NVMain Extension for Multi-Level Cache Systems

Asif Ali Khan, Fazal Hameed and Jeronimo Castrillon
Chair For Compiler Construction
Technische Universität Dresden, Germany
{ asif_ali.khan, fazal.hameed, jeronimo.castrillon }@tu-dresden.de

ABSTRACT

In this paper, we present an extension of the NVMain memory simulator. The objective is to facilitate computer architects to model complex memory designs for future computing systems in an accurate simulation framework. The simulator supports commodity memory models for DRAM as well as emerging non-volatile memories technologies such STT-RAM, ReRAM, PCRAM and hybrid models. The current publicly available version of NVMain, NVMain 2.0, offers support for main memory (using DRAM and NVM technologies) and a die-stacked DRAM cache. We extend the cache model of the simulator by introducing an SRAM cache model and its supporting modules. With this addition, designers can model hybrid multi-level cache hierarchies by using the die-stacked DRAM cache and SRAM caches. We provide a reference implementation of an optimized cache organization scheme for die-stacked DRAM cache along with a tag-cache unit that, together, reduces cache miss latency. To enable integration of the new features in the existing memory hierarchy, we make necessary changes to the memory controller. We provide functional verification of the new modules and put forward our approach for timing and power verification. We run random mixes of the SPEC2006 benchmarks and observe $\pm 10\%$ difference in simulation results.

CCS Concepts

•**Memory System Design** → Memory System Hirarichy; Memory Technologies; •**Memory Simulators** → Simulator Accuracy; Design options; Simulator Speed; Extendibility; Resilience;

Keywords

Memory Simulator; SRAM Cache; Cache Organization; Row Buffer; Tag-cache

RAPIDO, January 22–24, 2018, Manchester, United Kingdom

978-1-4503-6417-1/18/01…$15.00

DOI: https://doi.org/10.1145/3180665.3180672

1. INTRODUCTION

CPU and memory system are the two most important components in any computing system. From the performance perspective, both components are strictly interlinked. CPU frequency has rapidly increased in the past decade while frequency of the memory system has not scaled up at the same pace. The memory system has always been on the down side due its high access time and relatively low operating frequency, becoming a major bottleneck in modern days computing systems. The rise of multi-core systems has further worsened this problem and the available per core capacity and per core bandwidth has diminished further. Commodity memory technologies such as DRAM are unable to fill this ever-increasing processor memory gap.

DRAM is highly criticized for being expensive in terms of power as well. The major factor that dominates DRAM power dissipation is its periodic refreshes, background and leakage power. The newly emerged non-volatile memory (NVM) technologies such as spin-torque-transfer random-access memory (STT-RAM), phase-change random-access memory (PCRAM), and resistive random-access memory (ReRAM) are believed to alleviate these limitations. They have been advocated as potential DRAM replacements at various levels (main memory, die-stacked cache). While these technologies could supplement or supersede conventional memory technologies at various levels in the memory hierarchy, they have their own limitations. Simply replacement of DRAM with NVM technologies is not a viable option because the latter suffers from high write energy and write endurance issues.

This opens up new research directions and calls for exploration of appropriate memory technologies at each memory level. An ideal memory system would use the best of many memory technologies and fulfill the diverse demands of modern days applications. To design such system, it is vital for computer architects to use simulation tools and study the suitability of each technology. NVMain [24] is one such simulator that provides support for both DRAM and NVM technologies. It models energy and cycle accurate operations of main memory system. In its extended version NVMain 2.0 [25], die-stacked DRAM cache was introduced with the Alloy [26] model.

In this paper, we present extensions for the publicly available version of NVMain (NVMain 2.0) simulator. We provide a reference implementation of a new SRAM cache which could be used at various cache levels. For die-stacked cache, we provide implementation of various latency optimizations. To demonstrate this extension, we model a recently proposed

high associativity cache organization called LAMOST [9]. A supporting module called *tag-cache* (an SRAM based small memory unit that stores the tag information of recently accessed sets as explained in [14, 9, 20]) is employed on top of LAMOST to mitigate high DRAM cache tag lookup latency. We modify the memory controller and other NVMain modules to make provision for the new extensions.

