Abstract

In the modern world, computer system plays a vital role based on type of applications. Hence, there is a need to research and develop such system to improve its performance. In the field of computer architecture, cache address mapping acts between main memory and cache. Moreover, based on the internal or external request of the system, a bunch of words can be loaded onto cache memory. Due to the volatility nature of internal memory, the cache history gets abscond once the system is deactivated. However, the processor performance is based on various factors such as cache size and hit, write policy, type of cache mapping technique, CPU speed, front side bus, and depth of cache level. For cache, a number of standard cache addresses mapping are available. Among these, some of the well known techniques are set associative, fully associative and direct mapping technique. This paper reviews on non-linear cache address system in quadratic and cubic set associative cache address mapping. The standard set associative mapping is remapped with quadratic set associative technique for to secure the data in a non sequential fashion by having the standard mapping execution time. This work can also be applied to design the cache chip enhancement and improvement.

Keywords: Cache mapping, physical addressing, quadratic associative mapping, cubic associative mapping

1. Introduction

A very important feature of determining the effectiveness of level 2 caches is related to how the cache is mapped to the system memory. There are many different ways to allocate the storage in cache to the memory addresses it serves are


Direct Mapped Cache: Associative memories are expensive compared to random-access memories because of the added logic associate with each cell. The possibility of using a random access memory for the caches investigated the
CPU address of 15 bits divided in to 2 fields the nine least significant bits constitute Index field and the remaining six bits form the tag field. The following diagram shows the direct mapping method from the main to cache system.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Block</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 bits</td>
<td>14 bits</td>
<td>5 bits</td>
</tr>
</tbody>
</table>

Advantages

i. The tag memory is smaller than associative cache memory.

ii. The block field is used for the direct comparison between single fields.

Disadvantages

i. Consider what happens when a program references locations that are $2^{19}$ words apart, which is the size of the cache. Every memory reference will result in a miss, which will cause an entire block to be read into the cache even though only a single word is used.

![Diagram showing direct mapping method](image)

**Fig. 1 -** Direct mapping method

**Associative Cache Memory:**

The disadvantage of direct mapping is that two words with the same index in there address but with different tag values cannot reside in cache memory at the same time. A third type cache organization, called set associative mapping, in an improvement over the direct mapping organization.
Advantages

i. Any main memory block can be located into any cache slot.

ii. Irrespective of how irregular the data and program locations are, if a slot is available for the block, it can be kept in the cache.

Disadvantages

i. Considerable hardware above needed for cache bookkeeping.

ii. There must be a appliance for searching the tag memory in parallel.

**N-Way Associative Mapping:** “N” here is number typically 2,4,8… etc. This is a comparison between direct mapped and fully associative designs. The cache is broken into sets where each sets contain N cache lines, let’s say 4 i.e., 4 words per set. Then each memory address is assigned a set, and can be cached in any one of the four locations within the set that is assign to the other words with in each set the cache is associative. Hence it is called N-Way Associative Cache.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Set</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 bits</td>
<td>13 bits</td>
<td>5</td>
</tr>
<tr>
<td>5 bits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fully Associative: No Index

Direct Mapping: Large Index

![Diagram of a four way Set Associative Cache](image)

**Fig. 2 -** A four way Set Associative Cache.
Advantages

i. In our example the tag memory increases only slightly from the direct mapping and only two tags need to be searched for each memory reference. The set-associative cache is widely used in today’s microprocessors.

To measure set associative cache address mapping system using quadratic equation and to reduce time completion of cache address mapping system. The proposed quadratic cache mapping system aims to understand easily on associative cache address mapping and to access fast in cache address mapping system.

In the computer architecture, some of the standard cache addresses mapping techniques are direct mapping, set-associative mapping, and fully associative mapping. Generally, the cache mapping techniques differs with other through the way data is fetched and stored from the main memory onto cache and correspondingly. The data is referred from the cache through the policy called Locality of reference such as temporal and spatial reference. The temporal reference of a data refers a period of availability of data with respect to time and the spatial reference of a data refers to the location over the cache. The main aim of this paper is to experiment the time convolution of novel associative mapping with respect to standard associative mapping.

