Computing (2016) 98:279–301
DOI 10.1007/s00607-015-0464-7

Energy-aware assignment and scheduling for hybrid main memory in embedded systems

Guohui Wang[1,2] · Yong Guan[1] · Yi Wang[3,4] · Zili Shao[4]

Received: 29 May 2014 / Accepted: 27 May 2015 / Published online: 3 July 2015

Abstract In embedded systems, especially battery-driven mobile devices, energy is one of the most critical performance metrics. Due to its high density and low standby power, phase change memory (PCM), an emerging non-volatile memory device, is becoming a promising dynamic random access memory (DRAM) alternative. Recent studies have proposed the hybrid main memory architecture integrating both PCM and DRAM to fully take advantage of the properties of both memories. However, the low power performance of PCM in the hybrid main memory architecture has not been fully explored. Therefore, it becomes an interesting problem to utilize PCM and DRAM as hybrid main memory for energy optimization in embedded systems. In this paper, we present an energy optimization technique for hybrid main memory architecture. The objective is to fully utilize PCM to reduce the energy consumption while ensuring that the real-time performance of applications are guaranteed. We propose a two-phase approach to solve hybrid main memory address mapping problem. In the first phase, we calculate energy and time cost for each address based on the task models. Then the applications can be modeled as data-flow graph nodes, and different access times will associate with different energy consumption. In the second phase, for different memory types and the given timing constraint, we formulate the scheduling problem as an integer linear programming (ILP) model and obtain an optimal solution. The

✉ Yi Wang
csywang@comp.polyu.edu.hk

1 College of Information Engineering, Capital Normal University, Beijing, China

2 School of Mathematical Sciences, Capital Normal University, Beijing, China

3 Guangdong Province Key Laboratory of Popular High Performance Computers, Shenzhen University, Shenzhen, China

4 Embedded Systems and CPS Laboratory, Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong

ILP model can map a proper memory type for each address such that the total energy consumption can be minimized while the timing constraint is satisfied. In addition, we propose a heuristic approach to efficiently obtain a near-optimal solution. We conduct experiments on an ARM-based simulator. The experimental results show that our method can effectively reduce the energy consumption with the least system cost compared with the previous work.

Keywords PCM (phase change memory) · Hybrid memory architecture · Energy consumption · Embedded systems

Mathematics Subject Classification 68m07 Mathematical problems of computer architecture

1 Introduction

Energy consumption is one of the vitally important issues in the design of embedded systems. As main memory is becoming the major energy contributor to the whole system, reducing the energy consumption of main memory is significant for energy saving. Phase change memory (PCM), an emerging non-volatile memory technology, shows lots of advantages for energy optimization in embedded systems. It can reduce the energy consumption by consuming less leakage power compared to the conventional volatile memory, such as dynamic random access memory (DRAM). However, PCM incurs longer write energy and less write endurance. Therefore, a hybrid main memory architecture that incorporates PCM and DRAM could exploit both of their advantages and conceal the constraints for energy reduction in embedded systems.

The comparison for the characteristics of DRAM and PCM is shown in Table 1. PCM is similar to DRAM on bitwise operation, and it has better energy consumption than DRAM due to its better idling power. Therefore, it becomes a promising candidate to replace DRAM as main memory in the near future [1]. As current implementation of PCM still has very limited capacity, hybrid PCM and DRAM main memory architecture could be a possible solution that can directly integrate PCM into the existing system to reduce energy consumption.

Hybrid PCM and DRAM main memory architectures have been proposed in the literature [2,3]. PCM and DRAM are attached to the system bus and exposed to the operating system. They work side-by-side and provide the same address space to the operating system. Most existing work aims to optimize the page transfers between PCM and DRAM to improve the lifetime of PCM. Their design objectives are not to reduce the energy consumption of the hybrid main memory architecture. As PCM has very good performance in terms of energy consumption, it becomes a critical issue in embedded systems to fully utilize PCM to reduce the total energy demand of the system.

This paper presents energy optimization techniques for PCM and DRAM hybrid main memory system architecture. PCM is used to replace DRAM as much as possible, so that the energy consumption can be reduced by utilizing the non-volatile memory of PCM. We target at embedded applications, which normally have real-

Table 1 The comparison of DRAM and PCM [1,4]

Attributes	DRAM	PCM
Non-volatile	No	Yes
Erase required	bit	bit
Static power	1 W/GB	0.1 W/GB
Write latency	50 ns	180 ns
Write energy	0.1 nJ/bit	0.5 nJ/bit
Read latency	50 ns	70 ns
Read energy	0.1 nJ/bit	0.02 nJ/bit
Endurance	∞	10^8
Data retention	ms	Not f (cycles)

time requirements. The tradeoff between the energy saving and the time overhead needs to be jointly considered for a real-time task. The objective of this paper is to minimize energy consumption on PCM and the deadline of real-time applications. In this paper, the target problem is formulated as a hybrid memory address mapping problem. We present both an integer linear programming (ILP)-based approach and a heuristic approach to effectively solve the address allocation problem for hybrid main memory architecture to optimize energy consumption and help the application user to find the best size configurations of PCM and DRAM for a specific embedded application.

To solve the problem, we first present an ILP model, that can optimally map an address to the hybrid main memory architecture. This ILP model based approach can generate an optimal solution where the total energy consumption can be minimized while the timing constraint is satisfied. Since ILP model based approach takes a very long time to obtain the optimal solution, we also propose a practical algorithm to produce a near-optimal solution for the hybrid memory address mapping problem. In this algorithm, it maps each address with a minimal energy cost in different memory types and then interactively changes the address mapping so that the timing constraint can be satisfied with the minimal energy consumption.

We conduct a set of experiments on an ARM-based CPU simulator. We compared the experimental results of three algorithms, the greedy algorithm, the proposed ILP model based approach, and the heuristic-based hybrid memory address mapping algorithm. Three performance metrics, energy consumption, memory distribution, and time cost, are used to evaluate the performance. From the experimental results, the hybrid memory mapping algorithm achieves better performance compared to the greedy algorithm, and obtains near-optimal solution similar to that of ILP model based approach. In terms of time cost to generate the results, the hybrid memory address mapping algorithm can efficiently obtain the solution. These results prove the effectiveness of the proposed approaches.

The remainder of this paper is organized as follows. Section 2 introduces the background and the problem formulation of this paper. Then, Sect. 3 presents the proposed approaches to solve the hybrid memory address mapping problem. In Sect. 4, we present the experimental results with analysis. Finally, we conclude the paper in Sect. 6.

2 Background and problem formulation

In this section, we first introduce the background information of PCM. Then we present the system architecture. Finally, we present the target problem to be solved in the paper.

2.1 Phase change memory (PCM)

PCM is a type of non-volatile random-access memory. A PCM cell uses a special material, called phase change material, to remember a bit. In a PCM chip, cells are organized in the two-dimensional array just as shown in Fig. 1. The organization is similar to that of DRAM. In Fig. 1, the basic structure of PCM memory cell, consists of a standard NMOS transistor and a phase change device. A typical PCM cell in a phase change device is a small volume of phase change material, GST (Ge2Sb2Te5), which can exist in either a crystalline or amorphous state and two electrodes attached to the chalcogenide, one on each side.

PCM exploits the differences in the electrical resistivity of GST between the two states to store information. The phase change in GST can be achieved by heating a region of phase change material to a high temperature threshold using electrical-pulse generated Joule heat [5]. As shown in Fig. 2, for the SET operation, when GST is heated to a temperature between the crystallization temperature ($\sim$300 °C) and the melting temperature ($\sim$600 °C) over a period of time, GST turns into the crystalline state which corresponds to a logic '1'. For the RESET operation, when GST is heated above the melting point and quenched quickly, GST turns into the amorphous state which corresponds to logic '0'.