While some of these architectural features such as tag-cache support specific cache organizations [17, 9], others such as SRAM cache model can be used to model any level in the memory hierarchy. All the new modules are made configurable and can be enabled or disabled easily. Configuration parameters of each individual module are passed in a config file and can be varied as per design requirements. We believe these new features make NVMain more powerful and increase its application scope. Considering the design objectives, designers have more freedom to (a) model customized memory systems (b) choose memory technology for each level from a wider list of available options (DRAM, NVM, SRAM) (c) select the number of cache/memory levels (d) choose appropriate cache organization (Alloy, LAMOST, LAMOST with tag-cache) for die-stacked memory.

2. RELATED WORK

Simulation has become a powerful tool in computer architecture community that empowers designers to model their desired systems and predict the design objectives beforehand. Architectural simulation is mainly used for design space exploration and performance evaluation considering all design goals and performance parameters. As such, every leading processor manufacturing industry has developed its own simulation tool(s) to model new processor models and assess their feasibility. For instance, Mambo [4] by IBM is designed to model their past and future power designs. Its architectural support ranges from cell to embedded system to supercomputer (BlueGene, POWER7). Similarly, SimNow [2] by AMD and HAsim [23] by Intel are used to evaluate their future systems respectively. Some high speed architectural simulators have been developed to reduce the simulation time. Sniper [5], Graphite [21], SlackSimslacksim, P-Mambo [30] and COTSon [1] are to name a few. Open source system simulators such as Gem5 [3] exist that not only target micro-architecture of the processor but encompasses the whole system architecture. In a recent work [19], Gem5 has been coupled with SystemC that opens up a whole new set of options for system level design space exploration.

Unfortunately, some of these micro-architecture and system-level simulators do not model the memory system in detail. They consider a simplistic memory model without considering the complex organization of modern memory systems. More often, a fixed latency is associated with a memory access. In more advanced CPU simulators, bank conflicts are considered and every time it occurs, a fixed wait latency is added to the overall access time. This model significantly underestimates the actual characteristic of the memory system. This necessitates design of specialized memory simulators that consider all possible design features, model actual bus latencies, and report correct timing and energy parameters.

Of late, many such memory simulators have been published. DRAMSim [29], the first publicly available memory simulator, was designed to offer support for multiple types of

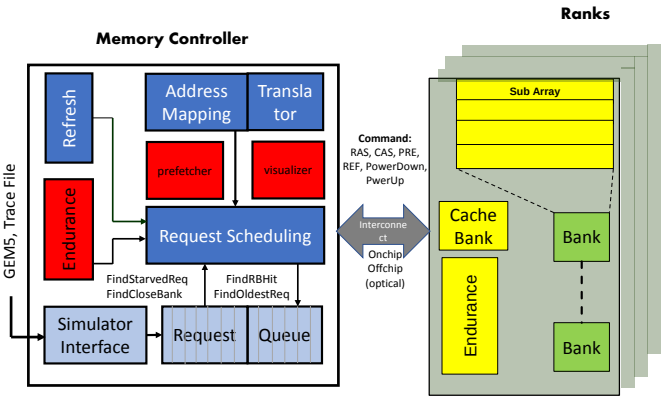

Figure 1: Overview of the NVMain Architecture [24]

memory technologies (SDRAM, DDR) and allow exploration of memory system concepts. Presently, its extended version DRAMSim2 [27] is openly available and is broadly used. It is a C++ based cycle-accurate model that provides most of the memory system design features. Design goals of the simulator were to keep it small and portable. However, this simplicity in controller architecture has barred it to be comparable with more recent and performance optimized controllers. Other memory simulators such as Ramulator [16], DRAMSys [15], NVMain and integrated simulators such as [28] have been proposed with different design goals. For instance, Ramulator offers support for an extended list of DRAM standards. DRAMSys provides a holistic framework that takes into consideration new DRAM based memory solutions such as JDEC DDR4, WIDE I/O, and HMC. It captures new aspects such as temperature and retention failures. In contrast to all these simulators, NVMain focuses on emerging NVM technologies while keeping the DRAM support intact. A brief comparison of these memory simulators with reference to design options is presented in Table 1.

