Data Backup Optimization for Nonvolatile SRAM in Energy Harvesting Sensor Nodes

Yongpan Liu, *Senior Member, IEEE*, Jinshan Yue, Hehe Li, Qinghang Zhao, Mengying Zhao, Chun Jason Xue, Guangyu Sun, *Member, IEEE*, Meng-Fan Chang, *Senior Member, IEEE*, and Huazhong Yang, *Senior Member, IEEE*

***Abstract*—Nonvolatile static random access memory (nvSRAM) has been widely investigated as a promising on-chip memory architecture in energy harvesting sensor nodes, due to zero standby power, resilience to power failures, and fast read/write operations. However, conventional approaches back up all data from static random access memory into nonvolatile memory when power failures happen. It leads to significant energy overhead and peak inrush current, which has a negative impact on the system performance and circuit reliability. This paper proposes a holistic data backup optimization to mitigate these problems in nvSRAM, consisting of a partial backup algorithm and a run-time adaptive write policy. A statistic dead-block predictor is employed to achieve dead block identification with trivial hardware overhead. An adaptive policy is used to switch between write-back and write-through strategy to reduce the rollback induced by backup failures. Experimental results show that the proposed scheme improves the performance by 4.6% on average while the backup power consumption and the inrush current are reduced by 38.1% and 54% on average compared to the full backup scheme. What is more, the backup capacitor size for energy buffer can be reduced by 40% on average under the same performance constraint.**

***Index Terms*—Cache memories, dead blocks, energy harvesting systems, nonvolatile static random access memory (nvSRAM), sensor nodes.**

I. Introduction

ENERGY harvesting systems have been widely used in habitat monitoring, volcano monitoring, and structural monitoring due to its ultralong operating time without maintenance [1]. However, ambient power inputs suffer from frequent failures. For instance, ambient radio frequency (RF) power varies according to power sources, frequency, distance, height, and obstacles [2]. Although independent sensing may be done on those devices under frequent interrupted supply, it may not work for more complicated history-dependent operations. In such cases, to ensure continuous operations on battery-less systems is a challenging task.

To maintain states under an intermittent power supply, emerging nonvolatile memory (NVM) has been proposed to replace volatile memories in various levels, such as register files, static random access memories (SRAMs), and dynamic random access memories. Typical emerging memory candidates include phase change memory [3], [4], spin-transfer torque magnetic RAM [5], [6], and resistive memory (RRAM) [7]–[10]. However, direct replacement of the volatile register/cache with NVM suffers from the drawbacks of high write energy, slow write speed, asymmetric read/write operations, and low endurance. In order to overcome these drawbacks, previous studies of hybrid memory technologies [11], loop retiming [12], and the 3-D-integrated hybrid memory technique [13] have explored the architectural level optimization to improve the performance of nonvolatile caches and nonvolatile main memory.

Another aspect of research focuses on narrowing the performance gap by combining SRAM and NVM, namely, the nonvolatile SRAM technique. Nonvolatile SRAM (nvSRAM) integrates an SRAM cell and a nonvolatile element in cell levels, forming a direct bit-to-bit connection in a 2-D or vertical arrangement to achieve fast parallel data transfer and power-on/off speed [14]. Therefore, it provides comparable power and performance metrics as SRAM, while keeping the nonvolatile capability when power failures happen. However, conventional backup strategies which back up full data from SRAM into NVM in parallel face unnecessary large backup energy consumptions and high inrush current, which may cause significant energy and reliability issues. As a result, various backup polices have been proposed for nonvolatile memories, such as the sequential backup [15] and compression techniques [16], [17]. However, sequential backup prolongs the backup time and compression techniques introduce extra hardware overhead. What is more, all of the above methods adopt an over-pessimistic *full backup* scheme, which assumes that full data should be stored and is not efficient.

This paper proposes a systematic partial backup strategy for nvSRAM. It consists of two techniques: 1) a partial backup technique with statistics-based dead-block prediction (SBDP)

Manuscript received April 21, 2016; revised August 30, 2016; accepted November 27, 2016. Date of publication January 5, 2017; date of current version September 14, 2017. This work was supported in part by NSFC under Grant 61674094, and in part by the Research Grants Council of the Hong Kong under Project CityU 11214115. This paper was recommended by Associate Editor J. Henkel. *(Corresponding author: Yongpan Liu.)*

Y. Liu, J. Yue, H. Li, Q. Zhao, and H. Yang are with the Department of Electrical Engineering, Tsinghua University, Beijing 100084, China (e-mail: ypliu@tsinghua.edu.cn).

M. Zhao is with the School of Computer Science and Technology, Shandong University, Jinan 250100, China.

C. J. Xue is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

G. Sun is with the Center for Energy-Efficient Computing and Applications, EECS School, Peking University, Beijing 100871, China.

M.-F. Chang is with the Department of Electrical Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan.

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

Digital Object Identifier 10.1109/TCAD.2017.2648841

and 2) a run-time adaptive write policy (AWP). Based on the statistics of recent used bits (RUBs), SBDP can significantly reduce the backup energy with trivial hardware overhead. Hence, it is suitable for nvSRAM backup in embedded applications. Besides, a run-time AWP is proposed to reduce the occurrences of rollback caused by excessive dirty blocks.

The fundamental challenges lie in identifying dead blocks accurately and efficiently based on a simple hardware and designing proper write policy switching method to trade-off between different kinds of performance penalties. The contributions of this paper are as follows.

1) We propose a systematic backup strategy for nvSRAM, which reduces energy and performance penalties in both the backup and executing stage.
2) A partial backup strategy with an energy-efficient SBDP method is presented. It can figure out dead blocks with trivial hardware overhead and its accuracy is comparable to state-of-the-art techniques.
3) An online scheduling scheme to accomplish the AWP process is proposed. The scheme can mitigate the performance penalty induced by partial backup.
4) We implemented the proposed techniques and evaluated them with MiBench [18] in gem5 simulator [19]. Experimental results show that the proposed scheme improves the performance by 4.6% on average while the backup power consumption and the inrush current are reduced by 38.1% and 54% on average compared to the full backup.

The rest of this paper is organized as follows. Section II discusses the background of the proposed strategy. Section III formulates the systematic backup strategy for nvSRAM. SBDP and AWP are illustrated in Sections IV and V, respectively. Section VI shows the experimental results and we conclude this paper in Section VII.

II. Background

This section first describes the typical architecture of an energy harvesting sensor node which is the target platform of this paper. After that, related works are provided.

A. Target Platform and Challenges

The target platform can be but not limited to the energy harvesting sensor node shown in Fig. 1. It consists of an energy harvesting module, a nonvolatile processor (NVP), voltage detection circuits (VDCs), bulk capacitors, peripheral sensors, off-chip data storage, and a wireless transceiver. The power supply system consists of the dynamic power management (DPM) circuit and two capacitors C_{bulk0} and C_{decoup}, which are connected in parallel with the PV cell. Because of the unstable characteristics of ambient energy harvesting, the power supply usually oscillates between ON and OFF state. Thus, the sensor node runs intermittently by turning the DPM load switch off and on to keep the supply voltage in an operational range [20]. The employment of nvSRAM enables the system to be suspended during power failures. The VDC detects a power drop when a power failure arrives. The VDC generates a backup signal to the NVP, which starts the backup

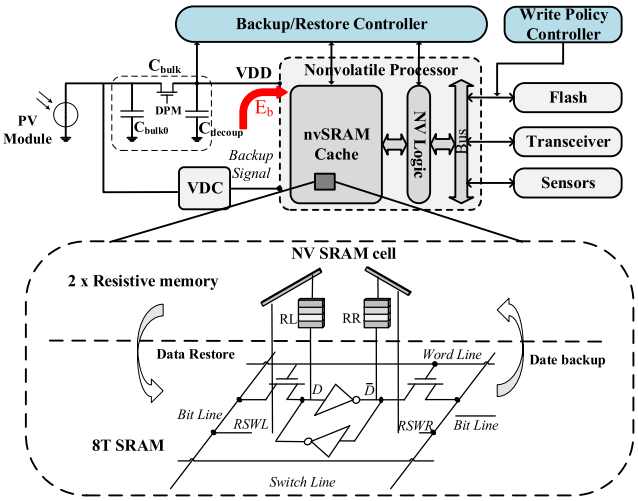


Fig. 1. Typical energy harvesting sensor node fabricated with nonvolatile memories. The structure of 8T2R nvSRAM [14] is depicted in the bottom figure.

operations. The capacitors provide energy to the workload in order to finish the backup process.

However, the backup process of nvSRAM is both time and energy consuming. Therefore, the major challenge of the system lies in the backup efficiency problem, which brings the following nontrivial challenges.

1) *Area Overhead:* Large C_{bulk} is required to provide enough energy for the backup process. The chip area overhead induced by a large on-chip C_{bulk} is nontrivial. For instance, a 4 KB nvSRAM in a state-of-the-art RRAM-based NVP [21], requires a 172 nF on-chip capacitor to supply the backup energy. It leads to over **1.6X** larger chip area compared to the original chip without C_{bulk}.
2) *Inrush Current Overhead:* The fully parallel backup causes a large inrush current up to several amperes, which probably burns surrounding circuits and induces IR drop to harm the reliability of the entire system [22].
3) *Performance Penalty:* The backup/restore process prolongs the total execution time.

In order to mitigate the above challenges, previous works have investigated various approaches to optimize the backup efficiency in term of both hardware and software levels. First, the structure of nvSRAM is optimized to accelerate the backup process. Second, plenty of backup policies for nvSRAM have been proposed to improve the efficiency.

B. Related Work

1) Nonvolatile SRAM: Chiu *et al.* [14] proposed an RRAM-based 8T2R nvSRAM depicted in Fig. 1. It consists of a 6T standard SRAM cell, 2T RRAM-switch (RSWL and RWSR) and two resistive devices (RL and RR) vertical-stacked over the 8T SRAM cell. This design achieves fast bit-to-bit parallel store/restore operations, but the parallel backup size is limited by the inrush current. Lee *et al.* [23] presented a 7T1R nvSRAM design to reduce the store energy by using

a single NVM device and suppress dc-short current during restore operations using a pulsed-overwrite scheme. Moreover, several works have been proposed to optimize the power gating technique of nvSRAM. Ohsawa *et al.* [24] presented a fabricated 32b fine-grained power gating technique applied to the memory cell array using the 4T2MTJ cell. Shuto *et al.* [25] explored the power gating architecture for nvSRAM to save the static power without degradation of the circuit performance. In summary, these works mainly focus on the optimization of nvSRAM in circuit level, while this approach focuses on the architectural-level backup optimization.

