Energy-Adaptive Scheduling of Imprecise Computation Tasks for QoS Optimization in Real-Time MPSoC Systems

Junlong Zhou†, Jianming Yan†, Tongquan Wei†, Mingsong Chen†, Xiaobo Sharon Hu‡
†Shanghai Key Lab of Trustworthy Computing, East China Normal University, Shanghai 200241, China
‡Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46656, USA

***Abstract*—The key issue of renewable generations such as solar and wind in energy harvesting system is the uncertainty of energy availability. The characteristic of imprecise computation that accepts an approximate result when energy is limited and executes more computations yielding better results if more energy is available, can be exploited to intelligently handle the uncertainty. In this paper, we first propose a task allocation scheme that adaptively assigns real-time imprecise computation tasks to individual processors considering uncertainties in renewable energy sources. The proposed task allocation scheme enhances energy efficiency by minimizing system energy consumption followed by adapting the execution of imprecise computation tasks to the energy availability. We then present a QoS-aware task scheduling scheme that determines the optional execution cycles of tasks allocated to processors. The proposed task scheduling scheme maximizes system QoS under the energy budget constraint.**

I. Introduction

With the proliferation of cyber physical applications in harsh environments where access is rather limited, it is desirable for a real-time embedded system deployed in such an environment to scavenge energy from renewable sources to sustain its perpetual operation. The key issue of renewable energy sources is the uncertainty of output. A system with intermittent renewable energy may fail to complete a task by its deadline due to lack of energy, resulting in a timing fault. Since a timely approximate result is preferable to a precise result delayed [1], imprecise computation can be used to avoid timing faults due to power fluctuation in energy harvesting systems. This is achieved by producing an approximate result of acceptable quality by the deadline when the system cannot produce an exact result in time due to lack of energy.

Extensive investigations have been made into the design of energy harvesting real-time systems. They mainly focus on improving the efficiency of exploiting renewable energy, reducing the capacity of energy storage, and minimizing the deadline miss rate of real-time tasks. For example, Severini et al. [2] presented an improved lazy scheduling algorithm. Abdeddaim et al. [3] designed an optimal fixed-priority solution to the problem of real-time scheduling that handles energy and timing constraints. An efficient global control algorithm was presented in [4] for real-time energy harvesting systems. However, task execution precision is not considered. Stavrinides et al. [5] combined imprecise computation and bin packing to evaluate the impact of input error on the performance of a heterogeneous distributed real-time system. Task graphs with end-to-end deadlines are dynamically scheduled, however, the energy design constraint is not taken into account.

This work was partially supported by Shanghai Municipal Natural Science Foundation (Grant No. 16ZR1409000), Natural Science Foundation of China (Grant No. 91418203 and 61672230), and ECNU Outstanding Doctoral Dissertation Cultivation Plan of Action (Grant No. PY2015047). The effort of X. Hu was supported in part by U.S. NSF (Grant No. CNS-1319904). T. Wei is the corresponding author, email: tqwei@cs.ecnu.edu.cn.

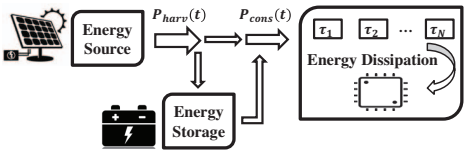

Fig. 1: The diagram of the system architecture.

Few works have investigated the energy management for imprecise computation-based real-time systems. Cortes et al. [6] proposed a two-phase approach to maximize rewards for real-time imprecise computation systems, and Yu et al. [7] presented a runtime scheduling algorithm to maximize system QoS under energy constraints. These two approaches, however, are not well suited for energy harvesting systems since the energy constraint for reward optimization is fixed. Considering the uncertainty of energy availability in harvesting systems, Kooti et al. [8] designed a two-step energy management scheme to maximize QoS. However, the presented scheme specifically targets applications in which dropouts of some of real-time tasks are allowed, thus doesnot comply with the stringent timing requirements of general real-time systems. In addition, all the above works only consider single-processor platforms, thus are not applicable to multiprocessor platforms.

In this paper, we propose an imprecise computation based task allocation and scheduling scheme for multiprocessor real-time systems that harvest energy from environments. The characteristic of imprecise computation can be exploited to intelligently handle the uncertainty of energy availability by trading off task execution precision for timeliness. Specifically, we introduce an energy-adaptive task allocation scheme which not only minimizes system energy consumption but also adapts to the execution of imprecise computation tasks to energy availability. We further propose a novel scheduling scheme to determine the number of optional cycles that each task can execute such that system QoS is maximized under the harvested energy constraint. We have conducted extensive simulations to verify the effectiveness of our schemes in reducing energy consumption and improving QoS. Results demonstrate that our schemes reduce energy consumption by up to 47.5% and improve system QoS by up to 147.8% compared to existing schemes.

