Memory

Tajana Simunic Rosing
Department of Computer Science and Engineering
University of California, San Diego.
Hardware platform architecture
Memory hierarchy

- Want inexpensive, fast memory
- Main memory
  - Large, inexpensive, slow, stores all data
- Cache
  - Small, expensive, fast memory stores a copy of likely accessed parts
  - L1, L2, L3
Memory

Efficiency is a concern:
- speed (latency and throughput);
- predictable timing
- energy efficiency
- size
- cost
- other attributes (volatile vs. persistent, etc)
Access-times

\[ \geq 2x \]

every 2 years

Source: P. Machanik
Caches and CPUs

- **Servers**: L1, L2 & L3 cache on chip
- **Embedded**: L1, L2 on chip
Cache

- Designed with SRAM, Usually on same chip as processor
- **Cache operation:**
  - Request for main memory access (read or write)
  - First, check cache for copy
    - cache hit
    - cache miss
- **Design choices**
  - cache mapping
    - Direct - each memory location maps onto exactly one cache entry
    - Fully associative – anywhere in memory, never implemented
    - Set-associative - each memory location can go into one of n set
  - write techniques
    - Write-through - write to main memory at each update
    - Write-back – write only when “dirty” block replaced
  - replacement policies
    - Random
    - LRU: least-recently used
    - FIFO: first-in-first-out
Cache impact on system performance

Most important parameters in terms of performance:
- Total size of cache (data and control info – tags etc)
- Degree of associativity
- Data block size

Larger caches -> lower miss rates, higher access cost
- Average memory access time \( (h_1=\text{L1 hit rate}, h_2=\text{L2 hit rate}) \)
  \[ t_{av} = h_1 t_{\text{L1}} + (h_2-h_1) t_{\text{L2}} + (1-h_2-h_1) t_{\text{main}} \]
- e.g., if miss cost = 20
  - 2 Kbyte: miss rate = 20%, hit cost = 2 cycles, access 5.6 cycles
  - 4 Kbyte: miss rate = 10%, hit cost = 3 cycles, access 4.7 cycles
  - 8 Kbyte: miss rate = 8%, hit cost = 4 cycles, access 4.8 cycles
Predictability

- Embedded systems are often real-time:
  - Have to guarantee meeting timing constraints.

- Pre run-time scheduling - predictability
  - Time-triggered, statically scheduled operating systems
  - Predictable cache design?
Scratch pad memories (SPM)

Hierarchy

main

SPM

processor

Example

ARM7TDMI well-known for low power consumption

Addres

0

scratch pad memory

no tag memory

FFF..
Why not just use a cache?

Worst case execution time (WCET) may be large

[Computed WCET vs Simulated Time]

[P. Marwedel et al., ASPDAC, 2004]
Why not just use a cache?

Energy for parallel access of sets, in comparators, muxes.

[R. Banakar, S. Steinke, B.-S. Lee, 2001]
Scratchpad vs. main memory currents

Example: Atmel ARM-Evaluation board

Current reduction: Factor of 3.02
Scratchpad vs. main memory energy

Example: Atmel ARM-Evaluation board

"Main" memory access takes longer

 FileNotFoundError savings 86%

energy reduction: factor of 7.06

100% predictable
Memory management units

- Memory management unit (MMU) translates addresses:

  - CPU
  - Logical address
  - Memory management unit
  - Physical address
  - Main memory
Memory Management Unit (MMU)

- Duties of MMU
  - Handles DRAM refresh, bus interface & arbitration
  - Takes care of memory sharing among multiple CPUs
  - Translates logic memory addresses from processor to physical memory addresses of DRAM

- Modern CPUs often come with MMU built-in
- Single-purpose processors can be used
Address translation

- Mapping logical to physical addresses.
- Two basic schemes:
  - **Segmented**
    - memory footprint can change dynamically
    - usually only a few segments per process; e.g. data and stack
  - **Paged**
    - size preassigned
  - can be combined (x86).

<table>
<thead>
<tr>
<th>SEGMENTATION</th>
<th>PAGING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Involves programmer</td>
<td>Transparent to programmer</td>
</tr>
<tr>
<td>Separate compiling</td>
<td>No separate compiling</td>
</tr>
<tr>
<td>Separate protection</td>
<td>No separate protection</td>
</tr>
<tr>
<td>Shared</td>
<td>No sharing</td>
</tr>
</tbody>
</table>
ARM memory management

- Memory region types:
  - section: 1 Mbyte block;
  - large page: 64 kbytes;
  - small page: 4 kbytes.
- An address is marked as section-mapped or page-mapped.
- Two-level translation scheme.
ARM address translation

