Power-Adaptive Computing System Design for Solar-Energy-Powered Embedded Systems

Qiang Liu, *Member, IEEE*, Terrence Mak, *Member, IEEE*, Tao Zhang, Xinyu Niu, Wayne Luk, *Fellow, IEEE*, and Alex Yakovlev, *Senior Member, IEEE*

***Abstract*— Through energy harvesting system, new energy sources are made available immediately for many advanced applications based on environmentally embedded systems. However, the harvested power, such as the solar energy, varies significantly under different ambient conditions, which in turn affects the energy conversion efficiency. In this paper, we propose an approach for designing power-adaptive computing systems to maximize the energy utilization under variable solar power supply. Using the geometric programming technique, the proposed approach can generate a customized parallel computing structure effectively. Then, based on the prediction of the solar energy in the future time slots by a multilayer perceptron neural network, a convex model-based adaptation strategy is used to modulate the power behavior of the real-time computing system. The developed power-adaptive computing system is implemented on the hardware and evaluated by a solar harvesting system simulation framework for five applications. The results show that the developed power-adaptive systems can track the variable power supply better. The harvested solar energy utilization efficiency is 2.46 times better than the conventional static designs and the rule-based adaptation approaches. Taken together, the present thorough design approach for self-powered embedded computing systems has a better utilization of ambient energy sources.**

***Index Terms*— Design optimization, energy harvesting, neural network, power adaptation.**

I. INTRODUCTION

ENVIRONMENTALLY embedded systems have developed rapidly in recent years. These systems are deployed in various environments, such as forests, rivers, buildings, and factories, to monitor the change of environmental conditions [1], [2].

A critical concern with regards to the design of the environmentally embedded systems is how to supply the electrical power both readily and reliably to allow the systems work perpetually. Batteries with limited capabilities are not an economical solution [3]. With the advancement in energy harvesting technologies [4], a new spectrum of power delivery strategies according to various environmental conditions and systematic volumetric is available. As a result, the environmentally embedded systems can be self-powered by harvesting ambient energy from the environments.

Energy harvesting powered system needs to adapt to the unstable nature of the ambient energy. Hence, power adaptability is one of the major design merits for the system. Power adaptability optimizes the system behavior according to the changing external power supply and thus allows the system to work in a long duration [5], [6].

Fig. 1 shows a model of the energy harvesting embedded systems [1], [2]. The energy harvester converts ambient energy to electricity. The power converter extracts power from the harvester and performs ac–dc or dc–dc conversion with the goal of transferring as much power as possible to the energy storage or application system. The energy storage buffers the harvested energy temporally and supplies energy to the application system, the embedded application-specific functional unit. If the harvested energy exceeds the energy consumed by the application system, the system is powered directly by the harvester and the surplus will charge the energy storage.

To improve the energy conversion efficiency, studies on various system modules shown in Fig. 1 were carried out in the past few years. A reconfiguration algorithm was proposed to increase the power conversion efficiency of a solar

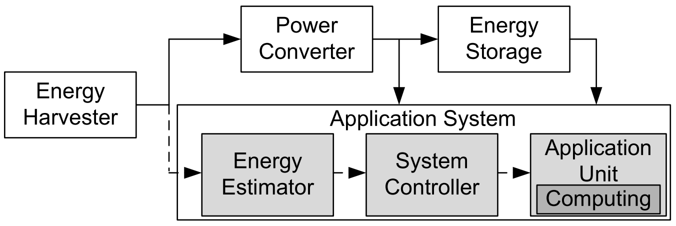


Fig. 1. Energy harvesting embedded system model.

Manuscript received October 11, 2013; revised April 21, 2014; accepted July 16, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61204022, in part by the Natural Science Foundation of Tianjin under Grant 12JCYBJC30700, in part by the Engineering and Physical Sciences Research Council of the United Kingdom, and in part by the European Union Seventh Framework Programme under Grant 248976 and Grant 257906.

Q. Liu and T. Zhang are with the School of Electronic Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: qiangliu@tju.edu.cn; todd@tju.edu.cn).

T. Mak is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong (e-mail: fstmakg@cse.cuhk.edu.hk).

X. Niu and W. Luk are with the Department of Computing, Imperial College London, London SW7 2AZ, U.K. (e-mail: niu.xinyu10@imperial.ac.uk; w.lukg@imperial.ac.uk).

A. Yakovlev is with the School of Electrical, Department of Electronic and Computing Engineering, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K. (e-mail: falex.yakovlevg@ncl.ac.uk).

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/TVLSI.2014.2342213

harvester [7]. A low-power maximum-power-point-tracking controller was implemented to achieve an efficient power converter [8], while a control algorithm for dc–dc converters was developed [9]. An efficient charging method for energy storage supercapacitors was presented to improve buffering efficiency [10]. Power management techniques to improve the consumption efficiency for the embedded application system were developed [2], [5], [11]–[15].

Many new advanced applications based on the environmentally embedded systems require real-time information processing such as image compression and data classification [16], [17]. However, most existing solar-energy-powered embedded systems, such as the wireless sensor networks, lack this capability. Therefore, this paper proposes a design approach for power-adaptive computing system, which equips the environmentally embedded systems with the required computing capability.

The design of the power-adaptive computing system consists of the design of the energy estimator, the system controller, and the computing unit, as shown in Fig. 1. We propose a thorough approach for designing and managing the power-adaptive computing system. Specifically, several design optimization techniques are applied and the design space is explored for a parallel computing structure with low-speed homogeneous processing units (PUs). Due to the interaction between the multiple optimization objectives and the non-linearity of the design space, the geometric programming technique is exploited to formulate the design space and provide the globally optimized solutions. The power profile of the designed computing unit is regulated using the clock gating technique. At run time, a simplified geometric programming optimization problem is solved to determine which PUs are clock-gated ON/OFF for a particular time slot.

This paper also extends the design optimization techniques [18]–[20] to design the parallel computing unit and the system controller with practical considerations. They are integrated with an energy estimator to provide a thorough solution for designing and managing the power-adaptive computing system. The proposed approach is comprehensively evaluated in various solar energy contexts. The contributions of this paper are as follows.

1) A geometric programming technique-based modeling approach is presented for designing the power-adaptive computing unit with multiple low speed homogeneous PUs. The computing unit adds additional computing capability as well as power adaptability to the environmentally embedded systems.
2) A convex model-based adaptation strategy is proposed for the power management of the computing unit. The adaptation strategy improves the energy consumption efficiency by maximizing the performance of the computing unit and by controlling the power consumption (i.e., power consumption does not exceed the harvested power).
3) Given the fact that the robustness of the energy harvesting powered adaptive systems relies on the accuracy of energy prediction [2], the artificial neural network is exploited to estimate the solar energy available in the future time slots. The neural network-based prediction with an accuracy of 2.45% is input to the system controller to provide a reliable adaptation control.
4) The proposed power-adaptive computing system was first implemented on an FPGA-based embedded hardware platform and then evaluated by a solar energy harvesting simulation framework. Five signal processing algorithms were used to exemplify the practical application of the proposed approach. The results demonstrate the achieved computing performance and improved energy utilization efficiency.

The rest of this paper is arranged as follows. Section II includes a comparison between the low-power and power-adaptive designs, as well as an introduction of various techniques for dynamic power management. Section III presents the proposed approach for designing the power-adaptive computing system. Section IV describes a practical instantiation of the proposed approach for five applications. The experimental results are shown in Section V. Section VI is the conclusion.

II. Background

A. Low-Power Design Versus Power-Adaptive Design

Designing power-adaptive systems is quite different from designing low power systems. Traditional design methods for low power systems aim to minimize the average power dissipation to increase the batteries-driven powerup duration of an electronic device. Hence, the main concern is the reduction of energy consumption. In contrast, the design goal of the power-adaptive computing systems in this paper is to maximize the system performance, under the constraint that the power consumption does not exceed the amount of power supply available. This constraint is also known as the energy neutral mode [5], where the energy consumption rate has to match with the availability of the power supply.

