An Energy Efficient Backup Scheme with Low Inrush Current for Nonvolatile SRAM in Energy Harvesting Sensor Nodes

Hehe Li, Yongpan Liu, Qinghang Zhao, Yizi Gu, Xiao Sheng, Guangyu Sun, Chao Zhang, Meng-Fan Chang, Rong Luo and Huazhong Yang

Tsinghua National Laboratory for Information Science and Technology, Dept. of Electronic Engineering, Tsinghua University, Haidian 100084, Beijing Center for Energy-efficient Computing and Applications, EECS School, Peking University, Haidian 100871, Beijing Dept. of Electrical Engineering, National Tsing Hua University, Hsin Chu 30013, Taiwan

Abstract—In modern energy harvesting sensor nodes, nonvolatile SRAM (nvSRAM) has been widely investigated as a promising on-chip memory architecture because of its zero standby power, resilience to power failures, and fast read/write operations. However, conventional approaches transfer all data from SRAM into NVM during the backup process. Thus, large on-chip energy storage capacitors are normally required. In addition, high peak inrush current is generated instantaneously, which has a negative impact on energy efficiency and circuit reliability.

To mitigate these problems, we propose a novel holistic backup flow, which consists of a partial backup process and a run-time pre-writeback scheme for nvSRAM based caches. A statistics based dead-block predictor is employed to achieve a fast and low power partial backup process. We also present an adaptive pre-writeback point allocation strategy to further reduce the backup load. Simulation results show that, with our proposed backup scheme, energy storage capacitance is reduced by 34% and inrush current is reduced by 54% on average compared to the conventional full backup scheme.

I. INTRODUCTION

Energy harvesting sensor nodes have been widely used in habitat monitoring, volcano monitoring and structural monitoring due to its ultra-long operating time without maintenance [1]. However, ambient power inputs suffer from the frequent failures due to unpredictable external factors. For instance, ambient 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, it is a must to ensure continuous operations on those battery-less systems with some state-keeping techniques.

In order to maintain state in intermittent power modes, emerging nonvolatile memory (NVM) has been proposed to replace volatile memories in sensor nodes, such as registers, register files and SRAMs. Typical approaches include spin transfer torque magnetic RAM [3], [4], resistive memory [5] and etc. However, the direct replacement method suffers from the drawbacks of nonvolatile memory, such as high write energy, slow write speed, asymmetric read/write operations and low endurance. Nonvolatile SRAM (nvSRAM) integrates a SRAM cell and a nonvolatile element in cell levels, forming a direct bit-to-bit connection in a 2D or vertical arrangement to achieve the fast parallel data transfer and fast power-on/off speed [6]. Therefore, it provides comparable power and performance metrics as SRAM, while keeping the nonvolatile capability when power failures happen.

However, backing up full data parallel into nvSRAM faces unnecessary large backup energy consumptions and big inrush currents, which may cause significant energy and reliability issues. As a result, prior researches have proposed solutions to mitigate above problems. Sequential backup reduces the inrush current by prolonging the backup time [7], but the backup energy is not reduced and the time is magnitudes longer than the parallel approach. Compression techniques [8], [9] help reducing data to be copied and backup energy consumption. However, compression leads to extra computing energy. What’s more, all of above methods adopt an overly pessimistic full backup scheme, which assumes that full data should be stored.

Recently, several partial backup techniques have been raised to avoid unnecessary backup operations. Circuits are proposed to eliminate redundancy in bit levels [10] of nvSRAM. The backup is executed only if there is a difference between the value in the SRAM cell and the one in the nonvolatile element. However, architecture-level redundancy is not fully considered. In fact, there are plenty of architecture-level works to predict dead blocks for cache prefetching and bypassing, such as PC-based [11], [12], counting-based [13] and time-based [14] predictions. Intuitively, these methods can be modified to identify dead blocks and eliminate unnecessary backup. However, as the motivation examples in Section II show, they introduce unacceptable hardware and energy overhead, which is not suitable for embedded applications, such as energy harvesting sensors. In addition, as these methods largely depend on the history trace, run-time computation introduces even more energy overhead. Taking all into considerations, an energy efficient dead block prediction is needed for energy...
This paper proposes a systematic backup strategy for nvSRAM based caches. It consists of two techniques: statistics based dead-block prediction (SBDP) and run-time pre-writeback. Based on the statistics of recent used bits (RUB), SBDP can significantly reduce the backup energy with trivial hardware overhead, which makes it suitable for nvSRAM backup in embedded applications. Besides, a run-time pre-writeback scheme is proposed to reduce the occurrences of rollback caused by excessive dirty blocks. They can separately reduce time and energy penalty in nvSRAM backup operations during the backup and run-time stage, respectively. The fundamental challenges lie in identifying dead blocks accurately and efficiently based on a simple hardware and designing proper pre-writeback strategy. Our specific contributions are listed as follows:

