Fully Associative Mapped Cache Memory System

David Murphy, Brent Simmons, Marc-Andre Couturier

University of New Brunswick Fredericton

Author Note

Table of Contents

[Abstract 3](#_Toc446172098)

[Reference System 4](#_Toc446172099)

[Specifications 4](#_Toc446172100)

[Design 4](#_Toc446172101)

[Matrix addition 5](#_Toc446172102)

[Enhanced System 6](#_Toc446172103)

[Specifications 6](#_Toc446172104)

[Design 6](#_Toc446172105)

[References 7](#_Toc446172106)

[Appendix A 8](#_Toc446172107)

# Abstract

The scope of this document is to describe in a concise way, the design, implementation and benchmarking of two systems. The first system, reference, is partially provided by the professor of the course. This system needs to be modified to meet the reference system requirements. The second system, enhanced, must implement a fully associative mapped cache to the reference. The two architectures must then be benchmarked to provide concrete evidence of: both architectures behaving as expected and to quantify the performance difference between the two architectures.

**Problem Statement**

We were tasked to design, simulate and implement a fully associative cache memory system in VHDL using Altera Quartus II and to compare it to a reference system without cache memory to verify the performance enhancement.

# Reference System

This system is meant to be the main building block of the cached system as well as a benchmark to assess the improvement achieved by the cached system.

## Specifications

1. Main memory size should be 4kbytes of 16 bits words (address width of 12 bits). See appendix for detailed steps.
2. The speed of memory module is controlled by the memory clock, which has to be changed to 1/8 of the CPU clock. iii.
3. System modification will require changes to the microprocessor module (instantiate the new memory module and new address width) and to the controller module (to deal with the new memory timing).

Design

The m9k memory was implemented as indicated in the project manual provided. We have used the “.mif “ file format to implement the memory. A for loop was used in the top level entity to reduce the memory clock to 1/8th of the system clock. Delays had to be introduced in the controller as to assure memory was read properly. That was necessary because of the assumption that the original systems’ memory would read/write in only 1 clock cycle. However, the m9k memory needs 1 extra clock cycle to operate properly. The system was then tested and proper behavior was observed.

Note. Things to be noted about the design choices made, ambiguities and other relevant information.

Matrix addition

The matrix addition program implements the m9k memory module with the required operations to perform a 5 x 5 matrix addition. Two new operational codes were implemented. These were necessary to provide us with the most optimized coding tools to do matrix addition on the system. The two operational codes are as follows:

1. Code 8: It is called mov5 and will read the memory location specified in a register and write the information read into a register. The pseudo code for the operation looks like this: RF[r2] <= mem[RF[r1]].
2. Code 9: It is called jz2 and will jump to a memory address regardless of the value of RF[1].

# Enhanced System

This architecture was implemented using the reference system as a base. It is to be tested using the same benchmark that was used to test the reference system. The cache system is expected to provide faster execution time compared to the reference system

## Specifications

1. Main memory size should be 4kbytes of 16 bits words (address width of 12 bits).
2. Memory should be implemented with M9K units as before.
3. Cache size should be 32 words of 16 bits (8 lines of 4 words).
4. Mapping scheme: fully associative.
5. Writing scheme: write back.
6. CPU will not access main memory directly but through the cache.

Design

Stuff

Note. Things to be noted about the design choices made, ambiguities and other relevant information.

References

*Intel. (n.d.). Cache Architecture. Retrieved February 26, 2016, from* [*http://download.intel.com/design/intarch/papers/cache6.pdf*](http://download.intel.com/design/intarch/papers/cache6.pdf)

Stallings, W. (2006). *Computer organization and architecture: Designing for performance* (8th ed.). Upper Saddle River, NJ: Pearson Prentice Hall.

Appendix A

Table 1

[Table Title]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Head | Column Head | Column Head | Column Head | Column Head |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |

Note: [Place all tables for your paper in a tables section, following references (and, if applicable, footnotes). Start a new page for each table, include a table number and table title for each, as shown on this page. All explanatory text appears in a table note that follows the table, such as this one. Use the Table/Figure style to get the spacing between table and note. Tables in APA format can use single or 1.5 line spacing. Include a heading for every row and column, even if the content seems obvious. To insert a table, on the Insert tab, tap Table. New tables that you create in this document use APA format by default.]