WiSE: When Learning Assists Resolving STT-MRAM Efficiency Challenges

Arash Salahvarzi*, Mohsen Khosroanjam*, Amir Mahdi Hosseini Monazzah*†, Hakem Beitollahi*, Umit Ogras‡, and Mahdi Fazeli§

Abstract—

Spin Transfer Torque Magnetic RAM (STT-MRAM) is one of the most promising on-chip technologies, which delivers high density, non-volatility, and near-zero leakage power. However, STT-MRAM suffers from three reliability issues, namely, read disturbance, write failure, and retention failure, that present significant challenges to its use as a reliable on-chip memory. All of these three reliability challenges become even more threatening with any increase in STT-MRAM cell temperature. Write operations are regarded as the main source of heat generation and temperature increase in STT-MRAM on-chip memories. This paper first presents experiments to show how the heat generated by consecutive writes affects the reliability of an STT-MRAM on-chip cache. Then, it proposes the WiSE framework, an approach to reduce the STT-MRAM-based cache memory temperature and improve its reliability. WiSE utilizes the Reinforcement Learning (RL) technique to detect high-density write operation patterns in STT-MRAM cache. To manage the write operations across the STT-MRAM caches, WiSE introduces a new temperature-aware replacement policy. The simulation results show that while WiSE imposes only about 1% performance overhead, it improves retention failure rate, read disturbance rate and write failure rate by 64%, 57%, and 47%, respectively, compared to Least Recently Used (LRU) replacement policy.

Index Terms—Reinforcement Learning, Reliability, Replacement Policy, STT-MRAM, Heat Accumulation

1 Introduction

Recently researchers have become increasingly interested in promising alternatives for Static RAM (SRAM) cells in the structure of on-chip Last Level Cache (LLC) memories [1]. Among emerging memory technologies, STT-MRAM has been regarded as one of the most promising candidates due to its better scalability, higher density, and lower leakage power [2], [3].

Reliability is one of the main concerns in all parts of integrated circuits from CPU to interconnects [4]. Among these components, cache memory is of decisive importance since it maintains the most valuable asset, i.e., data. Despite all of its advantages, STT-MRAM suffers from some reliability challenges that should be addressed to be employed as an on-chip memory [5]. Briefly, three failure mechanisms challenge the reliability of STT-MRAMs: 1) *read disturbance:* an unexpected value change of a memory cell during a read operation, 2) *write failure:* erroneously written value in an STT-MRAM cell during a write operation, and 3) *retention failure:* unintended value change of an STT-MRAM cell during idle times [6].

All three failure mechanisms of STT-MRAM cells are affected by temperature fluctuations [7]. Temperature increase reduces the *thermal stability factor* (Δ), the ratio of the thermal stability barrier to the operating temperature, which in turn, exponentially increases both read disturbance and retention failure rates. Besides, higher temperature increases write switching threshold current resulting in an increase in write failure rate [7].

Write operations are known to be the major source of heat generation and temperature increase in STT-MRAM cache memories [8], [9]. Accordingly, distributing write operations evenly across the memory banks of cache memories would efficiently distribute the heat over the STT-MRAM cache memory. On the other hand, the write operations in the cache are highly application-dependent. Since cache memory utilizes temporal and spatial localities in the application address space to maintain the most frequently referenced data, the write operation density in cache memories would be localized in some specific parts of the cache at each time epochs. Thus, usually, a few parts of cache memory banks would be hot while most parts of them would be cold.

To this end, if one finds high-density write operation patterns in the applications and distributes them across the memory banks, the peak temperature and average temperature of the cache memory banks would significantly decrease and many reliability issues of STT-MRAM caches would alleviate. The more accurate detection of high-density write operation patterns, the more reliability improvement achieve. Thus, to detect these patterns across the applications, we require an efficient method.

In this paper, first, we show how the heat generated by consecutive writes affects the reliability of STT-MRAM on-chip caches. Then, we introduce WiSE, an approach to

* *Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.*
† *School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.*
‡ *Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, Wisconsin, USA.*
§ *School of Information Technology, Halmstad University, Halmstad, Sweden.*
E-mails: arash_salahvarzi@alumni.iust.ac.ir, mohsen_khosroanjam@comp.iust.ac.ir, monazzah@iust.ac.ir, beitollahi@iust.ac.ir, uogras@wisc.edu, mahdi.fazeli@hh.se

Manuscript received ...
Manuscript accepted ...

alleviate the STT-MRAM-based cache memory temperature and improve its reliability. In WiSE, we try to track the write operation behavior across the memory banks of STT-MRAM cache through a Machine Learning (ML)-based technique. Indeed, ML provides automated methods that can detect patterns in data and analyze (classify) them. Notably, WiSE utilizes RL technique to detect high density write operation patterns in applications.

RL is a class of ML models where the learning process is based on evaluative feedbacks without any supervised signals. It aims to create agents similar to the humans, which learn for themselves by trial-and-error, solely from rewards or punishments, to develop successful strategies that eventually lead to the largest long-term rewards [10].WiSE utilizes the RL technique to detect high-density write operation patterns in applications.

Since replacement policy is the main contributor in placements of the blocks in the cache memories, after finding high density write operation patterns, to distribute them across the STT-MRAM cache's memory banks, we propose a new replacement policy. WiSE replacement policy tries to uniformly distribute high density write operation patterns all over the chip to reduce generated heat due to the write operations and improve reliability challenges.

To evaluate WiSE, we employ the gem5 cycle-accurate simulator [11], using a set of benchmarks from the *SPEC CPU2006* [12]. Our simulation results show that WiSE decreases the average and peak temperatures of cache blocks significantly up to 17.99°C and 26.36°C, respectively, compared to LRU. Based on the simulation results, compared to LRU, WiSE decreases retention failure rate by 64%, read disturbance rate by 57%, and write failure rate by 47%.

The remaining of this paper is organized as follows: In Section 2, the background of STT-MRAM and learning are presented. In Section 3, the motivation of this study is introduced. Afterward, in Section 4, WiSE is explained, followed by a comprehensive experimental study in Section 5. The most related previous studies are explored in Section 6. Finally, we conclude this paper in Section 7.

2 Preliminaries

In the following, we first discuss the basics of STT-MRAM technology and explore its reliability challenges. Then we review some preliminaries on ML.

2.1 STT-MRAM Fundamentals

Fig. 1 depicts the structure of an STT-MRAM bit-cell, which consists of an access transistor and a Magnitude Tunnel Junction (MTJ) device. An MTJ device, as shown in Fig. 1, consists of two independent ferromagnetic layers, namely reference layer and free layer, separated by a barrier oxide layer such as magnesium oxide (MgO). The Parallel (P) and Anti-Parallel (AP) orientation of the free layer compared to the reference layer puts the STT-MRAM cell in a low and high resistance state, respectively. These two resistance states are used to represent either '1' or '0' logic values [13].

To read the stored value in an STT-MRAM cell, the WL is activated, and a low sensing current is applied through the cell to measure the resistance of the MTJ. If the sensing current is higher than the reference current, the MTJ is in a low resistance state, and the content of the cell is considered to be '0'. Otherwise, the MTJ is in a high resistance state and the cell is regarded as '1'. During the write operation, a current greater than the critical switching current is passed through the bit-cell. Writing '0' or '1' is determined by the direction of the applied switching current between the Bit Line (BL) and Source Line (SL) while keeping the Word Line (WL) activated.

The major failure mechanisms of STT-MRAM cells are retention failure, read disturbance, and write failure [14],[15],[16]. A retention failure occurs when an idle cell flips without applying any intentional excitation source. The retention time of an MTJ is defined as the expected time until a data retention failure happens. The retention failure probability (P_{RF}) for a given time period (t) is calculated by Equation (1) [14]:

$$P_{RF} = 1 - exp(\frac{-t}{e^{\Delta}}) \tag{1}$$

where Δ is the thermal stability factor and is measured using the Equation (2) [14]:

$$\Delta = \frac{E_b}{K \times T} \tag{2}$$

During a read operation in an STT-MRAM cell, the content of a cell can change unintentionally due to read disturbance. The occurrence probability of a read disturbance in an STT-MRAM cell is calculated according to Equation (3) [17]:

$$P_{RD}(t_r) = 1 - exp(\frac{-t_r}{\tau} \times exp(\frac{-\Delta(I_{C_0} - I_r)}{I_{C_0}})) \tag{3}$$

where Δ is the thermal stability factor, τ is the attempt period, I_r is the read current, I_{C_0} is the critical MTJ switching current at 0°K, and t_r is the read pulse width.

