Hi-fi Playback: Tolerating Position Errors in Shift Operations of Racetrack Memory

Chao Zhang[1], Guangyu Sun[1,2], Xian Zhang[1], Weiqi Zhang[1], Weisheng Zhao[3], Tao Wang[1,2], Yun Liang[1,2], Yongpan Liu[4], Yu Wang[4], and Jiwu Shu[5]

[1]*Center for Energy-efficient Computing and Applications, Peking University, Beijing, 100871, China*

[2]*Collaborative Innovation Center of High Performance Computing, NUDT, Changsha 410073, China*

[3]*Spintronics Interdisciplinary Center, Beihang University, 100191, China*

[4]*Department of Electrical Engineering, Tsinghua University, 100084, China*

[5]*Department of Computer Science and Technology, Tsinghua University, 100084, China*

{zhang.chao, gsun, zhang.xian, zhangweiqi, wangtao, ericlyun} @pku.edu.cn

weisheng.zhao@u-psud.fr, ypliu26@gmail.com, {yu-wang, shujw} @tsinghua.edu.cn

Abstract

*Racetrack memory is an emerging non-volatile memory based on spintronic domain wall technology. It can achieve ultra-high storage density. Also, its read/write speed is comparable to that of SRAM. Due to the tape-like structure of its storage cell, a "shift" operation is introduced to access racetrack memory. Thus, prior research mainly focused on minimizing shift latency/energy of racetrack memory while leveraging its ultra-high storage density. Yet the reliability issue of a shift operation, however, is not well addressed. In fact, racetrack memory suffers from unsuccessful shift due to domain misalignment. Such a problem is called "***position error***" in this work. It can significantly reduce mean-time-to-failure (MTTF) of racetrack memory to an intolerable level. Even worse, conventional error correction codes (ECCs), which are designed for "bit errors", cannot protect racetrack memory from the position errors.*

In this work, we investigate the position error model of a shift operation and categorize position errors into two types: "stop-in-middle" error and "out-of-step" error. To eliminate the stop-in-middle error, we propose a technique called sub-threshold shift (STS) to perform a more reliable shift in two stages. To detect and recover the out-of-step error, a protection mechanism called position error correction code (p-ECC) is proposed. We first describe how to design a p-ECC for different protection strength and analyze corresponding design overhead. Then, we further propose how to reduce area cost of p-ECC by leveraging the "overhead region" in a racetrack memory stripe. With these protection mechanisms, we introduce a position-error-aware shift architecture. Experimental results demonstrate that, after using our techniques, the overall MTTF of racetrack memory is improved from $1.33\mu s$ *to more than 69 years, with only* 0.2% *performance degradation. Trade-off among reliability, area, performance, and energy is also explored with comprehensive discussion.*

ISCA '15, June 13 - 17, 2015, Portland, OR, USA

ACM 978-1-4503-3402-0/15/06 $15.00

http://dx.doi.org/10.1145/2749469.2750388

1. Introduction

Rapid advances of computing systems and applications make demands of enlarging capacity of on-chip memory. On the one hand, as the number of cores in both CMPs and GPUs keeps increasing, more data are expected to be cached on chip to leverage their locality. On the other hand, the prosperity of high-throughput computing applications also requires improvement of on-chip memory hierarchy to bridge an increasing bandwidth gap between processing elements and off-chip memory. However, the traditional SRAM technology cannot fully satisfy the demand due to its low scalability, high leakage power consumption, and vulnerability to soft errors. Consequently, various emerging technologies, such as STT-RAM, RRAM, and FBDRAM, have been extensively researched as potential alternatives of SRAM [45, 24, 35, 20, 37, 38, 40].

Racetrack memory, which is also known as domain wall memory (DMW), has attracted great attention of researchers because of its ultra-high storage density. Racetrack memory is a type of non-volatile memory based on spintronic technology. Compared to the other spintronic memory technologies (e.g. STT-RAM), racetrack memory provides even higher storage density by integrating multiple bits (domains) in a tape-like nanowire [26]. Previous research has demonstrated that its storage density is up to $10\times$ higher than that of STT-RAM [46]. To access all domains on the same nanowire, one or several access ports are uniformly distributed along the nanowire and shared by domains. For those domains aligned to access ports, data in them can be read similar to STT-RAM cells. Thus, when these bits are accessed, racetrack memory can achieve high performance comparable to STT-RAM [46, 44, 43], which makes it a promising candidate for on-chip memory design. However, to access other bits on the nanowire, the "shift" operations are required to move those bits to the nearest access ports.

Obviously, a shift operation induces extra timing and energy overhead. Thus, prior research on racetrack memory mainly focused on mitigating the shift overhead while leveraging its ultra-high storage density. For example, "block swapping" and "head management" techniques are proposed to reduce shift intensity when racetrack memory is employed as caches in generic processors [39, 44]. With these techniques, a racetrack memory based cache could achieve about 83% area reduction, 25% performance improvement, and 62% energy reduction compared with STT-RAM based counterpart. Venkatesan *et al.* proposed a racetrack memory based cache architecture for GPGPU with a shift aware promotion buffer. It could improve

GPGPU performance by 12% and reduce energy consumption by 70% over SRAM based cache hierarchy [43]. Without doubt, previous work has demonstrated that we can benefit from ultra-high storage density of racetrack memory even with extra shift latency and energy.

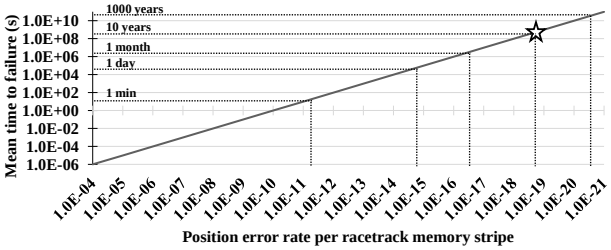

Figure 1: MTTF of a racetrack memory LLC [43] against different error rates.

Though several approaches have been proposed to mitigate its timing and energy overhead, the reliability issue of a shift operation is not well addressed yet. Simply speaking, there lack mechanisms to ensure that *domains are correctly shifted to be aligned with access ports.* Different from conventional "bit errors", an unreliable shift may result in a new type of error called "**position error**". Thus, although data stored in domains are unchanged, incorrect bits may be read from racetrack memory due to position errors. Also, a write operation may fail to update data because of position errors. Overall, position errors can have significant impact on the reliability of a racetrack memory design. As shown in Figure 1, for a racetrack memory cache in previous work [43], the position error rate needs to be at least lower than 10^{-19} to satisfy a requirement of 10-year mean-time-to-failure (MTTF) [25]. Unfortunately, a typical position error rate is in the range of $10^{-4} \sim 10^{-5}$ for different shift operations (more details in Section 3).

Even worse, conventional error correction codes (ECCs) proposed for transient bit errors cannot detect and correct such position errors efficiently. Since bit errors and position errors can be considered as orthogonal to each other, we need dedicated mechanisms for detection and correction of position errors, together with those for transient bit errors. Because of the tape-like structure of a racetrack memory cell, the case is analogous to that of a traditional cassette tape. When designing a hi-fi cassette deck, we mitigate noises caused by magnetic head sensing flaw (bit errors) and imperfect pitch caused by tape speed fluctuation (position errors), independently.

To mitigate the problem of position errors, we first investigate the position error model of a shift operation. Then, we propose techniques and architecture modification to tolerate position errors. The main contribution of this work can be summarized as follows:

- Based on a quantitative error model, we categorize position errors into two types, which are called "stop-in-middle" error and "out-of-step" error.
- To eliminate the stop-in-middle error, we propose a technique called sub-threshold shift (STS) to complete a more reliable shift in two stages.
- To detect and recover the out-of-step error, a protection mechanism based on position error correction code (p-ECC) is further proposed to provide different protection strength. With an analysis of p-ECC design cost, we trade latency/energy for storage density by leveraging domains in "overhead region" to store p-ECCs.
- Based on these techniques, we present a position-error-aware shift architecture to meet the reliability requirement.
- Experimental results demonstrate that, after using these techniques, a practical racetrack memory design with sufficient reliability can be achieved with moderate design cost.

The rest of this paper is organized as follows. In Section 2, we introduce background of racetrack memory technology. In Section 3, a detailed position error model is presented. In addition, we argue that conventional ECCs for transient bit errors cannot handle position errors efficiently. Then, two techniques called STS and p-ECC are proposed in Section 4. Based on them, we introduce our position-error-aware shift architecture in 5. Experimental results are presented and discussed in Section 6, followed by related work and conclusions.

2. Background

In this section, we first introduce the basics of racetrack memory, including the cell structure, read/write operations, and the shift operation. We then briefly review metrics and requirements for memory reliability.

2.1. Basics of Racetrack Memory

A racetrack memory cell is composed of a tape-like stripe and several access transistors. A typical cell structure [43] is illustrated in Figure 2 (a). The racetrack memory stripe is made of magnetic material. It contains a lot of domains (white blocks) isolated by domain walls (dark bricks). The magnetization direction (arrows) of a domain is programmed to store either bit '1' or bit '0'. Several transistors are connected to the stripe to perform read, write, and shift operations, respectively. They are called read access port.

