Main memory organization trade-offs with DRAM and STT-MRAM options based on gem5-NVMain simulation frameworks

Manu Komalan*, Oh Hyung Rock*, Matthias Hartmann*‡, Sushil Sakhare*, Christian Tenllado†, José Ignacio Gómez†, Gouri Sankar Kar*, Arnaud Furnemont*, Francky Catthoor*†, Sophiane Senni§, David Novo§, Abdoulaye Gamatie§ and Lionel Torres§

*IMEC, Leuven 3001, Belgium
Email: (Manu.Perumkunnil, Matthias.Hartmann, Francky.Catthoor)@imec.be
†Universidad Complutense de Madrid, Madrid 28040, Spain
‡KU Leuven 3001, Belgium
§LIRMM, University of Montpellier, CNRS, Montpellier, France
Email: (Sophiane.Senni, David.Novo, Abdoulaye.Gamatie, Lionel.Torres)@lirmm.fr

***Abstract*—Current main memory organizations in embedded and mobile application systems are DRAM dominated. The ever-increasing gap between today's processor and memory speeds makes the DRAM subsystem design a major aspect of computer system design. However, the limitations to DRAM scaling and other challenges like refresh provide undesired trade-offs between performance, energy and area to be made by architecture designers. Several emerging NVM options are being explored to at least partly remedy this but today it is very hard to assess the viability of these proposals because the simulations are not fully based on realistic assumptions on the NVM memory technologies and on the system architecture level. In this paper, we propose to use realistic, calibrated STT-MRAM models and a well calibrated cross-layer simulation and exploration framework, named SEAT, to better consider technologies aspects and architecture constraints. We will focus on general purpose/mobile SoC multi-core architectures. We will highlight results for a number of relevant benchmarks, representatives of numerous applications based on actual system architecture. The most energy efficient STT-MRAM based main memory proposal provides an average energy consumption reduction of 27% at the cost of 2x the area and the least energy efficient STT-MRAM based main memory proposal provides an average energy consumption reduction of 8% at the around the same area or lesser when compared to DRAM.**

***Index Terms*—NVM, Main Memory, Selector, Bit cell, NVMain**

I. Introduction

Minimizing the energy consumption of main memory under performance constraints has become one of the most important factors involved in delivering high-performance and low-power computing platforms. A study of server systems shows that main memory consumes 40% of server power by means of both the direct energy consumption and indirect energy contribution required to mitigate the power density/thermal effects of these systems [1]. This percentage of energy contribution is even higher when the system is in standby mode, a common case for mobile systems, typically to save energy. With the drive to bridge the memory wall, this problem has compounded. In DRAM, aggressive scaling of the cell capacitance threatens the sensing margin and obtaining enough data retention characteristics.

This has led to several proposals to utilize emerging non-volatile memory (NVMs) such as resistive RAM (RRAM), phase change memory (PCM) and spin-transfer torque magnetic RAM (STT-MRAM) at the main memory level [2] [3]. These NVMs provide advantages of dramatically reduced static energy consumption by eliminating the need for refresh while also boasting potential advantages in technology scalability and density. Unfortunately, these NVMs usually have a higher dynamic write energy than DRAM that can limit their energy advantage. STT-MRAM is presently one of the more promising and mature NVMs. It has high density (10-20 F^2 depending on bit-cell design), low power consumption, is fast relative to other NVMs and suffers minimal degradation over time (lifetime $\sim10^{16}$ cycles [4]. STT-MRAM can also be embedded into a SoC without altering or adversely impacting the baseline logic both in design and process integration.