II. System Architecture & Models

The system mainly consists of three modules: energy source, storage, and dissipation module. As shown in Fig. 1, the energy source module automatically scavenges renewable energy at the rate of $P_{harv}(t)$, and converts the scavenged energy into electrical energy. The energy storage module, typically in the form of a battery or supercapacitor, serves as a buffer against the uncertainty in harvested energy. A multiprocessor system-on-chip (MPSoC) is considered as the energy dissipation

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

module, which drains energy from either the energy source or storage module at the rate of $P_{cons}(t)$. The MPSoC system Θ consists of M processors $\theta_1, \theta_2, \cdots, \theta_M$, where processor θ_r $(1 \leq r \leq M)$ is characterized by a given supply voltage/frequency pair (v_r, f_r) [9], [10].

A. Task Model

Consider a frame-based task set Γ consisting of N independent real-time tasks $\tau_1, \tau_2, \cdots, \tau_N$, in which all tasks share a common deadline D that is also the frame size [11]. The whole frame is repeated until all task executions are finished. Tasks are assumed to be heterogeneous in the sense that different tasks exhibit different power consumptions on the same processor, even when executing at the same speed and temperature. This is due to that power consumptions of tasks strongly rely on circuit activities and usage patterns of different functional units [12]. Thus, the activity factor of a task, denoted by $\mu \in (0, 1]$, is introduced to capture how intensively functional units are being utilized by the task [13].

Every imprecise computation task τ_i $(1 \leq i \leq N)$ is logically decomposed into a mandatory part with execution cycles M_i and an optional part with execution cycles O_i. The mandatory part must execute to completion before the deadline and generate an acceptable result, while the optional part refines and improves the result produced by the mandatory part. The characteristic of an imprecise computation task τ_i is described by a quadruple $\tau_i : \{\mu_i, D, M_i, O_i\}$, where μ_i is the task activity factor and D is the common deadline. M_i is mandatory cycles of τ_i that must be completed before the deadline while O_i is the maximum optional cycles of τ_i. Since optional cycles are partially executed, we introduce a variable to represent the executed optional cycles of task τ_i, which is denoted as o_i and holds for $0 \leq o_i \leq O_i$. Then, the actual length of τ_i, measured by the total execution cycles, is expressed as

$$l_i = M_i + o_i. \tag{1}$$

B. Energy Model

The energy of the concerned system is modeled from the perspective of both supply and demand. We first describe the model for energy supply. Let $P_{harv}(t)$ be the harvesting power and $E_{harv}(t_1, t_2)$ be the energy scavenged from environments during time interval $[t_1, t_2]$, then $E_{harv}(t_1, t_2)$ is calculated as

$$E_{harv}(t_1, t_2) = \int_{t_1}^{t_2} P_{harv}(t)dt. \tag{2}$$

As illustrated in Fig. 1, a part of the harvested energy is consumed by the MPSoC system and the residuals are stored in the energy storage module. Both of the energy source and storage module can provide the energy to the energy dissipation module. Let $E_{sup}(t_1, t_2)$ be the supply energy available with the system during time interval $[t_1, t_2]$, and $E(t_1)$ be the energy stored in the battery at t_1, then we have

$$E_{sup}(t_1, t_2) = E_{harv}(t_1, t_2) + E(t_1). \tag{3}$$

We then describe the model for energy demand. The power consumption of a CMOS device can be modeled as the sum of static power P_{sta} and dynamic power P_{dyn}, that is,

$$P_{cons} = P_{sta} + P_{dyn}. \tag{4}$$

P_{sta} is independent of switching activity and maintains the system basic state. It can only be eliminated by turning off the system. P_{dyn} is related to processor switching activity and can be formulated as a function of supply voltage v and operating frequency f, i.e., $P_{dyn} \propto v^2 f$. Therefore, based on the system-level power model in [14], the overall power consumption of processor θ_r when executing task τ_i at (v_r, f_r) is

$$P_{cons}(\tau_i, \theta_r) = P_{sta,r} + C_r^{eff} \mu_i v_r^2 f_r, \tag{5}$$

where $P_{sta,r}$ is the static power of θ_r, C_r^{eff} is the effective switching capacitance of θ_r, and μ_i is the activity factor of τ_i.

The static power is always consumed to maintain basic circuits, and the dynamic power is only consumed when executing tasks. Thus, based on (5), the total energy consumed (or demanded) by the system during the scheduling horizon (i.e., the frame size D), denoted by E_{dem}, is calculated as

$$E_{dem} = \sum\nolimits_{r=1}^{M} (P_{sta,r} D + \sum\nolimits_{\tau_i \in \Gamma_r} C_r^{eff} \mu_i v_r^2 f_r \cdot \frac{l_i}{f_r}), \tag{6}$$

where Γ_r is the subset of tasks allocated to processor θ_r, and $\frac{l_i}{f_r}$ is the execution time of task τ_i at frequency f_r.

