Cost-Constrained QoS Optimization for Approximate Computation Real-Time Tasks in Heterogeneous MPSoCs

Tongquan Wei, *Member, IEEE*, Junlong Zhou, *Member, IEEE*, Kun Cao, Peijin Cong, Mingsong Chen, *Member, IEEE*, Gongxuan Zhang, *Senior Member, IEEE*, Xiaobo Sharon Hu, *Fellow, IEEE*, and Jianming Yan

***Abstract*—Internet of Things devices, such as video-based detectors or road side units are being deployed in emerging applications like sustainable and intelligent transportation systems. Oftentimes, stringent operation and energy cost constraints are exerted on this type of applications, necessitating a hybrid supply of renewable and grid energy. The key issue of a cost-constrained hybrid of renewable and grid power is its uncertainty in energy availability. The characteristic of approximate 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 an energy-adaptive task allocation scheme that optimally assigns real-time approximate-computation tasks to individual processors and subsequently enables a matching of the cost-constrained hybrid supply of energy with the energy demand of the resultant task schedule. We then present a quality of service (QoS)-driven task scheduling scheme that determines the optional execution cycles of tasks on individual processors for optimization of system QoS. A dynamic task scheduling scheme is also designed to adapt at runtime the task execution to the varying amount of the available energy. Simulation results show that our schemes can reduce system energy consumption by up to 29% and improve system QoS by up to 108% as compared to benchmarking algorithms.**

***Index Terms*—Approximate computation, hybrid energy systems, quality of service (QoS) optimization, real-time multiprocessor system-on-chip (MPSoC).**

Manuscript received April 23, 2017; revised August 10, 2017; accepted October 25, 2017. Date of publication November 13, 2017; date of current version August 20, 2018. This work was supported in part by the Shanghai Municipal Natural Science Foundation under Grant 16ZR1409000, and in part by the Natural Science Foundation of China under Grant 61672230. This paper was recommended by Associate Editor Z. Shao. *(Corresponding author: Mingsong Chen.)*

T. Wei, K. Cao, P. Cong, and J. Yan are with the Department of Computer Science and Technology, East China Normal University, Shanghai 200062, China.

J. Zhou and G. Zhang are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.

M. Chen is with the Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China (e-mail: mschen@sei.ecnu.edu.cn).

X. S. Hu is with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46656, USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2017.2772896

I. Introduction

AS A GLOBAL infrastructure for information society, Internet of Things (IoT) has enabled various emerging applications [1], [2]. Sustainable and intelligent transportation is one of such applications, where video-based detection systems or road side unit (RSU)-based vehicular ad-hoc networks are utilized for better traffic planning and management. IoT devices, such as video detectors and RSUs consume power of up to 10W [3], resulting in an unaffordable energy cost and thus hindering the deployment of intelligent transportation systems in metropolitan cities. This situation can potentially be mitigated from two approaches. One is for these systems to scavenge energy from renewable generations as complement to the energy drained from power grid. The other one is to perform approximate computing for applications like image or video-based object identification such that results of best quality is obtained by consuming a given amount of energy. In this paper, we are interested in designing such systems that are powered by hybrid energy and supposed to finish under budget a mission with best quality.

The key issue of a hybrid energy system under budget of the energy cost is its uncertainty. This uncertainty stems from the intermittence nature of renewable generation and time-varying pricing of electric grid. A system powered by such hybrid energy may fail to execute a task to completion by deadline due to lack of energy, resulting in a timing fault and degraded system performance. Approximate computation approach [4] can minimize the possibility that a task misses its deadline due to the nonstationarity of powering in hybrid energy systems. In the approximate computation approach, a task is decomposed into a mandatory part followed by an optional part. The mandatory part must execute to completion to produce an acceptable result while the optional part refines the generated result. Based on the observation that a timely approximate result is preferable to a precise result too late [4], the approximate computation technique can be used to avoid timing faults. 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. In this paper, we explore the design of a hybrid energy real-time system using the approximate computation technique.

The studied hybrid energy real-time system attempts to combine renewable generation and electric grid with energy storage to deliver a cost-constrained power supply. In fact, real-time embedded systems powered by conventional energy and renewable energy have been studied separately in the recent past. With regard to real-time embedded platforms that are either powered by batteries or directly connected to power grid, their energy management in the last two decades has concentrated on minimizing energy consumption for a longer lifetime and clear financial advantages [5]. On the other hand, for real-time embedded platforms that are powered by renewable generations, the focus of research is mainly on improving the efficiency to take advantage of renewable energy, reducing the capacity of energy storage, and minimizing the deadline miss rate of real-time tasks. For instance, Severini *et al.* [6] presented an improved version of lazy scheduling algorithm [7] for energy harvesting systems. The improved solution can foresee at runtime the task energy starving, hence allows obtaining a more conservative and efficient management of energy with respect to the original solution. Chetto and Queudet [8] showed that when the incoming ambient energy or task arrivals have a pure stochastic nature, the EDF scheduling algorithm remains an attractive scheduler owing to easy implementation and no need for estimating the stored or harvested energy. Abdeddaïm *et al.* [9] designed an optimal fixed-priority solution to the real-time scheduling problem that handles both energy and timing constraints for energy harvesting systems.

All the above works on renewable powered systems focus on improving energy efficiency and schedule timeliness for energy harvesting embedded systems, however, the concept of approximate execution is not considered. Stavrinides and Karatza [10] combined the approximate computation and bin packing strategy to evaluate the impact of input error on the performance of a heterogeneous distributed real-time system, however, the energy design constraint is not taken into account.

It is natural for approximate computing systems to maximize their QoS given a certain cost or energy budget. Cortes *et al.* [11] designed an approach to maximize rewards for real-time approximate computation systems. The voltage at which each task runs and the number of optional cycles are determined under the timing and energy constraint. Similarly, Yu *et al.* [12] presented a runtime scheduling algorithm for approximate computation tasks. The algorithm maximizes system QoS under energy constraints by optimally redistributing slack generated at runtime. Since these approaches assume a constant constraint of energy, they are not well suited for cost-constrained hybrid energy systems, where the available energy varies due to intermittence of renewable and dynamic pricing of grid power. Considering the uncertainty in availability of energy in embedded harvesting systems, Kooti *et al.* [13] designed an energy management technique to maximize system QoS. However, the scheme specifically targets applications in which QoS constraints allow drop out of some of real-time tasks, which does not comply with the stringent timing requirements of general real-time systems. In addition, the scheme is based on a platform of single processor only powered by harvesting energy, thus are not applicable for the state-of-the-art hybrid energy multicore processor platform.

In this paper, we propose an approximate computation-based task allocation and scheduling scheme for a heterogeneous multiprocessor system-on-chip (MPSoC) real-time system that is powered by a hybrid of energy harvested from environments and drained from power grid. The major contributions are summarized as follows.

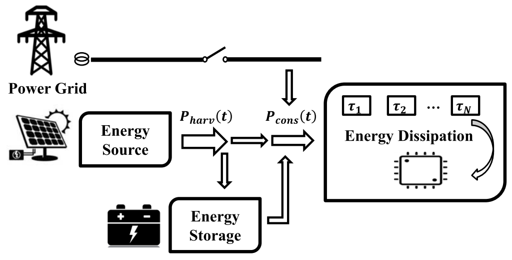


Fig. 1. Diagram of the system architecture.

1) We present a static energy-efficient task allocation scheme that adaptively assigns approximate computation tasks to individual processors considering the uncertainty in the hybrid of the renewable and grid energy. The energy-adaptive task allocation (ATA) scheme can effectively enable the matching of the hybrid energy supply with the energy consumption of the resultant task schedule.
2) We propose a static scheduling scheme for tasks allocated to individual processors. The proposed scheduling scheme determines the number of optional cycles of each task to be executed such that the system QoS is maximized under budget of the cost including energy and battery aging. The static task schedule is also adapted at runtime to the fluctuating amount of available energy.
3) We conduct extensive simulation experiments to verify the effectiveness of the proposed schemes in improving system energy efficiency and QoS. Simulation results have demonstrated that the proposed schemes have better performance as compared to benchmarking schemes.

The rest of this paper is organized as follows. Section II introduces the system architecture and models, and Section III defines the problem and gives the overall framework of the proposed scheme. Section IV presents the proposed uncertainty-aware and cost-constrained task allocation, and Section V describes the proposed QoS-driven task scheduling. The effectiveness of the proposed approach is verified in Section VI and concluding remarks are given in Section VII.

II. System Architecture and Models

The system consists of four major parts: 1) the energy source module; 2) storage module; 3) dissipation module; and 4) power grid. As shown in Fig. 1, the energy source module automatically scavenges renewable energy from environments at the power of $P_{\text{harv}}(t)$, and converts the renewable generation into electrical energy. The energy storage module, which is typically in the form of a battery, serves as a buffer against the uncertainty in harvested energy. When the harvested energy is more than the energy consumed by the dissipation module, the extra energy is stored in the storage module. A heterogeneous MPSoC is considered as the energy dissipation module, which drains energy at the rate of $P_{\text{cons}}(t)$ from the energy source, storage module, and/or power grid that provide a power supply of $P_{\text{sup}}(t)$. We are interested in a system that is designed under a cost budget. We assume that the harvested energy is free, thus, the amount of energy drained from the battery and/or grid depends upon the cost of battery aging and the dynamic price of grid electricity.

A. Processor and Task Model

The MPSoC system consists of M processors, denoted by $\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_M\}$, where each processor Θ_m $(1 \leq m \leq M)$ is a typical DVFS-enabled processor that can operate with a set of discrete supply voltage and frequency pairs $(v_{m,r}, f_{m,r})$ $(1 \leq r \leq x_m)$, where $v_{m,1} < \cdots < v_{m,r} < \cdots < v_{m,x_m}$, $f_{m,1} < \cdots < f_{m,r} < \cdots < f_{m,x_m}$, and x_m is the voltage/frequency level of Θ_m. Real-time tasks are supposed to schedule and execute on the system. Consider a task set Γ consisting of N independent real-time tasks $\{\tau_1, \tau_2, \ldots, \tau_N\}$. The task set Γ is a frame-based task set, in which all tasks share a common deadline d that is also the frame. Different tasks exhibit different power consumptions on the same processor, even executing at the same operating speed and temperature. This is due to the fact that power consumptions of tasks strongly rely on circuit activities and usage patterns of different functional units [14]. Thus, the activity factor of a task, denoted by μ (ranging in (0, 1]), is introduced to capture how intensively functional units are being utilized by the task [15].

We consider approximate computation tasks in this paper. Each 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 [16]. The mandatory part must execute to completion before the deadline and generate an acceptable result, while the optional part refines and improves the result. The characteristic of an approximate computation modeled task τ_i is therefore described by a quadruple $\tau_i : \{\mu_i, M_i, O_i, d\}$, where μ_i is the activity factor, and d is the common deadline. M_i is the 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 τ_i, which is denoted as o_i and holds for $0 \leq o_i \leq O_i$. Then, the actual length l_i of τ_i, measured by the total execution cycles, can be expressed as

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

B. Battery SoH Degradation Model

A practical and widely used state-of-health (SoH) degradation model of Li-ion batteries [17]–[19] is adopted to estimate the SoH degradation for cycled charging and discharging of a Li-ion battery cell. Before presenting the estimation of SoH degradation, we formally define the state-of-charge (SoC) of a battery at first, that is

$$\text{SoC} = \frac{C_{\text{bat}}}{C_{\text{full}}} \times 100\% \tag{2}$$

where C_{bat} is the amount of charge stored in the battery and C_{full} is the battery full charge capacity. The amount of SoH degradation, denoted by D_{SoH}, is defined as

$$D_{\text{SoH}} = \frac{C_{\text{full}}^{\text{nom}} - C_{\text{full}}}{C_{\text{full}}^{\text{nom}}} \times 100\% \tag{3}$$

where $C_{\text{full}}^{\text{nom}}$ is nominal value of C_{full} for a fresh new battery.

