DRAM

Today's topics:
Brief look at DRAM devices
Channel protocols & signalling
Memory controller issues
This is just a skim – CS7810 will have a more in depth treatment of lots of topics including this one

Plan Preview

- So far focus has been on-chip
  * processors, caches
  * and a bit of interconnect
  * brief look at parallel processing on 1 or more sockets
- Note that many big applications are I/O or memory bound
  * last 3 lectures before the 2nd midterm
    » DRAM – this one
      » 2 standards
        - JEDEC – DDRn – focus on this one – 64 bit slower data bus
        - Rambus – RDRAM – fast skinny bus
      » pace will be rapid – high level understanding is the goal
    » Disk and storage
      » non-volatile RAM (e.g. not disk)
    » goal is to give you a high level understanding
      » details are way too complicated to cover in 1.5 weeks

Key Item to Remember

- It is easy to predict SRAM behavior
  * even though discrete SRAM may well disappear in this decade
    » since cache buses (BSBs) are now extinct
- Hard to predict DRAM behavior
  * probabilistic resource availability
  * performance depends on controller and device model
    » small controller differences show up as big performance differences
  * access pattern has an even bigger effect than with caches
    » primarily since DRAM accesses are so much slower
- Disk performance is probabilistic as well
- Plus
  * lots of intermediate buffers, prefetch, ... issues
Typical PC

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

Basic DRAM Idea: bit/"mat"

Orthogonal address to save pins & cost

Sense amps now combined with row buffer
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 like PC's
    - x16 & x32
    - higher per bit cost used in high performance systems
- DRAM chip = lot's of memory arrays (mats)
  - mats operate under several regimes
    - unified
      - each access targets one bit/mat
    - independent
      - mats organized as subsets to create banks
      - concurrent bank access is the idea
      - intra-bank mats operate in unison
      - interfaced banks within a rank

Mat & Width Organization

- DRAM chip = lot's of memory arrays (mats)
  - mats operate under several regimes
    - unified
      - each access targets one bit/mat
    - independent
      - mats organized as subsets to create banks
      - concurrent bank access is the idea
      - intra-bank mats operate in unison
      - interfaced banks within a rank

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
    - Gate 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
        - restrictions: jitter, signal integrity, power
    - multiple on-die banks
      - may be internally or externally controlled

64 Mbit FPM DRAM (4096x1024x16)
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

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 a fully functional mat
  - fuse set
    - burn in, test and then fuse set

Sense Amp Operation

Sense Amp Waveforms
**DRAM Cell**

- Trench implementation now primarily used in eDRAM
- Stacked implementation in mainstream DRAM processes

**Ranks & Banks vs. DRAMs & DIMMs**

- JEDEC Interface
  - Address width depends on DRAM capacity
  - Control: RAS, CAS, D, Enable, CLK, 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

**JEDEC Interface**

- 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 CAS just a CAS
      - Open bank then Activate, PME, RAS, CAS
      - Wrong open bank then write-back and then ACT, PME, RAR, CAS
      - Lots of timing issues
    - Result: Latency varies
      - Sometimes 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 involve a lot of issues
  - Not as easy as you might guess if you want performance
  - Lots of device specific timing constraints
Simple SDRAM Timing

Note: pipelining possibilities

Timing Parameters (Micron Style)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tRC</td>
<td>interval between accesses to different rows in same bank = tRAS+tRP</td>
</tr>
<tr>
<td>tRRD</td>
<td>interval between two row activation commands to same SDRAM device</td>
</tr>
<tr>
<td>tRTP</td>
<td>interval between a read and a precharge command</td>
</tr>
<tr>
<td>tRFR</td>
<td>interval between refresh and activation commands</td>
</tr>
<tr>
<td>tRFC</td>
<td>interval between refresh and activation commands</td>
</tr>
<tr>
<td>tRAS</td>
<td>row access command to data restore interval</td>
</tr>
<tr>
<td>tRRTC</td>
<td>rank to rank switching time</td>
</tr>
<tr>
<td>tWR</td>
<td>write recovery time - interval between end of write data burst and a precharge command</td>
</tr>
<tr>
<td>tWTR</td>
<td>interval between end of write data burst and start of a column read command</td>
</tr>
<tr>
<td>tCWD</td>
<td>column write delay, CAS write to write data on the bus from the MC</td>
</tr>
<tr>
<td>tCMD</td>
<td>time command is on bus from MC to device</td>
</tr>
<tr>
<td>tCCD</td>
<td>column command delay - determined by internal burst timing</td>
</tr>
<tr>
<td>tBURST</td>
<td>data burst duration on the data bus</td>
</tr>
<tr>
<td>tAL</td>
<td>added latency to column accesses for posted CAS</td>
</tr>
</tbody>
</table>

