CS/ECE 6780/5780

Al Davis

Today's topics:

- **External memory**
  - overview of options
  - brief SRAM introduction
  - more detailed DRAM intro
    - mostly we'll care about DRAM chips
    - some DIMM coverage for added scope

Adding Memory Capacity

- **Inherent microcontroller problem**
  - limited amount of RAM & FLASH
    - » code tends to not change and be rather small
      - pick microcontroller w/ enough FLASH to hold your code
    - » so far your data has also been small and fits in RAM
    - » what happens if you have lots of data
      - common issue with data acquisition systems (DAS Chap. 12)
- **External memory choices**
  - SRAM – fast and easy to interface
    - » problems: expensive, power hungry, volatile
  - DRAM – lots of cheap bits
    - » problems: difficult interface, volatile
  - NVRAM (NV=nonvolatile)
    - » e.g. FLASH but other technologies exist that likely will replace FLASH
    - » pro's: cheap & non-volatile, low power if low usage
    - » con's: interface difficulty varies with technology

One More

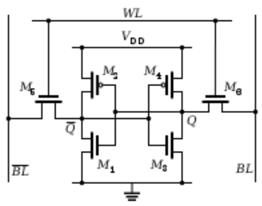
- **Disk**
  - **not really a reasonable embedded choice**
    - » fragile & power hungry
  - **note SSD's are basically big FLASH array's today**
    - » packaged in a disk interface like SATA
- **One notable exception**
  - **iPod "Classic"**
    - » although iPod shuffles use FLASH

Current Plan

- **Today**
  - **brief intro to SRAM**
  - **more details on DRAM**
- **Next Tuesday**
  - **broad view of the NVRAM playing field**
    - » as it looks today
      - note that things are changing rapidly
- **Next Thursday**
  - **survey of cool gizmo's**
    - » sensors, actuators, etc.
    - » namely things you may want to use to surround your microcontroller
- **Finish with midterm and 6780 project demo's**
  - **demo's will be in the appropriate Wed & Friday lab sessions**
    - » 5780 students in that lab session are expected to attend
      - of course all are welcome in both sessions if you're available.

SRAMs

- **Typically based on a 6T cell**

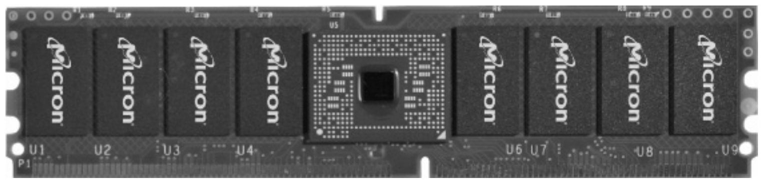


  - **cross coupled inverters hold value as long as there is power**
  - **Read – precharge BL's to *mid-voltage*, activate WL**
    - » **sense amps detect swing and value is latched to the output**
  - **Write drive BL's to the rail, activate WL**
    - » **note BL drivers must overpower cell transistors**
      - **so transistor sizing is critical to proper operation**

SRAM Interfaces

- **Simple model**
  - **read**
    - » **apply address and assert R**
    - » **wait for access delay and capture valid read data**
  - **write**
    - » **apply address, write data, and assert W'**
    - » **hold address and write data for appropriate hold time**
      - **and the data is written**
- **Modern SRAM's are more complicated**
  - **pipeline & burst models are common**
  - **interface is still fairly simple but timing changes a little bit**
    - » **for details see the Samsung data sheet on the class web site**
      - **K7A401809A: 256Kx18 synchronous SRAM**
      - **K7A403609A: 128Kx36 variant**
  - **compared to DRAM's this timing is quite straightforward**

DRAM: Overview & Devices



Reference: "Memory Systems: Cache, DRAM, Disk"

Bruce Jacob, Spencer Ng, & David Wang

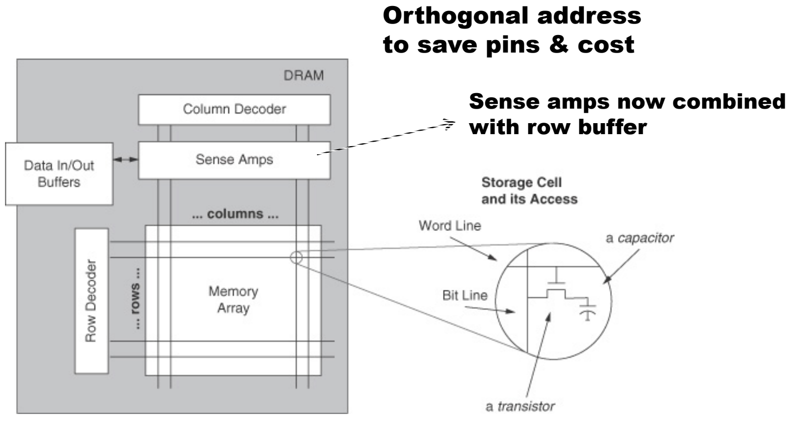
uncredited diagrams came from this book

NOTE: this is an expensive but very detailed treatment of both memory and storage systems.