Two optimization problems formulating the low power design ($\mathcal{P}_1$) and power-adaptive design ($\mathcal{P}_2$) are shown as follows:

$$\begin{aligned} \min \quad & P_c(\vec{x}) \\ \text{s.t.} \quad & T_{\text{exe}}(\vec{x}) \leq T_{\text{req}} \end{aligned} \qquad (\mathcal{P}_1)$$

$$\begin{aligned} \min \quad & T_{\text{exe}}(\vec{x}) \\ \text{s.t.} \quad & P_c(\vec{x}) \leq P_s(t) \\ & T_{\text{exe}}(\vec{x}) \leq T_{\text{req}} \end{aligned} \qquad (\mathcal{P}_2)$$

where $P_c(\vec{x})$ is system power consumption, $T_{\text{exe}}(\vec{x})$ is system execution time, T_{req} is execution time requirement, and $P_s(t)$ is system power supply changing over time.

Problems $\mathcal{P}_1$ and $\mathcal{P}_2$ are two formulations to show a performance–energy consumption tradeoff. Although they are related to performing certain tasks in a given time at lowest energy possible, they target different applications. The problem $\mathcal{P}_1$ implies that a system can run slowly as long as the execution time requirement is met to reduce power consumption, whereas the problem $\mathcal{P}_2$ means that the system can run as fast as possible, while the power consumption constraint is not violated. The system with faster computation speed can provide higher quality of service (QoS). This is particularly important in development wireless sensor networks for video and image sensing applications [21]. Given sufficient energy, the system can provide high-quality signal processing. This was not concerned in traditional low power designs as in $\mathcal{P}_1$. $\mathcal{P}_2$ presents the characteristics of the power-adaptive design. An intelligent control that adapts the system performance to the transient power supply constraint is required in a power-adaptive computing system.

B. Adaptive Power Management

Various dynamic approaches for system-level power management have been reviewed in [22]. The basic idea of these approaches is to switch the system to different operation modes according to different workloads. Different operation modes tradeoff power for speed. A recent development for determining the optimal switching rules for nonstationary workloads was reported in [23]. The target applications of these approaches are those with varying run-time workloads. In this paper, the target of the power-adaptive computing systems is the application scenarios, where the power supply varies over time. Although both varying workloads and power supply can be regarded as a net power change, in fact they are quite different. With the former, the system can choose not to change the operation modes and it might result in energy waste or delay in computation. In contrast with the latter, the system has to change the operation modes, otherwise the system might generate errors and break down eventually. For example, the harvested solar energy decreases dramatically in the future time slots. Therefore, the systems with varying power supply have a more demanding design requirement.

Works on adaptive power management in energy harvesting systems were presented in [2], [5], and [11]–[15]. There are two key elements for the adaptive power management: power modulation technique and power adaptation strategy. While the power adaptation strategies determine when and in which mode a system should work according to the harvested energy, the power modulation techniques regulate the system power consumption so that a system can work in multiple power modes.

Power modulation techniques such as duty cycling, dynamic voltage and frequency scaling (DVFS), power gating, and clock gating are widely used. Duty cycling changes the active time of the system components to adjust the power consumption. References [2], [5], [11], [12], and [24] used linear programming to determine the duty cycle of wireless sensors to adjust the data transmission rate. The DVFS technique controls system power consumption by adjusting the supply voltage and frequency. Zhang *et al.* [14] formulated the process of voltage and frequency selection and used enumeration to find the appropriate solution with respect to the system characteristics. Liu *et al.* [13] arranged tasks according to priority, so that high-priority tasks execute with the required voltage and frequency. Through enabling/disabling functional units and their clock sources, clock gating technique [25] regulates the system power consumption. These three techniques affect only the system dynamic power profile. The power gating technique modulates both the dynamic and static power by powering ON/OFF parts of the systems [26]. According to [22], DVFS and power gating introduce time delay between power mode transitions. In addition, DVFS has a limited dynamic range due to the system operational restrictions on clock frequencies and supply voltages. Power gating implemented on the top of clock gating can further increase the power modulation range, leading to more effective power-adaptive approach. However, the combined scheme requires further effort on new implementation and design of complicated control. These will be our future work.

The power adaptation strategies are generally divided into two categories: the rule-based and numerical model-based strategies. The former switches the system operation modes according to predefined rules, which specify an operation mode for a power supply range. This strategy is simple to implement, but it is often unable to reach the optimal condition for general cases [23], e.g., the mode transition costs cannot be ignored or the system operation modes are ambiguous. The latter formulates the decision process as optimization problems, such as linear programming [2], [5], [11], [12], and solves the problems in real time to find the optimal adaptation schemes.

We exploit the clock gating technique and the model-based strategy to manage the power consumption of the developed computing system. The process of determining the clock gating schemes is formulated as a geometric programming optimization problem by taking advantage of convex optimization techniques [27]. This dynamic management can adjust the number of active PUs of the computing unit in a time slot and, thus, modulates the peak power consumption more effectively.

III. Power Adaptive Computing System Design

The power-adaptive computing system is composed of three components: the computing unit, the system controller, and the energy estimator (Fig. 1). This section presents the proposed approaches for designing these three system components.

A. Computing Unit

The computing unit presented in this paper is capable of processing data with customized arithmetic functional units in hardware. We applied several design optimization techniques [18]–[20], including data reuse, loop pipelining and loop parallelization, and explore design space to determine an single-process multiple-data (SPMD) computation structure, as shown in Fig. 2.

In the computing structure, system operations were divided into three steps: data input, computation, and data output. In the data input step, operand data processed by all PUs were loaded into corresponding on-chip memories from the off-chip global memory. Each datum was loaded only once. In the computation step, all PUs with the same functionality processed different data in parallel. The structure of PUs was customized for the target applications, and operations in each PU were pipelined. In the output step, computation results were transferred back to the off-chip memory. All three steps were pipelined. Therefore, the computing structure achieved

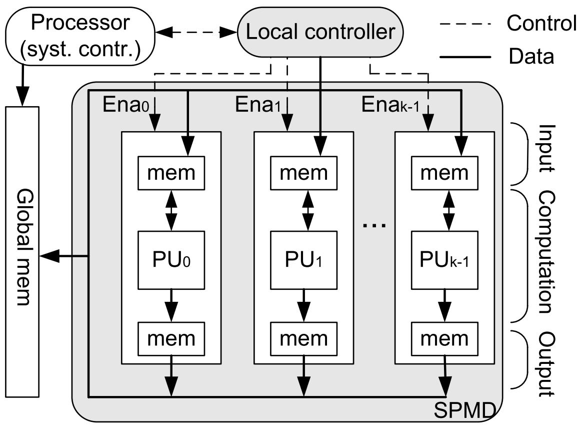


Fig. 2. Power adaptive enabled computing system.

hierarchical parallelism: global pipelining, local parallelization, and local pipelining, for a low clock frequency and high parallel computations.

The parallel computing structure was designed based on the decision made by the following optimization problem:

$$\begin{aligned} \min \quad & T(\vec{\rho}, \vec{k}, ii) \\ \text{s.t.} \quad & P(\vec{\rho}, \vec{k}, ii) \le P_{s_{\text{peak}}} \\ & R_{\text{mem}}(\vec{\rho}, \vec{k}) \le \text{Res}_{\text{ram}} \\ & R_{\text{comp}}(\vec{k}, ii) \le \text{Res}_{\text{comp}} \end{aligned} \qquad (\mathcal{P}_3)$$

where the system execution time T is minimized under the constraints of the peak power $P_{s_{\text{peak}}}$ of harvested energy, and the availability of memory resources Res_{ram} and computational resources Res_{comp} in hardware. The execution time model T, the power consumption model P, the memory resource utilization model R_{mem}, and the computational resource utilization model R_{comp} will be defined later. $\vec{\rho}$ is a data reuse variable vector determining the local memory space for each processing unit. $\vec{k} = (k_0, k_1, \ldots, k_{N-1})$ is a loop parallelization variable vector indicating k_l iterations of loop l in a N-level nested loop structure were executed in parallel. ii is the number of clock cycles of the local pipelining initiation interval in PUs.

