This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

Expected Energy Optimization for Real-Time Multiprocessor SoCs Running Periodic Tasks with Uncertain Execution Time

Kai Huang, Ke Wang, Dandan Zheng, Xiaowen Jiang, Xiaomeng Zhang, Rongjie Yan and Xiaolang Yan

***Abstract*—Energy optimization plays an increasingly critical role in designing an embedded real-time multiprocessor System on Chip (MPSoC). Dynamic Voltage Frequency Scaling (DVFS) and Dynamic Power Management (DPM) are preferable techniques to optimize energy consumption. However, previous DVFS and DPM algorithms were mostly designed for inter-task scheduling, without sufficient exploration on intra-task scheduling for further energy reduction. This paper presents a new intra-task scheduling approach considering the probabilistic distribution of task execution time, and it optimizes the mathematical expectation of power consumption (expected power consumption) for periodic dependent tasks with uncertain execution time running on MPSoCs using DVFS and DPM. The energy-efficient scheduling problem can be formulated by means of mixed integer linear programming (MILP) with the proposed technique. Moreover, we also propose a technique to compress the exploration space by reorganizing the probabilistic profiling information of all tasks. Our experimental results on synthetic and realistic benchmarks show that the proposed approach achieves up to 30% energy savings compared with other existing methods.**

***Index Terms*—Dynamic Voltage Frequency Scaling (DVFS), Dynamic Power Management (DPM), Probability, Intra-task Scheduling, Expected Energy Optimization**

I. Introduction

The increasing complexity and performance requirements of electronic systems have led to the widely use of multiprocessor architecture. High power consumption of modern multiprocessors becomes a major concern for system designers. On one hand, high power consumption decreases the lifetime for battery-driven mobile systems. On the other hand, it degrades the system reliability for overheating the entire chip and thus requiring expensive packaging and cooling technology. Therefore, optimizing power consumption for real-time applications is essential. To achieve this, some multiprocessors, such as ARM Cortex-A57 MPCore [19] and Intel XScale processors [20], provide multiple discrete voltage levels.

Power consumption of CMOS integrated circuits mainly consists of dynamic power from charging and discharging of capacitances and static power from leakage current. With the development of the semiconductor process, static power and dynamic power are of the same magnitude. Therefore, optimizing static and dynamic power are equally important.

Dynamic Voltage Frequency Scaling (DVFS) and Dynamic Power Management (DPM) are two major techniques for dynamic and static power optimization, respectively. DVFS takes advantage of the convex relationship between the frequency and the power consumption of a processor. If we admit that the frequency of a processor is proportional to its voltage, DVFS can achieve quadratic dynamic energy savings with only linear decrease of its frequency. Thus DVFS tends to use the lowest level of voltage/frequency when executing a task. As DVFS cannot save static power, previous studies [9], [10] using DVFS usually ignore static power or assume that static power is a constant value which exists all the time. DPM switches the processors to low power state when they are idle to save static power. However, it takes nonnegligible time and energy to switch to low power state. As a result, it is more energy-efficient to switch the processors to low power state only when the idle time is longer than the break-even time.

DVFS tends to use low voltage/frequency to save dynamic power which prolongs the execution time and shortens the idle time, thus the chance for DPM switching to low power state to save static power is reduced. Therefore, DVFS and DPM counteract with each other, and a trade-off should be achieved when using them both for energy optimization.

In a hard real-time system, each task has a deadline that cannot be violated. However, the execution time of some tasks is uncertain, which is mainly affected by hardware micro-architecture and software input stimulus. First, the execution time of a memory-accessing instruction is usually uncertain. Meanwhile the miss rate of the shared last level cache may vary under different circumstances. Second, different software input stimulus may result in different task execution time. Take the JBIG image encoding task as an example, the execution time would be much less if two adjacent lines of the image are exactly the same, for the second same line does not have to be encoded again. When tasks have uncertain execution time, the scheduler must guarantee that their worst case execution time (WCET) meet their deadlines. If the processor's voltage/frequency can be dynamically scheduled, the tasks' worst case execution cycles (WCECs) are utilized instead of WCET, because execution time equals to execution cycles divided by the processor's frequency.

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

The conservative method to schedule tasks with uncertain execution time merely guarantees that the WCECs meet their deadlines. However, the actual or average number of execution cycles of some tasks are far less than their WCECs, and this will result in a sub-optimal pessimistic scheduling. With the given probabilistic distribution of the tasks, the scheduler has the chance to generate a more energy-efficient scheduling. Through online or offline profiling, the probabilistic distribution of tasks' execution time can be obtained [21], [22]. For example, there is a task that has been executed for 1000 times. And among the 1000 times, the task consumes 1 million cycles for 900 times, 2 million cycles for 99 times, 3 million cycles for only once. According to the above profiling results, the first one million cycles are consumed for 1000 times, the second one million cycles are consumed for 100 times, and the last one million cycles are consumed for only once. In summary, the calculated probabilities of the first, second and last million cycles are 100%, 10% and 0.1%, respectively.

In this paper, we propose a novel energy optimization approach which combines intra-task scheduling, inter-task scheduling, DVFS and DPM for real-time MPSoCs. We assume that each processor has several discrete voltage/frequency levels, and can use DVFS and DPM independently. For a set of real-time applications and a mapping between tasks and processors, our approach can generate an optimal voltage/frequency assignment for each part of all tasks and a time-triggered scheduling, which minimizes the mathematical expectation of energy consumption of the real-time MPSoCs.

There are four main contributions in this paper:

- Based on the state-of-the-art technique that integrates DVFS and DPM to optimize energy consumption of real-time MPSoCs, we use the probabilistic distribution of the tasks' execution cycles for intra-task scheduling, which significantly decreases the mathematical expectation of energy consumption of the MPSoC.
- We develop a formulation that considers both the probabilistic distribution of the tasks and the voltage transition overheads for intra-task scheduling, and we solve it by mixed integer linear programming (MILP).
- We propose an approach to compress the exploration space by reorganizing the probabilistic profiling information of all tasks.
- Our experimental results on synthetic and realistic benchmarks demonstrate that this new approach can significantly decrease the mathematical expectation of energy consumption for real-time MPSoC, compared with other existing studies.

The rest of this paper is organized as follows: Section II reviews the related work. Hardware model, task model and problem definition are described in Section III. Section IV presents a motivating example. Section V formulates the expected energy optimization problem. Experimental results are presented in Section VI. Section VII concludes the paper.

II. Related Work

DVFS has been used for more than a decade [1]. With the evolvement of semiconductor process, the gap between static and dynamic power is narrowing. Therefore, combined DVFS+DPM approach is a preferable technique for low power systems. For uniprocessors, Devadas and Aydin [2] employed DVFS on uniprocessor and DPM on peripheral devices. Based on [2], Gerards and Kuper [3] minimized the energy consumption for independent frame-based tasks. However, these approaches cannot be used in MPSoC platforms. For MPSoCs with dependent tasks, Srinivasan and Chatha [4] proposed an approach that integrated DVFS into pipeline scheduling with MILP and applied DPM as the final design step. Wang [5] first transformed a periodic dependent task graph to an independent task graph, and then generated an energy-efficient scheduling with genetic algorithm. The state-of-the-art work [6] integrated DVFS and DPM globally, and generated the exact optimal time-triggered scheduling for dependent tasks with minimum total energy consumption.

Experimental results have shown that there is a wide variation between the best and the worst case execution time for many applications [7], [8]. For periodic independent tasks on uniprocessor systems, Xian and Lu [9] first calculated the frequency schedules for an ideal processor that could change the frequency continuously, and then used two adjacent frequencies to execute tasks. Chen [10] developed an algorithm to derive the optimal frequency scheduling for a single task, and then extended the algorithm to cope with periodic real-time tasks with different power characteristics. However, [9] and [10] ignored static power. Chen and Thiele [11] took DPM into consideration, and proposed an algorithm to derive the optimal frequency assignment without procrastination and extended it to apply procrastination scheduling for further energy reduction. But the method could only be applied to single task scheduling. For periodic independent tasks in multiprocessor systems, Xian and Lu [12] transformed the energy-aware scheduling problem to a probability-based load balancing problem and solved it with worst-fit decreasing bin-packing heuristic algorithm. But they also ignored the static power.

To the best of our knowledge, our work is the first to consider the probabilistic distribution of periodic dependent tasks for real-time MPSoCs, which integrates DVFS and DPM with scheduling globally and generates an exact optimal time-triggered intra-task scheduling and voltage assignment with minimized overall expected energy consumption.

III. Models and Problem Statements

In this section, we first introduce the hardware model, task model and some fundamental concepts which will later be used in this paper. And then the problem that we study is also described in detail.

A. Hardware Model

In this paper, we employ a distributed-memory multi-core hardware model, as shown in Fig. 1. Let U represent the

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

hardware architecture that consists of M cores, and $U = \{u_1, u_2, \cdots, u_M\}$. Each core has its local data and code memory, and is connected via a high-bandwidth interconnection network. If a core needs data that is not available in its local memory, it will send a request to the interconnection network, and the interconnection network will fetch data for the core.

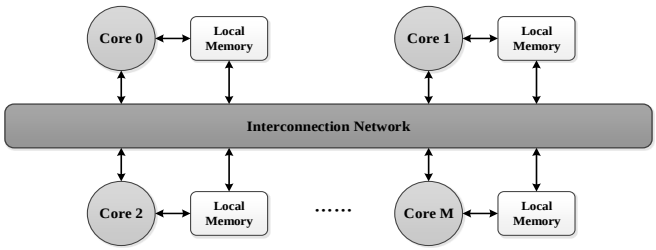


Fig. 1. Hardware Model

B. Task Model

In this paper, we study an application set running on the multi-core system. The application set is denoted as A, and it consists of a set of independent periodic applications denoted as J. And the applications can run in parallel. An application $J \in A$ is modeled as a directed acyclic task graph (DAG) $G(V, E, H)$, where V represents the set of tasks T, E records the set of dependencies between different tasks, and H is the set of the periods of the applications. It is assumed that the deadline D of the application equals to its period H.