2) Backup Policies: Various backup policies have been explored to improve the backup efficiency and lower the inrush backup current in traditional designs. Although the fast *parallel backup* policies [14], [21] can be reasonable for nonvolatile register files, it requires unaffordable backup energy and peak current when used in nvSRAM. *Sequential backup* reduces the inrush current by prolonging the backup time [15], but the backup energy is not reduced and the time is prolonged. To shorten the sequential backup time, progressive charging to power-gated modules [26] can be adopted under peak current constraint. Furthermore, *compression before backup* [16], [17] helps to reduce data to be copied with extra computation overhead. However, above methods adopt a *full backup* scheme to store all data in nvSRAM, which is unnecessary and energy inefficient. Several *partial backup* techniques have been raised to avoid unnecessary backup operations. Ma *et al.* [27] analyzed the partial backup strategies for in-order and out-of-order NVPs. But it does not take the impact of the cache part into consideration. Tsai *et al.* [28] proposed novel read circuits to eliminate redundancy in bit levels, where data is backed up only if there is a difference between the value in the SRAM cell and that in the nonvolatile element. However, both of them omit architecture-level optimization.

As previous works either back up all data or mainly focus on the device and circuit level optimization for nvSRAM, we explore the orthogonal architecture-level optimization to improve energy efficiency and reduce peak current, which can be combined with previous solutions.

III. Problem Overview

We first describe the hardware and performance models, followed by the goal of the backup procedure for nvSRAM. Then, how to achieve the goal by the system level backup procedure design is discussed. Finally, the proposed backup flow is presented.

A. Hardware Model

The structure of the hardware can be divided into three parts: 1) the energy harvester; 2) the workload which is taken as an energy consumer; and 3) the bulk capacitor C_{bulk} working as an energy buffer for the system.

1) Energy Harvester Model: Passive power-down occurs under various conditions for ambient energy harvesting energy sources. For instance, power-down occurs in an ambient RF harvesting system, when the power source moves out of the acceptable range or stops working. According to the energy harvesting system in Fig. 1, we use discrete power pulses to model the power collected by the energy harvester, denoted as a power pulse set $\mathcal{P} = \{\rho_1, \rho_2, \ldots, \rho_m, \ldots\}$. Since power-down triggered by events are independent and random, the power-on time T_i for power pulse ρ_i follows the exponential distribution, that is

$$f_{T_i}(t_i) = \begin{cases} \lambda e^{-\lambda t_i} & \text{if } t_i \geq 0 \\ 0 & \text{if } t_i < 0 \end{cases} \tag{1}$$

where $f_{T_i}(t_i)$ is the probability density function and $(1/\lambda)$ is the expected value of power-on time. The voltage and power-on time T_i associated with each power pulse ρ_i can be extracted from real energy harvesting traces [27].

2) Workload Model: The workload consists of memory, computational logic, and peripheral devices. Memory and computational logic are fabricated with nonvolatile devices. The nvSRAM is used as the basic unit of the M-way associative L1 cache with N blocks B$\{b_1, b_2, \ldots, b_N\}$. The least recently used (LRU) replacement strategy is employed. Therefore, each block b_i can be characterized by a 3-tuple (v_i, r_i, d_i), where v_i denotes the valid bit, r_i denotes the RUB used as a symbol for the reference order of intraset cache blocks, and d_i denotes the dirty bit for block i. For an NVP with a N-block nvSRAM, the energy consumption in the backup process can be expressed as follows:

$$E_{\text{bk}} = E_{\text{lg}} + E_b(N_b + N_{\text{et}}) \tag{2}$$

where E_{lg} is the backup consumption of the nonvolatile logic part, E_b is backup energy consumption of a cache block, N_b is the number of blocks to be backed up, and N_{et} is the equivalent extra blocks introduced by the hardware overhead. Compared to the full backup strategy, the proposed partial backup scheme can help to reduce the backup blocks, namely, $N_b \leq N$. The run-time parameters of the system consist of the frequency F and the system power consumption P.

3) Bulk Capacitor Model: The bulk capacitor provides energy for the workload to continuously execute and to back up data after power loss. The workload only operates in a feasible voltage range $[V_l, V_h]$. After power-off, the energy from C_{bulk} is first partially used for workload execution, after which the rest energy in C_{bulk} supports to complete the backup process. C_{bulk} is discharged to V_l when the backup process is finished. After that, the whole system enters into the power-off state. Once power is on, C_{bulk} is recharged from V_l to V_h. Therefore, the backup energy budget is expressed as

$$E_{\text{budget}} = C_{\text{bulk}} \frac{V_h^2 - V_l^2}{2}. \tag{3}$$

The backup policy design takes full use of the energy in C_{bulk} to execute tasks while guaranteeing the backup process.

B. Performance Model

Fig. 2 shows the diagram of the executed clock cycles, transition of work states, input power pulses, and the voltage of C_{bulk}. During a complete period of a power pulse, the system lives through the recovery, normal execution, extra-execution, backup, and idle modes for each power pulse.

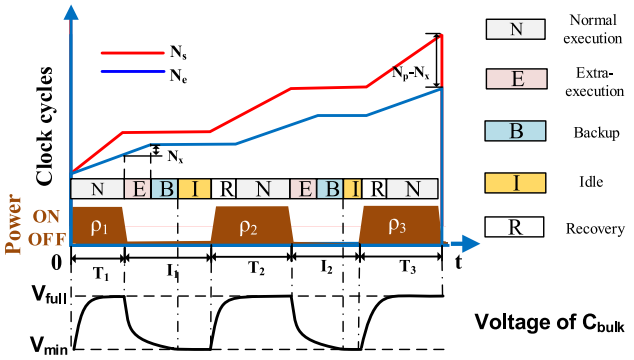


Fig. 2. Diagram of the power pulses ρ and the work states of the system. The curves show the elapsed cycles N_s and effective cycles N_e. The relationship between N_e, N_x, and N_s is also shown in the figure.

The blue line and the red line in Fig. 2 represent the increasing trend of the effective cycles N_e and elapsed cycles N_s. The *effective cycles*, denoted as N_e, is defined as the total amount of the cycles that are spent on the actual execution of the program. The *elapsed cycles*, denoted as N_s, are defined as the elapsed cycled during power-on time. Moreover, N_p represents the overhead cycles and N_x refers to the extra cycles powered by the capacitor energy. Therefore, their relationship is represented as follows:

$$N_e = N_s - N_p + N_x. \tag{4}$$

The penalty cycles N_p refers to the cycle overhead caused by the power interruptions. There are five kinds of performance penalties in a partial backup process.

1) *Backup-Failure Penalty:* It occurs when the capacitor energy cannot supply the nvSRAM to back up all dirty blocks. Assuming it happens at a probability of P_{fail}, the system should roll back to its previous backup point, which wastes N_{roll} cycles.
2) *Recovery Penalty:* It happens during the NVP initialization, denoted as N_{rec}, which consists of four parts: a) capacitor charging; b) reset IC delay; c) nonvolatile controller delay; and d) nonvolatile flip-flops (NVFFs) restore delay [29].
3) *Force-Discard Penalty:* It is induced in the partial backup process when all dirty blocks can be backed up, but the live blocks cannot. N_{force} cycles are induced to refetch data due to cache misses.
4) *Miss-Prediction Penalty:* It is caused by wrong predictions from the dead block prediction. A penalty will happen only when a live block is predicted to be dead. It leads to N_{miss} cycles to refetch cache blocks.
5) *Extra Write-Stall Penalty:* It is defined as the extra write stalls induced by temporary write through policies. It is denoted as N_{stall}. Details will be addressed in Section V.

In general, the total time penalty is expressed as follows:

$$N_p = N_{\text{roll}} + N_{\text{rec}} + N_{\text{force}} + N_{\text{miss}} + N_{\text{stall}}. \tag{5}$$

C. Optimization Goal

Table I summarizes all parameters and definitions in this paper. We use $N_e^{(k)}$, $N_s^{(k)}$, $N_x^{(k)}$, and $N_p^{(k)}$ to denote the effective cycles, elapsed cycles, extra execution cycles, and the penalty cycles of the task during the power pulse k. If n pulses are just enough for completing the program, the following inequation must be satisfied:

$$n = \underset{k}{\text{argmin}} \sum_{k} N_e^{(k)} \geq \text{SF} \tag{6}$$

where SF denotes the required computation cycles of the system. Furthermore, the actual execution time is represented as follows:

$$T_a = \sum_{i=1}^{n} T_i + \sum_{i=1}^{n-1} I_i. \tag{7}$$

The total backup energy is defined as follows:

$$E_t = \sum_{i=1}^{n} E_{\text{bk}}^{(k)}. \tag{8}$$

The main optimization objective is to maximize the backup efficiency η of all tasks, which is defined as follows:

$$\eta = \frac{1}{T_a E_t}. \tag{9}$$

Equation (9) indicates that the backup efficiency η is related to both the execution time and the total backup energy. The proposed partial backup policy helps to reduce the backup energy, which mitigates the chip area and inrush current problem. Meanwhile, backup energy reduction saves more energy for task execution, which is beneficial to the overall performance T_a.