- We propose a systematic backup strategy for nvSRAM based caches, which reduces energy and performance penalties in both the backup and executing stage.
- We present an energy-efficient statistics based dead blocks prediction (SBDP) method, which figures out dead blocks with trivial hardware overhead and its accuracy is comparable to state-of-art techniques.
- We design an adaptive writeback scheme (AWS) to accomplish the pre-writeback process. The scheme has negligible performance and energy overhead but reduces the backup energy significantly.
- Experimental results show that SBDP can achieve over 20% backup energy savings and reduce over 96% time penalty compared with state-of-art dead blocks prediction techniques. The combined SBDP+AWS approach can further reduce the backup energy by 30% and time penalty by 97%.

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

II. MOTIVATION

This section first describes the typical architecture of an energy harvesting sensor. After that, the limitations of nvSRAM backup are given out. Fig. 1 shows a typical energy harvesting nonvolatile sensor node. It consists of an energy harvesting module, a nonvolatile processor (NVP), peripheral sensors, data storage and wireless transceivers. Supposing a square waveform is generated by a vibration based energy harvester, the voltage detection circuit (VDC) detects a power drop when a power failure arrives. The VDC generates a backup signal to the NVP, which starts the backup operation in nvSRAM. Since the backup operations are powered by the energy in the capacitor $C_{bp}$, its capacity should be large enough to complete the data backup.

Although conventional full backup policy can be reasonable for nonvolatile register files, it brings nontrivial chip area overheads for nvSRAM. For instance, a 32KB nvSRAM based cache, which is common in ARM-cortex A8 processors [15], requires a 107.1nF on-chip capacitor to supply the backup energy. It leads to over 4.8X larger chip area compared to the original one if a fully-integrated solution is required. Moreover, it needs hundreds of milliseconds to be fully charged under typical energy harvesting currents, which can even make the system unworkable. Last but not the least, it causes a large inrush current up to several amperes, which has a negative impact on the reliability and limits the maximum nvSRAM capacity.

To solve those challenges, a partial backup method is needed to discard useless data during the backup operation. Dead block prediction (DBP) techniques can be used to detect the useless data, but traditional approaches may not work well in the energy harvesting scenario. Table I summarizes the performance of various state-of-art DBP strategies, including the simplest LRU algorithm and more complicated but accurate DBP techniques.

<table>
<thead>
<tr>
<th>DBP</th>
<th>Accuracy</th>
<th>False-alarm Rate</th>
<th>Predictor Structures</th>
<th>Cache Metadata</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>66%</td>
<td>4%</td>
<td>8KB</td>
<td>1KB</td>
<td>9KB</td>
</tr>
<tr>
<td>Cache Burst [16]</td>
<td>90%</td>
<td>4%</td>
<td>4KB</td>
<td>0.7KB</td>
<td>4.7KB</td>
</tr>
</tbody>
</table>

Table I compares prediction accuracy, false-alarm rate$^1$ and hardware overheads in an 8KB nvSRAM, which is a typical configuration in a nonvolatile sensor node[16]. As we can see, LRU method provides relatively low accuracy and high false-alarm rate, which may cause significant backup energy and performance penalty. However, other complicated DBP methods, which are dedicated to high performance applications, introduce unacceptable hardware overheads in an embedded system. They are even comparable to the nvSRAM size. What’s more, state-of-art DBP techniques need history records in the predictor tables, which requires even more nvSRAM and increases the backup energy. Therefore, a lightweight but accurate DBP method is badly needed for energy harvesting sensors.