3. SIMULATOR ARCHITECTURE

NVMain is a flexible memory simulator that supports DRAM and NVM technologies. For DRAM, it follows the same approach as other DRAM specific simulators in order to capture important features. For NVM devices, beside modeling timing and power parameters, it models NVM specific features as well; such as endurance, high write energy, and multi-level cells (MLC) capability. Major modules of the NVMain simulator include timing, power and endurance models. NVMain 2.0, the currently available version, extends support for sub-array level parallelism and MLC operations. *Object hooks* are introduced to strengthen the simulator hierarchy and allow requests inspection at a particular level.

3.1 NVMain Overview

NVMain is an object based model where every module is created as a separate object that can easily be integrated to or detached from the simulator. An overview of the base architecture of NVMain is shown in Figure 1.

Every object in the NVMain simulator such as the memory controller, the interconnect, the rank or the bank is responsible to model and capture its timing parameters. The

Table 1: Comparison of the Memory Simulators

Simulator	DRAM	die-stacked cache	NVM	Alloy [26]	LAMOST [9]	tag-cache	SRAM
DRAMSim2	✓	✗	✗	✗	✗	✗	✓
NVMain2.0	✓	✓	✓	✓	✗	✗	✗
Ramulator	✓	✗	✗	✗	✗	✗	✓
DRAMSys	✓	✗	✗	✗	✗	✗	✓
NVMainExt (proposed)	✓	✓	✓	✓	✓	✓	✓

timing parameters for DRAM devices are taken from their manufacturer data-sheets while for SRAM and NVM technologies, these parameters are obtained from CACTI [22] and NVSIM [8] tools respectively. The corresponding memory parameters such as t_{RCD}, t_{RP}, t_{BURST} are provided in a memory configuration file to the simulator. The simulator takes another configuration file that describes the overall memory system hierarchy and general configuration parameters such for number of ranks, banks, rows, columns, address mapping scheme, decoders, row buffer policies, queue models and queue sizes.

NVMain offers both single bank and inter-bank timing models. The single bank timing model tracks most commonly found parameters such as t_{RCD}, t_{RP} and t_{CAS} etc. The inter-bank timing model restricts the power consumption and current drawn by a single bank or different banks within a fix time period by introducing parameters like t_{FAW} (four activation window) and t_{RRD} (row to row activation delay). All timing parameters are considered before issuing a memory command to a particular module. Error message is generated in case a module violates the timings constraints.

In its power model, NVMain computes per device energy consumption where each device consists of multiple banks. The total energy consumption is calculated as the sum result of energy consumptions of all devices. For DRAM devices, the typical IDDx parameters are used to measure power consumption of read and write operations. For NVM technologies, to calculate the actual power consumption, NVMain takes per bit write energy from NVSIM. The endurance model keeps track of the total number of bits written and their values. Energy of the unchanged bits is subtracted from the total write energy consumption.

While the simulator provides a solid base for exploring the optimal memory system, the design features it offers are limited. The missing features include: a wide array of memory technologies, configurable number of memory levels, flexible cache controllers optimized for various performance parameters, tag-cache, predictors and multiple row buffer mapping schemes. The object-oriented structure of NVMain encourages development of new extensions and allow their easy integration.

3.2 NVMain Extensions

We provide reference implementations of new architectural features for NVMain that could be used to propose future memory systems. The new extensions strengthen the relatively simple cache model of the current NVMain simulator and opens up new design directions. Figure 2 highlights these features and demonstrates how they fit in the overall memory system design.

As highlighted in Figure 2, the die-stacked last level cache (LLC) can be modeled as DRAM or NVM technologies. For cache organization, designers have the choice to select appropriate scheme depending on the application requirements. Alloy cache [26] gives the best performance for applications having reduced miss-rate. Conversely, LAMOST [9] and LH [17] cache organizations perfom better than Alloy cache for applications having higher miss rates.

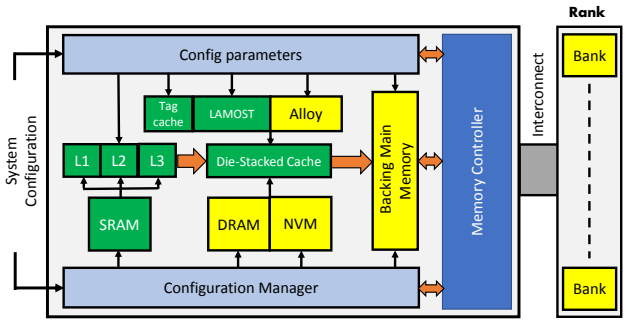

Figure 2: Extended NVMain Architecture (The green color highlights the extensions)

The proposed SRAM module can be used to model any cache level in the memory hierarchy. Typically, CPU simulators model only lower cache levels (level 1, 2 and 3). Existing NVMain simulator can be used to model die-stacked DRAM cache, typically used as LLC. Our extended version provides support to model multi-level cache hierarchy where lower cache levels can be realized with SRAM technology while higher cache level can be implemented using DRAM or NVM technology.

