ISBN 978-981-14-4787-7
Proceedings of 2020 the 10th International Workshop on Computer Science and Engineering (WCSE 2020)
Shanghai, 19-21 June, 2020, pp. 288-294
doi: 10.18178/wcse.2020.06.043

Real-time Task Allocation Based on Variable Scratchpad Memory

Jixiao Li and Weibi Bai [+]

School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, 400065, China

Abstract. Traditional memories, such as DRAM and SRAM, are widely used in embedded systems. With the increasing integration of CMOS, the static energy consumption of traditional memories is becoming more and more severe. Scratchpad memory belongs to on-chip memory, which acts on the cache layer, and it can be controlled by software. Random-access memory is usually used as the on-chip memory of the cache layer, but it has the problems of poor scalability, small storage density, and high static energy consumption, which seriously affects the system performance. Non-volatile memory has the advantages of reading speed, ultra-low static energy consumption, and high density compared with traditional memory. It is regarded as a new type of memory that is expected to replace traditional memory. In order to solve the problems of traditional memory, this article will combine the two types of storage units of NVM SLC/MLC, which can be converted to each other and the complementary advantages of the two storage units in performance and storage capacity, to achieve a new architecture of variable SPM, and design the corresponding task allocation algorithm to achieve the task set to meet the real-time schedulable requirements, while achieving the goal of the lowest energy consumption of the system.

Keywords: Non-volatile memory, Scratchpad memory, task allocation.

1. Introduction

With the development of data-intensive application technologies such as cloud computing, the Internet of Things, big data, and edge computing, embedded devices represented by smart terminal devices such as mobile phones have increasingly higher storage capacity and performance requirements. However, the traditional memory technology represented by dynamic random access memory (DRAM) and static random access memory (SRAM) has been challenging to achieve a huge breakthrough. Traditional scratchpad memory is mainly composed of SRAM, address decoding components and data output circuits, and software programs can control SPM. The main advantages of SRAM are its extremely high read-write performance and long life of 10^8. So it is usually used to replace the traditional cache [1]. However, SRAM also faces the problems of insufficient storage density and excessive static leakage power consumption. The average SRAM leakage is 25% to 50% of the total energy consumption, which will also seriously affect the performance of the system [2,3]. However, the emergence of new non-volatile memory brings hope to solve the problem of traditional memory. NVM has many advantages, such as low static energy consumption, high integration density, excellent shock resistance, fast reading speed, and non-volatile. In this paper, a highly variable SPM is constructed by combining the ultra-high access performance of SLC with the high storage density of MLC and their mutual conversion characteristics and proposes a related task allocation algorithm to make the system energy consumption overhead reaches the lowest while the task set meets real-time schedulability.

[+]Corresponding author. Tel.: + 86 18523502154.
E-mail address: 532994775@qq.com.

2. Related Work

2.1. Architectural Model

As shown in Figure 1, compared with traditional embedded systems, the main difference in this article is the use of on-chip SPM in the cache layer. Besides, the on-chip SPM is composed of a variable NVM, which can dynamically switch between SLC mode and MLC mode. Moreover, the embedded system in this article also uses DRAM as the main memory to store the operating data during the program running process.

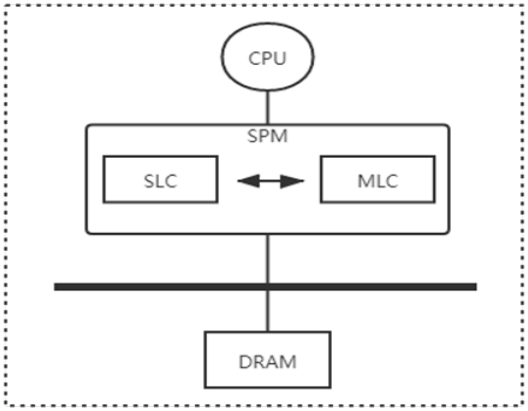

Fig 1: Architectural model

2.2. Related Studies

In this part, we will describe the static scheduling and data allocation research and dynamic task scheduling and data allocation research.

In static tasks and data allocation, Wang Z et al.[4] proposed a static data allocation algorithm based on WCET awareness, the purpose of which is to ensure that the upper limit of WCET, while allocating data variables to SPM to make the system Has the lowest energy cost. Gu et al.[5] studied the static task allocation and data allocation of multi-core systems based on multi-port SPM to minimize energy consumption and expressed the problem as an integer linear programming and combined with the algorithm to find the optimal solution to achieve the optimal task allocation and data allocation.