For instance, the system has the physical memory size of 16 MByte (2²⁴), the cache size 1024 KByte (2²⁰), and the block size is of 128Byte (2⁷). From the width of the SET field the number of rows is to be determined. Each set maximum refers to four blocks i.e., 4-way set-associative cache is applied onto the system. Thus the system consists of:

Number of Blocks in Physical memory,

\[ \text{Number of Blocks in Physical memory} = \frac{\text{Physical memory capacity}}{\text{Per Block Size}} \]

\[ = \frac{2^{24}}{2^7} \]

\[ = 2^{17} \text{ blocks} \]

Number of Sets = Number of Blocks / Number of ways

\[ = \frac{2^{17}}{2^2} \]

\[ = 2^{15} \text{ sets} \]

TAG Size

\[ \text{TAG Size} = \frac{\text{Physical size}}{\text{Cache Size}} \]

\[ = \frac{2^{24}}{2^{20}} \]

\[ = 2^4 \]
OFFSET Size = Physical size / (Number of Sets * TAG Size)

= \frac{2^{24}}{(2^{15} \times 2^4)}

= 2^5.

Hence, the Quadratic System fields are OFFSET (5 bit) and TAG (4 bit).

Fig. 3 - Direct Cache Mapping A[i] = A[j] mod Main memory

2. Motivation and Background

Lanfranco Lopriore et al., [3] a virtual address cache memory, whose operation is controlled explicitly by software presented. He also said that a new operand addressing mode reduce the complexity of cache management in logic to the capacity of the cache and solve the major problem of virtual address for this they did the mapping technique. Sambuddhi Hettiaratchi et al., [4] the cache address mapping is achieve various objectives such as minimizing cache conflicts and row switching activity for this they approach a Tile-Based Layout. Jung-Hoon Lee et al., [5] Lee the direct mapped cache is a favorite choice for short cycle time, but suffers from high miss rate hence they used the dual data cache is an approach to improve the is rate of direct-mapped cache without affecting the access time. They divided the cache system into two ways i.e., temporal locality and spatial locality. Inbum Jung et al., [6] in a multi-programmed system, when the
operating system switches contexts, in addition to the cost for handling the processes being swapped out and in, the cache performance of processors also can be affected. For this problem they used the concept of scheduling mechanism of keeping cache locality against context switching. Frank Mueller et al., [7] the low cache simulation overhead allows the interactive use of the analysis tool and scales with increasing associative. The framework is formally introduced in the pipeline mapped caches N. Chandramowliswaran et al., [1] describes that, the exponential growth of Information System needs a vital requirement to protect those data substantially from the prevention of unethical activities. To avoid such scenario in the internal memory system they are proposing a novel technique for **associative mapping** using **Graceful Code (GC) technique**.

The processor performance is directly depends on the cache, mapping technique, bandwidth, front side bus. In paper [1], their proposed work is focused on secured mapping over GC technique applied to acquire a demanding result. John S. Harper et al., [2] describes Cache behavior is complex and also unstable, but it is a critical factor affecting program performance. Quantitative predictions of miss-ratio and information to guide optimization of cache are required to evaluate cache. Cache simulation gives accurate predictions of miss-ratio, but little to direct optimization. Hence, the program execution time always lesser than simulation time. Many analytical models have been made, but concentrate mainly on direct-mapped caches, often for specific types of algorithm, to give qualitative predictions. Analytical models of cache are presented, applicable to numerical codes consisting mostly of array operations in looping constructs. **Set associative caches** are determined, through an extensive hierarchy of cache reuse and interference effects, including numerous forms of temporal and spatial locality. An advantage is that it indicates sources of cache interference. The accuracy validated through program fragments. The predicted miss-ratios are compared with simulations it will be within 15 percent. The evaluation time of the models is independent it depends upon the problem size, in general many orders of magnitude faster than simulation.

N. Chandramowliswaran et al., [8] deals with a novel idea of **set associative cache address mapping** using linear equation. The standard **set associative mapping** is remapped with linear set associative for to secure the data in a non sequential portion by having the standard mapping execution time. This is mainly focus on to design the cache enhancement and improvement. Stefano Di Carlo et al., [9] describe embedded microprocessor cache memories affect from observability and controllability creating problems during in system tests. So they apply procedure to transform
traditional march tests into software-based self-test programs for set associative cache memories with LRU replacement.