In this paper, we first look at the state of the art with respect to alternative main memory solutions (Section 2). We then perform a simple breakdown of the standard DDR4 DRAM architecture and it's power consumption (Section 3). This is followed by proposals of NVM based solutions of the main memory that utilize the DRAM interface and also have similar configuration and architecture (Section 4), a system level analysis (Section 5) and a comparative analysis of the different STT-MRAM based solutions (Section 6). The internally devel-

We gratefully acknowledge funding by the H2020-RIA and the French ANR frameworks, under the Mont-Blanc 3 and CONTINUUM projects, respectively grants number 671697 and ANR-15-CE25-0007-01. Part of this work has also received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 687973 - GREAT (hetero-Geneous integRated magnetic tEchnology using multifunctional standardized sTack (MSS))

oped cross layer framework (entitled **SEAT** - *S*ystem benchmarking for *E*nablement of *A*dvanced *T*echnologies) allows us to carefully validate the feasibility of proposals across the different layers of abstraction (from the device to the system and applications). This is a much more realistic approach due to the uniformity of the process and silicon validated simulation checks at multiple levels.

II. State of the Art

Very few studies thoroughly and conclusively analyze the suitability of STT-MRAM for main memory. Meza et al. [5] analyze architectural changes to enable small row buffers in NVMs and concludes that NVM main memories with reduced row buffer size can achieve up to 67% energy gain over DRAM at a cost of some performance degradation. Kultursay et al. [6] evaluate STT-MRAM as a main memory for SPEC CPU2006 workloads and show that, without any optimizations, early-design STT-MRAM is not competitive with DRAM. The authors also propose partial write and write bypass optimizations in the STT-MRAM main memory that achieves performance comparable to DRAM while reducing memory energy consumption by 60%.

K Rho et al. [7] propose a 4Gb STT-MRAM, featuring a compact 9F2-cell that was developed using advanced technology (with the DRAM transistor). The 9090nm2 STT-MRAM was realized by using a compact gate pitch for the wordline, and an even finer metal-0 pitch for the bitline and source line. The pMTJ process is used here, due to its low switching current and high TMR. The 4Gb STTMRAM is not only based on a compact cell technology, but also employs a high-density chip architecture and circuit techniques to maximize array density. Kazi et al. [8] conduct a preliminary assessment of HPC system performance impact with STT-MRAM main memory based on industry estimates and perform a sensitivity analysis that correlates overall system slowdown trend with respect to average device latency due to unavailability of reliable timing info. Their results demonstrate that the overall system performance of large HPC clusters is not particularly sensitive to main-memory latency. Therefore, STT-MRAM, as well as any other emerging non-volatile memories with comparable density and access time, can be a viable option for future HPC memory system design.

III. Power Breakdown

A. DRAM

The commercially available DRAM system has a top-down hierarchical structure, that consists of the DRAM controller, channel, rank, chip, bank as shown in Figure 1. A channel is usually composed of a controller, DIMM/s and a few ranks that share the same command/address and data bus. As the I/O data width of a single memory chip is limited, multiple chips are populated in a rank and operate in lockstep to feed a wider data bus. The term rank was created and defined by JEDEC and refers to a data block that is 64 bits wide. On systems that support Error Correction Code (ECC) an additional 8 bits are added, which makes the data block 72 bits wide. In this paper, we use Micron 4Gib DDR4 x8 MT40A512M8 @2133 MHz chip as an example and the baseline design [9]. The chip configuration is 512 Meg x 8. Since each chip can only provide 8-bit (8b) data (so-called 8 device), eight chips are organized in a single rank to satisfy the total bus width of 64-bit (64b). For the DIMM, we utlize the DDR4 SDRAM RDIMM (Registered DIMM). Inside one chip, 16 internal banks are deployed as DRAM cell arrays (bank groups of 4 by 4 for x4 and x8). A bank can be accessed independently so that bank level parallelism is extensively exploited to assure memory bandwidth.

In our case, each bank has 64K rows and 1K columns. If each row contains 1K (1,024) column address staring points and each column stores 8 bits (1 byte), this would mean each row (page) is 8,192 bits (1,024 x 8 bits) or 1K bytes per bank. It should be noted that each page of memory is segmented evenly across Bank n of each IC for the associated rank. The DDR4 SDRAM uses an 8n-prefetch architecture to achieve high-speed operation. The 8n-prefetch architecture is combined with an interface designed to transfer two data words per clock cycle at the I/O pins. A single READ or WRITE operation for the DDR4 SDRAM consists of a single 8n-bit wide, four-clock data transfer at the internal DRAM core and two corresponding n-bit wide, one-half-clock-cycle data transfers at the I/O pins. Even though we focus on the commercial DDR4 in this work, this breakdown and other NVM substitutions with the DRAM interface can be easily extended to other DRAM technologies that have similar hierarchy, such as low-power DDR (LPDDR) or 3D-stacked DRAM (e.g., Wide I/O, Hybrid Memory Cube). The DRAM controller is assumed to run at 1GHz. V_{dd} supply voltage is 1.2V +/-60mV and V_{pp} for wordline boost is 2.5V, 125mV/+250mV. The chip has a cycle time of 0.937ns @ CL = 16/15. The random access latency is thus around 42-45ns.

