Asymmetric-access Aware Optimization for STT-RAM Caches with Process Variations*

Yi Zhou
Beijing Institute of Technology
zhouyi.initial@gmail.com

Chao Zhang
Peking University
zhang.chao@pku.edu.cn

Guangyu Sun
Peking University
gsun@pku.edu.cn

Kun Wang
IBM Research - China
wangkun@cn.ibm.com

Yu Zhang
IBM Research - China
zhyu@cn.ibm.com

ABSTRACT

STT-RAM (Spin Transfer Torque Random Access Memory) has been extensively researched as a potential replacement of SRAM (Static RAM) as on-chip caches. Prior work has shown that STT-RAM caches can improve performance and reduce power consumption because of its advantages of high density, fast read speed, low standby power, etc. However, under the impact of process variations, using worst-case design can induce significant performance and power overhead in STT-RAM caches. In order to overcome the problem of process variations, we propose to apply the variable-latency access method to STT-RAM caches by introducing a variation-aware LRU (Least Recently Used) policy. Moreover, we show that simply applying traditional variable-latency access method is inefficient due to the read/write asymmetry. First, we demonstrate that a write-oriented data migration is preferred. Second, a block remapping is necessary to prevent some cache sets from being significantly affected by process variations. After using our techniques, the experimental results show that the performance can be improved by 13.8% and power consumption can be reduced by 14.1% compared to a prior approach [3].

Categories and Subject Descriptors

B.3.2 [**Memory Structures**]: Design Styles–*Cache memories*

Keywords

STT-RAM caches, process variations, asymmetric-access

1. INTRODUCTION

As the mainstream of computer system moves to the regime of multi-/many-core designs, the number of processing cores integrated on the same chip has been increasing. It also results in an increasing requirement for on-chip memory capacity to bridge the gap between processors and off-chip main memory. The traditional SRAM technology, however, cannot satisfy the requirement due to its problems of low cell density, high standby power, and vulnerable to soft errors [4], which lead to the well-known problem "memory wall" [14]. In order to attack this problem, STT-RAM has been extensively proposed to replace SRAM as on-chip memory because of its advantages of high cell density and immunity to soft errors [12, 13, 11].

Prior research has shown that replacing SRAM with STT-RAM in various levels of on-chip caches can help improve performance, reduce power consumption, and achieve a better reliability. Yet the main focus of prior research of STT-RAM is on how to leverage the density, power consumption advantages and try to mitigate the well-known issue of asymmetric-access. The impact of process variations on STT-RAM caches and corresponding optimization techniques, however, are not well studied at the architectural level. In fact, since STT-RAM is normally employed as lower level caches (e.g. L2) with large capacity, process variations have a significant impact STT-RAM caches. Moreover, it becomes more complicated when the character of asymmetric read/write access is considered.

For the popular one-transistor-one-MTJ (Magnetic Tunneling Junction) based STT-RAM cell, various design parameters, such as MTJ shape, MgO thickness, access transistor's gate length/width, etc., can be affected by process variations. Consequently, the characteristics of STT-RAM cells in an STT-RAM cache can be quite different from each other. In the simple worst-case design employed for STT-RAM caches, a large overhead can be induced for access latency and power consumption due to process variations. Sun *et al* proposed a process-variation aware data management for non-uniform access STT-RAM caches [12]. In their work, the write latency variation is comparable to that consumed on data movement among different banks. Thus, the data migration policy is tailored to adapt the variable write latency caused by process variations. However, their approach is not feasible for uniform access STT-RAM cache, especially when the read-write asymmetric access is considered.

In this work, we apply the variable-latency access method to uniform access STT-RAM caches by introducing a novel variation-aware LRU policy. Moreover, we show that simply applying traditional variable-latency access method is inefficient due to the read/write asymmetry. First, we demonstrate a write-oriented data migration is preferred. Second, a block remapping is necessary to prevent some cache sets from being significantly affected by process variations. After using our techniques, the experimental results show that the performance can be improved by 13.8% and power consumption can be reduced by 14.1%.

The rest of the paper is organized as follows. In the next section, we will give a brief review of STT-RAM and the impact of

*This work is supported by the National Natural Science Foundation of China 61202072.

