Memristor-Based Multithreading

Shahar Kvatinsky, Yuval H. Nacson, Yoav Etsion, Eby G. Friedman, Avinoam Kolodny, and Uri C. Weiser

Abstract— Switch on Event Multithreading (SoE MT, also known as coarse-grained MT and block MT) processors run multiple threads on a pipeline machine, while the pipeline switches threads on stall events (*e.g.*, cache miss). The thread switch penalty is determined by the number of stages in the pipeline that are flushed of in-flight instructions. In this paper, Continuous Flow Multithreading (CFMT), a new architecture of SoE MT, is introduced. In CFMT, a multistate pipeline register (MPR) holds the microarchitectural state of multiple different threads within the execution pipeline stages, where only one thread is active at a time. The MPRs eliminate the need to flush in-flight instructions and therefore significantly improve performance. In recent years, novel memory technologies such as Resistive RAM (RRAM) and Spin Torque Transfer Magnetoresistive RAM (STT-MRAM), have been developed. All of these technologies are nonvolatile, store data as resistance, and can be described as "memristors." Memristors are power efficient, dense, and fast as compared to standard memory technologies such as SRAM, DRAM, and Flash. Memristors therefore provide the opportunity to place the MPRs physically within the pipeline stages. A performance analysis of CFMT is compared to conventional SoE MT processors, demonstrating up to a 2X performance improvement, while the operational mechanism, due to the use of memristors, is low power and low complexity as compared to conventional SoE MT processors.

Index Terms — memristor; multithreaded processors; phase change memory; RRAM, STT-MRAM.

1 Introduction

Multithreading processors have been used to improve performance in a single core for the past two decades. One low power and low complexity multithreading technique is Switch on Event multithreading (SoE MT, also known as coarse grain multithreading and block multithreading) [1], [2], [3], [20], where a thread runs inside the pipeline until an event occurs (*e.g.*, a long latency event like a cache miss) and triggers a thread switch. The state of the replaced thread is maintained by the processor, while the long latency event is handled in the background. While a thread is switched, the in-flight instructions are flushed. The time required to refill the pipeline after a thread switch is refered to as the switch penalty. The switch penalty is usually relatively high, makes SOE MT less popular than simultaneous multithreading (SMT) [18] and fine-grain multithreading (interleaved multithreading) [4]. While fine-grain MT is worthwhile only for a large number of threads, the performance of SMT is limited in practice due to limitations on the number of supported threads (*e.g.*, two for Intel Sandy Bridge [5]).

In this paper, Continuous Flow Multithreading (CFMT), a novel microarchitecture, is proposed. The primary concept of CFMT is to support SoE MT for a large number of threads through the use of multistate pipeline registers (MPRs). These MPRs store the intermediate state of all instructions of inactive threads, eliminating the need to flush the pipeline on thread switches. This new machine is as simple as a regular SoE MT, and has higher energy efficiency while improving the performance as compared to regular SoE MT.

Hirst et al extends the SoE MT to differential multithreading (dMT) [19], proposing two threads running simultaneously in a single scalar pipeline for low cost microprocessors. CFMT takes a broader view of advanced SoE MT microarchitectures. CFMT extends SoE MT by enabling the use of numerous threads using multistate pipeline registers in deep pipeline machines. CFMT is applicable to any execution event that can cause a pipeline stall.

The development of new memory technologies, such as RRAM (Resistive RAM) [6] and STT-MRAM (Spin-Transfer Torque Magnetoresistive RAM) [7], enables MPRs since these devices are located in metal layers above the logic cells and are fast, dense, and power efficient. These memory technologies are referred to as memristors [8], [9].

The remainder of this paper is structured as follows: the microarchitecture of a conventional SOE MT is described and CFMT is proposed in section 2, the MPR is presented in section 3, emerging memory technologies and the basic structure of a memristor-based MPR are described in section 4, and a performance analysis for SOE MT and CFMT is presented in section 5, showing 2X theoretical performance improvements as compared to conventional SOE MT. The paper is summarized in section 6.

