electronics

Article

TA-CLOCK: Tendency-Aware Page Replacement Policy for Hybrid Main Memory in High-Performance Embedded Systems

Jun Hyeong Choi, Kyung Min Kim and Jong Wook Kwak *

Department of Computer Engineering, Yeungnam University, Gyeongsan 38541, Korea;
runchoice@ynu.ac.kr (J.H.C.); bl43a@ynu.ac.kr (K.M.K.)
* Correspondence: kwak@yu.ac.kr

Abstract: Recently, high-performance embedded systems have adopted phase change memory (PCM) as their main memory because PCMs have attractive advantages, such as non-volatility, byte-addressability, high density, and low power consumption. However, PCMs have disadvantages, such as limited write endurance in each cell and high write latency compared to DRAMs. Therefore, researchers have investigated methods for enhancing the limitations of PCMs. In this paper, we propose a page replacement policy called tendency-aware CLOCK (TA-CLOCK) for the hybrid main memory of embedded systems. To improve the limited write endurance of PCMs, TA-CLOCK classifies the page access tendency of the victim page through access pattern analysis and determines the migration location of the victim page. Through the classification of the page access tendency, TA-CLOCK reduces unnecessary page migrations from DRAMs to PCMs. Unnecessary migrations cause an increase in write operations in PCMs and the energy consumption of the hybrid main memory in embedded systems. Thus, our proposed policy improves the limited write endurance of PCMs and enhances the access latency of the hybrid main memory of embedded systems by classifying the page access tendency. We compared the TA-CLOCK with existing page replacement policies to evaluate its performance. In our experiments, TA-CLOCK reduced the number of write operations in PCMs by 71.5% on average, and it enhanced the energy delay product by 38.3% on average compared with other page replacement policies.

Keywords: non-volatile memory; phase change memory; hybrid main memory; page replacement policy; tendency classification; clock algorithm

Citation: Choi, J.H.; Kim, K.M.; Kwak, J.W. TA-CLOCK: Tendency-Aware Page Replacement Policy for Hybrid Main Memory in High-Performance Embedded Systems. *Electronics* **2021**, *10*, 1111. https://doi.org/10.3390/electronics10091111

Academic Editor: Seung-Ho Lim

Received: 26 March 2021
Accepted: 4 May 2021
Published: 8 May 2021

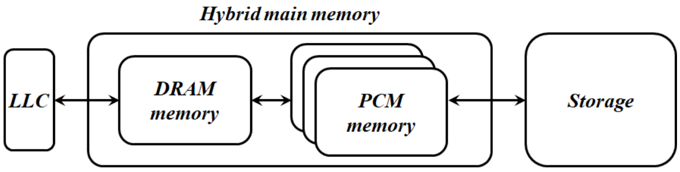
1. Introduction

Recently, demands for high-performance embedded systems have emerged because of the development of various smartphones, wearable devices, and Internet of Things (IoT) devices [1–5]. However, existing embedded systems have limitations because of their low performance, low battery capacity, and low storage capacity. Therefore, the demand for high-performance embedded devices has significantly increased in various embedded systems. For improved performance, embedded systems require high computing power, low power consumption, and high memory capacity [6]. Particularly, among the embedded system components, the power consumption of the main memory is 30–50% of the overall power consumption of the system [7]. Nevertheless, the battery capacity of embedded systems is limited; therefore, several studies on embedded systems have focused on reducing their power consumption to improve the limited battery lifespan.

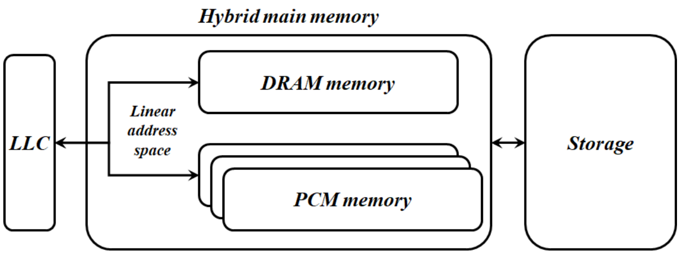
Dynamic random access memories (DRAMs) are widely used in the main memories of embedded systems because of their unlimited write endurance and low read/write latency. However, it is difficult to use DRAMs directly in the main memory of an embedded system with limited battery capacity because they have high power consumption, thereby requiring leakage power. Therefore, researchers have proposed a hybrid main memory architecture for resolving the limitations of DRAM and improving the performance of

DRAM alone main memory [8–11]. Phase change memory (PCM) is an attractive memory for embedded systems owing to its advantages, such as non-volatility, high density, and low power consumption [12–16]. However, PCMs have disadvantages, such as limited write endurance and high write latency compared to DRAMs. Therefore, hybrid main memories have been proposed to enhance the limitations of DRAMs and PCMs.

Figure 1 shows the organization methods of the hybrid main memory. The organization methods of hybrid main memories can be classified into hierarchical organization and parallel organization. Hierarchical organization comprises an upper DRAM cache layer and a lower PCM memory layer, where the DRAM cache layer incurs an additional hardware cost. Meanwhile, parallel organization comprises DRAMs and PCMs as the same layer. Parallel organization can reduce the additional hardware cost and use a wide range of the main memory space. Therefore, several researchers have studied parallel organization [8–10].



(**a**)



(**b**)

Figure 1. Organization methods of the hybrid main memory. (**a**) Hierarchical organization. (**b**) Parallel organization.

In this paper, we propose a hybrid main memory architecture and its page replacement policy called tendency-aware CLOCK (TA-CLOCK). TA-CLOCK consists of DRAM CLOCK and PCM CLOCK, and it determines the optimized location of requested pages using tendency classification. When the host system requires new read/write pages, TA-CLOCK first allocates pages in the DRAM CLOCK. To select the victim page in the DRAM, the DRAM CLOCK classifies its page access tendency. Afterward, TA-CLOCK determines the location of the selected victim page using write/read thresholds, which are determined by the write and read counts of the victim page. Based on the classification of the page access tendency, TA-CLOCK reduces unnecessary migrations between the DRAM CLOCK and PCM CLOCK. Additionally, it migrates dirty pages from the PCM CLOCK to the DRAM CLOCK to reduce the number of write operations in the hybrid main memory. As a result, TA-CLOCK improves the limited write endurance of the PCM and enhances the energy delay product (EDP) of the hybrid main memory.

The remainder of this paper is organized as follows. In Section 2, the background of PCMs is described and existing hybrid main memory policies are reviewed. The proposed TA-CLOCK hybrid main memory policy with its structure, algorithm, and operations is presented in Section 3. In Section 4, the performance of the TA-CLOCK is evaluated by comparison with existing main memory policies. Finally, conclusions are presented in Section 5.

2. Background and Related Works

In this section, we describe the background of PCMs and discuss existing hybrid main memory management policies.

2.1. Phase Change Memory

Recently, PCMs have been studied for use in the hybrid main memory of high-performance embedded systems because they have characteristics, such as low leakage power, high density, and non-volatility, compared to DRAMs [12–16]. Table 1 presents the details of the comparison of DRAMs and PCMs. PCMs are non-volatile, and they comprise phase change materials, such as chalcogenide glass [14]. Furthermore, they store data using the state transition of phase change materials. Phase change materials trasition into crystalline and amorphous states by heat to use electrical pulses. The crystalline state represents SET (value 1), and it has high optical reflexivity and low resistivity. Meanwhile, the amorphous state represents RESET (value 0), and it has low optical reflexivity and high resistivity.

Table 1. Comparison of DRAM and PCM.

	DRAM	PCM
Cell size (F^2)	6	4∼30
Data retention	∼64 ms	>10 years
Read latency (ns)	∼10	<10
Write latency (ns)	∼10	<50
Voltage (V)	<1	<3
Write energy (J/bit)	∼10 fJ	∼10 pJ
Write endurance	10^{15}	10^8∼10^9

Although PCMs have attractive advantages, such as low leakage power, high density, byte-addressability, and non-volatility, to replace DRAMs in embedded systems [12–14], they have disadvantages, such as high read/write latency with even asymmetric operation cycle requirement and limited write endurance. Thus, it is difficult to adopt PCMs in write-intensive systems because the write endurance of each cell in the PCM is approximately 10^8–10^9. Therefore, existing research has been focused on controlling the write operations of PCMs and providing improved access latency and write endurance by altering their hardware design. Nevertheless, PCMs cannot completely replace DRAMs in the main memory architecture owing to the advantages of DRAMs, such as unlimited write endurance and low read/write latency. Hence, recent research has proposed a hybrid main memory architecture to improve the limitations of PCMs and DRAMs by selectively exploiting their advantages.

