Incidental Computing on IoT Nonvolatile Processors*

Kaisheng Ma[1], Xueqing Li[1], Jinyang Li[2], Yongpan Liu[2], Yuan Xie[3], Jack Sampson[1],
Mahmut Taylan Kandemir[1], and Vijaykrishnan Narayanan[1]

{kxm505, lixueq, sampson, kandemir, vijay}@cse.psu.edu, lijy15@mails.tsinghua.edu.cn, ypliu@tsinghua.edu.cn, yuanxie@ece.ucsb.edu

[1]*Dept. of Computer Science and Engineering, The Pennsylvania State University* [2]*Dept. of Electronic Engineering, Tsinghua University*

[3]*Dept. of Electrical and Computer Engineering, University of California at Santa Barbara*

ABSTRACT

Batteryless IoT devices powered through energy harvesting face a fundamental imbalance between the potential volume of collected data and the amount of energy available for processing that data locally. However, many such devices perform similar operations across each new input record, which provides opportunities for mining the potential information in buffered historical data, at potentially lower effort, while processing new data rather than abandoning old inputs due to limited computational energy. We call this approach **incidental computing**, and highlight synergies between this approach and approximation techniques when deployed on a non-volatile processor platform (NVP). In addition to incidental computations, the backup and restore operations in an incidental NVP provide approximation opportunities and optimizations that are unique to NVPs.

We propose a variety of incidental approximation approaches suited to NVPs, with a focus on approximate backup and restore, and approximate recomputation in the face of power interruptions. We perform RTL level evaluation for many frequently used workloads. We show that these incidental techniques provide an average of 4.2X more forward progress than precise NVP execution.

CCS CONCEPTS

• **Computer systems organization** → **Single instruction, multiple data**; **Special purpose systems**; **System on a chip**; **Embedded hardware**;

ACM Reference Format:
Kaisheng Ma, Xueqing Li, Jinyang Li,Yongpan Liu, Yuan Xie, Jack Sampson, Mahmut Taylan Kandemir and Vijaykrishnan Narayanan. Incidental Computing on IoT Nonvolatile Processors. In *Proceedings of MICRO-50, Cambridge, MA, USA, October 14–18, 2017,* 15 pages.
https://doi.org/10.1145/3123939.3124533

*This work was supported in part by NSF ASSIST, and also by LEAST, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA, as well as NSF 1213052/1500848/1719160. This work was also supported in part by NSFC Grant 61674094 and the Beijing Innovation Center for Future Chip.

MICRO-50, October 14–18, 2017, Cambridge, MA, USA

ACM ISBN 978-1-4503-4952-9/17/10…$15.00
https://doi.org/10.1145/3123939.3124533

1 INTRODUCTION

Every shift in the way our devices are connected or powered brings with it a potential for revolution in the usage and capabilities of the systems built around them. Just as the transition from wired to wireless telephones led to unprecedented changes in our communications and the shift from wall-power to battery-power transformed our expectations for computational systems, the shift from battery-powered systems to self-powered systems promises to fuel the next revolution in the *Internet of Things* (IoT). The ability to power IoT devices using ambient, scavenged energy liberates them from the lifetime, deployment, and servicing limitations of a fixed battery: Tens of billions of IoT devices are expected to pervade consumer, industrial and public services by the end of the decade [10], and, for many of the target applications in these areas, the replacement of batteries or creating infrastructure to provide a wired power supply makes an IoT-scale approach impractical from a cost perspective.

While ambient energy sources are notoriously fickle, concurrent advances in energy harvesting, ultra-low power computation, and non-volatile memory have enabled a new generation of processors, known as *non-volatile processors* (NVPs), that can withstand the significant temporal variations and even short spurts of "no power" that are common in such power profiles. NVPs tightly integrate non-volatile memory elements into the logic fabric of the processor, thereby enabling almost instantaneous stopping and starting of execution via distributed backup and restore functionality for processor state. Recent device and circuits design exploration in emerging nonvolatile logic and embedded memory has made NVPs faster and more energy-efficient with lower overheads in handling *in situ* parallel distributed backup and restore operations [16–18]. For NVPs with microarchitectural hardware-managed backup [30], systems can make persistent progress even if only one instruction successfully completes between power interruptions, and software approaches to manage the semantics of intermittent computation have emerged [22] that can leverage non-volatile memory elements to provide correct execution despite power interruptions. Advances in energy harvesting efficiency has enabled the powering of NVPs

from RF energy [64], motion harvesters [73, 74], ambient lighting [65] and thermal variations [15], all of which exhibit significant instability.

Prior efforts on hardware-managed NVPs that perform local computation (in contrast to sense-and-transmit only IoT models) have focused on enhancing the efficiency of converting harvested energy into persistently executed instructions [13, 21, 30, 53, 54, 77]. These techniques primarily focused on (1) reducing the number and overheads of backups and restores and (2) adapting the compute architecture to exploit dynamic variations in incoming power which would otherwise be wasted due to limited energy storage capability. However, the forward progress metric used in these works does *not* directly capture higher level application semantics regarding the "utility" of the work performed: *In many IoT applications, temporal and interactivity requirements can make the quality of partial results, or even the existence of any response at all, more important than the fraction of instructions needed to eventually produce a "best-quality" result.*

Adding a "quality knob" provides flexibility in an NVP, where the need to make conservative decisions regarding energy reserves for backup operations can otherwise impose substantial overheads on execution. In an NVP, if the effort needed to ensure preservation of data is sufficiently reduced, some power emergencies may be avoided, improving response timeliness. Moreover, in addition to natural synergies with power management, accepting variable quality responses frees a harvesting system to apportion effort with respect to the continued relevance of the data being processed: If an NVP has been without power for some substantial time, resuming work on the input it was processing when power failed may have lower utility, from an application perspective, than moving on to processing the newest input. Discovering the optimal allocation/schedule for an NVP would depend not only on application-specific semantics, but also future knowledge of unpredictable power income.

To capture these notions, we introduce the concept of **incidental approximate computation**, wherein communicated application tolerance for approximate outputs is used to maintain timeliness of responses from an NVP while taking advantage of repetitive behaviors within the NVP's workload to apply any power surplus, when available, to improve the quality of abandoned older work. The paper makes the following contributions:

- We introduce *incidental computing*, wherein older computation is carried out in a best-effort fashion during the execution of newer computations. For the energy-harvesting NVP scenario explored in this paper, this is done through bitwidth-oriented approximation techniques in the datapath, memory, and backup-recovery modules to divide power and resources and provide differential guarantees of output quality between the current and prior computations. We also propose incidental recomputing, wherein the quality of older computations targeted for incidental computing can be gradually improved iteratively if picked up over multiple incidental computing passes.

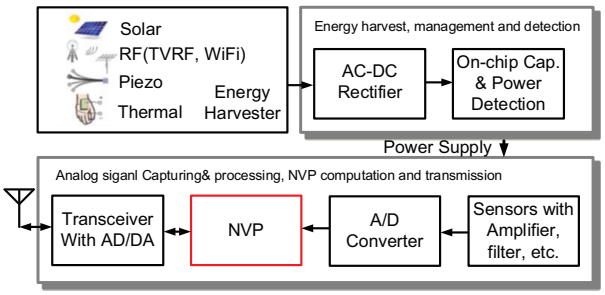

Figure 1: NVP-based energy-harvesting system.

- We propose incidental backup with several retention time matching models and supporting write circuits that can reduce the energy of backup operations through matching the retention time to the combination of the duration of power emergencies and the impacts of reduced fidelity to overall approximation quality.
- Collectively, the incidental approximation approaches improve forward progress by 4.28x improvement within tolerable quality loss.

2 SYSTEM MODEL

Figure 1 provides a block diagram for a general batteryless IoT system powered by ambient energy-harvesting techniques [19, 33, 39–41, 46, 60]. Such systems consist of a set of energy-harvesting, management, and detection components that comprise the power-provisioning front-end, as well as analog signal capturing and processing, NVP computation, and transmission units that implement the IoT tasks. Note that the front-end modules may vary depending on the energy sources. For our running example, we consider an NVP IoT platform in a "wristwatch form factor" using an unbalanced ring to harvest energy. The system uses an AC-DC rectifier following the rotational energy harvester. A capacitor is used to capture enough energy to ensure the NVP backup operation and to stabilize cycle-level execution voltages. As a result, the NVP is also in charge of system-level power policies that start, back up, or recover system state.

2.1 System Energy Distribution

We measured several prototype platforms in order to quantify the energy distribution of a typical wearable NVP system. The NVP system consists of an NVP running at 1MHz, costing 0.209mW, various sensors, and RF module with data rate 250kbps, costing 89.1mW. The distribution of energy needs between computation and communication varies significantly by application domain. For simple temperature sensing wireless sensing networks, NVP computation consumes 2.4% of total energy, for UV exposure metering, computation consumes 16.8% of energy, and, for more complex data processing like pattern matching and image processing, computation can consume 59.5% to 95% of energy depending on the data processing algorithms. These image signal processing algorithms are surprisingly common in many sensors, including gas sensing (spectrum analysis), water quality monitor (spectrum analysis

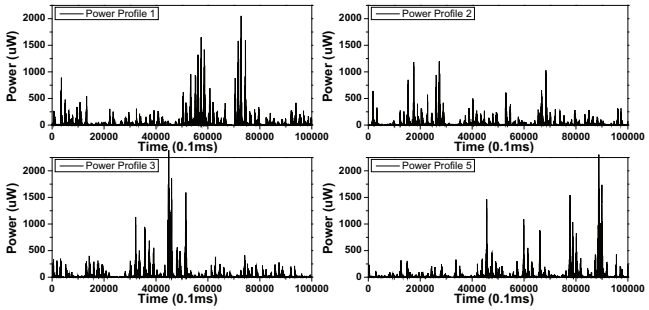

Figure 2: Power profiles of "watch" in daily life use

and image processing). Motivated by this, we focus our evaluation efforts on a collection of image signal processing kernels widely used in post-sensing data analysis as testbenches. One feature we observed is that the input data for these applications are usually buffered frame-by-frame, with no data dependencies between them. On the other hand, for the power levels available through an unbalanced ring class of harvester, more than 80% of the captured data may have to be abandoned in order to meet output deadlines due to weak data processing capability in the NVP. Processing the historical buffered data with incidental computing is one solution to get at least some low quality results, which can be enhanced later by "incidental recomputation", rather than abandoning these inputs entirely.

