HW/SW Co-design of Nonvolatile IO System in Energy Harvesting Sensor Nodes for Optimal Data Acquisition

Zewei Li[1], Yongpan Liu[1], Daming Zhang[1], Chun Jason Xue[3], Zhangyuan Wang[1], Xin Shi[1], Wenyu Sun[1], Jiwu Shu[2] and Huazhong Yang[1]
[1] Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing, China
[2]Department of Computer Science, Tsinghua University, Beijing, China
[3]Department of Computer Science, City University of Hong Kong, Hong Kong SAR
{ypliu, shujw}@tsinghua.edu.cn, jasonxue@cityu.edu.hk

ABSTRACT

Energy harvesting has been widely investigated as a promising alternative for future wearable sensors or internet-of-things. However, power and performance overhead is induced when IO operations are interrupted by power failures because non-preemptive characteristic of IO operations causes expensive re-executions. Furthermore, the state-of-art IO devices need long and power hungry initializing process, which makes IO operations inefficient in transient powered systems. This paper proposed a HW/SW co-design approach for nonvolatile IO system to maximize data acquisition. A ferroelectric flip-flop based nonvolatile IO architecture is adopted to reduce IO initialization overhead by 3-4 orders of magnitude. Based on the nonvolatile IO interface, we further formulate the optimal data acquisition as an INLP problem and a risk-aware online scheduler is presented to solve the problem efficiently. Experimental results show that the proposed HW/SW co-design architecture improves data acquisition by 2-5 times compared with conventional HW/SW architecture.

CCS Concepts

•**Hardware** → *Sensor devices and platforms;*

Keywords

Nonvolatile IO; HW/SW co-design; energy harvesting

1. INTRODUCTION

With the increase in popularity of wireless sensor networks, implantable and wearable devices, more and more computational ability is required from energy-limited systems, where usage lifetime has become one of the most critical design issues. Given weight and volume as the limiter, a huge amount of sensor nodes with batteries bring about expensive maintenance and polluted environment. Energy harvesting systems without battery manifest strong vitality recently due to ultra-long operation time without maintenance and pollution, including solar, thermal, wireless and vibration energy harvesting sources [1].

Different from the conventional battery powered sensors, energy harvesting systems suffer from unstable power supply due to the ambient environment. Both power amplitude and phase vary significantly, and power failures take place frequently. The power failure introduces nontrivial rollbacks of sensing, computing and transmitting tasks on sensor nodes. Ransford *et al.* [2] spreads computation tasks of RFID among different power failures, which suffers from slow and power hungry global checkpoint. To reduce checkpoint overheads, recently proposed nonvolatile processor techniques [3, 4, 5] adopt nonvolatile register and nonvolatile SRAM to back up data locally. While these techniques remedy power failures for computations on processor, little has been done for peripheral IO devices.

In fact, the operations of sensor nodes need the coordination between processor and IO devices, such as sensors, memory and wireless transceivers. For example, a typical task consists of initializing, sensing, transmitting(storing) and backup stages, and most of them cannot be interrupted by power failures. Once interrupted, those tasks are invalid and should be re-executed. Energy and time are wasted. Without carefully design, IO operations in sensor nodes face performance overheads or even reliability issues [6].

First, IO initialization overheads are quite large in each reboot. The state-of-art IO devices adopt volatile memory to store configuration data, which will be lost after power off. Examples in Sec. 2 show that restoring configuration data contributes up to 87% execution time for each meaningful sensing operation, which is unacceptable in transient powered systems with many reboots [2]. Thus, a more efficient IO architecture is needed to reduce the initialization overheads.

Second, the scheduling of IO operations should consider the hard-to-predict energy harvesting factors. Lots of IO scheduling work existed for stable powered system, where performance-aware algorithms [7] maximize throughput under the limitation of IO bandwidth and energy-aware schedulers reduce power by merging small IO requests for dynamic power management [8, 9]. However, none of them consider energy harvesting factors. There are also schedulers for computation tasks on energy harvesting platforms [10, 11]. However, no scheduler considers the non-preemptive characteristic of transient powered IO operations.