The adopted SoH degradation model estimates the battery SoH degradation in a cycled charging/discharging pattern, where a (charging/discharging) cycle is defined as a charging process of the battery cell from SoC_{low} to SoC_{high} followed by a discharging process from SoC_{high} to SoC_{low}. Thus, the average SoC and SoC swing in a cycle are calculated as

$$\text{SoC}_{\text{avg}} = \big(\text{SoC}_{\text{low}} + \text{SoC}_{\text{high}}\big)/2 \tag{4}$$

$$\text{SoC}_{\text{swing}} = \text{SoC}_{\text{high}} - \text{SoC}_{\text{low}}. \tag{5}$$

The battery SoH degradation $D_{\text{SoH,cycle}}$ during one cycle, accounting for SoC_{avg} and $\text{SoC}_{\text{swing}}$, is given by

$$D_1 = K_{co} \cdot e^{(\text{SoC}_{\text{swing}}-1) \cdot \frac{T_{\text{ref}}}{K_{ex} \cdot T_B}} + 0.2\frac{\eta}{\eta_{\text{life}}} \tag{6}$$

$$D_2 = D_1 \cdot e^{4K_{\text{SoC}} \cdot (\text{SoC}_{\text{avg}}-0.5)} \cdot (1 - D_{\text{SoH}}) \tag{7}$$

$$D_{\text{SoH,cycle}} = D_2 \cdot e^{K_T \cdot (T_B - T_{\text{ref}}) \cdot \frac{T_{\text{ref}}}{T_B}} \tag{8}$$

where K_{co}, K_{ex}, K_{SoC}, and K_T are battery specific parameters. T_B and T_{ref} are battery's operation temperature and reference temperature, respectively. η is the duration of this charging/discharging cycle and η_{life} is the calendar life of this battery. The total SoH degradation (in reference to a fresh battery) after $\mathcal{W}$ charging and discharging cycles is

$$D_{\text{SoH}} = \sum_{w=1}^{\mathcal{W}} D_{\text{SoH,cycle}}(w) \tag{9}$$

where $D_{\text{SoH,cycle}}(w)$ is the SoH degradation in the wth cycle.

However, the above SoH degradation model can only be applied to the fixed charging/discharging pattern, that is, the battery experiences the cycles with the same $\text{SoC}_{\text{swing}}$ and SoC_{avg}. In reality, a battery may not follow this pattern. Hence, a cycle-decoupling method was proposed [18] to build an improved SoH degradation model of battery for charging/discharging cycles with arbitrary patterns. The SoH degradation in an arbitrary cycle and the total degradation can thus be derived using (8) and (9), respectively.

C. 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 $P_{\text{harv}}(t)$ be the harvesting power and $E_{\text{harv}}(t_1, t_2)$ be the energy scavenged from environments during time interval $[t_1, t_2]$, then $E_{\text{harv}}(t_1, t_2)$ is calculated as

$$E_{\text{harv}}(t_1, t_2) = \int_{t_1}^{t_2} P_{\text{harv}}(t)dt. \tag{10}$$

As illustrated in Fig. 1, the energy source module, storage module, and power grid can provide the energy to the dissipation module. Let $E_{\text{sup}}(t_1, t_2)$ represent the system available supply energy during time interval $[t_1, t_2]$, and $E_{\text{bat}}(t_1, t_2)$ and $E_{\text{grid}}(t_1, t_2)$ indicate the energy drained from the battery and power grid during $[t_1, t_2]$, respectively, then we have

$$E_{\text{sup}}(t_1, t_2) = E_{\text{harv}}(t_1, t_2) + E_{\text{bat}}(t_1, t_2) + E_{\text{grid}}(t_1, t_2). \tag{11}$$

We then describe the model for energy demand. The total energy consumption of an IoT device depends on its processors, memory, disks, cooling system, and wireless components. Although, there are many research work [20]–[22] focusing on reducing system overall energy consumption, we only consider processor energy consumption in the energy dissipation model since processor is the dominant source of overall energy consumption of some systems like IoT security authentication system [23]–[25]. The power consumption of a processor can be modeled as the sum of static/leakage power consumption P_{sta} and dynamic power consumption P_{dyn}. The static power P_{sta} is temperature dependent and consumed by the leakage current required to maintain basic state of circuits. The leakage current changes super linearly with temperature. The dynamic power P_{dyn} is related to processor switching activity and can be estimated by a function of supply voltage v and frequency f, that is, $P_{\text{dyn}} \propto v^2 f$. Thus, the overall power consumption of processor Θ_m when executing task τ_i at the supply voltage/frequency $(v_{m,r}, f_{m,r})$ is formulated as

$$P_{\text{cons}}(\tau_i, \Theta_m) = \omega_1 \cdot v_{m,r} + \omega_2 \cdot v_{m,r} \cdot T_m(t) + C_{\text{eff},m} \cdot \mu_i \cdot v_{m,r}^2 \cdot f_{m,r} \tag{12}$$

where ω_1 and ω_2 are both non-negative architecture-dependent constants of Θ_m, $T_m(t)$ is the operating temperature of Θ_m at time instance t, $C_{\text{eff},m}$ is the effective switching capacitance of Θ_m, 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 (12), the total energy consumed (or demanded) by the system during the scheduling horizon $\mathcal{H}$, denoted by E_{dem}, is calculated as

$$E_{\text{dem}} = \sum_{m=1}^{M} \left(\omega_1 \cdot v_{m,r} \cdot \mathcal{H} + \int_0^{\mathcal{H}} \omega_2 \cdot v_{m,r} \cdot T_m(t) dt \right) + \sum_{m=1}^{M} \sum_{\tau_i \in \Gamma_m} \left(C_{\text{eff},m} \cdot \mu_i \cdot v_{m,r}^2 \cdot l_i \cdot \frac{\mathcal{H}}{d} \right) \tag{13}$$

where Γ_m is the subset of tasks allocated to processor Θ_m. Oftentimes, $\mathcal{H}$ is a multiple of d.

III. Problem Definition and Overall Framework

Our goal is to reduce the impact of uncertainties in the energy scavenged from ambient environments, and improve the efficiency of renewable energy and QoS of the system under a certain cost budget $\text{Cost}_{bdg,\mathcal{H}}$. The following sections describe the system cost and QoS function, formulate the QoS optimization problem, and overview the proposed solution.

A. System Cost Function

We adopt a slotted time model that deals with all the system decisions and constraints in discrete time intervals of equal length. More specifically, the whole scheduling horizon $\mathcal{H}$ is divided into $\mathcal{W}$ time slots with equal and constant length of Δt. We assume that the power grid has a dynamic pricing function, and the price of one unit of energy (in kWh) during the wth time slot is denoted by $PX_{\text{grid}}[w]$. In our formulation, solar energy is considered free for simplicity, though its cost can be normalized and incorporated in the formulation. Since solar energy is free, the total cost function $\text{Cost}_{\text{tot},\mathcal{H}}$ during $\mathcal{H}$ is composed of two parts. One is the cost $\text{Cost}_{\text{aging},\mathcal{H}}$ associated with battery aging and the other is the energy cost $\text{Cost}_{\text{grid},\mathcal{H}}$ due to charges from power grid. That is

$$\text{Cost}_{\text{tot},\mathcal{H}} = \text{Cost}_{\text{aging},\mathcal{H}} + \text{Cost}_{\text{grid},\mathcal{H}}. \tag{14}$$

Let Cost_{bat} be the cost to purchase a fresh new battery, $D_{\text{SoH},\mathcal{H}}$ be the amount of SoH degradation during $\mathcal{H}$, and $D_{\text{SoH,end}}$ be the amount of SoH degradation indicating the end-of-life of a battery, the cost of battery aging during $\mathcal{H}$ is

$$\text{Cost}_{\text{aging},\mathcal{H}} = \text{Cost}_{\text{bat}} \frac{D_{\text{SoH},\mathcal{H}}}{D_{\text{SoH,end}}}. \tag{15}$$

Oftentimes, Cost_{bat}, the cost to purchase and replace a battery is fixed, and the $D_{\text{SoH,end}}$ indicating the end-of-life of a battery takes value of 70% [19]. Since Cost_{bat} and $D_{\text{SoH,end}}$ are assumed constant, it can be seen from (15) that the cost of battery aging can be minimized by optimizing $D_{\text{SoH},\mathcal{H}}$, the amount of SoH degradation during the scheduling horizon $\mathcal{H}$.

For the wth time slot, let $P_{\text{sup}}[w]$ be the power supplied to the energy dissipation module, $P_{\text{harv}}[w]$ be the harvesting power of energy source module, and $P_{\text{bat}}[w]$ be the power drained from the battery. We assume that the grid electricity price $PX_{\text{grid}}[w]$ and the battery aging cost in the wth time slot is fixed. Then we define a binary variable β that is set to 1 when the electricity cost in a time slot is greater than the battery aging cost, and 0 otherwise. When the incurred cost of grid electricity is different from the battery aging cost in a time slot, the energy source of lower cost is selected to power the MPSoC. Given these and based on the principle of energy conservation illustrated in Fig. 1, the grid power can then be derived as $P_{\text{sup}}[w] - P_{\text{harv}}[w] - \beta \cdot P_{\text{bat}}[w]$, and its cost is

$$\text{Cost}_{\text{grid},\mathcal{H}} = \sum_{w=1}^{\mathcal{W}} PX_{\text{grid}}[w] \times \left(P_{\text{sup}}[w] - P_{\text{harv}}[w] - \beta \cdot P_{\text{bat}}[w]\right)\Delta t. \tag{16}$$

From the viewpoint of energy dissipation module, $P_{\text{sup}}[w]$ can also be written in the form of $P_{\text{dem}}[w] + \Delta P[w]$, that is, $P_{\text{sup}}[w] = P_{\text{dem}}[w] + \Delta P[w]$, where $\Delta P[w]$ is the difference between the power supply and demand of energy dissipation module. Thus, (16) can be expressed as

$$\text{Cost}_{\text{grid},\mathcal{H}} = \sum_{w=1}^{\mathcal{W}} PX_{\text{grid}}[w]\left(P_{\text{dem}}[w] + \Delta P[w] - P_{\text{harv}}[w] - \beta \cdot P_{\text{bat}}[w]\right)\Delta t. \tag{17}$$

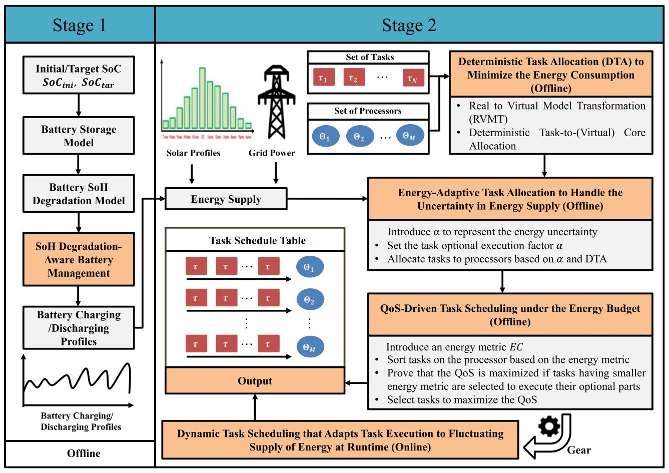
The solar harvesting profile $P_{\text{harv}}[w]$ can be estimated using historical data. The power drain of battery $P_{\text{bat}}[w]$ can be derived by an SoH degradation-aware battery management policy [18] that minimizes the amount of SoH degradation $D_{\text{SoH},\mathcal{H}}$ and hence $\text{Cost}_{\text{aging},\mathcal{H}}$. As the cost of battery aging $\text{Cost}_{\text{aging},\mathcal{H}}$ is derived and the cost budget $\text{Cost}_{bdg,\mathcal{H}}$ is given, the cost of grid power $\text{Cost}_{\text{grid},\mathcal{H}}$ is then determined. In (17), since $\text{Cost}_{\text{grid},\mathcal{H}}$, $P_{\text{harv}}[w]$, $P_{\text{bat}}[w]$, $PX_{\text{grid}}[w]$, Δt are all known, $P_{\text{dem}}[w] + \Delta P[w]$ is decided. When $P_{\text{dem}}[w]$ is optimized and $\Delta P[w] = 0$, the energy consumed per execution cycle is minimized and the energy supply matches with the energy demand. In other words, more energy can be used to execute task optional cycles for QoS improvement.