The communication between two tasks mapped on different cores is regarded as a special task, of which the execution time is modeled as the communication overhead. And then the communication task is integrated into the original task graph.

W_i denotes the WCEC of task $T_i \in V$, and it is divided into M_i bins. The k^{th} bin of task T_i contains b_{ik} cycles. Therefore, we have the following equation.

$$\sum_{k=1}^{M_i} b_{ik} = W_i \tag{1}$$

Let P_{ik} represent the probability that the k^{th} bin of task T_i is needed in each period. Thus, $P_i = \{P_{i1}, P_{i2}, \cdots, P_{iM_i}\}$ denotes the probabilistic distribution of task T_i.

In this paper, we follow the periodic time-triggered non-preemptive scheduling policy, which provides a fully deterministic real-time scheduling behavior. The task profiling information is denoted as R. For each task's profiling information $r_i \in R$, it consists of W_i, P_i, h_i and d_i, where W_i is the WCEC, P_i is the probabilistic distribution, h_i and d_i are the period and deadline of T_i, respectively. With the profiling information and the task mapping scheme, the scheduler determines the start time of each task, and the voltage/frequency assignment for each bin.

C. Energy Model

We assume that each core in the MPSoC supports DVFS and DPM. And each core has s discrete voltage/frequency levels, which are denoted as $\{(V_1, f_1), (V_2, f_2), \cdots, (V_s, f_s)\}$. The voltage (frequency) levels from $V_1(f_1)$ to $V_s(f_s)$ are sorted in ascending order, in which $V_1(f_1)$ is the lowest and $V_s(f_s)$ is the highest. The voltage/frequency level of a core can be changed independently without affecting other cores. As we combine inter-task and intra-task scheduling, the voltage/frequency of a core may change during the execution of a task, but must not change during the execution of a bin.

Given a time-triggered schedule S, the total energy consumption $E_t(S)$ in one hyper-period is as follows:

$$E_t(S) = E_d(S) + E_i(S) + E_s(S) + E_{m-ov}(S) + E_{v-ov}(S) \tag{2}$$

$E_d(S)$ is the total energy consumption when the cores are executing tasks. $E_i(S)$ and $E_s(S)$ are the total energy consumption in idle mode and sleep mode, respectively. $E_{m-ov}(S)$ and $E_{v-ov}(S)$ are the total energy consumed by mode switching and voltage switching, respectively. Next, we will present the formulation of each part of $E_t(S)$.

The total power consumed in a core consists of dynamic and static power. Dynamic power P_{dyn-j} is dissipated due to charging and discharging of capacitances, and it can be represented as follows:

$$P_{dyn-j} = C_{EFF} V_j^2 f_j \tag{3}$$

where C_{EFF} is the total effective switching capacitance of the core, V_j is the supplied voltage level, and f_j is the operating frequency. The analytical processor energy model from [13], [14] and [15] is employed, whose accuracy has already been verified with SPICE simulation. The cycle duration $t_{cycle-j}$ is calculated with a modified alpha model.

$$t_{cycle-j} = \frac{L_d \cdot K_6}{(V_j - V_{TH-j})^{\alpha}} \tag{4}$$

where K_6 is a technology-related constant and L_d is estimated as the average logic depth of all instructions' critical path of the processor. The threshold voltage is represented as follows:

$$V_{TH-j} = V_{TH1} - K_1 \cdot V_j - K_2 \cdot V_{BS} \tag{5}$$

where V_{TH1}, K_1, K_2 are all technology-related constants and V_{BS} is the body bias voltage.

Static power is dissipated due to the subthreshold leakage current $I_{subth-j}$ and the reverse bias junction current I_{junc}. Hence, the static power P_{sta-j} is given by:

$$P_{sta-j} = L_g \cdot (V_j \cdot I_{subth-j} + |V_{BS}| \cdot I_{junc}) \tag{6}$$

where I_{junc} can be estimated as a constant and $I_{subth-j}$ can be calculated as follows:

$$I_{subth-j} = K_3 \cdot e^{K_4 V_j} \cdot e^{K_5 V_{BS}} \tag{7}$$

where K_3, K_4 and K_5 are all technology-related constants. To avoid junction leakage power overriding the gain in lowering $I_{subth-j}$, V_{BS} should be constrained between -1V and 0V.

Let P_{on} be the intrinsic energy that is needed to keep the processor on (idle power). Thus the total power consumption $P_{power-j}$ with frequency f_j can be represented as follows:

$$P_{power-j} = P_{dyn-j} + P_{sta-j} + P_{on} \tag{8}$$

The energy consumption when executing the k^{th} bin of task T_i with frequency f_j is:

$$E_d(k, T_i, f_j) = P_{power-j} \cdot \frac{b_{ik}}{f_j} \tag{9}$$

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

Before introducing how to calculate idle energy $E_i(S)$ and sleep energy $E_s(S)$, we explain the definition of break-even time T_{BET} first. As the overhead of mode switching between active mode and sleep mode is nonnegligible, T_{BET} indicates the minimum time that the processor should stay in sleep mode. Only if the time interval is longer than T_{BET}, will the energy saving in sleep mode be larger than the mode switching overhead. T_{BET} is shown as follows:

$$T_{BET} = \max(t_{m-ov}, \frac{E_{m-ov} - P_{sleep} \cdot t_{m-ov}}{P_{idle} - P_{sleep}}) \quad (10)$$

where t_{m-ov} and E_{m-ov} are the time overhead and the energy overhead due to mode switching, respectively. Generally, the idle power P_{idle} is larger than the sleep power P_{sleep}.

When the idle time interval is shorter than T_{BET}, the core should stay in idle mode. And the energy consumption in idle mode E_i can be calculated as follows:

$$E_i = P_{idle} \cdot t_{idle} \quad (11)$$

When the idle time interval is longer than T_{BET}, the core should switch to sleep mode. The energy consumption in sleep mode E_s can be represented as follows:

$$E_s = P_{sleep} \cdot t_{sleep} \quad (12)$$

For modern processors that support DVFS such as Marvell PXA270 [18], frequency and voltage transition are both controlled by software which usually takes several hundred microseconds. Let t_{v-ov} denote the frequency and voltage transition time overhead. The energy overhead for frequency and voltage transition can be derived as follows:

$$E_{v-ov} = P_{power-j} \cdot t_{v-ov} \quad (13)$$

where $P_{power-j}$ is the power consumption at the voltage level before the transition.

D. Problem Statement

Given an application set A with task profile R, a multi-core SoC with M cores that each has s discrete voltage/frequency levels and task mapping for all tasks, the expected energy optimization problem is to find a voltage assignment for each bin and a time-triggered scheduling S, such that the total expected energy is optimized and all applications finish before their deadlines.

IV. MOTIVATION

This section presents a motivating example to show that it is more energy-efficient if the probabilistic distribution of the tasks are utilized for intra-task scheduling. Assume that there are two applications $J_1 = \{T_1^1, T_1^2, T_1^3, T_1^4\}$ and $J_2 = \{T_2^1, T_2^2, T_2^3, T_2^4\}$, which are modeled by DAG as shown in Fig. 2. T_1^i and T_2^i denote the i^{th} task of J_1 and J_2, respectively. The communication of two tasks in different cores is neglected for simplicity. The periods of the tasks in application J_1 and J_2 are 120ms and 60ms, respectively. Thus, the hyper-period (least common multiple) of application J_1 and J_2 is 120ms. Thus, in one hyper-period, the tasks in J_1 and J_2 are executed once and twice, respectively. All the tasks in application J_1 and J_2 are equally divided into two bins, so that the WCECs for the bins of the same task are equal. The probability that the first bin is consumed is 100% and 50% for the second bin.

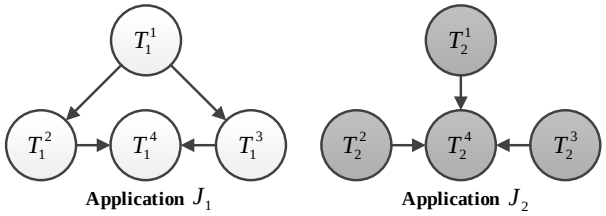


Fig. 2. DAGs for Application J_1 and J_2

The dual-core hardware architecture is represented by U, and $U = \{u_1, u_2\}$. Assume that each core has two discrete voltage/frequency levels, the high level f_H and the low level f_L. The frequency at the high level and the low level are normalized as 1 and 0.5, respectively. The dynamic power at the high level and the low level are 0.4W and 0.1W, respectively. The static power at the high level and the low level are 0.16W and 0.12W, respectively. Idle power is 0.15W. The mode switching time overhead and energy overhead are 25ms and 1mJ, respectively. Thus, T_{BET}=25ms can be calculated with (10). Task profiling and mapping information are shown in Table I. The tasks $\{T_1^2, T_1^4, T_2^2, T_2^3\}$ are mapped to core1, and the other tasks are mapped to core2. The WCETs (denoted as t_H) of the bins are profiled with f_H. If a bin is assigned with f_L, the actual WCET is $(f_H/f_L) \times t_H$.

Table I
Task Profiles and Mapping

Symbol	T_1^1	T_1^2	T_1^3	T_1^4	T_2^1	T_2^2	T_2^3	T_2^4
Mapping	u_2	u_1	u_2	u_1	u_2	u_1	u_1	u_2
First Bin WCET(ms)	2.5	4.5	3.5	8	2	4	6.5	8
First Bin Probability	100%	100%	100%	100%	100%	100%	100%	100%
Last Bin WCET(ms)	2.5	4.5	3.5	8	2	4	6.5	8
Last Bin Probability	50%	50%	50%	50%	50%	50%	50%	50%

We compare the results of three different schedules. Scheme A comes from [6]. Though it globally integrates DVFS and DPM, the probabilistic distribution of the tasks is neglected. Scheme B utilizes the schedule and voltage assignment of Scheme A. It achieves an improvement by switching the core to idle during the rest of the bins when a task finishes within a certain bin. Our Scheme C globally integrates DVFS, DPM and the probabilistic distribution of all tasks. Moreover, inter-task and intra-task scheduling are both used. The results of the three scheduling schemes are shown in Table II, Fig. 3 and Fig. 4. Table II shows the frequency assignment and the expected power consumption, where *H* denotes that the task or bin is assigned with f_H and vice versa. Fig. 3 illustrates the schedules of Scheme A and Scheme B. Fig. 4 shows the schedule of Scheme C. Scheme C achieves about 26.64% power savings compared with Scheme A, and about 4.95% power savings compared with Scheme B.