TABLE I
PARAMETERS AND VARIABLES IN THE SYSTEM MODEL

	Symbol	Description
Power	$\mathcal{P}$	Power pulse set, $\mathcal{P} = \{\rho_1, \rho_2, ...\rho_m, ...\}$
	T_k	Time duration of power pulse ρ_k
	I_k	Idle time duration after each time power pulse.
System	$\mathcal{B}$	A cache block set $\mathcal{B}\{b_1, b_2, ..., b_N\}$ for a M-way associative L1 cache with N blocks
	(v_i, d_i, r_i)	v_i denotes the valid bit for each cache block i. d_i denotes the dirty bit for the block i. r_i denotes the recently used bits (RUB) for cache block i.
	C_{bulk}	The capacitance volume of the bulk capacitor.
	$[V_l, V_h]$	The operation voltage range of the bulk capacitor.
	E_{lg}	Backup energy of the logic part in the nonvolatile processor.
	E_b	Backup energy of each cache block.
	$E_{bk}^{(k)}$	Backup energy on power pulse k.
	N_{et}	Equivalent extra blocks.
Performance	$N_e^{(k)}$	Effective cycles on power pulse k
	$N_p^{(k)}$	Penalty cycles on power pulse k
	$N_s^{(k)}$	Elapsed cycles on power pulse k
	$N_x^{(k)}$	Extra execution cycles on power pulse k
	S	Initial task execution time without power interruption.
	F	Operation frequency

D. Workflow of Proposed Method

As the optimization goal can be achieved by reducing the backup energy and maximizing the effective cycles of each

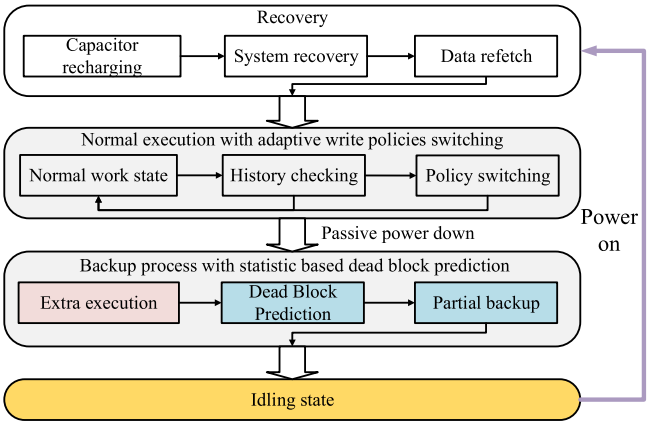


Fig. 3. Proposed partial backup flow.

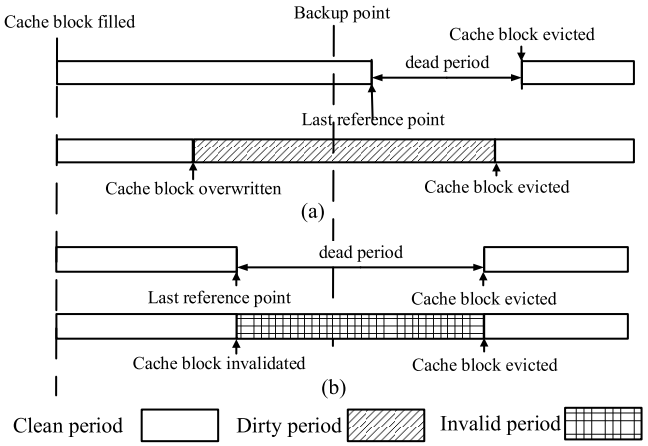


Fig. 4. Cache block classification. (a) Live block. (b) Dead block.

power pulse k. A systematic work flow for each power pulse is proposed to achieve the optimization goal in Fig. 3. The meaning of the colors in Fig. 3 matches those in Fig. 2. It consists of four stages. After power is on, the system starts with the recovery process. The backup capacitor is recharged and the data are reloaded from NVM. After that, cache data is refetched due to data loss in the partial backup process. Recovery penalties are caused in this stage. After the successful initialization, the system enters into the normal execution stage with an AWP scheme, which operates in the normal mode with the write-back policy. A write policy switching is triggered if the number of dirty blocks exceeds the maximal backup threshold. A history checking process is adopted to select the policy, followed by the policy switching. AWP is aimed to reduce the backup-failure and force-discard penalty so as to reduce N_p. Namely, this strategy targets to reduce the execution time T_a.

When a passive power down is detected, the system enters into the partial backup procedure. First, the execution process is continued by the energy in C_{bulk}. After the current task is completed, the dead block prediction is carried out to choose dead blocks. The partial backup only keeps the live blocks. The partial backup strategy aims to reduce E_{bk} and to prolong the current task execution. Therefore, it reduces T_a. Finally, the system enters into the power-off state. The proposed partial backup strategy includes two key techniques: 1) the dead block prediction technique and 2) the AWPs switching technique, which will be discussed in Sections IV and V.

IV. Statistics-Based Dead-Block Prediction Algorithm

This section presents the lightweight but accurate SBDP for the partial backup process. We introduce the concept of dead blocks and previous works on dead block prediction. After that, we discuss the theory basis and quantitative analysis of SBDP. Finally, the circuit implementation is given out.

A. Cache Block Classification

Fig. 4 shows the concept of live/dead blocks. The length of each hollow rectangle indicates the lifetime of each cache block. A cache block is alive between the moment when it is filled and the last reference time point by a read/write request. Furthermore, the death period is defined as the time interval between two successive cache lifetimes. Fig. 4(a) shows two cases of live blocks in a write-back cache. At the backup time point, both cache blocks are in a live period. If a cache block is modified, it is marked as a dirty block. The shadow rectangle in the median block presents a dirty period. These two cases have different performance penalties when discarded wrongly. Discarding clean & live blocks causes miss-prediction penalty or force-discard penalty, while discarding dirty blocks leads to backup failure penalties. Fig. 4(b) shows two cases of dead blocks. A cache block being in a dead period or in invalid state at the backup time point is classified into a dead block. Dead block prediction techniques can help to distinguish live and dead blocks.

In fact, there are plenty of architecture-level works to predict dead blocks, such as PC-based [30], [31], counting-based [32], and time-based [33] predictions. The PC-based [30], [31] prediction integrates PC bits into the cache tag fields and a history table is used to track the reference order of cache blocks. Counting-based [32] dead block prediction uses a counter to keep track of the approach to identify dead lines for cache prefetching. Time-based [33] prediction is based on the principle that a cache line will likely to be dead after it is replaced after a cache burst happens. However, these complex prediction methods are dedicated to high performance applications and introduce unacceptable hardware overheads for low power systems. Considering the burst count prediction in [33], it requires more than 2 KB extra hardware to deliver high accuracy. The hardware overhead is even comparable to the on-chip nvSRAM. The high storage overhead also introduces energy overhead to cause extra performance penalty, while SBDP can figure out dead blocks with much lower overhead.

B. Prediction Theory Basis

The main principle of SBDP lies in the correlation between the dead blocks and the cache replacement algorithm. LRU is a common cache replacement policy. When a cache set is full and a new block should be fulfilled, the block that has been

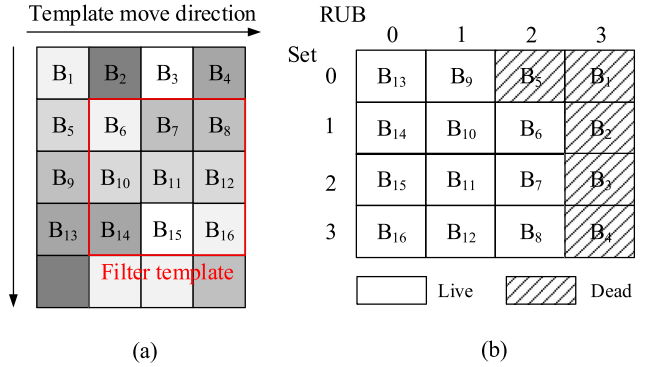


Fig. 5. Example of the correlation between the RUBs and the dead/live blocks. (a) Image filtering application and the movement of the template. (b) Data in the four-way associated cache.

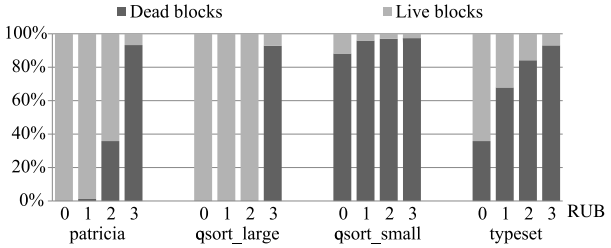


Fig. 6. Dead/live blocks ratio versus RUBs for four benchmarks from mibench.

unused for the longest time is replaced. RUBs are added to each cache block to represent the access order. Cache blocks with larger RUB are accessed longer time before those with smaller RUB. Fig. 5 takes the application of image filtering running on a four-way associated cache as an example to show the principle. Assume that the cache block size is equal to that of a single pixel. The filter template moves from left to right and from top to bottom in Fig. 5(a). Image blocks are filled into the corresponding cache set according to address mapping and the order of RUBs. Cache blocks marked with the shadow are dead blocks because they would not be covered by the template again. Fig. 5(b) gives us the intuition that cache blocks with large RUBs are more likely to be dead. Moreover, the correlation between the dead/live blocks and RUBs has the characteristic of time locality. It means that the distributions of dead/live blocks are stable in different cache sets for an application. Therefore, the theory basis of SBDP lies in the correlations between the distribution of RUBs and the classification of dead/live blocks.

Fig. 6 shows the average block type ratio under different RUBs when four benchmarks from Mibench [18] are running on a four-way associated cache. It is observed that the distribution of block type is different under various RUBs. In Patricia, all cache blocks of RUB $= 0$ and RUB $= 1$ are live. More than half of blocks are live when RUB $= 2$, while most of blocks are dead when RUB $= 3$. However, in qsort_large, almost all cache blocks are dead. Specifically, the ratio of dead/live blocks increases/decreases when the RUB becomes larger. This observation enables us to predict the block types (dead/live) based on the RUB information. As we can see, different benchmarks show various distributions. Therefore, a dynamic statistical approach should be used to train and get this correlation for different benchmarks.

C. Quantitative Analysis

We use the binary variable l_i to indicate the prediction results for each cache block, where $l_i = 1$ ($l_i = 0$) means that block i is predicted to be live (dead). The prediction problem can be formulated as a logistical regression problem, that is

$$P(l_i = 1|r_i) = h_\theta(r_i) \tag{10a}$$

$$P(l_i = 0|r_i) = 1 - h_\theta(r_i) \tag{10b}$$

where $h_\theta(x)$ is the hypothesis function and θ is a key parameter of the function. If $P(l_i = 1|r_i) > P(l_i = 0|r_i)$, then the corresponding cache block is predicted to be live. Otherwise, it is predicted to be dead. Therefore, the first step is to derive an approximate hypotheses function $h_\theta(x)$ by referring to historical inputs. Then, predictions are made based on outputs of $h_\theta(x)$. We adopt the perceptron learning to make the classification, which is the simplest neural network for single-layer perception. Compared with other candidate algorithms, such as back propagation network, radial basis function, and deep belief network, it achieves small error rate with trivial energy and hardware overhead. We further give the quantitative details of the training algorithm.

