IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 9, 2019, accepted February 19, 2019, date of publication March 1, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2902353

Energy-Efficient Intra-Task DVFS Scheduling Using Linear Programming Formulation

YANG QIN[1], GANG ZENG[2], (Member, IEEE), RYO KURACHI[1], (Member, IEEE), YIXIAO LI[1], YUTAKA MATSUBARA[1], (Member, IEEE), AND HIROAKI TAKADA[1], (Member, IEEE)

[1]Graduate School of Informatics, Nagoya University, Nagoya 464-8603, Japan
[2]Graduate School of Engineering, Nagoya University, Nagoya 464-8603, Japan

Corresponding author: Yang Qin (claire@ertl.jp)

This work was supported in part by the China Scholarship Council under Grant 201606090182.

ABSTRACT In real-time embedded systems, minimizing energy consumption is one of the most important tasks. Intra-task dynamic voltage and frequency scaling (DVFS) has been the subject of much research in the task boundary of time-constrained applications for energy reduction. The problem of optimizing energy consumption with respect to intra-task DVFS scheduling can be addressed by assigning proper operational frequencies to individual basic blocks in a program while guaranteeing the deadline. Based on the profile information of a task, we first formulate the problem in terms of integer linear programming (ILP) regarding different assumptions of transition overhead. To verify the effectiveness of ILP formulations, the most representative intra-task DVFS techniques are taken for comparisons. The results of the experiments demonstrate that the proposed ILP method achieves greater energy savings than the existing approaches. Moreover, it determines the optimal scheduling strategy in reasonable execution time for applications with a limited number of blocks.

INDEX TERMS Intra-task DVFS technique, time constrained applications, minimize energy consumption, ILP formulations.

I. INTRODUCTION

With the increasing demands for computing and communication, minimizing the energy consumption of time constrained tasks has become a critical issue in real-time embedded systems. Reducing power and energy consumption is usually considered as a primary design goal to prolong battery life and increase system reliability. As one of the most common techniques used in power management, dynamic voltage frequency scaling (DVFS), which involves a dynamic adjustment of the supply voltage and frequency, has been studied extensively [1]–[4]. The DVFS approach saves the energy by decreasing the voltage and frequency to appropriate levels based on the convex relationship between the power and speed settings.

According to the location of DVFS scheduling (i.e., inside or outside the task), existing DVFS algorithms can be classified into two categories: (i) intra-task and (ii) inter-task. The intra-task DVFS approach analyzes the execution flow of a task and adjusts the voltage and CPU frequency within a single task boundary. On the contrary, the inter-task DVFS method determines the supply voltage and frequency on a task-by-task basis. In other words, a set of optimal initial frequencies are defined by analyzing the interference and schedulability of the system to save energy. Although the inter-task approach is generally effective for reducing energy in a multi-task environment, some practical limitations cannot be neglected. For instance, inter-task scheduling cannot take advantage of the slack time in a single-task model and may not be effective when one task dominates the total execution time. Moreover, the scheduler, which carries out the scheduling activities in the inter-task approach, always cooperates with modifications of the operating system [5].

In this work, we focus on minimizing the energy for a single hard real-time task with the intra-task DVFS scheduling technique. In summary, we exploit the slack time of different execution paths and adopt DVFS by modifying the program as in [6] and [7]. First, path-based analysis is conducted with a set of input data, and a control-flow-graph (CFG) can be generated as a representation. Then, the execution cycle of each basic block is estimated and the probability of each path is concluded. The profiled information provides hints

The associate editor coordinating the review of this manuscript and approving it for publication was Rui Xiong.

about the locations to change the CPU speed; for example, a sequence of instructions can be inserted into the target program as a checkpoint or voltage/frequency transition code. The checkpoint approach is completed at the compiling time, and it works like a function call, which suggests a method for calculating the operational frequencies by referring to on-line information. Contrarily, the transition-code approach aims to compute the execution frequencies off-line, and the DVFS instructions with decided frequencies are inserted directly. Typically, the checkpoint processing consumes two different kinds of overhead: one for calculating appropriate execution frequencies and the other for switching these frequencies. Compared with inserting the transition code directly, the checkpoint solution could increase the programming complexity and overhead of a real-time system. In this work, we aim at studying the intra-task scheduling problem with transition instructions inserted directly to save energy.

However, finding the optimal intra-task DVFS strategy to successfully satisfy the minimum energy consumption and time constraint is difficult because a program usually has multiple execution paths, and the variations among them are large. To solve this problem, we propose an optimal approach of integer linear programming (ILP) formulation, which determines the supply frequency for each basic block and the locations to insert transition instructions. The novel contributions of this work are summarized as follows:

(1) Based on the profile information of a task, the optimal intra-task DVFS scheduling problem is formulated as an ILP model under the assumption of zero transition overhead. In this ILP model, the primary objective is to determine the optimal execution frequency for each basic block to achieve the minimum average energy.

(2) To handle the transition overhead of DVFS, the ILP formulation is extended to determine the optimal execution frequencies while identifying the best program locations to insert the transition instructions.

(3) To verify the efficiency of the proposed formulations, experiments on randomly generated tasks and a real benchmark are conducted. Experimental results demonstrate that compared with existing intra-task DVFS techniques, the ILP method obtains the highest energy savings.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the proposed energy model and introduces existing intra-task DVFS techniques with illustrative motivational examples. In Section IV, the problem of intra-task DVFS scheduling is formulated in terms of ILP models regarding different assumptions of transition overhead. Section V verifies the proposed models by simulation experiments and a real benchmark. Finally, Section VI concludes paper and presents some possible future work.

II. RELATED WORK

In recent decades, intra-task DVFS scheduling for real-time applications has been widely explored. In this section, we summarize the studies concerned with intra-task DVFS to optimize the system energy consumption. There also exist many well-performed inter-task DVFS approaches [8]–[10]. However, the scaling approach based on inter-task DVFS is on the task-by-task perspective, which makes a big difference to the role of intra-task method. For this reason, a full survey of inter-task DVFS scheduling is not introduced in this work.

In the intra-task DVFS approach, a compiler or software tool is used to analyze the program in advance. According to the profiled information of the task, the CPU frequency can be scaled in the program phase with considerable accuracy. Shin *et al.* [5] first proposed an intra-task scheduling algorithm based on the remaining-worst-case-execution-path (RWEP). In this algorithm, the number of execution cycles for the basic block was statically analyzed, and when the program did not follow the worst path, the remaining-worst-execution-cycle (RWEC) of each block was computed and the speed was updated along with the RWEC. Although this algorithm provided an automatic solution for DVFS at specified locations, it caused energy waste because the execution probability of each path was not taken into consideration. In [11], another energy-efficient algorithm, which involved the use of average-case-execution-path (ACEP), was proposed. The main motivation of the ACEP approach was to modify the reference path by exploring the probability of each execution path so that energy use could be reduced for common cases. Nevertheless, the impact of transition overhead was not investigated enough in the two approaches described above; thus, the scheduling strategies were able to be improved to achieve more energy saving. In 2011, Tatematsu *et al.* in [6] proposed a checkpoint-extraction method to deal with the transition overhead. After estimating the RWECs by using many execution traces, the locations of checkpoint candidates were selected. After that, a greedy algorithm was proposed to rank the checkpoints in order to reduce transition overhead. Based on the execution profile of a task, Seo *et al.* in [7] proposed an optimal intra-task DVFS scheduling approach called remaining-optimal-execution-path (ROEP). The method of ROEP was proposed to minimize the average energy consumption of all execution paths. However, to the best of our knowledge, ROEP could only solve the optimal execution frequencies in a continuously variable voltage environment with zero transition overhead. Although an extension of the scheduling algorithm was suggested to handle the transition overhead and adapted to discrete voltage processors, the scheduling result would deviate from the best solution.