Minimal Timing Equations

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

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

Mainstream Throughput Idea: DDRx

- Use both clock edges
  - DDR transfers 2 bits per cycle per lane
  - DDR2 transfers 4
  - DDRn transfers 2n
- Other sources of timing uncertainty
  - manufacturing variation, temperature, Miller side-wall effect, trace length
- Also add source synchronous clocking - enter DQS
- Timing variance creates synchronization issues
  - DDR device uses DLL/PDLL to sync with Mem.Ctl master clock
  - need to latch in the center of the data "eye"
  - delay proportional to RC
  - power proportional to CVf
Disturbing Trend

- DIMM capacity going up
  - process improvements yield more bits/die
- DRAM channel speed going up
  - DDRn
- # of DIMMs per channel going DOWN!
  - SDR - 8 DIMMs/channel
  - DDR - 4 DIMMs/channel
  - DDR2 - 2 DIMMs/channel
  - DDR3 - 1 DIMM/channel and higher latency
- isn't this a lower bound?
  - adding channels is expensive in CPU pins
- Why?
  - stub electronics problem on a JDEC broadcast bus
  - gets worse if bus speed increases - it's the stub thing
- Problem essence
  - not enough memory capacity per socket
  - huge server problem today

Signal Integrity

- Increasingly limiting in shrinking processes
  - gets even worse
  - as speeds increase
  - as trace length increases
- Multi-drop wires are a problem
  - very difficult to achieve perfect transmission line behavior in practice
  - impedance changes with
    - temperature
    - manufacturing variability
    - L & C effects of the neighborhood
  - signal reflections
  - result in signal distortion
  - made worse by noise
    - also a neighborhood problem
- DRAM systems
  - traces are long, and broadcast is the norm
  - intra- and inter-device

Multi-Drop Bus Complications

- Result
  - as speeds increase
    - #DIMMs per channel decrease
    - delay added by slow rise time and let ringing settle
      - hmm - faster means more delay - huh?
  - socketed DIMM connector adds another discontinuity
    - socket - PCB trace - connector - DIMM trace to DRAM die

Other Complications

- Skew
- Jitter
  - small fluctuations in signal propagation velocity due to
    - temperature, supply voltage, etc
- Inter-Symbol-Interference (ISI)
  - L & C induced cross-talk
- Bottom line
  - lots of practical barriers to increasing signal speed
Termination

- Key to minimizing reflections
  - but DRAM needs to be cheap
    - cheap SOJ and TSOP packages
  - large pin C & L's mismatched to trace impedance
  - OK for low freq. < 200 MHz
  - faster requires smaller pins \( \Rightarrow \) BGA (DDR) & FBGA (DDR2/3)
- Another termination issue
  - impedance inside vs. outside the package need to be isolated
    - series termination (DDR)
    - damps internal DRAM component reflection effects on the DIMM trace
    - programmable on die parallel termination (DDR2)
    - higher speeds \( \Rightarrow \) tighter reflection constraints
    - configuration register on-axis termination resistor switches
    - removes need to time for worst case configurations (max DIMMs)

Voltage Issues

- Low voltage swing
  - saves power and potentially improves speed
  - BUT: reduced noise immunity
  - \( \Rightarrow \) do differential signaling
    - CACTI did this for all of your HW4 experiments
    - problem - DRAM's have to be cheap
    - can't afford 2x data pins
- \( V_{\text{ref}} \)
  - provide a common voltage reference used by all inputs
    - adv: \( x+1 < 2x \) pins for interesting values of \( x \)
    - disadv: lose the common mode rejection of differential

Intel's FB-DIMM Compromise
FB-Dimm Problems?