Read Operation. Similar to STT-RAM, bit value in a domain is sensed out according to its magnetization direction. As shown in Figure 2 (a), a read-only port is attached to a reference domain with a pinned magnetization direction. Together with the domain aligned under it, the reference domain forms a sandwich structure magnetic tunneling junction (MTJ). MTJ has a low resistance (bit '0') when these two domains have parallel magnetization directions and has a high resistance (bit '1') when they have opposite directions. As shown in Figure 2 (a), the read port is controlled by a transistor connected to read word line (RWL). Since the read port can only read the bit in the aligned domain, a shift operation is needed before it reads other domains.

Shift Operation. Shift operations are based on a phenomenon called spin-momentum transfer caused by shift current, which is supplied by two transistors attached to both ends of the racetrack memory stripe. The driving current density needs to reach a threshold to enable movement of these domain walls. Because the spin-momentum is transferred from electrons, domain walls move opposite to the direction of current along the racetrack memory stripe in most magnetic materials.

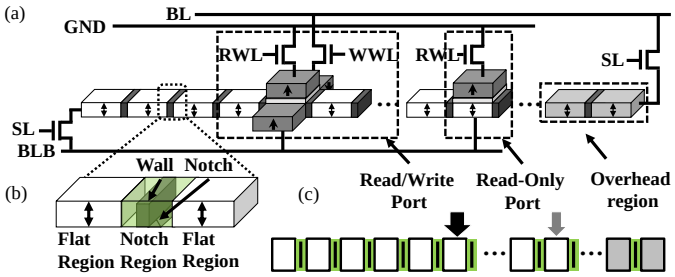

Figure 2: A racetrack memory cell: (a) physical layout, (b) details of notch, (c) architecture abstract.

Note that all domain walls move in the same direction with the same speed [**?**, 47].

In order to align domains in the racetrack memory stripe with the access ports after a shift operation, the stripe is etched with successive notches to pin the domain walls, as illustrated in Figure 2 (b). A notch region, which is affected by the notch, has much larger resistance to the shift speed compared with the flat region. Thus, a notch region works like a speed bump to pin a domain wall after the driving current is disabled. Ideally, domain walls should be pinned in proper notch regions to complete a successful shift operation.

In this work, the shift distance is denoted by "steps". For example, if all domain walls are shifted to right and stop in their neighbor notch regions, we refer it as a 1-step shift. Obviously, the maximum shift distance is determined by stripe length, domain length, and the number of access ports. In order to avoid data loss due to shift operations, extra domains are needed. When the maximum shift distance is *m*-step, *m* extra domains are added. These extra domains form an **overhead region**, which is also shown in Figure 2 (a) and (c).

Shift-based Write Operation. A write operation can be completed by a shift operation. Figure 2 (a) also shows a read/write access port. Compared to a read-only port, a read/write port requires one more transistor and two more reference domains with opposite pinned magnetization directions. Thus, we only need to control this extra transistor so that a proper bit is shifted from a reference domain into the domain to be updated. Note that a conventional write like STT-RAM is also feasible for racetrack memory. But it requires a larger transistor, due to larger current for write.

Number of Access Ports. Normally, access ports are distributed along the racetrack strip uniformly. For a fixed length racetrack memory stripe, the maximum shift distance is reduced with more access ports. However, adding extra ports may induce area overhead. More details can be found in the next section.

The architecture level abstract view of a racetrack memory cell can be is illustrated in Figure 2(c). In the rest of this paper, we will use it for discussion.

2.2. Reliability of Memory

Memory errors may result in silent data corruption (SDC), where a system generates erroneous outputs without attention, or a detected unrecoverable errors (DUE) [25]. Both of them reflect the reliability of the memory. They can be expressed using failures in time (FIT), number of failures in a billion (10^9) hours, or mean time to failure (MTTF), which is inversely related to FIT (11,415 FIT is equivalent to 10-year MTTF) [25]. IBM targets 1000-year SDC and 10-year DUE for its power4 systems [8]. In this work, we use these two numbers as a reference goal of the reliable racetrack memory design.

Error correction codes (ECC) such as parity check and extended hamming code (a.k.a. SECDED) are widely used in cache system of modern processors to tolerate memory errors. Parity check and ECC rely on the value of bits in their coding block to detect and correct the bit error [29]. Some processors employ parity check protection for L1 cache [1, 2, 22]. And the majority of processors use ECC to protect the last level cache, such as AMD K8 [1], UltraSPARC IV[2], Itanium 2 [22], Power4 [7], Alpha [19], and Intel Pentium 4 [3].

3. Position Error

In this section, the position error is modeled. In addition, we explain why conventional ECC cannot detect and correct position errors efficiently.

3.1. Error Modeling

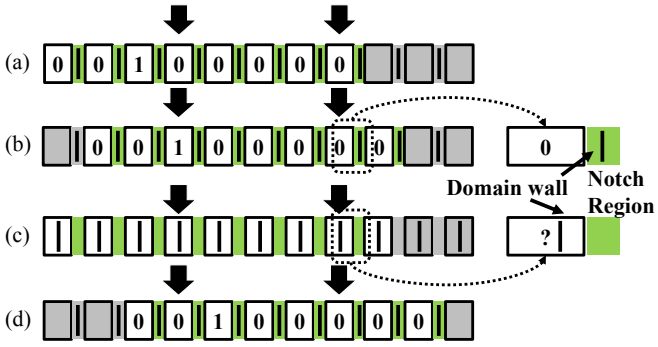

Figure 3: Illustration of position errors: (a) initial state before shifting, (b) a shift operation without errors, (c) a shift operation with a "stop-in-middle" error, (d) a shift operation with an "out-of-step" error.

We first define two types of position errors that may happen in a shift operation. In Figure 3 (a), all domain walls are in their initial state before being shifted. In order to access the domain containing bit '1', we need to shift all domain walls one step to the right. After a correct shift operation without any errors, the domains are in a state shown in Figure 3 (b). All domain walls are pinned in the notch regions. The domain containing bit '1' is ready to be read with the access port.

In Figure 3 (c), a position error happens because the domain walls are not pinned into notch regions. Such a position error is called "**stop-in-middle**" error in this work. In other words, the domain is not aligned properly to the access port. Thus, the value read out is uncertain, as highlighted in Figure 3 (c). A different type of error is illustrated in Figure 3 (d). In this scenario, all domain walls are stopped in notch regions after the shift operation. However, domains are over-shifted by one step. As shown in the figure, the domain containing bit '1' has passed the access port and an incorrect domain is accessed. Such a position error is called "**out-of-step**" error. Since the domain is over-shifted for one step, it is also defined as "+1

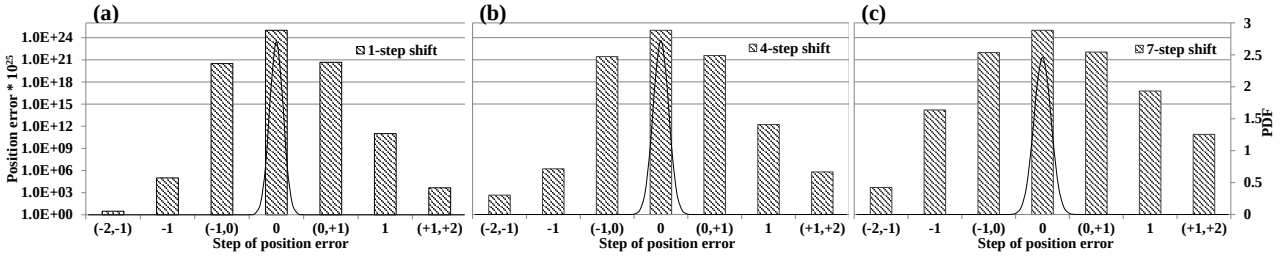

Figure 4: Probability distribution and its density function of position errors.

step" error. Similarly, a **"$\pm n$ step"** error means that domain walls are over-shifted/under-shifted for n steps.

$$\begin{aligned}(1+\alpha^2)\tfrac{dq}{dt} &= \tfrac{1}{2}\gamma\Delta(H_K\sin 2\psi - \pi H_T) + \alpha\gamma\Delta(H_A - \tfrac{Vq}{M_S d}) + (1+\alpha\beta)u \\ (1+\alpha^2)\tfrac{d\psi}{dt} &= -\tfrac{1}{2}\alpha\gamma(H_K\sin 2\psi - \pi H_T) + \gamma(H_A - \tfrac{Vq}{M_S d}) - \tfrac{\beta-\alpha}{\Delta}u\end{aligned} \tag{1}$$

The behavior of domains in a shift operation can be quantitatively modelled by Equation (1). It describes a one dimension model for the domain wall motion in in-plane racetrack memory [14]. Term q and ψ represent the position and the tilt angle of a domain wall, while $\frac{dq}{dt}$ and $\frac{d\psi}{dt}$ represent their time derivatives, respectively. Parameters α, β, and γ are the Gilbert damping constant, non-adiabatic spin transfer torque term, and the gyromagnetic ratio, respectively. The u represents the spin transfer torque, which is proportional to shift current density. H_K is the anisotropy field. H_T and H_A are the applied transverse field and lengthwise magnetic field. Both are zero in practical application. M_s is the saturation magnetization.

