Key Takeaways from Lecture

Memory
- Cascading Memory Blocks
- FIFOs
- Caches

Parallelism and Pipelining
Cascading Memory Blocks

2 \( 1K \times 8 \)  

To make a \( 1K \times 16 \)
Cascading Memory Blocks

2 1k x 8 \Rightarrow 2k x 8

A

Din

A[10]

Dout(7:0)

Berkeley
UNIVERSITY OF CALIFORNIA
Cascading Memory Blocks

2 read ports

A_w
Din

A
Din

A_1
Dat

A_2
Dat

2 read addresses
FIFOs (First In, First Out) Queue

--- c b a

read:  --- c b
write:  --- d c b

(valued into buffer)
FIFOs

If ptrs equal after write: Full
If equal after read: Empty
Caches

Memory Hierarchy

- Reg
- L1, L2, L3 caches
- DRAM
- Flash
- Harddrive

size

speed
Caches

Blocks of data from memory (Dram to L3)

(adjacent)

Cache Line
Caches (Direct Mapped)

Example:
- DRAM Size = 4 GiB
- Cache Line Size = 16 Bytes
- Cache Size = 64 Bytes (4 Lines)

Address: 0x00000000
- DRAM address

<table>
<thead>
<tr>
<th>Addr LSB</th>
<th>Memory</th>
<th>Index</th>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td></td>
<td>0</td>
<td>V Tag</td>
</tr>
<tr>
<td>0x00000010</td>
<td></td>
<td>1</td>
<td>V Tag</td>
</tr>
<tr>
<td>0x00000020</td>
<td></td>
<td>2</td>
<td>V Tag</td>
</tr>
<tr>
<td>0x00000030</td>
<td></td>
<td>3</td>
<td>V Tag</td>
</tr>
<tr>
<td>0x00000040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000050</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000060</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000070</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFFFFFFFFF0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- each line is written to a single location
- if index same but tag is different => conflict miss
- compulsory miss: when line is empty

Pros:
- simple

Cons:
- conflict misses
- evict data

Note:
- get used frequently
Caches (Fully Associative)

Example:
DRAM Size = 4 GiB
Cache Line Size = 16 Bytes
Cache Size = 64 Bytes (4 Lines)

Pros:
- no conflict misses
- if full and need to grab new value
  => capacity miss

Cons:
- a lot of comparing tag + valid bits
capacity
- cache replacement policy on misses

Cache Replacement Policy:
- least recently used (LRU)
Caches (Set Associative)

Example:
- DRAM Size = 4 GiB
- Cache Line Size = 16 Bytes
- Cache Size = 128 Bytes (8 Lines)
- 2 Way Set Associative:
  - 4 Indexes

 Tradeoff:
- Simplicity in HW
- Fewer conflict misses

---

<table>
<thead>
<tr>
<th>Address:</th>
<th>Tag</th>
<th>Cache Index</th>
<th>Byte Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000030</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000050</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000060</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00000070</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFFFFFFF0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- within a set, cache entries are fully associative
- replacement policy applies to entries in set
- 2 entries per set

---

Berkeley
UNIVERSITY OF CALIFORNIA
Parallelism

\[ y = cx^2 + bx + a \]

\[ y = \text{mult} \text{ (add) \ 2 \ mult/\text{reduces}} \text{ here} \]

\[ c \cdot x \rightarrow (c \cdot x) \rightarrow c \cdot x + b \rightarrow (c \cdot x + b) \cdot x \]

\[ \rightarrow [c \cdot x + b \cdot x] + a \]
Parallelism

\[
\begin{align*}
&c \cdot x \\ &\rightarrow \\ &c \cdot x \cdot x \\
&\rightarrow \\ &c x^2 + b x + a
\end{align*}
\]

\[
\begin{align*}
&b \cdot x \\ &\rightarrow \\ &b \cdot x + a
\end{align*}
\]

speed with area (HW)
Pipelining (higher throughput)
Quick Note on Pipelining With Feedback

- Pipelining in the presence of feedback is problematic due to the dependence.
- However, if the feedback loop includes operators that are associative and commutative, we may be able to make the feedback loop shorter.
  - Tightening the feedback path pushes some logic outside of the loop.
  - Logic outside of the feedback loop (feed forward) can usually be pipelined relatively easily.
Example from Lecture

Orig: $y[i] = (y[i-1] + x[i]) + a$

Reorg: $y[i] = y[i-1] + (x[i] + a)$

Feed Forward Section Pipelined
Homework Questions