# Designing Scratchpad Memory Architecture with Emerging STT-RAM Memory Technologies

Peng Wang,[1] Guangyu Sun,[1] Tao Wang,[1] Yuan Xie,[2] Jason Cong[1,3,4]
[1]Center for Energy-Efficient Computing and Applications, School of EECS, Peking University, Beijing, 100871, China
[2]Computer Science and Engineering Department, Pennsylvania State University, PA 16802, USA
[3]Computer Science Department, University of California, Los Angeles, CA 90095, USA
[4]UCLA/PKU Joint Research Institute in Science and Engineering
{wang_peng, gsun, wangtao}@pku.edu.cn, yuanxie@cse.psu.edu, cong@cs.ucla.edu

***Abstract*—Scratchpad memories (SPMs) have been widely used in embedded systems to achieve comparable performance with better energy efficiency when compared to caches. Spin-transfer torque RAM (STT-RAM) is an emerging nonvolatile memory technology that has low-power and high-density advantages over SRAM. In this study we explore and evaluate a series of scratchpad memory architectures consisting of STT-RAM. The experimental results reveal that with optimized design, STT-RAM is an effective alternative to SRAM for scratchpad memory in low-power embedded systems.**

## I. INTRODUCTION

Energy consumption is an important design issue for embedded systems. Since on-chip SRAM caches usually consume 25% to 45% of the total chip power [1], the on-chip memory in embedded processors is often configured as a software-managed scratchpad memory (SPM). The SPM does not have the tag array and relevant comparison logic that cache uses to support the fast lookup and dynamic mapping of data or instructions in off-chip memory. Therefore, it is more energy- and area-efficient than caches [1]. Moreover, SPM is managed by software, which can often provide better timing predictability in real-time systems [2].

As the technology advances, traditional SRAM-based on-chip memory has become a bottleneck for energy-efficient design due to its high leakage power. The emerging nonvolatile memory (NVM) technologies, such as the spin-transfer torque RAM (STT-RAM) and phase change RAM (PCRAM), are possible solutions for future memory systems. Compared to SRAM, STT-RAM and PCRAM provide higher density and lower leakage power. In addition, STT-RAM can significantly exceed PCRAM with respect to endurance, access latency, and dynamic power, while PCRAM has higher density. Previous works have showed that STT-RAM is more suitable for performance-critical last-level caches [3][4], while PCRAM is promising as an alternative for DRAM in the main memory [5]. Therefore, we primarily focus on STT-RAM-based design in this paper.

Although STT-RAM has many attractive characteristics, such as low leakage power and high density, there is one major drawback we need to overcome. Unlike SRAM, in which read and write operations consume the same time and energy, a write operation of STT-RAM needs much longer latency and higher energy than a read operation. Moreover, the latency and energy of write operations in conventional STT-RAM are several times larger than those of SRAM. Since on-chip memory is nearest to CPU in the memory hierarchy, reducing the access latency of on-chip memory is critical to performance.

New STT-RAM device cell designs have been developed to mitigate the low speed and high energy of write operations. Perpendicular MTJs (PMTJ) were developed [6][7] to achieve a low switching current while maintaining a high thermal stability for nonvolatility of STT-RAM. Through careful device-architecture co-optimization [8], we can significantly alleviate the write problem of STT-RAM SPM. In this paper we use the methodology in [8] to generate different types of STT-RAM configurations. We attempt to find an efficient SPM solution.

We also explore the design space of hybrid SPM architectures consisting of STT-RAM and SRAM in order to take advantage of the fast writes of SRAM. Because SPM is explicitly managed by software, programmers can tune the software manually or use special compiler support to counteract the negative effects of the write operations of STT-RAM SPM. We use a data allocation scheme aimed at the hybrid SPM architecture to reduce the number of writes on STT-RAM. The basic idea is to allocate the most-written data into SRAM and the most-read data to STT-RAM.

The major contributions of this work are summarized as follows:

- We explored a low-power SPM architecture that contains STT-RAM in an embedded system. According to our experimental results, the device-architecture co-optimized PMTJ-based STT-RAM is a potential replacement to SRAM-based SPM.
- We conducted a sensitivity analysis to various area ratios of a hybrid SPM architecture consisting of STT-

This work is supported by the National Natural Science Foundation of China 61103028, 61202072, and Research Fund for the Doctoral Program of Higher Education of China 20110001110099, 20110001120132.