The critical parameters used in the equation are introduced in Table 1, which includes domain wall width (Δ), pinning potential depth (V), pinning potential width (d), and flat region width (L). Shift current density J is calculated as a proper current density to drive domain walls. We select J as $2J_0$ to minimize the error rate. The threshold shift current density J_0 is the minimum current density to move domain walls out of notch region [14]. If J is too small, the rate of under-shifted position errors increases. On the contrary, if it is too large, the rate of over-shifted position errors increases.

Based on the Equation (1), the time that a domain wall passes a notch region and a flat region can be approximately expressed as:

$$\begin{cases} T_{flat} &= \frac{\alpha L}{(2\alpha-\beta)u} \\ T_{notch} &= \tau ln(1+\frac{d}{\delta l}), \end{cases} \tag{2}$$

where $\tau = \frac{\alpha M_s d}{V\Delta\gamma}$, $\delta l = \frac{udM_s(2\alpha-\beta)}{V\Delta\gamma} - L - d$. Thus, the theoretical time to shift a domain wall for N steps can be expressed as $T_N = N \times (T_{notch} + T_{rest})$. However, in real scenarios, variations of parameters in Table 1 may cause unintended movement and result in position errors. Normally, variations of these parameters come from two sources [23, 9]: (1) process variations and (2) environmental variations. Their variations are listed in Table 1.

Considering all these effects, we calculate the possibility of position errors based on 10^9 times Monte-Carlo simulation and its fitting curve, which is similar to the method used in [34]. In Figure 4, we present a probability distribution (PDF) of position errors for a single shift operation. Note that Figure 4 (a) (b) (c) are results for 1-step, 4-step and 7-step shift distance, respectively. The results for bar $x = \pm i$ are error rates for $\pm i$ out-of-step errors. The other bars represent error probability for "stop-in-middle" errors. For example, the bar for $x = (+1,+2)$ means that domain walls are over-shifted by one step and fail to stop in the notch region. The result for origin point $(x = 0)$ means the probability of a correct shift. Asymmetry of +/-k step-errors is because typical driving current is higher than threshold to facilitate shifting. Two important observations can be concluded from these results.

Table 1: Simulation parameters used in the model [14, 16]

Parameters	Mean	Standard Deviation
Domain wall width	$\bar{\Delta} = 5.00nm$	$\sigma_\Delta = 0.02\bar{\Delta}$
Pinning potential depth	$\bar{V} = 1.20J/dm^3$	$\sigma_V = 0.02\bar{V}$
Pinning potential width	$\bar{d} = 45nm$	$\sigma_d = 0.05\bar{d}$
Flat region width	$\bar{L} = 150nm$	$\sigma_L = 0.05\bar{d}$
Shift current density	$J = 1.24A/um^2$	by calculation

- Error rates increase with a longer shift distance.
- Error rates decrease sharply when $x > 1$. It means that the ± 1 out-of-step errors, $(-1,0)$, and $(0,+1)$ "stop-in-middle" errors are the critical problem to be handled.

Our model uses a conservative estimation of process variations; the error rate can be even higher in real cases. In addition, we focus on in-plane material in this work. Using perpendicular material can reduce the size of domain but may increase error rate at the same time [48].

3.2. Position Error vs Bit-Error ECC

We explain why conventional ECC designed for bit errors cannot work efficiently for position errors in this part. To be simplified, we note conventional ECC designed for bit errors as "b-ECC" in this work. We take single-error correction and double-error detection (SECDED) b-ECC protecting a 64-Byte data in on-chip memories as an example.

First, b-ECC is designed to detect unintended changes of data bits. When a position error happens, if the misaligned bit has same value as correct bit, ECC cannot detect it timely. For one case where **multiple bits** of one 64-Byte data are stored in one racetrack memory stripe, if a ± 1-step position error happens, all the bits are shifted ± 1-step. Thus b-ECC actually check another data instead of this one, and cannot detect the error. For the other case where only **one bit** of the data is located on each stripe (i.e. 512 stripes are need to store the data), b-ECC cannot detect the error unless the bit read out is different from the correct one. They both result in accumulation of multiple position errors on different stripes and fail b-ECC.

Second, even if ECC detects that misaligned bit is incorrect, it cannot decide the direction and steps of shit errors for certain. Thus, we have to refresh all data in stripes to refill the correct

data (if possible). However, such a refresh method induces thousands of extra shift operations, since all bits in the 512 racetrack memory stripes have to be read out. Unfortunately, the possibility that a second position error happens during the correction process is outstanding. For an 8-bit racetrack memory stripe, the possibility is about 0.17. And the MTTF after using "b-ECC" is 20*ms*, which is far from the reliability goal. It means that b-ECC may fail to work during the correction process. Thus, we propose a dedicated protection mechanism for position errors.

4. Position Error Correction Mechanisms

In this section, we first propose a technique called STS to eliminate almost all stop-in-middle errors. Then, we further propose p-ECC to detect and correct out-of-step (or step) errors.

4.1. STS: Sub-threshold Shift

We can find from Equation (1) that domain wall can only move in the flat region but stop in the notch region, when the driving current density J is reduced under J_0. In fact, when the driving current is close to J_0, it takes too long time for a domain wall to move out a notch region. In this work, such type of shift operation is called "sub-threshold shift" (STS).

Based on this observation, we propose a two-stage shift operation, which is described as follows:

- **Stage-1** For a N-step distance shift operation, a pulse of high driving current density ($2J_0$) is applied. The pulse width is calculated from Equation (2) as an ideal shift.
- **Stage-2** After stage-1, an extra pulse of driving current is applied to make a sub-threshold shift. The pulse width is set as 1*ns*.

According to Equation (1)(2), a driving pulse of 0.8*ns* is long enough to ensure that domain walls are moved out of flat regions and enter notch regions. Considering process variations, the pulse width is set to 1*ns*. Note that domain walls may move out notch regions if some notches are not etched successfully during fabrication. In fact, such rare malfunction racetrack stripes can be disabled during chip testing techniques, which is beyond the scope of this paper.

After applying STS technique, the stop-in-middle errors are almost eliminated. Compared to results in Figure 4, the probability of out-of-step errors, however, is increased significantly. The probability of out-of-step position errors is listed in Table 2. It is easy to understand that some stop-in-middle errors are transferred into out-of-step errors. For example, for those stop-in-middle errors happen in the flat region represented with $(+1,+2)$, they are turned into $+2$-step error after using STS technique. Note that a negative driving current can also be applied to eliminate stop-in-middle errors. For the same example, the only difference is that those stop-in-middle errors are turned into $+1$-step errors. In order to simplify discussion, we assume that a positive STS is applied in the rest of this work.

STS induces fixed overhead in a single shift operation (any distance), which is the extra cycles used in stage-2. According to the model, the latency for stage-2 is 1*ns*. The latency for stage-1 can be estimated as 0.4*ns* for value used in Table 1. Thus, the latency to shift N steps a time by STS is $\lceil 0.4/0.5N \rceil + 2$ cycles, if clock frequency is 2GHz. It needs 3 cycles to shift 1 step, and 8 cycles for a 7-step shift. Thus, a rule of thumb is that *larger steps are preferred to amortize the overhead induced by STS.* The hardware modification to shift controller is introduced in Section 5.

Table 2: Probability of out-of-step position error

Distance	$\pm k$ Step Error Rate		
	$k=1$	$k=2$	$k\geq 3$
1	4.55×10^{-5}	1.37×10^{-21}	too small
2	9.95×10^{-5}	1.19×10^{-20}	too small
3	2.07×10^{-4}	5.59×10^{-20}	too small
4	3.76×10^{-4}	1.80×10^{-19}	too small
5	5.94×10^{-4}	4.47×10^{-19}	too small
6	8.43×10^{-4}	9.96×10^{-18}	too small
7	1.10×10^{-3}	7.57×10^{-15}	too small

4.2. p-ECC Design

After using STS technique, the out-of-step errors become the main obstacle of achieving reliable shift operations. In order to mitigate this problem, we further propose position error correction codes (p-ECC). In this section, we use a racetrack memory stripe with eight data domains as an example. We first introduce basic idea of p-ECC with a simple case to detect a 1-step error. Then, we present a practical p-ECC design that can detect 2-step errors and correct 1-step errors. At last, we propose to leverage the overhead region in the racetrack memory stripe to reduce p-ECC's area cost.

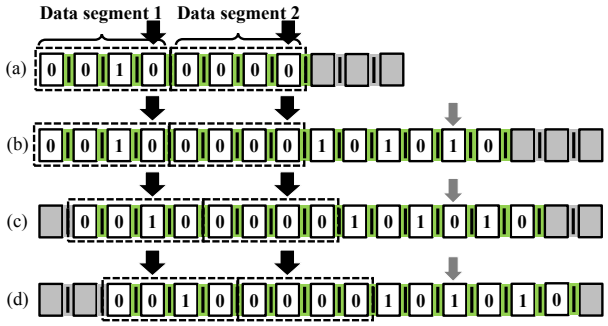

Figure 5: SED p-ECC. (a) original racetrack memory stripe, (b) adding p-ECC codes, (c) illustration of a correct shift, (d) illustration of detecting 1-step error.