III. Overall Framework

Due to the intermittent nature of renewable energy sources, the energy available to support system operation varies in a predefined time domain. To analyze the system at a fine granularity, the system operation with respect to energy is divided into three states: low, medium, and high energy state. The system is deemed to be in low energy state when the energy available during the scheduling horizon is insufficient to finish the execution of all the mandatory parts of tasks. The system is considered in high energy state if the energy available during the scheduling horizon is sufficient for all tasks to finish their mandatory and optional parts. The system is in medium energy state if it is neither in low nor high energy state. In medium energy state, the mandatory parts of all tasks are ensured to finish in time while not all the optional parts of tasks have enough energy to complete their executions.

We concentrate on systems in medium energy state, and design a task allocation and scheduling scheme to maximize the QoS. It has been shown in [1] and [6] that the QoS of a task in the imprecise computation system highly depends on task optional part and can be represented as a linear or concave function of task optional cycles. In addition, the more optional cycles the task executes, the higher QoS the task generates. Thus, similar to [6], we quantitatively define a simple yet effective QoS function for a system, which is the sum of the executed CPU cycles of optional parts of all the real-time tasks in the system. It is denoted by Q and is expressed as

$$Q = \sum\nolimits_{i=1}^{N} o_i, \tag{7}$$

where o_i is the executed optional cycles of task τ_i.

Problem Definition: Given a set Γ of N imprecise computation real-time tasks and a set Θ of M processors, design an energy-efficient task assignment that adapts to the uncertainty of system available energy, and a QoS-aware task scheduling scheme that determines the cycles of assigned tasks on individual processors to maximize system QoS. The system energy consumption cannot exceed the energy supply, and

the mandatory parts of all tasks must be finished before the deadline. In other words, the problem is formulated as

$$\begin{aligned} \text{Maximize:} \quad & Q = \sum\nolimits_{i=1}^{N} o_i \\ \text{Subject to:} \quad & 0 \le o_i \le O_i \\ & E_{dem} \le E_{sup} \\ & \sum\nolimits_{\tau_i \in \Gamma_r} \frac{M_i}{f(\tau_i)} \le D, \end{aligned}$$

where $f(\tau_i)$ is the operating frequency of task τ_i.

IV. Energy-adaptive Task Allocation

Our scheme first assumes deterministic energy sources and minimizes the energy consumption by intelligently assigning tasks to individual processors, then iteratively adapts the task allocation to the uncertainty in energy availability.

A. Deterministic Task Allocation (DTA)

We designed a task allocation heuristics for deterministic energy powered systems in [10] to minimize the energy consumption by utilizing heterogeneities of both processors and tasks, as briefly summarized in the following. The system energy consumption given in (6) can be written as

$$E_{dem} = \sum\nolimits_{r=1}^{M} P_{sta,r} D + \sum\nolimits_{r=1}^{M} C_r^{eff} v_r^2 \sum\nolimits_{\tau_i \in \Gamma_r} \mu_i l_i, \quad (8)$$

where the first item is the static energy consumption and the second item is the dynamic energy consumption. The static energy consumption is independent of task allocation. As a result, the system energy consumption E_{dem} is minimal if the dynamic energy consumption given by the second item of (8) is optimized via task allocation. Let E_{dyn} represent the dynamic energy consumption, which can be formulated into the product of two vectors, that is,

$$E_{dyn}(\mathcal{A}, \mathcal{B}) = \mathcal{A} \times \mathcal{B} = \mathcal{A}_1 b_1 + \cdots + \mathcal{A}_r b_r + \cdots + \mathcal{A}_M b_M, \quad (9)$$

where $\mathcal{A} = [\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_M]$ captures hardware dependent parameters and is constant for a given MPSoC system. $\mathcal{A}_r = C_r^{eff} v_r^2 \in \mathcal{A}$ is referred to as the power dissipation factor of processor θ_r. $\mathcal{B} = [b_1, b_2, \cdots, b_M]^T$ captures task related parameters, where $b_r = \sum_{\tau_i \in \Gamma_r} \kappa_i = \sum_{\tau_i \in \Gamma_r} \mu_i l_i \in \mathcal{B}$ is referred to as the power dissipation factor of subset Γ_r and $\kappa_i = \mu_i l_i$ is referred to as the power dissipation factor of task τ_i. It is clear that $\mathcal{B}$ is not constant since it depends on the task assignment strategy. For a given set Γ of real-time tasks, the sum of power dissipation factors of all tasks in the set is a constant and can be calculated as $\sum_{i=1}^{N} \kappa_i = \sum_{r=1}^{M} \sum_{\tau_i \in \Gamma_r} \mu_i l_i = \sum_{r=1}^{M} b_r = \mathcal{Y}$.