Obviously, the model $\mathcal{P}_3$ is a multiobjective optimization problem. We refined $\mathcal{P}_3$ based on the geometric program modeling technique [28]. The geometric program is the following optimization problem:

$$\begin{aligned} \min: \ & f_0(x) \\ \text{s.t.} \ & f_i(x) \le 1, \quad i = 1, \ldots, m \\ & h_i(x) = 1, \quad i = 1, \ldots, p \end{aligned}$$

where the objective function and inequality constraint functions are all in posynomial form, while the equality constraint functions are monomial. The geometric program has the feature of convexity and has efficient solvers for the globally optimal solution [28]. In [18], the geometric program has been applied to data reuse and parallelism co-optimization.

Extended from [18]–[20], the problem $\mathcal{P}_3$ was refined as follows. Unlike [19] where low power is the optimization objective, the objective in $\mathcal{P}_3$ is to design a parallel computing structure used in the solar energy harvesting environment. In addition, the peak power of the harvested solar energy, which was not preset in [20], was considered to ensure the designed computing unit work reliably in that environment. Furthermore, in [18] and [19], data input/output and computation were not parallelized, while here the execution of the parallel computing system is a pipelining of data input, computation, and data output. Therefore, the execution time is defined as

$$\begin{aligned} T(\vec{\rho}, \vec{k}, ii) = {} & (v-1) \times t + T_{\text{in}}(\vec{\rho}, \vec{k}) \\ & + T_{\text{comp}}(ii) + T_{\text{out}}(\vec{\rho}, \vec{k}) \end{aligned} \qquad (1)$$

$$t = \max(T_{\text{in}}(\vec{\rho}, \vec{k}), T_{\text{comp}}(ii), T_{\text{out}}(\vec{\rho}, \vec{k})) \qquad (2)$$

where T_{in}, T_{comp}, and T_{out} are respectively data input time, computation time, and data output time in the number of execution cycles. Among them, the maximum determines the global pipelining initiation interval t. v is the total number of times the three steps have to execute.

Each PU performs only arithmetic computations and could contain adders/subtracters, multipliers or comparators for the target applications, and only accesses its own local memory. All operations are pipelined with initiation interval ii. Local memory bandwidth, data dependence, and computation resources are the factors that determine ii and thus $T_{\text{comp}}(ii)$ [19].

T_{in} and T_{out} are the functions of variables $\vec{\rho}$ and $\vec{k}$. In practice, larger local memory size and more PUs mean more data transferred between off-chip and on-chip memories and thus longer input and output time. T_{in}, T_{comp}, and T_{out} will be deliberated in the next section for different applications.

The on-chip memory resource constraint is

$$K = \prod_{l=1}^{N} k_l \qquad (3)$$

$$R_{\text{mem}}(\vec{\rho}, \vec{k}) = K \times \sum_{i=0}^{R-1} \prod_{j=0}^{E_i - 1} \rho_{ij}^{\log_2 B_{ij}} \qquad (4)$$

where K is the total number of homogeneous PUs; R is the number of array references in a target application; E_i is the number of data reuse options of array reference i; B_{ij} is the size of on-chip memory required for data reuse option j of array i. ρ_{ij} takes values 1 or 2; taking 2 means data reuse option j is chosen for array i, otherwise 1. The details about data reuse options were presented in [18].

The computation resource constraint is defined as follows:

$$R_{\text{comp}}(\vec{k}, ii) = K \times r(ii) \qquad (5)$$

where $r(ii)$ is the number of computational resources used by a PU, which is related to the pipeline scheduling [19]. Usually, the shorter the pipelining interval is, the more the concurrent working of the computational resources is required.

Finally, the system power model was extended from [19] to model the power consumption of the computing unit and the global memory as follows:

$$P = (p_1 R_{\text{mem}} + p_2 R_{\text{comp}} + p_3)F + p_4 \frac{(T_{\text{in}} + T_{\text{out}})}{C_g} \qquad (6)$$

where the first bracket includes the power consumed by the local memory resources, the computational resources, and other control logic of the computing unit under clock frequency F; the second bracket is the power consumption of the global memory caused by data transfers, where C_g is the average number of cycles per global memory access.

The optimization problem $\mathcal{P}_3$ determines the computation structure with the peak speed. However, this peak speed is only achievable when the harvested solar energy reaches the peak value. If the harvested power is not sufficient to support all K PUs working, the system has to stop computation. To prolong working time, a number, less than K, of PUs can be designed, to trade speed for working time. The structure determined at design time is a static design and in Section V, we will show that the static designs have low energy efficiency. In the next section, behaviors of the computing system are dynamically adjusted corresponding to the power supply.

B. System Controller

The run-time adaptability of the parallel computing structure in this paper is realized by adjusting the number of PUs concurrently running, given that each PU can be clock gated independently. At run time, the task of the system controller is to select the most power-efficient clock gating scheme and configure the computing unit, so that the system does not consume more than the supplied power during a particular time slot. We exploited a numerical model built on $\mathcal{P}_3$ to determine the adaptation scheme.

Since the structure of each PU is fixed and is just clock gated ON/OFF, there is only one variable k now, which is the number of PUs gated ON. As a result, in $\mathcal{P}_3$, the resource constraints can be removed and T_{comp} becomes a constant. In addition, the instantly harvested solar energy should be considered now. At time slot i, the power consumption $P_c(i,k)$ should not be larger than the harvested energy $P_s(i)$ and the residual energy in the energy buffer after time slot $i-1$ $E_b(i-1)$, to achieve energy neutral [5]. Therefore, the optimization model simplified from $\mathcal{P}_3$ is

$$\begin{aligned}
&\min T(k) = (v-1)\times t + T_{\text{in}}(k) + T_{\text{comp}} + T_{\text{out}}(k)\\
&\quad \text{s.t.} \max\{T_{\text{in}}(k), T_{\text{comp}}, T_{\text{out}}(k)\} \le t\\
&\tau P_c(i,k) \le \tau P_s(i) + E_b(i-1) \qquad\qquad (\mathcal{P}_4)\\
&\qquad T(k) \le \tau\\
&\qquad 1 \le k \le K
\end{aligned}$$

where the execution time $T(K)$ is required to be not larger than the period τ of a time slot, ensuring the workloads assigned to the PUs be finished within that slot. This simplifies the operation mode of the adaptation and avoids context store and resume which would happen when the computation of workloads is across time slots. The period τ of a time slot is equal to the sampling period of solar energy.

We adopted an empirical power model [20], which is simple and accurate enough to make sure the computation system works correctly. For the power-adaptive computing system proposed in this paper, the system power consumption variation with different clock gating schemes can be expressed as

$$P_c(k) = P_{\text{const}} + k \times P_{\text{pu}} \tag{7}$$

where P_{pu} is the power consumption of a PU when it is running, and P_{const} contains the power consumption from the system controller, the energy estimator, and the local controller of the computing unit.

The ratio $\sigma = K \times P_{\text{pu}}/(P_{\text{const}} + K \times P_{\text{pu}})$ determines the dynamic range of the system power consumption. For implementing applications in hardware that has large P_{const}, small K, and simple PUs, the dynamic range could be considered small. The small dynamic range will limit the benefit of the proposed power-adaptive management approach. The percentage of energy efficiency improvement of the power-adaptive design over the static design with the peak speed could be computed as $((1-\sigma)+1)n_1/n_2$, where n_1 is the time taken by the power supply increasing from $P_{\text{const}} + P_{\text{pu}}$ to $P_{\text{const}} + K \times P_{\text{pu}}$ and n_2 is the time period when the power supply is greater than or equal to $P_{\text{const}} + K \times P_{\text{pu}}$. This energy efficiency improvement formula could give guidance on how to apply the proposed power-adaptive approach for applications and associated hardware platforms.

The system controller solves the optimization model $\mathcal{P}_4$ in real time to determine that k PUs can be enabled in time slot i, such that the system speed is maximized while the system power consumption is not more than the supplied power. The advantage of the model-based approach is the optimality of the obtained power adaptation solution. In addition, the optimization model can be easily extended to target different objectives and include more constraints, such as various QoS constraints, to satisfy different levels of requirements.

