International Journal of Engineering & Technology, 7 (3.29) (2018) 367-369

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

synchronization mechanism for multi- core processors

Poornima P. R [1] *, Padmaja Devi Ge [2], Rohan Kubde [3]

[1] PG scholar , Dept. of Electronics and Communication , Malnad College of Engineering, Hassan
[2] Associate Professor, Dept. of Electronics and Communication, Malnad College of Engineering, Hassan
[3] Assistant Professor, Dept. of Electronics and Telecommunication, SITS Narhe, Pune
**Corresponding author E-mail: poornimaravikumar231@gmail.com*

Abstract

The latest trend in processor design development to get the higher performance is to integrate multiple cores into a single IC or onto multiple dies but in a single chip package. The architectures of the microcontrollers need to look the strict demand within the embedded word. For essential writing, the concurrent programs the heart of multi-core revolution is shared data synchronization. Ideally, synchronization ought to be able to exploit accessible cores for its excellent performance. In this project, tend to present a design for hardware supported synchronization unit that would be enforced on chip and that should be directly accessible by all the various multiple cores. A hardware module memory access controller is used here. It has additionally seen that dual-ported memory will provide the better performance if the multiple cores use inherent parallelism by locking the shared memory by using the tactic called address sensitive method.

Keywords: *Multi-Core Processor; Synchronization; Dual-Ported Memory.*

1. Introduction

The growing market and demand for faster performance driving the processor industry to manufacture faster and smarter chips. One of the most classic techniques to improve performance by executing programs in a much quicker time is to clock chip at higher frequency. But if the chip frequency is increased beyond 4GHz panelizes more power dissipation and heat dissipation. Additional technique to improve performance is parallel processing, data parallelism and instruction parallelism. One such technique to significant improve in performance is multi-core processors have been exist since past decade [1]. A multi-core processor consists of more than one core on a chip. Functional units of multi core processor are cores made up of computational units and caches. The main feature of multi core is more than one happening at the same time. In multi core processors at the same time multiple tasks can be executed parallel by a separate core thus boosting the performance [2]. Changing from single core processor to multi core processors is not without any challenges. When multi core processors uses shared memory synchronization among cores is the performance bottleneck. When more than one processor trying to access shared memory simultaneously they need some way to ensure that memory will be used only by one core at a time this process is called synchronization. Here shared resource is memory system. So overload is imposing on memory. As multiple processors trying to access same-shared data simultaneously, data being invalidated can be prevented by synchronization [3].
Mainly synchronization among multi-core processors is addressed here. In this paper multi core, synchronization mechanism for shared memory is proposed.

2. Related work

Lock based and locks free are the existing synchronization mechanisms discussed in [4], [5]. In [4] lock based technique shared data get locked by one process get exclusive access and other processes need to access will be in busy waiting or get blocked. In [5] lock free technique shared data can be read and write concurrently without corrupting it. These two techniques are commonly used and use of atomic operations (uninterruptable). Processes frequently check if lock is free and acquire a lock if it is free in a single atomic operation. In [6] multi-core process synchronization is discussed where synchronization mechanism uses signaling scheme does not need support of atomic operations or disabling interrupts. Barrier is a fast synchronization mechanism to achieve parallel execution of tightly synchronizing streams of instructions. Barriers implemented in software using shared memory and lock does not need special hardware where involved process enters a barrier and wait for the other processes to leave together [7]. In [8] three fully programmable cores DSP, RISC and VLIW integrated in HiBRID SoC each core having specific class of algorithm their connected by 64 bit AMBA AHB bus and shares a on chip non caching dual ported memory for fast synchronization. Hardware configuration described in [8] is used in this paper. In [9] global synchronization unit is presented which provide all processors to access global state information from all the other processors in just few clock cycles. This is directly related to the topic presented in this paper.
In compare to prior synchronization scheme, proposed mechanism is based on chip non caching memory and shared by all the cores. While each processor get exclusive write access and all the other cores get read access for single port memory. For dual port memory two cores get write access remaining cores get read access.

