ISSN: 2321-7782 (Online)
Volume 2, Issue 1, January 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Scheduling- An Instrument to Optimize Cache Utilization for NVM's

Rani Shriram Lande[1]
Department of Computer Science and Engineering
Prof Ram Meghe College of Engineering and Management
Badnera, Amravati University
India

Dr. M. S. Ali[2]
Prof Ram Meghe College of Engineering and Management
Badnera, Amravati University
India

Abstract: ***Embedded systems are highly used in every field. Embedded system often required to provide Real-Time response. So, to deal with high volume of data embedded system uses flash memory as main memory. This paper presents the overview of the some of the techniques used to reduce the write operations on flash memory and increase the lifetime of flash memory.***

Keywords: Cache, Victim Cache, Flash Memory, Data Migration, Data Recomputation.

I. INTRODUCTION

Embedded system is frequently a system that is implemented for a particular purpose. In Real-Time environment, embedded system also required to provide Real-Time response. For that many embedded systems deal with large arrays of data that must be kept in memory. Embedded system requires large amount of memory for storage, as it deals with large amount of data. Large memory consumes more power. So, Non-Volatile Memory (NVM) such as flash memory can be used as main memory. Flash memory is structurally and functionally very similar to an EPROM computer storage device except for the fact that it is electrically erasable [1]. Flash memory has low-power consumption and high density.

Embedded system programs have very well ordered memory accesses that are known at compile time. To reduce the power consumption and to increase the density of the main memory task scheduling techniques are preferred. A scheduling helps to improve total memory accessed time and reduce write activity on NVM's. Scheduling also helps to solve large problems in a reasonable amount of time. Since the compiler can know the memory accesses of embedded system applications, it can schedule the tasks of the applications to maximize the effectiveness of the cache. ILP formulation is used to schedule tasks and concatenation scheduling to solve instances that are too large to solve with ILP.

II. PROBLEM DEFINITION

Flash memory has low-power consumption and high density. However, flash memory having two disadvantages. First, the write operations are much slower than read operations. Second, the lifetime of flash memory is depends on number of write operations. To increase the lifetime of flash memory there is need to reduce the write operations on flash memory. For that different techniques are used. All techniques target different architecture.

III. RELATED WORK

The use of caches in computing systems has been a widely adopted method for addressing long memory access latency, which has been exacerbated by rapid technology scaling. The advent of Chip Multiprocessors (CMPs) has increased the pressure on achieving good cache performance [31].

In [15], Liang Shi and Chun Jason xue et al. proposed smart victim cache architecture which can reduce the write activities on flash memory when applying flash memory as main memory. Authors purposed cache architecture which consists of victim

cache and write buffer only dirty pages/blocks are kept in victim cache. Victim cache is used to keep the dirty data from the last level cache (LLC), which is organized similar to traditional victim cache, but with an additional replace bit.