2.2. Hybrid Main Memory

The hybrid main memory adopts DRAMs and PCMs to improve the performance of embedded systems by selectively exploiting their advantages [8–11]. However, existing page replacement policies for main memories do not consider the characteristics of the hybrid main memory, especially PCMs, which are non-volatile memories. Therefore, when the hybrid main memory system adopts conventional page replacement policies, the performance of the embedded system is limited, and the policies cannot fully utilize the performance potential of non-volatile memories.

Zhang et al. proposed a hybrid main memory management policy called triple-based hybrid main memory (TriBHMM) [11], which comprises three components: L1 DRAM, L2 DRAM, and L2 PCM. It manages pages in the main memory using priority and the least-recently-used (LRU) algorithm to reduce energy consumption, memory access time, and PCM write counts. Lin et al. proposed history-aware LRU (HA-LRU), which is a page replacement policy for NAND flash-based storages [17]. HA-LRU manages pages in the main memory through the access history information using four LRU lists. However, these policies are based on the LRU algorithm and have a disadvantage in that they require a computation overhead when used in the main memory. This requirement is especially critical in embedded systems that utilize limited computing resources. Therefore, to reduce the computation overhead, recent studies have utilized the CLOCK algorithm.

The CLOCK algorithm was proposed for overcoming the disadvantages of LRU [10,18]. CLOCK consists of the circular list and the clock hand for managing pages in the main memory. The circular list of CLOCK manages pages, and each page include a reference bit. When the main memory requires the page replacement, the clock hand of CLOCK selects a victim page through the reference bit of the pointed page. If the reference bit of the pointed page is 0 (false), the pointed page is determined as the victim page and it is evicted to storages. In contrast, when the reference bit of the pointed page is 1 (true), it is changed to 0 and the clock hand moves to the next page in the circular list. In some cases, CLOCK maintains a dirty bit to manage write operations effectively.

Migration-optimized CLOCK (M-CLOCK) manages each page in the hybrid main memory by operation type [9]. It also manages frequently accessed pages to the DRAM using the write operation, and migrates frequently referenced pages to the PCM using the read operation. Moreover, it secures free pages in the DRAM by evicting unnecessary pages, which are unlikely to be referenced. However, in the early stage, M-CLOCK evicts DRAM pages to the storage because of the difficulty of accessing pattern identification. These problems arise because the characteristics of the dynamic access pattern in applications are not considered. As a result, M-CLOCK re-allocates the evicted pages from the storage to the DRAM, thereby reducing the performance of M-CLOCK in applications that frequently change the read/write access pattern.

Meanwhile, CLOCK with dirty bits and write frequency (CLOCK-DWF) is a page replacement policy for the hybrid main memory [10]. Unlike M-CLOCK, CLOCK-DWF determines page locations by the access type of the newly inserted pages. It allocates write pages to DRAM and read pages to PCM. Additionally, when a write access occurs in a page allocated to PCM, CLOCK-DWF migrates the accessed page to DRAM to reduce additional write operations in the PCM. However, it is difficult to utilize the limited write endurance of PCMs, especially in applications that include frequent random read accesses, because it allocates the read accesses only to PCM.

3. Design of TA-CLOCK

In this section, we propose a novel page replacement policy for the hybrid main memory, which improves the limited write endurance of PCMs and reduces the energy consumption of hybrid main memories. We refer to the proposed page replacement policy as TA-CLOCK. TA-CLOCK analyzes the access tendency of pages through the write/read threshold, and determines suitable page locations to overcome the limitations of PCMs in high-performance embedded systems. It has some specific characteristics which are summarized below:

1. TA-CLOCK improves the limited write endurance of PCMs by maintaining write pages in the DRAM CLOCK through the classification of the page access tendency.
2. TA-CLOCK reduces the energy consumption of the hybrid main memory in an embedded system by enhancing the write hit ratio of the DRAM CLOCK, thus reducing unnecessary page migrations between the DRAM CLOCK and PCM CLOCK.

3.1. Structure of TA-CLOCK

TA-CLOCK comprises DRAM CLOCK and PCM CLOCK for managing the read and write pages in the hybrid main memory. The overall structure of TA-CLOCK is presented in Figure 2. To manage pages in the hybrid main memory, TA-CLOCK utilizes the CLOCK algorithm, and the DRAM CLOCK maintains the read/write counts of each page to analyze the page access tendency. In our proposal, the page access tendency is classified by the write/read threshold, and it is used to determine the victim page to secure a free page in the DRAM CLOCK. Through this process, it maintains write-intensive pages in the DRAM CLOCK and reduces unnecessary page migrations between the DRAM CLOCK and PCM CLOCK.

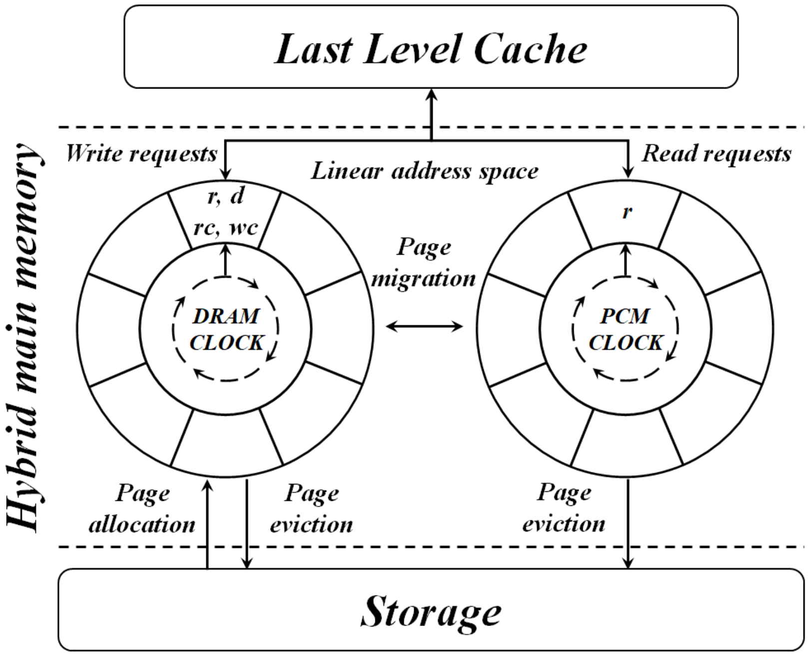


Figure 2. Structure of TA-CLOCK. Our policy comprises DRAM CLOCK and PCM CLOCK. The pages in the DRAM CLOCK maintain the reference bit r, dirty bit d, read counts rc, and write counts wc, and the pages in the PCM CLOCK maintain the reference bit r.

The DRAM CLOCK of the TA-CLOCK is the location where the requested read/write pages are allocated from the host system. It maintains a reference bit (r), dirty bit (d), read count (rc), and write count (wc) for each page. When read/write requests occur from the host system, the requested pages are inserted into the DRAM CLOCK. Additionally, when read/write hits occurs on the inserted page in the DRAM CLOCK, the TA-CLOCK increases the read/write counts of the accessed page. In the TA-CLOCK, the read/write counts of each page are used to analyze the page access pattern and classify the page access tendency. If the DRAM CLOCK does not have a free page, the TA-CLOCK selects a victim page and the migration location using the page access tendency.

Meanwhile, the PCM CLOCK of the TA-CLOCK is the location that manages the read-intensive pages migrated from the DRAM CLOCK using reference bit (r) for each page. When a write operation occurs in the PCM CLOCK, the TA-CLOCK migrates the accessed page to the DRAM CLOCK.