The overhead of the run-time adaptation includes the execution time and the power consumption of the system controller. The execution time of the controller is the time required to solve the optimization model $\mathcal{P}_4$. The measured solution time on our experiment platform is 0.3 s. The time overhead would not affect the system performance as long as the solution time is shorter than the time slot τ. The time slot usually is related to the sample period of the power supply, which depends on the change rate of the power source. If the change rate is quick, then the time slot could be increased to include multiple power samples, i.e., the adaptation resolution decreases. The power consumption overhead, however, is unavoidable. The measured power consumption for solving $\mathcal{P}_4$ on our experimental platform is 0.5 W.

C. Solar Energy Estimator

The run-time optimization approach, presented in the previous section, has to use the estimation of harvested solar energy to determine the adaptation scheme for the next time slot, because the harvested energy cannot be known for the future time slots. Therefore, the energy estimator is very important and is related to the robustness of the energy harvesting powered systems [2].

In this paper, we exploit a three-layer multilayer perceptron (MLP) neural network, as shown in Fig. 3, to predict the harvested solar energy $\bar{P}_s(i)$. The input layer has three nodes

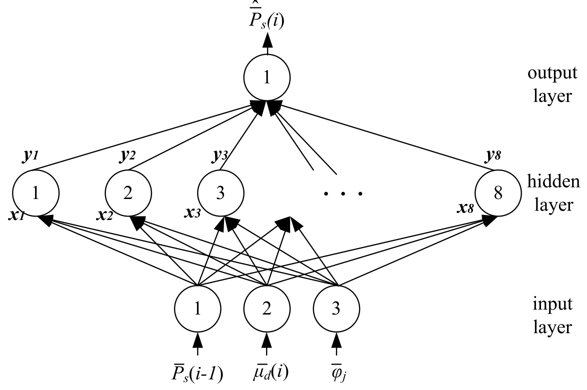


Fig. 3. Three-layer MLP neural network for solar energy prediction. The inputs are the average power in the previous time slot $\bar{P}_s(i-1)$, the average of power at the next time slot in the past days $\bar{\mu}_d(i)$ and the weather comparison between the current day and the previous days $\bar{\phi}_j$. The output $\hat{\bar{P}}_s(i)$ is the predicted average power for the next time slot.

corresponding to three input parameters: $\bar{P}_s(i-1)$, $\bar{\mu}_d(i)$, and $\bar{\phi}_j$. $\bar{P}_s(i-1)$ is the average power measured at the previous time slot $i-1$, $\bar{\mu}_d(i)$ is the average of power measured at time slot i in the past d, days and $\bar{\phi}_j$ is a weighted average of the ratios between the measured power at previous j time slots before time slot i and the average of power measured at the corresponding time slots in the past d days. These parameters similar to the parameters in [29], but we consider average power, because the energy harvested during a time slot is $\bar{P} \times \tau$.

The hidden layer contains eight neurons and the output layer has one node, which is the prediction of solar power values. The input to each neuron at the hidden layer is a weighted sum of the input parameters, and each hidden neuron contains a sigmoid activation function $y = 1/(1+e^{-x})$. The final prediction from the output neuron is the weighted sum of outputs from the hidden neurons.

The three-layer MLP needs to be trained over a set of training data $\{\bar{P}_s(i-1), \bar{\mu}_d(i), \bar{\phi}_j, \bar{P}_s(i)\}$, so that it can learn the relationship between the three inputs and the output $\bar{P}_s(i)$. Based on the 90-day solar energy data from the ORNL website [30], the trained three-layer MLP neural network achieves the average estimation error within 2.45%. The feed-forward computation of the three-layer MLP with eight hidden neurons is just about 0.95 ms for predicting once and can be ignored, compared with the solar energy sampling interval. Fig. 4 shows the performance of the MLP for predicting the harvested energy over 3 days with different weather conditions. The small power estimation error can be complemented by the energy buffer in the system, as shown in Fig. 1, and the $E_b(i-1)$ considered in $\mathcal{P}_4$. However, if the weather changes quickly, for example, the last day in Fig. 4, and the estimator overestimates the solar energy, then the energy buffer may empty quickly. This will cause the computing system to produce wrong results or even breakdown [2]. To avoid this situation, a threshold for the remaining energy in the buffer is set. When the remaining energy is lower than the threshold, the computing system comes into idle. This setting will be described in more details in the next section.

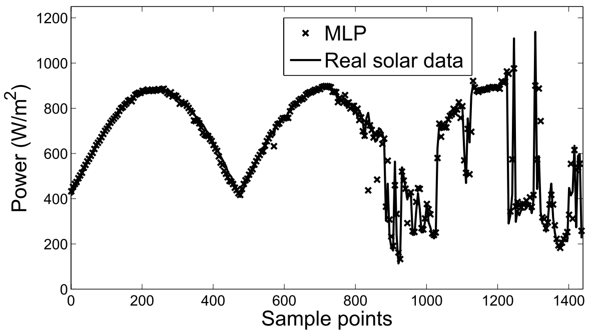


Fig. 4. Estimated solar energy versus real solar energy for 3 days with different weather conditions. Solar energy of each day from 8 A.M. to 4 P.M. is sampled for every 1 min.

So far, we have presented our approach for designing a power-adaptive computing system. The parallel computing unit is designed based on the geometric programming offline optimization and enables clock gating for power adaptation. The optimization problem is simplified and is implemented in the system controller to determine clock gating schemes in real time. Finally, an accurate solar energy estimator is built on a three-layer MLP neural network. In the next section, we will illustrate the approach by applying it to real applications, showing a simulation framework of the solar-energy-powered computing system, optimization model instantiation, and the related design space.

IV. Case Study

We developed power-adaptive computing systems for applications shown in Table I: matrix multiplication (MAT), k-means clustering (k-means), Sobel edge detection (Sobel), 1-d correlation, and n-body simulation (n-body), following the proposed design approach. The developed power-adaptive computing systems can be applied to distributed monitor networks to provide computation capability at sensory nodes. For example, Sobel extracts edges of subjects in images, then k-means that it classifies subjects in terms of their edge properties, and the data of interested subjects are transferred to the network at the end.

A. Simulation Framework

As mentioned earlier, this paper focuses on power-adaptive computing system design to improve the power consumption efficiency. Therefore, to evaluate the proposed approach, we have simulated an energy harvesting system with the following assumptions.

The real-time solar energy data from ORNL website [30] from 5 A.M. to 6 P.M. in August and September, in 2011, are used to simulate the harvested power, which is sampled every 1 min, i.e., $\tau = 1$ min. The data are on the range of 0–1125 W/m^2 and are scaled down as harvested in a 10 cm $\times$ 10 cm solar cell panel to power the target hardware platform, which consumes power on the range of 2.4–9.5 W. Here, we only consider the power consumption of the power-adaptive computing system. When considering the other components of the system in Fig. 1, the energy data can be scaled similarly.

TABLE I
BENCHMARKS

Application	Description	Parallel computing structure determined by $\mathcal{P}_3$
MAT	matrix size 1024×1024, 3 loop levels, single precision floating-point number.	94 fully pipelined PUs.
k-means	border detection and object recognition, 10^5 64-dimensional vectors partitioned into 128 clusters in 10 heuristic iterations, 4 loop levels, single precision floating-point number.	96 fully pipelined PUs.
Sobel	object edge detection in images, two 3×3 mask windows moving over an image, 4 loop levels, image size 288×352 pixels, 8-bit integer number.	143 fully pipelined PUs.
1-d correlation	moving one of two vectors over every elements of the other vector and computing sum of products, 2 loop levels, vector size 256 and 1024, single precision floating-point number.	96 fully pipelined PUs.
n-body	studying interactions between objects by calculating the total force acting on each individual particle, 3 loop levels, single precision floating-point number.	10 fully pipelined PUs.

The energy storage module is assumed to act as a buffer for two purposes: 1) to complement the inaccuracy of the energy estimator, i.e., when the predicted power is higher than the real harvested power, the excessive power comes from the energy storage [15] and 2) to keep the controller alive when there is not enough harvested energy, e.g., during night or terrible weather, and also provide certain level of QoS. Therefore, there is a bottom line for the stored energy in the energy buffer [15]. If the remaining energy in the buffer reaches to the bottom line, then the computing system is idle in the next time slot. The bottom line of the energy buffer is assumed to be the amount of energy to keep the controller alive until the next energy charge period. During daytime, the residual power between the power consumption and the power supply is stored in the buffer.