To evaluate the DRAM power consumption, we utilize Microns DDR4 power calculator along with our study on DRAM internal architecture. Figure 2 shows the power breakdown. The power breakdown in Figure 2 can be classified into 3 categories : (1) Background Power. The background power takes up 28.6% of total power that accounts for the static power and refresh power, which consists of the powers from ACT-STBY (active standby), PRE-STBY (precharged standby), 'ACT-PDN' (active powerdown), PRE-PDN (precharged powerdown) and REF (refresh). (2) RD/WR/Termination Power.3 About 43.5% of the power comes from RD/WR/Termination which represents the power for the data movement, including the powers from RD (read burst), WR (write burst), READ I/O (read bus driver), and Write ODT (write signal on-die termination). (3) Activation Power. This consists of 26.9% of the DRAM power. Considerable prior work has been proposed to reduce the Background Power or RD/WR/Termination Power.

B. NVM Main Memory

We consider three different alternative main memory solutions in this paper. Our first proposal consists of a bit-cell with the high performance p-MTJ stack was developed

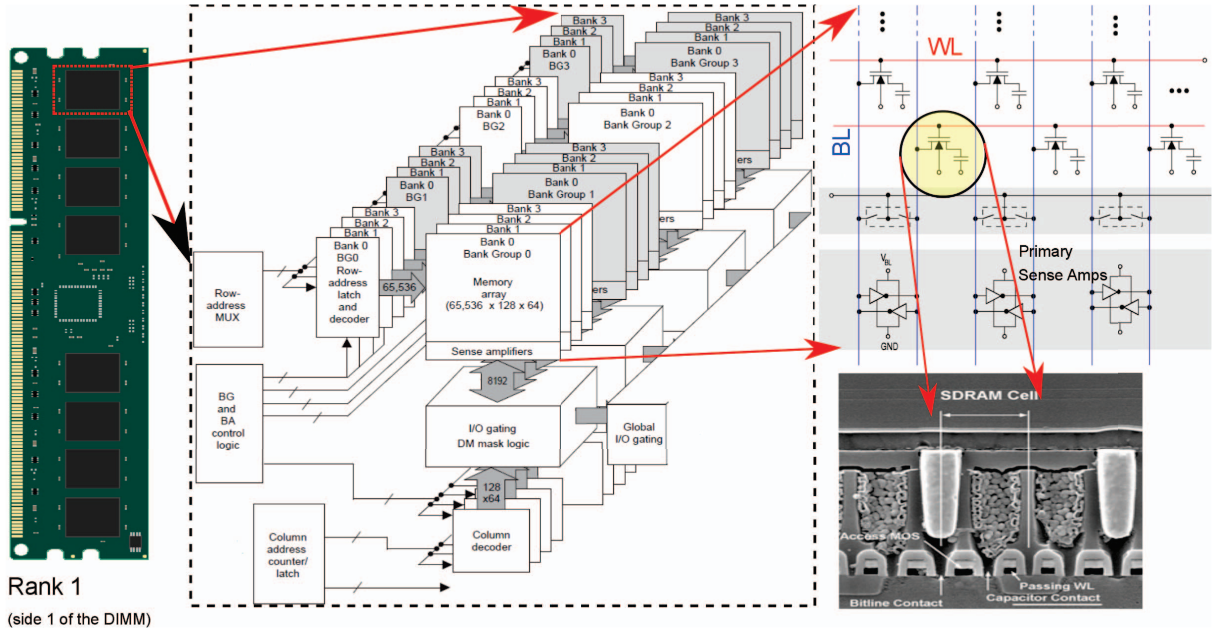

Fig. 1. DRAM internals

on 300mm Si wafers internally. The pMTJ stacks comprise of dual MgO interfaces interspersed by magnetic CoFeB electrodes, to improve the interfacial Perpendicular Magnetic Anisotropy (PMA). The free layer (FL) and reference layer (RL) consist of CoFeB-based multilayer stacks. The RL is coupled to a Co/Pt-based hard layer (HL) to form a synthetic anti-ferromagnet (iSAF), this allow us to keep our offset field near zero. The TMR is around 150% along with a resistance-area (RA) product (7 $\Omega\mu m^2$) from the CIPT measurement and the MTJ pillar dimension is 25nm. Here, we utilize the conventional planar transistor as a selector. This results in a unit area of 14F$_2$. V_{dd} supply voltage is taken to be 0.9V. The sense amplifier is enabled when the difference between the reference and the cell read out reaches around 70mV. The drawback here is that the unit cell area of STT-MRAM is still much larger than that of DRAM, making STT-MRAM not cost-competitive to contemporary DRAM. The random access latency here is 30ns.