Besides, to better optimize the energy consumption in a multi-task environment, some approaches that use combinations of the intra-task DVFS and inter-task DVFS are also proposed. Reference [12] first considered the combined problem in a single processor, however, the phase of intra-task DVFS was the same as ROEP in [7]. Quinones *et al.* in [13] described a DVFS approach for hard real-time systems that exploited the multicore-generated intra-task slack automatically at runtime. But the slack time exploited in work [13] was provided by intertask interferences of load

operations but not execution paths, which makes a difference to our research. Recently, Pinheiro *et al.* in [14] proposed a new combination of inter and intra-task approaches for energy saving with DVFS technique, but the idea of intra-task DVFS was from RWEP in [5]. In contrast to the above, this work exploits the optimal DVFS scheduling in terms of a single task. To adapt to discrete-voltage environment with varying transition overhead, we propose generalizing ILP formulations to calculate the optimal intra-task DVFS scheduling. After solving the optimal intra-task DVFS strategy, we will try to integrate it with inter-task DVFS technique in future.

There have also been a number of scheduling studies on multi-core systems [15]–[20]. The problem with these studies is that all the tasks were assumed to be executing at the worst case execution time. In practice, real-time applications usually exhibit large variances in the actual execution time and may finish much earlier than the estimated worst time. With regard to the intra-task DVFS approach, the slack time is reduced in different execution paths, and as a result, more energy saving is expected from the scheduling. Moreover, the study of task partitioning on a real-time embedded system was related to hardware configurations, e.g., memory management, cache size, and architecture design [21]–[24]. In this work, we aim at exploring energy-efficient scheduling with respect to an intra-task technique in a uniprocessor environment.

III. INTRA-TASK DVFS SCHEDULING

A. ENERGY MODEL

Complementary metal-oxide-semiconductor circuits have two main sources of power dissipation: dynamic power dissipation and static power dissipation. The static power is dissipated due to the leakage current, while the dynamic component occurs when the circuit is switching. Since latter term is usually the dominant term in most very large scale integrations (VLSIs), we consider applying DVFS technique to reduce the dynamic dissipation in this work, as in [5]–[7] and [14]. However, it is noteworthy that the static power also has a linear relationship with the supply voltage, thus scaling voltage/frequency to reduce the dynamic dissipation also has a positive impact on decreasing the static dissipation. In this paper, we model the dynamic dissipation as follows:

$$P_d = C_L V_{DD}^2 f_{CLK}. \tag{1}$$

In (1), P_d is the dynamic power dissipation, C_L is the load capacitance, V_{DD} is the supply voltage, and f_{CLK} is the clock frequency [25]. As can be seen, the dissipated power is linearly proportional to the switching frequency f_{CLK} and quadratically depends on the supply voltage V_{DD}; thus, lowering the clock frequency and voltage should be a satisfactory approach to reducing the power dissipation. However, we aim to reduce the energy consumption of the system rather than the power dissipation. The energy of a system is formally calculated as the power consumed during a certain period of time, which can be expressed as follows:

$$E_d = \int_0^T P_d = C_L V_{DD}^2 f_{CLK} T = C_L V_{DD}^2 N_{cycle}. \tag{2}$$

In the energy model of (2), E_d is the dynamic energy consumed by the system, T is the duration of the time interval, and N_{cycle} is the number of instruction cycles executed during the period of T. We observe that E_d is highly dependent on the supply voltage V_{DD}. However, the supply voltage cannot be scaled for free. The circuit delay T_d, which sets the switching frequency, is related to the supply voltage [26], [27]:

$$\frac{1}{f_{CLK}} \propto T_d \propto \frac{V_{DD}}{(V_{DD} - V_t)^\gamma}, \tag{3}$$

where V_t is the threshold voltage, and γ is saturation velocity index ($\gamma \in [2, 3]$). With a sufficiently small threshold voltage V_t, we assume that the supply voltage V_{DD} is linearly proportional to the clock frequency f_{CLK}, as introduced in previous work [26], [28], and we use the term ''frequency scaling'' to refer to the changes of voltage and corresponding speed. After that, the energy consumption of the system can be simplified by the following equation:

$$E = K f_{CLK}^2 N_{cycle}, \tag{4}$$

where K is a system-based parameter. Thus, energy can potentially be decreased by lowering the clock frequency f_{CLK}. For simplicity, we assume $K = 1$ in this study and calculate a relative energy value to compare different intra-task DVFS approaches.

As is known, a task may exhibit different behaviors depending on different input data and different execution paths, so it is difficult to conduct fair comparisons of different intra-task DVFS approaches. Therefore, we propose to use the following definition of average energy consumption to deal with the nonuniform execution paths [7]:

$$E_{ave} = \sum_{\forall path\pi} P(\pi) E(\pi). \tag{5}$$

In (5), π represents a possible execution path of a given task, while $P(\pi)$ denotes the probability and $E(\pi)$ denotes the energy consumption of this path. With the definition of average consumption, the problem of intra-task scheduling can be summarized as follows: with a task's profile information, how can the slack time be reduced by decreasing the execution frequency so that the average energy consumption of the task can be minimized while the deadline can be satisfied.

B. MOTIVATIONAL EXAMPLE

In this section, a motivational example that illustrates the conventional intra-task scheduling techniques and their limitations is introduced. As shown in Fig. 1, a simple hard real-time task τ with three basic blocks b_1, b_2, and b_3 is given, where n_1, n_2, and n_3 denote the number of execution cycles for each respective block. In this example, the task is composed of two execution paths. The execution probability of path1 ($b_1 \rightarrow b_2$) is $P(1) = 0.1$, while the

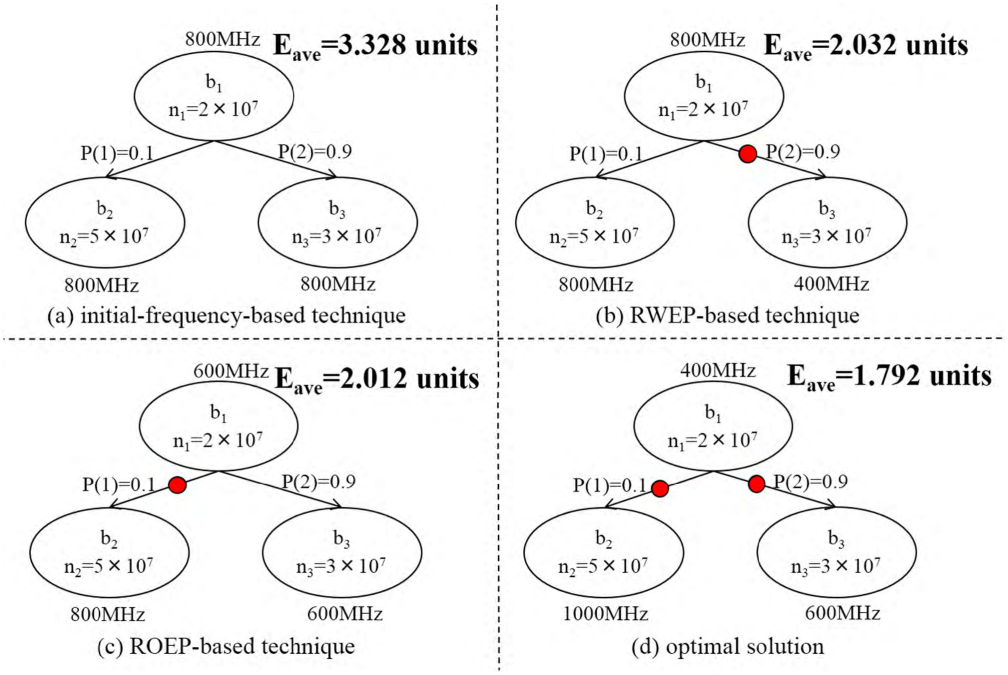


FIGURE 1. A motivational example of different intra-task scheduling techniques.

probability of path2 ($b_1 \rightarrow b_3$) is $P(2) = 0.9$. Assume that the deadline of the task is $Deadline = 100ms$, and the DVFS capability of a processor, e.g., Intel XScale processor, is [$150MHz$, $400MHz$, $600MHz$, $800MHz$, $1000MHz$] [29].

