ISSN: 2321-7782 (Online)
Volume 2, Issue 1, January 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Design and Implementation of Cache Controller for Multi-Core Systems Using Parallelization Method

Kalyani D. Kolhe[1]
Department of Electronics & telecommunication Engineering
G. H. Raisoni Institute of Engineering and Technology for women
Nagpur, India

Dr. U.M. Gokhale[2]
Department of Electronics & telecommunication Engineering
G. H. Raisoni Institute of Engineering and Technology for women
Nagpur, India

Darshan Pendhari[3]
Department of Electronics & telecommunication Engineering
G.H. Raisoni Institute of Engineering and Technology for women
Nagpur, India

Abstract: Multi-core processors are widely used across many application domains including general-purpose, embedded, network, graphics and digital signal processing (DSP). A multi-core processor is a single computing component with two or more independent actual central processing units called "cores", attracting many researchers to work for improvement in time management and power consumption of the system. Multi-core processing is a growing industry trend as single-core processors rapidly reach the physical limits of possible complexity and speed. It also improves the utilization and communication between the individual cores. There are many algorithm that are proposed by the researcher to fulfill the above requirements but still there is a wide scope to implement and propose in this area. Cache handling is one of the important issues which is having more impact on the performance of multi-core processor. One of the major issues in multi-core processor is cache coherence. Hence there is a requirement to implement a suitable and efficient method or design to handle.

So we are proposing a new concept of cache controlling by multilevel scheduling method, which utilized the concept of equal priority scheduling using parallelization. The proposed system is modeled using hardware descriptive language and simulation results are presented.

Keywords: Cache coherence, Multicore system, SoC

I. INTRODUCTION

Current processors typically embed many cores running at high speed. The main goal of proposed system is to assess the efficiency of software parallelism for low level arithmetic operations by providing a thorough comparison of several parallel modular multiplications. In computing, cache coherence (also cache coherency) refers to the consistency of data stored in local caches of a shared resource.

With additional cores, power consumption and heat dissipation become a concern. In a shared memory multiprocessor system with a separate cache memory for each processor, there is a possiblility to have many copies of any one instruction operand: one copy in the cache memory and one in each main memory. When one of the copy of an operand is changed, the other copies will also changed. Cache coherence ensures that the changes in the values of shared operands are propagated throughout the system in a timely fashion. To overcome this problem, caches are used to replicate the data and bring it closer to the requesting processors. Due to this the bus bandwidth requirement is less and also it is useful to minimize memory contention during memory sharing by different cores of Multi-core processor.

A. *Cache Memory*

Cache memory is random access memory (RAM) that a computer microprocessor can access more quickly than it can access RAM. As the microprocessor processes data, first it looks in the cache memory and if it finds the data there (from a previous reading of data), it doesn't have to do the reading of data operation which is more time consuming from larger memory. A memory cache, sometimes called a cache store, is a portion of memory which is made of high-speed static RAM (SRAM) instead of the slower and cheaper dynamic RAM (DRAM) used for main memory. Most programs access the same data or instructions over and over therefore memory caching is effective. Keeping as much of this information in SRAM, accessing the slower DRAM will be avoided by the computer.

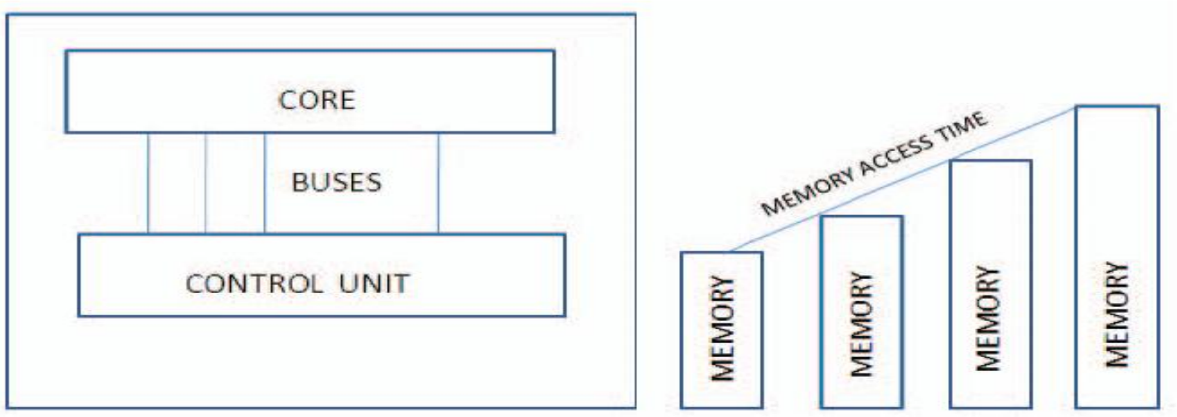

Figure 1. Memory Structure.

