**
        - **observed trend: as core # increases, $ lines/row approaches 1**
  - ***open page* memory systems - typical**
    - » **keep row buffer open hoping for the best**
      - **w/ additional energy cost**
- **Worst case**
  - **Precharge -> RAS -> single CAS –> precharge ....**
  - ***closed page* memory systems**
    - » **expect the worst but why not make the row smaller?**

Read and Write Sequences

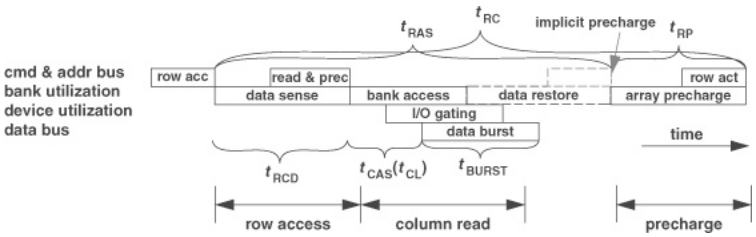




Note: % of time data bus bandwidth is utilized

Compound Commands

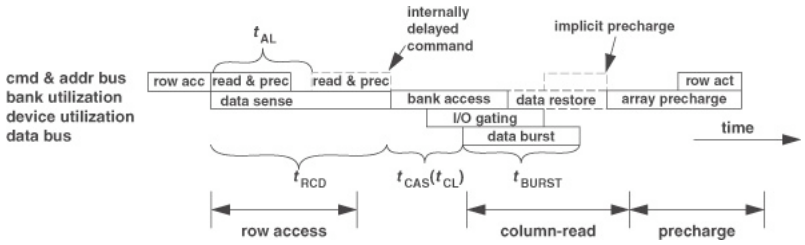
- **DRAM evolution**
  - **allows compound commands**
    - » **mem_ctlr options and scheduling complexity increase**
  - **column read and precharge**
    - » **use when next scheduled access is to a new row**
      - **2 commands rather than 3**
      - **timing constraints carried over however**



Other DDR2 Trends

- **t_{RAS} lockout**
  - **internal timer to make sure t_{RAS} isn't violated**
    - » **if col_rd_pch issues before data restore complete**
      - **device delays the implicit precharge command**
    - » **allows closed page systems to issue col_rd_pch w/ optimistic timing**
      - **mem_ctlr doesn't need to worry about precharge of random row access**
- **Posted CAS**
  - **CAS issued but delayed (posted) by t_{AL} cycles**
    - » **t_{AL} - added latency to column access**
      - **programmed into the device**
      - **usually once via initialization commands**
  - **XDR does same thing via CAS tag**
    - » **logs of mem_ctlr flexibility and complexity hides in this one**
    - » **1 simplification**
      - **MC can issue posted CAS immediately after read**
      - **t_{AL} is set to respect the other timing constraints once**

JEDEC Posted CAS



Other Considerations

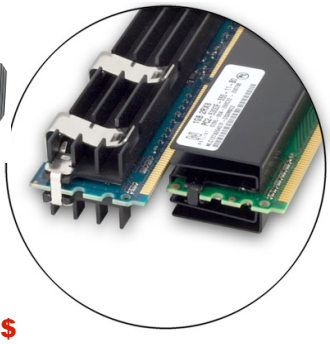
- **Until now**
  - view based on resource utilization & single bank timing constraints
- **Reality**
  - multi-bank DRAM devices & multi-rank DIMMs
    - » allows much higher resource utilization via pipelining
    - » but package (DRAM die & DIMM) limitations exist
      - peak current limited
        - remember the small pin count
      - thermal constraints
        - how many banks can remain active
  - enter package based timing parameters
    - » t_{FAW} - four bank activation window
      - time that 4 banks can be active (DDR2 and DDR3)
    - » t_{RRD} - row activate to row activate delay for any DRAM device
      - limits peak current profile
- **Combine to impact minimum scheduling times**