2.2 System architecture

PCM as one kind of non-volatile memory has the advantages in terms of excellent economy and high density. Therefore, it is promised as candidates to be employed as

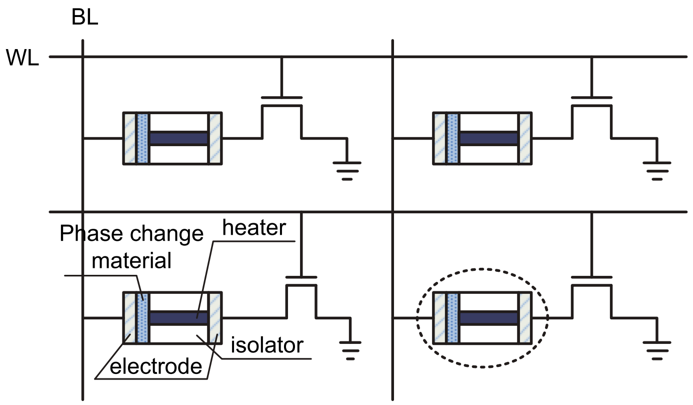


Fig. 1 A PCM cell array and the structure of a PCM cell [5]

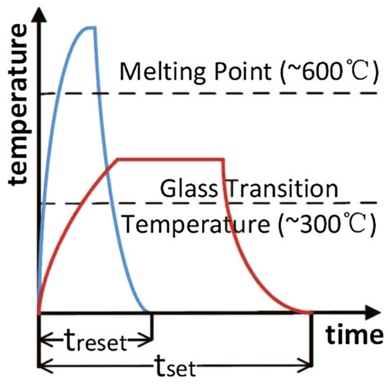


Fig. 2 The SET and RESET operations of PCM [6]

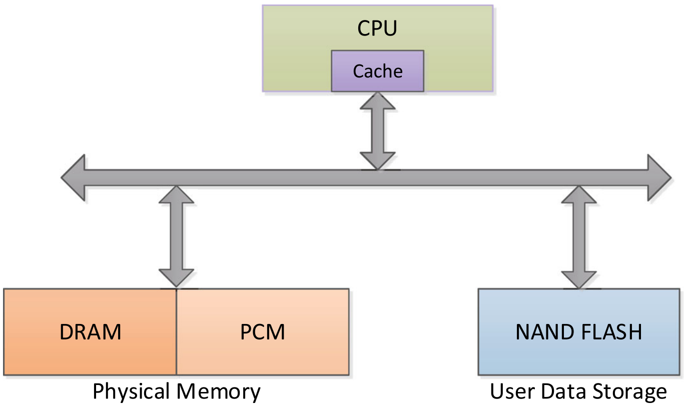


Fig. 3 A typical system architecture with hybrid main memory in embedded systems

main memory. PCM also has several disadvantages, such as long access latency. This makes PCM infeasible to be directly used as main memory. In order to fully utilize the properties of PCM, the hybrid main memory architecture that incorporates both PCM and DRAM, becomes a viable solution.

There have been several system architectures for hybrid main memory systems. Dhiman et al. [3] presented a hybrid PCM and DRAM memory architecture called PDRAM. This architecture is widely used in the literature, and we also adopt this architecture in our design. Figure 3 depicts the system architecture with the PCM and DRAM hybrid main memory in embedded system. The hybrid PCM and DRAM memory serve as the main memory. Both PCM and DRAM are attached to the memory bus, and they could be managed under a single physical address space. In this architecture, NAND flash is used as the secondary storage to store user data. Different from DRAM-based main memory, this hybrid main memory can store code and user data in the non-volatile PCM. It can reduce the energy consumption because PCM consumes less leakage power compared to DRAM. The total execution time will be extended, since PCM has much longer access latency.

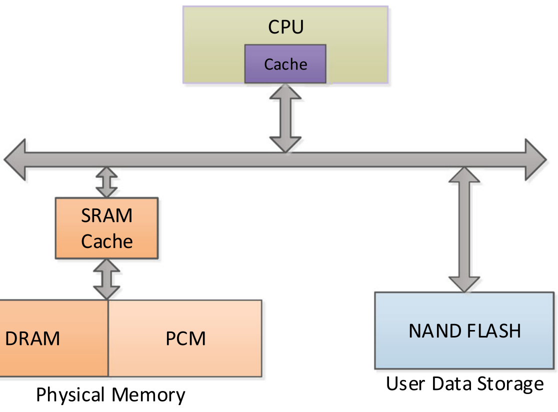


Fig. 4 A typical cache-based architecture with hybrid main memory

For the second type of hybrid system architecture, Qureshi et al. [2] presented an architecture that uses DRAM as the last level cache memory of PCM main memory. Zhou et al. [7] presented cache partitioning and replacement algorithms under this architecture. Figure 4 shows a typical cache-based architecture for hybrid main memory. This architecture can also utilize the features of PCM. The challenging problem in this architecture is the data caching policy between PCM and DRAM. With the increasing capacity of PCM, it could replace the DRAM as the main memory. In this paper, we focus on the first type of system architecture. We aim to design a data allocation strategy to assign data to either PCM or DRAM.

2.3 Problems formulation

In embedded systems, hardware and software can be specially designed and optimized for specific applications. The tasks run in embedded system normally require real-time performance. The target problem of this paper is that PCM is used to replace DRAM as much as possible, so the system energy can be reduced by utilizing the non-volatile memory of PCM with a timing constraint. It is an important problem to map an address to a proper type of memory such that the total energy can be minimized while the timing constraint is satisfied.

In this work, data-flow graph (DFG) is employed to model the *hybrid memory address mapping problem*. The target application is defined as an address-weighted directed graph (DFG) $G = \langle A, E, d\rangle$, where $A = \langle a_1, a_2, \ldots, a_N\rangle$ is the set of nodes, $E \subseteq A \times A$ is the edge set that defines the precedence relations among nodes in A, and $d(e)$ represents the number of delays for an edge e. Given an edge, $e = a \rightarrow v$, $d(e)$ means the data used as inputs in node a have previously been generated by node $d(e)$ iterations.

A special purpose hybrid memory architecture consists of different type of memory. Assume there are Q different memory types, $M_1, M_2, \ldots, M_Q$. $T(a_i)$ is used to represent the access time latency of each node $a_i \in A$ for different memory types: $T(a_i) = \{t_1(i), t_2(i), \ldots, t_M(i)\}$ where $t_j(i)$ denotes the access time latency of a_i for memory type P_j. $C(a_i) = \{c_1(i), c_2(i), \ldots, c_M(i)\}$ where $c_j(i)$ denotes the access energy consumption of a_i for memory type M_j. An address mapping process for a DFG is to assign a memory type to each address. Given an address mapping table of a DFG, we define the total energy cost to be the summation of access energy consumption of all address space. This model can deal with the case when all address space only maps to one type of memory. In this situation, we can set the execution time of a task as "infinite" if the address of this task can not be mapped to a memory type, where"infinite" means a value greater than the given timing constraint. Therefore, this address can not be mapped to this memory type because it does not satisfy the timing constraint.

Based on the above definitions, we formulate the target *hybrid memory address mapping problem* as follows:

Given Q different memory types: $M_1, M_2, \ldots, M_Q$, a DFG $G = \langle A, E, d\rangle$, where $V = \langle a_1, a_2, \ldots, a_N\rangle$, $T(a_i) = \{t_1(i), t_2(i), \ldots, t_Q(i)\}$ and $C(a_i) = \{c_1(i), c_2(i), \ldots, c_Q(i)\}$ for each node $a_i \in V$, and a timing constraint TC, the objective is to minimize the total cost under the timing constraint.

3 Energy-aware hybrid memory address mapping algorithms

The objective of this paper is to exploit the advantages of PCM and DRAM to reduce energy consumption. In this section, we present the proposed approach. We firstly introduce energy model and propose an ILP model to solve the problem. Integer linear programming provides a mechanism to get the optimal solution of a problem in which all of its constraints can be formulated as linear constraints of integer variables. The address mapping table which matches the given conditions can be obtained by the proposed approach. Based on the mapping table we can get the minimum energy consumption with the given timing constraint. Since the ILP-based approach needs to take a long time to obtain the optimal solution, we propose an algorithm called *hybrid memory address mapping algorithm*. This heuristic algorithm can efficiently obtain the address mapping table with less system cost.

