A Survey of Prototyping Platforms for Intermittent Computing Research

Harrison Williams
Virginia Tech
Blacksburg, Virginia, USA
hrwill@vt.edu

Matthew Hicks
Virginia Tech
Blacksburg, Virginia, USA
mdhicks2@vt.edu

Abstract

Batteryless energy harvesting platforms are gaining popularity as a way to bring next-generation sensing and edge computing devices to deployments previously limited by their need for batteries. Energy harvesting enables perpetual, maintenance-free operation, but also introduces new challenges associated with unreliable environmental power as systems face common-case, yet unpredictable power failures. Software execution on these devices is an active area of research: intermittently executed software must correctly and efficiently handle arbitrary interruption, frequent state saving/restoration, and re-execution of certain code segments as part of a normal operation. The wide application range for batteryless systems combined with strict limitations on size and performance means there is little overlap in batteryless system prototypes—platforms are chosen for familiarity or specific features in a given application. Unfortunately, the effectiveness of different intermittent computing approaches varies widely across devices. As a result, intermittent computing research is at best hard to generalize across platforms and at worst contradictory across studies.

This work explores several of the device-level differences that substantially affect intermittent system performance across eight low-power prototyping platforms. We examine system-level assumptions made by the major approaches to intermittent computing today and determine how compatible each approach is with each platform. The goal of this paper is to serve as a guide for researchers and practitioners developing intermittent systems to both understand the landscape of devices suitable for batteryless operation and to highlight how interactions between devices and the intermittent software running on them can profoundly affect both performance and high-level conclusions in intermittent systems research. We open source our device bring-up code and instructions to facilitate multi-board experiments for future approaches.

CCS Concepts

• **Computer systems organization** → **Embedded systems**; Sensor networks; • **Hardware** → Wireless devices.

Keywords

Energy harvesting, intermittent computing, prototyping

ENSsys '24, November 4–7, 2024, Hangzhou, China

ACM ISBN 979-8-4007-1296-8/24/11
https://doi.org/10.1145/3698384.3699612

ACM Reference Format:
Harrison Williams and Matthew Hicks. 2024. A Survey of Prototyping Platforms for Intermittent Computing Research. In *International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems (ENSsys '24), November 4–7, 2024, Hangzhou, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3698384.3699612

1 Introduction

The Internet-of-Things (IoT) is projected to expand to over 25 billion connected devices by 2030—the majority of which will be tiny, resource-limited sensor nodes [2]. Beyond massive-scale IoT networks, low-power embedded systems are increasingly being deployed for sensing and actuation in remote or otherwise inaccessible environments (e.g., in space [9, 10] or as medical implants [41]). Large-scale networks and extreme environments place unique demands on the systems operating within them. One central challenge is the need to operate without a conventional power source (i.e., a battery): the lifetime, maintenance, and deployability concerns of batteries make them a non-starter in many of the most promising next-generation applications.

A promising solution to the challenge of power delivery is energy harvesting: scavenging ambient energy directly from the environment to power bursts of operation using a solar panel, rectenna, or other appropriate harvesting circuit combined with a capacitor for short-term energy storage. Batteryless energy harvesting platforms allow practitioners to deploy powerful sensors and embedded processors untethered by traditional power supplies and have seen success in deployments such as environmental sensing and medical monitoring [1, 33, 41]. While energy harvesters can be built cheaper, run for longer, and operate in harsher environments than their battery-powered counterparts, the scarce and unreliable nature of environmental power introduces new challenges that must be overcome before energy harvesting enters the "mainstream" of mobile/IoT system design. The disparity between power input and consumption—most embedded devices still consume significantly more power than suitable energy harvesters currently supply—poses challenges for both hardware and software design, as systems must remain performant and correct in the face of frequent, unpredictable-yet-common-case power failure. Frequent power failures combined with general energy scarcity and unpredictability have introduced a variety of challenges not present on traditional energy-constrained (i.e., battery-powered) platforms.

Accordingly, energy harvesting and intermittent operation are active areas of research spanning the systems, architecture, and devices communities. Unfortunately, the field's broad range of potential applications combined with stringent energy/performance requirements means that batteryless prototypes are typically built

from the ground up to match the requirements of a specific deployment. The result is that individual studies are often poor guides for practitioners looking to build their own batteryless systems and that comparison across platforms is difficult, leading to different studies drawing inconsistent conclusions about the performance of competing intermittent computing techniques (§ 4.2). This paper seeks to identify platform-level parameters affecting the performance of intermittent systems, investigate their impact on different intermittent computing techniques, and explore how they vary across a diverse set of candidate devices for intermittent computing prototyping. We evaluate a set of software benchmarks across a range of energy-harvesting-class devices to demonstrate how the choice of hardware, architecture, and software dictates both application performance and the most effective approach to intermittent computation for a given platform. To support rapid prototyping for both future research and commercial deployment efforts, we explore a range of commercial devices available on ready-to-use development boards and publish our testbed and results publicly at https://github.com/FoRTE-Research/EnergyHarvestingDeviceTestbed.

