2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)

Achieving Stagnation-Free Intermittent Computation with Boundary-Free Adaptive Execution

Jongouk Choi
Virginia Tech
Blacksburg, VA, USA
jonchoi@vt.edu

Hyunwoo Joe
ETRI
Daejeon, Korea
hwjoe@etri.re.kr

Yongjoo Kim
ETRI
Daejeon, Korea
y.kim@etri.re.kr

Changhee Jung
Virginia Tech
Blacksburg, VA, USA
chjung@vt.edu

***Abstract*— This paper presents ELASTIN, a stagnation-free intermittent computing system for energy-harvesting devices that ensures forward progress in the presence of frequent power outages without partitioning program into recoverable regions or tasks. ELASTIN leverages both timer-based checkpointing of volatile registers and copy-on-write mappings of nonvolatile memory pages to restore them in the wake of power failure. During each checkpoint interval, ELASTIN tracks memory writes on a per-page basis and backs up the original page using custom software-controlled memory protection without MMU or TLB. When a new interval starts at each timer expiration, ELASTIN clears the write permission of all the pages written during the previous interval and checkpoints all registers including a program counter as a recovery point. In particular, ELASTIN dynamically reconfigures both the checkpoint interval and the page size to achieve stagnation-free intermittent computation and maximize forward progress across power outages. The experiments on TI's MSP430 board with energy harvesting traces show that ELASTIN outperforms the state-of-the-art scheme by 3.5X on average (up to orders of magnitude speedup) and guarantees forward progress.**

I. INTRODUCTION

Adoption of energy harvesting technologies in Internet of Things (IoT) has led to the advent of batteryless low-power embedded systems [1]–[5]. By leveraging ambient energy sources such as solar, thermal, wireless, vibration, and so on [6]–[16], energy harvesting devices are not only self-sustaining and maintenance-free but also eco-friendly, and they continue to be used in many areas: sensor, wearable, storage, and implantable medical devices [12], [13], [17]–[21].

However, due to the unreliable power source, energy-harvesting systems suffer from unpredictable and frequent power failure. They use a small capacitor as an energy buffer and intermittently compute only when enough energy is secured in the capacitor; when it is depleted, the systems die. This is so-called *intermittent computation* [22]. With the intermittent nature in mind, the researchers equip the energy harvesting system with nonvolatile memory (NVM) and some form of crash consistency to checkpoint necessary data and restore them across power outages.

The state-of-the-art intermittent computing schemes partition program into a series of recoverable regions (tasks) so that their re-executions always result in the same and correct output [23]–[28]. Such a recoverability is achieved by either compiler-directed idempotent region formation [23], [29]–[34] or user-based manual task partitioning [24]–[28]. In the wake of power failure, the prior schemes restart from the beginning of the interrupted region (task) after restoring the checkpoints saved at the region/task (task) boundary for correct recovery.

However, the region (task) based schemes [24], [26]–[28], [35], [36] face several critical issues. First, the schemes end up wasting the hard-won energy due to the lack of flexibility in the checkpoint interval; they make a checkpoint, that entails multiple energy-consuming NVM writes, at every pre-defined region (task) boundary, which would be unnecessary under stable energy-harvesting condition. The crux of the problem is that due to the compile-time fixed regions (tasks), checkpoint interval cannot be adapted to the underlying energy harvesting quality and the power outage behavior.

Unfortunately, the inability to adapt checkpoint interval can cause a more serious issue, i.e., making the system stagnate while consuming the hard-won energy; that is why the prior schemes [26], [28], [37]–[40] cannot ensure forward progress. If power outages repeatedly occur within a certain region (task) before it ends, the schemes continually attempt to re-execute the same interrupted region (task). This work refers to such a livelock-like situation as **stagnation**. Due to the small capacitance of an energy buffer, stagnation often occurs during the execution of long regions (tasks). Without solving the stagnation problem, all other efforts to make energy-harvesting systems reality would eventually fail, calling for a practical solution.

Last but not least, all prior works cannot handle capacitor malfunction issues such as excessive leakage and crack, that occur in reality and reduce the capacitance, thereby leading to incorrect recovery or even worse stagnation [41], [42]. According to Cronin *et al.*'s recent work [43]–[45], even fresh capacitors can be worn out due to physical access attacks. For example, attackers can damage the capacitor by injecting malicious voltage fluctuation into the target board. This urges the existing schemes to be robust against such security attacks for stagnation-free intermittent computation with correct recovery.

To address above issues, this paper presents ELASTIN, a stagnation-free intermittent computing system for energy-harvesting devices that ensures forward progress in the presence of frequent power outages. Unlike prior works, ELASTIN does not partition a program into recoverable regions or tasks;

978-1-7281-0678-6/19/$31.00 ©2019 IEEE
DOI 10.1109/RTAS.2019.00035

such a boundary-free nature allows ELASTIN to realize full potential of checkpoint adaptation. ELASTIN leverages both timer-based checkpointing of volatile registers and copy-on-write mappings of nonvolatile memory pages to restore them in the wake of power failure. During each checkpoint interval, ELASTIN tracks memory writes on a per-page basis and backs up the original page—i.e., the copy-on-write granularity, not a virtual memory page—using software-controlled memory protection without MMU or TLB[1]. When a new interval starts at each timer expiration, ELASTIN clears the write permission of all the pages written in the previous interval and checkpoints all registers including a program counter as a recovery point. ELASTIN reconfigures the checkpoint interval and the page size based not only on the underlying energy harvesting quality but also on the observed forward progress. Consequently, ELASTIN achieves stagnation-free intermittent computation, ensuring forward progress across power outages.

Finally, ELASTIN can survive the capacitor malfunction. ELASTIN's boundary-free adaptive execution makes it possible to adapt the checkpoint interval and the page size to even the cracked capacitor or one under attacks. For example, ELASTIN can ensure forward progress even for 50% of original capacitance with 2x faster leakage draining. The takeaway is that ELASTIN can improve both the capacitor security and the lifetime of energy-harvesting systems while maximizing forward progress even under exceptional circumstances.

The contributions of this paper are as following:

- ELASTIN strongly guarantees forward execution progress. Experimental results show that ELASTIN is able to complete all benchmark applications, whereas the state-of-the-art work cannot due to stagnation.
- ELASTIN's boundary-free checkpointing requires neither user intervention nor program partitioning for region (task) formation while its 2-dimensional adaptation of timer interval and page size can maximize the forward progress; ELASTIN achieves 3.5X average speedup over the state-of-the-art region based scheme.
- ELASTIN can handle the capacitor malfunction issues, thereby achieving forward progress even when the capacitor is under security attacks or simply worn out. This can eventually lengthen the life-time of the capacitor and the energy-harvesting system.

II. BACKGROUND AND MOTIVATION

A. System Model

Since power failure is the norm in energy-harvesting systems, they should have byte-addressable nonvolatile memory (NVM) for the efficient backup/recovery across the failures. TI's MSP430FR series of microcontrollers (MCU) have already integrated FRAM, though SRAM is still used due to the high write energy/latency of the current FRAM technology [46]. The MCUs are expected to have NVM soon as main memory thanks to new technologies such as STT-MRAM. Currently, ELASTIN targets MSP430 MCUs, 3-stage pipeline in-order core without cache, MMU, or TLB, where FRAM is used as main memory; SRAM is not used in our evaluation. Thus, only data in a processor, i.e., registers, are transient and will be lost on power failure; they need to be checkpointed for recovery.

[1]ELASTIN can be regarded as library OS that only offers memory protection and timer interrupt. In general, energy-harvesting devices do not run OS.

B. Curse of Stagnation

Suppose a program region/task whose execution time is greater than the power failure period, i.e., the time between the failures. If they periodically occur with the same frequency, the program ends up rolling back to the beginning of the same region again and again. That is because the failures keep occurring before the end of the region is reached, in which case the program just wastes harvested energy in vain making no forward progress. Section III shows how ELASTIN guarantees forward progress to overcome the curse of stagnation.

C. Lack of Checkpoint Adaptation

If the amount of harvested energy is sufficient, the energy-harvesting system does not have to frequently checkpoint to back up necessary program status due to low likelihood of power failure. On the other hand, if the harvesting energy source is weak or unstable, the system would need to checkpoint more frequently than usual. Unfortunately, all prior software schemes partition program to regions or restructuring it as tasks to form recoverable regions/tasks without considering the level of harvested energy. Since the schemes checkpoint program status at each region/task boundary fixed at compile time, they cannot adapt to the varying quality of harvested energy at run time. Even if power failure rarely occurs, the schemes can waste hard-won energy by performing an unnecessary checkpoint at every single boundary during the execution of consecutive regions (tasks). Even worse, the schemes can suffer from stagnation during the execution of a long region (task) when power outages occur frequently.

D. Capacitor (Energy Buffer) Vulnerability

Existing task based schemes [27], [47]–[49] take into account capacitor's behavior to improve forward progress. However, they may end up with stagnation due to their assumption that the capacitor can maintain its original quality (characteristic), which is not true in reality for several reasons.

First, the capacitor energy can be drained a lot faster than usual when the temperature goes up. This can be understood by Arrhenius law [41], [50], [51] that specifies the high impact of temperature on the leakage current, i.e., exponential leakage increase with rising temperature.