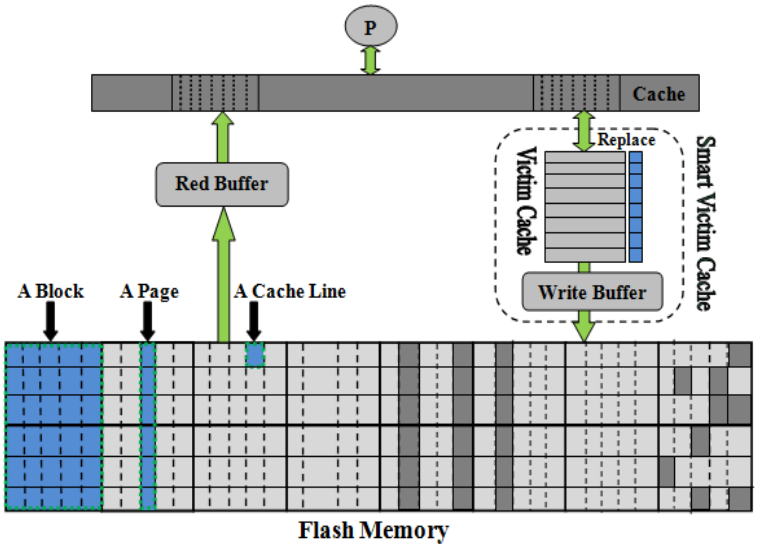

Fig 1. Architecture with victim cache

The write buffer keeps batch of lines from victim cache in queue based on the address of the lines. In this approach, only dirty blocks are kept in the victim cache, which can fully utilize the dirty blocks before the dirty blocks being written back to the main memory. To manage the victim cache ,to reduce write activities by careful selection of the set of cache lines three management techniques of victim cache is used. The techniques are as follows: 1] A Max lines-based approach: This approach used array to record the line counts information for each page. 2] History based approach: This approach proposed only write back data that have been least recently used and does not need a counter array. 3] Enhanced history based approach: In this approach, enhance the history based by first comparing the page tag of the evicted live from LLC and the selected line in LRU position of victim cache.

If both pages is equal to each other then the evicted line replaced tag would be set, if not it follow the history based approach to replace the lines and move forward to the MRU position. Authors presented experimental results which indicating that this approach can reduce write activities on flash main memory by 65.38% on average compared to traditional architecture.

The authors [16], Jingtang Hu and Yi He, Meikang Qie introduced another technique to reduce write activity on flash memory. Authors purposed technique for embedded chip multiprocessors (CMP) with scratch pad memory (SPM) ana non volatile main memory. The architecture designed by authors is as follows:

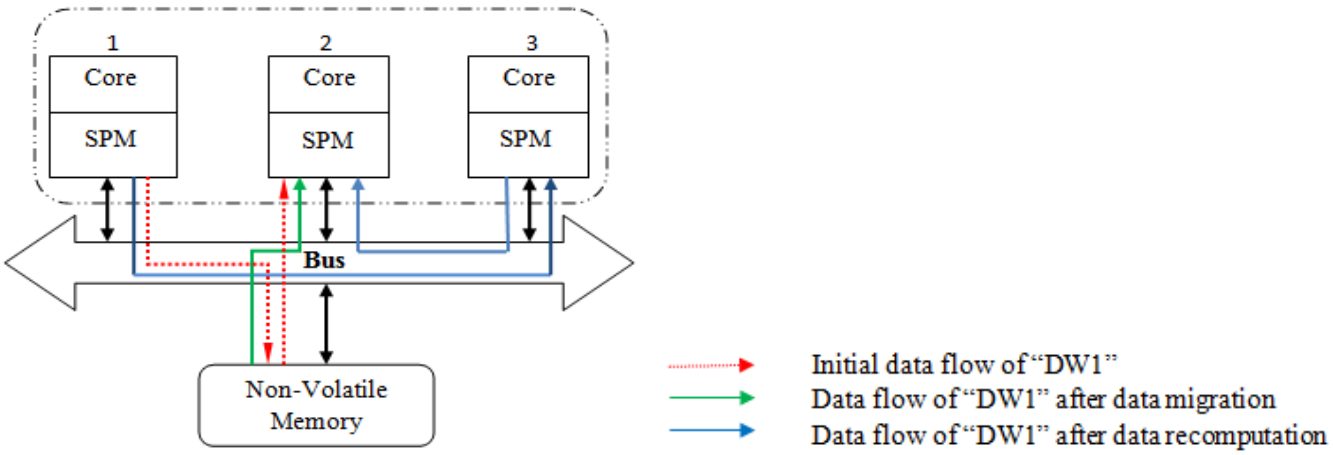

Fig 2. Example system with three cores

The smartly managed SPM will reduce the write activity on NVM. The authors purposed combine data migration and data recompilation to achieve the goal. In data migration, data is stored temporarily on the other core's SPM rather than written back to main memory. If data block migrated is a dirty block then authors called it as write-saving data migration and if the data block migrated is a clean block it called as read-saving data migration. In data recomputation, authors reduced the number of write activity by discarding the data which should have been written back to main memory and recomputing this data when it is needed. The authors purposed data migration problem as a shortest path problem. Through performance evaluation, the authors have also demonstrated that the data migration and recomputation can reduce the number of writes by 59.41% on an average. The finish time of programs is reduced by 31.81% on an average.