In dynamic tasks and data allocation, Wang Y et al.[6], proposed a dynamic greedy data allocation and task scheduling algorithm (DGDATS) that can sense the Changes, thereby dynamically planning data allocation and task scheduling strategies, and significantly reducing the energy consumption overhead of the system during operation based on a multi-core SPM system. Soliman et al. [7], introduce a compiler-controlled prefetch scheme for SPM in this paper, which is dynamic data allocation based on WCET. This method can make data transmission and tasks execute concurrently, to reduce the waiting time of the memory, reduce the response time of the system, and improve the performance of the system.

This article uses a single-core SPM system. We have built a dynamically variable SPM by combining the characteristics of SLC/MLC that can be converted to each other, the advantages of SLC ultra-high access performance, and the advantages of MLC high storage capacity. Then, we establish a mathematical model according to the access characteristics of the task, and the real-time constraints of the task is considered. Finally, we will design the corresponding task allocation algorithm to achieve the lowest system energy consumption cost while the real-time requirements are met.

3. Algorithms

In this section, we will discuss the details of the energy-aware real-time task allocation algorithm (EARTA). Before introducing the EARTA algorithm, we firstly give the computation model. Then, based on the computation model, a task allocation algorithm is proposed.

3.1. Computation Model

In this section, we will introduce the computation model of this article. To determine the optimal SLC/MLC SPM configuration and task assignment, we establish the problem of task assignment. First, we simulate the sizes of SLC SPM and MLC SPM in a variable SPM. Then, we propose a real-time constraint. Besides, we also simulate the task allocation among SLC SPM, MLC SPM, and DRAM. And finally, the objective function is given.

In the task allocation scheme, the configuration of the SLC/MLC in the variable SPM can be dynamically switched with multiple task sets with different characteristics. Therefore, the number of SLC/MLC configurations varies according to the characteristics of different task sets. However, there is no difference in physical structure between SLC and MLC, so the total number of SLC and MLC units in SPM must be equal to the total number of SPM units. This constraint model is as equation (1):

$$Size_{SLC} + \frac{Size_{MLC}}{b_m} = N_{spm} \tag{1}$$

In this equation, $Size_{SLC}$ indicates the number of SLC units under the current SPM configuration, $Size_{MLC}$ indicates the number of MLC units under the current SPM configuration andb_m indicates the size of an MLC unit.

In this article, each task in the task set must be stored in SLC SPM or MLC SPM or DRAM, so we set a variable A, which can represent the task allocation between SLC SPM, MLC SPM, and DRAM. $A_{i,k} = 1$It can be determined whether the task is allocated to SPM or main memory according to the value of letter k. If k = 1, the i-th task is in DRAM, if k = 2, the i-th task is in SLC SPM, if k = 3, the i-th task is in MLC SPM. The formula is as equation (2):

$$\forall_i \sum_{k=1}^{3} A_{i,k} = 1 \tag{2}$$

The energy consumption model of the algorithm in this paper is mainly composed of three types of energy consumption, which are the energy consumption of task access, the energy consumption of SLC/MLC conversion, and the energy consumption of DRAM static leakage. First, the task's access to energy consumption consists of two types of energy consumption. The first is when the task is stored in SPM, which can be expressed as equation (3):

$$E_{spm} = \sum_{k=2}^{3} \sum_{i=1}^{n} (A_{i,k} \times (e_{r_k} \times N_{r_i} + e_{w_k} \times N_{w_i})) \tag{3}$$

In this formula, e_{r_k} is the read operation power consumption in memory k, k = 1 is in DRAM, k = 2 is in SLC SPM, and k = 3 is in MLC SPM. N_{r_i} represents the number of read operations for the i-th task. e_{w_k} is the write operation power consumption in memory k, k = 1 is in DRAM, k = 2 is in SLC SPM, and k = 3 is in MLC SPM. N_{w_i} represents the number of write operations for the i-th task.

