Intra-task Scheduling for Storage-less and Converter-less Solar-Powered Nonvolatile Sensor Nodes

Daming Zhang* Shuangchen Li* Ang Li* Yongpan Liu* X.Sharon Hu† Huazhong Yang*
Dept. of Electronic Engineering, Tsinghua University, Beijing, 100084, China*
Dept. of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, U.S.†
ypliu@tsinghua.edu.cn*, zdm06@mails.tsinghua.edu.cn*

***Abstract*—Solar-powered sensor nodes without specific energy maintenance have shown great promise in many applications, but they suffer from large energy storage and power converter loss. The storage-less and converter-less architecture with nonvolatile processing units has been proposed to reduce the energy loss. However, the architecture is sensitive to solar variations, since there is no energy buffering. Traditional inter-task scheduling methods may not work well due to large variations of task execution time. To tackle the challenge, we develop an algorithm for intra-task scheduling to achieve better quality of service. The experimental results show that the intra-task scheduling algorithm reduces deadline miss rate by as much as 35% and improves energy utilization by close to 20%.**

***Index Terms*—Solar-Powered Nonvolatile Sensor Nodes, Storage-less and Converter-less, Intra-task Scheduling, Energy Utilization**

I. Introduction

Wireless sensor nodes (WSN) acquire information from the physical world, and the information is then processed and used for many different purposes. WSN are used in diverse applications from earthquake monitoring to warfare. Usually, such WSNs are powered by batteries, which need to be replaced or recharged periodically. With the rapid development of sensor networks, it is predicted that the world will enter into an era with a trillion WSNs by 2035 [1]. The number of WSNs on or around each person can reach a thousand or higher. It will be prohibitively complex or even impossible to maintain so many WSNs manually and frequently. Besides, many sensor nodes are deployed on hard-to-access structures [1], life animals [2] and dangerous places [3], where replacing batteries is quite difficult or impracticable. Therefore, self-powered sensor nodes without batteries are gaining popularity.

Among different energy harvesting sources, solar power has the highest power density and is easy to obtain [3]. Thus, solar-powered sensor nodes are widely used. Most of traditional solar-powered sensor nodes adopt the "harvest-store-use" architecture [3]. The power supply system contains storage units and input/output voltage regulators. The storage units can offer additional energy when solar power is low. The regulators can keep the input/output voltage at a required level to maximize power tracking and to make the sensor node operate correctly. The load system contains one or multiple general purpose processors.

Besides energy supply, quality of services (QoS) is also an important issue for sensor nodes. One key QoS metric is the deadline miss rates of tasks executed on the sensor nodes. In many applications, tasks are expected to be completed before their deadlines. For example, sensor nodes for body health monitoring collect ECG, EEG and motion data and these data should be processed by certain time for them to be useful. Reducing deadline miss rate leads to better monitoring effects and user experiences. Solar-powered sensor nodes without stable energy sources, however, suffer uncertainty in energy supply, which results in bad QoS during solar variations. Therefore, task scheduling and power management techniques for solar-powered sensor nodes are extremely important.

This work was supported in part by High-Tech Research and Development (863) Program under contract 2013AA01320 and Huawei Shannon Lab and the Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions under contract YETP0102.

A large amount of task scheduling work for solar-powered sensor nodes has been presented in recent years. They can be classified as offline and online scheduling methods. Offline scheduling methods make use of historical solar profiles and assume constant task timing parameters. Based on average values of the historical solar data, Kansal *et al.* [4] proposed an offline scheduling method of duty-cycling between active and low power modes on sensor nodes. The method was used to achieve perennial operation at a desired performance level. Audet *et al.* [5] presented static schedulers based on the energy consumption of recurring tasks, which tend to assign longer execution periods to tasks with higher energy requirements. Though their overhead is low, these offline scheduling methods are less effective since changes in both solar profiles and task execution status are unavoidable in the field.

Online scheduling methods for real-time solar-powered sensor nodes have been proposed in a number of papers (e.g. [6], [7], [8]). Lazy scheduling algorithm (LSA) is one of the most widely used algorithms. Moser *et al.* [9], [10], [6] proved that LSA can achieve the lowest deadline miss rate for given time and energy constraints if energy storage units of sensor nodes were replenished predictably. Recas *et al.* used a solar prediction method (Weather Conditioned Moving Average, WCMA) for LSA to deal with solar variations [7]. Zhu *et al.* [8] further improved LSA with decomposition and combination of task chains to improve QoS. However, none of the LSA based methods exploit slacks resulted from task execution time variations.

To utilize task slacks effectively, researchers developed scheduling algorithms with online dynamic voltage and frequency scaling (DVFS). In [11] and [12], Liu *et al.* developed an energy-aware scheduling method and a load-matching adaptive algorithm with online DVFS. Lin *et al.* [13] used a more accurate scheduling algorithm with online DVFS, where nonlinear models of DC-DC converters and storage units were taken into account. Recently, Wang *et al.* [14] proposed a load tuning based scheduling method with distributed online DVFS for solar-powered multicore systems. The method improved solar energy utilization ratio and QoS of the systems. However, all the above works are based on the traditional architecture employing energy storage and power converters.

Recently, Wang *et al.* [15] proposed a storage-less and converter-less solar-powered sensor node with a $3us$ wake-up-time nonvolatile processor [16]. The sensor node achieves nearly 90% energy utilization by eliminating energy loss from DC-DC converters and storage units if the solar power is smaller than the processor core power. However, the node still suffers from energy loss if the solar power is larger than the core power. To achieve higher energy utilization, it is better to use the power supply system in [15] with multiple heterogeneous nonvolatile processing units [16] (NPUs) (see Figure 1). With dynamic power management (DPM) on these NPUs, the storage-less and converter-less WSN with multiple NPUs (SCMN) achieves higher energy utilization.

However, current scheduling algorithms are not suitable for the SCMN architecture. Without energy buffers, the architecture suffers from uncertain execution delays of tasks during solar variation, which leads to low QoS. To improve QoS, intra-task scheduling, which schedules tasks at any time during their execution, can be extremely beneficial. However,

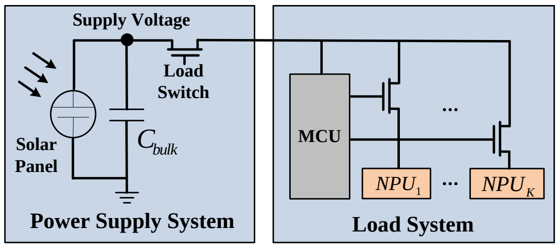


Fig. 1. The power supply and load systems of the SCMN architecture

current algorithms support only inter-task scheduling. Therefore, we need to design intra-task scheduling algorithms, which can schedule tasks effectively at any time during their execution.