4.2.1. Single Step Error Detection (SED)

As shown in Figure 5 (a), a racetrack memory stripe without any protection is used as a baseline. It has eight data domains and an overhead region of three domains. The total length of this racetrack memory stripe is 11 bits. We introduce a new term called "**data segment**". It is defined as the group of domains that can be accessed by a single read/write access port. The length of a segment is denoted by L_{seg} in the rest of this work. In this example, there are two 4-bit data segments in one stripe.

In Figure 5 (b), five extra domains have been added to store p-ECC bits for 1-step error detection. The value of these p-ECC bits are set to '10101' (from left to right). In order to access the p-ECC, an extra access port is added to the stripe. As shown in Figure 5 (c), we can find that this port is aligned to the fifth domain, counting from the right end of this stripe.

Since the values in these p-ECC domains are not changed during normal accesses to racetrack memory, only a **read** port is needed to reduce area overhead. More details can be found in subsection 4.2.3 about design overhead of p-ECC.

The detection of 1-step error using p-ECC is analogous to the parity check in traditional bit-error detection. After adding the SED p-ECC, 1-step error can be detected as follows:

- When domains are successfully shifted by even-step distance in any direction, the p-ECC bit read out should be equal to that read out before shift operation. If a 1-step error happens, the p-ECC bit read out is reversed.
- When domains are successfully shifted by odd-step distance in any direction, the p-ECC bit read out is reversed. If a 1-step error happens, the p-ECC bit read out is unchanged.

The example illustrated in Figure 5 demonstrates how p-ECC works. In order to read the only bit '1' in the stripe, all domains are shift to the right by one step. If domains are correctly shifted, bit '0' is read out from the p-ECC, shown as Figure 5 (c). However, if domains are over-shifted by one step (e.g. +1-step error happens), shown as Figure 5 (d), bit '1' is read out from p-ECC and a 1-step error is detected.

The SED p-ECC cannot correct the error because it is impossible to differentiate a $+1$-step error from a -1-step error. Due to the high error rate of 1-step error, such a SED protection is not enough. In the next subsection, we extend p-ECC to single step error correction and double steps error detection (SECDED).

4.2.2. p-ECC Design for SECDED

In order to correct ± 1-step errors, we first need to add an extra "guard" domain on both ends of the racetrack memory stripe. They prevent data loss when a 1-step error happens in either direction. These two guard domains are illustrated in Figure 6 (a). Then, we add extra domains to hold p-ECC bits for SECDED, as shown in Figure 6 (b). At the same time, extra read ports are needed to access these p-ECC bits during detection and correction. Since we are targeting SECDED, there are four potential states after a shift operation: (1) success, (2) $+1$-step error, (3) -1-step error, and ± 2-step error. Thus, we need to add two read ports to read out two p-ECC bits simultaneously, which are also illustrated in Figure 6 (b).

In the worst case, domains are shifted ($L_{seg}-1$)-step distance and a 2-step error happens at the same time. Thus, the total length of p-ECC is set as $L_{seg}+5$. It ensures that, even in the worst cases, those two read ports can still access valid p-ECC bits. For the example in Figure 6, the worst cases for shifting to both directions are illustrated in Figure 6 (c) and (d), respectively. Thus, we need $9 = 4+5$ domains for p-ECC.

These p-ECC bits are redrawn in Figure 6(e) in a cyclic style. When a shift to left direction happens, p-ECC bits change in clockwise according to the shift distance. When domains shift to right, they change in counter-clockwise accordingly. An example in Figure 6 is described as follows:

- All domains are in their initial states in Figure 6 (a).
- When domains are successfully shifted to the right by $4k$, $4k+1$, $4k+2$, and $4k+3$ steps, two bits read out from p-ECC should be '11', '10', '00', and '01' respectively.
- Based on these cyclic codes, both $+1$-step and -1-step errors can be identified. For example, when domains are supposed to be shifted by $4k$ steps, a $+1$-step error is detected if '10' are read out from p-ECC.
- After ± 1-step errors are detected, they can be corrected by shifting domains back by 1 step, accordingly.
- However, when 2-step errors happen, we only detect them but cannot correct them. It is because we cannot differentiate $+2$-step errors from -2-step errors.

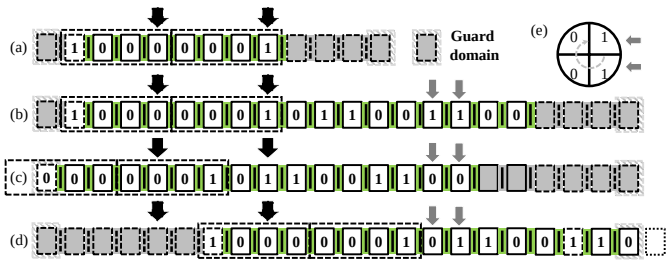

Figure 6: SECDEC P-ECC. (a) original racetrack memory stripe, (b) adding extra domains to protect the over-shift data lost, (c) worst case when domains are shifted left, (d) worst case when domains are shifted right, (e) cyclic code organization.

Compared to SED p-ECC, SECDED p-ECC requires more extra domain walls and read ports. In the next subsection, we will analytically discuss the overhead for correcting m-step position errors.

4.2.3. Correcting M-step Position Errors with p-ECC

Similar to the cyclic coding method for SECDED, we can further extend p-ECC to provide higher protection strength. It is analytically described as follows. In order to correct m-step ($m < L_{seg}-1$) position errors: (1) $2m$ extra domains are needed to guard data loss in worst cases; (2) the length of p-ECC is calculated as $L_{seg}-1+2m$; (3) $m+1$ extra read ports are needed. At the same time, the p-ECC can also detect (m+1)-step errors. However, the p-ECC induces overhead to area, latency, and energy consumption of a racetrack memory design.

Area overhead. Besides the extra domains added for p-ECC, adding extra read ports also impact area of racetrack memory. The racetrack memory stripe is stacked on transistors used for access ports. When there are only a few access ports, the total area of a racetrack memory stripe is mainly determined by the number of domains. Thus, the area overhead of adding one more read port is moderate (mainly from peripheral circuitry). However, if there are too many access ports, the total area is determined by the transistors. Thus, more overhead is introduced. In Figure 7, the area overhead of adding read ports is drawn with a 64-bit racetrack memory stripe based on previous models [43, 39] as an example.

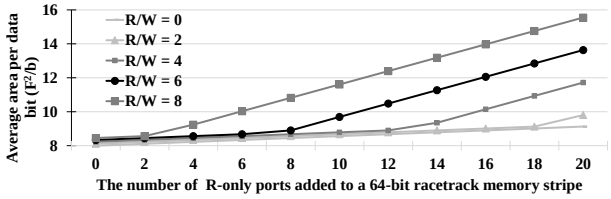

Figure 7: Overhead of adding read ports.

Latency/energy overhead. After using p-ECC, extra latency is induced in shift operation mainly due to two reasons. First, domain moving velocity is reduced due to longer stripes holding more domains. Second, extra latency is needed for error detection. Note that error detection of p-ECC may be processed at the same time with conventional ECC for read/write errors. Due to page limitation, it is not discussed in this paper. Similarly, extra energy consumption is caused by p-ECC. Detailed results can be found in evaluation section.

Apparently, overhead of p-ECC is closely related to the segment length L_{seg}. For example, when there is only one segment, the area overhead induced by p-ECC is more than 100%. In order to correct position errors for racetrack memory with long segment, we further propose a modified correction mechanism called p-ECC-O, which is introduced in the next subsection.

4.2.4. p-ECC-O: Leveraging Overhead Region

In previous approaches of p-ECC, bits in overhead region are not cared. In fact, if we carefully control bits stored in overhead region, these bits can also be employed for detection and correction of position errors. This is the basic idea of p-ECC-O. As shown in Figure 8, we add one write port at the end of each racetrack memory stripe. Then, we can control bits in overhead region with a "shift-and-write" operation. Note that one drawback is that shift can only be operated step by step because we need to write overhead region bit by bit.

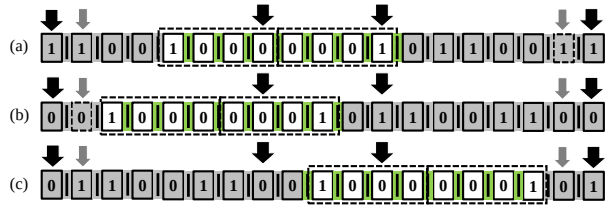

Figure 8: P-ECC (a) original racetrack memory stripe, (b) extra domains to protect the over-shift data lost. (c) worst case when domains are shifted left.

We still use the same example of 8-bit racetrack memory as an example to explain p-ECC-O. As shown in Figure 8, in order to achieve SECDED, we add four extra domains to each end of racetrack memory stripe. In addition, two ports are added to each end. As shown in Figure 8 (a), domains are in their left-most state. p-ECC in two overhead regions are also shown in this Figure. Similar to p-ECC in last subsections, cyclic codes are used.