3.2. TA-CLOCK Algorithm

In this section, we describe the main algorithm of our proposed page replacement policy. Table 2 describes the parameters used in this study. The DRAM CLOCK of the TA-CLOCK uses the read/write counts of each page to analyze and classify the page access tendency of a selected victim page for page replacement. The page access tendency is classified into four states: strong write (SW), weak write (WW), weak read (WR), and strong read (SR). The details of the classification of the page access tendency with their threshold values are described in Table 3. The calculation of the threshold value and its operation are as follows.

$$Write\_Threshold_{DRAM} = \frac{\sum_{i=1}^{page\_count} write\_count_i}{page\_count} \times \frac{1}{weight_{write}}, \quad (1)$$

$$Read\_Threshold_i = \left| \left(1 - \frac{write\_count_i}{read\_count_i}\right) \times \frac{1}{weight_{read}} \right|. \quad (2)$$

Table 2. Description of parameters for the analysis of the page access tendency.

Parameter	Description
i	The index of DRAM CLOCK page.
$Read\_Threshold_i$	The read threshold of page i for the classification of the page access tendency.
$Write\_Threshold_{DRAM}$	The write threshold of DRAM CLOCK for the classificati- on of the page access tendency.
$page\_count$	The number of pages in DRAM CLOCK.
$read\_count_i$	The read counts of page i.
$write\_count_i$	The write counts of page i.
$weight_{write}$	The weight of write operations (constant).
$weight_{read}$	The weight of read operations (constant).

Table 3. Classification of the page access tendency of DRAM pages and their conditions.

Tendency Classification	Condition
SW (Strong Write)	$write\_count_i \geq Write\_Threshold_{DRAM}$
WW (Weak Write)	$write\_count_i < Write\_Threshold_{DRAM}$ & $Read\_Threshold_i \geq 0.5$
WR (Weak Read)	$write\_count_i < Write\_Threshold_{DRAM}$ & $Read\_Threshold_i < 0.5$ & $Read\_Threshold_i \geq 0.25$
SR (Strong Read)	$write\_count_i < Write\_Threshold_{DRAM}$ & $Read\_Threshold_i < 0.25$

If the DRAM CLOCK does not have a free page, the TA-CLOCK selects a victim page in the DRAM CLOCK. $Write\_Threshold_{DRAM}$ in Equation (1) is used to determine the tendency of the write operation in a specific DRAM CLOCK page. TA-CLOCK analyzes and classifies the page access tendency of the victim page through $Write\_Threshold_{DRAM}$ which is calculated by the write counts of all the DRAM CLOCK pages ($write\_count_i$) and the page counts in the DRAM CLOCK ($page\_count$) by adjusting write weight ($weight_{write}$). The page access tendency of the selected page is classified by comparing the number of write counts of the selected page and the write threshold of the DRAM CLOCK. If the page access tendency of the selected page is *SW*, the TA-CLOCK maintains the selected page in the DRAM CLOCK to improve its write hit ratio and reduce the number of write counts in the PCM CLOCK.

Meanwhile, if the page access tendency of the selected page is not classified as *SW*, the TA-CLOCK analyzes the page access tendency of the selected page to determine the migration location of the selected page. $Read\_Threshold_i$ in Equation (2) is used to determine the migration location of the selected page by both read counts of page i ($read\_count_i$) and write counts of page i ($write\_count_i$) by adjusting read weight ($weight_{read}$). The page access tendency of the selected page is classified as *WW*, *WR*, and *SR* by the read threshold. To reduce the write counts in the PCM CLOCK, the TA-CLOCK considers the ratio of read/write counts in the selected page. If the page access tendency of the selected page is *WW*, the TA-CLOCK maintains the selected page in the DRAM CLOCK and reclassifies the page access tendency. However, if the selected page is classified as *WR*, it is evicted to the storage because the page access tendency of the selected page is read operation. As a result, the TA-CLOCK reduces the number of unnecessary migrations from the DRAM to the PCM, and it secures a free page in the DRAM CLOCK. Meanwhile, *SR* is a state that is expected to occur in additional read operations on the selected page. Therefore, the TA-CLOCK migrates the selected page to the PCM CLOCK to secure a new free page in the DRAM CLOCK. When the *SR* pages are migrated, the TA-CLOCK reduces the power consumption of the embedded system and improves the limited write endurance of the PCM.

Thus, TA-CLOCK determines the location of the victim page by classifying its page access tendency. Particularly, to reduce unnecessary migrations to the PCM CLOCK, the

TA-CLOCK classifies pages with weak tendencies as *WW* and *WR*. Therefore, the TA-CLOCK reduces the number of write operations in the PCM CLOCK and the number of unnecessary migrations, and enhances the write hit ratio in the DRAM CLOCK, which positively affects the EDP of the main memory system.

Algorithm 1 describes the TA-CLOCK in detail, and Table 4 presents the parameter definition. First, when page p requested from the host system hits the DRAM CLOCK, the TA-CLOCK updates the parameters according to the operation type (lines 1–8). If write operation occurs in page p, the TA-CLOCK sets its dirty bit as 1 and increases its write count by 1 (lines 2–4). Conversely, if read operation occurs on page p, the TA-CLOCK sets its reference bit as 1 and increases its read count by 1 (lines 5–7). The reason for increasing the read/write counts is to calculate the write and read thresholds when the DRAM CLOCK becomes full at a later time. Meanwhile, if page p hits the PCM CLOCK with a write operation, it is migrated to the DRAM CLOCK (lines 10–12). Conversely, if read operation occurs on page p in the PCM CLOCK, the TA-CLOCK sets its reference bit as 1 (lines 13–15). If page faults occur in the TA-CLOCK, and the DRAM CLOCK becomes full, the TA-CLOCK migrates a page in the DRAM CLOCK to the PCM CLOCK based on its page replacement policy (lines 19–21) and inserts page p into a free page in the DRAM CLOCK. If the inserted page p is referenced by the write operation, the TA-CLOCK sets its dirty bit as 1 and increases its write count by 1 (lines 24–26). However, if it is referenced by a read operation, its reference bit is set to 1, and its read count is increased by 1 (lines 27–29).

Algorithm 1 TA-CLOCK (page p, operation *op*).

1: **if** p in DRAM CLOCK **then**
2: **if** *op* = write **then**
3: $p_{dirty} \leftarrow$ set;
4: $p_{write\_count} \leftarrow p_{write\_count} + 1$;
5: **else**
6: $p_{ref} \leftarrow$ set;
7: $p_{read\_count} \leftarrow p_{read\_count} + 1$;
8: **end if**
9: **else if** p in PCM CLOCK **then**
10: **if** *op* = write **then**
11: // Decide whether the page migrates to DRAM CLOCK.
12: DRAMPageReplacement() $\leftarrow p$;
13: **else**
14: $p_{ref} \leftarrow$ set;
15: $p_{read\_count} \leftarrow p_{read\_count} + 1$;
16: **end if**
17: **else** // Page fault.
18: // No free page in DRAM CLOCK.
19: **if** DRAM CLOCK is full **then**
20: // Secure a free page and insert p into secured free page.
21: DRAMPageReplacement() $\leftarrow p$;
22: **else**
23: Insert p into free page in DRAM CLOCK.
24: **if** *op* = write **then**
25: $p_{dirty} \leftarrow$ set;
26: $p_{write\_count} \leftarrow 1$; //Initialize write counts of p.
27: **else**
28: $p_{ref} \leftarrow$ set;
29: $p_{read\_count} \leftarrow 1$; //Initialize read counts of p.
30: **end if**
31: **end if**
32: **end if**

Table 4. Definition of parameters in Algorithms 1 and 2.

Parameter	Description
p_{ref}	The reference bit of page p.
p_{dirty}	The dirty bit of page p.
p_{read_count}	The read count of page p.
p_{write_count}	The write count of page p.
$p_{tendency}$	The page access tendency of page p.