Second, even if the temperature is maintained, the capacitor can still malfunction. For example, due to mechanical or external pressure, the packaging material can be worn out, and the capacitor will be cracked eventually [52]. If cracked, the capacitor leaks the buffered energy more dramatically or charges only partial amount of the original capacitance.

Third, the capacitor quality can be deteriorated by security attacks that can physically access it and inject malicious

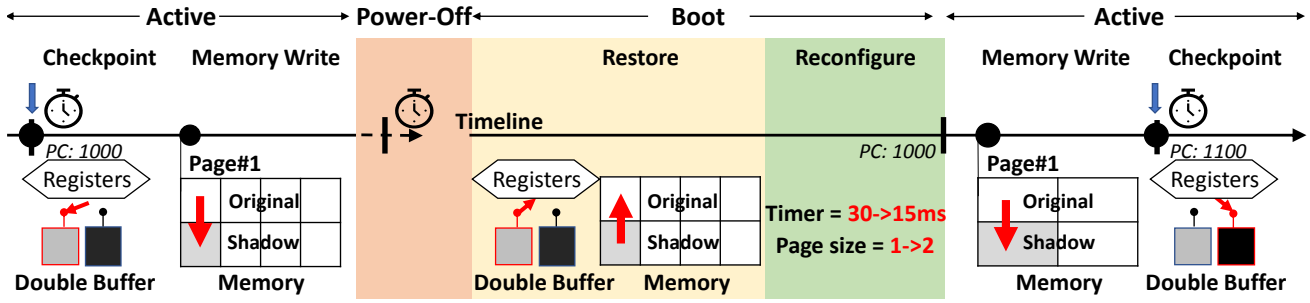


Fig. 1: Overall workflow: checkpoint interval can be adjusted when it ends at timer expiration and in the wake of power outage at boot time

voltage fluctuation to the system. As shown in the recent work [43], the attackers can inject square wave voltage fluctuation that generates extremely frequent checkpoints in the system, thus making capacitor malfunction occur much earlier. Section III-D details how ELASTIN addresses these issues.

III. Design Overview

To realize full potential of adaptive execution according to energy-harvesting condition, we design ELASTIN's backup and recovery mechanisms in a boundary-free way without inserting region or task boundaries to program. For this purpose, ELASTIN leverages both timer-based checkpointing of volatile registers (Section III-A) and copy-on-write mappings of nonvolatile memory (NVM) pages (Section III-B) to restore them in the wake of power failure. Figure 1 describes the overall workflow of ELASTIN.

A. Watchdog Timer Based Checkpointing of Volatile Registers

ELASTIN leverages a watchdog timer, that can be adjusted at both its expiration and boot time (Section III-C1), to form flexible checkpoint interval. At each timer expiration where the current checkpoint interval finishes and the new one is about to start, ELASTIN checkpoints all registers including the program counter (PC) to a reserved area in NVM. In case of power outage during the register checkpoint, ELASTIN leverages *double buffering* to leave at least one of the two buffers intact [23], [53]; see Figure 1.

Note that ELASTIN saves the registers for the new interval in case it is interrupted due to power failure. As described in Figure 1, when power comes back, ELASTIN uses the PC as a recovery point to restart the interrupted interval after restoring all the other registers; they serve as inputs to the interval to be restarted. As will be shown in Section III-B, in addition to a volatile register file, ELASTIN needs to make a copy of NVM pages, which is invalidated at both timer expiration and boot time, for correct recovery. Thus, the recovery process includes the page restoration as well.

B. Page Protection Based Backup of Nonvolatile Memory

The timer-based checkpointing alone can lead to a memory inconsistency problem. Consider an example shown in Figure 2. Here, an energy-harvesting system checkpoints between write#2 and the following read instruction and encounters a

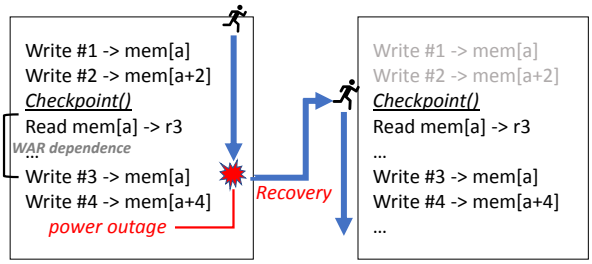


Fig. 2: Memory inconsistent recovery due to anti-dependence

power failure right after write#3. In this case, the write#3 and the Read instruction access to the same memory, mem[a]. Thus, these two instructions are anti-dependent, i.e., they form a WAR (write-after-read) dependence. In the wake of the power failure, the system starts from the most recently checkpointed point, thus subsequently reading mem[a]. However, it ends up reading not the original value but the one updated by write#3, thereby leading to incorrect recovery.

To address the memory inconsistency, during each checkpoint interval, ELASTIN tracks memory writes on a per-page basis and backs up the original page; in the wake of power failure, ELASTIN first reverts all the writes (including anti-dependent ones) performed in the interrupted interval using the backup page and then jumps back to the recovery PC (Section III-A) where the interval started. That way ELASTIN can restart the interrupted interval with original memory status as if it were being started for the first time.

To achieve this, ELASTIN leverages a conventional page protection mechanism of operating systems which tracks writes to non-writable page as a page fault and backs up the page with a copy-on-write mechanism [54]. In general, energy-harvesting systems do not run OS due to the scarce power supply, and thus we implemented custom page protection library; in a sense, ELASTIN can be regarded as a library OS that only supports page protection[2] and timer interrupt handling.

Interaction with Timer Based Checkpointing: When the watchdog timer is expired (i.e., the current checkpoint interval

[2]It is only for page backup and does not support virtual memory. The MCU of energy harvesting systems lacks MMU/TLB due to power constraint.

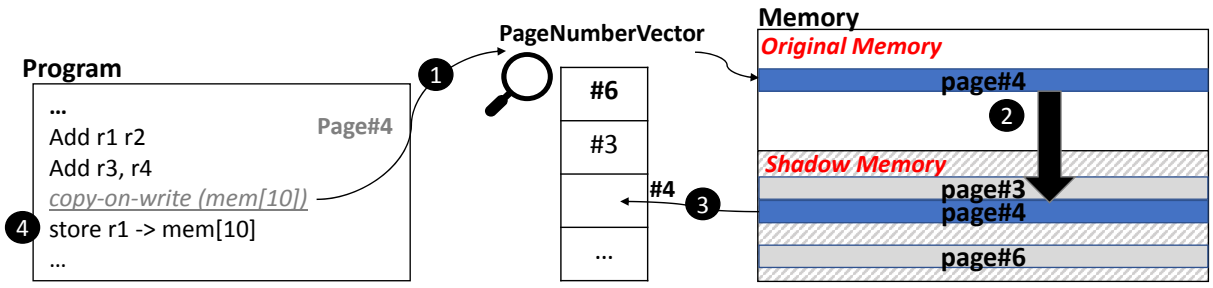


Fig. 3: copy-on-write backup: ❶ page number vector (PNV) lookup, ❷ copy the page to shadow, ❸ PNV insertion, ❹ memory write

has just been finished), ELASTIN clears the write permission of all the pages written in the interval. In other words, when the upcoming new interval starts, no page has a write permission. This gets the new interval ready to track its own writes and trigger copy-on-write for backing up the corresponding pages. In this way, ELASTIN can ensure that each interval starts with clean memory status.

Custom Software-Controlled Page Protection: To track memory writes and trigger their copy-on-write if needed, ELASTIN instruments store instructions at compile time while maintaining a page number vector (PNV) to record which page has a write permission at run time. For each store, ELASTIN first checks if the target page has a write permission by consulting PNV (❶ in Figure 3). If not, i.e., the page number is not found in PNV, ELASTIN creates a copy of the page (i.e., copy-on-write) in shadow memory or radix tree based data structure (❷); Section IV-B discusses the overhead of these alternatives. Then, to grant the page a write permission, ELASTIN inserts the page number (#4 in Figure 3) to PNV as a mark for the permission (❸). In this way, ELASTIN can preserve the copy of the page until the end of the current checkpoint interval so that the copy can be used to recover from possible outages during the interval. Finally, ELASTIN performs the write (❹).

On the other hand, if the page being stored has a write permission [3], i.e., the page number is found in PNV, ELASTIN skips both the page copy and the PNV insertion. In summary, for a store to writable pages, ELASTIN takes only two steps (❶→❹) while a store to non-writable pages goes through all four steps (❶→❷→❸→❹). Note that any power outage between these steps does not cause a memory inconsistency problem during the recovery as long as their order is enforced.

In the wake of power failure, ELASTIN reverts all the written pages (i.e., those populated in PNV) by using their original copy in shadow memory along with restoring all registers as shown in Figure 1. Obviously, this software-controlled page protection mechanism consumes the harvested energy for both page backup and restoration; Section III-C2 describes how ELASTIN adjusts the page size to minimize the copy-on-write overhead, and Section III-E shows how ELASTIN bounds the energy consumption to ensure forward progress.

[3]In other words, the page has already been accessed before in the current checkpoint interval.

Discussion: PNV is small enough to keep the lookup cost low. In energy-harvesting systems, the common case is that they encounter frequent power failures, e.g., in a few tens of milliseconds. During the short power-on period (one charge cycle run time), PNV is populated with only a handful number of pages in reality. Another reason for the small size of PNV is spatial locality; many stores fall into a few previously-populated pages during the short period of intermittent execution [55].