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

Table II
Frequency Assignment and Expected Power Consumption

	T_1^1	T_1^2	T_1^3	T_1^4	T_2^1	T_2^2	T_2^3	T_2^4	P(W)
A	H	L	H	H	H	L	L	H	0.74
B	H	L	H	H	H	L	L	H	0.61
C	H H	L H	H H	L H	H H	H H	H H	H H	0.58

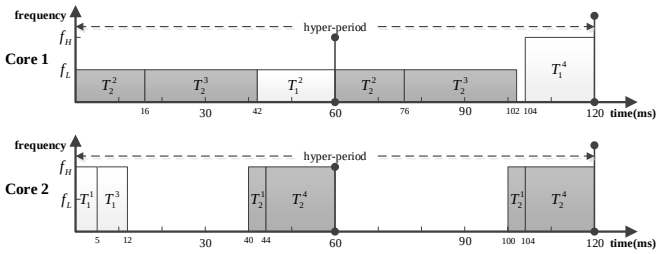


Fig. 3. Schedules of Scheme A and Scheme B

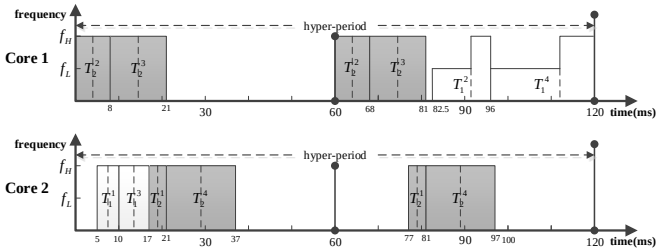


Fig. 4. Schedule of Scheme C

In Fig. 4, we can observe that by increasing the frequency of some tasks, the idle interval of core 1 can be larger than T_{BET} (25ms). Moreover, if T_2^3 finishes within the first bin, the sleep time interval is longer which further decreases the total expected power consumption. As T_2^3 can produce a longer expected sleep time interval, T_2^3 is executed after T_2^2. Among the tasks on core 1 in Fig. 4, the frequencies of the first bin of T_1^2 and T_1^4 are lower than those of the last bin. The reason is that the probabilities of the first bin and second bin are 100% and 50%, respectively. And it is more energy-efficient to lower the voltage of the bins with larger probability.

V. Proposed Approach

In this section, we formulate the expected energy optimization problem with MILP. Based on the state-of-the-art technique that integrates DVFS and DPM into task scheduling, we further utilize the probabilistic distribution of the tasks to optimize the mathematical expectation of energy consumption. Moreover, the voltage/frequency transition overhead for intra-task scheduling is also presented in the formulation. As the MILP problem may suffer from state explosion, we also propose a probability-aware refinement algorithm to decrease the number of bins of each task. To make our approach easier to understand, we list the scheduling variables, intermediate variables and constant values which are used in the MILP problem in Table III.

Table III
Variables and Constants in the MILP Problem

Scheduling Variables	
Variable Name	Description
c_{ijk}	Bin voltage assignment
s_i	Task start time

Constant Values		**Intermediate Variables**	
Constant Name	Description	Variable Name	Description
W_i	Task WCEC	t_{ik}	Bin execution time
b_{ik}	Bin execution cycles	t_i	Task execution time
M_i	Number of bins	y_{ik}	Bin voltage transition flag
t_{v-ov}	Voltage transition time overhead	t_{v-ov-i}	Task voltage transition time overhead
P_{ik}, Φ_{ik}	Bin probability	z_{pq}^{ij}	Task execution order flag
h_i, d_i	Task period and deadline	$\mathrm{ST}(T_i)$	Task start time
f_j	Core frequency	$\mathrm{FT}(T_{ik})$	Bin finish time
$P_{power-j}$	Core power with f_j	o_{ij}-b_{ij}	Adjacent task flag
T_{BET}	Break-even time	I_{ik}	Idle interval after each bin
ε	Task switching time overhead	m_{ik}	Sleep flag after each bin
H_r	Hyper-period of all tasks	tt_{ik}	$tt_{ik} = m_{ik} \cdot I_{ik}$
E_{m-ov}	Sleep mode energy overhead	x_{ijk}	$x_{ijk} = y_{ik} \cdot c_{ijk}$

A. Time-Triggered Task Scheduling

In this paper, we follow the periodic time-triggered non-preemptive scheduling policy which provides fully deterministic real-time behavior. Given the profiling information $r_i \in R$, which consists of W_i, P_i, h_i and d_i, the scheduler must decide the start time s_i of each task T_i and the voltage assignment for each bin. A set of binary variables c_{ijk} are used to formulate the frequency assignment of the k^{th} bin of task T_i: $c_{ijk} = 1$ if the k^{th} bin of task T_i is assigned with frequency f_j and vice versa. Therefore, the execution time t_{ik} of the k^{th} bin of task T_i is given as follows:

$$t_{ik} = \sum_{j=1}^{s} c_{ijk} \cdot \frac{b_{ik}}{f_j} \tag{14}$$

where b_{ik} is the number of cycles that the k^{th} bin of task T_i contains, and s is the number of voltage/frequency levels of the core. The execution time t_i of task T_i is shown as follows:

$$t_i = \sum_{k=1}^{M_i} \sum_{j=1}^{s} c_{ijk} \cdot \frac{b_{ik}}{f_j} \tag{15}$$

where M_i is the number of bins of task T_i. Each bin is assigned with only one frequency. And we have the following equation.

$$\sum_{j=1}^{s} c_{ijk} = 1 \tag{16}$$

where s is the number of voltage/frequency levels of the core.

Next we explain the necessity of formulating voltage/frequency transition overhead, and then present our formulation. For a task with 8×10^6 cycles on a 1GHz processor, if it is divided into 20 bins equally, the execution time for each bin is 400μs. Take Marvell PXA270 [18] as an example, the voltage/frequency transition time overhead is 600μs which is longer than the execution time of a bin. Therefore, it is necessary to formulate the voltage/frequency transition overhead when the tasks are divided into bins for intra-task scheduling.

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

If the voltage/frequency levels are different between two adjacent bins, there will be an overhead. A set of binary variables y_{ik} are used to formulate the existence of the voltage/frequency transition overhead: $y_{ik} = 1$ if the overhead exists between the k^{th} and $(k+1)^{th}$ bin of task T_i and vice versa. And y_{ik} and c_{ijk} have the following relationship:

$$\begin{cases} \sum_{j=1}^{s} (c_{ijk} - c_{ij(k+1)}) \cdot j = 0 \quad \Rightarrow \quad y_{ik} = 0 \\ \sum_{j=1}^{s} (c_{ijk} - c_{ij(k+1)}) \cdot j \neq 0 \quad \Rightarrow \quad y_{ik} = 1 \end{cases} \tag{17}$$

where s is the number of voltage/frequency levels of the core. To linearize y_{ik}, we present the following lemma.

LEMMA 1: Given a task T_i, let c_{ijk} represent a set of binary variables with $\sum_{j=1}^{s} c_{ijk} = 1$. y_{ik} is a binary variable and the relationship between c_{ijk} and y_{ik} is:

(a) $\sum_{j=1}^{s}(c_{ijk} - c_{ij(k+1)}) \cdot j = 0 \Rightarrow y_{ik} = 0$

(b) $\sum_{j=1}^{s}(c_{ijk} - c_{ij(k+1)}) \cdot j \neq 0 \Rightarrow y_{ik} = 1$

When y_{ik} is a positive part of the objective function that should be minimized, it can be linearized as follows.

$$\begin{cases} y_{ik} \geq \frac{1}{s-1} \cdot \sum_{j=1}^{s} (c_{ijk} - c_{ij(k+1)}) \cdot j \\ y_{ik} \geq -\frac{1}{s-1} \cdot \sum_{j=1}^{s} (c_{ijk} - c_{ij(k+1)}) \cdot j \end{cases} \tag{18}$$

PROOF: We can observe that:

$$-(s-1) \leq \sum_{j=1}^{s} (c_{ijk} - c_{ij(k+1)}) \cdot j \leq s-1 \tag{19}$$

When $\sum_{j=1}^{s}(c_{ijk} - c_{ij(k+1)}) \cdot j \neq 0$, the value of y_{ik} is 1 according to (18). When $\sum_{j=1}^{s}(c_{ijk} - c_{ij(k+1)}) \cdot j = 0$, y_{ik} could be both 0 and 1 according to (18). As y_{ik} is a positive part of the objective function, the MILP solver would choose $y_{ik} = 0$ to minimize the objective function.

Next we present the formulation of task dependency, deadline and non-preemption. First, the voltage transition time overhead t_{v-ov-i} of task T_i is given as follows:

$$t_{v-ov-i} = \sum_{k=1}^{M_i - 1} y_{ik} \cdot t_{v-ov} \tag{20}$$

where M_i is the number of bins of task T_i and t_{v-ov} denotes the time overhead per voltage/frequency transition.