1) Input Features Rearrangement: The input features r_i are discrete variables and the number of feasible values is set by the cache association. There are plenty of repeated training examples if all pairs of (r_i, l_i) are used for training, which leads to a lot of redundant operations in the training step. Instead, we use an input features rearrangement strategy to diminish the size of input features. It is achieved by normalizing the input features r_i to the ratio of dead/live blocks for each RUB $= j$

$$L_j = \frac{\sum_{i=1}^{N} I\{r_i = j, l_i = 1\}}{\sum_{i=1}^{N} I\{r_i = j\}} \text{ for } j = 0, 1, 2, \ldots, M-1 \tag{11a}$$

$$D_j = \frac{\sum_{i=1}^{N} I\{r_i = j, l_i = 0\}}{\sum_{i=1}^{N} I\{r_i = j\}} \text{ for } j = 0, 1, 2, \ldots, M-1 \tag{11b}$$

where $I(\bullet)$ is the indicator function. It equals to one when its argument is true, and vice versa. M is the degree of cache association. After applying the input feature rearrangement strategy, we get dead/live block ratio for each RUB $= j$ and the input vectors are $\boldsymbol{L}$ and $\boldsymbol{D}$. Because the prediction is carried out online, we use local input vectors ($\boldsymbol{D}^{(e)}$ and $\boldsymbol{L}^{(e)}$) as the training inputs. A decay rule is adopted to update ($\boldsymbol{D}^{(e)}$ and $\boldsymbol{L}^{(e)}$). After getting a new sample of the input vector tuple ($\boldsymbol{D}^{(s)}, \boldsymbol{L}^{(s)}$), the estimation of local input tuple is updated according to the following rule:

$$\boldsymbol{L}^{(e)}_{\text{new}} = \frac{1}{2}\boldsymbol{L}^{(e)}_{\text{old}} + \frac{1}{2}\boldsymbol{L}^{(s)} \tag{12a}$$

$$\boldsymbol{D}^{(e)}_{\text{new}} = \frac{1}{2}\boldsymbol{D}^{(e)}_{\text{old}} + \frac{1}{2}\boldsymbol{D}^{(s)} \tag{12b}$$

where $\boldsymbol{L}^{(e)}_{\text{old}}$ and $\boldsymbol{D}^{(e)}_{\text{old}}$ are estimated values based on previous $s-1$ samples. ($\boldsymbol{L}^{(e)}_{\text{new}}, \boldsymbol{D}^{(e)}_{\text{new}}$) is the new estimated tuple when the sth sample is finished. Equation (12) gives a higher weight to the latest sample because it is closer to the real distribution.

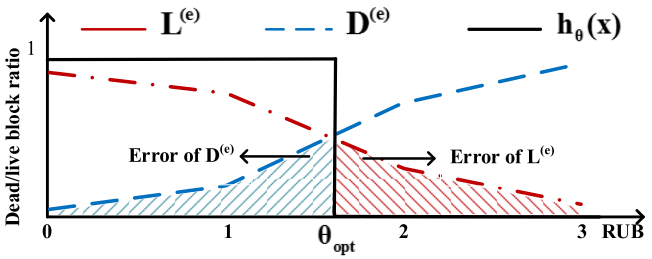


Fig. 7. Diagram of dead/live ratios and the optimal decision threshold θ_{opt}.

2) Hypotheses Function Type: In the perceptron learning, a threshold function denotes the output from the perceptron for input features, which is expressed as follows:

$$h_\theta(x) = \begin{cases} 1 & \text{if } x \le \theta \\ 0 & \text{if } x > \theta. \end{cases} \tag{13}$$

Function (13) predicts by comparing r_i with θ. The training target is to estimate the parameter θ in terms of input vectors $x = \boldsymbol{L}^{(e)}$ and $\boldsymbol{D}^{(e)}$, where $x = \boldsymbol{L}^{(e)}$ and $\boldsymbol{D}^{(e)}$ are estimated input vectors.

3) Training Method: We present a quantitative analysis on selecting threshold parameter θ to minimize miss-predictions. For a M-way associated cache with N blocks, the number of dead/live blocks N_d and N_l classified by $h_\theta(x)$ can be expressed as follows:

$$N_l = \sum_{j=0}^{[\theta]} L_j^{(e)} \frac{N}{M}, \quad N_d = \sum_{j=[\theta]+1}^{M-1} D_j^{(e)} \frac{N}{M} \tag{14}$$

where $y = [x]$ is the rounding function.

The principle of above equations is shown in Fig. 7. The red chain line and the blue dashed line represent the ratio of dead/live blocks versus the RUB number j. The threshold function $h_\theta(r_i)$ is shown as the solid line. Given a death threshold θ and a cache block with $r_i = j$, the block is classified into live block if $j < \theta$, because $h_\theta(r_i) = 1$. Else it is classified into dead blocks if $j > \theta$. The shadow area under the curve represents the total number of miss-predictions N_m, denoted as the cost function $J(\theta)$

$$J(\theta) = \left(\sum_{j=0}^{[\theta]} D_j^{(e)} + \sum_{j=[\theta]+1}^{M-1} L_j^{(e)} \right) \frac{N}{M}. \tag{15}$$

We choose θ to minimize $J(\theta)$ as follows:

$$\theta_{\text{opt}} = \underset{\theta}{\operatorname{argmin}} J(\theta). \tag{16}$$

Gradient descent or Newton's method can be adopted for optimization. However, these methods are ineffective because the derivative of (15) is hard to calculate. Intuitively, the optimal death threshold can be obtained at the intersected point in Fig. 7, where $\theta_{\text{opt}} = \{\theta | D_\theta^{(e)} = L_\theta^{(e)}\}$. The equality is true because the shadow area will increase if the death threshold deviates from the optimal point. Considering that $L_j + D_j = 1$ is always satisfied for any j, the solution of (16) is shown as

$$\theta_{\text{opt}} = \theta | D_\theta^{(e)} = L_\theta^{(e)} = 0.5 \tag{17}$$

where the optimal threshold value is achieved by searching $D_\theta^{(e)} = L_\theta^{(e)}$ for $\theta = j$.

4) Final Prediction Results: After estimating the tuple of $(\boldsymbol{L}^{(e)}, \boldsymbol{D}^{(e)})$ and deriving the optimal threshold parameter θ_{opt}, the predicted results for every cache block can be obtained. Since dirty (invalid) blocks should (not) be always backed up, we add dedicated logics to the hypothesis function. The final prediction results are expressed as follows:

$$\text{live}_i^{(\text{pre})} = \left(h_{\theta_{\text{opt}}}(r_i) \vee d_i\right) \wedge v_i \tag{18a}$$

$$\text{dead}_i^{(\text{pre})} = \neg \text{live}_i^{(\text{pre})} = \neg v_i \vee \left(\neg h_{\theta_{\text{opt}}}(r_i) \wedge \neg d_i\right) \quad \text{for } i = 1, 2, \ldots, N. \tag{18b}$$

Equation (18a) is based on the theory that dirty blocks or blocks with $r_i < \theta_{\text{opt}}$ must be backed up and invalid blocks do not need backup. Equation (18b) means that invalid blocks or clean blocks with $r_i > \theta_{\text{opt}}$ are predicted to be dead.

Algorithm 1 SBDP Algorithm

Input:

1: Cache blocks set B$\{b_1, b_2, \ldots b_N\}$, where each b_i is characterized by 3-tuple (v_i, d_i, r_i).
2: Sampling period T_s.

Output:

3: Prediction results set $\mathcal{P} = \{dead_i^{(pre)} | i \in [i, N]\}$.
4: **for** i=1 to N **do**
5: $\quad dead_i^{(pre)} \leftarrow 0$;
6: **end for**
7: Sampling an input tuple $(\boldsymbol{L}^{(s)}, \boldsymbol{D}^{(s)})$;
8: Initialize $(\boldsymbol{L}^{(e)}, \boldsymbol{D}^{(e)}) = (\boldsymbol{L}^{(s)}, \boldsymbol{D}^{(s)})$;
9: Timer t_{sam} starts timing;
10: **if** $t_{sam} \ge T_s$ **then**
11: $\quad$ Sampling an input tuple $(\boldsymbol{L}^{(s)}, \boldsymbol{D}^{(s)})$;
12: $\quad$ Update $(\boldsymbol{L}^{(e)}, \boldsymbol{D}^{(e)})$ by Equation (12);
13: $\quad \theta_{\text{opt}} = \theta | D_\theta^{(e)} = 0.5$;
14: $\quad$ **for** i=1 to N **do**
15: $\quad\quad$ Update $dead_i^{(pre)}$ and $live_i^{(pre)}$ by Equation (18);
16: $\quad$ **end for**
17: **end if**

D. SBDP Implementation

We further give the SBDP algorithm and its hardware implementation. Algorithm 1 outlines the main steps of SBDP. When a task starts, SBDP is invoked. Lines 1–5 are the initialization process, where both the prediction results set P (lines 1–3) and the estimated tuple $(\boldsymbol{L}^{(e)}, \boldsymbol{D}^{(e)})$ (lines 4 and 5) are reset. After that, the sampling timer starts the time. If the sampling timer reaches T_s, a new sampling operation is executed to update the estimated tuple (lines 8 and 9). Lines 11–13 calculate the optimal threshold θ_{opt} and make the predictions for all cache lines.

To make SBDP efficiently track the optimal threshold θ_{opt} without disturbing normal CPU process, we design a hardware

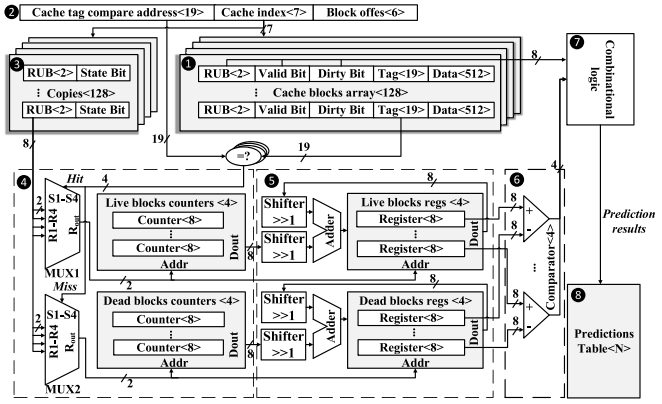


Fig. 8. Circuit of the statistics-based predictor.