B. System QoS Function

It has been shown that the quality of a task can be represented as a linear or concave function of optional cycles of the task [4]. The more cycles the optional part of the task executes, the higher QoS the task generates. Thus, 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. It is denoted by Q and is expressed as

$$Q = \sum_{i=1}^{N} o_i. \tag{18}$$

Clearly, the system QoS is a function of executed optional cycles of real-time tasks.



Stage 1
Stage 2
Initial/Target SoC
SoC_{ini}, SoC_{tar}
Battery Storage Model
Battery SoH Degradation Model
SoH Degradation-Aware Battery Management
Battery Charging /Discharging Profiles
Battery Charging/ Discharging Profiles
Offline
Solar Profiles
Grid Power
Set of Tasks
τ_1 τ_2 … τ_N
Set of Processors
Θ_1 Θ_2 … Θ_M
Deterministic Task Allocation (DTA) to Minimize the Energy Consumption (Offline)
Real to Virtual Model Transformation (RVMT)
Deterministic Task-to-(Virtual) Core Allocation
Energy Supply
Energy-Adaptive Task Allocation to Handle the Uncertainty in Energy Supply (Offline)
Introduce α to represent the energy uncertainty
Set the task optional execution factor α
Allocate tasks to processors based on α and DTA
Task Schedule Table
Output
QoS-Driven Task Scheduling under the Energy Budget (Offline)
Introduce an energy metric EC
Sort tasks on the processor based on the energy metric
Prove that the QoS is maximized if tasks having smaller energy metric are selected to execute their optional parts
Select tasks to maximize the QoS
Dynamic Task Scheduling that Adapts Task Execution to Fluctuating Supply of Energy at Runtime (Online)
Gear

Fig. 2. Main design flow of the proposed scheme.

C. Cost-Constrained QoS Optimization Problem

Given a set of approximate-computation real-time tasks and a set of heterogeneous processors powered by renewable and grid energy, design a task-to-processor allocation that adapts to the uncertainty of system available energy, and a task scheduling scheme that arranges assigned tasks on individual processors to maximize the system QoS under the constraint of a cost budget. We assume that the system energy demand cannot exceed the energy supply, and the mandatory parts of all the tasks must be finished before the common deadline d. The problem is formulated into the below form

$$\text{Maximize:} \quad Q = \sum_{i=1}^{N} o_i$$

$$\text{Subject to:} \quad 0 \le o_i \le O_i \tag{19}$$

$$E_{\text{dem}} \le E_{\text{sup}} \tag{20}$$

$$\sum_{\tau_i \in \Gamma_m} M_i / f(\tau_i) \le d \tag{21}$$

$$\text{Cost}_{\text{tot},\mathcal{H}} \le \text{Cost}_{bdg,\mathcal{H}}. \tag{22}$$

D. Overview of the Proposed Two-Stage Scheme

The key issue of the studied system is that uncertainties in the energy scavenged from ambient environments result in low energy utilization and/or high deadline miss rate. We address the problem by designing approximate computation-based task allocation and scheduling algorithms that enhance the energy efficiency, ensure system timing constraints, and maximize the system QoS under the constraint of a cost budget. The design flow of the proposed scheme is illustrated in Fig. 2. Given cost budget of a system, the uncertainties in energy supply of energy dissipation modules stem from the intermittence of renewable energy and dynamic pricing of grid energy. We tackle the uncertainties in stage 1 of the scheme by adopting an SoH degradation-aware battery management policy [18] that minimizes the cost of battery aging, and conducting an energy-ATA that minimizes the system energy demand and adapts the energy demand to the availability of energy supply including the grid and renewable energy.

In stage 1 of the proposed scheme, we first adopt an SoH degradation-aware battery management policy [18] to reduce the cost of battery aging. Based on the battery storage and SoH degradation model adopted, the SoH degradation-aware battery management policy [18] takes the battery initial SoC and target SoC as input and determines the battery charging/discharging current profile using convex optimization techniques. The generated charging/discharging current profile is a near-optimal charge management policy that can minimize the battery SoH degradation, thus extend the cycle life of battery. For a given cost budget formulated in (14), it is preferable to utilize free renewable energy, or battery storage and/or grid power, whichever incurs lower cost during the scheduling horizon $\mathcal{H}$. The cost of battery aging and grid power are given in (15) and (17), respectively.

We then design an energy ATA algorithm in stage 2 of the proposed scheme. The algorithm first introduces and initializes a variable α, named the task optional execution factor, to represent the uncertainty in energy supply. It allocates tasks to processors using a deterministic task allocation (DTA) scheme that consists of Real_to_Virtual_Model_Transformation (RVMT) and Deterministic Task-to-(Virtual) Core Allocation [26] based on the α. Then, it adapts system energy demand to renewable and grid supply of energy by using the task optional execution factor as a control knob to maximize the degree of match between system energy demand and supply. Through the DTA and task optional execution factor tuning, the system energy consumption is minimized, the energy supply is fully utilized, and the cost of grid energy is reduced.

The stage 2 of the proposed scheme aims at improving the QoS of the system after SoH degradation-aware battery management policy minimizes the cost of battery aging and the energy ATA scheme reduces the cost of grid energy. The QoS-driven task scheduling algorithm proposed in this stage first introduces an energy metric that indicates the importance of every task on the processor in terms of improving system QoS, then proves that the system QoS is maximized if tasks having smaller energy metric are selected to execute their optional parts. It finally develops a task selection scheme that chooses the tasks with smaller metric under the constraint of cost budget.

The above QoS-driven task scheduling is designed for the offline scenario under the assumption of a fixed energy supply. However, the actual energy harvested from environments may not be the same as the energy estimated by prediction techniques due to the fluctuating nature of renewable generation. In addition, the cost-constrained energy drained from the battery and power grid may vary due to different initial SoC of the battery and time varying price of grid power. Therefore, an online QoS-driven task scheduling is designed to adapt the offline schedules to fluctuating energy supply at runtime.

IV. Task Allocation Under Cost Constraint

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

A. DTA to Minimize Energy Consumption

The concerned system consists of heterogeneous multiprocessors, each of which is DVFS-enabled and supports a set of discrete supply voltage and frequency pairs. Hence, operating frequencies of tasks need to be determined during task allocation. To this end, we propose an RVMT method that converts the processor model with multiple voltage and frequency levels to a model of multiple virtual cores, each of which has a fixed supply voltage and frequency level. This method can effectively decrease one dimension of optimization for energy consumption by reducing task-to-(real) processor allocation and frequency selection problem to task-to-(virtual) core allocation problem. Below, we first show the proposed RVMT processor model transformation, analyze the energy optimality of allocating tasks to virtual cores and present a theorem on optimum task allocation, then develop a task-to-(virtual) core allocation heuristic based on the theorem.

1) Real_to_Virtual Model Transformation: The system energy consumption during the scheduling horizon $\mathcal{H}$ given in (13) can be rewritten as

$$E_{\text{dem}} = \sum_{m=1}^{M}\left(\omega_1 \cdot v_{m,r} \cdot \mathcal{H} + \int_0^{\mathcal{H}} \omega_2 \cdot v_{m,r} \cdot T_m(t)dt\right) + \frac{\mathcal{H}}{d}\sum_{m=1}^{M}\sum_{r=1}^{x_m}\left(C_{\text{eff},m} \cdot v_{m,r}^2 \sum_{\tau_i \in \Gamma_{m,r}} \mu_i \cdot l_i\right) \tag{23}$$

where $\Gamma_{m,r}$ is the subset of tasks allocated to processor Θ_m and operated at frequency $f_{m,r}$. The two items of (23) are the static and dynamic energy consumption, respectively. The static energy consumption depends on temperature while the dynamic energy consumption depends on task allocation. Thus, both thermal-aware task scheduling and energy-efficient task allocation are helpful to minimize the system overall energy consumption. Since thermal-aware task scheduling for minimizing the static energy consumption has been well studied in [26], we focus on energy-efficient task allocation to minimize the dynamic energy consumption in this paper.

Let E_{dyn} represent the dynamic energy consumption, which can be formulated into the product of two vectors, that is

$$\begin{aligned} E_{\text{dyn}} &= \frac{\mathcal{H}}{d}\sum_{m=1}^{M}\sum_{r=1}^{x_m}\left(C_{\text{eff},m} v_{m,r}^2 \sum_{\tau_i \in \Gamma_{m,r}} \mu_i l_i\right) \\ &= \frac{\mathcal{H}}{d}\sum_{m=1}^{M}\sum_{r=1}^{x_m} \mathcal{G}_{m,r}\mathcal{Q}_{m,r} = \frac{\mathcal{H}}{d}\mathcal{G}\mathcal{Q} \end{aligned} \tag{24}$$

where $\mathcal{G}_{m,r} = C_{\text{eff},m}v_{m,r}^2$, $\mathcal{Q}_{m,r} = \sum_{\tau_i \in \Gamma_{m,r}} \mu_i l_i$, M is the number of processors, and x_m is the number of frequency levels supported by the mth processor. Vector $\mathcal{G} = [\mathcal{G}_{1,1}, \mathcal{G}_{1,2}, \ldots, \mathcal{G}_{1,x_1}, \ldots, \mathcal{G}_{M,1}, \mathcal{G}_{M,2}, \ldots, \mathcal{G}_{M,x_M}]^T$ captures processor dependent parameters and vector $\mathcal{Q} = [\mathcal{Q}_{1,1}, \mathcal{Q}_{1,2}, \ldots, \mathcal{Q}_{1,x_1}, \ldots, \mathcal{Q}_{M,1}, \mathcal{Q}_{M,2}, \ldots, \mathcal{Q}_{M,x_M}]$ captures task related parameters. Here, $\mathcal{G}_{m,r} \in \mathcal{G}$ is referred to as the power dissipation factor of processor Θ_m, $\mathcal{Q}_{m,r} \in \mathcal{Q}$ is referred to as the power dissipation factor of subset $\Gamma_{m,r}$ allocated to processor Θ_m at frequency level $f_{m,r}$, and $\mu_i l_i$ is referred to as the power dissipation factor of task τ_i. It is clear that $\mathcal{G}$ is constant since $C_{\text{eff},m}$ and $v_{m,r}$ are known for the given MPSoC system Θ while $\mathcal{Q}$ is not since it depends on the task assignment and frequency selection (e.g., $\Gamma_{m,r}$). In addition, for a given task set Γ, the sum of power dissipation factor of tasks in the set is a constant (denoted by $\mathcal{V}$) and can be expressed as $\sum_{m=1}^{M}\sum_{r=1}^{x_m} \mathcal{Q}_{m,r} = \sum_{m=1}^{M}\sum_{r=1}^{x_m}\sum_{\tau_i \in \Gamma_{m,r}} \mu_i l_i = \sum_{i=1}^{N} \mu_i l_i = \mathcal{V}$.