We propose a HW/SW co-design of nonvolatile IO system for energy harvesting sensor nodes to optimize data acquisition. A nonvolatile IO (NVIO) architecture is proposed to retain IO states locally and reduce initialization overheads while a risk-aware scheduler is developed to maximize data acquisition for transient powered sensor nodes with NVIO architecture. Our specific contribu-

DAC '16, June 05-09, 2016, Austin, TX, USA

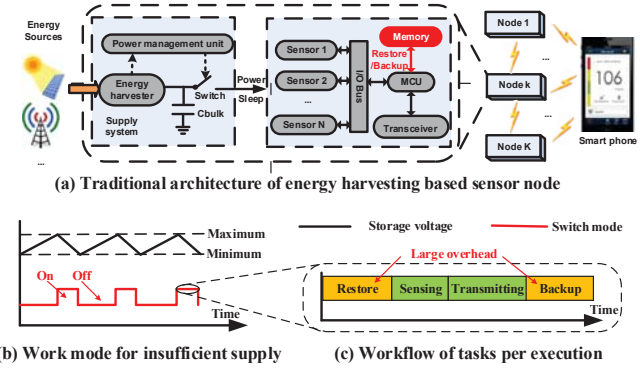
DOI: http://dx.doi.org/10.1145/2897937.2898029

tions are listed as below:

- Propose a ferroelectric NVFF(Nonvolatile Flip-flop)-based NVIO architecture to reduce IO initialization overheads, composed of a global power management unit and local NVIO interfaces.
- Extract system models for transient powered sensor nodes with NVIO and formulate an integer nonlinear programming offline scheduling problem to maximize data acquisition under given energy harvesting power traces and provide an upper bound for the proposed risk-aware heuristic online scheduler.
- Evaluate the architecture with seven benchmarks under three kinds of power traces and show that it is possible to achieve 2-5 times data acquisition by combining online IO scheduler with NVIO architecture compared with traditional sensor nodes under energy harvesting scenarios.

2. MOTIVATION

This section shows traditional energy harvesting based sensor nodes and then points out challenges of IO operations on them. As Fig. 1 has shown, energy harvesting powered sensor nodes can be deployed everywhere, which autonomously sense and transmit data to smart phones without maintenance. Each node consists of a power supply system and a computation system. The power supply system contains an energy harvester, a power switch, a power management unit (PMU) and a capacitor to buffer energy. When capacitor voltage V_{bulk} reaches a high threshold V_{max}, PMU closes power switch. Computation system executes sensing operations powered by both energy harvester and capacitor, which contains four steps: restore, sensing, transmitting and backup. If consumed power is larger than the harvested power, V_{bulk} starts to drop. Once a low threshold V_{min} is reached, PMU will generate a power down signal to all devices and computation system starts to back up configuration data. The process repeats according to the ambient energy profiles.



(a) Traditional architecture of energy harvesting based sensor node

(b) Work mode for insufficient supply (c) Workflow of tasks per execution

Figure 1: Architecture and workflow of the energy harvesting based sensor nodes

However, several IO operation challenges exist in such systems to prevent efficient data acquisition. First, large initialization overheads of restore and backup exits, leaving less time for meaningful sensing and transmitting operations. After each reboot, processor reads configuration data from nonvolatile memory and writes into sensors via a single IO bus. In backup step, all configuration data modified by user inputs have to be read and written into nonvolatile memory sequentially. Since both processes need the involvement of processor and nonvolatile memory with sequential read and write, they are quite slow and power hungry.

Second, previous task scheduler [10, 11] for energy harvesting systems omits the unpreemptive features of IO operations, where

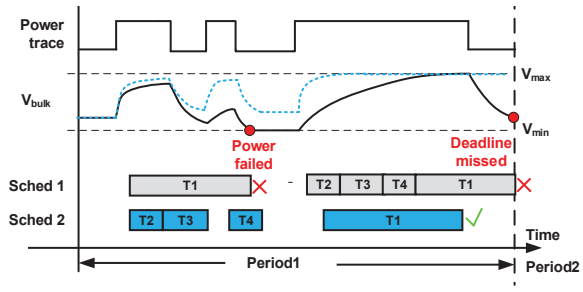


Figure 2: Motivation of energy harvesting based IO scheduling for optimal data acquisition

the capacitor voltage V_{max} is only selected to finish one restore-and-backup pair. It guarantees the system consistency in case sudden power loss happens at the beginning of a restore operation. However, such a scheduler causes significant performance and energy loss, because each IO operation needs much more energy than one restore-and-backup pair. Once the harvested power is not enough, capacitor voltage drops quickly to V_{min} in the middle of IO operation. All previous computing results are discarded and energy is wasted. Therefore, V_{max} should be adjusted at different time according to IO operation types and expected harvested power traces, instead of a fixed value in previous work.