III. PROBLEM FORMULATION

This section gives out the models and formulates the optimization problem. After that, the two-step flow is presented.

$^1$A false-alarm error occurs when a cache block is predicted to be dead but referred later. It leads to an extra time penalty as a cache miss happens.
A. Formulation

In a full backup process, the energy consumption of a N-block nvSRAM based cache is \( E_{fb} = N E_b \), where \( E_b \) is the backup energy consumption per block. The energy consumption in a partial backup \( E_{pb} \) can be expressed as follows:

\[
E_{pb} = E_{dbp} + E_b N_b
\]

(1)

where \( E_{dbp} \) is the energy consumption of the dead block prediction and \( N_b \) is the number of blocks to be backuped.

There are three kinds of sources to cause performance penalties in a partial backup process. 1) Backup failure penalties occur when the capacitor energy can not supply the nvSRAM to backup all dirty blocks. Assuming it happens at a probability of \( P_{fail} \), the system should roll back to its previous backup point due to the data inconsistency. We denote the rollback penalty as \( T_{rollback} \). 2) Force-discard penalties happen when all dirty blocks can be backed up, but the live blocks can not. Assuming \( N_{force} \) is the number of discarded live blocks, it will lead to a cache miss \( T_{miss} \) when the discarded block is accessed. 3) Mis-prediction penalties are caused by wrong predictions from DBP. A penalty will happen only when a live block is predicted to be dead. Therefore, a scale factor \( \alpha \) (typically 0.2-0.5) is used to represent such case among all mis-predictions. In general, the total time penalty is expressed as follows:

\[
T_{penalty} = P_{fail} T_{rollback} + (1 - P_{fail}) N_{force} T_{miss} \\
+ (1 - P_{fail}) \alpha (1 - Accuracy) T_{miss}
\]

(2)

This paper tries to to minimize \( T_{penalty} \) under the constraint of \( E_{budget} \), which can be shown as follows:

\[
obj : \min T_{penalty} \quad s.t. \quad E_{pb} < E_{budget}
\]

(3)

B. Proposed partial backup flow

We propose a systematic backup flow for the optimization problem in Fig. 2, which consists of a SDBP technique on the right side and an AWS on the left side. The former aims to reduce both mis-prediction and force-discard penalties. The latter reduces backup failure penalties and relaxes force-discard penalties.

IV. STATISTICS BASED DEAD-BLOCK PREDICTOR DESIGN

This section presents the lightweight but accurate SBDP for the partial backup process. We first discuss the observations and figure out the theory basis. After that, a circuit is implemented to support the SDBP.

A. Theory basis

The theory basis of the SBDP lies in the correlations between the distribution of recently used bits (RUB) and the classification of dead/live blocks. RUB is used in LRU cache replacement algorithm. It represents the order of access time for cache blocks in the same cache set. Fig. 3 shows the block ratio under different RUBs when four benchmarks from Mibench [17] are running on a four-way associated cache. Fig. 3 shows 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. Specifically, the ratio of dead/live blocks increases/decreases when the RUB becomes larger. This observation enable us to predict block type (dead/live) based on the RUB information. As we can see, different benchmarks show various distributions. Therefore, a dynamic statistical approach should be used.

B. Quantitative analysis

We present a quantitative analysis on the selection of RUB threshold and miss-predictions. First of all, we give the definitions of terms in Table. II. According to the definitions of terms, the number of predicted dead/live blocks can be express as follows:

\[
N_d = \sum_{j=TH}^{M-1} D_j \frac{N}{M}
\]

(4)

\[
N_l = \sum_{j=0}^{TH-1} L_j \frac{N}{M}
\]

(5)

The overall miss-predicted number \( N_m \) is as follows:

\[
N_m = \sum_{j=0}^{TH-1} D_j + \sum_{j=TH}^{M-1} L_j \frac{N}{M}
\]

(6)
It can be minimized by choosing an optimal $TH$. That is

$$TH_{opt} = \arg\min_{TH} N_m$$

(7)

Fig. 4 presents the distribution of dead/live blocks. Given a death threshold $TH$, the area under the curve represents the number of miss-predictions $N_m$. Intuitively, the optimal death threshold can be obtained when $TH_{opt} = \{TH|D_{TH} = L_{TH}\}$. If there are multiple points satisfying the condition, we choose the minimum integer $TH$ as the death threshold. Note that $D_j + L_j = 1$, $j = 1, 2, ..., M$, we have

$$TH_{opt} = TH|D_{TH} = 0.5$$

(8)

Obviously, the death threshold $TH$ needs dynamic adjustments according to different distributions under various benchmarks.