2.2 Turning Energy to Forward Progress

A wristwatch harvester can generate an average of 10uW to 40uW power in daily activities [73, 74]. However, these profiles are unstable, varying from 0 to 2000uW at a fine temporal granularity as shown in Figure 2. Assuming a processor operation threshold of 33uW, the system can experience 1000 to 2000 power emergencies in a 10s time window. Due to rollbacks and lost work in such an environment, many prior works propose **forward progress** (the number of instructions that have *persistently* committed) as a key "execution metric" for comparing the work done by processors.

A traditional strategy in energy-harvesting systems is to employ a volatile low power MCU or an MCU with checkpointing capability (e.g., FeRAM MSP430 is used in [39]) that waits before starting to execute while charging an energy storage device which must be large enough to store sufficient energy to complete an entire logical work unit, such as an image frame [39]. Systems operating on this paradigm will alternate between periods where they accumulate energy in the energy storage devices (ESD), and ones where powering the system drains the energy. While such a system is able to offer strong guarantees for execution once execution begins, this conventional solution has several limitations, including energy conversion efficiency overheads brought by frequently charging and discharging the capacitor, capacitor leakage [39, 55], minimum charging current(e.g. 20uA for the GZ115 [55]), and slow charging curve [55]). Moreover, if the incoming unit of work is too large, the incoming power may not be sufficient compared to leakage in the ESD [39], or there may be long periods of complete power outage that drain the accumulated charge, and it may take arbitrarily long to reach the threshold for beginning execution.

An alternative execution paradigm is to utilize only a small on-chip capacitor, i.e., one sufficient for backup operations and employ an NVP. This reduces capacitor leakage, and can improve front-end conversion efficiencies by mitigating the overheads of moving charge into and out of a large energy storage device at the cost of additional system complexity for the NVP, more complicated guarantees on the granularity of work accomplished once an execution period begins and the overheads imposed during more frequent backup and restore events during the execution of each logical unit of work. The two approaches can be seen as similar if the logical unit of work is at an instruction or similar granularity, thereby minimizing both charging time and charge lost if a shortfall occurs during a charging period between backup and recovery. Hybrid approaches have also been proposed. For example, Sheng *et al.* propose a dual channel front-end solution to overcome low charging efficiency [57] in which they design another power channel to bypass the energy storage device and connect directly to the load, and Ma *et al.* extend prior NVP models to maintain the capacitor energy level [24] within a bounded range for charging efficiency during execution rather than greedily consuming energy. *Thus, the key energy tradeoff between the two approaches is between the energy wasted on charging and discharging a capacitor with leakage and the backup and recovery overheads of NVP.*

In this work, we re-implement prior solution [24] and observe that the NVP-based execution approach can outperform the wait-compute scheme by 2.2X-5X for the power traces shown in Figure 2.

3 INCIDENTAL COMPUTING

There already exist various approximate computing techniques proposed in the literature (including dynamic bitwidth). In this work, in addition to employing several of these traditional techniques and evaluating them in energy harvesting scenarios, we propose a new approximation technique tailored for energy-harvesting computing, called *incidental computing*, and associated approximate backup/restore polices appropriate for both incidental computing and an NVP in an energy-harvesting domain.

3.1 Incidental Computing

Roll-forward Instead of Roll-back. We make the following key observations for IoT applications. In many deployment scenarios, catching up quickly after a power failure may take priority over the quality of response. Furthermore, such applications often contain kernels with independent loop iterations that could conceivably be skipped over in their entirety. However, skipping represents in a sense a maximum quality reduction, especially considering that each iteration, especially in image signal processing kernels, performs the same essential computation on different data. Finally, while average

power, even during periods of sufficient power to allow uninterrupted execution, is low in harvested systems, peak power can be substantially higher than average.

To take advantage of these observations, we propose *incidental* approximate SIMD computing for NVPs. Instead of rolling back after power failure, incidental computing *rolls forward* to process the most recent and (most of the time) most important new data. If there is additional power available beyond that needed to process the new data, then older data will be processed at reduced quality; incomplete executions from before a power failure are regarded as "incidental" and their importance drops over time. However, if the incidental low quality outputs computed indicate greater importance than expected and higher quality outputs are desired, *incidental recomputing* can be applied to enhance output quality.

Below, we discuss the details of incidental approximate SIMD computing. When a power failure happens, the computation states are backed up with the stored energy, and the data that are marked as incidental, like variable "src" in Figure 8, are backed up using the assigned unreliable storage policy in the NVM. When the power recovers, if a roll-forward is indicated ("incidental_recover_from"), instead of recovering from the backed-up PC, the PC is set to the place marked by "incidental_recover_from". From the application's perspective, the program rolls forward to process newer data from the buffer (in the example of program in Figure 8, it's a new frame of data). As a result, the newest captured data are always processed as the first priority.

During processing of the new data, the microarchitecture controller compares the current computation state to the state backed up using the old data. If there is a match, an SIMD strategy is applied to the old state and data. Note that, the computation precision of the newly-added SIMD for old data depends on the income power level under the control of the "incidental" pragma's "minbits" and "maxbits". In this way, a minimum quality can be guaranteed by "minbits", and the energy beyond the amount necessary for full precision processing of new data are instead applied to the old data for incidental SIMD computing. If the computation is interrupted again, both the new data and SIMDed old data become incidental, and a newest data computation begins from "incidental_recover_from". Note that multiple old data can be SIMDed. In our current implementation, at most 4-way SIMD can be achieved.

Recompute and Combine (RAC). We assume that, in general, the importance of data drops over time. If some old data are later found to be "interesting", and demand high precision output to validate "uncommon results", an *incidental recomputing* can be performed. Instead of inserting an interrupt into the current program, incidental recomputing employs incidental SIMD to recompute the old data, and tracks the precision of sub-component outputs. These two versions of the outputs can then be merged by combining the best precision sub-components from each run. After multiple recomputations and merges, we expect much better quality outputs as

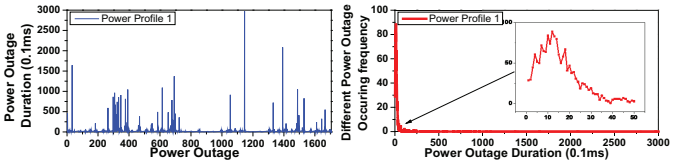

Figure 3: Power outage duration (left) and statistics (right)

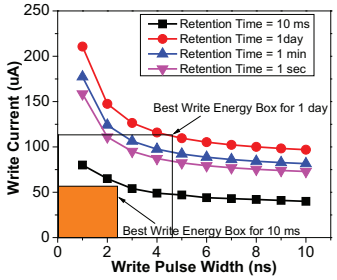

Figure 4: STT-RAM Write energy & retention time[12, 58, 63]

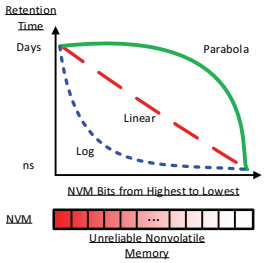

Figure 5: RTA

demonstrated in Section 8.5. It is important to emphasize that, in this incidental recompute method, a better quality result can be achieved without affecting the current data processing loop.

3.2 Incidental Backup

For the three power profiles shown in Figure 2, power supply unreliability would cause an NVP to perform as many as 1400 to 1700 backups per minute, costing 20.1% to 33% of the total income energy (simulated and measured with running testbenches shown in Figure 28). Approximate computing provides an opportunity to substantially mitigate these overheads by relaxing the reliability (i.e., write energy reduction brings the probability of flipped data storage beyond expected retention time) of the "lower order" NVM bits used to back up data during power emergencies, and using commensurately less energy for backup and recovery operations. Moreover, if the energy reserves needed for backup are reduced, fewer power emergencies may occur.

Retention Time Shaping (RTA). Current NVPs [13, 21, 53, 54, 77] utilize nonvolatile technologies with maximum retention times on the order of a decade or more, and parameters tuned to maximize both retention and reliability. However, most power emergencies in wearable harvesting devices last just a few ms, and are rarely more than a fraction of a second. Figure 3 plots the duration (left) and frequency of power emergencies (right) in the examined traces.

By matching the retention time to the power interval profile, the write energy can be significantly reduced. From the perspective of write energy for the backup operation, Figure 4 shows the relation between STT-RAM[1] write current and write pulse width for different retention times. We note that 77% of write energy can be saved, for instance, by reducing the

[1]ReRAM is an excellent option for infrequent backups. Here we choose STT-RAM mainly for endurance concerns for the backup rate associated with this specific energy harvester.

retention time from 1 day to 10 ms. However, applying a retention time reduction uniformly is very difficult to implement profitably for two main reasons: First, future power income is, in general, very difficult to predict, and, second, the cost of prediction failures can be very high.

Approximate computing eases the practical adoption of such an approach. Higher order bits are retained with longer duration, preventing catastrophic quality loss, while lower order bits can be unreliably persisted, saving energy. We consider three retention time reduction functions to shape the retention time in a way that reduces from the most significant bit to the least significant bit, as shown in Figure 5.