Fig. 2 gives a motivation example to compare two ways of IO scheduling in one period. In scheduling 1, T1 is executed when a low V_{maxl} is selected. Unfortunately, power trace is also low at the same time and V_{bulk} drops to V_{min} before T1 is finished. The sensor node has to enter into backup mode and all previous execution of T1 is discarded. After that, T1 is re-executed when a high V_{maxh} is used. However, the time is too limited to complete T1 before deadline. In scheduling 2, T2-T4 is firstly executed since power trace is low. The scheduler considers the risk by trading off the value of V_{bulk} and power trace prediction and selects V_{maxp} to execute T1. With the expected harvested power, T1 is completed at a much earlier time even if V_{maxp} alone has not enough energy to complete T1. This case implies that V_{max} selection and risk-aware task scheduling are two key factors to execute IO operations in transient powered systems.

To handle those challenges, a HW/SW co-design approach is needed for optimal data acquisition. We propose a distributed nonvolatile storage and controlling hardware to relieve high initialization overheads in the centralized system, based on which we develop an intelligent power management unit and risk-aware IO scheduling to support smart V_{max} selection.

3. NONVOLATILE IO SYSTEM DESIGN

This section first illustrates nonvolatile IO system design, containing two key modules: adaptive PMU and NVIO interface. After that, initialization overheads on sensor node with nonvolatile IO system are compared with those with volatile IO system.

3.1 System Architecture

Fig. 3 shows the architecture of sensor node with nonvolatile IO modules (NVIO). Traditional volatile IO adopts DFFs(D Flip-flops) to store the IO configuration data. When power failure occurs, these data is backed up in the remote centralized Flash memory through IO bus under the control of nonvolatile processor. In NVIO, the DFFs are replaced with NVFFs, which is composed of a normal DFF and two ferroelectric capacitors. A FFC (Flip-flop Controller) is used to generate the nonvolatile control signals with the input of W/S signals from PMU. The FFC consists of a timing block and a signal generating FSM (Finite State Machine), which generates the required nonvolatile control signals [3] for NVFFs.

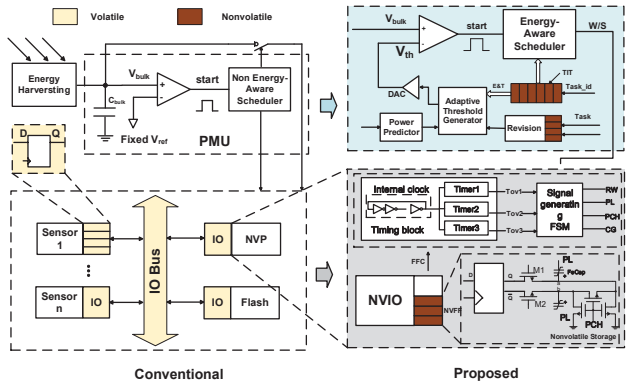


Figure 3: Nonvolatile IO system architecture for energy harvesting sensing node

When power failures happen, all configuration data is stored into local ferroelectric capacitors in parallel, leading to ultra-fast and energy-efficient data backup operation. When power is recovered, similar operations happen in the opposite direction and all configuration data is recovered.

Beside reducing the initialization overhead, we also propose an efficient PMU for IO operations. Different from the fixed V_{ref} supporting one backup-and-restore pair, the proposed PMU supports adaptive threshold voltage generation. V_{th} is decided based on the predicted power trace, sensing task energy consumption, priority and history information. A risk-aware scheduler is adopted to maximize the data acquisition. The control algorithm of PMU is illustrated in Section 5.

3.2 Evaluation of NVIO System

We extract system parameters from an ambient energy harvesting sensing platform based on NVP. Table 1 illustrates the system specifications.

Table 1: The parameters of prototype

Parameter	Value	Parameter	Value
Energy harvestor	Solar	Nonvolatile Processor	THU1010N
Process Technology	$0.13um$	Core Architecture	8051-based
Nonvolatile technology	Ferroelectric	Nonvolatile Memory	NVFF and FeRAM
Backup Energy	$23.1nJ$	Recovery Energy	$8.1nJ$
Backup Time	$7us$	Recovery Time	$3us$

To evaluate data acquisition, we investigate different sensors and extract their parameters from datasheet as shown in Table 2.

Table 2: The parameters of sensor chips

Sensor Name	Chip Model	Config. Data $(bytes)$	Sensing Data $(bytes)$	Time (ms)	Current (uA)	Voltage (V)
Proximity	APDS9130	12	2	2.7	250	3.6
Light Sensor	APDS9301	8	4	13.7	240	3
Light Sensor	NOA1305	6	2	6.25	120	3.6
Accelerometer	LIS331DLH	16	2	1	250	3.6
Gyroscope	L3G4200D	15	2	1.25	6100	3.6
Magnetic	HMC5843	3	6	20	800	3.3
Temperature	LM73	6	2	14	320	4.1

Since sensors have different data size and current consumption, the initialization time and energy vary. Fig. 4 and Fig. 5 compare time and energy consumption of one operation for sensor node with VIO(Volatile IO) and NVIO interface. The system execution time/energy can be reduced up to 17× and 25× for one data acquisition operation, consisting of restore, sensing, transfer and backup.