Key Items to Remember

- It is easy to predict SRAM behavior
  - even though discrete SRAM may well disappear in this decade
    - » cache buses (BSBs) are extinct now
    - » too power hungry & expensive for ES designs
- Hard to predict DRAM behavior
  - probabilistic resource availability
    - » due to refresh requirement
  - performance depends on controller and device model
    - » small controller differences show up as big performance differences

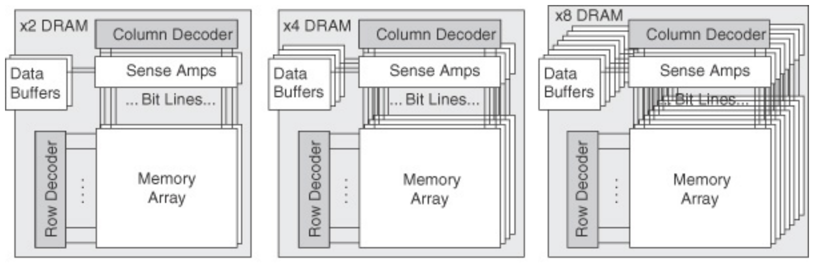
Simplified DRAM



It's All about Mats

- **DRAM devices come in several flavors**
  - **interface & speed: we'll deal with these later**
  - **width**
    - » **x4 & x8 are highest density die**
      - **used in price sensitive applications**
      - **best match for microcontroller systems due to narrow datapath**
    - » **x16 & x32**
      - **higher per bit cost used in high performance systems**
- **DRAM chip = lot's of memory arrays (mats)**
  - **mats operate under several regimes**
    - » **unison**
      - **each access targets one bit/mat**
        - – **x4 accesses 4 mats**
    - » **independent**
      - **mats organized as subsets to create banks**
        - – **concurrent bank access is the idea**
      - **intra-bank mats operate in unison**
    - » **interleaved banks**

Mat & Width Organization

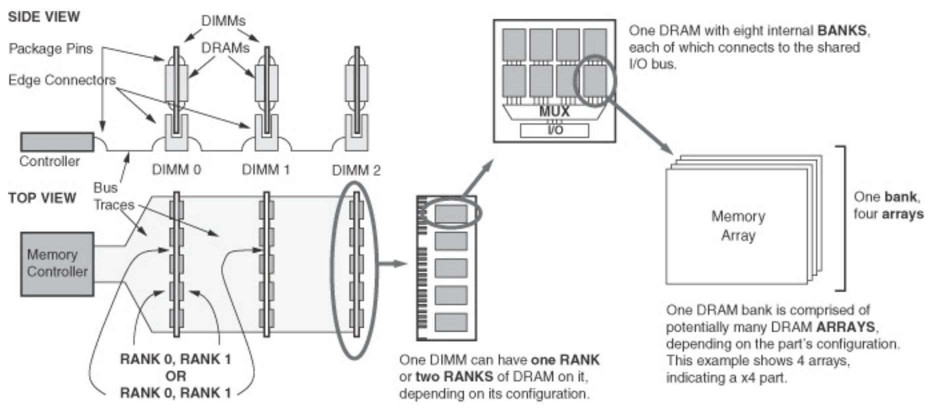


x2 DRAM x4 DRAM x8 DRAM

Slow Mat Problem

- **Mat access is slow**
  - **high-C word and bit lines**
    - » **bigger = slower**
      - **C for wire is linear in length at same width**
      - **Cgate is linear with size of row or column in the mat**
- **Interleave to speed up**
  - **mid-60's hack used on IBM 360/91 and Seymour's CDC 6600**
    - » **essentially a form of pipelining**
  - **if interface is n times faster than mat latency interleave n banks**
    - » **should be able to make things arbitrarily fast**
      - **in theory yes - in practice no**
        - – **constraints: jitter, signal integrity, power**
  - **multiple on-die banks**
    - » **may be internally or externally controlled**

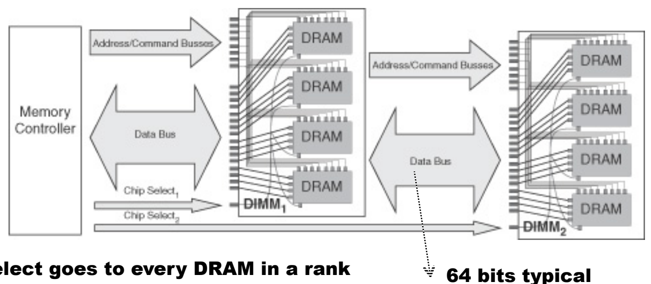
Ranks & Banks vs. DRAMs & DIMMs


SIDE VIEW
Package Pins
Edge Connectors
DIMMs
DRAMs
Controller
DIMM 0
DIMM 1
DIMM 2
TOP VIEW
Bus Traces
Memory Controller
RANK 0, RANK 1
OR
RANK 0, RANK 1
One DIMM can have one RANK or two RANKS of DRAM on it, depending on its configuration.
MUX
I/O
One DRAM with eight internal BANKS, each of which connects to the shared I/O bus.
Memory Array
One bank, four arrays
One DRAM bank is comprised of potentially many DRAM ARRAYS, depending on the part's configuration. This example shows 4 arrays, indicating a x4 part.

Note: ES will more likely be DRAM chip rather than DIMM

JEDEC Channel Interface

address width depends on DRAM capacity
control: RAS, CAS, Oenable, CLKenable, etc.



Chip select goes to every DRAM in a rank
Separate select per rank - 2 per DIMM common

64 bits typical
wider in high-end systems

See any problems on the horizon with this model?

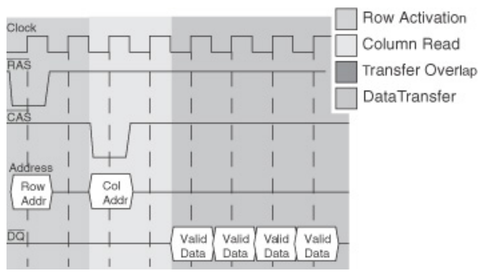
Memory Controller Issues

- DRAM control is tricky
  - CPU prioritizes memory accesses
    - » transaction requests send to Mem_Ctl
  - Mem_Ctl
    - » translates transaction into the appropriately timed command sequence
      - transactions are different
        - – open bank then it's just a CAS
        - – no open bank then Activate, PRE, RAS, CAS
        - – wrong open bank then write-back and then ACT, PRE, RAS, CAS
        - – lots of timing issues
      - result: latency varies
        - – often the command sequence can be stalled or even restarted
        - – refresh controller always wins
    - » now moving onto the CPU die
      - multi-core and multi-mem_ctl involves a lot of issues
- Fortunately microcontroller access need not be as hairy
  - treat it like an SRAM but with weirder timing restrictions