When domains are shifted to right, p-ECC bits in right overhead region are read out for position error detection and correction. At the same time, cyclic codes are shifted into left overhead region, as shown in Figure 8 (b). With protection of p-ECC in the right overhead region, we can ensure that these bits are correctly shifted into the left overhead region. Thus, when domains are shifted to left, these p-ECC bits in left overhead region can be used for position error detection and correction. Since, we add four extra domains in each end, we can still read out valid p-ECC bits even in the worst case (Figure 8 (c)).

Different from original p-ECC, design overhead of p-ECC-O is only determined by position errors to be corrected. If we want to correct m-step errors, we need to add $2(m+1)$ extra domains in each end of the racetrack memory stripe. In addition, we need to add m more read ports than original p-ECC. It is easy to find that p-ECC-O can achieve lower overhead than original p-ECC, when the segment length is large. The detailed storage overhead is analyzed in Section 6.3.

However, due to bit by bit "shift-and-write", p-ECC-O induces higher overhead to shift latency. As addressed in Section 4.1, after using STS to eliminate "stop-in-middle" error, a long distance shift is preferred to amortize extra latency for STS. Including the p-ECC latency overhead, the latency for a single 7-step shift is 9 cycles, compared to 28 cycles for 7 times 1-step shift operations. We can find that original shift achieves better shift performance than p-ECC-O. Thus, shift performance is traded for storage density by p-ECC-O technique. We can draw similar conclusion for overhead of energy consumption because of a similar reason. Consequently, p-ECC should be applied when we run latency-sensitive but capacity-insensitive applications. On the contrary, p-ECC-O is preferred when we run capacity-sensitive but latency-insensitive applications. More results about the trade-off can be found in Section 6.4.

4.3. p-ECC Initialization

In previous subsections, we have discussed how to detect and correct position errors with pre-programmed p-ECC in a racetrack memory stripe. The process of the pre-programming is called p-ECC initialization in this work. Apparently, an initialization is required after a racetrack memory is fabricated. In addition, it is also needed whenever a racetrack memory has to be flushed when uncorrectable errors happen or the system is crashed. Since position errors may also happen during initialization, extra effort is needed to ensure correct p-ECC is programmed into racetrack memory.

A straightforward method is to “program-and-test” iteratively. We take the case in Figure 6 as an example to describe this process as follows:

- **Step-1** p-ECC bits are written into the racetrack memory stripe from the left most port sequentially.
- **Step-2** These p-ECC bits are shifted step by step to the right till they reach the right most two read ports for p-ECC detection. During the shift process, p-ECC bits are read out for testing by all ports along the stripe. If any unexpected bits are detected, the initialization process restarts.
- **Step-3** These bits are shifted to the left most port step by step, and the a similar test in Step-2 is performed during shifting.
- **Step-4** Step-2 and Step-3 are repeated for enough rounds to ensure that p-ECC bits are correctly programmed.

For a racetrack memory stripe with 64 data domains, eight access ports, the error rate of initializing a SECDED p-ECC can be reduced to lower than 10^{-100} after one iteration. The expected latency of completing this initialization process is about 1200 cycles. Thus, the total initialization time for a 128MB racetrack memory should be less than 20ms.

5. Position Error Aware Shift Architecture

In this section, we first introduce how to extend a shift controller to support STS and p-ECC. Then, we define a concept called "safe distance" and propose a shift architecture under constraint of safe distance.

5.1. Error Aware Shift Controller

The overview of an error aware shift controller is shown in Figure 9. Those light-color blocks represent common components required for any racetrack memory design. Those dark-color blocks are extra components needed for position error detection/correction.

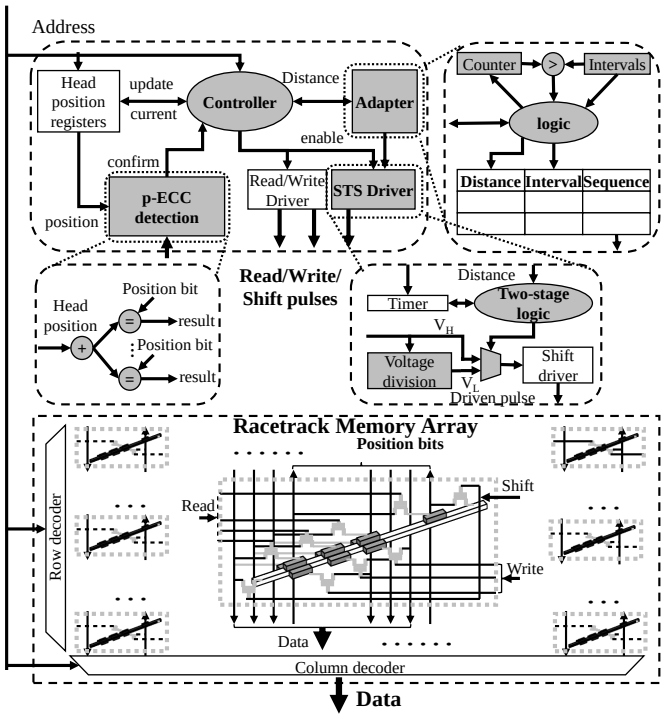

Figure 9: Overview of error aware shift controller.

"STS driver" in the figure enables sub-threshold shift. Two-stage logic and voltage division circuit are key designs in the driver. The two stage-logic sets the timer according to its input shift distance. The logic generates pulses according to timer duration, and selects high/low voltages to shift domains with different current density. Shift voltage is mainly controlled by a voltage division circuit or charge-pump [28].

The component "p-ECC detection" is responsible for position error detection. A customized *cyclic adder* is needed to generate expected p-ECC bits based on current p-ECC bits and the shift distance. We need add extra XOR gates for p-ECC bits comparison. Comparison results will be feed back to the controller. Any mismatch results in a detected error. Then, it will generate an extra shift for error correction, accordingly.

The "adapter" in Figure 9 is used to optimize a shift operation under a certain constraint of MTTF. Details are introduced in the next two subsections.

5.2. Safe Shift Distance

As introduced in our error model, the error rate of a shift operation is also related to its shift distance. In Table 2, the second column lists the 1-step error rates for different shift distances, after applying SECDED p-ECC. In this example, the segment length is set to eight. Thus, the distance of a single shift operation varies from 1-step to 7-step. Given a design goal of MTTF and using a specific p-ECC, we need to limit the longest (maximum) distance that can be performed by a single shift operation. This is called safe shift distance (or **safe distance** for simplicity) in this work.

Since MTTF is a statistical value, the safe shift distance is not only determined by the error rates but also related to shift intensity. Statistically, if the average shift intensity (shift operations per second) is denoted by I_{mean}, the error rate of one shift operation should not exceed $1/(T_{MTTF} \times I_{mean})$ in the worst case. With given error rate, the MTTF target determines the safe distance of a racetrack memory design. For the 64-bit racetrack memory stripe used in previous example, the relationship between safe distance and average shift intensity is listed in Table 3 (a).

Table 3: (a) safe distance vs. shift intensity. (b) safe shift sequences of a 7-step shift.

D_{safe}	Error rate	Intensity
1	1.37E-21	4.53G
2	1.19E-20	518M
3	5.59E-20	111M
4	1.80E-19	34.3M
5	4.47E-19	13.9M
6	9.96E-18	621K
7	7.57E-15	0.82K

Interval	Sequence	Lat.
2445260	7	9
76	4,3	13
26	3,2,2	16
12	2,2,2,1	19
9	2,2,1,1,1	22
6	2,1,1,1,1,1	25
3	1,1,1,1,1,1,1	28

Apparently, if the distance of a shift operation exceeds the safe distance, it has to be completed with a sequence of multiple shift operations. Given a safe distance D_{safe}, the optimal shift sequence for a request of D_{req} distance is calculated in a flow described in Algorithm 1.

Algorithm 1: Select shift sequence for a long distance.

Input : Required shift distance D_{ref}, a safe error rate P_{safe}
Output : Safe shift sequence $S = \{D_0, ..., D_k\}$
Find out all possible shift sequences for D_{ref};
for *each sequence* **do**
    calculate overall error rate of the sequence, p;
    calculate overall latency of the sequence, l;
end
Output the sequence that has smallest latency when $p < P_{safe}$.

In real cases, it is difficult to estimate average shift intensity of a racetrack memory because it varies significantly for different applications. A conservative solution is to estimate the safe distance using highest access frequency of a racetrack memory. For example, a 128MB racetrack memory used in Section 6 can support up to $83M$ accesses per second. Thus, the safe distance is set to 3 steps conservatively. In order to provide a better estiamtion, we further propose an adaptive shift architecture in the next subsection.

5.3. Adaptive Shift Architecture

Instead of using the safe distance of worst case all the time, we can calculate run-time shift intensity to select a proper safe distance for each shift operation. To simplify the discussion, we assume that the racetrack memory services only one request at one time. For a shift operation, the interval between it and the last shift operation is denoted by T_{inter}. Then, the run-time shift intensity can be calculated as $1/T_{inter}$, which is used to select the safe distance for the current shift operation. Note that if multiple requests are serviced simultaneously by an interleaving techniques, we only need to increase run-time intensity accordingly.