According to the above formulation, the power dissipation of processor set Θ can be characterized by $\mathcal{A} = [\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_M]$. For the sake of easy presentation, it is assumed that $\mathcal{A}_1 \le \mathcal{A}_2 \le \cdots \le \mathcal{A}_M$ holds. Similarly, the optimum power dissipation of subsets assigned to processors can be characterized by $\mathcal{B} = [\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_M]^T$, indicating the optimum task allocation solution can minimize the dynamic energy consumption by correlating the task assignment with processor power dissipation factors. Given these, we presented and proved the theorem below [10], which shows that the dynamic energy consumption is minimized when the processor with smaller power dissipation factor ends up with the subset of its allocated tasks having a larger power dissipation factor.

Theorem 1: *If $\mathcal{A}_1 \le \mathcal{A}_2 \le \cdots \le \mathcal{A}_M$ holds for $\mathcal{A} = [\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_M]$, and $\mathcal{B}_1 + \mathcal{B}_2 + \cdots + \mathcal{B}_M$ is fixed for $\mathcal{B} = [\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_M]^T$, then dynamic energy consumption $E_{dyn}(\mathcal{A}, \mathcal{B})$ is minimized if $\mathcal{B}_1 \ge \mathcal{B}_2 \ge \cdots \ge \mathcal{B}_M$ holds.*

The DTA [10] is developed based on the above theorem. It first sorts the processors in ascending order of processor power dissipation factors and sorts the tasks in descending order of task power dissipation factors. It then iteratively assigns tasks with larger power dissipation factors to processors with smaller power dissipation factors under the processor capacity limitation and timing constraint, and constructs task subsets assigned to individual processors in a first-fit manner.

B. Adaptive Task Allocation (ATA)

The DTA is designing for the systems of deterministic power supply. However, for the systems of intermittent power supply, it is imperative to tackle the uncertainty in energy sources. In this work, based on the aforementioned DTA, we handle the uncertainty by adapting the execution of imprecise computation tasks to the energy availability. Specifically, we introduce a variable α to denote the ratio of the number of executed optional cycles to the maximum optional cycles of a task, which is named as task optional execution factor and falls within the range of $[0, 1]$. Using the α, the total execution cycles l_i of task τ_i given in (1) can also be written as

$$l_i = M_i + \alpha \cdot O_i. \quad (10)$$

The key issue in dealing with the energy uncertainty is to derive the relationship between task optional execution factors and energy demand of the system. In other words, the optional execution factor of a task is expected to serve as a coarse adjustment knob to control the degree of match between system energy demand and supply. To this end, we first show that the energy demand of a system increases when the optional execution factor of a task in the given set increases, as described in Theorem 2. The proof of the theorem by construction is omitted due to page limit.

Theorem 2: *Given a task $\tau_i \in \Gamma$, its two different optional execution factors α and α', and the task allocation scheme in DTA, $E_{dem} > E'_{dem}$ holds if $\alpha > \alpha'$, where E_{dem} and E'_{dem} are the energy demand of the generated task schedule corresponding to α and α' of task τ_i, respectively.*

Since the optional execution factor of a task is positively related to system energy demand, we can decide the task optional execution factor for given energy demand using a simple yet effective binary search-based approach. We assume the optional execution factor of every task is the same such that the adaptive task allocation is conducted at a coarse granularity. We then decide the own optional execution cycles of every task in the task scheduling at a fine granularity. To handle the uncertainty in energy availability, we adapt system energy demand to renewable energy supply using the task optional execution factor as a coarse-grain control knob to adjust the system energy consumption. The energy-adaptive task allocation algorithm is described in Alg. 1. It takes as input task set Γ, processor set Θ, a sufficiently small positive number ϵ, and outputs M subsets. It first forecasts the energy E_{sup} available to support system operation, then compares E_{sup} with E_{low} and E_{high} to demarcate system energy state.

E_{low} represents energy consumption of the schedule that is generated by DTA only for mandatory parts of all tasks in set Γ, while E_{high} denotes energy consumption of the schedule that is generated by DTA for both mandatory and complete optional parts of all tasks in Γ. The system is deemed in low energy state if $E_{sup} \leq E_{low}$, and in high energy state if $E_{sup} \geq E_{high}$ holds. The task optional execution factors are reset to 0 for low energy state and set to 1 for high energy state.

The system is deemed to be in medium energy state when it is neither in high energy state nor in low energy state. A binary search-based approach is used to derive a common task optional execution factor α for all tasks. Based on α, Alg. 1 calculates task execution cycles using (10), calls DTA to allocate tasks, and computes energy demand E_{dem} using (6). Once energy demand E_{dem} of the task allocation is derived, Alg. 1 iteratively adapts the energy demand to the fluctuating supply of renewable energy E_{sup} according to $|E_{sup} - E_{dem}| > \epsilon$. The binary search-based method is also used in each iteration. The algorithm stops when energy demand of the generated schedule closely matches to the energy supply.