B. *Cache Controller*

Multi-core processors shares caches memory among multiple cores. Furthermore, multi-core processors typically employ a more complex interconnect mechanism to connect the cores, caches, and external memory interfaces that often includes switches and routers. In multi-core processor, cache coherency must also be considered. The mechanism which is used to connect the different cores with the same cache memory which is shared by the cores of processor and used to control the data transfer between cores and cache memory is called cache controller.

C. *Cache Coherence*

Cache coherence is the regularity or consistency of data stored in cache memory. Maintaining memory consistency and cache is imperative for multiprocessors or distributed shared memory (DSM) systems. Cache management is invented to ensure that data is not overwritten or lost. Different methods may be used to maintain cache coherency, including directory based coherence, bus snarfing and snooping. To maintain consistency, a DSM system invented these techniques and uses a coherency protocol, which is essential to system operations. Cache coherence is also known as cache consistency or cache coherency. This cache coherence problem is a critical correctness and performance-sensitive design point for supporting the shared-memory model. The cache coherence mechanisms govern communication in a shared-memory multiprocessor and also typically determine how the memory system transfers data between caches, memory and processor. future workloads will depend upon the performance of the cache coherent memory system, assuming the shared memory programming model remains prominent and continuing innovation in this realm is paramount to progress in computer design.

Here we have discussed different established methods for cache controller and we have proposed our own design and its implementation using VHDL.

II. CACHE CONTROLLER DESIGNS

Snoop filtering is designed to reduce energy consumption in snoopy bus based multi-core processor. According to the recent cache behaviors, jetty provides information as “what is available in local cache and what is not available”. Providing this information to the Cache controller, at relatively low cost certain snooping can be avoided. Satoetal proposed a cache aware thread scheduling policy for multi-core processors with multiple shared cache memories in. In the authors have introduced a technique for run time identification of data streams. A large number of coherence misses are eliminated by dynamically identifying sequence of memory accesses which correspond to a data stream. In embedded applications, system designers usually have much more detailed control over programs and hardware so that data sharing information is precise and deterministic. By using these advantages of this technique it is possible to avoid the speculative mechanisms present in general-purpose approaches and minimize area and power overhead.

Various Cache Controller Designing Techniques

Snooping: The first widely-adopted approach to cache coherence is snooping on a bus. A bus connects all components to an electrical, or logical, set of wires. A bus provides key ordering and atomicity properties that enable straightforward coherence operations. Snooping is the process where the individual caches monitor address lines for accesses to memory locations that they have cached. When a write operation is observed to a location of which a cache has a copy, the cache controller invalidates its own copy of the snooped memory location.

A snoop filter reduce the snooping traffic by maintaining a range of entries, each indicating a cache line that may be owned by single or multiple nodes. The snoop filter selects for replacement the entry representing the cache line or lines owned by hardly any nodes when replacement of one of the entries is required, as determined from a presence vector in each of the entries. A secular or other type of algorithm is used to refine the selection if more than one cache line is owned by the fewest number of nodes.

Round Robin Technique: In this cache controller is design for multiple core where single cache is shared by multiple cores of the processor. Here different priorities are allotted to all core and according to it scheduling is done. When all the cores wants to access the cache memory then core having highest priority will get the chance to use the cache memory first. That data which is available for one core is not provided on the data lines for other cores in order to avoid time and power consumption during accessing and processing unwanted data.

III. RESEARCH METHODOLOGY

Parallelism Technique:

As managing the multi-core processor cache memory is one of the biggest challenges in all processors, by processing the calculation overheads of address finding parallel system can increase the processing performance. Main goal of the proposed system is to make a use of Parallelization. System will manage the location and address fetching for defined address sing multiple processor execution.