DRAM Evolution

- Not that important
  - naming conventions vary by vendor to some extent
    - » Clocked - treat DRAM as a really slow SRAM
    - » Asynch DRAM - access and wait
      - still clocked but the timing provided by the command lines
    - » Fast Page Mode
      - add latches to the sense amps to form row buffer
    - » EDO
      - add latches to output drivers so data stays valid
    - » P/BEDO
      - add counter to cycle through successive width sized nibbles
    - » SDRAM - mid 90's - the bulk of the action now
      - clock now controls row select circuits as well
      - DDRx variants still SDRAM just higher bandwidth

Simple SDRAM Timing



Note: pipelining possibilities

Mainstream Throughput Idea: DDRx

- **Use both clock edges**
  - **DDR transfers 2 bits per cycle per lane**
    - » **DDR2 transfers 4**
    - » **DDRn transfers 2^n**
    - » **signal integrity and power limit clock speeds**
      - **particularly on long FR4 wire traces**
- **Also add source synchronous clocking - enter DQS**
  - **timing variance creates synchronization issues**
    - » **DDR device uses DLL/PLL to synch with Mem_Ctl master clock**
      - **note skew depends on where the DIMM sits in the chain**
    - » **need to latch in the center of the data "eye"**
  - **other sources of timing uncertainty**
    - » **manufacturing variation, temperature, Miller side-wall effect, trace length**
      - **delay proportional to RC**
      - **power proportional to CV^2**
- **There have been some latency improvements as well**
  - **unimportant for the embedded system context**
    - » **so we'll skip these**

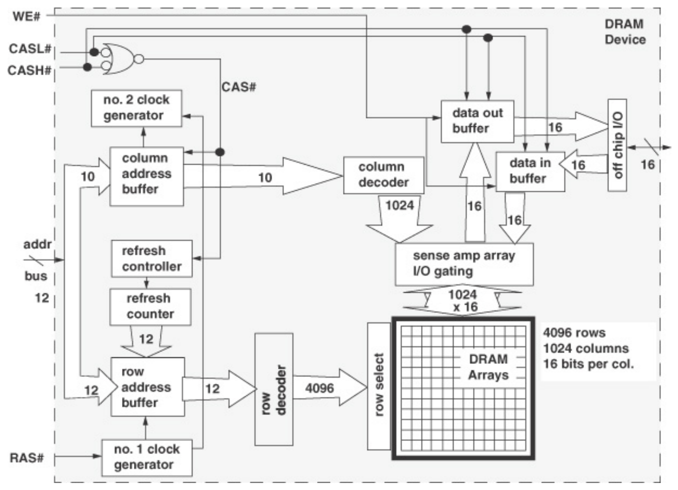
DRAM Systems Issues

- Architecture and scaling
  - DDRn causes 2^n prefetching
    - » I/O side faster but mat side is wider
- Timing fundamentally limited by signal integrity issues
  - lots can be done here but impact is cost/bit increase
- Pins vs. protocol
  - pin count has large cost adder
  - use them more efficiently ==> protocol change
    - » JEDEC moves slowly
- Power and Heat
  - the biggest concern now and in the future most likely
    - » early DIMMs consumed about 1W
      - ES use of one DRAM chip however
      - power down modes and low utilization mitigate the problem
      - refresh is still a culprit
      - 100 mW ES operation is realizable

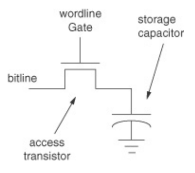
Slight Change of Focus

- Very brief device technology overview
  - background to help understand protocol issues
- Key issues
  - leaky devices
  - process differences
  - refresh requirements
  - how to build that pesky capacitor

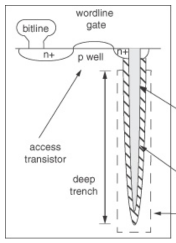
64 Mbit FPM DRAM (4096x1024x16)



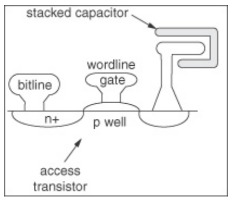
DRAM Cell



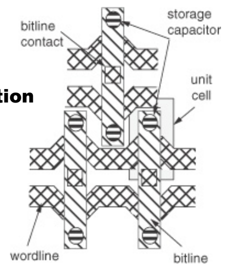
Logical View



Trench implementation now primarily used in eDRAM



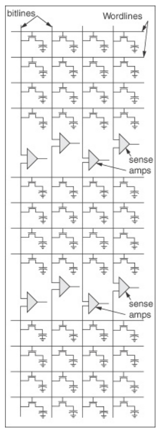
stacked implementation mainstream DRAM processes

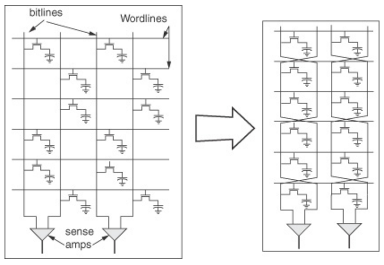


Leakage & Refresh

- Transistors are not ideal switches
  - leakage currents in DRAM processes are minimized
    - » but not to 0
  - leakage currents increase as Tsize goes down
    - » tricky balance of Vth, Vdd, and process
    - » additional increase with temperature
  - industry target - refresh every 32 - 64 ms

Folded vs. Open Bit-Line





Folded: 8F² - 2 bit line per cell
6F² version shipped by Micron using MIM (metal insulator metal C) in 2004