Write failure as another reliability concern occurs when the magnetic field direction of the free layer in an MTJ is not switched during the time that the write pulse is being applied. Equation (4) shows the probability of a write failure for an STT-MRAM cell [18]:

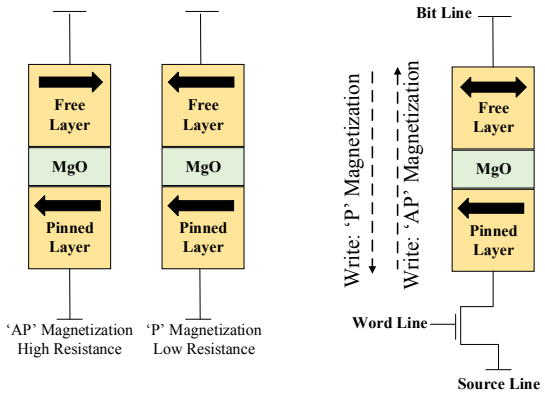


Fig. 1: Typical STT-MRAM bit-cell structure.

$$P_{WF}(t_w) = exp(-t_w \times \frac{2\mu_B p(I_w - I_{C_0})}{(c + ln(\Pi^2 \frac{\Delta}{4})) \times (em(1+p^2))}) \quad (4)$$

where Δ is the thermal stability factor, I_{C_0} is the critical MTJ switching current at 0°K, c is the Euler constant, e is the magnitude of electron charge, m denotes the magnetic momentum of the free layer, p is the tunneling spin polarization, μ_B is the Bohr magneton, I_w is the write current, and t_w is the write pulse width.

The temperature rise significantly increases the rates of the three types of errors mentioned above. Increasing the temperature reduces the thermal stability factor (Δ) of STT-MRAM cells, which results in an exponential increase in read disturbance and retention failure rates. Similarly, write current decreases in higher temperatures, which results in a higher write failure rate [7]. Accordingly, the high impact of temperature fluctuations on failure rates of STT-MRAM cells necessitates an efficient temperature management approach.

2.2 Learning

The ability to learn from the environment and improve performance through learning [19] is a primary and crucial property of a neural network. By tuning the inter-unit connection weights or strengths obtained by adapting to or learning from a set of training patterns [20], the network functionality is realized. Therefore, the quality of the learning process would have a significant impact on the functional accuracy of a neural network.

ML allows automatically detecting patterns in data and using these patterns to make decisions or predictions during the run-time [21]. ML approaches are traditionally divided into three broad categories: *Supervised learning* is the task of inferring a classification or regression from labeled training data, *Unsupervised learning* is the task of drawing inferences from data sets consisting of input data without labeled responses, and *RL* is the task of learning how agents ought to take sequences of actions in an environment to maximize cumulative rewards [22].

RL is a class of ML models in which the learning process is done through evaluative feedbacks without any supervised signals. As opposed to supervised learning, RL does not have direct access to the data labels. The labels for RL can be thought of as being dynamically generated and updated until convergence is achieved. The agent is placed in the training environment and is allowed to take action to explore the environment. With every action taken, a reward is given to the agent based on a user-defined goal. The reward quantifies the underlying goal; if the agent takes an action in accordance with the goal, the reward is higher and vice versa. The aim of RL is to learn a control policy that predicts actions that maximize the long-term rewards.

An RL problem can be formalized as a discrete time stochastic control process where an agent interacts with its environment. The agent starts, in a given state within its environment $s_0 \in S$, gathering an initial observation $\omega_0 \in \Omega$. At each time step t, the agent has to take an action $a_t \in A$. As illustrated in Fig. 2, it follows three sequences: (i) the agent obtains a reward $r(t) \in R$, (ii) the state transitions to $s(t+1) \in S$, and (iii) the agent obtains an observation $\omega_{t+1} \in \Omega$. This control setting was first proposed by [23] and later extended to learning by [24].

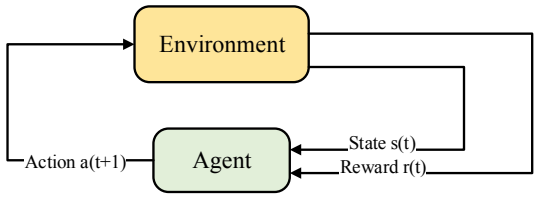


Fig. 2: Agent-Environment interaction in RL

An RL agent includes one or more of the following components:

- A representation of a value function that provides a prediction of how good each state or each state/action pair is,
- A direct representation of a policy to define how an agent selects actions, $\pi(s)$ or $\pi(s,a)$, or
- A model of the environment (the estimated transition function and the estimated reward function) in conjunction with a planning algorithm.

The first two components are related to what is called model-free RL. When the latter component is used, the algorithm is referred to as model-based RL.

3 Motivation

As mentioned earlier, the major source of heat generation and increase in temperature in STT-MRAM cache memories are write operations [8], [9]. If the generated heat is not evenly distributed all over the chip, some points of the chip will have a higher temperature than others. So, uneven heat distribution makes some points of the cache memory vulnerable to reliability challenges.

In the following, we will see how different workloads can cause high temperature in the L2 cache in case study applications. To this end, we use a combination of applications from *SPEC CPU2006* benchmark suite [12] as our case study application. The application includes `cactusADM`, `gcc`, `perlbench`, and `sjeng`. `cactusADM` is a combination of Cactus, an open-source problem-solving environment, and BenchADM, a computational kernel representative of many applications in numerical relativity. `gcc` generates code for an AMD Opteron processor. The benchmark runs as a compiler with many of its optimization flags enabled. `perlbench` is a cut-down version of Perl v5.8.7, the popular scripting language. `sjeng` is based on Sjeng 11.2, which is a program that plays chess and several chess variants and attempts to find the best move via a combination of alpha-beta or priority proof number tree searches, advanced move ordering, positional evaluation, and heuristic forward pruning.

We use the gem5 simulator [11] to simulate our case study application on a system with four cores. We consider an initial temperature of 84.85°c for each cache block [25]. Fig. 3 shows the comparison of temperature rate in four replacement policies, i.e., LRU, First In First Out (FIFO), Most Recently Used (MRU), and Random.

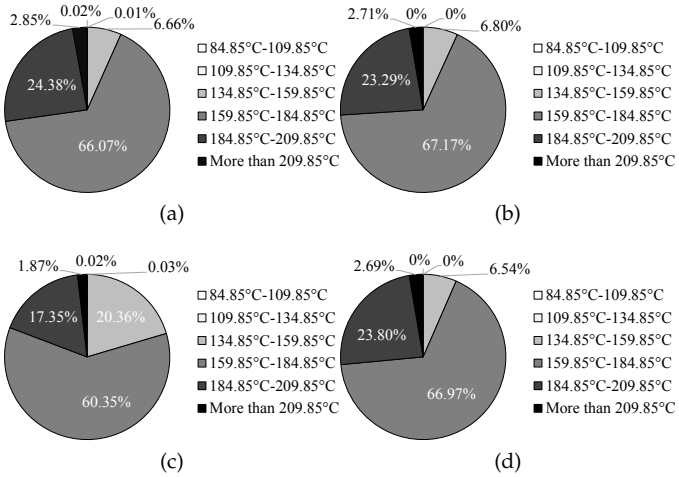


Fig. 3: Comparison of temperature rate in different replacement policies: (a) LRU, (b) FIFO, (c) MRU, and (d) Random.

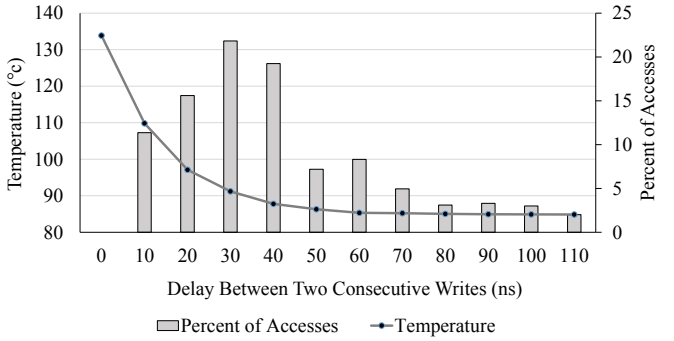


Fig. 4: The relation between 32nm STT-MRAM cell temperature rise and the delay between two consecutive writes as long as the percent of write accesses with different delays.

As can be seen from Fig. 3.(a), after running the application with LRU, about 6.66% of blocks have a temperature range of 134.85-159.85°c, 66.07% of blocks are in the range of 159.85-184.85°c, 24.38% of blocks have a temperature range of 184.85-209.85°c, and 2.85% of blocks have a temperature more than 209°c. For FIFO, as it is depicted in Fig. 3.(b), 6.8% of blocks have maximum temperature in the range of 134.85-159.85°c. The 159.85-184.85°c temperature range consists 67.17% of cache blocks. The temperature of 23.29% of blocks is in the range of 184.85-209.85°c. 2.71% of blocks have a temperature of more than 209°c.