GLSVLSI'13, May 2–3, 2013, Paris, France.

process variations on STT-RAM cache cells as well as the basic idea of variable-latency access method. Then, we will present our variation-aware LRU policy that can achieve a variable-latency access STT-RAM cache in Section 3. In addition, we will discuss the impact of asymmetric access and show that the remapping technique is necessary. The evaluation results are shown in Section 4, followed by a conclusion.

2. PRELIMINARIES

In this section, we first give a review of STT-RAM technology and how to model the effect of process variations on STT-RAM. Then, we introduce the method of variable-latency access to caches under process variations.

2.1 STT-RAM Basics

Different from traditional CMOS based SRAM, STT-RAM uses MTJ devices to store the information. A MTJ has two ferromagnetic layers (FL) and one oxide barrier layer (BL). The resistance of MTJ, which is used to represent information stored, is decided by the relative magnetization directions (MDs) of the two FLs. When the MDs are parallel or anti-parallel, the MTJ is in low (bit '0') or high resistance state (bit '1'). R_h and R_l are usually used to denote the high and the low MTJ resistance, respectively. In a MTJ, the MD of one FL (reference layer) is pinned while the one of the other FL (free layer) can be flipped by applying a polarized write current though the MTJ. In the "1T1J" cell structure, the MTJ write current is supplied by a NMOS transistor [11, 10].

2.2 Modeling of Process Variations of STT-RAM

In the popular "1T1J" STT-RAM cell, the transistor is used to control the access to the MTJ while the MTJ is used to story data. Both access transistor and MTJ can be affected by process variations. Due to different access mechanisms in read and write operations, the process variations of these two components have different impact on read and write to STT-RAM cells. Note that in this subsection, the read/write latency is just for a single STT-RAM cell, which should be differentiated from the cache access latency for a whole STT-RAM cache. In addition, we find that variation of a STT-RAM cell is significantly larger than that of the peripheral circuitry. Thus, we focus on the variation of the STT-RAM cell in this work.

The read variation is mostly determined by variations on parameters of the access transistor, which include effective channel length/width, threshold voltage, oxide thickness, etc. Similar to prior research, variations of these parameters of transistors can be modeled with normal distributions using a well-know method [1, 9, 2]. Having distributions of these parameters, we can run Monte-Carlo based simulation to model the characters of all access transistors in a STT-RAM cache. Then, these numbers can be used to calculate the access latency and energy consumption of a read operation to cells in a STT-RAM cache. Obviously, these cells will demonstrate different access latency and energy, which are projected in the variable-latency access method in the next subsection.

Compared to read, the write operation of STT-RAM can have even lager variances of latency and energy consumption due to process variations of both the access transistor and MTJ. For the MTJ, the process variations have effects on several design parameters, such as MTJ shape, MgO thickness, and the localized fluctuation of magnetic anisotropy [11, 13]. All these parameters are related to the programming current, which is key factor for access latency and energy of a write operation. Wen *et al* and Sun *et al* investigated the typical distributions in various STT-RAM cell designs. A dual-exponential function about programming current is used to provide an excellent accuracy, which can be described in equation (1) [13]. Due to page limitation, the detailed relationship among those design parameters and characters of write operation is not presented in this paper. For those readers who are interested in the model details, please refer to Wen's and Sun' work for reference.

$$f(I) = \begin{cases} a_1 e^{b_1(I-\mu)} & I \le \mu \\ a_2 e^{b_2(\mu-I)} & I > \mu \end{cases} \quad (1)$$

2.3 Variable-latency Cache Access

Variable-latency access is an effective method to mitigate the impact of process variations on caches [3, 8, 11, 7]. In a cache using variable-latency access technique, the cache access latency is no longer a fixed value as in a traditional worst-case cache design. Instead, the access latency has a variance depending on the affect of process variations on STT-RAM cells. In other words, the access to some parts of the caches can be faster than the rest parts due to different effects of process variations on the STT-RAM cells. Since the access granularity is a cache line, the access latency of a cache line is decided by the slowest STT-RAM cell belonging to the cache line. It should be addressed that the energy consumption to a cache line also varies in a cache because the energy consumption is related to both access latency and current. Note that the definition of variable-latency access cache is different from that of a Non-uniform cache architecture (NUCA). For the later one, the non-uniform access latency is caused by the specific architecture design (e.g. network-on-chip) and also exists even when there are no any process variations.

