Energy-efficient and Reliable Inference in Nonvolatile Memory under Extreme Operating Conditions

SALONIK RESCH, S. KAREN KHATAMIFARD, ZAMSHED I. CHOWDHURY, MASOUD ZABIHI, ZHENGYANG ZHAO, HUSREV CILASUN, JIAN-PING WANG, SACHIN S. SAPATNEKAR, and ULYA R. KARPUZCU, University of Minnesota, Twin Cities, USA

Beyond-edge devices can operate outside the reach of the power grid and without batteries. Such devices can be deployed in large numbers in regions that are difficult to access. Using machine learning, these devices can solve complex problems and relay valuable information back to a host. Many such devices deployed in low Earth orbit can even be used as nanosatellites. Due to the harsh and unpredictable nature of the environment, these devices must be highly energy-efficient, be capable of operating intermittently over a wide temperature range, and be tolerant of radiation. Here, we propose a non-volatile processing-in-memory architecture that is extremely energy-efficient, supports minimal overhead checkpointing for intermittent computing, can operate in a wide range of temperatures, and has a natural resilience to radiation.

CCS Concepts: • **Computer systems organization → Architectures;**

Additional Key Words and Phrases: Processing in memory, beyond-edge computing

ACM Reference format:
Salonik Resch, S. Karen Khatamifard, Zamshed I. Chowdhury, Masoud Zabihi, Zhengyang Zhao, Husrev Cilasun, Jian-Ping Wang, Sachin S. Sapatnekar, and Ulya R. Karpuzcu. 2022. Energy-efficient and Reliable Inference in Nonvolatile Memory under Extreme Operating Conditions. *ACM Trans. Embedd. Comput. Syst.* 21, 5, Article 57 (December 2022), 36 pages.
https://doi.org/10.1145/3520130

1 INTRODUCTION

Beyond-edge devices collect energy from the environment, allowing them to operate off the grid and without a battery [13, 63]. This enables them to function in environments that were previously considered as impossible, such as in the remote wilderness [91], within the human body [41], and out in space [82]. This capability opens up many opportunities for new applications. Running machine learning algorithms on beyond-edge devices is particularly attractive due its versatility [39]. Utilizing **neural networks (NN)** or **support vector machines (SVM)**, a wide variety of problems can be solved.

However, engineering devices to operate beyond the edge is difficult. As they must collect energy from their environment, the power source by construction is unreliable. The devices must

Authors' address: S. Resch, S. K. Khatamifard, Z. I. Chowdhury, M. Zabihi, Z. Zhao, H. Cilasun, J.-P. Wang, S. S. Sapatnekar, and U. R. Karpuzcu, University of Minnesota, Twin Cities, 200 Union Street SE, Minneapolis, Minnesota, USA, 55455; emails: {resc0059, khatami}@umn.edu.

1539-9087/2022/12-ART57 $15.00
https://doi.org/10.1145/3520130

frequently power off, turning back on when energy is available. This is referred to as *intermittent* computing, which comes with performance and energy efficiency overheads. To prevent total loss of information during intermittent operation, beyond-edge devices must do additional work in three categories [34]: (1) *Backup* refers to saving data and the current architectural state (which often entails writes to non-volatile memory); (2) *Dead* corresponds to re-performing work that could not be saved on the last shutdown; and (3) *Restore* encapsulates all work associated with re-starting the device after a shutdown. Beyond the additional latency and energy overheads, intermittency also makes it a challenge to guarantee correctness of a program. Power interuptions can introduce memory inconsistencies, which can easily lead to incorrect operation [19, 82]. For conventional embedded systems, sophisticated software strategies are required to ensure that an interruption at any point during operation does not induce corruption [39, 110].

Previous work has shown that non-volatile **processing-in-memory (PIM)** architectures are promising for use in beyond-edge devices. As a representative example, MOUSE [107] is a PIM architecture that delivers high performance and extreme energy efficiency using low complexity checkpointing mechanisms. It has three advantages over traditional architectures:

(1) Inherently intermittent safe logic operations;
(2) Automatic and instantaneous data backup;
(3) Highly energy-efficient and highly parallel operations.

Advantage (1) enables **minimal overhead accelerator utilizing spintronic RAM for energy-harvesting applications (MOUSE)** to simplify its checkpointing strategies. Operations performed in the memory can be interrupted or performed multiple times without introducing corruption. Hence, data remains consistent as long as operations are performed sequentially. Advantage (2) comes from processing in non-volatile memory, and directly reduces the overhead for checkpointing. Typically, a device has to write volatile data back to memory to save progress before shutdown. Since MOUSE does all its computation in non-volatile memory, progress is saved automatically after every operation. Finally, Advantage (3) enables high performance within low power budgets.

Recently, there has been much excitement about the use of beyond-edge devices as nanosatellites [83] deployed in **low Earth orbit (LEO)**. Such devices can provide valuable services such as security along with agricultural [137], environmental or structural monitoring [83]. Nanosatellites can be much more cost effective than traditional monolithic satellites. However, orbital deployment for use as a satellite further challenges engineering such devices. For example, the cost of communication now becomes much greater than the cost of computation (even more so than for terrestrial deployment) [39, 83]. This shifts emphasis towards performing more computation and holding more data on the device, and away from frequent communication [27]. MOUSE is well-suited for this challenge as it has a large memory capacity (due to consisting nearly entirely of high density non-volatile memory), enabling it to potentially go long periods of time and store many results before data transmission becomes necessary.

An additional challenge for beyond-edge devices deployed in LEO is that they must operate in a wide range of temperatures. Satellites can get both very cold (−170°C) and very hot (123°C) [74]. Large scale satellites can be engineered to perform temperature modulation [9]. However, small, cheap beyond-edge devices can typically not use such strategies. Fortunately, **complementary metal-oxide semiconductors (CMOS)** can perform well across this wide temperature range, and the performance of CMOS circuits actually tends to increase with decreasing temperature [124, 147]. However, cold operation can have an adverse effect on non-volatile memory, where the energy efficiency degrades [52, 71, 106, 146, 150].

Another complication of orbital deployment is radiation. Even in terrestrial deployment, radiation can induce soft errors in CMOS circuits [10], potentially corrupting the architectural state. Without the Earth's atmosphere to shield radiation from space, satellites are exposed to much higher levels of radiation. Non-volatile memory is at an advantage in this domain, as the memory devices it uses are highly resistant to radiation [94]. However, non-volatile memory still relies on CMOS circuitry for memory access and external control, which also applies to non-volatile PIM. Circuit level strategies can be used to mitigate the impact of radiation on CMOS hardware [118], which can incur significant power, latency, and area overheads.

In this work, we extend the design of MOUSE [107] to be suitable for orbital deployment. We demonstrate that MOUSE can operate over a wide temperature range (despite non-ideal impacts on non-volatile memory) and evaluate its performance at the extremes. MOUSE has an inherent resilience to radiation due to ideal properties of the non-volatile memory it uses [37, 65], but it still requires CMOS circuitry to drive operations. We show that even in the presence of the overhead for the hardening of CMOS circuitry to radiation [157], MOUSE remains highly energy-efficient and performant. We also extend the MOUSE PIM instruction set [107] and add architectural support for branch instructions, which increases the programmability of the device. Finally, we introduce more hardware-efficient column activation mechanisms for enabling logic in the memory. The result is a programmable, high performance, and extremely energy-efficient beyond-edge device that is suitable for deployment in space. In summary, using MOUSE [107] as a representative case study, our contributions are as follows:

(1) Demonstration that the non-volatile in-memory logic works under a wide operating temperature range and evaluation of the impact on performance.
(2) Evaluation of the overhead in adapting all PIM circuitry to withstand high radiation.
(3) Extension of the MOUSE instruction set for enhanced programmability.
(4) Addition of more efficient hardware mechanisms for enabling logic in the memory.

The rest of the article is as follows. In Section 2, we provide a background on non-volatile processing-in-memory. In Section 3, we detail the architecture and cover how it maintains correctness in Section 4. We discuss additional challenges of orbital deployment in Section 5. The evaluation is set up in Section 6, and the results are reported in Section 7. Finally, related work is discussed in Section 8, and we conclude the article in Section 9.

2 NON-VOLATILE PIM

Any non-volatile memory technology can be used as a PIM substrate, including RRAM [142] and PC-RAM [73]. In this work, we focus on **Magnetoresistive RAM (MRAM)** [97, 139], which features both high density and high endurance. Due to its nearly ideal properties, MRAM can even be considered as a universal memory replacement [29] and a few commercial products are already available [1, 2]. **Spin-torque transfer (STT)** MRAM uses the **magnetic tunnel junction (MTJ)** as its memory element and each memory cell contains one MTJ and one access transistor.

STT-MRAM arrays can be minimally modified to enable PIM. An example is **Computational RAM (CRAM)** [17], which MOUSE [107] is based on. It has a unique advantage in that it does not require sense amplifiers or any digital circuitry in the array periphery to perform computation. The structure of CRAM supports Boolean logic operations directly within the memory, simply by adjusting voltage along the bitlines (to logic function specific levels). The computation remains *entirely* inside the array at all times. In this work, use two variants of CRAM, an optimized version of the standard STT [108] and an extension that increases energy efficiency with a **spin-Hall effect (SHE)** channel [153].

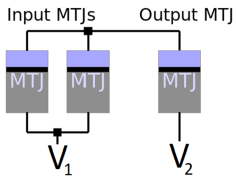

Fig. 1. MTJs connected for a two-input logic gate. Fixed (free) layer is colored in grey (light blue).

2.1 Magnetic Tunnel Junctions

MTJs are the resistive memory elements used by STT-MRAM. MTJs consist of two magnetic layers (a fixed layer and a free layer) separated by a thin insulator. The polarity of the free layer can change, but the fixed cannot. If the two magnetic layers are aligned (referred to as the **parallel (P)** state), then the MTJ has low resistance and is assigned the logic value 0. When the layers are not aligned (the **anti-parallel (AP)** state), the MTJ has high resistance and is assigned logic value 1. The MTJ changes state if a (relatively) large amount of current is passed through it. The state it transitions to is determined by the direction of the current. If electrons flow from the free (fixed) layer to the fixed (free) layer, then the MTJ switches to the AP (P) state. The current required to change the state is referred to as I_{switch}. A current greater than or equal to I_{switch} induces switching and a current below I_{switch} leaves the state as is. The voltage required to induce switching can be referred to as V_{switch}, and is determined by $V_{switch} \geq I_{switch} \times R_{MTJ}$, where R_{MTJ} is the resistance of the MTJ. Hence, V_{switch} also depends on the state (P or AP) of the MTJ. As will be discussed in Section 5.1, the operating temperature has a significant impact on this voltage.

To read the value of the MTJ, a voltage below V_{switch} (to avoid switching) is applied across it. The amount of current that passes through it is a function of its resistance (state), which gets detected by a sense amplifier. To write, i.e., change the state of the MTJ, a voltage higher than V_{switch} is applied across it. This induces a sufficient amount of current to change the MTJ state.

2.2 Logic Operations with MTJs

MTJs can be used to perform logic operations if they are connected together in a circuit. An example of a circuit that can perform a two-input logic operation is shown in Figure 1. Two input MTJs are connected in parallel, which are in series with an output MTJ. Before performing the logic gate, the two inputs can be in any state. However, the output MTJ must be preset to a known value. After the logic gate is performed, the state of the output MTJ changes as a function of the two input MTJs, following the truth table of the corresponding logic gate.

For example, a NAND gate requires the output to be preset to 0 (low resistance). A voltage is then applied across the terminals V_1 and V_2, such that electrons flow from the input MTJs to the output MTJ. If both inputs are 1 (high resistance), then the current is sufficiently low to prevent switching of the output MTJ, which will remain at 0. If either of the inputs is 0 (low resistance), then there is enough current to induce switching of the output MTJ. As electrons are flowing from the free layer to the fixed layer of the output MTJ, it can only switch to 1.[1] Hence, the output MTJ reflects the logical NAND of the two inputs; 0 if both are 1, and 1 if either is 0.

Different gates, such as NOT, AND, and N(OR) can be performed by changing the number of inputs, the direction of the current, and the preset of the output. Sequences of these gates can be used to perform more complex operations. For example, a full-adder can be implemented using 9 NAND gates. Multi-bits additions and multiplications can be performed using sequences of full-additions. As the gate set is universal, any computation can be implemented.

[1]Due to the direction of the current, the output MTJ can only switch to 1 and cannot switch back to 0.

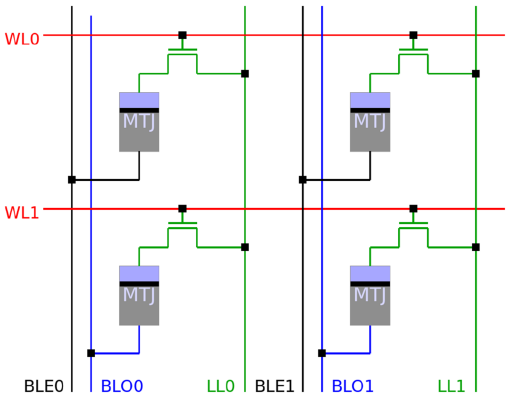

Fig. 2. Four cells in two columns and two rows in 1T1M (one access transistor, one MTJ) STT configuration.

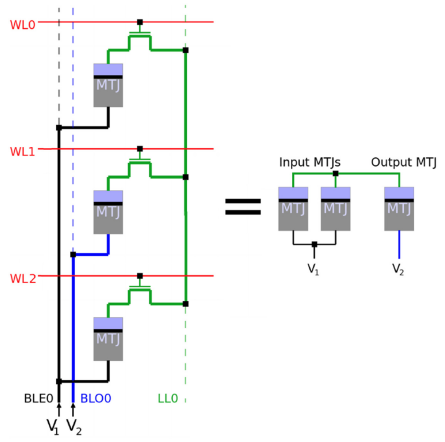

Fig. 3. Demonstration of two-input gate performed within the array.

2.3 STT Array Architecture

The optimized STT variant of CRAM is nearly identical to a standard STT-MRAM array [108]. Four cells located in two adjacent rows and columns are shown in Figure 2. It has one MTJ and one access transistor in each cell. A standard STT-MRAM array contains two bitlines (typically referred to as bitline and bitline bar). CRAM has three bitlines per column, **bit line even (BLE)**, **bit line odd (BLO)**, and the **logic line (LL)**. BLE connects to even rows and BLO connects to odd rows (where even or odd refer to the parity of the row number). LL connects to all rows through the access transistors. The existence of three bitlines is essential for enabling computation, which is explained below. As with standard STT-MRAM, there is a **wordline (WL)** in each row that controls the access transistor. We now describe how memory and logic operations are performed in the array.

Read: To read from row n, activate WLn. Apply a voltage differential, V_{read}, across LL and the BLE/BLO. Current can be sensed on the bitlines. V_{read} should be lower than V_{switch}.

Write: To write to row n, activate WLn. Apply a voltage differential, V_{write}, across LL and the BLE/BLO. To write 0 (1), the voltage on BLE/BLO should be higher (lower) than on LL. V_{write} should be larger than V_{switch}.

Logic Operation: To perform a logic gate with two inputs in rows n_1 and n_2, and the output in row m, preset row m by performing a write.[2] n_1 and n_2 must have the same parity (i.e., both even or both odd), and m, the opposite. Activate WLn_1, WLn_2, and WLm. Apply a voltage differential, V_{logic}, across BLE and BLO. Due to the parity requirement, in Figure 1, if V_1 is connected to BLE, V_2 must be connected to BLO, and vice versa. LL connects the free layers (in light blue) of the input and the output MTJs. Current travels from one bit line (either BLO or BLE, depending on the parity of the input cells), through the MTJs in rows n_1 and n_2, through the LL, through the MTJ in row m, and back to the other bitline. Depending on the states of the MTJs in rows n_1 and n_2, the state of the MTJ in row m will either change or not. Figure 3 shows how a NAND gate can be performed inside the array. V_{logic} must be within a specified range for each type of logic operation [108, 152].

Only one operation (read, write, or logic) can be performed in each column at a time. However, operations can proceed in many columns simultaneously. The restriction is that (within a single

[2]This write needs only to be performed if the initial state is different than the corresponding preset value.

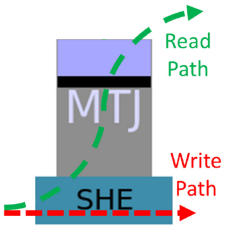

Fig. 4. An MTJ integrated with a spin-Hall effect (SHE) channel has separate read and write paths.

array) it should be the same operation (i.e., type of logic gate) on the same row designation for inputs and outputs. For example, a NAND gate can be performed in all columns with the inputs in rows n_1 and n_2 and the output in row m.

