Design Insights of Non-volatile Processors and Accelerators in Energy Harvesting Systems

Keni Qiu
Capital Normal University, China
qiukn@cnu.edu.cn

Mengying Zhao
Shandong University, China
zhaomengying@sdu.edu.cn

Zhenge Jia, Jingtong Hu
University of Pittsburgh, USA
zhenge.jia,jthu@pitt.edu

Chun Jason Xue
City University of Hong Kong, China
jasonxue@cityu.edu.hk

Kaisheng Ma, Xueqing Li, Yongpan Liu
Tsinghua University, China
kaisheng,xueqingli,ypliu@tsinghua.edu.cn

Vijaykrishnan Narayanan
Pennsylvania State University, USA
vxn9@psu.edu

ABSTRACT

There is growing interest in deploying energy harvesting processors and accelerators in Internet of Things (IoT). Energy harvesting harnesses the energy scavenged from the environment to power a system. Although it has many advantages over battery-operated systems such as lightweight, compact size, and no necessity of recharging and maintenance, it may suffer frequently power-down and a fluctuating power supply even with power on. Non-volatile processor (NVP) is a promising architecture for effective computing in energy harvesting scenarios. Recently, non-volatile accelerators (NVA) have been proposed to perform computations of deep learning algorithms. In this paper, we overview the recent studies of NVP and NVA across the layers of hardware, architecture, software and their co-design. Especially, we present the design insights of how the state-of-the-art works adapt their specific designs to the intermittent and fluctuating power conditions with the energy harvesting technology. Finally, we discuss recent trends using NVP and NVA in energy harvesting scenarios.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; **Embedded systems**; **Embedded hardware**;

KEYWORDS

Energy harvesting; Non-volatile processor; PIM; Architecture; Checkpointing; Task scheduling

ACM Reference Format:
Keni Qiu, Mengying Zhao, Zhenge Jia, Jingtong Hu, Chun Jason Xue, Kaisheng Ma, Xueqing Li, Yongpan Liu, and Vijaykrishnan Narayanan. 2020. Design Insights of Non-volatile Processors and Accelerators in Energy Harvesting Systems. In *Great Lakes Symposium on VLSI 2020 (GLSVLSI '20), September 7–9, 2020, Virtual Event, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3386263.3407596

GLSVLSI '20, September 7–9, 2020, Virtual Event, China

ACM ISBN 978-1-4503-7944-1/20/09…$15.00
https://doi.org/10.1145/3386263.3407596

1 INTRODUCTION

The Internet-of-Things (IoT) devices are growing at a rapid speed because of their huge impact in many sectors. There are an estimated 31 billion devices in 2020 [1]. Historically, battery is the primary power source for many mobile and embedded edge IoT devices. However, progress in battery technology has not paced that of the power demand of smart devices. Consequently, the large form factor, limited charge storage and high replacement cost of batteries have become impediments to capabilities at edge devices. In contrast, energy harvesting (EH), a technique enabling the applications to scavenge energy from the ambient environment can dramatically extend the operating lifetime of the system [2–4].

An energy harvester extracts power from the ambient environment and provides an attractive power alternative in many sensing application scenarios such as implantable medical devices, hazardous war-zone systems, and outer space electronics, where it is difficult to rely only on batteries. Solar [5], thermal [6], RF signal [7], piezoelectric [2], kinetic power [8] are all promising sources of energy. For example, the Ultra-Low Power Sensor Evaluation Kit (ULPSEK) [6] for evaluation of biomedical sensors and monitoring applications is a wearable, multi-parameter health sensor powered by an efficient body heat harvester. ULPSEK can measure and process electrocardiogram, respiration, motion and body temperature. However, ambient harvestable energy is usually unstable and difficult to utilize effectively. With such unreliable power supply, EH-powered computing systems have to run intermittently.

Since traditional computing systems are designed to operate with a stable power supply, they cannot make much continuous progress with the intermittent power sources. In the worst case, they cannot complete even a single task because they have to restart due to power outage. In addition, a non-adaptive system often cannot adjust to the power fluctuations, leading to low power efficiency. Therefore, the critical challenge of an energy harvesting system is how to make a system operate correctly and efficiently under unreliable power supply.

