Buffer-Controlled Cache for Low-Power Multicore Systems

Marven Calagos
Department of Electrical Engineering
University of Texas Rio Grande Valley
Edinburg, Texas, U.S.A.

Yul Chu
Department of Electrical Engineering
University of Texas Rio Grande Valley
Edinburg, Texas, U.S.A.

***Abstract*— This paper proposes a buffered dual-access mode cache to reduce power consumption in multicore embedded systems. This cache is called as Buffer-Controlled Cache (*BCC*). The proposed cache includes a pre-cache buffer to determine how to access the cache in two different ways. The BCC scheme shows better prediction rates and lower power consumption than other popular low-power caches, such as Phased cache and Way-prediction cache. We have used Multi2Sim and McPAT simulators to evaluate various cache systems by using the multicore-oriented Parsec benchmark programs. Our experimental results show that the proposed cache scheme improves the power consumption by 54% over the other popular cache scheme, such as Way-prediction cache.**

Keywords—microprocessors; cache; low power; multicore; Multi2Sim; McPAT; Parsec

I. INTRODUCTION

Current computers and mobile devices play an important role in daily use, either for entertainment, communication, or work. These devices are multifunctional and have great demands because of its processing capabilities. Multicore processors can fit this role and have been widely used in general purpose computing area, even in mobile devices. For such hand-held battery-powered devices, since low-power consumption has been a critical factor to design, there has been a great need for research and development in this area [1]. Another factor that limits the performance of multicore processors is cache coherence for shared or distributed memory [2]: That is because, when different cores with private caches share a common memory resource, inconsistent data may be raised. Currently, active research in this area is ongoing to mitigate the problem. In this paper, we propose the low-power BCC scheme and have done several experiments through various arrangements of cores, L1 caches, and L2 caches to determine the optimal cache configuration for multicore processors. We believe the optimized cache configuration will affect cache coherence and overall performance to a better way.

According to [3], low-power cache improvements can be made from the architectural level. As the complexity of processor design increases, it will become more complicated to estimate power consumption. This paper uses a software-based simulator, McPAT, to compute power consumption for various cache schemes since it is easily updated and modified.

To improve L1 cache performance, it has been popular to use highly-associative caches like 16- or 32-way since they have an inherent property that provides better performance by reducing the conflict misses: Conflict misses develop due to imperfect allocation of entries in a cache [9]. However, highly-associative caches significantly increase power consumption because of the simultaneous accesses to all the banks in parallel, i.e., the *n*-way cache has *n* banks to access simultaneously. Therefore, this paper approaches how to accurately predict only one bank out of *n* banks (e.g., *n*-way) and how to restrict access to the predicted bank in reducing power consumption. We found that this can also improve cache latency since the proposed cache can be accessed like a direct-mapped cache.

The rest of the paper is organized as follows: Section 2 explores the related works; Section 3 presents the proposed cache scheme; Section 4 deals with experiment methodology and performance metrics; Section 5 provides the evaluation results and discussion; and finally the conclusions are provided in Section 6.

II. RELATED WORKS

An inhibiting aspect in multicore processors is the ability to exploit parallelism in software programs [4]. If a program is not written with parallelism, then it may not run any faster in a multicore system. However, processor performance for a software application can be increased by using specialized chips or coprocessors for its specific purposes, such as media and graphics application programs. These specialized chips are highly efficient and can bypass the less efficient stages during computation in general processor.

ARM's big.LITTLE technology [5] is a power-optimization processing to deliver peak-performance capacity based on an asymmetric design by combining different types of cores, such as performance and energy-efficient cores, in one chip. In order to improve performance for various applications simultaneously in parallel, all the cores can be used independently. If the load is minimal, only the energy-efficient cores are used. From the software point of view, it appears as a homogeneous multicore processor. These cores can be linked via a high-speed interconnect network. Access to memory via

978-1-5090-1801-7/16/$31.00 ©2016 IEEE

an interconnect can inhibit performance by delaying communication among cores. A solution to this is a high-speed buffer as the form of an L4 cache [6].

Intel's Turbo Boost allows different cores to be completely switched off and the frequency of the remaining threads can be raised [7]. This temporary performance boost is only limited by the thermal capacity of the chip. The ability to control the voltage or frequency among different cores and other processing units can also increase performance and power management.