It may be desirable to perform computation in all columns, or in just a subset of columns. The peripheral circuitry determines which columns participate in every operation. The mechanism for activating columns is covered in Section 3.3, and the instructions that control column activation are covered in Section 3.4.

Effectively, each column acts as an independent thread that has access to the memory cells within the column. This is highly analogous to the SIMD lanes of a GPU architecture, where each lane (column) performs the same operation on different data. The *active* columns act like the bit-mask in a GPU, where only the active subset columns perform the operation. However, each column can only perform Boolean logic gates (whereas a GPU has full access to an ALU). Complex arithmetic/logic translates into performing a sequence of Boolean gates in each column, where each gate operation can proceed in parallel, in a lock-step fashion. Hence, computations in each column are relatively slow, but performance is achieved via a high degree of parallelism.

2.4 SHE Array Architecture

The energy efficiency of MTJ write and logic can be significantly improved by utilizing a SHE channel [153]. SHE channels are compatible with both CMOS and MTJs and prototypes have been successfully demonstrated [36]. Proposed memory technology based on this same technology is called **spin-orbit torque (SOT)** MRAM [36, 97].

The SHE channel provides a more efficient mechanism for switching the state of the MTJ. An MTJ augmented by a SHE channel is shown in Figure 4. There are two current paths through the device. For the write path, current passes only through the SHE channel. Despite not travelling through the MTJ body, this current can change the MTJ state. As a result, a lower current density is required and the voltage V_{write} can be lowered. This benefit also extends to logic operations and V_{logic} However, the read path is still through the MTJ body, and read operations remain the same.

Four augmented cells in two rows and two columns are shown in Figure 5. For the SHE design, there are two word lines per row, **word line for read (WLR)** and **word line for write (WLW)**. WLR connects the cell to the read path, via t_{read}. WLW connects the cell to the write path, via t_{write}. t_{write} connects the SHE channel directly to LL, allowing current to bypass the MTJ body. t_{write} is activated when the memory cell is being written, or when it is the output of a logic operation. t_{read} connects one end of the MTJ to LL, causing current to travel through the MTJ body. t_{read} is activated when the memory cell is being read, or when it is an input of a logic operation.

The energy efficiency provided by the SHE channel is highly beneficial for beyond-edge devices. Reducing the amount of energy per operation can reduce the total execution time (if the device is limited by the available energy), as will be shown in the evaluation. Beyond energy efficiency, the SHE channel also enables more robust logic operations. When performing logic with STT, the

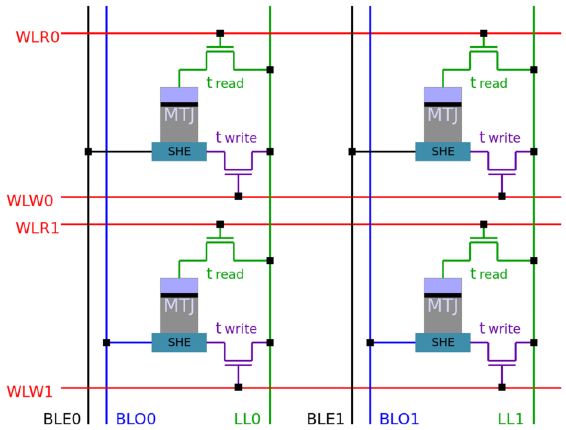

Fig. 5. Four cells in two columns and two rows in 2T1M SHE configuration.

resistance of the output MTJ is in series with the input MTJs (Figure 1). With SHE, only the SHE channel is in series with the input MTJs. This makes it easier to distinguish between the resistances of the input MTJs, reducing susceptibility to voltage fluctuations [153].

2.5 Memory-centric Architectures

Over the past few decades, CPUs have achieved larger speedups than memory technologies [44]. Consequently, memory has become the limiting factor for performance and energy efficiency. This has been referred to as the *memory wall*, and has led to the development of technologies and architectures performing computation closer to the memory to avoid data transfer bottlenecks.

Memory-array-based computing has been integrated into a number of accelerators, where it is used in tandem with external dedicated hardware [123, 128, 140, 143]. In this type of architecture, the memory array acts like a hybrid of scratchpad memory and logical units, rather than a traditional memory structure. A common approach is to use a cross-bar structure. Inputs are supplied as voltages along the rows and results are processed by sense amplifiers on the columns, before being passed to digital circuitry for non-linear operations [16].

***In-storage processing* (ISP)** attempts to alleviate the memory wall for larger-scale computing (e.g., in data centers) by integrating logic cores closer to the memory. These systems typically deploy small processors or FPGAs near memory arrays, potentially on the same chip [58, 60, 62, 109, 131, 132].

Processing Near Memory has more fine-grained integration of computation and memory. It maintains standard memory structures but moves logic units to the array periphery [125]. In this approach, data is loaded from memory, computed on by nearby digital circuitry, and then stored back into the array. Examples include the Hybrid Memory Cube [100] and TESSERACT [3].

PIM represents the extreme, where computation occurs in the memory itself. PIM technologies include Pinatubo [76], which uses non-volatile memory, and Ambit [122], which uses DRAM. Both of these require sense amplifiers to perform computation. Multiple rows in the memory are read simultaneously. The sense amplifiers perform computation by discerning between different possible analog values. The results are then immediately written back. In contrast, the PIM architecture we use, CRAM [17, 108, 151], performs the logic directly in the memory cells. As the data does not have to be pulled to the array periphery, this has been referred to as *true* in-memory computing [151].

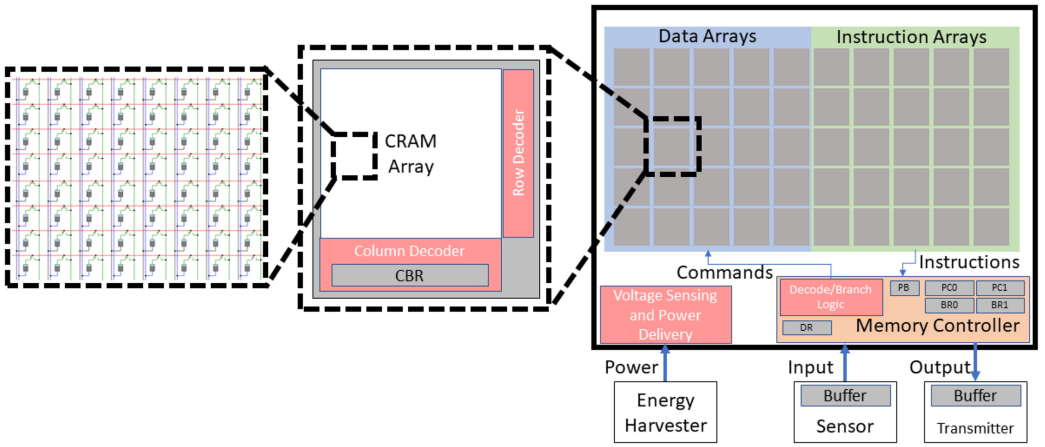

Fig. 6. Overview of MOUSE. Each memory array contains an array of MTJs, a row and column decoder, and a non-volatile register storing the column bitmask. Sense amplifiers are required for reads but are not used in computation. The memory controller contains non-volatile registers to maintain the architectural state.

3 MOUSE DESIGN

In this section, we describe the architecture of MOUSE and show how it is uniquely well suited for beyond-edge deployment. MOUSE utilizes CRAM arrays and minimal support circuitry. The computations performed in MOUSE are energy-efficient, highly parallel, and have an inherent robustness to intermittent operation. The basic architecture and operation semantics are the same as our previous work [107]. However, we expand the instruction set, improve the method of activating memory to perform computation, and add hardware support for branch instructions.

3.1 Hardware Organization

Figure 6 shows the architecture of MOUSE. It consists predominantly of CRAM arrays.[3] MOUSE can afford to have a large number of arrays (and hence more memory than is typical for a beyond-edge device) due to the ideal properties of MRAM. The memory arrays are not energy costly as MRAM has near zero standby power. Hence, MOUSE does not have to worry about static power with large memory capacity (as would be the case for SRAM/DRAM). Additionally, the area overhead of MRAM is very small.[4] For example, NVSIM [30] reports the size of a 64 MB STT-MRAM array (nearly twice the capacity required by our largest benchmark) as 15.12 mm^2. Commercial products of 256 MB and 1 GB STT-MRAM memory manufactured by Everspin come in a packages that are 130 mm^2 [1, 2]. For reference, MSP430FR5994 micro-controller, commonly used as a subcomponent of beyond-edge devices [20, 39, 45–47, 114], consumes over 100 mm^2. Additionally, as computation is performed within the memory arrays, there is no need for an external processor or area costly volatile memory (such as SRAM). Nearly the entire area budget of MOUSE is available for memory arrays. Each CRAM array contains 1,024 rows and 1,024 columns. In addition to the arrays, MOUSE requires the following minimal hardware to drive operations and maintain the architectural state:

(1) A memory controller that reads, decodes, and issues instructions;
(2) A non-volatile register for the **program counter (PC)**;

[3]Each array also contains a row decoder and column decoder.
[4]The SHE configuration consumes more area than STT because of the second transistor, which we detail in Section 6. However, the area budget still remains modest.

(3) A 128 byte **data register (DR)** that facilitates reads and writes;
(4) Two non-volatile registers, BR1 and BR2, for branch evaluation;
(5) Voltage sensing circuitry for monitoring the power source.

Minimal hardware is required for the memory controller.[5] With the exception of resolving branches (covered in Section 3.4.4) and updating architectural variables, its sole responsibility is repeatedly reading instructions, decoding them, and broadcasting them to the CRAM arrays. We use a highly simplified instruction set, covered in Section 3.4, hence decoding requires very little computation. The DR is the same size as one row of the MOUSE arrays and is used for intermediate storage when transferring data to and from different arrays. BR1 and BR2 hold data near the memory controller, enabling quick comparison tests for branch resolution. Finally, the voltage sensing circuitry is standard in beyond-edge devices and is as described in Reference [80].

3.2 Row Activation

In standard memory, a row decoder activates wordlines for read and write operations. As described in Section 2.3 and depicted in Figure 3, logic operations require the activation of multiple rows (up to three) simultaneously. To avoid increasing complexity of the row decoder, we use a latching mechanism that holds wordlines high after a row activation [76]. In this manner, the row decoder can activate rows sequentially with normal operation. The hardware cost is two transistors per row. Additionally, each logic operation must wait for the three sequential activations, which increases latency.

3.3 One-hot Column Decoder

Typically, it is desirable to drive logic operations in every column simultaneously. However, it is frequently preferable to perform computation only in a subset of the columns, leaving data in other columns unperturbed. Hence, in addition to a row decoder, we also need a column decoder that will select which columns participate in each operation.

Column activation patterns are different than for rows. With rows, 1–3 rows are activated for every instruction, and the rows that are activated are typically different for each consecutive instruction. When it comes to column activation, typically many columns are activated simultaneously (commonly all columns or a large subset). Additionally, columns tend to remain active for long periods of time—(de)activating columns is a rare event.

The original MOUSE design relies on a decoder that allows bulk addressing [107, 122]. Activation for the column decoder follows the same semantics as for the row decoder, except that there are reserved addresses that correspond to groups of columns, and up to 5 column addresses can be specified simultaneously [107]. In this work, we propose One-hot bitmask decoding to reduce the complexity of the decoder.

Rather than an address, a bitmask is supplied to the column decoder. In a 1:1 bit to column scheme, each bit indicates whether a specific column should be activated or not. The bitmask is stored in a 1,024-bit non-volatile register (corresponding to the 1,024 columns in each array) within each CRAM array. We call this register the ***column bitmask register*** **(CBR)**. The CBR can be written with a standard write operation. The advantage of One-hot bitmask activation is that the column decoder complexity is low: no addresses need to be resolved. The activation signal of each bit can be supplied directly to the columns. The disadvantage is that each activation of the columns (if the column addresses are changing) must be preceded by a write operation to CBR.

[5]Our memory controller is a standard memory controller that has been augmented with the capability to read, decode, and issue PIM instructions. We simply refer to it as the memory controller for the remainder of the article.

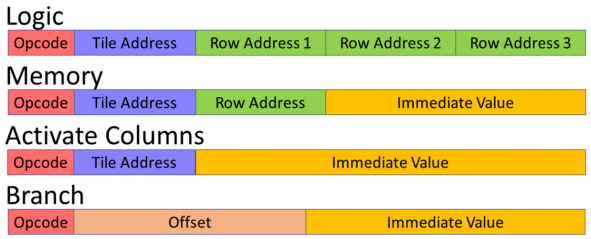

Fig. 7. MOUSE instruction formats. There are three types of instructions, logic, memory, and an additional activate columns instruction for configuration. Opcodes are 5 bits; array (tile) addresses, 9 bits; and row addresses 10 bits each. Branch offset is 20 bits. Remaining bits are used by instructions that use an optional immediate value.

It is possible to use different bit-to-column ratios. Fore example, a 1:32 scheme could be used, where each bit activates a consecutive set of 32 columns. This would allow a 32-bit mask to activate all 1,024 columns. The drawback is reduced flexibility as such a bitmask cannot activate less than 32 columns at a time. If computation requires less than 32 columns, then additional (and unnecessary) operations will be performed in all 32 columns. This wastes energy. As a 1,024-bit bitmask is easily handled by the CBR, we maintain a 1:1 ratio.

3.4 Instructions

Instructions for MOUSE are 64-bit and have the formats shown in Figure 7. There are four categories of instructions, which we explain here.

3.4.1 Memory Instructions. Reads and writes are the standard memory operations, but have additional overhead due to support for intermittent operation. The DR is a non-volatile register the same size as the rows of the CRAM arrays (128B) that holds data between read and write instructions. A read instruction reads from a CRAM array (at the specified address) and writes the data into the DR. A write instruction reads data from the DR and writes it into a CRAM array (at the specified address). Hence, if there is a power interruption between consecutive reads and writes, the DR will maintain the data being transferred—circumventing the need to re-perform the prior read operation. In addition to memory operations that use the DR, there is also a write immediate instruction, which allows instructions to write data directly into memory.

3.4.2 Logic Instructions. Logic instructions correspond 1:1 with logic gates (as covered in Section 2.2). For example, NOT, (N)AND, and (N)OR are all individual instructions. The instruction specifies the CRAM array address the operation is to be performed in and the row addresses of the logic gate (which rows the inputs and outputs reside in). NOT requires two row addresses (1 input, 1 output) and all others require three row addresses (two inputs, one output). For example, a NAND instruction may specify that it is to be performed in CRAM array 15, with inputs in rows 7 and 9, and the output in row 12. We restrict logic operations to at most two inputs, which are shown to be reliable [108, 152]. As COPY, XOR, and XNOR gates are not natively supported in CRAM, there are only five unique logic instructions. Analogous to vector instructions, many logic gates can be performed in parallel (triggered by a single instruction), as long as their inputs and output reside in the same rows. The number of parallel gates depends on which columns are active, discussed in Section 3.4.3.

The CRAM array address can specify a single array, or multiple arrays with bulk addressing [122]. There are reserved memory addresses that correspond to groups of memory arrays. For

example, it may be desirable to trigger an operation in all CRAM arrays. We designate array address 111111111 as a reserved address, to send an instruction to all arrays.

3.4.3 Activate Columns Instruction. It is necessary to specify which columns should participate in each operation. As noted in Section 3.3, consecutive operations typically use the same columns. Hence, which columns to activate changes infrequently. To take advantage of this, we use a strategy where columns are activated and then held active. All following logic operations will be performed in the columns that are held active. To (de)activate columns, we use a dedicated instruction, the **activate columns (AC)** instruction. As described in Section 3.3, the column decoder simply activates the columns depending on the values in CBR. Hence, a column activation consists of two components:

(1) A write to the CBR (*set*);
(2) Triggering of the column decoder to activate the corresponding columns (*activate*).

Typically, an AC instruction handles both components. However, when restarting the device it is only necessary to do the second. Hence there are two variants of the AC instruction, one that does both parts (set and activate) and one that only does the second (activate).

The write to CBR acts like a standard write. As noted in Section 3.4.1, a write can use the value in the DR or an immediate field in the instruction. Hence, there are a total of three unique versions of the AC instruction:

(1) Re-activate: Activate using pre-existing value in CBR;
(2) Set and Activate: Use data in DR to set CBR and then activate;
(3) Set and Activate (Immediate): Use data in the immediate field of instruction to set CBR and then activate.

3.4.4 Branch Instructions. Branch instructions involve an update to the PC in the event a logical condition holds true. As the logic required to evaluate the condition (e.g., checking equality of two numbers) is not complex, it can be implemented efficiently within the memory controller.