Wei-che Tseng and Qingfeng proposed scheduling technique to optimize cache utilization for NVM for the architecture with software controlled on-chip memory such as scratch-pad memory and main memory. But, by keeping limitation of NVM in mind, to reduce write activity on non-volatile memory such as Flash memory, authors proposed two scheduling techniques: Write-aware Scheduling and Recomputation [25]. In the proposed first technique authors used ILP scheduling. With the help of ILP formulation they derived objective function which provide optimal schedule of the task and SPM block replacement to reduce write activity on NVM's. But solving ILP formulation is NP-complete. So, Shuffle scheduling technique is suggested to solve large problem. In second technique, to avoid write operation on main memory the basic idea is discard the sum of computed data and when dependent task require that data it again read the necessary operand from main memory and recomputed the result. But, it is done only in case where recomputation cost or execution cost is lesser than the writing cost.

With the proposed techniques, authors speed up programs completion time and extend non-volatile memory's lifetime. These methods can reduce the number of write activities on non- volatile memory by 55.71% on average. Thus the lifetime of NVM is extended to 2.5 times as long as before on average. The completion time of programs can be reduced by 56.67% on systems with NOR flash memory and by 47.63%on systems with NAND flash memory on average.

In [24], Wei-Che Tseng and Edwin H.-M. Sha et al. have taken the reference of how the growing use of embedded systems in every field. But embedded system requires large amount of memory for storage, as it deals with large amount of data. Large memory consumes more power. For such type of systems authors have proposed that DRAM currently used as main memory be replaced by non-volatile memories (NVM) such as flash memory. But NVMs are limited by their endurance and long write latencies. To reduce the power consumption and increase the density of the main memory in this paper the authors proposed task scheduling techniques for the common embedded system architecture: Single core architecture with a hardware controlled cache and non-volatile main memory. To minimize the main memory access time, reduce the write activity and extend the lifetime of the NVM, optimally schedule tasks by ILP formulation and concatenation scheduling to solve instances that are too large to solve with ILP. They gave the objective function

$$\min t_r\Big[\sum_{s=1}^{nsets}\sum_{a=1}^{assoc}\sum_{p=1}^{m} C_{1,s,a,p} + \sum_{\ell=2}^{n}\sum_{p=1}^{m} Y_{\ell,p}\Big] + t_w\Big[\sum_{\ell=2}^{n}\sum_{p=1}^{m} Z_{\ell,p} + \sum_{p=1}^{m} D_{n,p}\Big]$$

Through performance evaluation, the authors have also demonstrated that the proposed concatenation task scheduling techniques can reduce the total memory access time by an average of 9.99% and increase the lifetime of the NVM by 26.66% than list scheduling. When compared with list scheduling, ILP can reduce the total memory access time by an average of 12.39% and increase the lifetime of NVM by 38.74%.

In [26], Ming-hwa Wang et al. introduced a flexible scheduling algorithm i.e. Hybrid Scheduling to improve the run-time performance of embedded systems. Through limited experiment authors achieved 3% -22.89% improvements in reduction of wait-time and 2% t- 9.23% improvement in the reduction of turnaround time. Hybrid scheduling has better performance than FCFS and RR as well as important for multi-tasking embedded system. Along with that, code overhead is low and the implementation is flexible. So, as a fixed scheduling algorithm, RR and FCFS can provide good performance in certain

condition, but not for all conditions. In contrasts, the hybrid scheduling monitors the input tasks and dynamically adjusts the scheduling algorithm to improve performance under changing condition.

Cache played an important role in every modern computing system to improve the performance by bridging the main memory and CPU speeds. However, cache increases execution time unpredictability due to its adaptive and dynamic nature. Due to the power hungry nature of cache, it consumes vast amount of energy.