4. OPTIMAL DATA ACQUISITION

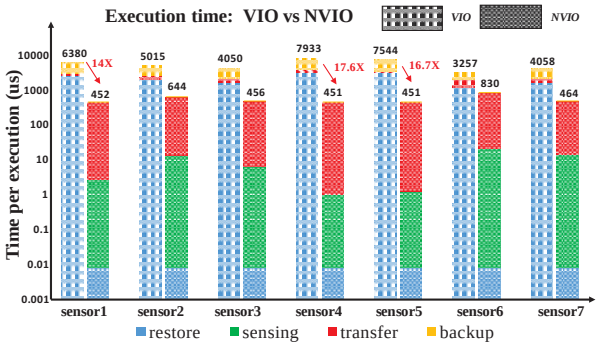


Figure 4: Time comparison between VIO and NVIO

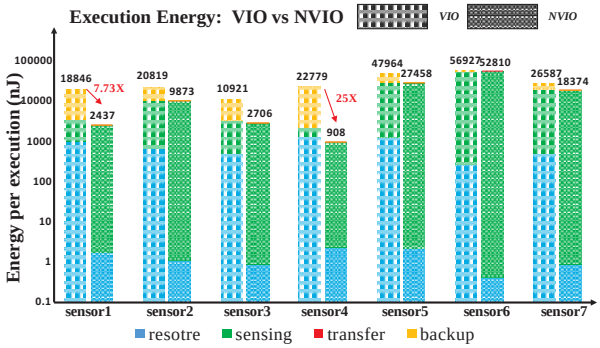


Figure 5: Energy comparison between VIO and NVIO

This section describes the system model for the energy harvesting based sensor node with proposed NVIO interface. After that, the optimal data acquisition problem is formulated.

4.1 System Modeling

The parameters of the system model are demonstrated in Table 3, including power trace, task and NVIO related parameters. Without losing generality, a set of periodic sensing tasks are executed on the energy harvesting sensor node. Each period is divided into several time slots, whose length is Δt_s and the total number is N_s^e. In each slot, the power supply can be regarded as constant when the time slot is short enough. $P_{i,k}^e$ is the average power of the ith time slot. Different sensors sense data and send them to the MCU periodically, and we denote different sensor tasks as $\tau_1, \tau_2, \cdots, \tau_{N_\tau}$. Each task has several parameters. w_n is the task weight. P_n^τ and T_n^τ represent average load power and task execution time. $S_{c\_n}^\tau$ and $S_{t\_n}^\tau$ are the data size of configuration and transmission of each task. In the NVIO interface, the baud rate B_{IO} and data transmission power E_{IO} are two important parameters. T_b and T_r are backup and restore time delay. E_b and E_r are the energy consumption of backup and restore operation per byte data.

4.2 Scheduling Variables

The boolean variable $x_{i,n}^\tau$ denotes the scheduling result of task τ_n in the ith time slot. If task τ_n is executed, $x_{i,n}^\tau = 1$, otherwise $x_{i,n}^\tau = 0$. Once task is started, sensor configuration data is restored at first, and then the sensor samples the physical information and converts it into digital signals. After that, the sensor transfers these data to the processor. Finally, configuration data are backed up and the sensor enters into sleep state. ΔE_i is the migrated energy in the ith time slot while E_i^c is the remaining energy stored in the capacitor. They can be calculated as follows:

$$\Delta E_i = (P_i^e - \sum\nolimits_{n=1}^{N_\tau} x_{i,n}^\tau \cdot P_n^\tau) \cdot \Delta t_s \quad (1)$$

Table 3: Parameters and variables of the system model

	Symbol	Description
Power Trace	$(N_s^e, \Delta t_s)$	Number of time slots in a period, each lasting Δt_s (ms)
	P_i^e	Power of harvested energy in the ith time slot
Task	V	Task set, $V = \{\tau_1, \tau_2, \cdots, \tau_{N_\tau}\}$
	w_n^τ	Sensing data weight of τ_n
	(P_n^τ, T_n^τ)	Average execution power and time length of τ_n (mW, us)
	$(S_{c\_n}^\tau, S_{t\_n}^\tau)$	Size of configuration data and sensing data of τ_n
NVIO	B_{IO}	Baud rate of IO bus
	E_{IO}	Energy consumption of IO data transmission (nJ per byte)
	(T_b, T_r)	Backup and restore time (ns)
	(E_b, E_r)	Backup and restore energy (nJ) per byte
Variable	$x_{i,n}^\tau$	Scheduling result of τ_n in the ith time slot

$$E_i^c = E_0 + \sum_{t=1}^{i} \Delta E_t \tag{2}$$

where E_0 is the initial energy stored in the capacitor.