- There are many
  - daisy chain causes varied response time
  - bit lane retiming additional latency problem
- Already considered a 1-trick pony
- Enter BoB - Buffer on Board - the new Intel hack
  - use a tree rather than a daisy chain for 4x DDR3
  - BoB placement
    - motherboard or on a memory card riser
  - problem - another buffer stage in the memory hierarchy
  - OK if prefetch strategy is working for you
- AMD has/had? a similar variant
  - Socket 3 Memory Extender (G3MX) micro-buffer
    - effort now seems to have been cancelled

DRAM Systems Issues

- Power and Heat
  - the biggest concern now and in the future most likely
    - early DIMMs consumed about 1W
    - FB-DIMMs now at 10W
- Servers
  - goal
    - 3x more channels and 8x more DIMMs per channel
    - looks like 250 W per socket just for memory
  - huge problem now
  - definite time for a rethink
    - industry momentum
    - standards
    - DRAM commodity == super low margins
    - not thriving in a healthy proposition

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

Refresh Trends

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

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Vdd</th>
<th># Banks</th>
<th># Rows</th>
<th>Row Size</th>
<th>Refresh Count</th>
<th>$t_{RFC}$ ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2</td>
<td>1.8V</td>
<td>8</td>
<td>1</td>
<td>64</td>
<td>2048 16384</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>1</td>
<td>8192</td>
<td>2048 16384</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>1</td>
<td>8192</td>
<td>2048 16384</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>1</td>
<td>8192</td>
<td>2048 16384</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>1</td>
<td>8192</td>
<td>2048 16384</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>1</td>
<td>8192</td>
<td>2048 16384</td>
<td>1</td>
</tr>
</tbody>
</table>
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 wins since memory bus can still be active
          • peak power wins 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

DIMMs and DRAMs

<table>
<thead>
<tr>
<th>DIMM Type</th>
<th>DRAM IC Type</th>
<th>Bus Clock Rate (MHz)</th>
<th>Memory Clock Rate (MHz)</th>
<th>non-ECC Channel Width</th>
<th>ECC Channel Width</th>
<th>Peak Memory Bandwidth (GB/s)</th>
<th>Core Latency (bus cycles)</th>
<th>DIMM pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR-200</td>
<td>PC1600</td>
<td>100</td>
<td>100</td>
<td>16</td>
<td>16</td>
<td>1.6</td>
<td>2-3</td>
<td>184</td>
</tr>
<tr>
<td>DDR-266</td>
<td>PC2100</td>
<td>133</td>
<td>133</td>
<td>16</td>
<td>16</td>
<td>2.133</td>
<td>2-3</td>
<td>184</td>
</tr>
<tr>
<td>DDR-333</td>
<td>PC2700</td>
<td>167</td>
<td>167</td>
<td>16</td>
<td>16</td>
<td>2.667</td>
<td>2-3</td>
<td>184</td>
</tr>
<tr>
<td>DDR-400</td>
<td>PC3200</td>
<td>200</td>
<td>200</td>
<td>16</td>
<td>16</td>
<td>3.2</td>
<td>2-3</td>
<td>240</td>
</tr>
<tr>
<td>DDR2-400</td>
<td>PC2-3200</td>
<td>100</td>
<td>200</td>
<td>32</td>
<td>32</td>
<td>3.2</td>
<td>3-9</td>
<td>240</td>
</tr>
<tr>
<td>DDR2-533</td>
<td>PC2-4200</td>
<td>133</td>
<td>266</td>
<td>64</td>
<td>64</td>
<td>4.267</td>
<td>3-9</td>
<td>240</td>
</tr>
<tr>
<td>DDR2-667</td>
<td>PC2-5300</td>
<td>167</td>
<td>333</td>
<td>64</td>
<td>64</td>
<td>5.333</td>
<td>3-9</td>
<td>240</td>
</tr>
<tr>
<td>DDR2-800</td>
<td>PC2-6400</td>
<td>200</td>
<td>400</td>
<td>64</td>
<td>64</td>
<td>6.4</td>
<td>3-9</td>
<td>240</td>
</tr>
<tr>
<td>DDR3-800</td>
<td>PC3-6400</td>
<td>100</td>
<td>400</td>
<td>64</td>
<td>64</td>
<td>6.4</td>
<td>3-9</td>
<td>240</td>
</tr>
<tr>
<td>DDR3-1066</td>
<td>PC3-8500</td>
<td>133</td>
<td>533</td>
<td>64</td>
<td>64</td>
<td>8.53</td>
<td>3-9</td>
<td>240</td>
</tr>
<tr>
<td>DDR3-1333</td>
<td>PC3-10600</td>
<td>167</td>
<td>667</td>
<td>64</td>
<td>64</td>
<td>10.67</td>
<td>3-9</td>
<td>240</td>
</tr>
<tr>
<td>DDR3-1600</td>
<td>PC3-17000</td>
<td>200</td>
<td>1066</td>
<td>64</td>
<td>64</td>
<td>18.06</td>
<td>3-9</td>
<td>240</td>
</tr>
</tbody>
</table>