The second proposal borrows from [7]. Two different supply voltages are present here to help boost the WL and compensate for the weak DRAM transistor ($V_{dd}1$ = 1.2V and $V_{dd}2$ = 1.8V). The chip operation is similar to that of mobile LPDDR2-S4B DRAM, resulting in a random read latency of 50ns. This chip is however about 1.4 times smaller than the previous proposal. The third and the final proposal involves the use of a highly nonlinear bi-directional selector device that opens up the possibility of the stacking to reduce the unit area and be cost competitive with contemporary DRAM [10]. This results in a unit area of 4F$_2$. The drain voltages of selected WL/BL driving transistors are 1.8/0 V, and the gate voltage of selected WL/BL driving transistors is 3.3 V. The line resistance was assumed to be 2 Ohm/sq, and the parasitic capacitance was assumed to be 30 aF/cell. These values are typical of parasitic parameters in crossbar arrays using 25-nm technology. The STT-MTJ parameters in this proposals are as follows: TMR of approximately 110% and a parallel resistance (RP) was 25 kOhm for the dimensions of 30 30 nm^2. Presently, since selector technology is not fully optimized, the higher leakage contribution compared to conventional transistor options can lead to an increase in background power. The random access latency here is 40ns. The power breakdown profile in Figure 2 alludes to this. Thus, while the 3rd proposal has the smallest unit area and will be the most cost competitive, it will also be the most power hungry option.

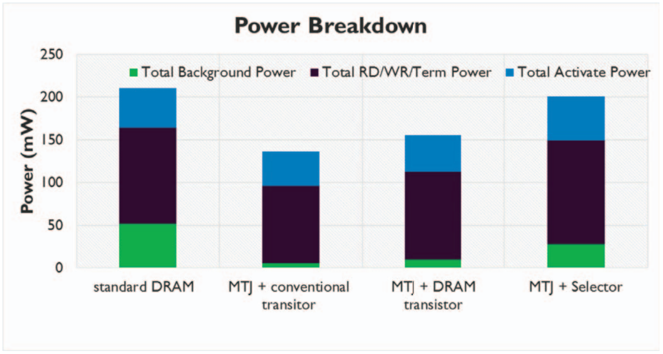

Fig. 2. A comparative power breakdown of the different Main memory solutions.

IV. System level Analysis

The system level analysis is carried out by combining two modeling and simulation tools: gem5 [11] and NVMain

[12]. From input parameters characterizing a given system configuration, these tools are co-simulated in order to produce execution statistics including execution time, memory transactions and related latencies, power/energy consumption.