RAM and SRAM. The experimental results show that the hybrid SPM with a 2:1 SRAM:STT-RAM area ratio achieves the best performance and energy-delay product, while the pure STT-RAM configuration outperforms the others in energy.

The remainder of this paper is organized as follows. Related work is discussed in Section II. Section III presents the scheme used for allocation of SPM data. The experiments for different SPM configurations are presented and analyzed in Section IV. Section V concludes our work.

## II. RELATED WORK

Effective utilization of SPM is critical for an SPM-based system. Research on automatic data allocation for SPM has focused on how to place the frequently used data in a program in SPM so as to maximize for both improved performance and energy consumption. The process of data allocation determines the location of each piece of data in SPM. Compilers or programmers need to decide on an appropriate data allocation before execution. Several compiler-based data allocation techniques, e.g. [9][10][11][13], have been proposed. The existing techniques can be classified into two major categories: static methods and dynamic methods.

In the static methods, the contents of the SPM are determined before the execution of a program and never change during the execution. These static allocation approaches either use greedy strategies to find an efficient solution, or model the problem as a knapsack problem or an integer linear programming problem (ILP) to find an optimal solution. The profiling method is often used. Avissar et al. [9] proposed a memory allocation scheme for global variables that is optimal in relation to the profiling provided. Cong et al. [10] proposed a compile-time technique which uses the SPM to reuse the data with consideration of prefetching in affine-indexed arrays.

In the dynamic methods, the contents of the SPM may change during the program execution. Dynamic methods based on compile-time techniques change the SPM allocation based only on compile time decisions and profiling information. Udayakumaran et al. [13] proposed a dynamic data allocation method for global and stack data. Verma et al. [11] proposed an overlay-based memory allocation method for both code and data that uses ILP to find the optimal memory allocation and minimize energy consumption for a given profile. Dynamic methods are usually more efficient than static ones in exploiting the benefits of SPMs.

None of the methods above consider the latency and energy overhead associated with the write operations of STT-RAM because they are aimed at the traditional SPM consisting of SRAM. Hu et al. [12] design an SPM data allocation algorithm to counteract negative effects of the write operation of STT-RAM. However, their design uses the PCRAM as the SPM, which is not an ideal choice compared to STT-RAM in terms of performance and energy. They also did not evaluate the impact of using different device technologies on the overall system performance and energy.

## III. DATA ALLOCATION SCHEME FOR HYBRID SPM

In hybrid SPM architecture, STT-RAM SPM and SRAM SPM share the same address space with the main memory. The CPU can load data from both SPMs, and data can be migrated between them using special instructions.

We use a dynamic data allocation scheme based on Hu et al.'s scheme [12]. A program is divided into several program regions by a compiler, as in [13]. Data allocation is performed for each program region before execution using compiler-inserted code at the beginning of a program region. The scheme allocates the most-written data into SRAM and the most-read data into STT-RAM optimally in a region.

First, we evaluate costs for each data item in the region. We define three costs, S(D), T(D) and M(D), which are the total costs for the data D if D is allocated in SRAM, STT-RAM, and main memory, respectively. The costs can be computed as Equation 1.

$$Total\ cost = \sum(migrating\ cost + number\ of\ read \times cost\ per\ read + number\ of\ write \times cost\ per\ write) \quad (1)$$

Here the cost per read/write is the access time of reading/writing SPM, and the moving cost is the time needed for migrating data between SRAM SPM, STT-RAM SPM, and main memory. We can obtain the number of read or write in the region through profiling.

Second, we allocate each data item to the appropriate place to minimize the total cost. A variant of the dynamic programming solution in [12] to the multiple knapsack problem is used to select the best place for each data item.

## IV. EXPERIMENTS

### A. Experimental Setup

We evaluate the proposed design on a simulation platform built upon Simics [14] with GEMS [15]. Table I provides the parameters used in our baseline model. Notice that the configuration of the CPU and main memory remains the same through all simulations. Our test benches consist of nine benchmark applications from MiBench [16]. We use Udayakumaran's algorithm [13], which is a greedy algorithm, to manage the data allocation for pure SRAM SPM.

TABLE I. BASELINE SIMULATION PARAMETERS

| | |
|---|---|
| Simulator | Simics with GEMS |
| CPU | Single 1GHz processor |
| SPM | 16KB, 2-cycle latency SRAM |
| Main Memory | 64MB DRAM, 100-cycle access |
| Benchmark | MiBench |

We use the NVM simulator NVsim [17] to estimate the read/write latencies and energy consumption for a given size of SRAM and STT-RAM. We use 45nm technologies with the tool and select three different types of optimization targets for SPM—latency optimized, energy-delay-product (EDP) optimized, and energy optimized STT-RAM. Table II shows the parameters of SPM used in our experiments.