3.1 Energy and time model

Energy consumption in main memory can be divided into two parts. They are static energy consumption and dynamic energy consumption. Static energy consumption includes the leakage energy and the refresh energy. The leakage power is the energy consumed even when memory is idle, and the leakage power of PCM is negligible compared to that of DRAM. DRAM memory cells store data in small capacitors that lose their charge over time and must be recharged. The recharged process is called refresh which is used for retaining its data.

In this paper, the notations used in this paper are summarized in Table 2. The time costs for each address on DRAM and PCM are defined as $T_d(i)$ and $T_p(i)$, respectively.

Table 2 The notations used in this paper

Notations	Description
a_i	The i-th address
$\widehat{e}_{dr}/\widehat{e}_{dw}$	Energy consumption of one bit read/write on DRAM
$\widehat{e}_{pr}/\widehat{e}_{pw}$	Energy consumption of one bit read/write on PCM
e_{dl}/e_{pl}	Leakage power on DRAM/PCM
BW	Bandwidth of each read/write for the systems
$N_r(i)/N_w(i)$	The number of reads/writes on a_i
S_d/S_p	The size of DRAM/PCM used on the systems
t_{dr}/t_{dw}	Time latency of each read/write on DRAM
t_{pr}/t_{pw}	Time latency of each read/write on PCM
e_{pr}/e_{pw}	Energy consumption of each read/write on PCM
$T_d(i)/T_p(i)$	Time cost for a_i access on DRAM/PCM
T_{total}	Total time cost for all address space
$E_d(i)/E_p(i)$	Energy consumption of a_i on DRAM/PCM
E_{total}	Total energy consumption for all address space
TC	Timing constraint

The energy consumptions for each address on DRAM and PCM are defined as $E_d(i)$ and $E_p(i)$, respectively.

Let $N_r(i)$ and $N_w(i)$ be the read and write numbers for the address i. Time cost for address a_i accessing DRAM $T_d(i)$ can be obtained by Eq. 1. Similarly, time cost for address a_i accessing PCM $T_p(i)$ can be obtained by Eq. 2.

$$T_d(i) = t_{dr} \times N_r(i) + t_{dw} \times N_w(i) \tag{1}$$

$$T_p(i) = t_{pr} \times N_r(i) + t_{pw} \times N_w(i) \tag{2}$$

The energy consumption for the A_i access on DRAM $E_d(i)$ can be calculated based on Eq. 3.

$$E_d(i) = e_{dr} \times N_r(i) + e_{dw} \times N_w(i) \tag{3}$$

where

$$e_{dr} = \widehat{e}_{dr} \times BW + e_{dl} \times S_d \times t_{dr}$$
$$e_{dw} = \widehat{e}_{dw} \times BW + e_{dl} \times S_d \times t_{dw}$$

The energy consumption for address a_i accessing PCM $E_p(i)$ can be obtained by Eq. 4.

$$E_p(i) = e_{pr} \times N_r(i) + e_{pw} \times N_w(i) \tag{4}$$

Table 3 The time cost and energy consumption for address space for different types of memory

Address	DRAM (M_1)		PCM (M_2)	
	Time (T_1)	Energy (E_1)	Time (T_2)	Energy (E_2)
a_1	$T_{M_1}(1)$	$E_{M_1}(1)$	$T_{M_2}(1)$	$E_{M_2}(1)$
a_2	$T_{M_1}(2)$	$E_{M_1}(2)$	$T_{M_2}(2)$	$E_{M_2}(2)$
$a_{\dots}$	$T_{M_1}(\cdots)$	$E_{M_1}(\cdots)$	$T_{M_2}(\cdots)$	$E_{M_2}(\cdots)$
a_N	$T_{M_1}(N)$	$E_{M_1}(N)$	$T_{M_2}(N)$	$E_{M_2}(N)$

where

$$e_{pr} = \widehat{e}_{pr} \times BW + e_{pl} \times S_p \times t_{pr}$$
$$e_{pw} = \widehat{e}_{pw} \times BW + e_{pl} \times S_p \times t_{pw}$$

Based on the above four equations, we summarize the time cost and the energy consumption of the hybrid memory architecture. The models are presented in Table 3. This table describes all attributes for each address using different types of memory. From the table, the given DFG is a simple path $a_1 \to a_2 \to, \cdots, \to a_N$.

3.2 ILP formulations

In this section, an ILP model is presented to obtain an optimal solution. The objective is to find the optimal location scheme for minimizing the system energy consumption within the timing constraint for each task. Given two different memory types, a DFG $G = \langle A, E, d \rangle$, and a timing constraint *TC*, we can formulate an ILP model as follows.

$$Min\ E_{total} = \sum_{i=1}^{N} (E_d(i) \times P(i)) + \sum_{i=1}^{N} (E_p(i) \times \overline{P}(i)) \tag{5}$$

Subject to

$$\begin{cases} T_{total} = \sum_{i=1}^{N} (T_d(i) \times P(i)) + \sum_{i=1}^{N} (T_p(i) \times \overline{P}(i)) \le TC \\ \sum_{i=0}^{N} P(i) \times BW \le S_d \\ \sum_{i=0}^{N} \overline{P(i)} \times BW \le S_p \\ P(i) + \overline{P(i)} = 1 \\ P(i) \in \{0, 1\} \\ \overline{P(i)} \in \{0, 1\} \end{cases} \tag{6}$$

For each address a_i, there are two variables $P(i)$ and $\overline{P(i)}$. Variable $P(i)$ represents whether task a_i is located in DRAM, which is given in Eq. 7. Variable $\overline{P(i)}$ represents whether task a_i is located in PRAM, which can be determined by Eq. 8.

$$\begin{cases} p(i) = 0, & \textit{if address } a_i \textit{ is not assigned in DRAM} \\ p(i) = 1, & \textit{if address } a_i \textit{ is assigned in DRAM} \end{cases} \tag{7}$$

$$\begin{cases} \overline{p(i)} = 0, & \textit{if address } a_i \textit{ is not assigned in PRAM} \\ \overline{p(i)} = 1, & \textit{if address } a_i \textit{ is assigned in PRAM} \end{cases} \tag{8}$$

In the ILP model, the objective function seeks to minimize the maximum value of all address nodes. In order to generate the optimal solution, the ILP model considers the following constraints:

1. *The timing constraint*. Constraint 1 represents that the execution time of the whole task is equal to or less than the timing constraint.
2. *The bound of the memory size*. These two memories can be considered together as a whole memory with the size $S_d + S_P$. And each kind of memory must be assigned a physical address that is less than or equal to its size.
3. *Location of the each address*. For each address, it is either assigned to the DRAM or to the PRAM.

As the ILP model integrates several constraints that are showed in Eq. 6, these constraints greatly reduce search space for finding the optimal solution. The proposed approach can efficiently solve the problem and provide the optimal address location scheme to ensure that the energy consumption is minimized within the timing constraint.

To illustrate how to generate the optimal mapping table, we use Fig. 5 as an example. Based on the initial time cost and energy consumption for address space for different types of memory in Fig. 5a and the timing constraint $TC = 24$, we can obtain the mapping method for minimal energy consumption within the timing constraint.

For example, based on the data in Fig. 5a, we reorder the node in the table according to the running time in PCM (in Fig. 5b) and build a tree (in Fig. 5c). Then, we employed branch and bound approach to find the optimal solution. The details of approach is given at Algorithm 1. The calculation steps and the results are presented in Fig. 5d.