In multicore system, synchronization can be a hardware solution to the problem of cache coherency [8]. Coherency contributes to major locks and conflicts which result in poor performance since locks and conflicts can occur from inconsistent data because of improperly managed cache coherency. When data is required but not available, a lock or a stall is needed in data processing. As a solution, synchronization is necessary by prioritizing access to memory only to cores that are requesting: The address being accessed by the core is locked to prevent other cores from accessing it. In addition, synchronization can provide fairness in regards to memory access with affecting performance.

At the cache level, several methods have been proposed to reduce power consumption. For example, Nicolaescu et al. [9] attempts to reduce power consumption by allowing access to only one bank in a n-way set-associative cache by using a Way Determination Unit (WDU): The WDU is implemented to exploit high line address locality by recording previously accessed cache line addresses and their way or bank number. Nicolaescu et al. [9] claims that it saves power on average of 66% for an 8-way set-associative cache. Using high-associativity in L1 cache is mostly beneficial for way-prediction schemes and similar designs.

Inoue et al. [10] proposed a low-power cache by combining the speed of conventional cache and the power savings of Phased cache [11] with Way-prediction cache [12]. This method has a performance-aware mode (switching between conventional and Way-prediction) and an energy-aware mode (switching between Phased and Way-prediction). It provides better performance than the original Way-prediction cache. The performance-aware mode reduces performance overhead by 17%. The energy-aware mode reduces power consumption by 73%.

III. BUFFER-CONTROLLED CACHE (BCC)

This paper proposes a cache scheme to reduce power consumption by using a buffer and dual-access modes for multicore systems: those are Most Recently Used (MRU) buffer, Phased-mode, and Way-prediction mode. The MRU buffer determines how the cache is accessed, Phased-mode or Way-prediction mode. Fig. 1 shows the access mode sequence of BCC scheme based on the following steps:

1) Access to the MRU buffer to compare the memory address and the MRU buffer tag
2) If the tags match, access the cache using the Way-prediction mode
3) If not, access it using the Phased-mode
4) In the case of a cache miss, the Phased mode is used.

In the Way-prediction mode, only one tag and one data is accessed (refer to Fig. 3). In the Phased-mode, all the tags and one data are accessed (refer to Fig. 4). The latest *m* MRU tag entries are stored in the buffer, where *m* refers to the number of cache lines. Only the MRU tag of each cache line is stored in the buffer. Fig. 2 shows the address space of an entry for a 16-way (set-associative) cache. For this experiment, all cache configurations will use a 20-bit tag. Therefore, each line of the MRU buffer will also have a 20-bit tag plus four bits (16-way) to determine the location of the MRU way.

As the cache size increases, the tag size of the buffer also increases. However, for a given (fixed) cache size, as the associativity increases, the number of buffer entries will be decreased. For example, a 32KB 4-way cache will have 256 entries in the buffer, but a 32KB 16-way cache will have only 64 entries in the buffer. Therefore, the indexing of the buffer is exactly the same as the main cache, such as 32KB 16-way cache. In Fig. 2, that is why it is unnecessary to have index bits in the address space of the buffer.

In Fig 3, if the tag entry exists in the buffer, the cache is accessed using the Way-prediction mode and will require only one cycle. If not, the cache is accessed using Phased mode and requires two cycles, as shown Fig. 4.

A disadvantage of the original Way-prediction cache is the power and delay penalties incurred during a prediction miss. This is mitigated by the use of an MRU buffer. The Way-prediction mode relies on the MRU information for prediction.