Fig. 3.(c) shows that having MRU as the replacement policy causes to have 20.36% of cache blocks in a temperature range of 134.85-159.85°c. 60.35% of cache blocks have a maximum temperature between 159.85°c and 184.85°c. The range 184.85-209.85°c consists of 17.35% of blocks while 1.87% of blocks have a maximum temperature of more than 209°c. For the Random replacement policy, Fig. 3 shows that 6.54% of cache blocks have a temperature between 134.85°c and 159.85°c and the temperature of 66.97% of cache blocks is in the range of 159.85-184.85°c. The range 184.85-209.85°c consists of 23.8% of cache blocks while 2.69% of cache blocks have a temperature of more than 209.85°c.

As can be seen, the percent of cache blocks that have a temperature in a specific range is different for different replacement policies. These observations confirm that one of the main factors in write distribution non-uniformity is the decisions made by cache replacement policy, as they determine into which cache block the incoming data should be written.

4 WiSE IN DETAILS

This section describes how WiSE distributes the generated heat in an STT-MRAM cache. To this end, WiSE benefits from an RL-based approach that tries to distribute write operations all over the cache memory banks and manage the generated heat to improve STT-MRAM reliability challenges. In the following, first, we explain how we map our thermal management problem to an RL-based problem at the first sub-section. Then, in the second sub-section, we will introduce the WiSE replacement policy in detail.

4.1 How to Utilize RL in WiSE

Increasing chip temperature can have significant adverse effects on the performance, reliability, power consumption, and lifetime of the chip. Some destructive mechanisms such as electron migration, stress migration, and dielectric breakdown are temperature-dependent and will speed up with increase in temperature. Increasing the temperature also decreases the reliability of an STT-MRAM chip. If the generated heat accumulates in a specific part of the chip, that part becomes a hot-spot and can cause read disturbance (see Equation (3)), write failure (see Equation (4)), or retention failure (see Equation (1)). So, controlling the generated heat plays an important role in improving the performance, reliability, and energy consumption of the chip.

As mentioned earlier, one of the main sources of heat generation in STT-MRAM memory cells is the write operation. Consecutive writes significantly increase the temperature of a cell and treat its reliability. To explore the generated heat after the write operation, we used the MTJ cell model proposed in [25]. Accordingly, we considered a set of workloads from the SPEC CPU 2006 benchmark suite [12][1]. Fig. 4 depicts the temperature of the cell (left vertical axis) and access distribution (right vertical axis) based on the delay between two consecutive writes. As can be seen in the figure, the longer the delay between two consecutive writes, the less heat is generated.

Fig. 5 shows the temperature rise for different write latencies at different technology node sizes in two scenarios. In the first scenario, a technology node size-independent voltage (current) is applied to the STT-MRAM cell during the write operation. In this scenario, we just change the technology node size of the MTJ cell. In the latter scenario, we apply different write pulse voltages (same as technology node size introduced Vdd) during the write operation. Fig. 5 reveals that the most critical contributor in heat generation of STT-MRAM cell during the write operation is its applied write voltage (current). As shown in this figure, when the write voltage is fixed at 0.7v, the amount of generated heat

1. The details of the benchmarks and simulation settings can be found in Table 1 and Table 2

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3163438, IEEE Transactions on Emerging Topics in Computing

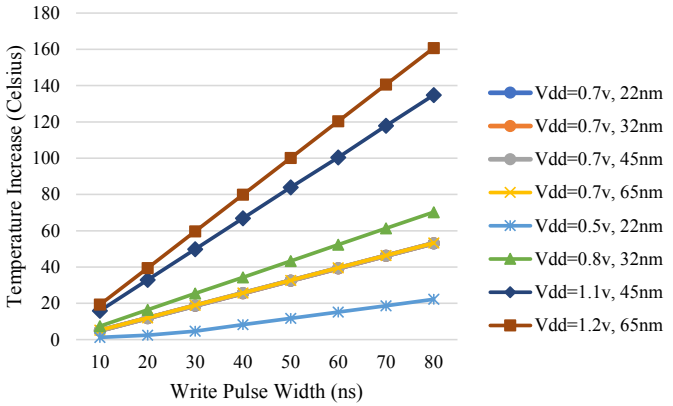


Fig. 5: The amount of temperature increase for STT-MRAM write operation at different technology node sizes.

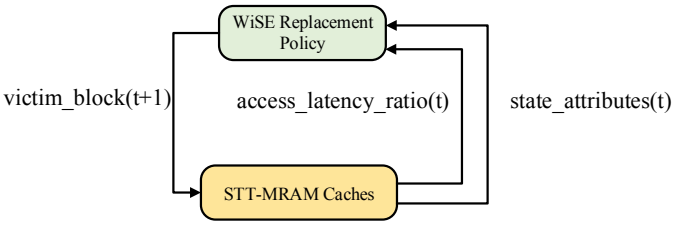


Fig. 6: The RL model used in WiSE-enabled cache.

during the write operations is not noticeably different for various technology node sizes. On the other hand, when we change the write pulse voltage (current) as the technology node size varies, the amount of generated heat changes significantly. Considering the second scenario, the temperature rise due to the write operation in the STT-MRAM cell becomes smaller as the technology size becomes smaller. It should be noted that the primary cell's temperature was 84.85°c (358°K) during the experiments.

Since the write operation is performed in different cells, the cell in which the write operation is performed remains idle for a certain period. If there is no write request to that cell, the temperature of that cell will decrease. Indeed, the cell in which the write operation is not performed will lose its heat and cool down over time. Equation (5) shows the relation between time and cooling rate for an STT-MRAM cell [26].

$$\frac{T - T_\infty}{T_i - T_\infty} = e^{\frac{-t}{\tau}} \tag{5}$$

where T is the temperature at any moment, T_i is initial temperature, T_∞ is final temperature, t is time, and τ is thermal time constant.

In WiSE, we aim to increase the idle time of each STT-MRAM cache cell as much as possible with the hope of decreasing the cell temperature. Accordingly, if we consider Figure 2 model for a system that utilizes the RL management strategy, we should first fit the model to our problem. Figure 6 depicts the RL model that we use in the WiSE-enabled cache. The environment in our RL model is the STT-MRAM cache and our agent is the WiSE replacement policy. As can be seen in Figure 6, WiSE RL actions are the

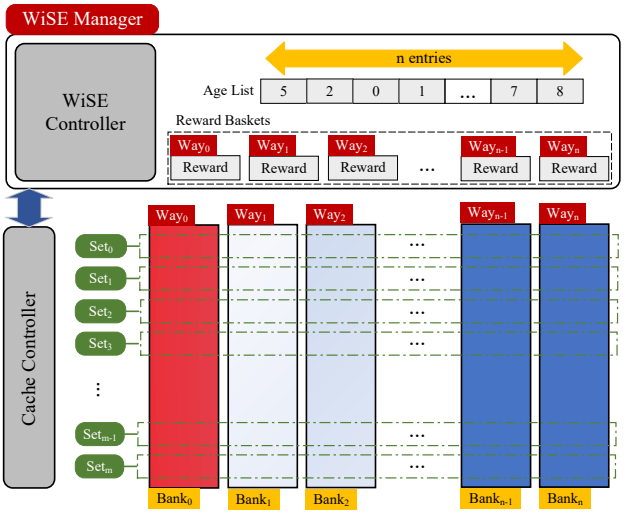


Fig. 7: An abstract view of different parts of a system equipped with WiSE approach.

WiSE replacement policy decisions on the victim blocks. For WiSE RL rewards, different strategies can be considered. For example, we can consider a positive reward of '1' whenever the incoming write request is written at the most relaxed memory bank recently, and a negative reward of '-1' for the time that the incoming write request is written at the most recently used memory bank. Finally, for the WiSE RL state attributes, we consider access latency ratios of different WiSE-enabled cache memory banks.

4.2 WiSE-enabled Cache Architecture

Fig. 7 depicts the WiSE-enabled cache architecture. As can be seen in Fig. 7, compared with the traditional n-way set associative STT-MRAM-based cache, the WiSE-enabled cache benefits from a *WiSE Manager* module which is resided next to an ordinary cache controller. Generally, *WiSE Manager* is responsible for alleviating the hot spots in STT-MRAM memory banks which are assigned to the cache ways and increase the reliability of STT-MRAM cache according to the facts (and formulas) which are mentioned in Section 2.

It is illustrated in Fig. 7 that *WiSE Manager* itself consists of three main parts, i.e., *WiSE Controller*, *Age List*, and *Reward Baskets*. In the following, we will explain the details of each part.