To secure a free page in the DRAM CLOCK, the TA-CLOCK uses the DRAMPageReplacement() algorithm, whose details are described in Algorithm 2. The DRAM CLOCK hand points to page p in the DRAM CLOCK (line 1). When the reference bit of page p is set, the TA-CLOCK resets its reference as 0, and the clock hand points to the next page (lines 3–5). Conversely, if the reference bit of page p is 0, the TA-CLOCK classifies its page access tendency to secure a free page in the DRAM CLOCK (lines 6–25). If the dirty bit of page p is set as 1, the TA-CLOCK calculates the write threshold of the DRAM CLOCK using Equation (1), and classifies the page access tendency of page p (lines 7–9). Subsequently, if the page access tendency of page p is SW, the TA-CLOCK retains it in the DRAM CLOCK and analyzes the page access tendency of the next page to secure a free page in the DRAM CLOCK (line 21). Meanwhile, if the page access tendency of page p is not SW, the TA-CLOCK calculates its read threshold using Equation (2) (lines 10–11). Moreover, if its page access tendency is *WW*, the clock hand of TA-CLOCK points to the next page in the DRAM CLOCK and repeats the above processes (lines 13–14). Furthermore, if the page access tendency of page p is *WR*, the TA-CLOCK evicts it to the storage and secures a free page in the DRAM CLOCK (lines 15–16). In contrast, if the page access tendency of page p is *SR*, the TA-CLOCK migrates it to the PCM CLOCK (lines 17–18). Finally, if page p is a clean page, it is discarded from the DRAM CLOCK (lines 22–23).

Algorithm 2 DRAMPageRelpacement (*void*).

1: $p \leftarrow$ GetCurrentClockPointer();
2: **while** page p is not empty **do**
3:   **if** p_{ref} is set **then**
4:     $p_{ref} \leftarrow$ reset;
5:     $p \leftarrow$ GetNextClockPointer();
6:   **else**
7:     **if** p_{dirty} is set **then** // page p is dirty page.
8:       Calculate $Write\_Threshold_{DRAM}$ by Equation (1);
9:       Classify the page access tendency of page p;
10:       **if** $p_{tendency}$ is not *SW* **then**
11:         Calculate $Read\_Threshold_p$ of page p by Equation (2);
12:         Classify the page access tendency of page p;
13:         **if** $p_{tendency}$ is *WW* **then**
14:           $p \leftarrow$ GetNextClockPointer();
15:         **else if** $p_{tendency}$ is *WR* **then**
16:           Evict page p to storage;
17:         **else if** $p_{tendency}$ is *SR* **then**
18:           Migrate page p to PCM CLOCK;
19:         **end if**
20:       **end if**
21:       $p \leftarrow$ GetNextClockPointer();
22:     **else** // page p is clean page.
23:       Discard page p;
24:     **end if**
25:   **end if**
26: **end while**

3.3. Scenarios of TA-CLOCK

In this section, we provide representative scenarios of the TA-CLOCK proposed in this paper. In these scenarios, pages that are allocated to the DRAM CLOCK are marked as $P(x, y)$, where P represents the logical page number of each page. In the DRAM CLOCK pages, x and y represent the read and write counts, respectively. However, in the PCM CLOCK pages, x denotes the reference bits. In the presented scenarios, we assumed that the victim page of the DRAM CLOCK is already selected by the DRAM page replacement algorithm of the TA-CLOCK.

Figure 3 shows an example of page allocation in the DRAM CLOCK and page migration from the DRAM CLOCK to the PCM CLOCK. In this example, the TA-CLOCK selects page A as the victim page. The $Write\_Threshold_{DRAM}$ is 2.5, and $Read\_Threshold_A$ is 0.2, using Equations (1) and (2), respectively. When a write operation occurs in page K, page A is selected as the victim page by the DRAM page replacement policy of TA-CLOCK (*Step 1*). Because the page access tendency of page A is *SR*, the TA-CLOCK migrates page A to the PCM CLOCK to secure a free page in the DRAM CLOCK for page K. To insert page A into the PCM CLOCK, the TA-CLOCK discards page O from the PCM CLOCK using the CLOCK algorithm (*Step 2*). Subsequently, the TA-CLOCK migrates page A to the location of the evicted page O in the PCM CLOCK (*Step 3*). Finally, it inserts page K into the location of the migrated page A in the DRAM CLOCK (*Step 4*).

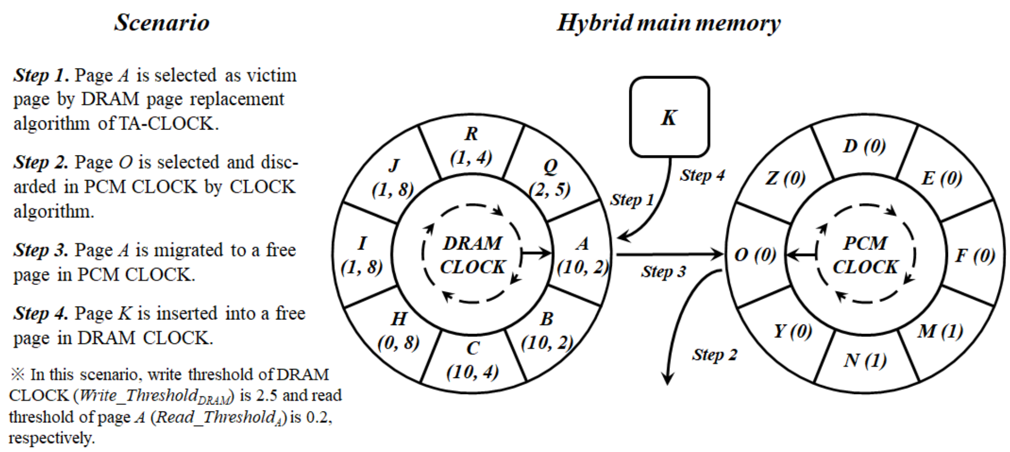


Figure 3. Example of page allocation in the DRAM CLOCK and page migration from the DRAM CLOCK to the PCM CLOCK.

Figure 4 shows an example of page eviction from the DRAM CLOCK to the storage and new page allocation in the DRAM CLOCK. When page N is requested from the host system, the TA-CLOCK selects page B as the victim page because its write count is less than $Write\_Threshold_{DRAM}$ (*Step 1*). In this case, $Read\_Threshold_B$ is 0.3, and the TA-CLOCK classifies the page access tendency of page B as *WW*. Therefore, page B is evicted to the storage by the DRAM page replacement policy of the TA-CLOCK (*Step 2*). Finally, page N is inserted into the location of page B of the DRAM CLOCK (*Step 3*).

When write operations occur again in the dirty page of the PCM CLOCK, the TA-CLOCK migrates or discards the selected victim page using its page access tendency. Figure 5 presents an example of page exchange between the DRAM CLOCK and PCM CLOCK. When a write operation occurs in dirty page A in the PCM CLOCK, the TA-CLOCK migrates page A to the DRAM CLOCK (*Step 1*). Subsequently, the TA-CLOCK selects the victim page to secure a free page page A in the DRAM CLOCK. In this example, the TA-CLOCK selects page N in the DRAM CLOCK as the victim page because $Read\_Threshold_N$ and its page access tendency are 0 and *SR*, respectively (*Step 2*). Finally, page N is inserted into the location of page A in the PCM CLOCK (*Step 3*).

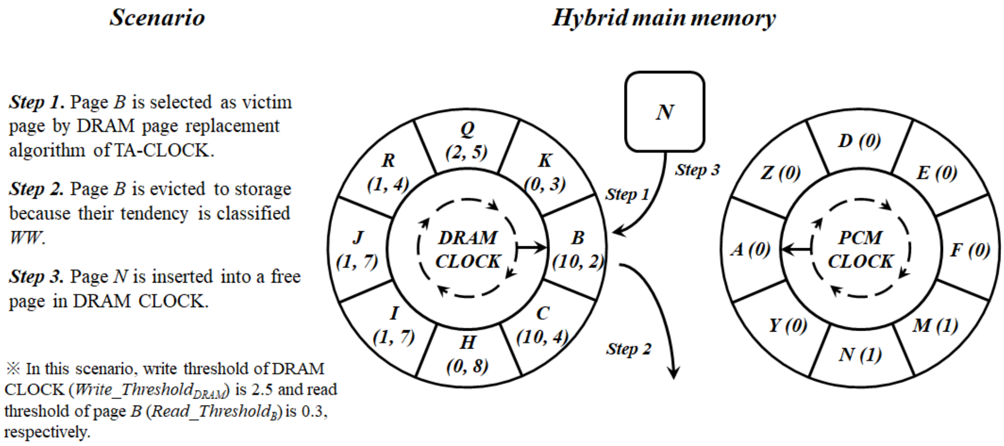


Figure 4. Example of page allocation in the DRAM CLOCK and page eviction from the DRAM CLOCK to the storage.