4.3 Problem Formulation

We aim to maximize the number of sensing tasks executed successfully on energy harvesting based sensor nodes, as shown in (3).

Objective:

$$\max \sum_{i=1}^{N_s^e} \sum_{n=1}^{N_\tau} w_n^\tau \cdot x_{i,n}^\tau / T_n^\tau \tag{3}$$

Subject to:

$$E_{i-1}^c + \Delta E_i \geq 0 \tag{4}$$

$$\sum_{n=1}^{N_\tau} x_{i,n}^\tau \leq 1 \tag{5}$$

$$x_{i,n}^\tau = 1,\ where\ i \subseteq [s, s+l],\ x_{s,n}^\tau = 1,\ l \cdot \Delta t_s = T_n^\tau \tag{6}$$

Three constraints must be met. Constraint (4) is the power supply constraint. In each time slot, the supply energy plus the storage in capacitor must be larger than the executing task energy consumption. Constraint (5) means that at most one sensor can dominate the IO bus simultaneously. Constraint (6) gives the nonpreemptive requirement, where energy is wasted once the task is interrupted. Since if-then-else judgement is involved in constraint (6), the scheduling problem can be formulated as an integer nonlinear programming (INLP) problem.

The complexity of the formulation is O($2^{N_\tau \cdot N_s^e}$), where N_τ is the number of tasks and N_s^e is the number of time slots in a period. It can be solved by a nonlinear programming solver(like LINGO). In addition, the optimal results can serve as an upper bound for data acquisition in the experiments. However, the optimal results are obtained offline with known power traces, it cannot be effectively used for online scheduling due to both the high computation complexity and unknown power traces in practical applications. Thus, we develop an online task scheduling algorithm in Section 5.

5. ALGORITHM DESIGN

This section presents the online scheduling algorithm and related parameter calculations for optimal data acquisition.

5.1 Risk-aware Online Scheduling

Algorithm 1 shows the workflow of Risk-aware Online SchEduling (ROSE). The inputs include the parameters from task and power trace, and V_{min} is the minimal voltage to support system running. The output is the task scheduling results $\{x_{i,n}^\tau\}$ for optimal data acquisition. Line 1 represents the initialization process. β and γ are the risk and revision factor. Each task has three states: non-execution (0), execution (1) and failure (2). Lines 5-34 are the main loop, where we choose different control strategies according to the system execution states. When no task is executed ($state$=0), scheduling process is invoked. The future energy harvesting is predicted and the risk and calibration factor β and γ are calculated. An adaptive voltage threshold Vth'_n is calculated. If current capacitor voltage is above Vth'_n, task τ_n is executed and $state$ is set to 1. Once the task is started, the capacitor voltage is monitored constantly. Once $V_{bulk} < V_{min}$, the system enters the failure state. Current task configuration data is backed up and system returns to non-execution state. The bulk capacitor voltage V_{bulk} is updated when task τ_n is executed:

$$V_{bulk,i} = ({V_{bulk,i-1}}^2 + \frac{2 \cdot (P_i^e - P_n^\tau) \cdot \Delta t_s}{C_{bulk}})^{\frac{1}{2}} \tag{7}$$

Algorithm 1: Task scheduling algorithm

```
   input : {w_n^τ, P_n^τ, T_n^τ, S_{c_n}^τ, S_{t_n}^τ}, {N_s^e, P_i^e, Δt_s}, V_min
   output: {x_{i,n}^τ}
1  Initialization: Set V_bulk, β, γ and {δ_n}, t to V_ini, 1, 1, 1 and 0;
2                  Set {x_{i,n}^τ} and state to 0;
3                  Generate task information table;
4  for i = 1 to N_s^e do
5      if state=0 then
6          predict average power in future ΔT_p time slots;
7          calculate risk factors β_i and revision factor γ_{n,i};
8          set n = 1;
9          while n ≤ N_τ do
10             if δ_n = 1 then
11                 Vth'_{n,i}=func(β_i, γ_{n,i}, E_h, Vth_n) (11)
12                 if V_{bulk,i} ≥ Vth'_{n,i} then
13                     exe task, state = 1, t = 0, x_{i,n}^τ = 1,
                       break;
14                 end
15             end
16             n = n + 1;
17         end
18     end
19     if state = 1 then
20         calculate V_{bulk,i} by Equation (7);
21         if V_{bulk,i} ≤ V_min then
22             execution failed, set state,t to 2 and 0;
23         else
24             t = t + 1;
25             if t ≥ T_n^τ then
26                 enter non-execution, set state, t and δ_n to 0;
27             end
28         end
29     end
30     if state = 2 then
31         back up current task, update V_{bulk,i};
32         enter non-execution, set state to 0;
33     end
34     i = i + 1;
35 end
36 return {x_{i,n}^τ}.
```