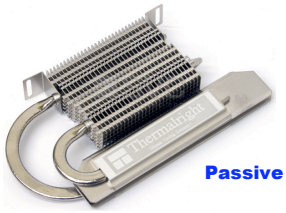
Hot DRAMs & Packaging

source: random web photos

Heat spreaders: DDR 1st step

Fins and Fans: DDR2 and beyond



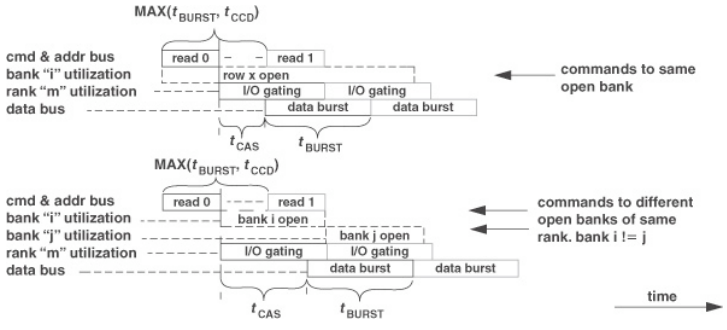


Passive heat pipes

$$$ Ka-ching $$$

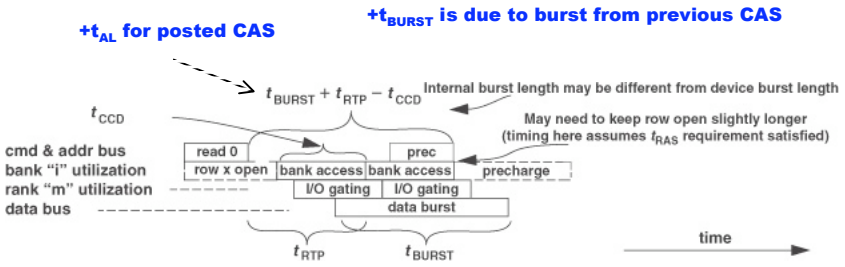
Pipelining Reads

- Typically $t_{BURST} > t_{CCD}$
  - except DDR3 where t_{CCD} = 4 cycles
    - » so general form is to pick the maximum



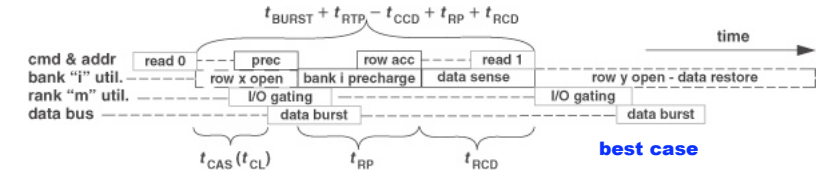
Read to Precharge Timing

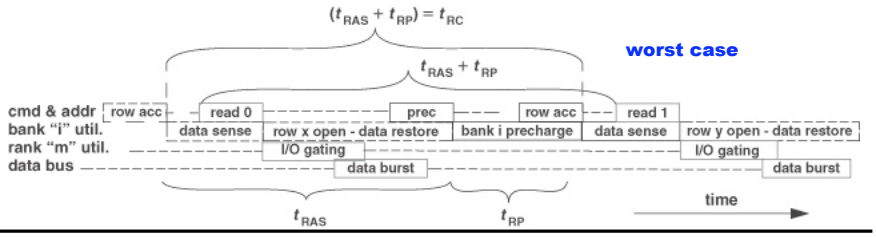
- **Burst may consist of multiple internal bursts**
  - **Interleaved or phased mat returns for bandwidth improvement**
  - **t_{RTP} - read to precharge command interval**
    - » **more general: tRTP+(N-1)*tCCD for N internal bursts**
  - **sense amps kept open to drive multiple internal bursts through the I/O circuitry**