The recent adoption of non-volatile processor (NVP) in energy harvesting devices is deemed as state-of-the-art solution to the intermittent computing. The NVP processor is usually equipped with emerging non-volatile memory (NVM) to maintain in-process data and store configurations between power outages. Thus, the most important advantage of NVP compared with a conventional processor is that it can survive power shortage without losing their

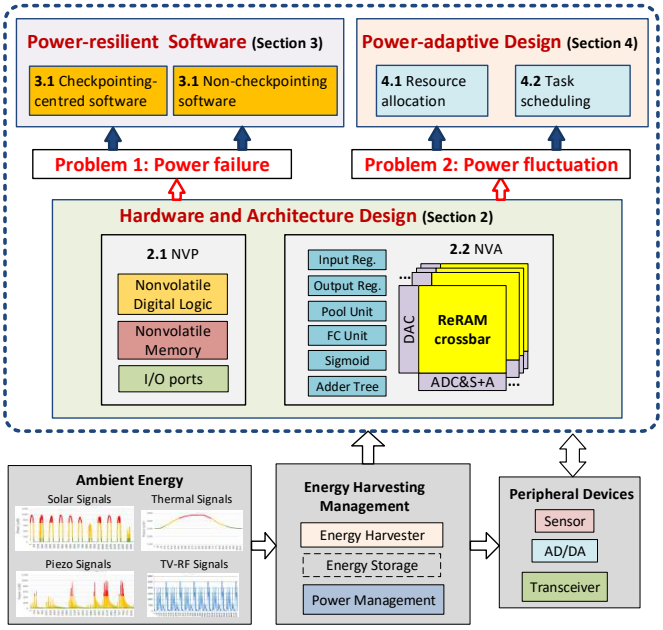

Figure 1: Overview of the survey.

status meanwhile with fast wake-up property, which fits well with the intermittent characteristic of the ambient power [9].

Most recently, many works have proposed non-volatile accelerator (NVA) architecture to embed a computing accelerator to offload inference tasks on IoT nodes rather than merely transmitting raw sensor data [10, 11]. Since an NVA has to work under fluctuating power supply, it often features of lightweight and reconfigurable design along with resilient software scheduling.

As NVP and NVA provide circuit-/architecture-level support to fit the intermittent computing, adaptive software design is highly necessary to co-operate with these emerging architectures to address the unique challenges from harvestable power supply: (i) power failure and (ii) power fluctuation. The software-level work can be divided into two categories. One category studies how to perform failure-resilient execution to guarantee correctness with or without checkpointing; the other category studies how to allocate hardware resources or schedule tasks to adaptively accommodate the fluctuating power input.

In this paper, we summarize the recent architectural design and software optimizations in the domain of energy harvesting system, covering the evolution of NVP and NVA, checkpointing-centred software optimizations, non-checkpointing approaches and adaptive hardware/task scheduling. Figure 1 summarizes the overview of topics covered in this paper according to different design levels. A harvester collects ambient energy. The energy management module regulates the harvested energy to power NVP/NVA and multiple peripherals. In order for the system to be resilient to power failure and power fluctuation, failure-resilient and adaptive software strategies are presented for execution correctness and optimizations.

The remainder of this paper is organized as follows. Section 2 presents the hardware- and architecture-level evolution of NVP and NVA. Section 3 summarizes the software techniques used to guarantee a power-resilient execution. Section 4 summarizes adaptively resource reconfiguring and task scheduling techniques in order to achieve high power efficiency. Section 5 discusses the future study prospects and Section 6 concludes the paper.

2 RECENT NON-VOLATILE PROCESSING HARDWARE AND ARCHITECTURES

2.1 NVP

The essential difference between NVP and traditional volatile processor is data saving to survive power failures, which needs backup logic to save volatile data to non-volatile memory. Wang et al. [12] propose a structure of non-volatile flip-flop (NVFF), where a non-volatile cell is attached to the volatile part for data saving with a localized fully-parallel bit-to-bit transfer. Li et al. [13] propose a set of intrinsically nonvolatile latches and DFFs by harnessing the built-in non-volatility of negative capacitance field-effect transistors (NCFETs), with fJ-level energy and ns-level intrinsic latency for a backup plus restore operation. Thirumala et al. [14] propose a structure of dual mode ferroelectric transistor based NVFF, which can dynamically configure between volatile and non-volatile modes, leading to automatic backup.

The design of NVFF can also take system features into consideration. Since the storage operation is unnecessary for unused memory cells, Liu et al. [15] propose to utilize self-write-termination (SWT) approach to eliminate needless backups. Besides, it supports to retain data during short power failure stages, leading to reduced number of backups and restores. This adaptive NVSRAM with SWT design enables a higher clock frequency and lower storage energy consumption. A ReRAM-based non-volatile flip-flop is further developed with fast-off, fast-wake-up features [16]. It leverages appropriate resistance range and sufficient resistance ratio of ReRAM to avoid loading too much capacitance/DC current on the power rail. Compared with the previous NVFF [17], this SWT-NVFF can achieve more than 93% reduction of store energy and more than 27× reduction of restore time.

NVFF enables fast backup with simple control, however, may induce large area overhead [21]. Thus for large-size memory space such as high-level cache and main memory, instead of directly employing NVFF to save all data, they usually attach non-volatile memory for selective data backup, where emerging non-volatile memory is preferred than Flash [22–25].

On the basis of backup logic, the architecture of NVP systems have been explored [26]. In [26], three architectures of NVP systems pertaining to Non-Pipelined configuration (NP), N-Stage-Pipeline (NSP) and Out-of-Order Processor (OoO), as well as several data backup strategies are discussed, providing a design guide to find optimal configuration of energy harvesting systems.