C. Circuit implementation

In order to make the SBDP to efficiently track the optimal death threshold $TH_{opt}$, we implement a circuit in Fig. 5, which shows a SBDP circuit for a 4-way associated cache with $N$ blocks. In this instance, 2-bit RUB fields are used for the 4-way associated cache. Two counter array are used to monitor the number of dead and total sample blocks. After restoration, the state bits for all cache blocks are set to zero. If a dead block with RUB=i is detected and the state bit is zero, both the corresponding dead block counter i and the sampled block counter i are increased by 1. Otherwise, only the sampled block counter is increased by 1. The state bit is set to 1 after the corresponding block’s state (dead/live) is confirmed. After some times, assuming the dead-block counter=N and the sampled block counter=M for RUB=i, we have the optimal death threshold for RUB when $N/M = 0.5$. The four comparators’ outputs represent the death threshold($TH$). In order to avoid complex dividers in hardware, we adopt right shifters and comparators to realize the threshold choosing. Since dirty(invalid) blocks should(not) be always backed up, we use a dedicated logic as follows:

$$\text{Dead bit} = VB||\{(\text{Result} & DB)\}$$

(9)

where VB represents to the valid bit, DB is the dirty bit. Final results of dead block prediction are written into the $N \times 1$ bits dead block table, whose entries represent the prediction results for all blocks.

V. ADAPTIVE PRE-WRITEBACK SCHEME

As Equation 2 shown, the total time penalty is also associated the backup failure and the force discard. We propose a pre-writeback scheme to mitigate those penalties.

A. Theory basis

Fig 6 shows an example of a backup failure and a pre-writeback. If the total number of dirty blocks exceeds the maximum backup size, a backup failure occurs. In the run-time pre-writeback scheme, a backup warning threshold is set to limit the maximum number of dirty blocks. When its number exceeds the threshold, a pre-writeback is triggered. A number of dirty blocks with large RUBs are written back to the nonvolatile parts. The overall number of dirty blocks is reduced and the force-discard penalties are also relieved because the number of live blocks becomes smaller. However, pre-writeback may lead to nontrivial energy and performance overhead if they are executed frequently. We should adjust the number of pre-writeback blocks $N_b$ dynamically.

Assuming the probability of backup failure is $P_0$, the overall probability of backup failure is as follows, when there are $N_i$ times of power intervals:

$$P_F = 1 - (1 - P_0)^{N_i}$$

(10)

If $N_b$ dirty blocks are written back, the probability $P'_0$ becomes:

$$P'_0 = \frac{D - N_b}{D} P_0$$

(11)

where $D$ is the number of dirty blocks above the maximum data size in Fig 6. The extra writeback time is $T_{w} = \gamma N_b T_w$, where $0 < \gamma < 1$ is a scale factor. $T_w$ is the writeback time of a cache block. The overall time penalty is expressed as follows.

$$T_{pen} = (1 - (1 - \frac{(D - N_b)P_0}{D})^{N_i}T_{fb} + \gamma N_b T_w$$

(12)
Equation 12 includes two kinds of penalties: backup failure and writeback penalties. If \( N_i \) increases, the backup failure penalty decreases while writeback increases. \( T_{rb} \) denotes the rollback time penalty, which is the accumulated time from the last successful backup point. By taking derivation of \( T_{pen} \) to \( N_b \), we have

\[
N_b = D - \frac{D}{P_0} \left[ 1 - \sqrt[N_b]{\frac{\gamma T_{wb}}{N_b P_0 T_{rb}}} \right] \tag{13}
\]

Therefore, \( N_b \) is positively associated with \( N_i \). If the number of dirty blocks exceeds the maximum backup size and \( N_b > 0 \), a writeback will happen.