3.3 Hybrid memory address mapping algorithm

The hybrid memory address mapping problem is NP-complete. In this section, we present a new mapping algorithm called *hybrid memory address mapping algorithm* to solve this problem for hybrid memory architecture. The basic idea is to assign each address with the best cost ratio, which is determined by each address and each of its unmarked types. The best cost ratio can be calculated by Eq. 9,

$$\textit{costRatio} = \textit{IncreaseEnergy/ReduceTime} \tag{9}$$

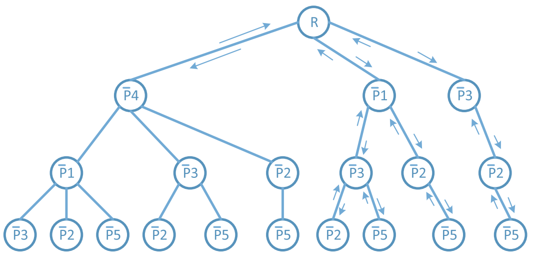
Address	DRAM		PCM	
	Time	Energy	Time	Energy
a_1	2	14	6	2
a_2	4	17	10	6
a_3	3	16	8	4
a_4	1	12	5	1
a_5	5	19	11	7

*Timing constraint TC = 24

(a)

$$\begin{tikzcd} a_4 \arrow[r] & a_1 \arrow[r] & a_3 \arrow[r] & a_2 \arrow[r] & a_5 \end{tikzcd}$$

(b)



(c)

STEP	PCM	DRAM	Time	Energy	STEP	PCM	DRM	Time	Energy
S1	a3,a2,a5	a1,a4	32		S8	a4,a2,a5	a1,a3	31	
S2	a3,a2	a1,a4,a5	26		S9	a4,a2	a1,a3,a5	25	
S3	a3	a1,a2,a4,a5	20	66	S10	a4,a3,a5	a1,a2	30	
S4	a1,a2,a5	a3,a4	31		S11	a4,a3	a1,,a2,a5	24	55
S5	a1,a2	a3,a4,a5	25		S12	a4,a1,a5	a2,a3	29	
S6	a1,a3,a5		29		S13	a4,a1	a2,,a3,a5	33	55
S7	a1,a3		24	54					

⬭ To satisfy the timing constraint

○ Minimum energy

(d)

Fig. 5 An example of the proposed ILP-based approach

```
Algorithm 1 Branch and Bound Algorithm.
Input:
    Address node ordered as $a_4, a_1, a_3, a_2, a_5$, and timing constraint ($TC$).
Output:
    An optimal address mapping table.
 1: for i = 1 to N
 2:   $T = \sum_{i=0}^{N} (P_i \times T_1(i) + \overline{P_i} \times T_2(i))$
 3:   if $T > T_{min}$
 4:     Back up to parent node
 5:   else
 6:     $E = \sum_{i=0}^{n} (P_i \times E_1(i) + \overline{P_i} \times E_2(i))$, and save $T$, $E$, $P_i$ and $\overline{P_i}$.
 7:     then back to the recent bifurcation point
 8: end for
 9: Finding the minimum energy in the saving nodes
10: $E$ is the minimum system energy consumption within timing constraint $TC$ and an optimal address
    mapping table can be obtained.
```

ReduceTime and IncreaseEnergy are the corresponding reducing time and the increasing energy if a_i is changed its current mapping from one type of memory to another type of memory. The ratio is used to represent the average increasing energy per reducing time unit. Since we want to reduce the energy consumption with the minimal execution time increase, we pick up the nodes with the maximal ratio. We map the

Algorithm 2 Hybrid Memory Address Mapping Algorithm.

```
Input:
    $M$ different types of Memory, an address path $a_1 \rightarrow a_2 \rightarrow, ..., \rightarrow a_N$, and timing constraint ($TC$).
Output:
    An optimal address mapping table.
1: for i = 1 to N
2:   for j = 1 to (Q-1)
3:     for k = j+1 to Q
4:       $costRatio(i, j) = {|E(j) - E(k)|}/{T(k) - T(j)}$.
5:     end for
6:   end for
7: end for
8: Reorganize the address sequence based on the costRatio,
9: Map the address to suitable type memory from the largest costRatio to the small one. At the same time,
   the total execution time needs to be recorded.
10: while $T_{total} \geq TC$
11:   Moving the largest time consumption address to the time-saving type memory.
12:   Recalculate $E_{total}$ and $T_{total}$.
13: end while
14: $E_{total}$ is the minimum system energy consumption within timing constraint $TC$ and an optimal address
    mapping table can be obtained.
```

node with maximal ratio to the suitable memory type, and mark it. The procedure is repeated until the timing constraint is satisfied or we can not reduce the energy consumption any more. The details of algorithm are given at Algorithm 2.

We use an example in Fig. 6 to illustrate our approach. The data given in Fig. 5a are used in this example. Assuming that the maximum of PCM memory has four cells. In the first step, we calculate the *costRatio* by Eq. 9. Then we reorder the nodes in the table according to *costRatio*, and this step is shown in Fig. 6a. In the second step, four addresses which have the maximum value of *costRatio* are selected to map to PCM. In the last step, the timing cost T and energy consumption E are calculated. If the

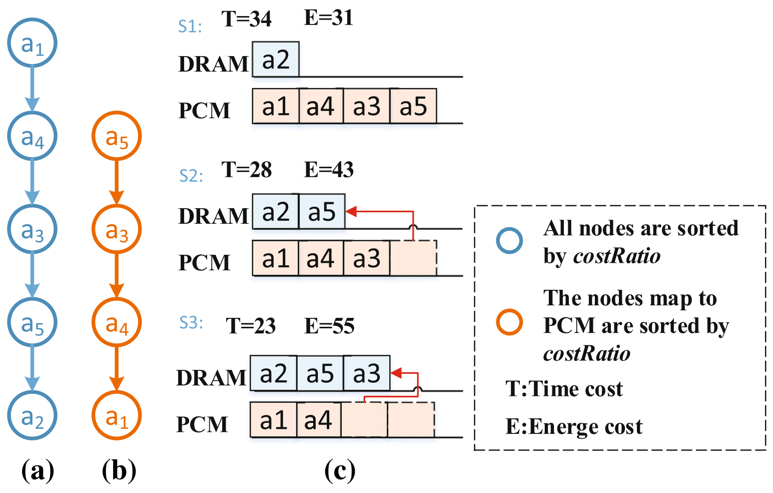


Fig. 6 An example of the proposed hybrid memory address mapping algorithm

timing cost T is larger than timing constraint TC, then the first node in Fig. 6b will be moved to DRAM until the timing consumption T is smaller than or equal to timing constraint TC. The details are presented in Fig. 6c.

4 Evaluation

This section presents experimental results with analysis. We compare and evaluate our proposed approaches with the previous work on a set of benchmarks. The objective of the evaluation is to quantify the gains of our approaches over the representative work in terms of three performance metrics: energy consumption, memory distribution and time cost. In this section, we first introduce the experimental setup. Then we present the experimental results and discussion.

4.1 Experimental setup

In the experiments, memory access traces are captured through a modified version of SkyEye [8]. SkyEye can simulate ARM CPU based hardware system. We operate SkyEye on a PC with Core-i5 CPU and 2GB memory running Ubuntu operating system. The traces in our experiments include six different applications: *basicmath small*, *bit*, *dijkstra large*, *pbmsrch large*, *qsort* and *sha* running on an ARM-based embedded system from MiBench [9]. These applications are typical operations in embedded systems. The characteristics of the benchmarks are given in Table 4.

In the experiments, we adopted the hybrid memory system architecture depicted at Fig. 3. Two types of memory, DRAM and PCM, are used in this architecture. PCM is used to replace a small part of DRAM. They are sharing the same physical address space. We set the total size of the main memory as 512 MB. When dynamically adjusting the distribution of the memory size for hybrid main memory system, the leakage power of PCM also will be slightly changed. In order to simplify the calculation, we consider the leakage power of PCM with the fixed value. The leakage power is calculated based on the results with the size configurations of 508 MB of DRAM and 4 MB of PCM.

Table 4 The characteristics of the benchmarks

Benchmarks	Memory size (B)	Memory access count			Ratio of read operations (reads:total) (%)
		Reads	Writes	Total	
basicmath small	2,476	15,800,688	10,413,652	26,214,340	60.3
bit	2,472	4,659,609	1,107,433	5,767,042	80.8
dijkstra large	893,016	4,273,444	3,066,522	7,339,966	58.2
pbmsrch large	30,388	67,827	358,097	425,924	15.9
qsort	1,280,604	3,909,870	3,430,102	7,339,972	53.3
sha	9,164	56,956	16,698	73,654	77.3

4.2 Results and discussion

In this section, we present and discuss the experimental results. We compare the proposed ILP-based approach and the proposed heuristic approach with the Greedy algorithm in terms of energy consumption, memory distribution, and time cost.