This paper makes the following three contributions:

- We identify device-level parameters that dictate the performance of different intermittent computing approaches, including microarchitecture (§ 3), analog components (§ 4.2), and memory technology (§ 4.1).
- We quantify the performance of representative software across platforms and explore how different implementations and platforms fill the trade space for execution time, memory consumption, and energy consumption (§ 3.1).
- Finally, we qualitatively compare the platforms we explore and recommend platforms and intermittent systems for prototyping applications with different requirements (§ 5).

Device	Core	Max Clock (MHz)	SRAM (KB)	NVM (KB)
FE310-G002[13]	RISC-V	320	16	32000*
Apollo3[4]	Cortex-M4	96	384	1000
Apollo4 Lite[3]	Cortex-M4	192	1000	2000
M2L31KIDAE[29]	Cortex-M23	72	168	512
SAMD21G18[28]	Cortex-M0+	48	32	256
SAML11E16[27]	Cortex-M23	32	16	64
MSP430F5529[34]	MSP430X	25	8	128
MSP430FR5994[18]	MSP430X	16	8	256

Table 1: Microcontrollers we evaluate as part of the testbed repository. *Size set by external flash chip.

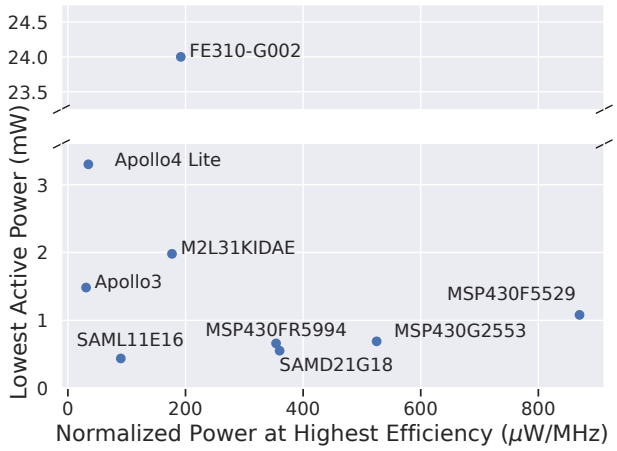

Figure 1: Comparison of operating points at maximum efficiency and minimum power for candidate devices.

2 Background

Energy harvesting systems face several design challenges that set them apart from traditional battery- or mains-powered low-power devices, in addition to the traditional size, weight, and power constraints of IoT-scale computation. While ongoing research efforts continue to push down the power consumption of mobile computing platforms, today's devices consume significantly more power than energy harvesting circuits can supply. As a result, batteryless systems execute intermittently, in short bursts of operation punctuated by long recharge times as the energy harvester refills the buffer capacitor. Batteryless systems are therefore primarily *energy-constrained*: increasing efficiency increases performance by allowing the system to run for longer. While low-power architectures are valuable for batteryless systems, processor efficiency is only one part of the story: the energy overhead of techniques to mitigate common-case power failures affects overall efficiency, and for developers choosing between commercial off-the-shelf devices the most efficient core may not be part of the overall most efficient platform for intermittent operation.

The primary challenge for effective intermittent execution is extending long-running computation across short, unpredictable power cycles. There is a large body of work exploring different strategies for system-level "checkpoints", in which all volatile state (e.g., software values, hardware registers) is committed to a persistent store and restored to continue execution on the next power-on period [5, 7, 20–24, 32, 38, 40]. While prior work also shows that architectural modification is a power technique for intermittent computing [12, 42], we focus on checkpointing techniques applicable to commercially available low-power devices. Different checkpointing strategies make different assumptions about the platform software is running on; we explore in Section 4 how the performance and feasibility of different checkpointing techniques depends on the underlying hardware. Checkpointing overhead must be balanced with common-case execution performance, binary size, and other concerns typical of ultra-constrained low power systems.

3 Microarchitecture

As more computation is pushed to deeply embedded edge devices, the active-mode efficiency and performance of the microcontroller core is becoming increasingly important. Highly efficient processors are available at a range of performance points, ranging from 8- and 16-bit processors designed for ultra-small sensing platforms to larger ARM- and RISC-based systems suitable for higher-end deployments such as small satellites. To encompass the corresponding

AES256	Chacha20	RSA	ECDH	SHA256
TinyAES	Portable8439	TinyRSA	TinyECDH	Gladman
Gladman	Chacha-AVR	Navin	BSD	Saddi
MbedTLS	MbedTLS	BearSSL	MbedTLS	MbedTLS
	RFC7539			

Table 2: Algorithms we evaluate across our device testbed.

range of potential batteryless system deployments we evaluate a set of devices spanning this performance range, detailed in Table 1. Each device in Table 1 targets low-power embedded operation and is available as part of a development board for rapid evaluation.