Non-volatile registers, BR1 and BR2, reside in the memory controller and are used for condition evaluation. We support simple standard branches based on BR1 and BR2:

(1) beq BR1 BR2: branch if BR1 and BR2 contents are equal;
(2) bge BR1 BR2: branch if BR1 is equal or greater than BR2;
(3) beqz BR1: if BR1 equal to 0.

Hence, the memory controller evaluates a simple condition based on BR1 and/or BR2 and updates the PC accordingly. Additional instructions are required to write values to BR1 and BR2. This follows the same semantics as writing to the CBR. A dedicated instruction writes to BR1 or BR2, and the value can come either from the DR or an immediate field in the instruction.

Branch instructions increase programmability by enabling function calls, repetitions of computational blocks, and handling I/O events. However, as the computation for branch instructions happens in the memory controller, it cannot capitalize on the extreme energy efficiency and the high degree of parallelism provided by the CRAM arrays. Therefore, efficiency tends to decrease with larger share of branch instructions in the instruction mix.[6]

3.4.5 Compilation. Compiling high-level code to MOUSE instructions (or any PIM substrate) requires knowledge of the PIM hardware to make efficient use of available parallelism. This is

[6] Avoiding branch instructions is easy for machine learning applications. For our benchmarks, we do not need any branch instructions during a single inference pass. Branches are used only to repeat inference or to handle I/O.

similar to compiling Open-CL or CUDA code for GPU architectures. Unfortunately, no equivalent standard software compiler exists for PIM. There is a rich design space, where a multi-dimensional trade-off exists between efficiency, area, power, and performance. Higher degrees of parallelism are possible by spreading computation out over more columns. However, this increases power, consumes more area, and adds communication overhead, which reduces energy efficiency. Our strategy rather is to minimize area by using as few columns as possible, to maximize energy efficiency. Our data layout is similar to a number of other works that have mapped applications to PIM substrates [76, 122], including machine learning algorithms [108].

3.4.6 Issuing Instructions. While operations can occur in multiple arrays simultaneously, arrays do not operate autonomously. All operations are triggered by the memory controller (discussed in more detail in Section 4). Effectively, there is a single controlling "thread," and hence there are no concurrency concerns between individual arrays. CRAM arrays in MOUSE hold both data and the instructions. For clarity, we categorize arrays into *instruction arrays* and *data arrays* based on their contents. However, as all arrays have identical hardware, arrays can be re-categorized to fit the programmer's needs.

Instructions and required data are written into the arrays prior to deployment. During operation, the memory controller repeatedly fetches an instruction from the instruction arrays, decodes it, and then broadcasts it to the data arrays. Instructions are performed entirely sequentially. The next instruction does not start until the previous has finished. This is to guarantee correctness, which we will cover further in Sections 3.6 and 4.

Generally, different instructions can take different time to complete. This is mainly because instructions can activate different numbers of rows. To guarantee that all instructions complete in time, for each instruction, the memory controller conservatively allocates as much time as the the longest instruction takes before starting the next instruction. This time lapse forms a *cycle*. This conservative approach to issuing instructions comes with a performance cost, as opposed to a more complex event-driven strategy. We opt for the conservative approach for three reasons: (1) MOUSE already delivers higher performance than representative alternatives for beyond-edge computing (as we cover in Section 6), hence aggressive optimization is unnecessary. (2) Complex issue logic would be less energy-efficient and make it more difficult to guarantee correctness under intermittent operation. (3) Simplicity is a strength for beyond-edge devices. In the end, energy efficiency (rather than high performance hardware) is the limiting factor for performance beyond the edge [40]. Designs consuming less energy complete programs faster, because they spend less time waiting for the harvested energy to reach sufficient levels for forward progress in computation.

3.5 Power Draw

As power sources for beyond-edge devices are highly variable, it is undesirable to connect them directly to the compute circuitry. A solution is to utilize an energy buffer (capacitor), which is charged by the power source. The device can consume energy from this buffer, without having to match the power supplied from the source in real time [82]. Power delivery systems such as Capybara [20] are specifically designed to harvest energy and reliably power beyond-edge devices in this manner. Because MOUSE uses such an energy buffer, it accumulates energy over time and consumes it in bursts. This allows MOUSE to consume more power during its power-on time than the power source provides. At the same time, it is possible to program MOUSE to fine-tune its power consumption. Instructions determine the degree of parallelism, i.e., the number of active columns. Higher degrees of parallelism, i.e., many active columns, result in higher By controlling the number of active columns on a per instruction basis, the programmer can explore the power versus performance trade-off.

3.6 Intermittent Processing

Beyond-edge devices are powered by unreliable sources and must frequently shutdown. A key challenge is maintaining program correctness during frequent power cycles, which is referred to as intermittent processing. Energy is a precious resource for beyond-edge devices, and any energy spent on guaranteeing correctness would not be available for normal program execution. Hardware for intermittent computing is therefore tightly constrained by energy efficiency in providing a correctness guarantee.

Previous work covers sophisticated software and hardware strategies for intermittent processing on more traditional architectures [19, 38, 90, 111]. MOUSE operates in significant contrast to these strategies, in that MOUSE is able to checkpoint after *every* instruction and still maintain correctness with extremely limited additional hardware, which boils down to only a valid copy of the PC and an additional non-volatile status bit. This strategy is extremely simple and would be crude for more traditional architectures, while more conventional, more sophisticated or complex strategies are unsuitable and unnecessary for MOUSE.

As MOUSE performs the computation in non-volatile memory, progress is automatically saved after *every* operation and MOUSE can checkpoint after every operation with very low overhead. Hence, there is no additional backup operation required, which typically represents a complex task of very high overhead in traditional systems. Specifically, when MOUSE restarts, only two pieces of information are required:

(1) the last instruction that was completed (valid value of the PC),
(2) which columns were active,

where the memory controller writes (checkpoints) the PC into a non-volatile register after the completion of every instruction. We provide a detailed discussion on the correctness of the PC in Section 4.2.

In the worst case, MOUSE loses power after an instruction is completed, but before the PC can be updated and saved. When power is restored, MOUSE re-issues the same instruction, performing it for a second time. However, this does not break correctness as the same result is obtained if a single instruction is repeated multiple times, i.e., each such repetition is *idempotent* [49, 136], as covered in Section 4.1. The only requirement is that the PC checkpointing happens strictly after each instruction is performed.

Checkpointing after every instruction not only minimizes the work potentially lost on shutdown but also simplifies the restart process. The correctness guarantee comes from each operation being *idempotent*, which does not apply to sequences of operations (over multiple instructions). This is because the inputs to logic operations can be overwritten over the course of multiple instructions. If we re-perform multiple instructions, then these input values may be incorrect. To guarantee correctness when repeating multiple instructions, software-level policies can help but incur additional and unnecessary complexity for MOUSE.

The second requirement on restart is to restore the previously active columns. As the active columns are stored in the CBR of each memory array, all we need is for the memory controller to issue a *re-active AC instruction*. The column decoder in each memory array then re-activates its columns and the memory controller can resume issuing instructions. We cover correctness during intermittent operation in grand detail in Section 4.

3.7 System Integration

When performing the computation for inference, MOUSE is a self-contained system. Memory arrays hold all the instructions and data and the memory controller drives operations. As a full-fledged beyond-edge device, MOUSE is integrated with an energy-harvesting power source, a

sensor to provide input data, and a transmitter. We assume that input data is stored in a non-volatile buffer within the sensor. The sensor is given a memory address (as if it was a memory array), where MOUSE can use read instructions to retrieve data from it. Additionally, the sensor has a non-volatile valid bit, which indicates if new input data is ready. When MOUSE is ready to receive new input data, it can check the valid bit and begin reading from the sensor and writing the data into the MOUSE *data arrays*. These reads and writes are controlled by instructions in the *instruction arrays*, hence data transfer is a software controlled (programmable) process.

When MOUSE finishes inference, the memory controller reads out the data from the arrays, and writes it into a non-volatile buffer for the transmitter. This buffer, as well, is is given a memory address (as if it was a memory array), where standard write instructions can be used. In this work, we focus only on the accelerator and do not consider any overhead for the sensor or transmitter.

The programmability of the data transfer process is important. For example, it is possible that MOUSE loses power during the process of transferring data in. If power is not available for an extended period of time, then when MOUSE restarts there may be a new set of data in the sensor. MOUSE can handle this with branch instructions. If the data in the sensor is timestamped, then the first instruction in the transfer process can be used to copy the timestamp of the first data chunk into *BR1*. The last instruction in the transfer process can be used to copy the timestamp of the last data chunk into *BR2*. MOUSE can then check the equivalence of *BR1* and *BR2*, and branch back in case of a mismatch to the beginning of the transfer to overwrite old data.

4 INTERMITTENT CORRECTNESS GUARANTEE

Beyond-edge devices need to ensure program correctness in the presence of power outages. If not handled carefully, then interruption due to power outage can corrupt the architectural state. In this section, we show how MOUSE remains correct, even in the presence of unexpected power outages. There are two crucial components: the correctness of individual in-memory operations when interrupted or re-performed (Section 4.1) and the correctness of architectural state variables in transitions between states (Section 4.2). As MOUSE checkpoints after every instruction, we need to only show that each instruction and the transitions between instructions remain correct when interrupted. In the following, we show that all instructions and transitions are *idempotent* [49, 136], which means that they produce the same results, even if repeated multiple times. The key to remaining idempotent is not over-writing data that is required on restart (or if the data gets overwritten, it should be in a manner that does not change the computation outcome). The architectural state variables and their protection mechanisms are listed in Table 2. Note that the correctness guarantee covered in this section applies only to interruptions and power outages. It does not cover errors in the computation itself or perturbations due to soft errors from radiation.

4.1 Operation Level Correctness

In this section, we cover the correctness of individual operations performed in the memory when interrupted and re-performed. We assume the most general case, where the power can be cut at any moment (unexpectedly). Hence, we need to consider all possibilities for when (during its processing) an operation can get interrupted.

4.1.1 Logic Operations. All logic operations are threshold operations (the output MTJ either switches or it does not). Hence, there are only two stages for each logic operation, pre- and post-switching. In the following, we use an AND operation as an example. However, the observations here apply to all gates.

To perform an AND gate, the output MTJ must be in the logic 1 (high resistance) state. Voltage is applied across the two inputs and the output (as in Figure 1), such that electrons flow from

Table 1. Four Possible Cases for Re-performing an Interrupted AND Gate

	Output did not switch before interruption	**Output did switch before interruption**
Output should not switch	Input MTJs prevent the switching of the output MTJ, both before and after interruption.	Not possible. Input MTJs prevent switching of output MTJ. By construction, repetition cannot induce switching.
Output should switch	Inputs and output did not change prior to interruption. Second attempt has same inputs and will produce correct output.	The output has already switched to 0 (correct output). Second attempt has the wrong output preset value. However, due to the *direction* of the current, the output MTJ will remain at 0.

The output MTJ either should or should not switch for correct operation, and it either did or did not prior to the power being cut.

Table 2. Architectural State Variables and How They Are Protected Under Power Interruptions

Variable	Volatility	Protection Mechanism
Program Counter	Non-Volatile	Duplicated, valid copy is read only
Parity Bit	Non-Volatile	Only flipped after instruction has finished. Flip is an atomic operation
BR1 and BR2	Non-Volatile	Write operation guarantee (Section 4.1.2)
CBR	Non-Volatile	Write operation guarantee (Section 4.1.2)
DR	Non-Volatile	Read and Write operation guarantee (Section 4.1.2)
Active Columns	Volatile	Bitmask stored in CBR. Re-actived on restart with AC instruction (Section 3.4.3)
Active Rows	Volatile	Activated by every instruction
Data	Non-Volatile	Idempotent logic operations (Section 4.1), Read and Write operation guarantee (Section 4.1.2)

the fixed layer to the free layer of the output MTJ. This current can potentially change the state of the output MTJ to 0. If either of the two inputs is 0 (low resistance state), then the current becomes sufficiently high to switch the output to 0. If both inputs are 1, then the current is too low and the output keeps its state of logic 1. We now consider what happens when this operation is interrupted due to a power outage, and when we need to re-perform AND a second time once power is restored.

Consider first the case where the output MTJ *should not* switch. This means that the states of the input MTJs are preventing the output MTJ from switching. Hence, the output MTJ could not have switched prior to the interruption. When we re-perform the operation, the initial states of all MTJs are the same. This is identical to performing AND the first time, and again the output MTJ does not switch.

Now consider the case where the output MTJ *should* switch. In this case, there are two possibilities: (1) The output MTJ did not switch before the interruption and, (2) the output MTJ did switch prior to the interruption. For possibility (1), when re-performing the operation, the initial states of all MTJs are the same. Hence, performing AND the second time is identical to performing it the first time (minus the interruption). The second time, the operation finishes, and the output MTJ switches as desired. In possibility (2), the MTJs do not keep their initial state: the output MTJ has already switched to 0, whereas it should be preset to 1. Still, the AND operation remains correct when performing it a second time. This is because the current applied can only cause the output MTJ to switch to 0, it cannot revert it back to 1. Hence, after performing AND a second time, the output MTJ will remain in the 0 state, as desired.

All four possible cases are listed in Table 1. The catch here is that repeating a logic gate is effectively the same as performing the gate for a longer duration. Doing so results in an identical outcome, whether the output MTJ switched before interruption (i.e., power outage) or not.

4.1.2 Memory Operations. Re-performing a read operation has no effect on the read data, reading it a second time will produce the same results. Re-performing a write will overwrite whatever was written the first time. If the write was not successful the first time (due to interruption), then it will be successful the second time. If the write was successful the first time, then the same value

will be written twice. As noted in Section 3.4.1, read (write) instructions can involve a write to (read from) the *DR*. These are protected by the idempotency of both read and write operations—a memory operation does not write to any address/register that it also reads.

4.1.3 Column Activation. Column activation involves a write to the CBR in a data array and then a triggering of the column decoder. The write to the CBR is kept correct by the same semantics as memory operations (a write can be performed multiple times). The column activation by the column decoder does not change any non-volatile data and hence cannot introduce corruption. The volatile state is entirely lost on shutdown and will be overwritten on restart.

4.1.4 Summary. Power interruptions can waste energy (due to re-performing instructions) but cannot corrupt the data in memory. Idempotency of all instructions guarantees that they produce the same results, even if performed multiple times. Moreoever, idempotency is not required beyond a single instruction as only one instruction is performed between each checkpoint.

4.2 Maintaining Correct State

The previous section showed that the individual operations performed in the memory are idempotent. This is necessary but not sufficient for correctness. MOUSE has to guarantee that the memory controller can drive the operations and maintain the architectural state in an intermittent safe fashion.

4.2.1 Memory Controller. The memory controller repeatedly reads instructions from the *instruction arrays*, decodes them, and broadcasts them to the *data arrays*. The memory controller waits for a sufficient time for each instruction to complete before updating the PC. The PC must be stored in a non-volatile register to prevent loss on shutdown. However, concern remains if the update to the PC gets interrupted. If power is lost during a write to the PC register, then it can be corrupted—resulting in incorrect behavior on startup.

We circumvent this issue by maintaining two PC registers, PC0 and PC1, and an additional non-volatile parity bit. If the parity bit is 0, then PC0 is valid, and if the parity bit is 1, then PC1 is valid. When the memory controller updates the PC, it takes the value in the valid PC register, updates it accordingly, and stores it into the invalid PC register. Then it flips the parity bit, indicating the advancement to the next instruction. Therefore, the memory controller never writes to the valid PC register, and there is no risk of corruption.

The setting of the parity bit is analogous to the committing of an instruction in traditional architectures. As the parity is a single bit, the operation is *atomic* and cannot be interrupted mid-way through. The parity bit either is set or not. If an interruption occurs before the parity bit is set, then the memory controller re-issues the same instruction on restart, which is safe to do (Section 4.2.2). If the interruption occurs after the parity bit is set, then the instruction is completed and the memory controller issues the next instruction on restart, as depicted in Figure 8.

There are other non-volatile registers that hold the architectural state. These include the DR, branch registers (BR1 and BR2), and the CBR in each array. These registers are protected by the same semantics as in Section 4.1.2. Updates (i.e., writes) to these registers are guaranteed to be completed before the memory controller commits the corresponding instruction. No register is both read and written by the same instruction, so no required data can be corrupted.

Active columns is part of the architectural state. When MOUSE restarts, these columns need to be re-activated. The non-volatile CBR in each memory array maintains the currently valid bitmask for active columns. All that is required is for the column decoders to re-activate these columns. To achieve this, the memory controller issues a re-activate columns instruction to all arrays on restart (Section 3.4.3).