The second is the access energy consumption when the task is stored in DRAM. In this case, the static leakage energy consumption of DRAM needs to be considered, p_{dram} represents static leakage power, so the access energy consumption model can be expressed as equation (4):

$$E_{dram} = \forall_{k=1} \sum_{i=1}^{n} (A_{i,k} \times \left((e_{r_k} \times N_{r_i} + e_{w_k} \times N_{w_i}) + (t_{r_k} \times N_{r_i} + t_{w_k} \times N_{w_i}) \times p_{dram} \right)) \tag{4}$$

The total access energy consumption of a task is the sum of the access energy consumption of a task stored in SPM and the access energy consumption of a task stored in DRAM. Therefore, it can be expressed as equation (5):

$$E_a = E_{spm} + E_{dram} \tag{5}$$

Because SPM can dynamically change its capacity with the different needs of each task set, the energy consumption of SLC to MLC conversion can be considered as the energy consumption caused by the change in the number of SLC units required by two adjacent task sets. Therefore, it can be expressed as equation (6):

$$E_c = \left| SS_{taskset_j} - SS_{taskset_{j-1}} \right| \times e_c \tag{6}$$

The scheduling strategy in this article uses the EDF scheduling strategy. The EDF algorithm is a dynamic priority scheduling algorithm. When a scheduling event occurs, the EDF algorithm always chooses the task to execute, which has the latest deadline. For the EDF algorithm, according to the literature [8], the sufficient and necessary condition for schedulable task sets is that the total CPU utilization does not exceed 100%, because the task period P_i can be obtained from the input parameters, and the access time of the task C_i can be obtained according to the read and write characteristics of the task, so the real-time constraint can be described as the formula (7):

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1 \tag{7}$$

Finally, the purpose of this article is to minimize energy consumption on the premise that the task set is real-time schedulable, so the objective function is designed to minimize energy consumption and can be expressed as the equation (8):

$$optimal_{result} = \sum_{j=1} \min(E_a + E_c) \tag{8}$$

3.2. Energy-aware Real-time Task Allocation Algorithm

In this section, we will introduce the EARTA algorithm in detail to obtain an approximately optimal result. The EARTA algorithm consists of three parts. First, we construct a 3-D matrix through the idea of dynamic programming 01 backpack. The matrix records the lowest energy consumption under each SPM of each SLC and MLC number. Then, trace the source of each SLC/MLC SPM configuration, find the task with the lowest energy consumption under each SLC/MLC SPM configuration and record it in an array. Finally, the real-time judgment is performed on each case recorded in the candidate scheme set, and finally, a target scheme is obtained, that is, a task allocation scheme that meets the real-time requirement and has the lowest energy consumption. The EARTA algorithm is shown in Figure 2.

Algorithm 1 EARTA

Input: Task set: $T = \{T_1, T_2, T_3 ... T_n\}$; Each task's size and period: S_i, P_i; SPM initial size: N_{spm}; Each Task's energy in each storage media: E_{DRAM_i}, E_{SLC_i}, E_{MLC_i}.

Output: Optimal task assignment: R(T); Energy of task set: $Energy_{optimal}$; The entire task set execution time: $Time_{optimal}$.

```
Initialize Energy_optimal ← ∞;
Initialize Time_optimal ← 0;
Initialize the optimal task assignment R(T) ← NIL;
Initialize dp matrix V[n, N_spm, 2 × N_spm]
for i ← 1 to n do
    for SS ← 0 to N_spm do
        for SM ← 0 to 2 × N_spm do
            if SS = 0 and SM = 0 then
                V[i, SS, SM] = V[i − 1, SS, SM] + E_DRAM_i
            else if S_i <= SS then
                V[i, SS, SM] = min(V[i−1, SS, SM]+E_DRAM_i, V[i−1, SS−S_i, SM]+E_SLC_i, V[i−1, SS, SM−S_i] + E_MLC_i)
            else if S_i <= SM then
                V[i, SS, SM] = min(V[i − 1, SS, SM] + E_DRAM_i, V[i − 1, SS, SM − S_i] + E_MLC_i)
            else
                V[i, SS, SM] = V[i − 1, SS, SM] + E_DRAM_i
            end if
            Record[i, SS, SM] ⇐ ("Assignment of task i (DRAM/SLC/MLC)").
        end for
    end for
end for
Invoking matrix backtracking algorithm to get pre-task-assignment PreR[N_spm].
for j ← 0 to N_spm do
    Compute n tasks' real-time constraints, ∑_{i=1}^{n} C_i/P_i
    if ∑_{i=1}^{n} C_i/P_i <= 1 then
        if PreR[j].energy < Energy_optimal then
            Update Energy_optimal to PreR[j].energy.
            Update Time_optimal to ∑_{i=1}^{n} C_i.
            Update R(T) to PreR[j].
        end if
    end if
end for
```