Our three retention time reduction policies are: linear (Equation 1), log (Equation 2), and parabola (Equation 3). B stands for bit index; for this example, it is from 1 to 8, and the T is the retention time, whose unit is 0.1ms.

Note that different kernels, and even different regions within these kernels, are differently sensitive to retention time shaping, which leads to different tradeoffs between energy savings and quality reduction. The log policy fits applications that have higher tolerance for approximation, such as neural network inference. The linear policy is suited for most applications, such as FFT, iFFT, etc. The parabola policy is the most conservative in maintaining upper bit fidelity. It is designed to match some algorithms that show significant quality loss when bitwidth is reduced under 4 bits.

We propose these three policies based on observations of the relationship between bitwidth precision and final result quality, considering both program features as well as power source profiles [29]. Through a quantitative analysis of the relationship, with MATLAB as a tool for regression and expression, we provide three retention time policies to trade off between energy and qualities.

$$T = 427B - 426 \quad (1)$$

$$T = \pi^{B-1} + 9 \quad (2)$$

$$T = -61B^2 + 976B - 905 \quad (3)$$

4 ARCHITECTURAL SUPPORT

In this section, we introduce the architectural support needed to implement the incidental approximate computing concept outlined in Section 3.

Microarchitecture Support for Incidental Computing. The high-level design changes and microarchitectural support needed for incidental computing are illustrated in Figure 6. A control unit (Figure 6, bottom) dynamically controls *whether approximation should be used and, if so, how.* The main task of this unit is to set the number of precise and approximate bits for SIMD for different hardware components based on the available power level. One of the outputs of this unit is approximation control bits, which are used to control approximation in the register files, ALU, pipeline flip-flops, and data memory. Approximation can also be globally disabled by a running program via unsetting the AC_EN register, overriding the decisions of the control module and forcing "full precision" execution.

Another important functionality is to manage the control transfers of incidental computing, namely *from which point the SIMD execution should begin*. This is decided by comparing the current PC and the values of key loop variables (e.g. induction variables analyzed by the compiler at compile time, such as variable "n" in Figure 8) against buffered values. To implement this, an additional circular nonvolatile buffer within the controller records the PC of the last N (four, in our implementation) resume-points from which the SIMD operation can begin. Once a power failure happens, the system states are backed up: the resume-point PC is backed up in a 2Byte*4 size buffer made of nonvolatile flip-flops in the controller; the register files are stored in multi-version non-volatile registers. The oldest value is overwritten (discarded in FIFO order). When incidental SIMD is enabled, the current PC is compared against stored resume-point PCs. If the current PC matches one of the stored PCs, the controller has the modified register file generate a bit-vector indicating which register values associated with the matching resume-point PC have values identical to the current register values. This vector is then combined with a compiler-generated mask. Once matches in both PC and the mask-indicated variables are observed, SIMD width is increased and the buffer storing the SIMDed resume-point PC is cleared.

A few other microarchitectural units also need to be modified to enable incidental computing:

Configurable approximate ALU: Direct extension to 4 versions. The ALU also performs approximate computation using the techniques proposed by [8, 75]. Our ALU can also work on packed SIMD operands.

Power-gated register files: Each register is designed with nonvolatile logic, has an AC bit, and is extended from 8 bits to 32 bits (4 versions) for incidental computing. These extensions can be powered off when incidental computing is not employed. Comparison circuits are also added to indicate an identical match between the current register value and the values of prior versions. Comparison is governed by the controller, which both enables the comparison circuits and specifies the version to compare against.

Data memory: The versioned NVM memory (there is no cache in this simple 5-stage-pipeline NVP) extends each word from 8 bits to 32 bits to support SIMD. An additional 3 bits for each data (12 bits in total for 4 SIMDs) are put to track the precision of the data. The memory implements the max, min, and other intra-bundle operations for merging the recomputed results.

Recomputation is implemented through an instruction marks a pragma-indicated control point into the nonvolatile PC buffer for comparison. The difference between incidental recomputation and normal incidental SIMD is that the programmer can set up the PC to be the one marked with "incidental recover from" rather than the one before power fails. After the recomputation finishes, an instruction requests the controller to use the multi-version memory to combine the new results with previous ones according to a specified policy conveyed

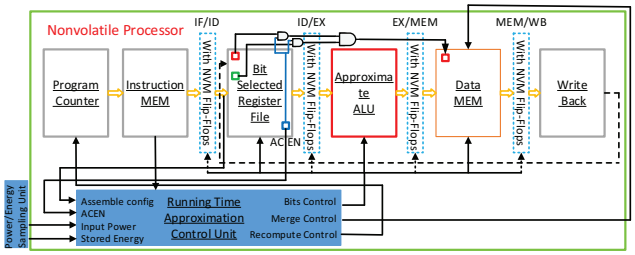

Figure 6: NVP execution approximation scheme. Global AC enable bit "AC EN" can be configured via writing to specific register.

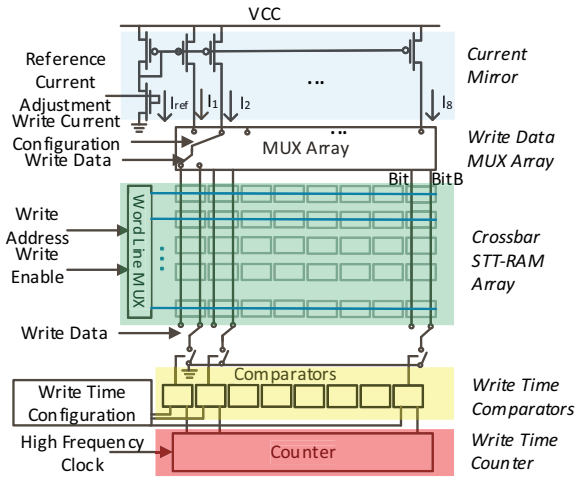

Figure 7: Proposed dynamic retention control scheme

by the programmer. No further execution can occur until the controller has completed the combination operation, using a state machine to iterate over the specified memory region one pair of memory values at a time. The combination options supported by our memory are max-precision (metadata max), max-value, min-value, and sum.

Hardware Support for Incidental Backup. Considering that the write time and write current both affect retention time, the NVM write circuits can be redesigned as shown in Figure 7. As shown in the cross bar STT-RAM array part (green background color), one STT-RAM has three nodes, "Bit", "BitB" and "Write Enable" signals for one bit cell. The write data can be changed through flipping the current direction of "Bit" and "BitB", under the control of the "Write Enable" transistor. The big idea of implementing such write operation supporting dynamic retention time is to apply write current control on one line in either "Bit" or "BitB", and to use the other line to control the write duration time. Similar retention time trade-offs can also be observed from ReRAM, PCRAM[42, 72], and FeRAM[56], and our dynamic retention time control scheme can be extended to these devices.

The write current for different retention time is generated by a current mirror, shown in blue background in Figure 7. I_{ref} is a baseline current. Different write currents, from I_1 to I_8, can be generated with little overhead by provisioning multiple output current mirror circuits with different W/L ratio of PMOS transistors, because the maximum current variation ratio is less than 3X from 1 day to 10ms, requiring only a small number of variant W/L ratios. The STT-RAM process variation[76] can be adjusted with the "Current Adjustment" signal during test in fabrication. Different currents can be selected in the MUX array according to different configurations (Log, linear etc). The write current is connected to either "Bit" or "BitB", depending on "Write Data".

The other line of "Bit" or "BitB" controls the write time. A high frequency 4-bit counter (sub ns per cycle) is implemented, and a comparator for each column of cells is designed to compare the counted time with the pre-set threshold stored in the nonvolatile "Write Time Configuration" module. Once the counter time reaches the threshold, the write operation is terminated by breaking the connection to GND.

The overheads for such a write module include 2-3X larger area for the current mirror, tens of transistors in MUX array, a 4-bit counter, and 8 comparators. The total overhead is less than 200 transistors per STT-RAM sub-array. We have implemented a behavior-level model of this module in RTL.

5 SOFTWARE SUPPORT

Pragma support: In order to support incidental computing, four pragmas are provided, as listed in Table1:

- incidental (src, minbits, maxbits, policy). This pragma indicates the allowed range of bitwise precision (with which a variable can be approximated) as well as its storage approximation policy. The first line in Figure 8 indicates that variable "src" can be approximated within a range of [minbits, maxbits] and the retention time policy is "linear".
- incidental_recover_from (variable). This pragma is used only with induction variables in looping constructs. The example line 4 in Figure 8 means that variable "frame" is used to indicate a fixed recovery restart point. Instead of recovering fromthe last instruction, the NVP skips over the remainder of a partially completed loop iteration to begin a new iteration.
- recompute(buf, minbits). Once some data are found to be "interesting", and we want to perform a recomputation to further improve the output quality with "minbits". When we obtain the new results, we use the the next pragma to merge the new results with the previous ones.
- assemble(buf, assemble_mode). This merges the new data with the previous, using one of the following strategies: sum, max, min, or higherbits. The "higherbits" option means that the results computed with higher bits cover the results of the lower bits.

Programmer's role: The programmer can use our pragmas to provide the compiler with three pieces of information: (i) to guarantee a minimum output quality, the programmer needs to mark the data that can be approximated, and how they can be approximated; (ii) the programmer also needs to indicate where the program should recover from; and finally (iii) in cases where some data are deemed "interesting" (as opposed

Semantics - begins with "#pragma ac"	Explanations
incidental (src, minbits, maxbits, policy)	variable "src" can be [minbits, maxbits] dynamically, and retention "policy"
incidental_recover_from (variable)	indicate a fixed recovery restart point
recompute(buf, minbits)	used to force minimum bits during recomputation
assemble(buf, assemble_mode)	merge buf with previsous ones with mode (sum, max, min, and higherbits)

Table 1: Semantics for supporting incidental computing

to "incidental"), the programmer needs to indicate how re-computation will be performed and how the results of this recomputation will be combined with old data to generate results with higher precision.