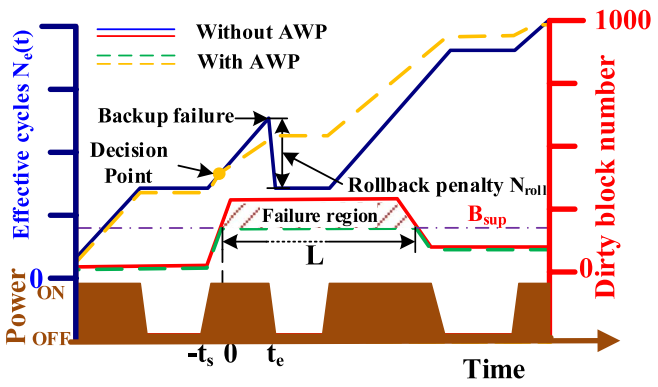


Fig. 9. Comparison between the scheme with and without AWP switching.

implementation of SBDP in Fig. 8. It takes a four-way associated 32 KB cache with 64B block size as an example (❶). The address field consists of a 19-bit tag, a 7-bit index and a 6-bit block offset (❷). ❸ and ❹ are the sampling modules to sample $(\boldsymbol{L}^{(s)}, \boldsymbol{D}^{(s)})$. If a sampling signal is detected, ❸ duplicates RUB fields from the cache. The state bits are adopted to indicate whether cache blocks are confirmed to be dead or live. First, all state bits are reset to 0. A live block is detected if a cache hit occurs and the state bit of the accessed block in ❸ is zero. The RUB of the accessed cache block is selected by a cache index in ❷ and MUX1 in ❹. The corresponding counter in live block counter arrays is increased by 1. On a cache miss, a dead block is detected if the state of the evicted block is zero. The sampling module reads the RUB information of the evicted cache block indexed by the cache index and MUX2. The corresponding counter in the dead block counter arrays is increased by 1. After recording a dead/live block, the state bit is set to one. When all state bits are set to one, a sampling operation is finished. ❺ is the recording module to implement (12). It consists of four one-bit right shifters for half-division and two 8-bit accumulators for updating. Four comparators (❻) are used to compare $\boldsymbol{L}^{(e)}$ with $\boldsymbol{D}^{(e)}$ and derive θ_{opt} according to (17). Combinational logics in ❼ implement (18b) and output the prediction results $\text{dead}^{(\text{pre})}$ for each cache block. Finally, results are written into the prediction table (❽).

The size of metadata (❸) is $(\log 2(M)+1)\times N$ for a M-way associated cache with N blocks. The SBDP consists of several live block counters and the dead block registers. If the size of each counter/register is R bit, the total size of SBDP is $4R\times M$. Hence, the total structure overhead is $N(\log 2(M)+1)+4RM$. A prototype chip with SBDP modules is designed under 65 nm CMOS/RRAM process [22]. The RTL implementation consists of four 4 KB RRAM-based nvSRAM macros and 16 SBDP modules inside. The SBDP accelerates the backup operation by nearly 30% against the sequential backup strategy under peak current constraint, while the prediction error rate is 1.24% on average.

V. Adaptive Write Policy

Write-back is attractive in embedded applications since it saves power and memory bandwidth by reducing the amount of writes to the lower level memory. However, the total penalty is associated with the backup failure and force discard penalty, when dirty blocks exceed the maximal backup threshold. Replacing write-back policy by write-through policy can avoid this event, but nontrivial write-stall penalty N_{stall} is introduced. Therefore, we propose an AWP scheme to achieve a proper balance.

A. Theory Basis

A backup failure occurs when the number of dirty blocks exceeds the upper bound. The event of a backup failure, denoted as F, is defined as

$$F : \sum_{i=1}^{N} I(\text{dead}_i = 1) > B_{\text{sup}} \tag{19}$$

where B_{sup} is the maximum supported block number. It is derived by (2) and (3)

$$B_{\text{sup}} = \frac{E_{\text{budget}} - E_{\text{lg}}}{E_b} - N_{\text{et}}. \tag{20}$$

The philosophy of AWP is to limit the quantity of dirty blocks by switching between the write-back policy and the write-through policy dynamically. Fig. 9 illustrates the difference between the behavior of a work scheme without AWP and an improved scheme with AWP. The x-axis represents the time and the left y-axis represents the accumulated effective cycles. The supported backup blocks B_{sup} is drawn in the chain line. The solid line shows the effective cycles and the number of dirty blocks with the basic write-back policy. The dashed line represents the effective cycles and the dirty blocks of AWP. The area between the dirty block number and the supported backup size B_{sup} forms a failure region. If a passive power down occurs in a failure region, a backup failure is induced. The system must roll back to the previous successful backup point.

An intuitive policy to avoid backup failure is to switch from the write-back policy to write-through when the number of dirty blocks reaches B_{sup}. This time point is marked as the decision time point in Fig. 9. However, it may lead to nontrivial performance overhead since the write-back policy is

proved to be more efficient than the write-through policy by reducing total write stalls. We denote the extra stall cycles caused by the write-through policy as $N_{\text{stall}}(t)$. Therefore, estimated total effective cycles under the write-back policy or the write-through policy should be compared before switching the writing policy.

B. Switching Strategy

We present the quantitative models and derive the optimal adaptive switching policy. When the processor reaches the decision point, the system determines whether switching to the write through policy or not. If the system using the original write-back policy, it will enter the failure region and has a probability to fail. Otherwise, if the system switches to the write-through policy, no more dirty blocks are generated but extra write stall penalties are introduced. Thus, the decision is made by comparing the overall performance profits by two policies.

Because the exponential distribution followed by the power on time T_i has the memoryless property, the distribution of the remaining waiting time for the next power down is the same. As a result, the probability that a backup failure will occur at the decision point is shown as follows:

$$P_{\text{fail}} = P(F) = P(t_e < L) = \int_0^L f(t)dt = 1 - e^{-\lambda L} \tag{21}$$

where F denotes the event of a backup failure defined by (19) and t_e is the power-off time point and L is the length of the failure region in Fig. 9. Assume that the start time of a failure region is the zero time point. The start time t_s and the end time t_e of the power supply is marked in Fig. 9. For the write-back strategy, the roll-back cycles penalty N_{roll} is defined as follows if a power-down occurs at time t:

$$N_{\text{roll}}(t_e) = \begin{cases} f_{\text{clk}}(t_e + t_s) & \text{if } t_e < L \\ 0 & \text{if } t_e > L \end{cases} \tag{22}$$

where f_{clk} is the clock frequency. $N_{\text{stall}}(L) = 0$ in the write-back policy. The expectation of $N_e(L)$ for a failure region is

$$\begin{aligned} E(N_e(L)) = {} & E(N_s(L)) - E(N_{\text{roll}}(L)) \\ & - E(N_{\text{rec}}(L)) - E(N_{\text{force}}(L)) - E(N_{\text{miss}}(L)). \end{aligned} \tag{23}$$

$N_{\text{roll}}(L) = 0$ if a write-through policy is adopted. The expectation of $N'_e(L)$ is expressed as follows:

$$\begin{aligned} E\big(N'_e(L)\big) = {} & E(N_s(L)) - E(N_{\text{rec}}(L)) \\ & - E\big(N'_{\text{force}}(L)\big) - E\big(N'_{\text{miss}}(L)\big) - E\big(N'_{\text{stall}}(L)\big). \end{aligned} \tag{24}$$

The difference between $E(N_e(L))$ and $E(N'_e(L))$ is

$$\begin{aligned} & E(N_e(L)) - E\big(N'_e(L)\big) \\ & \quad = E\big(N'_{\text{force}}(L)\big) + E\big(N'_{\text{miss}}(L)\big) + E\big(N'_{\text{stall}}(L)\big) \\ & \qquad - E(N_{\text{force}}(L)) - E(N_{\text{miss}}(L)) - E(N_{\text{roll}}(L)). \end{aligned} \tag{25}$$

N_{force} and N_{miss} are induced only if no backup failure happens, while N'_{force} and N'_{miss} always exist because the write-through policy can avoid the backup failures. If a backup failure happens, the system has to roll back to the last power-on time point ($-t_s$). Therefore, the items in (25) are calculated as follows:

$$E(N_{\text{force}}(L)) = \int_L^{\inf} N_{\text{force}}(L) f(t_e) dt_e = N_{\text{force}}(L) e^{-\lambda L} \tag{26a}$$

$$E(N_{\text{miss}}(L)) = \int_L^{\inf} N_{\text{miss}}(L) f(t_e) dt_e = N_{\text{miss}}(L) e^{-\lambda L} \tag{26b}$$

$$E\big(N'_{\text{force}}(L)\big) = N_{\text{force}}(L) \quad E\big(N'_{\text{miss}}(L)\big) = N_{\text{miss}}(L) \tag{26c}$$

$$E\big(N'_{\text{stall}}(L)\big) = N_{\text{stall}}(L) \tag{26d}$$

$$\begin{aligned} E(N_{\text{roll}}(L)) &= \int_0^L f_{\text{clk}}(t_e + t_s) f(t_e) dt_e \\ &= \frac{f_{\text{clk}}}{\lambda} - \frac{f_{\text{clk}} e^{-\lambda L}(\lambda L + 1)}{\lambda} - f_{\text{clk}} t_s \big(e^{-\lambda L} - 1\big). \end{aligned} \tag{26e}$$

Equation (25) can be rewritten as follows:

$$\begin{aligned} & E(N_e(L)) - E\big(N'_e(L)\big) \\ & \quad = \big(1 - e^{-\lambda L}\big)(N_{\text{miss}}(L) + N_{\text{force}}(L) - f_{\text{clk}} t_s) \\ & \qquad + N_{\text{stall}}(L) - \frac{f_{\text{clk}}}{\lambda} + \frac{f_{\text{clk}} e^{-\lambda L}(\lambda L + 1)}{\lambda}. \end{aligned} \tag{27}$$