One key trend enabling higher efficiency embedded cores is favorable power/frequency scaling: running the low-power devices in Table 1 at higher frequencies tends to increase power efficiency, as energy consumption does not increase at the same rate as clock frequency. For example, increasing clock frequency on the MSP430FR5994 from 1 to 16 MHz only increases energy consumption by a factor of ~6.5. On battery-powered devices, these high efficiency/high power operating modes motivate a "sprint-to-completion" operating model where the system runs at a high frequency to complete some workload then enters a low-power sleep mode to conserve energy.

On batteryless systems, however, designers must consider the overhead associated with power failure. Higher power dissipation—regardless of efficiency—forces the system to operate at a lower duty cycle and endure more power failures. The extra effort needed to mitigate power failures introduces system-level overhead (e.g., checkpointing or re-execution of interrupted tasks) that is not well represented in a chip-level $\mu W/MHz$ figure. As a result, designers of intermittent systems need to balance efficiency with absolute power consumption to maximize overall performance.

We visualize the efficiency/minimum power trade space in Figure 1, which plots each device according to its normalized power consumption at its *most efficient operating point* (x axis) versus its absolute power consumption at its *lowest power operating point* (y axis). With the notable exception of the SAML11E16, few devices offer both low absolute active power and high energy efficiency per clock cycle. As the impact of power failures varies with checkpointing strategy, the optimal strategy depends on where the target device falls on this efficiency/absolute power trade space. Approaches with lower per-restart overhead (e.g., continuous checkpointing) are better suited for high-efficiency/high-power devices, while those with high per-restart overhead (e.g., just-in-time) will maximize overall performance on a lower absolute-power device.

3.1 Software Performance

While clock rate and energy consumed per clock cycle give valuable insights for predicting performance and efficiency, processor architecture and instruction set are important influences on overall performance. Differences in architecture, software implementation, and compiler performance mean that high-level parameters such as clock rate and efficiency are not enough to predict the fastest/lowest power setup for an embedded system. We explore these differences by running a set of different implementations of cryptographic algorithms—detailed in Table 2—on each device.

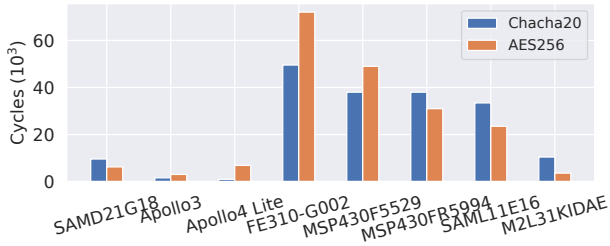

Figure 2: Execution time (in clock cycles) of symmetric-key encryption across devices.

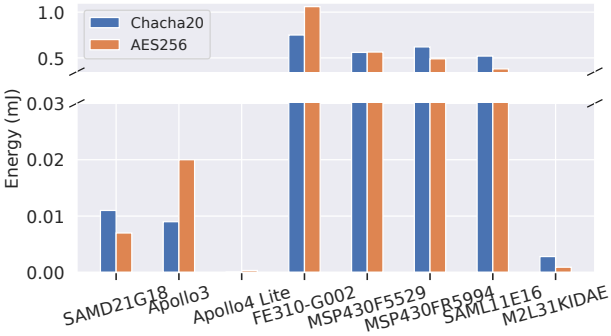

Figure 3: Energy consumption of symmetric-key encryption across devices.

We compare the overall execution time, energy consumption, and binary size of each algorithm on each device to determine the "best in class" combination of software and hardware for a given application. No single architecture, algorithm, or implementation is best: to illustrate, we plot the runtime of the fastest-running symmetric-key encryption implementations on each platform in Figure 2. The fastest algorithm across the eight devices is evenly split: AES is faster on four of our test devices, while Chacha20 is faster on the other four. Comparing these values to the energy consumption (Figure 3) of an iteration of each algorithm across devices demonstrates both inter- and intra-device changes: for example, the Apollo3 completes both encryptions significantly faster than the SAMD21G18, but consumes significantly more energy on AES. On the MSP430F5529, AES takes significantly longer to complete but ultimately consumes less energy than ChaCha20. In the highly-constrained, single-application deployments typical of energy harvesting platforms, datasheet power numbers are insufficient: developers must consider the interactions between software and the architecture it runs on. Optimizing for other constraints reveals additional tradeoffs; for example, the AES and ChaCha20 binaries on the MSP430F5529 consume significantly less NVM than on the other platforms.

4 Checkpointing Support

Power and size constraints limit the amount of energy batteryless systems can buffer for a single operating period, forcing software to execute in short bursts separated by power cycles. Extending long-running operations across power cycles requires preserving volatile state to a persistent store in the form of a checkpoint, from which the system can restart after a power failure and eventually complete arbitrarily long-running processes. Broadly, several approaches to checkpointing exist—each of which makes different assumptions and demands of the underlying system:

Just-In-Time (JIT) checkpointing systems [5, 24, 32, 38] monitor the supply voltage on the buffer capacitor and store a copy of all volatile state in Non-Volatile Memory (NVM) when a drop in supply voltage indicates the system will soon lose power. JIT systems minimize software overhead because software runs unmodified until power loss; one checkpoint is taken per power failure, and only the minimal set of data needed to correctly continue execution is committed to NVM. However, JIT systems realize this low software overhead at the cost of strong hardware assumptions: the system must include analog voltage monitoring components (e.g., an analog-to-digital converter or voltage comparator). While many energy-harvesting-class devices use mixed-signal processes and include analog measurement components, we explore in Section 4.2 how the power consumption of these integrated components often considerably reduces overall energy efficiency.