Extensive research has been done on the variable-latency cache architecture. One variable-latency cache architecture uses a delay storage to record the delay time for each cache line [3]. In another variable-latency cache design, they enabled the variable access cache by modifying the function units and adding special queues to store the dependent instructions [8]. Similarly, a LA (Latency Aware)-LRU policy is put forward to mitigate the process variation [7]. They also uses the delay-storage architecture, and dynamically swapping data using the LA-LRU policy. The optimization targets in these works are traditional SRAM caches so that the asymmetric read/write access to STT-RAM caches are not considered. Sun *et al* proposed a variation aware data management for non-uniform cache architectures. They compensate write time variations via dynamic data migration [11]. Their data migration method, however, is not feasible for uniform caches. Considering the limitation of prior works, we propose a novel LRU policy with consideration of data migrations for asymmetric access STT-RAM caches, which will be introduced in next section.

3. VARIABLE-LATENCY ACCESS STT-RAM CACHE DESIGN

In this section, we first introduce the basic structure of our STT-RAM cache that can support variable-latency access. Then, we propose a novel LRU policy tailored to enable the data migrations in the variable-latency access cache. Finally, we will discuss the impact of cache line remapping and analyze design overhead.

3.1 Basic Structure

With the process variations, the caches lines in a STT-RAM cache demonstrate various access latency, in respect of read and write operations. For example, the distribution of write latency for a STT-RAM cache with 512 cache lines are shown in Fig-

ure 1. In our variable latency cache, we partition these cache lines into two parts, which are called the fast-access part (FAP) and the slow-access part (SAP). The partitioning is based on the distribution of the access latency to different cache lines. Since the STT-RAM cache has asymmetric read/write access latency, we can have different partitioning policies based on read latency, write latency, or a combination of them. The impact of partitioning policies is discussed later in subsection 3.3.

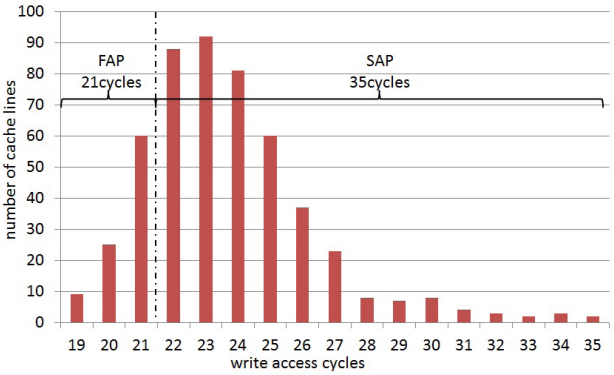

Figure 1: An example of distribution for cache write latency.

In the example of Figure 1, we partition the STT-RAM cache based on write latency and set 21 *clock cycles* as the threshold to differentiate SAP from FAP. In other words, for the cache lines that can be written within 21 *cycles*, they are grouped together as the FAP of the STT-RAM cache. The rest caches lines, which have the longer write latency than 21 *cycles*, are organized as the SAP of the STT-RAM. The discussion of selecting a proper threshold will be introduced in subsection 3.3. For each part of the STT-RAM cache, the write latency is fixed and determined by the longest access latency of the cache line in the part. For the same example in Figure 1, the write latency to SAP and FAP are 21 *cycles* and 35 *cycles*, respectively.

The organization of FAP and SAP for STT-RAM cache is different from traditional variable-access latency cache structure in prior work [3, 8, 7]. In prior work, the access latency is identified with a fine granularity, which is a single cache line. It means that the access to each cache line can be finished as soon as possible. For our variable-latency access STT-RAM cache, however, the cache is partitioned with a coarse granularity (e.g. SAP and FAP), and the access latency to a cache line is decided by the slowest cache line in the same part.

It is easy to find that the traditional fine granularity based structure can achieve a better performance than ours for the *static data allocation*. Static data allocation means that a data are placed in fixed positions (cache blocks) before they are evicted. On the contrary, a *dynamic data allocation* means that data can be moved from SAP to FAP if they are frequently accessed so that the performance is improved. It should be addressed that a dynamic data allocation is necessary for variable-latency access cache due to temporal data locality. Otherwise, if most frequently accessed data are always allocated in the cache lines with slow access latency, we may gain little benefits of using variable latency access cache.