In [27], B. Ward and J. Herman et al. 2013 presented several techniques for managing shared caches on multi-core systems within the mixed-critically scheduling framework. They implemented it on a quad-core ARM machine.

In [28], C. Belwal and A. Cheng et al. 2013 proposed utilization based necessary and sufficient scheduling condition for a Software Transactional Memory (STM) using lazy conflict detection.

In [21], *N. Guan, Wang Yiand Ge yu et al. 2009* proposed to use cache space isolation technique to avoid cache contention for hard real-time tasks running on multi-cores with shared caches. They proposed scheduling for real-time task with both timing and cache space constraints, which allow each task to use a fixed number of cache partitions and make sure that at any time at most one running task occupied the cache partition. They proposed schedulability test for non-preemptive fixed-priority scheduling for multicores with shared L2 cache, encoded as a linear programming problem.

In [29], S. Mittal noticed some issues related to NVM. He claimed that even though NVM offer high density, low leakage power and better scalability compared to SRAM but NVMs are not strictly superior to SRAM on all design parameters. Specially, PCM has very high write latency/energy and low write endurance [30]. He proposed a way-based SRAM-PCM Hybrid cache design as a solution to these issues. In hybrid cache 2 out of 8 ways are designed using SRAM and the remaining ways are designed using PCM. Hybrid cache aims to utilize the best features of both SRAM and NVM. Hybrid cache aims to leverage the fast access speed and high write endurance of SRAM and low-leakage power and high density of PCM. He also proposed cache replacement policy called dead fast block (DFB) to manage the hybrid cache. DFB evict dead SRAM ways much before they reach the bottom of the least recently used (LRU) stack.

Execution time predictability is a crucial factor for the success of real-time systems. Energy requirement is also crucial for embedded systems as they suffer from limited resources. For modern real-time embedded systems, the performance, power consumption, and predictability – all are important.

In [31], Xiaoxia Wu, Jian Li and Ram Rajamony et al. proposed two types of hybrid cache architecture, namely inter cache Level HCA (LHCA) and intra cache level or cache region based HCA (RHCA). In LHCA, the levels in a cache hierarchy can be made of disparate memory technologies. While in RHCA a single level of cache can be partitioned into multiple regions and each of a different memory technology.

Authors studied and provided comparison between different memory architecture:

TABLE I
Comparison of Various Memories

Features	SRAM	eDRAM	MRAM	PRAM
Density	Low	High	High	Very High
Speed	Very Fast	Fast	Fast read Slow write	Slow read Very slow write
Dyn. Power	Low	Medium	Low read; High Write	Medium read; High write
Leak Power	High	Medium	Low	Low
Non-volatile	No	No	Yes	Yes
Scalability	Yes	Yes	Yes	Yes

Authors have also evaluated low-overhead intra cache data movement power-aware policies and their hardware support to both improve cache performance and reduce power.

In [32], Abu Asaduzzaman et al. have developed a methodology to optimize cache for real-time embedded systems. This methodology is effective for both single-core and multi-core systems. These cache modeling and optimization technique for systems to improve performance / power ratio.

To improve execution time predictability for real-time systems a cache locking technique is used. To improve cache locking performance authors have presented Miss Table (MT) at cache level and use victim cache (VC) to increase cache hits. MT holds information about memory blocks, which may cause more misses if not locked. VC temporarily stores the victim blocks from level-1 cache to improve cache hits.

Authors claimed that existing solutions do not address the performance, power consumption, and predictability issues together and are not very effective for multi-core architecture. So, authors developed a cache optimization methodology to analyze and improve the predictability and performance/power ratio of real-time embedded systems at the same time. MT is used to improve cache replacement performance and VC is used to improve cache hits by supporting stream buffering. Authors experimentally proved that MT reduces 33% in mean delay per task and a reduction of 41% in total power consumption and VCs while locking 25% of level-2 cache size in a 4-core system. It is also observed that execution time predictability can be improved by avoiding more than 50% cache misses while locking one-fourth of the cache size.