In this paper, we introduce an algorithm to realize online intra-task scheduling on the SCMN architecture. Our contributions are listed as below:

- Proposing a system model of the SCMN architecture for intra-task scheduling design.
- Developing an algorithm to realize online intra-task scheduling on the SCMN architecture. The algorithm is launched by a trigger mechanism, which takes variations of both solar power and task status into consideration. Then task priorities are calculated with an artificial neural network (ANN), which is trained by an optimal intra-task priority solution. Finally, a task selection algorithm is developed based on the task priorities at the scheduling point.
- Validating our algorithm on the SCMN architecture with random and real benchmarks. The experimental results show that the algorithm reduces deadline miss rate by as much as 35% and improves energy utilization by close to 20%.

II. Motivation and Challenges

This section presents the motivation and challenges in designing an online intra-task scheduling approach on the SCMN architecture.

A. Motivation

Figure 2 shows a motivation example, where an intra-task scheduling algorithm is compared with an inter-task one. The example contains four tasks (τ_1 to τ_4) with their deadlines (D_1 to D_4). Task allocations and dependencies are also given at the top of Figure 2.

In Figure 2, the inter-task algorithm only schedules tasks when the corresponding NPUs are available. Tasks cannot be interrupted during execution. Thus, solar variations lead to energy loss and extend execution delays, when τ_1 and τ_2 are executed. Finally, τ_1, τ_2 and the following τ_3 miss their deadlines. In the intra-task algorithm, however, tasks can be interrupted and scheduled based on both solar variation and task status during execution. Thus, energy can be utilized more effectively and more deadlines can be satisfied. Eventually, only τ_1 misses its deadline and the finish times of all the tasks are earlier than those obtained by the inter-task algorithm.

In general, an intra-task scheduling algorithm, if designed properly, may achieve much higher energy utilization and QoS than an inter-task one on the SCMN architecture.

B. Challenges

For the SCMN architecture with real-time varying solar power, intra-task scheduling can assign any task to be executed or not in any time slot. Thus, the complexity of of a brute-force algorithm is O($2^{N \cdot M}$), where N is the number of tasks and M denotes the number of time slots in a period. The huge space search space brings us two big challenges to design online intra-task scheduling algorithms: (i) When to stop executing one task and start another one? (ii) How to schedule task execution effectively based on the solar power and tasks status? We propose an approach to tackle these challenges in Section 4.

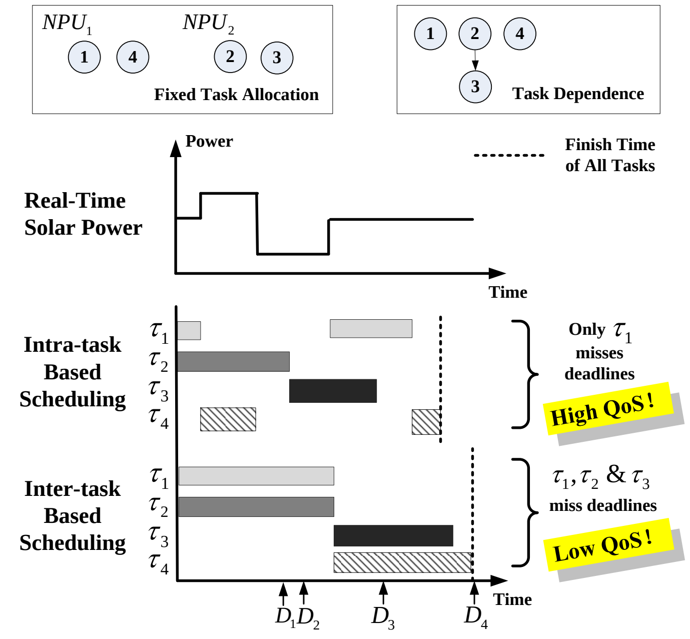


Fig. 2. A motivation example

III. System Modeling and Problem Formulation

This section first describes the system model and defines scheduling variables for online intra-task scheduling. After that, the problem formulation is presented.

TABLE I
PARAMETERS AND VARIABLES OF THE SYSTEM MODEL

	Parameters	Description
Task	D_i	Deadline of τ_i
	L_i	Total execution delay of τ_i
	P_i	Maximal power of τ_i (mW)
	$E_{i,j}$	Task dependence from τ_i to τ_j
System	T	Scheduling period of tasks, $t \in T$
	$P^s(t)$	Solar power at t (mW)
	A_k	Task allocation on NPU_k
Scheduling	$x_i(t)$	$x_i(t) = 1$, if τ_i is executed during the time slot t; otherwise,$x_i(t) = 0$.
	$p_i(t)$	Average power consumption of τ_i during the time slot t (mW)
	$l_i^{rem}(t)$	Remaining execution cycles of τ_i on t

A. Task related parameters

We consider a set of real-time tasks to be executed on a solar-powered sensor node. Table I summarizes the parameters and scheduling variables[1] of the system model. We use a directed acyclic graph $G(V, E)$ to describe task dependencies. V is the task set and E is the edge set. In V, there are N tasks (denoted as $\{\tau_1, \tau_2, \cdots, \tau_N\}$), which are executed by K NPUs. Each task (τ_i) has three parameters: D_i, L_i and P_i. D_i is the deadline of τ_i. L_i and P_i are τ_i's the number of execution cycles and maximal power. In E, $E_{i,j}$ denotes the task dependence from τ_i to τ_j. $E_{i,j} = 1$, if τ_j depends on the results of τ_i; Otherwise, $E_{i,j} = 0$.