When the dynamic data allocation is employed, our variable-latency access cache structure can outperform those in prior work for three reasons [3, 8, 7]. First, data migration is feasible to be realized when there are only SAP and FAP. On the contrary, in the traditional structure, there will be many different levels of access latency, which can significantly increase the design complexity since data migration can happen between any two cache lines with different access latency. Moreover, when the cache remapping technique is applied (introduced later in the paper), design complexity of a fine granularity policy is further significantly increased. In fact, the dynamic allocation policy is not introduced in most of traditional structures [3, 8]. Second, a coarse partitioning granularity can help reduce the overhead of tracing access latency of different cache lines. In traditional structure, a table is needed to record the access latency of each individual cache line. Such a table can induce non-trivial overhead, especially for STT-RAM cache which normally has the large capacity and significant variance of write latency. Third, the design complexity for variable-access latency control can be reduced. For example, as introduced in Bennaser's work [3], a buffer is needed to enable the variable-latency access. The design complexity of such a buffer and related control logic increases with the total number of different access latency. Since we only have two different access latency for SAP and FAP, the design can be really simple.

3.2 A Novel LRU Policy for Data Migration

As discussed in the last subsection, a dynamic data allocation is critical to exploit the benefits of using variable-latency access cache. In Sun's work, dynamic data allocation is enabled through data migration among different cache banks in the NUCA [11]. This data migration method, however, is not feasible for the uniform cache architecture. First, the data migration technique in NUCA can be enabled without the existence of process variations. Sometimes, the migration may even depends on the infrastructure of the NUCA (e.g. network-on-chip). When we apply the data migration in uniform cache, the extra design overhead must be considered. Second, the data locality is not considered in their approach, which may destroy the LRU replacement policy. Third, using the DPVA-NUVA-2 policy, both read and write operations can trigger the data migration. We will discuss in next subsection that such a combination of migration technique can harm the performance in uniform cache. Thus, we propose a new LRU policy, which can also enables the data migration at the same time.

The new LRU policy is illustrated in Figure 2 with a set of cache lines belonging to an eight-set-associative STT-RAM cache. As shown in the figure, the cache lines of the FAP are represented using blocks with shade. In this example, there are cache lines in FAP and five cache lines in SAP. In order to simplify the discussion, we assume that the positions of these cache lines in FAP are physically next to each other. The number labeled on top of each cache line is used to represented the position of the cache line in the LRU stack. The cache line with number 7 is the one that is just accessed. The cache line with number 0 is always evicted when a cache replacement happens.

Figure 2 (a)-(c) demonstrates the initial filling process for an empty cache set. As shown in Figure 2 (b), the data are always allocated in the FAP when there are slots in it. Figure 2 (c) shows the status when half of the cache set is filled. Note that the most recently loaded data are allocated in the SAP (the fifth cache line from left) at this time. Our experiments show that there is little enhancement if we carry out data swapping policy in the filling process. It is because high frequent data reuse is rare at the initialization stage. Moreover, these data may generate more swap operations when the data swap policy is applied. Considering the extra power consumption and latency for data swapping, we do not swap data in the filling process.

Figure 2 (d) and (e) shows the process when a cache replacement happens. When new data are loaded and the cache set is full, the cache line labeled with 0 is evicted. At the same time, we need to check whether the evicted cache line is in the SAP. If it is true, a data migration is triggered. The cache block