Translation table base register

descriptor
1st level table

concatenate

1st index 2nd index offset

concatenate

descriptor
2nd level table

physical address
Memory: basic concepts

- Stores large number of bits
  - $m \times n$: $m$ words of $n$ bits each
  - $k = \log_2(m)$ address input signals
  - or $m = 2^k$ words
  - e.g., 4,096 x 8 memory:
    - 32,768 bits
    - 12 address input signals
    - 8 input/output data signals

- Memory access
  - r/w: selects read or write
  - enable: read or write only when asserted
  - multiport: multiple accesses to different locations simultaneously
Write ability/ storage permanence

- Traditional ROM/RAM
  - ROM
    - read only, bits stored without power
  - RAM
    - read and write, lose stored bits without power
- Distinctions blurred
  - Advanced ROMs can be written to
    - e.g., EEPROM
  - Advanced RAMs can hold bits without power
    - e.g., NVRAM

Write ability and storage permanence of memories, showing relative degrees along each axis (not to scale).
**RAM: “Random-access” memory**

- **Volatile, read/write at run time**
- **Internal structure more complex**
  - a word consists of several memory cells, each storing 1 bit
  - each input and output data line connects to each cell in its column
  - rd/wr connected to every cell

- **SRAM: Static RAM**
  - Memory cell uses flip-flop to store bit
  - Requires 6 transistors
  - Holds data as long as power supplied

- **DRAM: Dynamic RAM**
  - Memory cell uses MOS transistor and capacitor to store bit
  - More compact than SRAM
  - “Refresh” required due to capacitor leak
    - word’s cells refreshed when read
  - Typical refresh rate 15.625 microsec.
  - Slower to access than SRAM
Extended data out DRAM

- Improvement of FPM (full page mode) DRAM
- Extra latch before output buffer
  - allows strobing of cas before data read operation completed
- Reduces read/write latency by additional cycle
(S)ynchronous and Enhanced Synchronous (ES) DRAM

- SDRAM latches data on active edge of clock
- Eliminates time to detect ras/cas and rd/wr signals
- A counter is initialized to column address then incremented on active edge of clock to access consecutive memory locations
- ESDRAM improves SDRAM
  - added buffers enable overlapping of column addressing
  - faster clocking and lower read/write latency possible
Rambus DRAM (RDRAM)

- More of a bus interface architecture than DRAM architecture
- Data is latched on both rising and falling edge of clock
- Broken into 4 banks each with own row decoder
  - can have 4 pages open at a time
- Capable of very high throughput
DRAM integration problem

- SRAM easily integrated with CPU
- DRAM more difficult
  - Different chip making process between DRAM and conventional logic
  - Goal of conventional logic (IC) designers:
    - minimize parasitic capacitance to reduce signal propagation delays and power consumption
  - Goal of DRAM designers:
    - create capacitor cells to retain stored information
SRAM vs. DRAM

- **SRAM:**
  - Faster.
  - Easier to integrate with logic.
  - Higher active power consumption per bit.
- **DRAM:**
  - Denser.
  - Must be refreshed.
Comparing RAM

- **Register file**
  - Fastest
  - But biggest size

- **SRAM**
  - Fast (e.g. 10ns)
  - More compact than register file

- **DRAM**
  - Slowest (e.g. 20ns)
    - And refreshing takes time
  - But very compact
  - Different technology for large caps.
Memory power trends

- DRAM: 2X size / chip / 3 year
  - Growth rate lower than the requirement for system performance growth
- Numerically, more memory chips will be needed in a system to match the requirement for growth in system performance
- Memory chip power is unlikely to decrease
  - $P = \text{active power} + \text{leakage power} + \text{refresh power}$
  - DRAM can be put in standby mode, i.e., no active power, but refresh and leakage power are still present at reduced levels
ROM Types

- **Mask or fuse programmed ROM**
- **Erasable Programmable ROM (EPROM)**
  - Uses “floating-gate transistor” in each cell
  - Programmer uses higher-than-normal voltage so electrons tunnel into the gate
    - Electrons become trapped in the gate
    - Only done for cells that should store 0
    - Other cells will be 1
  - To erase, shine ultraviolet light onto chip
    - Gives trapped electrons energy to escape
    - Requires chip package to have window

- **Electronically-Erasable Programmable ROM (EEPROM)**
  - Erasing one word at a time electronically

- **Flash memory**
  - Like EEPROM, but large blocks of words can be erased simultaneously

- **EEPROM & FLASH are in-system programmable**
EEPROM: Electrically erasable programmable ROM

- Programmed and erased electronically
  - higher than normal voltage
  - can program and erase individual words
  - can be erased and programmed 1000s of times

- Better write ability
  - can be in-system programmable with built-in circuit to provide higher than normal voltage
  - writes very slow due to erasing and programming

- Similar storage permanence to EPROM (about 10 years)
- Far more convenient than EPROMs, but more expensive
Flash Memory

- Extension of EEPROM
  - Same floating gate principle
  - Same write ability and storage permanence