Fig 2 : The EARTA algorithm

As shown in Figure 2. First, initialize the result variables. After the variables are initialized, the algorithm uses the idea of dynamic programming to construct the energy matrix $V[n, N_{spm}, 2 * N_{spm}]$. According to the proposed ILP model, tasks have different energy consumption costs when stored in different storage. When the number of SLC and MLC is 0, the i-th task can only be stored in DRAM, and the lowest energy consumption at this time is $V\ [i-1, SS, SM]\ +\ E_{DRAM_i}$. If the size of the task is Less than the size of the current SLC, the minimum energy consumption at this time is $min\ (V\ [i-1, SS, SM]\ +\ E_{DRAM_i}, V\ [i-1, SS-S_i, SM]\ +\ E_{SLC_i}, V\ [i-1, SS\ , SM-S_i]\ +\ E_{MLC_i})$, if the size of the task is larger than the size of the current remaining SLC and smaller than the current remaining The size of MLC, then the minimum energy consumption at this time is $min\ (V\ [i-1, SS, SM]\ +\ E_{DRAM_i}, V\ [i-1, SS, SM-S_i]\ +\ E_{MLC_i})$, after obtaining the lowest energy consumption, use a 3-D $Record$ array as the mapping array to record the task allocation scheme under the lowest energy consumption. After the looping, the energy consumption matrix $V[n, N_{spm}, 2 * N_{spm}]$ And the task assignment matrix $\mathrm{Record}[n, N_{spm}, 2 * N_{spm}]$ is completed.

Then, the EARTA algorithm will call the matrix backtracking MB algorithm, taking $\mathrm{Record}[n, N_{spm}, 2 * N_{spm}]$ as input, and finally obtaining an alternative array $PreR[N_{spm}]$, which records the lowest energy consumption in various SLC/MLC configurations. And the elements of this array will be a candidate for the optimal task allocation scheme. We can determine whether the elements in the array meet the real-time requirements by formula (7), after traversing $PreR[N_{spm}]$ and then update the result variables. Finally, the task allocation scheme R(T) that meets the real-time performance and the lowest energy consumption, and the optimal energy consumption and time variables are output.

3.3. Matrix Backtracking Algorithm

This section mainly introduces the MB algorithm. The main idea of the MB algorithm is to trace the optimal allocation scheme under each SLC/MLC configuration according to the mapping matrix $\mathrm{Record}[n, N_{spm}, 2 * N_{spm}]$ and the constraint equation (1), and then store the traced result in the array $PreR[N_{spm}]$. At this point, the array $PreR[N_{spm}]$ is the candidate scheme array. The MB algorithm is shown as Figure 3.

Algorithm 2 MB

Input: Record matrix: Record[n,N_{spm},2×N_{spm}]; Task set: $T = \{T_1, T_2, T_3 ... T_n\}$; Each task's size: S_i; SPM initial size: N_{spm}.
Output: Pre-task-assignment: PreR[N_{spm}].

1: *Initialize* $PreR[N_{spm}] \leftarrow \varnothing$;
2: **for** $j \leftarrow 0$ *to* N_{spm} **do**
3: $SS \leftarrow j$
4: $SM \leftarrow 2 \times (N_{spm} - SS)$
5: $i \leftarrow n$
6: **while** $i! = 0$ **do**
7: *Record the assignment of the nth task from* $Record[i, SS, SM]$.
8: **if** *the nth task is assigned to SLC* **then**
9: $SS \leftarrow SS - S_i$
10: **end if**
11: **if** *the nth task is assigned to MLC* **then**
12: $SM \leftarrow SM - S_i$
13: **end if**
14: $i{-}{-}$
15: **end while**
16: *Record the entire task set in this SLC/MLC configuration to* $PreR[j]$.
17: **end for**

Fig 3: The MB algorithm

The source tracing process of the MB algorithm is mainly as follows. First, we can know the number of MLCs when the SLC is a specific value through the constraint equation (1). Then according to the matrix $\mathrm{Record}[n, N_{spm}, 2 * N_{spm}]$, the task with the lowest energy consumption under the corresponding number of SLC and MLC can be obtained. If the task is allocated to the SLC, the remaining space of the SLC is $SS - S_i$, and the MLC number will also change. If the task is allocated to the MLC, the remaining space of the MLC

is $SM - S_i$. At this time, the number of SLC will also change as the MLC changes. If the task is allocated to the DRAM, the number of MLC and SLC does not change. When the task loop traversal ends, an optimal task allocation scheme in the SLC/MLC configuration will be obtained and recorded in the array $PreR[N_{spm}]$. Finally, after traversing the number of all units in the SPM, a complete candidate array $PreR[N_{spm}]$ can be obtained and output.

