Alternative Architectures for Computer System Performance Enhancement

Hasan Krad

***Abstract*—Computer-based solutions for scientific applications require higher performance computer systems than ever. In fact, the improvement of the computer system performance through the improvement of the performance of hardware chips has almost reached its physical limitation and scientists started to seek the required enhancement of the system performance through new alternative approach to design computer systems using methods such as pipelining and parallel processing, with advanced processor architectures such as that of the Niagara family of processors that has been pioneered by Oracle®. In this paper, we will address some of these concepts and related issues and present a synopsis of some important aspects of such architectures while emphasizing on the pipelining techniques that lead to the performance enhancement of such systems.**

***Index Terms* — Pipelining, Multi-threaded, Performance, Multi-core, and System-on-chip (SoC).**

I. INTRODUCTION

The advancement of computer system architecture is motivated by everyday problems that demand accurate and fast solutions. The system architecture is modeled, from a programmer point of view, by its instruction set, addressing modes, registers, virtual memory, etc., while it is modeled, from an architect point of view, by CPUs, buses, physical memory, caches, pipelines, etc. In reality, during the last four or five decades, the *von Neumann* computer system architecture has gone through many evolutional changes and improvements in order to enhance the performance. However, during the last two decades or so, we started to see a new development of revolutional approaches to enhance the system performance using new techniques such as pipelining and parallel processing [1,2].

Pipelining is an efficient way to offer parallelism in computer systems. During any given time, each stage of the pipeline will be doing computation on a subtask that is belonging to a different task. In reality, the time required to finish the processing of a single task is increased, because of the buffering needed between the pipeline stages, but since many tasks are performed simultaneously through the pipeline by overlapping, the task completion rate is much higher compared to the sequential execution of the tasks [2,3,4]. Recently, Oracle® has established a distinguished series of microprocessors with abundant multithreading capabilities known as the Niagara family. One of the latest members of this family, the SPARC T4, is known for its improved performance due to the enhancements that are added to its multithreaded core, which offers up to 5x integer and 7x floating-point single-thread performance improvements in comparison with its predecessors from the SPARC T-series [2,5,6,7]. Furthermore, Oracle® did collaborate with Fujitsu on a series of microprocessors for Unix servers known as the SPARC64 family. The 10th generation of this family, named SPARC64 X processor, is distinguished with its great capabilities of handling huge data and its support of High Performance Computing (HPC) features that significantly enhance the performance and throughput of the processor [2,8]. In this work, we will highlight the architectures of SPARC64 X and SPARC T4 and present the way arithmetic pipelining techniques were deployed in the processors to improve their performance. The following sections of this paper are organized as follows: Section II discusses some pipelining concepts and issues, section III explains the SPARC64 X microarchitecture and its detailed design, section IV discusses the SPARC T4 architecture design and focuses on its main hardware features, section V presents the evaluation of both SPARC64 X and SPARC T4 and provides an assessment of both technologies in terms of their capabilities and overall performance, section VI ends the paper with some concluding remarks.

Prof. Hasan Krad, Department of Computer Science and Engineering, Qatar University/College of Engineering, Doha, Qatar.

II. PIPELINING CONCEPTS AND ISSUES

Pipelining offers an economical way of realizing parallelism in computer systems in order to enhance the system performance. At any given time, each stage of the pipeline will be performing a subtask belonging to a different task. However, the processing time required to complete a task is not reduced by the pipeline. In fact, it is increased, due to the buffering needed between the stages of the pipeline, but since several tasks are processed concurrently by the pipeline, the task completion rate is higher compared to sequential processing in a non-pipelined system. The total processing time of a program consisting of several tasks is shorter, compared to sequential execution of tasks. The throughput of an *N*-stage pipelined processor is ideally *N* times of the non-pipelined processor. Almost all computer systems today employ pipelining techniques to one degree or another. Furthermore, the ideal performance is not achievable due to many issues. For example, since all *N* stages are active simultaneously, as *N* becomes large, the memory traffic increases *N-fold*, thus making the memory module a *bottleneck*. Also since in pipelining the maximum clock rate is determine by the slowest stage, it may appear to be possible to speed up the pipeline arbitrarily by dividing each stage into smaller and smaller parts. However, because the delay of the staging registers is fixed, the pipeline will eventually reach a point where the amount of improvement in speed is negligible, and the cost of making such improvement