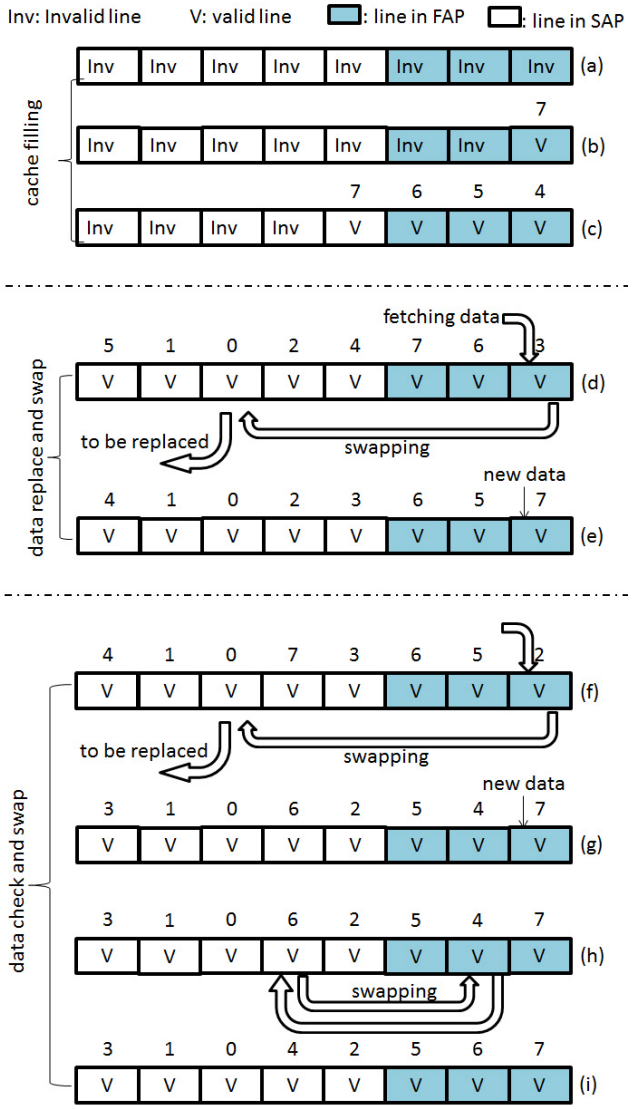

Figure 2: The illustration of LRU with the consideration of data migrations.

with smallest label number (e.g. the least recently used cache line in FAP), which is 3 in this example is, moved to the evicted cache line. Then, the new data is allocated in FAP (e.g. the last cache block in Figure 2 (e)). Such a replacement can promise that the most frequently loaded data are always allocated in the FAP.

As shown in Figure 2 (f), it is possible that the most recently accessed cache line resides in the SAP. Such scenarios usually happen during the initial filling process and also exist when there are some frequently accessed data that have a long reuse distance. In order to mitigate such problems, an extra check-and-swap step is adapted during the replacement. As shown in Figure 2 (g) to (i), after the replacement, if the cache labeled with number 6 is detected in the SAP, another data swapping is employed to move the SAP. With such a LRU policy, most data that are frequently accessed can be kept in the FAP to improve the performance.

Note that the data migration is only triggered during the cache replacement to avoid some useless data migration caused by cache hits, which are used in previous work [11]. In addition, we only demonstrates our data migration method with a perfect LRU policy to simplify the discussion. In fact, such a method can also be applied to some fake LRU polices in real processors. Due to page limitation, we will not discuss the details in this work.

3.3 Impact of Management Policies

The efficiency of variable-latency access is related to different cache management policies. In this subsection, we will discuss the impact of two important policies including partitioning policy and the cache line remapping.

3.3.1 Partitioning Policy

Obviously, the STT-RAM cache partitioning for SAP and FAP is one of the most important key factor for a variable-latency access cache design. First, it is critical to choose what type of access latency as the basis of latency partitioning. Second, the threshold for separating SAP from FAP also has an important impact on the efficiency of cache design.

As discussed in subsection 2.1, the variance of read and write latency is decided by different design parameters in STT-RAM. Thus, read latency and write latency have different distributions, although there is a correlation between them. To this end, we have to choose from read latency, write latency, or the combination of them for the cache partitioning. The comparison among these three partitioning methods is discussed as follows. In prior work for traditional SRAM caches, the read latency is normally used because the read latency is on critical path of execution time and the access latency is similar for read and write operations.

The case for STT-RAM caches, however, is quite different from that in SRAM caches. The reason is in three-folds: 1) The latency of a write operation is much longer than that of a read latency so that read operations may be blocked by write operations for a long period [10, 12]; 2) STT-RAM is usually used in lower cache levels like L2 caches. If a L1 cache is write-through based, the intensity of write operations can be significantly higher than that of the read operations in the L2 cache. Thus, the access time of write operations may dominate the total cache access time; 3) For each data migration, extra write operations are induced. Since the variance of read latency is also quite smaller than that of the write operation, a large number of subsequent accesses to the same data is needed to amortize the overhead of data migration. Later in section 4, we will show that the read-based partitioning cannot effectively improve the performance and may even degrade the performance sometime when a frequently accessed data is migrated to a cache line with short read latency but long write latency. Moreover, when the energy consumption is consider at the same time, a partitioning based on write operation is more attractive because cache lines in FAP also consume less energy in a write operation.