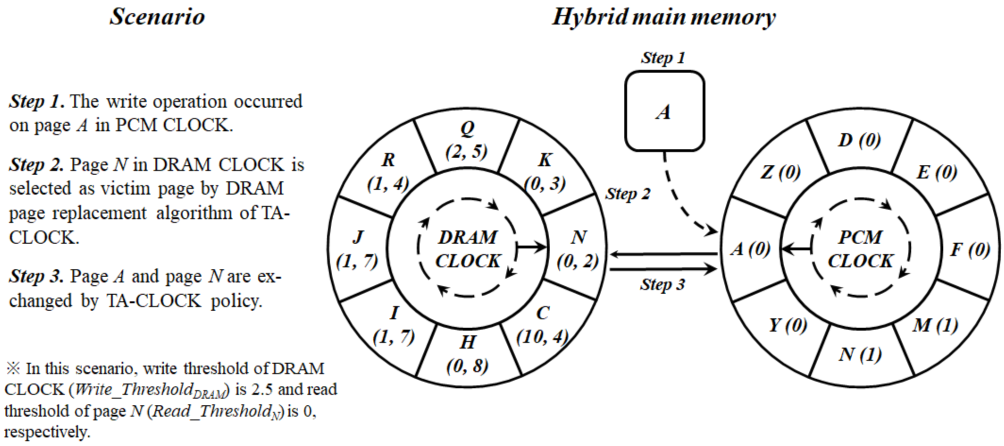


Figure 5. Example of page exchange in the DRAM CLOCK and the PCM CLOCK.

4. Performance Evaluation

In this section, we describe the evaluation environment and discuss the performance enhancement of our proposed approach in comparison with existing page replacement policies.

4.1. Experimental Setup

We used the gem5 simulator to evaluate the performance of the TA-CLOCK and existing main memory page replacement policies [19]. To guarantee the reliability of the experiments, we adopted the full system mode of the gem5 simulator. In our experiments, we measured the performance of our proposed policy by adjusting the ratio of DRAM to PCM in the hybrid main memory for comparison with existing hybrid main memory policies. In addition, we used ten billion simulated instructions in our experiments for evaluating the performance of the TA-CLOCK, compared with CLOCK, CLOCK-PRO, M-CLOCK, and CLOCK-DWF by the following indicators: the number of write operations in the PCM, the number of migrations between DRAM and PCM, the hit ratio of write operations in DRAM and the EDP [9,10,20]. We used $weight_{write}$ and $weight_{read}$ as 25 and 100, respectively. The remaining parameters of our experiments are summarized in Table 5.

We used the MiBench embedded benchmark program to compare and analyze the performance of our proposed policy with other policies [21]. MiBench is frequently used to evaluate the performance of various embedded systems [22–24]. In the experiments, we used benchmarks for performance evaluation with different memory footprint sizes and read/write ratios. In addition, we performed experiments by adjusting the size of the hybrid main memory to consider the corresponding memory footprint size of workloads, as described in Table 6.

Table 5. Experimental parameters.

Parameter	Description	
Read/Write latency	DRAM (ns) PCM (ns)	50/50 50/350
Read/Write energy	DRAM (nJ/bit) PCM (nJ/bit)	0.1/0.1 0.2/1.0
Static power	DRAM (W/GB) PCM (W/GB)	1 0.1
Storage	Access latency (ms)	5

Table 6. Workload specifications.

Workload	Memory Footprint (KB)	Read/Write Ratio (%)
basicmath	26,140	71/29
blowfish	17,261	63/37
dijkstra	19,112	72/28
patricia	19,281	76/24
qsort	25,735	58/42
stringsearch	16,716	72/28

4.2. Experimental Results

In this subsection, we present the number of write operations in the PCM, number of migrations between the DRAM and PCM, DRAM write hit ratio, and EDP of our proposed policy. The PCM write count is an important indicator of the performance of hybrid main memories. PCMs have a high write latency and limited write endurance compared to DRAMs. As a result, frequent write operations to the PCM reduces the performance of hybrid main memories. To solve this problem, the TA-CLOCK reduces the number of write operations in the PCM CLOCK by reducing page allocations to the PCM and uncessary migrations between the DRAM and PCMs.

Figure 6 shows the number of write operations in the PCM of the TA-CLOCK compared with those of CLOCK, CLOCK-Pro, M-CLOCK, and CLOCK-DWF. The TA-CLOCK reduced the number of write operations in the PCM by averages of 93.6%, 92.1%, 41.6%, and 58.7% compared with those of CLOCK, CLOCK-Pro, M-CLOCK, and CLOCK-DWF, respectively. This shows that TA-CLOCK effectively performs state classification using Equations (1) and (2). Hence, it can utilize the limited endurance of PCM in the hybrid main memory of embedded systems.

When write operations frequently occur to a specific area of hybrid main memories, the write endurance of the system is reduced. TA-CLOCK alleviates the non-uniformity of write operations on a specific page by reducing the number of write operations in PCMs. Figure 7 shows the number of PCM writes, grouped by block units. TA-CLOCK shows the least number of write operations to whole areas of PCM, and write operations are evenly distributed. Table 7 described the average write counts and standard deviation on the PCM for each page replacement policy.

Figure 8 shows the number of migrations between the DRAM and PCM in the hybrid main memory. To improve the performance of embedded systems and reduce the number of write operations in PCMs, hybrid main memories need to reduce the number of unnecessary migrations of victim pages. The TA-CLOCK classifies the page access tendency into four types to determine the location of the victim page. As a result, TA-CLOCK reduced the number of migrations between the DRAM and PCM in the hybrid main memory by averages of 54.8% and 59.8% compared with those of M-CLOCK and CLOCK-DWF, respectively.

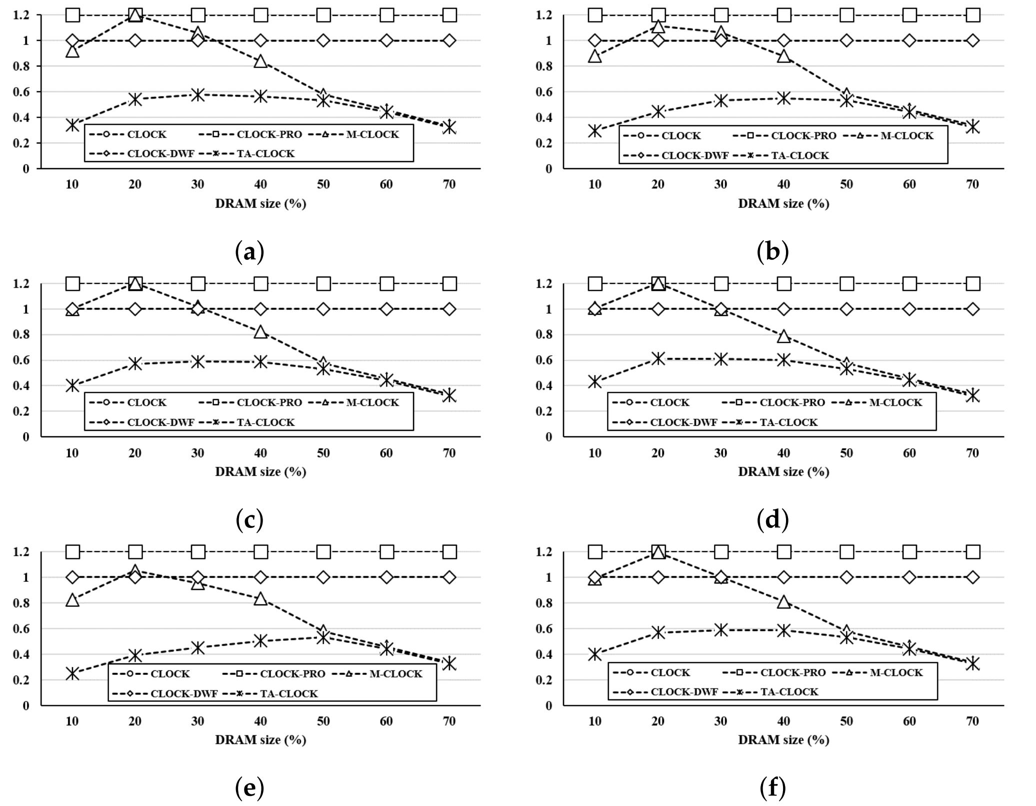


Figure 6. Normalized write operations in the PCM (values of 1.2 or higher are indicated as 1.2). (**a**) basicmath; (**b**) blowfish; (**c**) dijkstra; (**d**) patricia; (**e**) qsort; (**f**) stringsearch.