In particular, the size of PNV never grows unboundedly. To avoid stagnation not only at run time but also at boot time, ELASTIN bounds the number of pages, that can be populated at run time, by taking into account their restoration cost at boot (i.e., recovery) time. Section III-D shows how ELASTIN bounds the number of the populatable pages.

C. Adaptive Execution

To enable energy efficient intermittent computation, ELASTIN dynamically adjusts the checkpoint interval and the page size at both timer expiration time and boot time if needed.

1) Checkpoint Interval Adaptation: ELASTIN reconfigures the checkpoint interval by taking into account the condition (quality) of the energy harvesting source. This harvesting condition is an important factor for ELASTIN to determine whether the checkpoint interval should be adjusted or not. If the quality of the harvested energy is sufficiently good, there is no need to frequently checkpoint at run time; not doing so can make a better forward progress by saving the high energy of NVM writes required for the register checkpoint and the page backup. In contrast, if the harvested energy is not enough, a system should checkpoint before the impending power outage. In light of this, ELASTIN leverages the timer itself to figure out the underlying energy harvesting condition.

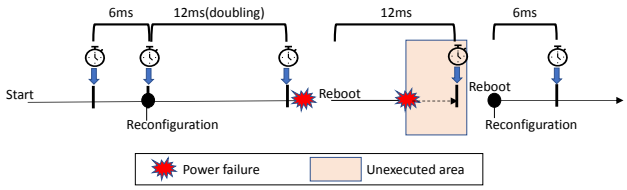


Fig. 4: Timer reconfiguration example

Figure 4 describes how ELASTIN reconfigures the checkpoint interval. If the timer expires two times in a row [4] while the energy-harvesting system is active, ELASTIN assumes that the system have gone through good energy harvesting condition. Thus, it doubles the checkpoint interval at the second timer expiration. The rationale behind this heuristic is that at the second timer expiration, at least the first checkpoint interval turns out to be unnecessary because it did not encounter a power outage; the harvested energy was that sufficient. However, the second interval should not be considered as unnecessary because the next interval may encounter a power outage.

On the other hand, if the timer has never expired since the last reboot, i.e., *checkpoint counter* is 0, then ELASTIN assumes that the system is under poor energy-harvesting condition. The intuition here is that the harvested energy was insufficient to pass even the first checkpoint interval without interruption due to power failure. With that in mind, ELASTIN sets the interval as a half of the last timer value in the wake of the power failure, i.e., at the reboot time. This particular approach (i.e., timer-halving mechanism) helps the system to overcome the stagnation problem for most of the time, though there are a few exceptional cases; Section III-D shows how ELASTIN handles them for stagnation-free intermittent computation.

2) Page Size Adaptation: To reduce the *copy-on-write* overhead, ELASTIN attempts to find the optimal page size; the spatial locality of memory writes is likely to vary due to program phase behavior [56], and therefore the best page size might vary for each phase.

In our current design, the memory page size cannot be changed at run time, which would otherwise cause significant metadata (e.g., PNV) updates overheads and a subtle correctness issue due to power failure between them. Instead, ELASTIN reconfigures the page size at reboot time as shown in Figure 1 to make the adaptation easier and still find the best size across power outages.

In the wake of a power outage, ELASTIN first restores all the pages populated in PNV. Then, it measures the cost of the current page configuration by the product of the page size and the number of populated pages, i.e., the size of PNV. Finally, ELASTIN resets the page size to the best-performing one using the decision logic of adaptive execution [57], [58].

That is, as decision runs, ELASTIN tries a set of page sizes to select the best among them across power outages; Section IV-E shows how the set is determined. Even though the best page size is selected at the end of decision runs, it is not fixed for the upcoming reboot times. Instead, at every reboot time, ELASTIN measures the cost of its current pick, which is compared to the most recent costs of the other page sizes, to see if it is still the best or one of them becomes the new best.

[4]To detect this, we use a metadata variable called *checkpoint counter*.

D. Challenges in Forward Progress Guarantee

When the system repeatedly starts at the same recovery point due to stagnation, ELASTIN reduces the checkpoint interval by halving the watchdog timer value in the wake of each power outage (Section III-C1). In this simple way, ELASTIN can effectively avoid the stagnation problem.

However, there are a couple of challenges that must be addressed to ensure forward progress for stagnation-free intermittent computation; (1) an excessive pages populated during a checkpoint interval, and (2) capacitor (i.e., energy buffer) malfunction due to wear-out, environmental factors such as temperature change, physical access attacks, and so on. First, if there are too many populated pages which must be restored at recovery time, the system may be stagnated in the middle of the recovery process. Second, if the capacitor is malfunctioning, the system may suffer from stagnation—e.g., the buffered energy is not enough to complete even single page backup or restoration.

To tackle these potential stagnation problems, ELASTIN defines thresholds for each condition: (1) the quota (i.e., maximum number) of populatable pages during a given checkpoint interval and (2) the lower bound of the power-on period (i.e., one charge cycle run time) of the energy harvesting system while its capacitor works fine. This paper assumes that the lower bound as the worst case scenario to ensure forward progress even in the most harsh situation, i.e., the lower bound is called **WCPT** (the worst case power consuming time) [5].

E. Stagnation-free Adaptation Solution

In this section, we first delve into WCPT and then show how to use it for detecting the capacitor malfunction problem. Finally, we show how WCPT can be used as a basis for solving the other problem, i.e., how to bound the number of the pages populated in a checkpoint interval.

1) Worst Case Power Consuming Time: Specifically, we define WCPT as follows: *how long can an energy harvesting system sustain its execution under the maximum power consumption mode?* To figure this out for the target energy harvesting system, ELASTIN analyzes its capacitor, i.e., energy buffer [6]. This is motivated by the insight that energy harvesting systems do not boot until the capacitor (energy buffer) is fully charged as with commodity systems such as WISP [61]. In the wake of each power outage, it is thus assured that the program can make as much progress as the fully charged capacitor allows, even if no additional energy is harvested. Section IV-F shows how ELASTIN calculates WCPT with this in mind and discusses how the calculation can be extended in case the system is equipped with other components such as sensors.

[5]ELASTIN can precisely bound WCPT due to the MSP430 MCU's simple architecture and execution environment, i.e., in-order core without cache/OS.

[6]A capacitor is used as an energy buffer [59], [60]. When an electric component depends upon a specific amount of power, the energy buffer is placed to provide the required power.

2) Energy Buffer Malfunction: Once WCPT is obtained, ELASTIN leverages it to detect the capacitor malfunction problem based on the following invariant: *the power-on period of an energy harvesting system should not be shorter than its WCPT*—as long as the capacitor works well. That is, if this invariant does not hold, the capacitor is malfunctioning. However, it is impossible for a timer to measure the power-on period because the timer value is reset on a power outage; Section IV-C shows how ELASTIN checks the invariant without measuring the power-on period.

If the capacitor turns out to be malfunctioning based on the invariant checking, ELASTIN treats this situation as an exception and switches to its handling mode. At the reboot time, ELASTIN first decreases the page size to the minimum (2 bytes) and then sequentially restores the registers and pages one by one in case there is an insufficient amount of energy for their restoration in a the batch manner. With the exception handling mechanism, ELASTIN can avoid stagnation even if the capacitor malfunctions—provided the system can run at least a single read/write instruction without interruption [7].

3) Populatable Pages: ELASTIN also leverages WCPT to determine the maximum number of the populatable pages with their boot-time restoration cost in mind. To ensure that at recovery (boot) time, all the pages populated in the last checkpoint interval can be safely restored, the total page restoration time must be shorter than WCPT, i.e., $Number\_of\_Pages * Single\_Page\_Restore\_Time < WCPT$; otherwise, power failure may occur in the middle of the restoration process. For the threshold of $Number\_of\_Pages$, ELASTIN therefore uses the maximum value among those that satisfy the above inequality. In this way, when ELASTIN reconfigures the page size at boot time, the threshold is also updated according to the new page size. If the number of the pages populated in a checkpoint interval happens to exceed the threshold, which is detected by checking a metadata variable called *populated page counter*, ELASTIN makes an additional checkpoint right at the moment. This allows ELASTIN to safely restore all the pages at the next reboot time without interruption due to power failure.

IV. IMPLEMENTATION

A. Register Checkpointing, Permission Clearing Protocol

As shown in Section III-B, at each timer expiration, ELASTIN checkpoints all registers including PC with double buffering and invalidates out the write permission of all pages. For this purpose, ELASTIN maintains two bits: (1) a *double buffer index bit* that is toggled at the end of the register file checkpointing and (2) a *PNV valid bit* whose reset invalidates the write permission of all pages. Note that these two bits must be atomically updated. Otherwise, a power outage between the two separate updates leads to incorrect recovery; in the wake of the power outage, ELASTIN ends up reverting the pages written in the formerly finished interval though it is about to start a new interval from the checkpointed PC, not the former. To avoid the incorrect recovery, ELASTIN updates the two bits in a single store instruction that guarantees failure atomicity [62]. Once they atomically updated, ELASTIN clears out all page numbers in PNV by using a single DMA operation [8] as will be shown in Section IV-D; the amount of the DMA write is determined by *populated page counter*. Once it is successfully done, ELASTIN finally sets the counter to zero before starting the new interval.