B. System related parameters

The system of the solar-powered sensor node contains three parameters. T=$\{1, \cdots, t, \cdots, M\}$ is the set of time slots. $P^s(t)$ denotes the solar power at the time slot t. It is the scheduling period of tasks during which all the tasks must be completed and is referred to as scheduling

[1] Note that we use lower case letters for variables and upper case letters for parameters/constants whenever appropriate.

period. We assume that tasks can be preempted at any time slot t and M is the number of time slots in the period. Note that tasks in $[M+1,2M]$ are independent of those in $[1,M]$. A_k is the task allocation on NPU k and meets the NPU resource constraint: each task can only be executed by a certain NPU while an NPU executes one task at the same time.

C. Scheduling variables

We define scheduling variables as follows. $x_i(t)$ is an independent variable, where $x_i(t) = 1$ if τ_i is executed at t and $x_i(t) = 0$ otherwise. Based on $x_i(t)$, we defined two intermediate variables: $p_i(t)$ and $l_i^{rem}(t)$. $p_i(t)$ is the average power consumption of τ_i during the time slot t and it is calculated as follows.

$$p_i(t) = \begin{cases} 0 & \text{if } x_i(t) = 0 \\ P_i & \text{else if } \sum_{k=1}^{N} x_k(t) \cdot P_k \leq P^s(t) \\ P^s(t) \cdot \frac{P_i}{\sum_{k=1}^{N} x_k(t) \cdot P_k} & \text{otherwise} \end{cases} \tag{1}$$

That is, $p_i(t) = 0$, if τ_i is not executed; $p_i(t) = P_i$, if the solar power is sufficient ($\sum_{k=1}^{N} x_k(t) \cdot P_k \leq P^s(t)$); $p_i(t) = P_i/[\sum_{k=1}^{N} x_k(t) \cdot P_k]$, which is a fraction of the available solar power, if the solar power is less than the load power ($\sum_{k=1}^{N} x_k(t) \cdot P_k$). $l_i^{rem}(t)$ is the remaining execution cycles of τ_i during the time slot t, which is used to describe its current status. $l_i^{rem}(t)$ is calculated as follows.

$$l_i^{rem}(t) = L_i - [\sum_{i=1}^{t} x_i(t) \cdot p_i]/P_i \tag{2}$$

where $[\sum_{i=1}^{t} x_i(t) \cdot p_i]/P_i$ is the executed cycles of τ_i. $l_i^{rem}(t) = L_i$, if τ_i has never been executed; $l_i^{rem}(t) = 0$, if τ_i is completed.

D. Problem formulation

Given the parameters and variables defined above, the scheduling problem can be formulated as an optimization problem. Our goal is to find a schedule $\{x_i(t)\}$ for all t ($t \in T$) that minimizes the deadline miss rate of the tasks executed on the solar-powered sensor node.

$$\min \sum_{i=1}^{N} \theta(l_i^{rem}(D_i))/N \tag{3}$$

where $\theta(\bullet)$ is calculated as follows.

$$\theta(l_i^{rem}(D_i)) = \begin{cases} 1 & \text{if } l_i^{rem}(D_i) > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{4}$$

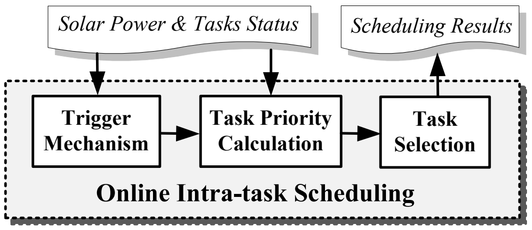
That is, τ_i misses its deadline, if $l_i^{rem}(D_i) > 0$; Otherwise, τ_i meets its deadline.

IV. Proposed algorithm for intra-task scheduling

In this section, we present our framework for online intra-task scheduling on the SCMN architecture. We first give an overview of the framework and then discuss each part in detail in the subsequent subsection.

A. Algorithm framework

According to Section II-B, two basic challenges are presented: when and how to schedule tasks on the SCMN architecture during solar variations? To tackle the challenges, an algorithm framework for intra-task scheduling is presented in Figure 3. It contains three parts. The trigger mechanism is used to choose scheduling points based on variations of both the solar power and task status. Then, task priority calculation has been done with an ANN trained by an optimal intra-task priority solution. Finally, task selection is performed based on task priorities and scheduling results are generated. We illustrate each part in the following subsections.



Solar Power & Tasks Status
Scheduling Results
Trigger Mechanism
Task Priority Calculation
Task Selection
Online Intra-task Scheduling

Fig. 3. The algorithm framework

B. Trigger mechanism

As online intra-task scheduling can perform on any time slot of task execution, the first challenge is to choose scheduling points. Observing variations of the solar power and task status on the SCMN architecture, we find that intra-task scheduling should be done in the following situations: (i) current tasks are completed; (ii) current tasks miss their deadlines; (iii) solar variations happen, which impacts execution delays of current tasks. Therefore, we use a trigger mechanism to launch the scheduling process, which stores all the situations as triggers: task finishing, deadline missing and solar variations. The three kinds of triggers are defined as follows.

$over_i$ denotes the task finishing trigger of τ_i. The scheduling point is triggered, if $over_i = 1$, which means τ_i finishes ($l_i^{rem}(t) = 0$); Otherwise ($over_i = 0$), the scheduling does not work. The trigger is also used in the inter-task scheduling algorithms. In the worst case, the number of scheduling points triggered by $\{over_i\}$ is N, which is the task number.

$miss_i$ denotes the deadline missing trigger of τ_i. The scheduling point is triggered, $miss_i = 1$, which means τ_i misses its deadline ($l_i^{rem}(D_i) > 0$) and there are no other following tasks depending it; Otherwise ($miss_i = 0$), the scheduling does not work. In the worst case, the number of scheduling points triggered by $\{dm_i(D_i)\}$ is N, which is the task number.

$sv(t)$ denotes the solar variation trigger at t, which is calculated as follows.

$$\begin{aligned} sv(t) &= \Delta P^s(t, ts) - \delta \\ \Delta P^s(t, ts) &= |P^s(t) - P^s(ts)| \end{aligned} \tag{5}$$

where $\Delta P^s(t, ts)$ is the solar variation between $P^s(t)$ and $P^s(ts)$. $P^s(ts)$ is the solar power used at the last scheduling point ts. δ is the solar variation threshold, which is determined by users. The scheduling point is triggered, if $sv(t) > 0$, which means the solar variation is larger than the threshold; Otherwise, the scheduling does not work. As $sv(t)$ is detected based on δ, smaller δ means a larger number of scheduling points triggered by $sv(t)$. In the worst case, the number is M, which is the number of time slots in T. We further analyze scheduling effects and overheads of intra-task scheduling under different solar variation thresholds (δ) in Section V-C1.