Task-based approaches [7, 21, 22, 37] eliminate the need for hardware voltage monitoring by instead requiring programmers to refactor code into self-contained tasks; a purpose-built compiler inserts code to checkpoint volatile state between tasks, "locking in" progress at the end of each task. A task management runtime rolls back partially updated state from an interrupted task before re-executing it to maintain correctness when code accesses NVM as part of its normal execution [31]. Software written for these systems is naturally resilient to power failures and guaranteed to make forward progress as long as each task is small enough to complete in a single power cycle. Checkpoints in task-based systems are more frequent but smaller than JIT systems because post-task checkpoints only need to update global state that the prior task actually changed. Task-based systems trade increased software overhead—checkpoints are recorded to NVM on every task transition, regardless of energy conditions—in exchange for avoiding voltage monitoring components.

Automatic techniques [20, 23, 40] eliminate the need for hardware support or programmer intervention using compile-time analysis to back up state to NVM throughout execution. Automatic compiler-based systems place small, frequent checkpoints based on different metrics: Ratchet and WARio [20, 40] insert checkpoints as necessary to split non-idempotent memory accesses (i.e., code segments which have different effects on global state depending on the number of times they execute) to reduce rollback overhead when the system loses power. Chinchilla [23] checkpoints at the granularity of basic blocks and explicitly rolls back uncheckpointed non-idempotent accesses after a power failure.

Automatic systems combine the advantages of task-based and JIT approaches: like task-based systems, they eliminate the need for potentially-expensive hardware support. Like JIT techniques, automatic systems do not require the programmer to refactor code—an important consideration for large code bases or applications where the specific source code implementation is important (e.g., certified cryptographic libraries). The primary limitation of automatic techniques is instead that their execution models *depend on non-volatile main memory*. Compared to JIT and task-based systems, automatic approaches take far more frequent (but smaller) checkpoints and reduce checkpointing overhead by maintaining runtime state in non-volatile main memory. As the access overhead associated with off-chip NVM would dramatically increase the cost of frequent checkpointing, automatic approaches target devices with high-performance on-chip NVM. Unfortunately, few commercially available systems with suitable NVM are available today, limiting the applicability of automatic hardware-free techniques (§5).

4.1 Memory Considerations

Intermittent systems' dependence on frequently checkpointing volatile state to a persistent store means that NVM is on the "critical path" of overall system performance. For developers limited to commercial devices, NVM is intrinsically tied to other on-chip components (e.g., architecture, analog features, etc.) depending on the devices available from vendors. Table 3 shows the on-chip NVM type for our evaluated set of systems.

Four of the eight devices we evaluate use flash as their on-chip NVM. The prevalence of flash memory represents a hurdle for intermittent systems using commercial platforms: flash is mature, inexpensive, and easy to integrate onto microcontroller dies—strongly incentivizing vendors to choose flash for NVM on standard low-power systems which use NVM primarily for code storage. Unfortunately, the energy-intensive and endurance-limited nature of flash writes makes flash particularly poorly suited for intermittent computation. Even at low rates of checkpointing using JIT systems, typical intermittent systems exhaust a flash memory's endurance in a matter of days to weeks [17]. As a result, research prototypes and deployed intermittent systems depend on new types of NVM.

Several emerging memories are available commercially on low-power microcontrollers, often referred to collectively as Non-Volatile Random Access Memory (NVRAM). Three such technologies have so far been integrated into energy-harvesting-class devices: Resistive RAM (RRAM) [6], Spin-Transfer Torque Magnetoresistive RAM (STT-MRAM) [30], and Ferroelectric RAM (FRAM) [35]. Each of these technologies significantly reduces write overhead and eliminates the endurance problems associated with flash, making them suitable for the NVM-intensive access patterns required of intermittent computing.

Unfortunately, the relative immaturity of these technologies introduces tradeoffs developers must consider when designing an intermittent system. Improving checkpointing performance using integrated NVRAM comes at a tradeoff of common-case execution performance because current NVRAMs significantly underperform flash memory during read-intensive operations [26, 36]. Systems combining different NVM technologies, such as using integrated flash for code storage/execution in conjunction with an external NVRAM chip for checkpoints, are a promising option for systems that experience relatively infrequent power failures.