B. Memory Organization

ELASTIN divides the whole nonvolatile memory into four areas: main (original) memory, shadow memory, register double buffer, and reserved memory for PNV and the rest of various metadata, i.e., the checkpoint counter, valid bits for checkpoint and PNV, a performance table of page sizes, the populated page counter, thresholds for the number of populatable pages and WCPT, and so on.

The biggest problem with shadow memory is that it occupies a half of the total memory size, thus failing to run those applications that have high memory footprints. To overcome this challenge, ELASTIN proposes another design choice, radix tree memory management; as OS implements the page table using a radix tree, we used the same kind of data structure. By using radix tree as backup page storage, ELASTIN can increase available main memory size for applications at the expense of the increased page search overhead. Section V evaluates the performance overhead of both shadow memory and radix tree.

C. Invariant Checking for Capacitor Malfunction Detection

To detect capacitor malfunction, ELASTIN uses the invariant of an intact capacitor, i.e., the power-on period (one charge cycle run time) should not be shorter than WCPT; see Section III-E2. However, ELASTIN cannot use a timer to measure the period because the timer value will be reset on power outage. To achieve the invariant checking without a timer, ELASTIN relies on the following observation: when the first checkpoint is not made—due to power outage—since the last reboot, we can infer that the power-on period must be less than the checkpoint interval; this is the reason ELASTIN to halve the interval (Section III-C1). With this in mind, ELASTIN detects the capacitor malfunction as follows.

While the capacitor malfunctions, ELASTIN keeps decreasing the checkpoint interval due to frequent power outages. Thus, after many outages and resumptions, the interval would eventually become WCPT at some recovery time. At the moment, if it turns out that no checkpoint was performed since the last boot, i.e., *checkpoint counter* is 0, then we know that the power-on period is definitely shorter than the interval (WCPT). Thus, we conclude that capacitor is malfunctioning. In short, when a checkpoint interval is the minimum (WCPT), if the checkpoint is not made before power failure, ELASTIN switches to the exception handling mode (Section III-E2).

[7] If the capacitor cannot even secure energy required for one memory instruction, ELASTIN assumes that the system is completely unusable.

[8] Even if the DMA operation fails due to power failure, ELASTIN does not lead to incorrect recovery. In the wake of the power failure, ELASTIN simply starts the DMA operation over and follows the rest of the protocol.

D. DMA-Based Fast Page Copy

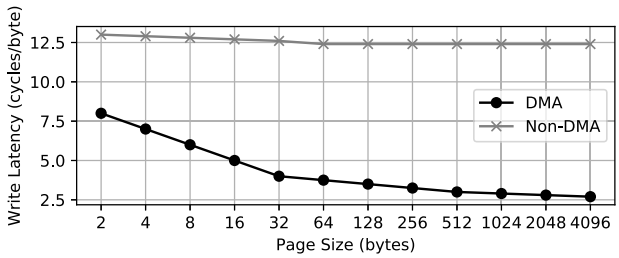


Fig. 5: NVM write latency with DMA (Cycles per byte)

ELASTIN's copy-on-write mechanism entails NVM writes for the page copy. However, due to the nature of NVM, memory writes incur very significant latency [63]–[72]. To reduce the overhead, ELASTIN leverages direct memory access (DMA) hardware accelerator available in the target energy harvesting system. Figure 5 demonstrates that a single byte DMA transfer takes only 8 cycles which is about 1.5X faster than the standard memory copy without DMA. When the copy size is larger than a single byte [73]. DMA copy becomes 4∼5X faster.

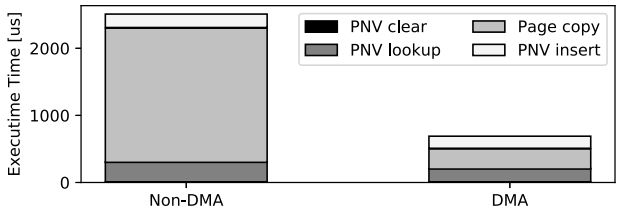


Fig. 6: Copy-on-write overhead breakdown in stable power input case. With the page size of 256 bytes, the major overhead comes from the page copy.

We also measured the impact of DMA on ELASTIN's per-page based copy-on-write mechanism for all of our benchmarks. Figure 6 shows the average execution time breakdowns of the copy-on-write for a 256B page with and without DMA. The page copy overhead occupies the most significant portion, and the PNV lookup overhead follows. With DMA, the overall execution time becomes about one third of the original time without DMA; this results from the large reduction of the page copy overhead which is about 6X speedup. In particular, the PNV clearing overhead is negligible because all page numbers are cleared by one DMA operation. In contrast, PNV insertions cannot be batched, since they are far apart from each other. Even though DMA can be leveraged for them, the DMA initialization and completion costs offset the benefit. Consequently, ELASTIN takes advantage of the DMA only for the page copying and the page clearing.

E. Page Size Adaptation Range

ELASTIN's page size selection is based on a series of decision runs for testing each size; see Section III-C2. Since most of them are suboptimal, ELASTIN tries to minimize the decision runs by limiting the range of page sizes to be tested.

To find the optimal page size, it is necessary to understand the tradeoff between the cost of PNV copying and the cost of page clearing. For example, if the page size is too small, it may incur frequent page copies causing expensive PNV clearing cost at reboot time or timer expiration. In contrast, if the page size is too large, the system may consume too much energy for copying even on page. In addition, the spatial locality of memory writes is another important factor. The high locality lets ELASTIN skip the page copy and PNV insertion since the memory writes are likely to be concentrated on a few pages. while the low locality increases them since many writes tend to touch many different pages.

The tradeoff is affected by the locality, e.g., with the high locality can amortize the cost of a large page copy by many subsequent writes whose address falls into the same page. To a large extent, the locality significantly varies across applications due to their different pattern of memory writes. With that in mind, we empirically measured the performance of each page size for all of our benchmarks. Figure 7 shows the average execution time overhead of the best 4 page size configurations, i.e., 32B, 64B, 128B, 256B. As a result, ELASTIN's adaptive execution uses them for decision runs, i.e., the page size adaptation range is 32∼256 bytes.

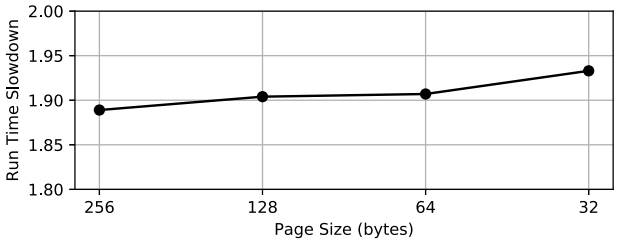


Fig. 7: Average execution time overhead of the best 4 page size configurations for all benchmarks when DMA and shadow memory are used with stable power input, i.e., no power failure.

F. Worst Case Power Consuming Time

In this paper, WCPT is defined as: how long a program can sustain its execution under the maximum power consumption mode of the microcontroller (MCU) which drains the energy from the capacitor at the highest rate. To measure WCPT, ELASTIN needs to know the energy buffer size (capacitance), because the MCU may rely on only the buffer without any input from harvesting energy sources in the worst case. For a given capacitance of the energy buffer (e.g., 47μF in WISP5), it provides the MCU with the operating voltage from its starting point (V_{max}) to the power outage point (V_{min}). Then, ELASTIN estimates the available energy input as follows:

$$Available\ Energy\ Input = \frac{1}{2} C_{buf} * (V_{max}^2 - V_{min}^2). \quad (1)$$

For the maximum power consumption estimation of MCU, ELASTIN leveraged the following equation [74] :

$$E_{tot} = P_{tot}t = V_{dd}I_{leak}t + C_{msp}V_{dd}^2 \quad (2)$$

where V_{dd}, I_{leak}, and C_{msp} are input voltage to MCU, leakage current, and the MCU capacitance, respectively. ELASTIN considers the input voltage to MCU by taking into account the capacitor discharge behavior, since the capacitor cannot consistently provide the same amount of power. ELASTIN models the input voltage variation while the energy buffer is discharged with a simple equation: $v_o(t) = V_o e^{\frac{-t}{CR}}$ for which the capacitance (C) is already given by $Equation$ 1, and the resistance (R) can be calculated by Ohm's law, $R = V/I$. ELASTIN views the MCU as a huge constant resistor, R, under the maximum power consumption mode. That is, ELASTIN refers to the MCU manual to figure out the maximum current (I)—that the device can consume—and use it to calculate the resistance (R). For I_{leak} and C_{msp}, ELASTIN refers to the manual as well; If a certain MCU's manual does not specify them in any case, ELASTIN can adapt the typical leakage current and capacitance model as one used in [74]. With all these findings, the available energy input obtained by $Equation$ 1 should be always greater than the energy consumption of the underlying MCU given by $Equation$ 2. With that in mind, ELASTIN calculates the WCPT by calculating a threshold t in the following equation:

$$\frac{1}{2}C_{buf} * (V_{max}^2 - V_{min}^2) > V_o e^{\frac{-t}{CR}}(I_{leak})t + C_{msp}(V_o e^{\frac{-t}{CR}})^2 \quad (3)$$

In particular, this is applicable to commodity energy harvesting devices. For instance, WISP5 consists of 47μF energy buffer and MSP430FR5969. This MCU consumes 2650μA at 3.0V, 16MHz, in an active mode [75]. The MCU starts to operate at 2.4V and performs down to 1.8V while the resistance value of the MCU is 1133Ω. Therefore, the resulting WCPT is approximately 11.6ms.