- Fast erase
  - Large blocks of memory erased at once, rather than one word at a time
  - Blocks typically several thousand bytes large

- Writes to single words may be slower
  - Entire block must be read, word updated, then entire block written back

- Used with embedded systems storing large data items in nonvolatile memory
Definition of **Storage Class Memory**

- A new class of data storage/memory devices
  - many technologies compete to be the ‘best’ SCM

**SCM features:**
- Non-volatile (~ 10 years)
- Fast Access times (~ DRAM like)
- Low cost per bit more (DISK like – by 2015)
- Solid state, no moving parts

**SCM blurs the distinction** between
- MEMORY (*fast, expensive, volatile*) and
- STORAGE (*slow, cheap, non-volatile*)
Phase-change RAM

Access device (transistor, diode)

PCRAM “programmable resistor”

Bit-line

Word-line

Voltage - temperature

“RESET” pulse

“SET” pulse

Voltage

Temperature

Time

Potential headache: High power/current → affects scaling!

Potential headache: If crystallization is slow → affects performance!
STT-RAM (MRAM)

- Number of variations:
  - Electric field organizes the electrons in a magnetic tunnel so 1 bit of information can be stored in a space slightly larger than the size of an electron
  - Spin-polarized current switches magnetic bits, which consumes less power and enhances scalability
- Non-volatile, as fast reads as SRAM, endurance of DRAM
Memory/Storage Stack Latency Problem

Century

Time in ns

10^10
10^9
10^8
10^7
10^6
10^5
10^4
10^3
10^2
10
1

Human Scale

second

Storage

Get data from TAPE (40s)

Access DISK (5ms)

Access FLASH (20 us)

Access PCM (100 – 1000 ns)

Memory

Get data from DRAM or PCM (60ns)

Get data from L2 cache (10ns)

CPU operations (1ns)
If you could have SCM, why would you need anything else?
SCM in a large System

1980
- CPU
- Logic
- RAM
- DISK
- TAPE

2008
- CPU
- Logic
- RAM
- FLASH SSD
- DISK
- TAPE

2020
- CPU
- Logic
- RAM
- SCM
- DISK
- TAPE
Active vs passive power

- The blue area marks active power in the power equations
- The red area marks passive power in the power equations
  - Passive power is unproductive. It just causes heat
  - For memories it is leakage and refresh power, which is typically smaller than maximum active power
  - For disks it is keeping the motor spinning and the standby power of the electronics, which is typically larger than the maximum active power
  - For PCM it is the leakage and small standby power and is typically much much smaller than the maximum active power.

\[
P_M = V_{dd} I_{\text{leak}} + V_{dd} I_{\text{refresh}} + \alpha CV_{dd}^2 f \\
P_D = \kappa d^{4.6} r^{2.8} + I_{i\&c} V + \alpha I_{s\&t} V \\
P_{PCM} = I_{\text{standby}} V_{dd} + \alpha I_{\text{active}} V_{dd}
\]

$\alpha$ is the portion of time that the device is active and productive
$\kappa$ is the normalized power of the disk motor
## Focus on memory/storage stack

**Goal:** eliminate passive power and make active power more efficient

### Issues
- Redesign of DRAM and Disks to eliminate passive power
  - Possible but not probable
- DRAM has fast turn off and turn on times, but it is volatile
  - Turning off DRAM when not active causes data loss
- Disks are nonvolatile and turn on/off in ~ 20-30 seconds
  - On/off time too long for practical active storage systems
  - Storage systems that manage power in this manner are called MAID system.
  - So far, only used for archive systems

### Opportunities
- How can PCM be used to virtually eliminate passive power?
  - Active power is much greater than passive power
  - Turn on/off time ~50us
- How can data be laid out to minimize active power?
  - Memory/storage pools
  - Hierarchy
- How can active power be used more efficiently?
  - Device design
  - System architecture
  - Exploit virtualization (management challenge)
  - Exploit accelerators
PDRAM: combining PCM and DRAM
## DRAM vs. PRAM
(Phase Change RAM)

- Storing data: Altering the material state
  - Crystalline
  - Amorphous
- Comparison with DRAM

<table>
<thead>
<tr>
<th></th>
<th>PRAM</th>
<th>DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-volatile?</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Scaling</td>
<td>😊</td>
<td>😞</td>
</tr>
<tr>
<td>Read speed</td>
<td>😊 Moderately Slower than DRAM</td>
<td>😊</td>
</tr>
<tr>
<td>Write speed</td>
<td>😞 Slower than DRAM</td>
<td>😊</td>
</tr>
<tr>
<td>Power</td>
<td>😊 Standby &lt; DRAM</td>
<td>😞 Standby</td>
</tr>
<tr>
<td></td>
<td>😊 Read &lt; DRAM</td>
<td>😊 Read</td>
</tr>
<tr>
<td></td>
<td>😞 Write &gt; DRAM</td>
<td>😊 Write</td>
</tr>
<tr>
<td>Write endurance</td>
<td>😊10⁹</td>
<td>Very High</td>
</tr>
</tbody>
</table>