Beyond pure performance concerns, the relative rarity of NVRAM-based embedded systems means a system containing some desirable on-chip feature such as a specific processor or peripheral device may not be available with integrated NVRAM. In these cases, developers are limited to low-checkpoint-rate JIT or task-based approaches using discrete NVRAM chips [8]. Higher-end devices such as the FE310 [13] offer one possible solution through a Quad SPI (QSPI) memory mapped interface, which allows software to access external memory devices similarly to internal on-chip memory. As discrete NVRAM chips supporting QSPI are available [14], this is a promising approach to bringing automatic techniques to platforms where the increased board size/cost is not prohibitive.

An alternative line of research targets systems where NVRAM is not available and instead explores retaining program state using time-dependent non-volatility in SRAM, the standard volatile memory technology for low-power embedded systems [37, 38]. These systems take advantage of the fact that SRAM's typical data retention voltage is well below the minimum operating voltage of the entire microcontroller (~400 mV versus 1.8V in many cases), resulting in a scenario following a power failure where software cannot execute but SRAM is functionally non-volatile for minutes to hours as leakage slowly brings the system supply voltage down from the software brown-out voltage to SRAM's data retention voltage. TotalRecall [38] uses SRAM's time-dependent non-volatility in the context of a JIT checkpointing system by calculating a checksum over all volatile state in SRAM before a power failure, storing checkpointed data "in-place" in SRAM and verifying its integrity on recovery using the checksum. Camel [37] extends this approach to the task-based model by logically splitting the SRAM into volatile and non-volatile "worlds", placing working data in the volatile world and writing inter-task checkpoints to the non-volatile world.

SRAM-based approaches bring intermittent computing to systems without high-performance NVM but introduce their own set of associated limitations. No fully automatic, compiler-only intermittent computing system exists, limiting current SRAM-based systems to the JIT or task-based models. On a hardware level, using SRAM to store data across power cycles requires SRAM's supply voltage rail to be *unregulated*. This is because the linear regulators integrated into low-power microcontrollers typically explicitly drive their output voltage low when their input voltage falls below the system's minimum to prevent a reverse current condition. As a result, SRAM's supply rail is driven to 0 as soon as the device's input voltage falls below the minimum for software operation—preventing SRAM from retaining state across power failures.

Table 3 illustrates which of the devices we evaluate includes an internal regulator that would interfere with SRAM-based checkpointing. With the exception of the FE310-G002, none of the devices we test are compatible with SRAM-based intermittent computation without hardware modification. Broadly, we expect SRAM-based intermittent computing to be a better fit for devices outside the mid-range systems we focus on here. Higher end devices such as the FE310 require a regulated core voltage but save die space for better core performance by requiring designers to include an external regulator, allowing users to add capacitance directly to the supply rail to extend SRAM's data retention. Lower end, cheaper devices using older technology nodes (e.g., the MSP430G2553 [16]) do not require the core voltage to be regulated.

	Current (μA)				
Device	**Core**	**Comp.**	**ADC**	**NVM Type**	**SRAM Regulated?**
FE310-G002	8000	N/A	N/A	QSPI External	No
Apollo3	494	10	300	Flash	Yes
Apollo4 Lite	1100	10	300	MRAM	Yes
M2L31KIDAE	660	70	400	RRAM	Yes
SAMD21G18	184	16	1250	Flash	Yes
SAML11E16	146	6	150	Flash	Yes
MSP430F5529	360	70	220	Flash	Yes
MSP430FR5994	220	150	230	Flash	Yes

Table 3: Device-level parameters that influence checkpointing performance. Comparator and ADC current include internal voltage reference generation.

4.2 Analog Components

Many of the devices best suited for energy harvesting applications are mixed-signal systems originally designed for battery-powered sensing deployments. As a result, they include sensing components useful for energy harvesters such as voltage comparators and Analog-to-Digital Converters (ADCs) which allow software to measure supply voltage and estimate energy remaining before power failure. Voltage supervision is both a necessary component of JIT checkpointing and a valuable tool for batteryless operation in general, as software can use knowledge of energy conditions to schedule tasks or modulate performance according to incoming/buffered energy [11, 25, 39].

Integrated analog components save cost and board space but come with their own set of drawbacks. On-chip comparators and ADCs tend to underperform their discrete counterparts in both performance and energy consumption due to cost and size constraints. Combined with the superior downscaling of digital circuitry compared to analog components [39], voltage supervision on commercial low-power devices tends to represent a significant portion of overall power consumption. We detail the current consumption of the on-chip comparator and ADC, compared to the minimum active-mode current consumption of the digital core for context, for our testbed devices in Table 3. For each device, the on-chip comparator draws several orders of magnitude more power than a comparable discrete device [19]. In the case of the MSP430FR5994, the comparator and internal bandgap voltage reference combined can draw more power than the entire active-mode consumption of the digital core (depending on cache hit rates) [18].

