International Journal of Research in Engineering, Science and Management
Volume-2, Issue-2, February-2019
www.ijresm.com | ISSN (Online): 2581-5792

A Literature Review on Uniprocessor to Multiprocessor in Computer Architecture

Bhargavi Gedela[1], Lohitha Bheemisetty[2]
[1,2]*UG Student, Department of ECE, Saveetha School of Engineering, Chennai, India*

***Abstract*: This paper describes about the evolution of uniprocessor to multiprocessor in computer architecture technology. One that is depicted and talked about is a characteristic argumentation of the transformative methodologies. This alternative spreads the issues of finding, portraying, controlling, and planning parallel handling. It is a generally useful approach that is relevant to a wide range of information handling and PC design. It is a generally useful approach that is appropriate to a wide range of information handling and PC arrangement. In some random application would sell out the speed of the constituent microchip in respect to that of the uniprocessor, in extent to the measure of direct processor movement engaged with that application. The equipment configuration incorporated a MP switch component included a time-multiplexed pipeline whose heart was a common cache serving to the processors. The product configuration depended on minor changes to a current continuous multiprogramming OS. Two models are analysis of memory interference they are discrete-time queuing model and Markov model. It is technique used to multiprocessor information address follows on a broadly accessible RISC uniprocessor for an explicit class of parallel applications. A microarchitecture can have sorted out the instructions and presented efficiently executed many instructions per cycle.**

***Keywords*: Uniprocessor, multiprocessor, pipeline, RISC, parallel handling, Markov model, multiprogramming.**

1. Introduction

Computer architects use modeling and simulation techniques to evaluate the performance of competing design solutions before physically constructing new machines. For example, the performance of the memory hierarchy of a shared memory multiprocessor is crucial to the overall performance of the multiprocessor. Adequate performance evaluation of a multiprocessor memory hierarchy requires a trace-driven simulation approach, since the performance of cache and memory controllers, cache coherence protocols, and processor-memory interconnection networks are highly dependent on the memory access patterns of parallel applications. The design and multiprocessor check for the PowerPC 604 information reserve efficiently checks the information reserve design, rationale, and execution rightness and gives the affirmation that the PowerPC 604 chip's forceful equipment and programming execution is completed accurately in the uniprocessor and multiprocessor condition.

OS executes a value-based disseminated memory (TDM) with page-granularity, which is gotten to through the processors memory the executives unit. All articles and most of the working framework piece live in a solitary substantial location space shared by all hubs. present day implanted frameworks target Multiprocessor Frameworks On-Chip (MPSoC)-based stages. The structures of these frameworks are winding up progressively mind boggling and are ineffectively bolstered by existing dialects and toolchains. In a common memory multiprocessor, the memory framework gives get to the data to be recessed and mechanisms for inter-process correspondence.

2. Survey on uniprocessor to multiprocessor:

Parallel handling in different structures has created fascinating improvements with regards to the PC field. This model solved the problems of describing, locating, controlling and scheduling parallel processing, high.

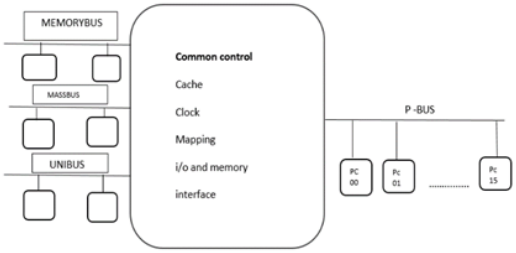

Fig. 1. Microprocessor system

Degree of parallelism some special protected Functions need to be accumulated lock-out feature temporarily restrict the access [1]. The multiprocessor configuration depicted here seems to offer points of interest in giving secluded steady development over a restricted and all around characterized scope of execution. This ranges that of ordinary minicomputer families, as it may be anticipated through the last 50% of the following decade. Without increasingly viable issue decay systems than are accessible today, uniprocessors will remain the more suitable methods for tending to applications embodied by single-entrusting [2]. Computer architects use modeling and simulation techniques to assess the execution of contending structure arrangements before physically constructing new machines. Adequate performance evaluation of a multiprocessor memory generation requires as follows since the execution of reserve and memory controllers, cache coherence

protocols, and processor-memory interconnection networks are exceptionally reliant on the memory get to examples of parallel applications [3]. In a multiprocessing domain present the reserve coherence problem. At the point when various processors keep up privately reserved duplicates of an exceptional shared memory area [4]. MINT ("MIPS mediator") includes the memory reference generator and reenactment library parts of a program-driven test system in a coordinated bundle. MINT has been cautiously designed for execution by utilizing a novel reenactment procedure, tuning the speed of every part, and by picking instruments that interface proficiently. MINT is planned explicitly for multiprocessor reenactment, so the inside instruments are tuned for that undertaking [5].