Additional Constraints

• Power - It’s the biggest problem as things get “better”?
  • 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
        • multiples lead indicates a cache line
          • for large areas of core
  • Remember
    • large current profile changes
      • cause timing delays
        • hit line jitter depends on Vdd
          • Ohm’s law V = IR
            • not just a good idea - its the law

Double Edged Sword

• Active power
  • $P_a = aCV^2$
• non-adiabatic charging regime
  • $-\frac{1}{2}P$ 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 no more P and leakage
        • leakage impacts resource availability
          • can’t ignore refresh and the 64 ms standard target
Hot DRAMs & Packaging

source: random web photos

Heat spreaders: DDR 1st step
Fins and Fans: DDR2 and beyond

Passive heat pipes

$SSS$ Ka-ching $SSS$

Memory Controller Requirements

- Manage data movement to/from DRAM
  - device level
    - electrical & timing restrictions
    - error correction
      - typical parity just means retry and flag
  - system level
    - arbitration fairness
      - will be necessary in multiple core/mem_ctlr configurations
    - maximize system performance
      - command scheduling
    - multiple conflicting performance metrics however
      - heat, power consumption, latency, bandwidth

- Lots of options increase complexity
  - variety of timing parameters & command sequences
  - specific to the target device
  - scheduling for some optimality target
  - lots of queuing theory applies here

Top-Level View

- 3 top-level policy/strategies
  - row buffer management policy
  - address mapping scheme (MC, channel, rank, bank, row, col)
    - what's the right swizzle?
  - memory transaction and command ordering strategy

- Large body of research
  - partially due to huge timing differences
    - processors get faster & DRAM is fairly flat
  - seems to be reported primarily by the circuit community
    - according to recent look by Dave and Manu
    - ISPLED – Int. Symp. on Low Power Electronics and Design
    - and a bunch of reference cores put out by industry
    - main game played by northbridge chipset vendors

Basic MC Components

- Note
  - as memory access cost increases w.r.t. compute on CPU's
    - combining transaction and command scheduling is important
  - address translation targets rank and bank
    - transaction turned into a series of DRAM commands
    - optimization options occur with interleaved transactions
      - while still respecting device timing restrictions

School of Computing
University of Utah

45 CS6810
Row Buffer Management

- **Open-Page**
  - **good**
    - both temporal and spatial locality exist in access pattern
      - spatial: amortizes large row activate energy cost
      - temporal: energy to keep row open results in improved bandwidth
        - latency limited by \( t_{CAS} \) only
  - **bad**
    - energy: active row but no accesses
    - time: precharge, activate, access if target row is inactive
      - better to perform a col-rd-precharge command when new row is known
- **scheduling issues**
  - similar to dynamic instruction issue
    - performance increases with a larger window
      - except when window is always slightly filled
    - multi-core/MC changes the probability
    - dependent and anti-dependent issues must be tracked
    - note write buffer in XDR (sound familiar?)

Concluding Remarks

- **Whirlwind tour – phew!**
  - **Take homes**
    - understand role of MC, channel, rank, bank, row & column
    - large mat delay & broadcast commands
      - MC role is to overlap commands optimally
      - best bandwidth \( \rightarrow \) keep data bus active
      - open and closed row scheduling policy idea
    - **challenges for the future**
      - signal integrity limits bus speed
      - cpu pin count limits channel width
  - **Multi-core and improved process technology**
    - only makes things worse
    - more compute power \( \rightarrow \) higher memory pressure
      - caches help and are critical
      - but they can't catch everything
    - power is and will continue to be a fundamental constraint