In [33], B. Jacob et al. have proposed several software-oriented cache management schemes to allow real-time systems to make use of on chip SRAM caches. Author compared the operation and organization of caches as found in general-purpose processors, microcontrollers, and DSPs and also discussed designs for embedded real time systems.

In real-time embedded systems, energy consumption reduction is one of the best important techniques since most of these systems are battery-operated devices. Processor idle time (also known as slack time) provides a unique opportunity to reduce the overall energy consumption by putting the system into sleep mode using Dynamic Power Management (DPM) techniques [34][36].

In [34], W. Wang and P. Mishra et al. have proposed Scheduling-Aware Cache Reconfiguration (SACR), a methodology for using reconfigurable caches in real-time systems with preemptive tasks. Scheduling-Aware Cache Reconfiguration (SACR), provides an efficient and near optimal cache tuning strategy based on static program profiling for both statically and dynamically scheduled real-time systems. The goal is to optimize energy consumption with performance considerations via reconfigurable cache tuning while ensuring that the majority of task deadlines are met.

Authors claimed that this is the first approach integrating dynamic cache reconfigurations into real-time embedded systems. Authors also demonstrated that a 50% reduction on average in the overall energy consumption of the cache subsystem in soft real time embedded systems.

In [35], Phase-Change Memory (PCM) has used as a low-power alternative over DRAMs as main memory architecture that is especially helpful for energy-aware embedded real-time systems. PCM have three drawbacks: its high latency, high energy consumption when writing, and low endurance.

To oppose power consumption, PCM has emerged as alternative main memory architecture; however, PCM is slow, and thus architects use it together with a small DRAM cache. Authors proved that prioritizing requests at the bottleneck resource, namely PCM, using task information, increases the chance that tasks meet their deadlines. Authors also discussed how to carry out the communication between the OS and the hardware to make priority information feasible at the memory controller. Authors have suggested that reordering PCM requests based on criticality of the requests to reduce latency and the number of missed deadlines. Authors also claimed that this is the first technique which inserts real-time priorities in the memory controller.

In [38], J. Anderson and B. Ward considered the problem in the context of shared caches in safety-critical cyber physical embedded system. Authors considered the problem of adding proper shared cache management to MC2 (mixed-criticality on multicore). To address this problem they considered several cache management schemes that utilize page coloring in some way. Under page coloring, pages of physical memory are assigned "colors" in such a way that ensures that differently colored pages cannot cause cache conflicts.

Page coloring technique alone can often completely eliminate inter-task cache conflicts in real-time systems. But, the problem of optimally allocating colors is NP-hard in the strong sense [37]. Also, due to constraints on memory, which can arise for a variety of reasons (e.g., the need to support many processing modes, etc.), conflict-free color assignments may be unobtainable. So,authors proposed colouring technique to enable coloring to be more flexibly utilized. Specifically, they assumed colors as shared resources to which accesses must be arbitrated, either by a real-time locking protocol or a scheduling algorithm. They considered cache management in a variant of MC2 in which both higher-criticality (HC) hard realtime (HRT) tasks and lower-criticality (LC) soft real-time (SRT) tasks must be supported.

Authors considered cache locking and cache scheduling as two basic cache-management approaches. Under cache locking, portions of the cache are viewed as nonpreemptive resources that are accessible via a locking protocol. Under cache scheduling, portions of the cache are viewed as preemptive resources that are "scheduled." This approach results in a scheduling problem that is quite difficult generally; however, authors showed that known uniprocessor schedulability analysis can be used to provide a sufficient schedulability condition that performs well in many cases. We note that in some special cases, these cache management strategies are obviated, in which case these approaches reduce to cache partitioning. Authors assumed periodic task systems scheduled under the MC2 mixed-criticality framework.

IV. CONCLUSION

This paper gives an overview of the various optimization scheduling techniques which is suitable for different architecture. Even though ILP scheduling is NP-complete problem, ILP Formulation can used to optimally solve the problem. More relevant results can be obtained by applying an ILP formulation.