Algorithm 1: Energy-Adaptive Task Allocation

Input: Task set Γ, processor set Θ, sufficiently small positive number ϵ
Output: Task-to-processor assignment $\Gamma_1, \Gamma_2, \cdots, \Gamma_M$

estimate the energy E_{sup} available during the scheduling horizon $[t, t+D]$ by (3);
if $E_{sup} \leq E_{low}$ **then**
  $\alpha = 0$, call **DTA** to allocate tasks; // *In low energy state*
end
else if $E_{sup} \geq E_{high}$ **then**
  $\alpha = 1$, call **DTA** to allocate tasks; // *In high energy state*
end
else // *In medium energy state*
  $min = 0, max = 1, \alpha = (min + max)/2$;
  compute execution cycles for all tasks by (10) based on α, call **DTA** to allocate tasks, and compute energy demand E_{dem} by (6);
  while $|E_{sup} - E_{dem}| > \epsilon$ **do**
    if $E_{dem} > E_{sup}$ **then**
      $max = \alpha$, $\alpha = (min + max)/2$;
    end
    else
      $min = \alpha$, $\alpha = (min + max)/2$;
    end
    update l_i using (10) based on α, allocate tasks using **DTA** based on l_i, and update E_{dem} of the task schedule using (6);
  end
end

V. QoS-Driven Task Scheduling

We focus on a system the available energy of which varies in the range from E_{low} to E_{high}. In such a system, the energy available is enough to finish all mandatory parts but not sufficient to finish both mandatory and optional parts of all tasks. Thus, after tasks are assigned to processors, the goal becomes to maximize system QoS under the energy constraint by selecting tasks whose optional parts should be executed.

A. Task Selection (TS) for QoS Optimization

As given in (7), the QoS of a system is defined as the sum of executed CPU cycles of optional parts of all tasks in the given set. To maximize the system QoS under the energy constraint, we propose to develop a task selection scheme that chooses certain tasks and executes the optional parts of these tasks. Let E_r denote the energy consumed by tasks in subset Γ_r allocated to processor θ_r, then E_r is given by

$$E_r = \sum\nolimits_{\tau_i \in \Gamma_r} [P_{cons}(\tau_i, \theta_r) \cdot \frac{o_i + M_i}{f_r}] \tag{11}$$

where $P_{cons}(\tau_i, \theta_r)$ is the power consumption of task τ_i on processor θ_r. The energy demand or consumption of the system can be expressed as the sum of energy consumed by individual processors. It approximates closely to the system energy supply by a parameter $\epsilon > 0$ since tasks are assigned to individual processors using Alg. 1. Thus, we have

$$E_{sup} \approx E_{dem} = \sum\nolimits_{r=1}^{M} E_r = \sum\nolimits_{r=1}^{M} \sum\nolimits_{\tau_i \in \Gamma_r} [\frac{P_{cons}(\tau_i, \theta_r)}{f_r} \cdot o_i] + \sum\nolimits_{r=1}^{M} \sum\nolimits_{\tau_i \in \Gamma_r} [\frac{P_{cons}(\tau_i, \theta_r)}{f_r} \cdot M_i], \tag{12}$$

where the first item denoted by E_{optl}, is the energy consumption of optional parts while the second item denoted by E_{mand}, is the energy consumption of mandatory parts. When a scheduling decision is made at the start of the scheduling horizon, the estimated energy supply E_{sup} is a fixed value. In addition, the second item E_{mand} shows that the energy consumed by all allocated mandatory parts during the horizon is invariable. As a result, the total energy consumption of optional parts given by the first item E_{optl} is fixed.

The task selection problem for QoS optimization is hence refined into finding the optimal optional cycles for all allocated tasks under the constraint of a fixed energy budget E_{optl}. Let

$$EC_{r,i} = \frac{P_{cons}(\tau_i, \theta_r)}{f_r} \tag{13}$$

be an energy metric that gives the energy consumption of task τ_i on processor θ_r per execution cycle. Since every task is only allowed to bind with one processor, EC_i can be used to instead of $EC_{r,i}$ after task assignment is determined. In addition, we sort all tasks allocated to processors in the ascending order of EC_i for the sake of easy presentation. Thus, the energy consumed by optional parts of all tasks can be written as

$$E_{optl} = EC_1 \cdot o_1 + EC_2 \cdot o_2 + \cdots + EC_N \cdot o_N, \tag{14}$$

where $EC_1 < EC_2 < \cdots < EC_N$ holds. In (14), the metric EC_i and task executed optional cycles $o_i = \alpha \cdot O_i$ are determined once tasks are bound to individual processors. However, the α is derived in the task allocation for maximizing energy efficiency. Now we aim to maximize system QoS in the task scheduling under the energy budget by re-determining the executed optional cycles o_i of tasks using an individual optional execution factor α_i.

As mentioned above, EC_i is determined once tasks are bound to individual processors. Moreover, the LHS of (14) is determined after task allocation. Thus, our goal of optimizing system QoS, which is given in (7) as the sum of executed optional cycles of all tasks, becomes maximizing the sum of o_i for $1 \leq i \leq N$ under the constraint of (14). This can be achieved by developing a task selection scheme that wisely chooses certain tasks and executes their optional cycles. Given these, we present and prove a theorem below, which shows that system QoS is maximized if the tasks having smaller energy metric are selected to execute optional parts. The theorem can be proved by construction. We omit the proof due to page limit.