TABLE II. CONFIGURATION OF 45NM 16KB STT-RAM SPM AND COMPARISON WITH SRAM

| | *Latency opt. STT-RAM* | *EDP opt. STT-RAM* | *Energy opt. STT-RAM* | *SRAM* |
|---|---|---|---|---|
| Area ($mm^2$) | 0.031 | 0.021 | 0.018 | 0.093 |
| Read latency (ns) | 0.906 | 0.835 | 0.813 | 1.085 |
| Write latency (ns) | 0.997 | 3.135 | 5.128 | 1.085 |
| Read energy (pJ) | 10.951 | 7.091 | 6.516 | 20.589 |
| Write energy (pJ) | 29.56 | 21.87 | 22.65 | 14.501 |
| Leakage (mW) | 0.924 | 0.616 | 0.606 | 7.916 |

## B. *Experimental Results*

We explore two different approaches to leverage STT-RAM for scratchpad memory design: the first approach is a direct replacement of SRAM with a pure STT-RAM scratchpad memory; the second approach is a hybrid SRAM/STT-RAM scratchpad memory design.

**Pure STT-RAM Scratchpad Memory.** First we evaluated the case of direct replacement of SRAM scratchpad with STT-RAM scratchpad. We compared the performance (in terms of instruction per cycle, IPC) and the energy consumption of different configurations. We used the conventional pure SRAM (ALL-SRAM) SPM design as the baseline. Figure 1 shows the IPC performance of the simulated SPM designs normalized to the baseline ALL-SRAM case

***Performance.*** On average, implementing the SPM using the "energy optimized" STT-RAM design results in more than an 8% IPC degradation due to the longer write latency. However, the performance of the SPM using the "latency optimized" and "EDP optimized" STT-RAM design is significantly improved compared to that of the SRAM SPM. The average normalized IPCs of latency optimized and EDP optimized are 1.24 and 1.10, respectively. The performance improvement of latency optimized and EDP optimized SPM compared to the SRAM baseline comes from the shorter read latency, even though its write latency is still longer (as shown in Table II). However, SPM read accesses are far more frequent than write accesses in most benchmarks. In some benchmarks, for example mp3enc, the latency optimized design achieves a better than 30% improvement in IPC.

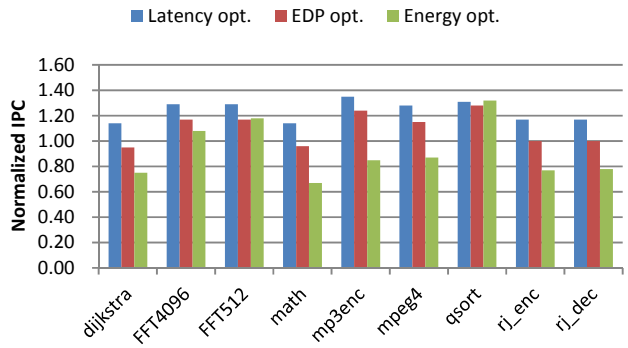


Figure 1. Comparison of performance.

***Energy.*** The energy consumptions of different SPM designs normalized to the baseline ALL-SRAM cases are summarized in Figure 2. The results show that implementing the SPM with the latency optimized STT-RAM is much less energy-efficient than the SRAM by 20%. The EDP optimized and energy optimized STT-RAM SPM designs achieved significant energy savings of 18% and 31%, respectively, compared to the SRAM baseline.

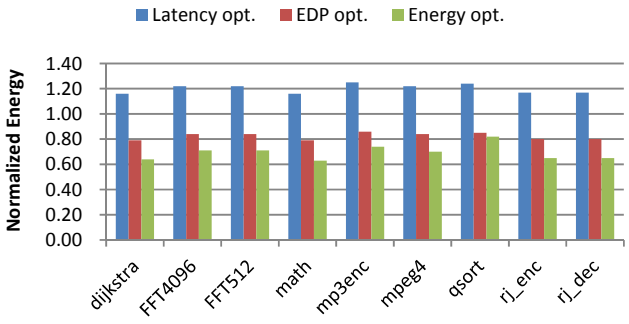


Figure 2. Comparison of energy consumption.