Based on the trigger mechanism, we find all scheduling points for online intra-task scheduling.

C. Task priority calculation

After solving the problem of choosing scheduling points by the trigger mechanism, we face the second challenge: how to select tasks to be executed or not at each scheduling point? For task selection, we need to calculate current task priorities. Thus, we define task priorities and develop an optimal intra-task priority solution for online task priority calculation.

1) Task priority definition: To define task priorities, all related metrics of tasks and the solar power should be concerned. When considering task deadlines, we prefer that a task with an earlier deadline gets a higher priority based on the EDF (Earliest Deadline First) rule, as tasks with earlier deadlines miss their deadlines easily; When considering task energy, we prefer that a task with smaller execution energy gets a higher priority, as larger energy consumption of tasks supplied by the limited solar energy are easier to cause deadline missing of other tasks. When considering task dependencies, we prefer that a task gets a higher priority with more depended tasks, as it is the bottleneck of the system and needs to be executed earlier. Besides, the solar power also impacts task priorities based on their status, as the supplied power source for task scheduling is changeable. For example, tasks with early deadlines may be given up during execution when the solar power drops; tasks with large energy consumption may be chosen to execute when the solar power increases.

Based on the consideration, the task priority of τ_i (m_i, smaller m_i means the higher priority) is defined as follows.

$$m_i = \sum\nolimits_{k=1}^{3} w_k \cdot M_i^k \tag{6}$$

$\{M_i^k\}$ are metrics of τ_i, which are calculated as follows.

$$M_i^1 = \frac{D_i}{\max\{D_k | k \in [1, N]\}} \tag{7}$$

$$M_i^2 = \frac{P_i \cdot L_i}{\max\{P_k \cdot L_k | k \in [1, N]\}} \tag{8}$$

$$M_i^3 = (1 + \sum\nolimits_{k=1}^{N} C_{i,k})^{-1} \tag{9}$$

where $C_{i,j}$ is the data dependence from τ_i to τ_j in $G(V, E)$. $C_{i,j} = 1$, if there are data transmission from τ_i to τ_j; Otherwise, $C_{i,j} = 0$. For metrics of τ_i, M_i^1 denotes the impact of the deadline; M_i^2 denotes the impact of the energy consumption; M_i^3 denotes the impact of the task dependency.

$\{w_k\}$ are the weights of metrics ($\{M_i^k\}$), which denotes the impact of the solar power based on the current execution status of tasks. Thus, $\{w_k\}$ change during solar variations. In order to achieve optimal $\{w_k\}$ for task priority calculation at each scheduling point, we propose an optimal intra-task priority solution, which is described in Section IV-C2.

2) Optimal intra-task priority method: To get optimal $\{w_k\}$, we propose an optimal intra-task priority solution, which is based on both the historical solar power and current task status. The solution contains two parts. In part 1, a mixed integer nonlinear programming (MINLP) formulation for optimal scheduling results ($\{x_i(t)\}$) is developed; In part 2, optimal $\{w_k\}$ are achieved by a linear programming (LP) formulation based on the scheduling results.

In part 1, the inputs of the optimal MINLP formulation at scheduling point $t0$ consist the solar power ($\{P^s(t)\}, t \in [t0, M]$) and the current task status at $t0$ ($\{l_i^{rem}(t0)\}$). Based on the inputs, the formulation is shown as follows.

objective: $$\min \sum\nolimits_{i=1}^{N} \theta(l_i^{rem}(D_i))/N, \tag{10}$$

subject to:

(a) Task dependency constraint: (11)

$$f_i < s_j, \quad \forall i, j, \text{ if } E_{i,j} = 1,$$

(b) Solar energy constraint: (12)

$$\sum\nolimits_{i=1}^{N} x_i(t) \cdot p_i(t) \le P^s(t), \quad \forall t \in [t0, M],$$

(c) Task energy constraint: (13)

$$\sum\nolimits_{t=t0}^{M} p_i(t) = P_i \cdot l_i^{rem}(t0), \forall i \in \{i | \tau_i \text{ is done}\},$$

(d) NPU resource constraint: (14)

$$\sum\nolimits_{\forall i \in A_k} x_i(t) \le 1, A_k = \{i | \tau_i \text{ runs on } NPU_k\},$$
$$\forall t \in [t0, M], \forall k \in [1, K].$$

where s_i and f_i are τ_i's start and completion time. The objective is to minimize the deadline miss rate based on the optimal scheduling variables ($\{x_i(t)\}$). Besides, there are four constrains in the formulation. Task dependency constraint (11) means that τ_j starts only after all its depending tasks are completed; Solar energy constraint (12) means that the load power consumption is no more than the solar power supply; Task energy constraint (13) means that the total energy consumption of τ_i is a constant; NPU resource constraint (14) means an NPU can only run one task at the same time. The formulation can be solved by a nonlinear programming solver (like LINGO). Besides, the optimal deadline miss rate is also serves as a lower bound in the experiment (Section V).

In part 2, we achieve the optimal $\{w_k\}$ as follows. A reference task priority set $\{m_i^{ref}\}$ can be easily get based on the optimized scheduling results ($\{x_i(t)\}$). Then, optimal $\{w_k\}$ are achieved based on $\{m_i^{ref}(t)\}$ by solving the following LP formulation as follows.

$$\min \sum\nolimits_{i=1}^{N} |m_i^{ref} - \sum\nolimits_{k=1}^{3} w_k \cdot M_i^k| \tag{15}$$

In this way, optimal $\{w_k\}$ are gained on the scheduling point $t0$.

3) ANN training: In the optimal intra-task priority method above, the optimal formulation (Equation 10-14) is an NP-hard problem, which has an exponential complexity. The complexity of the LP formulation (Equation 15) is O(N^2), where N is the variable number. Besides, the number of intra-task scheduling points is large based on the trigger mechanism. Thus, the solution cannot be directly used for online task priority calculation.