The gem5 computer architecture simulator [11] provides an accurate evaluation of system performance thanks to its high configurability for a fine grained architecture modeling. Its full-system simulation mode runs unmodified operating systems. It has been used for the accurate modeling and evaluation of modern multicore architectures, such as ARM big.LITTLE [13]. NVMain [12] is an architectural-level simulator that enables to model main memory design with both DRAM and emerging non-volatile memory technologies. It is well-integrated together with the gem5 simulator, which makes it possible to achieve cycle-approximate multicore system simulation together with cycle-level simulation of a variety of main memory technologies in the system. In addition, NVMain can also accommodate traces as input for facilitating design space exploration by considering either user-defined emerging memory technologies or those provided together with the simulator. In this paper, we consider both options by calibrating a few DRAM models based on available datasheets.

A given memory configuration in NVMain is described by several tens of parameters regarding the corresponding architecture for which a template is illustrated in Figure 3, and its timing and energy consumption. These parameters are leveraged by during the co-simulation NVMain to estimate the delay and energy consumption of the main memory in the execution of a particular workload. For available memories, these configuration parameters are gathered from datasheet information and are very accurate.

A. gem5-NVMain co-simulation

In order to obtain a good simulation accuracy, gem5 and NVMain need a tight interaction. NVMain receives memory requests from gem5 at precise time instants; these requests should trigger different NVMain actions (i.e., request enqueueing, scheduling, transfer to main memory, bank latency computation and transfer back to controller) that need to be executed (to preserve dependencies) before gem5 can proceed with its simulation. Accordingly, gem5 and NVMain have been integrated (by NVMain developers) in the co-simulation environment illustrated in Figure 4 as part as the used framework.

An interfacing object, namely NVMainMemory, is created inheriting from both, the *AbstractMemory* gem5 class and the *NVMObject* NVMain class. As a gem5 object, the newly created interfacing object can respond to the *recvTimingReq* method triggered by the last level cache during the gem5 simulation. Each received request includes a gem5 memory request packet, namely *pkt*, whose information is transferred onto a new NVMain memory request packet, namely *req*, that is then submitted to the NVMain simulator. To keep track of the issued memory requests, the interfacing object stores both the *pkt* and the corresponding *req* in a *map* data structure. Once NVMain has completed the submitted memory operation, the interfacing object is notified through the *RequestCompleted* method. Then, the corresponding *pkt* is retrieved from the *map* data structure and sent back to gem5 as part of the request completed confirmation through the *sendTimingResp* method, which is triggered by a *RespondEvent* (previously scheduled in the gem5 event queue by the interfacing object). Furthermore, the *NVMainMemory* interfacing object and the NVMain simulator are cycled every clock cycle through the *tick* method, which propagates the standard gem5 cycling to NVMain through the *Cycle* method.

B. Framework soundness evaluation

Experiments have been carried out to evaluate the correctness of gem5-NVMain framework by considering a set of relevant applications (from the *SPEC CPU2006* benchmark suite [14]) running on a system with the aforementioned Micron 4Gib DDR4 x8 MT40A512M8 @2133 MHz main memory baseline design. The experimental setup is detailed in Table I. An important motivation of this evaluation framework is to enable a rapid design space exploration while integrating NVMs into main memory. For this reason, our first experiment focuses on the relevance assessment of gem5 system call emulation (SE) and full system (FS) simulation modes. The SE mode is used to simulate individual applications without the need to model devices or an operating system (OS). System calls are emulated by calling the host OS, enabling a fast simulation. On the other hand, FS mode executes both user-level and kernel-level instructions and models a complete system including the OS and devices. This includes support for interrupts, exceptions, privilege levels, and I/O devices. For illustration, Table II summarizes an evaluation of the *perlbench* application in both SE and FS mode combined with NVMain.