- **WiSE Controller:** Generally, it is responsible for implementing Algorithm 1. By executing Algorithm 1, the *WiSE Manager* can efficiently track the temperature rise in the cache's ways and force the traditional replacement policies (like LRU) which are executed in the cache controller to evict a different block to alleviate the way's temperature.
- **Age List:** It is a *WiSE Manager*-provided space that tracks the order of write operations in the ways. It is much like the age bit concept, which is traditionally used in LRU replacement policy to monitor the block usage history in each set of an n-way set-associative cache. However, the *Age List* focuses on the write operations in the ways (memory banks). The *Age*

List is directly updated by the *WiSE controller*. The leftmost section in the *Age List* is the least frequently written way and the rightmost section in it is the most frequently written way. The bit-width and the number of sections in the *Age List* depend on the number of ways in the cache.

- **Reward Baskets:** It is a *WiSE Manager*-provided space that is used to save the reward of each way. It is computed according to Algorithm 1. The number of baskets in the reward basket depends on the number of cache ways. The bit-width of each basket can be computed by Equation (6).

$$RB_{bit-width} = \log_2\lceil(PositiveMaxValue - NegativeMaxValue + 1\rceil \quad (6)$$

4.3 WiSE Replacement Policy

This section describes the WiSE replacement policy. It is important to remember that the physical structure of the n-way set associative cache consists of several memory banks. Indeed, there is a relationship between the number of cache ways and the number of memory banks. In most cases, for an n-way set associative cache, we have an n-multiple number of memory banks.

As we mentioned earlier, the write operations in the STT-MRAM cache memories are the main contributor in generating the heat across the STT-MRAM cells. The write operations in cache memories are requested in two scenarios. In the first scenario, a write operation is issued when a cache block that is already found in the cache is updated. In the second scenario, when a block is requested in a cache and it is not found in it, this block should be taken from the lower-level (nearer to main memory) memory module in hierarchical memory system infrastructure.

In the second scenario, the cache replacement policy is the main contributor to placing the new block in a specific position of the cache memory and issues the write operation on that position. Indeed, here when a miss occurs, a new block must be written to cache. The replacement policy chooses a block as a victim block and replaces it with the new one. Here in the WiSE replacement policy, we aim to control these kinds of write requests in a way that the generated heat from them would be as low as possible. To this end, the WiSE replacement policy chooses the victim block from the memory banks that experience fewer write operations recently.

Algorithm 1 outlines the WiSE management policy. Algorithm 1 is called each time a read or write request arrives at the WiSE-equipped cache memory. As can be seen in Algorithm 1, at line 1, first we check if we can find the requested block on the cache (hit) or not (miss). Then, at line 2, if the request is a read and it was a hit in the cache, there is no need to perform any action since the read operation on the STT-MRAM cache is not challenging (compared to the write operation). In line 5, if the request is a write (write-back) and it was a hit in the cache, we should update the reward baskets that are assigned to each way. To this end, first, the way number that the hit block resides in is retrieved at line 6.

Algorithm 1: WiSE Algorithm

Input: readRequest, writeRequest, blockAddress, RewardBaskets, ageList, nextVictimWay, maxWayNumber
Output: Updating rewardBaskets and introducing the victimBlock if applicable

```
1  found = (!foundSet(blockAddress) and
     !foundWay(blockAddress));
2  if (found == 1) and readRequest then
3  |  return null
4  end
5  if (found == 1) and writeRequest then
6  |  n = foundWay(blockAddress));
7  |  if n == ageList[maxWayNumber] then
8  |  |  if RewardBasket_n == PositiveMaxValue then
9  |  |  |  RewardBasket_n = 0;
10 |  |  |                    ▷ Clearing the good history!
11 |  |  else
12 |  |  |  RewardBasket_n+ = 1;
13 |  |  end
14 |  |                ▷ Write hit in least recently written way
15 |  end
16 |  if n == ageList[0] then
17 |  |  if RewardBasket_n == NegativeMaxValue then
18 |  |  |  nextVictimWay = n;
19 |  |  |     ▷ Introducing the next victim block in advance!
   |  |  |  RewardBasket_n = 0;
20 |  |  |                    ▷ Clearing the bad history!
21 |  |  else
22 |  |  |  RewardBasket_n− = 1;
23 |  |  end
24 |  |                ▷ Write hit in most recently written way
25 |  end
26 |  ageList[0] = n;
27 |  return null
28 end
29 if found ≠ 1 then
30 |                              ▷ Miss in read or write request
31 |  if nextVictimWay ≠ null then
32 |  |  set = foundSet(blockAddress);
33 |  |  victimBlock = set.block[nextVictimWay];
34 |  |  ageList[0] = nextVictimWay;
35 |  |  return victimBlock
36 |  else
37 |  |  LRU(blockAddress);
38 |  |  ageList[0] = foundWay(blockAddress));
39 |  end
40 end
```

As illustrated in Fig. 7, we consider a reward basket for each way in WiSE-equipped cache memory. The most significant bit in each way's basket is considered as a sign bit since we can have positive or negative rewards for different write operations in the ways. For example, a five bits way's reward basket can contain numbers from -15 (*NegativeMaxValue*) to +15 (*PositiveMaxValue*). For the situations that the write request hits on the coolest way (lines 7 to 15), we will increase the corresponding way's reward basket by one unless its value does not become above *PositiveMaxValue*. As can be seen in lines 8 to 13, if the target way's reward basket already contains *PositiveMaxValue* before this write hit request, first we clear the target way's basket to zero and then increase it by 1.

For the situations that the write request hits on the hottest way (lines 16 to 25), we will decrease the corresponding way's reward basket by one unless its value does not become less than *NegativeMaxValue*. As can be seen in lines 17 to 23, if the target way's reward basket already contains *NegativeMaxValue* before this write hit request, we

will consider the target way as the next victim way to replace and then clear the target way's basket to zero and decrease it by 1.

If the request is a read or a write and it was missed in the cache (line 29), first we check if *NextVictimWay* is null or not. If we have a victim way, then the replacement will be done in the corresponding set of that way (lines 32 to 33). Then, the *ageList[]* will be updated (line 34) and the victim way comes to the first place in this array which means that this way is the most recently used. If we have no victim way (*NextVictimWay* is null), the algorithm uses LRU replacement policy to find the victim block for replacing and updates the *ageList[]* accordingly (lines 37 to 39).

5 Simulation System Setup and Results

In this section, first, we introduce the system setup used to explore the efficiency of WiSE. Then, we will explore the experimental results of this study.

5.1 Simulation System Setup

We applied WiSE to a shared L2 cache of a 4-core ARM processor simulated in gem5 [11]. Although there is no limitation in WiSE design to be applied to any cache level in the memory hierarchy, we applied it to the L2 cache for illustration. Several past studies on adopting STT-MRAM in on-chip memories, such as SPM (Scratchpad Memories) and cache, declared that employing STT-MRAM is only practical for L2 and lower-level on-chip memories [27], [28], this is because STT-MRAM has limitations on dynamic energy consumption, performance, and endurance that make it more suitable for L2 cache and lower levels of on-chip memories.

To calculate the temperature of the STT-MRAM cache at different timestamps during the run-time, we model the temperature of STT-MRAM cache block at three levels. The first level calculates the temperature rise due to applying the write pulse to the STT-MRAM cell. At this level, we use the HSPICE model in [25]. The initial temperature of the STT-MRAM cell introduced in this model was 84.85°c. This model contains a Landau-Lifshitz-Gilbert (LLG) equation solver and computes the final temperature based on the STT-MRAM physics informed model and its applied voltage. The parameters and variables included in the STT-MRAM circuit-level simulation are shown in Table 3. Please note that some parameters of modeled STT-MRAM cells like Tunnel Magneto-Resistance (TMR) and critical switching current are temperature-dependent and vary by changing the temperature of the STT-MRAM cell. The values and dynamic behaviour of these kinds of parameters are efficiently modeled at [25] and the interested readers can refer to [25] for more information. Then, at the second level, we model an STT-MRAM cache at the gem5 simulator and associate a temperature to each STT-MRAM block in the L2 cache. When each write operation is performed across the STT-MRAM block, we utilize the information taken from the HSPICE model and update the temperature of the block. On the other hand, when the STT-MRAM blocks are idle across the STT-MRAM cache, based on Equation (5), we update the temperature of each block as time passes.