Therefore, we need an effective model to replace the optimal solution by offline data fitting. The inputs for data fitting are the historical solar power ($\{P^s(t)\}, t \in [t0, M]$) and current task status ($\{l_i^{rem}(t0)\}$). For further simplification, $\{P^s(t)\}$ are expressed by two parameters: P_{avg}^s and P_{var}^s, which are the average value and variance of $\{P^s(t)\}$. The outputs for data fitting are the corresponding optimal $\{w_k\}$. In order to capture the complex relationship between the inputs and outputs effectively, we choose a back propagation (BP) ANN model with single hidden layer[2] for offline training and use it to calculate the optimal $\{w_k\}$ online.

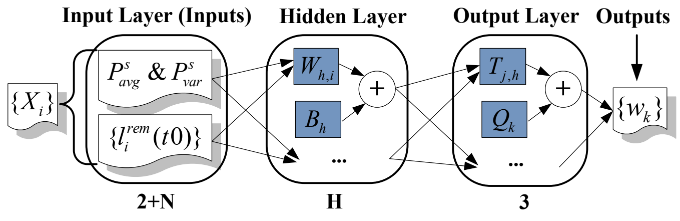


Fig. 4. The architecture of the BP ANN

We present the ANN's architecture in Figure 4. It contains three layers: input layer, hidden layer and output layer. The input layer contains $2 + N$ elements (solar power $P_{avg}^s \& P_{var}^s$ and task status $\{l_i^{rem}(t0)\}$), where N is the number of task status. We use a set $\{X_i\} = \{P_{avg}^s, P_{var}^s, l_i^{rem}(t0)\}$ to store the inputs. The hidden layer contains H neurons, where H is defined by users. The nonlinear activation function ($f(x)$, a sigmoidal function) and the output function (O_h) in the layer are presented as follows.

$$f(x) = (1 + e^{-x})^{-1} \tag{16}$$

$$\begin{gathered} O_h = f(\sum\nolimits_{i=1}^{2+N} W_{h,i} \cdot X_i + B_h) \\ i \in [1, 2+N], h \in [1, H] \end{gathered} \tag{17}$$

where $\{W_{h,i}\}$ and $\{B_h\}$ are the weights and thresholds of the hth neurons in the hidden layer. The output layer contains three neurons, which is the number of $\{w_k\}$. In the layer, the nonlinear activation function is shown in Equation 16 and the output function (Y_k) is presented as follows.

$$\begin{gathered} Y_k = f(\sum\nolimits_{h=1}^{H} T_{j,h} \cdot O_h + Q_k) \\ h \in [1, H], k \in [1, 3] \end{gathered} \tag{18}$$

where $\{T_{j,h}\}$ and $\{Q_k\}$ are the weights and thresholds of the kth neurons in the output layer.

Then, we train the ANN offline to get the parameters (weights and thresholds of all neurons) with samples (the inputs and outputs above).

[2] Other models including machine learning models can also be used for offline data fitting.

For once training, all parameters are adjusted and the error E_p is calculated as follows.

$$E_p = 0.5 \times \sum_{k=1}^{3} (w_k - w_k^{ann})^2 \quad (19)$$

where $\{w_k\}$ are the expected outputs gained by the optimal solution and $\{w_k^{ann}\}$ are the calculated outputs gained by the trained ANN. Besides, iteration number n_{ann} for the ANN training is determined by error threshold E_r and iteration number constraint n_{const}. The ANN needs to be trained with more samples, if $E_p > E_r$ & $n_{ann} < n_{const}$; Otherwise, the training is done and the ANN parameters are achieved. The complexity of calculating the trained ANN is O($H + 3$), which is equal to the total neuron number in the ANN. What's more, we analyze the neuron number (H) and the error threshold (E_r) in the hidden layer, which have influences on iteration numbers and online scheduling effects. The analysis is presented in Section V-C2.

In this way, task priorities ($\{m_i\}$) for online intra-task scheduling can be achieved effectively by calculating the trained ANN. The impacts of the ANN are tested in Section V-C3.

D. *Task selection*

Based on the calculated task priorities in Section IV-C, the algorithm of task selection at $t0$ is presented as follows. The inputs are $\{P_i\}$, $P^s(t0)$ (in Table I) and calculated task priorities $\{m_i\}$. The outputs are the scheduling results ($\widetilde{x}_i(t0)$) at $t0$.

Algorithm 1: Task Selection on $t0$

```
   input : $\{P_i\}$,$P^s(t0)$, $\{m_i\}$
   output: online scheduling results $\{\widetilde{x}_i(t0)\}$
1  if $S_p$ has been updated then
2      initial $\{\widetilde{x}_i(t0)\}$ as zeros, set $n = 1$;
3      sort $\{m_i\}$ in ascending order;
4      while $\sum_{i=1}^{N} \widetilde{x}_i(t) \cdot P_i < P^s(t0)$ & $n \le N$ do
5          get$\tau_i$ with the n^th priority in $\{m_i\}$;
6          if $\tau_i$'s corresponding NPU is available & $l_i^{rem}(t0) > 0$
           then
7              set $\widetilde{x}_i(t0) = 1$ and execute $\tau_i$ on the NPU.
8          end
9      end
10 else
11     the scheduling remains the same
12 end
```

In the algorithm, initialization is done (Line 2) and $\{m_i\}$ are sorted (Line 3). Then, tasks with higher priorities in $\{m_i\}$ are chosen to be executed (Line 5&7), if the load power consumption is less than the solar power supply (Line 4). For those uncompleted tasks, which have higher priorities but cannot be executed due to data dependencies or resource unavailable (Line 6), the scheduling remains the same (Line 11). In this way, we get the scheduling results $\{\widetilde{x}_i(t0)\}$ at $t0$. Finally, the deadline miss rate is achieved when all tasks are completed.

V. Experimental evaluation

In this section, we show the experiment setup and compare the proposed algorithm with the WCMA-based LSA and the optimal MINLP formulation (Section IV-C2) on the SCMN architecture. The effectiveness of the proposed algorithm is presented under different solar conditions in both short term and long term. Then, we analyze the algorithm under different solar variation thresholds (δ), neuron numbers (H) and ANN's error thresholds (E_r). Finally, the ANN's impacts and the algorithm's overheads are presented.

A. *Experiment setup*

We use three random cases and a real case as benchmarks in the experiment. The random cases are constructed by more than ten kinds of tasks (AES encoder/decoder, JPEG encoder/decoder, FIR filter etc. [17], [18]), where HDL files of tasks are achieved by C2RTL tools [17]. The numbers of tasks, task dependence edges and NPUs in the random benchmarks are presented as follows.