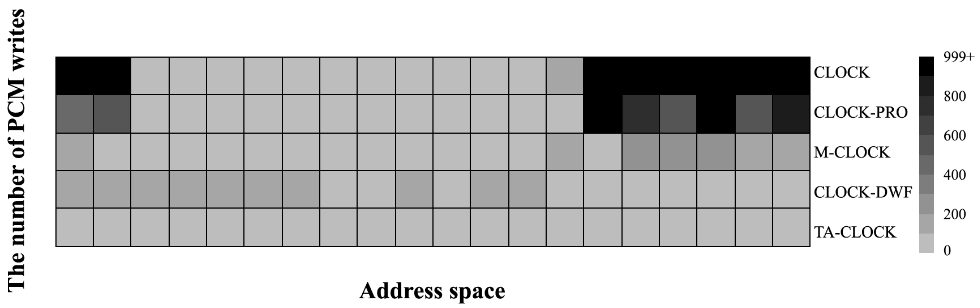


Figure 7. Distribution of write counts in the PCM.

Table 7. The average and standard deviation of write counts in PCM pages.

Page Replacement Policy	The Average Write Counts for Each Page on the PCM	The Standard Deviation of Write Counts on the PCM
CLOCK	18.54	52.77
CLOCK-PRO	4.69	13.18
M-CLOCK	1.57	1.06
CLOCK-DWF	1.48	0.5
TA-CLOCK	0.85	0.35

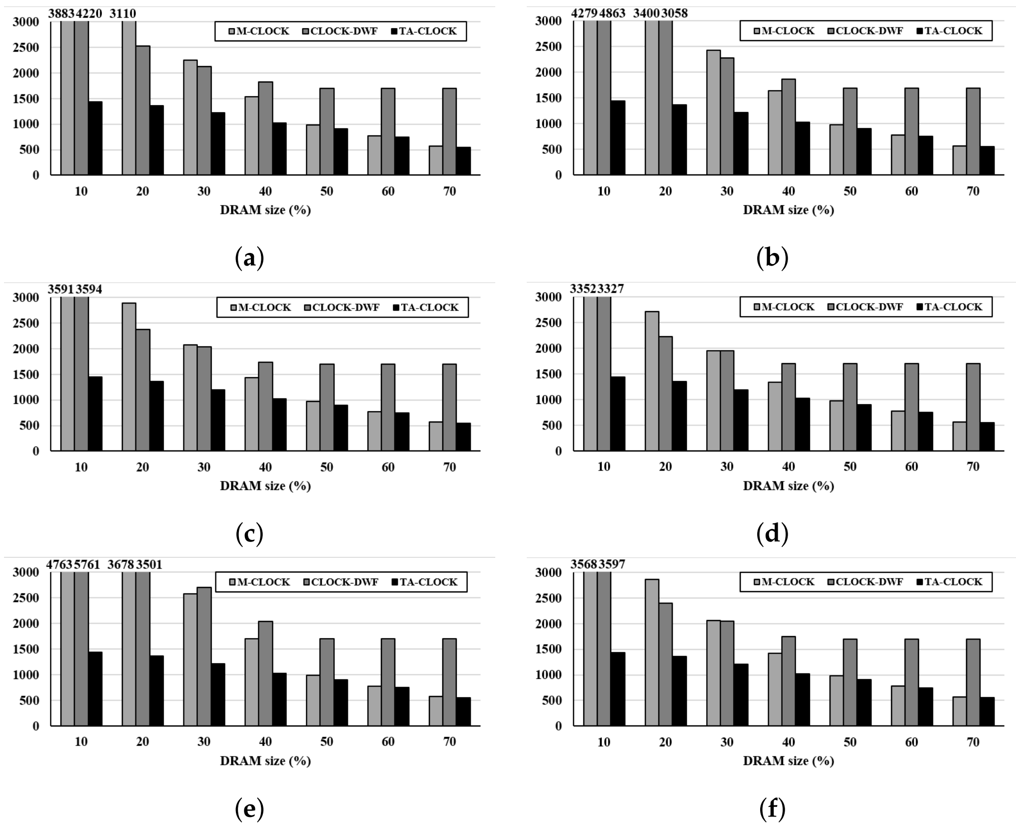
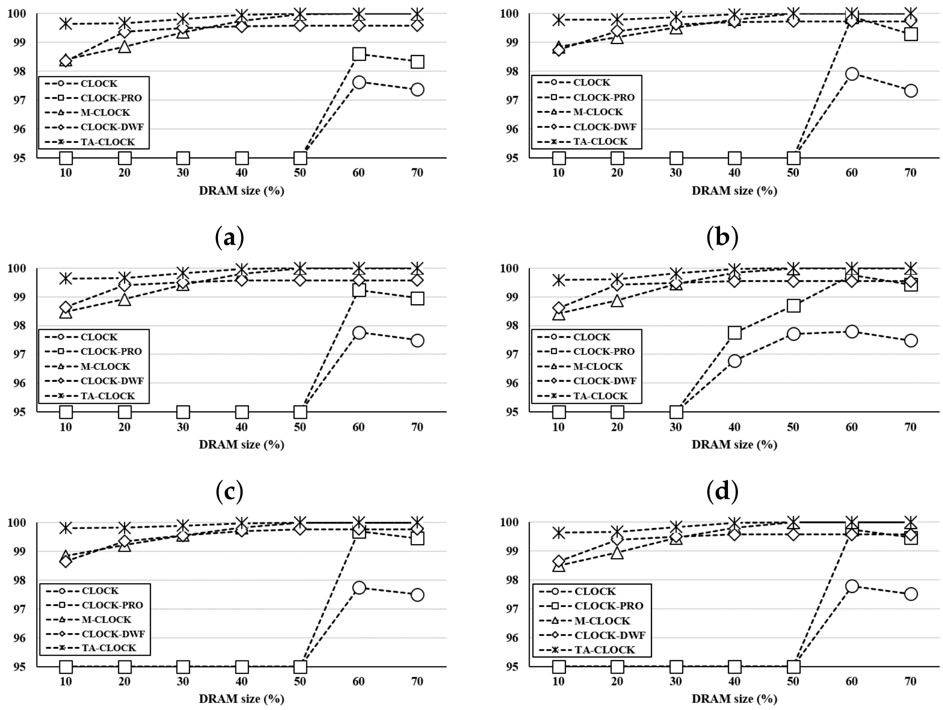


Figure 8. Number of migrations between DRAM and PCM (values of 3000 or higher are indicated as 3000). (**a**) basicmath; (**b**) blowfish; (**c**) dijkstra; (**d**) patricia; (**e**) qsort; (**f**) stringsearch.

