Can apparent superluminal neutrino speeds be explained as a quantum weak measurement?

M V Berry¹, N Brunner¹, S Popescu¹ & P Shukla²

¹H H Wills Physics Laboratory, Tyndall Avenue, Bristol BS8 1TL, UK
²Department of Physics, Indian Institute of Technology, Kharagpur, India

Abstract

Probably not.
Direct-Mapped Cache

Tag Memory | Valid Bits | Cache Memory | Main Memory block numbers
---|---|---|---
30 | 1 | | 025651276807936
9 | 1 | 257513230576817937
1 | 1 | 25851476817938
• | • | • | •
• | • | • | •
1 | 1 | 2555117678191255

Tag field, 5 bits
One Cache line, 8 bytes

Memory Address:
<table>
<thead>
<tr>
<th>5</th>
<th>8</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Group</td>
<td>Byte</td>
</tr>
</tbody>
</table>
Very Small Direct Mapped Cache

4 sets, one line per block, 2 bytes per block, 4-bit addresses

<table>
<thead>
<tr>
<th>Address (decimal)</th>
<th>Tag bit ((t=1))</th>
<th>Index bits ((s=2))</th>
<th>Offset bits ((b=1))</th>
<th>Block number (decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>01</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>01</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>10</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>00</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>00</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>01</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>01</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
Why index via the middle bits?

4-set cache

High-order bit indexing

Middle-order bit indexing

Set index bits
Set-Associative Cache

- Primary cache memory has \( S \) regions of \( E \) blocks memory of \( B \) size.
- Secondary memory has \( S' \) blocks of memory of \( B \) size.
- \( S' \) is typically a multiple of \( S \), and each region can store \( E \) blocks of memory.
- Each block will map into a specific cache region based on the block number modulus \( S \).
- If there is an empty block in a region then existing blocks will not be overwritten; otherwise, some existing data must be evicted.
Set-Associative Cache (2-way)
Set-Associative Cache (2-way)

Main Memory : ?
Cache Memory : ?
Set-Associative Cache (2-way)

Main Memory: \[ 2^5 \times 2^8 \times 2^3 = 2^{16} \text{ bytes} = 2^6 \text{ KB} = 64 \text{ KB} \]

Cache Memory: \[ 2^8 \times 2^3 \times 2 = 2^{12} \text{ bytes} = 2^2 \text{ KB} = 4 \text{ KB} \]
6.30

A) CT: [11-4], CI: [3-2], CO:[1-0]

B) • No, Unknown
• Yes, N/A
• Yes, C0
Fully Associative Cache

- Primary cache memory has 1 region ($S=1$)
- What does this imply?
• **Placement Strategies** -- where to place an incoming block in the cache.

• **Replacement Strategies** -- Which block to replace when a cache miss occurs.

• **Read and Write Policies** -- how to handle reads and writes upon cache hits and misses.
Hits and Misses

• What happens when there is a:
  – Cache Hit on a read or write?
  – Cache Miss on a read or write?

• Are the same things done for both the direct-mapped and set-associative caches?
Cache Hits Write Policy

- There are basically two methods:
  - *write through*: updates both the cache and main memory.
  - *write back* -- updates only the cache, and uses a **dirty bit** to indicated that it should be stored in main memory when purged from the cache.
Cache Miss Read Policy

• **direct-mapped cache:**
  – the valid bit is checked
  
  • **if not valid**, the block is read in, possibly forwarding the requested word;
  
  • **if valid**, the current block is eliminated (with consideration for the dirty bit) then the new block is loaded.