4.2.1 Energy consumption and memory distribution

Figure 7 presents the energy consumption of each benchmark under different timing constraints. The horizontal axis indicates timing constraints, and the vertical axis shows energy consumption. The experimental results show the clear trend that energy consumption decreases with the increasing of the timing constraint. When the timing constraint increases by 5 %, the energy consumption will be decreased with more than 10 %. In Fig. 7, the experimental results also show that the proposed heuristic-based approach can achieve near-optimal results for various benchmarks compared with the optimal results obtained by the ILP model. Our hybrid memory address mapping algorithm also has better performance than the greedy algorithm.

The memory distributions for each benchmark with different timing constraints are shown in Table 5. In the table, Column "TC" represents the given timing constraint. To set a baseline, the first timing constraint for each benchmark is the minimum execution time. The results obtained from different algorithms show that, with the minimum timing constraint, all addresses are mapped to DRAM. When the timing constraint is increased from 5 to 20 %, the memory distribution is significantly changed. The larger timing constraints, the greater PCM will be occupied. For benchmark *qsort* with greedy algorithm and hybrid memory address mapping algorithm, almost all the memory space of PCM is utilized. That is because, write operations are time consuming, and these operations will result in constant access to the PCM address.

4.2.2 Time cost

Table 6 presents the average time cost to obtain the results for each benchmark. From the table, the ILP model based approach takes very long time to get the results compared with the other two approaches even when the given address numbers are not very big. When the address number is more than 223,254, the ILP model based approach even can not get the result. Among these three algorithms, the proposed hybrid memory address mapping algorithm can achieve the best performance.

In order to present how the timing constraint affects the system time cost, Fig. 8 presents the time cost for each benchmark under three different algorithms. The horizontal axis indicates timing constraints, and the vertical axis represents time cost. It can be seen that ILP model based approach is most time consuming and the proposed hybrid memory address mapping algorithm can save some system time cost than the greedy algorithm. The system cost of ILP model based approach is relatively balanced with different timing constraints. But the system costs of the greedy algorithm and the hybrid memory address mapping algorithm with the minimum timing constraint is much larger than the system costs with other timing constraints.

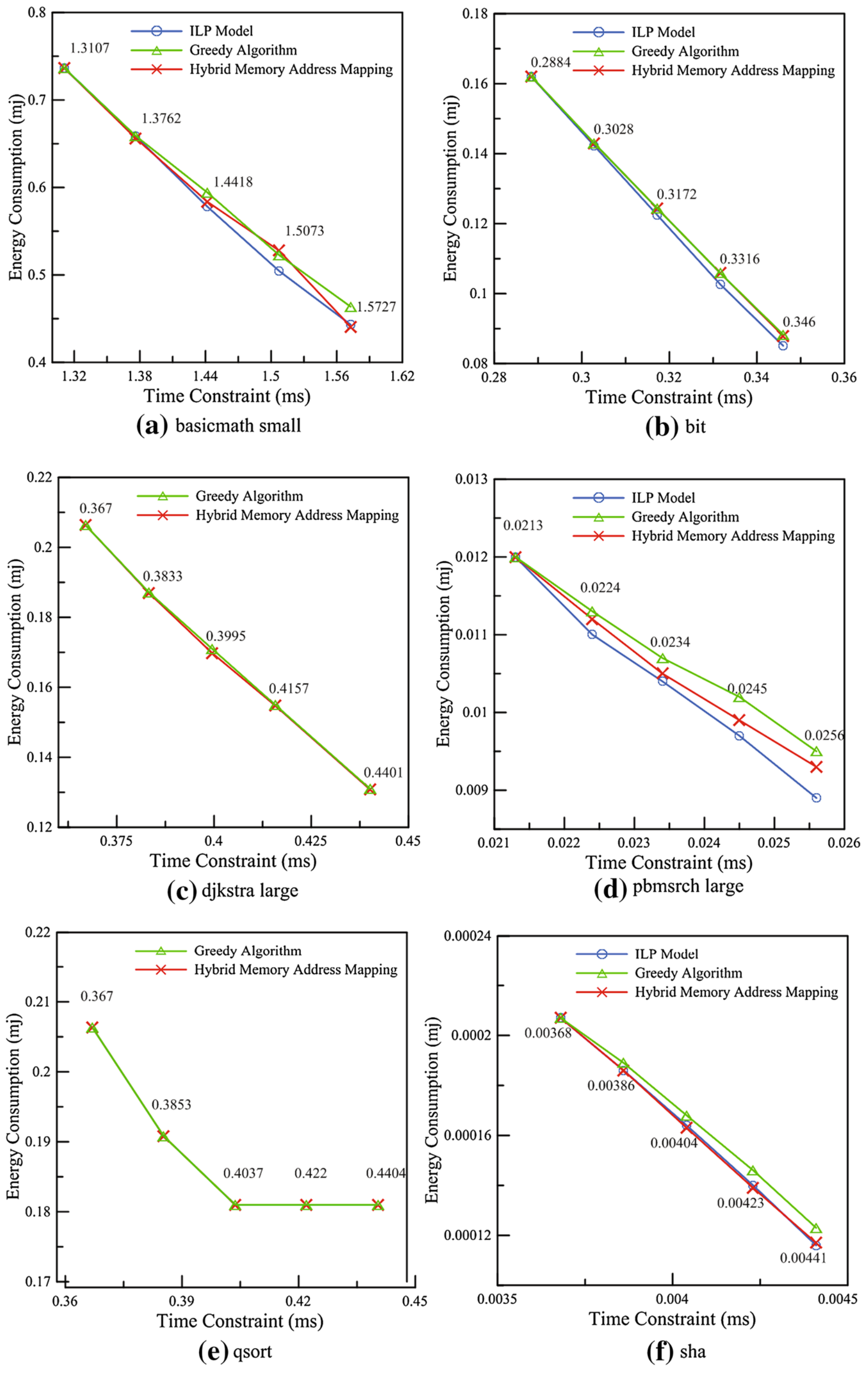


Fig. 7 The energy consumptions for each benchmark

Table 5 Levels of energy consumption for each benchmark

Benchmark	TC (ms)	ILP Model		Greedy algorithm		Hybrid memory address mapping	
		DRAM(B)	PCM(B)	DRAM(B)	PCM(B)	DRAM (B)	PCM(B)
basicmath small	1.3107	2,476	0	2,476	0	2,476	0
	1.3762	1,316	1,160	724	1,752	568	1,908
	1.4418	1,284	1,192	572	1,904	476	2,000
	1.5073	1,180	1,296	500	1,976	452	2,024
	1.5727	1,196	1,280	472	2,004	436	2,040
bit	0.2884	2,472	0	2,472	0	2,472	0
	0.3028	1,324	1,148	588	1,884	560	1,912
	0.3172	1,272	1,200	524	1,949	496	1,976
	0.3316	1,276	1,196	488	1,984	460	2,012
	0.3460	1,332	1,140	468	2,004	440	2,032
dijkstra large	0.3670	–	–	893,016	0	893,016	0
	0.3833	–	–	891,012	2,004	891,008	2,008
	0.3995	–	–	891,004	2,012	891,000	2,016
	0.4157	–	–	890,996	2,020	890,988	2,028
	0.4401	–	–	890,984	2,032	890,976	2,040
pbmsrch large	0.0213	30,388	0	30,388	0	30,388	0
	0.0224	29,420	968	29,288	1,100	29,264	1,124
	0.0234	29,268	1,120	29,124	1,264	29,104	1,284
	0.0245	29,228	1,260	28,960	1,428	28,948	1,440
	0.0256	28,980	1,408	28,796	1,592	28,782	1,606
qsort	0.3670	–	–	1,280,604	0	1,280,604	0
	0.3853	–	–	1,278,564	2,040	1,278,564	2,040
	0.4037	–	–	1,278,556	2,048	1,278,556	2,048
	0.4220	–	–	1,278,556	2,048	1,278,556	2,048
	0.4404	–	–	1,278,556	2,048	1,278,556	2,048
sha	0.00368	9,164	0	9,164	0	9,164	0
	0.00386	8,012	1,152	7,452	1,712	7,348	1,816
	0.00404	7,980	1,184	7,368	1,796	7,296	1,868
	0.00423	7,892	1,272	7,312	1,852	7,248	1,916
	0.00441	7,900	1,264	7,264	1,900	7,200	1,964

The experimental results reflect the effectiveness of the proposed approaches. For example, Figs. 7a and 8a show the result of energy consumption and system cost for the benchmark *basicmath*. From Fig. 7a, the minimum time constraint is 1.3107 ms, and the maximum energy consumption is 0.7367 mJ. When the timing constraint increases by 5 % at 1.3762 ms, the saving in energy consumption is 10.61 %. When the timing constraint increases by 15 % at 1.5073 ms, the system time cost is 27.0258, 0.0368 and 0.0092 s for ILP model based approach, the greedy algorithm and the hybrid memory address mapping algorithm, respectively.