TABLE I
ARCHITECTURE CONFIGURATION

System element	Configuration
Processor	64-bit RISC ARMv8 single core, out-of-order $2GHz$
L1 Instruction cache	Private, $64kB$
L1 Data cache	Private, $64kB$
L2 cache	Shared, $1MB$
Main Memory	2 channels 2 ranks per channel 8 chips per rank Memory controller at $1GHz$ Micron 4Gib DDR4 x8 MT40A512M8 @2133 MHz chip

TABLE II
SYSTEMCALL EMULATION VS FULL SYSTEM SIMULATION

Parameter	SE	FS
Runtime (second)	0.006	0.0104
Main memory reads (kilo)	17	21
Main memory writes (kilo)	2	43
Main memory bandwidth (MB/s)	205	400

The SE mode estimates an execution time of 0.006 seconds, instead, the FS simulation estimates 0.0104 seconds. The time

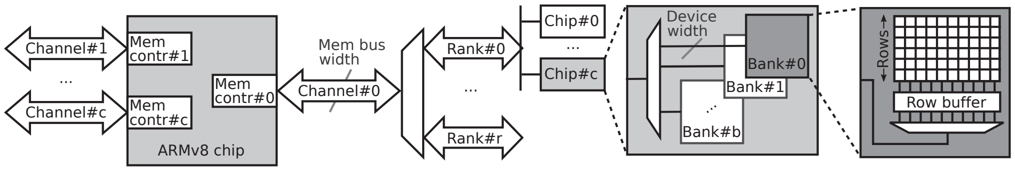

Fig. 3. NVMain memory architecture template that needs to be instantiated in the configuration files.

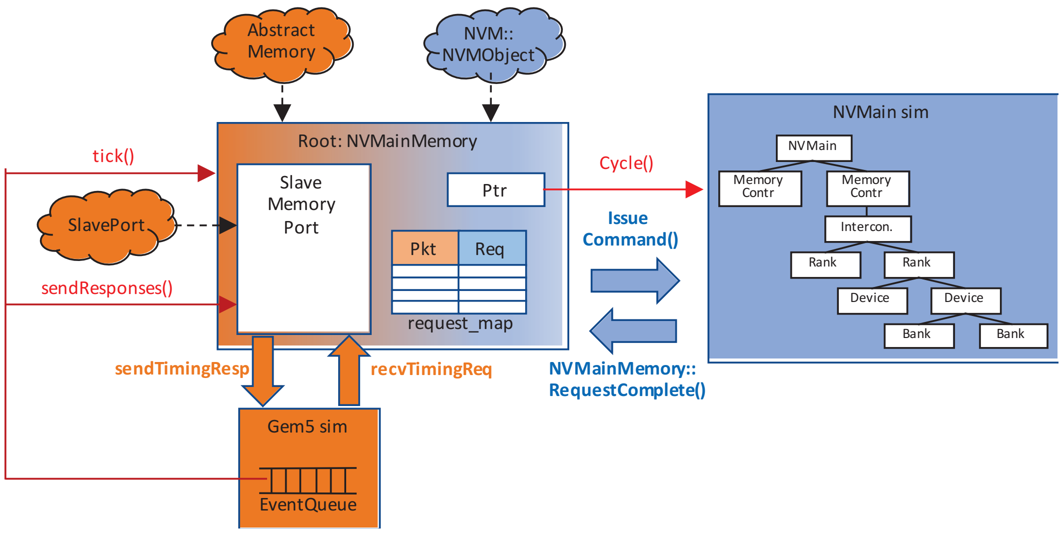

Fig. 4. Software architecture of the gem5/NVMain co-simulation.

difference comes from the fact that SE does not amount for the time spent by the architecture in the handling of OS system calls. However, such events are important to consider when evaluating a particular memory architecture. The number of memory reads and writes to main memory can differ by an order of magnitude between the FS and SE mode. Alternative fast gem5 simulation approaches such as [15] could be rather considered. Now, let us consider the gem5 FS mode together with NVMain in order to evaluate the accuracy of the framework compared to the main memory design reference (i.e. Micron DDR4). A set of system activity statistics are collected through Table III, based on the execution of the *SPEC CPU2006 perlbench*, *h264ref*, *gobmk* and *bzip2* applications.