Discussion: Thanks to the simplicity of the above analytical model, it is easy for ELASTIN to incorporate other system components in the WCPT calculation. For example, if the system is equipped with other components, e.g., sensors and actuators. For this purpose, ELASTIN needs to update the resistance part of $Equation$ 3, i.e., $R = \frac{V}{I_{MCU}+I_{Sensor}+I_{Actuator}}$. To figure out the maximum current of the components, ELASTIN simply refers to their manuals as usual.

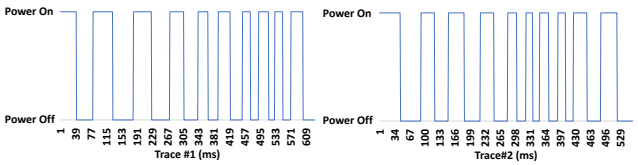
V. EVALUATION

We conducted all the experiments on TI's MSP430FR5994 Launchpad development kit board[9] and implemented ELASTIN described in Section III as a runtime library. To instrument nonvolatile memory (NVM) writes (Figure 3), we implemented a source-to-source translator using the LLVM compiler infrastructure [76]. Then, the instrumented program and the runtime library are compiled and linked using TI's MSP430 GCC toolchain to generate the binary executable.

[9]FRAM is used as main memory, and we do not use SRAM at all.

To compare ELASTIN with Ratchet [23], the state-of-the-art region based work, we ported it to MSP430 since it was originally implemented for ARM [77]. Note that we omitted Ratchet's timer based checkpointing, because it does not work—i.e., it may cause incorrect recovery—for those idempotent regions that contain WARAW (Write-After-Read-After-Write) dependence as admitted by the author [23]. We evaluated both ELASTIN and Ratchet for total 11 benchmarks comprised of a subset of MiBench applications [78], [79] and others from prior works [28], [36]. All the benchmark applications were compiled with standard -O3 optimization.

A. Intermittent Computing Platform



(a) Power trace 1 (b) Power trace 2

Fig. 8: Realistic intermittent power traces (simplified)

We developed a special power generator board with TI's MSP430FR5969 to mimic various power outages and resumptions as with prior work [80]. The power generator board provides supply voltage between 0 to 3.3V, directly to the target evaluation board (i.e., TI's MSP430FR5994) through GPIO pins to power it on/off at will based on power input traces. Unlike prior works [23], [25], [26], [28], [55] that do not vary the power failure frequency, we randomly increase and decrease the power-on period to model various energy sources and environments, which serves to stress-test ELASTIN. The minimum bound of the power-on period is set to 15ms [10] while the minimum bound to a half of the execution time of the smallest application among our benchmarks. With that in mind, we synthesized two power traces for our intermittent computing experiments as shown in Figure 8. At power-on, the power generator board provides voltage to the target board while it cuts the voltage at power-off for outage.

B. Execution Time Overhead Analysis with No Power Failure

We first analyze ELASTIN's execution time overhead when the power source is stable, i.e., there is no power failure. Here, we set the baseline to the uninstrumented binaries that have no checkpoint/restart support. We measured the overhead of ELASTIN for 11 benchmark applications varying the page size from 2B to 4KB and alternating the backup page storage between a shadow memory and a 2-level radix tree data structure. Figure 9 shows the normalized overhead of ELASTIN compared to the baseline when shadow memory is used while 10 shows that when the 2-level Radix tree is used. Overall, the average overhead of ELASTIN is 88%

[10]WISP5 [61], a commodity energy-harvesting system, has an a capacitor of 47μF, and it can sustain about 15ms as one charge cycle run time [28].

with the best page size of 256 bytes in shadow memory (see Figure 7) while 2-level Radix tree results in 243% with the same page size as the best. Nevertheless, it would be a mistake to take this to mean that ELASTIN incurs such a significant overhead for intermittent computation. Recall that frequent power failures are the norm in energy-harvesting systems, and this particular experiment has no power failure at all; Section V-C evaluates the performance impact of ELASTIN on intermittent computation with frequent power outages.

As shown in the figures, some applications such as bitcnt, dijkstra, and stringsearch prefer larger page size. Even if the copy-on-write of a large page size is expensive due to high volume of NVM copy, the cost is amortized by high spatial locality in the following memory writes. In contrast, the other applications show performance degradation when the page size is bigger than 256 bytes. That is because the cost of such a large page copy cannot be paid off in the applications due to their low spatial locality. The takeaway is that the best page size varies depending on application characteristics. Furthermore, the best page might vary even during program execution due to phase behavior [56]. In such a case, ELASTIN's boundary-free adaptive execution can find the right page size across power failures.

C. Execution Time Overhead Analysis with Power Outages

To evaluate the forward execution progress in the presence of power failures, we measured the application's completion time i.e., the execution time taken to complete the application across power failures. In Figure 11 and Figure 12, total three cases are compared using the two power traces shown in Figure 8: the state-of-the-art [23], i.e., Ratchet in the legend, ELASTIN with timer only adaptation, i.e., ELASTIN (timer), and ELASTIN with both timer and page size adaptation, i.e., ELASTIN (timer+page). Note that in Figure 11 (trace#1) and Figure 12 (trace#2), we set the baseline to our approach, i.e., ELASTIN (timer+page) because Ratchet makes many applications stagnate.

As shown in the figures, Ratchet [23] incurs stagnation problem in five applications on both traces. For the rest applications where Ratchet does not stagnate, ELASTIN (timer+page) outperforms Ratchet on average by 3.5X and 3X for trace#1 and trace#2, respectively. Also, it turns out that ELASTIN (timer+page) improves ELASTIN (timer) on average by 40% and 8% for trace#1 and trace#2, respectively. This confirms that ELASTIN's boundary-free 2-dimensional (timer and page size) adaptation works effectively.

Interestingly, Figure 11 shows that Ratchet could outperform ELASTIN for basicmath. That is because in basicmath, Ratchet happens to have the optimal size of regions which corresponds to the input power cycle characteristics, i.e., trace#1. Thus, when a different power trace is used, Ratchet cannot beat ELASTIN, which is confirmed by our experiment with trace#2. As shown in Figure 12, for the same application (basicmath), ELASTIN significantly outperforms Ratchet under trace#2.

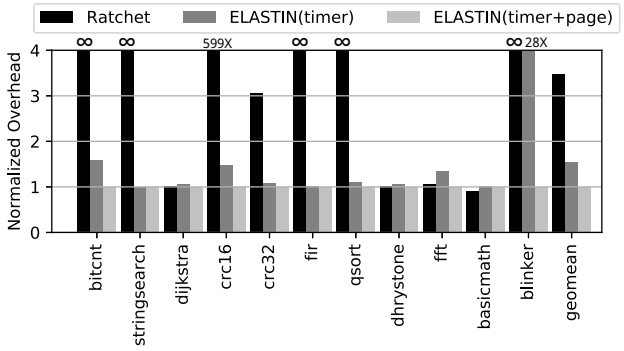


Fig. 11: Application completion time in the presence of power failures using trace#1: the bar of stagnated applications reaches ∞, and the geomean of Ratchet is calculated only for non-stagnated applications.

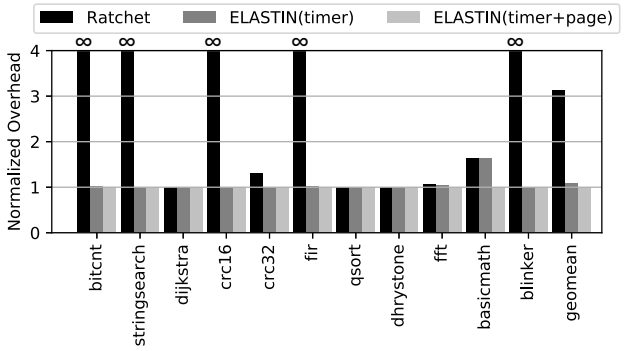


Fig. 12: Application completion time in the presence of power failures using trace#2: the bar of stagnated applications reaches ∞, and the geomean of Ratchet is calculated only for non-stagnated applications.

D. Energy Consumption Breakdown across Power Outages

We also analyzed the energy consumption breakdown across power outages. Figure 13 shows that *copy-on-write* and the register checkpoint (ckpt in the legend) do not consume significant amount of energy, i.e., less than 1% on average. That is because the average number of copied pages in intermittent computation is only about 1 or 2 thanks to spatial locality and high power outage frequency; as shown in Figure 6, the PNV lookup overhead is trivial as well, and thus overall copy-on-write overhead is not significant.