5.2 Parameter Calculation

Task information table generation: The task feature is maintained in the Task Information Table(TIT). Each task item contains the task priority ρ_n^τ, minimal capacitor voltage Vth_n, execution time length T_n^τ and task progress δ_n^τ. Under the limited energy and time constraints, the task priority ρ_n^τ is related with task weight,

task energy consumption and execution time, and it is calculated as follows:

$$\rho_n^\tau = \frac{w_n^\tau}{P_n^\tau \cdot (T_n^\tau)^2} \tag{8}$$

The minimal capacitor voltage Vth_n represents the minimal capacitor energy to support a whole sensor task without any power input.

$$Vth_n = (\frac{2 \cdot P_n^\tau \cdot T_n^\tau}{C_{bulk}} + V_{min}^2)^{\frac{1}{2}} \tag{9}$$

The task progress δ_n^τ records the task execution state. $\delta_n^\tau = 0$ when task τ_n is completed, otherwise $\delta_n^\tau = 1$.

Power prediction: We adopt WCMA algorithm to predict the future energy of ΔT_p time slots [12]. WCMA is well suited for predicting a series of data with kind of periodical characteristic. Since the harvested energy is impacted by periodical circumstances, like solar energy, wind energy, heat energy etc. Note that some other prediction methods can also be used.

Adaptive threshold calculation: Considering the energy harvesting and task execution requirement, it is good to execute the right sensor task at the right time without energy waste or deadline miss. From the view point of energy, the future available energy should be predicted and fully utilized. Considering the future energy input, we can start task execution before the capacitor voltage meets Vth_n. Besides, from the view point of time, the task execution time and deadline requirement should be considered. When time is getting closer to the task deadline, there is more reason to believe that the left unexecuted tasks should try to meet the incoming deadline before capacitor is fully charged. Both factors inspire us to bring forward the task execution to make the best of the limited time and energy. We introduce the risk factor β_i to represent the deadline requirement.

$$\beta_i = \alpha \cdot \frac{(N_s^e - i) \cdot \Delta t_s - \sum_{n=1}^{N_\tau} \delta_n^\tau \cdot T_n^\tau}{N_s^e \cdot \Delta t_s - \sum_{n=1}^{N_\tau} T_n^\tau} \tag{10}$$

Historical prediction result is also used to adjust the threshold generation strategy. Intuitively speaking, if too many task execution failures occur, the adaptive adjustment should be more conservative. Otherwise, we should try more aggressive adjustment to make best use of energy. The calibration factor $\gamma_{n,i}$ is involved and the final threshold generation method is shown as follows:

$$Vth'_{n,i} = \beta_i^{\gamma_{n,i}} \cdot (Vth_n{}^2 - \frac{2 \cdot P_i^{e'} \cdot \Delta T_p}{C_{bulk}})^{\frac{1}{2}} \tag{11}$$

$$\gamma_{n,i} = \lambda \cdot \sum_{t=1}^{i} (S_n - F_n) \tag{12}$$

where S_n and F_n are the successful and failed times of τ_n, and $P_i^{e'}$ is the prediction power of the following ΔT_p time slots.

6. EXPERIMENTS

This section first shows the experimental setup, then separate hardware and software effects are explored. Finally, the HW/SW co-design benefits on task completeness is compared with the conventional IO architecture and algorithms. The optimal offline scheduling results are given out as the upper bound.

6.1 Experimental Setup

Seven benchmarks (B1-B7) are evaluated in this paper, which are composed of real sensing tasks shown in Table 1. Table 4 shows the energy and time overheads of seven benchmarks extracted either from measured results or from the datasheet. Four types of real power traces in different energy harvesting situations [1] are adopted.

