# Programmable Cache Memory

F. Laleh [1] & M. Fathi [2]

[1] Aria Technology Group
[2] Fast Processing Research Lab., Dept. of Computer Eng., IUST

## Extended Abstract

One of the most important problems in computational speed of the high performance microprocessor based systems, is the slow speed of the dynamic memory (DRAM) with regard to the speed of CPU. To overcome this problem, a popular approach is to use *cache memory*. This is a physical solution to the problem, i.e., a physically high speed static memory (SRAM) is used as a part of memory. It contains the most common used blocks of data, so the CPU is engaged with this part of memory at the most times. This subject is displayed by a factor that is named *hit ratio*.

In this paper, a new method for increasing the hit ratio is introduced which is referred to as the *programmable cache memory*. This method is a combination of both the physical and logical approaches, i.e., in addition to use a high speed SRAM, a programmable logical circuit is added to the cache controller in order to increase the hit ratio.

The foundation of cache memory is based on a property, which is called the *locality of reference* [1, 2]. It refers to the locality of data, which are used by a proposed program at runtime. If the cache size is large enough and the data addresses are not varying widely, then a high hit ratio will be reachable [2, 3]. But, there are some applications in which we are engaged with more than one segment (segment is a continuous part of the memory which generally has a size larger than the cache size) of main memory, concurrently, e.g., in image processing applications or matrix operations. In these cases, using the direct mapped cache method is not very useful and only fully and set associative methods can be helpful. The advantage of the last two methods is their ability to work with the several segments of system memory at the same time. In some applications the size of memory segments which must be processed, is huge and each index in a segment is used only once, so none of the cache methods will be helpful, because the storage of index which will not be used, has no advantage. In fact the main disadvantage of all the previous cache methods is that they cannot provide data which have not been used before by the program at runtime. This subject is referred to as the *cold start misses* [2]. To cover this problem, some prefetching methods have been proposed [4]. We propose a programmable cache which is able to provide the data which have not been used before, while the program uses several different data segments with sizes larger than the cache size. The proposed method runs a programmable prefetching process while the CPU is running other instructions.

The programmable cache memory is made of three main parts: Data registers, control and timer registers, and cache driver program. Data registers contain the addresses and data, which are used by the CPU. This part of our proposed cache is a small fully associative cache memory. Control registers and timers are used to determine when and from where the valid data must be read or written to the memory. The control registers and timers are programmed by the cache driver program in order to provide the valid data at the desired time in data registers. The cache driver program investigates a proposed program before runtime and determines the approximate times of its memory operations at the runtime according to the CPU cycles, and the location and order of the instructions which is used in the proposed program and then programs the control registers and timers. There are situations in which the program contains interrupts or conditional branches, so the driver cannot determine the precise time of R/W operations. In these cases, the proposed cache method acts as a fully associative cache memory while in other situations it acts as mentioned before. The following figure shows the block diagram of the proposed cache memory.

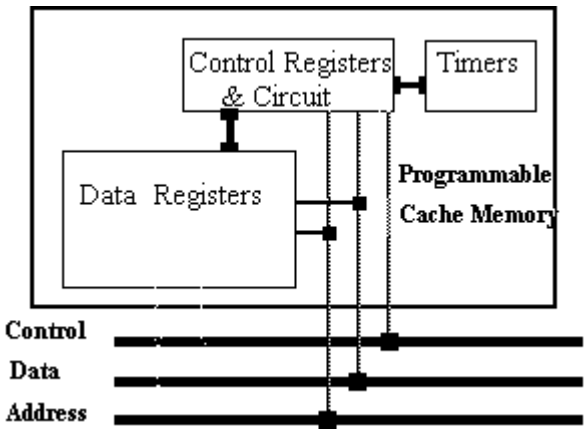


Fig. 1- The block diagram of the programmable cache memory

In this paper a full map of the programmable cache memory will be given. Then, the structure of its driver program will be expressed. Finally, we investigate the performance of the proposed method in a matrix processing application , when it is compared to the fully associative and other prefetching methods.

## References:

[1] K. Hwang, *Advanced Computer Architecture,* McGraw-Hill, 1993.

[2] J.l. Hennessy and D.A. Patterson, *Computer Architecture A Quantitative Approach,* Second Ed., Morgan Kaufmann, CA, 1996.

[3] M.D. Hill, *Aspect of Cache Memory and Instruction Buffer Performance,* Ph.D. Thesis, University of Calif. at Berkeley, Computer Science Division, Tech. Rep. UCB/CSD 87/381, 1987.

[4] N.P. Jouppi, "Improving direct mapped cache performance by the addition of a small fully associative cache and prefetching buffers," *Proc. 17th Annual Int'l Symposium on Computer Architecture,* Seattle, pp. 364-373, 1990.