Theorem 3: *Given the energy budget E_{optl} of optional parts of*

Algorithm 2: Determine Task Optional Execution Cycles

Input: Subsets $\Gamma_1, \Gamma_2, \cdots, \Gamma_M$ produced by Alg. 1
Output: Tasks with updated optional execution cycles

for $r = 1$ to M **do**
  for $i = 1$ to $size(\Gamma_r)$ **do**
    calculate the energy metric $EC_{r,i}$;
  end
  sort the tasks allocated to θ_r in ascending order of $EC_{r,i}$;
end
$E_{optl} = E_{sup} - E_{mand}$;
for $r = 1$ to M **do**
  for $i = 1$ to $size(\Gamma_r)$ **do**
    if $E_{optl} > 0$ **then**
      if $EC_{r,i} \times O_i \leq E_{optl}$ **then**
        $o_i = O_i$, $\alpha_i = 1$;
      end
      else
        $o_i = \frac{E_{optl}}{EC_{r,i}}$, $\alpha_i = \frac{E_{optl}}{EC_{r,i} \times O_i}$;
      end
      $E_{optl} -= EC_{r,i} \times o_i$;
    end
    else
      $o_i = 0$, $\alpha_i = 0$;
    end
    $i \leftarrow i + 1$;
  end
end

task set Γ *as defined in (14), the system QoS as defined in (7) is maximized if the tasks having smaller energy metric* EC *are selected to execute optional parts.*

Based on Theorem 3, we conclude that executing optional parts of tasks with smaller energy metric can improve system QoS. A heuristic is thus designed to determine the execution cycles of optional parts of all the tasks, as shown in Alg. 2. The algorithm first calculates EC, the energy metric that gives the energy consumption per execution cycle of a task and sorts the tasks allocated to individual processors in the ascending order of the metric, then calculates the energy budget E_{optl} for optional parts of all tasks. It derives optional execution cycles for each task based on the analysis that executing optional parts of tasks with smaller energy metric can improve the system QoS. If energy budget E_{optl} for optional parts is large enough, maximum optional cycles of task τ_i can be executed by $o_i = O_i$; otherwise, only $o_i = E_{optl}/EC_{r,i}$ optional cycles can be executed. No optional execution cycles will be executed if the energy budget is exhausted. After task τ_i is examined, the energy budget is updated accordingly and the process moves to the next task. The algorithm outputs all the tasks once their optional execution cycles o_i are updated.

VI. Evaluation

To evaluate the effectiveness of our proposed scheme in reducing energy consumption and improving QoS, we have performed extensive simulation-based studies. To be specific, we first compare the energy consumption of our adaptive task allocation (ATA) with that of HWGA [15] to validate the efficiency of our scheme in reducing energy consumption. We then compare the QoS of our ATA with that of DTA [10] under varying energy supply to validate the efficiency of our scheme in improving QoS at the task allocation stage. We finally compare our task selection (TS) strategy with baseline method Rand and Reve, and benchmarking algorithm critical-task-first (CTF) [8] to validate the efficiency of our scheme in improving QoS at the task scheduling stage. The algorithms used in the

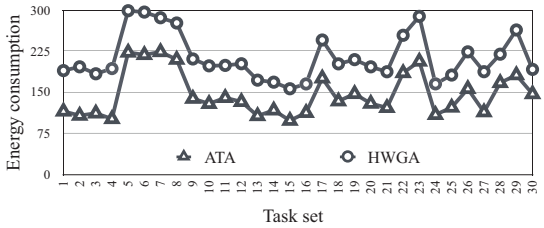

Fig. 2: Energy consumption of 30 task sets using our ATA scheme and benchmarking algorithm HWGA.

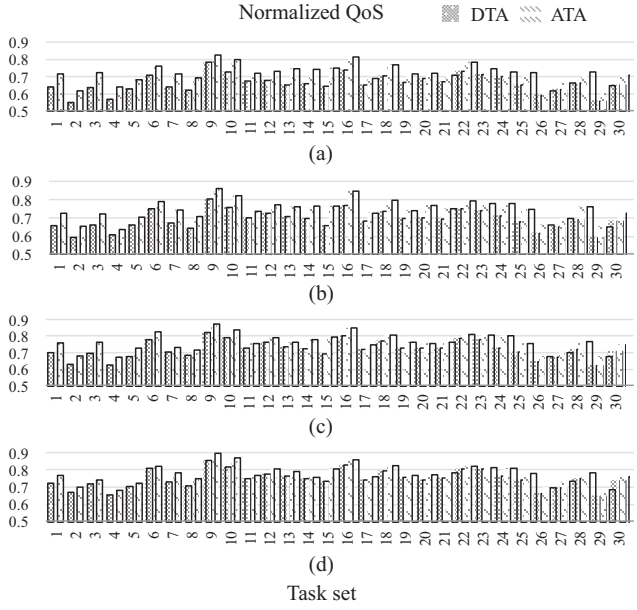

Fig. 3: Normalized QoS of 30 task sets using DTA and ATA under varying energy supply. (a) $E_{sup} = 0.75E_{high}$, (b) $E_{sup} = 0.8E_{high}$, (c) $E_{sup} = 0.85E_{high}$, (d) $E_{sup} = 0.9E_{high}$.