Compiler's role: The compiler uses the parameters provided by these pragmas to set some of the bits/values in our architecture: (i) the "AC" bit for variables like "src"; (ii) a recovery program counter; (iii) key variables within a loop, like "n" in the code example in Figure 8, in order to accurately match the break points; (iv) if the target code cannot be incidentally SIMDized, it is replicated multiple times, each with different inputs; and (v) we generate an instruction that configures a register to trigger hardware-based data merging.

Our current implementation does not support incidental SIMD optimizations for programs with loop-carried dependencies, although individual variable bitwidth approximations can still be used. If there are library calls in the code, the corresponding library routines need to be optimized and recompiled or the pragmas will be ignored by the compiler due to scoping and precise execution will be employed.

Note that, while prior research [7] proposed various directives for approximate computing, some of the actions taken by our compiler when processing our directives are entirely different from prior approaches, as we target an NVP based environment. Other approximation techniques [7, 43, 50] target "active" approximation, while our approach is "passive" due to limited energy in energy harvesting scenarios. That is, in our model, approximation is exogenously induced by insufficient power on a computation that is precise both in default and preference.

6 PUTTING IT ALL TOGETHER

```
#pragma ac incidental (src,2,8,linear);          -----(a1)
#pragma ac incidental (src,6,8,linear);          -----(a2)
unsigned char src[RowSize][ColSize] = {99,105,114,x,x...x};
#pragma ac incidental_recover_from(frame);       ----- (b)
for (unsigned int frame=0; frame < 3000; frame ++)
  for(n=0;n<RowSize;n++) ...
```

Figure 8: Example program with annotations

We now go over an example program fragment (Figure 8) to show how pragmas are used and how they interact with the underlying hardware. This sample program implements a portion of the median kernel.

We mainly focus on how to setup the first two types of pragmas and parameters, since the re-computation pragma is easier to understand. The first line marked with (a1) indicates

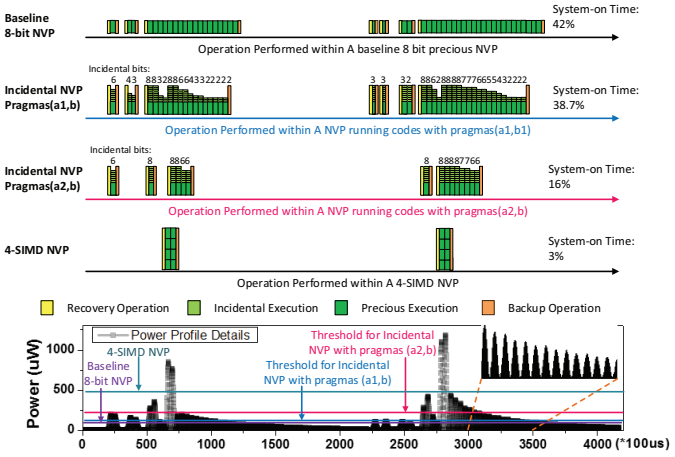

Figure 9: Timing based behavior analysis

that the programmer wants variable "src", which is a frame of data from an image sensor, to be incidentally computed between 2 bits and 8 bits, and the unreliable memory policy is linear. Line 4 marked with (b) indicates that the recovery program counter should be marked to the instruction that begins the update of the induction variable "frame".

The bottom part of Figure 9 shows a portion of the detailed power profile of power profile 2 in Figure 2. The grayscale power profile shows that, although there are some power spikes around 700*100us and 2700*100us, they are not "dark" (in color) enough (maintaining high power) to generate ample energy. The baseline 8-bit NVP has lowest threshold individually, leading to 42% system-on time. And the 4-SIMD has highest threshold, resulting in 3% system-on time (regarded as 3%*4=12%). Both of them can not achieve best forward progress. The system start threshold for incidental pragmas (a1,b) - [2 bits to 8 bits] is lower than that of (a2,b) - [6 bits to 8 bits]. The system performs recovery, computation or recomputation, and backup as shown in the top and middle portions of Figure 9. Due to different threshold levels, the NVP running pragmas (a2,b) has at least 6 bits, to guarantee a minimum output quality. However, the system is on in 16% of the total time period due to higher threshold. In comparison, the NVP running pragmas (a1,b) runs more instructions than all other solutions (although system-on is 38.7%, lower than 8-bit NVP 42%, but FP is 3.7X if incidental results are also considered, compared to 8-bit NVP), with more backup and recovery operations, and generates lower quality incidental outputs.

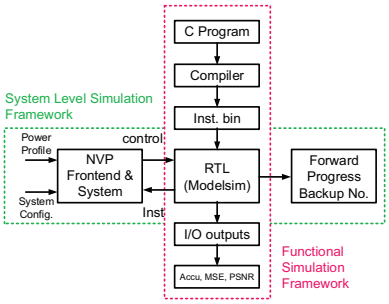

Figure 10: Our NVP framework with both functional and system-level simulation.

The programmer should set the minbits lower if the application is to be run faster, but with low quality incidental outputs. If, however, the low quality outputs turn out to be "interesting", the programmer may choose to recompute to obtain better quality outputs. Note that, if we look into the power profiles, we can see that in tiny scales (Figure 9 bottom right), there are still lots of glitches in the power, meaning different computation bitwidths for different elements in the "src" array. Note that this is more aggressive than the always-high-quality (a2,b) configuration.

For "incidental_recover_from", we suggest putting it near a buffer of data (e.g., before processing a whole frame). The programmer can also put it in inner loop, which can help to increase the output quality. Optimal placement also depends on the characteristics of the expected power profiles. More specifically, if the power profile is likely to have very frequent power interrupts (much shorter than the time required for processing a whole frame), putting the pragma in inner loop can help to improve the output quality. Note however that this happens only when the system is powered by WiFi or by a very quick vibration like 10kHz. For solar and thermal energy sources on the other hand, as long as the algorithm is not too complex, putting it near per "frame" is recommended.

For the recomputation, the programmer is in charge of checking whether there are low-quality outputs indicating that a higher precision might be beneficial (for instance, to reduce false positives in scenarios with asymmetric recall and precision impact from bitwidth reduction). If there are, he/she can use the recomputation pragmas to recompute and merge to improve the output quality.

7 SIMULATION AND VALIDATION

Our simulation framework consists of two parts as shown in Figure 10. The first part is a functional simulator, the core of which is a modified 8051 RTL [21], which we further modified with support for incidental computing logic and approximate memory. For framework compatibility, the inputs are generated as ROM arrays, and the outputs are generated through GPIO P2 and P3. We compiled the source code, and modified it to embed the "AC" bits. The RTL running in Modelsim initializes the ROM, RAM, etc. The quality analysis for image outputs is performed by computing PSNR and MSE in MATLAB.

The second part of our framework is a system-level simulator derived from the work by *Ma et al.* [30]. This system level simulation implemented in Matlab, and Python handles the system-level components including parameters and features of analog front-end circuits, capacitor etc., which cannot be implemented in RTL. The inputs to this simulator are the power profiles sampled every 0.1ms and the system configuration parameters such as the system capacitor size, capacitor leakage, chip leakage, front-end circuit efficiency, system start threshold, backup energy threshold, and recovery threshold. This system-level simulator controls the RTL simulator steps and gets the decoded instructions in order to decide various polices that dictate energy consumption. The system-level simulator, together with the functional simulator, generate important output metrics such as the *amount of forward progress* and the *number of backups*.

To test the effectiveness of our approach, we used several image processing and pattern matching kernels from MiBench [9] for the following reasons: Firstly, as discussed in Section 2.1, the computation in NVP dominates the energy consumption in the whole system for systems where sensing tasks utilize computationally intensive post-sensing algorithms, as are common in image processing and pattern matching. Secondly, these kernel are widely used in many post-sensor processing algorithms including gas sensing (spectrum analysis), water quality monitor (spectrum analysis and image processing), and power spectrum analysis of heart-rate variability. Finally, image processing kernels have also been employed on other successfully prototyped energy harvesting platforms in the literature [39, 41, 70].

To validate our choice of NVP execution rather than wait-compute for an energy-harvesting image-processing platform, we simulate frame rates for a volatile wait-compute platform (33uW power volatile MCU, the same model adapted in the NVP. Sensor power is not included) powered by the same "watch" energy harvester. For a 256*256 image size, similar to that used in other prototyped platforms [39, 41], susan.corners, susan.edges, and jpeg.encode are 1.65s, 4.9s, 12.55s per frame individually. An NVP solution without approximation [27] can improve this to 0.97s, 2.28s, 5.22s per frame individually with no quality loss. Incidental techniques can further improve to 0.3s, 0.59s, 1.2s per frame individually with minimum preset tolerable quality loss as shown in Table 2. Based on these observations, we believe that image/signal processing kernels running direclt on IoT devices can be an important workload in the energy-harvesting domain.

8 RESULTS AND DISSCUSSION

In this section, we first evaluate the impact of our individual approximation schemes on quality of output (adding noise or losing detailed information), and then quantify the forward

progress contributions due to individual techniques employed. Finally, we provide results from our holistic evaluation.

8.1 Bitwidth v.s. Quality

We use two key metrics to quantify the output quality compared against an 8-bit non-approximate baseline, namely, mean squared error (MSE) and peak signal-to-noise ratio (PSNR). To establish quality baselines, we first investigate MSE and PSNR for fixed-known-correct bit approaches, varying the bitwidth. This will allow us to ground our exploration of approximation in the NVP context from an output quality viability perspective. The N-bit reduced-quality ALU preserves the upper N bits and produces random outputs for the lower $8 - N$ bits, whereas the non-preserved bits in the reduced quality memory are truncated, and the operations using their values are treated as shifted N-bit operations. The approximate ALU models gradient VDD for different bits in ALU [8, 75], hence the addition of noisy bits rather than truncation. Figures 11 and 13 show the image output from three testbenches, namely, sobel, median, and integral, for ALU and memory bitwidth reductions, respectively.