Open: 6F² - 1 bit line per cell

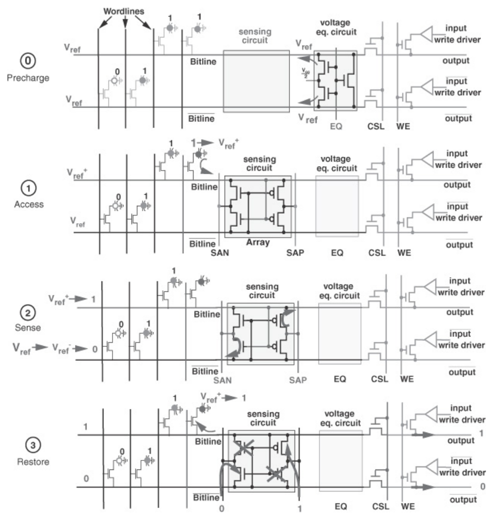
Issues

- Open
  - requires dummy array segments at mat edge
    - » balance C characteristics of bit-line pairs
  - more noise susceptibility
  - combine to dilute the cell size advantage
- Folded
  - differential sense amps have better common-mode noise rejection properties
    - » e.g. alpha particle or neutron spike shows up on both sides
  - current industry focus
    - » new folding strategies show up regularly in circuit venues

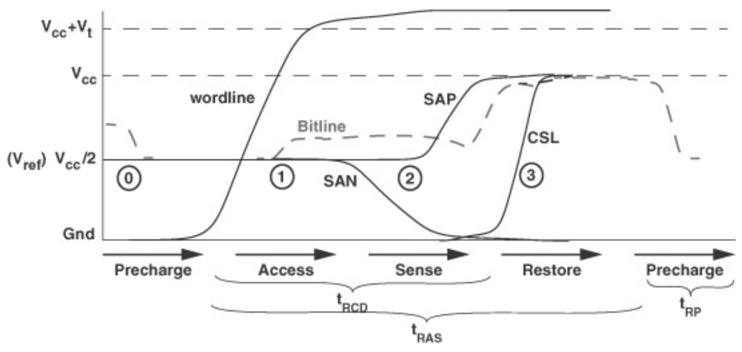
Sense Amps

- Small stored charge requires high sensitive amps
  - use differential model
    - » reference voltage precharged to half-way mark
    - » then look at which way the charge goes to determine value
      - noise margins must exist and trick is to keep them small
      - problematic as devices shrink
- Roles
  - 1: basic sense value
  - 2: restore due to the destructive read
    - » 2 variants in play
      - restore instantly or restore on row close
  - 3: act as a temporary storage element (row buffer)
    - » how temporary depends on restore choice

Sense Amp Operation



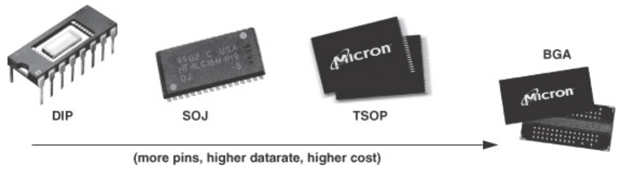
Sense Amp Waveforms



Decoders & Redundancy

- Defects occur and yields have to be high
  - rules of a low margin business
- Redundant rows, columns, and decoders
  - fuses are used to isolate defective components
  - appearance is of a fully functional mat
  - fuse set
    - » burn in, test and then fuse set

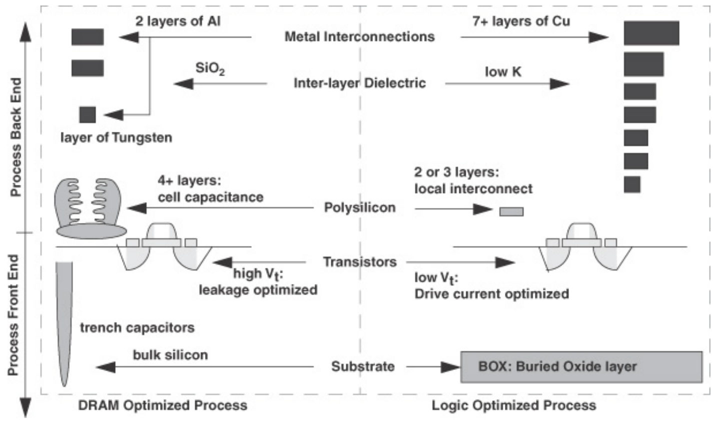
Packaging, Performance, Cost



ITRS 2002	2004	2007	2010	2013	2016
process (nm)	90	65	45	32	22
CPU pin count	2263	3012	4009	5335	7100
cents/pin	1.88	1.61	1.68	1.44	1.22
DRAM pin count	48-160	48-160	62-208	81-270	105-351
cents/pin	0.34-1.39	0.27-0.84	0.22-0.34	0.19-0.39	0.19-0.33

Pressure runs wild!!

DRAM vs. Logic Process



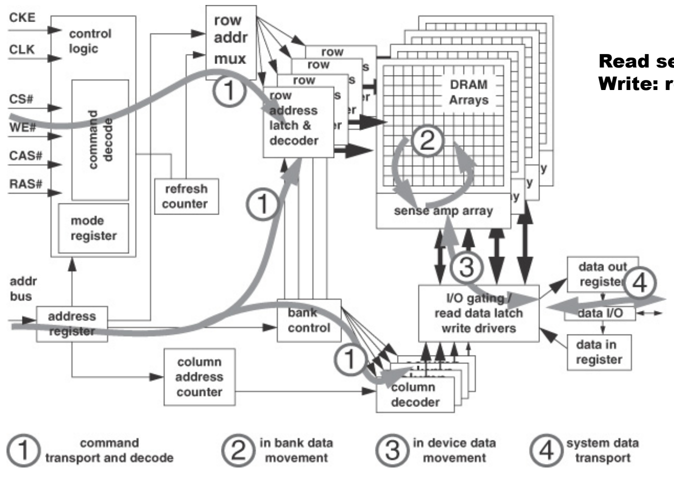
Hybrid Processes Coming

- **IBM was the pioneer**
  - **start with logic process**
  - **add extra layers to create high-C DRAM cells**
    - » **multiple oxide thicknesses**
      - **fast leaky transistors**
      - **slow less-leaky transistors**
    - » **enables eDRAM**
    - » **also helps with power issues**
      - **leakage is a big deal**
      - **only use fast transistors on the critical CPU path**
      - **use slow T's for non-critical path and memory blocks**
- **Current usage in transition**
  - **from high-performance SoC's to mainstream CPU**
    - » **issues do become more tricky as feature size shrinks**
    - » **but power is the nemesis so you do what you have to**