Having the run-time safe distance, we can select a proper shift sequence for any distance shift operations. For any shift distance, we need to record the relationship between a shift sequence and its interval threshold. Such a relationship for a 7-step shift operation is illustrated in Table 3. For example, if the interval is only 3 cycles, this shift operation has be to completed in seven times of 1-step shift. These relationships for all shift distances are stored in "Adapter" as a table. A counter is used to trace the interval. These components are also illustrated in Figure 9. Since only one global table and one counter is needed, the overhead is trivial.

The p-ECC considering safe distance is denoted as "**p-ECC-S**" to simplify discussion in this work. Two approaches in this subsection and the last one are labeled as "p-ECC-S adaptive" and "p-ECC-S worst" in next evaluation section, respectively.

6. Evaluation

In this section, we evaluate the impact of our techniques on reliability, area, performance, and energy of racetrack memory.

6.1. Experiments Setup

A full system simulator gem5 [6] is employed for performance simulation. Its cache model is extended to support racetrack memory, and to report detailed shift statistics. Given error rates for different shift operations, we track run-time errors that may happen during simulation of different workloads. Both silent data corruption (SDC) and detected unrecoverable errors (DUE) MTTF can be calculated based on error numbers and simulated execution time.

The area and read/write/shift latency/energy for an ideal racetrack memory is estimated from a circuit level model [46]. To evaluate area, timing, and energy overhead of error correction, we implement RTL codes for extra hardware and synthesize them with 45nm technology node. Then, these number are applied on ideal racetrack memory model for comparison.

The detailed configuration of the simulated system is listed in Table 4. For the last level cache (LLC) design, there are three options of memory technologies: SRAM, STT-RAM, and racetrack memory. The process technology node is 45nm for all memories. LLC area is kept similar for different technologies. Paramters for SRAM and STT-RAM LLCs are obtained from NVSim [12].

Racetrack memory device parameters are set by Table 1. Data mapping is same to previous work [43]: A 64B cache line is interleaved onto 512 stripes. Each stripe has 64 data domains divided into eight segments (i.e. eight read/write ports) by default. Note that the shift latency and energy numbers in Table 4 are for 1-step shift distance. Baseline has same number of ports as that using p-ECC (O/S). The benchmarks come from PARSEC [5] suite. It provides multi-threading programs for all domain from desktop to server applications [5].

Table 4: The detailed configurations of evaluation system.

Unit	Configurations		
CPU	4 single Alpha cores, 2GHz, 1-way issue		
L1	split I/D, 32KB/32KB, 2-way, 64B,LRU, private, R/W: 1/1-cycle, 0.074/0.074nJ, 23.4mW		
L2	1MB shared by 2 cores, 4-way, 64B, LRU R/W lat.: 7/7-cycle, R/W E: 0.407/0.386-nJ, 681.5mW		
L3	16-way, 64B, share, LRU		
	SRAM	STT-RAM	RM
	4MB, R/W: 24/22-cycle, 0.802/0.761-nJ 2673.5mW	32MB, R/W: 27/41-cycle, 1.056/2.093-nJ 862.2mW	128MB, R/W/S: 24/24/4-cycle, 0.956/0.952/1.331-nJ 948.4mW
Mem.	Dual Channel DDR3, 1600MHz, 100-cycle, 38.10nJ, 12.8GB/s.		

6.2. Reliability Evaluation

Figure 10 shows the SDC MTTF performance with different protection mechanisms. The baseline is only $1.33\mu s$. The basic SED p-ECC can easily improve the SDC MTTF to $3.56 \times 10^5 s$ (about 10 hours). And SECDED p-ECC(-O) can further improve it to more than 1000 years, which can fulfill the reliability target [8]. Since the position error aware shift architecture further improves the reliability, p-ECC-S can achieve SDC MTTF target. Thus we will only focus on DUE MTTF as the reliability metric in the rest of this section.

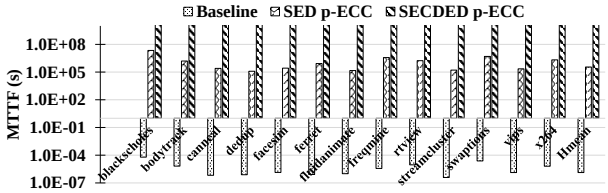

Figure 10: SDC MTTF under different protection.

The DUE MTTF performance is shown in Figure 11. Because the SED p-ECC can only detect the position error without correction, the risk of DUE is high (MTTF is small). After using SECDED p-ECC, the DUE MTTF is increased to about 10^5 seconds (about 1 day) on average. But it's not good enough. Thus, the safe distance is critical for a reliable shift operation. As shown in Figure 11, the DUE MTTF after p-ECC-S worst approach is increased to about $1.68 \times 10^{10} s$ (532-year). Even though the adaptive approach (i.e. p-ECC-S adaptive), reduces it to about $2.18 \times 10^9 s$ (i.e. 69-year), it still satisfies the 10-year DUE MTTF target [8].

To prove feasibility of p-ECC-S and p-ECC-O, we further evaluate DUE MTTF of racetrack memory with various configurations in Figure 12, under the same error rate. The data length (number of data domains) of a stripe varies from 32 to 128. For a fixed data length, the number and size of a segment

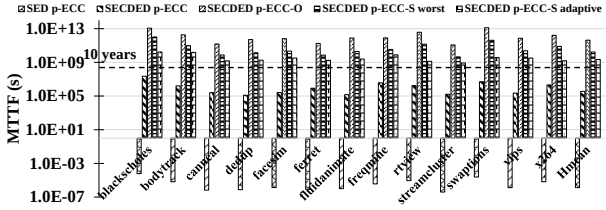

Figure 11: DUE MTTF under different protection.

also change for different configurations (e.g. 16×2, 8×8 in the figure). Note that the effect of adding more ports can be estimated by increasing the segment number. Both methods shows enough protection strength. For p-ECC-S, MTTF increases as the segment length is reduced. Simply speaking, it is because the average shift distance decreases with segment length. With the constraint of safe distance, p-ECC-S always works with different segment lengths. For p-ECC-O, MTTF is almost kept same for different configurations because the maximum shift distance is always limited to 1-step. That's why both methods provide same protection when $L_{seg} = 2$. Note that p-ECC-O achieves the highest DUE MTTF, paying cost in time/energy performance to use the overhead region. In following subsections, we will explore their tradeoff among performance, energy and reliability.

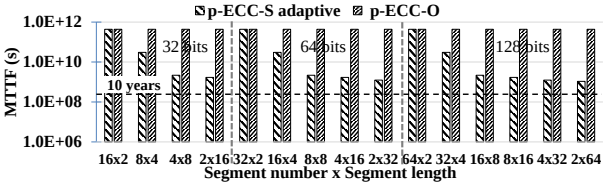

Figure 12: Sensitivity analysis of MTTF.

6.3. Area Evaluation

Overhead of position error detection/correction is induced by extra domains to save p-ECC codes and control logic. Table 5 lists the overhead with the default configuration ($L_{seg} = 8$). The time cost of detection and correction for p-ECC are about 0.3*ns* and 1.3*ns*. The capacity overhead from extra domains is 17.6%. And the controller is small enough. The p-ECC-S adaptive case costs more to detect/correct errors and has more complex control, but achieves better time performance.

Table 5: Design overhead of position error protection

Approaches	Detection/stripe		Correction/stripe		Area	
	Time (ns)	Energy (pJ)	Time (ns)	Energy (pJ)	Cell (%)	Controller (um^2)
STS	0.82	1.31	0.82	1.31	N/A	1.94
p-ECC	0.34	3.73	1.34	6.16	17.6	54.0
p-ECC-O	0.34	3.74	1.34	9.90	15.7	54.0
p-ECC-S worst	0.38	3.75	1.35	6.17	17.6	54.3
p-ECC-S adaptive	0.61	3.86	1.37	6.19	17.6	109.4

In Figure 13, using the LLC as an example, we compare the impact of our techniques on average area per bit for various configurations. Note that the area of controller and other peripheral circuitry is also counted. Baseline is for racetrack memory without any protection. We didn't include p-ECC and "p-ECC-S worst" since they have same cell area overhead as "p-ECC-S adaptive". When the segment length is small ($L_{seg} < 8$), area overhead is trivial for both p-ECC-S and p-ECC-O. p-ECC-O becomes more efficient for a large segment length ($L_{seg} \geq 16$). However, it induces more overhead in performance and energy, which is discussed in the next subsection.

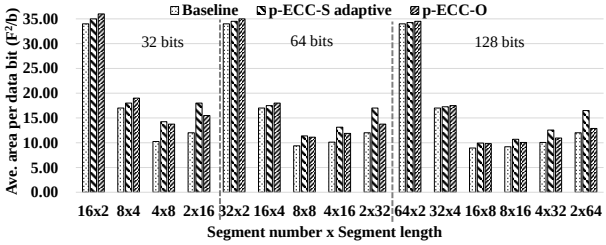

Figure 13: Sensitivity analysis for area overhead.