The proposed power-adaptive approach can use both field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) to implement the parallel computing system. According to [31], FPGAs with embedded multipliers and block memories require on average 7 times more power compared with ASICs, for the same circuit design. However, the increased nonrecurring engineering cost makes low to midvolume ASIC production unaffordable [32]. In addition, systems based on energy harvesting usually are deployed in the wide and remote areas. The costs for system maintenance and update also need to be considered. Therefore, in practice, the hardware implementation platform should be decided depending on the scale of applications and the volume of the power supply. This paper uses an FPGA-based platform to demonstrate the application of the proposed approach.

Specifically, the developed power-adaptive system is implemented on an experimental platform with an ARM processor and an FPGA. The parallel computing system is mapped onto a Virtex5-330t FPGA. The measured peak power consumption of the FPGA implementations of the five applications at 100 MHz is about 7.3 W, and when idle, the FPGA is reconfigured with a blank configuration to make the power consumption negligible. The system controller and energy estimator are implemented on the ARM processor and the measured peak power consumption of the ARM processor is about 2.4 W. This power overhead can be reduced if an FPGA platform with embedded processor is used for the implementation.

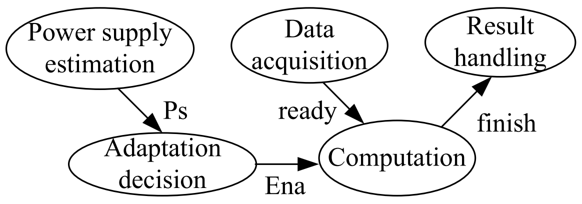


Fig. 5. Function blocks of the power-adaptive computing system.

The relevant function blocks of the energy harvesting system are shown in Fig. 5 and the system works as follows. Sensed data are stored in the global memories. The estimator estimates the supply power for the next 1 min, and the controller determines a proper clock gating scheme based on the estimation. Both estimation and the control decision have to be made in 1 min before the next time slot starts. After that, the system controller sends clock gating signals (`Ena`) to the local controller to configure the parallel computing system. When the computing system completes the task, the local controller sends a finish signal back to the system controller and the latter handles results.

Remember that the computing system should finish the assigned task within a time slot defined in $\mathcal{P}_4$. In Section IV, solar energy is sampled every 1 min, while processing one data set for the applications completes within a few seconds.

B. Approach Instantiation

In this section, we demonstrate how to follow the proposed approach to develop a power-adaptive computing system for MAT.

At design time, a parallel hardware design of MAT, determined by the optimization problem $\mathcal{P}_3$ on the target FPGA, is $k_1 = 94, k_2 = 1, k_3 = 1$ and $ii = 1$. In this design, 94 iterations of the outermost loop of MAT are executed in 94 PUs in parallel, where one PU performs the multiplication of one row of two input matrices and generates one row of resultant matrix; the innermost loop of MAT is fully pipelined. This design can obtain the final resultant matrix in 0.36 s at 100-MHz clock frequency, including the data transfer time between the global and local memories.

Once we implement the design on the FPGA, we could measure the power consumption of the computation system. Experimenting with several clock gating schemes, the fitted

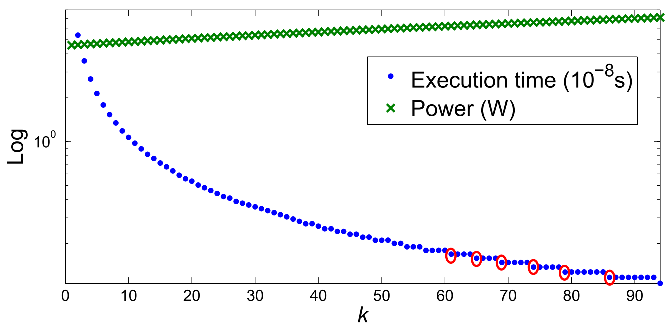


Fig. 6. MAT run-time optimization design space. Each circle indicates the design consuming the least power while achieving the same speed as those in its right for a certain range of k.

power model for MAT is

$$P_c(k) = 6.96 + 0.027k. \tag{8}$$

For the run-time optimization model, three parts of the execution time in the number of clock cycles are refined as

$$T_{\text{in}}(k) = k \times numCol \tag{9}$$
$$T_{\text{comp}} = lengthPipe \times numCol \tag{10}$$
$$T_{\text{out}}(k) = k \times numCol \tag{11}$$

where $numCol$ is the number of columns of the matrix and $lengthPipe$ is the scheduled length of pipelining the innermost loop. Bringing these functions into problem $\mathcal{P}_4$ instantiates the optimization problem for MAT. Moreover, since the structure of the PU is fixed, $lengthPipe$ is known at this stage; it is 1024 clock cycles for pipelining the innermost loop of MAT. This value is much larger than the upper bound of k, that is, $K = 94$. Therefore, the run-time optimization problem for MAT is

$$\min \quad T(k) = \upsilon \times T_{\text{comp}} + T_{\text{in}}(k) + T_{\text{out}}(k) \tag{12}$$
$$\text{subject to} \quad \tau P_c(i,k) \le \tau P_s(i) + E_b(i-1) \tag{13}$$
$$T(k) \le \tau \tag{14}$$
$$1 \le k \le 94 \tag{15}$$
$$numCol \times k^{-1} \times \upsilon^{-1} \le 1. \tag{16}$$

This is a geometric programming model and thus can quickly converge to a globally optimal solution k, given the supplied power $P_s(i)$.

The design space of this optimization problem is shown in Fig. 6, where we can observe that as k increases the power consumption keeps increasing while the execution time decreases and sometimes holds constant. The reason for the nondecreased execution time when k increases to some values is that the execution time shown in (12) is dominated by the first addend. For certain values of k, the integer variable υ, which is derived in (16), does not change. This leads to designs that have the same speed but the different power consumption. The target of the optimization problem is to find the design with the fastest speed and the least power consumption given the supplied power, as circled in Fig. 6.

We followed the same procedure to develop the similar computing systems for k-means clustering, Sobel, 1-d correlation, and n-body simulation as well with different instantiations of problems $\mathcal{P}_3$ and $\mathcal{P}_4$. The number of parallel PUs for each benchmark is also shown in Table I.

V. Experimental Results

The results shown in the previous section were obtained after synthesis, placement and routing, and mapping onto the hardware platform. The power consumption of the FPGA was obtained by monitoring the current flowing over a current sense resistor on the FPGA card. The power consumption of the ARM processor was measured by placing a ac/dc current clamp at the power supply. With these experimental data and the instantiated execution time and power models, as presented in the previous section, the evaluation of the proposed approach is presented in this section.

For each application, four parallel computing system designs listed below are implemented and compared.

1) Static design with the maximum number of PUs (S_1), determined by our proposed model $\mathcal{P}_3$.
2) Static design with the moderate number of PUs (S_2), obtained from the previous static design by reducing the number of implemented PUs, trading speed for power consumption.
3) Power-adaptive design following the rule-based strategy (RB design) [23]. In the rule-based strategy, the power adaptability of the parallel computing system is realized as follows. The operation power range $[P_{c_l}, P_{c_u}]$ of the computing system is partitioned into several segments. Based on (7), a segment i is associated with a rule, that is, if $P_{c_i} \le P_s \le P_{c_{i+1}}$, then the clock gating scheme $k = \lfloor P_{c_i} - P_{const}/P_{pu} \rfloor$. Each rule, determined at design time, specifies a power range and a corresponding clock gating scheme. The rules are put in a lookup table and are switched at run time according to P_s.
4) Power-adaptive design following the proposed convex model based strategy (CMB design). The clock gating scheme of the parallel computing system is determined by solving $\mathcal{P}_4$ at run time.

The comparison criteria of the four designs is the harvested energy utilization efficiency

$$\text{Energy utilization efficiency} = \frac{E_c}{E_s} \tag{17}$$

where E_s is the harvested energy supplied to the power-adaptive computing system and E_c is the energy consumed by the power-adaptive computing system. An ideal situation is the energy utilization efficiency equal to one, i.e., all supplied energy is utilized without waste.