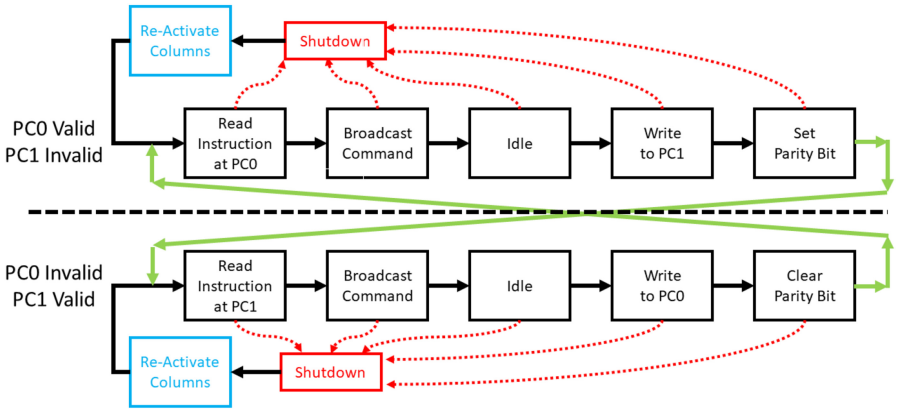

Fig. 8. Memory controller's state transitions to ensure the correctness of the program counter as MOUSE transitions from one instruction to the next. Effect of interruptions are dashed and highlighted in red, corrective measures in blue, and forward progress (guaranteed completion of an instruction) in green.

4.2.2 Data in Arrays. The previous section showed that the memory controller itself remains correct during intermittent operation. We must also ensure that the memory controller does not generate any signals that corrupt the data residing in the memory arrays.

The memory controller broadcasts instructions to the data arrays. This broadcast is not atomic, and thus can be interrupted at any stage. However, all the operations that it can trigger are idempotent (Section 4.1), meaning they can safely be interrupted at any point in their progression. As a direct result, the broadcast cannot cause corruption as its only effect is the initiation of the operation. Power can be cut before the broadcast reaches a memory array, while the operation is being performed, or after the operation has finished—none of these cases can introduce error.

5 IMPACT OF ORBITAL DEPLOYMENT

An exciting domain for beyond-edge devices is LEO, where they can act as nanosatellites [83]. One aspect of LEO deployment is that the cost of communication becomes much greater than computation (even more so than for terrestrial deployment) [39, 83]. MOUSE is especially well suited for LEO deployment as it has a much larger memory capacity relative to other beyond-edge devices as MOUSE nearly entirely entails high density non-volatile memory. Due to the large memory capacity, MOUSE can go long periods of time without offloading data. However, orbital deployment also introduces challenges related to operating temperature and radiation. We discuss here how MOUSE can tolerate such conditions.

5.1 Temperature

Satellites in LEO can experience a wide range of temperatures, from −170°C to 123°C [74]. Maintaining the proper temperature on large scale satellites is an important engineering challenge [9]. Nanosatellites, however, typically do not have sufficient resources for any temperature modulation yet they need to operate properly across a wide range of temperatures. Non-volatile memory characteristics are very sensitive to temperature [106], which further challenges this situation for MOUSE.

MTJ resistance increases with decreasing temperature, to the extent that the resistance at −170°C can be 30% higher than at room temperature [71, 147]. This increases the voltage required to write MTJs, and consequently, energy consumption. The SHE architecture is less sensitive than STT, as the SHE channel is metallic (to be more specific, the resistance does not increase with

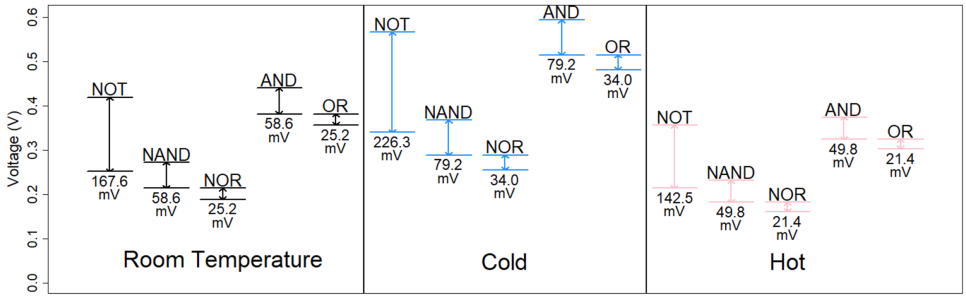

Fig. 9. Voltage ranges for correct operation of each logic gate at different temperatures, room temperature (27°C), Cold (−170°C), and hot (123°C).

decreasing temperature). Therefore, write operations with SHE remain largely unaffected. However, SHE still requires current to travel through the body of input MTJs for read and logic operations. Hence, the overall energy efficiency of SHE still decreases. As a result, PIM using MTJs (or other non-volatile technologies, as well) is less energy-efficient at cold temperatures [106]. However, the change is modest, within approximately 10% more energy consumption (relative to room temperature), even at cryogenic (77K) temperatures [106]. Given that MTJs are extremely energy-efficient [151, 153], this increase in energy consumption remains tolerable. Additionally, there is a benefit of cold temperature. The ratio between the high and low resistance state increases [71, 150]. This leads to more robust logic gates, which are less susceptible to voltage fluctuations [108, 152].

The inverse is true at high temperatures. The overall MTJ resistance and the ratio between the high and low resistance states are both lower. MTJ resistance at 123°C is roughly 86% of its resistance at room temperature. This translates into more energy-efficient MTJ logic gates, which at the same time are more susceptible to voltage fluctuations. The latter challenges the power delivery system. Power systems such as Capybara [20] become necessary to ensure that the proper voltage is applied across a variety of temperatures. Switched-capacitor voltage converters [59, 103, 103] can generate necessary voltages to facilitate correct operation. We cover the overhead of voltage generation/conversion in Section 6.

In contrast to the resistive memory, the peripheral circuitry in MOUSE benefits from cold temperatures. At colder temperatures CMOS transistors have higher ON current [147], switch faster [99], feature a higher trans-conductance, and incur a lower leakage current due to a steeper sub-threshold slope [124]. However, MOUSE does not benefit significantly from these characteristics. This is because the non-volatile memory already has extremely low static power, and the latency is limited by the switching time of the MTJs. CMOS performance can degrade with increasing temperature due to increased leakage current, while typical CMOS devices can operate well up to 175°C [61]. Radiation hardened bulk CMOS technology can increase this range further to 250°C [61, 81]. Hence, CMOS technology is well suited to operate within the expected temperature range of LEO satellites. We discuss the overhead of our CMOS components further in Section 6.

5.1.1 Voltage Margins. Logic operations with MTJs discussed in Section 2.2 require proper voltages to be applied across the inputs and output. Correct operation can only be the case if each gate-specific voltage is within a specified range [108, 151]. The required voltage depends on the logic operation (which determines the number of inputs and the output preset), the resistances of the MTJs (R_P and R_{AP}), and the switching current (I_{switch}). Because the MTJ resistance changes as a function of the temperature, the proper voltage ranges do, as well. Figure 9 shows the voltage ranges for different logic operations at different temperatures. MTJ resistance is higher at cold

temperatures, making the required voltages higher. At high temperatures, the MTJ resistance is lower. In addition to the voltages being lower, the ranges are also smaller. This reduces the margin for error in the voltage supply. MOUSE relies on a power delivery system that can reliably supply the appropriate voltage. This becomes an even more challenging task considering temperature fluctuations.

5.2 Radiation

Radiation can cause errors in electrical circuits. When a stray energetic particle strikes the hardware, it induces a voltage spike that can travel along the circuit and potentially alter output voltages or flip logical values [118]. Corrupted logic values give rise to *soft errors*. Technology scaling already makes any circuit ever more susceptible to soft errors [145]. However, when deployed in orbit, beyond-edge devices are exposed to significantly higher levels of radiation, which translates into frequent particle strikes.

CMOS components of MOUSE (the memory controller and peripheral circuitry, respectively) are susceptible to soft errors. Radiation can disrupt the CMOS logic in the memory controller or the voltage supplied by the power delivery circuitry (required to drive logic operations in memory). The consequences vary greatly depending on the location. The impact of soft errors in computation will likely be minor due to the resilience of machine learning to noise. Hirtzlin et al. [50] showed that frequent bitflips in binary neural networks implemented in STT-MRAM result in negligible degradations in accuracy. However, a bit flip in the memory controller (corrupting the architectural state) can lead to undefined behavior. For example, if the memory controller experiences an error where it incorrectly updates the program counter, the device could attempt to read instructions from invalid addresses. Such an error could permanently disable the device. Hence, hardening MOUSE to radiation is imperative.

Many mitigation techniques exist for soft errors, which can be categorized into-system level, device-level, or circuit-level [117, 118, 120]. As MOUSE must stay correct in transitions from each instruction to the next (due to check-pointing after each instruction), and as increasing complexity also increases the overhead for correctness guarantees, system-level mitigation is less appropriate—i.e., soft errors should be caught before they manifest themselves at the system (architecture) level. Additionally, MOUSE relies on pre-existing CMOS devices, making device-level mitigation impossible.[7] Hence, circuit-level mitigation is the most suitable approach.

MOUSE relies on switched-capacitor circuits to supply appropriate voltages to the memory, which can be particularly susceptible to radiation. Circuit-level techniques can be especially helpful in this case, e.g., in the form of additional feedback paths to counteract the impact of particle strikes [33]. Thereby, if a single path experiences a voltage spike (or drop), an alternate path can take over to compensate. When properly designed, the impact on the final output can be minimized even if a large disturbance is experienced at the input of a circuit. This comes at a cost in area and energy efficiency due to the larger number of transistors per circuit.

MOUSE also relies on CMOS logic circuits within the memory controller to decode instructions and send commands to the memory. CMOS logic can be hardened to soft errors with a variety of circuit-level techniques. Redundancy can be added, either in space [98] (with area overhead) or time [92, 95] (with latency overhead), where outputs are checked for consistency. Increasing the node capacitance and transistor drive current (at a cost of area and energy) can also reduce the electrical susceptibility to particle strikes [157]. More sophisticated strategies of lower area overhead involve creating "transmission gates" between stages of a circuit, filtering out pulses from particle strikes [70, 119, 156].

[7] A major exception is pre-existing properties of MTJ devices, discussed later.

While MOUSE does critically rely on CMOS circuitry, the vast majority of MOUSE's computation and all of its memory involve MTJs. Fortunately, MTJs are considerably more robust to soft errors than alternative technologies While short-lived voltage pulses suffice to change the state of CMOS circuits, MTJs require a significant current (a few microamps) for a sustained period of time (a few nanoseconds) to switch states. Hence, particle strikes are highly unlikely to flip MTJ states. In fact, MTJs are shown to be highly resilient to radiation from heavy ions [21, 66], neutrons [105], protons [53], and gamma rays [53, 105]. Recently, Montoya et al. [94] demonstrated that MTJs are even resilient to radiation that is 100× greater than what is observed on particularly harsh interplanetary travel [11, 154]. Therefore, MTJs represent leading candidates for space applications [37, 65]. Since MOUSE consists mostly of MTJs, it is less susceptible to radiation than traditional architectures. As only minimal CMOS circuitry is required external to the memory arrays, circuit level strategies to increase CMOS resilience to radiation [157] incurs a relatively low overhead.

6 EVALUATION SETUP

Benchmarks: The exact use case of beyond-edge devices can vary significantly, applications include agricultural monitoring [83, 137], security, and structural and environmental monitoring [27]. However, general sensor processing algorithms can be used to solve a wide variety of problems. We use benchmarks that are representative for many possible use cases—machine learning inference on data sets which are tenable for beyond-edge devices. The specific input problem will vary depending on the user, however, the computation involved should remain highly similar.

We implement two machine learning algorithms, SVM and BNN. Both are widely used and light weight, which makes them highly suitable for the beyond-edge domain. For both, we used only operations that are efficient in MOUSE, all bit-wise and integer arithmetic. Machine learning applications are well-suited for integer arithmetic as they remain robust under approximation. Fixed-point representation using integer arithmetic is sufficient to achieve high accuracy [54]. We designed customized SVM implementations and trained and tested them in R [102]. We were able to achieve similar accuracy as standard SVM implementations from libSVM [14]. For inference, the main computation is effectively performing the dot product between an input vector and each of the support vectors. The results of these dot products are then squared, multiplied by a set of coefficients, and finally summed together. By construction, SVMs have two class outputs, where the sign of the output value is the classification. We extend to multi-class classification by training a separate SVM for each possible output class, where each has the task of identifying a single class. BNNs are neural networks that use neurons and weights represented by a single bit each [23]. This enables multiplications to be replaced by XNOR operations and addition is simplified to a popcount operation. This gives BNNs extreme energy efficiency. Previous work has efficiently mapped BNNs onto FPGAs, including FINN [134] and FP-BNN [77]. We copy their network configurations exactly. We modify the algorithms only in transforming them to run on our PIM substrate. Hence, our accuracy is identical.

Data Sets: For small-scale image recognition, we use MNIST [72]. The task is digit recognition, where a 28×28 pixel image with 8-bit precision is to be classified into one of ten digits (0–9). We use both BNNs and SVMs on this benchmark. With SVM, the pixels are a 784 element vector. We also create a binarized version, where pixels that have a value below $255/4 \approx 63$ are assigned 0 and those above are assigned 1. This allows us to replace multiplications with AND gates, significantly reducing the time, energy, and area overhead. For BNNs, we use the network configurations of FPGA-based FINN [134] and FP-BNN [77]. FINN [134] uses binarized input, has three hidden layers of 1,024 neurons (bits) each, and the output layer has 10 neurons with 10-bit precision. FP-BNN [77] 8-bit inputs, has three hidden layers of 2,048 neurons each, and the output layer is 10 neurons with 16-bit precision.

Table 3. Parameters for MTJ Devices

Parameter	Modern	Projected
P State Resistance	3.15 kΩ	7.34 kΩ
AP State Resistance	7.34 kΩ	76.39 kΩ
Switching Time	3 ns [96, 113]	1 ns [55, 151]
Switching Current	40 μA [113]	3 μA [151]

Human Activity Recognition (HAR) [4] is a data set that has accelerometer and gyroscope measurements from a smartphone, which is carried by participants performing a variety of activities. The problem is to classify the physical activity the individual is performing. Each input is a vector of 561 elements. We convert the input to fixed point representation with 8-bit precision.

ADULT [67] contains census information. The problem is to classify whether an individual makes greater than $50K per year or not. We use a reformatted version of the data set from libSVM [14]. Each input is a 15 element vector where each element is an 8-bit integer.

Performance and Energy Model: We use an in-house simulator to evaluate MOUSE. We set each array in MOUSE to 1,024 × 1,024, which is a recommended subarray size for non-volatile memories from NVSIM [30]. Our simulator tracks all instructions issued by the memory controller and accounts for the time and energy consumed by each. An instruction can consume energy by performing the following actions:

(1) Reading the instruction from the instruction array.
(2) Sending the instruction to the data arrays.
(3) The activation of rows for computation.
(4) The activation of columns for computation.
(5) The switching energy of the MTJs in memory.
(6) Update of the program counter and parity bit

Items (1) and (6) always occur, while the remaining items occur depending on the instruction type. All energy consumption comes either from the MTJ devices or from the peripheral circuitry. The models for both are discussed below.

We simulate with both modern MTJ parameters [112] and with projections of MTJ parameters expected to be possible within a few years [151, 153]. MTJs are expected to be significantly more energy-efficient as the technology matures. Two techniques will enable a reduction in the switching current, (1) decreasing the damping constant of ferromagnetic materials [31, 93, 116] and (2) using a dual-reference layer structure [28, 51]. It is possible switching currents will be as low as 1 μA, however, we assume 3 μA to be conservative. The parameters we use are shown in Table 3. For Modern MTJs, we use only the STT architecture, and for projected MTJs, we use both the STT and SHE architectures. The benefit of SHE is providing a more efficient write mechanism. We model the SHE channel as a 1 kΩ resistance. This provides a conservative estimate of SHE energy efficiency.

Due to the different switching times of modern and projected MTJs, we clock MOUSE at different speeds for each. With Modern MTJs MOUSE operates at 30.3 MHz clock rate (33 ns per cycle) and for projected MTJs MOUSE operates at 90.9 MHz clock rate (11ns per cycle). This enables sufficient time for instruction read, decode, and the peripheral circuitry latency and MTJ switching time.

For modeling peripheral circuitry, we take data from NVSIM [30], which reports the relative overhead of peripheral circuitry for modern MRAM memory. We set the overhead of MOUSE so that it consumes the same relative share of total latency and energy as reported by NVSIM.