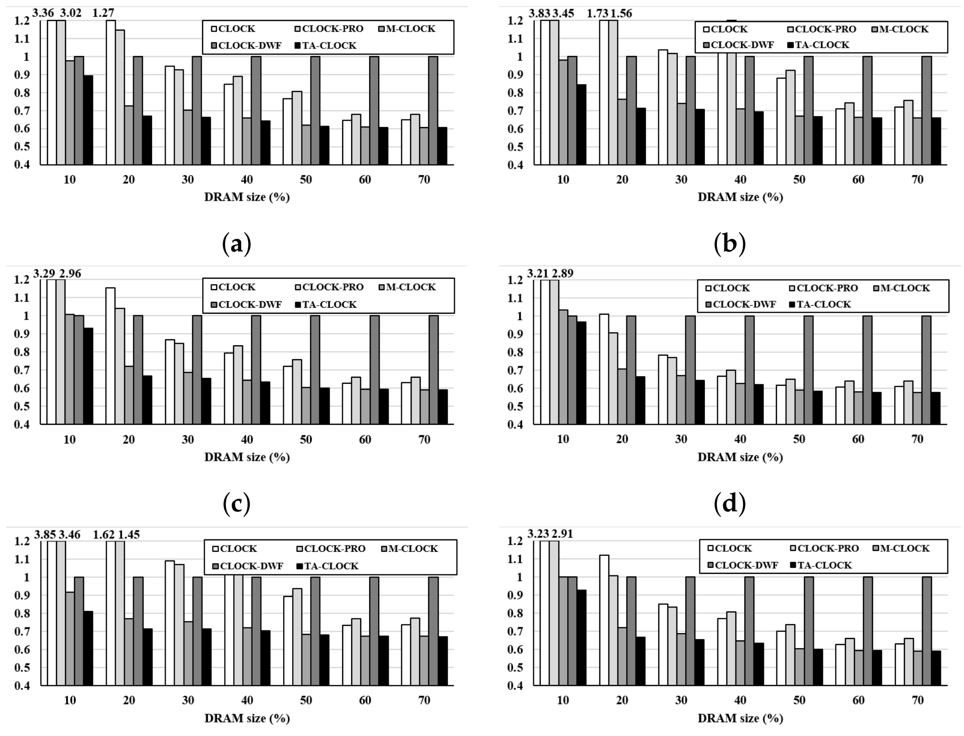
Figure 9 shows the write hit ratio of DRAM in the hybrid main memory. When frequent write operations occur in the PCMs of hybrid main memories, the performance of embedded systems is reduced, and it is difficult to utilize the limited lifespan of PCMs. Therefore, in embedded systems with hybrid main memories, research has been conducted toward designing write operations to occur in DRAM. In our experiments, TA-CLOCK had a higher DRAM write hit rate than other techniques, with averages of 21.1%, 19.6%, 0.3%, and 0.4% compared with those of CLOCK, CLOCK-Pro, M-CLOCK, and CLOCK-DWF, respectively, as shown in Figure 9. From this result, we can infer that TA-CLOCK effectively maintains write pages in the DRAM CLOCK by classifying the page access tendency of the victim page. Consequently, it reduces the PCM write count and the number of migrations between the DRAM and PCM, as observed in Figures 6 and 8.

The reduction of the energy consumption of hybrid main memories is important because embedded systems have limited battery capacity. Figure 10 shows the EDP of the TA-CLOCK normalized to CLOCK-DWF. EDP is an important metric for performance and energy consumption. As shown by Figure 10, TA-CLOCK reduced the EDP by averages of 51.6%, 48.1%, 3.8%, and 49.6% compared with those of CLOCK, CLOCK-Pro, M-CLOCK, and CLOCK-DWF, respectively, by reducing the number of write operations in the PCM. This result is supported by previous experimental results, in which write counts in the PCM and number of migrations between the DRAM and PCM were reduced, and the write hit ratio in the hybrid main memory was improved. Consequently, TA-CLOCK enhances the EDP of the hybrid main memory.



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**)

Figure 9. The hit ratio of write operations in DRAM (values of 95 or less are indicated as 95). (**a**) basicmath; (**b**) blowfish; (**c**) dijkstra; (**d**) patricia; (**e**) qsort; (**f**) stringsearch.



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**)

Figure 10. Normalized energy delay product (values of 1.2 or higher are indicated as 1.2). (**a**) basicmath; (**b**) blowfish; (**c**) dijkstra; (**d**) patricia; (**e**) qsort; (**f**) stringsearch.

Finally, the analysis of the time and memory complexity required by TA-CLOCK is as follows. At first, the results of time overhead are similar to those of EDP, which is the result of Figure 10. In fact, EDP considers the time overhead for each policy, as well as energy requirement, and the result of it shows final requirements in terms of time and energy. The memory space overhead is caused by the additional bits required for managing TA-CLOCK in hybrid main memories. Table 8 describes the average memory space overhead by adjusting the ratio of DRAM to PCM in the hybrid main memory for each page replacement policy. TA-CLOCK requires 8 bits as 1 reference bit, 1 dirty bit, 3 bits for read count and 3 bits for write count, and 1 bit for each PCM page as the reference bit. Therefore, the average memory space overhead of TA-CLOCK is 0.0115% for the whole memory capacity, and memory space overhead for all CLOCK based policies is negligible considering the whole memory capacity.

Table 8. Comparison of memory space overhead for each page replacement policy.

Page Replacement Policy	DRAM	PCM	Memory Space Overhead on Average
CLOCK	2 bits	2 bits	0.0061%
CLOCK-PRO	2 bits	2 bits	0.0061%
M-CLOCK	2 bits	3 bits	0.0079%
CLOCK-DWF	7 bits	2 bits	0.0122%
TA-CLOCK	8 bits	1 bit	0.0115%

The contributions of TA-CLOCK include reduction in the number of write operations in the PCM, which positively affects the write endurance and write latency, and reduction of the energy consumption of embedded systems. Figure 11 shows the average results of all workloads in the hybrid main memory. In the experiment results, TA-CLOCK efficiently reduced the number of write operations in the PCM by reducing the number of unnecessary migrations between the DRAM and PCMs. Additionally, it improved the hit ratio of write operations in the DRAM and enhanced the EDP. As shown by the average results in Figure 11a, the TA-CLOCK reduced the number of write operations in the PCM by 71.5% on average compared with other policies. Figure 11b shows the normalized migration counts between DRAM and PCM. The TA-CLOCK reduced the number of migrations by 57.3% on average compared with CLOCK-DWF and M-CLOCK. Figure 11c shows the hit ratio of the write operations in the DRAM. The TA-CLOCK improved the write hit ratio in the DRAM by 10.4% on average, compared with other policies. Finally, Figure 11d shows the normalized EDP in the hybrid main memory. In this experiment, TA-CLOCK reduced the EDP by 38.3% on average. Based on these results, TA-CLOCK improves the limited lifespan of PCMs with enhanced EDP compared with other policies.

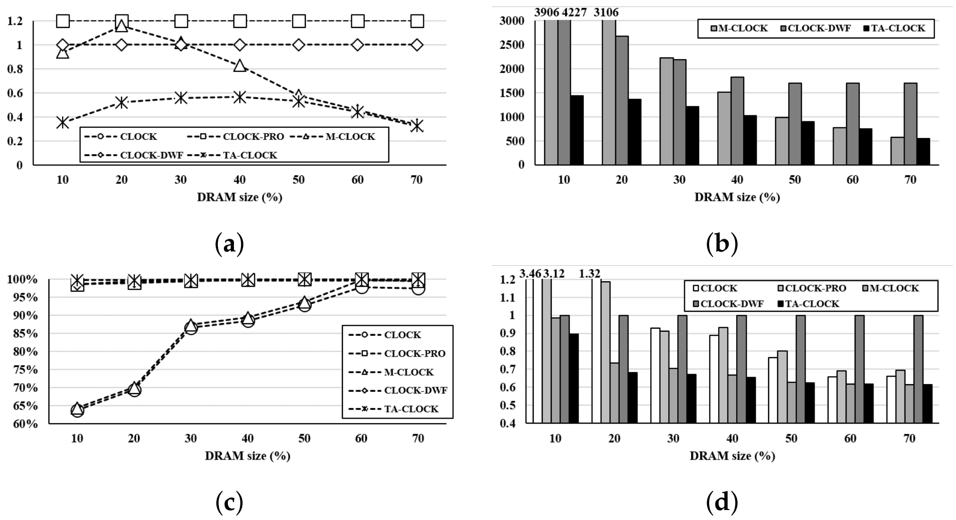


Figure 11. Average results of all workloads. (**a**) Normalized write operations in the PCM; (**b**) Number of migrations between DRAM and PCM; (**c**) The hit ratio of write operations in DRAM; (**d**) Normalized energy delay product.