The power consumption behavior of the four designs of MAT, k-means, and Sobel are shown in Figs. 7, 9, and 10, respectively, where the dashed line represents the variation of the power supply and the solid line represents the power consumption. The results are shown based on a 3-day duration: day 1 was sunny, day 2 was a day with sunny morning and cloudy afternoon, and day 3 was cloudy.

Figs. 7(a) and 9(a) show the first static design S_1 of MAT and K-means clustering determined by our proposed model $\mathcal{P}_3$. These fast designs with maximum number of PUs

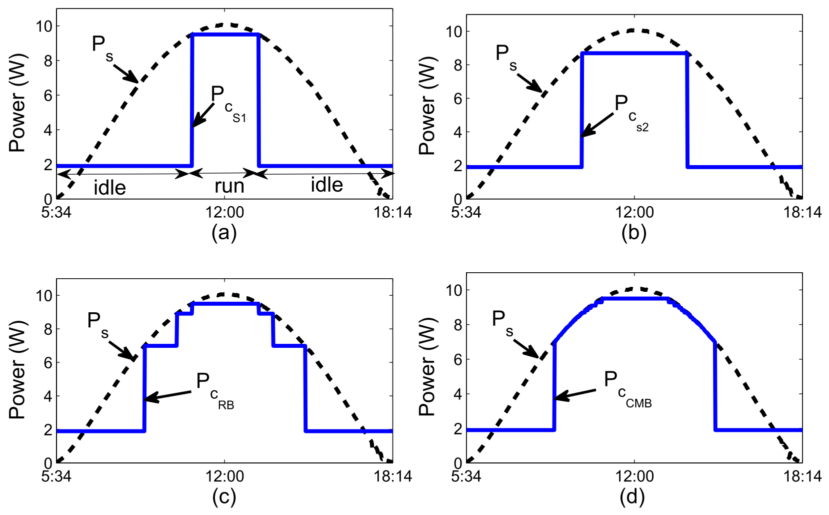


Fig. 7. Power variation of MAT at day 1. P_s: supplied power and P_c: power consumption. (a) Static design S_1 with 94 PUs. (b) Static design S_2 with 64 PUs. (c) Rule-based adaptive design. (d) Convex-model-based adaptive design.

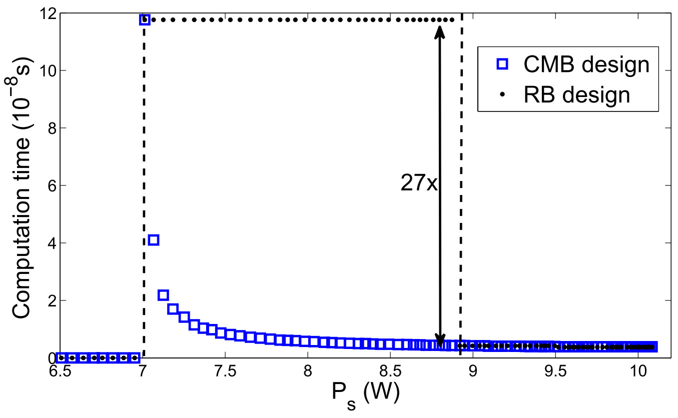


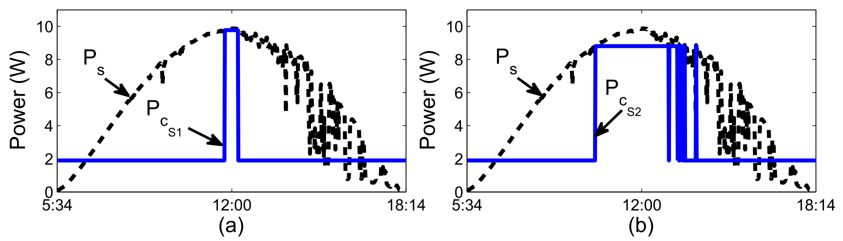
Fig. 8. Computation speed comparison between the RB and CMB designs for MAT. There are three zones, where in the middle zone the CMB designs are faster than the RB designs and in the left and right zones designs from both approaches achieve the same speed.

available require high power and thus can only work for a short period of time. The second static design S$_2$ with fewer number of PUs for MAT and K-means clustering is shown in Figs. 7(b) and 9(b), respectively. This design with lower power requirement can run earlier, longer but slower than the previous design. These two static designs only work when the constant power requirement is met. Otherwise, the SPMD computing system will be idle (Figs. 7 and 9) and only the ARM processor will be available to provide a certain level of computation QoS. The ARM processor used has a low computing capability. The results measured for MAT show that the ARM processor can only perform 0.78 million floating point operations per second per watt, while the SPMD computing system with 94 PUs can perform at about 1.1 GFLOPS/W. Therefore, keeping the SPMD system working as long and as fast as possible should be the design goal.

Aiming at this goal, the power-adaptive design with clock gating schemes determined by the rule-based run-time strategy described in Section III-B for MAT and K-means are shown in Figs. 7(c) and 9(c), respectively. The SPMD system operation power range is partitioned into three segments, thus three working modes in the system are represented as three power levels in Figs. 7 and 9. For K-means, due to the fact that the power supply changes more significantly in the afternoon of day 2, the computing system falls into the idle state several times, as shown in Fig. 9(c). Compared with the previous two static designs, this RB design works longer. However, the discretization results in inefficient usage of the supplied power, represented by the gaps between the power supply line and the power consumption line in Figs. 7(c) and 9(c).

The CMB designs for MAT and K-means are shown in Figs. 7(d) and 9(d), respectively. The energy utilization used as metric in this paper is actually the area outlined by the solid line and the x-axis in Figs. 7 and 9. In Figs. 7 and 9, it is clear that the areas outlined in Figs. 7(d) and 9(d) is larger than the areas in the other three subfigures (i.e., the energy utilization efficiency of the CMB design is the highest under the same power supply). Compared with the static design S_1, the CMB design works in a longer duration in Figs. 7(a) and 9(a). Compared with the static design S_2 in Figs. 7(b) and 9(b), the CMB design works for a longer duration and at a higher speed with all PUs running when the power supply is sufficient. Compared with the RB design, the CMB design has a smaller gap with the power supply line.

As mentioned before, the power adaptability stems from the adaptive computing behavior. This can be shown in Fig. 8. Designs obtained by using the RB and CMB strategies are shown. Fig. 8 could be divided into three zones. There is not enough power supply in the left zone, so the computing system remains idle. Starting from the left edge of the middle zone, as the power constraint P_s increases, more and more PUs of the MAT parallel computing structure are switched on and thus the computation time decreases. In the middle zone, due to the real-time optimization and the fine control, the CMB designs are up to 27 times faster than the RM designs. In the right zone, designs from both strategies are working at the full speed, and the markers representing both types of design are overlapped.



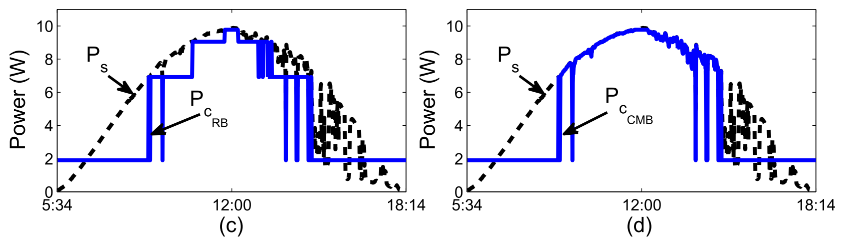
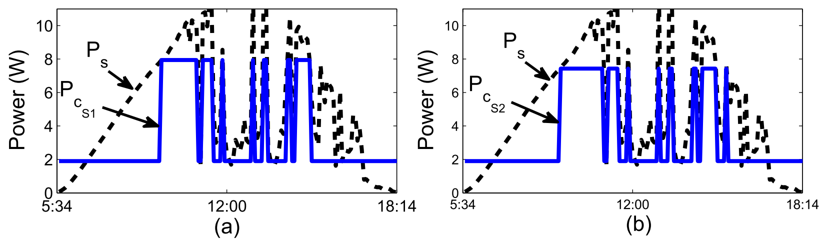


Fig. 9. Power variation of k-means clustering at day 2. P_s: supplied power and P_c: power consumption. (a) Static design S_1 with 96 PUs. (b) Static design S_2 with 64 PUs. (c) Rule-based adaptive design. (d) Convex-model-based adaptive design.