3.2.1 SRAM Cache Model

SRAM is a fast memory technology. It has small access time compared to DRAM and NVM technologies but is more expensive. It is used at lower cache levels (close to the processor) to bridge the processor and (slow)memory speed gap. NVMain simulator in its present form does not support SRAM. As a result, it can not model the lower cache levels. We introduce a reference model of SRAM cache which can be used as Level 1(L1), Level 2(L2) or Level 3(L3) cache.

To model the actual complex memory system, designers should define all cache levels in the memory hierarchy. Unfortunately, existing memory simulators offer only higher cache level(s). With this SRAM model, NVMain is capable to model the entire memory system by its own. Every parameter of the SRAM model is configurable and can be changed as per design goals. We adopt existing cache bank model of the simulator with varied parameters to implement SRAM specific functionalities. For every new request, the address translator of the SRAM module retranslates the physical address of the request and sets the SRAM related memory partitions. The SRAM cache is checked for

a hit/miss and based on the type of request, it is serviced accordingly (detailed discussion on the flow of commands to serve a read or write request is beyond the scope of this paper). A sub-request (e.g. cache line eviction, cache fill), if any, generated by this module is solely owned by it and has to be deleted after its completion. Further, this module forwards request to the next level in the memory hierarchy based on the system configuration. The next level, if exists, is added as a child to the SRAM module and can be accessed via an *object hook*. Appropriate requests and responses are generated between parent and child modules where parent corresponds to the current level and child corresponds to the next level in the memory hierarchy.

3.2.2 LAMOST Cache Organization

Presently, NVMain implements the Alloy cache organization for its die-stacked cache and it is by-default selected. Designers have no choice to change the cache organization. Alloy cache is a direct-mapped model that suffers from high miss rate and high off-chip memory latency. We implement a new cache organization that overcomes the limitations of the Alloy cache. For demonstration, we model LAMOST with a supporting tag-cache that reduces the cache hit/miss latency. In addition, we develop new row and set mapping schemes and make necessary changes to the memory controller. More details of the mapping schemes can be found in [10].

3.2.3 Tag Cache

Tag cache, fundamentally, is not a level in the memory system. Rather, it is a supporting unit for the die-stacked cache. Tag-cache stores the tag information of the most recently accessed cache sets. Future accesses to the same set(s) in die-stacked cache result in reduced access latency. Cache organizations such as ATCache [14] and LAMOST [9], when used with tag-cache, positively benefit from it. Detailed architectural benefits of tag-cache and various cache organizations can be found in [26, 14, 17, 10, 20]. For correctness, the simulator makes sure the right system configuration and generates an error otherwise.

4. EVALUATION

NVMain can be used in both trace based simulation as well full system simulation mode. For full system simulation, we connect NVMain to gem5 by applying the publicly available patch and run SPEC benchmarks on it. Gem5 uses the memory system modeled by NVMain instead of using its default memory model. An abstract overview of the full system configuration is presented in Figure 3. We use Simpoint [12] in order to reduce the benchmarks execution time. The idea is, to run a sub-set of instructions (for each benchmark) that imitates the behavior of the whole benchmark. In trace based simulation, we generate traces from Gem5 and provide trace file(s) as input to the NVMain simulator.

For functional verification of the new extensions, we use random mixes of the SPEC2006 [13] benchmarks. For the same system configuration, we run SPEC benchmarks in sim-zesto simulator [18] and NVMain. For ten experimental runs, we observe performance parameter (miss-rate) of L1, L2 and L3 caches and report the average measures in Table 2.