comparison are described as follows. HWGA [15] integrates a worst-fit based partition heuristic with the genetic algorithm to generate a task allocation that reduces energy consumption while satisfying all system constraints. DTA [10] is a task allocation that reduces the dynamic energy consumption by assigning the subset having a larger power dissipation factor to the processor having a smaller power dissipation factor. Rand is a baseline method that randomly selects tasks to complete their optional cycles under the energy budget. Contrary to our TS, Reve chooses tasks having larger metric EC to execute their optional parts. CTF [8] assigns QoS-critical jobs higher priorities to complete their optional cycles, where QoS-critical jobs are defined as tasks with larger maximum optional cycles.

We perform our simulations based on a 2×3 MPSoC system ($M = 6$). Our processor model is built on 65nm technology [16]. The task activity factors μ are uniformly distributed in $[0.4, 1]$, which demonstrates the heterogeneous nature of tasks [13]. The size N of task set Γ is set to 100. The worst case execution cycles (WCEC) of tasks are assumed to be in the range $[4 \times 10^7, 6 \times 10^8]$ [17]. Each task τ_i is instantiated by randomly picking two WCECs from the range, one is for the mandatory part M_i and the other is for the maximum optional part O_i. The common deadline D is assumed to be $1.5\sum_{i=1}^{N} M_i/f_{max}$, where f_{max} is the maximum processor frequency. 30 task sets are constructed in the simulation.

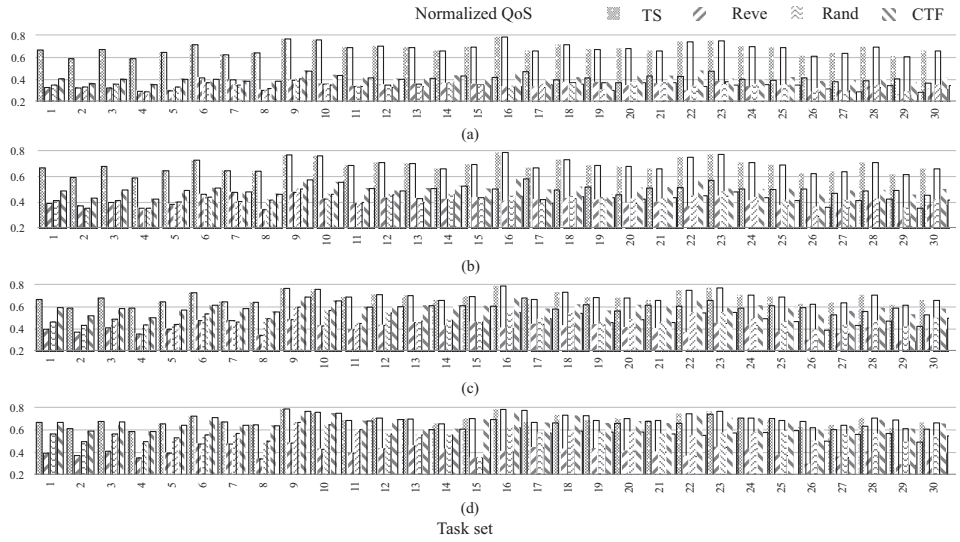

Fig. 4: Normalized QoS of 30 task sets using TS, Reve, Rand, and CTF under varying energy supply. (a) $E_{sup} = 0.75E_{high}$, (b) $E_{sup} = 0.8E_{high}$, (c) $E_{sup} = 0.85E_{high}$, (d) $E_{sup} = 0.9E_{high}$.

The energy consumption of 30 task sets using our ATA scheme and benchmarking algorithm HWGA [15] are shown in Fig. 2. It has been demonstrated in the figure that our ATA scheme consumes less energy (33.7% on average) than that of HWGA. Furthermore, the improvement of energy savings by our scheme over HWGA can be up to 47.5%. For instance, the energy consumed by executing task set 4 using ATA and HWGA are 101.5J and 193.2J, respectively.

Fig. 3 presents the normalized QoS achieved by the system when executing the optional cycles of tasks in 30 sets using the proposed DTA and ATA scheme under varying energy supply. The results given in the figure show that better performance in improving QoS can be achieved by ATA as compared to DTA, especially when the energy supply is low. To be specific, the QoS of ATA is 9.1%, 8.2%, 6.1%, and 4.2% higher than that of DTA on average when $E_{sup} = 0.75, 0.8, 0.85$, and $0.9E_{high}$, respectively. The highest improvement achieved by ATA over DTA is 16.3%. For example, in the case of $E_{sup} = 0.75E_{high}$ and task set 15, the normalized QoS achieved by ATA and DTA are 0.749 and 0.644, respectively.

