CACHE MEMORY & SRAM/DRAM TIMING

An introduction to cache memories and their architecture
Microprocessor and Microcontrollers Course – Isfahan University of Technology – Mohammad Sadegh Sadri
Storages

- CPU needs memory
  - Program
  - Data
- Memories
  - Volatile memory
    - RAM
  - Non-Volatile memory
    - Hard disk
    - Flash
Storage Speed

- RAM is faster than
  - Hard disk and flash
- Access to RAM is easier
  - Compared to hard disk and flash
Basic Idea

- Hard disk is slow
- Access to it is difficult
- So:
  - Put a memory between hard disk and CPU
  - Now move *Mostly Used Data and Instructions* to this memory
  - Allow CPU to read from/write to this memory
Basic Idea Expansion (1)

- DRAM memory
  - Large
  - Slow
- SRAM memory
  - Fast
  - Small
- Idea
  - Put a fast SRAM memory between DRAM and CPU
  - Store Mostly Used Data and Instructions on this SRAM memory
  - Allow CPU to use this memory
Basic Idea Expansion (2)

- When CPU needs an instruction
- Look into fast SRAM memory and check
  - If the instruction exist $\rightarrow$ Give it to CPU
  - If the instruction does not exist $\rightarrow$
    - Read it from large DRAM
    - Give it to CPU
    - Update the contents of SRAM memory
Basic Idea Expansion (3)

• When CPU wants to store/read a data
  • Look into fast SRAM memory
  • Check if any of SRAM memory locations has been associated with this memory address
    • If yes: store the data into this location (or read data from this location)
    • If no: store data into main DRAM memory
      • And also in SRAM memory
Cache Memory

- A “faster” small memory
- Stores frequently used data
- To shorten access time
- Caches are everywhere:
  - CPU cache
  - Hard disk cache
  - Operating system cache
  - ...
Multi-Level Caches

Main Memory
Larger than caches
Slower than caches

Level 3 Cache
Larger than L2
Slower than L1

Level 2 Cache
Larger than L1
Smaller than L2

Level 1 Data Cache
Smallest
Fastest Nearest

Level 1 Instruction Cache
Smallest
Fastest Nearest

CPU
Basic Cache Operation

1 Cache Line (Row, Entry)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Data Block</th>
<th>Valid Bit</th>
</tr>
</thead>
</table>

Main Memory

<table>
<thead>
<tr>
<th>Index</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>xyz</td>
</tr>
<tr>
<td>1</td>
<td>pdq</td>
</tr>
<tr>
<td>2</td>
<td>abc</td>
</tr>
<tr>
<td>3</td>
<td>rgf</td>
</tr>
</tbody>
</table>

Cache Memory

<table>
<thead>
<tr>
<th>Index</th>
<th>Tag</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>abc</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>xyz</td>
</tr>
</tbody>
</table>
Direct Mapped Cache

- Main memory is divided into blocks
- Each block can be cached in only one cache row
- Fasted search speed (best hit time, worst hit rate)
Direct Mapped Cache

Each location in main memory can be cached by just one cache location.
Direct Mapped Cache

• Can some times be very bad!
• Suppose both of A, B and C are stored in the same memory block in the following code

```c
for ( i = 0 ; i < 10; i++ )
    A[i] = B[i] + C[i];
```
Fully Associative Cache

- Each main memory location
  - Can be cached in all of the cache memory lines
- When CPU wants to access a memory location
  - All of the cache lines should be searched first
- Slowest search (worst hit time)
- Best hit rate
- There are some techniques to speed up the search
Set-Associative Cache

- Cache will be divided into sets
  - Each set is 2, 4, 8 or ... cache lines
- The content of each memory block
  - Can be cached in any of cache lines in a specific set
- Doubling the associativity
  - For example from direct mapped to 2-way
  - Has same effect on hit rate as: doubling the cache size
- Trade off between hit rate, and hit time
2-Way Set Associative Cache

![Diagram of 2-Way Set Associative Cache]

- Main Memory
  - Index 0
  - Index 1
  - Index 2
  - Index 3
  - Index 4
  - Index 5
  -...

- Cache Memory
  - Index 0, Way 0
  - Index 0, Way 1
  - Index 1, Way 0
  - Index 1, Way 1
AMD Athlon 64 Cache

Main Memory
<= 8 GB

Other CPUs

L2 Unified
1 MB 16-way

L2 ITLB
512 entries
4-way

L2 DTLB
512 entries
4-way

L1 Instruction Cache
64KB 2-way

L1 ITLB
4 KB
32 entries
full assoc

L1 DTLB
4/2 MB
8 entries
full assoc

L1 Data Cache
64KB 2-way 2 ports
DRAM VS. SRAM

Detailed operation of DRAM and SRAM memories
Asynchronous SRAM

- Example: CYPRESS CY7C10612DV33 (16Mbits: 1M*16, 10ns)
What is chip enable?

- Usually CPU is connected to multiple devices on the same bus.
What is chip enable? (2)

- Usually CPU is connected to multiple devices on the same bus.

Only 1 device can use the bus at the same time.
What is chip enable? (3)

- Usually CPU is connected to multiple devices on the same bus.

Only 1 device can use the Bus at the same time.
Async-SRAM Read Cycle
Async-SRAM Write Cycle
Synchronous SRAM

- Similar to Async SRAM
- But
  - Every Read and Write operation is done with Clock pulse edges
- Example:
  - Cypress CY7C1460
  - 36MBits – 2M*18
  - No-Bus-Latency Synchronous SRAM (NoBL Architecture)
- No-Bus-Latency:
  - There is no delay for switching between read and write operations
CY7C1462
Read/Write Timing

- Page 22, component’s data sheet
DRAM Memory

- Access to DRAM memory is very complicated (compared to SRAM)
- DRAM accept commands (there is not just a simple WEn signal)
- DRAM accepts Address in 3-phases
  - Bank Address
  - Row Address
  - Column Address
Micron MT46V64M16

- Total number of : 1G bits
  - 4 Banks
  - 16M lines
  - Each line is 16Bits
Timings

• Page 53 Datasheet
• Page 64 Datasheet