Let ξ represent the task switching time overhead. The task dependency $T_a \rightarrow T_b$ indicates that the start time s_b of task T_b is later than the finish time of task T_a.

$$s_a + \sum_{k=1}^{M_a} \sum_{j=1}^{s} c_{ajk} \cdot \frac{b_{ak}}{f_j} + t_{v-ov-a} + \xi \leq s_b \tag{21}$$

Task T_b must finish no later than its deadline to meet the hard real-time deadline constraint.

$$s_b + \sum_{k=1}^{M_b} \sum_{j=1}^{s} c_{bjk} \cdot \frac{b_{bk}}{f_j} + t_{v-ov-b} + \xi \leq d_b \tag{22}$$

As the scheduling is non-preemptive, any two tasks mapped to the same core must not overlap in time. Let T_p^i denote the i^{th} instance of task T_p. We adopt the method in [6] that uses a set of binary variables z_{pq}^{ij} to describe the execution order of task T_p^i and T_q^j: $z_{pq}^{ij} = 1$ if T_p^i finishes before the start of T_q^j and vice versa. Therefore, if $p \neq q$, we have $z_{pq}^{ij} + z_{qp}^{ji} = 1$, which indicates z_{pq}^{ij} and z_{qp}^{ji} contradict with each other. Let H_r and H_{pq} denote the hyper-period of all tasks and the hyper-period of task T_p and T_q, respectively. $TC(T_p)$ denotes the set of tasks that are mapped to the same core as T_p. The following equations formulate the non-preemptive constraints.

$$\forall T_p, T_q \in TC(T_p), a = 0, \cdots, (\frac{H_{pq}}{h_p} - 1), b = 0, \cdots, (\frac{H_{pq}}{h_q} - 1):$$

$$a \cdot h_p + s_p + \sum_{k=1}^{M_p} \sum_{j=1}^{s} c_{pjk} \cdot \frac{b_{pk}}{f_j} - (1 - z_{pq}^{ij}) H_r + t_{v-ov-p} + \xi \leq b \cdot h_q + s_q \tag{23}$$

$$b \cdot h_q + s_q + \sum_{k=1}^{M_q} \sum_{j=1}^{s} c_{qjk} \cdot \frac{b_{qk}}{f_j} - z_{pq}^{ij} H_r + t_{v-ov-q} + \xi \leq a \cdot h_p + s_p \tag{24}$$

The constraints (23) and (24) guarantee that the execution time of task T_p and T_q will not overlap.

B. DPM Representation

Next we propose an approach to formulate the mathematical expectation of energy consumption in idle and sleep mode. If task T_i finishes before its WCEC, the corresponding core would switch to idle or sleep mode after the current bin. For example, when task T_i finishes within the k^{th} bin b_{ik}, the mode switching actually happens at the end of bin b_{ik}.

Previous studies did not consider the uncertainty of execution time thus failed to use slack time to optimize the mathematical expectation of energy consumption. To illustrate the problem, an example is given in Fig. 5. Task T_i and T_j are mapped to the same core. Task T_i is divided into two bins, denoted as b_{i1} and b_{i2}, and task T_i is executed before task T_j. The variables $I(TS_{i1})$ and $I(TS_{i2})$ denote the idle intervals after task T_i if task T_i finishes within the first bin and the second bin, respectively. Assume that the relationship of $I(TS_{i1})$, $I(TS_{i2})$ and T_{BET} is $I(TS_{i2}) \leq T_{BET} \leq I(TS_{i1})$. Previous studies considered the finish point of task T_i as shown in Fig. 5(a). Thus, the core should stay in idle mode during the time interval $I(TS_{i2})$. In reality, task T_i could also finish at the end of b_{i1} as shown in Fig. 5(b). And the core would enter sleep mode during $I(TS_{i1})$ to save more power. Previous scheduling techniques would generate sub-optimal results as they ignored the situation shown in Fig. 5(b).

Next we formulate the probabilistic distribution of the time interval between two adjacent tasks, and then the mathematical expectation of energy consumption in idle and sleep mode. As shown in Fig. 6, the time interval between task T_i and T_j is determined by the start time s_i of task T_i, the start time s_j of task T_j and the probabilistic distribution P_i of task T_i.

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

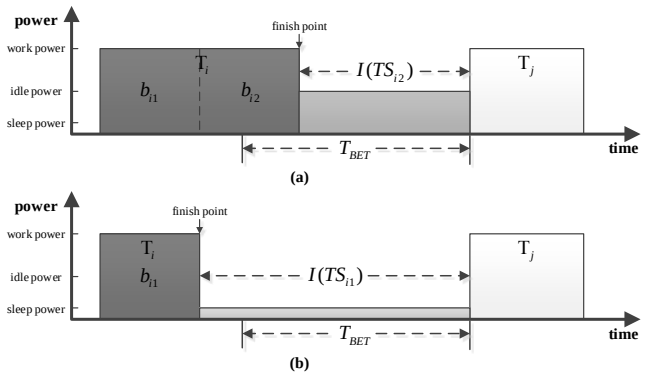


Fig. 5. Schedules Affected by Uncertain Execution Time

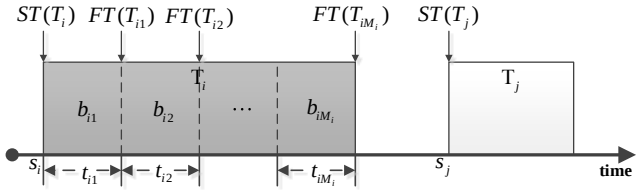


Fig. 6. Time Interval Formulation

In Fig. 6, task T_i is divided into M_i bins. As described in Section III, P_{ik} is the probability that the k^{th} bin of task T_i is needed in each period. Thus, the probability that task T_i finishes within the k^{th} bin (not the last bin) is $\Phi_{ik} = P_{ik} - P_{i(k+1)}$, and it is $\Phi_{iM_i} = P_{iM_i}$ for the last bin. $\mathrm{ST}(T_i)$ represents the start time of task T_i. The execution time t_{ik} of the k^{th} bin of task T_i can be calculated with (14). When task T_i finishes within the k^{th} bin, the finish time is denoted as $\mathrm{FT}(T_{ik})$. And $\mathrm{FT}(T_{ik})$ is calculated as follows.

$$FT(T_{ik}) = ST(T_i) + \sum_{c=1}^{k}\sum_{j=1}^{s} c_{ijc} \cdot \frac{b_{ic}}{f_j} \tag{25}$$

Based on the analysis above, the probabilistic distribution of the time interval between task T_i and T_j is formulated below.

Table IV
Probabilistic Distribution of the Time Interval

Probability	Time Interval
$\Phi_{i1} = P_{i1} - P_{i2}$	$\mathrm{ST}(T_j) - \mathrm{FT}(T_{i1})$
⋮	
$\Phi_{ik} = P_{ik} - P_{i(k+1)}$	$\mathrm{ST}(T_j) - \mathrm{FT}(T_{ik})$
⋮	
$\Phi_{iM_i} = P_{iM_i}$	$\mathrm{ST}(T_j) - \mathrm{FT}(T_{iM_i})$

Next we formulate the time interval after each task with the technique in [6]. First, the execution order variables z_{pq}^{ij} are used to construct matrix O and matrix B. If T_j is the adjacent task after T_i, $o_{ij} - b_{ij} = 1$ holds. Otherwise, $o_{ij} - b_{ij} = 0$ holds. Second, the concept of virtual task is used to formulate the last idle interval in one hyper-period. The period of the virtual task is the hyper-period of all the tasks. And the execution time of virtual task is zero. VTS and VTE are the virtual tasks at the start and the end of one hyper-period, respectively. Therefore, the idle interval I_{ik} after the k^{th} bin of task T_i can be determined as follows.

$$I_{ik} = \begin{cases} \sum_{T_j \neq VTE} (o_{ij} - b_{ij})(ST(T_j) - FT(T_{ik})) & T_i \neq VTE \\ \sum_{T_j \neq VTE} (o_{ji} - b_{ji})(ST(T_i) - FT(T_{jk})) & T_i = VTE \end{cases} \tag{26}$$

I_{VTS} and I_{VTE} represent the first and last idle interval, respectively. We actually use $I_{VTS} + I_{VTE}$ to calculate the idle interval after the last task instance. We can see that $o_{ij} - b_{ij}$ is a binary variable, and $0 \le ST(T_j) - FT(T_{ik}) \le H_r$. Thus, (26) can be linearized according to Lem. 5.4 in [6].

Next we formulate the expected energy consumption in idle and sleep mode. A set of binary variables m_{ik} are used to denote the mode-switching decision for each time interval I_{ik}. If the idle interval is longer than T_{BET}, the core should enter sleep mode. Thus, the value of m_{ik} is determined as follows:

$$\begin{cases} I_{ik} \ge T_{BET} \Rightarrow m_{ik} = 1 \\ I_{ik} < T_{BET} \Rightarrow m_{ik} = 0 \end{cases} \tag{27}$$

The idle interval I_{ik} is bounded by the period h_i of task T_i. And Constraint (27) can be linearized using the method in [6].

$$\frac{I_{ik} - T_{BET}}{h_i} < m_{ik} \le \frac{I_{ik}}{T_{BET}} \tag{28}$$

Thus, the mathematical expectation of idle and sleep interval of a core in one hyper-period can be given as follows.

$$E(t_{sleep}) = \sum_{i=1}^{N}\sum_{k=1}^{M_i} m_{ik} \cdot \Phi_{ik} \cdot I_{ik} \tag{29}$$

$$E(t_{idle}) = \sum_{i=1}^{N}\sum_{k=1}^{M_i} (1 - m_{ik} \cdot \Phi_{ik}) \cdot I_{ik} \tag{30}$$

where N is the number of task instances in one hyper-period. It is obvious that $0 \le I_{ik} \le h_i$, and m_{ik} is a binary variable. So (29) and (30) can be linearized with the same method as (26).