[Diagram of PRAM and DRAM comparison]
System Design

- Use both PRAM & DRAM
  - Low standby & read power of PRAM
  - High write endurance and low write power of DRAM

- Challenges
  - Hide drawbacks of PRAM
    - Write endurance ($10^9$ write cycles)

- Design goals:
  - Uniformly distribute writes across PRAM (wear leveling)
  - Direct writes to DRAM
  - Hybrid hardware/software solution
    - Hardware: PDRAM MemoryController
    - Software: PRAM Memory Manager
PDRAM Memory Controller

- Aware of memory partition
  - Routes requests based on address

- Low Cost write accounting:
  - Access Map
  - Access Map Cache

- Interrupts:
  - Page Swap
  - Bad Page

<table>
<thead>
<tr>
<th>Page Addr</th>
<th>Write Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1001</td>
<td>500</td>
</tr>
<tr>
<td>0x10ae</td>
<td>25</td>
</tr>
</tbody>
</table>
PRAM Memory Manager

- **Objective:** Uniformly distribute writes across PRAM pages

- **Memory Allocator**
  - **Free list:** free pages (not yet allocated)
  - **Threshold free list:** pages freed by page swapper
  - **Used free list:** pages freed by applications

- **Swap** free and **threshold-free** list pointers when free list becomes empty

- **Page Swapper**
  - Handle page-swap/bad page page interrupts
  - Bad page interrupt: mark page as invalid (writes > $10^9$)
  - Page-swap interrupt (writes > swap threshold):
    - Replace old page with a new page
    - Put the old page in threshold free list

**Which new page?**
- **Policy decision**
New Page Policies

- **Next PRAM page policy (Uniform policy)**
  - Allocate new page from the PRAM free list
    - Can be used in PRAM only configuration
    - Exploits wear leveling
    - Does not benefit from heterogeneity

- **Next DRAM page policy (Hybrid policy)**
  - Allocate new page from the DRAM free list
    - High probability of that page being write intensive
  - Advantages of moving write intensive pages to DRAM:
    - Reduces page swap interrupts
    - Reduces number of writes to PRAM
      - Good for reliability and power
Endurance Analysis

- Assume application that writes to two addresses:
  - Map to different rows in the memory bank
- Every write results in a writeback
  - Assume no wear leveling
  - Write Latency ($T_w$) = 150ns
    - Assume the application writes every $T_w$ period for worst case
  - Max write cycles ($N_w$) = $10^9$
  - # of writes for $N_w$ writebacks = $2N_w$
  - Time = $T_w \times 2N_w = 300s!$
Evaluation

- Baseline System: 4GB of DDR3 SDRAM
- Two experimental systems:
  - **Hybrid System**: 1GB SDRAM + 3GB PRAM
  - **Uniform System**: 4GB PRAM
- Workloads
  - Simulate first 5 billion instructions

### Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>rpi (%)</th>
<th>wpi (%)</th>
<th># of pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>applu</td>
<td>1.94</td>
<td>0.93</td>
<td>24435</td>
</tr>
<tr>
<td>bzip2</td>
<td>0.12</td>
<td>0.08</td>
<td>24600</td>
</tr>
<tr>
<td>facerec</td>
<td>0.6</td>
<td>0.5</td>
<td>2240</td>
</tr>
<tr>
<td>gcc</td>
<td>0.15</td>
<td>0.06</td>
<td>2781</td>
</tr>
<tr>
<td>sixtrack</td>
<td>0.01</td>
<td>0.008</td>
<td>7601</td>
</tr>
</tbody>
</table>
Performance Overhead

- Average overhead of < 6% for the hybrid system
- Average overhead of 30% for the uniform system
  - Slower access times of PRAM
- Includes overhead of access map and page swap

![Graph showing performance overhead for different applications]

- applu
- bzip2
- facerec
- gcc
- sixtrack
- average

- Uniform
- Hybrid
Energy Savings

- Average Energy Savings of 30%
- Includes energy consumption for access map & page swaps

Bar chart showing energy savings for different applications and the average.
Access map cache (hit rate)

- Average hit rate of 90%
- Infrequent access map updates

![Bar chart showing hit rates for art, applu, bzip2, facerec, gcc, sixtrack, and the average, with a hit rate of 90% for most applications]
Summary

- Memory hierarchy
  - Needs: speed, low power, predictable
- Cache design
  - Mapping, replacement & write policies
- Memory types
  - ROM vs RAM, types of ROM/RAM
- Storage class memories
  - DRAM/PCM combo, SRAM/MRAM
Sources and References