Table 4: Parameters of the sensing tasks

Arch.	Item	Benchmarks						
		B1	B2	B3	B4	B5	B6	B7
NVIO	Energy (uJ)	2.44	9.86	2.70	0.91	27.4	52.8	18.3
	Time (us)	452	643	456	451	451	830	464
VIO	Energy (uJ)	18.8	20.9	10.9	22.8	47.9	56.9	26.6
	Time (us)	6380	5015	4050	7933	7545	3257	4059

6.2 Hardware Effects

The hardware effects on data acquisition are evaluated based on the optimal offline INLP scheduling. Fig. 6 presents the task completeness based on the VIO and NVIO interfaces for seven benchmarks. X-axis is the seven benchmarks while Y-axis is the task completeness. Compared with the VIO architecture, the task completeness can be increased by 66.7% on the NVIO architecture. The improvement lies in the large backup/restore overhead reduction and more energy spared for task execution.

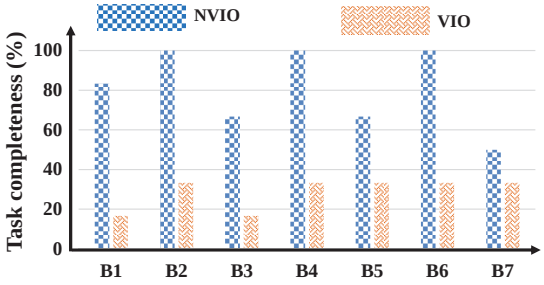


Figure 6: Comparison of the VIO and NVIO architectures

Furthermore, we analyze the impact of the energy distribution of power traces on task completeness. Fig. 7 presents the relationship between task completeness and the energy distribution in a period with 80 time slots. Three kinds of distributions are studied, where most energy lies in the early time slots(1st-20th slots), middle time slots(30th-50th slots) and late time slots(60th-80th slots) .

Counterintuitively, even if the total energy in each period is sufficient, 50% tasks on the VIO architecture still fail to be completed, because in the late time slots, many sensing tasks on nodes with VIO do not have enough time to be executed. However, NVIO interface can finish those tasks due to their shorter initialization time.

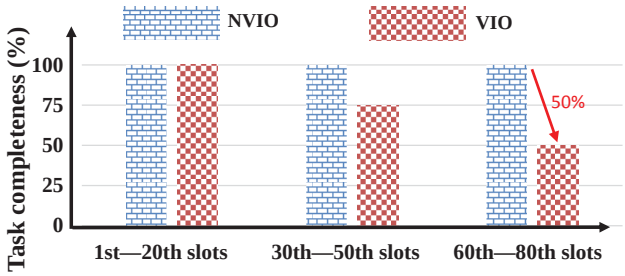


Figure 7: Analysis of energy distribution of power trace

6.3 Software Exploration

The ROSE algorithm with adaptive and fixed V_{th} is compared with traditional aggressive algorithm [10] (baseline), which executes task as soon as the energy accumulated in capacitor is just enough for a restore-and-store pair. In this part, ROSE algorithm adopts two approaches to choose the capacitor voltage to start tasks. When a fixed V_{th} is used, the energy storage in capacitor should be large enough to make sure the task can be finished even without any harvested energy. To achieve even better performance, an adaptive V_{th} is used to start tasks with energy prediction and risk analysis. The optimal offline scheduling results are used as the

upper bound and all other results are normalized with the INLP results. Fig. 8 shows the relative task completeness of all algorithms for seven benchmarks. The proposed algorithm achieves up to 50% improvements.

Fig. 9 shows the relative task completeness of all algorithms for three kinds of power traces including early, middle and late distribution. For ROSE with adaptive V_{th}, the relative task completeness becomes lower (from 1.0 to 0.67) when most energy arrives later. Due to overestimation of power in future, ROSE tends to use more aggressive scheduling, inducing energy wastes in case rollbacks happen due to power failure. On the opposite, ROSE with fixed V_{th} is more conservative and tasks tend to be executed much later. However, they suffer from time limitation when energy comes too late. The traditional algorithm is the worst of the three for omitting the unpreemptive features of IO operations.