4. Experiments

The experiments in this paper use two simulators, Gem5 and NVmain. Gem5 is a modular discrete event-driven full system simulator. NVmain [9] is a memory simulator developed for non-volatile memory, and it can also simulate traditional memories such as DRAM. We simulate the embedded system with SPM by jointly compiling Gem5 and NVmain, and finally combine the approximate algorithm to simulate task allocation. We use Mibench as benchmarks. The combination of tasks is shown in Table 1. We compare the initial size of 2MB SPM with pure DRAM architecture, 2MB pure SLC SPM, 4MB pure MLC SPM and Offline-EDF algorithm [10] to get comparison data. The comparison results are shown in Figure 4.

Table 1: Bencmarks

Task set	tasks
T1	qsort_small, bitcount, jpeg, dijkstra, blowfish, FFT
T2	basicmath_small, susan, typeset, patricia, sha, stringsearch
T3	sha, qsort_small, bitcount, susan, jpeg, stringsearch
T4	typeset, dijkstra, patricia, blowfish, sha, FFT
T5	dijkstra, FFT, qsort_small, sha, bitcount, blowfish
T6	jpeg, dijkstra, FFT, susan, qsort_small, bitcount

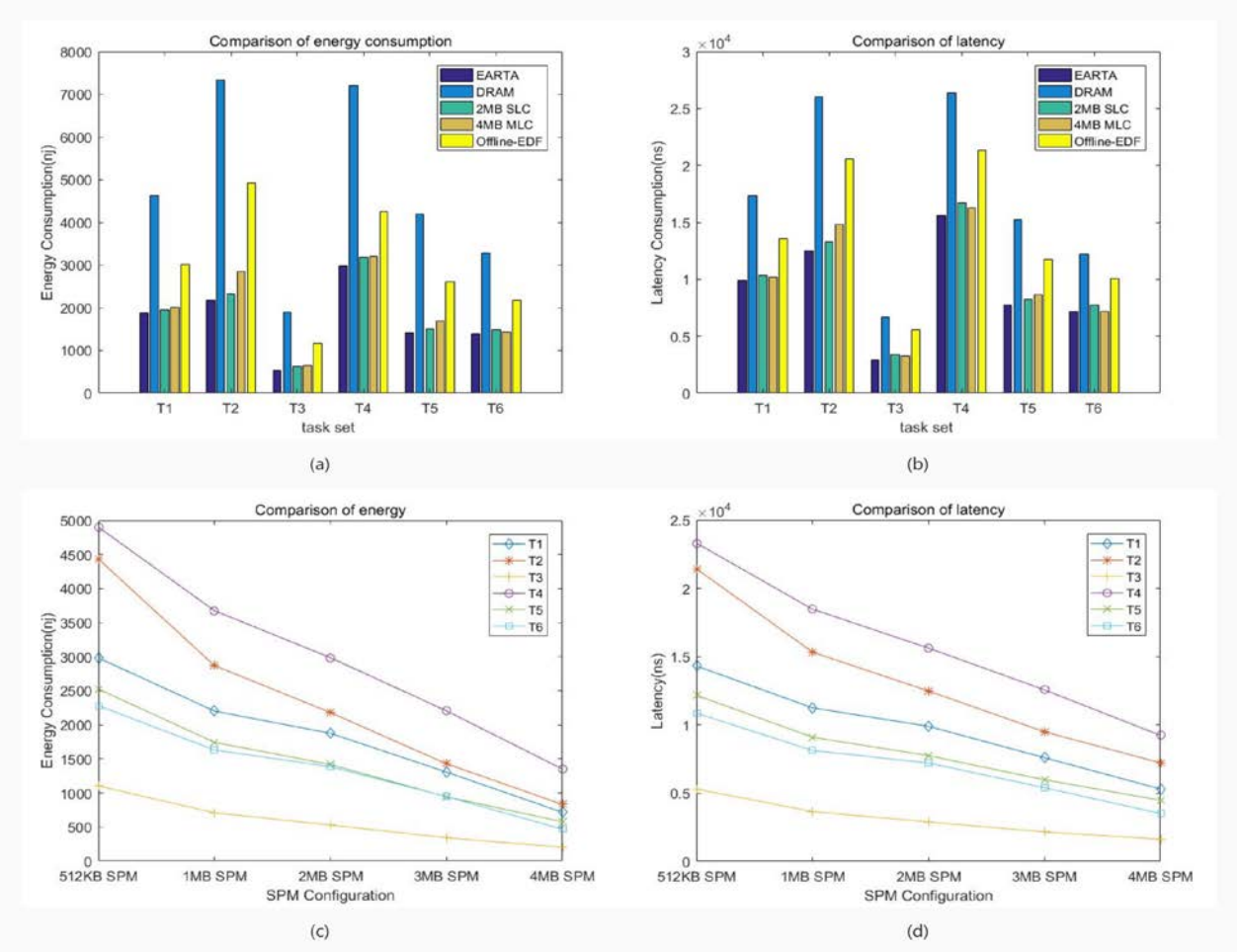

Fig 4 : Experimental comparison chart

From (a) and (b) in Figure 4, it can be seen that the algorithm in this paper has obvious effects on the reduction of system energy and performance. Compared with the DRAM-only architecture, it reduced the energy consumption by 64.1% and reduced the latency by 47.1%. Compared with the 3MB SLC SPM architecture, it reduced the energy consumption by 7.3% and reduced the latency by 7.4%. Compared with the 6MB MLC SPM architecture, the average energy consumption is reduced by 12.1%, and the latency is reduced by 7.5%. Compared with Offline-EDF algorithm [10], the average energy consumption is reduced by 43.3%, and the latency is reduced by 33.8%. It can be seen from Figures (c) and (d) that when the SPM is larger, the SPM hit rate is higher, the effect of our algorithm is more obvious, and the energy consumption and delay are lower.

5. Conclusion