We first evaluate MOUSE with continuous power (using a power source that can supply as much power as MOUSE desires). Then, we evaluate with an energy-harvesting power source where

MOUSE will have to operate intermittently. We model the energy harvester as a (small) constant power source that is filling an energy buffer (capacitor). When MOUSE is off, the power source charges the capacitor and the voltage will rise. When MOUSE is on, it will consume the energy and the voltage will drop. MOUSE will shut off when the voltage hits a pre-defined minimum value, hence the voltage on the capacitor will fluctuate within a specified range. When the voltage hits the lower end of the range, power is instantaneously cut—MOUSE does not do any preparation for the shutdown. We start all benchmarks with a capacitor that has voltage just below the cutoff, hence all benchmarks begin with an initial charging time. Modern MTJs and Projected MTJs have different operating voltages [151], so we use a different voltage range for each technology. We let the voltage fluctuate between between 400 and 420 mV when using Modern MTJs and between 100 and 120 mV when using Projected MTJs. Switched-capacitor converters are used for upconversion and downconversion [43] to supply the required voltages for all operations. All required voltages can be acquired by using conversion ratios of 0.75, 1, 1.5, and 1.75 [59, 103]. We evaluate MOUSE on the power supplied by the converter, the evaluation does not include regulator efficiency overhead. The converter may have an efficiency anywhere between 35-80%, hence the energy harvester may need to provide roughly 1.25–2.85× the energy that MOUSE consumes. As noted in Section 3.4.6, a single instruction is performed in every cycle. A portion of the cycle must be dedicated to changing the output voltage of the converter (if consecutive operations require different voltage levels). The time overhead can be overlapped with the row activations.

It is desirable to match the capacitor size to the expected energy consumption. Hence, we also use different capacitor sizes for modern and project MTJs. We use a 100 µF capacitor (energy buffer) with Modern MTJs and a 10 µF capacitor for Projected MTJs. The optimal capacitor size depends on the technology and the program being executed. When deployed, a system such as Capybara [20] could be used to tune the parameters of the energy buffer.

Given that energy-harvesting power sources can vary significantly in how much power they can provide, we sweep the power source over a wide range. At the low end, we test from 60 µW, which is approximately what can be harvested from a 1 cm^2 thermal energy harvester running on body heat [64, 75]. This is well below the operating power of MOUSE. At the high end, we use 5 mW, which is the same power harvested by the beyond-edge device SONIC [39]. This can nearly power MOUSE continuously. Beyond-edge devices deployed as satellites will likely use solar cells as power sources [83]. The amount of power that can be harvested will depend on the size of the cells (typically very small) and their orientation, which is likely to change over time.

Area Overhead: The CRAM arrays used in MOUSE have a similar area overhead as MRAM arrays. The extra overhead of STT CRAM is an extra bit line per column, which is a minor impact. For SHE CRAM, a second transistor and SHE channel is required in each cell, which has a significant impact.

We base our cell area estimates on Zabihi et al. [152]. We use configurations where the access transistors have a resistance less than 1 kΩ and give an extra 10% to account for spacing and layout issues. The access transistors and MTJs can be placed on separate layers. As the transistors are much larger, they dominate the area overhead. As the SHE architecture has twice as many transistors, it is approximately twice as large. We use NVSIM [30] to estimate the area overhead of peripheral circuitry. NVSIM reports the percentage of chip area that must be dedicated to the peripheral circuitry for different memory sizes. We increase the area overhead for each benchmark accordingly. Our conservative area estimates are shown in Table 4.

Impact of Temperature: MTJs have been demonstrated to function over a wide range of temperatures [71, 150]. However, the MTJ resistance increases at colder temperatures, which will increase energy consumption. We test MOUSE both at −170°C (cold) and 123°C (hot). To model the impact on MTJs, we take data from Yuan et al. [150]. For cold temperatures, we conservatively

Table 4. Area Required for MOUSE for Different Benchmarks and Configurations

Benchmark	Total Memory	Modern STT [152]	Projected STT [152]	SHE
SVM MNIST	64 MB	28.04	21.27	42.54
Binarized	8 MB	2.99	2.27	4.53
SVM HAR	16 MB	5.97	4.53	9.06
SVM ADULT	1 MB	0.39	0.29	0.58
BNN FINN MNIST	8 MB	2.99	2.27	4.53
BNN FPBNN MNIST	16 MB	5.97	4.53	9.06

Units are in mm^2.

estimate the MTJ resistance increases by 30%. For the STT architecture, this increases the write, read, and logic energy consumption by 30%. For SHE, the write energy remains unaffected as the SHE channel (which is metallic) is use for write operations. However, energy consumption still increases for read and logic operations. The CMOS circuitry will generally perform better at cold temperatures, having a lower latency and potentially lower energy [99, 124, 147]. However, to be conservative, we assume no additional efficiency of the peripheral circuitry. The latency improvement of CMOS does not benefit MOUSE as we choose to maintain the same clock rate across temperature ranges. Hence, the latency of each instruction remains the same. At hot temperatures, the MTJ resistance drops by approximately 13% [150]. We model this in an identical fashion to cold temperatures, where we change the energy efficiency of each operation.

Impact of Radiation: As noted in Section 5.2, MTJs have an inherent resilience to radiation. However, the CMOS components of MOUSE remain vulnerable. Errors in the CMOS circuits can lead to undefined behavior. Hence, in order for MOUSE to work in orbital deployment, these errors must be suppressed. Circuit-level strategies can make CMOS circuits resistant to soft errors [118, 157]. These strategies come with a power and delay cost. We choose to be conservative and assume a large overhead. We assume a 60% increase in CMOS energy and a 10% increase in CMOS latency, one of the largest overheads reported by Zhou et al. [157]. Hence, the performance and efficiency of an orbitally deployed MOUSE will be reduced, relative to that of its counterpart designed for terrestrial deployment.

Baseline for Comparison: We compare MOUSE with SONIC [39], a beyond-edge device that performs machine learning inference on the same benchmarks we use. As SONIC was evaluated at room temperature, we must estimate its performance at different temperature ranges. To be conservative, we assume SONIC can fully exploit the benefit of CMOS operation at cold temperatures, increasing performance by 30% [99]. We also assume it suffers no negative consequences of varying temperature (hot or cold) and it pays no overhead resilience to radiation. We also compare against estimates of the vector architecture MANIC [40]. We also give MANIC overly optimistic assumptions, a 30% boost in performance and no overhead for temperature or radiation. MANIC was not evaluated on end-to-end inference, rather on computational kernels required for inference (i.e., convolution). Hence, we rely on rough estimates of its performance on the same benchmarks. We follow the authors' statement, that MANIC 9.6× more energy-efficient than SONIC [40].

7 EVALUATION

Continuous Power: Continuously powered MOUSE at room temperature and related work is reported in Table 5. MOUSE implements both BNNs and SVMs. SONIC [39] is a beyond-edge device that uses TI-MSP430FR5994 microcontroller to run neural networks on the same benchmarks. For reference, our custom SVM implementation and optimized SVMs from libSVM [14] are run on a Intel Haswell 5-2680v3 processor. To be conservative, we account only for the processor power

Table 5. Continuously Powered MOUSE at Room Temperature (Using STT Design and Modern MTJ Devices) and Related Work Under Continuous Power

Benchmark	Latency (µs)	Energy (µJ)	#SV	I/D Mem (MB)	Area (mm^2)	Accuracy
SVM (CPU)						
MNIST	169,824	5,094,702	11,813	—	—	97.55
MNIST (Binarized)	192,370	5,771,085	12,214	—	—	97.37
HAR (integer) [4, 133]	127,494	3,824,822	2,809	—		95.96
ADULT	4,368	131,052	1,909	—	—	76.12
MOUSE SVM (Modern STT)						
MNIST	23,116	1,700	11,813	4.5/30.0	28.04	97.55
MNIST (Binarized)	6,071	81.43	12,214	1.25/6.0	2.99	97.37
HAR (integer) [4, 133]	11,312	575.8	2,809	2.25/10.0	5.97	94.57
ADULT	1,104	9.06	1,909	0.25/0.5	0.39	76.12
MOUSE BNN (Modern STT)						
MNIST (Binarized) FINN	1,605	18.04	NA	3.15/1.71	2.99	98.4
MNIST FP-BNN	2,150	125.4	NA	4.20/8.00	5.97	98.24
libSVM [14]						
MNIST	7,830	234,900	8,652	—	—	98.05
MNIST (Binarized)	19,037	571,116	23,672	—	—	92.49
HAR (integer)	1,701	51,042	2,632	—	—	93.69
ADULT	379	11,370	15,792	—	—	78.62
SONIC [39]						
MNIST	2,740,000	27,000	NA	0.256	>100	99
HAR	1,100,000	12,500	NA	0.256	>100	88

The CPU does not benefit from MNIST binarization as it still performs 64-bit integer multiplication.

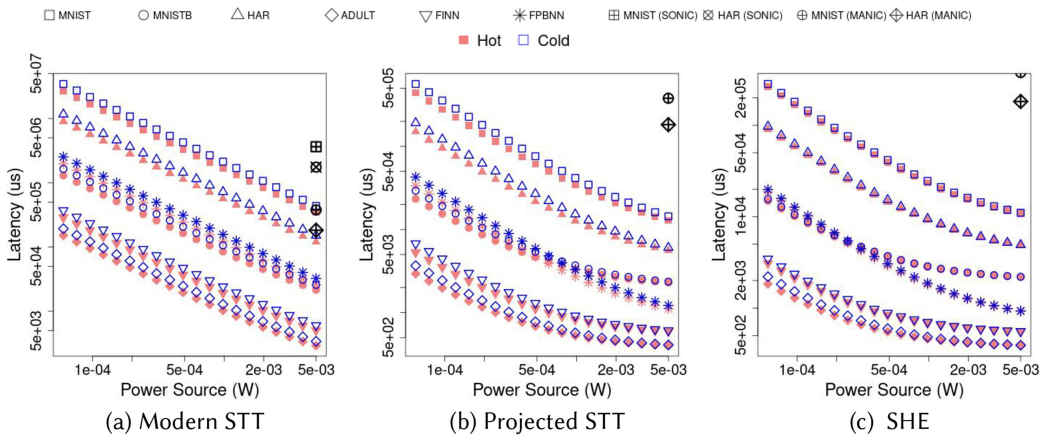

Fig. 10. Latency (µs) vs. Power Source (W) for each MOUSE configuration and SONIC [39]. MOUSE at hot temperature is shown in Red/Filled shapes and at cold temperature is shown in Blue/empty shapes.

consumption and assume it operates at its idle power. Overall, MOUSE has a significant energy efficiency advantage and a competitive latency. Notably, MOUSE consumes more memory than SONIC. However, this is reasonable as MOUSE consists nearly entirely of non-volatile memory, which has high density. MOUSE does not require external logic or area costly volatile memory.

Intermittent Operation: We now evaluate MOUSE with intermittent computation, where a small power source is charging a capacitor that MOUSE can draw energy from. The latency (including time powered off) of all benchmarks with each MTJ device (and different operating temperatures) over the range of power sources (60 µW–5 mW) is plotted in Figure 10, along with a comparison to SONIC [39] and MANIC [40]. All MOUSE configurations are able to significantly

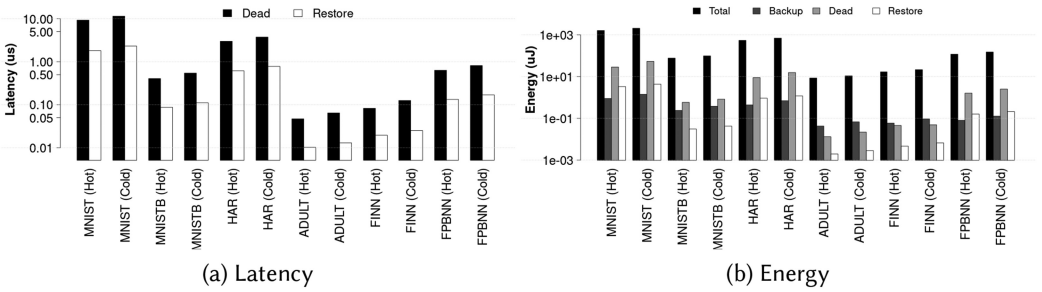

Fig. 11. Latency/Energy Breakdown: Modern STT.

outperform SONIC for the same power budget. Despite conservative estimates of MTJ performance, conservative estimates of peripheral circuitry, and very optimistic estimation of MANIC across the temperature range (30% boost in performance and no overhead for temperature or radiation) MOUSE has a similar performance with MANIC. On the MNIST data set, if MOUSE uses 8-bit inputs, then its latency is 0.91× (1.15×) that of MANIC at hot (cold) temperatures. On the HAR data set, MOUSE has a latency that is 0.66× (0.83×) that of MANIC at hot (cold) temperatures. Hence, MOUSE has better performance on average, with better results at warmer temperatures.

At cold temperatures MOUSE has a higher latency than when hot. At 60 µW, overall cold is 23.4% slower on average. While MOUSE has the same clock rate and issues instructions at the same rate, the instructions consume more energy when cold. Hence, MOUSE will run out of energy and have to power off more frequently. Temperature has a varying level of impact on each MTJ technology. Modern STT has a 33.3% higher latency and Projected STT has 28.5% higher latency at cold temperature, across all benchmarks. SHE is less effected by temperature, because write and logic operations use the SHE channel, which is not only more energy-efficient but less affected by temperature. SHE has an 8.6% higher latency across all benchmarks at cold temperature.

Independent of temperature, SHE is the most energy-efficient. Because of this it drains the capacitor less often, and hence has fewer power outages leading to the overall lowest latency. Projected STT has a lower latency than Modern STT, as it can operate at higher frequency (11 ns per instruction versus 33 ns) and it is more energy-efficient.

MOUSE spends negligible amounts of energy while powered off. Hence, the energy consumption is nearly independent of the power supply. The vast majority of the energy is dedicated to normal program execution. A small portion is dedicated to overhead for intermittent execution, which will vary depending on the number of interruptions (which is determined by energy efficiency and the capacitor size). The total energy is plotted in Figure 11(b) for Modern STT; in Figure 12(b) for Projected STT; and in Figure 13(b) for SHE; assuming a 60 µW power source.

There are metrics specific to beyond-edge devices that indicate how efficient the checkpointing strategy is [115]. In addition to the total energy, we report the Backup energy, Dead energy, and the Restore energy. Backup refers to any actions required prior to shutdown to save the state. For more traditional architectures, this involves writing data back to non-volatile memory. For MOUSE, the only backup operations are saving the PC, flipping the parity bit, and writing values into the CBR (to indicate which columns are active). MOUSE does the first two on every instruction, and the second only AC instructions. Dead refers to any computation that must be re-performed after restart (which was lost due to the shutdown). For MOUSE, this is at most re-performing the very last instruction. Restart is any actions required to put the device back into operating condition after a shutdown. For MOUSE, this is the re-activation of columns with an AC instruction.

Backup has no associated latency, as MOUSE's backup operations are overlapped with normal program execution. However, we do report Dead latency, which is the time it takes to re-perform

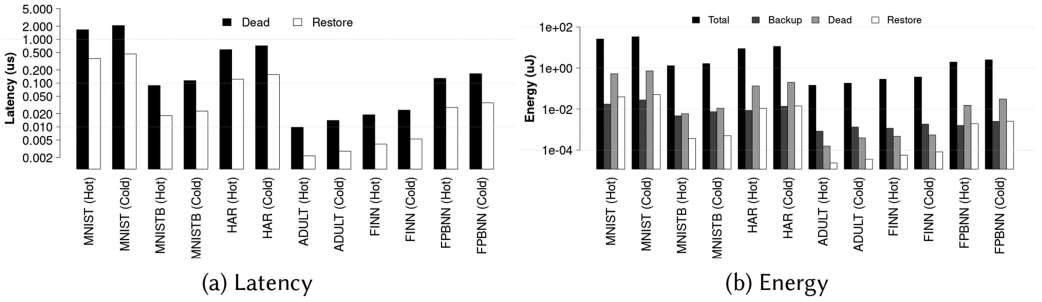

Fig. 12. Latency/Energy Breakdown: Projected STT.