Table 6 The average time cost for each benchmark

Benchmarks	Address num.	Average cost (s)		
		ILP model	Greedy algorithm	Hybrid memory address mapping
basicmath small	619	16.1532	0.0429	0.0149
bit	618	19.1918	0.0692	0.0195
dijkstra large	223,254	–	0.8582	0.4468
pbmsrch large	7,597	3612.93	0.0711	0.0261
qsort	320,151	–	0.6801	0.6503
sha	2,291	137.3502	0.0537	0.0271

From the experimental results, we can conclude that the hybrid memory address mapping algorithm has better performance compared with the greedy algorithm. For DFGs with limited number of addresses, ILP model based approach could be used to obtain the optimal solution. While DFGs become too big for the ILP model to solve, the proposed heuristic algorithm can still efficiently give results. For solving the general problem, our hybrid memory address mapping algorithm is recommended to be used, which gives near-optimal results with less system cost compared with the ILP model. The proposed hybrid memory address mapping algorithm not only gives a better solution to the hybrid address mapping problem, but also has less system cost compared with the greedy algorithm. These results prove the effectiveness of the proposed scheme.

4.2.3 Comparison for cache-based architecture

This section presents the experimental results for the cache-based hybrid PCM architecture. In this architecture, SRAM is used as the cache to store the instructions and data from main memory. We applied our technique to the cache-based hybrid PCM architecture by placing a SRAM cache between hybrid main memory and the CPU. Based on the parameters of SRAM [10], we developed cache-based DRAM/PCM architecture simulator. The size of the cache is set to 4 KB. The cache replacement algorithm applied LRU (least recently used), which are shown in Table 4. We compared the time cost and energy consumption for the hybrid architecture and the cache-based hybrid architecture. The experimental results are presented in Table 7.

From the experimental results, the cache-based hybrid architecture can reduce the time cost and energy consumption for most of benchmarks. Compared with the hybrid architecture, the cache-based hybrid architecture can achieve an average reduction of 58.2 % for time cost and an average reduction of 89.5 % for energy consumption. We notice that, for benchmarks *qsort* and *sha*, the performance of cache-based hybrid architecture is worse than that of hybrid architecture. We found that the hit ratio of benchmark *qsort* is only 55.5 % and that of benchmark *sha* is 84.1 %. The low hit ratio of these benchmarks shows that it experiences a large number of hit misses. The

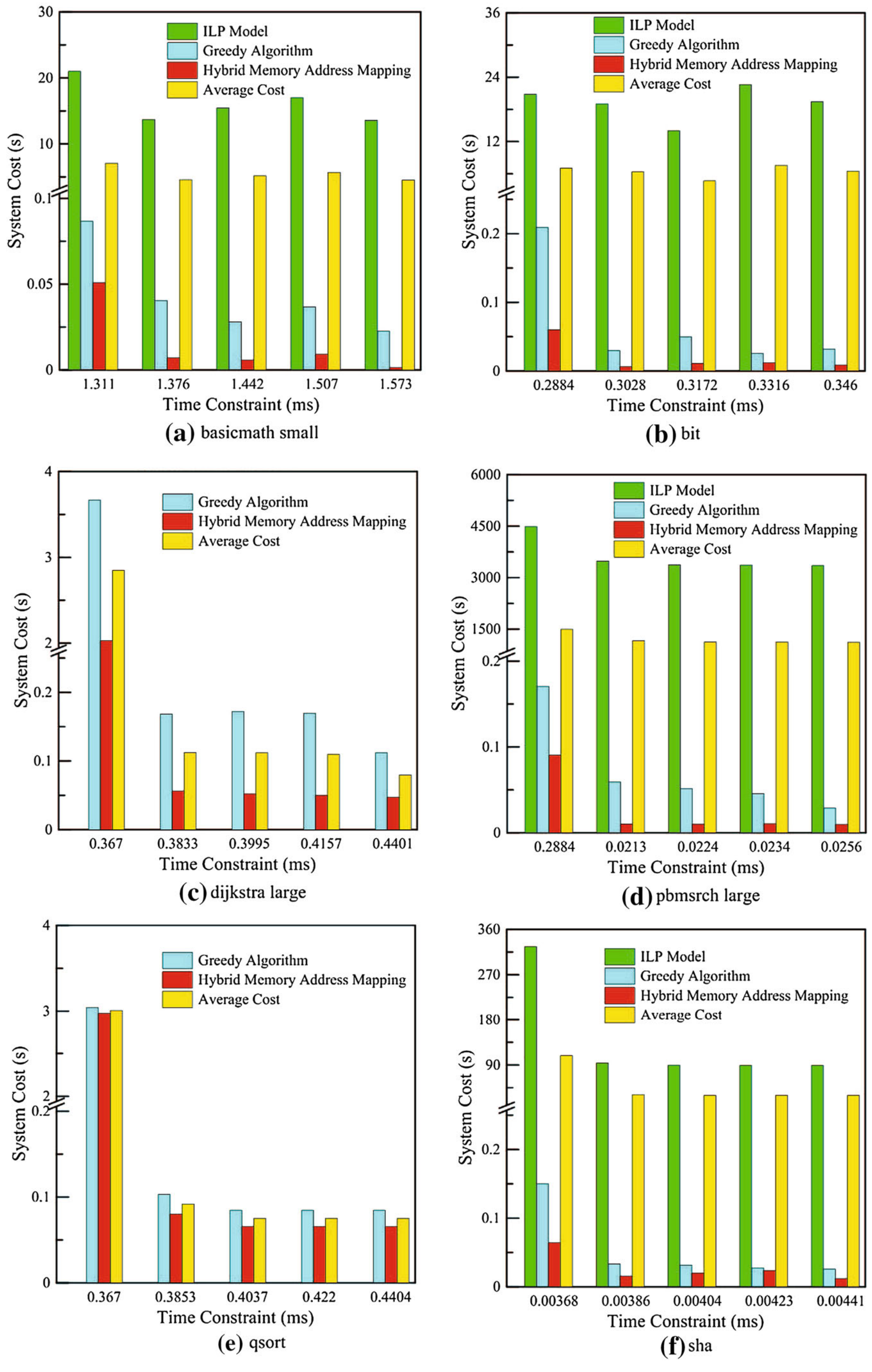


Fig. 8 The system cost for each benchmark under different timing constraints

Table 7 The time and energy consumption for hybrid architecture and cache-based hybrid architecture

Benchmark	Hybrid architecture		Cache-based hybrid architecture		
	Time (ms)	Energy (mJ)	Hit ratio (%)	Time (ms)	Energy (mJ)
basicmath small	1.5727	0.4408	99.9	0.5211	0.0417
bit	0.3460	0.0879	99.9	0.0726	0.0058
dijkstra large	0.4401	0.1308	90.2	0.1339	0.0105
pbmsrch large	0.0256	0.0093	85.4	0.0107	0.00098
qsort	0.4404	0.1810	55.5	0.5153	0.4200
sha	0.00441	0.00012	84.1	0.00201	0.00016

hit miss leads to extra search time in SRAM and costs more energy compared to those in hybrid architecture.