Then we need to formulate the probabilistic distribution Φ_{ik} of the last time interval when $T_i = VTE$. The probabilistic distribution of the last time interval is determined by the task adjacent to VTE, and it can be formulated as follows.

$$\Phi_{ik} = \sum_{j=1}^{N} (o_{ji} - b_{ji}) \cdot \Phi_{jk} \qquad (T_i = VTE) \tag{31}$$

where o_{ji} and b_{ji} come from matrix O and matrix B, respectively. Thus, the last part of (29) and (30) is shown as follows.

$$m_{ik} \cdot I_{ik} \cdot \Phi_{ik} = m_{ik} \cdot I_{ik} \cdot \sum_{j=1}^{N} (o_{ji} - b_{ji}) \cdot \Phi_{jk} \qquad (T_i = VTE) \tag{32}$$

In order to linearize (32), intermediate variable $tt_{ik} = m_{ik} \cdot I_{ik}$ is used. Because m_{ik} is a binary variable and I_{ik} is bounded by H_r, tt_{ik} can be linearized with the same method as (26). Then we have the following representation.

$$m_{ik} \cdot I_{ik} \cdot \Phi_{ik} = \sum_{j=1}^{N} (o_{ji} - b_{ji}) \cdot \Phi_{jk} \cdot tt_{ik} \qquad (T_i = VTE) \tag{33}$$

As $0 \le \Phi_{jk} \cdot tt_{ik} \le H_r$ and $(o_{ji} - b_{ji})$ is a binary variable, (33) can also be linearized with the same method as (26).

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

From (32), we can see that k in $m_{ik} \cdot I_{ik}$ represents the number of bins that task T_i contains, and k in Φ_{jk} represents the number of bins that task T_j contains. Thus, it is recommended that all tasks are divided into the same number of bins.

C. Objective Function

The mathematical expectation of the energy overhead for mode switching in one hyper-period is given as follows:

$$E(E_{m-ov-total}) = \sum_{i=1}^{N} \sum_{k=1}^{M_i} \Phi_{ik} \cdot m_{ik} \cdot E_{m-ov} \tag{34}$$

where E_{m-ov} is the energy overhead per mode switching. The mathematical expectation of the voltage/frequency transition energy overhead in one hyper-period is given as follows:

$$\begin{aligned} E(E_{v-ov-total}) &= \sum_{i=1}^{N} \sum_{k=1}^{M_i-1} P_{i(k+1)} \cdot y_{ik} \cdot t_{v-ov} \cdot \sum_{j=1}^{s} c_{ijk} P_{power-j} \\ &= \sum_{i=1}^{N} \sum_{k=1}^{M_i-1} P_{i(k+1)} \cdot t_{v-ov} \cdot \sum_{j=1}^{s} y_{ik} c_{ijk} P_{power-j} \end{aligned} \tag{35}$$

In order to linearize (35), intermediate variable $x_{ijk} = y_{ik} c_{ijk}$ is used, and x_{ijk} can be linearized as follows.

$$\begin{cases} 0 \le x_{ijk} \le 1 \\ x_{ijk} \le y_{ik} \\ x_{ijk} \le c_{ijk} \\ x_{ijk} \ge y_{ik} + c_{ijk} - 1 \end{cases} \tag{36}$$

Up to now, we have presented the formulation for the expected energy optimization and task scheduling problem. In this paper, our goal is to minimize the mathematical expectation of energy consumption in one hyper-period and the objective function of the MILP problem is given below.

$$\begin{aligned} E(E_{total}) = &\sum_{\forall T_i} \frac{H_r}{h_i} \sum_{k=1}^{M_i} P_{ik} \sum_{j=1}^{s} c_{ijk} P_{power-j} \frac{b_{ik}}{f_j} + \\ &P_{sleep} E(t_{sleep}) + P_{idle} E(t_{idle}) + \\ &\sum_{i=1}^{N} \sum_{k=1}^{M_i} \Phi_{ik} m_{ik} E_{m-ov} + E(E_{v-ov-total}) \end{aligned} \tag{37}$$

D. Refinement Algorithm

The exploration space of the MILP problem will increase exponentially with linear increase of the number of bins for each task. A direct way to solve this problem is to decrease the number of bins for each task, which implies combining several adjacent bins together. Normally, the bins are combined in a balance way, that is, the same number of adjacent bins are combined. But this way may not be energy-efficient, as it neglects the probabilistic distribution of the tasks.

Next we present an example to demonstrate that it is more energy-efficient to consider the probabilistic distribution when decreasing the number of bins. Assume a task T_i with the period of 50ms, and the WCEC is 5×10^7. It is equally divided into 20 bins and the probabilistic distribution is shown in Fig. 7. The bin number should be reduced to 5 to compress the exploration space. The processor works at the frequency of 1GHz. The working power of the processor is 0.8W and the idle power is 0.1W. We compare the mathematical expectation of energy consumption of two different schemes, the balance way and the other that considers the probabilistic distribution (refinement algorithm).

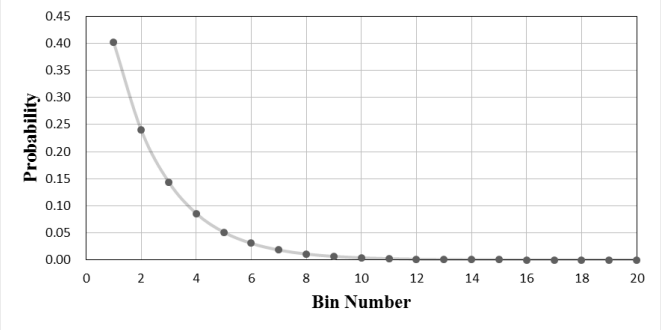


Fig. 7. Probabilistic Distribution of Each Bin

Fig. 8 and Table V show the probabilistic distribution and the bin size after decreasing the number of bins, respectively. Bin size is the number of original bins in a combined bin. For example, the bin size of the first combined bin is 4, as it consists of the first 4 original bins.

The mathematical expectation of power consumption of the balance way and the refinement algorithm are 0.26W and 0.21W, respectively. The refinement algorithm achieves 21% expected power savings with respect to the balance way. The mathematical expectation of idle time in one period with the balance way is 38.5ms, and it is 42.5ms with the refinement algorithm which is 10.2% longer than that with the balance way. Decreasing the number of bins neglects some of the probabilistic information. The balance way in the example neglects the probabilistic information of the second, third and fourth original bin. Task T_i has relatively large probability to finish within the second, third and fourth original bin. Therefore, the balance way has shorter expected idle time and larger expected power consumption.

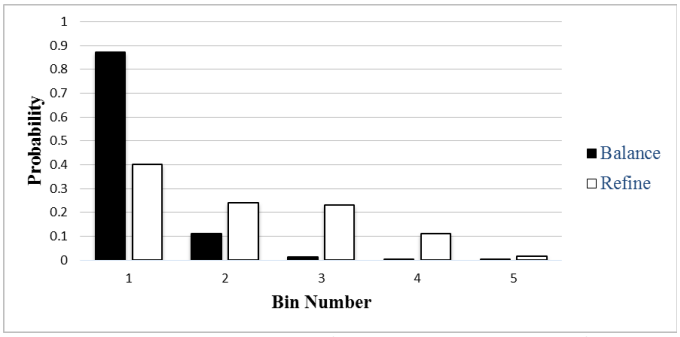


Fig. 8. Probabilistic Distribution after Decreasing the Number of Bins

Table V
Bin Size after Decreasing the Number of Bins

	Bin1	Bin2	Bin3	Bin4	Bin5
Balance	4	4	4	4	4
Refinement	1	1	2	4	12

In this paper, we propose an algorithm to decrease the

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

number of bins considering the probabilistic distribution of the tasks. It can generate a more energy-efficient probability profiling information for the MILP problem.

As the number of bins should be decreased before solving the MILP problem, the voltage assignment of each bin is not available for the refinement algorithm. Thus, it is assumed that each bin has the same number of cycles and is executed with the same frequency. Thus each bin has the same execution power and idle power, which are denoted by $power$ and $ipower$, repectively. The main idea of the refinement algorithm is to find a set of bin sizes, so that the mathematical expectation of energy consumption of the task is minimized.

Next the approach to decrease the number of bins is presented. Suppose a task originally contains n bins and the bin number should be decreased to m. Let x_k be the bin size of the k^{th} new bin. And we have the following equation.

$$\sum_{k=1}^{m} x_k = n \tag{38}$$

Φ_k is used to represent the probability that the task finishes within the k^{th} original bin. Φ_k' is used to represent the probability that the task finishes within the k^{th} combined bin. And Φ_k' can be formulated as follows.

$$\Phi_k' = \begin{cases} \sum_{i=1}^{x_1} \Phi_i & k = 1 \\ \sum_{i=1+\sum_{j=1}^{k-1} x_j}^{\sum_{j=1}^{k} x_j} \Phi_i & k = 2 \cdots (m-1) \\ \sum_{i=n-x_m+1}^{n} \Phi_i & k = m \end{cases} \tag{39}$$

Let P_k' denote the probability that the k^{th} combined bin is needed, and it can be represented as follows.

$$P_k' = \begin{cases} 1 & k = 1 \\ 1 - \sum_{i=1}^{k-1} \Phi_k' & k = 2 \cdots (m-1) \end{cases} \tag{40}$$

The mathematical expectation of energy consumption $E(Etotal)$ after decreasing the number of bins can be formulated as follows.

$$\begin{aligned} E(Etotal) &= \sum_{k=1}^{m} P_k' x_k \cdot power + \sum_{k=1}^{m-1} \Phi_k' (n - \sum_{i=1}^{k} x_i) \cdot ipower \\ &= n \cdot power - (power - ipower) \sum_{k=1}^{m-1} (x_{k+1} \cdot \sum_{i=1}^{k} \Phi_k') \end{aligned} \tag{41}$$

The refinement algorithm in this paper is to find a set of bin sizes denoted as x_k, such that $E(Etotal)$ is minimized. All possible values of x_k are calculated to find the best answer, and the complexity of the algorithm is $O(n^{m-1})$. As the decreased bin number m is usually small, the complexity of the refinement algorithm is acceptable.