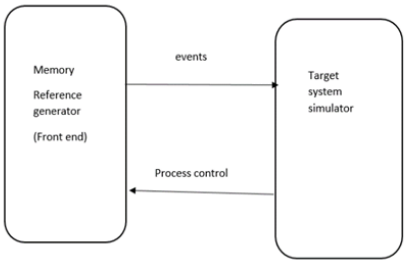

Fig. 2. A program-driven simulator

Upgrades in processor execution come about in two different ways - propels in semiconductor innovation and propels in processor microarchitecture. To continue the historical rate of increment in processing power, it is critical for the two sorts of advances to proceed. It is nearly sure that clock frequencies will keep on expanding. The micro architectural challenge is to issue numerous guidelines per cycle and to do as such effectively [6].The choice of a specific programming model determines how a programmer views the machine, independent of the machine's actual structure [7]. One of the objectives of Mach has been to explore the relation between hardware and software memory designs also, to structure a memory the executes framework that would be portable multiprocessor registering motors just as conventional uniprocessors.

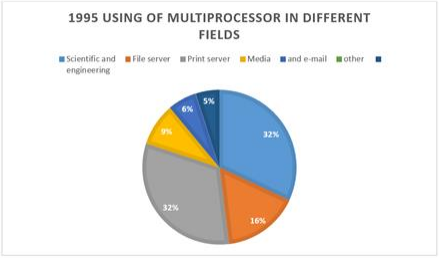

Fig. 3. In 1995 using of multiprocessor in different fields

The market has produced 15% grows annually. The predominant applications in the 1995 market and the anticipated market for the equivalent applications in 2000, as indicated by Dataquest. An "application" not appeared, in any case a critical bit of programming for a mutual memory machine, is the working framework [8].

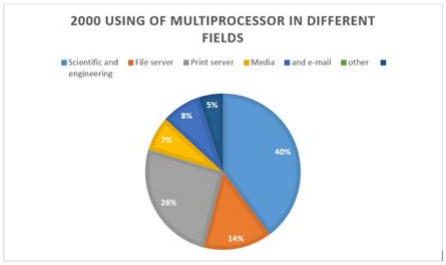

Fig. 4. In 2000 using of multiprocessor in different fields.

The main challenges faced in designing the Multiprocessor Systems-on-chip (MPSoC) are caused due to the complexity in applications. The MPSoC includes the integration of digital logic, embedded processors and mixed-signal circuits into a heterogeneous multiprocessor. Due to the integration of these technologies on to the single chip causes many challenges while power consumption and also software challenges. These MPSoCs are needed to designed in such a way that they have high potential security. We can make considerable changes in them by studying the traditional computer designing principles [9]. In the Chip multiprocessor (CMP) architecture the cache sharing has some of the negative impacts. The cache used in the CMP is L2 cache sharing, this cache sharing has significant impacts on the threads. In the Chip multiprocessor (CMP) architecture the cache sharing has some of the negative impacts. The cache used in the CMP is L2 cache sharing, this cache sharing has significant impacts on the threads. The impact is non-uniform due which the performance of some of the threads is slow downed [10]. The severe performance problems caused due to this cache are cache thrashing, sub optimal throughput and thread starvation. Three performance models are used to study the impact of cache sharing on co-scheduled threads to which circular sequence profile of each thread or isolated L2 cache stack distance is the input and output is taken as the number of extra L2 cache misses in cache sharing in each thread. Each a differs in accuracy prediction and complexity. Among these three models the inductive probability model gave the accurate results [11]. As we moved from uniprocessor to multiprocessor in recent years, now the main focus is on increasing the number of cores on chip. The Holistic Multicore Design is coined by professor Rakesh Kumar of UIUC which can give high general purpose performance, energy efficiency and reliability etc. We easily join two cores in conventional multiprocessors but putting cores on a single chip is the difficult task which need to be achieved in near future [12]. The uniprocessor is added with multiprocessor support for distributed operating system with transactional distributed memory (TDM). Synchronization and data consistency issues can be resolved by transactional memory. Rainbow OS which contains TDM as well as distributed operating system for PC clusters. The Rainbow OS consists of two parts namely

distributed kernel and local kernel. The distributed kernel is a part of global TDM and the local kernel is private and local to each station. Introducing multiprocessor capabilities into the uniprocessor is a tedious process both in hardware and software implementation and improving their architecture [13]. The MPSoC are solution for complex systems in loot of applications such as multimedia, networking and low power consumption. The reconfigurable or field programmable gate array (FPGA) technology is the new trend. The FPGA is also called as Multiprocessor-On-Programmable Chip (MPoPC). Not only proto types are implemented but real designs are also made based on FPGA technology. There are three main systems on the architecture based on this technology. These three approaches are point-to point, shared-bus and network on chip. These three approaches implemented physically for the communication connections. And there are two possible ways to connect and exchange information between two processors [14]. They are shared memory and message passing. The MPSoCs are classified as heterogeneous, homogenous and shared memory multiprocessor systems. The common