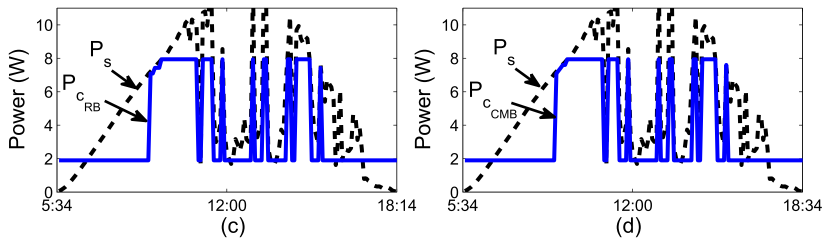


Fig. 10. Power variation of Sobel edge detection at day 3. P_s: supplied power and P_c: power consumption. (a) Static design S_1 with 143 PUs. (b) Static design S_2 with 74 PUs. (c) Rule-based adaptive design. (d) Convex-model-based adaptive design.

Unlike Figs. 7 and 9, the four designs of Sobel in Fig. 10 have similar power consumption variations. This is partially due to the fact that the structure of PUs for Sobel edge detection is simple, which involves only addition/subtraction and shift, and as a result variations of the system power consumption are insignificant for most of the clock gating schemes. Another reason is due to the serious variation of the power supply.

The energy utilization efficiencies and their comparisons of all designs in 3 days are summarized in Table II. From column three to column six, the energy utilization efficiency of the designs increases, and the CMB design is the highest (up to 96% and with an average of 85%). Over the five benchmarks, for MAT, k-means and n-body the CMB design has the higher efficiency than for Sobel and 1-d correlation due to the simpler structure of PUs in the Sobel edge detection and 1-d correlation algorithms. Over the 3-day period, the CMB design has the higher efficiency in day 2, which has several moderate fluctuations in power supply, and the lower efficiency in day 3, in which the power supply increases and decreases drastically as shown in Fig. 9.

Furthermore, as shown in Table II, compared with the static design with the maximum number of PUs (EE_{s1}), the CMB design improves the energy utilization efficiency up to 2.46 times with an average of 1.58 times in five benchmarks. Compared with the static design with fewer number of PUs (EE_{S2}), the CMB design improves the efficiency up to 1.55 times with an average of 1.30 times. When comparing with the RB design, the efficiency improvement of the CMB design is up to 1.09 times with an average of 1.05 times. If the RB strategy can enumerate all possible system states, the strategy will result in designs the same as CMB. This is only feasible when the number of system states is small. The Sobel benchmark shown in Table II belongs to this case. The advantage of using the rule-based adaptive strategy is the adaptability response time (time for comparing with different

TABLE II

ENERGY UTILITY EFFICIENCY (EE). EE_{s_1}: STATIC DESIGN s_1 OF THE FIVE BENCHMARKS WITH 94, 96, 143, 96, AND 10 PUs, RESPECTIVELY; EE_{s_2}: STATIC DESIGN s_2 OF THE FIVE BENCHMARKS WITH 64, 64, 72, 64, AND 4 PUs, RESPECTIVELY; EE_{rb}: RB DESIGN; AND EE_{cmb}: CMB DESIGN

	Application	EE_{s1}	EE_{s2}	EE_{rb}	EE_{cmb}	$\text{EE}_{cmb}/\text{EE}_{s1}$	$\text{EE}_{cmb}/\text{EE}_{s2}$	$\text{EE}_{cmb}/\text{EE}_{rb}$
day1	MAT	56%	67%	83%	86%	1.54×	1.28×	1.04×
	k-means	50%	65%	83%	88%	1.76×	1.55×	1.06×
	Sobel	71%	71%	78%	78%	1.10×	1.10×	1.00×
	1-d correlation	38%	61%	77%	81%	2.16×	1.34×	1.05×
	n-body	68%	67%	87%	93%	1.37×	1.39×	1.07×
day2	MAT	50%	67%	85%	90%	1.80×	1.34×	1.06×
	k-means	40%	65%	84%	91%	2.28×	1.40×	1.08×
	Sobel	76%	77%	82%	82%	1.08×	1.06×	1.00×
	1-d correlation	35%	56%	80%	85%	2.46×	1.50×	1.06×
	n-body	68%	69%	88%	96%	1.42×	1.38×	1.09×
day3	MAT	59%	64%	77%	80%	1.36×	1.25×	1.04×
	k-means	56%	63%	78%	81%	1.45×	1.29×	1.04×
	Sobel	68%	70%	74%	74%	1.09×	1.06×	1.00×
	1-d correlation	55%	61%	73%	75%	1.36×	1.24×	1.02×
	n-body	65%	71%	86%	92%	1.41×	1.30×	1.07×
Avg		57%	66%	81%	85%	1.58×	1.30×	1.05×

conditions and determining a mode), while the convex model-based strategy needs to solve a system of equations. If an adaptive system requires a quick response time, which cannot be met by the convex model-based strategy, the rule-based strategy is more promising.

VI. CONCLUSION

Here, we propose a thorough approach for designing and managing power-adaptive computing systems to provide computing ability for solar-energy-powered embedded systems. First, the geometric program modeling and optimization technique are exploited to design a computing unit that contains multiple parallel low speed and individually clock-gated processing units. Then, a system controller in run time solves a simplified geometric programming model to decide how many processing units are clock-gated ON, such that the power consumption of the computational system does not exceed the power supplied and the system computation speed is maximized. Third, to ensure the reliable power adaptation, an intelligent neural network is trained to estimate the incoming solar energy. The proposed approach has been evaluated by applying to five applications on an FPGA-based hardware platform. In a solar energy harvesting simulation environment, our power-adaptive designs can improve the harvested energy utilization efficiency by 2.46 times compared with the static and the rule-based adaptive designs.

The limitations of this paper and the potential improvements in the future are the following.

1) The current approach and platform only support the clock gating technique, which may not be effective enough for applications with simple arithmetic architecture, like Sobel edge detection algorithm. In the future, we will consider to integrate other dynamic power modulation techniques, e.g., power gating and DVFS, to increase the power dynamic range.
2) The current approach sets a threshold for the energy buffer to shut down the system when energy buffer is nearly empty due to energy overestimation. In the future, we will adjust the estimator using a feedback based on the estimation error and the buffer residual, to prolong the working status of the computing system.
3) The current approach focuses on improving utilization of harvested energy in the computing system. In the future, we will investigate how the efficiency of the other components of the energy harvesting system is related to the energy usage, and will try to improve the efficiency of the whole system while maintaining QoS.

REFERENCES

[1] V. Raghunathan and P. H. Chou, “Design and power management of energy harvesting embedded systems,” in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Oct. 2006, pp. 369–374.

[2] C. Moser, L. Thiele, D. Brunelli, and L. Benini, “Adaptive power management for environmentally powered systems,” *IEEE Trans. Comput.*, vol. 59, no. 4, pp. 478–491, Apr. 2010.

[3] C. Lu, V. Raghunathan, and K. Roy, “Micro-scale energy harvesting: A system design perspective,” in *Proc. 15th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2010, pp. 89–94.

[4] A. Khaligh, P. Zeng, and C. Zheng, “Kinetic energy harvesting using piezoelectric and electromagnetic technologies—State of the art,” *IEEE Trans. Ind. Electron.*, vol. 57, no. 3, pp. 850–860, Mar. 2010.

[5] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, “Power management in energy harvesting sensor networks,” *ACM Trans. Embedded Comput. Syst.*, vol. 6, no. 4, p. 32, Sep. 2007.

[6] D. Dondi, A. Bertacchini, D. Brunelli, L. Larcher, and L. Benini, “Modeling and optimization of a solar energy harvester system for self-powered wireless sensor networks,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 7, pp. 2759–2766, Jul. 2008.