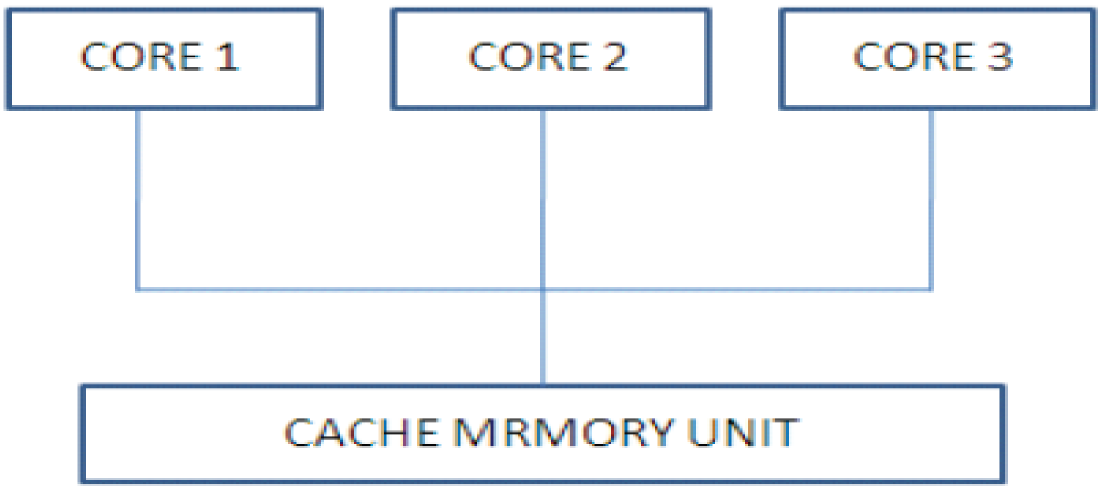

Figure 2. Cache memory shared by three cores

Work Done Uptil Now:

Proposed Cache Controller Using 4-way Associative Cache Memory

Cache Memory is a part of RAM. Therefore, total memory is equal to only RAM and not equal to RAM plus Cache Memory. Cache memory consists of two parts:

i. *tag memory*

ii. *data memory*

Tag memory is the one were addresses of some part of RAM memory location is stored and Data memory contains the data of respective memory location. Here we have used 4-way associative method in which cache memory is divided into 4 parts each containing 256 memory locations. Each memory is of 32 bits and having address of 22 bits. Each part is connected to a separate comparator to compare the address given to the processor with each addresses of cache memory. One counter is also design which is of 8 bits (since each part contain 256 location and 256= 2^8). When the requested address arrives at the processor, the first memory location address of each part compare this address with it and if the address is match then the process goes to next step. If not the process continues with remaining locations. Whenever the requested address matches with one of the address of cache tag memory then this process is known as cache hit and if match is not found then it is known as cache miss. After verifying the location the data of that memory location is given at the output. Now if the requested address matches with cache tag memory then to identify the part from 4 existing parts of which the address is match we have assign the parts by binary representation as “00”, “01” , “10” and “11”. VHDL code is written for counter, comparator, cache tag memory, cache data memory and controller and also for representing 4 parts with binary representation. Finally the VHDL code using structural modeling is written to design the whole cache controller with cache memory using 4way associative method. Then stimulation is done.

Figure 3. Shows the set associative cache memory organization, Figure 4. Shows the RTL view of cache controller with Cache memory and Figure 5. Shows the stimulation result.