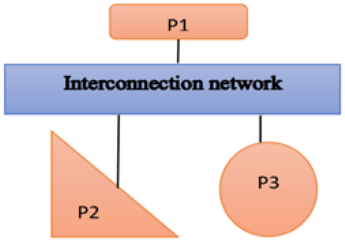

Fig. 5. Heterogeneous multiprocessor systems

Type of MPSoCs is the heterogeneous multiprocessor system (Fig.5). This system usually implemented in the embedded system applications which require heterogeneous behavior [15].

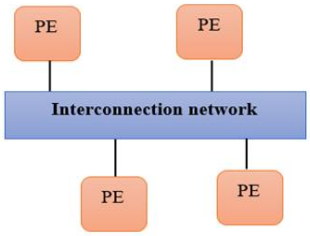

Fig. 6. Homogenous multiprocessor systems

The above fig. 6 represents the homogenous multiprocessor system. These are general-purpose identical processors. In this systems we can increase the number of processors without making any changes in the architecture. The software development is also in this type of systems. Depending on the memory architecture the multiprocessor systems can be divided into two types they are shared memory (Fig. 7) and distributed memory (Fig. 8).

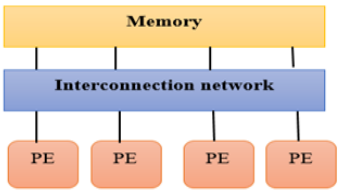

Fig. 7. Shared memory multiprocessor system

In shared memory systems there will be a single memory which is used by all the processors. The memory location will be visible to all the processors. There exists synchronization mechanism such as semaphore, barriers and locks. Different processors can easily share the information among them. The two widely used shared memory architecture threads are POSIX threads and OpenMP [16]. Due to the limited band width this architecture has poor scalability.

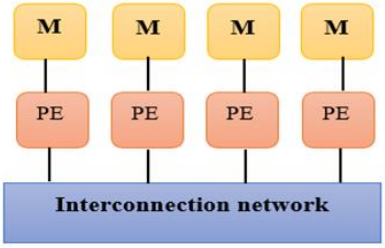

Fig. 8. Distributed memory multiprocessor system

Distributed memory systems contains separate memory for each processor. Due to this the message passing protocols are used for exchange of information among the multiple processors. Due to sharing of memory only these machines are more scalable [17]. The synchronization of the different memories used is the major problem in this systems. To resolve this problem Bisynchronization method is implemented. To perform lock and barrier function two hardware IP cores are used. The design methodology in building the FPGA based multiprocessors is of two types:

- Manually, implementing the design flow as per the instructions of FPGA companies.
- Automatically, application mapping using specific architecture.
- The hand tuned design is the most common design provided by the FPGA companies. This method is used for small system designs. The main problems in this design are synchronization, cache coherency and interrupt control.
- This automatic synthesis design consists of tools that can map with the applications. The main limiting factor is the amount of on-chip memory. And another limitation is it doesn't has standards.

The new degree of freedom in designing the multiprocessor systems is obtained from the run-time reconfigurability features of FPGAs [18]. So far only the hardware limitations are

discussed. Now, let us see main challenges faced in software implementation in the MPSoCs. The complex architectures implemented are poorly supported by existing toolchains and languages. The developed code need to analyze the worst-case execution time (WCET) and worst-case response time (WCRT). Due this advancement there is large increase in transistor density and also maximum clock rates [19]. There exists programming model and platform mismatch. The three mostly widely used programming languages in the world are C, C++ and Java [20]. The model driven engineering can be used to track the challenges (MDE). The MADES approach and TouchHMore approach are the two major approaches used in tracking the challenges. The MADES approach is MDE approach used for the design, simulation, validation, code generation and deployment of multi-processor embedded systems [21]. The TouchHMore approach uses a model driven approach which can capture the key characteristics of the platform to meet the specific requirements the tool chain can be customized. The prediction of WCET and its analysis for complex embedded systems is one of the significant problems. In small systems the simulation and testing is enough for the analysis [22].