becomes prohibitive. Although clocking constraints may determine the ultimate physical limit to the number of stages in the pipeline, the maximum number of stages may not be optimal when other factors, such as cost and pipelining overhead, are considered. In that case, a cost/performance tradeoff is to be considered. Other issues that we need to deal with is the overhead of the interrupts as a deep pipelining increases the interrupt overhead, control hazard, structural hazard, collision handling, etc.

III. SPARK64 X PROCESSOR

SPARC64 X processor is designed for mission-critical UNIX server and fabricated in enhanced 28 nm high-k metal-gate (HKMG) CMOS. It runs at 3.0GHz with a peak performance of 382 gigaflops and contains 16 identical cores with 24 MB shared L2 cache and system/DDR3/PCIe interfaces on 588 mm^2 die area. Figure 1 shows theSPARC64 X pipeline which has very similar structure to the Unix processor SPARC64 VII, except that almost all parts have been enhanced [2,9].

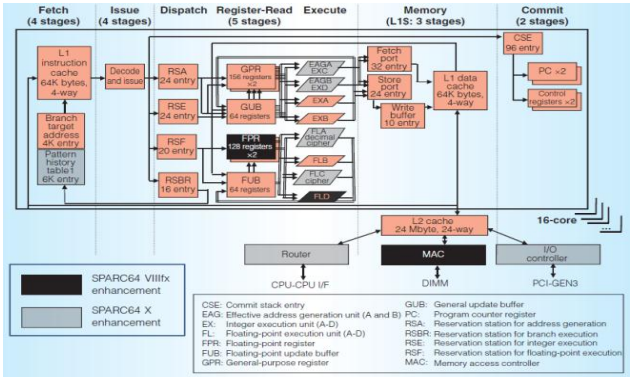

Fig. 1 SPARC64 X processor pipeline.

SPARC64 X processor has about 3-billion (2,950 M) transistors, 4 channels of 1.6 GHz DDR3 controller, 2 ports of PCIe Gen3 controller, and 5 ports of system interface controller. Its memory system uses ccNUMA and a cache coherence control unit for multichip systems, with up to 64 processors, is integrated into L2 cache control module in order to reduce the latency, area, and power consumption [2,8,9,10].

Each one of these cores can run 2 threads in parallel on 4 integer pipelines, two of pipelines can be used as effective address generators (EAG) for load/store pipelines and the other two as floating-point pipelines. Its instruction set architecture is based on SPARC-V9, Joint Programming Specifications (JPS), and HPC-ACE. The memory and Input/Output controllers are embedded in the chip which is directly connected to DDR3-DIMM with a peak throughput of 102 Gbytes/Sec. [2,9]. Five ports of interface between CPUs are embedded to increase performance and reduce production cost.

In pipelining and concurrent systems, many instructions are executed in the pipeline in different stages or concurrently on different executing units, possibly out of order. However, some of these instructions may be data-dependent. If this is the case, a data-hazard occurs. This problem has a negative impact on the system performance, because it reduces the system throughput rate. There are three cases of data-hazard that might occur in a pipeline or concurrent systems. In the first case, known as *Read-After-Write* (RAW), a data-hazard occurs if an instruction tries to read data that a previous instruction in the pipeline did not finish updating yet. In the second case, known as *Write-After-Read* (WAR), a data-hazard occurs if an instruction tries write data that other concurrent instruction did not finish reading yet. This condition can create a problem in a concurrent execution environment. In the third case, known as *Write-After-Write* (WAW), a data-hazard occurs if an instruction tries to write data concurrently with another instruction to the same location. This condition too can create a problem in a concurrent execution environment [1].