***Algorithm** 1:Decrease the Number of Bins (DNB)*

Input: Probabilistic distribution Φ_k, Original bin number n, Decreased bin number m.
Output: Energy optimal bin sizes $x_k (k = 1 \cdots m)$.

```
1   x_1 ← 1, ⋯, x_{m-1} ← 1, x_m ← (n − m + 1);
2   calculate E(Etotal);
3   E(Etotal)_min ← E(Etotal);
4   for (k ← 1 ⋯ m) do
5       x_k_min ← x_k;
6   end
7   while x_k is not the last set do
8       generate the next set of x_k;
9       calculate E(Etotal);
10      if E(Etotal) < E(Etotal)_min do
11          E(Etotal)_min ← E(Etotal);
12          for (k ← 1 ⋯ m) do
13              x_k_min ← x_k;
14          end
15      end
16  end
```

In the algorithm, line 1 initializes the variables x_k. The bin size of the last combined bin is initialized as $(n - m + 1)$ and it is 1 for the other bins. Line 2 calculates the mathematical expectation of energy consumption with (41), and the energy consumption is stored in the variable $E(Etotal)_{min}$ in line 3. In line 4, the corresponding bin sizes are stored in x_k_min. From line 7 to line 16, the algorithm traverses all patterns of x_k, calculates the mathematical expectation of energy consumption and obtains the optimal bin sizes x_k.

VI. Performance Evaluations

This section describes the experiment setup and presents the case studies. We use synthetic and printer-image-processing benchmarks to demonstrate the effectiveness of our approach. The 70nm technology energy parameters of the processor in [13], [14] and [15] are used. The CPLEX solver is used to solve the MILP problem.

A. Experiment Setup

TGFF [23] is used to generate 7 periodic task graphs. Task graph 1 and 2 are normal with medium in and out task degree. Task graph 5 is thin with low in and out task degree. Task graph 6 is fat with large in and out task degree. Task graph 3, 4 and 7 are of serial-parallel structure.

The method in [9] is utilized to construct the periods, WCECs and probabilistic distribution of all the tasks. And the probabilistic distribution of the tasks are Gaussian and exponential as suggested in [16] and [17]. First, the period of each task graph is chosen between 40ms and 100ms. Second, the WCEC (W) is chosen randomly between 10^5 and 2.5×10^7 under the constraint that 4-core CPU utilization at 1.5GHz in the worst case does not exceed 1. Third, the probabilistic distribution of each task is determined. Gaussian distribution has two parameters: mean (μ) and standard deviation (σ). For each task, μ is chosen randomly between 0 and W, and $\sigma = W/6$.

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

Exponential distribution has one parameter λ ($\mu = 1/\lambda, \sigma^2 = 1/\lambda^2$). For each task, μ ($1/\lambda$) is chosen the same value as the Gaussian distribution for a fair comparison. Each task is divided into 5, 10 and 20 bins equally, and the probabilistic distribution is calculated with Matlab.

Printer-image-processing benchmark is based on the multi-functional laser printer system, whose functions include image scanning, image correction and enhancement and image printing. Image correction is needed because of the inherent defects of the scan sensors. This application is denoted as task graph 8. The scan images are not comfortable for human eyes, image enhancement is therefore needed and this application is denoted as task graph 9. When printing images, the background should be removed and the bit width should be reduced, and this application is denoted as task graph 10. All the applications above should be done within a given deadline which determines the performance of the printer. The probabilistic distribution is profiled with random images of random sizes. Apparently, there is a great potential for energy savings, as the probability of WCEC is merely 2%.

The combination of these applications is shown in Table VI.

The experiments are conducted in a 4-core hardware architecture system. The energy model is based on a 70nm technology processor in [13], [14] and [15], whose accuracy has been verified by SPICE simulation. Table VII shows the dynamic and static power consumption under different voltage levels, which is used in [6].

Table VI
Task Graph Sets

	Task Graph	Task Number
Set1	2, 3	19
Set2	1, 4	17
Set3	1, 2, 5	20
Set4	6, 7	26
Set5	8, 9, 10	31

Table VII
Dynamic and Static Power Consumption

V_{dd} (V)	0.85	0.8	0.75	0.7	0.65
f (GHz)	2.10	1.81	1.53	1.26	1.01
P_{dyn} (mW)	655.5	489.9	370.4	266.7	184.9
P_{sta} (mW)	462.7	397.6	340.3	290.1	246

According to [14], the idle power P_{on} is 276mW and the sleep power P_{sleep} is 80 µW. The state transition energy overhead is 385µJ. The total state transition time overhead t_{sw} and the voltage/frequency transition time overhead t_{v-ov} are 10ms and 600µs, respectively, which can be found in [18].

To evaluate the performance of our approach, we compare the mathematical expectation of energy consumption with the state-of-the-art technique [6] and its extension:

- Optimal DVFS-DPM (DVFS-DPM-OPT) in [6]: integrating DVFS and DPM globally with scheduling.
- Applying DVFS-DPM-OPT, then switching the core to idle mode if a task is finished before its WCEC (Sub-OPT): an extension of DVFS-DPM-OPT for a fare comparison.
- Optimal DVFS-DPM-Probability (OPT): integrating DVFS, DPM and the probabilistic distribution of all tasks with scheduling.

B. Experimental Results

Fig. 9 and Fig. 10 show the expected power consumption of the three techniques for 4 synthetic task sets with Gaussian and exponential distribution, respectively. Fig. 11 shows the expected power consumption of the printer-image-processing benchmark. The probabilistic distribution of each task is represented with 5, 10 and 20 bins. As seen from the experimental results, Sub-OPT fails to achieve expected power savings at all the task sets. The reason is that DVFS-DPM-OPT and Sub-OPT ignore the probabilistic distribution when scheduling. In contrast, OPT globally integrates DVFS, DPM and the probabilistic distribution of all tasks. Furthermore, inter-task and intra-task scheduling are both utilized. For example, OPT can accelerate the bins with low probability, thus prolongs the sleep time and saves static power. It can also assign low voltage/frequency to the bins with large probability to save dynamic power. Moreover, OPT can put the task that has a large probability to finish early in front of an idle interval to generate a larger idle interval in scheduling.

For each synthetic task with Gaussian and exponential distribution, they share the same mean and WCEC. But the expected power consumption of the same task set has a great difference as shown in Fig. 9 and Fig. 10. So the probabilistic distribution of the tasks affects the mathematical expectation of power consumption tremendously. The bins near the end of the tasks with exponential distribution have much lower probability than that with Gaussian distribution. Thus, they can be assigned with high voltage/frequency without much increase in the expected power consumption, and this will save time for other bins. Moreover, the bins with large probability can be assigned with a low voltage/frequency to save dynamic power, or the idle time can be prolonged for sleep mode.

The probabilistic distribution of the tasks in the realistic benchmark are different from Gaussian or exponential distribution. The main factors that affect the execution time of the realistic tasks are the resolution of the image, the feature of the image, the variations in the delay of the memory-accessing instructions and the miss rates of the last level of the shared cache. As the major image resolutions are 150dpi and 300dpi, the probabilistic distribution of the tasks are dual-peak. As shown in Fig. 11, OPT achieves expected power savings with respect to Sub-OPT, which demonstrates the effectiveness of our approach for realistic benchmarks.

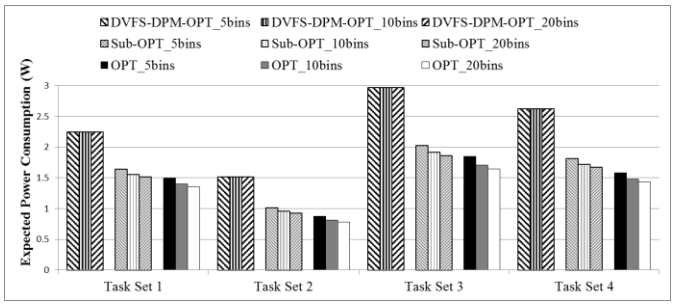


Fig. 9. Expected Power Consumption with Gaussian Distribution

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

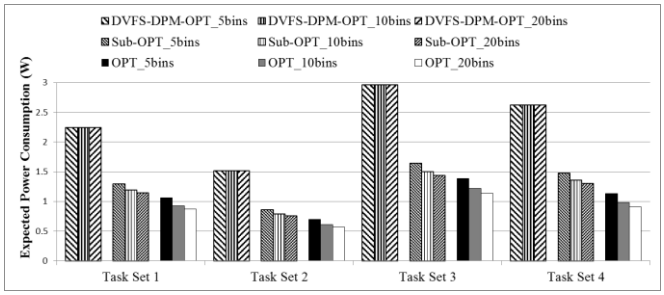


Fig. 10. Expected Power Consumption with Exponential Distribution

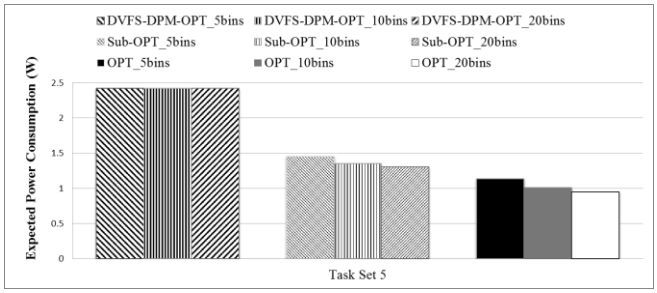
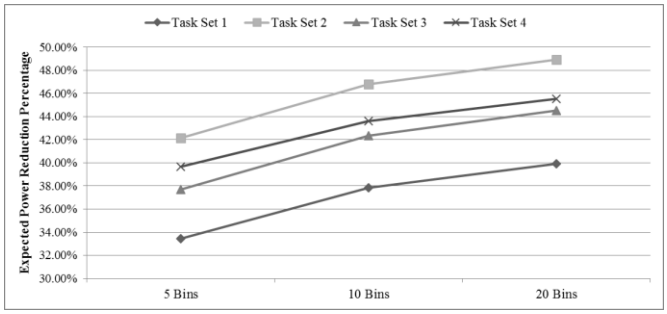
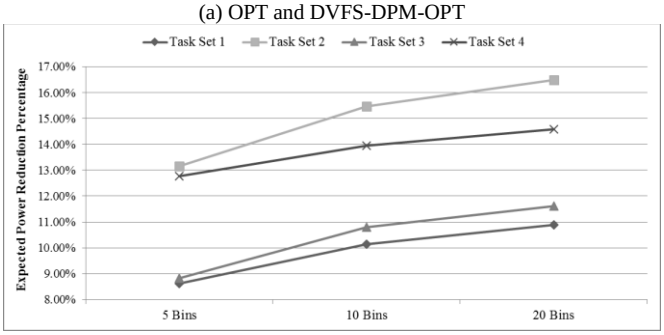


Fig. 11. Expected Power Consumption with Realistic Distribution

Moreover, the experimental results in Fig. 9, Fig. 10 and Fig. 11 show that the expected power consumption decreases when the number of bins increases. There are two reasons. One direct reason is that the expected idle time is prolonged. If one bin is divided into two sub-bins, there may be a possibility that the task finishes within the first sub-bin. Therefore, the expected idle time is prolonged when one bin is divided into two and the core is idle during the second sub-bin. It should be noted that both Sub-OPT and OPT can take advantage of the prolonged expected idle time. The other reason is that OPT can do intra-task scheduling and make full use of the probabilistic distribution to generate a more energy-efficient scheduling and voltage assignment. This can be seen from Fig. 12(b), Fig. 13(b) and Fig. 14.