Table 2: Functional Verification of the New Extensions

Cache Level	sim-zesto normalized miss-rate	NVMain normalized miss-rate	Difference(%)
L1 (32 KB)	1.00	0.96	+4
L2 (256 KB)	1.00	1.09	−9
L3 (8 MB)	1.00	0.95	+5

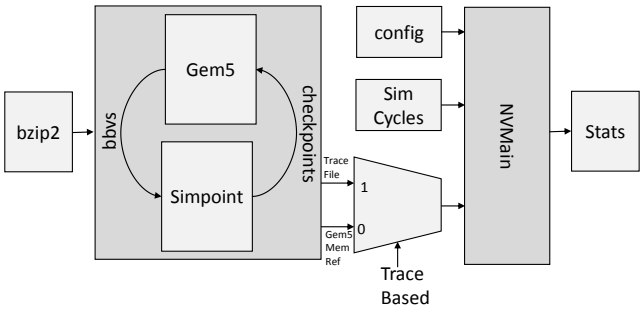

Figure 3: Full system simulation overview

Simulation results of the NVMain are in ±10% of the sim-zesto results. The memory model in sim-zesto is simplistic and can not accurately model LLC. Therefore, for the cache organization scheme LAMOST, we compare our results with the actual results presented in [9] and observe that they are in the accuracy range of ±5%.

For speed comparison, we run both sim-zesto and NVMain for 3B (3×10^9) instructions and observe that NVMain is around 10 times faster than sim-zesto. The timing and energy models of NVMain have already been verified. However, in future, we plan to verify these models for new extensions as well using Verilog model (for timing) and the publicly available DRAMPower2 [7] for (energy).

5. CONCLUSIONS

We have presented an extended version of the NVMain simulator. Considering the fundamental design goals — flexibility, simple user interface and scalability — of the simulator, we have provided reference implementations of SRAM cache, an optimized cache organization for die-stacked cache and a tag-cache model. The newly added design features widen the list of design options and enable customized design modelling. We have outlined the existing simulator architecture in brief and the new extensions in detail. We have run random mixes of the SPEC benchmarks in NVMain and observed that simulation results of the new extensions are in conformance with state-of-the-art. We will use this simulator to investigate memory systems with emerging memory technologies like [11] in the context of the Orchestration project [6].

Acknoledgments

This work was partially funded by the German Research Council (DFG) through the Cluster of Excellence 'Center for Advancing Electronics Dresden' (cfaed).

6. REFERENCES

[1] E. Argollo, A. Falcón, P. Faraboschi, M. Monchiero, and D. Ortega. Cotson: Infrastructure for full system simulation. *SIGOPS Oper. Syst. Rev.*, 43(1):52–61, Jan. 2009.

[2] R. Bedicheck. Simnow: Fast platform simulation purely in software. In *Hot Chips 16*, 2004.

[3] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.

[4] P. Bohrer, J. Peterson, M. Elnozahy, R. Rajamony, A. Gheith, R. Rockhold, C. Lefurgy, H. Shafi, T. Nakra, R. Simpson, E. Speight, K. Sudeep, E. Van Hensbergen, and L. Zhang. Mambo: A full system simulator for the powerpc architecture. *SIGMETRICS Perform. Eval. Rev.*, 31(4):8–12, Mar. 2004.

[5] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 52:1–52:12, New York, NY, USA, 2011. ACM.

[6] J. Castrillon, M. Lieber, S. Klüppelholz, M. Völp, N. Asmussen, U. Assmann, F. Baader, C. Baier, G. Fettweis, J. Fröhlich, A. Goens, S. Haas, D. Habich, H. Härtig, M. Hasler, I. Huismann, T. Karnagel, S. Karol, A. Kumar, W. Lehner, L. Leuschner, S. Ling, S. Märcker, C. Menard, J. Mey, W. Nagel, B. Nöthen, R. Peñaloza, M. Raitza, J. Stiller, A. Ungethüm, A. Voigt, and S. Wunderlich. A hardware/software stack for heterogeneous systems. *IEEE Transactions on Multi-Scale Computing Systems*, Nov. 2017.

[7] K. Chandrasekar, B. Akesson, and K. Goossens. Improved power modeling of ddr sdrams. In *Proceedings of the 2011 14th Euromicro Conference on Digital System Design*, DSD '11, pages 99–108, Washington, DC, USA, 2011. IEEE Computer Society.

[8] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):994–1007, July 2012.