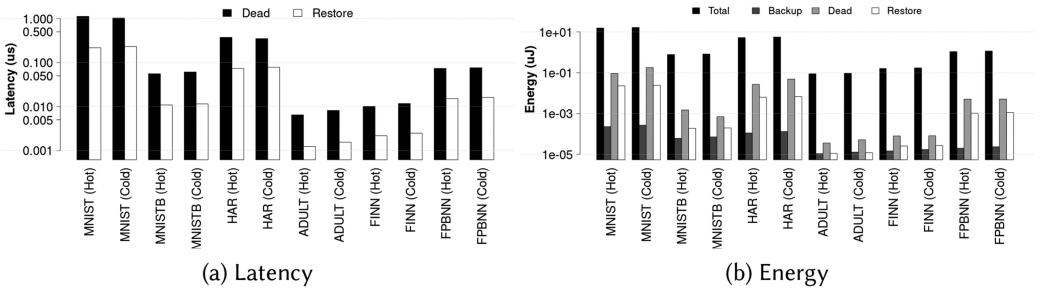

Fig. 13. Latency/Energy Breakdown: SHE.

the last instruction, and the Restore latency, which is the time it takes to re-activate columns. To remain efficient, the Backup, Restore, and Dead latency and energy should be low.

Overhead for Backup, Restore, and Dead are reported in Figure 11 for Modern STT; in Figure 12, for Projected STT; and in Figure 13, for SHE. Note that the y-axis is log scale. The total energy encapsulates all energy used for computation, as well as Backup, Restore, and Dead energy. Also note the total latency is provided for all architectures in Figure 10—where the breakdown figures capture the data for the 60 µW power source.

The overheads for Backup, Dead, and Restore increase with cold temperature. This is for two reasons. The first is that the actions required for each will cost more energy due to the MTJ characteristics. For exampling, writing the PC value or re-performing the last instruction will involve MTJ operations, which will take more energy at cold temperatures. The second reason is that the overall lower energy efficiency at cold temperatures leads to more power outages. At cold temperature, across all benchmarks and technologies, MOUSE restarts 24.4% more often than at hot temperatures. This increases the number of instructions that need to be re-performed and the number of times architectural state variables will be saved.

Modern STT is the least energy-efficient, which means it must restart the most. Because of this it has the largest relative Dead energy. At the extremely low power of 60 µW, on average, across all benchmarks, Dead energy is 0.98% (1.09%) of the total energy at hot (cold) temperature. The projected MTJs have lower overhead, where Dead energy (on average) becomes 0.796% (0.804%) of the energy for Projected STT and 0.194% (0.323) of the total for SHE at hot (cold) temperatures. Dead latency, however, is 0.068% (0.084%) of the total for Modern STT, 0.040% (0.050%) of the total for Projected STT, and 0.020% (0.020%) of the total for SHE with hot (cold) temperatures. Restore is only 0.013% (0.016%) of the latency and 0.066% (0.069%) of the energy for Modern STT; 0.008% (0.010%) of the latency and 0.048% (0.049%) of the energy for Projected STT; and 0.0037% (0.0040%)

of the latency and 0.0436% (0.0436%) of the energy for SHE with hot (cold) temperatures. As Restore latency and energy is due to peripheral circuitry, SHE has no advantage over STT for an individual restart. However, SHE still requires fewer restart operations due to its overall increased energy efficiency. Backup energy is, on average across all benchmarks, 0.304% (0.337%) for Modern STT; 0.350% (0.340%) for Projected STT; and 0.009% (0.009%) for SHE. Backup has no associated latency as it is performed at the same time as each instruction on every cycle. Overall the Backup, Dead, and Restore overheads increase only modestly at cold temperatures. Hence, the checkpointing mechanisms remain efficient across the wide temperature range and MOUSE is suitable for use as an intermittent accelerator in the harsh environments of LEO.

Restore and Dead latency and energy are all zero for the case of a continuously powered system. This is because there are no power outages and, hence, never a need to restart the system or re-perform any potentially unfinished instructions.

CMOS hardening decreases efficiency due the peripheral circuitry consuming more energy. The overhead varies across benchmarks and technologies, but tends to be higher at lower power (due to requiring more restarts, which incurs re-activation of the peripheral circuitry) and higher temperature (due to peripheral circuitry having a larger percentage share of the total energy). At 60 μW, the lowest power tested, CMOS hardening increases energy consumption by 26.9% on average (44.8% at worst) in cold and by 32.3% on average (49.2% at worst) at hot temperature.

8 RELATED WORK

Orbital Edge Computing was proposed by Denby and Lucia [26, 27] as a new model for satellite computation. The authors describe architectures for computational nanosatellites. Additionally, they proposed a strategy called the computational nanosatellite pipeline, which parallelizes computation across collections of satellites to reduce latency. MOUSE could be used as a sub-component within such computational satellites.

Traditional architectures have been significantly modified to be intermittent safe. A strategy has been to tightly integrate non-volatile memory with volatile registers to enable a fast and more efficient backup process just prior to shutdown. These architectures are known as **non-volatile processors (NVP)** [80, 88]. A system utilizing a THU1010N non-volatile processor was analyzed, where trade-offs in checkpointing strategies are evaluated [80]. Follow up work has increased the resilience of NVPs to power interruptions [86, 87]. The NVP in Reference [86] can complete the FFT benchmark from MiBench [42] in 4.2 ms. Cilasun et al. [24] evaluated FFT implementations on CRAM, the same PIM substrate that MOUSE uses. Performing a similarly sized problem, the best latency they were able to achieve is 1.63 ms. However, adapting this implementation to be intermittent safe in the same manner in MOUSE would add a latency overhead. PIM has been incorporated into beyond-edge devices previously, using RRAM arrays for acceleration [126]. However, this design still requires a CPU to perform logic and orchestrate control. PIM is only a sub-component of the system, hence the efficient checkpointing strategy of MOUSE cannot be applied to this architecture.

ResiRCA [101] uses an adaptable RRAM crossbar accelerator for MAC (multiply+accumulate) operations for CNNs. The architecture is able to adapt to varying levels of input power to efficiently utilize the PIM components. However, a battery is required to maintain an external controller. Additionally, a significant amount of computation occurs outside the memory array (only MACs are processed by the memory). Hence, the MOUSE's checkpointing mechanism is also not applicable to this architecture. Many RRAM accelerators have been developed [127, 129, 144, 148]. However, these architectures only use the RRAM array as an accelerator for specific operations. The full system contains much additional circuitry and logic in addition to the memory arrays. This

significantly increases the difficulty to adapt to intermittent processing. Additionally, they require analog-to-digital converters for every operation, which has a large area and energy overhead.

Capybara [20] uses a re-configurable hardware energy storage mechanism and a software interface that allows the specification of energy needs for different tasks. This gives the system more flexibility in satisfying the requirements of different kinds of tasks. In this work, we assumed a constant capacitor size, however, Capybara could enable variable size energy buffers to more closely match the requirements of each application.

Hibernus [8] is a system that reactively hibernates and wakes up. This is a similar shutdown policy to MOUSE. However, Hibernus performs an additional back-up operation before shutting down, whereas MOUSE does not need to.

Many strategies have proposed to enable more traditional systems to operate intermittently. CleanCut [19] works with LLVM to compile programs with checkpoints, and uses a statistical energy model to find potential non-terminating paths. Chinchilla [90] uses adaptive checkpointing, where the frequency of checkpoints is a function of the number of interrupts. Coati [111] developed methods to ensure correctness of concurrent threads in the presence of interrupts for intermittent systems. The What's Next Intermittent Architecture [35] uses approximation to improve performance. Rather than following an all-or-nothing approach, What's Next computes approximate results and continually improves the output. If an acceptable output is achieved, then it will skip to processing the next input. This enables the device to process more inputs as it does not waste time and energy achieving unnecessary accuracy.

The EH model [115] is a design space exploration tool for energy-harvesting architectures. As noted by the authors, energy-harvesting systems can generally be divided into two types, (1) multi-backup, which perform many backups between power outages, and (2) single back-up, which only save state once before a power outage. Multi-backup systems include Mementos, [104], DINO [84], Chain [18], Alpaca [89], Mayfly [48], Ratchet [136], and Clank [49]. Single-backup systems include Hibernus [7], QuickRecall [56], and many others [5, 6, 12, 79, 85]. MOUSE is a multi-backup system as it is constantly saving the architectural state.

Many PIM architectures exist, such as Pinatubo [76], for DRAM with Ambit [122], and for SRAM with Neural Cache [32]. These technologies target traditional memory hierarchies and have not considered intermittent operation. Ambit and Neural Cache are not suitable for energy harvesting as they are volatile technologies. Pinatubo has the potential to be adapted and used similarly as CRAM in MOUSE. However, Pinatubo uses logic external to the memory array for some operations. This adds complexity, which is difficult to manage during intermittent execution. Additionally, Pinatubo requires sense amplifiers for every operation, which comes with a high energy cost.

Neural networks [15, 143] and BNNs [130, 149] have been previously mapped to PIM substrates for acceleration, including on CRAM [108]. However, such designs have not considered intermittent computing and would be unsuitable for the beyond-edge domain.

A number of high performance and low power accelerator exist, but which have not been adapted for intermittent execution. The Phoenix processor [121] is an extremely low power processor with a sophisticated sleep strategy. A number of accelerators have demonstrated high performance and energy efficiency on inference. PuDianNao [78] is an ASIC accelerator that also targets SVM. The XNOR Neural engine is microcontroller-based system for BNN acceleration [22]. An in/near memory SRAM substrate is proposed in Reference [138], which performs bit-serial arithmetic, and which was shown to have high performance and efficiency on the AlexNet [69] network. A number of PIM accelerators also exist, including a BNN accelerator for Cifar-10 image classification [57], an analog SRAM accelerator for MNIST classification [155], and another that does both MNIST and Cifar-10 classification [135]. Adapting such accelerators for intermittency

is not straight-forward and would likely come, if at all possible, at significant performance and efficiency cost.

Orthogonal to our work, recent papers have made progress on problems relevant in the energy-harvesting domain. Low power and accurate time keeping was developed in Reference [25]. SRAM was used as an efficient check-pointing memory, being able to maintain state for short periods of power off time [141]. A new platform for intermittent computing is proposed in Reference [68], which simplifies the task of adapting pre-existing embedded applications to work in intermittent environments.

9 CONCLUSION

In this article, we improve the hardware efficiency and programmability of MOUSE [107], a non-volatile PIM accelerator for beyond-edge computing to to enable orbital deployment. Specifically, we expand the PIM instruction set and add architectural support for branch instructions for enhanced programmability. We develop more efficient mechanisms for column activation, reducing the complexity of the peripheral circuitry. We show that MTJ devices and supporting CMOS circuitry operate correctly across a wide temperature range. Even accounting for the overhead to maintain resilience against radiation, this advanced architecture features high performance and extreme energy efficiency. Combined with the guarantee for intermittent safe operation and inherent low-cost checkpointing mechanisms, the final result is a design well suited for use as a nanosatellite in low Earth orbit.

REFERENCES

[1] Everspin Technologies. 2019. Retrieved from https://www.everspin.com/supportdocs/EMD3D256M08G1-150CBS1.

[2] Everspin Technologies. 2019. Retrieved from https://www.everspin.com/family/emd4e001g?npath=3557.

[3] Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi. 2015. A scalable processing-in-memory accelerator for parallel graph processing. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*. 105–117.

[4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21st European Symposium on Artificial Neural Networks (ESANN'13)*.

[5] Faycal Ait Aouda, Kevin Marquet, and Guillaume Salagnac. 2014. Incremental checkpointing of program state to NVRAM for transiently powered systems. In *Proceedings of the 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC'14)*. IEEE, 1–4.

[6] Domenico Balsamo, Anup Das, Alex S. Weddell, Davide Brunelli, Bashir M. Al-Hashimi, Geoff V. Merrett, and Luca Benini. 2016. Graceful performance modulation for power-neutral transient computing systems. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 35, 5 (2016), 738–749.

[7] Domenico Balsamo, Alex S. Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M. Al-Hashimi, Geoff V. Merrett, and Luca Benini. 2016. Hibernus++: A self-calibrating and adaptive system for transiently powered embedded devices. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 35, 12 (2016), 1968–1980.

[8] Domenico Balsamo, Alex S. Weddell, Geoff V. Merrett, Bashir M. Al-Hashimi, Davide Brunelli, and Luca Benini. 2014. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embed. Syst. Lett.* 7, 1 (2014), 15–18.

[9] Mark Barton and Jon Miller. 2005. Modular thermal design concepts: Thermal design of a spacecraft on a module level for LEO missions.

[10] Robert Baumann. 2005. Soft errors in advanced computer systems. *IEEE Design Test Comput.* 22, 3 (2005), 258–266.

[11] Todd Bayer, Brent Buffington, Jean-Francois Castet, Maddalena Jackson, Gene Lee, Kari Lewis, Jason Kastner, Kathy Schimmels, and Karen Kirby. 2017. Europa mission update: Beyond payload selection. In *Proceedings of the IEEE Aerospace Conference*. IEEE, 1–12.

[12] Gautier Berthou, Tristan Delizy, Kevin Marquet, Tanguy Risset, and Guillaume Salagnac. 2017. Peripheral state persistence for transiently-powered systems. In *Proceedings of the Global Internet of Things Summit (GIoTS'17)*. IEEE, 1–6.

[13] Anantha P. Chandrakasan, Denis C. Daly, Joyce Kwong, and Yogesh K. Ramadass. 2008. Next generation micro-power systems. In *Proceedings of the IEEE Symposium on VLSI Circuits*. IEEE, 2–5.

[14] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2, 3 (2011), 27.

[15] Lerong Chen, Jiawen Li, Yiran Chen, Qiuping Deng, Jiyuan Shen, Xiaoyao Liang, and Li Jiang. 2017. Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar. In *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 19–24.

[16] Xizi Chen, Jingbo Jiang, Jingyang Zhu, and Chi-Ying Tsui. 2018. A high-throughput and energy-efficient RRAM-based convolutional neural network using data encoding and dynamic quantization. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC'18)*. IEEE, 123–128.

[17] Zamshed Chowdhury, Jonathan D. Harms, S. Karen Khatamifard, Masoud Zabihi, Yang Lv, Andrew P. Lyle, Sachin S. Sapatnekar, Ulya R. Karpuzcu, and Jian-Ping Wang. 2017. Efficient in-memory processing using spintronics. *IEEE Comput. Architect. Lett.* 17, 1 (2017), 42–46.

[18] Alexei Colin and Brandon Lucia. 2016. Chain: Tasks and channels for reliable intermittent programs. In *ACM SIGPLAN Notices*, Vol. 51. ACM, 514–530.

[19] Alexei Colin and Brandon Lucia. 2018. Termination checking and task decomposition for task-based intermittent programs. In *Proceedings of the 27th International Conference on Compiler Construction*. ACM, 116–127.

[20] Alexei Colin, Emily Ruppel, and Brandon Lucia. 2018. A reconfigurable energy storage architecture for energy-harvesting devices. In *ACM SIGPLAN Notices*, Vol. 53. ACM, 767–781.

[21] Y. Conraux, J. P. Nozieres, V. Da Costa, M. Toulemonde, and K. Ounadjela. 2003. Effects of swift heavy ion bombardment on magnetic tunnel junction functional properties. *J. Appl. Phys.* 93, 10 (2003), 7301–7303.

[22] Francesco Conti, Pasquale Davide Schiavone, and Luca Benini. 2018. XNOR neural engine: A hardware accelerator IP for 21.6-fJ/op binary neural network inference. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 37, 11 (2018), 2940–2951.

[23] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1. Retrieved from https://arXiv:1602.02830.

[24] Hüsrev Cilasun, Salonik Resch, Zamshed Iqbal Chowdhury, Erin Olson, Masoud Zabihi, Zhengyang Zhao, Thomas Peterson, Jian-Ping Wang, Sachin S. Sapatnekar, and Ulya Karpuzcu. 2020. CRAFFT: High resolution FFT accelerator in spintronic computational RAM. In *Proceedings of the 57th Annual ACM/IEEE Design Automation Conference*.

[25] Jasper de Winkel, Carlo Delle Donne, Kasim Sinan Yildirim, Przemysław Pawełczak, and Josiah Hester. 2020. Reliable timekeeping for intermittent computing. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems*. 53–67.

[26] Bradley Denby and Brandon Lucia. 2019. Orbital edge computing: Machine inference in space. *IEEE Comput. Architect. Lett.* 18, 1 (2019), 59–62.

[27] Bradley Denby and Brandon Lucia. 2020. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems*. 939–954.

[28] Zhitao Diao, Alex Panchula, Yunfei Ding, Mahendra Pakala, Shengyuan Wang, Zhanjie Li, Dmytro Apalkov, Hideyasu Nagai, Alexander Driskill-Smith, Lien-Chang Wang, et al. 2007. Spin transfer switching in dual MgO magnetic tunnel junctions. *Appl. Phys. Lett.* 90, 13 (2007), 132508.