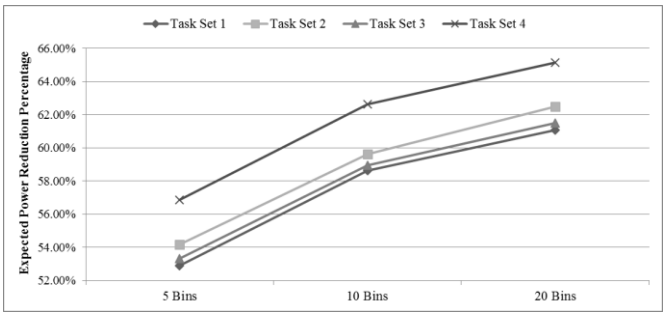
(a) OPT and DVFS-DPM-OPT



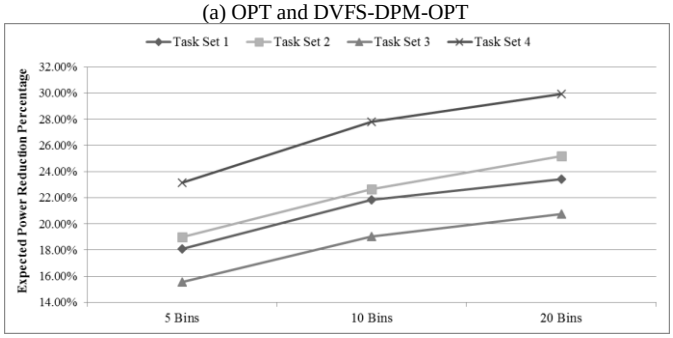
(b) OPT and Sub-OPT

Fig. 12. Power Savings of OPT with Gaussian Distribution

The experimental results in Fig. 12, Fig. 13 and Fig. 14 show that the percentage of expected power savings that OPT achieves with respect to DVFS-DPM-OPT and Sub-OPT. As seen from Fig. 12, OPT achieves 32% to 50% expected power savings with respect to DVFS-DPM-OPT, and 8% to 17% with respect to Sub-OPT when the tasks are with Gaussian distribution. In Fig. 13, OPT achieves 52% to 66% expected power savings with respect to DVFS-DPM-OPT, and 14% to 30% with respect to Sub-OPT when the tasks are with exponential distribution. In Fig. 14, OPT achieves 53% to 60% expected power savings with respect to DVFS-DPM-OPT, and 21% to 27% with respect to Sub-OPT for realistic applications. In Fig. 12(b), Fig. 13(b) and Fig. 14, OPT achieves more expected power savings with respect to Sub-OPT when the number of bins increases in all the synthetic and realistic task sets. This indicates OPT not only takes the advantage of the prolonged expected idle time, but also make full use of the probabilistic distribution to generate a more energy-efficient intra-task scheduling and voltage assignment when the number of bins increases.



(a) OPT and DVFS-DPM-OPT



(b) OPT and Sub-OPT

Fig. 13. Power Savings of OPT with Exponential Distribution

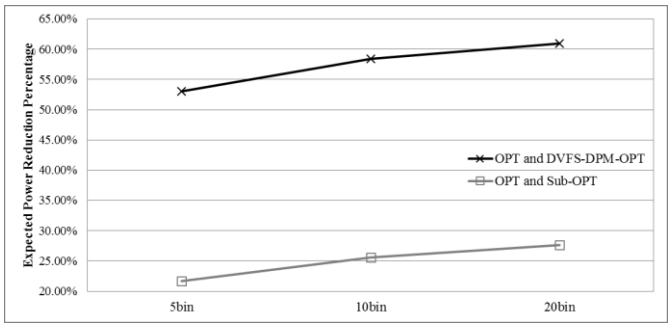


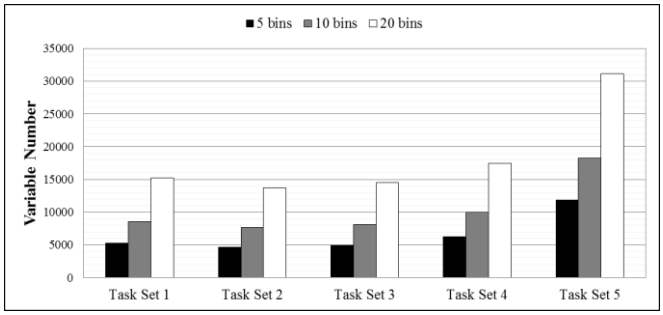
Fig. 14. Power Savings with Realistic Distribution

As seen from Fig. 12, Fig. 13 and Fig. 14, the mathematical expectation of power consumption decreases when the number of bins of each task increases. But increasing the number of

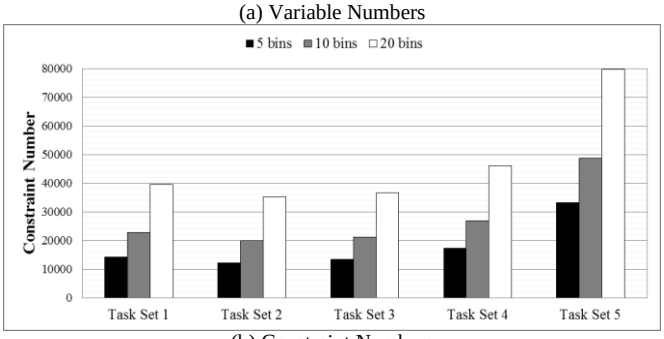
2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

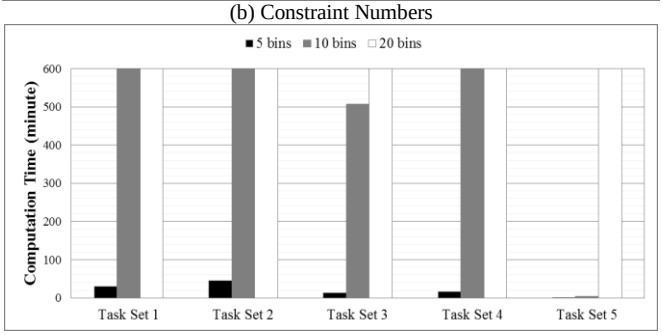
bins would generate more variables and constraints, which results in larger exploration space and longer computation time. Therefore, we conduct experiments to illustrate how the number of bins will affect the number of variables, constraints and the computation time.



(a) Variable Numbers



(b) Constraint Numbers

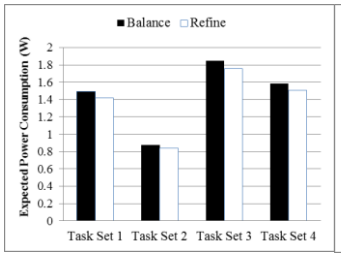


(c) Computation Time

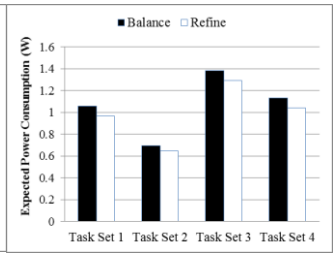
Fig. 15. Variable Numbers, Constraint Numbers and Computation Time of Different Number of Bins

Fig. 15 shows the experimental results of the four task sets with Gaussian distribution and one task set with realistic distribution, where the time limit of the MILP solver is set to 600 minutes. As seen from Fig. 15(a) and (b), the variable and constraint number increase linearly with the number of bins. Among the increased variables, a large percent of them are integer variables, which tremendously prolong the execution time as shown in Fig. 15(c). All the task sets with 5 bins can generate the results in 45 minutes, while only task set 3 and 5 with 10 bins generate the results within 600 minutes, and all the task sets with 20 bins fail to generate the results in 600 minutes. Although the variable number and constraint number of the realistic task set 5 are larger than the other task sets', the computation time is shorter. The reason is that the exploration space of task set 5 is smaller than the other task sets, as the dependencies of the tasks in task set 5 are simpler than those of other task sets. The print-image-processing tasks are usually executed in a dedicated order, for example, the background should be removed first and then halftoning algorithm is used to shorten the width of all the pixels to be printed. But when the tasks in task set 5 contain 20 bins, it is also impossible to get the results in 600 minutes. Thus, decreasing the number of bins is a highly recommended option for our approach.