We first consider ALU bit-quality reduction. As seen in Figure 12(a), the MSE for median and integral increases significantly when using less than 3 bits in the ALU. In contrast, for sobel, the MSE increases dramatically when there are fewer than 6 bits, indicating that sobel is not as amenable to fixed-width approximation as median. Figure 12(b) shows PSNR for reduced bit widths; traditionally, above 20-40 dB is considered a good PSNR response. For median and integral, even operating at a bitwidth of 1 can provide quality above 20 db, and 40 dB is achieved at 4-6 bits whereas sobel cannot achieve even 20 dB with anything less than full precision.

Bit reduction in memory produces somewhat distinct outputs from ALU precision reduction. As can be seen in Figure 14, the MSE measure of quality drops further than with ALU bit reduction. PSNR in Figure 14 (b) is similar to the approximate ALU solution. The PSNR metric is more similarly affected by either adding noise or losing detail compared to MSE, which is more sensitive to loss than noise.

8.2 Progress vs. Quality

Two other key evaluation parameters are forward progress (in the NVP sense of *persistent* forward progress), represented by number of instructions committed with a given power profile, and the number of backups. Figure 15 shows the forward progress achievable when the number of reliable bits in both the ALU and memory are reduced in tandem. By reducing the bits from full precision (8 bits in the 8051 NVP) to 1 bit, the forward progress doubles. The number of instructions executed increases for two reasons: not only is the power per operation reduced, but the lower power consumption and reduced local state to back up combine to trigger fewer power emergencies and to provide a lower activation threshold, leading to a higher duty cycle. As can be observed in Figure 16, the number of backups reduces by an average of 45% when the number of bits is reduced from 8 to 1.

8.3 Dynamic Bitwidth Approximation

In dynamic bitwidth, the computation of incidental approximation and memory bits changes along with the power profile. While this approach does not provide a bitwidth guarantee stronger than the above 1-bit solution, it will, in practice, execute at a much higher average bitwidth. Figure 18, shows the change in precision over time for three of our power profiles, and summarizes the distribution across bitwidths at the right of the figure. Figure 17 shows the output for the median testbench. Examining the MSE and PSNR in Figure 19 and the forward progress depicted in Figure 20 reveals that the execution quality of the dynamic bitwidth approximation is roughly comparable to a 2-bit solution, but the dynamic approach achieves an additional 20% forward progress over the similar quality fixed-bit approximation approach.

As previously mentioned, some kernels, such as the sobel testbench, are not as amenable to approximation as others. Rather than allowing dynamic bitwidth to be entirely determined by power profiles, it may sometimes be necessary to guarantee a higher minimum quality of results. For median, we find that the 4-bit-minimum-dynamic approach achieves similar MSE and PSNR results, across three power profiles, (MSE = 1.46, 1.72, and 1.72. PSNR = 46.5dB, 45.7dB, and 45.8dB) to a 7-bit fixed bitwidth solution, while achieving 22% more forward progress in Figure 21.

8.4 Backup and Recovery Approximation

The quality of outputs, both visually, as seen in Figure 26 left part, and by PSNR, in Figure 24, is similar for different retention time polices, although both retention time failures and MSE scores vary more widely. As seen in Figure 22, the retention failure counts for each bit vary significantly across both power profiles and policies, ranging from 15 to 1200 retention time violations. Surprisingly, the log retention policy has the best MSE (Figure 23), as well as the best PSNR (Figure 24), among the three policies tested. From this, we conclude that either the relatively low total number of bit errors on higher bits, while higher for the log policy, is still well within the tolerance of the approximable algorithms and the result reflects random variations in output quality (i.e. noise in the measurement of noise sensitivity), or some amount of noise is preferable for the applications examined. While the latter is possible, it seems the less likely of the two for the majority of the kernels examined in this work.

All the retention reduction policies reduce the backup energy requirements. Since the energy used to perform backups is reduced, it is available for increased computation, resulting in an average of 50% forward progress improvement, as shown in Figure 25. The log policy frees the greatest amount of energy and the parabola policy the least, with consistent trends in the forward progress benefits from these policies.

8.5 Recomputation

As previously shown, reducing the bitwidth in an NVP can improve forward progress, ***producing an earlier output at the cost of quality.*** For instance, as seen in Figure 15, an initial,

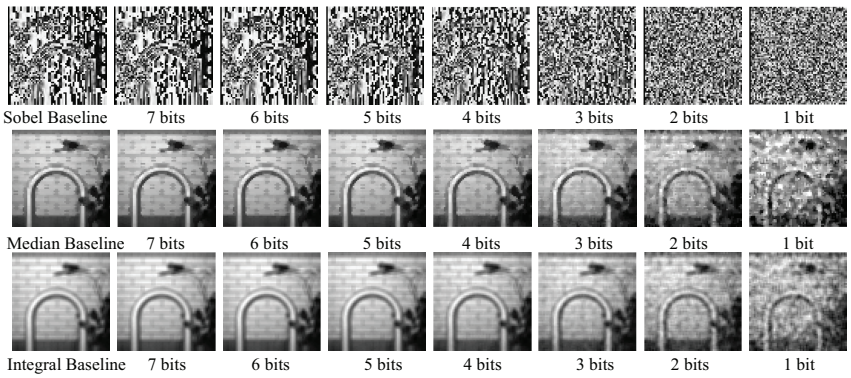

Figure 11: Impact of approximate ALU on image quality.

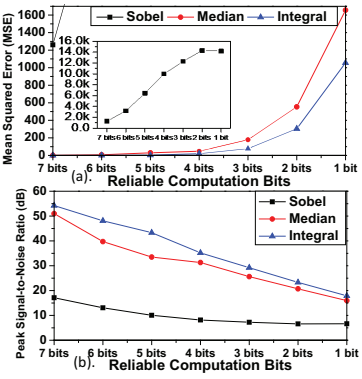

Figure 12: AC ALU MSE PSNR

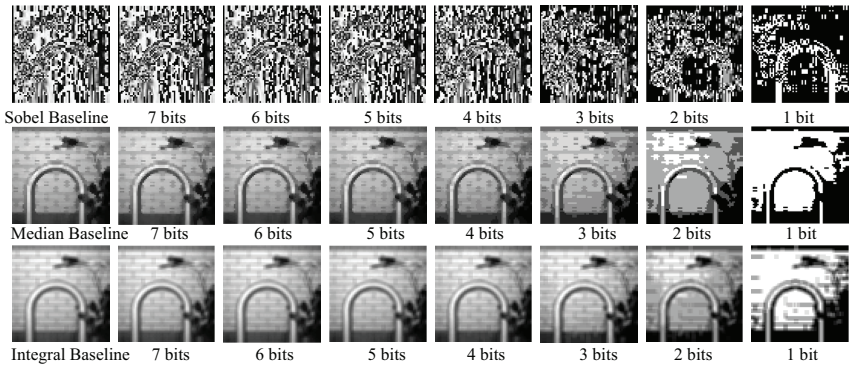

Figure 13: Impact of approximate memory on image quality.

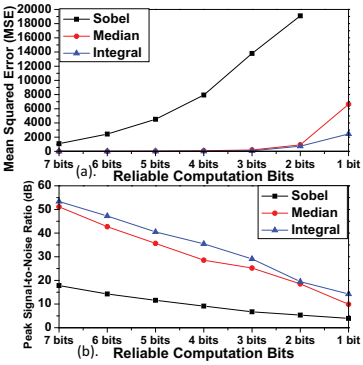

Figure 14: Unreliable Memory MSE PSNR

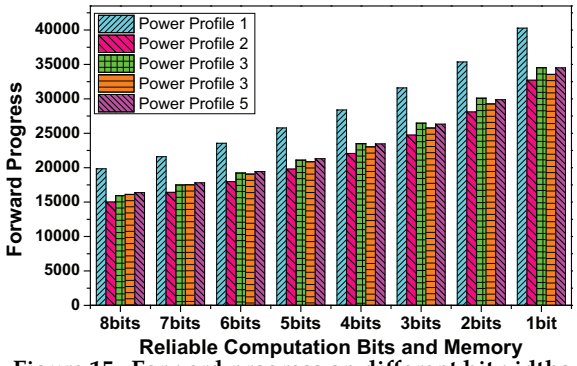

Figure 15: Forward progress on different bitwidths.

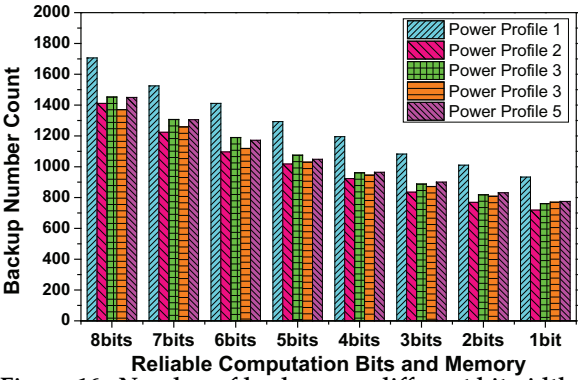

Figure 16: Number of backups on different bitwidths.

(a), Median with power profile 1 (b), Median with power profile 2 (c), Median with power profile 3

Figure 17: Impact of dynamic bitwidth on median.