Fig. 1. Buffer-Controlled Cache access pattern

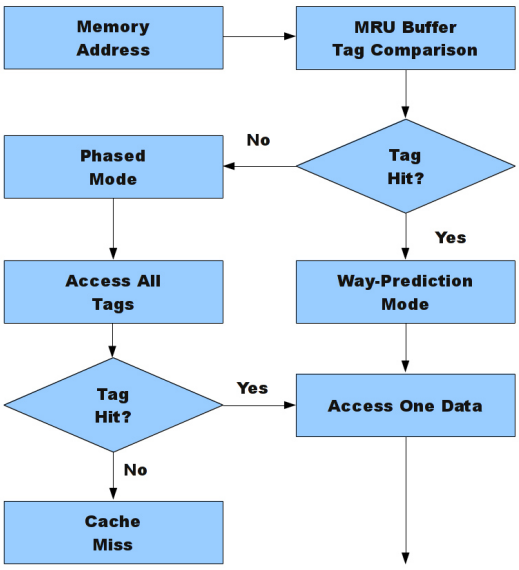


Fig. 2. Address space of a Most Recently Used buffer for a 16-way cache

By comparing the MRU entries before accessing the cache, the power consumption penalty of a prediction-miss can be avoided. The Phased mode access has the benefit of reduced energy because only one data block is accessed, compared to all data blocks of a prediction-miss of the Way-prediction cache.

Because of temporal locality, the MRU buffer entries are the most likely memory locations to be referenced again. Based on spatial locality, the next access data are likely to be located in the same block as the last access. As a result, the Way-prediction mode will have more accesses than Phased-mode. This is desirable since the Way-prediction mode consumes less power and has faster access than the Phased-mode.

Fig. 3. The Way-prediction Mode

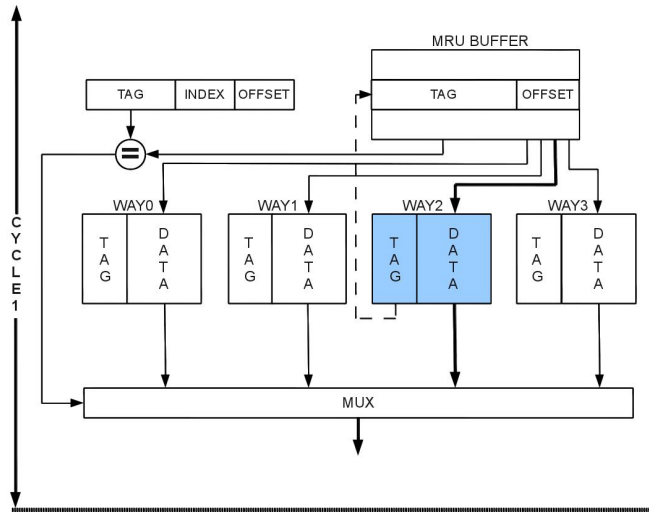
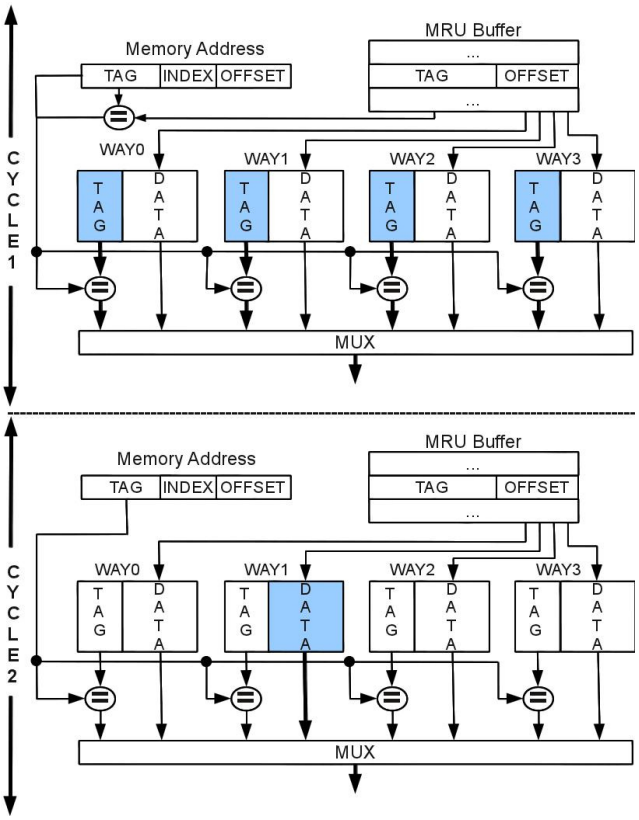


Fig. 4. The Phased Mode



In Fig. 3 and Fig. 4, the MRU buffer is placed between the CPU and the cache. Therefore, a possible delay is considered regardless of buffer size. However, this delay does not negatively affect the Energy-Delay Product (EDP) of the scheme. The EDP (energy delay product) is a common metric used in evaluating microprocessors. The EDP shows the effect of any delay to decrease energy consumption, and takes into account any increased execution time and access time.