This paper designs the energy consumption and delay model according to the read and write characteristics of the task, and constructs the objective function, which aims to minimize the system energy consumption under the premise that the task set meets the real-time schedule. Then, the energy-aware real-time task allocation algorithm (EARTA) based on variable SPM is proposed in this paper. Subsequently, the candidate task assignment scheme is obtained by a mapping matrix combined with the matrix backtracking algorithm (MB), and then the optimal scheme is found based on the real-time constraint. Finally, a comparative experiment is performed through a simulator. The experimental results verify that the algorithm in this paper can bring considerable energy consumption reduction to the system.

6. References

[1] Yooseong Kim, David Broman, Aviral Shrivastava. WCET-Aware Function-Level Dynamic Code Management on Scratchpad Memory. ACM Trans. Embedded Comput. Syst. 16(4): 112:1-112:26 (2017).

[2] R. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, and P. Marwedel. Scratchpad memory: Design alternative for cache on-chip memory in embedded systems. 10th Int. Symp. Hardw./Softw. Codesign,2002, pp. 73–78.

[3] Y. Wang, K. Li, J. Zhang, and K. Li. Energy optimization for data allocation with hybrid SRAMCNVM SPM. IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 65, no. 1, pp. 307–318, Jan. 2018.

[4] Z. Wang, Z. Gu, and Z. Shao. WCET-aware energy-efficient data allocation on scratchpad memory for real-time embedded systems. IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 11,pp. 2700–2704, Nov. 2015.

[5] S. Gu, Q. Zhuge, J. Yi, J. Hu, and E. H.-M. Sha. Optimizing task and data assignment on multi-core systems with multi-port SPMs. IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 9, pp. 2549–2560, Sep. 2014.

[6] Yan Wang, Kenli Li, Keqin Li. Dynamic Data Allocation and Task Scheduling on Multiprocessor Systems With NVM-Based SPM. IEEE Access 7: 1548-1559 (2019).

[7] M. R. Soliman and R. Pellizzoni. WCET-driven dynamic data scratchpad management with compiler-directed prefetching, in Proc. LIPIcs-Leibniz Int. Inform., vol. 76, 2017, pp. 24:1–24:23.

[8] C. L. Liu, James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. J. ACM 20(1): 46-61 (1973).

[9] Matthew Poremba, Tao Zhang, Yuan Xie. NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems. Computer Architecture Letters 14(2): 140-143 (2015).

[10] Zhang Z, Jia Z, Liu P, et al. Energy efficient real-time task scheduling for embedded systems with hybrid main memory[J]. Journal of Signal Processing Systems, 2016, 84(1): 69-89.