- R1: 4 tasks, 0 edges, 3 NPUs;
- R2: 6 tasks, 2 edges, 4 NPUs;
- R3: 8 tasks, 4 edges, 5 NPUs.

The real case is wild animal monitoring [2] (WAM), which contains eight tasks[3]. In the WAM case, the total execution delays of tasks (L_i) are gained by Modelsim simulation while the maximal power (P_i) of tasks is achieved by DC Compiler synthesis. Besides, deadlines (D_i), total periods (T), dependencies ($E_{i,j}$) and allocations (A_k) of tasks in all benchmarks are defined by users based on the applications and hardware NPUs.

A solar light intensity database in 2012 [19] is used in our experiment. The corresponding harvested solar power $\{P^s(t)\}$ in the database are achieved based on our solar panel, which has an area of 24.5cm^2 and an energy conversion efficiency of 0.06% [15]. The solar power from Jan. to Jun. are used to train the ANN in the proposed algorithm. Besides, the solar power of five individual days (7:00∼17:00) in Sep. are chosen for short term evaluation, where the solar profiles are presented in Figure 5 (a) and their average values and variances are shown in Figure 5 (b). We also use the solar power from July. to Aug. for long term evaluation.

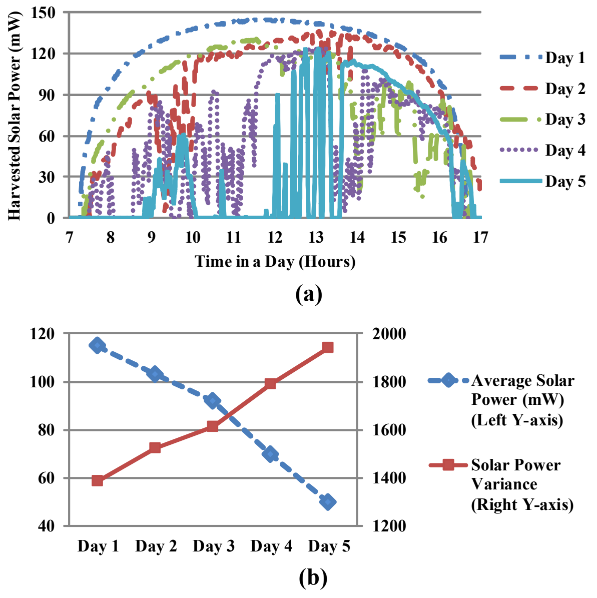


Fig. 5. Harvested solar power (7:00∼17:00) in the five days

The configurations of the proposed algorithm are presented as follows: the time slot (ΔT) is one minute and the solar power threshold (δ) is $10mW$; the neuron number (H) in the hidden layer of the ANN is 10; the normalized ANN's error threshold (E_r) is 0.4 and the iteration number constraint (n_{const}) is 1000. The algorithms in the experiment are implemented and evaluated by running Matlab on an Intel $3GHz$ notebook with $4G$ RAM.

B. *Comparison of the algorithms*

Without loss of generality, the proposed algorithm is compared with both WCMA-based LSA (W-LSA) ([10], [12]) and the optimal MINLP formulation on the SCMN architecture. W-LSA is an inter-task scheduling method, which is widely used on the traditional architecture of solar-powered sensor nodes with energy buffers. The optimal MINLP formulation can achieve optimal scheduling results and minimum deadline miss rates based on known solar profiles, which are used as the lower bounds of deadline miss rate for evaluation. We show deadline miss rates and

[3] The eight tasks are periodic locating, heart rate sampling, voice recording, audio processing, emergency response, audio compressing, local storing and data transmitting.

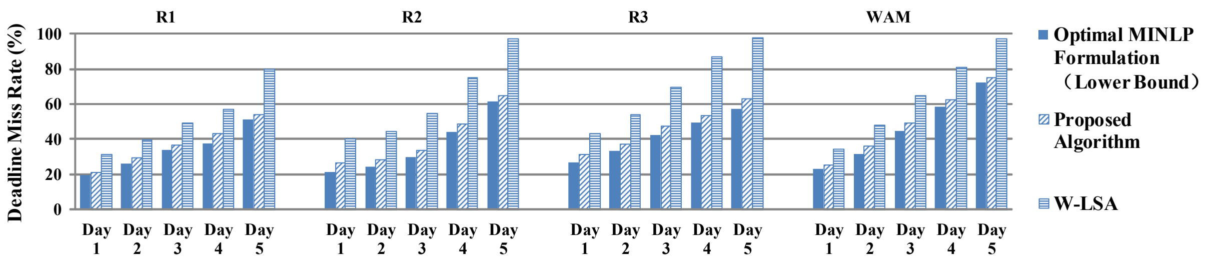


Fig. 6. Deadline miss rate (%) comparison of algorithms in the short term

energy utilizations of the algorithms in both short term (daily) and long term (monthly).

We compare algorithms with all benchmarks in the short term (five individual days). Figure 6 shows deadline miss rates of them. Compared with W-LSA, the proposed algorithm reduces the deadline miss rate by 34.6% at most (Day 5 in R3). Moreover, as the solar power turns smaller from Day 1 to Day 5 (see Figure 5 (b), the average value goes smaller and variation goes larger), the deadline miss rate reduction becomes greater in each benchmark (e.g. from 11.4% to 34.6% in R3). As the inter-task scheduling methods (W-LSA) has no abilities to schedule tasks in their execution periods, it suffers from much worse QoS without energy buffers under more drastic solar variation. However, the proposed intra-task one always works effectively, where its deadline miss rates are very close to the lower bounds (only a 4.50% degradation on average).

According to the analysis of deadline miss rates and energy utilizations of two algorithms, Table II presents the energy utilization improvements (the proposed algorithm vs W-LSA) in five days. The average improvement varies from 16.2% to 24.9% in different benchmarks. What's more, as the solar power turns smaller from Day 1 to Day 5, the improvement becomes larger in each benchmark (e.g. from 18.1% to 30.1% in R3).