Consecutive Reads

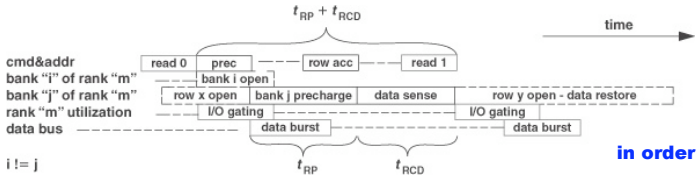
- **Different rows same bank**
  - **best case: tRAS elapsed and mats have been restored**
  - **worst case: have to wait for tRAS to complete data restore phase**





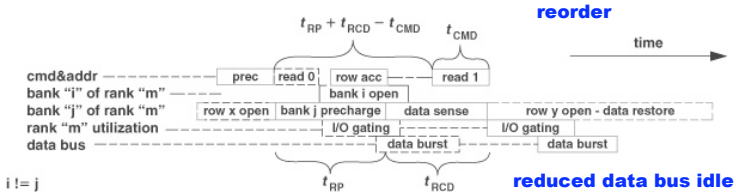
Bank Read Conflict

- **Consecutive reads to different banks in same rank w/ conflict**
  - **2nd read to an inactive row**
  - **improvement if commands can be reordered by mem_ctlr**



in order

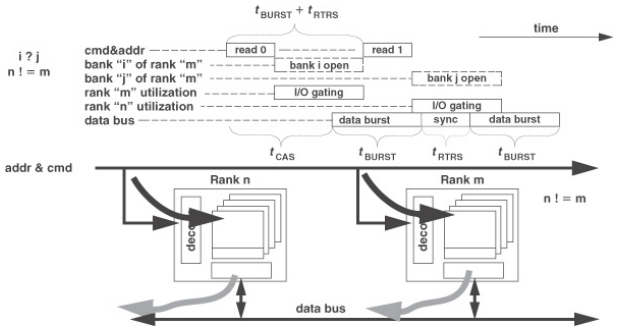
reorder



reduced data bus idle time

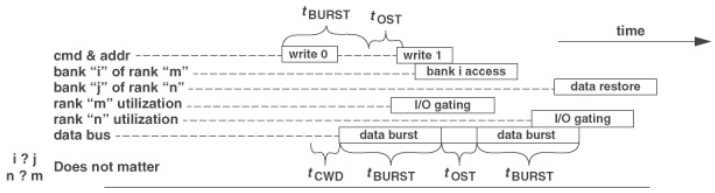
Consecutive Reads to Different Ranks

- **Pipeline option more restricted than to same rank different bank command sequences**
  - **depends on**
    - » **system level synchronization issues & DRAM operating rate**
      - **synchronization is very tricky due to bit-lane jitter and varying trace lengths**
    - » **t_{RTRS} - rank to rank switching time due to resynchronization issues**



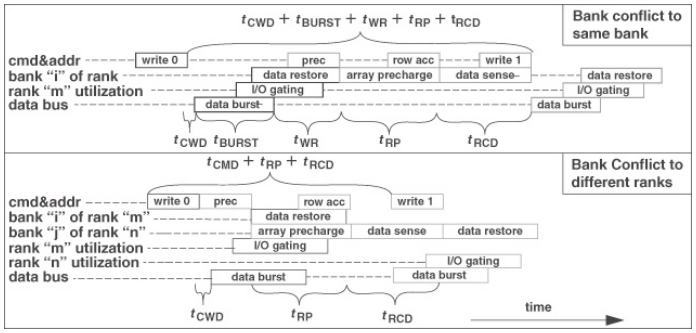
Consecutive Writes

- **Different ranks but to open banks**
  - **may be pipelined depending on the bus termination strategy**
    - » **data in this case comes from a common source**
      - **hence no need to give up control of shared bus**
    - » **termination strategy gets hairier as x gets larger in DDRx**
      - **on die termination (ODT) for DDR2**
        - **problem: 2 cycles to turn on ODT and 2.5 cycles to turn it off**
    - » **t_{OST} - time it takes to switch ODT from rank to rank**
      - **in reality this applies to reads as well but typically $t_{RTRS} > t_{OST}$**