DIMMs and DRAMs

DRAM chip type	DIMM Stick Type	Bus Clock Rate (MHz)	Memory Clock Rate (MHz)	Channel Bandwidth (GB/s)	non-ECC Channel Width	ECC Channel Width	Prefetch Buffer Width	Vdd	Read Latency Typical (bus cycles)	DIMM pins
DDR-200	PC-1600	100	100	1.6	64	72	2	2.5	2-3	184
DDR-266	PC-2100	133	133	2.133	64	72	2	2.5	2-3	184
DDR-333	PC-2700	167	167	2.667	64	72	2	2.5	2-3	184
DDR-400	PC3200	200	200	3.2	64	72	2	2.5	2-3	184
DDR2-400	PC2-3200	100	200	3.2	64	72	4	1.8	3-9	240
DDR2-533	PC2-4200	133	266	4.267	64	72	4	1.8	3-9	240
DDR2-667	PC2-5300	167	333	5.333	64	72	4	1.8	3-9	240
DDR2-800	PC2-6400	200	400	6.4	64	72	4	1.8	3-9	240
DDR3-800	PC3-6400	100	400	6.4	64	72	8	1.5	?	240
DDR3-1066	PC3-8500	133	533	8.53	64	72	8	1.5	?	240
DDR3-1333	PC3-10600	167	667	10.67	64	72	8	1.5	?	240
DDR3-1600	PC3-17000	200	1066	18.06	64	72	8	1.5	?	240

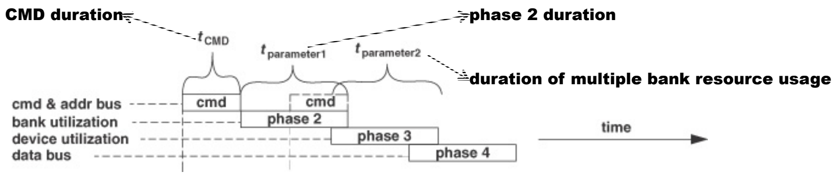
Generic Protocol Structure



Read sequence
Write: reverse 2,3,4

Abstract Command Structure

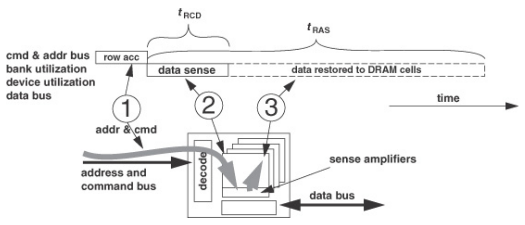
- **Reality**
  - **huge variety of command sequences possible**
    - » **all with heavily constrained timing issues**
      - **2 roles of timing**
        - **1) physical: latency, set-up and hold, signal integrity, lane retiming**
        - **2) power: limit concurrency to stay under thermal/power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



Note other overlaps - also specified by timing parameters

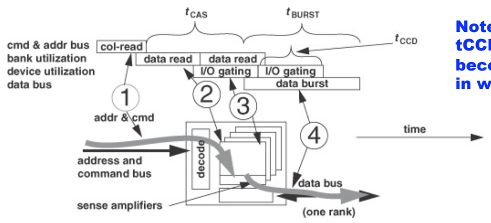
Row Access Command

- **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**

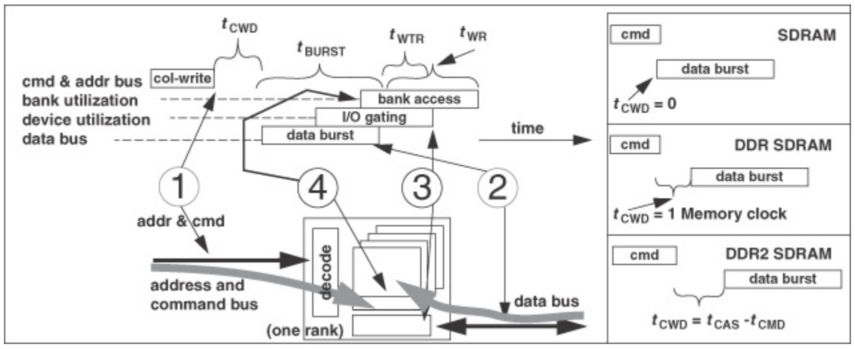


Note: some devices have tCCD>tBurst where tCCD becomes the limiting factor in what can happen next

Column Write Command

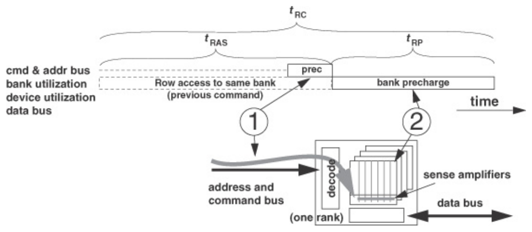
- **Move data from mem_ctrl to sense amps**
  - **timing parameters**
    - » **t_{CWD} - delay between col-write and data valid on bus from mem-ctrlr**
      - **some per device differences differences**
        - – SDRAM: t_{CWD} is typically 0
        - – DDR - typically 1 memory clock cycle
        - – 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**