If (27) <0, the system needs to be switched to the write-through policy at the decision point. Otherwise, the write-back policy is kept. Typically, $N_{\text{miss}}(L)$ and $N_{\text{force}}(L)$ are small enough to be ignored compared with other factors in (27). If the extra write stall number increases linearly with time t, $N_{\text{stall}}(L)$ can be further expressed as $N_{\text{stall}}(L) = n_s L f_{\text{clk}}$ where n_s is the average extra number of write stalls. Therefore, $E(N_e(L)) - E(N'_e(L)) < 0$ is equivalent to the following inequation:

$$n_s L - \big(1 - e^{-\lambda L}\big) t_s - \frac{1}{\lambda} + \frac{e^{-\lambda L}(\lambda L + 1)}{\lambda} < 0. \tag{28}$$

We can define TH as

$$\text{TH} = n_s L - \big(1 - e^{-\lambda L}\big) t_s - \frac{1}{\lambda} + \frac{e^{-\lambda L}(\lambda L + 1)}{\lambda}. \tag{29}$$

The write policy switching occurs only if TH < 0. The mathematical expression is in accordance with the intuitive analysis in the following aspects. First, TH < 0 is less likely to be satisfied with the increasing of n_s. It means that the system should stay in the write-back policy, if the runtime performance overhead caused by write-through dominates more. Second, TH has a negative correlation with t_s. It is because that if the last backup time point is far away from the current time point, the rollback penalty will increase and thereby the policy transition will occur more easily. Third, there is not a monotonic relationship between TH and L. It is because the probability of a power failure increases with the increasing of L. But the number of extra write stalls induced by write-through policy also increases. Therefore, both write-back and

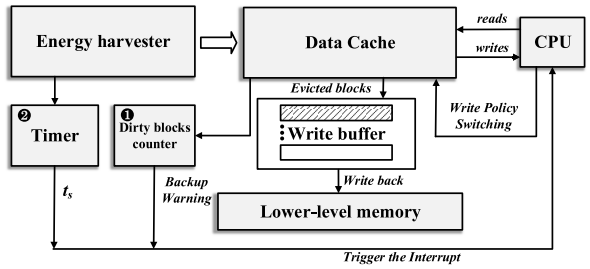


Fig. 10. Hardware implementation of the AWP.

TABLE II
SIMULATION SETUP

Component	Parameter	Value
Processor	Cell	0.13μm CMOS + NVFF [35]
	Core	1 core, 0.9V, 25MHz, pipeline CPU
Cache	Cell	RRAM based nvSRAM [14]
	Structure	4-way association, 8KB&8KB I/D, 64B block
	R/W latency	1/1 clock cycle
	Store energy	1.714nJ/2Kb
	Restore energy	1.06nJ/2Kb
Off-chip Mem	Mem type	Flash
	I&D size	512KB&2GB
	R/W latency	300 clock cycle

write-through policies have their superiority and are likely to be adopted.

C. AWP Implementation

The organization of the multilevel cache consisting of a write buffer and a policy switching module is shown in Fig. 10. To implement the AWP algorithm, the system monitors the total amount of dirty blocks by the dirty block counter (❶). A backup warning signal is triggered each time when the number of dirty blocks exceeds the threshold B_{sup}. The CPU interrupt is triggered and TH is calculated by CPU according to (29), where t_s is provided by a timer (❷). λ is obtained through some energy prediction techniques such as the EWMA algorithm [34]. L and n_s are estimated by referring to the program history. If TH < 0, the CPU sends a signal to trigger the policy switching. Therefore, only a dirty block counter and a timer are needed, which causes trivial hardware overhead.

VI. EXPERIMENTAL RESULTS

In this section, we provide comprehensive evaluations to demonstrate the efficiency of the proposed backup flow. We first give the experiment setup. Furthermore, we compare the SBDP with other state-of-the-art techniques and analyze the peak current influence. Finally, we show the tradeoff between backup capacitor and performance gain.

A. Experiment Setup

We implement the NVP with nvSRAM in gem5 [19] to model a typical computation system on a sensor board.

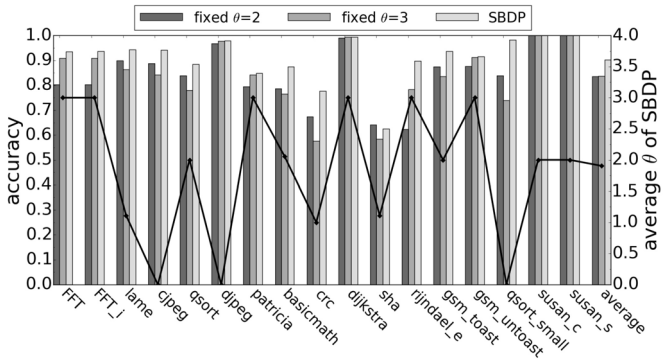


Fig. 11. Prediction accuracy of SBDP compared with fixed threshold θ predictors. The solid line shows the variation of death threshold (θ) of SBDP for different benchmarks.

The simulator is configured to model a fabricated NVP chip [35]. The parameters of nvSRAM is derived from the previous work [14] and listed in Table II. All benchmarks come from Mibench [18]. We fast forward 10M instructions at the beginning to warm up caches and execute 60M instructions of a single benchmark.

B. SBDP Evaluation

SBDP is to properly change the threshold parameter θ of the hypotheses function dynamically. Fig. 11 shows the prediction accuracy of SBDP compared with the fixed threshold policies ($\theta = 2$ and $\theta = 3$). As shown by the solid line, the threshold (θ) of SBDP varies from 0 to 3 under different benchmarks. It validates that the proposed architecture can dynamically track and update RUB. Compared with the fixed threshold policy, the proposed adaptive threshold policy outperforms or equals to the fixed threshold policy. The average accuracy of SBDP is up to 92% while those in fixed threshold policies are 83.6% ($\theta = 2$) and 83.7% ($\theta = 3$).

Table III compares prediction accuracy, hardware and performance overhead induced by the proposed strategy with several state-of-the-art dead-block prediction methods [Reftrace [30] and cache burst [33] and adaptive write-back policy (AWS)]. Some interesting results are observed. Although the prediction accuracy of SBDP is not the highest, it outperforms all other strategies with more accurate predictions by introducing the smallest 17.2% performance overhead. This is because other complicated dead block prediction techniques introduce more backup energy overhead due to large storage overhead. They cause large performance penalties from force discard and backup failures, which dominate the overall performance penalties.

Moreover, SBDP+AWP can provide better performance than SBDP only approach. With the help of AWP, the backup failure penalty decreases from 38.5% to 0% at the expense of a few extra write-stall penalties. The total performance penalty is reduced to 17.2%. Compared with SBDP+AWS strategy [36], SBDP+AWP strategy can further reduce the performance penalty by 16.1% due to the elimination of backup failures.

TABLE III
COMPARISON OF ACCURACY, STORAGE OVERHEAD, AND PERFORMANCE OVERHEAD FOR DIFFERENT DEAD BLOCK PREDICTIONS

Prediction strategies	Accuracy	Hardware overhead	Backup energy overhead	Miss-prediction penalty	Force-discard penalty	Backup-failure penalty	Recovery penalty	Extra write-stall penalty	Perf. penalty
SBDP	92%	**80B**	**0.536nJ**	0.0029%	0.045%	38.5%	0.01 %	-	38.6%
SBDP(with AWP)	92%	**80B**	**0.536nJ**	0.0042%	0.072%	0%	0.01%	17.1%	**17.2%**
SBDP(with AWS [36])	92%	**80B**	**0.536nJ**	0.0036%	0.037%	23.1%	0.01%	10.2%	33.3%
Reftrace [30]	88%	432B	2.89nJ	0.0018%	0.022%	76.9%	0.01%	-	77.0%
Cache burst [33]	96%	2022B	13.5nJ	0.0040%	0.019%	84.6%	0.01%	-	84.6%

C. Performance Comparison With Full Backup Strategy

To give an intuitive analysis of the system performance, a power pulse set with uniform distribution is adopted. We use the average power length T_a, duty cycle of the power pulses D, the average performance penalties N_p^a and average extra cycles N_x^a during one power pulse to estimate the system performance. According to (6) and (7), the number of power pulses for a task is $[\mathrm{SF}/(T_aF + N_x^a - N_p^a)]$, where S is the initial task execution time and F is the system frequency. The actual execution time is listed as

$$T = \frac{T_aSF}{D\left(T_aF + N_x^a - N_p^a\right)} \tag{30}$$

$$T \propto \frac{1}{T_aF + N_x^a - N_p^a}. \tag{31}$$

We define the performance net income $G = N_x^a - N_p^a$ to denote the gap between extra execution cycles and penalty cycles. And (31) is rewritten as $T \propto [1/(T_aF + G)]$. G is a major factor influencing the performance. The baseline is the full backup strategy. Fig. 12(a) shows the percent of performance improvement of SBDP to baseline, where the y-axis is the percent of execution time reduction. Fig. 12(b) compares G between SBDP and baseline.

According to Fig. 12(a), we notice that the performance of SBDP is always superior to that of baseline. Furthermore, the performance net income G of SBDP is always larger than baseline in Fig. 12(b). The performance improvement rate fluctuates under different capacitor sizes. The full backup point marked in Fig. 12(a), denoted as C_{full}, indicates the capacitor size, which can exactly support the backup of all cache blocks. It is interesting to observe that the performance improvement is not monotonic when the capacitor volume varies. When $C = 25$ nF increases to $C_{\text{full}} = 33.7$ nF, the improvement decreases because SBDP can discard dead blocks efficiently. When $C = 40$ nF increases to $C = 50$ nF, the performance improvement decreases. It is because the two-order derivative of T to G is less than zero, when C increases and $G(\text{partial}) - G(\text{full})$ remains constant. Counter-intuitively, we can observe that improvement increases from $C = 33.7$ nF to $C = 40$ nF. It is because $G(\text{partial}) - G(\text{full})$ increases dramatically from C_{full} to $C = 40$ nF, and there exists some extra write stall cycles at the point of C_{full}, which is removed at the points larger than C_{full}. Moreover, the backup energy of SBDP is reduced by 38.1% compared to baseline at C_{full}.