3. Synchronization in multi- core processor

The order of execution of threads is modified by synchronization mechanism. Synchronization coordinates the execution of threads and also manages shared data. The primary spotlight is on the best way to accomplish reliable communication between the processor cores using the on-chip shared memory.

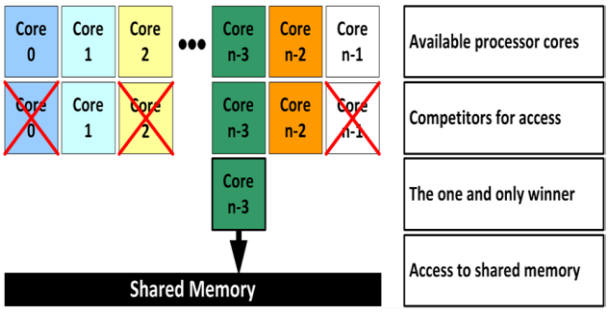

Fig. 1: Efficient Access in a Race Condition.

a) Problem

When race condition arises more than one core trying to access memory, so overload impose on it. Therefore to improve performance of system this becomes a potential bottleneck. However, problem is synchronization among number of cores is necessary and mechanism is needed to give access to one core at time in a race condition with following demands,

- In a race cores which really want to access should attend it and may become its winner.
- The race itself should not take more time and ideally winner has to select in one clock cycle.
- To get access core should not waits indefinitely.
- Every core in a race condition should serve in a finite time.

All the above demands can be are met with proposed synchronization mechanism.

b) Concept

To fulfill above requirements synchronization mechanism is proposed. In this mechanism cores which are really want to access memory are considered for race condition as shown in fig 1. To give access to all available cores in a finite time simple round robin scheme is utilized which require minimal resources for their implementation. Cores in the waiting state are get blocked to minimize the power consumption and get access when their turn comes. . In the worst case when all the available processor cores wanna access shared memory at the same time, this lead to different waiting times to all the available processors. To pick the access order dynamic priority scheme is utilized, so any processor core to be discriminated or favored can be avoided. Dynamic priority scheme is based on earliest deadline first scheduling and least slack time scheduling. A global locking bit is shared by all the processor available in the system to offer a processor to use any arbitrary location in the shared memory.

When accessing a little zone of memory, locking the complete shared memory is not proficient. Therefore rather than locking entire shared memory, just blocks of memory can be locked by cores, by utilizing strategy called address sensitive locking scheme. This empowers concurrent read-and write access to areas of shared memory, is exhibited for two cores in fig 2.

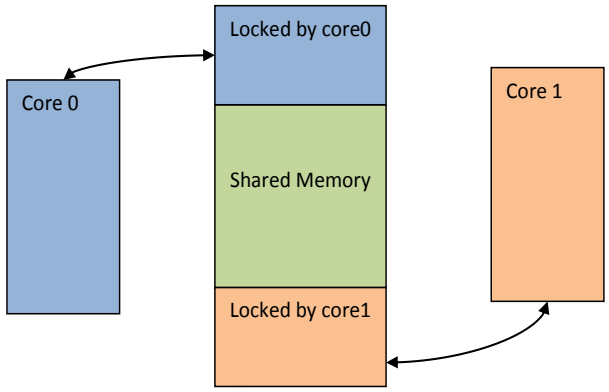

Fig. 2: Parallel Access to Different Memory Regions by Address-Sensitivity.

Instead of locking whole shared memory blocks of memory can be locked by using address sensitive locking scheme. So concurrently data can be read and write to the memory efficiently as shown in fig 2.

c) Realization

To control access to shared memory a unit called memory access controller is designed and developed. It consists of core side and inter-core logic as shown in figure 3.

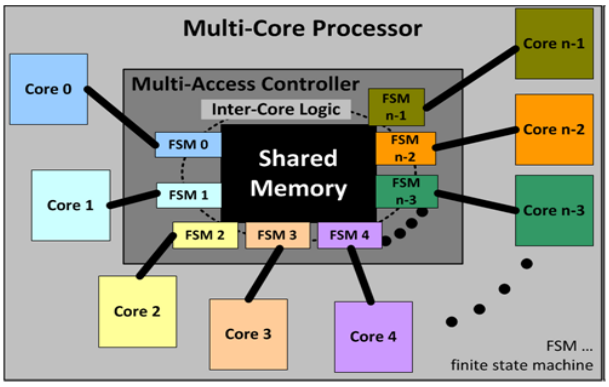

Fig. 3: Memory Access Controller, Abstraction.