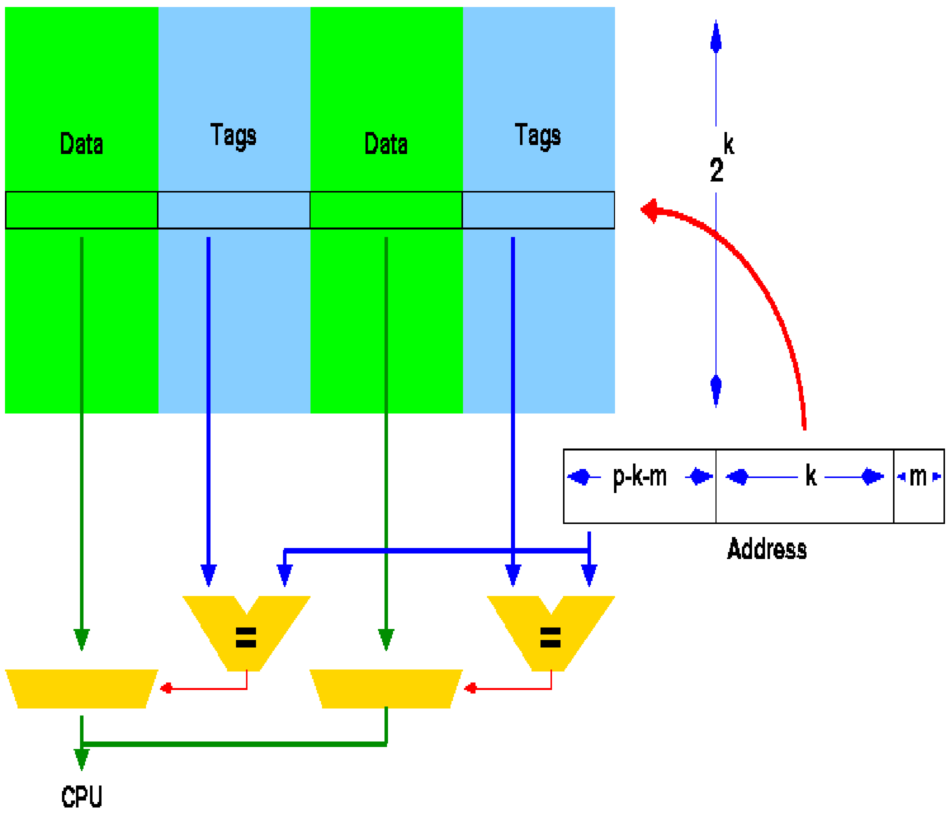

Figure 3. Set associative cache memory organization

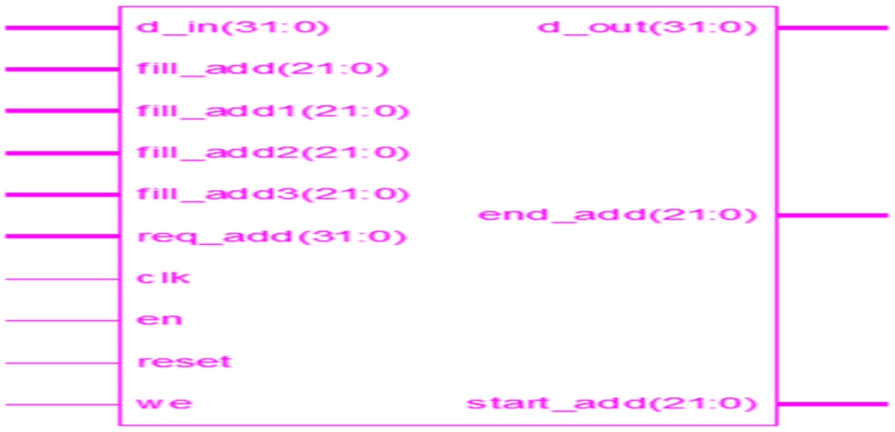

Figure 4. RTL view

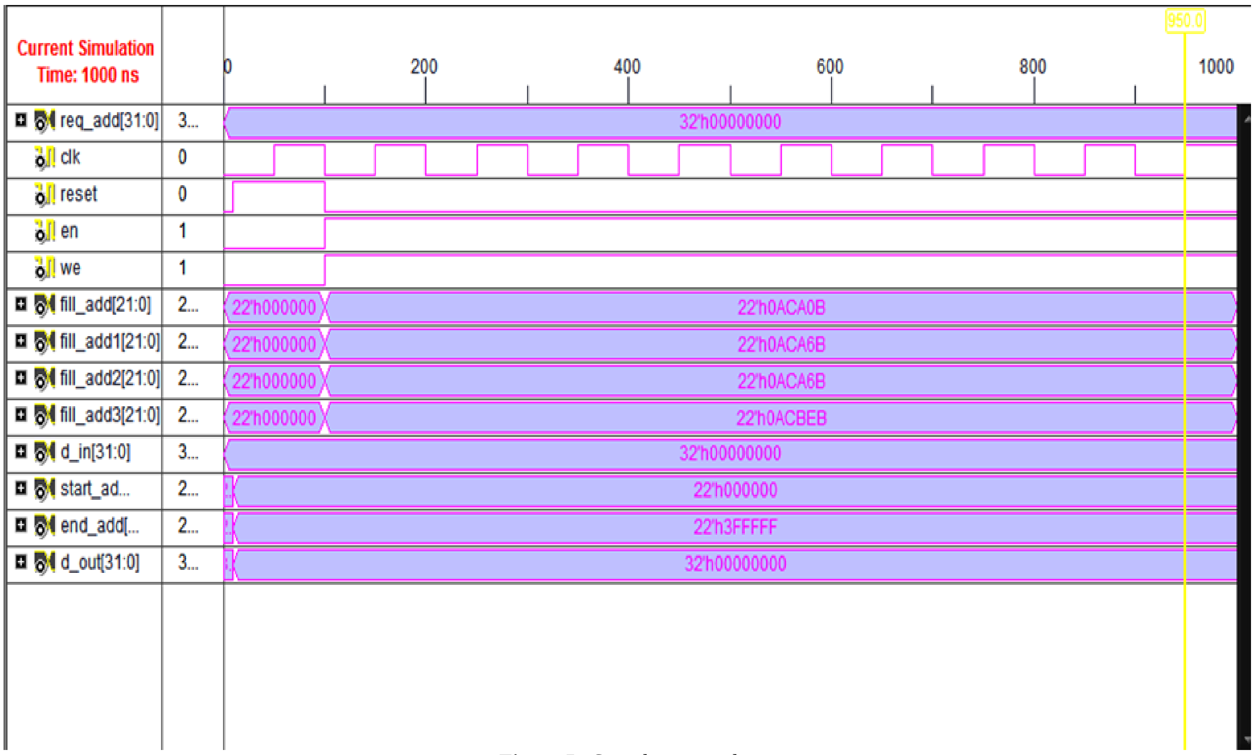

Figure 5. Stimulation results

Here the requested is not present in the cache tag memory means cache miss have taken place, so the address range of cache tag memory is shown in start_add and end_add..

IV. Software to be used

Xilinx Software

VHDL: VHDL (VHSIC Hardware Description Language) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits (ICs).It can also be used as a general purpose parallel programming language. VHDL is an IEEE and U.S. Department of Defense standard for electronic system descriptions. It is also becoming increasingly accepted in private industry as experience with the language grows and supporting tools become more widely available. Therefore, to facilitate the transfer of system description information, an understanding of VHDL will become increasingly important. VHDL enables hardware modelling from the gate to system level. Formal verification of correctness of a design: require mathematical statement of the required functions of the system.

The leading advantage of VHDL is that it allows the behavior of the required system to be described (modeled) and verified (simulated) before synthesis tools translate the design into real hardware (gates and wires), when used for systems designs.

Another benefit is that VHDL allows the description of a concurrent system. VHDL is a dataflow language, unlike procedural computing languages such as BASIC, C, and assembly code, which all run sequentially, single instruction at a time.

A VHDL project is multipurpose. Being once created, a block's calculation can be used in many projects. However, many functional and formational block parameters can be tuned (, memory size, element base, block composition, capacity parameters and interconnection structure).

A VHDL project is portable. Being created for one base element, a computing device project can be implemented on another element base, for example Very large scale integrated circuits with various technologies.

V. Conclusion