We begin the explanation of intra-task DVFS scheduling with the initial-frequency-based technique, as seen in Fig. 1(a). The initial-frequency-based technique aims to minimize the slack time by decreasing the frequency at the beginning in order to reduce the energy while meeting the deadline. Moreover, the frequency of each block is not changed during the execution of the program. In this example, the initial frequency that minimizes the energy consumption should be set tightly to the deadline: $f_1 = f_2 = f_3 = WCEC/Deadline = (n_1 + n_2)/Deadline = 700MHz$, where $WCEC$ means worst-case-execution-cycle. However, given the discrete variability of DVFS capability, f_1 is finally set as $800MHz (> 700MHz)$. After that, the average consumption of task τ by referring to (5) can be computed as $E_{ave} = P(1) \cdot E(b_1 \rightarrow b_2) + P(2) \cdot E(b_1 \rightarrow b_3) = 0.1 \cdot [(800MHz)^2 \cdot (2 \times 10^7) + (800MHz)^2 \cdot (5 \times 10^7)] + 0.9 \cdot [(800MHz)^2 \cdot (2 \times 10^7) + (800MHz)^2 \cdot (3 \times 10^7)] = 3.328$ units of energy.

By observing the scheduling result of initial-frequency-based technique, when task τ follows the worst path ($b_1 \rightarrow b_2$), the task can be finished executing at the deadline, but when task τ takes the other path ($b_1 \rightarrow b_3$), it may benefit from a period of slack time to decrease the supply frequency for energy saving. A representative example is the RWEP-based solution, in which the frequency is allowed to decrease when the program does not follow the worst path. In this strategy, the frequency of block b_3 is set according to the remaining execution cycles and remaining execution time, which should be $f_3 = n_3/(Deadline - n_1/f_1) = 400MHz$. Therefore, the average energy consumption can be computed as $E_{ave} = 0.1 \cdot [(800MHz)^2 \cdot (2 \times 10^7) + (800MHz)^2 \cdot (5 \times 10^7)] + 0.9 \cdot [(800MHz)^2 \cdot (2 \times 10^7) + (400MHz)^2 \cdot (3 \times 10^7)] = 2.032$ units. Since b_3 is set at a different execution frequency from the predecessor b_1, the transition instructions are required to insert between b_1 and b_3, presented as the red circle in Fig. 1(b). Obviously, the RWEP-based intra-task technique can achieve more energy saving compared with the initial-frequency setting because of the DVFS transition. However, the scheduling result is not optimal and the algorithm can still be improved.

For reducing energy consumption further, another path-based scheduling technique called ROEP is introduced to optimally determine the execution frequency for each basic block. Different from the above two methods, ROEP-based scheduling formulates the average energy consumption as a function of the initial speed f_1. After the analysis, the average energy E_{ave} shows a convex relationship with the starting speed f_1. In other words, an optimal starting frequency f_1 that minimizes E_{ave} can theoretically be calculated. In the scheduling approach of ROEP, $f_1 = 493MHz$, $f_2 = 841MHz$, and $f_3 = 505MHz$ is supposed to be the optimal solution that minimizes E_{ave} (the derivation process can be seen in [7]). Although the ROEP-based technique provides a function to compute the best execution speed for each basic block, it is only applicable to a continuous DVFS environment. To the best of our knowledge, most general-purpose commercial processors support DVFS with discrete frequency levels.

In the example of Intel XScale processor, $f_1 = 600MHz(> 493MHz)$, $f_2 = 800MHz$, and $f_3 = 600MHz$ should be finally set to satisfy the deadline constraint of task τ. Therefore, the average energy is calculated as $E_{ave} = 0.1 \cdot [(600MHz)^2 \cdot (2 \times 10^7) + (800MHz)^2 \cdot (5 \times 10^7) + 0.9 \cdot [(600MHz)^2 \cdot (2 \times 10^7) + (600MHz)^2 \cdot (3 \times 10^7)] = 2.012$ units.

The major approaches used in intra-task scheduling in the literature have been introduced above. It is noteworthy that the approach of RAEP in [11] is not taken as a comparison in this paper because it has been verified that RAEP performs worse than RWEP in previous work [7]. However, based on our initial findings, the optimal supply frequency of each basic block can theoretically be solved using linear programming formulations. As illustrated in Fig. 1(d), the best decisions of three blocks can be computed as $f_1 = 400MHz$, $f_2 = 1000MHz$, and $f_3 = 600MHz$, and the minimized average energy should be $E_{ave} = 0.1 \cdot [(400MHz)^2 \cdot (2 \times 10^7) + (1000MHz)^2 \cdot (5 \times 10^7)] + 0.9 \cdot [(400MHz)^2 \cdot (2 \times 10^7) + (600MHz)^2 \cdot (3 \times 10^7)] = 1.792$ units, saving 46.2% energy compared to an initial-frequency-based setting, 11.8% compared to the RWEP-based technique, and 10.9% compared to the ROEP-based technique. Consequently, we can generalize the intra-task scheduling problem as an ILP formulation to achieve the minimum average energy based on the profiled information of a task. The details of the ILP generation are presented in Section IV.

IV. LINEAR PROGRAMMING FORMULATIONS

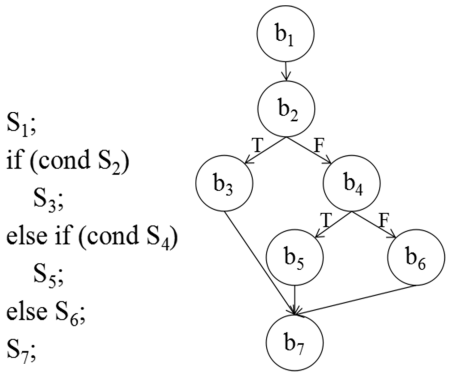
In this section, we address intra-task DVFS scheduling for a hard real-time task as ILP formulations. We first propose ILP(1) to explore the optimal supply frequency for each basic block under the assumption of zero transition overhead. After that, an extended ILP(2) is adapted to deal with the constant transition overhead. Note that when transition overhead is considered in the intra-task DVFS scheduling, ILP(2) should not only calculate the optimal supply frequencies as ILP(1) does, but it should also explore the best locations of DVFS insertion. The definitions and notations used in this work are given in Table 1.

A. ILP(1): OPTIMAL INTRA-TASK SCHEDULING WITH ZERO TRANSITION OVERHEAD

Consider a time constrained program τ. Fig. 2 is the CFG representation of the program, where each node represents a basic block, and each edge represents the execution dependency between the blocks. Let b_1 be the beginning basic block, which corresponds to the statement of S_1 in Fig. 2(a). Generally, a computer program may have multiple conditional branches. In this case, assume blocks b_2 and b_4 are conditional blocks representing the conditional expressions of S_2 and S_4 in Fig.1(a), respectively. For each conditional block, we abstract the conditional statements as True/False branches. In summary, for each basic block, the number of predecessors could be more than one (e.g., block b_7), but the successors are limited within two (True/False branches).

TABLE 1. Notations and definitions used in this work.

Symbol	Notations and their definitions
π	A possible execution path in task τ
$P(\pi)$	Execution probability of path π
$E(\pi)$	Energy consumption of path π
n	Number of basic blocks in task τ
m	Number of frequency levels provided by processor
$cycle(i)$	Number of execution cycles in block b_i
$f(j)$	The jth frequency level provided by processor
D	Required deadline of task τ
ΔE	Energy consumption during a DVFS transition
Δt	Time consumption during a DVFS transition
$x_{i,j}$	Decision variable of frequency setting; $x_{i,j} = 1$ means block b_i is executing at $f(j)$
$l_{i,j}$	Decision variable of DVFS location; $l_{i,j} = 1$ means a DVFS transition is inserted in the left branch of block b_i executing at $f(j)$
$r_{i,j}$	Decision variable of DVFS location; $r_{i,j} = 1$ means a DVFS transition is inserted in the right branch of block b_i executing at $f(j)$



(a) example program (b) CFG representation

FIGURE 2. CFG representation of an example program.

Fig.2 shows the procedure of the control-flow-graph with three conditional statements.