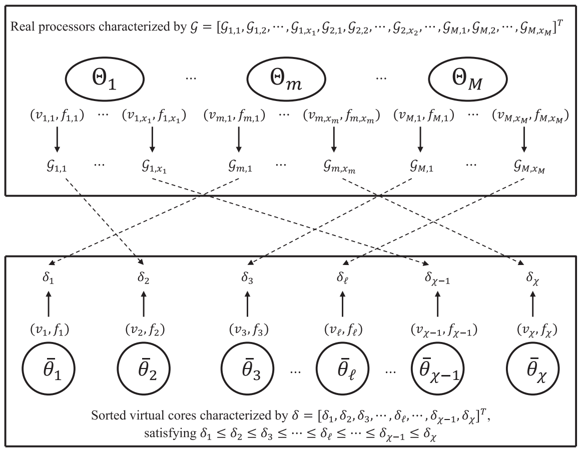

Real processors characterized by $\mathcal{G} = [\mathcal{G}_{1,1}, \mathcal{G}_{1,2}, \cdots, \mathcal{G}_{1,x_1}, \mathcal{G}_{2,1}, \mathcal{G}_{2,2}, \cdots, \mathcal{G}_{2,x_2}, \cdots, \mathcal{G}_{M,1}, \mathcal{G}_{M,2}, \cdots, \mathcal{G}_{M,x_M}]^T$
Θ_1 ⋯ Θ_m ⋯ Θ_M
$(v_{1,1}, f_{1,1})$ ⋯ (v_{1,x_1}, f_{1,x_1}) $(v_{m,1}, f_{m,1})$ ⋯ (v_{m,x_m}, f_{m,x_m}) $(v_{M,1}, f_{M,1})$ ⋯ (v_{M,x_M}, f_{M,x_M})
$\mathcal{G}_{1,1}$ ⋯ $\mathcal{G}_{1,x_1}$ $\mathcal{G}_{m,1}$ ⋯ $\mathcal{G}_{m,x_m}$ $\mathcal{G}_{M,1}$ ⋯ $\mathcal{G}_{M,x_M}$
δ_1 δ_2 δ_3 δ_ℓ $\delta_{\chi-1}$ δ_χ
(v_1, f_1) (v_2, f_2) (v_3, f_3) (v_ℓ, f_ℓ) $(v_{\chi-1}, f_{\chi-1})$ (v_χ, f_χ)
$\bar{\theta}_1$ $\bar{\theta}_2$ $\bar{\theta}_3$ ⋯ $\bar{\theta}_\ell$ ⋯ $\bar{\theta}_{\chi-1}$ $\bar{\theta}_\chi$
Sorted virtual cores characterized by $\delta = [\delta_1, \delta_2, \delta_3, \cdots, \delta_\ell, \cdots, \delta_{\chi-1}, \delta_\chi]^T$,
satisfying $\delta_1 \le \delta_2 \le \delta_3 \le \cdots \le \delta_\ell \le \cdots \le \delta_{\chi-1} \le \delta_\chi$

Fig. 3. Illustration of the RVMT transformation.

According to the above formulation, the power dissipation of the MPSoC system can be characterized by vector $\mathcal{G} = [\mathcal{G}_{1,1}, \mathcal{G}_{1,2}, \ldots, \mathcal{G}_{1,x_1}, \ldots, \mathcal{G}_{M,1}, \mathcal{G}_{M,2}, \ldots, \mathcal{G}_{M,x_M}]^T$. Since the number of power dissipation factors in the vector $\mathcal{G}$ is greater than the number of processors supporting multiple voltage levels, we propose a model transformation method that converts the DVFS-enabled MPSoC system of multiple voltage levels into a virtual core system, each core of which supports only one voltage level. During the transformation, the virtual cores obtained from a certain processor in the MPSoC system are assumed to share the same characteristics as the processor except for the supply voltage and frequency. As a result, each element in the power dissipation vector $\mathcal{G}$ corresponds to a core in the virtual core system. We then sort the cores in the increasing order of power dissipation factors.

Fig. 3 illustrates the proposed RVMT processor model transformation. As demonstrated in the figure, the virtual core system can be represented by $\bar{\Theta} = \{\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_\ell, \ldots, \bar{\theta}_{\mathcal{X}}\}$, where every core $\bar{\theta}_\ell \in \bar{\Theta}$ $(1 \le \ell \le \mathcal{X}$, and $\mathcal{X} = \sum_{m=1}^{M} x_m)$ has a fixed voltage and frequency pair (v_ℓ, f_ℓ). The virtual core system $\bar{\theta}$ is then characterized by vector $\delta = [\delta_1, \delta_2, \ldots, \delta_\ell, \ldots, \delta_{\mathcal{X}}]^T$ and $\delta_1 \le \delta_2 \le \cdots \le \delta_\ell \le \cdots \le \delta_{\mathcal{X}}$ holds, where $\delta_\ell = C_{\text{eff},\ell}v_\ell^2$ is referred to as the power dissipation factor of $\bar{\theta}_\ell$. Accordingly, the power dissipation of subsets allocated to virtual cores can be represented by vector $\xi = [\xi_1, \xi_2, \ldots, \xi_\ell, \ldots, \xi_{\mathcal{X}}]$, where $\xi_\ell = \sum_{\tau_i \in \Gamma_\ell} \mu_i l_i$ is referred to as the power dissipation factor of subset allocated to $\bar{\theta}_\ell$, and $\xi_1 + \cdots + \xi_\ell + \cdots + \xi_{\mathcal{X}} = \mathcal{V}$ holds. Thus, the dynamic energy consumption given in (24) can be expressed as

$$E_{\text{dyn}} = \frac{\mathcal{H}}{d}\delta\xi. \tag{25}$$

Let $\xi^* = [\xi_1^*, \xi_2^*, \ldots, \xi_\ell^*, \ldots, \xi_{\mathcal{X}}^*]$ be a vector that denotes the optimum power dissipation factors of subsets allocated to individual virtual cores, and E_{dyn}^* be the dynamic energy consumption of the optimum task allocation. We prove below that the dynamic energy consumption E_{dyn} given in (25) is minimized when the virtual core with smaller power dissipation factor (δ) ends up with the subset of its allocated tasks having a larger power dissipation factor (ξ^*).

Theorem 1: If the virtual core power dissipation factors $\delta_1 \le \delta_2 \le \cdots \le \delta_{\mathcal{X}}$ hold for $\delta = [\delta_1, \delta_2, \ldots, \delta_{\mathcal{X}}]^T$, and the sum of the corresponding task subset power dissipation factors $\xi_1^* + \xi_2^* + \cdots + \xi_{\mathcal{X}}^*$ is fixed for $\xi^* = [\xi_1^*, \xi_2^*, \ldots, \xi_{\mathcal{X}}^*]$, then the

Algorithm 1: Deterministic Task-to-Core Assignment

```
Input: 1) Task set represented by $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$
       2) MPSoC represented by $\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_M\}$
Output: task-to-(virtual) allocation $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_X\}$
1  transform the real processor system $\Theta$ to virtual core system $\overline{\Theta}$ by
   $\overline{\Theta} = \mathrm{RVMT}(\Theta)$, where the $\mathcal{X}$ cores of $\overline{\Theta}$ are in increasing order of
   their power dissipation factors;
2  for $l = 1$ to $\mathcal{X}$ do
3      initialize the utilization and task subset of
       virtual core $\overline{\theta}_l \in \overline{\Theta}$ by $U(\overline{\theta}_l) = 0$ and $\Gamma_l = \varnothing$;
4  $l = 1$;
5  while $\Gamma \neq \varnothing$ and $\ell \leq \mathcal{X}$ do
6      sort the tasks $\tau_i \in \Gamma$ in the decreasing order of task power
       dissipation factors $\mu_i l_i$ using heapsort;
7      for $i = 1$ to $size(\Gamma)$ do
           /* use First-Fit to group tasks */
8          if $\mathrm{RT}(\tau_i, \overline{\theta}_\ell) ==$ true and $u(\tau_i, \overline{\theta}_\ell) + U(\overline{\theta}_\ell) \leq 1$ then
9              assign task $\tau_i$ to core $\overline{\theta}_\ell$;
10             update $\Gamma$, $\Gamma_\ell$, and $U(\overline{\theta}_\ell)$ by $\Gamma = \Gamma - \tau i$, $\Gamma_\ell = \Gamma_\ell + \tau_i$,
11             and $U(\overline{\theta}_\ell) = U(\overline{\theta}_\ell) + u(\tau_i, \overline{\theta}_\ell)$;
12     $\ell \leftarrow \ell + 1$;
13 if $\Gamma \neq \varnothing$ and $\ell > \mathcal{X}$ then
14     exit(1) ;                    /* Exit when infeasible */
```

dynamic energy consumption E_{dyn} is minimized to E^*_{dyn} if $\xi^*_1 \geq \xi^*_2 \geq \cdots \geq \xi^*_\ell \geq \cdots \geq \xi^*_{\mathcal{X}}$ holds.

2) Deterministic Task-to-(Virtual) Core Allocation: We propose a suboptimal task allocation heuristic which is motivated by the theorem presented in Section IV-A1, that is, allocating the subset having a larger power dissipation factor to the virtual core having a smaller power dissipation factor can minimize the dynamic energy consumption. The heuristic operates as follows. Tasks in the subset with the maximum power dissipation factor is allocated to the virtual core with the minimum power dissipation factor, and tasks in the subset with the next maximum power dissipation factor is allocated to the virtual core with the next minimum power dissipation factor. This process repeats until all subsets of tasks are allocated to cores. In addition, the constraints of task deadline and processor capacity limit are examined during the allocation.

The details of the heuristic are described in Algorithm 1. Inputs to the algorithm are task set Γ and processor set Θ. Let $\bar{\Theta}$ be the corresponding virtual core system, and $\bar{\theta}_\ell$ be the ℓth virtual core of the virtual system model. Let $U(\bar{\theta}_\ell)$ be the utilization of virtual core $\bar{\theta}_\ell$, which is calculated as the sum of utilization of tasks assigned to the core, and $u(\tau_i, \bar{\theta}_\ell)$ be the utilization of task τ_i if it is assigned to virtual core $\bar{\theta}_\ell$, which is calculated as the quotient of task execution time and frame size.

The algorithm first transforms the MPSoC to a virtual core system using $\bar{\Theta} = \mathrm{RVMT}(\Theta)$, in which the cores are arranged in the increasing order of power dissipation factors. The utilization and subset of all virtual cores $\bar{\theta}_\ell$ $(1 \leq \ell \leq \mathcal{X})$ are initialized by $U(\bar{\theta}_\ell) = 0$ and $\Gamma_\ell = \varnothing$, respectively. It then iteratively implements the process of task-to-(virtual) core allocation if the task set is not empty and not all the cores in $\bar{\Theta}$ have been considered. Before starting the iteration, the core index ℓ is set to 1. In each round of iteration, the algorithm first sorts the tasks of set Γ in the decreasing order of task power dissipation factors $\mu_i l_i$, then allocates tasks to cores and construct subset of tasks in a first fit manner under the real-time constraint $\mathrm{RT}(\tau_i, \bar{\theta}_\ell) ==$ *true* and processor capacity limit $u(\tau_i, \bar{\theta}_\ell) + U(\bar{\theta}_\ell) \leq 1$. $\mathrm{RT}(\tau_i, \bar{\theta}_\ell)$ is a procedure that checks the real-time constraint of a task and returns *true* if the task is finished before its deadline, and *false* otherwise. If allocating task τ_i to core $\bar{\theta}_\ell$ can satisfy these constraints, the task is allocated to the core, and task set Γ, subset Γ_ℓ, and utilization $U(\bar{\theta}_\ell)$ are hence updated. The process then moves to the next iteration and considers the allocation of the next task. Otherwise, the task is not assigned and the process directly moves to the next iteration. If there is no feasible schedule for the system under the constraints, the algorithm exits.