The accuracy of the prediction technique is crucial to reducing power consumption. BCC cache incorporates the Way-prediction scheme as it provides a high level of accuracy. Unlike other cache schemes, BCC does not change the performance of the cache since it has been designed to improve power consumption. While the number of accesses to the cache remain the same, cache activity is reduced using the dual-access modes: The Way-prediction mode and the Phased mode. The main difference between BCC cache and the scheme in [10] is the method of deciding how the cache is access, namely - the MRU buffer.

The BCC scheme is evaluated using the cache coherence protocol embedded with Multi2Sim simulator [13]. Multi2Sim implements a 6-state coherence protocol called NMOESI.

IV. Experimental Methodology

The BCC scheme is evaluated for its performance and power consumption using Multi2Sim and McPAT simulators. For the evaluation, the following parameters were used: 64KB cache size, 32 sets, 16-way associativity, and 128B block size. All experiments were evaluated using 2 cores, and 2 threads per core. All other parameters (such as core frequency, latency, technology, etc.) are left as defaults. Varying too many variables would over-complicate the experiment. Most of these parameters are also shared between Multi2Sim and McPAT.

Multi2Sim (version 4.2) [13] was used to implement the cache and the buffer. Multi2Sim is a multicore simulator including memory and interconnect networks. It is used for simulating multicore and multi-threaded systems. Multi2Sim was modified and extended to implement the common definitions of the Phased cache and Way-prediction cache. The buffer shares many elements of a cache and was also implemented as a separate architecture.

McPAT (version 1.3) [14] was used to compute power consumption for the different cache configurations. This simulator models the power, area, and timing for multi-threaded and multicore architectures. McPAT is also compatible with Multi2Sim.

Parsec benchmarks (version 2.1) were used for all simulations [15]. Parsec is a collection of benchmarks that focus on multicore and multi-threaded processors. Parsec excels at stressing the shared-memory paradigm of multicore processors. The selected benchmarks reflect commonly used commercial programs. The benchmarks employ workloads

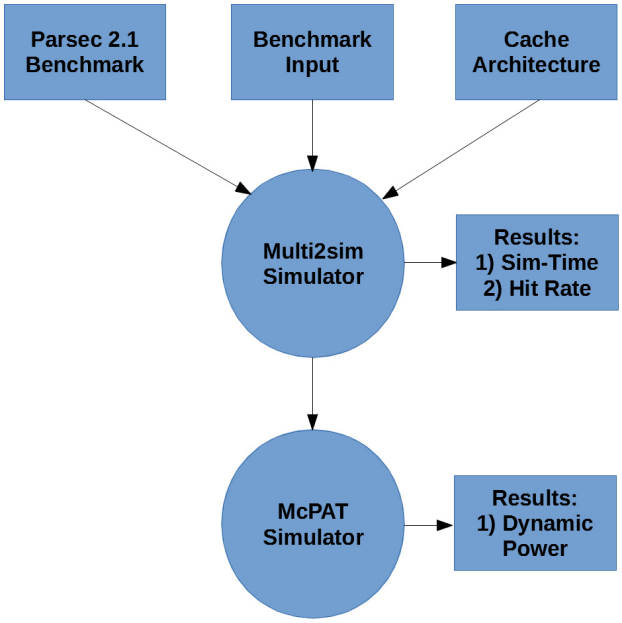
such as systems programs and parallelization models that many other benchmarks lack. New and emerging methods for benchmarking applications are continuously added. Parsec is available to the public and therefore used by many researchers for multicore simulation.

The simulation tools interact well with each other. Multi2sim lacks a proper power simulator. However, all of its outputs parameters can be used by McPAT to calculate power consumption. Furthermore, the Parsec benchmark tools were designed to take full advantage of Multi2Sim. These simulators were selected because of the ability to evaluate the architectural design as a whole. In contrast, cycle-accurate simulators would not be appropriate as the design does not provide hardware specifications at the level required for cycle level analysis.