Assume a target processor and a time constrained task τ are given. The time constrained task is represented by n basic blocks, and the execution cycle of block b_i is expressed as $cycle(i)$. The target processor scales a set of discrete frequencies $\{f(1), f(2), \ldots, f(m)\}$, where $1 < 2 < \cdots < m$. Let j denote the jth frequency level configured by the processor, and a binary variable $x_{i,j}$ is defined to represent the frequency decision of block b_i. For example, if block b_i is executing at frequency $f(j)$, $x_{i,j}$ is set as 1; otherwise, $x_{i,j}$ is set as 0. Let Γ denote a set of execution paths of task τ. Then, the energy consumption of a possible path π ($\pi \in \Gamma$) can be defined as the sum of all the blocks that pass through the path. The scheduling problem with the intention of minimizing average energy consumption can now be formulated as ILP(1):

$$Min\ E_{ave} = \sum_{\forall \pi \in \Gamma} \sum_{i \in \pi} \sum_{j=1}^{m} P(\pi) cycle(i) f(j)^2 x_{i,j}, \qquad (6)$$

subject to

$$\sum_{i\in\pi}\sum_{j=1}^{m}\frac{cycle(i)x_{i,j}}{f(j)} \le D, \quad (\forall\pi \in \Gamma). \tag{7}$$

$$\sum_{j=1}^{m} x_{i,j} = 1, \quad (i = 1, 2, \cdots, n). \tag{8}$$

$$x_{i,j} \in [0, 1], (i = 1, 2, \cdots, n; j = 1, 2, \cdots, m). \tag{9}$$

In (6), $P(\pi)$ represents the execution probability of path π, and the section of $i \in \pi$ represents all the blocks that belong to this path. According to the definition of energy model introduced in Section III.A, the energy consumption of path π can be expressed as $\sum_{i\in\pi}\sum_{j=1}^{m} cycle(i)f(j)^2x_{i,j}$, and the average energy of all execution paths can be formulated as (6).

Besides, the section of $\sum_{j=1}^{m}\frac{cycle(i)x_{i,j}}{f(j)}$ in constraint (7) represents the execution time of block b_i, so constraint (7) guarantees that the total execution time of the blocks that pass through path π is restricted to deadline D. In addition, the constraints in (8) and (9) ensure that each block i from $\{1, 2, \cdots, n\}$ is executing at only one frequency without DVFS transition. The goal of ILP(1) is to assign each block a proper execution frequency, so that the average energy consumption is minimized and the time constraint of each path is satisfied.

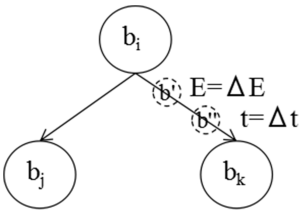


FIGURE 3. Assumption of DVFS transition.

B. ILP(2): OPTIMAL INTRA-TASK SCHEDULING WITH CONSTANT TRANSITION OVERHEAD

In the previous section, we formulated ILP(1) to assign the optimal execution frequency for each basic block, but during the execution of a task, a transition overhead is encountered for switching the frequencies. In this work, assume a processor is capable of instruction $change\_f\_V(f_{CLK})$, which controls the supply voltage and clock frequency dynamically. Given a linear relationship between the supply voltage and speed f_{CLK}, use f_{CLK} to denote the changes of voltage and frequency simultaneously. Furthermore, since the transition overhead is dependent on the processing core and there is no accurate model for estimating the transition overhead, we simply assume that two types of overhead are consumed during the DVFS transition. As seen in Fig. 3, when DVFS transition occurs between b_i and b_k, one virtual block b' consumes a constant transition energy, denoted as ΔE, and the other virtual block b'' consumes a constant transition time, denoted as Δt.

Since the transition overhead is encountered during the scaling, the clock frequency cannot be assigned arbitrarily as in ILP(1). In this section, we define another two decision variables called $l_{i,j}$ and $r_{i,j}$ to represent the locations of DVFS insertion. If block b_i is executing at $f(j)$ while its left successor is not, a DVFS transition is inserted in the left branch of block b_i, and $l_{i,j}$ is set as 1. Similarly, if the right successor is not executing at the same frequency as block b_i, then $r_{i,j}$ is set as 1 and a DVFS transition happens in the right branch of block b_i. As explained in Fig.2, when we generate the CFG representation of an example program, the number of successors from a conditional block is no more than two. As a result, it is reasonable to use $l_{i,j}$ and $r_{i,j}$ to represent the DVFS locations. In the case that block b_i has one single successor (e.g., b_1 in Fig. 2(b)), only $l_{i,j}$ is used to represent the decision of DVFS insertion.

Equations (10) and (11) present the definitions of the two parameters, where $succ[i, L]$ and $succ[i, R]$ represent the left and right block succeeded from block b_i, respectively. Take Fig. 1(d) for example; the top block b_1 is executing at the second level of frequency, which is $f(2) = 400MHz$, while its left successor b_2 is working at $f(5) = 1000MHz$. In this case, we have $x_{1,2} = 1$, while $x_{2,5} = 1$. For a reminder, $x_{i,j} = 1$ means that block b_i is executing at the *jth* frequency level. When $j = 2$, we can calculate $x_{1,2} - x_{2,2} = 1 - 0 = 1$, and then $l_{1,2}$ is set as 1 according to (10). $l_{1,2} = 1$ identifies that when block b_1 is executing at $f(2)$, which is $400MHz$ in this case, a DVFS instruction is inserted in its left branch. On the contrary, when $j \neq 2$, we always have $x_{1,j} = 0$. No matter how we set $x_{2,j}$, it must contribute to $l_{1,j} = 0$. In this paper, $l_{1,j} = 0$ implies that when block b_i is executing at frequency $f(j)$, no DVFS transition is happened between block b_i and its left successor $b_{succ[i,L]}$.

$$l_{i,j} = \begin{cases} 1, & if\ x_{i,j} - x_{succ[i,L],j} = 1 \\ 0, & if\ x_{i,j} - x_{succ[i,L],j} \neq 1 (= 0/-1). \end{cases} \tag{10}$$

$$r_{i,j} = \begin{cases} 1, & if\ x_{i,j} - x_{succ[i,R],j} = 1 \\ 0, & if\ x_{i,j} - x_{succ[i,R],j} \neq 1 (= 0/-1). \end{cases} \tag{11}$$

By observing (10) and (11), it is not difficult to see that both the equations are in a non-linear form, and this will undoubtedly complicate the solving process. For this reason, we reformulate (10) using the restrictions in restrictions (16)-(18), and we reformulate (11) as (19)-(21). For example, if $x_{i,j} - x_{succ[i,L],j}$ is computed as 1, $l_{i,j} = 1$ can be derived from (16)-(18); otherwise, if $x_{i,j} - x_{succ[i,L],j}$ is decided to be 0 or -1, $l_{i,j} = 0$ can be obtained by using the linear expressions. Together with the constraints in (14)-(15), the optimal intra-task scheduling problem with the constant transition overhead is expressed as ILP(2):

$$\begin{aligned} Min\ \ & E_{ave} \\ &= \sum_{\forall\pi\in\Gamma} P(\pi)[\sum_{i\in\pi}\sum_{j=1}^{m} cycle(i)f(j)^2x_{i,j} \\ &\quad + \Delta E(\sum_{\forall i\cap succ[i,L]\in\pi}\sum_{j=1}^{m} l_{i,j} + \sum_{\forall i\cap succ[i,R]\in\pi}\sum_{j=1}^{m} r_{i,j})], \end{aligned} \tag{12}$$

subject to

$$\sum_{i\in\pi}\sum_{j=1}^{m}\frac{cycle(i)x_{i,j}}{f(j)} \le D - \Delta t\Big(\sum_{\forall i\cap succ[i,L]\in\pi}\sum_{j=1}^{m} l_{i,j} \sum_{\forall i\cap succ[i,R]\in\pi}\sum_{j=1}^{m} r_{i,j}\Big), (\forall\pi \in \Gamma). \quad (13)$$