TABLE III
MAIN MEMORY STATISTICS

Parameter	perlbench	h264ref	gobmk	bzip2
Main memory reads	$1.13e6$	$5.24e7$	$4.32e7$	$9.51e7$
Main memory writes	$8.42e5$	$2.78e7$	$4.11e7$	$4.48e7$
Row buffer hits	$5.66e5$	$2.96e7$	$4.84e7$	$1.67e7$
Row buffer misses	$1.41e6$	$5.06e7$	$3.59e7$	$1.23e8$
Average latency (ns)	51.64	50.2	42.28	56.46
Average queue latency (ns)	10.4	12.19	35.88	1.79
Average total latency (ns)	62.04	62.39	78.15	58.25
Simulation cycles	$1.31e10$	$1.88e10$	$2.49e10$	$1.26e10$
Wakeup count	$3.03e7$	$4.16e8$	$3.82e8$	$7.09e8$

The row buffer locality exploited by the memory controller is indicated by the row buffer hits/misses values. Obviously, an execution optimised for minimal main memory latency will aim at minimising the more expensive row buffer misses. The total main memory access latency is split between average memory access latency and average queue latency. The latter is the average time a memory request spends in the memory queue waiting to be served. Finally, one can also observe how often the memory simulation was triggered by comparing the wakeupCount with respect to the total number of simulation cycles. Another experiment has been conducted with 12 applications of the *PARSEC* 3.0 benchmark suite [16] demonstrating that NVMain rarely accounts for more than 15% of the overall simulation time (Figure 5, right). Moreover, it is observed that the NVMain contribution on the total simulation time is not correlated to the main memory bandwidth (Figure 5). Among all the simulated *PARSEC* applications, the simulation time varies from about 4 to 36 hours. Each simulation traces consider a minimum of $1,500,00$ samples which corresponds to about a 4-hour execution time.

V. COMPARATIVE SYSTEM LEVEL ANALYSIS AND CONCLUSION

Figure 6 highlights a relative comparison between the different NVM main memory solutions and DRAM from a system energy consumption point of view. From the graphs it is quite clear that the NVM solutions fare better than the DRAM almost every-time. It is only when the write access dominates the benchmark/application suite that the

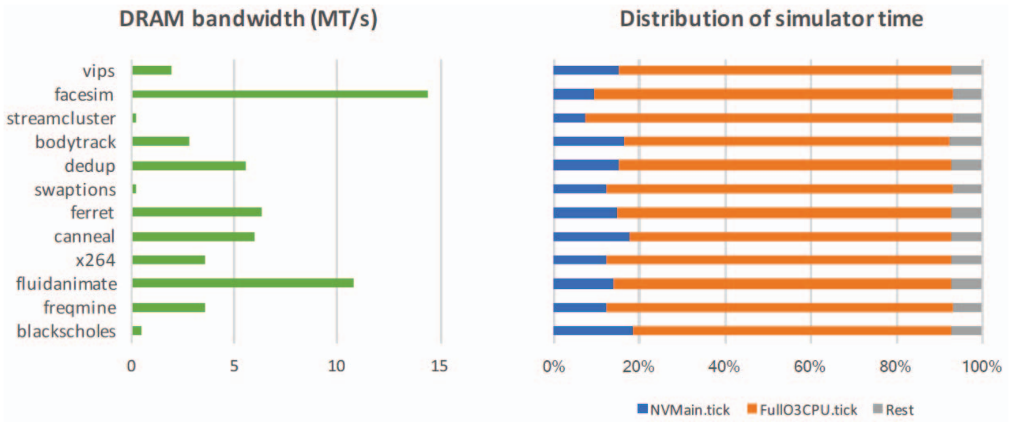

Fig. 5. DRAM bandwidth and Full System simulation time breakdown.

NVM energy consumption shoots up. This can be attributed to the higher write energy of MRAM compared to DRAM and it's own read (asymmetric nature). The high resistance MTJ cells in 'MTJ + Selector' option leads to increased RC delay and thus longer access times. This combined with the higher voltage requirements due to the immaturity of the selector technology leads to the relatively higher energy energy consumption (only 8% average energy reduction across the different benchmarks compared to conventional DRAM). The 'MTJ + selector' option will be most viable in the future if and when the selector is optimized for leakage and dynamic power. While, the 'MTJ + conventional transistor' is the most energy efficient among the NVM proposals (27% average energy reduction across the different benchmarks compared to conventional DRAM), it is also the least cost effective due to the large cell size (more than 2 times that of DRAM). Finally, it must be mentioned that while we assume a DRAM like interface for these NVM proposals, the full benefits of NVMs can only be leveraged through micro-architectural modifications and custom interfaces that maximize the benefits of NVMs.