The third choice is to consider both read and write at the same time. This method, however, has a worse performance compared to the write-based partitioning. It is because a competition of same data may happen between a cache line with short read latency and a cache line with short write latency. Such a competition can result in a "ping-pong" style data migration and induce huge overhead. Consequently, only the pure read-based and write-based partitioning is considered in this work.

Another important issue in partitioning policy is the ratio between SAP and FAP. On one hand, if the SAP is too large, we cannot gain much benefits because most cache lines have the access latency of SAP, which is equal to that of the worst case in all cache lines. In addition, there will be intensive competition for the cache lines in the FAP, which may induce more data migrations. On the other hand, if the FAP is too large, more

cache lines with long access latency will be included in the FAP so that the access latency to FAP will be increased. Thus, the benefits of data migration will also be reduced because the gap of access latency between SAP and FAP is shrunk. The optimized ratio between SAP and FAP is related to the distribution of process variations on STT-RAM cells and the data access pattern. For the STT-RAM cache and workloads in section 4, the results show that the optimized ratio is in the range of 4 to 7. Thus, the threshold 21 *cycles* in subsection 3.1 is plausible, for the ratio between SAP and FAP is nearly 4.

3.3.2 Cache Line Remapping

Due to the correlation of process variations, the physical position allocation of cache lines in FAP or SAP can be quite non-uniform. It means that the ratio between SAP and FAP can vary a lot all through the whole cache. Consequently, the efficiency of using variable-latency access cache will be decreased. In order to mitigate this problem, a cache remapping method introduced in prior research need to be adopted to achieve a uniform distribution of ratio between SAP and FAP [6]. Figure 3 shows an example of such a remapping method. Note that the number labeled on the cache lines represent the index of cache set that those cache lines belong to. The cache lines with shade color represent the cache lines in FAP. We can find that, after remapping, the ratio between SAP and FAP become uniform among different cache sets.

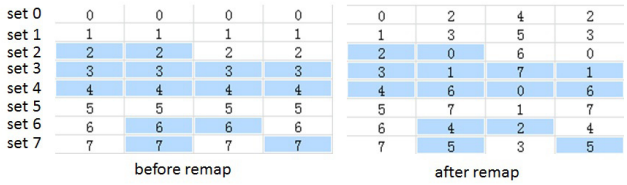

Figure 3: An example of cache line remapping.

4. EVALUATION

In this section, we first evaluate our variable-latency access cache with different management policies. Then, we compare our optimization design to the worst case design and an approach from prior work [3], in respect of performance and energy consumption.

4.1 Experiment Setup

For system level simulation, we use SIMICS simulator to evaluate performance. It is configured to model an eight-core processor. Each core is Ultra-SPARC-like with a 2GHz frequency. There are three levels of caches. The IL1/DL1 caches are SRAM based and the capacities are set to 64KB/64KB. The L2 cache is a 16-way 4MB STT-RAM cache. The simulator captures data addresses from all loads, stores, and prefetch operations. We use the information to calculate the memory access intensity, and use that to compute the energy consumption of the cache hierarchy. Our workloads are sets of multi-threaded benchmarks from PARSEC.

In order to estimate the access latency and energy consumption of different caches, we extend the widely used tool NVSim [5] to support STT-RAM technology with process variations considered. The parameters used in this work are listed in Table 1 and Table 2. Figure 4 shows the probability of programming current in write operations, which are fitted based on Monte-Carlo simulation for different STT-RAM cells. Note that in our simulations, extra latency and power consumption are included for the data migration. For each data migration, we add extra write access latency (30 *cycles*) and corresponding power consumption.

Table 1: Device Parameters

Device	Parameters	Mean
Transistor	Channel Length L	$45nm$
	Channel Width W	design dependent
	Threshold Voltage V_{th}	$0.466V$
MTJ	MgO Thickness τ	$2.2nm$
	Cross Section A	$45 \times 90nm^2$

Table 2: 3σ parameters of the transistor.