B. ATA to Handle Uncertainties in Energy Availability

Based on the DTA, this section handles the uncertainty in energy sources by adapting the execution of approximate-computation tasks to the energy availability. We introduce a variable α to denote the ratio of the number of executed optional cycles to the maximum optional cycles of a task. The α is named as task optional execution factor and falls within the range [0, 1]. Using α, the total execution cycles of task τ_i given in (1) can be rewritten as

$$l_i = M_i + \alpha \cdot O_i. \tag{26}$$

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.

Theorem 2: Given task $\tau_i \in \Gamma$, its two different optional execution factors α and α', and the task allocation scheme in Algorithm 1, the inequality $E_{\text{dem}} > E'_{\text{dem}}$ 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 determine the task optional execution factor for given energy demand by using a simple yet effective binary search-based approach. We assume that the optional execution factor of every task is the same such that the ATA is conducted at a coarse granularity. We refine the optional execution factors of tasks such that each task has its own optional execution factor at the task scheduling stage, as shown in Section V.

To handle the uncertainty in energy availability, we adapt system energy demand to fluctuating energy supply by using the task optional execution factor as a coarse-grain control knob to adjust the system energy consumption. The proposed ATA algorithm is described in Algorithm 2. It takes as input task set Γ, processor set Θ and the given cost budget $\mathrm{Cost}_{bdg,\mathcal{H}}$, and outputs $\mathcal{X}$ subsets $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_{\mathcal{X}}\}$. It first derives the budget-constrained energy E_{sup} that is available to support the system operation during the scheduling horizon $\mathcal{H}$ based on (11). It then utilizes a binary search-based approach to derive a common task optional execution factor for all tasks in set Γ. It initializes the common task optional execution factor α to 0.5. It calculates task execution cycles using (26), calls Algorithm 2 to allocate tasks, and computes energy demand E_{dem} using (13). Once energy demand E_{dem} of the task allocation is derived, the algorithms iteratively adapt the energy demand to the fluctuating supply of energy E_{sup} according to $|E_{\text{sup}} - E_{\text{dem}}| > \epsilon$, where ϵ is a sufficiently small

Algorithm 2: Iterative Task Allocation Based on Task Optional Execution Factors

```
Input: Task set $\Gamma$ && MPSoC $\Theta$ && cost budget $Cost_{bdg,\mathcal{H}}$ && a
       sufficiently small positive number $\in$
Output: task to virtual core assignment $\{\Gamma_1, \Gamma_2, \dots, \Gamma_X\}$
1  derive available energy Esup during $[t, t+\mathcal{H}]$ using (11);
2  $min = 0, max = 1, \alpha = (min + max)/2$;
3  calculate task execution cycles for all tasks using Eq. (26) based on
   $\alpha$, call Alg. 1 to allocate tasks, and calculate energy demand $E_{dem}$
   using Eq. (13);
4  while $|E_{sup} - E_{dem}| > \in$ do
5      if $E_{dem} > E_{sup}$ then
6          $max = \alpha, \alpha = (min + max)/2$;
7      else
8          $min = \alpha, \alpha = (min + max)/2$;
9      update $l_i$ for all tasks using Eq. (26) based on $\alpha$;
10     allocate tasks using Alg. 1 based on $l_i$;
11     update $E_{dem}$ of the task schedule using Eq. (13);
```

positive number. Algorithm 2 stops when energy demand of the generated schedule closely matches to the energy supply, and returns the allocation solution.

V. QoS-Driven Task Scheduling

This section maximizes the system QoS by determining tasks whose optional parts should be executed. Once these tasks are picked out, a runtime scheme needs to be designed to adapt task schedule to the fluctuating energy availability.

A. Static Task Selection for QoS Optimization

As given in (18), 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 QoS of a system 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_ℓ denote the energy consumed by tasks in subset Γ_ℓ allocated to the virtual core $\bar{\theta}_\ell$, then E_ℓ is given by

$$E_\ell = \sum_{\tau_i \in \Gamma_\ell} P_{\text{cons}}(\tau_i, \bar{\theta}_\ell) \cdot \frac{o_i + M_i}{f_\ell} = \sum_{\tau_i \in \Gamma_\ell} \frac{P_{\text{cons}}(\tau_i, \bar{\theta}_\ell)}{f_\ell} \cdot o_i + \sum_{\tau_i \in \Gamma_\ell} \frac{P_{\text{cons}}(\tau_i, \bar{\theta}_\ell)}{f_\ell} \cdot M_i \quad (27)$$

where $P_{\text{cons}}(\tau_i, \bar{\theta}_\ell)$ is the power consumption of task τ_i on core $\bar{\theta}_\ell$, f_ℓ is the operating frequency of core $\bar{\theta}_\ell$, o_i is the actual optional execution cycles of task τ_i, and M_i is the mandatory part of task τ_i. The energy demand or consumption of the system can be expressed as the sum of energy E_ℓ consumed by individual cores. It approximates closely to the energy supply of the system by a parameter $\epsilon > 0$ since tasks are assigned to individual cores using the energy adaptive algorithm (i.e., Algorithm 2) presented in Section IV-B. Thus, we have

$$E_{\text{sup}} \approx E_{\text{dem}} = \sum_{\ell=1}^{\mathcal{X}} E_\ell = \underbrace{\sum_{\ell=1}^{\mathcal{X}} \sum_{\tau_i \in \Gamma_\ell} \frac{P_{\text{cons}}(\tau_i, \bar{\theta}_\ell)}{f_\ell} \cdot o_i}_{E_{\text{optl}}} + \underbrace{\sum_{\ell=1}^{\mathcal{X}} \sum_{\tau_i \in \Gamma_\ell} \frac{P_{\text{cons}}(\tau_i, \bar{\theta}_\ell)}{f_\ell} \cdot M_i}_{E_{\text{mand}}} \quad (28)$$

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

The problem of selecting tasks for QoS optimization is hence refined into finding optimal optional cycles for all allocated tasks under the energy budget constraint. Let

$$EC_{\ell,i} = \frac{P_{\text{cons}}(\tau_i, \bar{\theta}_\ell)}{f_\ell} \quad (29)$$

be an energy metric that gives the energy consumption of task τ_i on core $\bar{\theta}_\ell$ per execution cycle. Since every task is only allowed to bind with one core, we use EC_i rather than $EC_{\ell,i}$ to denote the energy metric of τ_i on $\bar{\theta}_\ell$ after task assignment is determined. In addition, we sort all tasks allocated to cores in ascending order of EC_i for the sake of easy presentation. Thus, the energy consumed by optional parts of all tasks is

$$E_{\text{optl}} = EC_1 \cdot o_1 + EC_2 \cdot o_2 + \cdots + EC_N \cdot o_N \quad (30)$$

and $EC_1 < EC_2 < \cdots < EC_N$ holds. In (30), the metric EC_i and task executed optional cycles $o_i = \alpha \cdot O_i$ are determined once tasks are bound to individual cores. 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 redetermining the executed optional cycles o_i of tasks using an individual optional execution factor α_i.

Clearly, EC_i is determined once tasks are bound to individual cores. Moreover, the LHS of (30) is determined after task allocation. Thus, our goal of optimization for system QoS, which is given in (18) as the sum of executed optional cycles of all tasks, becomes maximizing the sum of o_i for $1 \le i \le N$ under the constraint of (30). 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 the QoS is maximized if the tasks having smaller energy metric are selected to execute their optional parts.

Theorem 3: Given the energy budget E_{optl} of optional parts of the task set Γ, the QoS of the system, as defined in (18), is maximized if the tasks having smaller energy metric are selected to execute their optional parts.

Based on the theorem, we conclude that executing optional parts of tasks with smaller energy metric can improve the system QoS. A heuristic is thus designed to determine the execution cycles of optional parts of all tasks, which is shown in Algorithm 3. The algorithm describes the procedure to redetermine optional execution cycles of tasks in subsets for maximizing the system QoS. 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 cores in 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. Given energy metric EC_i and

Algorithm 3: Determine Task Optional Execution Cycles

```
Input: Subsets of assigned tasks produced by Alg. 2
Output: Tasks with optional execution cycles updated
1  for $\ell = 1$ to $\mathcal{X}$ do
2      for $i = 1$ to $size(\Gamma_\ell)$ do
3          calculate the energy metric $EC_i$;
4      sort the tasks allocated to core $\bar{\theta}_\ell$ in ascending order of $EC_i$ using heapsort;
5  $E_{optl} = E_{sup} - E_{mand}$;
6  for $\ell = 1$ to $\mathcal{X}$ do
7      for $i = 1$ to $size(\Gamma_\ell)$ do
8          if $E_{optl} > 0$ then
9              if $EC_i \times O_i \le E_{optl}$ then
10                 $o_i = O_i, \alpha_i = 1$;
11             else
12                 $o_i = \frac{E_{optl}}{EC_i}, \alpha_i = \frac{E_{optl}}{EC_i \times O_i}$;
13             $E_{optl} -= EC_i \times o_i$;
14         else
15             $o_i = 0, \alpha_i = 0$;
16         $i \leftarrow i + 1$;
```

Algorithm 4: Adapt Task Execution to Fluctuating Energy at Runtime

```
Input: Optional execution cycles $\{o_1, o_2, \ldots, o_N\}$ determined by
       Alg. 3 && subset $\Gamma_\ell$ of tasks on core $\bar{\theta}_\ell$
1  sort tasks in $\Gamma_\ell$ in ascending order of metric EC using heapsort;
2  $h_l = \frac{\sum_{i=1}^{size(\Gamma_\ell)} EC_i \cdot (M_i + o_i)}{f^{-1}(cost_{bdg})}$;
3  $i \leftarrow 1$;
4  while $i \le size(\Gamma_\ell)$ do
5      $l_i = (o_i + M_i)$;
6      while $EC_i \cdot l_i > h_\ell \cdot f^{-1}(Cost_{bdg}) \cdot \frac{l_i}{f_\ell \mathcal{H}}$ && $l_i \ge M_i$ do
7          $l_i = \sigma \cdot l_i$ ;   /* scale the length $l_i$ to reduce
           energy demand of task $\tau_i$ */
8      while $h_\ell \cdot f^{-1}(Cost_{bdg}) \cdot \frac{l_i}{f_\ell \mathcal{H}} - EC_i \cdot l_i > \in$ && $l_i \ge M_i$ do
9          $l_i = (1 + \sigma) \cdot l_i$ ;   /* scale the length $l_i$ to
           increase energy demand of task $\tau_i$ */
10     if $h_\ell \cdot f^{-1}(Cost_{bdg}) \cdot \frac{l_i}{f_\ell \mathcal{H}} - EC_i \cdot l_i \le \in$ && $l_i \ge M_i$ then
11         execute task $\tau_i$ with execution cycles of $l_i$;
12     else
13         drop task $\tau_i$ and $l_i = 0$ ; /* drop $\tau_i$ when the energy
           supply is insufficient */
14     update battery storage;
15     $i \leftarrow i + 1$;
```

maximum optional cycles O_i of τ_i, the energy consumed by the task can be expressed as $EC_i \times O_i$. If energy budget E_{optl} for optional parts is large enough, maximum optional cycles of τ_i can be executed by $o_i = O_i$; otherwise, only $o_i = E_{\text{optl}}/EC_i$ optional cycles can be executed. No optional execution cycles will be executed if the energy budget is exhausted. After τ_i is examined, the energy budget is updated accordingly and the process moves to the next task. The algorithm outputs all tasks once their optional execution cycles o_i are updated.