[7] D. Nguyen and B. Lehman, “An adaptive solar photovoltaic array using model-based reconfiguration algorithm,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 7, pp. 2644–2654, Jul. 2008.

[8] O. Lopez-Lapena, M. T. Penella, and M. Gasulla, “A new MPPT method for low-power solar energy harvesting,” *IEEE Trans. Ind. Electron.*, vol. 57, no. 9, pp. 3129–3138, Sep. 2010.

[9] J. L. Agorreta, L. Reinaldos, R. Gonzalez, M. Borrega, J. Balda, and L. Marroyo, "Fuzzy switching technique applied to PWM boost converter operating in mixed conduction mode for PV systems," *IEEE Trans. Ind. Electron.*, vol. 56, no. 11, pp. 4363–4373, Nov. 2009.

[10] F. I. Simjee and P. H. Chou, "Efficient charging of supercapacitors for extended lifetime of wireless sensor nodes," *IEEE Trans. Power Electron.*, vol. 23, no. 3, pp. 1526–1536, May 2008.

[11] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Adaptive power management in energy harvesting systems," in *Proc. DATE*, Apr. 2007, pp. 1–6.

[12] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Robust and low complexity rate control for solar powered sensors," in *Proc. DATE*, Mar. 2008, pp. 230–235.

[13] S. Liu, Q. Qiu, and Q. Wu, "Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting," in *Proc. DATE*, Mar. 2008, pp. 236–241.

[14] B. Zhang, R. Simon, and H. Aydin, "Energy management for time-critical energy harvesting wireless sensor networks," in *Proc. 12th Int. Symp. Stabilization, Safety, Secur. Distrib. Syst.*, 2010, pp. 236–251.

[15] Y. Levron, D. Shmilovitz, and L. Martínez-Salamero, "A power management strategy for minimization of energy storage reservoirs in wireless systems with energy harvesting," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 3, pp. 633–643, Mar. 2011.

[16] H. Wang, D. Estrin, and L. Girod. "Preprocessing in a tiered sensor network for habitat monitoring," *EURASIP JASP Special Issue Sensor Netw.*, vol. 2003, no. 4, pp. 392-401, Mar. 2003.

[17] R. A. Manesha and M. Velladurai, "Efficient image transmission in wireless sensor networks using wavelet coded preprocessing technique," *Int. J. Comput. Netw. Wireless Commun.*, vol. 2, no. 2, pp. 218–224, Apr. 2012.

[18] Q. Liu, G. A. Constantinides, K. Masselos, and P. Cheung, "Combining data reuse with data-level parallelization for FPGA-targeted hardware compilation: A geometric programming framework," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 3, pp. 305–215, Mar. 2009.

[19] Q. Liu, T. Todman, and W. Luk, "Combining optimizations in automated low power design," in *Proc. DATE*, Mar. 2010, pp. 1791–1796.

[20] Q. Liu, T. Mak, J. Luo, W. Luk, and A. Yakovlev, "Power adaptive computing system design in energy harvesting environment," in *Proc. Int. Conf. Embedded Comput. Syst., Archit., Model. Simul.*, Jul. 2011, pp. 33–40.

[21] K. Akkaya and M. Younis, "An energy-aware QoS routing protocol for wireless sensor networks," in *Proc. Int. Conf. Distrib. Comput. Syst. Workshops*, May 2003, pp. 710–715.

[22] L. Benini, A. Bogliolo, R. Bogliolo, and G. D. Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 3, pp. 299–316, Jun. 2000.

[23] Z. Ren, B. H. Krogh, and R. Marculescu, "Hierarchical adaptive dynamic power management," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 409–420, Apr. 2005.

[24] D. Noh, L. Wang, Y. Yang, H. K. Le, and T. Abdelzaher, "Minimum variance energy allocation for a solar-powered sensor system," in *Proc. 5th IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, 2009, pp. 44–57.

[25] H. Li, S. Bhunia, Y. Chen, K. Roy, and T. N. Vijaykumar, "DCG: Deterministic clock-gating for low-power microprocessor design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 3, pp. 245–254, Mar. 2004.

[26] A. B. Kahng, S. Kang, T. S. Rosing, and R. Strong, "Many-core token-based adaptive power gating," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 8, pp. 1288–1292, Aug. 2013.

[27] J. Mattingley and S. Boyd, "Real-time convex optimization in signal processing," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 50–61, May 2010.

[28] S. Boyd and L. Vandenberghe, *Convex Optimization*. Singapore: Cambridge Univ. Press, 2004.

[29] M. I. Ali, B. M. Al-Hashimi, J. Recas, and D. Atienza, "Evaluation and design exploration of solar harvested-energy prediction algorithm," in *Proc. DATE*, Mar. 2010, pp. 142–147.

[30] (Oct. 2010). *Oak Ridge National Laboratory (ORNL) RSR Web Site*. [Online]. Available: http://www.nrel.gov/midc/ornl_rsr/

[31] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[32] M.-H. Ho *et al.*, "Architecture and design flow for a highly efficient structured ASIC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 3, pp. 424–433, Mar. 2013.

Qiang Liu (M'14) received the B.S. and M.Sc. degrees from the School of Electronic Information Engineering, Tianjin University, Tianjin, China, in 2001 and 2004, respectively, and the Ph.D. degree from the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., in 2008.

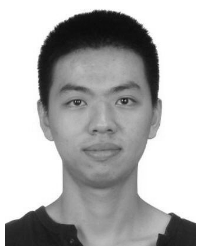
He was a Research Associate with the Department of Computing, Imperial College London, from 2009 to 2011. He is currently an Associate Professor with the School of Electronic Information Engineering with Tianjin University. His current research interests include low-power and power-adaptive design, VLSI design, and reconfigurable computing

Terrence Mak (S'05–M'09) received the B.Eng. and M.Phil. degrees in systems engineering from the Chinese University of Hong Kong, Hong Kong, in 2003 and 2005, respectively, and the Ph.D. degree from Imperial College London, London, U.K., in 2009.

He was with the School of Electrical, Electronic and Computer Engineering, Newcastle University, Newcastle upon Tyne, U.K., as a Lecturer, from 2010 to 2012. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong. His current research interests include VLSI/FPGA architecture design, networks-on-chip, green and sustainable computing, and systems for biomedical applications.

Tao Zhang received the dual B.S. degree in electronic engineering and management engineering from Tianjin University, Tianjin, China, in 1998, and the M.S. and Ph.D. degrees in signal and information processing from Electronic and Information Engineering School, Tianjin University, in 2001 and 2004, respectively.

He is currently an Associate Professor with Tianjin University. His current research interests include signal processing, digital audio processing, and DSP system application.



Xinyu Niu received the Degree from Fudan University, Shanghai, China, and the M.Sc. and Ph.D. degrees from Imperial College London, London, U.K.

His current research interests include developing applications and tools for reconfigurable computing that involves runtime reconfiguration.

Wayne Luk (F'09) is a Professor of Computer Engineering with Imperial College London, London, U.K. His current research interests include theory and practice of customizing hardware and software for specific application domains, such as multimedia, financial modeling, medical computing, and high-level compilation techniques, tools for high-performance computers, and embedded systems.

Prof. Luk is a fellow of the British Computer Society. He was a recipient of the Research Excellence Award from Imperial College London, and 11 awards for his publications from various international conferences.

Alex Yakovlev (M'97–SM'–) received the D.Sc. degree from Newcastle University, Newcastle upon Tyne, U.K., in 2006, and the M.Sc. and Ph.D. degrees from the Saint Petersburg Electrical Engineering Institute, Saint Petersburg, Russia, in 1979 and 1982, respectively.

He held the positions of Assistant and Associate Professor with the Department of Computing Science from 1982 and 1990. Since 1991, he has been with Newcastle University, where he is a Professor and leads the Microelectronic Systems Design Research Group at the School of Electrical and Electronic Engineering. His current research interests include modeling and design of asynchronous, concurrent, real-time, and dependable systems-on-a-chip.

Prof. Yakovlev is a Dream Fellow of Engineering and Physical Sciences Research Council, U.K.