Overall, the overhead of ELASTIN comes from the re-execution cost; in the legend, 'forward' means the energy consumption for a portion of execution time that has never been restarted, thus it is not an overhead technically. Even if ELASTIN reconfigures the checkpoint interval by halving the previously selected interval when the system dies without forward progress, the re-defined checkpoint interval may not help for the first time to make progress. For example, to get out of stagnation, ELASTIN might need to perform multiple times of checkpoint interval halving across power failures, and the re-execution of the reduced intervals consumes the harvested energy. As shown in Figure 13, this re-execution overhead is

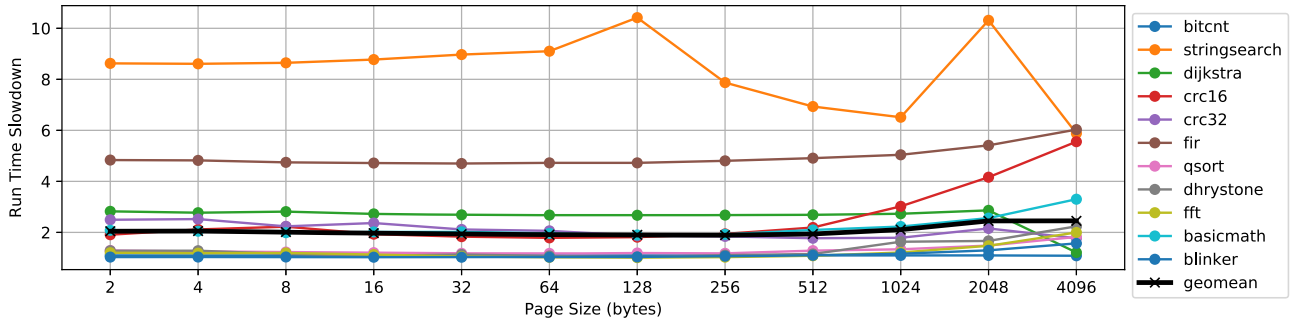


Fig. 9: Normalized performance overhead of ELASTIN *(shadow memory design)*

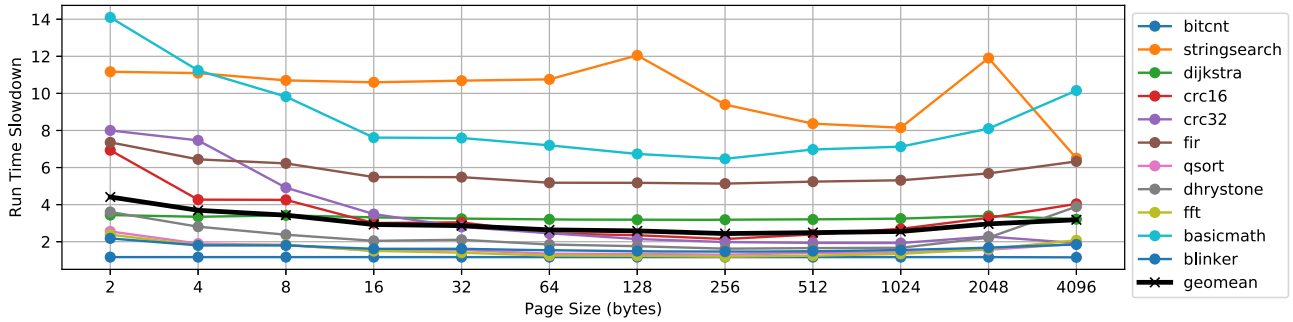


Fig. 10: Normalized performance overhead of ELASTIN *(2-level radix tree design)*

about 39% on average.

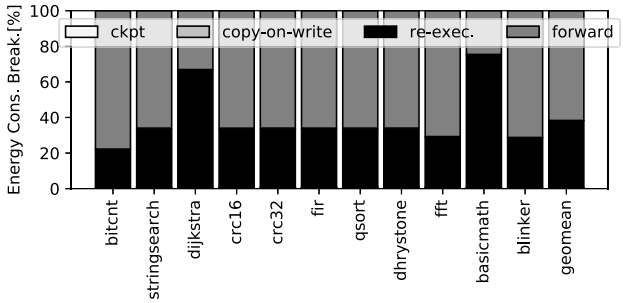


Fig. 13: Energy consumption breakdown of ELASTIN

E. Exception Handling for Capacitor Malfunction

Capacitor (i.e., energy buffer) can malfunction either by natural worn-out or physical security attacks [43]. For example, when cracked, the capacitor leaks the buffered energy more quickly and the original capacitance is significantly reduced [41]. To have a scenario of the malfunctioning capacitor, we set the power-on period to 5X lower than the normal minimum bound mimicking the cracked energy buffer. That is, the system only runs for 3ms intermittently. As shown in Figure 14, Ratchet was unable to complete all of benchmark applications. In contrast, ELASTIN successfully completes them all. This implies that ELASTIN is not only robust against capacitor malfunction but also capable of working with smaller capacitance, e.g., less than 10μF. Consequently, we believe that ELASTIN enables using a smaller capacitor, which should be a desired approach for smaller chips required for IoT industry such as wearable markets.

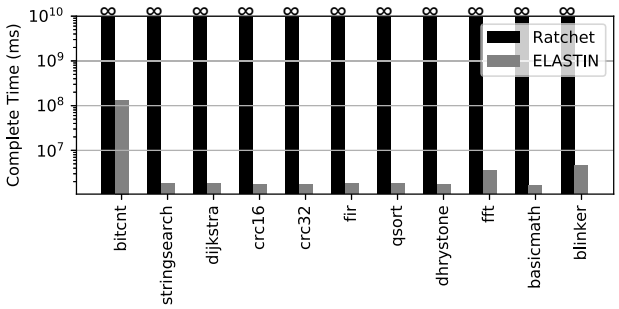


Fig. 14: ELASTIN*'s robustness against capacitor malfunction*

VI. RELATED WORKS

The problem of ensuring data consistency and improving the forward progress of an intermittently powered system is at the heart of energy harvesting computing. Various methods have been proposed from hardware designs to software solutions.

a) Hardware Schemes: To solve the problem, researchers have designed the nonvolatile processor. Wang *et al.* propose utilizing nonvolatile flip-flops (NVFF), as their fundamental

block for register checkpointing [40]. NVFF leverages a hybrid CMOS and ferroelectric technology in which a backup ferroelectric capacitors (FeCap) are coupled to a standard CMOS D latches. Lui *et al.* [8], [81] expand on the work of Wang *et al.* by additionally designing nonvolatile SRAM while Liu and Jung [8], [82] design nonvolatile gated store buffer for consistency-aware checkpointing; their designs are roughly the same hybrid technique as NVFF.

The benefit of such technologies would be a fast transfer of bit data because of the close proximity to the storage source. However, to checkpoint volatile states to NVFF, the scheme requires a voltage monitor. For instance, when the system detects voltage drops, the processor backs up volatile states to NVFF. However, Cronin *et al.*'s recent work reported that the design is vulnerable to frequent bit flip attacks [43], [44] or checkpoint failure [43], [45]. ELASTIN does not rely on unconventional hardware support yet it can address the energy buffer issue by treating the capacitor malfunction as an exception.

Colins *et al.* propose a reconfigurable energy buffer with multiple capacitor banks [49]. Since a different task may require a different amount of energy, they attempt to reconfigure the energy capacity to match the application demand. However, this scheme overlooks the capacitor malfunction issues that can be caused by wear-out, environmental factors such as temperature change, and even physical access attacks. When the capacitor is malfunctioning, the scheme does not work properly. In contrast, ELASTIN can survive the capacitor malfunction issues with its exception handling mode that works orthogonally to the capacitor.

b) Software Schemes: A number of software solutions have been proposed for the past few years. They differ how the correct placement of checkpoints, what occurs during checkpointing, and whether the method is automated or able to ensure forward progress. As for the correct placement of checkpoints, two starkly different approaches are given. In the research by Xie *et al* [35], [36], they present algorithms to figure out when is a safe to perform checkpoints and how to minimize them as a heuristic. This design revolves around the idea of severing anti-dependencies that appear in the code by placing a checkpoint between the dependent load-store pair.

As for the other approach [26], [28], [37], [53], this burden is placed on programmers. For example, they are required to determine re-executable task boundaries on their own by taking into account potential memory inconsistency and power failure during the task execution. That is, it is up to programmers to make good judgment of whether a task is considered idempotent or free from stagnation. Unfortunately, all of these software schemes end up splitting an program into several regions (or tasks) by re-compilation or user-intervention. In contrast, ELASTIN never places such a burden on end users thanks to its boundary-free and fully-automated nature.

Colins *et al.* [27] propose to use an energy debugger [47], [48] to deal with the stagnation phenomenon. They also found that the state-of-the-arts including task-based schemes [26], [28], [37], [53] suffer from the "non-terminating" bug and tried to partition the stagnated tasks. However, the scheme requires multi-step user interventions, i.e., energy profiling, energy checking over the profiled paths, and boundary placement, and so on, without considering energy buffer malfunction/vulnerability. Although the energy debugger approach can measure the whole system energy including MCU, sensors, and actuators, it cannot measure the worst case energy consumption thereby failing to ensure forward progress. It is rather a very complicated process to measure the maximum current consumption of the entire system, because it requires expensive experimental settings such as heating/cooling chambers for precise measurement. In fact, the manufacturers of the MCU, sensors, and other components measure the maximum current in that expensive way, considering temperature variation and other factors. In light of this, ELASTIN simply refers to the manuals to figure out the maximum current and incorporates it to the analytical model for WCPT calculation.