TABLE II
ENERGY UTILIZATION IMPROVEMENTS (%) IN THE SHORT TERM

Benchmark	Day 1	Day 2	Day 3	Day 4	Day 5	Average
R1	10.8	13.7	15.6	19.2	21.5	16.2
R2	14.1	16.7	18.2	23.5	24.7	19.4
R3	18.1	22.8	25.3	28.2	30.1	24.9
WAM	13.4	15.8	18.9	21.7	24.5	18.9

We also compare the deadline miss rates and energy utilizations of algorithms with the WAM case in the long term. Figure 7 (a) shows the daily deadline miss rates of two months. Compared with W-LSA, the proposed algorithm reduces the deadline miss rates by 19.7% on average and it always performs better, where the minimum deadline miss rate reduction is 1.6%. Moreover, the deadline miss rates of our algorithm are very close to those of the optimal MILP formulation, with an only 4.9% degradation on average. Figure 7 (b) presents the daily energy utilizations, where the proposed algorithm always achieves higher energy utilizations (19.2% on average) than W-LSA.

From the validations in both the short and long terms, we conclude that our algorithm, almost as good as the optimal MILP formulation, is much better than W-LSA.

C. Analysis of the proposed algorithm

We further analyze three factors in the proposed algorithm with the WAM case in Day 3. The factors are: solar variation threshold (δ) of the trigger mechanism, neuron number (H) in the hidden layer of the ANN and the error threshold (E_r). Then, the ANN's impact and the algorithm's overheads are presented.

1) Analysis of solar variation threshold: Figure 8 presents the deadline miss rates and numbers of intra-task scheduling points under different solar variation thresholds (δ), where the left Y-axis is the deadline miss rate and the right Y-axis denotes the number of scheduling

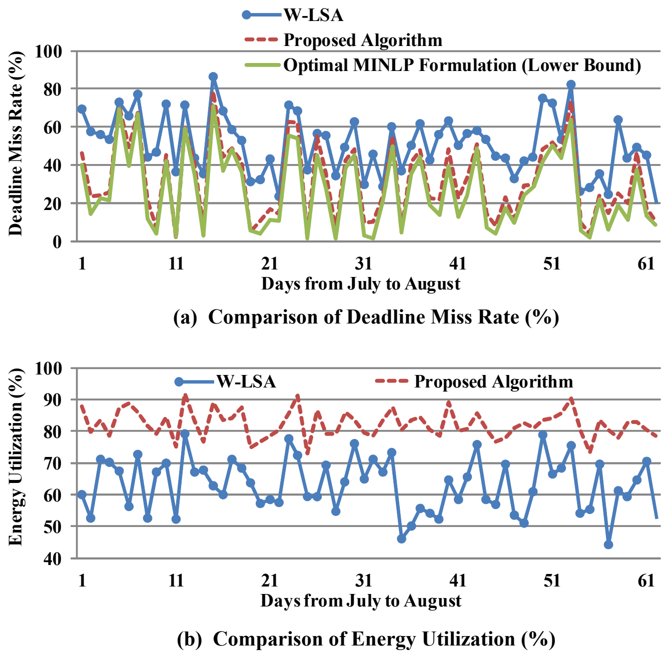


Fig. 7. Comparison of algorithms in the long term

points. As δ decreases from $80mW$ to $2.5mW$ in the X-axis, the number of scheduling points increases rapidly, which means the intra-task scheduling is more granular during solar variation. Besides, the deadline miss rate decreases as δ does. On certain constraint points (e.g. $\delta = 2.5mW$ and $\delta = 5mW$), however, the deadline miss rate does not decrease any more. Because small solar variations at those points does not impact the scheduling results gained by the proposed algorithm.

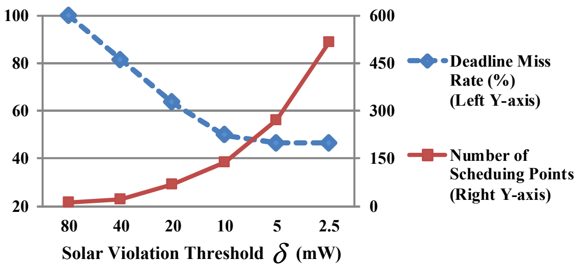


Fig. 8. Scheduling effects under different solar variation threshold δ (mW)

2) Analysis of neuron number and error threshold: Figure 8 presents the deadline miss rates and iteration numbers under different neurons numbers (H), where the left Y-axis is the deadline miss rate and the right Y-axis denotes the iteration number. When H decreases from 20 to 10 in the X-axis, the iteration number becomes larger, as the ANN with more neurons (more parameters of weights and thresholds) are easier to meet E_r. The deadline miss rate nearly stays the same during H reduction from 20 to 10, as E_r for ANN training is constant. However, the deadline miss rate increases while the iteration number reaches the constraint (n_{const}), when H continues to decrease from 7 to 4. Because E_r is not met after n_{const} iterations. Therefore, we can find the minimum neuron number for ANN training while meeting E_r.

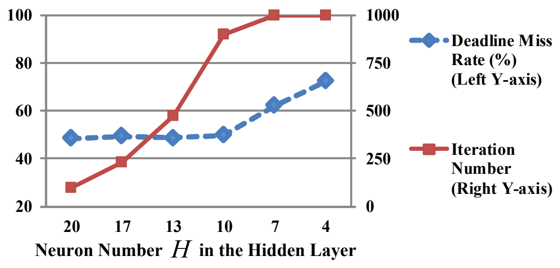


Fig. 9. Scheduling effects under different neuron number H

Furthermore, we analyze the impacts of error thresholds (E_r). Figure 10 presents the deadline miss rates and minimum neuron numbers under different E_r, where the left Y-axis is the deadline miss rate and the right Y-axis denotes the minimum neuron number. As E_r decreases in the X-axis, the minimum neuron number increases while the deadline miss rate turns smaller until it reaches the optimal result.