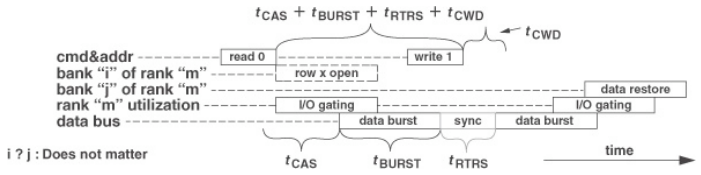
Consecutive Writes

- **Bank conflict - 2nd write to an inactive row**
  - **same rank - bigger delay due to t_{BURST} and t_{WR}**
  - **different rank - more overlap**
  - **for now best case assume t_{RAS} has been satisfied**



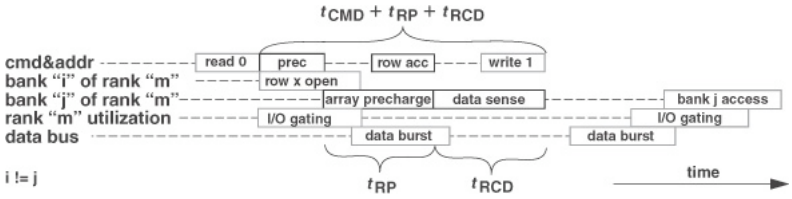
Write After Read: Open Banks

- Pipelining possible if requests are to open banks
  - timing control is primarily restricted by burst length
    - » no new timing parameter for this one - phew!
    - » different banks allows tighter packing
      - since no new row needs to be precharged & data restore time is overlapped
    - » note this case can have a lot of variance in different DRAM technologies



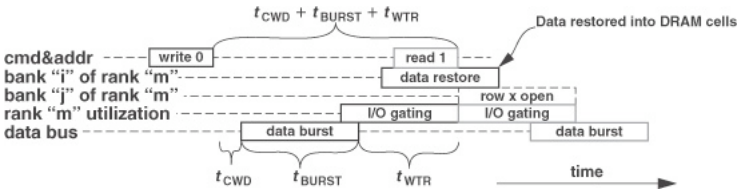
Write After Read: Bank Conflict

- Different banks, bank conflict, no reordering
  - best case for data already restored in old open row
    - » e.g. time > t_{RAS} has passed



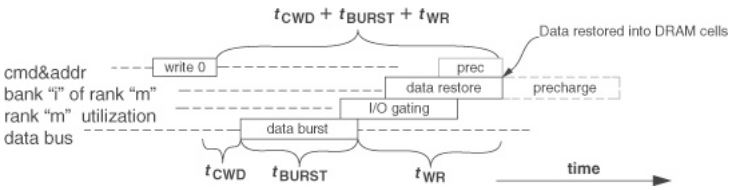
Read After Write

- **Same rank, open banks**
  - **main issue is the reversal of the data flow**
    - » **RAW vs. WAR**
      - **write first is worse since data restore time is needed**
        - **hence RDRAM uses write buffers to improve performance**
        - **allows I/O gating to be used by another command**
        - **effectively allows HW support for dynamic command reordering**
      - **controlled by t_{WTR} constraint**
    - » **shared I/O gating happens in both cases but with different timing restrictions**



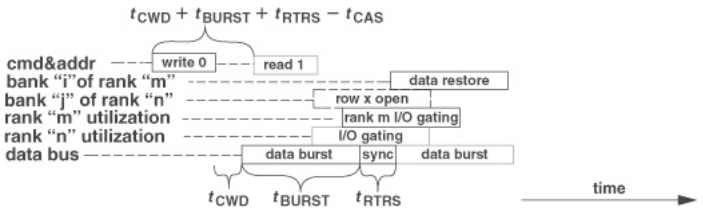
Write to Precharge Timing

- **Subtle difference**
  - **write to read timing vs write to precharge**
  - **due to I/O mux gating time needed to drive the data into the sense amps**
    - » **hence write to precharge must additionally wait for the data to be restored in the mats**