**SRAM/STT-RAM Hybrid Scratchpad Memory.** In order to leverage the benefit of both SRAM (faster write) and STT-RAM memory (higher density), we propose a SRAM/STT-RAM hybrid scratchpad memory. We evaluate the proposed hybrid SPM with different SRAM and STT-RAM area ratios. The total area is the fixed value of a 64KB SRAM SPM. For example, the 1:1 area ratio means that the hybrid SPM has 32KB of SRAM and 128KB of STT-RAM, while the baseline SPM has 64KB of SRAM. Since STT-RAM can be made about four times denser than SRAM, these two SPMs have a similar silicon area. In the experimental results, we use the EDP optimized STT-RAM design for the hybrid design. To be consistent with the previous section, we normalize the simulation results to the ALL-SRAM design.

Figure 3 compares the normalized performance results. We can see that the 2:1 SRAM:STT-RAM area ratio is the best, with an average 36% improvement over the baseline. Figure 4 shows the corresponding energy result with EDP optimized STT-RAM. We can see that on average the "ALL STT-RAM" configuration is the best, with 29% less energy.

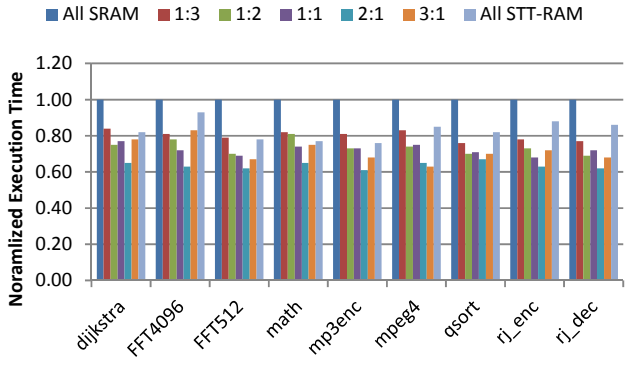


Figure 3. Normalized performance comparison of various area ratios for hybrid SPM design. (normalized to ALL-SRAM baseline, with EDP optimized STT-RAM design).

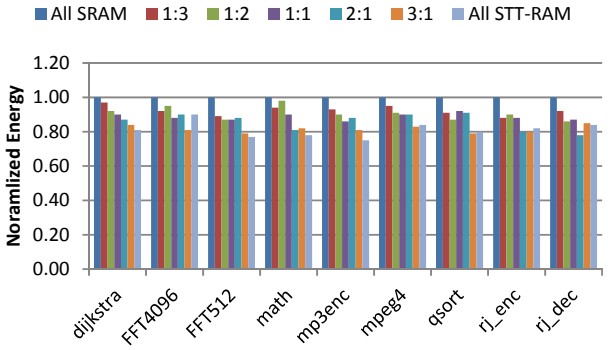


Figure 4. Normalized energy comparison of various area ratio hybrid SPM design (normalized to ALL-SRAM baseline, with EDP optimized STT-RAM design).

Figure 5 shows the corresponding energy-delay product (EDP) result with EDP optimized STT-RAM. The 2:1 configuration achieves the best results on average, reducing 45% of total energy-delay.

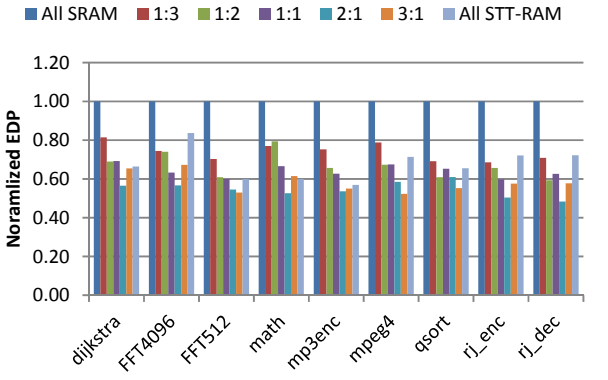


Figure 5. Normalized energy-delay product (EDP) comparison of various area ratios for hybrid SPM design (normalized to ALL-SRAM baseline, with EDP optimized STT-RAM design).

There are two reasons why the performance and energy-efficiency of the hybrid SPM are improved. First, the allocation scheme allocates the most-read data into STT-RAM and the most-written data into SRAM. Therefore, we can take advantage of the efficient read of STT-RAM and reduce the expensive writes to the STT-RAM. Second, since STT-RAM has a higher density than SRAM, for the same area, the hybrid SPM has a larger capacity. Therefore, there are fewer off-chip memory accesses in the hybrid SPM than in smaller SRAM SPM. The evaluation results also show that there is no single hybrid SPM design which can outperform the others in terms of performance, power, or EDP. It means that a re-configurable SPM design that can dynamically adapt configurations and data management has the potential to further improve STT-RAM SPM design.