On top of the profile directed task partitioning, Chinchilla [25] recently introduced a dynamic adaptive checkpoint scheme using a timer and a undo logging mechanism. Chinchilla also reconfigures the timer interval according to system performance, but it checkpoints at pre-determined program point determined by the energy debugger [27], [48] for forward progress. As this paper discussed, capacitor can be worn out, and thus the energy buffer can behave differently from its original behavior. Consequently, Chinchilla is unable to guarantee forward progress though it requires substantial manual effort. In addition, its undo logging scheme fixed the logging granularity (i.e., 8 bytes) without DMA support resulting in lower performance. In contrast, ELASTIN is user-intervention-free, boundary-free, and fully-adaptive. Moreover, it strongly guarantees forward progress even when the energy buffer is malfunctioning or under attacks.

VII. SUMMARY

This paper presents ELASTIN, a stagnation-free intermittent computing system that ensures forward progress in the presence of frequent power outages. ELASTIN leverages both timer-based checkpointing of volatile registers and copy-on-write mappings of nonvolatile memory pages to restore them in the wake of power failure. Unlike prior works, ELASTIN does not partition program into recoverable regions or tasks. The boundary-free nature allows ELASTIN to realize full potential of adaptive execution, adjusting both the checkpoint interval and the page size at will. Consequently, ELASTIN can achieve stagnation-free intermittent computation and maximize forward progress across power outages.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their valuable comments and Matthew Hicks for confirming our understanding of Ratchet and its timer based checkpointing. At Virginia Tech, this work was supported by NSF grants 1750503 (CAREER Award) and 1814430, and by Google/AMD Faculty Research Awards. At ETRI, this work was supported by the ICT R&D program MSIT/IITP 2017-0-00051.

REFERENCES

[1] E. C. Lab, "Wearable technology and the internet of things," 2016. https://www.ericsson.com/assets/local/networked-society/consumerlab/reports/wearable-technology-and-the-internet-of-things-ericsson-consumerlab-2016.pdf.

[2] T. A. D. A. E. T. W. Group, "Theinternetofme:howwearabletechischangingtheinternetofthings," 2017.

[3] C. Perera, C. H. Liu, and S. Jayawardena, "The emerging internet of things marketplace from an industrial perspective: A survey," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 4, pp. 585–598, 2015.

[4] S. Mauilk, "Wearables and internet of things 2015," 2015. http://www.peoplepowerco.com/wp-content/uploads/2015/09/wp-wearables-iot.pdf.

[5] M. S. Reserach, "Wearable devices: The internet of things becomes personal," 2014. http://byinnovation.eu/wp-content/uploads/2014/11/MORGAN-STANLEY-BLUE-PAPER_Internet-of-Things.pdf.

[6] B. Campbell, B. Ghena, and P. Dutta, "Energy-harvesting thermoelectric sensing for unobtrusive water and appliance metering," in *Proceedings of the 2nd International Workshop on Energy Neutral Sensing Systems, ENSsys '14, Memphis, Tennessee, USA, November 6, 2014*, pp. 7–12, 2014.

[7] L. Rizzon, M. Rossi, R. Passerone, and D. Brunelli, "Wireless sensor networks for environmental monitoring powered by microprocessors heat dissipation," in *Proceedings of the 1st International Workshop on Energy Neutral Sensing Systems*, ENSSys '13, (New York, NY, USA), pp. 8:1–8:6, ACM, 2013.

[8] Y. Lui, Z. Li, H. Li, Y. Wang, X. Li, K. Ma, S. Li, M.-F. Chang, J. Sampson, Y. Xie, J. Shu, and H. Yang, "Ambient energy harvesting nonvolatile processors: From circuit to system," in *Proceedings of the 52nd Annual Design Automation Conference*, DAC '15, (New York, NY, USA), ACM, 2015.

[9] H. Jayakumar, K. Lee, W. S. Lee, A. Raha, Y. Kim, and V. Raghunathan, "Powering the internet of things," in *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, ISLPED '14, (New York, NY, USA), pp. 375–380, ACM, 2014.

[10] H. Jayakumar, A. Raha, and V. Raghunathan, "Quickrecall: A low overhead hw/sw approach for enabling computations across power cycles in transiently powered computers.," in *VLSI Design*, pp. 330–335, IEEE Computer Society, 2014.

[11] H. G. Lee and N. Chang, "Powering the iot: Storage-less and converter-less energy harvesting," in *Proceedings of Asia and South Pacific Design Automation and Conference (ASP-DAC)*, 2015.

[12] C. Wang, N. Chang, Y. Kim, S. Park, Y. Liu, H. G. Lee, R. Luo, and H. Yang, "Storage-less and converter-less maximum power point tracking of photovoltaic cells for a nonvolatile microprocessor," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pp. 379–384, Jan 2014.

[13] W. S. Lee, H. Jayakumar, and V. Raghunathan, "When they are not listening: Harvesting power from idle sensors in embedded systems," in *Proceeding of the 5th International Green Computing Conference (IGCC),*, 2014.

[14] S. Beeby and N. White, *Energy Harvesting for Autonomous Systems*. Artech House Series Smart Materials, Structures, and Systems, Artech House, Incorporated, 2014.

[15] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive wi-fi: Bringing low power to wi-fi transmissions.," in *NSDI*, vol. 16, pp. 151–164, 2016.

[16] H. Aantjes, A. Y. Majid, and P. Pawełczak, "A testbed for transiently powered computers," *arXiv preprint arXiv:1606.07623*, 2016.

[17] J. Hester, K. Storer, L. Sitanayah, and J. Sorber, "Towards a language and runtime for intermittently-powered devices," *sleep*, vol. 9, p. 10, 2016.

[18] K. S. Yıldırım, H. Aantjes, A. Y. Majid, and P. Pawełczak, "On the synchronization of intermittently powered wireless embedded systems," *arXiv preprint arXiv:1606.01719*, 2016.

[19] E. Nwafor, A. Campbell, D. Hill, and G. Bloom, "Towards a provenance collection framework for internet of things devices," in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 1–6, IEEE, 2017.

[20] S. Nirjon, "Lifelong learning on harvested energy," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 500–501, ACM, 2018.

[21] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, 2018.

[22] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel, "Intermittent computing: Challenges and opportunities," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 71, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[23] J. V. D. Woude and M. Hicks, "Intermittent computation without hardware support or programmer intervention," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, (Savannah, GA), pp. 17–32, USENIX Association, 2016.

[24] B. Lucia and B. Ransford, "A simpler, safer programming and execution model for intermittent systems," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 575–585, ACM, 2015.

[25] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, (Carlsbad, CA), pp. 129–144, USENIX Association, 2018.

[26] A. Colin and B. Lucia, "Chain: Tasks and channels for reliable intermittent programs.," in *In Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pp. 514–530, ACM, 2015.

[27] A. Colin and B. Lucia, "Termination checking and task decomposition for task-based intermittent programs," in *Proceedings of the 27th International Conference on Compiler Construction*, pp. 116–127, ACM, 2018.

[28] A. C. Kiwan Maeng and B. Lucia, "Alpaca: intermittent execution without checkpoints," in *Proc. ACM Program. Lang.1, OOPSLA, Article 96*, October 2017.

[29] M. A. de Kruijf, K. Sankaralingam, and S. Jha, "Static analysis and compiler design for idempotent processing," in *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '12, (New York, NY, USA), pp. 475–486, ACM, 2012.

[30] S. Feng, S. Gupta, A. Ansari, S. A. Mahlke, and D. I. August, "Encore: low-cost, fine-grained transient fault recovery," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 398–409, ACM, 2011.

[31] Q. Liu, C. Jung, D. Lee, and D. Tiwari, "Compiler-directed lightweight checkpointing for fine-grained guaranteed soft error recovery," in *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*, pp. 228–239, IEEE, 2016.

[32] Q. Liu, C. Jung, D. Lee, and D. Tiwari, "Clover: Compiler directed lightweight soft error resilience," in *Proceedings of the 16th ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems 2015 CD-ROM*, LCTES'15, (New York, NY, USA), pp. 2:1–2:10, ACM, 2015.

[33] Q. Liu, C. Jung, D. Lee, and D. Tiwari, "Compiler-directed soft error detection and recovery to avoid due and sdc via tail-dmr," vol. 16, p. 32, ACM, 2016.

[34] Q. Liu, C. Jung, D. Lee, and D. Tiwarit, "Low-cost soft error resilience with unified data verification and fine-grained recovery for acoustic sensor based detection," in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, pp. 1–12, IEEE, 2016.

[35] M. Xie, C. Pan, J. Hu, C. Yang, and Y. Chen, "Checkpoint-aware instruction scheduling for nonvolatile processor with multiple functional units," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pp. 316–321, IEEE, 2015.

[36] M. Xie, M. Zhao, C. Pan, J. Hu, Y. Liu, and C. J. Xue, "Fixing the broken time machine: consistency-aware checkpointing for energy harvesting powered non-volatile processor," in *Proceedings of the 52nd Annual Design Automation Conference*, p. 184, ACM, 2015.

[37] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on rfid-scale devices," *Acm Sigplan Notices*, vol. 47, no. 4, pp. 159–170, 2012.

[38] M. Xie, M. Zhao, C. Pan, J. Hu, Y. Liu, and C. Xue, "Fixing the broken time machine: Consistency-aware checkpointing for energy harvesting powered non-volatile processor," in *Proceedings of The 52nd IEEE/ACM*

Design Automation Conference (DAC 2015), DAC '15, (New York, NY, USA), ACM, 2015.

[39] Q. Li, M. Zhao, J. Hu, Y. Liu, Y. He, and C. J. Xue, "Compiler directed automatic stack trimming for efficient non-volatile processors," in *Proceedings of the 52nd Annual Design Automation Conference*, p. 183, ACM, 2015.