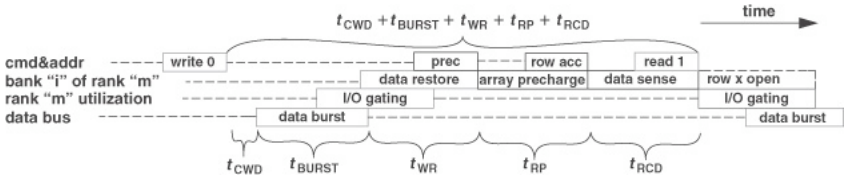
Read After Write

- **Different ranks, open banks (no bank conflict)**
  - **data movement change timing issue doesn't apply to the different rank case**
    - » **but rank switching synchronization time does come into play**



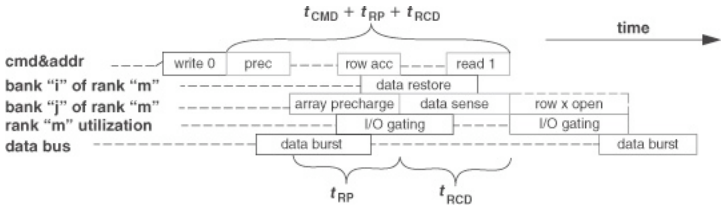
Read After Write

- **Bank conflict this time**
  - **assumes t_{RAS} (data restore) time has already elapsed**
  - **write recovery time must be respected**
  - **NOTE: if there was a write buffer**
    - » **then a write commit command would be necessary**
    - » **OR retrieve from write buffer which is not currently being done**
      - **It's that density and cost/bit thing again**



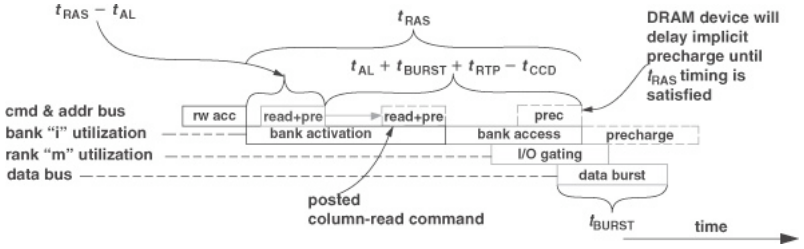
Read After Write

- **Same rank, bank conflict, no reordering**
  - **plus best case - data restore complete**
- **Issues**
  - **re-ordering will help**
  - **many relative timing constraints in play**
    - » **I/O gating is critical in this case**
    - » **min scheduling time is:**
      - **max ($t_{CMD}+t_{RP}+t_{RCD}$, $t_{CWD}+t_{BURST}+t_{WR}$)**



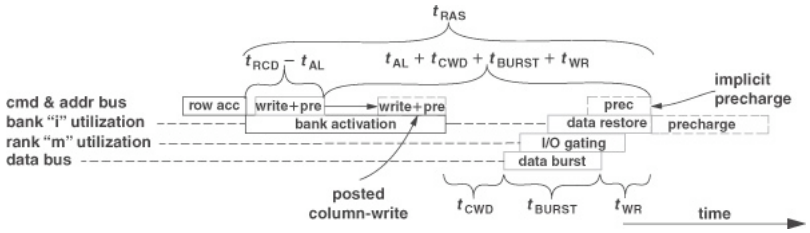
Col_Rd_&_Precharge Command

- **previously**
  - **precharge after column read minimum timing**
    - » **$t_{BURST}+t_{RTP}-t_{CCD}$**
    - » **$+t_{AL}$ if it's a posted read**
  - **unified read and precharge command would be the same**
    - » **but there is an issue of respecting t_{RAS} data restore time**
    - » **DDR2 has additional support to delay precharge to insure t_{RAS} req's**



Col_Wr_Precharge Command

- Tricky - well what isn't with DRAM?
  - t_{RAS} could be defined to include reads and writes
    - » this is the case here but not necessarily true in general
    - » depends on how complicated you want the mem_ctlr to be
      - BEWARE - how t_{RAS} is defined for the components you actually target