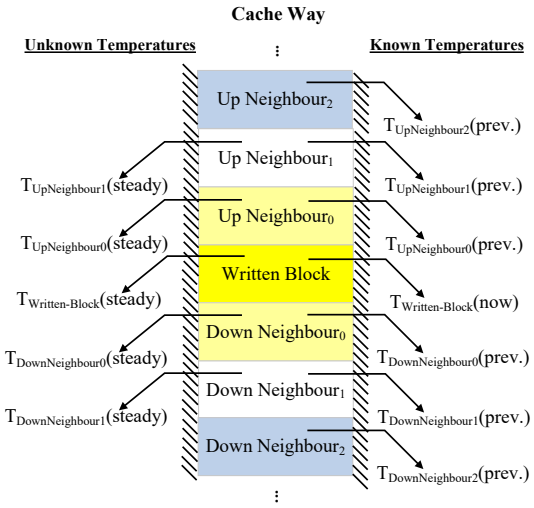


Fig. 8: The heat dissipation model considered in the experiments.

Algorithm 2: Heat dissipation modeling.

Input: $T_{WB(now)}$, $T_{UN0(prev.)}$, $T_{UN1(prev.)}$, $T_{UN2(prev.)}$, $T_{DN0(prev.)}$, $T_{DN1(prev.)}$, $T_{DN2(prev.)}$
Output: $T_{WB(steady)}$, $T_{UN0(steady)}$, $T_{UN1(steady)}$, $T_{DN0(steady)}$, $T_{DN1(steady)}$

1 $T_{WB(temp)} = T_{UN0(temp)} = Average(T_{WB(now)} + T_{UN0(prev.)})$;
2 $T_{UN0(steady)} = T_{UN1(temp)} = Average(T_{UN0(temp)} + T_{UN1(prev.)})$;
3 $T_{UN1(steady)} = Average(T_{UN1(temp)} + T_{UN2(prev.)})$;
4 $T_{WB(steady)} = T_{DN0(temp)} = Average(T_{WB(temp)} + T_{DN0(prev.)})$;
5 $T_{DN0(steady)} = T_{DN1(temp)} = Average(T_{DN0(temp)} + T_{DN1(prev.)})$;
6 $T_{DN1(steady)} = Average(T_{DN1(temp)} + T_{DN2(prev.)})$;

At the third level, we model heat dissipation across the STT-MRAM blocks. While none of the previous studies consider the heat dissipation effects of writing an STT-MRAM block on its neighbours, here we try to model this phenomenon based on some basic concepts of heat transfer knowledge. To this end, we assume that the heat dissipation of STT-MRAM write operation affects at most only affects its two consecutive neighbour blocks at its upside and downside. Since we consider the blocks of cache sets at each way are manufactured in unique banks (chips) (see Fig. 7), we do not consider left and right sides neighbours for cache blocks. Accordingly, the heat dissipation due to the write operation in an STT-MRAM cache block affects the neighbour blocks according to the following scenario depicted in Fig. 8.

There are two sets of temperatures, i.e., known temperatures and unknown temperatures, as shown in Fig. 8. Based on our assumptions, we assume that each neighbour can affect its neighbour temperature up to their average temperatures. In this regard, the unknown temperatures can be calculated by Algorithm 2.

Note that the temperature variables are defined at three moments in Algorithm 2, i.e., previous, now, and steady times. *WB* stands for Written Block, *UN* stands for Up Neighbour, and *DN* stands for Down Neighbour. After each

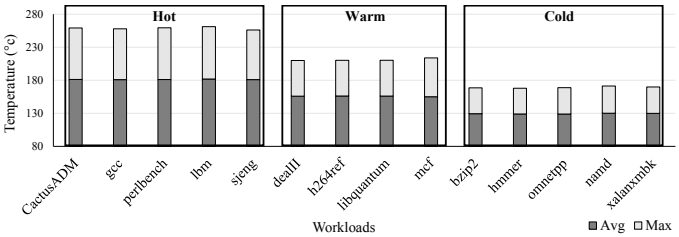


Fig. 9: Classification of benchmarks based on their generated temperature in the L2 cache

TABLE 1: gem5 settings

Parameter	Value	Parameter	Value
ISA	ARMv7-A	L1 Size,Assoc	32KB, 4
No. of Cores	4	L2 Size,Assoc	4MB, 8
Cache Config	L1 (Private) L2 (Shared, WiSE-enabled)	Cache Block Size	64B
L1 Write Latency	2 Cycle	L2 Write Latency	40 Cycle
L1 Read Latency	2 Cycle	L2 Read Latency	20 Cycle

write operation during the experiments, we update the temperature of all the affected blocks by considering the mentioned three levels of temperature modeling.

The details of the simulation environment are shown in Table 1. NVSim [29] is used to retrieve the latencies, energy consumption, and area of a 4MB STT-MRAM based L2 cache. Eventually, to evaluate the area and power consumption overheads of the peripheral circuits in the WiSE controller, we used Synopsys Design Compiler® with $45nm$ technology library.

From the workload perspective, we used a set of benchmarks from the SPEC CPU2006 benchmark suite [12] to evaluate WiSE. To better show the efficiency of the WiSE at different applications, we classified the benchmarks based on the maximum and average temperature they generated in the L2 cache. We intuitively categorized all the benchmarks into three groups,i.e., hot, warm, and cold, depicted in Fig. 9. As can be seen in Fig. 9, we have five benchmarks in the hot group whose peak temperatures in the L2 cache are above 250°c. On the other hand, we have four and five benchmarks in warm and cold categories, respectively. The peak temperature of the L2 cache for benchmarks in the warm category is above 200°c while it is above 170°c in the cold category. Finally, we made 12 combinations of these benchmarks introduced in Table 2.

5.2 Simulation Results

In the following, we evaluate WiSE from two perspectives, i.e., reliability and energy consumption. Also, we discuss the performance and area overheads of WiSE. For the sake of comparison, we implemented and compared the following replacement policies:

- **LRU:** The victim block is always the one that has been used less frequently than the others in the cache target set, recently.
- **FIFO:** The evicted block is the one that comes first in the cache target set in comparison with the others.
- **MRU:** The evicted block is the one which was the last one accessed in the cache target set.

TABLE 2: SPEC CPU2006 benchmarks combinations as multi-programmed workloads in the evaluation of a quad-core processor

Workloads	Benchmarks			
	Core1	Core2	Core3	Core4
Hot1	cactusADM	gcc	perlbench	sjeng
Hot2	cactusADM	gcc	perlbench	lbm
Hot3	cactusADM	gcc	sjeng	lbm
Hot4	cactusADM	perlbench	sjeng	lbm
Warm1	dealII	lbm	h264ref	libquantum
Warm2	dealII	lbm	h264ref	mcf
Warm3	dealII	lbm	libquantum	mcf
Warm4	dealII	h264ref	libquantum	mcf
Cold1	bzip2	mcf	hmmer	omnetpp
Cold2	bzip2	mcf	hmmer	namd
Cold3	bzip2	soplex	xalancmbk	mcf
Cold4	namd	mcf	hmmer	omnetpp

TABLE 3: Parameters and variables included in MTJ simulation

Parameter and Variable	Description	Default Value
α	Gilbert Damping Coefficient	0.0062
MS0	Saturation Magnetization at 0°K	1210
P0	Polarization Factor at 0°K	0.69
γ	Gyro-Magnetic Constant	$1.76\times10^{11}s^{-1}T^{-1}$
Ts	Free Layer Thickness	2.44nm
tmp0	Initial Temperature	358°k
RA	Resistance-Area Product	5
F^2	Feature Size	45nm

- **Random:** The evicted block is randomly selected among the target set, regardless of the usage history of the blocks.
- **TA-LRW:** The victim block is always the one that recently has been written less frequently than the others in the cache target set.

It is worth mentioning that the WiSE approach is completely orthogonal with the ECC technique. Indeed, the WiSE approach is categorized among the fault avoidance techniques while the ECC technique is categorized among fault tolerance techniques. Thus, the operational phases of the WiSE and ECC techniques do not interfere with each other.

5.2.1 Reliability

As we have mentioned earlier, most STT-MRAM reliability challenges are temperature-dependent. An increase in temperature causes a reduction in thermal stability factor, Δ that increases the probability of read disturbance and retention failure. Temperature rise also reduces the write current that results in increasing the write failure rate.