- *S. Kvatinsky, Y. H. Nacson, A. Kolodny, and U. C. Weiser are with the Electrical Engineering Department, Technion – Israel Institute of Technology, Haifa, Israel 32000. E-mail: skva@ tx.technion.ac.il*
- *Y. Etsion is with the Electrical Engineering and Computer Science Departments, Technion – Israel Institute of Technology, Haifa, Israel 32000.*
- *E. G. Friedman is with the Electrical Engineering and Computer Science Department, University of Rochester, Rochester, NY 14627.*

Manuscript received 27th November 2012.

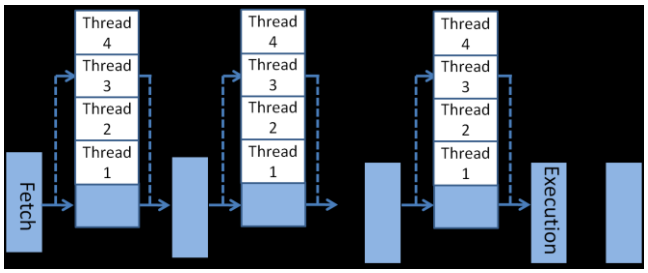

Fig. 1. Continuous Flow Multithreading (CFMT) pipeline structure. A set of multistate pipeline registers (MPRs) is located between every two pipeline stages. Each MPR maintains a single bit of the state of an instruction from all threads. The number of MPRs is the number of bits required to store the entire state of an instruction in the specific pipline stage.

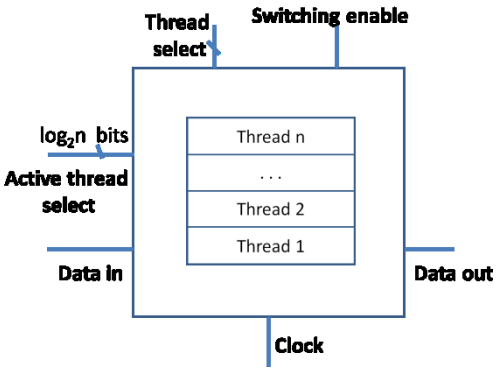

Fig. 2. The logic structure of a multistate pipeline register (MPR). An MPR maintains a single bit of the state of an instruction from all threads (stores *n* bits of data), where only one thread is active at a time. The MPR is synchronized by the processor clock and can switch the active thread.

2 Continuous Flow Multithreading (CFMT)

To reduce the thread switch penalty, a new thread switching mechanism for SOE MT is proposed. In CFMT, pipeline registers are replaced by MPRs, as shown in Figure 1. For each pipeline stage, an MPR stores the state of the instructions from all threads. Thus, in the case of a thread switch, there is no need to flush all subsequent instructions. The processor saves the state of each instruction from the switched thread in the relevant MPR in each pipeline stage, while handling the operation of the long latency instruction in the background. Instructions from the new active thread are inserted into the pipeline from the MPR, creating a continuous flow of instructions within the pipeline. When no thread switching is required, the pipeline operates as a regular pipeline and each MPR operates as a conventional pipeline register. When the long latency instruction is completed, the result is written directly into the MPR in the background. In CFMT, the thread switch penalty is determined by the time required to change the active thread in the MPR, i.e., the time required to read the state of the new, previousely inactive thread from the MPR. For a fast MPR, the thread switch penalty is significantly lower than in conventional SOE MT and the performance therefore increases significantly.