B. Dynamic Energy Adaptation for QoS Optimization

The proposed scheme generates a cost-constrained task schedule at design time by mapping tasks to processors and executing optional cycles of those tasks that can maximize the system QoS. However, due to the fact that the amount of renewable and grid energy available under a cost budget fluctuates, the static task schedule needs to be adjusted at runtime. Therefore, we provide a dynamic task scheduling algorithm that adapts task execution to fluctuating energy available at runtime. The dynamic algorithm is developed based on the QoS-aware static algorithm described in Algorithm 4.

Consider core $\bar{\theta}_\ell$ with allocated tasks in subset Γ_ℓ. Energy available for core $\bar{\theta}_\ell$ to execute tasks includes renewable energy harvested from environments, and energy drained from battery and grid. Due to the intermittence of renewable energy and time varying price of grid power, the amount of energy available to support the operation of the system under a given cost budget fluctuates. Assume that the actual energy available will be allocated to individual cores in proportion to the optimal energy value derived using Algorithm 4. The actual energy available is then weighted by a factor h_ℓ, which is defined as the ratio of the energy consumed by size(Γ_ℓ) tasks on core $\bar{\theta}_\ell$ to the energy available for the system under the cost budget of Cost$_{bdg}$. Let $f^{-1}(\text{Cost}_{bdg})$ denote the energy available for the given cost budget Cost$_{bdg}$, then the factor h_ℓ is given by

$$h_\ell = \frac{\sum_{i=1}^{\text{size}(\Gamma_\ell)} EC_i \cdot (M_i + o_i)}{f^{-1}(\text{Cost}_{bdg})}. \tag{31}$$

The dynamic algorithm running on core $\bar{\theta}_\ell$ is given in Algorithm 4. It takes as input the N task optional execution cycles determined by Algorithm 4 and the subset Γ_ℓ of tasks allocated to core $\bar{\theta}_\ell$. To achieve a higher system QoS, the algorithm sorts tasks on the core in ascending order of energy metric EC so that it first executes tasks with smaller energy consumption per execution cycle. It calculates the factor h_ℓ for $\bar{\theta}_\ell$ using (31). For task τ_i in subset Γ_ℓ, the algorithm calculates the execution cycles l_i of the task. The energy demand of the task is expressed as $EC_i \cdot l_i$ and the energy available on $\bar{\theta}_\ell$ during time interval (l_i/f_ℓ) is expressed as $h_\ell \cdot f^{-1}(\text{Cost}_{bdg}) \cdot (l_i/f_\ell \mathcal{H})$. The dynamic algorithm aims to adapt the static task schedule to the variation in energy supply. That is, it tries to reduce the runtime mismatch between the energy supply and demand before a task is executed. On one hand, if energy demand of τ_i is greater than energy supply of the task, then execution cycles of the task and in turn the energy demand of the task are iteratively reduced by an empirical factor $\sigma \in [0, 1]$. The iteration stops when the energy demand is less than or equal to the energy supply. On the other hand, if the energy supply of τ_i is greater than the energy demand of the task by a small positive number $\epsilon > 0$, the execution cycles of the task and in turn the energy demand of the task are iteratively increased by an empirical factor $\sigma \in [0, 1]$. After execution cycles and energy demand of τ_i are updated in accordance with the fluctuation of energy supply, the task is executed if its execution cycles l_i is greater than its mandatory cycles M_i; otherwise, τ_i is dropped and l_i is reset to 0. The energy available for core $\bar{\theta}_\ell$ is updated and the procedure moves to the next task.

VI. Evaluation

A. Experimental Settings

We validate the proposed cost-constrained task allocation and scheduling scheme through two sets of simulation experiments. The first set of simulations is based on synthetic real-time tasks while the second set of simulations is based on real-life benchmarks. As introduced in Sections IV and V, our scheme is composed of four algorithms. The first one is a DTA (Algorithm 1) proposed in [26] that minimizes the energy consumption by utilizing heterogeneities of both processors and tasks. The second one is an ATA (Algorithm 2) that uses a common optional execution factor to handle the uncertainty in energy availability and adopts DTA to allocate tasks. The third one is a static task selection (STS, Algorithm 3) that selects certain tasks to complete their optional parts for maximizing QoS under the energy budget. The tasks with smaller *EC* are given higher priority to execute optional parts. The three algorithms are developed at design time. The fourth one is a dynamic task scheduling (DTS, Algorithm 4) which adjusts the static schedule to adapt to the fluctuating harvested energy at runtime.

Accordingly, four comparative experiments are carried out in each set of simulations to validate our scheme from different perspectives. First, we compare the energy consumption of our DTA scheme [26] with that of the hybrid worst-fit genetic algorithm (HWGA) [27] to validate the efficiency of our ATA algorithm in saving energy, which is developed based on DTA. We also investigate the degree of match between energy demand and supply by using our ATA to validate the efficiency of our ATA in utilizing the energy supply, which specifically exploits an optimal common task optional execution factor to maximize the degree of match. Second, we compare the QoS of our ATA with that of DTA under a given cost constraint to validate the efficiency of our scheme in improving QoS at task allocation stage. Third, we compare the QoS of our STS with that of baseline method RAND, RVS, and benchmarking algorithm critical-task-first (CTF) [13] to validate the efficiency of our scheme in improving QoS at task scheduling stage. Finally, to validate the effectiveness of our runtime scheme DTS, we compare the QoS of DTS with that of the dynamic algorithm, gradient curve shifting (GCS) [12]. These algorithms are described below.

1) *DTA* [26] is an energy-efficient task allocation algorithm designed for systems of deterministic energy supply. It can minimize dynamic energy consumption by assigning the subset of tasks having a large power dissipation factor to the processor having a small power dissipation factor.
2) *HWGA* [27] is a state-of-the-art approach that integrates a worst-fit-based partitioning heuristic with the genetic algorithm to generate a task allocation that reduces the energy consumption while satisfying all system constraints.
3) *RAND* is a method that randomly selects tasks whose optional parts are to be completed under the budget.
4) *RVS* is a method that selects the task whose energy metric is larger to execute first, which is exactly the opposite of our proposed task selection algorithm STS.
5) *CTF* [13] is a method that assigns QoS-critical jobs higher priorities such that their optional cycles can be completed first. The QoS-critical jobs are defined as tasks with larger maximum optional cycles.
6) *GCS* [12] is a dynamic scheduling method that determines the best allocation of slack cycles generated at runtime for maximizing QoS under energy constraint.

B. Simulation for Synthetic Real-Time Tasks

We perform this simulation based on an MPSoC system with eight heterogeneous processors ($M = 8$). Our processor

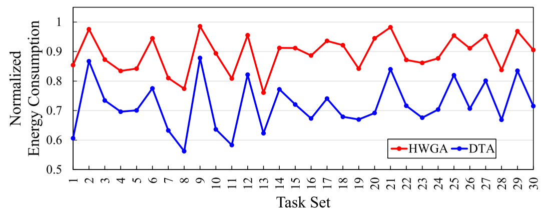


Fig. 4. Normalized energy consumption of 30 task sets using our DTA and benchmarking algorithm HWGA [27].

model is built on 65 nm technology and the parameters of processors can be found in [26], [28], and [29]. Task activity factors μ are uniformly distributed in the interval [0.4, 1], demonstrating the heterogeneous nature of tasks [15]. The size N of task set Γ is set to 200. The worst case execution cycles (WCEC) of tasks in the set Γ are assumed to be in the range of $[4\times10^7, 6\times10^8]$. Each task τ_i is instantiated by randomly picking two WCECs from the range, one is for its mandatory part M_i and the other is for its maximum optional part O_i. The common deadline d of tasks is assumed to be $1.5\sum_{i=1}^{N} M_i/f_{\max}$, where $f_{\max}$ is the maximum frequency. Multiple task sets are constructed to validate the proposed scheme in the simulation.

Solar source is selected as the renewable generation, and the trace of harvesting power $P_{\text{harv}}(t)$ is generated according to the equation $P_{\text{harv}}(t) = \left|\psi \cdot \Psi(t) \cdot \cos([t/70\pi]) \cdot \cos([t/100\pi])\right|$ [7], [30], where ψ is a coefficient and $\Psi(t)$ is a normally distributed random variable with variance 1 and mean 0. The ψ is used to generate two different $P_{\text{harv}}(t)$, the first of which is taken as predicted harvesting power and the second of which is used as actual harvesting power. The real-time pricing model offered by Ameren Illinois Corporation [31] is adopted as the pricing model of grid electricity. A Li-lion battery bank equipped with 4 Ah normal capacity and 15 V terminal voltage [18] is served as the energy storage module. The whole scheduling horizon $\mathcal{H}$ is set to 24 h and divided into 24 time slots with equal and constant length of 1 h.

1) Evaluate the Effectiveness of Our ATA Scheme in Energy Management: Our ATA scheme is developed based on the energy-efficient DTA scheme [26] that minimizes the system energy consumption by fully exploiting the heterogeneities of both processors and tasks, and an optimal task common optional execution factor that can maximize the degree of match between system energy demand and supply. To evaluate the effectiveness of our ATA in energy management, we first compare our DTA with benchmarking algorithm HWGA [27] in terms of energy consumption, then investigate the degree of match between energy demand and supply using our ATA.

Fig. 4 shows the normalized energy consumption of 30 task sets using our DTA scheme [26] and benchmarking algorithm HWGA [27]. It has been demonstrated in the figure that our DTA consumes less energy as compared to HWGA. To be specific, the energy consumption of DTA is 19.7% lower than that of HWGA on average. Furthermore, the energy savings achieved by DTA over HWGA can be up to 29.0%. For example, the normalized energy consumed by executing task set 1 using HWGA and DTA are 0.854 and 0.607, respectively. In addition to the efficiency of reducing energy consumption achieved by DTA, our scheme ATA can make the most of energy supply by introducing task optional execution factor. Fig. 5 plots the energy supplied by the harvesting system and

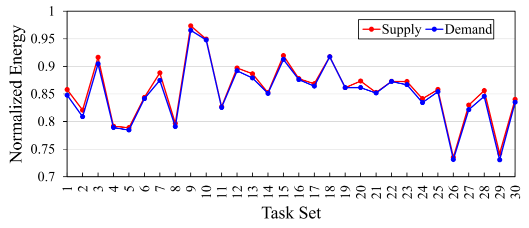


Fig. 5. Normalized energy supply and demand of our proposed ATA algorithm for executing 30 task sets.

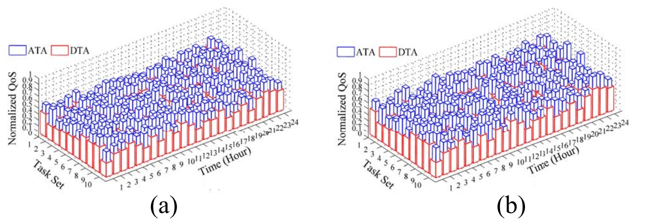


Fig. 6. Normalized QoS of ten synthetic task sets over 24 h using the proposed DTA and ATA scheme under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

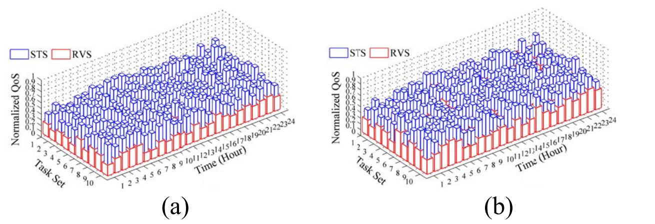


Fig. 7. Normalized QoS of ten synthetic task sets over 24 h using the proposed STS and baseline method RVS under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

the energy demanded by our ATA for executing 30 task sets. It can be easily seen from the figure that our scheme achieves a close match between energy supply and demand.