The power consumption of integrated components is a crucial consideration when prototyping batteryless systems because it affects the high-level conclusions drawn from experiments on those systems. For example, the study describing Samoyed [24] compares the overall performance of JIT and task-based checkpointing and concludes that JIT approaches typically reduce end-to-end overhead compared to task-based ones owing to high checkpointing overhead in task-based systems. As the software overhead of task-based approaches falls, however, the performance gap has closed enough that analog components' power consumption must be considered. Camel—the state of the art task-based system [37]—reduces overhead over prior works by a factor of 2, and the authors demonstrate

Device	JIT	Automatic	Task-based
FE310-G002	○	◒	◒
Apollo3	◒	○	◒
Apollo4 Lite	●	●	●
M2L31KIDAE	●	●	●
SAMD21G18	◒	○	◒
SAML11E16	◒	○	◒
MSP430F5529	◒	○	◒
MSP430FR5994	◒	●	●

Table 4: Compatibility of commercial devices with different intermittent computing approaches. Symbols: ○ None, ◒ Some, ● Strong.

that it significantly outperforms JIT approaches on a typical low-power device [15] when low-power discrete analog components are not an option. Both researchers and industrial developers of energy harvesting systems should consider the implications and requirements of different intermittent systems and how they interact with prototyping platforms.

5 Prototyping Recommendations and Challenges

The unique demands of intermittent operation, variability across both checkpointing systems and hardware platforms, and ad hoc nature of many existing studies makes choosing the best platform for intermittent systems development difficult. To serve as a guide for researchers and developers, we classify each device in our testbed as having no, some, or strong compatibility with each intermittent computing approach in Table 4. Our focus is on combining *ease of prototyping* with *overall performance*. We classify a hardware/software combination of systems as having:

- **Strong** compatibility if the hardware can be used without modification (i.e., development board only) and at typically low overhead
- **Some** compatibility if the hardware can be used unmodified with a given system at a performance penalty or would require only minor modification
- **No** compatibility if the system combination is infeasible or the platform requires extensive modification.

In general, few devices are strongly compatible with every intermittent computing approach. With the exception of the Apollo4, M2L31, and MSP430FR lines of devices, each system listed requires external NVRAM to store checkpoints of persistent state. The most immediate consequence of this requirement is that it precludes automatic, compiler-based checkpointing systems which depend on small and frequent updates to NVM. The aforementioned QSPI interface allows the FE310 partial compatibility with automatic checkpointing systems, although extra work is required to mitigate the possibility of power loss during a QSPI transfer. Given the potential of fully-automatic, compiler-based checkpointing systems, we consider overcoming the need for integrated NVRAM in these approaches to be a compelling open research question.

JIT and task-based systems introduce considerably less NVM pressure and so are more amenable to using off-chip NVRAM. The primary limiting factor for JIT systems is the need for voltage supervision, which can represent a significant energy burden. The Apollo3/4, M2L31KIDAE, SAMD21G18, SAML11E16, and MSP430F5529 all contain hardware to monitor voltage with a single bit of precision—enough for checkpointing—at a modest to low energy cost, allowing efficient intermittent computation with a discrete NVRAM as the only external component. The MSP430FR5994, despite its integrated NVRAM, spends a significant portion of incoming power on voltage supervision (Table 3) when using only integrated components and so is better suited for approaches that totally eschew the need for voltage supervision. The FE310, fabricated on a digital-only process, requires a number of additional discrete components (e.g., a comparator and voltage reference in addition to an NVRAM chip) for JIT operation.

Task-based approaches minimize the hardware barrier to entry for intermittent computing: they do not require NVRAM main memory or voltage supervision. This maximizes their compatibility with prototyping systems "out of the box": the Apollo4, M2L31, and MSP430FR devices are well suited for task-based operation without modification, and the other devices would only require an external NVRAM. The tradeoff of task-based systems is that they shift the development burden onto the programmer: source code must be completely refactored to a task-based model, and overall performance depends on programmer decisions and understanding of runtime energy dynamics. For hardware-constrained systems, however, task-based approaches remain a feasible option.

6 Conclusion

Batteryless energy harvesting systems have the potential to revolutionize IoT deployments by enabling long-lived, high-performance sensing and computing systems in applications previously limited by the need for batteries. Effective intermittent software execution is a key component of a batteryless future, but the field today is dominated by research prototypes designed and optimized for platforms with specific features and performance profiles. We carry out a high-level study exploring the device-level assumptions made by each major intermittent computing approach and examine how available commercial prototyping systems fit these assumptions. We find that interactions between the intermittent computing system and the actual device it runs on profoundly affect overall performance: no one approach or platform is best across all scenarios, and few commercial devices are compatible with all intermittent support modalities. We hope our work serves both as an inspiration for more research towards accessible, general-purpose intermittent systems and as a guide for practitioners looking to transfer research prototypes into deployable systems.

Acknowledgments

The project depicted is sponsored by the Defense Advanced Research Projects Agency. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. Approved for public release; distribution is unlimited. This material is based upon work supported by the National Science Foundation under Grant No. 2240744.