3 Multi-State Pipeline Register (MPR)

The logic structure of a multistate pipeline register (MPR) is shown in Figure 2. Each MPR stores data for multiple threads, one bit per thread. The total size of an MPR is therefore *n* bits, where *n* is the maximum number of threads. For each pipeline stage, the state of the instruction is stored in a set of MPRs with common control signals for thread management and switching. The MPR has one active thread (the current thread) for which the data can be read and written during operation of the processor, as in a regular pipeline register. During a thread switch, the active thread changes while the data of the previously active thread is maintained in the MPR. The MPR can therefore store data for all threads running in the machine. The time required to change the active thread in the MPR depends on the specific circuit structure of the MPR. This time determines the thread switch penalty of CFMT. A typical thread switch penalty in CMFT is in the range of 1 to 3 clock cycles, a significant improvement as compared to SOE MT (typically 8 to 15 clock cycles).

4 Emerging Memory Technologies

Over the past decade, new technologies have been considered as potential replacements for the traditional SRAM/DRAM-based memory system to overcome scaling issues, such as greater leakage current. These emerging technologies include PCM (Phase Change Memory) [10], PMC (Programmable Metallization Cell, also known as CBRAM) [11], FeRAM (Ferroelectric RAM) [12], RRAM (Resistive RAM) [9], and STT-MRAM (Spin Transfer Torque Magnetoresistive RAM) [13].

While the physical mechanism for these emerging memory technologies is different, all of these technologies are nonvolatile with varying resistance and can therefore be considered as memristors [8]. These emerging memory technologies are fabricated by introducing a special insulator layer between two layers of metal which can be integrated into a CMOS process, stacked vertically in multilayer metal structures physically above the active silicon transistors. This fabrication technique provides a high density of memory bits above a small area of active silicon. Memristive memory cell sizes are approximately 1 to 4 F^2 for RRAM and 8 to 45 F^2 for STT-MRAM, as compared to SRAM (60 to 175 F^2) and DRAM (4 to 15 F^2) [14], where F is the minimum feature size in the technology.

RRAM and STT-MRAM are both relatively fast [15]. STT-MRAM does not exhibit any endurance issues, while it is believed that the endurance issue of RRAM will be overcome in the near future [16]. Since memristors are dense, fast, and power efficient, these devices are attractive for use within the processor as an MPR. The basic structure for a set of memristor-based MPRs is shown in Figure 3.

For a memristor-based MPR, each thread has its own memristor-based layer, while the bottom CMOS layer is used for the active thread running within the pipeline. The bottom layer consists of standard CMOS pipeline registers, compatible with CMOS logic. During a thread

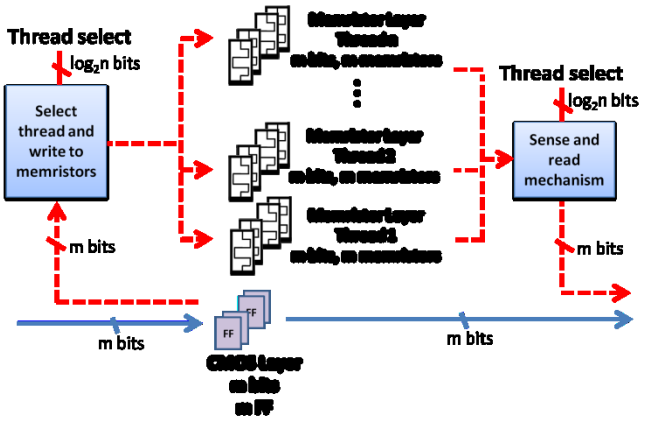

Fig. 3. Set of memristor-based multistate pipeline registers (MPRs). Each thread has its own memristor-based layer, where every bit is stored in a single memristor. The active thread is located in the bottom CMOS layer. During regular operation of the pipeline, only the CMOS layer is active (blue line) and all memristor-based layers are disabled, exploiting the nonvolatility of the memristors to save power. During a thread switch (red dashed line), the data from the CMOS layer is written into the relevant memristor-based layer, while the state of the new active thread is read and transferred to the next pipeline stage.

switch, data is copied from the CMOS layer to a specific memristor-based layer that corresponds to the previously active thread. The data from the new active thread is read into the next pipeline stage that receives the state of the new thread. When no thread switch occurs, only the bottom CMOS layer is active and the memristor layers are in standby mode. It is possible to completely disable the memristor layers and save power due to the nonvolatility of memristors.