All of the simulation tools (Multi2Sim, McPAT, and Parsec) were compiled using a modern Linux operation system, Red Hat Enterprise Linux, for the 64-bit little endian architecture. Fig. 5 shows the simulation model used for this experiment: Benchmarks were compiled to be used specifically for Multi2sim. The BCC scheme was implemented in Multi2Sim. The output of Multi2Sim was used to calculate the dynamic power for the simulation through McPAT.

Fig. 6 shows the six multicore cache configurations. We use 7 configurations in total including another configuration, which is a single core with a single thread for comparison purpose. In Fig. 6, the four threads (in Config. 1) share the main memory. Each thread has its own L1 and L2 cache. In Config. 2, each thread has its own L1 cache but shares a L2 cache. In Config. 3, each thread has its own L1 cache, but the cores share a L2 cache and main memory. In Config. 4, within a core, the threads share L1 and L2 caches. But the cores share main memory. In Config. 5, two threads share L1 caches, but the cores share L2 and main memory.

Fig. 5. Simulation Model



Finally, in Config. 6, all cores and threads share L1 and L2 caches as well as main memory. The goal of multi-core simulation is to determine which cache distribution design is optimum for BCC scheme.

V. Evaluation

The experiment was conducted using five benchmarks from the Parsec benchmark suite. These benchmarks were selected for the diverse methods of stressing a multicore processor.

The benchmarks are:

1. Blackscholes - numerical computation using partial differential equations for option pricing
2. Bodytrack - image sequencing simulation for tracking movement of a person
3. Canneal - simulated cache-aware annealing to optimize routing cost of a chip design
4. Fluidanimate - fluid dynamics animation using the smoothed particle hydrodynamics method
5. x264 - a popular video encoding standard using the H.264 codec

Fig. 6. Cache Configurations in Multi2Sim [13]

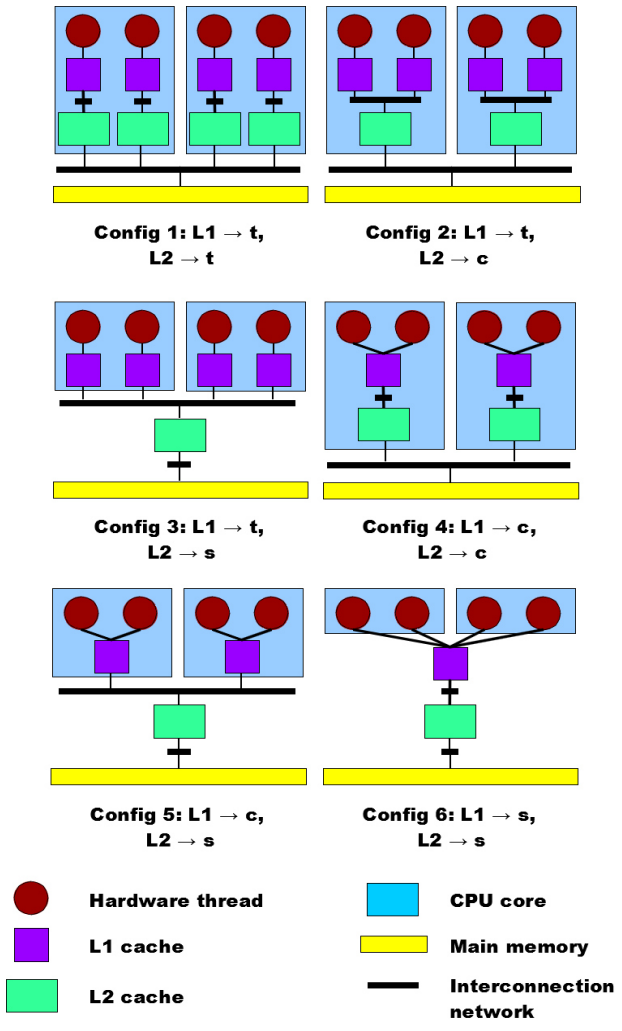


Fig. 7 shows the power consumption of the Way-prediction cache for seven configurations including single core and single thread configuration. In Fig. 7, the downward progression in power consumption (in order from Config. 1 to 6) is expected since the number of cache memories, L1 and L2, are reduced to make low-power consumption (refer to Fig. 6).