6.4. Performance and Energy Evaluation

Total shift latency for different workloads are compared in Figure 14 with different protection techniques. The baseline is a racetrack memory without any protection, which has no constrain on maximum shift distance. Compared with the p-baseline, p-ECC-O method introduces about $2\times$ latency overhead due to 1-step maximum shift distance. After considering safe distance, "p-ECC-S worst" reduces latency overhead to 23%. And it is further reduced to 23% after using "p-ECC-S adaptive".

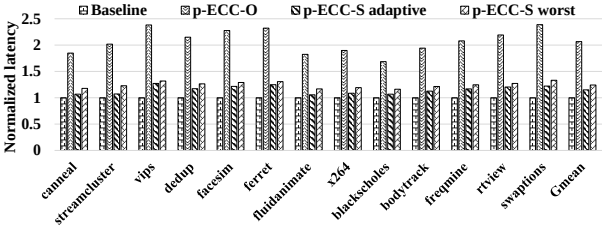

Figure 14: Relative shift latency of racetrack memory.

For the LLC with different configurations, we compare average shift latency in Figure 15. When the segment length is small, "p-ECC-S adaptive" and p-ECC-O induce trivial extra shift latency. When the segment is long, however, "p-ECC-S adaptive" becomes more efficient because shift distance is relaxed according to access intensity.

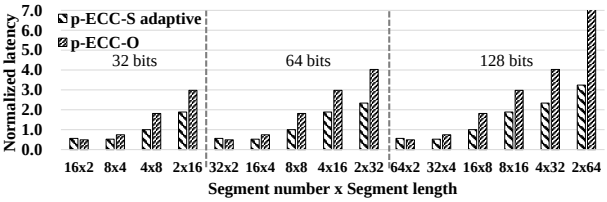

Figure 15: Sensitivity analysis of shift latency.

In Figure 16, we analyze the impact of position error protection on full system execution time for different workloads. To provide a comprehensive comparison, we also present results of using SRAM LLC, STT-RAM LLC, and an ideal racetrack memory LLC, shift latency of which is entirely removed. The

workloads are mainly divided into two types: (1) capacity sensitive and (2) capacity insensitive. For capacity insensitive workloads (small working sets), execution time is reduced little when the LLC capacity is increased after using racetrack memory. For capacity sensitive workloads, execution time of using racetrack memory is reduced substantially because of less miss rate, compared to SRAM and STT-RAM.

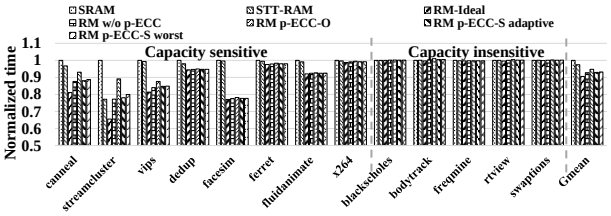

Figure 16: Overall execution time.

After using STS and p-ECC, execution time is increased due to extra shift latency. Fortunately, the overhead is moderate. As shown in Figure 16, p-ECC-O only increases the execution time about 2% on average over that without any protection. The reason is in two folds: (1) some cache accesses do not need shift operations; (2)shift latency is not always on critical path. For some latency sensitive workloads (e.g. streamcluter), however, the overhead caused by error correction may even offset the benefits of miss rate reduction. On average, "p-ECC-S worst" and "p-ECC-S adaptive" only cause about 0.5% and 0.2% performance overhead, respectively.

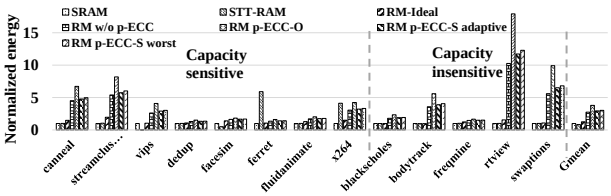

Figure 17: The shift energy of p-ECC.

To address energy overhead caused by position error protection, we compare LLC dynamic energy in Figure 17. Dynamic energy is similar for SRAM, STT-RAM and a racetrack memory without protection. Obviously, dynamic energy increases significantly after applying position error protection. The p-ECC-O consumes 46% more dynamic energy than that without any protection. The "p-ECC-S worst" and "p-ECC-S adaptive" reduce dynamic energy overhead to 14% and 20%, respectively.

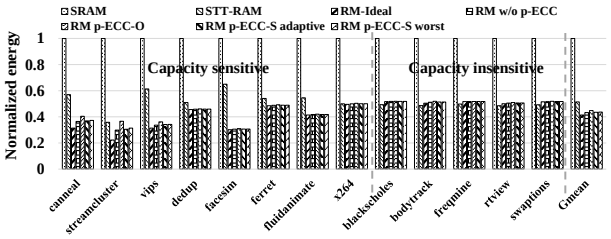

Figure 18: Analysis of energy consumption benefits.

Figure 18 shows impact of error detection on total energy benefits. The results include dynamic energy for read, write, error detection, and leakage power for all cache levels, and dynamic energy of main memory. The average energy reduction using STT-RAM L3 cache is 53.1% over SRAM. The energy reduction for "p-ECC-O" and "p-ECC-S adaptive" are 53.1% and 54.1%, respectively. Compared to STT-RAM, even considering energy overhead for error detection, we can still gain benefits of using racetrack memory because of less accesses to main memory.

7. Related Work

Recent research on error correction code (ECC) focuses on how to reduce the overhead, such as VS-ECC [4], CPPC [21], little-space ECC [31], etc. The work explores trade-off between reliability and area/performance/energy. [36, 15, 41, 18]. Besides conventional ECC for transient errors, other approaches have also attracted attention. For example, acoustic wave detector based architecture [42] and light weight error detection method for GPU [17] have been proposed recently. In addition, software based methods are also efficient to detect [30, 10] and recovery [13, 11] different errors.

As different non-volatile memory technologies emerge, the detection and correction of errors in NVM are well addressed recently. Seong *et al*. proposed Tri-level cell of PCM and SAFER for stuck-at errors to achieve reliable memory system [34, 33]. Schechter *et al* proposed ECP to replace ECC for resistive memory, in order to handle permanent errors [32]. Qureshi *et al* proposed adaptive ECC to reduce the overhead of error correction for PCM [27].

However, since the shift operation is unique to racetrack memory, detection and correction of position errors are not covered in previous research yet. Thus, we model and define the position errors, and propose corresponding error correction mechanism.

8. Conclusions

Racetrack memory is attractive because of its ultra-high density, fast access speed, and non-volatility. However, one obstacle of using racetrack memory is that position errors happen during shift operation. Our model demonstrates that the raw position error rate is unacceptable (upto 0.1% for a 7-step shift) for modern memory design requirement. Unfortunately, conventional ECC designed for bit errors cannot work efficiently for position error, which leads to significant degradation of reliability. To overcome this problem, we first introduce STS technique to convert stop-in-middle errors into out-of-step errors. Then,we further propose p-ECC to mitigate out-of-step position errors. We explore trade-off between reliability and design overhead with various p-ECC architectures. Experimental results show that we can improve SDC MTTF of racetrack memory to more than 1000 years and DUE MTTF to 69 years using position error aware adaptive shift architecture (p-ECC-S adaptive). The performance and energy overhead are only 0.2% and 20%, respectively.

9. Acknowkedgements

This work is supported by National High-tech R&D Program of China (No.2013AA013201), NSFC (No.61202072, No.61471015, No.61373026). This work is also supported by Huawei Shannon Lab Project and the Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions under contract YETP0102.

References

[1] "Amd eighth-generation processor architecture." ADVANCED MICRO DEVICES, October 2001. [Online]. Available: http://intel80386.com/amd/k8_architecture.pdf

[2] "Ultrasparc iv processor architecture overview technical whitepaper." SUN microsystems, February 2004. [Online]. Available: http://laser.cbs.cnrs.fr/IMG/pdf/SUN-usiv-arch.pdf

[3] "Intel pentium 4 processor on 90nm process datasheet." INTEL, February 2005. [Online]. Available: http://download.intel.com/design/Pentium4/datashts/30056103.pdf

[4] A. R. Alameldeen, I. Wagner, Z. Chishti, W. Wu, C. Wilkerson, and S.-L. Lu, "Energy-efficient cache design using variable-strength error-correcting codes," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011, pp. 461–472. [Online]. Available: http://doi.acm.org/10.1145/2000064.2000118

[5] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," Princeton University, Tech. Rep. TR-811-08, January 2008.

[6] 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*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[7] D. Bossen, J. M. Tendler, and K. Reick, "Power4 system design for high reliability," *Micro, IEEE*, vol. 22, no. 2, pp. 16–24, Mar 2002.

[8] D. C. Bossen, "CMOS Soft Errors and Server Design," *IEEE 2002 Reliability Physics Tutorial Notes, Reliability Fundamentals*, vol. 121, pp. 07–1, 2002.

[9] C. Burrowes, A. P. Mihai, D. Ravelosona, J. V. Kim, C. Chappert, L. Vila, A. Marty, Y. Samson, F. Garcia-Sanchez, L. D. Buda-Prejbeanu, I. Tudosa, E. E. Fullerton, and J. P. Attane, "Non-adiabatic spin-torques in narrow magnetic domain walls," *Nat Phys*, vol. 6, no. 1, pp. 17–21, 01 2010.