$$\sum_{j=1}^{m} x_{i,j} = 1, (i = 1, 2, \cdots, n). \quad (14)$$

$$x_{i,j} \in [0, 1], (i = 1, 2, \cdots, n; j = 1, 2, \cdots, m). \quad (15)$$

$$l_{i,j} - (x_{i,j} - x_{succ[i,L],j}) \ge 0 \quad (\forall i \cap succ[i, L] \in \pi; j = 1, 2, \cdots, m). \quad (16)$$

$$2l_{i,j} - (x_{i,j} - x_{succ[i,L],j}) \le 1 \quad (\forall i \cap succ[i, L] \in \pi; j = 1, 2, \cdots, m). \quad (17)$$

$$l_{i,j} \in [0, 1], (i = 1, 2, \cdots, n; j = 1, 2, \cdots, m). \quad (18)$$

$$r_{i,j} - (x_{i,j} - x_{succ[i,R],j}) \ge 0 \quad (\forall i \cap succ[i, R] \in \pi; j = 1, 2, \cdots, m). \quad (19)$$

$$2r_{i,j} - (x_{i,j} - x_{succ[i,R],j}) \le 1 \quad (\forall i \cap succ[i, R] \in \pi; j = 1, 2, \cdots, m). \quad (20)$$

$$r_{i,j} \in [0, 1], (i = 1, 2, \cdots, n; j = 1, 2, \cdots, m). \quad (21)$$

In the objective function of ILP(2), the energy consumption is composed with two distinct parts. One is the energy of basic blocks, and the other is the energy of DVFS transitions. The section of $\forall i \cap succ[i, L] \in \pi$ represents all the blocks whose left successors are in path π, while $\forall i \cap succ[i, R] \in \pi$ represents the blocks with right successors in path π. Therefore, the sum of DVFS transition times in path π can be expressed by using $l_{i,j}$ and $r_{i,j}$. Take Fig. 2(b) for example; if the program follows the path of $b_1 \rightarrow b_2 \rightarrow b_4 \rightarrow b_6 \rightarrow b_7$, the number of DVFS transitions expressed by $l_{i,j}$ and $r_{i,j}$ should be $\sum_{j=1}^{m}(l_{1,j} + r_{2,j} + r_{4,j} + l_{6,j})$.

Additionally, the constraint in (13) is proposed to guarantee the deadline of execution paths. Unlike the restriction of (7) in ILP(1), the constriction in (13) takes the DVFS transition overhead into consideration, so the execution time of a single path includes both the DVFS transition and blocks' execution. Moreover, the constraints of (16)-(18) and (19)-(21) are linear expressions of (10) and (11), and the purpose is to obtain the best locations for DVFS insertion.

V. EVALUATION RESULTS

To demonstrate the efficiency of the proposed ILP formulations, evaluation experiments of randomly generated tasks as well as a real benchmark are conducted. In this section, we present the experimental setup, procedures, and results of the tested approaches.

A. SETUP

Assume that a processor scales a range of frequencies in the range [200MHz,1400MHz], with a minimum step size of 100MHz, as with the Arm Cortex-A7 in [30]. For simplicity, it is supposed that the processor consumes no power in idle mode. The LINGO software is employed as the LP solver [31], implemented on an Intel(R) Core(TM) i7-3770 CPU (@3.4GHz) with 8.00 GB installed memory.

Since the ILP method can achieve the optimal DVFS scheduling result without interruption, we only choose four representative approaches of the previous intra-task techniques as comparisons: (i) the highest-frequency-based technique, (ii) the initial-frequency-based technique, (iii) RWEP-based technique [5], and (iv) ROEP-based technique [7]. In particular, the first highest-frequency-based technique chooses a single high speed for the entire task and there is no frequency scaling during the execution in this strategy. On the contrary, the second initial-frequency-based technique changes the frequency only at the start of a program, and the initial frequency is dependent on the worst-case-execution-cycles and the required deadline, i.e., $f\_ini = WCEC/Deadline$. In addition, the RWEP approach is based on the remaining-worst-case-execution-path, and the operational frequency of block b_i is determined using the RWECs: $f_i = RWEC_i/Deadline_i$. Correspondingly, the ROEP method calculates clock frequency using the remaining-optimal-execution-cycles (ROECs): $f_i = ROEC_i/Deadline_i$.

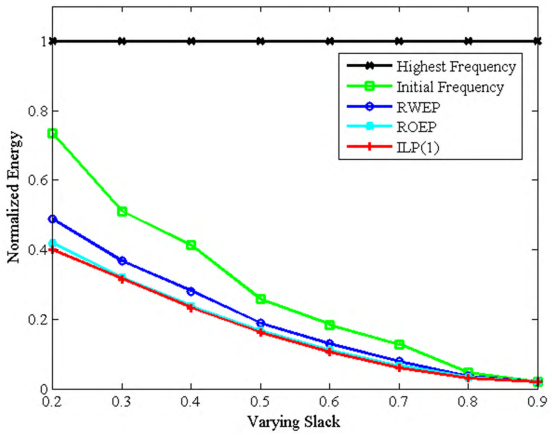


FIGURE 4. Normalized energy with respect to highest-frequency-based technique under zero transition overhead assumption at different *Slack*.

B. EXPERIMENTS OF RANDOMLY GENERATED TASKS

In the first experiment, we randomly generate 10 tasks to test our ILP models. The average results are calculated from the 10 experiments, as plotted in Figs. 4-9 and Table 2.

1) RANDOMLY GENERATED TASKS WITH ZERO TRANSITION OVERHEAD

In this experiment, an intra-task DVFS scheduling problem with zero transition overhead is investigated. We generate 10 tasks all with 30 basic blocks and then set different slack factors, which are defined as $Slack = (Deadline - WCET)/Deadline$. Note that $WCET$ is the worst-case-execution-time of a task, and it is estimated off-line. As long as the factor $Slack$ is set, the $Deadline$ of a task can be determined.

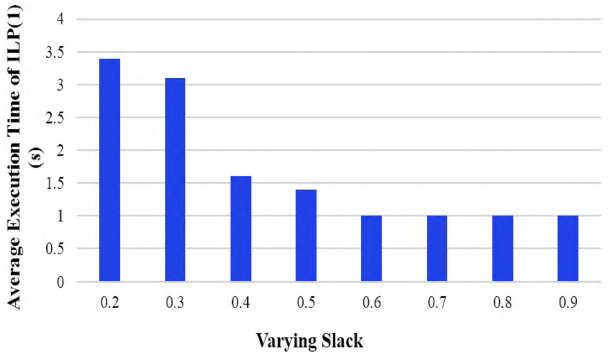


FIGURE 5. Average execution time of ILP(1) at different *Slack*.

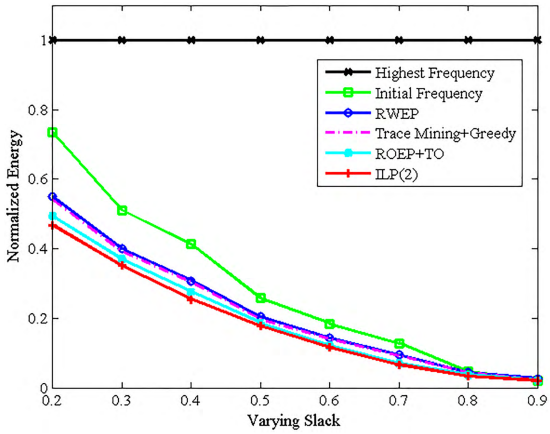


FIGURE 6. Normalized energy with respect to highest-frequency-based technique under constant transition overhead assumption at different *Slack*.