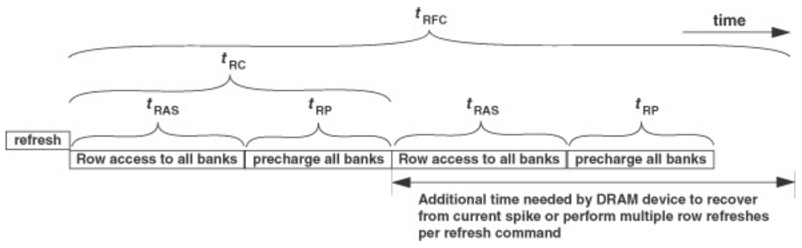


Refresh

- 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

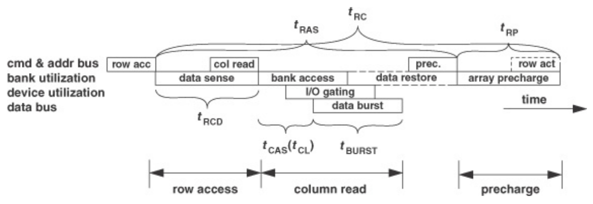
Other Refresh Options

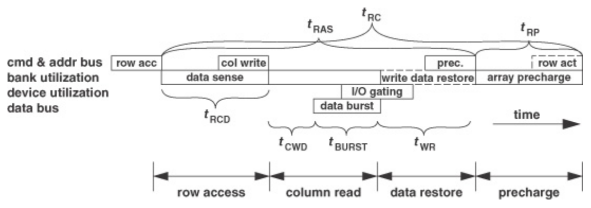
- All have control overhead
  - usually pushed to memory controller
    - » since device vendors need to minimize $/bit
      - device could do it
        - – classic cost-performance dilemma
- Separate bank refresh
  - allow a bank to be refreshed
    - » while other bank accesses are still allowed
      - 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

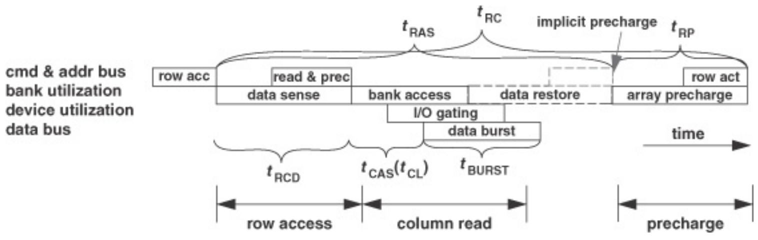




Note: % of time data bus bandwidth is utilized

Compound Commands

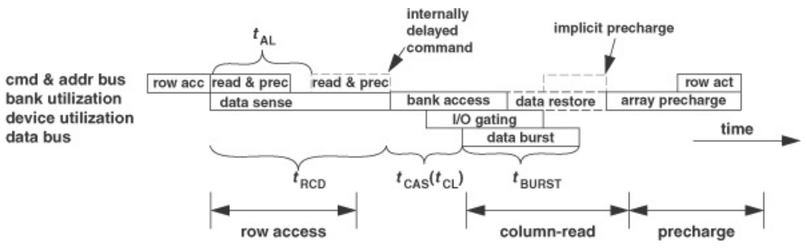
- **DRAM evolution**
  - **allows compound commands**
    - » **mem_ctlr options and scheduling complexity increase**
  - **column read and precharge**
    - » **use when next scheduled access is to a new row**
      - **2 commands rather than 3**
      - **timing constraints carried over however**



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**

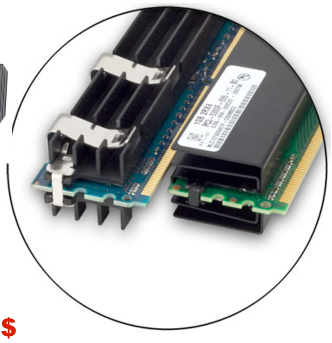
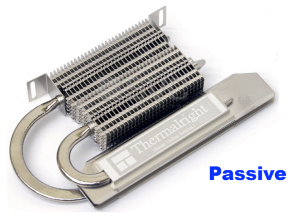
Hot DRAMs & Packaging (for fun)