B. Adaptive pre-writeback algorithm

**Algorithm 1** Adaptive pre-writeback algorithm

**Input:** Task set, task condition, \( P_0 \), \( T_{wb} \), \( \gamma \)

**Output:** Pre-writeback decision

1. Initial average power on-time \( T_{on} \leftarrow 0 \), average task processing time \( \bar{T} \leftarrow 0 \), \( N_i \leftarrow 0 \)
2. If a task starts then
3. Counting D and \( T_{rb} \) on-line
4. If the maximum backup size is reached then
5. Calculating optimal \( N_b \) by equation 13.
6. If \( N_b > 0 \) then
7. Executing the pre-writeback process
8. end if
9. end if
10. end if
11. If a task is finished then
12. Getting the task processing time \( T_{proc} \rightarrow last \)
13. \( T = \alpha T_{proc} \rightarrow last + (1 - \alpha)T_{on} \), where \( \alpha \) is a weight factor
14. end if
15. If a passive power down occurs then
16. Getting the last power-on time \( T_{on} \rightarrow last \)
17. \( T_{on} = \beta T_{on} \rightarrow last + (1 - \beta)T_{on} \), where \( \beta \) is a weight factor
18. \( N_i = T/T_{on} \)
19. Entering into the partial backup state
20. end if

Algorithm 1 is an adaptive pre-writeback algorithm. It is designed to on-line calculate \( N_b \) and make the pre-writeback decisions. Line 2 to line 10 represents the pre-writeback scheme during task processing. When the maximum backup size threshold is reached, a pre-writeback process is triggered if \( N_b > 0 \). The predicted power-on time and average task processing time is obtained as a weighted sum of the last power-on time, processing time length and the last prediction as shown in line 11 to 14 and line 15 to 20 in algorithm 1. In practice, \( \alpha \) and \( \beta \) are selected to be 0.5. This policy is able to track the variation of processing time and average power-on time. They are used to calculated the power-off times \( N_i \), as shown in line 18 of algorithm 1.

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-art techniques. Finally, we show the tradeoff between the backup capacitor and the performance.

### A. Experiment setup

We implement the NVP with nvSRAMs in gem5 [18]. It is configured to model a fabricated nonvolatile processor [16]. We replaced the 8KB SRAM with nvSRAM. All important parameters are listed in Table III.

<table>
<thead>
<tr>
<th>Component</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor (nvMCU)</td>
<td>Single core, 25MHz with 2-width issue</td>
</tr>
<tr>
<td>On-chip cache (nvSRAM)</td>
<td>I&amp;D size: 8kB&amp;8kB (4-way association)</td>
</tr>
<tr>
<td></td>
<td>Block size: 64B</td>
</tr>
<tr>
<td></td>
<td>R/W latency: 1/1 clock cycle</td>
</tr>
<tr>
<td></td>
<td>Store energy: 1.714nj/2kb</td>
</tr>
<tr>
<td></td>
<td>Restore energy: 1.06nj/2kb</td>
</tr>
<tr>
<td>Off-chip memory (Flash)</td>
<td>I&amp;D size: 512kB&amp;512kB</td>
</tr>
<tr>
<td></td>
<td>R/W latency: 300 clock cycle</td>
</tr>
</tbody>
</table>

All benchmarks come from Mibench [17]. For each benchmark, we select ten random backup points and use their average values for evaluations. We fast forward 10M instructions at the beginning to warm up caches and execute 30M instructions of a single benchmark.

### B. Statistics based dead-block prediction evaluation

Fig. 7 shows the prediction accuracy of SBDP compared with fixed death threshold policies (\( TH = 2 \) and \( TH = 3 \)). As shown by the solid line, the death threshold (\( TH \)) of the SBDP varies from 0 to 3 under different benchmarks. It validates that the proposed architecture can dynamically track and update RUB information. Compared with the fixed threshold policy, our proposed adaptive threshold policy always outperforms or equals to the fixed threshold policy. The average accuracy of SBDP is 91% while those in fixed threshold policies are 67% (\( TH = 2 \)) and 82% (\( TH = 3 \)).