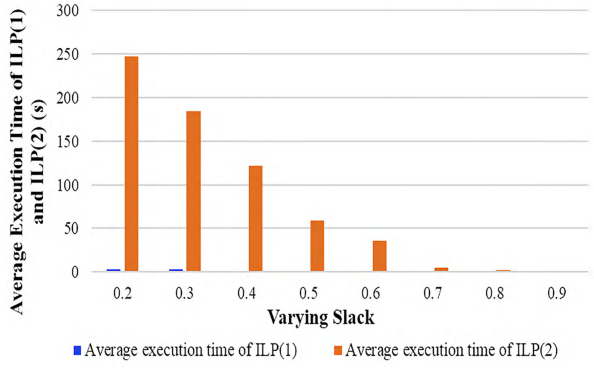


FIGURE 7. Comparison of average execution times of ILP(1) and ILP(2) at different *Slack*.

a: THE EFFECTIVENESS OF ENERGY SAVING

The average energy consumption in (5) is used to evaluate the effectiveness of different intra-task strategies. Take the results of the highest-frequency-based technique as the base line; the normalized energy of different approaches, i.e., initial-frequency setting, RWEP, ROEP, and ILP(1) are calculated respectively, as illustrated in Fig. 4. It can be observed that with the increase of *Slack*, the normalized energy of four intra-task techniques with respect to highest-frequency-based technique all show a downward trend. The reason is that large *Slack* contributes to a big difference between *WCET* and *Deadline*, so it is more likely to perform intra-task DVFS scheduling to reduce energy. When *Slack* is loosened to a large extent, i.e., $Slack = 0.9$ in this experiment, all the blocks executing at the lowest frequency $f = 200MHz$ can satisfy the restriction of the deadline. In other words, the energy cannot be further saved by decreasing the CPU speed. With respect to the energy consumption, we also find that ILP(1) always achieves the highest energy savings. Compared with the second performed best ROEP, simulation experiments prove that the ILP method reduces energy consumption by an average of 4.4%. Additionally, we analyzed the maximum saved energy of 10 random tasks at different *Slack*. It is found that ILP(1) can save a maximum of 19.5% energy compared to ROEP when *Slack* is increased to 0.8.

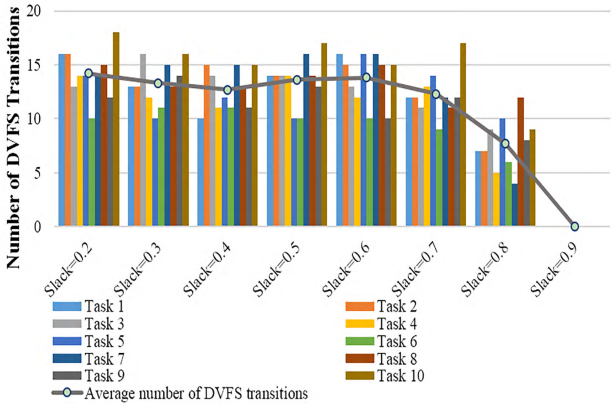


FIGURE 8. Number of DVFS transitions at different *Slack*.

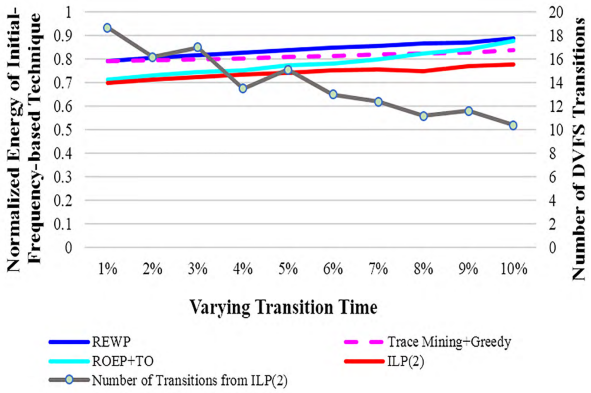


FIGURE 9. Normalized energy with respect to initial-frequency-based technique for varying transition time.

TABLE 2. Average execution time of ILP(2) to compute the optimal scheduling result ($Slack = 0.5$, $\Delta t = 5\%WCET$).

Number of blocks	10	30	50	100	200
Average execution time	1s	42.9s	8m31s	8h40m8s	$\geq$ 24h
Number of transitions	5	14	22	38	-

b: AVERAGE EXECUTION TIME OF ILP(1)

To evaluate the performance of the generalized formulation, we analyze the average execution time of ILP(1) at

different *Slack*. As shown in Fig. 5, ILP(1) always solves the optimal intra-task strategy in efficient time, with a maximum of 11s according to experimental results. Also, we observe that larger *Slack* contributes to less execution time of ILP(1). The reason is that the loosened slack time may lead to a lower initial frequency of the scheduling result, which then causes a reduction of frequency candidates. In other words, the searching space of the frequency is reduced with larger *Slack* and the execution time of ILP is undoubtedly shortened.

2) RANDOMLY GENERATED TASKS WITH CONSTANT TRANSITION OVERHEAD

In the previous section, we discussed the intra-task DVFS scheduling problem under the assumption of zero transition overhead. However, the DVFS transition overhead is encountered for switching the frequencies in practice. In this section, we conduct experiments by assuming two types of constant overhead: transition energy ΔE and transition time Δt. In practice, the transition overhead is dependent on the processing core and frequency change. In this experiment, assume that the transition time Δt is 5% of *WCET* for each task in particular, while the transition energy ΔE is linear with Δt, as in [7].

a: THE EFFECTIVENESS OF ENERGY SAVING

As introduced in Section IV.B, intra-task DVFS scheduling problem with the consideration of transition overhead can be generalized as calculating optimal execution frequency for each basic block while finding the best locations for DVFS insertion. We compare the ILP(2) model with three representative intra-task DVFS scheduling solutions: RWEP [5], Trace Mining+Greedy [6], and ROEP+TO [7]. Note that TO is an abbreviation for transition overhead, and Trace Mining+Greedy and ROEP+TO are expressly proposed for reducing the transition energy with the consideration of transition overhead. The approach of Trace Mining+Greedy is an extension of RWEP, which follows the execution paths to calculate the RWECs, but it substitutes Greedy to decide DVFS locations. The approach of ROEP+TO is an extension of ROEP, which chooses the ROECs to calculate execution frequencies, while a greedy algorithm is used to remove unnecessary transition instructions. We evaluate the average energy of different scheduling algorithms, and a downtrend of normalized energy with rising *Slack* can be observed, as in Fig. 6. The reason that larger slack time provides a higher probability to utilize intra-task DVFS scheduling has been explained in last experiment, so further discussion is not given here. Besides, experimental results show that ILP(2) saves more energy than the three other compared techniques. Summarily, if the transition overhead is counted in the DVFS scheduling, ILP(2) can be used to solve the optimal scheduling solution, achieving a maximal 34.3% when $Slack = 0.8$ and an average 7% energy reduction compared with ROEP+TO.

b: AVERAGE EXECUTION TIME OF ILP(2)

The average execution time of ILP(2) is also observed to compare the difference with ILP(1). Unlike ILP(1), which solves the supply frequency for each block, ILP(2) should solve the frequency decision and DVFS location at the same time. Therefore, the execution time of ILP(2) is believed to be longer than that of ILP(1). The average execution of two ILP models at different *Slack* is illustrated in Fig. 7. We observe that the average execution time of ILP(2) shows the same decreasing tendency as ILP(1) with the increase of *Slack*. Also, it is easy to see that ILP(2) spends much longer time than ILP(1) to derive the optimal scheduling strategy when a constant transition overhead is considered. However, the execution time of ILP(2) is still acceptable.

c: THE NUMBER OF DVFS TRANSITIONS

In this experiment, we analyze the optimal number of DVFS insertions for the 10 random tasks. While all the tasks are generated with the same number of basic blocks and execution flow, the execution cycle of each block and the probability distribution of execution paths are different. Fig. 8 shows the number of DVFS transitions for each experiment at different *Slack*. It can be seen that even when the execution flow of the tasks remains the same, different execution cycles, probability distribution, and slack time contribute to a different best solution. We then calculate the average number of DVFS transitions of the 10 experiments at different *Slack*, as seen by the gray line in Fig. 8. We find that when *Slack* is varying within a certain range, the number of DVFS transitions almost remains balanced, but when *Slack* increases beyond this range, it causes a dramatic reduction of DVFS number. In this example, when *Slack* is above 0.7, the number of DVFS shows a declining tendency, and when it reaches 0.9, energy cannot be saved by inserting DVFS instructions.