The main part of this work lies in the possibility of applying state-of-the-art memory test algorithms to embedded cache memories without any hardware or performance overheads and guarantee that detection of typical faults arising in nanometer CMOS technologies. The results got by constructing test programs for the LEON3 microprocessor show that it is possible to protect the fault coverage of the original march tests. The results also consider control blocks of the cache such as validity bits and control circuits, providing reasonable coverage also on these blocks.

3. Non-linear Set associative Mapping

The proposed is deal with the implementation of quadratic based set associative cache address mapping system using quadratic equation $y = (a_0 + a_1x + a_2x^2) \mod n$.

Solution Methodology

In this proposed system, the concept of set associative mapping techniques associated with direct and fully associative mapping. The cache lines are grouped into sets. The number of lines, ‘n’ in set can vary from 2 to 16. Set associative mapping will be divided into three parts.

<table>
<thead>
<tr>
<th>TAG</th>
<th>SET</th>
<th>OFFSET</th>
</tr>
</thead>
</table>

A. Quadratic Algorithm

STEP 1: Initialize all variables to $x$, $a_0$, $a_1$, $a_2$.

STEP 2: Get the inputs for $a_0$, $a_1$, $a_2$ from the user.

STEP 3: If $a_1$ is odd and $a_2$ is even,

$$f(x) = a_0 + a_1x + a_2x^2$$ and

$$y = f(x) \mod n.$$.

STEP 4: Else check the input values $a_1$ and $a_2$.

STEP 5: End.

For instance $[a_0, a_1, a_2] = [1, 3, 6]$ is shown in Table 1.

<table>
<thead>
<tr>
<th>X</th>
<th>f(x)</th>
<th>Y Encoded Set</th>
</tr>
</thead>
</table>

Table - 1: Quadratic Mapping
B. Cubic Algorithm

STEP 1: Initialize all secured variables to $x, a_0, a_1, a_2, a_3$

STEP 2: Get the inputs for real constant $a_0$, odd integer $a_1$, and pair of even integers $a_2$ & $a_3$

If $a_1 \% 2 == 1$ and $a_2 \% 2 == 0$ and $a_3 \% 2 == 0$

Goto step 3

Else repeat step 1

STEP 3: Compute $f(x) = (a_0 + a_1x + a_2x^2 + a_3x^3)$

STEP 4: Find its equivalent remapped block,

$y = f(x) \mod n$

STEP 5: End

For instance the private secured variables $[a_0, a_1, a_2, a_3] = [1, 3, 6, 4]$ is shown in Table 2.

Table - 2: Cubic Mapping

<table>
<thead>
<tr>
<th>$X$</th>
<th>$f(x)$</th>
<th>Y Encoded Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>172</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>365</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>666</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>1099</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>1688</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>2457</td>
<td>9</td>
</tr>
</tbody>
</table>
Here quadratic based set associative cache address mapping is based on the parameters $a_0$, $a_1$, $a_2$ and cubic by $a_0$, $a_1$, $a_2$, $a_3$.

And also, the time complexity of the existing algorithm takes only $O(n)$.

4. Conclusions

In this work, both quadratic and cubic based set associative cache address mapping has been presented by picking the parameters $a_0$, $a_1$, $a_2$ and $a_4$. Hence, the above remap clearly states that there exists unique one to one mapping from the search of cubic inputs, $X[0]$ to $X[15]$. Moreover, this work can also be extended to $n$ polynomial recursive mapping.

The time complexity of the existing algorithm acquires only invariable time $O(1)$. So the power consumption will be utilized effectively during the replacement of words.

References


5. Jung-Hoon Lee “ A new cache architecture based on temporal and spatial locality” Journal of Parallel Processing Laboratory, Department of Computer Science, 10 November 2000 PRESS.

6. Inbum Jung “A scheduling policy for preserving cache locality in a multiprogrammed system” Department of Computer Science, 30 November 2000 PRESS.