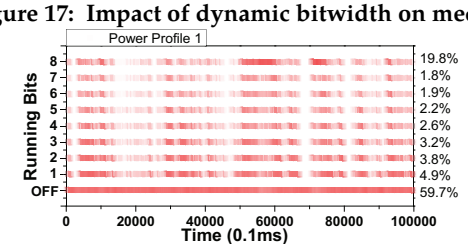

Figure 18: Utilization of bitwidthsof profile 1 for Median

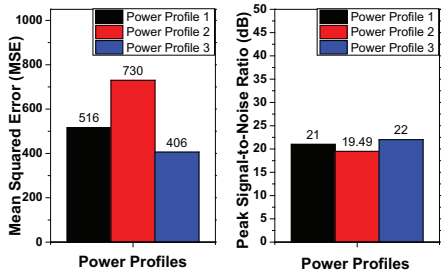

Figure 19: QoS of dynamic bitwidth on Median.

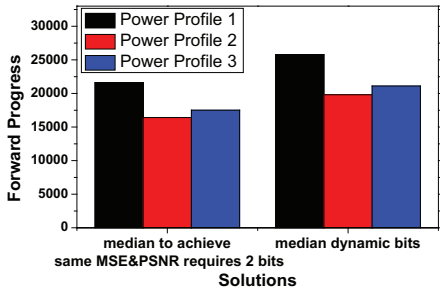

Figure 20: FP of dynamic bitwidth for Median.

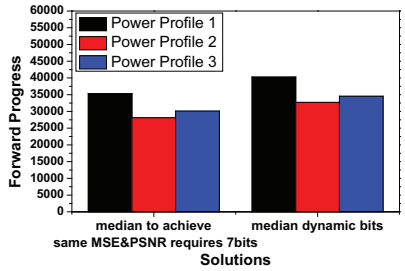

Figure 21: FP of the "MinBits=4" for Median.

4-bit result can be produced roughly 1.5x faster than an 8-bit result. For an application with a real-time deadline, any remaining slack time could be used to improve image quality, whereas, because of power-uncertainty, always waiting for the 8-bit full precision result will more frequently miss deadlines. For algorithms whose outputs are derived from highly independent computations, such as in many image processing kernels, recomputation can be used to replace output elements that were computed with low precision with higher precision recomputed outputs.

Below, we present an exploration of the potential benefits of the incidental recomputation using a *model* that always performs entire output passes with dynamic precision and then takes the highest precision output pixel from each and merges them. Figure 26 right part shows the outputs of recomputation with varying minimum bitwidths and Figure 27 shows the quality improvement as a function of the *additional recomputation passes.* Note that there is little value in recomputation beyond four to five passes. However, the approach employed is able to capitalize on random variation in the input power profile to perform iterative improvement.

8.6 Putting It All Together

Testbench	Target QoS / Achieved ?	MinBits	Recompute	Backup
Integral	PSNR 20dB / Yes	2 bits	No	Parabola
Median	PSNR 50dB / Yes	4 bits	2 times	Linear
Sobel	PSNR 8dB / Yes	4 bits	2 times	Linear
JPEG	150% Size / No: 3% unmet	3 bits	No	Log

Table 2: Targeting at QoS, fine-tuned incidental policies.

All these incidental techniques together can provide adjustable tradeoffs between QoS (quality of service) and forward progress, which provides programmers with a design space to play with through a debug-test-modify loop until the QoS reaches the minimum requirements (this is akin to experimenting with various parallelization options in OpenMP before picking up the right one). Table 2 shows an example of policies that we have fine-tuned to target QoS. For all testbenches except JPEG, we define our QoS target in terms of PSNR. For all the power profiles tested, the listed targets are always achieved except for the JPEG testbench. In the JPEG encoding testbench we apply incidental computing only on motion estimation, wherein approximation-induced error affects only the size of the compressed output. We define our QoS target for this kernel to be an output size that is no more than 50% larger than the full-precision compressed output. Across all power profiles, 97% of the 25000 compressed output frames from JPEG met QoS. For all kernels, incidental approximation is applied with full precision in the current iteration and dynamic bitwidth for incidental loop executions.

Based on our observation and experiences, the programmers should first decide the "minbits" to make the QoS above the QoS threshold, then reduce the "minbits", and try to fine-tune the incidental backup policy and the recompute times to compensate the QoS loss. We also suggest to employ linear incidental backup when average power is expected to be higher (e.g. scenarios akin to profiles 1, 4) and parabola when average power is low (e.g. profiles 2, 3, 5); preference for the logarithmic policy over linear/parabola is strongly kernel-specific. If the expected power characteristics are unknown, a lookup table or machine learning based mapping from the sampled power to configurations can be applied.

Figure 28 plots the gains of an incidental NVP with fine-tuned policies in Table 2. We observe a 4.3X improvement over precise NVPs [21]. Several factors contribute to the gains of incidental approximation: (1) omitting execution of some instructions, replaced with incidental computing, (2) dynamic approximation reduces power consumption, and (3) incidental computing provides the SIMD benefits of reduced instruction fetch energy. Overall, our gains vary substantially from testbench to testbench, due to, in large part, the different predefined pragma loop lengths. The variation among different power profiles is largely due to slight variation of the energy per instruction within these testbenches.

9 RELATED WORK

Energy harvesting is attractive for IoT. To support operation under an intermittent power supply, many works across the computation hierarchy have been explored, including OS and high-level synthesis approaches [36, 68], programming language and compiler approaches [6, 45], HW/ SW approaches [11], and software based approaches [2, 3, 67]. We classify these approaches as active checkpointing [35], in which the programmer needs to identify the essential data and software invokes a backup before energy runs out.

Another alternative is passive checkpointing. By designing distributed or centralized nonvolatile logic or storage elements

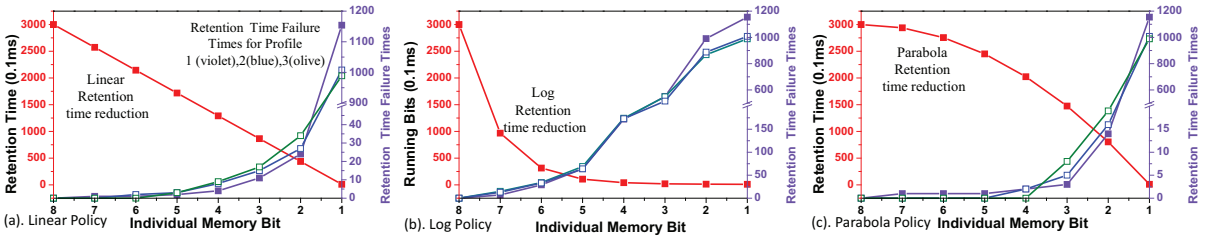

Figure 22: Backup, retention, and failure times with different bitwidths for (a) Linear, (b) Log, and (c) Parabola

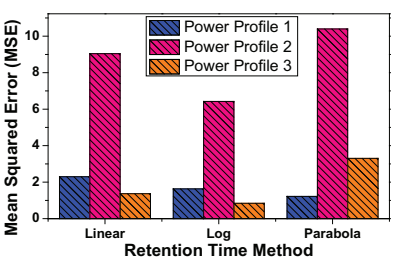

Figure 23: MSE v.s. retention.

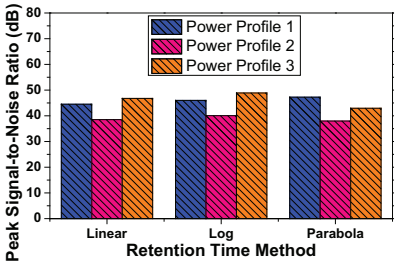

Figure 24: PSNR v.s. retention.

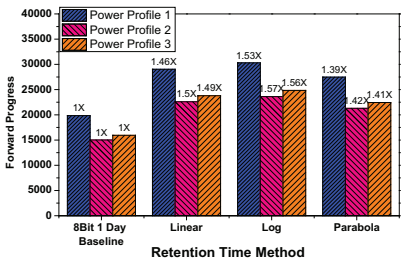

Figure 25: FP improvement.

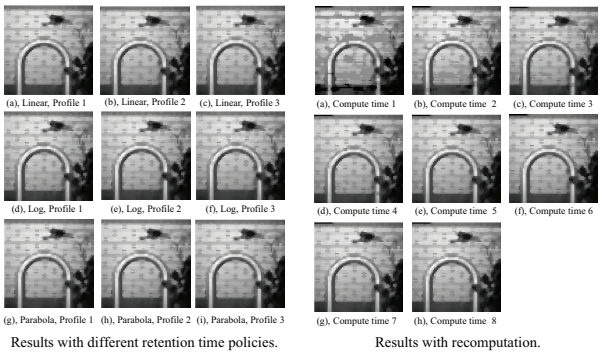

Figure 26: Visual results on different retention time policies(left), and recomputation(right).

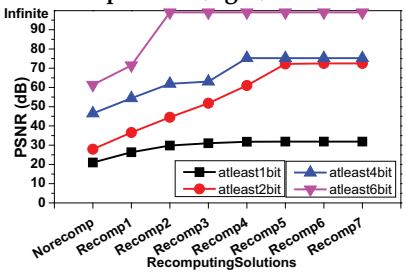

Figure 27: Impact of recomputation on quality.

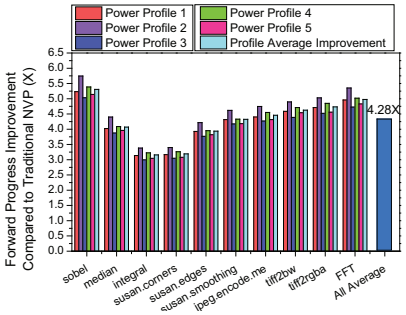

Figure 28: FP gain of incidental computing & backup

managed at a microarchitectural level, computation state can be checkpointed *transparently to software* before power outages using stored energy. This is called a nonvolatile processor [4, 23, 25, 27, 28, 30, 32, 33, 44, 46, 61]. Various materials and circuits with nonvolatility can be adopted to design different types of NVPs, e.g., FeRAM-based NVPs [13, 69], ReRAM-based NVPs[21], and MRAM-based NVPs[53, 54].