d: THE EFFECT OF TRANSITION OVERHEAD

Experiments are also conducted on random tasks with different transition overheads. The transition time Δt, which means the time taken during the DVFS transition, varies from 1% to 10% of *WCET*, as illustrated in Fig. 9. Assume that the transition energy ΔE, which is the amount of energy consumed during the transition, is linearly dependent on Δt. Through the analysis of experimental results, we find that the larger the transition overhead, the less the energy savings. From Fig. 9, we can observe the normalized energy of RWEP, Trace Mining+Greedy, ROEP+TO, and ILP(2) with respect to initial-frequency setting all have an increasing tendency. This is because larger transition time causes larger energy cost of DVFS transition, so less energy can be saved from the DVFS scheduling. Also, with the increase of transition overhead, a declining tendency can be observed from the number of DVFS insertions, as the gray line shows in Fig. 9.

e: THE EFFECT OF BLOCK NUMBER

The execution time of ILP also relies on the size of the scheduling problem. As a result, we test a set of random tasks with a varying number of blocks in this experiment. In fact, the number of execution paths may also have an impact on the execution time in intra-task scheduling, but the number of paths usually increases nearly exponentially with that of blocks. For this reason, we only evaluate the number of basic blocks for the task and use it as a representative of the problem scale. According to the experimental results, we find that the execution time of ILP(2) is positively correlated with the number of blocks. As can be seen in Table 2, when the number is limited to 100, ILP(2) solves the optimal intra-task DVFS scheduling in an acceptable time; otherwise, when the number is increased to 200, the execution time would be longer than 24 hours. Besides, it is obvious that when we assume a constant transition overhead that the number of DVFS transitions shows a rising trend with an increase in the number of blocks.

Since the applications are generally small and medium in embedded systems, we did not evaluate the large-size applications in this work. However, it is worth noting that the proposed ILP formulations are also applicable to solve large-size scheduling problems. For example, by limiting an upper execution time of LP Solver, a near-optimal intermediate solution can be calculated instead of the true optimal solution [32].

C. EXPERIMENT ON A REAL BENCHMARK

Besides the randomly generated tasks, we also test some programs that are practically used in embedded systems. In this experiment, we carry out the experiment on a real benchmark suite called CHStone. The CHStone benchmark consists of twelve real-world applications with various domains, such as arithmetic, media processing, security, and microprocessing. All the programs are written in standard C language, and the programs are self-contained, so no external libraries are necessary. In this section, we use the case study of the program DFADD to illustrate the validity of the ILP approach. In the first step, we conduct a path evaluation with 46 different input data and draw the execution flow of the main function *softfloat.c* as 59 interdependent blocks. After that, task profiling is conducted in which the execution cycles of the blocks are evaluated. Assume that the transition time Δt is 5% of the *WCET*, and the transition energy ΔE is linear with Δt. The normalized energy of different intra-task DVFS strategies at different *Slack* are illustrated in Fig. 10. According to the experimental results, we conclude that the ILP approach always solves the optimal scheduling result and achieves significant energy savings over existing intra-task techniques. Specifically, the ILP approach saves an average of 35.9% more than the initial-frequency setting, 14.1% more than RWEP, 12.3% more than Trace Mining+Greedy, and 9.4% more than ROEP+TO. Besides, the execution time of ILP is acceptable: average of 1h20m15s with different slack times. The number of optimal DVFS insertions is also

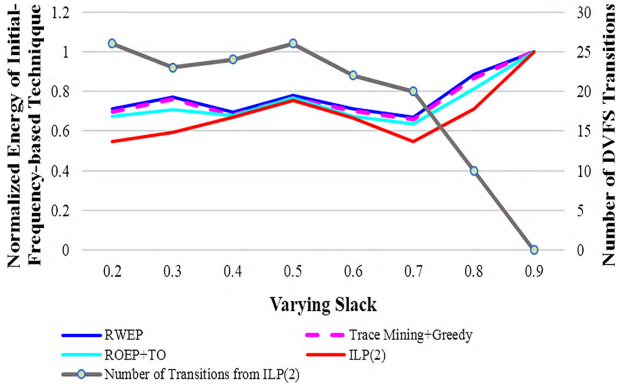


FIGURE 10. Normalized energy with respect to initial-frequency-based technique at different *Slack*.

evaluated in this experiment. Results show that the tendency of DVFS insertion with varying *Slack* is the same as in Fig. 8; it first remains balanced, and then it decreases dramatically. When the *Slack* is increased to a certain extent (*Slack* = 0.9 in this test) the intra-task DVFS cannot be applied for reducing the energy consumption anymore because the lowest frequency is capable of guaranteeing the deadline of each path.

VI. CONCLUSION

In this study, we explored intra-task DVFS scheduling for a hard real-time task to reduce energy consumption. To the best of our knowledge, this is the first work that considers the ILP method to achieve the optimal intra-task scheduling strategy in a single-task environment. With the objective of minimizing average energy, we generate two ILP formulations regarding different transition overheads. First, we generalize the intra-task DVFS scheduling problem as ILP(1) with the assumption of zero transition overhead. Determining out the optimal execution frequency for each basic block is set as the primary objective. To handle the practical transition overhead, we then extend the model to ILP(2), which simultaneously determines the supply frequencies and program locations for inserting transition instructions. In order to assess the optimality of the proposed formulations, randomly generated tasks as well as a real benchmark were tested, and both of the experiments showed that the ILP method achieves obvious energy savings compared to existing intra-task techniques. Compared with the second best approach, ILP(1) saved an average of 4.4% and a maximum of 19.5%, while ILP(2) saved an average of 7% and a maximum of 34.3%. Additionally, an experiment on a real benchmark called CHStone was conducted, and the experimental results on DFADD showed that the ILP approach achieves an average saving of 35.9% compared to the initial-frequency setting, 14.1% compared to RWEP, 12.3% compared to Trace Mining+Greedy, and 9.4% compared to ROEP+TO. For future work, we plan to test the real transition overhead of DVFS and improve the existing ILP formulations with a more precise transition model. Also, we will consider integrating the

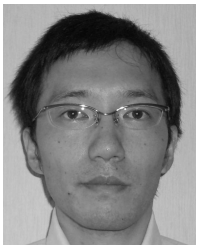
inter-task DVFS with intra-task scheduling for a multi-task environment.