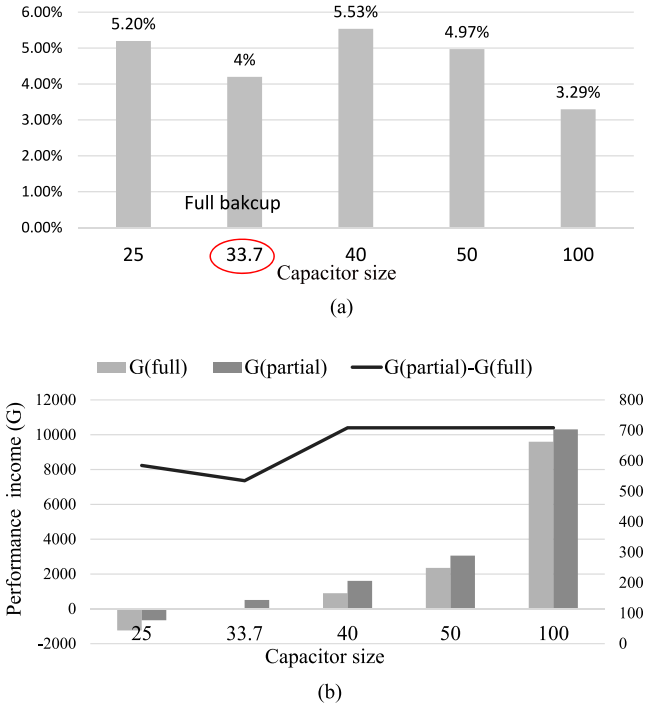


Fig. 12. Performance comparison between the proposed partial backup strategy and the full backup strategy. The performance income factor G refers to the gap the extra execution cycles and the penalty cycles. (a) Performance improvement rate to the baseline. (b) Performance income G compared to the baseline.

D. Peak Current Analysis

In this section, we provide the analysis of the peak current comparison between SBDP and the full backup strategy. If the peak current constraint is added to the backup process, distributed backup proposed in [22] should be used to mitigate the inrush current hence the recovery time is prolonged. Fig. 13 shows the backup time comparison between SBDP and the full backup method under peak current constraint. We notice that the performance improvement rate becomes larger with the decreasing of the peak current constraint, because the recovery energy and time of SBDP is shorter. Therefore, SBDP is less sensitive to the peak current constraint and performs better in peak current limited designs. The backup current can be further reduced by combining SBDP with other orthogonal backup techniques under peak current constraint, such as compression before backup, progressive charging, sequential backup, and so on.

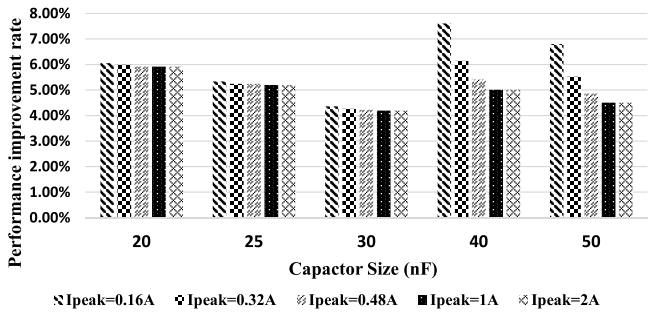


Fig. 13. Performance improvement rate with different peak current constraint.

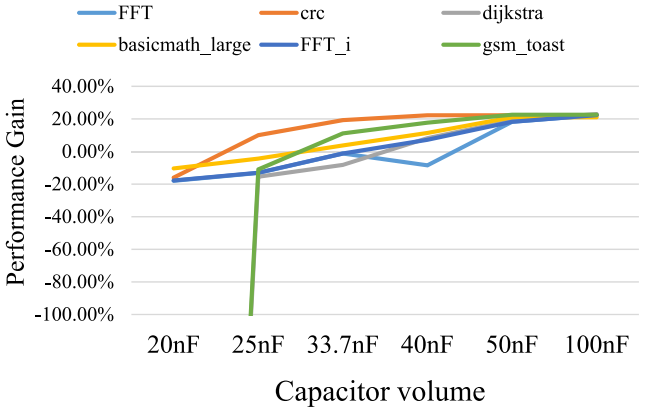


Fig. 14. Performance gain versus capacitor size.

E. Capacitor Selection for Different Benchmarks

The capacitor size C_{bulk} has great impacts on the execution time of all benchmarks. Fig. 14 shows the performance gain versus backup capacitor volumes for different benchmarks. The performance gain denotes the time reduction rate of SBDP to the execution time without power interruptions. Fig. 14 indicates that the performance gain increases as the backup capacitor volume increases. To select a proper capacitor size to finish all data backup in a task, different benchmarks have different capacitor requirements. If the performance overhead budget is set to 10%, a 20 nF capacitor is sufficient for basicmath_large but a 27 nF C_{bulk} should be used for FFT_*i*. Note that the performance gain of gsm_toast drops dramatically with the decreasing of capacitor volume, because backup failures happen if C_{bulk} is too small. On average, the capacitor size is reduced by 40%. Moreover, if a performance oriented selection strategy is used, a large off-chip capacitor should be adopted. However, a very large C_{bulk} is not needed in most cases, because the performance gain tends to be steady after a certain threshold. It is limited by the power-off interval and the capacitor charging time. The overall performance can be improved by 4.6% by choosing a proper capacitor.

VII. Conclusion

This paper proposes an energy efficient, low inrush current partial backup scheme for nvSRAM-based caches in nonvolatile sensor systems. We reduce the backup burdens through a partial backup strategy along with an AWP scheme. A statistics-based dead-block predictor is proposed to identify dead blocks accurately with low power consumption. An AWP scheme is employed to further reduce the overall performance overhead caused by energy limitation. Experimental results show that our approach can reduce the backup capacitance by 40% compared to the conventional full backup strategy and the inrush current by 54% on average in the area oriented design while improve the performance by 4.6% in the performance oriented design. In the future, the proposed backup scheme will be integrated into nvSRAM to meet the increasing demand of low power applications, such as mobile phones, PCs, and servers. The energy efficiency can be further improved with the near- or sub-threshold design techniques.

References

[1] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 443–461, 3rd Quart., 2011.

[2] H. J. Visser, A. C. F. Reniers, and J. A. C. Theeuwes, "Ambient RF energy scavenging: GSM and WLAN power density measurements," in *Proc. 38th Eur. Microw. Conf.*, Amsterdam, The Netherlands, Oct. 2008, pp. 721–724.

[3] Z. Shao, Y. Liu, Y. Chen, and T. Li, "Utilizing PCM for energy optimization in embedded systems," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Amherst, MA, USA, 2012, pp. 398–403.

[4] W.-S. Khwa *et al.*, "A resistance drift compensation scheme to reduce MLC PCM raw BER by over 100X for storage class memory applications," in *Proc. IEEE Int. Solid State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2016, pp. 134–135.

[5] Y. Chen, W.-F. Wong, H. Li, and C.-K. Koh, "Processor caches built using multi-level spin-transfer torque ram cells," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Fukuoka, Japan, 2011, pp. 73–78.

[6] Z. Sun *et al.*, "Multi retention level STT-RAM cache designs with a dynamic refresh scheme," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchit.*, Porto Alegre, Brazil, 2011, pp. 329–338.

[7] M.-F. Chang *et al.*, "17.5 A 3T1R nonvolatile TCAM using MLC ReRAM with sub-1ns search time," in *Proc. IEEE Int. Solid State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2015, pp. 1–3.

[8] J. Wang, X. Dong, and Y. Xie, "Point and discard: A hard-error-tolerant architecture for non-volatile last level caches," in *Proc. 49th Annu. Design Autom. Conf.*, San Francisco, CA, USA, 2012, pp. 253–258.

[9] S.-S. Sheu *et al.*, "A 4Mb embedded SLC resistive-ram macro with 7.2ns read-write random-access time and 160ns MLC-access capability," in *Proc. IEEE Int. Solid State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2011, pp. 200–202.

[10] M.-F. Chang *et al.*, "19.4 embedded 1Mb ReRAM in 28nm CMOS with 0.27-to-1V read using swing-sample-and-couple sense amplifier and self-boost-write-termination scheme," in *IEEE Int. Solid State Circuits Conf. Dig. Tech. Papers (ISSCC)*, San Francisco, CA, USA, 2014, pp. 332–333.

[11] F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori, "Evaluation of hybrid memory technologies using SOT-MRAM for on-chip cache hierarchy," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 3, pp. 367–380, Mar. 2015.

[12] K. Qiu, M. Zhao, Q. Li, C. Fu, and C. J. Xue, "Migration-aware loop retiming for STT-RAM-based hybrid cache in embedded systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 3, pp. 329–342, Mar. 2014.

[13] Y. Wang, H. Yu, and W. Zhang, "Nonvolatile CBRAM-crossbar-based 3-d-integrated hybrid memory for data retention," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 5, pp. 957–970, May 2014.

[14] P.-F. Chiu *et al.*, "Low store energy, low VDDMIN, 8T2R nonvolatile latch and SRAM with vertical-stacked resistive memory (memristor) devices for low power mobile applications," *IEEE J. Solid-State Circuits*, vol. 47, no. 6, pp. 1483–1496, Jun. 2012.