To determine the thread switch penalty for a memristor-based MPR, only sensing the memristor layer of the new active thread is considered since the copy operation of the bottom CMOS layer to a memristor layer can be masked using buffers. This latency is determined by the read time of a memristor (sensing the data in the memristive layer). Due to the high density of memristors, our preliminary design of the memristor-based MPR shows that the area overhead can be neglected (less than 0.1% of the pipeline area for 16 active threads [23]). This overhead is primarily due to the write mechanism and can be further optimized by separating the read and write mechanisms.

5 Performance Analysis

The performance (in CPI - cycles per instruction) of an SoE processor depends upon whether the number of threads is sufficient to overlap long latency events. Two regions of operation exist in SoE processors, depending upon the number of threads running in the machine. The *unsaturated region* is the region where the number of threads is fewer than the number required for concealing a long latency event. The behavior of the pipeline in this region is illustrated in Figure 4a. The analytic model assumes that the execution behavior in the pipeline is periodic. The period is determined by the execution of $1/r_m$ instructions from the same thread, where r_m is the average

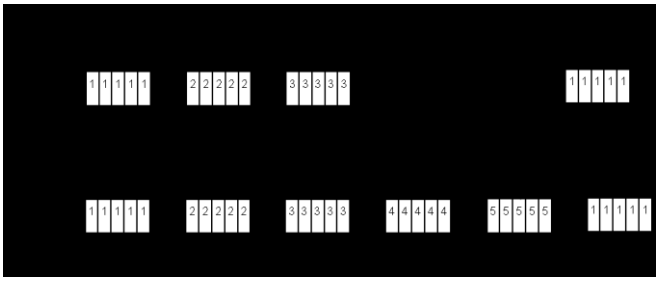

Fig. 4. The executed instructions in the two regions: (a) the unsaturated region, and (b) the saturation region. Each block is an instruction. The numbers indicate the thread number.

fraction of memory operations in the instruction stream. One instruction is a long latency instruction (*i.e.*, the instruction that triggers the thread switch; in this paper, an L1 cache miss is assumed as the trigger, with a miss penalty of P_m cycles) and the remaining instructions are low latency instructions with an average CPI of CPI_{ideal}. During execution of the long latency instruction, other instructions from different threads run within the machine. For these instructions, a periodic behavior is again assumed which also triggers a thread switch. For the unsaturated region, it is assumed that there is an insufficient number of instructions to overlap the P_m cycles required to execute the long latency instruction. The *CPI* in the unsaturated region is

$$CPI_{unsat} = \frac{CPI_{ideal} + P_m \cdot r_m \cdot MR(n)}{n}, \qquad (1)$$

where n is the number of threads running in the machine and $MR(n)$ is the miss rate of the L1 cache. Note that CPI_{unsat} is limited by CPI_{sat}, determined in (2).

When a sufficient number of threads runs on the machine, the long latency instruction can be completely overlapped, and a second region, named the *saturation region*, is reached. In the saturation region, the thread switch penalty (P_s clock cycles) influences the behavior, which effectively limits the number of threads (above a specific number of threads there is no change in performance). The behavior of the pipeline in the saturation region is illustrated in Figure 4b. Assume all of the threads exhibit the same average behavior and $P_m >> CPI_{ideal}/r_m$ (*i.e.*, the miss penalty is significantly longer than the execution time of the short latency instructions). The *CPI* in the saturation region is

$$CPI_{sat} = CPI_{ideal} + P_s \cdot r_m \cdot MR(n) \qquad (2)$$

In a conventional SOE MT, the switch penalty P_s is determined by the number of instructions flushed during each switch. In CFMT, however, the switch penalty is the MPR read time T_m, *i.e.*, the time required to read the state from the MPR and transfer this state to the next pipeline stage. In the case of a memristor-based MPR, the switch penalty is the time required to read the data from the memristor layer. From (2), if the value of T_m is lower than P_s, the performance of the processor in the saturation region is significantly improved, where the speedup is

$$Speedup_{sat} = 1 + \frac{r_m \cdot MR(n)}{CPI_{ideal} + T_m \cdot r_m \cdot MR(n)} \cdot (P_s - T_m). \qquad (3)$$