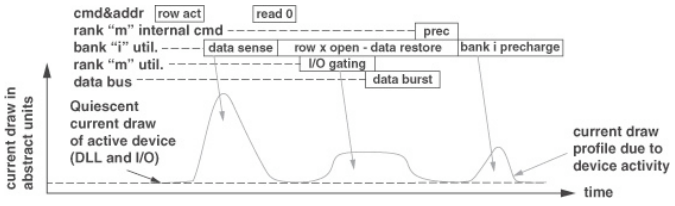
Additional Constraints

- Power - it's the biggest problem as things get "better?"
- Rules
  - first rule - things must work
  - second rule - things must get faster
  - third rule - devices must protect themselves
    - » Intel learned this the hard way
    - » for DRAM this is enforced via timing constraints
- Row activation in the main culprit
  - K's of bits moved to the sense amp latches
    - » question is how much of them do you use
      - multi-core land indicates a cache line
        - for large num's of cores
- Remember
  - large current profile changes
    - » cause timing delays
      - bit lane jitter depends on Vdd
      - Ohm's law V = I/R
        - not just a good idea - it's the law

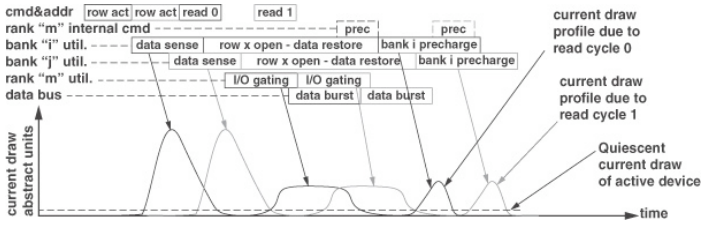
Double Edged Sword

- **Active power**
  - $P_a = aCV^2f$
- **non-adiabatic chargine regime**
  - **~.5P given off as heat**
    - » **the other half is returned to the power supply**
    - » **Vdd variations on the power lines are an issue**
      - **also supply tolerance to high variance loads is a design issue**
        - **requires over provisioning**
  - **higher temps increase passive P component**
- **Faster is better**
  - **except for power since both f and a go up**
    - » **hence so does P and leakage**
      - **leakage impacts resource availability**
      - **can't ignore refresh and the 64 ms standard target**

Power Profiles

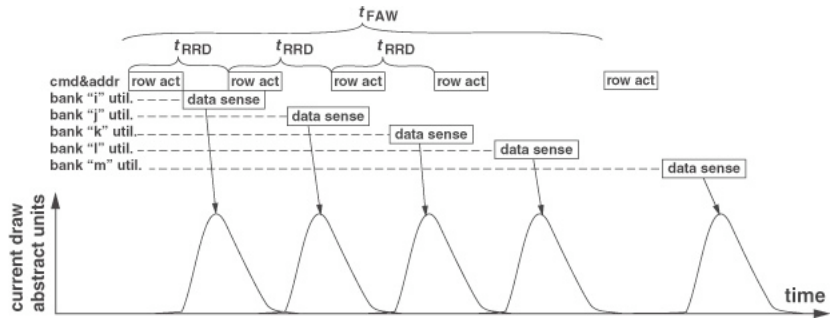


simple read



pipelined read

Hence Delay 5th Row Activate



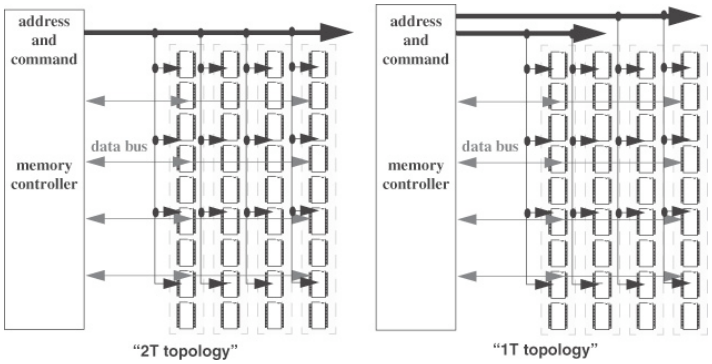
Enter Power Driven Timing Parameters

- Limit row activation - t_{RRD}
- # of active banks
  - conflict between performance and power
  - current limit is 4 bank activation window t_{FAW}
- both get worse as device width goes up