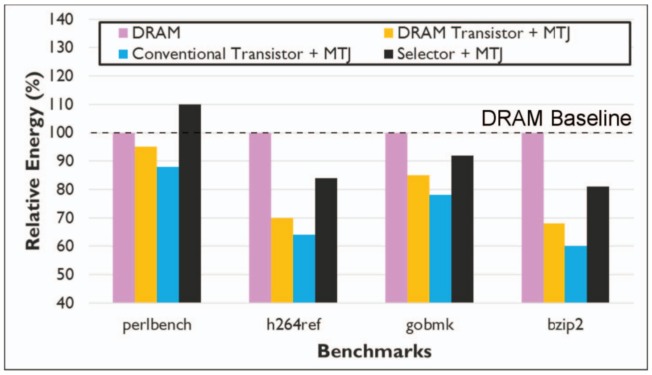

Fig. 6. The relative energy consumption of the NVM based main memory solutions and conventional DRAM

References

[1] Q. Deng, L. Ramos, R. Bianchini, D. Meisner, and T. Wenisch, "Active low-power modes for main memory with memscale," *IEEE Micro*, vol. 32, no. 3, pp. 60–69, May 2012.

[2] S. Li, P. Chi, J. Zhao, K. T. Cheng, and Y. Xie, "Leveraging nonvolatility for architecture design with emerging nvm," in *IEEE NVMSA*, Aug 2015, pp. 1–5.

[3] Y. Xie, "Modeling, architecture, and applications for emerging memory technologies," *IEEE Design Test of Computers*, vol. 28, no. 1, pp. 44–51, Jan 2011.

[4] D. Apalkov *et al.*, "Spin-transfer torque magnetic random access memory (stt-mram)," *J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 2, pp. 13:1–13:35, May 2013.

[5] J. Meza, J. Li, and O. Mutlu, "Evaluating row buffer locality in future nonvolatile main memories," Carnegie Mellon University, IBM Watson Research Center, Tech. Rep. SAFARI Technical Report No. 2012-002, 12 2017.

[6] E. Kltrsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating stt-ram as an energy-efficient main memory alternative," in *IEEE ISPASS*, April 2013, pp. 256–267.

[7] K. Rho *et al.*, "23.5 a 4gb lpddr2 stt-mram with compact 9f2 1t1mtj cell and hierarchical bitline architecture," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 396–397.

[8] K. Asifuzzaman *et al.*, "Performance impact of a slower main memory: A case study of stt-mram in hpc," in *MEMSYS*. New York, NY, USA: ACM, 2016, pp. 40–49.

[9] "4gb ddr4 sdram," Micron Technology inc., Tech. Rep. 4gb ddr4 dram.pdf - Rev. H 5/17 EN, 2012.

[10] H. Lim, S. Lee, and H. Shin, "Switching time and stability evaluation for writing operation of stt-mram crossbar array," *IEEE Transactions on Electron Devices*, vol. 63, no. 10, pp. 3914–3921, Oct 2016.

[11] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[12] 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*, vol. 14, no. 2, pp. 140–143, July 2015.

[13] A. Butko, F. Bruguier, A. Gamati, G. Sassatelli, D. Novo, L. Torres, and M. Robert, "Full-system simulation of big.little multicore architecture for performance and energy exploration," in *2016 IEEE MCSOC*, Sept 2016, pp. 201–208.

[14] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006. [Online]. Available: http://doi.acm.org/10.1145/1186736.1186737

[15] A. Butko, R. Garibotti, L. Ost, V. Lapotre, A. Gamatie, G. Sassatelli, and C. Adeniyi-Jones, "A trace-driven approach for fast and accurate simulation of manycore architectures," in *ASPDAC*, Jan 2015, pp. 707–712.

[16] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *PACT*. New York, NY, USA: ACM, 2008, pp. 72–81.