[29] Xiangyu Dong, Xiaoxia Wu, Guangyu Sun, Yuan Xie, Helen Li, and Yiran Chen. 2008. Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement. In *Proceedings of the 45th ACM/IEEE Design Automation Conference*. IEEE, 554–559.

[30] Xiangyu Dong, Cong Xu, Yuan Xie, and Norman P. Jouppi. 2012. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 31, 7 (2012), 994–1007.

[31] Philipp Dürrenfeld, Felicitas Gerhard, Jonathan Chico, Randy K. Dumas, Mojtaba Ranjbar, Anders Bergman, Lars Bergqvist, Anna Delin, Charles Gould, Laurens W. Molenkamp, et al. 2015. Tunable damping, saturation magnetization, and exchange stiffness of half-Heusler NiMnSb thin films. *Phys. Rev. B* 92, 21 (2015), 214424.

[32] Charles Eckert, Xiaowei Wang, Jingcheng Wang, Arun Subramaniyan, Ravi Iyer, Dennis Sylvester, David Blaauw, and Reetuparna Das. 2018. Neural cache: Bit-serial in-cache acceleration of deep neural networks. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*. IEEE Press, 383–396.

[33] Patrick R. Fleming, Brian D. Olson, W. Timothy Holman, Bharat L. Bhuva, and Lloyd W. Massengill. 2008. Design technique for mitigation of soft errors in differential switched-capacitor circuits. *IEEE Trans. Circ. Syst. II: Express Briefs* 55, 9 (2008), 838–842.

[34] Karthik Ganesan, Joshua San Miguel, and Natalie Enright Jerger. 2019. The what's next intermittent computing architecture. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA'19)*. IEEE, 211–223.

[35] Karthik Ganesan, Joshua San Miguel, and Natalie Enright Jerger. 2019. The what's next intermittent computing architecture. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA'19)*. IEEE, 211–223.

[36] Kevin Garello, Farrukh Yasin, S. Couet, Laurent Souriau, J. Swerts, S. Rao, Simon Van Beek, Wonsub Kim, Enlong Liu, S. Kundu, et al. 2018. SOT-MRAM 300mm integration for low power and ultrafast embedded memories. In *Proceedings of the IEEE Symposium on VLSI Circuits*. IEEE, 81–82.

[37] Simone Gerardin and Alessandro Paccagnella. 2010. Present and future non-volatile memories for space. *IEEE Trans. Nuclear Sci.* 57, 6 (2010), 3016–3039.

[38] Graham Gobieski, Nathan Beckmann, and Brandon Lucia. 2018. Intermittent Deep Neural Network Inference.

[39] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence beyond the edge: Inference on intermittent embedded systems. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*. 199–213.

[40] Graham Gobieski, Amolak Nagi, Nathan Serafin, Mehmet Meric Isgenc, Nathan Beckmann, and Brandon Lucia. 2019. Manic: A vector-dataflow architecture for ultra-low-power embedded systems. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 670–684.

[41] Hayit Greenspan, Bram Van Ginneken, and Ronald M. Summers. 2016. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Trans. Med. Imag.* 35, 5 (2016), 1153–1159.

[42] Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, and Richard B. Brown. 2001. MiBench: A free, commercially representative embedded benchmark suite. In *Proceedings of the 4th Annual IEEE International Workshop on Workload Characterization (WWC'01)*. IEEE, 3–14.

[43] Ramesh Harjani and Saurabh Chaubey. 2014. A unified framework for capacitive series-parallel DC-DC converter design. In *Proceedings of the IEEE Custom Integrated Circuits Conference*. IEEE, 1–8.

[44] John L. Hennessy and David A. Patterson. 2011. *Computer Architecture: A Quantitative Approach*. Elsevier.

[45] Josiah Hester, Travis Peters, Tianlong Yun, Ronald Peterson, Joseph Skinner, Bhargav Golla, Kevin Storer, Steven Hearndon, Kevin Freeman, Sarah Lord, et al. 2016. Amulet: An energy-efficient, multi-application wearable platform. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM, 216–229.

[46] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. 2015. Tragedy of the coulombs: Federating energy storage for tiny, intermittently-powered sensors. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 5–16.

[47] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 19.

[48] Josiah Hester, Kevin Storer, and Jacob Sorber. 2017. Timely execution on intermittently powered batteryless sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 17.

[49] Matthew Hicks. 2017. Clank: Architectural support for intermittent computation. In *Proceedings of the ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA'17)*. IEEE, 228–240.

[50] Tifenn Hirtzlin, Bogdan Penkovsky, Jacques-Olivier Klein, Nicolas Locatelli, Adrien F. Vincent, Marc Bocquet, Jean-Michel Portal, and Damien Querlioz. 2019. Implementing binarized neural networks with magnetoresistive ram without error correction. In *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH'19)*. IEEE, 1–5.

[51] G. Hu, J. H. Lee, J. J. Nowak, J. Z. Sun, J. Harms, A. Annunziata, S. Brown, W. Chen, Y. H. Kim, G. Lauer, et al. 2015. STT-MRAM with double magnetic tunnel junctions. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM'15)*. IEEE, 26–3.

[52] Xiao-Di Huang et al. 2020. Forming-free, fast, uniform, and high endurance resistive switching from cryogenic to high temperatures in W/AlO x/Al 2 O 3/Pt bilayer memristor. *IEEE Electron Device Lett.* 41, 4 (2020), 549–552.

[53] Harold Hughes, Konrad Bussmann, Patrick J. McMarr, Shu-Fan Cheng, Robert Shull, Andrew P. Chen, Simon Schafer, Tim Mewes, Adrian Ong, Eugene Chen, et al. 2012. Radiation studies of spin-transfer torque materials and devices. *IEEE Trans. Nuclear Sci.* 59, 6 (2012), 3027–3033.

[54] Kyuyeon Hwang and Wonyong Sung. 2014. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS'14)*. IEEE, 1–6.

[55] Guenole Jan, Luc Thomas, Son Le, Yuan-Jen Lee, Huanlong Liu, Jian Zhu, Ru-Ying Tong, Keyu Pi, Yu-Jen Wang, Dongna Shen, et al. 2014. Demonstration of fully functional 8Mb perpendicular STT-MRAM chips with sub-5ns writing for non-volatile embedded memories. In *Proceedings of the Symposium on VLSI Technology: Digest of Technical Papers*. IEEE, 1–2.

[56] Hrishikesh Jayakumar, Arnab Raha, and Vijay Raghunathan. 2014. QuickRecall: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers. In *Proceedings of the 27th International Conference on VLSI Design and 13th International Conference on Embedded Systems*. IEEE, 330–335.

[57] Hongyang Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma. 2019. A programmable embedded microprocessor for bit-scalable in-memory computing. In *Proceedings of the IEEE Hot Chips 31 Symposium (HCS'19)*. IEEE, 1–29.

[58] Sang-Woo Jun, Ming Liu, Sungjin Lee, Jamey Hicks, John Ankcorn, Myron King, Shuotao Xu, et al. 2015. Bluedbm: An appliance for big data analytics. In *Proceedings of the ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA'15)*. IEEE, 1–13.

[59] Wanyeong Jung, Sechang Oh, Suyoung Bang, Yoonmyung Lee, Dennis Sylvester, and David Blaauw. 2014. 23.3 A 3nW fully integrated energy harvester based on self-oscillating switched-capacitor DC-DC converter. In *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC'14)*. IEEE, 398–399.

[60] Yangwook Kang, Yang-suk Kee, Ethan L. Miller, and Chanik Park. 2013. Enabling cost-effective data processing with smart SSD. In *Proceedings of the IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST'13)*. IEEE, 1–12.

[61] Holger Kappert, Norbert Kordas, Stefan Dreiner, Uwe Paschen, and Rainer Kokozinski. 2015. High temperature SOI CMOS technology and circuit realization for applications up to 300°C. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'15)*. IEEE, 1162–1165.

[62] Sungchan Kim, Hyunok Oh, Chanik Park, Sangyeun Cho, Sang-Won Lee, and Bongki Moon. 2016. In-storage processing of database scans and joins. *Info. Sci.* 327 (2016), 183–200.

[63] Sangkil Kim, Rushi Vyas, Jo Bito, Kyriaki Niotaki, Ana Collado, Apostolos Georgiadis, and Manos M. Tentzeris. 2014. Ambient RF energy-harvesting technologies for self-sustainable standalone wireless sensor platforms. *Proc. IEEE* 102, 11 (2014), 1649–1666.

[64] Sangkil Kim, Rushi Vyas, Jo Bito, Kyriaki Niotaki, Ana Collado, Apostolos Georgiadis, and Manos M. Tentzeris. 2014. Ambient RF energy-harvesting technologies for self-sustainable standalone wireless sensor platforms. *Proc. IEEE* 102, 11 (2014), 1649–1666.

[65] Daisuke Kobayashi, Kazuyuki Hirose, Takahiro Makino, Shinobu Onoda, Takeshi Ohshima, Shoji Ikeda, Hideo Sato, Eli Christopher Inocencio Enobio, Tetsuo Endoh, and Hideo Ohno. 2017. Soft errors in 10-nm-scale magnetic tunnel junctions exposed to high-energy heavy-ion radiation. *Japan. J. Appl. Phys.* 56, 8 (2017), 0802B4.

[66] Daisuke Kobayashi, Yuya Kakehashi, Kazuyuki Hirose, Shinobu Onoda, Takahiro Makino, Takeshi Ohshima, Shoji Ikeda, Michihiko Yamanouchi, Hideo Sato, Eli Christopher Enobio, et al. 2014. Influence of heavy ion irradiation on perpendicular-anisotropy CoFeB-MgO magnetic tunnel junctions. *IEEE Trans. Nuclear Sci.* 61, 4 (2014), 1710–1716.

[67] Ron Kohavi. 1996. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, Vol. 96. Citeseer, 202–207.

[68] Vito Kortbeek, Kasim Sinan Yildirim, Abu Bakar, Jacob Sorber, Josiah Hester, and Przemysław Pawełczak. 2020. Time-sensitive intermittent computing meets legacy software. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems*. 85–99.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

[70] Jeetendra Kumar. 2005. Use of pass transistor logic to minimize the impact of soft errors in combinational circuits. In *Proceedings of the Workshop on System Effects of Logic Soft Errors*.

[71] Lili Lang et al. 2020. A low temperature functioning CoFeB/MgO-based perpendicular magnetic tunnel junction for cryogenic nonvolatile random access memory. *Appl. Phys. Lett.* 116, 2 (2020).

[72] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[73] S. H. Lee, H. C. Park, M. S. Kim, H. W. Kim, M. R. Choi, H. G. Lee, J. W. Seo, S. C. Kim, S. G. Kim, S. B. Hong, et al. 2011. Highly productive PCRAM technology platform and full chip operation: Based on 4F 2 (84nm pitch) cell scheme for 1 Gb and beyond. In *Proceedings of the International Electron Devices Meeting*. IEEE, 3–3.

[74] LEO Temperatures 2021. Retrieved from https://www.oreilly.com/library/view/diy-satellite-platforms/9781449312756/ch01s05.html.

[75] Vladimir Leonov. 2013. Thermoelectric energy harvesting of human body heat for wearable sensors. *IEEE Sensors J.* 13, 6 (2013), 2284–2291.

[76] Shuangchen Li, Cong Xu, Qiaosha Zou, Jishen Zhao, Yu Lu, and Yuan Xie. 2016. Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 173.

[77] Shuang Liang, Shouyi Yin, Leibo Liu, Wayne Luk, and Shaojun Wei. 2018. FP-BNN: Binarized neural network on FPGA. *Neurocomputing* 275 (2018), 1072–1086.

[78] Daofu Liu, Tianshi Chen, Shaoli Liu, Jinhong Zhou, Shengyuan Zhou, Olivier Teman, Xiaobing Feng, Xuehai Zhou, and Yunji Chen. 2015. Pudiannao: A polyvalent machine learning accelerator. In *ACM SIGARCH Computer Architecture News*, Vol. 43. ACM, 369–381.

[79] Qingrui Liu and Changhee Jung. 2016. Lightweight hardware support for transparent consistency-aware checkpointing in intermittent energy-harvesting systems. In *Proceedings of the 5th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*. IEEE, 1–6.

[80] Yongpan Liu, Zewei Li, Hehe Li, Yiqun Wang, Xueqing Li, Kaisheng Ma, Shuangchen Li, Meng-Fan Chang, Sampson John, Yuan Xie, et al. 2015. Ambient energy-harvesting nonvolatile processors: From circuit to system. In *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 150.

[81] R. Lowther, W. Morris, D. Gifford, D. Duff, and R. Fuller. 2011. Latchup immunity in high temperature bulk CMOS devices. In *Proceedings of the Additional Conferences (Device Packaging, HiTEC, HiTEN, and CICMT)*. 000215–000220.

[82] Brandon Lucia, Vignesh Balaji, Alexei Colin, Kiwan Maeng, and Emily Ruppel. 2017. Intermittent computing: Challenges and opportunities. In *Proceedings of the 2nd Summit on Advances in Programming Languages (SNAPL'17)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[83] Brandon Lucia, Brad Denby, Zachary Manchester, Harsh Desai, Emily Ruppel, and Alexei Colin. 2021. Computational nanosatellite constellations: Opportunities and challenges. *GetMobile: Mobile Comput. Commun.* 25, 1 (2021), 16–23.

[84] Brandon Lucia and Benjamin Ransford. 2015. A simpler, safer programming and execution model for intermittent systems. In *ACM SIGPLAN Notices*, Vol. 50. ACM, 575–585.

[85] Giedrius Lukosevicius, Alberto Rodriguez Arreola, and Alex S. Weddell. 2017. Using sleep states to maximize the active time of transient computing systems. In *Proceedings of the 5th ACM International Workshop on Energy-Harvesting and Energy-Neutral Sensing Systems*. ACM, 31–36.

[86] Kaisheng Ma, Xueqing Li, Jinyang Li, Yongpan Liu, Yuan Xie, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2017. Incidental computing on IoT nonvolatile processors. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'17)*. IEEE, 204–218.

[87] Kaisheng Ma, Xueqing Li, Huichu Liu, Xiao Sheng, Yiqun Wang, Karthik Swaminathan, Yongpan Liu, Yuan Xie, John Sampson, and Vijaykrishnan Narayanan. 2017. Dynamic power and energy management for energy-harvesting nonvolatile processor systems. *ACM Trans. Embedded Comput. Syst.* 16, 4 (2017), 1–23.

[88] Kaisheng Ma, Yang Zheng, Shuangchen Li, Karthik Swaminathan, Xueqing Li, Yongpan Liu, Jack Sampson, Yuan Xie, and Vijaykrishnan Narayanan. 2015. Architecture exploration for ambient energy-harvesting nonvolatile processors. In *Proceedings of the IEEE 21st International Symposium on High Performance Computer Architecture (HPCA'15)*. IEEE, 526–537.

[89] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2017. Alpaca: Intermittent execution without checkpoints. *Proc. ACM Program. Lang.* 1 (2017), 96.

[90] Kiwan Maeng and Brandon Lucia. 2018. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI'18)*. 129–144.

[91] Milos Manic, Kasun Amarasinghe, Juan J. Rodriguez-Andina, and Craig Rieger. 2016. Intelligent buildings of the future: Cyberaware, deep learning powered, and human interacting. *IEEE Industr. Electron. Mag.* 10, 4 (2016), 32–49.

[92] David G. Mavis and Paul H. Eaton. 2002. Soft error rate mitigation techniques for modern microcircuits. In *Proceedings of the 40th Annual IEEE International Reliability Physics Symposium*. IEEE, 216–225.

[93] S. Mizukami, D. Watanabe, M. Oogane, Y. Ando, Y. Miura, M. Shirai, and T. Miyazaki. 2009. Low damping constant for Co2 FeAl Heusler alloy films and its correlation with density of states. *J. Appl. Phys.* 105, 7 (2009), 07D306.

[94] Eric Arturo Montoya, Jen-Ru Chen, Randy Ngelale, Han Kyu Lee, Hsin-Wei Tseng, Lei Wan, En Yang, Patrick Braganca, Ozdal Boyraz, Nader Bagherzadeh, et al. 2020. Immunity of nanoscale magnetic tunnel junctions with perpendicular magnetic anisotropy to ionizing radiation. *Sci. Rep.* 10, 1 (2020), 1–8.

[95] Michael Nicolaidis. 1999. Time redundancy based soft-error tolerance to rescue nanometer technologies. In *Proceedings of the 17th IEEE VLSI Test Symposium*. IEEE, 86–94.