4.2.4 Comparison for pure DRAM, pure PCM and hybrid architecture

This section presents the comparison for pure DRAM, pure PCM and hybrid architecture. The experimental results are shown in Fig. 9. In terms of time cost, pure DRAM achieves the best results, and pure PCM suffers from the longest time. The time cost of the proposed hybrid architecture ranges between pure DRAM and pure PCM, and it can obtain similar performance as that of pure DRAM. Our proposed hybrid architecture can achieve an average reduction of 67 % compared to that of pure PCM. This shows the effectiveness of the proposed scheme.

The experimental results of energy consumption for these three schemes are presented in Fig. 9b. Not surprisingly, the energy consumption of the proposed hybrid architecture costs more than that of pure PCM and less than that of pure DRAM. The proposed hybrid architecture can reduce an average 34 % of energy consumption compared with pure DRAM, and slightly increase energy consumption compared with pure PCM. For some benchmarks, such as *basicmath small*, *dijkstra large* and *pbmsrch large*, the energy consumptions of proposed hybrid architecture and pure PCM can achieve similar energy consumption. Therefore, the proposed hybrid architecture can find the trade-off between pure DRAM and pure PCM.

5 Related work

Several techniques have been proposed to reduce write activities of PCM including differential write [5], compression [11,12], Flip-N-Write [13], and row-buffer locality-aware data placement [14]. Zhou et al. [5] proposed a set of techniques, such as redundant bit removal and row shifting, to prolong the lifetime of PCM-based main memory. Write activity reduction is achieved mainly based on the idea of data-comparison write (DCW), by which a write to PCM is ignored if its designated PCM cell holds the same value. Zhang and Li [15] presented a hybrid PRAM/DRAM memory architecture and exploited an OS-level paging scheme to improve PCM write

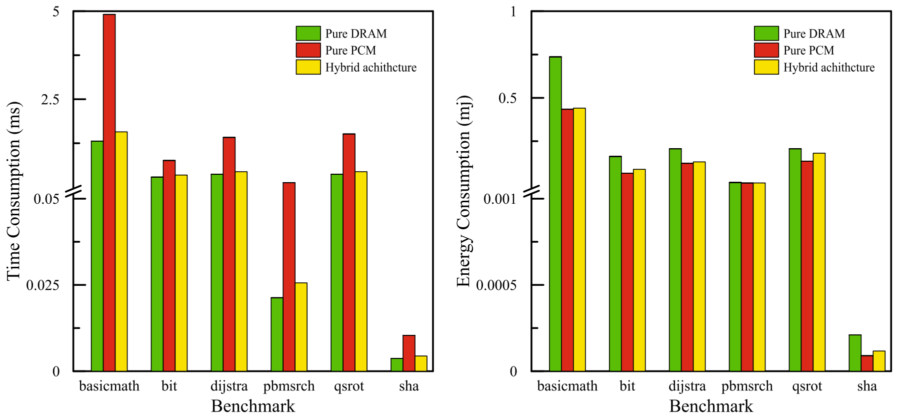


Fig. 9 The comparison of time cost and energy consumption with pure DRAM, pure PCM and hybrid architecture for each benchmark

performance and lifetime. Cho et al. propose a simple architectural technique to replace a PCM write operation with a more efficient read-modify-write operation for reducing redundant bit programming [13]. Joo et al. [16] proposed an energy and endurance aware PCM cache design which reduces write activities by read-before-write and data inverting techniques. Sun et al. [11] proposed a PCM architecture of frequent-value storage based on data locality. Write intensity is reduced by storing frequently written values in compressed form.

Wear leveling techniques that aim to evenly distribute write activities include hot/cold line shifting and segment swapping [5], randomized mapping such as Start-Gap [17] and Security Refresh [18], and adaptive wear leveling with online attack detector [19]. To defend against malicious attacks, randomized approaches have been proposed to randomly distribute writes [17–20]. Wear-leveling-aware encryption techniques have been proposed in [21,22]. Though various wear leveling techniques have been studied for other non-volatile memories such as NAND flash [23,24], they cannot be applied to PCM-based embedded systems directly due to the different characteristics of PCM and flash memory. Hu et al. presented software-based management strategies for hybrid PCM main memory [25–29]. Performance could be improved through wear-leveling, write activity reduction, and energy optimization. Our technique can be combined with these techniques to further improve the performance.

Some software-level techniques are also proposed. Hu et al. [30] proposed write-aware scheduling and recomputation, to schedule the tasks in the program with the consideration of write activities in main memory, for minimizing write activities in non-volatile memories such as PCM. In [31], data migration and code optimization techniques are proposed to avoid write-backs of shared data, for extending the lifetime of PCM-based main memory. ER [32] utilizes data compression to prolong the lifetime of PCM. Ferreira et al. [33] present write minimization, unnecessary writes reduction, and a wear-leveling scheme to increase the lifetime of PCM-based main memory. Software dispatch [34] distributes data to different memories depending on data access characteristics. Besides, In [35], Dong et al. study the endurance variation of PCM cells, and propose a variant of wear leveling mechanism, through physical address re-mapping and data swapping, to balance wear rates of PCM cells across the whole PCM chip. In [36], Bathen et al. present a run-time memory manager, called HaVOC, that virtualizes the hybrid on-chip memory space and supports efficient sharing of distributed ScratchPad Memories (SPMs) and NVMs. ViPZonE [37] is a system level optimization technique for allocating memory in the kernel, however, it focuses on reducing energy consumption in DRAM-based systems.

Compared to previous work, this paper has different strategies and targets, i.e., using the hybrid main memory architecture with both PCM and DRAM to explore their potentials to improve the energy efficiency. Besides, this paper focuses on the address allocation strategy to optimally allocate the memory space. Therefore, different from previous work, this paper takes a first step to propose address mapping schemes for hybrid main memory architecture for optimizing the energy consumption while considering the timing constraint. The approach proposed in this paper is especially important for resource-constrained embedded systems, since both energy consumption and real-time performance are critical issues in such systems.

6 Conclusion

In this paper, we studied the problem of minimizing energy consumption for the PCM and DRAM hybrid main memory architecture. We proposed an optimal address mapping algorithm to map a proper memory type for each address such that the total energy consumption can be minimized while the timing constraint is satisfied. We also presented a heuristic approach to achieve an efficient and near-optimal solution. Experimental results show that the proposed approach can significantly reduce the energy consumption with the least system cost compared with representative techniques. In the future, we plan to investigate the use of our approach to large-scale systems with multiple concurrent tasks. We will also extend our approach to other hybrid main memory infrastructures and propose a general model that can be applied to other system architectures.

Acknowledgments The work described in this paper is partially supported by the grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (GRF 152138/14E), National Natural Science Foundation of China (Project 61272103 and Project 61373049), National 863 Program 2013AA013202, the Hong Kong Polytechnic University (4-ZZD7,G-YK24, G-YM10 and G-YN36), the Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges Under Beijing Municipality (IDHT20150507), Guangdong Province Key Laboratory Project (2012A061400024), Guangdong Natural Science Foundation (2014A030310269), and Natural Science Foundation of SZU (201534).