5. Conclusions

In recent times, high-performance embedded systems have used PCM as their main memory. However, it is difficult to directly apply PCMs to the main memory of embedded systems owing to their disadvantages, including limited write endurance and high write latency. In this paper, we proposed TA-CLOCK, a novel page replacement policy for the hybrid main memory of high-performance embedded systems. TA-CLOCK classifies page access tendency in the hybrid main memory through access pattern analysis. To analyze the page access tendency, it uses read and write thresholds based on the page access counts of each page. By classifying the page access tendency, TA-CLOCK determines the location of the victim page, reduces unnecessary page migrations, and improves the hit ratio of write operations in the hybrid main memory. Additionally, it reduces the EDP in the hybrid main memory because it reduces the number of read and write operations in the storage. In the experiment, TA-CLOCK reduced the number of write operations in the PCM by 71.5% compared to CLOCK, CLOCK-Pro, M-CLOCK, and CLOCK-DWF, and it reduced the number of page migrations between the DRAM and PCM in the hybrid main memory by an average of 57.3%. Therefore, TA-CLOCK improves the write hit rate of DRAMs and enhances the EDP of hybrid main memories by averages of 10.4% and 38.3%, respectively, compared with existing techniques.

Author Contributions: Conceptualization, J.H.C. and J.W.K.; methodology, J.H.C. and J.W.K.; software, J.H.C. and K.M.K.; validation, J.H.C., K.M.K. and J.W.K.; formal analysis, J.H.C. and J.W.K.; investigation, J.H.C., K.M.K. and J.W.K.; resources, J.H.C. and K.M.K.; data curation, J.H.C. and K.M.K.; writing—original draft preparation, J.H.C.; writing—review and editing, J.W.K.; visualization, J.H.C. and K.M.K.; supervision, J.W.K.; project administration, J.W.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1F1A1065788).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mittal, S.; Vetter, J.S.; Li, D. A survey of architectural approaches for managing embedded DRAM and non-volatile on-chip caches. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *26*, 1524–1537. [CrossRef]

2. Xia, F.; Jiang, D.J.; Xiong, J.; Sun, N.H. A survey of phase change memory systems. *J. Comput. Sci. Technol.* **2015**, *30*, 121–144. [CrossRef]
3. Mittal, S.; Vetter, J.S. A survey of software techniques for using non-volatile memories for storage and main memory systems. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *27*, 1537–1550. [CrossRef]
4. Mittal, S.; Vetter, J.S. A survey of architectural approaches for data compression in cache and main memory systems. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *27*, 1524–1536. [CrossRef]
5. Mittal, S. A survey of power management techniques for phase change memory. *Int. J. Comput. Aided Eng. Technol.* **2016**, *8*, 424–444. [CrossRef]
6. Carballo, J.A.; Chan, W.T.J.; Gargini, P.A.; Kahng, A.B.; Nath, S. ITRS 2.0: Toward a re-framing of the Semiconductor Technology Roadmap. In Proceedings of the 2014 IEEE 32nd International Conference on Computer Design (ICCD), Seoul, Korea, 19–22 October 2014; pp. 139–146.
7. Dayarathna, M.; Wen, Y.; Fan, R. Data center energy consumption modeling: A survey. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 732–794. [CrossRef]
8. Lin, Y.J.; Yang, C.L.; Li, H.P.; Wang, C.Y.M. A buffer cache architecture for smartphones with hybrid DRAM/PCM memory. In Proceedings of the 2015 IEEE Non-Volatile Memory System and Applications Symposium (NVMSA), Hong Kong, China, 19–21 August 2015; pp. 1–6.
9. Lee, M.; Kang, D.H.; Kim, J.; Eom, Y.I. M-CLOCK: Migration-optimized page replacement algorithm for hybrid DRAM and PCM memory architecture. In Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, 13–17 April 2015; pp. 2001–2006.
10. Lee, S.; Bahn, H.; Noh, S.H. CLOCK-DWF: A write-history-aware page replacement algorithm for hybrid PCM and DRAM memory architectures. *IEEE Trans. Comput.* **2013**, *63*, 2187–2200. [CrossRef]
11. Zhang, H.; Wang, X. TriBHMM: An Energy-Efficient and Latency-Aware Hybrid Main Memory. In Proceedings of the 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, China, 16–18 December 2019; pp. 1451–1456.
12. Boukhobza, J.; Rubini, S.; Chen, R.; Shao, Z. Emerging NVM: A survey on architectural integration and research challenges. In *ACM Transactions on Design Automation of Electronic Systems (TODAES)*; ACM: New York, NY, USA, 2017; Volume 23, pp. 1–32.
13. Yu, S.; Chen, P.Y. Emerging memory technologies: Recent trends and prospects. *IEEE Solid State Circuits Mag.* **2016**, *8*, 43–56. [CrossRef]
14. Qureshi, M.K.; Srinivasan, V.; Rivers, J.A. Scalable high performance main memory system using phase-change memory technology. In Proceedings of the 36th Annual International Symposium on Computer Architecture, Austin, TX, USA, 20–24 June 2009; pp. 24–33.
15. Kim, J.G.; Kim, S.D.; Yoon, S.K. Q-Selector-Based Prefetching Method for DRAM/NVM Hybrid Main Memory System. *Electronics* **2020**, *9*, 2158. [CrossRef]
16. Sun, H.; Chen, L.; Hao, X.; Liu, C.; Ni, M. An Energy-Efficient and Fast Scheme for Hybrid Storage Class Memory in an AIoT Terminal System. *Electronics* **2020**, *9*, 1013. [CrossRef]
17. Lin, M.; Yao, Z.; Xiong, J. History-aware page replacement algorithm for NAND flash-based consumer electronics. *IEEE Trans. Consum. Electron.* **2016**, *62*, 23–29. [CrossRef]
18. Bansal, S.; Modha, D.S. CAR: Clock with Adaptive Replacement. In Proceedings of the FAST'04: Proceedings of the 3rd Usenix Conference on File and Storage Technologies, San Francisco, CA, USA, 31 March–2 April 2004; Volume 4, pp. 187–200. Available online: https://www.usenix.org/legacy/events/fast04/ (accessed on 7 May 2021).
19. Binkert, N.; Beckmann, B.; Black, G.; Reinhardt, S.K.; Saidi, A.; Basu, A.; Hestness, J.; Hower, D.R.; Krishna, T.; Sardashti, S.; et al. The gem5 simulator. In *ACM SIGARCH Computer Architecture News*; ACM: New York, NY, USA, 2011; Volume 39, pp. 1–7.
20. Jiang, S.; Chen, F.; Zhang, X. CLOCK-Pro: An Effective Improvement of the CLOCK Replacement. In Proceedings of the USENIX Annual Technical Conference, General Track, Anaheim, CA, USA, 10–15 April 2005; pp. 323–336.
21. Guthaus, M.R.; Ringenberg, J.S.; Ernst, D.; Austin, T.M.; Mudge, T.; Brown, R.B. MiBench: A free, commercially representative embedded benchmark suite. In Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538), Austin, TX, USA, 2 December 2001; pp. 3–14.
22. Niu, N.; Fu, F.; Yang, B.; Yuan, J.; Lai, F.; Zhao, C.; Wang, J. PRO: A periodical reset optimized page migration scheme for hybrid memory system. *J. Syst. Archit.* **2020**, *111*, 101786. [CrossRef]
23. Monazzah, A.M.H.; Rahmani, A.M.; Miele, A.; Dutt, N. CAST: Content-Aware STT-MRAM Cache Write Management for Different Levels of Approximation. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2020**, *39*, 4385–4398. [CrossRef]
24. Li, W.; Shuai, Z.; Xue, C.J.; Yuan, M.; Li, Q. A wear leveling aware memory allocator for both stack and heap management in pcm-based main memory systems. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 228–233.