References

1. M. Wu and W. Zwaenepoel, "envy: a non-volatile, main memory storage system," in ASPLOS-VI: Proceedings of the sixth international conference on Architectural support for programming languages and operating systems, San Jose, California, United States, 1994, pp. 86–97.
2. K. Lee and A. Orailoglu, "Application specific non-volatile primary memory for embedded systems," in CODES/ISSS '08: Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis, Atlanta, GA, USA, 2008, pp. 31–36.
3. "Onenand features and performance," 2007, [Online]. Available: http://www.samsung.com/global/business/semiconductor/products/fusionmemory/Products OneNA ND.html [Accessed: Sept. 10, 2009].
4. P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," SIGARCH Comput. Archit. News, vol. 37, no. 3, pp. 14–23, 2009.
5. B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," SIGARCH Comput. Archit. News, vol. 37, no. 3, pp. 2–13, Jun. 2009.
6. R. Desikan, C. R. Lefurgy, S. W. Keckler, and D. Burger, "Onchip mram as a high-bandwidth low-latency replacement for dram physical memories," Department of Computer Science, The University of Texas at Austin, Tech. Rep. TR-02-47, 2002.
7. J. Hu, C. Xue, W. Tseng, Q. Zhuge, and E. Sha, "Minimizing write activities to non-volatile memory via scheduling and recomputation," in Application Specific Processors (SASP), 2010 IEEE 8th Symposium on, 2010, pp. 101–106.
8. Y. He, C. J. Xue, C. Xu, and E. Sha, "Co-optimization of memory access and task scheduling on MPSoC architectures with multi-level memory," in Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific, 2010, pp. 95–100.
9. C. Lee, M. Potkonjak, and W. Mangione-Smith, "Mediabench: A tool for evaluating and synthesizing multimedia and communications systems," in micro. Published by the IEEE Computer Society, 1997, p. 330.
10. P. Wu, Y. Chang, and T. Kuo, "A file-system-aware FTL design for flash-memory storage systems," in Proceedings of the Conference on Design, Automation and Test in Europe, Nice, France, 2009, pp. 393–398.