![Fig. 7. Prediction accuracy of SBDP compared with fixed TH predictors. The solid line shows the variation of death threshold (TH) of SBDP for different benchmarks.](image-url)

Table IV compares the SBDP with several state-of-art dead-block prediction methods. Our approach performs comparative prediction accuracy but achieves the best performance. It shows some interesting counterintuitive results. The proposed scheme introduced only 0.25% performance overhead, which outperforms all other strategies with more accurate predictions. This is because those high complicated DBP techniques suffer from very high backup energy, which leads to large performance penalties from force discard and backup failures. They compromise the energy savings from the reduction of the miss-predictions. Moreover, the SBDP consumes much less prediction energy because it only reads 8x32 bits counters along...
with a 128-bit dead block table. Other prediction strategies need to read a large prediction table and greatly increase the backup overhead. Furthermore, our approach introduces less metadata on caches, while other predictors need to add extra fields such as PC [11] into cache structures. Therefore, the force-discard penalty and backup-failure penalty is reduced. Finally, the SDBP+AWS can provide better performance and less force discard penalties than the pure SDBP approach. It validates the effectiveness of the AWS approach.

C. Capacitor-performance co-optimization

Experimental results show that our approach can reduce the backup capacitance by 34% and the inrush current by 54%. The proposed backup strategy can be further used in other nonvolatile architectures such as mobile phones, PCs and servers. In the future, more works will be done to implement the proposed backup scheme. NVSRAM along with the backup scheme will be used more widely to meet the increasing demand of low power devices.

TABLE IV COMPARISON OF ACCURACY, STORAGE OVERHEAD AND PERFORMANCE OVERHEAD FOR DIFFERENT DBPS

<table>
<thead>
<tr>
<th>Prediction strategies</th>
<th>Accuracy</th>
<th>Extra storage</th>
<th>Cache metadata</th>
<th>Total hardware overhead</th>
<th>Backup energy overhead</th>
<th>Miss-prediction penalty</th>
<th>Force-discard penalty</th>
<th>Backup-failure penalty</th>
<th>Total performance penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDBP</td>
<td>91%</td>
<td>48B</td>
<td>16B</td>
<td>64B</td>
<td>0.429%</td>
<td>0.15%</td>
<td>0.19%</td>
<td>0</td>
<td>0.25%</td>
</tr>
<tr>
<td>SDBP (with pre-writeback)</td>
<td>91%</td>
<td>48B</td>
<td>16B</td>
<td>64B</td>
<td>0.429%</td>
<td>0.15%</td>
<td>0.10%</td>
<td>0</td>
<td>0.25%</td>
</tr>
<tr>
<td>Reftrace [11]</td>
<td>88%</td>
<td>8KB</td>
<td>1KB</td>
<td>9KB</td>
<td>630B</td>
<td>0.23%</td>
<td>0</td>
<td>0</td>
<td>0.50%</td>
</tr>
<tr>
<td>Cache burst [14]</td>
<td>96%</td>
<td>4KB</td>
<td>0.7KB</td>
<td>4.7KB</td>
<td>573B</td>
<td>0.09%</td>
<td>0.22%</td>
<td>0</td>
<td>0.23%</td>
</tr>
</tbody>
</table>

Fig. 8. Performance overhead vs. backup capacitor volume for the benchmark suite. The rollback time is set half of the task processing time, thus the maximum performance loss is 50% when a power failure occurs.

Fig. 8 shows the performance overhead vs. backup capacitor volume for different benchmarks. It indicates that the performance overhead decreases as the backup capacitor volume increases. The minimum capacitor volume for a full-backup scheme is 35nF. However, our analysis shows that a 23nF backup capacitor, marked as the cost-efficient point, can finish the backup process with only 0.03% performance overhead. In such a case, the total on-chip capacitance can be reduced by 34%. Moreover, the inrush current is decreased by 54% because the total backup load is reduced by discarding dead blocks. It implies that the designers can explore different design space to trade off performance overhead and capacitor volume, which may be a critical factor for tiny sensor nodes.

REFERENCES