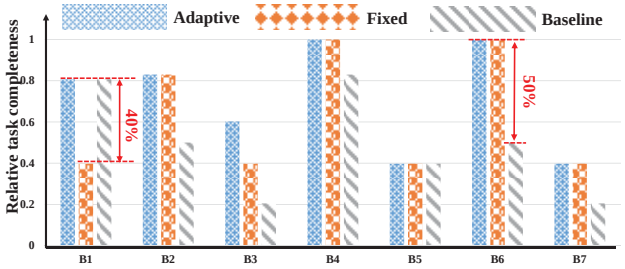


Figure 8: Relative task completeness for seven benchmarks

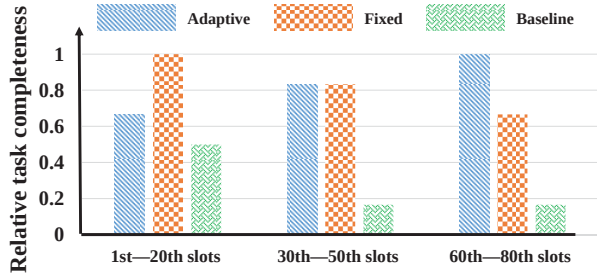


Figure 9: Relative task completeness for three kinds of power traces

6.4 HW/SW Co-design Benefits

In this part, we explore the system task completeness improvement adopting both hardware and software designs. NVIO and VIO architecture, together with three kinds of task scheduling algorithms as discussed in section 6.3, form six combinations for seven benchmarks. As shown in Fig. 10, the HW/SW co-design in red grid bar improves the task completeness by 2-5 times compared with conventional architecture using VIO and baseline algorithm in blue slash bar. In the largest benefit improvement application the hardware occupies up to 60% improvement and the software takes up the rest.

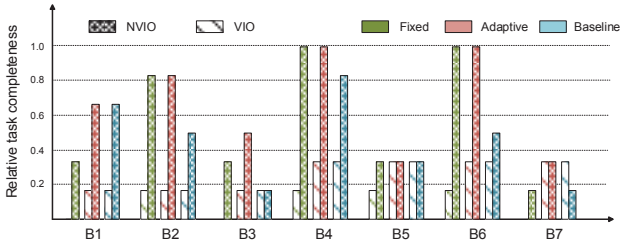


Figure 10: Task completeness improvement of HW/SW co-design

7. CONCLUSIONS

This paper demonstrates that the nonvolatile IO greatly reduces the response time and energy consumption during power failures, leaving more time and energy for effective execution in energy harvesting applications. Furthermore we proposed a offline and online sensor task scheduling algorithm to maximize the data throughput. The study shows that the previous task scheduling algorithm induces large energy waste omitting the unpreemptive features of IO operations. The fixed V_{th} selection has the problem of deadline miss. The data throughput of ROSE online scheduling algorithm is very close to the offline INLP algorithm. The experimental result shows that the HW/SW co-design improves the task completeness by 2-5 times compared with conventional architecture using VIO and baseline algorithm.

8. ACKNOWLEDGMENTS

This work was supported in part by NSF Grant #61271269 and under grant from the Research Grants Council of the Hong Kong SAR, China (Project No. CityU 11214115). Corresponding author: Yongpan Liu,ypliu@tsinghua.edu.cn.

9. REFERENCES

[1] K. Ma and et al. Architecture exploration for ambient energy harvesting nonvolatile processors. In *HPCA'15*, pages 526–537, 2015.

[2] R. Benjamin and et al. Mementos: system support for long-running computation on rfid-scale devices. *ACM SIGPLAN Notices*, 47(4):159–170, 2012.

[3] Yiqun Wang and et al. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *ESSCIRC'12*, pages 149–152, 2012.

[4] Y. Liu and et al. 4.7 a 65nm reram-enabled nonvolatile processor with 6x reduction in restore time and 4x higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic. In *ISSCC'16*, 2016.

[5] Yongpan Liu and et al. Ambient energy harvesting nonvolatile processors: from circuit to system. In *DAC'15*, pages 1–6, 2015.

[6] Mimi Xie and et al. Fixing the broken time machine: Consistency-aware checkpointing for energy harvesting powered non-volatile processor. In *DAC'15*, pages 1–6, 2015.

[7] Joel C. Wu and et al. Storage access support for soft real-time applications. In *RTAS'04*, pages 1–8, 2004.

[8] V. Swaminathan and et al. Energy-conscious, deterministic i/o device scheduling in hard real-time systems. *TCAD*, 22(7):847–858, 2003.

[9] H. Cheng and et al. Online energy-aware i/o device scheduling for hard real-time systems. In *DATE'06*, pages 1055–1060, 2006.

[10] Domenico Balsamo and et al. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *Embedded Systems Letters, IEEE*, 7(1):15–18, 2015.

[11] Daming Zhang and et al. Deadline-aware task scheduling for solar-powered nonvolatile sensor nodes with global energy migration. In *DAC'15*, pages 1–6, 2015.

[12] J.R. Piorno and et al. Prediction and management in energy harvested wireless sensor nodes. In *Wireless VITAE'09*, pages 6–10, May 2009.