Note that in the unsaturated region, the exact *CPI* of the

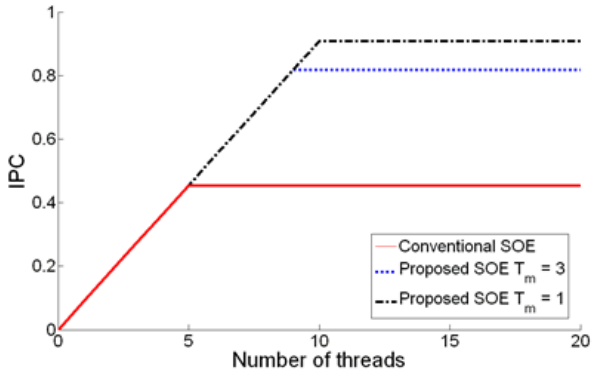

Fig. 5. The IPC of the Continuous Flow MT (CFMT) as compared to a conventional SoE MT processor (straight line). The memristor read time, which determines the thread switch penalty, is three clock cycles and one clock cycles. The IPC of CFMT is twice greater (2X improvement) than a conventional SOE MT for T_m = 1 cycle for a constant miss rate, MR = 0.25, r_m = 0.25, P_s = 20 cycles, and P_m = 200 cycles.

CFMT is slightly better (lower) than a conventional SoE MT processor due to the improved switch penalty. The IPC of the proposed machine as compared to a conventional SoE machine is shown in Figure 5. The performance of the proposed machine exhibits a 2X performance improvement for a constant miss rate when operating in the saturation region. For varying miss rates (particularly with large P_m), the behavior of the *CPI* is similar to the behavior reported in [17]. Preliminary simulations have been performed on GEM5 [21], exhibiting a saturation performance improvement of approximately 50% for the SPEC MCF benchmark [22].

6 Conclusions

In this paper, a new architecture for a multithread processor, Continuous Flow Multithreading (CFMT), is proposed. This architecture is based on multi-state pipeline registers (MPR) to save the thread state in the case of an event (*e.g.*, an L1 cache miss). CFMT greatly reduces the thread switch penalty and eliminates the wasted energy of repeating instructions.

An analytic model of the performance of a conventional SoE MT and the CFMT is described. It is shown that a CFMT processor can exhibit up to a 2X performance improvement as compared to a conventional SoE MT. CFMT has a simple control mechanism and can therefore maintain more threads than modern SMT processors. The performance of the CFMT architecture is comparable to SMT processors with lower complexity and power consumption.

Emerging memristive technologies enhable low power MPRs that can maintain a large number of threads in the same area of the regular pipeline registers. The memristor-based MPR demonstrates the attractiveness of memristors as a means to overcome power and performance deficiencies of existing system structures, and opens opportunities for novel processor microarchitectures.

Acknowledgments

This work was supported by the Hasso Plattner Institute. The authors thank Ravi Patel for his comments and area overhead estimation and to Nimrod Wald and Guy Satat for their help in evaluating the architecture.