2) Evaluate the Effectiveness of Our ATA and STS Scheme for QoS Improvement: Two comparative experiments are carried out to validate the effectiveness of our proposed static QoS-aware task selection scheme under varying cost budgets. We utilize a normalized cost budget in the comparative study, the maximum of which is set to 1. That is, the normalized cost budget varies in the range of (0, 1]. The first comparative study is conducted from the perspective of task allocation, where two allocation schemes, the DTA without considering energy uncertainty and ATA considering energy uncertainty, are compared with respected to QoS. Fig. 6 presents the normalized QoS obtained when executing the optional cycles of tasks in ten sets over a scheduling horizon of 24 h using the proposed DTA and ATA scheme under varying cost budgets. As compared to DTA, ATA achieves higher QoS of up to 55.1%, especially when the cost budget is low. For example, the average QoS of ATA are 51.8% and 42.1% higher than that of DTA for $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively.

The second comparative experiments investigate the impact of STS on system QoS. To be specific, we compare the QoS of our STS with that of baseline task selection methods RAND, RVS, and benchmarking algorithm CTF [13] to validate the efficiency of our scheme in improving QoS at the task scheduling stage. Fig. 7 shows the normalized QoS obtained when

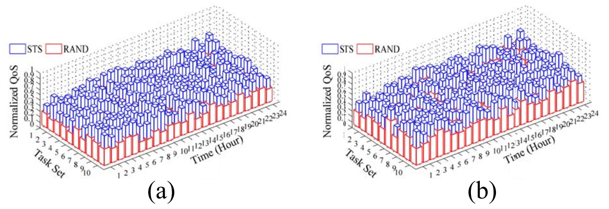


Fig. 8. Normalized QoS of ten synthetic task sets over 24 h using the proposed STS and baseline method RAND under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

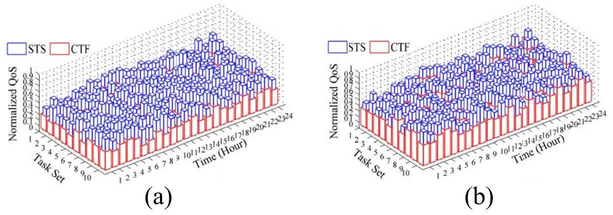


Fig. 9. Normalized QoS of ten synthetic task sets over 24 h using our STS and benchmarking method CTF [13] under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

executing optional cycles of tasks in ten sets of tasks using the proposed STS and the baseline method RVS under the cost budget of $\text{Cost}_{bgt} = 0.6$ and 0.7. As compared to RVS, the average QoS of our STS scheme are 97.6% and 72.4% higher in the case of $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively. Fig. 8 compares the proposed STS and baseline method RAND in terms of the QoS obtained when executing optional cycles of tasks under the cost budget of $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively. It can be seen from the figure that the average QoS of STS are 75.2% and 51.8% higher than that of RAND for $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively. A comparative study of the proposed STS and benchmarking method CTF is demonstrated in Fig. 9. The figure indicates that the average QoS of our STS scheme are 61.2% and 39.7% higher than that of the CTF method for $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively.

From the above analyses, we can draw the followings.

1) From the perspective of task allocation, the ATA that considers the uncertainty in energy availability outperforms the DTA that does not in terms of system QoS.
2) As to selecting tasks on individual processors to be executed, the STS outperforms methods RVS, RAND, and CTF in terms of system QoS. This is because our STS utilizes heterogeneities of both tasks and processors for QoS improvement under given cost constraints.
3) Our ATA and STS scheme outperform the benchmarking methods regarding to system QoS improvement, especially, when the given cost budget is low. This is because when the given cost budget and the corresponding available energy is low, our ATA and STS scheme can efficiently utilize the intermittent available energy so that a larger portion of the optional part of a task is finished.

3) Evaluate the Effectiveness of Our DTS Scheme for QoS Improvement: The effectiveness of our dynamic task scheduling scheme, referred to as DTS, is also verified in the simulation. We compare the proposed dynamic task scheduling scheme DTS with benchmarking algorithm GCS [12] in terms of runtime QoS. Fig. 10 gives the normalized QoS achieved

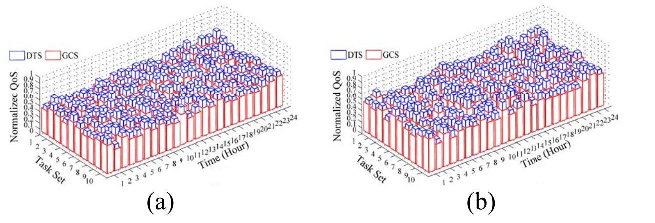


Fig. 10. Normalized QoS of ten synthetic task sets over 24 h using our DTS and benchmarking method GCS [12] under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

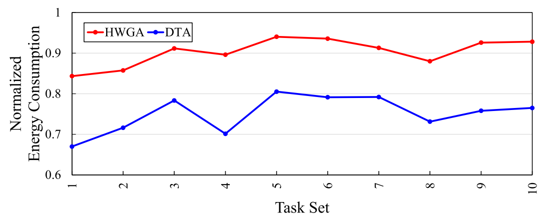


Fig. 11. Normalized energy consumption of ten real-life task sets using our DTA and benchmarking algorithm HWGA [27].

when executing optional cycles of tasks in ten sets under the cost budget of $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively.

We can observe from the figure that the average QoS of our DTS scheme are 17.7% and 15.1% higher than that of the GCS [12] algorithm for $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively. This is primarily due to the fact that our proposed DTS utilizes heterogeneities of both processors and tasks to enhance system energy efficiency in a way so that the energy consumed per execution cycle is minimized. Owing to the enhanced energy efficiency achieved by judiciously assigning tasks to processors, DTS could not only guarantee the completion of task mandatory parts, but also execute more task optional parts. In addition to the enhanced energy efficiency that leads to a higher QoS, the DTS could directly improve QoS by intelligently selecting tasks on individual processors to execute their optional parts. On the contrary, the GCS [12] improves system QoS by only using slack claimed in the runtime, which renders relatively small headroom for QoS improvement.

C. Simulation for Real-Life Benchmarks

In this simulation, a heterogeneous MPSoC system [23] that consists of an AMD Athlon processor with three supply voltage/frequency levels and an TI DSP processor with two supply voltage/frequency levels is adopted. The three supply voltage and frequency pairs of AMD Athlon are (0.89 V, 1.8 GHz), (1.12 V, 2.4 GHz), (1.34 V, 3 GHz), and the two supply voltage and frequency pairs of TI DSP are (0.98 V, 2.0 GHz), (1.42 V, 3.0 GHz) [23], respectively. The settings of solar source, Li-lion battery bank, and real-time pricing are the same as that of simulation for synthetic real-time tasks. The tool MEGA [32] that incorporates approximate computation is utilized in this simulation to generate real-life benchmarks.

1) Evaluate the Effectiveness of Our ATA Scheme in Energy Management: Fig. 11 shows the normalized energy consumption of ten real-life task sets using our DTA scheme [26] and benchmarking algorithm HWGA [27]. It has been demonstrated in the figure that our DTA consumes less energy as compared to HWGA. To be specific, the energy consumption of DTA is 16.9% lower than that of HWGA on

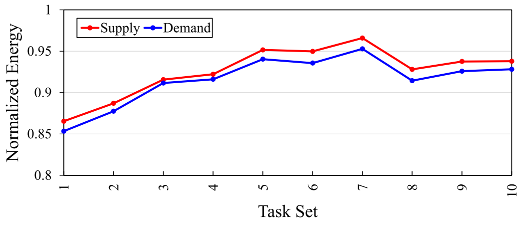


Fig. 12. Normalized energy supply and demand of our proposed ATA algorithm for executing 10 real-life task sets.

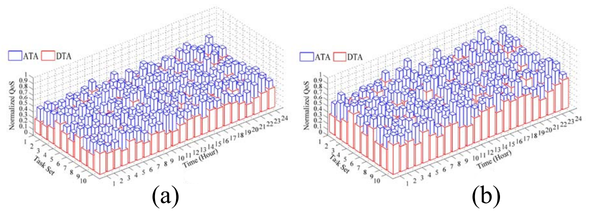


Fig. 13. Normalized QoS of ten real-life task sets over 24 h using our DTA and ATA scheme under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

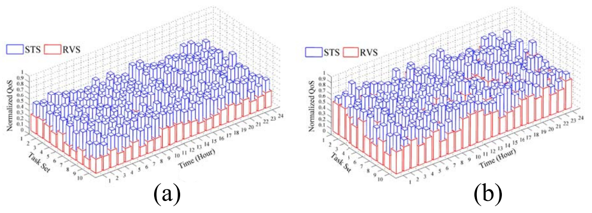


Fig. 14. Normalized QoS of ten real-life task sets over 24 h using our STS and baseline method RVS under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

average. Furthermore, the energy savings achieved by DTA over HWGA can be up to 21.7%. For example, the normalized energy consumed by executing the real-life task set 4 using HWGA and DTA are 0.896 and 0.701, respectively. Fig. 12 plots the energy supplied by the harvesting system and the energy demanded by our ATA scheme for executing ten real-life task sets. It can be easily seen from the figure that our scheme achieves a close match between energy supply and demand.

2) Evaluate the Effectiveness of Our ATA and STS Scheme for QoS Improvement: Fig. 13 presents the normalized QoS obtained when executing the optional cycles of real-life tasks in ten sets over a scheduling horizon of 24 h using the proposed DTA and ATA scheme under varying cost budgets. Compared to DTA, ATA achieves a higher QoS of up to 68.8%, especially when the cost budget is low. For example, the average QoS of ATA are 59.6% and 48.3% higher than that of DTA for $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively. Fig. 14 shows the normalized QoS obtained when executing optional cycles of real-life tasks in ten sets using the proposed STS algorithm and baseline method RVS under the cost budget of $\text{Cost}_{bgt} = 0.6$ and 0.7. As compared to RVS, the average QoS of STS are 108.6% and 63.8% higher for $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively. Fig. 15 compares the proposed STS algorithm and baseline method RAND in terms of the QoS obtained when executing optional cycles of real-life tasks in ten sets under the cost budget of $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively. It can be

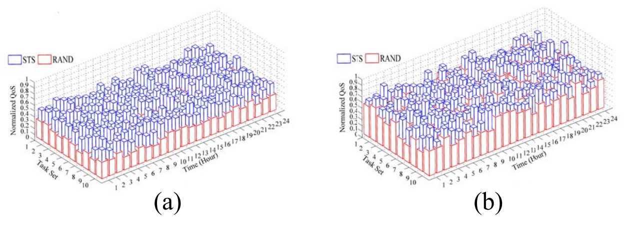


Fig. 15. Normalized QoS of ten real-life task sets over 24 h using our STS and baseline method RAND under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

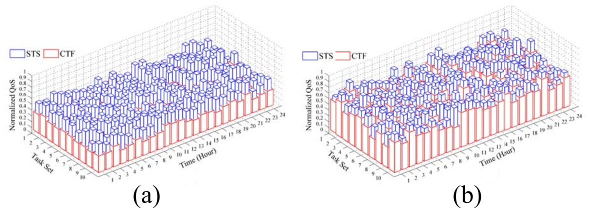


Fig. 16. Normalized QoS of ten real-life task sets over 24 h using our STS and baseline method CTF [13] under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

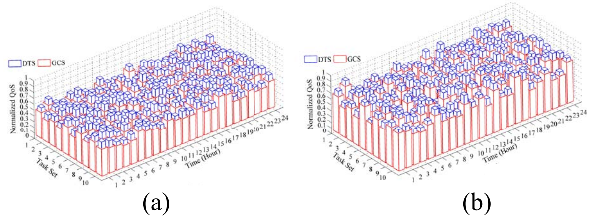


Fig. 17. Normalized QoS of 10 real-life task sets over 24 h using our DTS and benchmarking method GCS [12] under varying cost budgets. (a) $\text{Cost}_{bgt} = 0.6$ and (b) $\text{Cost}_{bgt} = 0.7$.

seen from the figure that the average QoS of STS are 78.6% and 40.4% higher than that of RAND for $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively. A comparative study of the proposed STS algorithm and benchmarking method CTF is demonstrated in Fig. 16. The figure indicates that the average QoS of STS are 69.9% and 26.7% higher than that of CTF for $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively.