Micron			
Device Configuration	512 Mb x 4	256 Mb x 8	128 Mb x 16
Bus width	4	8	16
Bank count	8	8	8
Row Count	16384	16384	8196
Column Count	2048	1024	1024
Row Size	8192	8192	16384
tRRD ns	7.5	7.5	10
tFAW	37.5	37.5	50

Partitioned Address and Command Bus

- **Alleviates variable trace length to some extent**
  - **binary tree partition**
    - » **doesn't work for DDR2 and DDR3**
    - » **but similar to BoB**



Summary Timing Parameters

Parameter	Description
tAL	added latency to column accesses for posted CAS
tBURST	data burst duration on the data bus
tCAS	interval between CAS and start of data return
tCCD	column command delay - determined by internal burst timing
tCMD	time command is on bus from MC to device
tCWD	column write delay, CAS write to write data on the bus from the MC
tFAW	rolling temporal window for how long four banks can remain active
tOST	interval to switch ODT control from rank to rank
tRAS	row access command to data restore interval
tRC	interval between accesses to different rows in same bank = tRAS+tRP
tRCD	interval between row access and data ready at sense amps
tRFC	interval between refresh and activation commands
tRP	interval for DRAM array to be precharged for another row access
tRRD	interval between two row activation commands to same DRAM device
tRTP	interval between a read and a precharge command
tRTRS	rank to rank switching time
tWR	write recovery time - interval between end of write data burst and a precharge command
tWTR	interval between end of write data burst and start of a column read command

Summary Minimal Timing Equations

A=row access
R=col_rd
W=col_wr
P=precharge
F=Refresh
s=same
d=different
a=any

Prev	Next	Rank	Bank	Min. Timing	Notes
A	A	s	s	tRC	
A	A	s	d	tRRD	plus tFAW for 5th RAS same rank
P	A	s	d	tRP	
F	A	s	s	tRFC	
A	R	s	s	tRCD-tAL	tAL=0 unless posted CAS
R	R	s	a	Max(tBURST, tCCD)	tBURST of previous CAS, same rank
R	R	d	a	tBURST+tRTRS	tBURST prev. CAS diff. rank
W	R	s	a	tCWD+tBURST+tWTR	tBURST prev CASW same rank
W	R	d	a	tCWD+tBURST+tRTRS-tCAS	tBURST prev CASW diff rank
A	W	s	s	tRCD-tAL	
R	W	a	a	tCAS+tBURST+tRTRS-tCWD	tBURST prev. CAS any rank
W	W	s	a	Max(tBURST, tCCD)	tBURST prev CASW same rank
W	W	d	a	tBURST+tOST	tBURST prev CASW diff rank
A	P	s	s	tRAS	
R	P	s	s	tAL+tBURST+ tRTP-tCCD	tBURST of previous CAS, same rank
W	P	s	s	tAL+tCWD+tBURST+tWR	tBURST prev CASW same rank
F	F	s	a	tRFC	
P	F	s	a	tRFC	

Projects

- Note this is an abstracted view
  - individual devices may vary and do
    - » particularly RAMBUS
  - project ideas
    - » #1
      - specify the things that count for a particular technology
      - and then write a mem_ctlr that respects these constraints
      - It's all about scheduling
    - » #2
      - simulate RAMBUS vs. JEDEC for various workloads
      - pick your technology step
    - » #3
      - evolution is not always good
      - compare memory performance SDRAM, DDR, DDR2, DDR3 …
        - use basic timing model but with device specific values