[9] F. Hameed, L. Bauer, and J. Henkel. Simultaneously optimizing dram cache hit latency and miss rate via novel set mapping policies. In *2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pages 1–10, Sept 2013.

[10] F. Hameed, L. Bauer, and J. Henkel. Architecting on-chip dram cache for simultaneous miss rate and latency reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(4):651–664, April 2016.

[11] F. Hameed, C. Menard, and J. Castrillon. Efficient stt-ram last-level-cache architecture to replace dram cache. In *Proceedings of the International Symposium on Memory Systems (MemSys'17)*, MEMSYS '17, pages 141–151, New York, NY, USA, Oct. 2017. ACM.

[12] G. HAMERLY. Simpoint 3.0 : Faster and more flexible program analysis. *Workshop on Modeling, Benchmarking and Simulation, 2005*, 2005.

[13] J. L. Henning. Spec cpu2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17, Sept. 2006.

[14] C.-C. Huang and V. Nagarajan. Atcache: Reducing dram cache latency via a small sram tag cache. In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, PACT '14, pages 51–60, New York, NY, USA, 2014. ACM.

[15] M. Jung, C. Weis, and N. Wehn. Dramsys: A flexible dram subsystem design space exploration framework. *IPSJ Transactions on System LSI Design Methodology*, 8:63–74, 2015.

[16] Y. Kim, W. Yang, and O. Mutlu. Ramulator: A fast and extensible dram simulator. *IEEE Comput. Archit. Lett.*, 15(1):45–49, Jan. 2016.

[17] G. Loh and M. D. Hill. Supporting very large dram caches with compound-access scheduling and missmap. *IEEE Micro*, 32(3):70–78, May 2012.

[18] G. H. Loh, S. Subramaniam, and Y. Xie. Zesto: A cycle-level simulator for highly detailed microarchitecture exploration. In *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 53–64, April 2009.

[19] C. Menard, J. Castrillón, M. Jung, and N. Wehn. System simulation with gem5 and systemc the keystone for full interoperability. 2017.

[20] J. Meza, J. Chang, H. Yoon, O. Mutlu, and P. Ranganathan. Enabling efficient and scalable hybrid memories using fine-granularity dram cache management. *IEEE Computer Architecture Letters*, 11(2):61–64, July 2012.

[21] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal. Graphite: A distributed parallel simulator for multicores. In *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pages 1–12, Jan 2010.

[22] N. Muralimanohart and N. Balasubramonian, R.and Jouppi. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 3–14, December 2007.

[23] M. Pellauer, M. Adler, M. Kinsy, A. Parashar, and J. Emer. Hasim: Fpga-based high-detail multicore simulation using time-division multiplexing. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pages 406–417, Feb 2011.

[24] M. Poremba and Y. Xie. Nvmain: An architectural-level main memory simulator for emerging non-volatile memories. In *2012 IEEE Computer Society Annual Symposium on VLSI*, pages 392–397, Aug 2012.

[25] M. Poremba, T. Zhang, and Y. Xie. Nvmain 2.0: A user-friendly memory simulator to model (non-)volatile memory systems. *IEEE Computer Architecture Letters*, 14(2):140–143, July 2015.

[26] M. K. Qureshi and G. H. Loh. Fundamental latency trade-off in architecting dram caches: Outperforming impractical sram-tags with a simple and practical design. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-45, pages 235–246, Washington, DC, USA, 2012. IEEE Computer Society.

[27] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. Dramsim2: A cycle accurate memory system simulator. *IEEE Comput. Archit. Lett.*, 10(1):16–19, Jan. 2011.

[28] J. Stevens, P. Tschirhart, C. Mu-Tien, I. Bhati, P. Enns, J. Greensky, Z. Chisti, L. Shih-Lien, and B. Jacob. An integrated simulation infrastructure for the entire memory hierarchy: Cache , dram, nonvolati le memory, and disk. *Intel Technology Journal*, 17(1):184 – 200, 2013.

[29] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob. Dramsim: A memory system simulator. *SIGARCH Comput. Archit. News*, 33(4):100–107, Nov. 2005.

[30] K. Wang, Y. Zhang, H. Wang, and X. Shen. Parallelization of ibm mambo system simulator in functional modes. *SIGOPS Oper. Syst. Rev.*, 42(1):71–76, Jan. 2008.