3) Evaluate the Effectiveness of Our DTS Scheme for QoS Improvement: Fig. 17 gives the normalized QoS achieved when executing optional cycles of real-life tasks in ten sets under the cost budget of $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively. We can observe from the figure that the average QoS of our DTS scheme are 20.8% and 13.5% higher than that of the GCS [12] algorithm for $\text{Cost}_{bgt} = 0.6$ and 0.7, respectively.

VII. Conclusion

We propose to utilize the characteristic of approximate computation to intelligently handle uncertainties in energy availability of an MPSoC real-time system that is powered by a hybrid of energy harvested from environments and drained from power grid. For systems of such power supply, we design a static energy-ATA scheme and an QoS-driven task scheduling scheme that not only reduce the energy consumption but also improve the system QoS. We also design a dynamic task scheduling algorithm that adapts task execution to fluctuating energy available at runtime. We conduct extensive simulations to validate the effectiveness of our schemes. Results show that our algorithms can reduce energy consumption by up to 29% and improve system QoS by up to 108% as compared to benchmarking schemes.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful suggestions. The proof of theorems are omitted due to page limit.

References

[1] M. Elhebeary, M. Ibrahim, M. Aboudina, and A. Mohieldin, “Dual-source self-start high-efficiency micro-scale smart energy harvesting system for IoT applications,” *IEEE Trans. Ind. Electron.*, vol. 65, no. 1, pp. 342–351, Jun. 2017.

[2] J. Zhou, J. Yan, T. Wei, M. Chen, and X. S. Hu, “Energy-adaptive scheduling of imprecise computation tasks for QoS optimization in real-time MPSoC systems,” in *Proc. DATE*, Lausanne, Switzerland, 2017, pp. 1402–1407.

[3] (2015). *Peek Traffic Corporation, Video Detection Products*. [Online]. Available: http://peektraffic.com/products_video_detection.php

[4] J. W. S. Liu *et al.*, *Algorithms for Scheduling Imprecise Computations*. New York, NY, USA: Springer, 1991.

[5] 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, 2016, Art. no. 7.

[6] M. Severini, S. Squartini, and F. Piazza, “Energy-aware lazy scheduling algorithm for energy-harvesting sensor nodes,” *Neural Comput. Appl.*, vol. 23, nos. 7–8, pp. 1899–1908, 2013.

[7] C. Moser, D. Brunelli, L. Thiele, and L. Benini, “Real-time scheduling for energy harvesting sensor nodes,” *Real Time Syst.*, vol. 37, no. 3, pp. 233–260, 2007.

[8] M. Chetto and A. Queudet, “Clairvoyance and online scheduling in real-time energy harvesting systems,” *Real Time Syst.*, vol. 50, no. 2, pp. 179–184, 2014.

[9] Y. Abdeddaïm, Y. Chandarli, and D. Masson, “The optimality of PFPasap algorithm for fixed-priority energy-harvesting real-time systems,” in *Proc. ECRTS*, Paris, France, 2013, pp. 47–56.

[10] G. L. Stavrinides and H. D. Karatza, “Scheduling real-time DAGs in heterogeneous clusters by combining imprecise computations and bin packing techniques for the exploitation of schedule holes,” *Future Gener. Comput. Syst.*, vol. 28, no. 7, pp. 977–988, 2012.

[11] L. A. Cortes, P. Eles, and Z. Peng, “Quasi-static assignment of voltages and optional cycles in imprecise-computation systems with energy considerations,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 10, pp. 1117–1129, Oct. 2006.

[12] H. Yu, B. Veeravalli, and Y. Ha, “Dynamic scheduling of imprecise-computation tasks in maximizing QoS under energy constraints for embedded systems,” in *Proc. ASPDAC*, Seoul, South Korea, 2008, pp. 452–455.

[13] H. Kooti, N. Dang, D. Mishra, and E. Bozorgzadeh, “Energy budget management for energy harvesting embedded systems,” in *Proc. RTCSA*, Seoul, South Korea, 2012, pp. 320–329.

[14] Y. Liu, H. Yang, R. P. Dick, H. Wang, and L. Shang, “Thermal vs energy optimization for DVFS-enabled processors in embedded systems,” in *Proc. ISQED*, San Jose, CA, USA, 2007, pp. 204–209.

[15] H. Huang, V. Chaturvedi, G. Quan, J. Fan, and M. Qiu, “Throughput maximization for periodic real-time systems under the maximal temperature constraint,” *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 2, pp. 1–22, 2014.

[16] M. Amirijoo, J. Hansson, and S. H. Son, “Specification and management of QoS in real-time databases supporting imprecise computations,” *IEEE Trans. Comput.*, vol. 55, no. 3, pp. 304–319, Mar. 2006.

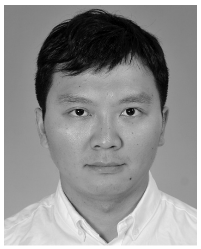
[17] A. Millner, “Modeling lithium ion battery degradation in electric vehicles,” in *Proc. CITRES*, Waltham, MA, USA, 2010, pp. 349–356.

[18] Y. Wang, X. Lin, Q. Xie, N. Chang, and M. Pedram, “Minimizing state-of-health degradation in hybrid electrical energy storage systems with arbitrary source and load profiles,” in *Proc. DATE*, Dresden, Germany, 2014, pp. 1–4.

[19] T. Cui *et al.*, "Optimal control of PEVs for energy cost minimization and frequency regulation in the smart grid accounting for battery state-of-health degradation," in *Proc. DAC*, San Francisco, CA, USA, 2015, pp. 1–6.
[20] C. Perera, D. S. Talagala, C. H. Liu, and J. C. Estrella, "Energy-efficient location and activity-aware on-demand mobile distributed sensing platform for sensing as a service in IoT clouds," *IEEE Trans. Comput. Soc. Syst.*, vol. 2, no. 4, pp. 171–181, Dec. 2015.
[21] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.
[22] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto, "Survey of energy-cognizant scheduling techniques," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1447–1464, Jul. 2013.
[23] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2867–2876, Nov. 2014.
[24] W. Trappe, R. Howard, and R. S. Moore, "Low-energy security: Limits and opportunities in the Internet of Things," *IEEE Security Privacy Mag.*, vol. 13, no. 1, pp. 14–21, Jan./Feb. 2015.
[25] P. Schaumont, "Security in the Internet of Things: A challenge of scale," in *Proc. DATE*, Lausanne, Switzerland, 2017, pp. 674–679.
[26] J. Zhou *et al.*, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 8, pp. 1269–1282, Aug. 2016.
[27] S. Saha, Y. Lu, and J. S. Deogun, "Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems," in *Proc. RTCSA*, Seoul, South Korea, 2012, pp. 41–50.
[28] W. Liao, L. He, and K. M. Lepak, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 7, pp. 1042–1053, Jul. 2005.
[29] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature-constraint hard real-time periodic tasks," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 329–339, Aug. 2010.
[30] J. Chen, T. Wei, and J. Liang, "State-aware dynamic frequency selection scheme for energy-harvesting real-time systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 8, pp. 1679–1692, Aug. 2013.
[31] *Day Ahead Pricing*, Ameren Illinois, St. Louis, MO, USA, 2017. [Online]. Available: https://www2.ameren.com/RetailEnergy/RealTimePrices
[32] Y. G. Tirat-Gefen, D. C. Silva, and A. C. Parker, "Incorporating imprecise computation into system-level design of application-specific heterogeneous multiprocessors," in *Proc. DAC*, Anaheim, CA, USA, 1997, pp. 58–63.

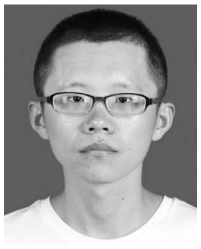
Tongquan Wei (M'11) received the Ph.D. degree in electrical engineering from Michigan Technological University, Houghton, MI, USA, in 2009.

He is currently an Associate Professor with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interests include real-time embedded systems, green and reliable computing, parallel and distributed systems, and cloud computing.



Junlong Zhou (S'15–M'17) received the Ph.D. degree in computer science from East China Normal University, Shanghai, China, in 2017.

He was a Visiting Scholar with the University of Notre Dame, Notre Dame, IN, USA, from 2014 to 2015. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His current research interests include real-time embedded systems, cloud computing, and cyber physical systems.



Kun Cao is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China.

His current research interests include high performance computing, multiprocessor systems-on-chip, and cyber physical systems.

Peijin Cong received the B.S. degree from the Department of Computer Science and Technology, East China Normal University, Shanghai, China, in 2016, where she is currently pursuing the master's degree.

Her current research interest includes power management in mobile devices.

Mingsong Chen (S'08–M'11) received the B.S. and M.E. degrees from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006, respectively, and the Ph.D. degree in computer engineering from the University of Florida, Gainesville, FL, USA, in 2010.

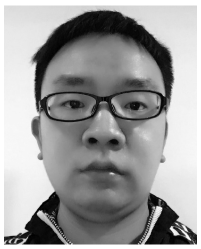
He is currently a Full Professor with the Department of Embedded Software and Systems, East China Normal University, Shanghai, China. His current research interests include design automation of cyber-physical systems, formal verification techniques, and mobile cloud computing.

Gongxuan Zhang (SM'12) received the B.S. degree in electronic computer from Tianjin University, Tianjin, China, in 1983, and the M.S. and Ph.D. degrees in computer application from the Nanjing University of Science and Technology, Nanjing, China, in 1991 and 2005, respectively.

He was a Senior Visiting Scholar with the Royal Melbourne Institute of Technology, Melbourne, VIC, Australia, from 2001 to 2002 and the University of Notre Dame, Notre Dame, IN, USA, in 2017 for three months. Since 1991, he has been with the Nanjing University of Science and Technology, where he is currently a Professor with the School of Computer Science and Engineering. His current research interests include multicore and parallel processing and distributed computing.

Xiaobo Sharon Hu (S'85–M'89–SM'02–F'16) received the B.S. degree from Tianjin University, Tianjin, China, the M.S. degree from the New York University Tandon School of Engineering, Brooklyn, NY, USA, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA.

She is currently a Professor with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA. Her current research interests include real-time embedded systems, low-power system design, and computing with emerging technologies. She has authored over 250 papers in the above areas.



Jianming Yan received the master's degree from the Department of Computer Science and Technology, East China Normal University, Shanghai, China, in 2016.

He is currently a Senior Software Engineer with Meituan.com Corporation, Beijing, China. His current research interest includes task allocation and scheduling techniques in heterogeneous real-time MPSoC systems.