Fig. 4 shows the normalized QoS achieved by the system when executing the optional cycles of tasks in 30 sets using the proposed TS, and benchmarking methods Reve, Rand, and CTF [8] under varying energy supply. The results in Fig. 4 show that our TS has the best performance in improving QoS among the four methods. Specifically, when $E_{sup} = 0.75E_{high}$, the QoS of TS is 100.3%, 97.1%, and 65.7% higher than that of Reve, Rand, and CTF [8] on average, respectively; when $E_{sup} = 0.8E_{high}$, the QoS of TS is 71.7%, 61.7%, and 36.3% higher than that of Reve, Rand, and CTF [8] on average, respectively; when $E_{sup} = 0.85E_{high}$, the QoS of TS is 68.8%, 40.3%, and 15.2% higher than that of Reve, Rand, and CTF [8] on average, respectively; when $E_{sup} = 0.9E_{high}$, the QoS of TS is 71.8%, 24.3%, and 3% higher than that of Reve, Rand, and CTF [8] on average, respectively. Moreover, the highest improvement achieved by TS is up to 147.8%. For instance, in the case of $E_{sup} = 0.75E_{high}$ and task set 22, the normalized QoS of TS and Reve are 0.741 and 0.299, respectively.

VII. Conclusion

In this paper, we proposed to utilize the characteristic of imprecise computation to intelligently handle the uncertainty in renewable energy powered real-time embedded systems. We designed an energy-adaptive task allocation scheme and a QoS-driven task scheduling scheme that can not only reduce the energy consumption but also improve the QoS of the system. We conducted extensive simulations to validate the effectiveness of our schemes. Our algorithms were shown to reduce energy consumption by up to 47.5% and improve system QoS by up to 147.8% compared to existing schemes.

References

[1] J. Liu et al. Algorithms for scheduling imprecise computations, *Springer US*, 1991.
[2] M. Severini et al. Energy-aware lazy scheduling algorithm for energy-harvesting sensor nodes, *NCA*, vol. 23, no. 7-8, pp.1899-1908, 2013.
[3] Y. Abdeddaim, et al. The optimality of PFPasap algorithm for fixed-priority energy-harvesting real-time systems, *ECRTS*, pp. 47-56, 2013.
[4] X. Lin et al. A framework of concurrent task scheduling and dynamic voltage and frequency scaling in real-time embedded systems with energy harvesting, *ISLPED*, pp. 70-75, 2013.
[5] G. Stavrinides et al. Scheduling real-time DAGs in heterogeneous clusters by combining imprecise computations and bin packing techniques for the exploitation of schedule holes, *FGCS*, vol. 28, no. 7, pp. 977-988, 2012.
[6] L. Cortes et al. Quasi-static assignment of voltages and optional cycles in imprecise-computation systems with energy considerations, *IEEE TVLSI*, vol. 14, no. 10, pp. 1117-1129, 2006.
[7] H. Yu et al. Dynamic scheduling of imprecise-computation tasks in maximizing QoS under energy constraints for embedded systems, *ASPDAC*, pp. 452-455, 2008.
[8] H. Kooti et al. Energy budget management for energy harvesting embedded systems, *RTCSA*, pp. 320-329, 2012.
[9] M. Awan et al. Energy aware partitioning of tasks onto a heterogeneous multi-core platform, *RTAS*, pp. 205-214, 2013.
[10] J. Zhou et al. Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems, *IEEE TCAD*, vol. 35, no. 8, pp. 1269-1282, 2016.
[11] J. Zhou et al. Stochastic thermal-aware real-time task scheduling with considerations of soft errors, *Elsevier JSS*, vol. 102, pp. 123-133, 2015.
[12] Y. Liu, et al. Thermal vs energy optimization for DVFS-enabled processors in embedded systems, *ISQED*, pp. 204-209, 2007.
[13] H. Huang et al. Throughput maximization for periodic real-time systems under the maximal temperature constraint, *ACM TECS*, vol. 13, no. 2s, 2014.
[14] B. Zhao et al. On maximizing reliability of real-time embedded applications under hard energy constraint, *IEEE TII*, vol. 6, no. 3, pp. 316-328, 2010.
[15] S. Saha et al. Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems, *RTCSA*, pp. 41-50, 2012.
[16] G. Quan et al. Feasibility analysis for temperature constraint hard real-time periodic tasks, *IEEE TII*, vol. 6, no. 3, pp. 329-339, 2010.
[17] R. Jayaseelan et al. Temperature aware task sequencing and voltage scaling, *ICCAD*, pp. 618-623, 2008.