IV. SPARK T4 PROCESSOR

SPARC T4 processor has revealed a significant increase in the single-threaded computational capabilities compared to the SPARC T3 while it provides double the per-thread throughput performance. It uses T3's system-on-chip (SoC) components and is socket-compatible with SPARK T3. It improved the overall cryptographic performance and maintained the compatibility with chip multi-threading technology (CMT) and SPARC Version 9 (V9). It also extends the reliability and serviceability (RAS) capabilities over T3. It has 8 SPARC S3 cores, that runs at 3.0 GHz, with the hardware support for 8 threads per each core. Figure 2 shows the SPARC S3 core pipeline, which has 16-stage integer pipe, a 27-stage floating-point graphic pipe, and a 20-stage load-store pipe [2,5].

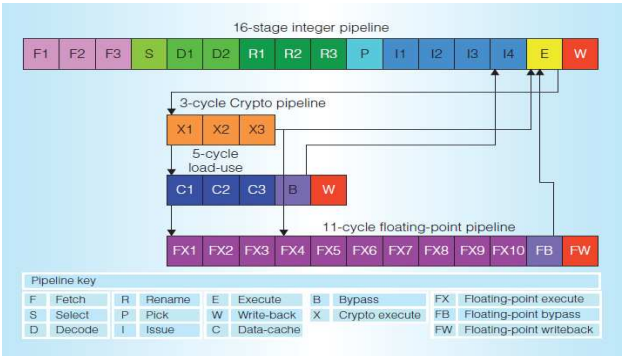

Fig. 2 SPARC S3 processor core pipeline.

SPARC T4 integrates an out-of-order execution, instruction-based cryptography, advanced branch prediction, and a 3-level cache hierarchy. Level 1 and level 2 are private within each core while level 3 is shared among all the 8 cores. These architectural advancements deliver a 5X improvement in single-thread integer performance over SPARC T3 [2,5,6,7].

SPARC T4 has 8 high performance SPARC S3 cores that share an 8-way banked L3 cache with a capacity of 4 Mbyte , each bank is 16-way set associative cache with a line size of 64 bytes. Each core has a full hardware support to execute 8 hardware threads, an out-of-order execution, instruction-based cryptography, 3-lelvels of cache (two private levels, L1 and L2, and a shared level, L3), and a dual-issue execution engine with a dynamically threaded

16-stage integer pipeline and operates at speed of 3.0 GHz. These architectural enhancements offer a 5X improvement in a single-thread integer performance over T3. The shared L3 cache is connected to two dual-channel, on-chip DDR3-1066 memory controllers that communicate with Dual In-line Memory Modules (DIMMs) via four high-speed serial links. SPARC T4 supports two (x8) PCI Express Gen 2 ports, provides dual 10-Gbit Ethernet ports, and its coherency links support up to 4 SPARC T4 processors in a system without any extra glue logic [2,5,6,7].

V. ARCHITECTURE EVALUATION

Oracle® continues to be successful in addressing the growing computing power needed for the industry's commercial workloads by consistently improving the throughput performance of the SPARC processors family. Figure 3 shows the throughput and the single-thread performance of the T4 processor vs. its predecessors T1-T3 [2,6].

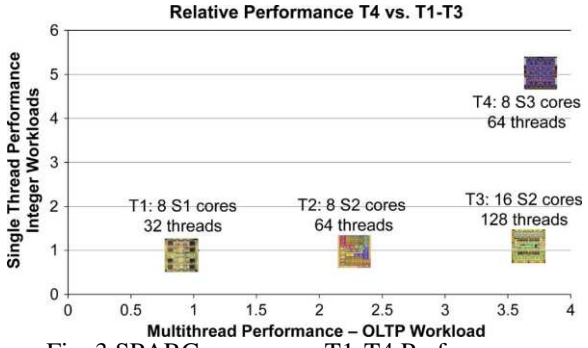

Fig. 3 SPARC processors T1-T4 Performance.