Fig. 8 shows the power consumption of the BCC scheme. A significant reduction in power can be seen depending on the benchmark and configurations. The addition of a buffer adds a small amount of delay and power consumption. However, this negated by the significant savings in speed and power consumption in bypassing certain tag and data accesses to the cache. Fig. 9 summaries the results for power consumption reduction for the BCC scheme over the Way-prediction cache.

As can be seen, a minimum reduction of 12% in Config 6 and a maximum of 54% in Config 1. A greater reduction in Config 1 can be seen because of the greater number of cache components used, therefore more opportunities for power savings. In addition, in Config 1, not all cores or threads are used 100% of the time.

Fig. 7. Total power consumption of the Way-prediction cache (watts)

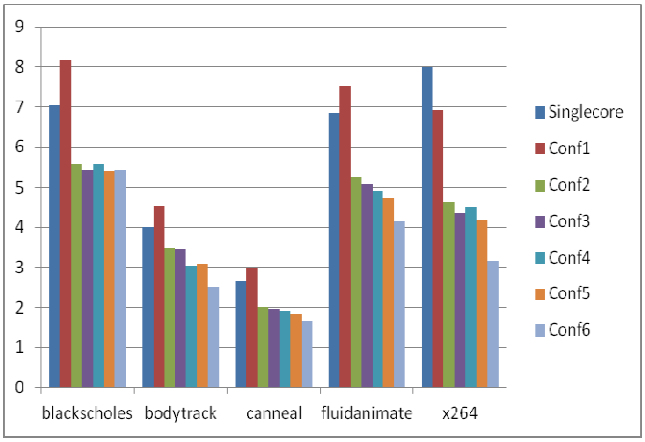


Fig. 8. Total power consumption of the Buffer-Controlled Cache scheme (watts)

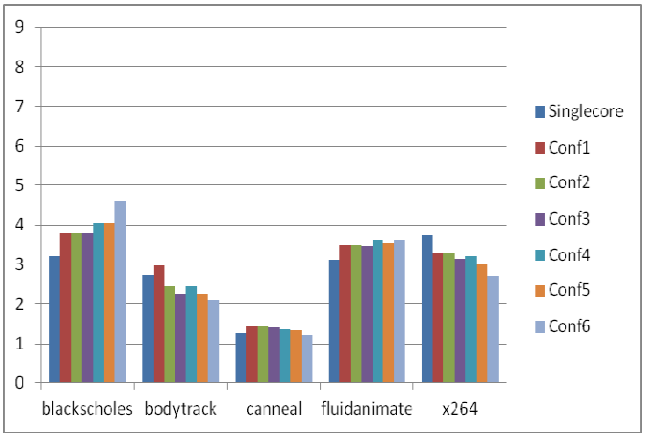


Fig. 9. Power consumption reduction (%)

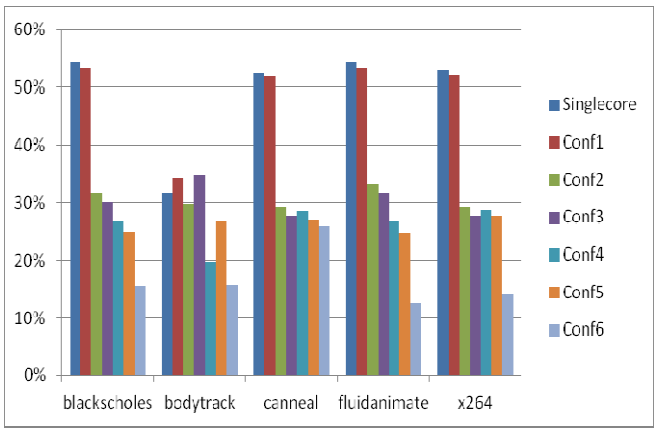


Fig. 10. Total simulation time (seconds)