$Width$	$Length$	V_{th}
$-15\%\mu$	$+15\%\mu$	$+15\%\mu$

4.2 Experimental Results

First, we compare the performance of different benchmarks with variable-latency access caches using different partitioning policies in Figure 5. The y-axis represent the normalized execution time of different benchmarks. The first set of results labeled with *Worst* are for the baseline case, in which worst case design method is used in STT-RAM cache design. The second set of results labeled with *VAL-R* are for the cache using variable-latency access caches with a read-based partitioning policy. Similarly, the third set of results labeled with *VAL-W* are for the cache using variable-latency access caches with a write-based partitioning policy. The results show that the read-based partitioning cannot help improve the performance even with data migration supported. It is because the variance of a read latency is not significant and read operations only contribute a small fraction to the total access numbers. On the contrary, using write-based partitioning can significantly improve performance. Thus, the write-based partitioning should be used for variable-latency access STT-RAM caches. For the rest of the section, all variable-latency access caches will use write-based partitioning.

In order to study the impact of ratio between SAP and FAP, we assume a perfect allocations of cache blocks in SAP and FAP. In this perfect allocation, each cache set has an identical number of cache lines belonging to FAP. Then, we vary this number from 1 to 8 and compare the normalized reduction of execution time for a typical workload (*bodytrack*) in Figure 6. Note that the results for other workloads show the similar trend and we hide them due to page limitation. The results show that the best performance can be achieved for most benchmarks when the cache line numbers of FAP are in the range of 2 to 4. Consequently, we estimate an sub-optimized ratio between SAP and FAP as $(16-3)/3$, or 4.3.

The impact of remapping on performance is shown in Figure 7. *VAL-NoRemap* represents the normalized execution time without using cache remapping. *VAL-Remap* represents the normalized execution time for the case of using cache remapping. the ratio between SAP and FAP of the whole cache is still set to about 4.3 for both cases. It can be found that the performance can be significantly improved with remap method. The reason is that, without cache remapping, the ratio between SAP and FAP in many caches are quite different from that of the whole cache. Thus, the benefits of using variable-latency access caches are decreased significantly.

We compare our method to a prior approach using the same

$$f(I) = \begin{cases} 0.0802e^{0.1604(I-84.77)} & I \leq 84.77 \\ 0.0802e^{0.1604(84.77-I)} & I > 84.77 \end{cases}$$

Figure 4: Fitting equation of programming current

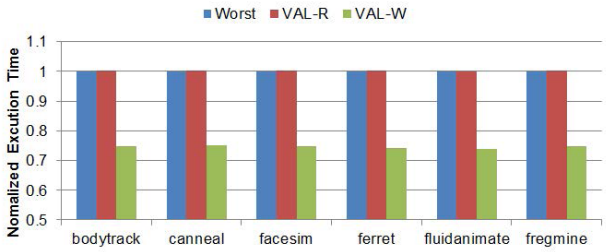

Figure 5: Comparison of performance with different partitioning policies.

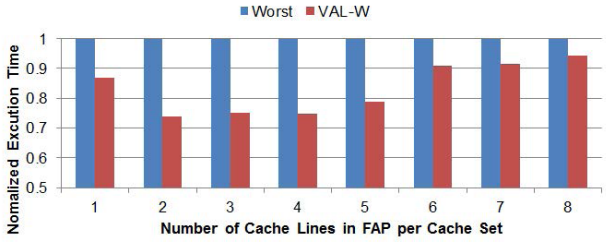

Figure 6: Comparison of performance with different ratios between SAP and FAP.

experimental setup, in respect of performance and energy consumption [3]. The results of normalized execution time are shown in Figure 8. The label *VAL-Static* represents the set of results using the variable-latency access cache design in Bennaser's work [3]. On average, the performance can be improved by about 13.8%, compared to the prior approach [3]. Also, we compare the results of normalized energy consumption in Figure 9. We can draw a similar conclusion for the results of energy consumption. It is because the write energy consumption of a STT-RAM cell is roughly proportional to the product of programming latency and current. Since the write energy dominates in the total energy consumption of STT-RAM caches, a similar conclusion can be drawn in the comparison. On average, the energy consumption can be reduced by about 14.1%. Note that the VAL-Static architecture in Bennaser's work uses fine granularity, but without cache line remapping. The result shows that our coarse granularity policy with cache line remapping and dynamic data migration works better than their approach. Theoretically speaking, fine granularity with remapping and data migration can perform better. However, as discussed in subsection 3.1, the design complexity of such a fine-granularity architecture is significantly higher.