References

[1] Miran Alhaideri, Michael Rushanan, Denis Foo Kune, and Kevin Fu. 2013. The Moo and Cement Shoes: Future Directions of A Practical Sense-Control-Actuate Application. http://terraswarm.org/pubs/111.html Presented at First International Workshop on the Swarm at the Edge of the Cloud (SEC'13 @ ESWeek), Montreal..

[2] allan Jay. 2024. Number of Internet of Things (IoT) Connected Devices Worldwide 2024: Breakdowns, Growth & Predictions. https://financesonline.com/number-of-internet-of-things-connected-devices/.

[3] Ambiq. 2023. Apollo4 Blue Lite SoC. https://www.mouser.com/datasheet/2/1494/Apollo4_Blue_Lite_Datasheet-3317499.pdf.

[4] Ambiq. 2024. Apollo3 and Apollo3 Blue SoC. https://ambiq.com/wp-content/uploads/2020/10/Apollo3-Blue-SoC-Datasheet.pdf.

[5] Domenico Balsamo, Alex Weddell, Geoff Merrett, Bashir Al-Hashimi, Davide Brunelli, and Luca Benini. 2014. Hibernus: Sustaining Computation during Intermittent Supply for Energy-Harvesting Systems. In *IEEE Embedded Systems Letters*.

[6] Tsai-Kan Chien, Lih-Yih Chiou, Shyh-Shyuan Sheu, Jing-Cian Lin, Chang-Chia Lee, Tzu-Kun Ku, Ming-Jinn Tsai, and Chih-I Wu. 2016. Low-Power MCU With Embedded ReRAM Buffers as Sensor Hub for IoT Applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 6, 2 (2016), 247–257. https://doi.org/10.1109/JETCAS.2016.2547778

[7] Alexei Colin and Brandon Lucia. 2016. Chain: Tasks and Channels for Reliable Intermittent Programs. In *International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. 514–530.

[8] Jasper de Winkel, Vito Kortbeek, Josiah Hester, and Przemysław Pawełczak. 2020. Battery-Free Game Boy. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 3, Article 111 (sep 2020), 34 pages. https://doi.org/10.1145/3411839

[9] Bradley Denby, Krishna Chintalapudi, Ranveer Chandra, Brandon Lucia, and Shadi Noghabi. 2023. Kodan: Addressing the Computational Bottleneck in Space. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 392–403. https://doi.org/10.1145/3582016.3582043

[10] Bradley Denby and Brandon Lucia. 2020. Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 939–954. https://doi.org/10.1145/3373376.3378473

[11] H. Desai and B. Lucia. 2020. A Power-Aware Heterogeneous Architecture Scaling Model for Energy-Harvesting Computers. *IEEE Computer Architecture Letters* 19, 1 (2020), 68–71.

[12] Matthew Hicks. 2017. Clank: Architectural Support for Intermittent Computation. In *International Symposium on Computer Architecture (ISCA)*. 228–240.

[13] SiFive Inc. 2021. SiFive FE310-G002. https://starfivetech.com/uploads/fe310-g002-datasheet-v1p2.pdf.

[14] Infineon. 2022. 8Mb EXCELON™ Ultra Ferroelectric RAM (F-RAM). https://www.mouser.com/datasheet/2/196/Infineon_CY15B108QSN_108BKXI_DataSheet_v05_00_EN-3106498.pdf.

[15] Texas Instruments. 2013. MIXED SIGNAL MICROCONTROLLER. https://www.ti.com/lit/ds/symlink/msp430g2955.pdf.

[16] Texas Instruments. 2013. MSP430G2x53, MSP430G2x13 Mixed Signal Microcontroller datasheet (Rev. J). http://www.ti.com/lit/ds/symlink/msp430g2553.pdf.

[17] Texas Instruments. 2018. MSP430 Flash Memory Characteristics (Rev. B). http://www.ti.com/lit/an/slaa334b/slaa334b.pdf.

[18] Texas Instruments. 2018. MSP430FR5964—MSP430FR599x, MSP430FR596x Mixed-Signal Microcontrollers. http://www.ti.com/lit/ds/symlink/msp430fr5964.pdf.

[19] Texas Instruments. 2018. TLV7081 Nano-Power, 4-Bump WCSP, Small-Size Comparator. https://www.ti.com/lit/ds/symlink/tlv7081.pdf.

[20] Vito Kortbeek, Souradip Ghosh, Josiah Hester, Simone Campanoni, and Przemysław Pawełczak. 2022. WARio: efficient code generation for intermittent computing. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (San Diego, CA, USA) *(PLDI 2022)*. Association for Computing Machinery, New York, NY, USA, 777–791. https://doi.org/10.1145/3519939.3523454

[21] Brandon Lucia and Benjamin Ransford. 2015. A Simpler, Safer Programming and Execution Model for Intermittent Systems. In *Conference on Programming Language Design and Implementation (PLDI)*. 575–585.

[22] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2017. Alpaca: Intermittent Execution Without Checkpoints. In *International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. 96:1–96:30.