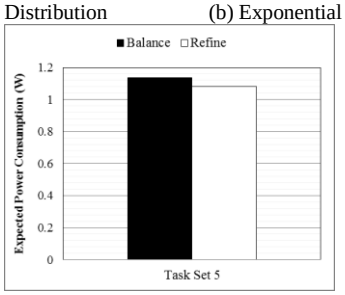
Finally, we demonstrate the effectiveness of the refinement algorithm. The bin number is decreased from 20 to 5 in the balance way and with the refinement algorithm. OPT technique is used to calculate the mathematical expectation of power consumption. The experimental results are shown in Fig. 16. The refinement algorithm achieves 4.5% to 5% expected power savings with respect to the balance way with Gaussian distribution, 6.5% to 8.5% with exponential distribution, and 4.9% with the realistic distribution. Decreasing the number of bins neglects some of the probabilistic information. The probability-aware refinement algorithm can generate a more energy-efficient bin grouping scheme to retain the more important probabilistic information in the profiling results. Meanwhile, the experimental results indicate that better profiling information may result in a more energy-efficient scheduling without much profiling overhead. Back to Fig. 12, Fig. 13 and Fig. 14, the expected power reduction percentage increases rapidly from 5 bins to 10 bins, while it slows down from 10 bins to 20 bin. When the number of bins is too large, increasing the number of bins is less helpful, and would greatly enlarge the exploration space of the MILP problem. Therefore, choosing the appropriate bin number also plays an important role for the energy-efficient scheduling problem.



(a) Gaussian Distribution



(b) Exponential Distribution



(c) Realistic Distribution

Fig. 16. Expected Power Consumption of the Balance and Refinement Algorithm

VII. Conclusion

This paper presents an intra-task scheduling approach considering the probabilistic distribution of task execution time. And it optimizes the mathematical expectation of power consumption for periodic dependent tasks with uncertain execution time running on MPSoC using DVFS and DPM. Based on this approach, the expected power optimization problem is formulated with MILP. Meanwhile the voltage/frequency transition overhead for intra-task scheduling is also presented in the formulation. Our technique can generate an optimal

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

time-triggered non-preemptive intra-task schedule and an optimal voltage assignment for each bin, which minimizes the expected power consumption of the MPSoC. Furthermore, we also propose a probability-aware refinement algorithm to decrease the number of bins. It compresses the exploration space with better probabilistic information for the MILP problem. The experimental results on synthetic and realistic benchmarks demonstrate the effectiveness of our approach compared with other existing studies.

Following is a brief flow for our approach. First, construct the probabilistic distribution of all tasks with profiling, and map the tasks to the cores. Second, apply the refinement algorithm to decrease the number of bins for all tasks. Third, formulate the MILP problem with the technique in Section V. Finally, solve the MILP problem with MILP solver. Then the start time of each task and the voltage assignment of each bin could be obtained.

For future work, we will improve the refinement algorithm. The bin number should be further decreased for the tasks whose expected power consumption are not sensitive to the change of the bin number. And we will study the energy-efficient mapping methodology for periodic dependent tasks with uncertain execution time running on MPSoCs.

REFERENCES

[1] M. Weiser et al. "Scheduling for reduced CPU energy," Mobile Computing. Springer US, 1994, pp.449-471.

[2] V. Devadas and H. Aydin. "On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications," IEEE Trans. Comput., vol. 61, no. 1, pp. 31-44, Jan. 2012.

[3] M. E. T. Gerards and J. Kuper, "Optimal DPM and DVFS for frame-based real-time systems," ACM Trans. Archit. Code Optim., vol. 9, no. 4, Jan. 2013.

[4] K. Srinivasan and K. S. Chatha. "Integer linear programming and heuristic techniques for system-level low power scheduling on multiprocessor architectures under throughput constraints," J. Integr., vol. 40, no. 3, pp. 326-354, 2007.

[5] Y. Wang et al. "Overhead-aware energy optimization for real-time streaming applications on multiprocessor System-on-Chip," ACM Trans. Des. Autom. Electron. Syst., vol. 16, no. 2, pp. 29-33, 2011.

[6] G. Chen, K. Huang and A. Knoll. "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," ACM Trans. Embed. Comput. Syst., vol. 13, no. 3, Mar. 2014.

[7] F. Mueller and J. Wegener. "A Comparison of Static Analysis and Evolutionary Testing for the Verification of Timing Constraints," J. Real-Time Syst., vol. 21, no. 3, pp. 241-268, Nov. 2001.

[8] W. Yuan and K. Nahrstedt. "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," ACM SIGOPS Oper. Syst. Rev., vol. 37, no.5, pp. 149-163, Dec. 2003.

[9] C. Xian, Y. H. Lu and Z. Li. "Dynamic voltage scaling for multitasking real-time systems with uncertain execution time," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 27, no. 8, pp. 1467-1478, Aug. 2008.

[10] J. J. Chen. "Expected energy consumption minimization in DVS systems with discrete frequencies," in 2008 Proc. Applied Computing Symp., pp. 1720-1725.

[11] J. J. Chen and L. Thiele. "Expected system energy consumption minimization in leakage-aware DVS systems," in 2008 Low Power Electronics and Design Symp., pp. 315-320.

[12] C. Xian, Y. H. Lu and Z. Li. "Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time," in Proc. 44th Annu. Design Automation Conf., 2007, pp. 664-669.

[13] R. Jejurikar, C. Pereira and R. Gupta. "Leakage aware dynamic voltage scaling for real-time embedded systems," In Proc. 41st Annu. Design Automation Conf., 2004, pp. 275-280.

[14] W. Wang and P. Mishra. "Leakage-aware energy minimization using dynamic voltage scaling and cache reconfiguration in real-time systems," in 2010 VLSI Design Conf., pp. 357-362.

[15] S. M. Martin, K. Flautner and T. Mudge, et al. "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in Proc. Computer-Aided Design Conf., 2002, pp. 721-725.

[16] R. Xu, C. Xi, R. Melhem and D. Mosse. "Practical PACE for embedded systems," in Proc. 4th Embedded Software Conf., 2004, pp. 54-63.

[17] Y. Zhang, Z. Lu, J. Lach, K. Skadron and M. R. Stan. "Optimal procrastinating voltage scheduling for hard real-time systems," in Proc. 42nd Annu. Design Automation Conf., 2005, pp. 905-908.

[18] Marvell PXA270 Processor Electrical, Mechanical, Thermal Specification, Marvell, 2009.

[19] ARM Inc. "Cortex-A57 Serious Family". Available: http:// www.arm.com/products/processors/cortex-a

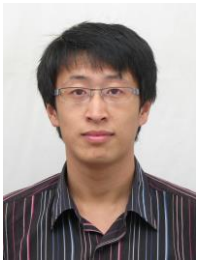
[20] Intel Inc. "Intel Core Processors". Available : http:// ark.intel.com

[21] C. Rusu, R. Xu, R. Melhem and D. MossÃ, "Energy-efficient policies for request-driven soft real-time systems," in Proc. Euromicro Conf. Real-Time Syst., 2004, pp. 175–183.

[22] W. Yuan and K. Nahrstedt. "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," in Proc. ACM Symp. Oper. Syst. Principles, 2003, pp. 149–163.

[23] R. P. Dick, D. L. Rhodes and W. Wolf. "TGFF: task graphs for free," in Proc. of Internal Workshop on Hardware/Software Codesign, 1998, pp. 97-101.

Kai Huang received the B.S.E.E. degree from Nanchang University, Nanchang, China, in 2002, and the Ph.D. degree in engineering circuit and system from Zhejiang University, Hangzhou, China, in 2008. From 2006 to 2006, he was a short-term Visitor with the TIMA Laboratory, Grenoble, France. From 2009 to 2011, he was also a Post-Doctoral Research Assistant with the Institute of VLSI Design, Zhejiang University. In 2010, he also worked as a Collaborative Expert in VERIMAG Laboratory, Grenoble, France. Since 2012, he has been an Associate Professor with the Department of Information Science and Electronic Engineering, Zhejiang University. His current research interests include embedded processors and SoC system-level design methodology and platforms.



Ke Wang received the B.S. degree from the Department of Electrical Engineering, Zhejiang University, China, in 2011. He is currently pursuing the Ph.D. degree at the Institute of VLSI Design, Zhejiang University, China. His research interests include the energy-efficient task mapping and scheduling algorithm on multiprocessor SoC and low power IC design methodology.

Dandan Zheng received the PHD from Department of Information Science & Electronic Engineering, Zhejiang University, China, in 2009. She is currently a Research Assistant of College of Electrical Engineering at the University of Zhejiang. Her research interests are VLSI physical design of nano technology. Currently the main research is the low power physical design of MPSOC.

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2853621, IEEE Transactions on Sustainable Computing

Xiaowen Jiang received the B.S. degree from Jiangxi Normal University, China, in 2012. He is currently pursuing the Ph.D. degree at the Institute of VLSI Design, Zhejiang University, China. His current research interests include multiprocessor software exploration, real-time systems and energy-efficient scheduling.

Xiaomeng Zhang received the B.S. degree from the Department of Electrical Engineering, Zhejiang University, China, in 2013. She is currently pursuing the Ph.D. degree at the Institute of VLSI Design, Zhejiang University, China. Her current research interests include multiprocessor software exploration and multi-thread code generation.

Rongjie Yan received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, Beijing, China, in 2007. She is an Assistant Researcher with the Institute of Software, Chinese Academy of Sciences. She has spent two years at VERIMAG, Grenoble, France, where she focused on compositional and incremental verification methodology, and correctness-by-construction of component-based systems. Her current research interests include modeling and formal verification of embedded systems. Recently, she worked on extra-function analysis of embedded systems

Xiaolang Yan received the B.S.E.E. and M.S.E.E. degrees from Zhejiang University, Hangzhou, China, in 1968 and 1981, respectively. From 1993 to 1994, he was a Visiting Scholar at Stanford University, Palo Alto, CA, USA. From 1994 to 1999, he was a Professor and the Dean of the Hangzhou Institute of Electronic Engineering, Hangzhou, China. Since 1999, he has been a Professor, the Dean of the Information Science and Engineering College, and the Director of the Institute of VLSI Design, Zhejiang University. His current research interests include embedded CPU design, SoC design methodology, and design for manufacturability.

2377-3782 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.