References

[1] R. Gabor, S. Weiss, and A. Mendelson, "Fairness Enforcement is Switch On Event Multithreading," *ACM Transactions on Architecture and Code Optimization*, Vol. 4, No. 3, Article 15, pp. 1-34, September 2007.
[2] J. M. Borkenhagen, R. J. Eickemeyer, R. N. Kalla, and S. R. Kunkel, "A Multithreaded PowerPC Processor for Commercial Servers," *IBM Journal of Research and Development*, Vol. 44, No. 6, pp. 885-898, November 2000.
[3] C. McNairy and R. Bhatia, "Montecito - The Next Product in the Itanium Processor Family," *Hot Chips* 16, August 2004.
[4] B. J. Smith, "Architecture and Applications of the HEP Multiprocessor Computer System," *Proceedings of SPIE Real Time Signal Processing IV*, pp. 241-248, 1981.
[5] L. Gwennap, "Sandy Bridge Spans Generations," *Microprocessor Report* (www.MPRonline.com), September 2010.
[6] R. Waser and M. Aono, "Nanoionics-based Resistive Switching Memories," *Nature Materials*, Vol. 6, pp. 833-840, November 2007.
[7] Y. Huai, "Spin-Transfer Torque MRAM (STT-MRAM) Challenges and Prospects," *AAPPS Bulletin*, Vol. 18, No. 6, pp. 33-40, December 2008.
[8] L. O. Chua, "Memristor – the Missing Circuit Element," *IEEE Transactions on Circuit Theory*, Vol. 18, No. 5, pp. 507-519, September 1971.
[9] R. Waser, R. Dittmann, G. Staikov, and K. Szot, "Redox-Based Resistive Switching Memories – Nanoionic Mechanisms, Prospects, and Challenges," *Advanced Materials*, Vol. 21, No. 25-26, pp. 2632-2663, July 2009.
[10] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," *Proceedings of the Annual International Symposium on Computer Architecture*, pp. 2-13, June 2009.
[11] M. N. Kozicki and W. C. West, "Programmable Metallization Cell Structure and Method of Making Same," *U. S. Patent No. 5,761,115*, June 1998.
[12] J. F. Scott and C. A. Paz de Araujo, "Ferroelectric Memories," *Science*, Vol. 246, No. 4936, pp. 1400-1405, December 1989.
[13] Z. Diao et al, "Spin-Transfer Torque Switching in Magnetic Tunnel Junctions and Spin-Transfer Torque Random Access Memory," *Journal Of Physics: Condensed Matter*, Vol. 19, No. 16, pp. 1-13, 165209, April 2007.
[14] International Technology Roadmap for Semiconductor (ITRS), 2009.
[15] A. C. Torrezan, J. P. Strachan, G. Medeiros-Riveiro, and R. S. Williams, "Sub-Nanosecond Switching of a Tantalum Oxide Memristor," *Nanotechnology*, Vol. 22, No. 48, pp. 1-7, December 2011.
[16] J. Nickel, "Memristor Materials Engineering: From Flash Replacement Towards a Universal Memory," *Proceedings of the IEEE International Electron Devices Meeting*, December 2011.
[17] Z. Guz, E. Bolotin, I. Keidar, A. Kolodny, A. Mendelson, and U. C. Weiser, "Many-Core vs. Many-Thread Machines: Stay Away From the Valley," *Computer Architecture Letters*, Vol. 8, No. 1, pp. 25-28, May 2009.
[18] D. M. Tullsen, S. J. Eggers, and H. M. Levy, " Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proceedings of the Annual International Symposium on Computer Architecture*, pp. 392-403, June 1995.
[19] J. W. Haskins, K. R. Hirst, and K. Skadron, " Inexpensive Throughput Enhancement in Small-Scale Embedded Microprocessors with Block Multithreading: Extensions, Characterization, and Tradeoffs," *Proceedings of the IEEE International Conference on Performance, Computing, and Communications*, pp. 319- 328, April 2001.
[20] M. K. Farrens and A. R. Pleszkun, "Strategies for Achieving Improved Processor Throughput," *Proceedings of the Annual International Symposium on Computer Architecture*, pp. 362-369, May 1991.
[21] http://www.m5sim.org/
[22] SPEC CPU2006 benchmark suite. http://www.spec.org/cpu2006/
[23] Private discussion with Ronny Ronen, Intel.