WiSE tries to reduce the chip temperature by increasing the delay between two consecutive writes. Fig. 10 shows the maximum temperature of L2 cache blocks in different replacement policies for 12 hot, warm, and normal combinations according to Table 2. As it is clear from Fig. 10, WiSE keeps the L2 cache at a lower temperature compared to other replacement policies. For the hot combinations, Fig 10.(a) shows that on average, WiSE keeps the maximum temperature of 15% of blocks in the range of 109.85-134.85°c, 65.21% of blocks in the range of 134.85-159.85°c, 17.58% of blocks in the range of 159.85-184.85°c, 1.73% of blocks in the range of 184.85-209.85°c, and 0.27% of blocks in the range of more than 209.85°c. On average, WiSE decreases

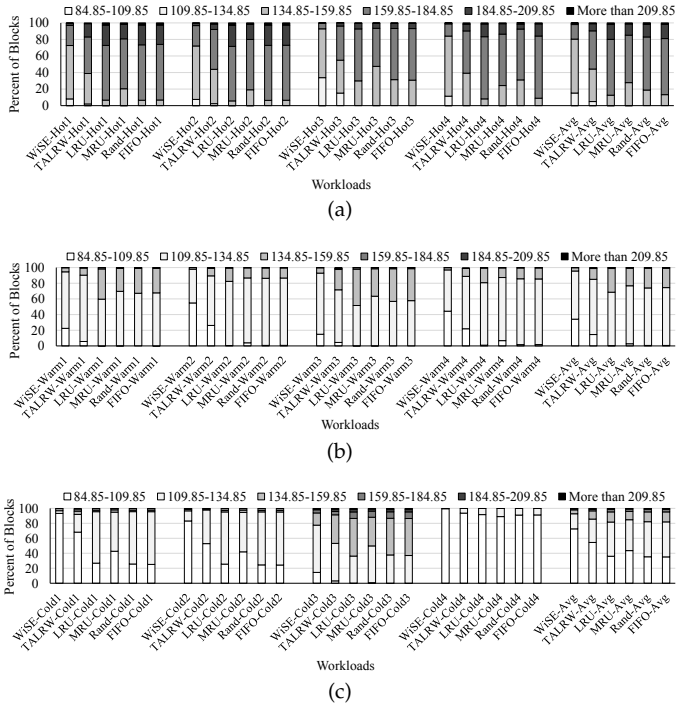


Fig. 10: Maximum temperature of L2 cache with different replacement policies for combinations: (a) hot, (b) warm, and (c) cold.

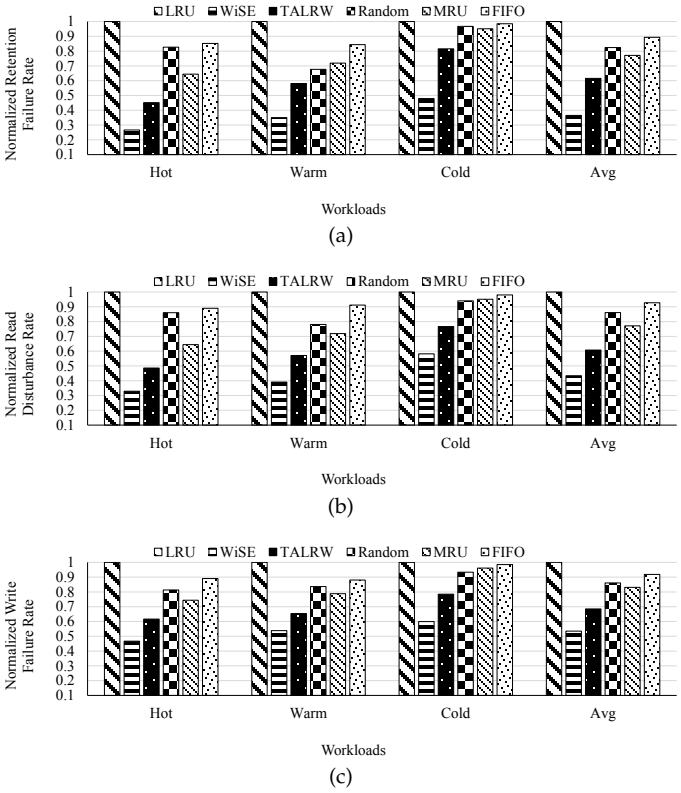


Fig. 11: Comparison of failure rates for different configurations, normalized to LRU: (a) Retention failure rate, (b) Read disturbance rate, and (c) Write failure rate.

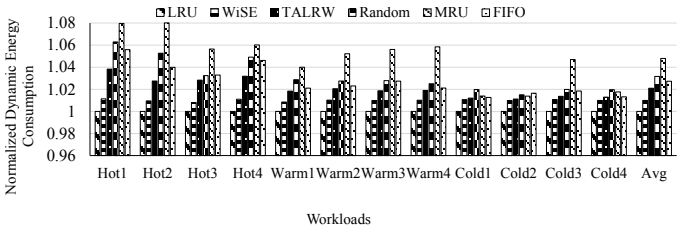


Fig. 12: Dynamic energy consumption of L2 cache, normalized to LRU.

the maximum temperature by 26.36°c over LRU. This temperature reduction over TA-LRW, MRU, Random, and FIFO is 16.55°c, 18.43°c, 22.82°c, and 24.13°c, respectively.

For the warm category, Fig. 10.(b) shows that WiSE keeps the temperature of 34.14% of blocks in the range of 84.85-109.85°c, 61.45% of blocks in the range of 109.85-134.85°c, 4.29% of blocks in the range of 134.85-159.85°c, 0.11% of blocks in the range of 159.85-184.85°c, 0.003% of blocks in the range of 184.85-209.85°c, and 0% of blocks in the range of more than 209.85°c. In this category, WiSE decreases the maximum temperature by an average of 10.36°c, 16.74°c, 12°c, 11.05°c, and 14.32°c over TA-LRW, LRU, MRU, Random, and FIFO, respectively.

Considering the cold benchmarks, Fig. 10.(c) shows that WiSE keeps the temperature of 72.64% of blocks in the range of 84.85-109.85°c, 20.14% of blocks in the range of 109.85-134.85°c, 5.24% of blocks in the range of 134.85-159.85°c, 1.5% of blocks in the range of 159.85-184.85°c, and 0.46% of blocks in the range of 184.85-209.85°c. On average, WiSE decreases the maximum temperature by 10.88°c over LRU. This temperature reduction over TA-LRW, MRU, Random, and FIFO is 9.37°c, 10.16°c, 12.47°c, and 14.5°c, respectively.

Fig. 11 shows the comparison of reliability for different configurations normalized to LRU. As can be seen in Fig. 11, the temperature decrease delivered by WiSE noticeably alleviated the reliability concerns of the STT-MRAM cache. It is illustrated in Fig. 11.(a) that WiSE decreases the retention failure rate on average by 64% compared to LRU. On the other hand, compared to LRU, WiSE decreases the read disturbance rate and write failure rate by 57% (Fig. 11.(b)) and 47% (Fig. 11.(c)), respectively.

5.2.2 Energy Consumption

Fig. 12 shows the dynamic energy consumption of the L2 cache for WiSE and other replacement policies. As can be seen, on average, the dynamic energy consumption of WiSE is about near the LRU and is lower than Random, MRU, and FIFO. According to Fig. 12, WiSE increases the energy consumption by about 1.6% compared to LRU. This is because an l2 cache equipped with WiSE always uses LRU for replacing a cache block and every time that the reward basket of a way reaches the $NegativeMaxValue$, it uses WiSE replacement policy to evict a block from this way with the hope that the new incoming block to the target way does not have the same intensive frequent write behavior. Accordingly, the dynamic energy consumption of the WiSE-equipped cache is much like the LRU-quipped cache.

From the static energy consumption perspective, WiSE-equipped cache benefits from the modules depicted in Fig. 7. These modules and their corresponding circuits impose

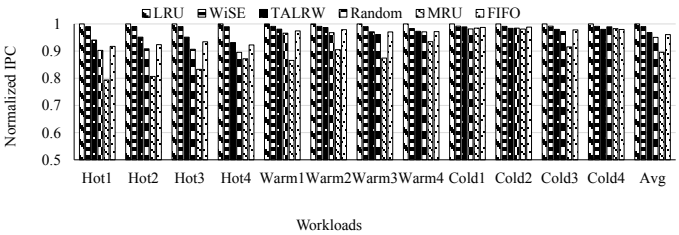


Fig. 13: IPC, normalized to LRU.

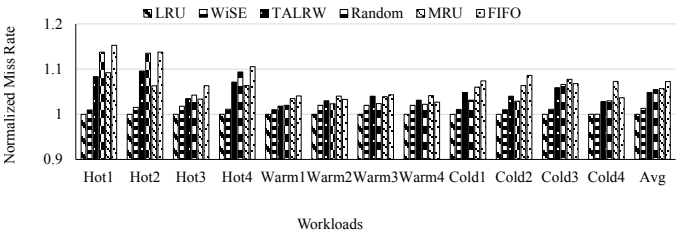


Fig. 14: Miss rate of L2 cache, normalized to LRU.