[40] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Zhao, M.-F. Chiang, Y. Yan, B. Sai, and H. Yang, "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," in *2012 Proceedings of the ESSCIRC*, ESSCIRC '12, (Piscataway, NJ, USA), pp. 149–152, IEEE Press, 2012.

[41] A. Teverovsky, "Insulation resistance and leakage currents in low-voltage ceramic capacitors with cracks," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 4, no. 7, pp. 1169–1176, 2014.

[42] R. Shigeta, T. Sasaki, D. M. Quan, Y. Kawahara, R. J. Vyas, M. M. Tentzeris, and T. Asami, "Ambient rf energy harvesting sensor device with capacitor-leakage-aware duty cycle control," vol. 13, pp. 2973–2983, IEEE, 2013.

[43] P. Cronin, C. Yang, D. Zhou, K. Qiu, X. Shi, and Y. Liu, "'the danger of sleeping', an exploration of security in non-volatile processors," in *Hardware Oriented Security and Trust Symposium (AsianHOST), 2017 Asian*, pp. 121–126, IEEE, 2017.

[44] P. Cronin, C. Yang, and Y. Liu, "A collaborative defense against wear out attacks in non-volatile processors," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2018.

[45] P. Cronin, C. Yang, and Y. Liu, "Reliability and security in non-volatile processors, two sides of the same coin," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 112–117, IEEE, 2018.

[46] T. Instruments, "Msp430fr family of ultra low-power microcontrollers," 2015. http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/msp/ultra-low_power/msp430frxx_fram/what_is_fram.page.

[47] A. Colin, G. Harvey, B. Lucia, and A. P. Sample, "An energy-interference-free hardware-software debugger for intermittent energy-harvesting systems," *ACM SIGOPS Operating Systems Review*, vol. 50, no. 2, pp. 577–589, 2016.

[48] A. Colin, G. Harvey, A. P. Sample, and B. Lucia, "An energy-aware debugger for intermittently powered systems," *IEEE Micro*, vol. 37, no. 3, pp. 116–125, 2017.

[49] A. Colin, E. Ruppel, and B. Lucia, "A reconfigurable energy storage architecture for energy-harvesting devices," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 767–781, ACM, 2018.

[50] S. Arrhenius, "On the reaction velocity of the inversion of cane sugar by acids," in *Selected readings in chemical kinetics*, pp. 31–35, Elsevier, 1967.

[51] D. Ratkowsky, J. Olley, T. McMeekin, and A. Ball, "Relationship between temperature and growth rate of bacterial cultures.," *Journal of Bacteriology*, vol. 149, no. 1, pp. 1–5, 1982.

[52] S. W. Freiman and R. C. Pohanka, "Review of mechanically related failures of ceramic capacitors and capacitor materials," *Journal of the american ceramic society*, vol. 72, no. 12, pp. 2258–2263, 1989.

[53] B. Lucia and B. Ransford, "A simpler, safer programming and execution model for intermittent systems," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '15, (New York, NY, USA), pp. 575–585, ACM, 2015.

[54] R. E. Bryant and D. R. O'Hallaron, *Computer Systems: A Programmer's Perspective Plus MasteringEngineering with Pearson eText – Access Card Package*. Pearson, 3rd ed., 2015.

[55] M. Hicks, "Clank: Architectural support for intermittent computation," in *In Proceedings of ISCA '17*, ACM, 2017.

[56] X. Shen, Y. Zhong, and C. Ding, "Locality phase prediction," *ACM SIGPLAN Notices*, vol. 39, no. 11, pp. 165–176, 2004.

[57] C. Jung, D. Lim, J. Lee, and S. Han, "Adaptive execution techniques for smt multiprocessor architectures," in *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pp. 236–246, ACM, 2005.

[58] J. Lee, J.-H. Park, H. Kim, C. Jung, D. Lim, and S. Han, "Adaptive execution techniques of parallel programs for multiprocessors," *J. Parallel Distrib. Comput.*, vol. 70, pp. 467–480, May 2010.

[59] M. Chen, K. K. Afridi, and D. J. Perreault, "Stacked switched capacitor energy buffer architecture," *IEEE Transactions on Power Electronics*, vol. 28, no. 11, pp. 5183–5195, 2013.

[60] X. Wang, D. M. Vilathgamuwa, and S. S. Choi, "Determination of battery storage capacity in energy buffer for wind farm," *IEEE Transactions on Energy Conversion*, vol. 23, no. 3, pp. 868–878, 2008.

[61] "Wisp5 wiki," 2017. https://wisp5.wikispaces.com/WISP+Home.

[62] Q. Liu, J. Izraelevitz, S. K. Lee, M. L. Scott, S. H. Noh, and C. Jung, "ido: Compiler-directed failure atomicity for nonvolatile memory," in *51st Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2018, Fukuoka, Japan, October 20-24, 2018*, pp. 258–270, 2018.

[63] C. Pan, M. Xie, J. Hu, Y. Chen, and C. Yang, "3m-pcm: Exploiting multiple write modes mlc phase change main memory in embedded systems," in *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, p. 33, ACM, 2014.

[64] H. Aghaei Khouzani, Y. Xue, C. Yang, and A. Pandurangi, "Prolonging pcm lifetime through energy-efficient, segment-aware, and wear-resistant page allocation," in *Proceedings of the 2014 international symposium on Low power electronics and design*, pp. 327–330, ACM, 2014.

[65] T. Wang, D. Liu, Z. Shao, and C. Yang, "Write-activity-aware page table management for pcm-based embedded systems," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pp. 317–322, IEEE, 2012.

[66] H. A. Khouzani, C. Yang, and J. Hu, "Improving performance and lifetime of dram-pcm hybrid main memory through a proactive page allocation strategy," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pp. 508–513, IEEE, 2015.

[67] M. Zhao, Y. Xue, C. Yang, and C. J. Xue, "Minimizing mlc pcm write energy for free through profiling-based state remapping," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pp. 502–507, IEEE, 2015.

[68] C. Liu and C. Yang, "Improving multilevel pcm reliability through age-aware reading and writing strategies," in *Computer Design (ICCD), 2014 32nd IEEE International Conference on*, pp. 264–269, IEEE, 2014.

[69] M. Zhao, L. Shi, C. Yang, and C. J. Xue, "Leveling to the last mile: Near-zero-cost bit level wear leveling for pcm-based main memory," in *Computer Design (ICCD), 2014 32nd IEEE International Conference on*, pp. 16–21, IEEE, 2014.

[70] C. Wang and S. Chattopadhyay, "Lawn: boosting the performance of nvmm file system through reducing write amplification," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2018.

[71] C.-C. Ho, Y.-M. Chang, Y.-H. Chang, H.-C. Chen, and T.-W. Kuo, "Write-aware memory management for hybrid slc-mlc pcm memory systems," *ACM SIGAPP Applied Computing Review*, vol. 17, no. 2, pp. 16–26, 2017.

[72] T.-Y. Chen, Y.-H. Chang, S.-H. Chen, C.-C. Kuo, M.-C. Yang, H.-W. Wei, and W.-K. Shih, "wrjfs: A write-reduction journaling file system for byte-addressable nvram," *IEEE Transactions on Computers*, 2018.

[73] "Maximizing write speed on the msp430™ fram," Feb 2015. Accessed: 2018-10-14.

[74] A. Sinha and A. P. Chandrakasan, "Jouletrack-a web based tool for software energy profiling," in *In Proceedings of the 38nd Annual Design Automation Conference*, DAC '01, 2001.

[75] "Msp430fr59xx mixed-signal microcontrollers (rev. f)," Mar 2017. Accessed: 2017-11-08.

[76] C. Lattner and V. Adve, "Llvm: A compilation framework for lifelong program analysis & transformation," in *Proceedings of the International Symposium on Code Generation and Optimization*, CGO '04, (Washington, DC, USA), pp. 75–, IEEE Computer Society, 2004.

[77] M. Hicks, "Thumbulator: Cycle accurate armv6-m instruction set simulator," 2016.

[78] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pp. 3–14, IEEE, 2001.

[79] Q. Liu, X. Wu, L. Kittinger, M. Levy, and C. Jung, "Benchprime: Effective building of a hybrid benchmark suite," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, p. 179, 2017.

[80] A. Rodriguez Arreola, D. Balsamo, A. K. Das, A. S. Weddell, D. Brunelli, B. M. Al-Hashimi, and G. V. Merrett, "Approaches to transient computing for energy harvesting systems: A quantitative evaluation," in *Proceedings of the 3rd International Workshop on Energy Harvesting & Energy Neutral Sensing Systems*, ENSsys '15, (New York, NY, USA), pp. 3–8, ACM, 2015.

[81] K. Ma, X. Li, J. Li, Y. Liu, Y. Xie, J. Sampson, M. T. Kandemir, and V. Narayanan, "Incidental computing on iot nonvolatile processors," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 204–218, ACM, 2017.

[82] Q. Liu and C. Jung, "Lightweight hardware support for transparent consistency-aware checkpointing in intermittent energy-harvesting systems," in *Non-Volatile Memory Systems and Applications Symposium (NVMSA), 2016 5th*, pp. 1–6, IEEE, 2016.