source: random web photos

You won't be using these in ES designs

Heat spreaders: DDR 1st step

Fins and Fans: DDR2 and beyond

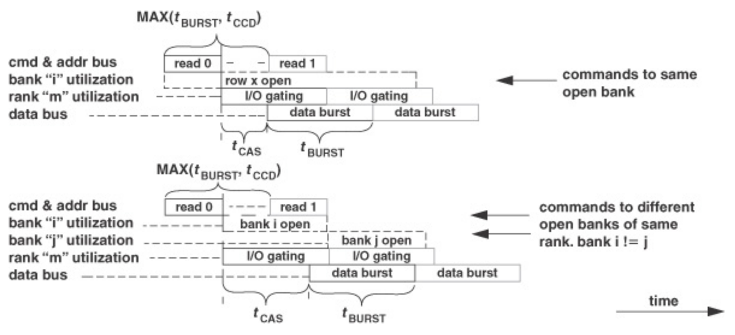



Passive heat pipes

$$$ Ka-ching $$$

Pipelining Reads

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

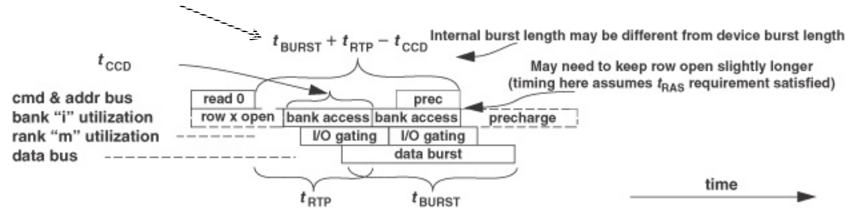


Read to Precharge Timing

- **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**

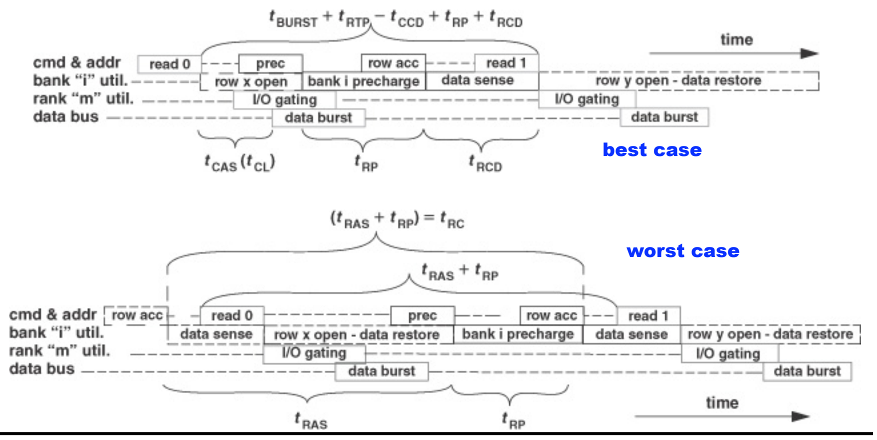
$+t_{AL}$ for posted CAS

$+t_{BURST}$ is due to burst from previous CAS


$t_{BURST} + t_{RTP} - t_{CCD}$
Internal burst length may be different from device burst length
t_{CCD}
May need to keep row open slightly longer
(timing here assumes t_{RAS} requirement satisfied)
cmd & addr bus
bank "i" utilization
rank "m" utilization
data bus
read 0
prec
row x open
bank access
bank access
precharge
I/O gating
I/O gating
data burst
t_{RTP}
t_{BURST}
time

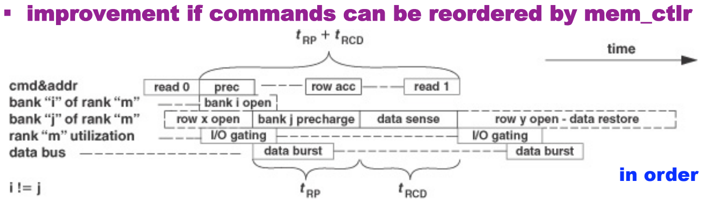
Consecutive Reads

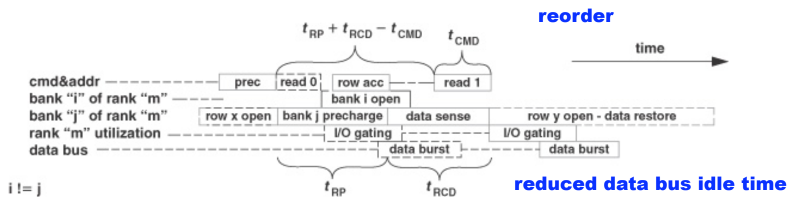
- **Different rows same bank**
  - **best case: tRAS elapsed and mats have been restored**
  - **worst case: have to wait for tRAS to complete data restore phase**



Bank Read Conflict

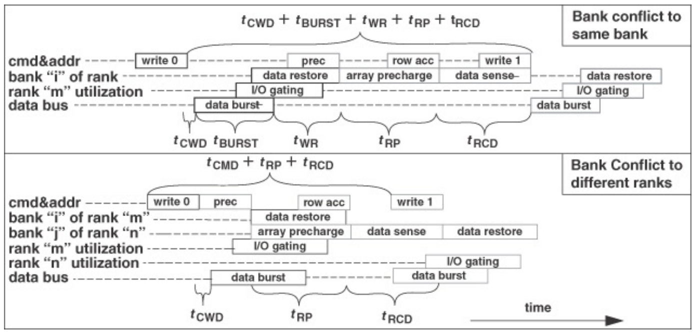
- **Consecutive reads to different banks**
  - **2nd read to an inactive row (for single DRAM chip ignore rank issues)**
  - **improvement if commands can be reordered by mem_ctlr**