## V. CONCLUSIONS

STT-RAM has many attractive characteristics, such as low leakage power and high-density, which make it a promising universal memory replacement in low-power embedded systems. In this paper we verified three types of STT-RAM as scratchpad memory replacements in an embedded system. We also evaluated the hybrid SPM, which consists of STT-RAM and SRAM, to leverage the benefit of both SRAM and STT-RAM. According to our experimental results, the hybrid SPM architecture can outperform SPMs consisting of either pure SRAM or pure STT-RAM.

## REFERENCES

[1] R. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, and P. Marwedel, "Scratchpad memory: design alternative for cache on-chip memory in embedded systems," in *Proc. CODES*, 2002, pp. 73–78.

[2] P. Marwedel, L.Wehmeyer, M. Verma, S. Steinke, and U. Helmig, "Fast, Predictable and Low Energy Memory References through Architecture-aware Compilation," in *Proc. ASPDAC*, 2004, pp. 4-11.

[3] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Hybrid Cache Architecture with Disparate Memory Technologies," in *Proc. ISCA*, 2009, pp. 34–45.

[4] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs," in *Proc. HPCA*, 2008, pp. 239–249.

[5] B. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *Proc. ISCA*, 2009, pp. 2–13.

[6] P. Khalili Amiri, Z. M. Zeng, J. Langer, H. Zhao, G. Rowlands, Y. J. Chen, I. N. Krivorotov, J. P. Wang, H. W. Jiang, J. A. Katine, Y. Huai, K. Galatsis, and K. L. Wang, "Switching current reduction using perpendicular anisotropy in CoFeB-MgO magnetic tunnel junctions," *Applied Physics Letters*, vol. 98, no. 11, pp. 112 507–112 507–3, 2011.

[7] Z. R. Tadisina, A. Natarajarathinam, B. D. Clark, A. L. Highsmith, T. Mewes, S. Gupta, E. Chen, and S. Wang, "Perpendicular magnetic tunnel junctions using co-based multilayers," *Journal of Applied Physics*, vol. 107, no. 9, pp. 09C703 –09C703–3, 2010.

[8] C. Xu, D. Niu, X. Zhu, S. H. Kang, M. Nowak, and Y. Xie, "Device-architecture co-optimization of STT-RAM based memory for low power embedded systems," In *Proc. ICCAD*, 2011, pp. 463-470.

[9] O. Avissar, R. Barua, and D. Stewart, "An optimal memory allocation scheme for scratch-pad-based embedded systems," *ACM Trans. Embedded Comput. Syst.*, vol. 1, no. 1, pp. 6–26, 2002.

[10] J. Cong, H. Huang, C. Liu, and Y. Zou, "A Reuse-Aware Prefetching Scheme for Scratchpad Memory,". In *Proc. DAC*, 2011, pp. 960-965.

[11] M. Verma and P. Marwedel. "Overlay techniques for scratchpad memories in low power embedded processors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 8, pp. 802–815, 2006.

[12] J. Hu, C. J. Xue, Q. Zhuge, W.-C. Tseng, and E. H.-M. Sha, "Towards energy efficient hybrid on-chip scratch pad memory with non-volatile memory," in *Proc. DATE*, 2011, pp. 1–6.

[13] S. Udayakumaran and R. Barua, "Compiler-decided dynamic memory allocation for scratch-pad based embedded systems," in *Proc. CASES*, 2003, pp. 276–286.

[14] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A Full System Simulation Platform," in *IEEE Computer*, 2002, pp. 50–58.

[15] M. Martin, D. Sorin, B. Beckmann, M. Marty, M. Xu, A. Alameldeen, K. Moore, M. Hill, and D. Wood, "Multifacet's General Execution-Driven Multiprocessor Simulator(GEMS) Toolset," in *Computer Architecture News*, 2005, pp. 92–99.

[16] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proc. WWC*, 2001, pp. 3–14.

[17] X. Dong, N. P. Jouppi, and Y. Xie, "Pcramsim: System-level performance, energy, and area modeling for phase-change ram," in *Proc. ICCAD*, 2009, pp. 269-275.