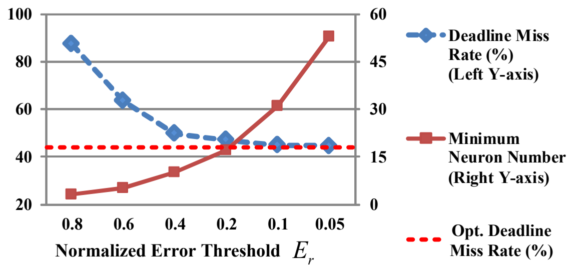


Fig. 10. Scheduling effects under different error thresholds E_r

3) Analysis of the ANN: We analyze the impacts of the ANN in the proposed algorithm based on the R2 case. Figure 11 shows deadline miss rates of the algorithm with ANN and non-ANN in five days. For the non-ANN algorithm, we use the same weight ($w_1 = w_2 = w_3$) to calculate task priorities. Compared with the non-ANN one, the algorithm with ANN reduces the deadline miss rates by 24.3% at most. Besides, the deadline miss rate reduction gets larger when the solar power becomes more effective from Day 1 to Day 5. Because the trained ANN takes the real-time solar power variation into consideration. The ANN is a great contributor to the effectiveness of the algorithm.

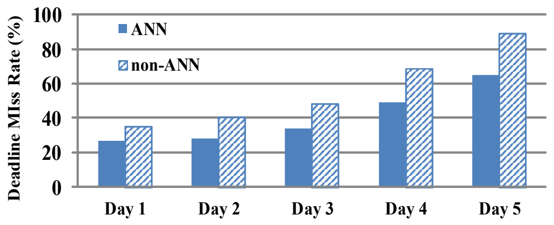


Fig. 11. Deadline miss rate (%) comparison of ANN and non-ANN

4) Analysis of the overheads: We analyze the overheads (execution delay and power consumption) of the proposed algorithm, where the configurations are presented in Section V-A. We run the algorithm on a nonvolatile 8051 processor with the WAM case. The execution delay is gained by an oscilloscope and the power consumption is tested by Data Data Acquisition (DAQ) on the Labview. On each intra-task scheduling point, the execution delay is about $11ms$ and power consumption is about $5.75mW$. Considering the numbers of scheduling points in each day (e.g. from Day 1 to Day 5) are 73∼515, the energy overheads are 0.01%∼0.68%. Therefore, the proposed algorithm is suitable to execute on solar-powered sensor nodes.

VI. Conclusions

In this paper, we propose a trigger based algorithm for online intra-task scheduling on the storage-less and converter-less solar-powered sensor node with multiple heterogeneous nonvolatile processor units. In the algorithm, the intra-task scheduling is triggered by variations of both solar power and task status. Then, an ANN is used for online task priority calculation. The ANN is trained offline by an optimal intra-task priority solution, which is also used to evaluate the effects of the algorithm. The experimental results of of random and real benchmarks show that the algorithm reduces deadline miss rate by as much as 35% and improves energy utilization by close to 20%.

References

[1] W. Liu, X. Fei, T. Tang, P. Wang, H. Luo, B. Deng, and H. Yang, "Application specific sensor node architecture optimization - experiences from field deployments," in *2012 17th Asia and South Pacific Design Automation Conference (ASP-DAC),*, pp. 389–394, 2012.
[2] W. Liu, X. Fei, T. Tang, R. Li, P. Wang, H. Luo, K. Li, L. Zhang, B. Deng, and H. Yang, "Design and implementation of a hybrid sensor network for milu deer monitoring," in *2012 14th International Conference on Advanced Communication Technology (ICACT)*, pp. 52–56, 2012.
[3] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: survey and implications," in *IEEE Communications Surveys and Tutorials, Volume 13, Issue 3*, pp. 443–461, 2011.
[4] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," in *ACM Transactions on Embedded Computing Systems (TECS) - Special Section LCTES'05; Volume 6, Issue 4*, pp. 1–35, 2007.
[5] D. Audet, L. C. Oliveira, N. MacMillan, D. Marinakis, and K. Wu, "Scheduling recurring tasks in energy harvesting sensors," in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 277–282, 2011.
[6] C. Moser, J. Chen, and L. Thiele, "Reward maximization for embedded systems with renewable energies," in *14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '08)*, pp. 247–256, 2008.
[7] J. Piorno, C. Bergonzini, D. Atienza, and T. Rosing, "Hollows: A power-aware task scheduler for energy harvesting sensor nodes," in *Journal of Intelligent Material Systems and Structures, Volume 21, Issue 12*, pp. 1317–1335, 2010.
[8] T. Zhu, A. Mohaisen, P. Yi, and D. Towsley, "Deos: Dynamic energy-oriented scheduling for sustainable wireless sensor networks," in *2012 Proceedings IEEE INFOCOM*, pp. 2363–2371, 2012.
[9] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Lazy scheduling for energy harvesting sensor nodes," in *Working Conference on Distributed and Parallel Embedded Systems (DIPES)*, pp. 125–134, 2006.
[10] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Adaptive power management in energy harvesting systems," in *Design, Automation and Test in Europe Conference and Exhibition (DATE '07)*, pp. 1–6, 2007.
[11] S. Liu, Q. Qiu, and Q. Wu, "Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting," in *Design, Automation and Test in Europe, (DATE '08)*, pp. 236–241, 2008.
[12] S. Liu, J. Lu, Q. Wu, and Q. Qiu, "Load-matching adaptive task scheduling for energy efficiency in energy harvesting real-time embedded systems," in *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, pp. 325–330, 2010.
[13] X. Lin, Y. Wang, S. Yue, N. Chang, and M. Pedram, "A framework of concurrent task scheduling and dynamic voltage and frequency scaling in real-time embedded systems with energy harvesting," in *2013 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, pp. 70–75, 2013.
[14] Y. Wang, R. Chen, Z. Shao, and T. Li, "Solartune: Real-time scheduling with load tuning for solar energy powered multicore systems," in *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 101–110, 2013.
[15] C. Wang, N. Chang, Y. Kim, S. Park, Y. Liu, H. Lee, R. Luo, and H. Yang, "Storage-less and converter-less maximum power point tracking of photovoltaic cells for a nonvolatile microprocessor," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 379–384, 2014.
[16] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Zhao, M. Chiang, Y. Yan, B. Sai, and H. Yang, "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," in *In Proceeding of European Solid State Circuits Conference (ESSCIRC)*, pp. 149–152, 2012.
[17] S. Li, Y. Liu, S. Hu, X. He, Y. Zhang, P. Zhang, and H. Yang, "Optimal partition with block-level parallelization in c-to-rtl synthesis for streaming applications," in *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 225–230, 2013.
[18] Y. Zhang, "Image engineering (i) image processing (second edition),"
[19] Measurement and Instrumentation Data Center (MIDC), http://www.nrel.gov/midc/.