11. Y. Chu, J. Hsieh, Y. Chang, and T. Kuo, "A set-based mapping strategy for flash-memory reliability enhancement," in Proceedings of the Conference on Design, Automation and Test in Europe, Nice, France, 2009, pp. 405–410.
12. Y. Chang and T. Kuo, "A commitment-based management strategy for the performance and reliability enhancement of flash-memory storage systems," in Proceedings of the 46th Annual Design Automation Conference, 2009, pp. 858–863.
13. L. Jiang, Y. Du, Y. Zhang, B. R. Childers, and J. Yang, "Lls: Cooperative integration of wear-leveling and salvaging for pcm main memory," Dependable Systems and Networks, International Conference on, vol. 0, pp. 221–232, 2011.
14. M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in ACM SIGARCH Computer Architecture News, vol. 37, 2009, pp. 24–33
15. L. Shi, C. J. Xue, J. Hu, W. Tseng, X. Zhou, and E. H. M. Sha, "Write activity reduction on flash main memory via smart victim cache," in Proceedings of the 20th symposium on Great lakes symposium on VLSI, Providence, Rhode Island, USA, 2010, pp. 91–94.
16. J. Hu, C. J. Xue, W. Tseng, Y. He, M. Qiu, and E. H. M. Sha, "Reducing write activities on non-volatile memories in embedded CMPs via data migration and recomputation," in Proceedings of the 47th Design Automation Conference, 2010, pp. 350–355.
17. G. Dhiman, R. Ayoub, and T. Rosing, "Pdram: a hybrid pram and dram main memory system," in Proceedings of the 46th Annual Design Automation Conference, 2009, pp. 664–469.
18. X. Wu, J. Li, L. Zhang, E. Speight, and Y. Xie, "Power and performance of read-write aware hybrid caches with nonvolatile memories," in Proceedings of the Conference on Design, Automation and Test in Europe, 2009, pp. 737–742.
19. X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Hybrid cache architecture with disparate memory technologies," SIGARCH Comput. Archit. News, vol. 37, no. 3, pp. 34–45, Jun. 2009. [Online]. Available: http://doi.acm.org/10.1145/1555815.1555761
20. P. Mangalagiri, K. Sarpatwari, A. Yanamandra, V. Narayanan, Y. Xie, M. J. Irwin, and O. A. Karim, "A low-power phase change memory based hybrid cache architecture," in Proceedings of the 18th ACM Great Lakes symposium on VLSI, 2008, pp. 395–398.
21. N. Guan, M. Stigge, W. Yi, and G. Yu, "Cache-aware scheduling and analysis for multicores," in Proceedings of the seventh ACM international conference on Embedded software, 2009, pp. 245–254.
22. M. Bhadauria and S. A. McKee, "An approach to resourceaware co-scheduling for CMPs," in Proceedings of the 24th ACM International Conference on Supercomputing, 2010, pp. 189–199.
23. W. Tseng, C. Xue, Q. Zhuge, J. Hu, and E. Sha, "Optimal scheduling to minimize non-volatile memory access time with hardware cache," in VLSI System on Chip Conference (VLSISoC), 2010 18th IEEE/IFIP, 2010, pp. 131–136.
24. Jingtong Hu, Chun Jason Xue, Qingfeng Zhuge, Wei-CheTseng, Edwin, H.-M. Sha, "Scheduling to Optimize Cache Utilization for Non-Volatile Main-Memories", IEEE Transaction(COMPUTERS), PP- 99 ,1 , 2013
25. V. Zivojnovic, J. Martinez, C. Schlager, and H. Meyr, "Dspstone: A dsp-oriented benchmarking methodology," in ICSPATICSPAT' 94: In Proceedings of the International Conference on Signal Processing Applications and Technology, Dallas, Texas, USA, 1994.
26. C. Hwang, "Effective scheduling in mobile embedded system" [Online] Available.(Accessed: 2013)
27. B. Ward, J. Herman, C. Kenna, and J. Anderson,. "Making Shared Caches More Predictable on Multicore Platforms", Proceedings of the 25th Euromicro Conference on Real-Time Systems, pp. 157-167, July 2013. Winner, outstanding paper award.
28. H. Cho, Binoy Ravindran and E. Douglas Jensen, "Synchronization for an optimal real-time scheduling algorithm on multiprocessors" [Online] Available.(Accessed: Aug 6,2013)
29. S. Mittal, "A Technique for Efficiently Managing SRAM-NVM Hybrid Cache", [v1] Fri, 1 Nov 2013 12:52:59
30. S. Mittal, "Energy Saving Techniques for Phase Change Memory (PCM)," Iowa State University, Tech. Rep., 2013.
31. Xiaoxia Wu, Jian Li and Ram Rajamony, "Hybrid Cache Architecture with Disparate Memory Technologies" [Online] Available (Accessed: Dec 12,2013)
32. Abu Asaduzzaman, "Cache Optimization For Real-Time Embedded Systems",Dec 2009.
33. B. Jacob, "Cache Design for Embedded Real-Time Systems" [Online] Available.(Accessed: Dec 17,2013)
34. W. Wang and P. Mishra, "SACR: Scheduling-Aware Cache Reconfiguration for Real-Time Embedded Systems" [Online] Available.(Accessed: Dec 13, 2013)
35. Miao Zhou,Santiago Bock, Alexander P.Ferreira, Bruce Childers, Rami Melhem,Daniel Moss'e, "Real-Time Scheduling for Phase Change Main Memory Systems" [Online] Available.(Accessed: Aug 05,2013)
36. L. Benini, G. De Micheli, "A survey of design techniques for system level dynamic power management", TVLSI, 8(3):299-316, June 2000.
37. B. Bui, M. Caccamo, L. Sha, and J. Martinez. "Impact of cache partitioning on multi-tasking real time embedded systems", In RTCSA '08.
38. J. Anderson and B. Ward, "Making Shared Caches More Predictable on Multicore Platforms", [Online] Available.(Accessed: Dec 1,2013)