3. Conclusion

The main reason for the evolution of uniprocessor to multiprocessor is the increasing scientific advancements and applications. Connecting many uniprocessors is tedious work which have both hardware and software challenges. So, due this we move to multiprocessors. Integrating the different cores on the single is the challenging task for the architects. These kind of devices are MPSoCs. The main challenges faced during the design of MPSoCs is also mentioned in this paper. The RPGA technology can implemented in MPSoCs to solve some of the hardware problems like synchronization and memory sharing. The different architectural solutions already existing are also mentioned in this paper. The software challenges faced in MPSoCs can be tracked mainly using two approaches like MADES approach and TouchHMore approach. There is a need to be some more development made in improvising the tools that can simulate, run and debug the software in the virtual design of the architectures.

References

[1] G. M. Amdahl, "New Concepts in Computing System Design," Proc. IRE, vol. 50, (May 1962) pp. 1073-1077.
[2] R. J. Swan, S. H. FuLler, and D. P. Sieworek,"Cm: A modular multi microprocessor," AFIPS Conf. Proc. NCC, vol. 46, 1977, pp.637-644.
[3] R. L. Sites and A. Agarwal, “Multiprocessor cache analysis using atum,” in Proceedings of the 15th Annual International Symposium on Computer Architecture, pp. 186 - 195, May 1988.
[4] JR. Goodman“,U sing Cache Memory to Reduce Processor-Memo Trrya ffic,” Pi-or.10th Ann. Symp. Computer Architecture, June 1983, pp. 124.131.
[5] R. Larus, Alvin R. Lebeck, James C. Lewis, and David A. Wood. The Wisconsin Wind Tunnel: Virtual Prototyping of Parallel Computers. In A CM SIGME TR ICs Intern at ion a1 Conference on Measurement and Modeling of Computer Systems, May 1993.
[6] E. Rotenberg, S. Bennett, and J. E. Smith. Trace cache: A low latency approach to high bandwidth instruction fetching. 29th Intl. Symp. on Microarchitecture, pages 24–34, Dec 1996.
[7] J.-Y. Tsai and P.-C. Yew, “The Super threaded Architecture: Thread Pipelining with Runtime Data Dependence Checking and Control Speculation,” in Proc. Int’l Conf. Parallel Architectures and Compilation Techniques, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 35-46.
[8] B. cariliey: seeking the balance :large smp ware houses pp-44-48.
[9] Wayne Wolf, “The Future of Multiprocessor Systems-on-Chips” Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium.
[10] S. Dutta, R. Jensen, and A. Rieckmann, “Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems,” IEEE Design and Test of Computers, September/October 2001, pp. 21-31
[11] Dhruba Chandra, Fei Guo, Seongbeom Kim, and Yan Solihi, “ Predicting Inter-Thread Cache Contention on a Chip Multi-Processor Architecture”, Proceedings of the 11th Int’l Symposium on High-Performance Computer Architecture.
[12] Dean M. Tullsen “Holistic Design of Multiple-Core Architectures” International Symposium on Parallel and Distributed Computing.
[13] Thilo schimtt, Patrick Schmidt et al “Adding multiprocessor support to a uniprocessor distributed operating system with transactional distributed memory”.
[14] Z. Wang and O. Hammami, “External DDR2-constrained NOC-based 24-processors MPSOC design and implementation on single FPGA,” in Proceedings of the 3rd International Design and Test Workshop (IDT ’08), pp. 193–197, December 2008.
[15] J. Dykes, P. Chan, G. Chapman, and L. Shannon, “A multiprocessor system-on-chip implementation of a laser based transparency meter on an FPGA,” in Proceedings of the International Conference on Field Programmable Technology (ICFPT ’07), pp. 373–376, December 2007.
[16] B. Nichols, D. Buttlar, and J. P. Farrell, Pthreads Programming, O’Reilly & Associates, Sebastopol, Calif, USA, 1996.
[17] “The openmp api specification for parallel programming,” http://openmp.org/wp.
[18] A. Kumar, S. Fernando, Y. Ha, B. Mesman, and H. Corporaal, “Multiprocessor systems synthesis for multiple use-cases of multiple applications on FPGA,” ACM Transactions on Design Automation of Electronic Systems, vol. 13, no. 3, pp. 1–27, 2008.
[19] S. Boyd-Wickizer, H. Chen, R. Chen, Y. Mao, F. Kaashoek, R. Morris, A. Pesterev, L. Stein, M. Wu, Y. Dai, Y. Zhang, and Z. Zhang. Corey: an operating system for many cores. In Proceedings of the 8th USENIX conference on Operating systems design and implementation, OSDI’08, pages 43–57, 2008
[20] TIOBE Software BV. TIOBE Programming Community Index for May 2012. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, May 2012.
[21] The MADES Consortium. The MADES Project. http://www.madesproject.org/, 2011.
[22] B. Chamberlain, D. Callahan, and H. Zima. Parallel Programmability and the Chapel Language. Int. J. High Perform. Comput. Appl., 21(3):291–312, 2017.