[10] L. Chen and Z. Zhang, "Memguard: A low cost and energy efficient design to support and enhance memory system reliability," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ser. ISCA '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 49–60. [Online]. Available: http://dl.acm.org/citation.cfm?id=2665671.2665683

[11] M. de Kruijf, S. Nomura, and K. Sankaralingam, "Relax: An architectural framework for software recovery of hardware faults," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 497–508. [Online]. Available: http://doi.acm.org/10.1145/1815961.1816026

[12] 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*, vol. 31, no. 7, pp. 994–1007, July 2012.

[13] S. Feng, S. Gupta, A. Ansari, S. A. Mahlke, and D. I. August, "Encore: Low-cost, fine-grained transient fault recovery," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44. New York, NY, USA: ACM, 2011, pp. 398–409. [Online]. Available: http://doi.acm.org/10.1145/2155620.2155667

[14] M. Hayashi, "Current driven dynamics of magnetic domain walls in permalloy nanowires," Ph.D. dissertation, Stanford University, 2006.

[15] R. Huang and G. E. Suh, "Ivec: Off-chip memory integrity protection for both security and reliability," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 395–406. [Online]. Available: http://doi.acm.org/10.1145/1815961.1816015

[16] A. Iyengar and S. Ghosh, "Modeling and analysis of domain wall dynamics for robust and low-power embedded memory," in *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*. ACM, 2014, pp. 1–6.

[17] H. Jeon and M. Annavaram, "Warped-dmr: Light-weight error detection for gpgpu," in *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*, Dec 2012, pp. 37–47.

[18] X. Jian and R. Kumar, "Adaptive reliability chipkill correct (arcc)," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, Feb 2013, pp. 270–281.

[19] R. Kessler, "The alpha 21264 microprocessor," *Micro, IEEE*, vol. 19, no. 2, pp. 24–36, Mar 1999.

[20] H. Lee, P. Chen, T. Y. Wu, Y. Chen, C. Wang, P. Tzeng, C. H. Lin, F. Chen, C. Lien, and M. J. Tsai, "Low power and high speed bipolar switching with a thin reactive ti buffer layer in robust hfo2 based rram," in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, Dec 2008, pp. 1–4.

[21] M. Manoochehri, M. Annavaram, and M. Dubois, "Cppc: Correctable parity protected cache," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011, pp. 223–234. [Online]. Available: http://doi.acm.org/10.1145/2000064.2000091

[22] C. McNairy and D. Soltis, "Itanium 2 processor microarchitecture," *Micro, IEEE*, vol. 23, no. 2, pp. 44–55, March 2003.

[23] I. M. Miron, T. Moore, H. Szambolics, L. D. Buda-Prejbeanu, S. Auffret, B. Rodmacq, S. Pizzini, J. Vogel, M. Bonfim, A. Schuhl, and G. Gaudin, "Fast current-induced domain-wall motion controlled by the rashba effect," *Nat Mater*, vol. 10, no. 6, pp. 419–423, 06 2011.

[24] A. Mishra, X. Dong, G. Sun, Y. Xie, N. Vijaykrishnan, and C. Das, "Architecting on-chip interconnects for stacked 3d stt-ram caches in cmps," in *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, June 2011, pp. 69–80.

[25] S. Mukherjee, J. Emer, and S. Reinhardt, "The soft error problem: an architectural perspective," in *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, Feb 2005, pp. 243–247.

[26] S. S. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," *Science*, vol. 320, no. 5873, pp. 190–194, 2008.

[27] M. K. Qureshi, "Pay-as-you-go: Low-overhead hard-error correction for phase change memories," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44. New York, NY, USA: ACM, 2011, pp. 318–328. [Online]. Available: http://doi.acm.org/10.1145/2155620.2155658

[28] A. Raychowdhury, B. Geuskens, J. Kulkarni, J. Tschanz, K. Bowman, T. Karnik, S.-L. Lu, V. De, and M. Khellah, "Pvt-and-aging adaptive wordline boosting for 8t sram power reduction," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, Feb 2010, pp. 352–353.

[29] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge University Press, 2009. [Online]. Available: http://books.google.com/books?id=0gwqxBU_t-QC

[30] S. K. Sastry Hari, M.-L. Li, P. Ramachandran, B. Choi, and S. V. Adve, "mswat: Low-cost hardware fault detection and diagnosis for multicore systems," in *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42. New York, NY, USA: ACM, 2009, pp. 122–132. [Online]. Available: http://doi.acm.org/10.1145/1669112.1669129

[31] Y. Sazeides, E. Özer, D. Kershaw, P. Nikolaou, M. Kleanthous, and J. Abella, "Implicit-storing and redundant-encoding-of-attribute information in error-correction-codes," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-46. New York, NY, USA: ACM, 2013, pp. 160–171. [Online]. Available: http://doi.acm.org/10.1145/2540708.2540723

[32] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ecp, not ecc, for hard failures in resistive memories," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 141–152. [Online]. Available: http://doi.acm.org/10.1145/1815961.1815980

[33] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee, "Safer: Stuck-at-fault error recovery for memories," in *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '43. Washington, DC, USA: IEEE Computer Society, 2010, pp. 115–124. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2010.46

[34] N. H. Seong, S. Yeo, and H.-H. S. Lee, "Tri-level-cell phase change memory: Toward an efficient and reliable memory system," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 440–451. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485960

[35] S.-S. Sheu, M.-F. Chang, K.-F. Lin, C.-W. Wu, Y.-S. Chen, P.-F. Chiu, C.-C. Kuo, Y.-S. Yang, P.-C. Chiang, W.-P. Lin, C.-H. Lin, H.-Y. Lee, P.-Y. Gu, S.-M. Wang, F. Chen, K.-L. Su, C.-H. Lien, K.-H. Cheng, H.-T. Wu, T.-K. Ku, M.-J. Kao, and M.-J. Tsai, "A 4mb embedded slc resistive-ram macro with 7.2ns read-write random-access time and 160ns mlc-access capability," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, Feb 2011, pp. 200–202.

[36] J. Sim, G. H. Loh, V. Sridharan, and M. O'Connor, "Resilient die-stacked dram caches," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 416–427. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485958

[37] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. Stan, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, Feb 2011, pp. 50–61.

[38] 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*, March 2012, pp. 1507–1512.

[39] Z. Sun, W. Wu, and H. Li, "Cross-layer racetrack memory design for ultra high density and low power consumption," in *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, May 2013, pp. 1–6.

[40] S. Tang, A. Keshavarzi, D. Somasekhar, F. Paillet, M. Khellah, Y. Ye, S. Lu, and V. De, "Floating-body dynamic random access memory with purge line," May 11 2006, uS Patent App. 11/289,621. [Online]. Available: http://www.google.com/patents/US20060098482

[41] A. N. Udipi, N. Muralimanohar, R. Balsubramonian, A. Davis, and N. P. Jouppi, "Lot-ecc: Localized and tiered reliability mechanisms for commodity memory systems," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ser. ISCA '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 285–296. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337159.2337192

[42] G. Upasani, X. Vera, and A. González, "Avoiding core's due & sdc via acoustic wave detectors and tailored error containment and recovery," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ser. ISCA '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 37–48. [Online]. Available: http://dl.acm.org/citation.cfm?id=2665671.2665682

[43] R. Venkatesan, S. Ramasubramanian, S. Venkataramani, K. Roy, and A. Raghunathan, "Stag: Spintronic-tape architecture for gpgpu cache hierarchies," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, June 2014, pp. 253–264.

[44] R. Venkatesan, V. Kozhikkottu, C. Augustine, A. Raychowdhury, K. Roy, and A. Raghunathan, "Tapecache: A high density, energy efficient cache based on domain wall memory," in *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '12. New York, NY, USA: ACM, 2012, pp. 185–190.

[45] Z. Wang, D. Jimenez, C. Xu, G. Sun, and Y. Xie, "Adaptive placement and migration policy for an stt-ram-based hybrid cache," in *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*, Feb 2014, pp. 13–24.

[46] C. Zhang, G. Sun, W. Zhang, F. Mi, H. Li, and W. Zhao, "Quantitative modeling of racetrack memory, a tradeoff among area, performance, and power," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*. IEEE, 2015, pp. 100–105.

[47] Y. Zhang, W. Zhao, D. Ravelosona, J.-O. Klein, J. Kim, and C. Chappert, "Perpendicular-magnetic-anisotropy cofeb racetrack memory," *Journal of Applied Physics*, vol. 111, no. 9, pp. 093 925–093 925–5, May 2012.

[48] W. Zhao, Y. Zhang, H.-P. Trinh, J.-O. Klein, C. Chappert, R. Mantovan, A. Lamperti, R. Cowburn, T. Trypiniotis, M. Klaui, J. Heinen, B. Ocker, and D. Ravelosona, "Magnetic domain-wall racetrack memory for high density and fast data storage," in *Solid-State and Integrated Circuit Technology (ICSICT), 2012 IEEE 11th International Conference on*, Oct 2012, pp. 1–4.