[96] Hiroki Noguchi, Kazutaka Ikegami, Keiichi Kushida, Keiko Abe, Shogo Itai, Satoshi Takaya, Naoharu Shimomura, Junichi Ito, Atsushi Kawasumi, Hiroyuki Hara, et al. 2015. 7.5 A 3.3 ns-access-time 71.2 μW/MHz 1Mb embedded STT-MRAM using physically eliminated read-disturb scheme and normally-off memory architecture. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'15)*. IEEE, 1–3.

[97] Fabian Oboril, Rajendra Bishnoi, Mojtaba Ebrahimi, and Mehdi B. Tahoori. 2015. Evaluation of hybrid memory technologies using SOT-MRAM for on-chip cache hierarchy. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 34, 3 (2015), 367–380.

[98] Roystein Oliveira, Aditya Jagirdar, and Tapan J. Chakraborty. 2007. A TMR scheme for SEU mitigation in scan flip-flops. In *Proceedings of the 8th International Symposium on Quality Electronic Design (ISQED'07)*. IEEE, 905–910.

[99] Bishnu Patra et al. 2017. Cryo-CMOS circuits and systems for quantum computing applications. *IEEE J. Solid-State Circ.* 53, 1 (2017).

[100] J. Thomas Pawlowski. 2011. Hybrid memory cube (HMC). In *Proceedings of the IEEE Hot Chips 23 Symposium (HCS'11)*. IEEE, 1–24.

[101] Keni Qiu, Nicholas Jao, Mengying Zhao, Cyan Subhra Mishra, Gulsum Gudukbay, Sethu Jose, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2020. ResiRCA: A resilient energy-harvesting ReRAM crossbar-based accelerator for intelligent embedded processors. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA'20)*. IEEE, 315–327.

[102] R Core Team. 2016. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. Retrieved from https://www.R-project.org/.

[103] Yogesh K. Ramadass and Anantha P. Chandrakasan. 2007. Voltage scalable switched capacitor DC-DC converter for ultra-low-power on-chip applications. In *Proceedings of the IEEE Power Electronics Specialists Conference*. IEEE, 2353–2359.

[104] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: System support for long-running computation on RFID-scale devices. In *ACM SIGARCH Computer Architecture News*, Vol. 39. ACM, 159–170.

[105] Fanghui Ren, Albrecht Jander, Pallavi Dhagat, and Cathy Nordman. 2012. Radiation tolerance of magnetic tunnel junctions with MgO tunnel barriers. *IEEE Trans. Nuclear Sci.* 59, 6 (2012), 3034–3038.

[106] Salonik Resch, Husrev Cilasun, and Ulya Karpuzcu. 2021. Cryogenic PIM: Challenges and opportunities. *IEEE Comput. Architect. Lett.* (2021).

[107] Salonik Resch, S. Karen Khatamifard, Zamshed I. Chowdhury, Masoud Zabihi, Zhengyang Zhao, Husrev Cilasun, Jian-Ping Wang, Sachin S. Sapatnekar, and Ulya R. Karpuzcu. 2020. MOUSE: Inference in non-volatile memory for energy-harvesting applications. In *Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'20)*. IEEE, 400–414.

[108] Salonik Resch, S. Karen Khatamifard, Zamshed Iqbal Chowdhury, Masoud Zabihi, Zhengyang Zhao, Jian-Ping Wang, Sachin S. Sapatnekar, and Ulya R. Karpuzcu. 2019. PIMBALL: Binary neural networks in spintronic memory. *ACM Trans. Architect. Code Optimiz.* 16, 4 (2019), 41.

[109] Siavash Rezaei, Kanghee Kim, and Eli Bozorgzadeh. 2018. Scalable multi-queue data transfer scheme for fpga-based multi-accelerators. In *Proceedings of the IEEE 36th International Conference on Computer Design (ICCD'18)*. IEEE, 374–380.

[110] Emily Ruppel and Brandon Lucia. 2019. Transactional concurrency control for intermittent, energy-harvesting computing systems. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 1085–1100.

[111] Emily Ruppel and Brandon Lucia. 2019. Transactional concurrency control for intermittent, energy-harvesting computing systems. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 1085–1100.

[112] Daisuke Saida, Saori Kashiwada, Megumi Yakabe, Tadaomi Daibou, Naoki Hase, Miyoshi Fukumoto, Shinji Miwa, Yoshishige Suzuki, Hiroki Noguchi, Shinobu Fujita, et al. 2016. Sub-3 ns pulse with sub-100 μA switching of 1x–2x nm perpendicular MTJ for high-performance embedded STT-MRAM towards sub-20 nm CMOS. In *Proceedings of the IEEE Symposium on VLSI Technology*. IEEE, 1–2.

[113] Daisuke Saida, Saori Kashiwada, Megumi Yakabe, Tadaomi Daibou, Naoki Hase, Miyoshi Fukumoto, Shinji Miwa, Yoshishige Suzuki, Hiroki Noguchi, Shinobu Fujita, et al. 2016. Sub-3 ns pulse with sub-100 μA switching of 1x–2x nm perpendicular MTJ for high-performance embedded STT-MRAM towards sub-20 nm CMOS. In *Proceedings of the IEEE Symposium on VLSI Technology*. IEEE, 1–2.

[114] Alanson P. Sample, Daniel J. Yeager, Pauline S. Powledge, Alexander V. Mamishev, and Joshua R. Smith. 2008. Design of an RFID-based battery-free programmable sensing platform. *IEEE Trans. Instrument. Measure.* 57, 11 (2008), 2608–2615.

[115] Joshua San Miguel, Karthik Ganesan, Mario Badr, Chunqiu Xia, Rose Li, Hsuan Hsiao, and Natalie Enright Jerger. 2018. The EH model: Early design space exploration of intermittent processor architectures. In *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'18)*. IEEE, 600–612.

[116] H. Sato, E. C. I. Enobio, M. Yamanouchi, S. Ikeda, S. Fukami, S. Kanai, F. Matsukura, and H. Ohno. 2014. Properties of magnetic tunnel junctions with a MgO/CoFeB/Ta/CoFeB/MgO recording structure down to junction diameter of 11 nm. *Appl. Phys. Lett.* 105, 6 (2014), 062403.

[117] Selahattin Sayil. 2016. *Soft Error Mechanisms, Modeling and Mitigation*. Springer.

[118] Selahattin Sayil. 2019. A survey of circuit-level soft error mitigation methodologies. *Analog Integr. Circ. Signal Process.* 99, 1 (2019), 63–70.

[119] Selahattin Sayil, Archit H. Shah, Md Adnan Zaman, and Mohammad A. Islam. 2015. Soft error mitigation using transmission gate with varying gate and body bias. *IEEE Design Test* 34, 1 (2015), 47–56.

[120] Ronald D. Schrimpf and Daniel M. Fleetwood. 2004. *Radiation Effects and Soft Errors in Integrated Circuits and Electronic Devices*, Vol. 34. World Scientific.

[121] Mingoo Seok, Scott Hanson, Yu-Shiang Lin, Zhiyoong Foo, Daeyeon Kim, Yoonmyung Lee, Nurrachman Liu, Dennis Sylvester, and David Blaauw. 2008. The phoenix processor: A 30pW platform for sensor applications. In *Proceedings of the IEEE Symposium on VLSI Circuits*. IEEE, 188–189.

[122] Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry. 2017. Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 273–287.

[123] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. 2016. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News* 44, 3 (2016), 14–26.

[124] M Shin et al. 2014. Low temperature characterization of 14nm FDSOI CMOS devices. In *Proceedings of the 11th International Workshop on Low Temperature Electronics (WOLTE'14)*.

[125] Gagandeep Singh, Lorenzo Chelini, Stefano Corda, Ahsan Javed Awan, Sander Stuijk, Roel Jordans, Henk Corporaal, and Albert-Jan Boonstra. 2018. A review of near-memory computing architectures: Opportunities and challenges. In *Proceedings of the 21st Euromicro Conference on Digital System Design (DSD'18)*. IEEE, 608–617.

[126] Fang Su, Wei-Hao Chen, Lixue Xia, Chieh-Pu Lo, Tianqi Tang, Zhibo Wang, Kuo-Hsiang Hsu, Ming Cheng, Jun-Yi Li, Yuan Xie, et al. 2017. A 462GOPs/J RRAM-based nonvolatile intelligent processor for energy-harvesting IoE system featuring nonvolatile logics and processing-in-memory. In *Proceedings of the Symposium on VLSI Technology*. IEEE, T260–T261.

[127] Xiaoyu Sun, Xiaochen Peng, Pai-Yu Chen, Rui Liu, Jae-sun Seo, and Shimeng Yu. 2018. Fully parallel RRAM synaptic array for implementing binary neural network with (+1, −1) weights and (+1, 0) neurons. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*. IEEE Press, 574–579.

[128] Tianqi Tang, Lixue Xia, Boxun Li, Yu Wang, and Huazhong Yang. 2017. Binary convolutional neural network on RRAM. In *Proceedings of the 22nd Asia and South Pacific Design Automation Conference (ASP-DAC'17)*. IEEE, 782–787.

[129] Tianqi Tang, Lixue Xia, Boxun Li, Yu Wang, and Huazhong Yang. 2017. Binary convolutional neural network on rram. In *Proceedings of the 22nd Asia and South Pacific Design Automation Conference (ASP-DAC'17)*. IEEE, 782–787.

[130] Tianqi Tang, Lixue Xia, Boxun Li, Yu Wang, and Huazhong Yang. 2017. Binary convolutional neural network on RRAM. In *Proceedings of the 22nd Asia and South Pacific Design Automation Conference (ASP-DAC'17)*. IEEE, 782–787.

[131] Mahdi Torabzadehkashi, Siavash Rezaei, Vladimir Alves, and Nader Bagherzadeh. 2018. Compstor: An in-storage computation platform for scalable distributed processing. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW'18)*. IEEE, 1260–1267.

[132] Mahdi Torabzadehkashi, Siavash Rezaei, Ali Heydarigorji, Hosein Bobarshad, Vladimir Alves, and Nader Bagherzadeh. 2019. Catalina: In-storage processing acceleration for scalable big data analytics. In *Proceedings of the 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'19)*. IEEE, 430–437.

[133] UCI Machine Learning Repository 2019. Retrieved from https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+\using+smartphones.

[134] Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. 2017. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 65–74.

[135] Hossein Valavi, Peter J. Ramadge, Eric Nestler, and Naveen Verma. 2019. A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute. *IEEE J. Solid-State Circ.* 54, 6 (2019), 1789–1799.

[136] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent computation without hardware support or programmer intervention. In *Proceedings of the12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*. 17–32.

[137] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. 2017. Farmbeats: An iot platform for data-driven agriculture. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)*. 515–529.

[138] Jingcheng Wang, Xiaowei Wang, Charles Eckert, Arun Subramaniyan, Reetuparna Das, David Blaauw, and Dennis Sylvester. 2019. A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing. *IEEE J. Solid-State Circ.* 55, 1 (2019), 76–86.

[139] Jian-Ping Wang, Mahdi Jamaliz, Angeline Klemm Smith, and Zhengyang Zhao. 2016. Magnetic tunnel junction based integrated logics and computational circuits. In *Nanomagnetic and Spintronic Devices for Energy-Efficient Memory and Computing*. Wiley, 133.

[140] Qiwen Wang, Xinxin Wang, Seung Hwan Lee, Fan-Hsuan Meng, and Wei D. Lu. 2019. A deep neural network accelerator based on tiled RRAM architecture. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM'19)*. IEEE, 14–4.

[141] Harrison Williams, Xun Jian, and Matthew Hicks. 2020. Forget failure: Exploiting SRAM data remanence for low-overhead intermittent computation. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems*. 69–84.

[142] H.-S. Philip Wong, Heng-Yuan Lee, Shimeng Yu, Yu-Sheng Chen, Yi Wu, Pang-Shiu Chen, Byoungil Lee, Frederick T. Chen, and Ming-Jinn Tsai. 2012. Metal–oxide RRAM. *Proc. IEEE* 100, 6 (2012), 1951–1970.

[143] Lixue Xia, Tianqi Tang, Wenqin Huangfu, Ming Cheng, Xiling Yin, Boxun Li, Yu Wang, and Huazhong Yang. 2016. Switched by input: Power-efficient structure for RRAM-based convolutional neural network. In *Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference (DAC'16)*. IEEE, 1–6.

[144] Lixue Xia, Tianqi Tang, Wenqin Huangfu, Ming Cheng, Xiling Yin, Boxun Li, Yu Wang, and Huazhong Yang. 2016. Switched by input: Power-efficient structure for RRAM-based convolutional neural network. In *Proceedings of the 53rd ACM/EDAC/IEEE Annual Design Automation Conference (DAC'16)*. ACM, 125.

[145] Kodai Yamada, Haruki Maruoka, Jun Furuta, and Kazutoshi Kobayashi. 2018. Sensitivity to soft errors of NMOS and PMOS transistors evaluated by latches with stacking structures in a 65 nm FDSOI process. In *Proceedings of the IEEE International Reliability Physics Symposium (IRPS'18)*. IEEE, P–SE.

[146] Jeng-Bang Yau et al. 2017. Hybrid cryogenic memory cells for superconducting computing applications. In *Proceedings of the 35th International Cosmic Ray Conference (ICRC'17)*.

[147] Mustafa Berke Yelten. [n.d.]. Cryogenic DC characteristics of low threshold voltage (VTH) n-channel MOSFETs. *Balkan J. Electric. Comput. Eng.* 7, 3 ([n.d.]).

[148] Shimeng Yu, Zhiwei Li, Pai-Yu Chen, Huaqiang Wu, Bin Gao, Deli Wang, Wei Wu, and He Qian. 2016. Binary neural network with 16 Mb RRAM macro chip for classification and online training. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM'16)*. IEEE, 16–2.

[149] Shimeng Yu, Zhiwei Li, Pai-Yu Chen, Huaqiang Wu, Bin Gao, Deli Wang, Wei Wu, and He Qian. 2016. Binary neural network with 16 Mb RRAM macro chip for classification and online training. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM'16)*. IEEE, 16–2.

[150] L. Yuan et al. 2006. Temperature dependence of magnetoresistance in magnetic tunnel junctions with different free layer structures. *Phys. Rev. B* 73, 13 (2006).

[151] Masoud Zabihi, Zamshed Iqbal Chowdhury, Zhengyang Zhao, Ulya R. Karpuzcu, Jian-Ping Wang, and Sachin S. Sapatnekar. 2018. In-memory processing on the spintronic CRAM: From hardware design to application mapping. *IEEE Trans. Comput.* 68, 8 (2018), 1159–1173.

[152] Masoud Zabihi, Arvind K. Sharma, Meghna G. Mankalale, Zamshed Iqbal Chowdhury, Zhengyang Zhao, Salonik Resch, Ulya R. Karpuzcu, Jian-Ping Wang, and Sachin S. Sapatnekar. 2020. Analyzing the effects of interconnect parasitics in the STT CRAM in-memory computational platform. *IEEE J. Explor. Solid-State Comput. Dev. Circ.* 6, 1 (2020), 71–79.

[153] Masoud Zabihi, Zhengyang Zhao, D. C. Mahendra, Zamshed I. Chowdhury, Salonik Resch, Thomas Peterson, Ulya R. Karpuzcu, Jian-Ping Wang, and Sachin S. Sapatnekar. 2019. Using spin-Hall MTJs to build an energy-efficient in-memory computation platform. In *Proceedings of the 20th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 52–57.

[154] Stephanie A. Zajac, Amanda N. Bozovich, Bernard G. Rax, Joe Davila, Duc Nguyen, Wilson P. Parker, Aaron J. Kenna, Steven S. McClure, Jason L. Thomas, Kelly W. Stanford, et al. [n.d.]. Updated compendium of total ionizing dose (TID) test results for the europa clipper mission. In *Proceedings of the IEEE Radiation Effects Data Workshop (in Conjunction with 2020 NSREC)*. IEEE, 1–4.

[155] Jintao Zhang and Naveen Verma. 2019. An in-memory-computing DNN achieving 700 TOPS/W and 6 TOPS/mm 2 in 130-nm CMOS. *IEEE J. Emerg. Select. Top. Circ. Syst.* 9, 2 (2019), 358–366.

[156] Quming Zhou, Mihir R. Choudhury, and Kartik Mohanram. 2008. Tunable transient filters for soft error rate reduction in combinational circuits. In *Proceedings of the 13th European Test Symposium*. IEEE, 179–184.

[157] Quming Zhou and Kartik Mohanram. 2005. Gate sizing to radiation harden combinational logic. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 25, 1 (2005), 155–166.

Received 15 July 2021; revised 29 January 2022; accepted 19 February 2022