References

1. Eilertand S, Leinwander M, Crisenza G (2009) Phase change memory: a new memory enables new memory usage models. In: Proceedings of the 2009 IEEE international memory workshop (IMW '09), pp 1–2

2. Qureshi MK, Srinivasan V, Rivers JA (2009) Scalable high performance main memory system using phase-change memory technology. In: Proceedings of the 2009 international symposium on computer architecture (ISCA '09), pp 24–33
3. Dhiman G, Ayoub R, Rosing T (2009) PDRAM: a hybrid pram and dram main memory system. In: Proceedings of the 46th ACM/IEEE design automation conference (DAC '09), pp 664–669
4. Dong X, Xu C, Xie Y, Jouppi N (2012) NVSim: a circuit-level performance, energy, and area model for emerging nonvolatile memory. IEEE Trans Comput Aided Des Integr Circuits Syst 31(7):994–1007
5. Zhou P, Zhao B, Yang J, Zhang Y (2009) A durable and energy efficient main memory using phase change memory technology. In: Proceedings of the 36th annual international symposium on computer architecture (ISCA '09), New York, NY, USA. ACM, pp 14–23
6. Tian W, Zhao Y, Shi L, Li Q, Li J, Xue C et al (2013) Task allocation on nonvolatile-memory-based hybrid main memory. IEEE Trans Very Large Scale Integr (VLSI) Syst 21(7):1271–1284
7. Zhou M, Du Y, Childers B, Melhem R, Mossé D (2012) Writeback-aware partitioning and replacement for last-level caches in phase change main memory systems. ACM Trans Archit Code Optim 8(4):53:1–53:21
8. Yu C, Jie R, Hui Z, Chun SY (2006) Dynamic binary translation and optimization in a whole-system emulator-skyeye. In: Proceedings of the 2006 international conference on parallel processing workshops (ICPP '06), pp 327–336
9. Guthaus M, Ringenberg J, Ernst D, Austin T, Mudge T, Brown R (2001) Mibench: a free, commercially representative embedded benchmark suite. In: Proceedings of the 2001 IEEE international workshop on workload characterization (WWC-4 2001), pp 3–14
10. Powell M, Yang SH, Falsafi B, Roy K, Vijaykumar T (2000) Gated-vdd: a circuit technique to reduce leakage in deep-submicron cache memories. In: Proceedings of the 2000 international symposium on low power electronics and design, 2000. ISLPED '00, pp 90–95
11. Sun G, Niu D, Ouyang J, Xie Y (2011) A frequent-value based pram memory architecture. In: Proceedings of the 16th Asia and South Pacific design automation conference (ASP-DAC '11), pp 211–216
12. Cintra M, Linkewitsch N (2013) Characterizing the impact of process variation on write endurance enhancing techniques for non-volatile memory systems. In: Proceedings of the ACM SIGMETRICS/international conference on measurement and modeling of computer systems (SIGMETRICS '13), New York, NY, USA. ACM, pp 217–228
13. Cho S, Lee H (2009) Flip-N-write: a simple deterministic technique to improve pram write performance, energy and endurance. In: Proceedings of the 42nd annual IEEE/ACM international symposium on microarchitecture (MICRO-42 '09), pp 347–357
14. Yoon H, Meza J, Ausavarungnirun R, Harding R, Mutlu O (2011) Row buffer locality-aware data placement in hybrid memories. Technical Report. CMU
15. Zhang W, Li T (2009) Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures. In: Proceedings of the 18th international conference on parallel architectures and compilation techniques (PACT '09), pp 101–112
16. Joo Y, Niu D, Dong X, Sun G, Chang N, Xie Y (2010) Energy and endurance-aware design of phase change memory caches. In: Proceedings of the 2010 design, automation test in Europe conference exhibition (DATE '10), pp 136–141
17. Qureshi M, Karidis J, Franceschini M, Srinivasan V, Lastras L, Abali B (2009) Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In: Proceedings of the 42nd annual IEEE/ACM international symposium on microarchitecture (MICRO-42 '09), pp 14–23
18. Seong NH, Woo DH, Lee HHS (2010) Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In: Proceedings of the 37th annual international symposium on computer architecture (ISCA '10), New York, NY, USA. ACM, pp 383–394
19. Qureshi M, Seznec A, Lastras L, Franceschini M (2011) Practical and secure PCM systems by online detection of malicious write streams. In: Proceedings of IEEE 17th international symposium on high performance computer architecture (HPCA '11), pp 478–489
20. Seznec A (2010) A phase change memory as a secure main memory. Comput Archit Lett 9(1):5–8
21. Kong J, Zhou H (2010) Improving privacy and lifetime of PCM-based main memory. In: Proceedings of 2010 IEEE/IFIP international conference on dependable systems and networks (DSN '10), pp 333–342
22. Chhabra S, Solihin Y (2011) i-NVMM: a secure non-volatile main memory system with incremental encryption. In: Proceedings of the 38th annual international symposium on computer architecture (ISCA '11), New York, NY, USA. ACM, pp 177–188

23. Chang YH, Hsieh JW, Kuo TW (2007) Endurance enhancement of flash-memory storage systems: an efficient static wear leveling design. In: Proceedings of the 44th annual design automation conference (DAC '07), New York, NY, USA. ACM, pp 212–217
24. Chang LP, Kuo TW (2002) An adaptive striping architecture for flash memory storage systems of embedded systems. In: Proceedings of the 8th IEEE real-time and embedded technology and applications symposium (RTAS '02), pp 187–196
25. Hu J, Zhuge Q, Xue CJ, Tseng WC, Sha EHM (2013) Software enabled wear-leveling for hybrid PCM main memory on embedded systems. In: Design, automation test in Europe conference exhibition (DATE), 2013, pp. 599–602
26. Hu J, Xie M, Pan C, Xue C, Zhuge Q, Sha E (2014) Low overhead software wear leveling for hybrid PCM + DRAM main memory on embedded systems. IEEE Trans Very Large Scale Integr (VLSI) Syst PP(99):1–6
27. Hu J, Xue CJ, Zhuge Q, Tseng WC, Sha EHM (2013) Write activity reduction on non-volatile main memories for embedded chip multiprocessors. ACM Trans Embed Comput Syst 12(3):77:1–77:27
28. Hu J, Xue C, Zhuge Q, Tseng WC, Sha EM (2011) Towards energy efficient hybrid on-chip scratch pad memory with non-volatile memory. In: Design, automation test in Europe conference exhibition (DATE), 2011, pp 1–6
29. Hu J, Xue C, Zhuge Q, Tseng WC, Sha E (2013) Data allocation optimization for hybrid scratch pad memory with SRAM and nonvolatile memory. IEEE Trans Very Large Scale Integr (VLSI) Syst 21(6):1094–1102
30. Hu J, Tseng WC, Xue C, Zhuge Q, Zhao Y, Sha EM (2011) Write activity minimization for nonvolatile main memory via scheduling and recomputation. IEEE Trans Comput Aided Des Integr Circuits Syst 30(4):584–592
31. Hu J, Xue C, Tseng WC, He Y, Qiu M, Sha EM (2010) Reducing write activities on non-volatile memories in embedded CMPs via data migration and recomputation. In: Proceedings of the 47th ACM/EDAC/IEEE design automation conference (DAC '10), pp 350–355
32. Jiang L, Zhang Y, Yang J (2012) ER: elastic RESET for low power and long endurance MLC based phase change memory. In: Proceedings of the 2012 ACM/IEEE international symposium on low power electronics and design (ISLPED '12), New York, NY, USA. ACM, pp 39–44
33. Ferreira A, Zhou M, Bock S, Childers B, Melhem R, Mosse D (2010) Increasing PCM main memory lifetime. In: Proceedings of the 2010 design, automation test in Europe conference exhibition (DATE '10), pp 914–919
34. Li Y, Chen Y, Jones AK (2012) A software approach for combating asymmetries of non-volatile memories. In: Proceedings of the 2012 ACM/IEEE international symposium on low power electronics and design (ISLPED '12), New York, NY, USA. ACM, pp 191–196
35. Dong J, Zhang L, Han Y, Wang Y, Li X (2011) Wear rate leveling: lifetime enhancement of PRAM with endurance variation. In: Proceedings of the 48th ACM/EDAC/IEEE design automation conference (DAC '11), pp 972–977
36. Bathen L, Dutt N (2012) HaVOC: a hybrid memory-aware virtualization layer for on-chip distributed scratchpad and non-volatile memories. In: Proceedings of the 49th ACM/EDAC/IEEE design automation conference (DAC '12), pp 447–452
37. Bathen LAD, Gottscho M, Dutt N, Nicolau A, Gupta P (2012) ViPZonE: OS-level memory variability-driven physical address zoning for energy savings. In: Proceedings of the eighth IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis (CODES+ISSS '12), New York, NY, USA. ACM, pp 33–42