[15] X. Pan and R. Teodorescu, "NVSleep: Using non-volatile memory to enable fast sleep/wakeup of idle cores," in *Proc. IEEE 32nd Int. Conf. Comput. Design (ICCD)*, Seoul, South Korea, 2014, pp. 400–407.
[16] Y. Wang *et al.*, "A compression-based area-efficient recovery architecture for nonvolatile processors," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, Dresden, Germany, Mar. 2012, pp. 1519–1524.
[17] X. Sheng, Y. Wang, Y. Liu, and H. Yang, "SPAC: A segment-based parallel compression for backup acceleration in nonvolatile processors," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, Grenoble, France, Mar. 2013, pp. 865–868.
[18] M. R. Guthaus *et al.*, "Mibench: A free, commercially representative embedded benchmark suite," in *Proc. IEEE Int. Workshop Workload Characterization*, Austin, TX, USA, Dec. 2001, pp. 3–14.
[19] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.
[20] C. Wang *et al.*, "Storage-less and converter-less maximum power point tracking of photovoltaic cells for a nonvolatile microprocessor," in *Proc. ASP-DAC*, Singapore, 2014, pp. 379–384.
[21] Y. Liu *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 *Proc. IEEE Int. Solid State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2016, pp. 84–86.
[22] Y. Liu, F. Suy, Z. Wangy, and H. Yang, "Design exploration of inrush current aware controller for nonvolatile processor," in *Proc. IEEE Non Volatile Memory Syst. Appl. Symp. (NVMSA)*, Hong Kong, 2015, pp. 1–6.
[23] A. Lee *et al.*, "RRAM-based 7T1R nonvolatile SRAM with 2x reduction in store energy and 94x reduction in restore energy for frequent-off instant-on applications," in *Proc. Symp. VLSI Circuits (VLSI Circuits)*, Kyoto, Japan, 2015, pp. C76–C77.
[24] T. Ohsawa *et al.*, "A 1 Mb nonvolatile embedded memory using 4T2MTJ cell with 32 b fine-grained power gating scheme," *IEEE J. Solid-State Circuits*, vol. 48, no. 6, pp. 1511–1520, Jun. 2013.
[25] Y. Shuto, S. Yamamoto, and S. Sugahara, "Comparative study of power-gating architectures for nonvolatile SRAM cells based on spintronics technology," in *Proc. IEEE Asia–Pac. Conf. Circuits Syst. (APCCAS)*, 2014, pp. 699–702.
[26] S.-H. Chen, Y.-L. Lin, and M. C.-T. Chao, "Power-up sequence control for MTCMOS designs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 3, pp. 413–423, Mar. 2013.
[27] K. Ma *et al.*, "Architecture exploration for ambient energy harvesting nonvolatile processors," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Burlingame, CA, USA, 2015, pp. 526–537.
[28] H.-J. Tsai *et al.*, "Leveraging data lifetime for energy-aware last level non-volatile SRAM caches using redundant store elimination," in *Proc. 51st Annu. Design Autom. Conf. Design Autom. Conf.*, San Francisco, CA, USA, 2014, pp. 1–6.
[29] Y. Liu *et al.*, "Ambient energy harvesting nonvolatile processors: From circuit to system," in *Proc. 52nd Annu. Design Autom. Conf.*, San Francisco, CA, USA, 2015, p. 150.
[30] A.-C. Lai, C. Fide, and B. Falsafi, "Dead-block prediction & dead-block correlating prefetchers," in *Proc. 28th Annu. Int. Symp. Comput. Archit.*, Gothenburg, Sweden, 2001, pp. 144–154.
[31] J. Ahn, S. Yoo, and K. Choi, "DASCA: Dead write prediction assisted STT-RAM cache architecture," in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Orlando, FL, USA, 2014, pp. 25–36.
[32] M. Kharbutli and D. Solihin, "Counter-based cache replacement and bypassing algorithms," *IEEE Trans. Comput.*, vol. 57, no. 4, pp. 433–447, Apr. 2008.
[33] H. Liu, M. Ferdman, J. Huh, and D. Burger, "Cache bursts: A new approach for eliminating dead blocks and increasing cache efficiency," in *Proc. 41st Annu. IEEE/ACM Int. Symp. Microarchit.*, 2008, pp. 222–233.
[34] D. R. Cox, "Prediction by exponentially weighted moving averages and related methods," *J. Roy. Stat. Soc. B (Methodol.)*, vol. 23, no. 2, pp. 414–422, 1961.
[35] Y. Wang *et al.*, "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," in *Proc. ESSCIRC*, Bordeaux, France, 2012, pp. 149–152.
[36] H. Li, "An energy efficient backup scheme with low inrush current for nonvolatile SRAM in energy harvesting sensor nodes," in *Proc. Design Autom. Test Europe Conf. Exhibit.*, Grenoble, France, 2015, pp. 7–12.

Yongpan Liu (SM'15) received the B.S., M.S., and Ph.D. degrees from the Electronic Engineering Department, Tsinghua University, Beijing, China, in 1999, 2002, and 2007, respectively.

He is currently an Associate Professor with the Department of Electronic Engineering, Tsinghua University. He has published over 100 peer-reviewed conference and journal papers and led over six chip design projects for sensing applications, including the first nonvolatile processor (THU1010N). His research is supported by NSFC, 863, 973 Program and Industry Companies, such as Huawei, Shenzhen, China, Rohm, Kyoto, Japan, and Intel, Santa Clara, CA, USA. He holds 7 authorized Chinese patents and 1 authorized U.S. patent. His current research interests include nonvolatile computation, low power very large scale integration design, emerging circuits and systems, and design automation. These projects lead to the first nonvolatile processor THU1010N and a series of advanced versions. The line of processors has been adopted for the research of self-powered sensing platforms in 7 universities

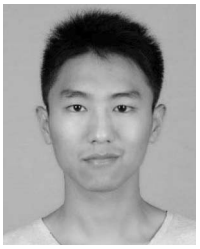
Dr. Liu was a recipient of the Micro Top Pick 2016, the Best Paper Award in HPCA 2015, two Design Contest Awards in International Symposium on Low Power Electronic Design (ISLPED) 2012 and 2013, and three Best Paper Nominations in ASPDAC 2013, 2016, and 2017. He served as the General Chair for AWSSS 2016, the Exhibition Chair for Asian Solid-State Circuit Conference (A-SSCC) 2015, the Publicity Chair for International Conference Computer Design 2015-2016 and BioCAS 2016, and the Panel Chair for AsiaHOST 2016. He also served as a Program Committee Member for leading conferences, including Design Automation Conference (DAC), Asian South Pacific Design Automation Conference, ISLPED, ICCD, real-time system symposium, and A-SSCC. He is an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and *IET Cyber-Physical Systems: Theory & Applications*. He co-founded and serves as the Vice Chair for ACM SIGDA North China Chapter and co-founded IEEE CEDA Beijing Chapter. He has given 26 invited talks in industry/academia. He is an ACM Member.

Jinshan Yue received the B.S. degree from the Electronic Engineering Department, Tsinghua University, Beijing, China, in 2016, where he is currently pursuing the Ph.D. degree.

His current research interests include nonvolatile memory, high-performance simulator, and high energy efficiency neural network accelerator design.

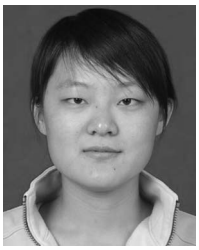
Hehe Li received the B.S. and M.S. degrees from the Electronic Engineering Department, Tsinghua University, Beijing, China, in 2013 and 2016, respectively.

He is currently an Embedded Software Engineer with TP-LINK Company Ltd., Shenzhen, China. His current research interests include nonvolatile memory, high-performance simulator design, and real-time embedded systems.



Qinghang Zhao received the B.S. degree in electrical engineering from Xidian University, Xi'an, China, in 2013. He is currently pursuing the Ph.D. degree with Tsinghua University, Beijing, China.

His current research interests include thin-film transistor circuit modeling and design, and flexible electronics technology.



Mengying Zhao received the B.E. degree from the School of Computer Science and Technology, Shandong University, Jinan, China, in 2011, and the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, Hong Kong, in 2015.

She is currently an Assistant Professor with the School of Computer Science and Technology, Shandong University. Her current research interests include computer architecture, embedded and real-time systems, and nonvolatile memory.

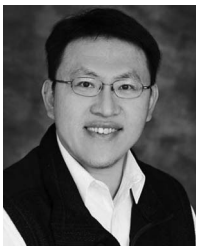
Chun Jason Xue received the B.S. degree in computer science and engineering from the University of Texas at Arlington, Arlington, TX, USA, in 1997, and the M.S. and Ph.D. degrees in computer science from the University of Texas at Dallas, Richardson, TX, USA, in 2002 and 2007, respectively.

He is currently an Associate Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include nonvolatile memories, embedded systems, and software/hardware codesign.

Guangyu Sun (M'09) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 2003 and 2006, respectively, and the Ph.D. degree in computer science from Pennsylvania State University, State College, PA, USA, in 2011.

He is currently an Assistant Professor of CECA with Peking University, Beijing. His current research interests include computer architecture, very large scale integration design, and electronic design automation. He has published over 60 journals and refereed conference papers in the above areas.

Dr. Sun is an Associate Editor of the *ACM Journal on Emerging Technologies in Computing*. He is a member of ACM and CCF.



Meng-Fan Chang (M'05–SM'14) received the M.S. degree from Pennsylvania State University, State College, PA, USA, and the Ph.D. degree from the National Chiao Tung University, Hisnchu, Taiwan.

He is currently a Full Professor with National Tsing Hua University (NTHU), Hisnchu. He was with industry for over ten years. From 1996 to 1997, he designed memory compilers in Mentor Graphics, Warren, NJ, USA. From 1997 to 2001, he designed embedded SRAMs and Flash in Design Service Division, Taiwan Semiconductor Manufacturing Company, Hsinchu. From 2001 to 2006, he was a Co-Founder and the Director of IPLib Company, Hsinchu, Taiwan, where he developed embedded SRAM and ROM compilers, Flash macros, and Flat-cell ROM products. His current research interests include circuit designs for volatile and nonvolatile memory, ultra-low-voltage systems, 3-D-memory, circuit-device interactions, and memristor logics for neuromorphic computing.

Dr. Chang was a recipient of the Academia Sinica (Taiwan) Junior Research Investigators Award in 2012, the Ta-You Wu Memorial Award of National Science Council (NSC-Taiwan) in 2011, and numerous awards from Taiwans National Chip Implementation Center (CIC), NTHU, MXIC Golden Silicon Awards, and ITRI. He is the corresponding author of numerous ISSCC and very large scale integration (VLSI) Symposia papers. He is an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS (VLSI), the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, and *IEICE Electronics*. He has been serving on technical program committees for ISSCC, IEDM, A-SSCC, ISCAS, VLSI-DAT, and numerous international conferences. He has been serving as an Associate Executive Director for Taiwan's National Program of Intelligent Electronics since 2011.

Huazhong Yang (SM'13) was born in Ziyang, China, in 1967. He received the B.S. degree in microelectronics in 1989, and the M.S. and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1993 and 1998, respectively.

He joined the Department of Electronic Engineering, Tsinghua University, Beijing, in 1993, where he has been a Full Professor since 1998. He has authored and co-authored over 300 technical papers and 70 granted patents. His current research interests include wireless sensor networks, data converters, parallel circuit simulation algorithms, nonvaltile processors, and energy-harvesting circuits.

Dr. Yang is a Specially Appointed Professor of the Cheung Kong Scholars Program.