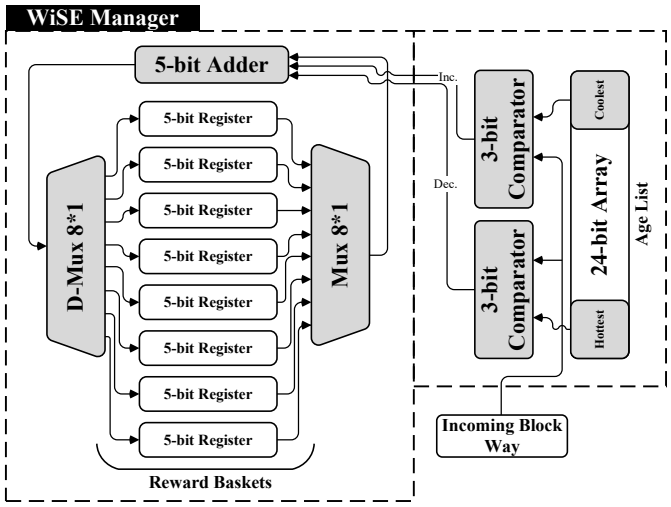


Fig. 15: The peripheral circuit of WiSE.

static energy consumption on the design. We have synthesized these circuits in Synopsys Design Compiler® with $45nm$ technology library. The synthesis results show that WiSE only imposes 0.64 percent static energy overhead compared to an LRU-equipped L2 cache.

5.2.3 Performance

We have chosen Instruction Per Cycle (IPC) as the metric for showing system performance. Fig. 13 shows the IPC overhead after applying WiSE normalized to the LRU (the highest performance scenario among the other replacement policies). As can be seen, the IPC after applying WiSE is higher than MRU, Random, and FIFO and is about near to the LRU. WiSE decreases the IPC by about 1% compared to the LRU, that is negligible.

As the second metric that represents the system performance, we have considered the miss rate of the L2 cache. Fig. 14 illustrates the miss rate of L2 cache for different replacement policies. According to Fig. 14, on average, applying WiSE only increases the miss rate of L2 cache by about 1.3% compared to the LRU, that is lower than Random, MRU, and FIFO.

5.2.4 Area

Fig. 15 shows the detailed modules of peripheral circuits that we added to the conventional architecture of an 8-way L2 cache as WiSE Manager (see Fig. 7). Note that these peripheral circuits are designed to operate with an 8-way L2 cache that we considered in our simulations. By changing the configuration of the caches, the design depicted in Fig. 15 will be modified slightly, but the concept behind the design will not change. As can be seen from the right side of Fig. 15, a 24-bit width array is used as the age list. Each cell of this array contains a 3-bit number that shows a way number in an 8-way cache. Two 3-bit comparators are used to compare the hottest way and the coolest way (the first cell and the last cell of the age list) to the way associated with the incoming address. In the left side of Fig. 15, eight 5-bit registers are used as the reward basket of each way. A multiplexer is used to choose a reward basket and add a reward to it through a 5-bit adder. The result is added to the corresponding reward basket through a demultiplexer.

From the area perspective, as we mentioned previously, the circuits that we added to the traditional architecture of the L2 cache leads to a small area overhead. We have synthesized these circuits in Synopsys Design Compiler® with $45nm$ technology library. The total area of an STT-MRAM L2 cache obtained from NVSim was equal to 4.931 mm^2. The WiSE Manager peripheral circuits has an area equal to $9.794{\times}10^{-4} mm^2$. The synthesis results show that WiSE only imposes 0.02 percent area overhead in comparison with an LRU-equipped L2 cache.

6 Related Work

As we mentioned in Section 2, STT-MRAM suffers from three reliability challenges, i.e., retention failure, read disturbance, and write failure. There have been several studies to address these reliability challenges of STT-MRAM caches.

Some studies like [30], [31], and [32] use Error Correction Codes (ECCs) to reduce the write error rate in STT-MRAM caches. The authors in [30] propose a repair mechanism based on an SEC-DED code to increase the reliability of embedded non-volatile memories. In [31], a particular type of cache block called tolerable block is further identified among the faulty ones and ECCs are used to increase the number of tolerable blocks. ECCs impose energy, area, and performance overhead. This is because the majority of data bits that are written in cache blocks are zeros and do not need protection. ECCs are only needed when most of the incoming data bits are ones.

To overcome write failure of STT-MRAM caches, [33] and [34] have proposed increasing the amplitude of write current to overcome the write failure problem. In this circuit-level scheme, the width of the access transistor is augmented to improve the current driving ability of the access transistor. Consuming extra energy and enhancing the probability of oxide breakdown are side effects of this scheme.

Performing an additional read operation after each write operation is the method used in [7], [35]. In this method, when a write operation is done, the data of memory is read

and if the data is incorrect, the write operation is repeated. This process continues until the data is written correctly. These additional read operations increase the probability of read disturbance.

Several research efforts have been devoted to addressing the retention failure of STT-MRAM. ECCs are also used in some studies to reduce retention failure [14]. As mentioned earlier, ECCs impose energy, area, and performance overhead. Some researches increase the STT-MRAM cell retention time [36], [37]. This is done by changing the cell thermal stability factor or MTJ layer characteristics.

ECCs have been widely used to overcome read disturbance [38] which results in energy, area, and performance overheads. An additional write operation after each read operation is done in some researches to guarantee the correction of possible read error [39]. This causes an increase in the write failure rate. Reducing read current to reduce read disturbance is another method that has been used in previous studies [36]. So, accurate sense amplifiers are needed to read the cell content correctly.

Several research efforts have been devoted to taking advantage of RL to improve the reliability and energy consumption of STT-MRAM caches. The authors in [40] propose a method to reduce the number of refreshes needed to maintain cache data integrity. Leveraging the reduced cell area of STT-MRAM, they create a refresh confidence table to learn the interleaving access duration of the blocks through their access pattern and to decide if a block needs to be refreshed. Eliminating unnecessary refreshes helps to reduce dynamic energy as well as the memory access stalls required when the data is being refreshed. To overcome the large write latency and write energy of STT-MRAM-based embedded systems, [41] performs transfer learning (TL) on meta environments and RL on the last few layers of a deep convolutional network. While the STT-MRAM stores the meta-model from TL, an on-die SRAM stores the weights of the last few layers. Thus, all the real-time updates via RL are carried out on the SRAM arrays. The authors in [42] employ a customized RL algorithm to minimize the latency and power overhead caused by scrub operation in tailored STT-MRAM without incurring additional retention errors. [43] proposes a cache management method based on the RL that uses the write intensity and reuses information of cache access requests to design a cache allocation policy and optimize energy consumption. The key idea of this work is to use the RL algorithm to get the weight for the set allocating to SRAM or STT-MRAM by learning from the energy consumption of cache line sets. The algorithm allocates a cache line in a set to the region with greater weight.

To the best of our knowledge, except for [44], none of the previous studies have attempted to mitigate heat generation and high temperature of STT-MRAM cache memories using a replacement policy. The authors in [44] propose a cache replacement policy, so-called Thermal-Aware Least Recently Written (TA-LRW), to uniformly distribute the temperature and prevent heat accumulation. TA-LRW sorts the blocks based on their last write access and evicts the least-recently written block on a cache miss. To prevent heat accumulation, the target block to eviction is selected physically far enough from the previously written block in the cache set. Please note that TA-LRW granularity of STT-MRAM cache thermal management is at block-level while WiSE granularity of STT-MRAM cache thermal management is at way (memory bank)-level. Thus, these two approaches are orthogonal to each other.

7 Conclusions

STT-MRAM offers great promise for replacing SRAM-based memories. However, read disturbance, retention failure, and write failure are serious reliability challenges for the development of STT-MRAM. The probability of occurring each of these three reliability challenges is significantly increased in higher temperatures. Write operations are regarded as the main source of heat generation and temperature increase in STT-MRAM cache memories. To reduce the STT-MRAM-based cache memory temperature and improve its reliability, we have proposed WiSE in this paper. WiSE utilizes RL technique to detect high density write operation patterns in ways that leads to a significant increase in cache ways temperature. The experiments show that WiSE reduces retention failure rate, read disturbance rate and write failure rate by 64%, 57%, and 47%, respectively, compared to LRU, while it imposes minor overheads to the system from performance, static energy consumption, and area point of view.