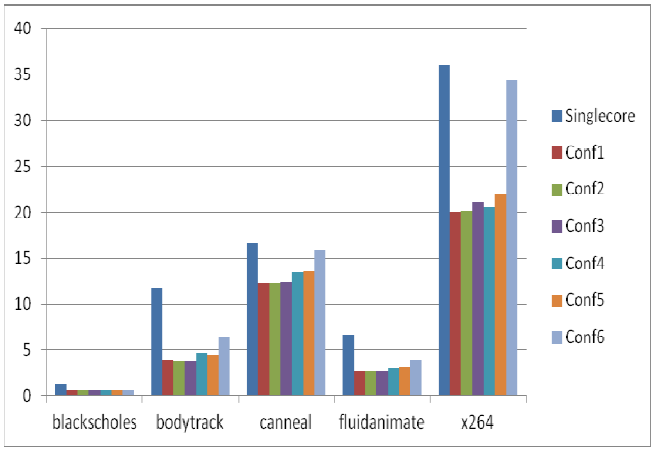


Fig. 10 shows the total simulation time of the BCC scheme. This does not take into account the emulation time, which can be several hours long. Based on the simulation time and power consumption, it is recommended to use Config. 1, Config. 2, or Config. 3 along with the BCC scheme. These configurations provide a balance of power and speed.

VI. CONCLUSION

In this paper, the BCC scheme has been proposed to improve the power consumption for multicore systems, compared to the Phased and Way-prediction caches under multicore environments. The BCC scheme is a dual-access mode cache scheme that uses MRU tag entries in a buffer to determine the access mode, Way-prediction or Phased. By using this scheme, the energy consumption can be reduced with minimal access-time increase. Our experimental results show that the BCC scheme reduces power consumption by 12%-54% for 6 configurations over the Way-prediction cache.

REFERENCES

[1] Q. Wang, Y. Tang, Z. Li, J. Wang, "Design and implementation of the multicore architecture teaching experiment platform," in Advanced Computational Intelligence (ICACI), 2012 IEEE Fifth International Conference on, 18-20 Oct. 2012, pp. 72-78.

[2] A. D. Joshi, N. Ramasubramanian, "Comparison of significant issues in multicore cache coherence," in Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on, 8-10 Oct. 2015, pp. 108-112.

[3] M. Peng, Y. Hu, "A Power Model Combined of Architectural Level and Gate Level for Multicore Processors," in Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on, 16-18 July 2013, pp. 1652-1655.

[4] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, D. Burger, "Executable Dark Silicon Performance Model": http://research.cs.wisc.edu/vertical/DarkSilicon, last access Dec. 3, 2015.

[5] Big.Little Technology, last access on February 1, 2016. http://www.arm.com/products/processors/technologies/biglittleprocessing.php

[6] C. Märtin, "Multicore Processors: Challenges, Opportunities, Emerging Trends," in Proceedings Embedded World Conference 2014, 25-27 February 2014.

[7] Intel Turbo Boost Technology 2.0, last access on January 10, 2016, http://www.intel.com/architecture-and-technology/turbo-boost.

[8] C. Stoif, M. Schoeberl, B. Liccardi, J. Haase, "Hardware synchronization for embedded multi-core processors," in Circuits and Systems (ISCAS), 2011 IEEE International Symposium on , 15-18 May 2011, pp. 2557-2560.

[9] D. Nicolaescu, A. Veidenbaum, A. Nicolau, "Reducing power consumption for high-associativity data caches in embedded processors," in Design, Automation and Test, Europe Conference and Exhibition, 2003, pp. 1064-1068.

[10] K. Inoue, H. Tanaka, "Adaptive Mode Control for Low-Power Caches Based on Way-prediction Accuracy," in IEICE Trans. Fundamentals. vol. E88, Dec. 2005, pp. 3274-3281.

[11] R. K. Megalingam, K. B. Deepu, I. P. Joseph, V. Vikram, "Phased set associative cache design for reduced power consumption," in Computer Science and Information Technology, ICCSIT 2009. 2nd IEEE International Conference on , 8-11 Aug. 2009, pp. 551-556.

[12] K. Inoue, T. Ishihara, K. Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," in Low Power Electronics and Design, 1999 International Symposium on, 17-17 Aug. 1999, pp. 273-275.

[13] R. Ubal, B. Jang, P. Mistry, D. Schaa, D. Kaeli, "Multi2Sim: A Simulation Framework for CPU-GPU Computing," in Proc. of the 21st International Conference on Parallel Architectures and Compilation Techniques, Sep. 2012.

[14] Sheng Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on , vol., no., 12-16 Dec. 2009, pp.469-480.

[15] C. Bienia, "Benchmarking Modern Multiprocessors", Princeton University, Jan. 2011.