[23] Kiwan Maeng and Brandon Lucia. 2018. Adaptive Dynamic Checkpointing for Safe Efficient Intermittent Computing. In *USENIX Conference on Operating Systems Design and Implementation (OSDI)*. 129–144.

[24] Kiwan Maeng and Brandon Lucia. 2019. Supporting Peripherals in Intermittent Systems with Just-in-time Checkpoints. In *SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. 1101–1116.

[25] Kiwan Maeng and Brandon Lucia. 2020. Adaptive Low-Overhead Scheduling for Periodic and Reactive Intermittent Execution. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation* (London, UK) *(PLDI 2020)*. Association for Computing Machinery, New York, NY, USA, 1005–1021. https://doi.org/10.1145/3385412.3385998

[26] Andrea Maioli and Luca Mottola. 2021. ALFRED: Virtual Memory for Intermittent Computing. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems* (Coimbra, Portugal) *(SenSys '21)*. Association for Computing Machinery, New York, NY, USA, 261–273. https://doi.org/10.1145/3485730.3485949

[27] Microchip. 2019. Ultra Low-Power, 32-bit Cortex-M23 MCUs with TrustZone, Crypto, and Enhanced PTC. https://ww1.microchip.com/downloads/en/DeviceDoc/SAM-L10L11-Family-DataSheet-DS60001513F.pdf.

[28] Microchip. 2021. SAM D21/DA1 Family. https://ww1.microchip.com/downloads/aemDocuments/documents/MCU32/ProductDocuments/DataSheets/SAM-D21-DA1-Family-Data-Sheet-DS40001882H.pdf.

[29] Nuvoton. 2024. M2L31 Series Datasheet. https://www.nuvoton.com/export/resource-files/en-us--DS_M2L31_Series_EN_Rev1.00.pdf.

[30] Guillaume Patrigeon, Pascal Benoit, Lionel Torres, Sophiane Senni, Guillaume Prenat, and Gregory Di Pendina. 2019. Design and Evaluation of a 28-nm FD-SOI STT-MRAM for Ultra-Low Power Microcontrollers. *IEEE Access* 7 (2019), 58085–58093. https://doi.org/10.1109/ACCESS.2019.2906942

[31] Benjamin Ransford and Brandon Lucia. 2014. Nonvolatile Memory is a Broken Time Machine. In *Workshop on Memory Systems Performance and Correctness*.

[32] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: System Support for Long-Running Computation on RFID-Scale Devices. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[33] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith. 2008. Design of an RFID-Based Battery-Free Programmable Sensing Platform. *IEEE Transactions on Instrumentation and Measurement* 57, 11 (Nov 2008), 2608–2615.

[34] Texas Instruments. 2020. MSP430F552x, MSP430F551x Mixed-Signal Microcontrollers. https://www.ti.com/lit/ds/symlink/msp430f5529.pdf.

[35] Texas Instruments. 2023. Low-Power FRAM Microcontrollers and Their Applications. https://www.ti.com/lit/wp/slaa502a/slaa502a.pdf.

[36] Harrison Williams, , and Matthew Hicks. 2025. A Software Caching Runtime for Embedded NVRAM Systems. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[37] Harrison Williams, Saim Ahmad, and Matthew Hicks. 2024. A Difference World: High-performance, NVM-invariant, Software-only Intermittent Computation. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. USENIX Association, Santa Clara, CA, 1223–1238. https://www.usenix.org/conference/atc24/presentation/williams

[38] Harrison Williams, Xun Jian, and Matthew Hicks. 2020. Forget Failure: Exploiting SRAM Data Remanence for Low-Overhead Intermittent Computation. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 69–84.

[39] Harrison Williams, Michael Moukarzel, and Matthew Hicks. 2021. Failure Sentinels: Ubiquitous Just-in-Time Intermittent Computation via Low-Cost Hardware Support for Voltage Monitoring. In *International Symposium on Computer Architecture (ISCA)*. 665–678.

[40] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent Computation without Hardware Support or Programmer Intervention. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 17–32.

[41] X. Wu, I. Lee, Q. Dong, K. Yang, D. Kim, J. Wang, Y. Peng, Y. Zhang, M. Saliganc, M. Yasuda, K. Kumeno, F. Ohno, S. Miyoshi, M. Kawaminami, D. Sylvester, and D. Blaauw. 2018. A 0.04MM316NW Wireless and Batteryless Sensor System with Integrated Cortex-M0+ Processor and Optical Communication for Cellular Temperature Measurement. In *2018 IEEE Symposium on VLSI Circuits*. 191–192.

[42] Yuchen Zhou, Jianping Zeng, Jungi Jeong, Jongouk Choi, and Changhee Jung. 2023. SweepCache: Intermittence-Aware Cache on the Cheap. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture* (<conf-loc>, <city>Toronto</city>, <state>ON</state>, <country>Canada</country>, </conf-loc>) *(MICRO '23)*. Association for Computing Machinery, New York, NY, USA, 1059–1074. https://doi.org/10.1145/3613424.3623781