Calculating the physical memory address of instruction to switch processor execution to particular address is very bulky work for processor which increase the processor overheads but which cannot be skipped so by designing better cache memory and its management we can improve the system performance. The proposed architecture will be very cost efficient as the required additions to the system software and the hardware architectures are minimal.

References

1. Man-Ki Yoon, Sibin Mohan, Jaesik Choi, Jung-Eun Kim, and Lui Sha, "SecureCore: A Multicore-based Intrusion DetectionArchitecture for Real-Time Embedded Systems"IEEE paper 2013.
2. VinayHanumaiah, SarmaVrudhula, "Energy-efficient Operation of Multi- core Processors by DVFS, Task Migration and Active Cooling." IEEE Trans. 2013.
3. Young-Si Hwang and Ki-Seok Chung, "Dynamic Power Management Technique for Multicore Based Embedded Mobile Devices". IEEE Trans. 2013.
4. Seung Hun Kim, Sang Hyong Lee, Minje Jun, Byunghoon Lee, Won Woo Ro, Eui-Young Chung and Jean-Luc Gaudiot, "*C-Lock*: Energy Efficient Synchronization for Embedded Multicore Systems" IEEE Trans. 2013.
5. Wenting Wang, QiushuangTian, "Cache Coherence Verification of a Multi-core Processor" IEEE Trans. Paper 2012.
6. Vipin S. Bhure, Dr. Dinesh Padole, "Design of Cache Controller for Multi-core Systems Using Multilevel Scheduling Method" International Conference paper 2012.
7. Qiujie Cui, Shangda Yang, Xin Wang, YichengGuo and Zhi Li, "Multi Core with Individual L1 Cache and Shared L2 Cache", EECS470 Final Project 2012 .
8. T. Usui, R. Behrends, J. Evans, and Y. Smaragdakis, "AdaptiveLocks: Combining Transactions and Locks for Efficient Concurrency,"Journal of Parallel and Distributed Computing, vol. 70, no. 10,pp. 1009–1023, 2010.
9. D. Molka, D. Hackenberg, R. Schone, and M.S. Muller, "Memory Performance and Cache Coherency Effects on an Intel Nehalem Multiprocessor System", in *2009 18th International Conference on Parallel Architectures and Compilation Techniques*, September 2009.
10. JJ. Treibig, G. Hager, and G. Wellein, "Multi-core architectures: Complexities of performance prediction and the impact of cache topology", Regionales Rechenzentrum Erlangen, Friedrich-Alexander Universiat Erlangen-urnberg, Erlangen, Germany, October 2009.