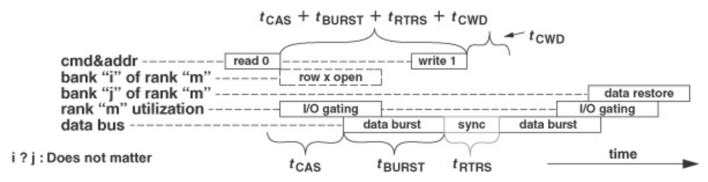
Consecutive Writes

- **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**



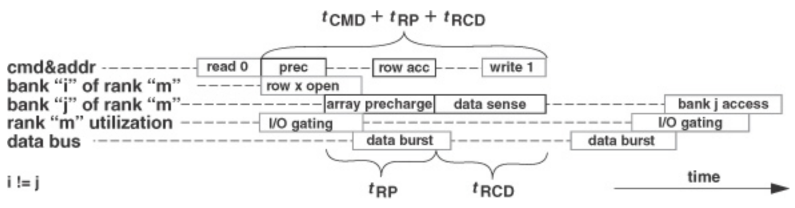
Write After Read: Open Banks

- 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



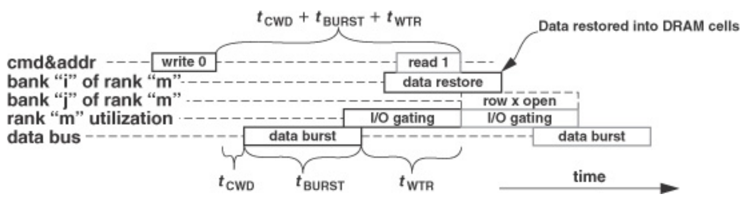
Write After Read: Bank Conflict

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



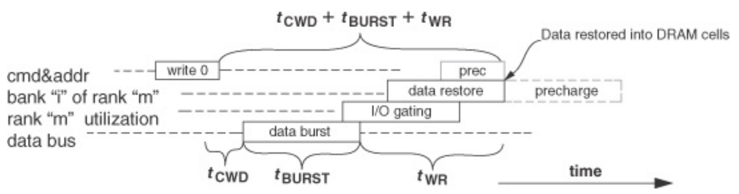
Read After Write

- **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**



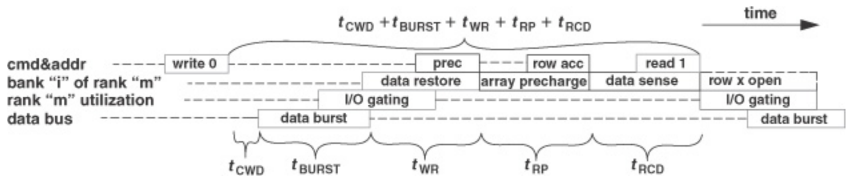
Write to Precharge Timing

- **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**



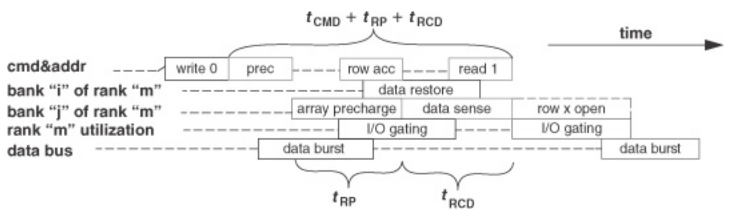
Read After Write

- **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**



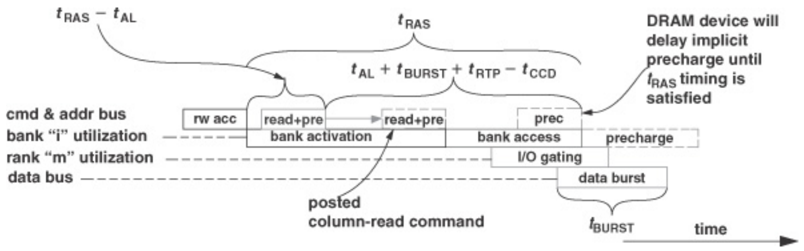
Read After Write

- **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 ($t_{CMD}+t_{RP}+t_{RCD}$, $t_{CWD}+t_{BURST}+t_{WR}$)**



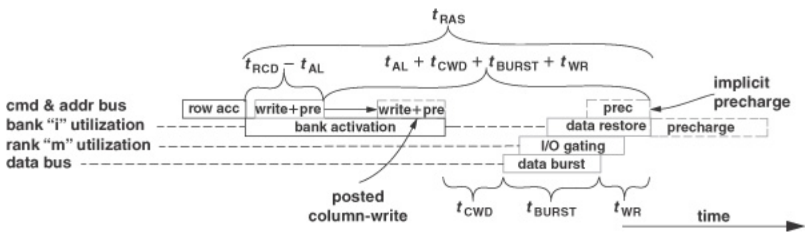
Col_Rd_&_Precharge Command

- 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



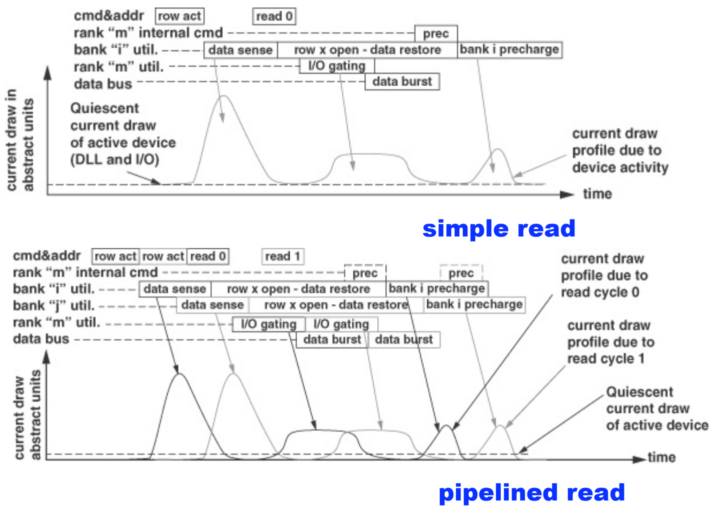
Additional Constraints

- 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 *"overfetch"* 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
  - good thing if there is a lot of locality
    - » this is likely the common ES case
- 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

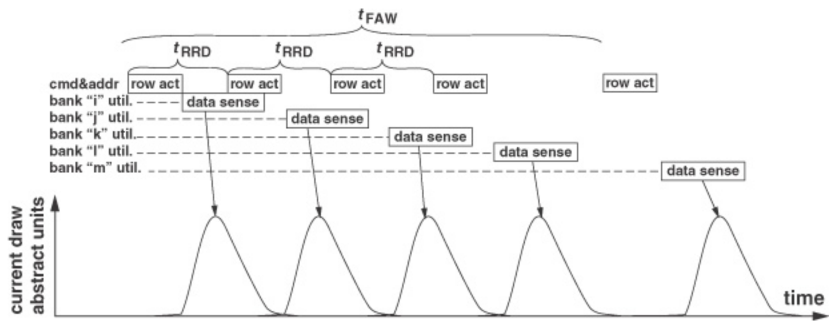
Double Edged Sword

- 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



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

School of Computing
University of Utah

CS 5780

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

School of Computing
University of Utah

CS 5780

Summary Minimal Timing Equations

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

This is what you really care about

For single DRAM chip – ignore rank column

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	

Concluding Remarks

- Whirlwind introduction
  - point is that there are a lot of tricking timing constraints that have to be understood to achieve maximum DRAM throughput
- Fortunately
  - for microcontroller interfaces
    - » it's usually possible to abstract most of the hair away
    - » since you usually just want to use a DRAM chip as a way to store a lot of data
      - high locality
      - simplified command structure if you treat it much like an SRAM
  - so most of this hair can be ignored
    - » slow microcontroller → slow access rate
  - for reference
    - » typical DRAM datasheet from Micron is on the class web site