The throughput, which is based on the on-line transaction process(OLTP) benchmark TPC-C, has doubled with the increasing of the on-chip thread counts for T1 to T3, while a single-thread performance remained nearly constant when running integer workloads. To bridge this gap between the single-thread and throughput performance, T4 processor uses a new out-of-order, dual-issue, high-frequency core (S3). This improvement in T4's single-thread performance results in better scalability that can address a broader array of applications and a faster response time for each thread. T4 demonstrations 5 times improvement for single-thread performance with integer workloads. The physical design enhancement of T4 processor and the extensive power management features allows it to deliver the largest single-thread performance improvement in SPARC history while maintaining the same high throughput in the same power envelope of its predecessor [2,5,6]. Figure 4 shows the performance of SPARC64 X compared to the previous SPARC64 processors based on standard benchmarks and the power efficiency of software on chip (SWoC) [2,9].

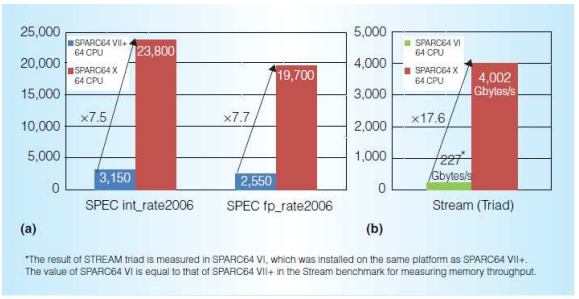

Fig. 4 SPARC64 X Performance.

SPARC64 X demonstrations 7 times better on integer and floating-point throughput (Figure 4-a) and more than 17 times better on memory throughput (Figure 4-b) than the previous SPARC64. This great performance improvement is achieved by the new microarchitecture major enhancements. For example, the pattern history table is newly implemented to improve the branch prediction. The increase in size of the reservation stations and various queues has improved the integer performance. Also the new engines for SWoC such as cipher and decimal operation are implemented in floating-point pipelines [2,9].

VI. CONCLUSION

Computer system performance through evolutional methods has almost reached its physical limitation and researchers are using revolutional methods such as pipelining and parallel processing to achieve the mandatory enhancement of system performance by using advanced processor architectures. Advanced microarchitecture such as that of the Niagara family of processors that has been pioneered by Oracle® were discussed. An overview of some of the important aspects of such architectures was presented while we stressed on the pipelining techniques that lead to such a system performance enhancement.

REFERENCES

[1] Hwang, K., "Advanced Computer Architecture: Parallelism, Scalability, Programmability", McGraw-Hill, 1993, ISBN 0-07-031622-8.
[2] Krad, H., "Performance Enhancement of Computer Systems Through Revolutional Approaches", *The Second International Conference on Technological Advances in Electrical, Electronics and Computer Engineering* (TAEECE2014), 2014, pp 59-64.
[3] Shiva, S. G., "Advanced Computer Architectures", CRC Taylor & Francis, 2006, ISBN 0-8493-3758-5.
[4] Stallings, W., "Computer Organization & Architecture: Designing for Performance", 6th Edition, Prentice-Hall, 2003, ISBN 0-13-049307-4.
[5] M. Shah et al., "SPARC T4: A Dynamically Threaded Server-on-a-Chip," *IEEE Micro*, vol. 32, no. 2, pp. 8–19, 2012.
[6] J. L. Shin et al., "The Next Generation 64b SPARC Core in a T4 SoC Processor," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 82–90, 2013.
[7] J. L. Shin et al., "The 3.0GHz 64-thread SPARC T4 processor," *IEEE Asian Solid State Circuits Conference (A-SSCC)*, pp. 21–24, 2012.
[8] R. Kan et al., "The 10th Generation 16-Core SPARC64 X Processor for Mission Critical UNIX Server," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 32–40, 2014.
[9] T. Yoshida et al., "SPARC64 X: Fujitsu's New-Generation 16-Core Processor for Unix Servers," *IEEE Micro*, vol. 33, no. 6, pp. 16–24, 2013.
[10] R. Kan et al., "A 10th generation 16-core SPARC64 processor for mission-critical UNIX server," *2013 IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, pp. 60–61, 2013.