Many of the active checkpointing works [2, 3, 39, 41, 45, 67] are validated on a TI MSP430 with on-chip FeRAM, which can be considered an embryonic form of NVP, with software-based methods for active check-pointing. The different approaches for achieving continuous computation under unstable power supply have differing tradeoffs. The active method is modest in cost, but it is bounded by the backup speed and energy. Passive checkpointing can save system initialization time and energy when powered up, but is difficult to design and can potentially impose operational overheads.

In this work, rather than utilizing the programming language and compiler to achieve continuous computation, we use it for incidental computing, with continuity of computation fully offloaded to the NVP. By combining incidental computing and energy harvesting NVPs, we find optimization opportunities in both incidental computation and backup.

There is a substantial body of work focusing on approximate computing in the general purpose computing domain. A statistical guarantee method in controlling quality has recently been proposed for an approximate accelerator [31]. Quality detection and error correction by exact recomputing on host processor is proposed by Khudia *et al.* [14]. A pipeline-parallel approach for producing progressively higher quality output across multi-kernel execution chains via iterative recomputation is described by Miguel *et al.* [52]. Tang *et al.* improve bank-level parallelism for irregular applications [66]. A self-tuning approximation with quality feedback control for graphics engines is proposed by Samadi *et al.* [48]. Hardware support for

approximate operations includes voltage scaling and speculation [1] and multi-voltage setups [7].

Another form of approximation is approximate storage [26] . Approximation in DRAM based on data criticality is explored and optimized [20]. Approximate storage in solid-state memories is employed by Sampson *et al.* [51] and load-value approximation is developed by Miguel *et al.* [34]. Approximation is often a system-level approach, requiring support at multiple layers, cross-layer optimization and co-design. Recent work [62, 71] continue to examine compiler and programming level support for approximation, and a pure software based solution [47] targeting GPU approximation has been explored. An architecture using signal significance to vary approximation levels in an inter-frame motion estimator is presented [38]. Code acceleration with limited-precision analog computation is developed [59]. Configurable trade-offs between precision and energy are explored [37]. A "Rely" programming model for verifying unreliable hardware is developed [5], but random power failures are not modeled. Approximation in energy-harvesting is explored in software by Sampson *et al.* [49], but not targeted on nonvolatile processors.

The key point of divergence for our approach is optimizing approximate computing to a specific application scenario - energy harvesting - with the help of traditional NVPs to handle the unstable power supply. The application requirements of post-processing sensed data in real-time and locally, with limited harvested energy, challenges traditional NVPs. As a result, approximate computing alone cannot solve the problem because the newly-sensed data are still urgent to process, while historical buffered data's value drops over time. Observing this, our approach focuses on the incidental computing of historical buffered data, and proposes incidental re-computing to enhance the quality without affecting processing the newest data. Incidental computing offers appealing opportunities in the notion of gradient approximate backup and recovery, which tries to match the data importance and retention time to power outages. In combination, NVPs, approximation and incidental computing open new areas for optimizing energy harvesting IoT systems.

10 CONCLUSION

Technology trends leading to the proliferation of IoT devices operating on harvested energy demand a corresponding revolution of the abilities of processors to adapt to unstable power supplies. Adopting approximate computing approaches in NVPs not only improves their forward progress, but it also provides a means to optimize for responsiveness and efficiency and synergizes with unique features of NVPs, namely, frequent backup and recovery operations. We introduce the concept of "incidental computing" to address opportunistic responsiveness versus quality tradeoffs under unstable power income, and implement and evaluate an instantiation of the incidental computing approach based on memory and datapath approximation within an NVP. Overall, the incidental computing improves forward progress by an average of 4.28x over a baseline "precise" NVP, of which 1.4x is attributable to NVP-specific backup and restore approximations.

REFERENCES

[1] A. Bacha and R. Teodorescu, "Using ecc feedback to guide voltage speculation in low-voltage processors," *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 306–318, 2014.

[2] D. Balsamo, A. S. Weddell, A. Das, A. R. Arreola, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini, "Hibernus++: a self-calibrating and adaptive system for transiently-powered embedded devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 12, pp. 1968–1980, 2016.

[3] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, "Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 15–18, 2015.

[4] P. Bogdan, M. Pajic, P. P. Pande, and V. Raghunathan, "Making the internet-of-things a reality: From smart models, sensing and actuation to energy-efficient architectures," in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES '16. New York, NY, USA: ACM, 2016, pp. 25:1–25:10.

[5] M. Carbin, S. Misailovic, and M. C. Rinard, "Verifying quantitative reliability for programs that execute on unreliable hardware," in *ACM SIGPLAN Notices*, vol. 48, no. 10. ACM, 2013, pp. 33–52.

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

[7] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," *ACM SIGPLAN Notices*, vol. 47, no. 4, pp. 301–312, 2012.

[8] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: imprecise adders for low-power approximate computing," *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, pp. 409–414, 2011.

[9] 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," *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pp. 3–14, 2001.

[10] S. Jankowski, J. Covello, H. Bellini, J. Ritchie, and D. Costa, "The internet of things: Making sense of the next mega-trend," *IoT primer*, 2014.

[11] 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 and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference on*. IEEE, 2014, pp. 330–335.

[12] A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. R. Das, "Cache revive: architecting volatile stt-ram caches for enhanced performance in cmps," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 243–252.

[13] S. Khanna, S. C. Bartling, M. Clinton, S. Summerfelt, J. A. Rodriguez, and H. P. McAdams, "An fram-based nonvolatile logic mcu soc exhibiting 100digital state retention at vdd= 0 v achieving zero leakage with < 400-ns wakeup time for ulp applications," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 95–106, Jan 2014.

[14] D. S. Khudia, B. Zamirai, M. Samadi, and S. Mahlke, "Rumba: an online quality management system for approximate computing," *ACM SIGARCH Computer Architecture News*, 2015.

[15] V. Leonov, T. Torfs, P. Fiorini, and C. V. Hoof, "Thermoelectric converters of human warmth for self-powered wireless sensor nodes," *IEEE Sensors Journal*, vol. 7, no. 5, pp. 650–657, May 2007.

[16] X. Li, S. George, K. Ma, W. Y. Tsai, A. Aziz, J. Sampson, S. K. Gupta, M. F. Chang, Y. Liu, S. Datta, and V. Narayanan, "Advancing nonvolatile computing with nonvolatile ncfet latches and flip-flops," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. PP, no. 99, pp. 1–13, 2017.

[17] X. Li, K. Ma, S. George, W. S. Khwa, J. Sampson, S. Gupta, Y. Liu, M. F. Chang, S. Datta, and V. Narayanan, "Design of nonvolatile sram with ferroelectric fets for energy-efficient backup and restore," *IEEE Transactions on Electron Devices*, vol. 64, no. 7, pp. 3037–3040, July 2017.

[18] X. Li, J. Sampson, A. Khan, K. Ma, S. George, A. Aziz, S. K. Gupta, S. Salahuddin, M. F. Chang, S. Datta, and V. Narayanan, "Enabling energy-efficient nonvolatile computing with negative capacitance fet," *IEEE Transactions on Electron Devices*, vol. 64, no. 8, pp. 3452–3458, Aug 2017.

[19] X. Li, H. Liu, U. D. Heo, K. Ma, S. Datta, and V. Narayanan, "Rf-powered systems using steep-slope devices," *New Circuits and Systems Conference (NEWCAS)*, pp. 73–76, 2014.

[20] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: saving dram refresh-power through critical data partitioning," *ACM SIGPLAN Notices*, vol. 47, no. 4, pp. 213–224, 2012.

[21] Y. Liu, Z. Wang, A. Lee, F. Su, C. P. Lo, Z. Yuan, C. C. Lin, Q. Wei, Y. Wang, Y. C. King, C. J. Lin, P. Khalili, K. L. Wang, M. F. Chang, and H. Yang, "A 65nm reram-enabled nonvolatile processor with 6x reduction in restore time and 4x higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic," *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 84–86, Jan 2016.

[22] 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*, ser. PLDI '15, 2015, pp. 575–585.

[23] K. Ma, X. Li, S. Li, Y. Liu, J. J. Sampson, Y. Xie, and V. Narayanan, "Non-volatile processor architecture exploration for energy-harvesting applications," *IEEE Micro*, vol. 35, no. 5, pp. 32–40, 2015.

[24] K. Ma, X. Li, H. Liu, X. Sheng, Y. Wang, K. Swaminathan, Y. Liu, Y. Xie, J. Sampson, and V. Narayanan, "Dynamic power and energy management for energy harvesting nonvolatile processor systems," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 4, pp. 107:1–107:23, May 2017.

[25] K. Ma, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Dynamic machine learning based matching of nonvolatile processor microarchitecture to harvested energy profile," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 670–675, 2015.

[26] K. Ma, X. Li, J. Sampson, Y. Liu, Y. Xie, and V. Narayanan, "Nonvolatile processor optimization for ambient energy harvesting scenarios," *15th Non-volatile Memory Technology Symposium (NVMTS 2015)*, 2015.

[27] K. Ma, X. Li, S. R. Srinivasa, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Spendthrift: Machine learning based resource and frequency scaling for ambient energy harvesting nonvolatile processors," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, 2017, pp. 678–683.

[28] K. Ma, X. Li, K. Swaminathan, Y. Zheng, S. Li, Y. Liu, Y. Xie, J. J. M. Sampson, and V. Narayanan, "Nonvolatile processor architectures: Efficient, reliable progress with unstable power." *IEEE Micro*, vol. 36, no. 3, pp. 72–83, 2016.