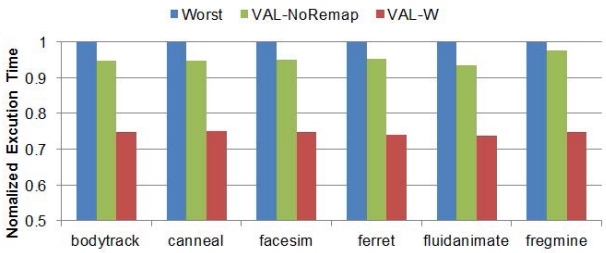

Figure 7: Comparison of performance for cases using and without using remapping.

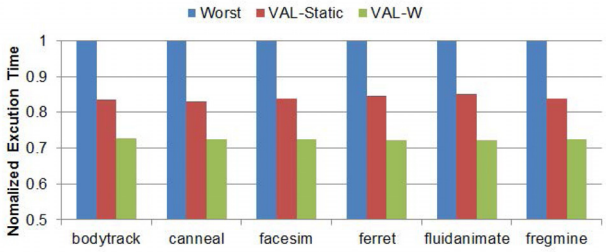

Figure 8: Comparison of performance for different variable-latency access methods.

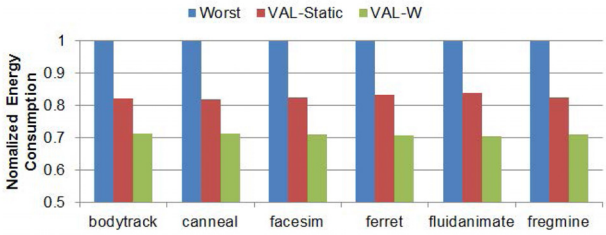

Figure 9: Comparison of energy for different variable-latency access methods.

5. REFERENCES

[1] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, pages 900 – 907, nov. 2003.

[2] A. Agarwal, D. Blaauw, V. Zolotov, S. Sundareswaran, M. Zhao, K. Gala, and R. Panda. Statistical delay computation considering spatial correlations. In *Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific*, pages 271 – 276, jan. 2003.

[3] M. Bennaser, Y. Guo, and C. Moritz. Data memory subsystem resilient to process variations. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(12):1631 –1638, dec. 2008.

[4] Y. G. Choi, S. Yoo, S. Lee, and J. H. Ahn. Matching cache access behavior and bit error pattern for high performance low vcc l1 cache. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 978 –983, june 2011.

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

[6] M. Hussain and M. Mutyam. Block remap with turnoff: A variation-tolerant cache design technique. In *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*, pages 783 –788, march 2008.

[7] A. Jain, A. Shrivastava, and C. Chakrabarti. La-lru: A latency-aware replacement policy for variation tolerant caches. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, pages 298 –303, jan. 2011.

[8] S. Ozdemir, A. Mallik, J. C. Ku, G. Memik, and Y. Ismail. Variable latency caches for nanoscale processor. In *Supercomputing, 2007. SC '07. Proceedings of the 2007 ACM/IEEE Conference on*, pages 1 –10, nov. 2007.

[9] G. Sun, C. Xu, and Y. Xie. Modeling and design exploration of fbdram as on-chip memory. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 1507 –1512, march 2012.

[10] G. Sun, Y. Zhang, Y. Wang, and Y. Chen. Improving energy efficiency of write-asymmetric memories by log style write. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ISLPED '12, pages 173–178, New York, NY, USA, 2012. ACM.

[11] Z. Sun, X. Bi, and H. Li. Process variation aware data management for stt-ram cache design. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ISLPED '12, pages 179–184, New York, NY, USA, 2012. ACM.

[12] Z. Sun, X. Bi, H. H. Li, W.-F. Wong, Z.-L. Ong, X. Zhu, and W. Wu. Multi retention level stt-ram cache designs with a dynamic refresh scheme. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44 '11, pages 329–338, New York, NY, USA, 2011. ACM.

[13] W. Wen, Y. Zhang, Y. Chen, Y. Wang, and Y. Xie. Ps3-ram: A fast portable and scalable statistical stt-ram reliability analysis method. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 1187 –1192, june 2012.

[14] M. Yamaoka and H. Onodera. A detailed vth-variation analysis for sub-100-nm embedded sram design. In *SOC Conference, 2006 IEEE International*, pages 315 –318, sept. 2006.