REFERENCES

[1] M. Bambagini, M. Marinoni, H. Aydin, and G. Buttazzo, "Energy-aware scheduling for real-time systems: A survey," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 1, p. 7, 2016.
[2] C. Da-Ren, C. Young-Long, and C. You-Shyang, "Time and energy efficient DVS scheduling for real-time pinwheel tasks," *J. Appl. Res. Technol.*, vol. 12, no. 6, pp. 1025–1039, 2014.
[3] Y. Qin, G. Zeng, R. Kurachi, Y. Matsubara, and H. Takada, "Execution-variance-aware task allocation for energy minimization on the big.LITTLE architecture," *Sustain. Comput., Inform. Syst.*, to be published. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537917303906
[4] S. Li and F. Broekaert, "Low-power scheduling with DVFS for common RTOS on multicore platforms," *ACM SIGBED Rev.*, vol. 11, no. 1, pp. 32–37, 2014.
[5] D. Shin, J. Kim, and S. Lee, "Intra-task voltage scheduling for low-energy hard real-time applications," *IEEE Design Test Comput.*, vol. 18, no. 2, pp. 20–30, Mar. 2001.
[6] T. Tatematsu, H. Takase, G. Zeng, H. Tomiyama, and H. Takada, "Checkpoint extraction using execution traces for intra-task DVFS in embedded systems," in *Proc. 6th IEEE Int. Symp. Electron. Design, Test Appl. (DELTA)*, Jan. 2011, pp. 19–24.
[7] J. Seo, T. Kim, and J. Lee, "Optimal intratask dynamic voltage-scaling technique and its practical extensions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 1, pp. 47–57, Jan. 2006.
[8] W. Kim, J. Kim, and S. Lyul-Min, "Dynamic voltage scaling algorithm for fixed-priority real-time systems using work-demand analysis," in *Proc. ACM Int. Symp. Low Power Electron. Design*, 2003, pp. 396–401.
[9] M. K. Bhatti, C. Belleudy, and M. Auguin, "An inter-task real time DVFS scheme for multiprocessor embedded systems," in *Proc. IEEE Conf. Design Archit. Signal Image Process. (DASIP)*, Oct. 2010, pp. 136–143.
[10] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 3s, p. 111, 2014.
[11] D. Shin and J. Kim, "A profile-based energy-efficient intra-task voltage scheduling algorithm for hard real-time applications," in *Proc. ACM Int. Symp. Low Power Electron. Design*, 2001, pp. 271–274.
[12] J. Seo, T. Kim, and N. D. Dutt, "Optimal integration of inter-task and intra-task dynamic voltage scaling techniques for hard real-time applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2005, pp. 450–455.
[13] E. Quinones, J. Abella, F. J. Cazorla, and M. Valero, "Exploiting intra-task slack time of load operations for DVFS in hard real-time multi-core systems," *ACM SIGBED Rev.*, vol. 8, no. 3, pp. 32–35, 2011.
[14] D. Pinheiro, R. Gonçalves, E. Valentin, H. de Oliveira, and R. Barreto, "Inserting DVFS code in hard real-time system tasks," in *Proc. IEEE VII Brazilian Symp. Comput. Syst. Eng. (SBESC)*, Nov. 2017, pp. 23–30.
[15] J. Singh, A. Gujral, H. Singh, J. U. Singh, and N. Auluck, "Energy aware scheduling on heterogeneous multiprocessors with DVFS and duplication," in *Proc. IEEE 17th Int. Conf. Parallel Distrib. Comput., Appl. Technol. (PDCAT)*, Dec. 2016, pp. 105–112.
[16] A. Colin, A. Kandhalu, and R. R. Rajkumar, "Energy-efficient allocation of real-time applications onto single-isa heterogeneous multi-core processors," *J. Signal Process. Syst.*, vol. 84, no. 1, pp. 91–110, 2016.
[17] D. Liu, J. Spasic, P. Wang, and T. Stefanov, "Energy-efficient scheduling of real-time tasks on heterogeneous multicores using task splitting," in *Proc. IEEE 22nd Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2016, pp. 149–158.
[18] G. Zeng, Y. Matsubara, H. Tomiyama, and H. Takada, "Energy-aware task migration for multiprocessor real-time systems," *Future Gener. Comput. Syst.*, vol. 56, pp. 220–228, Mar. 2016.
[19] A. Elewi, M. Shalan, M. Awadalla, and E. M. Saad, "Energy-efficient task allocation techniques for asymmetric multiprocessor embedded systems," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 2s, p. 71, 2014.
[20] S. Pagani, A. Pathania, M. Shafique, J.-J. Chen, and J. Henkel, "Energy efficiency for clustered heterogeneous multicores," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 5, pp. 1315–1330, May 2017.
[21] G. Zeng, H. Tomiyama, and H. Takada, "A generalized framework for energy savings in hard real-time embedded systems," *IPSJ Trans. Syst. LSI Des. Methodol.*, vol. 2, pp. 167–179, 2009.
[22] H. Takase, H. Tomiyama, and H. Takada, "Partitioning and allocation of scratch-pad memory for priority-based preemptive multi-task systems," in *Proc. Conf. Design, Autom. Test Eur.*, 2010, pp. 1124–1129.
[23] Y.-J. Chen, W.-W. Chang, C.-Y. Liu, C.-E. Wu, B.-Y. Chen, and M.-Y. Tsai, "Processors allocation for MPSoCs with single ISA heterogeneous multi-core architecture," *IEEE Access*, vol. 5, pp. 4028–4036, 2017.
[24] E. Rotem, U. C. Weiser, A. Mendelson, R. Ginosar, E. Weissmann, and Y. Aizik, "H-EARtH: Heterogeneous multicore platform energy management," *Computer*, vol. 49, no. 10, pp. 47–55, Oct. 2016.
[25] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, vol. 83, no. 4, pp. 498–523, Apr. 1995.
[26] D. Shin and J. Kim, "Optimizing intratask voltage scheduling using profile and data-flow information," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 369–385, Feb. 2007.
[27] T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 584–594, Apr. 1990.
[28] G. Xie, G. Zeng, J. Jiang, C. Fan, R. Li, and K. Li, "Energy management for multiple real-time workflows on cyber–physical cloud systems," *Future Gener. Comput. Syst.*, to be published. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1731066X
[29] (2005). *Intel Xscale Microarchitecture: Benchmarks*. [Online]. Available: http://web.archive.org/web/20050326232506/developer.intel.com/design/intelxscale
[30] ODROID. *ODROID-XU3*. Accessed: 2013. [Online]. Available: https://developer.arm.com/graphics/development-platforms/odroid-xu3
[31] *LINDO Systems*. Accessed: 2017. [Online]. Available: https://www.lindo.com/
[32] Y. Qin, G. Zeng, R. Kurachi, Y. Matsubara, and H. Takada, "Energy-aware task allocation for large task sets on heterogeneous multiprocessor systems," in *Proc. IEEE 16th Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Oct. 2018, pp. 158–165.
[33] (2015). *CHStone*. [Online]. Available: http://www.ertl.jp/chstone/

YANG QIN received the master's degree in software engineering from Southeast University, China, in 2015. She is currently pursuing the Ph.D. degree with the Graduate School of Informatics, Nagoya University. Her research interests mainly include embedded systems, real-time systems, and power-aware scheduling.

GANG ZENG (S'03–M'06) received the Ph.D. degree in information science from Chiba University, in 2006. From 2006 to 2010, he was a Researcher and then an Assistant Professor with the Center for Embedded Computing Systems, Graduate School of Information Science, Nagoya University, where he is currently an Associate Professor with the Graduate School of Engineering. His research interests mainly include power-aware computing and real-time embedded system design. He is a member of IPSJ.



RYO KURACHI (M'18) received the bachelor's degree in applied electronics from the Tokyo University of Science, the master's degree in management of technology from the Tokyo University of Science, in 2007, and the Ph.D. degree in information science from Nagoya University, in 2012, where he is currently a Designated Associate Professor with the Center for Embedded Computing Systems. He was a Software Engineer with AISIN AW Co., Ltd. for few years. His research interests include embedded systems and real-time systems. Within that domain, he has investigated topics such as in-vehicle networks and real-time scheduling theory and embedded system security.

YIXIAO LI received the B.E. degree in software engineering from East China Normal University, in 2012, and the Master of Information Science degree from Nagoya University, in 2015, where he is currently a Researcher with the Center for Embedded Computing Systems, Graduate School of Informatics. His research interests include real-time operating systems, embedded multi/many-core systems, and software platforms for embedded systems.

YUTAKA MATSUBARA (M'13) received the Ph.D. degree in information science from Nagoya University, in 2011, where he was a Researcher, and then an Assistant Professor with the Center of Embedded Computing Systems, from 2009 to 2018, and currently an Associate Professor with the Graduate School of Informatics. His research interests include real-time operating systems, real-time scheduling theory, and system safety and security for embedded systems. He is a member of USENIX, IPSJ, and IEICE.

HIROAKI TAKADA (M'92) received the Ph.D. degree in information science from the University of Tokyo, in 1996, where he was a Research Associate, from 1989 to 1997. He was a Lecturer and then an Associate Professor with the Toyohashi University of Technology, from 1997 to 2003. He is currently a Professor with the Institutes of Innovation for Future Society, Nagoya University, where he is also a Professor and the Executive Director of the Center for Embedded Computing Systems, Graduate School of Informatics. His research interests include real-time operating systems, real-time scheduling theory, and embedded system design. He is a member of the ACM, IPSJ, IEICE, JSSST, and JSAE.