[29] K. Ma, M. Liao, X. Li, Z. Huan, and J. Sampson, "Evaluating tradeoffs in granularity and overheads in supporting nonvolatile execution semantics," *ISQED*, 2017.

[30] K. Ma, Y. Zheng, S. Li, K. Swaminathan, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Architecture exploration for ambient energy harvesting nonvolatile processors," *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pp. 526–537, 2015.

[31] D. Mahajan, A. Yazdanbakhsh, J. Park, B. Thwaites, and H. Esmaeilzadeh, "Towards statistical guarantees in controlling quality tradeoffs for approximate acceleration," *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3, pp. 554–566, 2015.

[32] G. V. Merrett, "Invited: Energy harvesting and transient computing: A paradigm shift for embedded systems?" in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 1–2.

[33] G. V. Merrett and B. Al-Hashimi, "Energy-driven computing: Rethinking the design of energy harvesting systems," 2017.

[34] J. S. Miguel, M. Badr, and N. E. Jerger, "Load value approximation," *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 127–139, 2014.

[35] A. Mirhoseini, B. D. Rouhani, E. Songhori, and F. Koushanfar, "Chime: Checkpointing long computations on interm ittently energized iot devices," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 4, pp. 277–290, Oct 2016.

[36] A. Mirhoseini, E. M. Songhori, and F. Koushanfar, "Idetic: A high-level synthesis approach for enabling long computations on transiently-powered asics," in *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*. IEEE, 2013, pp. 216–224.

[37] N. Mishra, H. Zhang, J. D. Lafferty, and H. Hoffmann, "A probabilistic graphical model-based approach for minimizing energy under performance constraints," *ACM SIGPLAN Notices*, vol. 50, no. 4, pp. 267–281, 2015.

[38] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator," *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*, pp. 195–200, 2009.

[39] S. Naderiparizi, Z. Kapetanovic, and J. R. Smith, "Battery-free connected machine vision with wispcam," *GetMobile: Mobile Computing and Communications*, vol. 20, no. 1, pp. 10–13, 2016.

[40] S. Naderiparizi, Z. Kapetanovic, and J. R. Smith, "Rf-powered, backscatter-based cameras," in *Antennas and Propagation (EUCAP), 2017 11th European Conference on*. IEEE, 2017, pp. 346–349.

[41] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith, "Wispcam: A battery-free rfid camera," in *RFID (RFID), 2015 IEEE International Conference on*. IEEE, 2015, pp. 166–173.

[42] C. Pan, M. Xie, C. Yang, Y. Chen, and J. Hu, "Exploiting multiple write modes of non-volatile main memory in embedded systems," in *ACM Transactions on Embedded Computing Systems (TECS)*. ACM, 2017.2.

[43] J. Park, X. Zhang, K. Ni, H. Esmaeilzadeh, and M. Naik, "Expax: A framework for automating approximate programming," Georgia Institute of Technology, Tech. Rep., 2014.

[44] R. Perricone, I. Ahmed, Z. Liang, M. Mankalale, X. S. Hu, M. Kim, Chris H.and Niemier, S. Sapatnekar, and J.-P. Wang, "Advanced spintronic memory and logic for nonvolatile processors," in *International conference and exhibition for the design and engineering of systems-on-chip and embedded systems.*, Mar 2017.

[45] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on rfid-scale devices," *SIGARCH Comput. Archit. News*, vol. 39, no. 1, pp. 159–170, Mar. 2011.

[46] A. Rodriguez, D. Balsamo, A. Das, A. S. Weddell, D. Brunelli, B. Al-Hashimi, and G. V. Merrett, "Approaches to transient computing for energy harvesting systems: A quantitative evaluation," in *ENSsys 2015*, 2015.

[47] M. Samadi, D. A. Jamshidi, J. Lee, and S. Mahlke, "Paraprox: Pattern-based approximation for data parallel applications," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 35–50, 2014.

[48] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke, "Sage: Self-tuning approximation for graphics engines," *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 13–24, 2013.

[49] A. Sampson, A. Baixo, B. Ransford, T. Moreau, J. Yip, L. Ceze, and M. Oskin, "Accept: A programmer-guided compiler framework for practical approximate computing," *University of Washington Technical Report UW-CSE-15-01*, vol. 1, 2015.

[50] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: Approximate data types for safe and general low-power computation," in *ACM SIGPLAN Notices*, vol. 46, no. 6. ACM, 2011, pp. 164–174.

[51] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, "Approximate storage in solid-state memories," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 3, p. 9, 2014.

[52] J. San Miguel and N. E. Jerger, "The anytime automaton," *International Symposium on Computer Architecture*, 2016.

[53] S. Senni, L. Torres, A. Gamatié, and G. Sassatelli, "Non-volatile processor based on MRAM for ultra-low-power IoT devices," *ACM Journal of Emerging Technologies in Computing (JETC)*, vol. 00, no. 00, 2017.

[54] S. Senni, L. Torres, G. Sassatelli, and A. Gamatie, "Non-volatile processor based on mram for ultra-low-power iot devices," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 2, pp. 17:1–17:23, Dec. 2016.

[55] C.-X. G. seris datasheet, "https://www.cap-xx.com/resource/cap-xx-g-series-datasheets/."

[56] A. Sheikholeslami and P. G. Gulak, "A survey of circuit innovations in ferroelectric random-access memories," *Proceedings of the IEEE*, vol. 88, no. 5, pp. 667–689, 2000.

[57] X. Sheng, C. Wang, Y. Liu, H. G. Lee, N. Chang, and H. Yang, "A high-efficiency dual-channel photovoltaic power system for nonvolatile sensor nodes," in *Non-Volatile Memory Systems and Applications Symposium (NVMSA), 2014 IEEE*. IEEE, 2014, pp. 1–2.

[58] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*. IEEE, 2011, pp. 50–61.

[59] R. St Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger, "General-purpose code acceleration with limited-precision analog computation," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 505–516, 2014.

[60] F. Su, Y. Liu, Y. Wang, and H. Yang, "A ferroelectric nonvolatile processor with 46 us system-level wake-up time and 14 us sleep time for energy harvesting applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 3, pp. 596–607, March 2017.

[61] F. Su, K. Ma, X. Li, T. Wu, Y. Liu, and V. Narayanan, "Nonvolatile processors: Why is it trending?" in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 966–971.

[62] X. Sui, A. Lenharth, D. S. Fussell, and K. Pingali, "Proactive control of approximate programs," *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 607–621, 2016.

[63] K. Swaminathan, R. Pisolkar, C. Xu, and V. Narayanan, "When to forget: A system-level perspective on stt-rams," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*. IEEE, 2012, pp. 311–316.

[64] V. Talla, B. Kellogg, B. Ransford, S. Naderiparizi, S. Gollakota, and J. R. Smith, "Powering the next billion devices with wi-fi," *arXiv preprint arXiv:1505.06815*, 2015.

[65] Y. K. Tan and S. K. Panda, "Energy harvesting from hybrid indoor ambient light and thermal energy sources for enhanced performance of wireless

sensor nodes," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 9, pp. 4424–4435, Sept 2011.
[66] X. Tang, M. Kandemir, P. Yedlapalli, and J. Kotra, "Improving bank-level parallelism for irregular applications," in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*. IEEE, 2016, pp. 1–12.
[67] TI, "Ctpl "compute through power loss" software utility, https://e2e.ti.com/blogs_/b/msp430blog/archive/2015/05/29/what-is-compute-through-power-loss."
[68] J. Van Der Woude and M. Hicks, "Intermittent computation without hardware support or programmer intervention," in *Proceedings of OSDIâĂŹ16: 12th USENIX Symposium on Operating Systems Design and Implementation*, 2016, p. 17.
[69] 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 *ESSCIRC (ESSCIRC), 2012 Proceedings of the*. IEEE, 2012, pp. 149–152.
[70] A. S. Weddell, M. Magno, G. V. Merrett, D. Brunelli, B. M. Al-Hashimi, and L. Benini, "A survey of multi-source energy harvesting systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 905–908.
[71] D. Wong, N. S. Kim, and M. Annavaram, "Approximating warps with intra-warp operand value similarity," *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 176–187, 2016.
[72] Y. Xie, *Emerging Memory Technologies: Design, Architecture, and Applications*. Springer Science & Business Media, 2013.
[73] T. Xue, X. Ma, C. Rahn, and S. Roundy, "Analysis of upper bound power output for a wrist-worn rotational energy harvester from real-world measured inputs," *Journal of Physics: Conference Series*, vol. 557, no. 1, p. 012090, 2014.
[74] T. Xue and S. Roundy, "Analysis of Magnetic Plucking Configurations for Frequency Up-Converting Harvesters," *Journal of Physics Conference Series*, vol. 660, no. 1, p. 012098, Dec. 2015.
[75] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," *Proceedings of the International Conference on Computer-Aided Design*, pp. 48–54, 2013.
[76] W. Zhao, X. Zhao, B. Zhang, K. Cao, L. Wang, W. Kang, Q. Shi, M. Wang, Y. Zhang, Y. Wang *et al.*, "Failure analysis in magnetic tunnel junction nanopillar with interfacial perpendicular magnetic anisotropy," *Materials*, vol. 9, no. 1, p. 41, 2016.
[77] M. Zwerg, A. Baumann, R. Kuhn, M. Arnold, R. Nerlich, M. Herzog, R. Ledwa, C. Sichert, V. Rzehak, P. Thanigai, and B. O. Eversmann, "An 82 ua/mhz microcontroller with embedded feram for energy-harvesting applications," in *2011 IEEE International Solid-State Circuits Conference*, Feb 2011, pp. 334–336.