To permit easy scaling in terms of processor cores a completely nonspecific plan of the MACtrl has been created and implemented in the hardware description language Verilog. In case of concurrent requests a code optimized algorithm is used that selects core that is allowed to access the shared memory in order to keep the design as small as possible. The core having highest priority among the available cores get access in a next clock cycles for a concurrent requests. The cores which want to access memory are considered remaining cores are masked. In order to execute multiple accesses without interruption also, global locking is implemented using a variation of the algorithm. MActrl stores the upper and lower address of the memory blocks as these addresses loaded Address-sensitive locking method gets activated. Then the controller locks a block of memory depending upon address location core want to access.

4. Result and discussion

All the synchronization mechanisms described above are simulated for eight cores. The specified design is modeled using Xilinix ISE Suite 14.5 and to validate functionality simulation is also performed. Core 1 and core2 requesting to write into different memory by using address sensitive scheme at single clock tick data is writing to memory locations. Core 3 to core 8 are requesting for read access simultaneously, on priority basis gets access simultaneously, so all requesting cores are served in finite time using round robin scheme. This complete operation is done memory access controller its simulation result is shown in fig 4

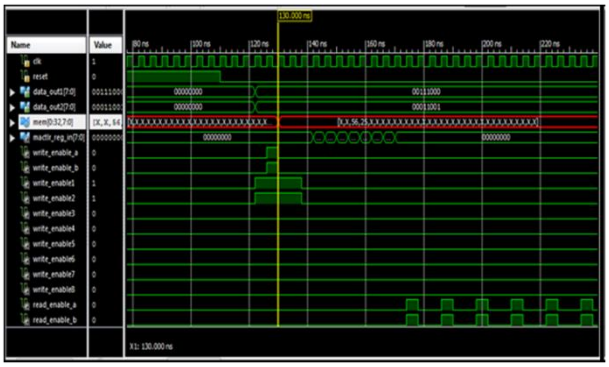

Fig. 4: Simulation Results of Memory Access Control.

5. Conclusion

The problem of synchronization among cores is solved efficiently using MACtrl. This can be integrated into on chip hardware. Dual port RAM is efficiently used using address sensitive scheme. Problems of overload, starvation and dead lock for poor performance are avoided.

References

[1] V Rajarama, "Multi-core microprocessors", RESONANCE | PP 1177 December 2017.

[2] Balaji Venu "Multi Core Processor – Overview" Computer Science | Hardware Architecture [v1] Sun, 16 Oct 2011.

[3] Christian Stoif," hardware synchronization for embedded multi core processors" 2011 IEEE International Symposium of Circuits and Systems (ISCAS) DOI: 10.1109/ISCAS.2011.5938126.

[4] J. M. Mellor-Crummey and M. L. Scott, "Algorithms for Scalable Synchronization on Shared-Memory Multiprocessors", ACM Trans. On Computer Systems, 9(1), February 1991.

[5] M.P. Herlihy, "Wait-free synchronization", ACM Transactions on Programming Languages and Systems, 13(1):124--149, January 1991.

[6] Arun Joseph, Nagu R Dhanwada "Process synchronization in multi core systems using on chip memories" 2014 13th International Conference on Embedded Systems.

[7] Mohammed Mahmudur Rahman, "Process Synchronization in Multi-Processor and Multi Core Processors" 2012 International Conference on Informatics, Electronics & Vision (ICIEV).

[8] H.-J. Stolberg et al., "HiBRID-SoC: A multi-core System-on-Chip architecture for multimedia signal processing applications," Universitaet Hannover, Germany, 2003.

[9] E. W. Lynch and G. F. Riley, "Hardware supported time synchronization in multi-core architectures," in ACM/IEEE/SCS 23rd workshop on principles of advanced and distributed simulation. IEEE Press, 2009, pp. 88–94.