References

[1] A. Salahvarzi *et al.*, "NOSTalgy: Near-Optimum Run-time STT-MRAM Quality-Energy Knob Management for Approximate Computing Applications," *IEEE TC*, 2020.
[2] K. Wang *et al.*, "Low-power non-volatile spintronic memory: STT-RAM and beyond," in *Applied Physics*, 2013.
[3] M. Wang *et al.*, "Current-induced magnetization switching in atom-thick tungsten engineered perpendicular magnetic tunnel junctions with large tunnel magnetoresistance," *Nature communications*, 2018.
[4] Z. Shirmohammadi *et al.*, "3d-dycac: Dynamic numerical-based mechanism for reducing crosstalk faults in 3d ics," in *2017 IEEE International High Level Design Validation and Test Workshop (HLDVT)*, 2017, pp. 87–90.
[5] A. Chintaluri *et al.*, "A model study of defects and faults in embedded Spin Transfer Torque (STT) MRAM arrays," in *Proc. of ATS*, 2015.
[6] Y. Zhang *et al.*, "STT-RAM Cell Design Considering MTJ Asymmetric Switching," *SPIN Journal*, 2012.
[7] X. Bi *et al.*, "Analysis and optimization of thermal effect on stt-ram based 3-d stacked cache design," in *Proc. of ISVLSI*, 2012.
[8] G. Sun *et al.*, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *Proc. of HPCA*, 2009.
[9] M. V. Beigi *et al.*, "Tesla: Using microfluidics to thermally stabilize 3d stacked stt-ram caches," in *Proc. of ICCD*, 2016.
[10] R. S. Sutton *et al.*, "Learning to predict by the methods of temporal differences," in *Machine Learning*, 1988.
[11] N. Binkert *et al.*, "The Gem5 Simulator," *SIGARCH Comput. Archit. News*, 2011.
[12] J. L. Henning, "SPEC CPU2006 benchmark descriptions," in *ACM SIGARCH Computer Architecture News*, 2006.
[13] T. Endoh *et al.*, "An Overview of Nonvolatile Emerging Memories—Spintronics for Working Memories," *IEEE JETCAS*, 2016.
[14] H. Naeimi *et al.*, "STT-MRAM scaling and retention failure," in *Intel Technology Journal (ITJ)*, 2013.
[15] L. Jiang *et al.*, "Improving read performance of stt-mram based mainmemories through smash read and flexible read," in *Proc. of ASP-DAC*, 2016.
[16] H. Sun *et al.*, "Architectural exploration to enable sufficient mtj devicewrite margin for stt-ram based cache," in *IEEE TMAG*, 2012.
[17] R. Bishnoi *et al.*, "Read Disturb Fault Detection in STT-MRAM," in *Proc. of ITC*, 2014.

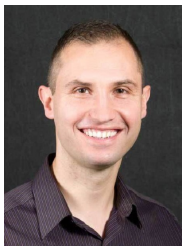
[18] B. Wu *et al.*, "Temperature Impact Analysis and Access Reliability Enhancement for 1T1MTJ STT-RAM," *IEEE Transactions on Reliability*, 2016.
[19] S. Haykin, "Neural Networks: A comprehensive foundation," 1998.
[20] K. Gurney, "An introduction to neural networks," 1997.
[21] M. B. Christopher, "Pattern recognition and machine learning," in *Springer*, 2006.
[22] K. P. Murphy, "Machine Learning: A Probabilistic Perspective," 2012.
[23] R. Bellman, "Dynamic Programming," 1957.
[24] A. G. Barto *et al.*, "Neuronlike adaptive elements that can solve difficult learning control problems," in *IEEE SMC*, 1983.
[25] J. Kim *et al.*, "A technology-agnostic MTJ SPICE model with user-defined dimensions for STT-MRAM scalability studies," in *Proc. of CICC*, 2015.
[26] T. L. Bergman *et al.*, "introduction to heat transfer," 2011.
[27] S. Wang *et al.*, "Word- and partition-level write variation reduction for improving non-volatile cache lifetime," *ACM Transaction on Design and Automation Electronic System*, vol. 23, no. 1, pp. 4:1–4:18, 2017.
[28] L. Chang *et al.*, "Evaluation of spin-Hall-assisted STT-MRAM for cache replacement," in *proc. of NANOARCH*, 2016, pp. 73–78.
[29] X. Dong *et al.*, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," *IEEE TCAD*, 2012.
[30] P. Skoncej, "ECC with Increased Hard Error Correction Capability for Memory Reliability Improvement," in *Proc. of NVMTS*, 2014.
[31] T.-Y. Hsieh *et al.*, "Cost-Effective Enhancement on Both Yield and Reliability for Cache Designs Based on Performance Degradation Tolerance," *IEEE TVLSI*, 2017.
[32] M. Talebi *et al.*, "Rocky: A robust hybrid on-chip memory kit for the processors with stt-mram cache technology," *IEEE TC*, pp. 1–1, 2020.
[33] E. I. Vatajelu *et al.*, "Power-Aware Voltage Tuning for STT-MRAM Reliability," in *Proc. of ETS*, 2015.
[34] H. Lee *et al.*, "A Word Line Pulse Circuit Technique for Reliable Magnetoelectric Random Access Memory," *IEEE TVLSI*, 2017.
[35] H. Sun *et al.*, "Design techniques to improve the device write margin for MRAM-based cache memory," in *Proc. of GLSVLSI*, 2011.
[36] W. Zhao *et al.*, "Design considerations and strategies for high-reliable stt-mram," *Elsevier Microelectronics Reliability (MR)*, 2011.
[37] J. Li *et al.*, "Design paradigm for robust spin-torque transfer magnetic RAM (STT MRAM) from circuit/architecture perspective," *IEEE TVLSI*, 2010.
[38] S. M. Seyedzadeh *et al.*, "Leveraging ecc to mitigate read disturbance, false reads and write faults in stt-ram," in *Proc. of DSN*, 2016.
[39] R. Takemura *et al.*, "Highly-scalable disruptive reading scheme for gb-scale spram and beyond," in *Proc. of IMW*, 2010.
[40] S. Suman *et al.*, "Reinforcement Learning Based Refresh Optimized Volatile STT-RAM Cache," in *Proc. of ISVLSI*, 2020.
[41] I. Yoon *et al.*, "Transfer and Online Reinforcement Learning in STT-MRAM Based Embedded Systems for Autonomous Drones," in *Proc. of DATE*, 2019.
[42] M. M. Shihab *et al.*, "Couture: Tailoring STT-MRAM for Persistent Main Memory," in *4th Workshop on Interactions of NVM/Flash with Operating Systems and Workloads*, 2016.
[43] F. Hao *et al.*, "An Energy Consumption Optimization and Evaluation for Hybrid Cache Based on Reinforcement Learning," *Journal of Computer Research and Development*, 2020.
[44] E. Cheshmikhani *et al.*, "TA-LRW: A Replacement Policy for Error Rate Reduction in STT-MRAM Caches," *IEEE TC*, 2018.

Arash Salahvarzi received the M.Sc degree in computer engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 2018. He is currently a researcher in Cyber-Physical Systems (CPS) Laboratory at IUST. His research interests include low power design, emerging nonvolatile memory technologies, fault-tolerant embedded system design, and storage systems.

Mohsen Khosroanjam received the B.Sc. degree in computer engineering from Shiraz University, Shiraz, Iran, in 2017 and the M.Sc. degree in computer engineering from Iran university of Science and technology, Tehran, Iran, in 2020. His research interests include real-time systems, and reconfigurable architectures.

Amir Mahdi Hosseini Monazzah received his Ph.D degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2017. He was a member of the Dependable Systems Laboratory from 2010 to 2017. As a Visiting Researcher, he was with the Embedded Systems Laboratory, University of California, Irvine, CA, USA from 2016 to 2017. As a postdoc fellow he was with the school of computer science, institute for research in fundamental sciences (IPM), Tehran, Iran from 2017 to 2019. He is currently a faculty member of the School of Computer Engineering, Iran University of Science and Technology (IUST), Tehran, Iran. His research interests include investigating the challenges of emerging nonvolatile memories, hybrid memory hierarchy design, and IoT applications.

Hakem Beitollahi received the B.Sc degree in Computer Engineering from University of Tehran ,and the M.Sc degree from Sharif University of Technology, Tehran, Iran and the PhD degree from Katholieke Universiteit Leuven, Belgium in 2002, 2005, and 2012, respectively. He is currently an assistant professor and the head of hardware and computer systems architecture branch at the Iran University of Science and Technology in the school of Computer Engineering. His research interests currently include real-time systems, fault tolerance and dependability, re-configurable computing, and hardware accelerators.



Umit Y. Ogras received the PhD degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2007. He was a Research Scientist with Intel, Mountain View, CA, USA, from 2008 to 2017, and Arizona State University, Tempe, AZ, USA, from 2013 to 2020. He is currently an Associate Professor with the University of Wisconsin–Madison, Madison, WI, USA. His research interests include energy-efficient embedded systems, wearable Internet-of-Things, flexible hybrid electronics, and multi-core architectures.

Mahdi Fazeli received his M.Sc and Ph.D. degrees in computer engineering both from Sharif University of Technology, Tehran, Iran, in 2005 and 2011, respectively. He is currently an associate professor at the School of Information Technology, Halmstad University, Sweden. His research interests include hardware security and trust, reliable VLSI circuits and systems, energy efficient computing, and dependable embedded systems.