2.2 NVA

NVAs enable a new paradigm to offload neural network (NN) tasks on IoT nodes rather than cloud computation especially under a bandwidth-constrained circumstance. An NVA can be integrated to an NVP on one die or be an off-chip device beside an NVP.

Targeting the energy harvesting intelligent IoT devices, Su et al. [19] propose a non-volatile intelligent processor (NIP). In the NIP, the 1T1R organized ReRAM crossbar can directly use the resistance values stored in ReRAM cells as the NN weights to execute matrix-vector-multiplication (MVM) operations through a low

Table 1: Evolution of NVP and NVA

Type	Volatile Processor	NVP		NVA		
Publication	[18]	[12] (THU1010N)	[15] (NVP-SWT)	[19] (NIP)	[10] (STICKER-T)	[20] (Liquid Silicon)
Technology	Flash	FeRAM (130 nm)	ReRAM (65 nm)	ReRAM (150 nm)	Transpose SRAM (65 nm)	ReRAM (130 nm)
Power/Energy efficiency	450 μW/MHz (Active power)	160 μW/MHz (Active power)	33 μW/MHz (Active power) 30.3GOPs/J (Energy efficiency)	10.9 mW (Power consumption) 469.7GOPs/J (Energy efficiency)	13.3-339.2 mW (Power consumption) 0.39-140.3TOPs/W (Power efficiency)	60.95 TOPs/W (Power efficiency)
Backup time	6 ms	7 μs	4 μs	200 ns	N/A	N/A
Restore time	3 ms	3 μs	20 ns	100 ns	N/A	N/A
Area	N/A	N/A	1.56×2.86 mm^2	3.69 mm^2	3.0×2.5 mm^2	1.83×4 μm^2
Application type	General embedded applications	General embedded applications	General embedded applications	FCNN	CNN+FC+RNN	Machine learning + big data applications
Feature highlights	Off-chip memory for data backup	Automatic backup logic with 3μs wake-up time	nvSRAM+SWT & Adaptive retention+SWT	Low-power PIM	A unified MVM supporting variable NNs	A 2D array of identical tiles with flexibly programming

power MVM engine. In this MVM engine, binary input vector is directly linked to the word line, and the outputs are obtained from a 1-to-3-bit adaptive sense amplifiers (SAs) at the end of the bit-lines. The proposed NIP can achieve 470GOPs/J energy efficiency at 10MHz clock frequency, which is 13× higher than the prior design performance [27].

Architectures proposed to support unified NN algorithms often suffer from area-inefficiency and/or energy-inefficiency. Yue et al. [10] propose a unified NN processor, STICKER-T, which supports a variety of NN algorithms (e.g. CNN/FC/RNN). In this design, the workloads of CNN/FC/RNN are transformed into a unified MVM in the transpose domain by adopting a block-circulant algorithm. In STICKER-T, the 6T HBST-TRAM-based 2D data-reuse MAC array provides bit-serial MAC units to support 1b-to-12b precision. It can achieve 8.1× higher TOPS/mm^2 and 4.2× higher energy efficiency at 4b accuracy, compared to the most advanced unified NN processor [28].

To further improve flexibility/programmability to fit diverse machine learning and big data applications, Zha et al. [20] proposed a non-volatile fully programmable processing-in-memory (PIM) processor based on ReRAM, *Liquid Silicon*. Through programming the memory array and connected nodes, the 2D array of identical tiles in Liquid Silicon can be configured into computation, storage, search, and neural network modes. Liquid Silicon can be reprogrammed in a FPGA-like manner to flexibly change the ratio of on-chip computing and memory resources to accommodate a variety of machine learning and big data applications.

The superior characteristics of STICKER-T and Liquid Silicon with low power, high energy efficiency and flexible configurations make them very likely be used in energy harvesting scenarios.

Table 1 presents the design highlights and comparisons of several typical NVPs and NVAs.

3 FAILURE-RESILIENT SOFTWARE

3.1 Checkpointing-centred software

The prime objective of NVP systems is to guarantee the correctness with existence of checkpointing. It is observed that there may be errors with backup and resumptions [37]. For example, error may occur if an interruption happens during backup, or, certain instructions are re-executed after power resumption, leading to duplicate data modifications [38] or I/O operations [39]. To solve this problem, Lucia et al. [23] develop a system model based on backup and data versioning techniques to correctly record system status.

On the basis of system correctness, checkpointing-centred NVP systems face two design issues, i.e., *what* to back up and *when* to back up.

Theoretically, all volatile data that help system resumption need to be backed up. However, directly backing up all volatile data, especially in cache and main memory, would incur high performance and energy cost. This motivates researches on reducing the content needed to be backed up. Wang et al. [21] use compression techniques to build backup-efficient NVPs. Xie et al. [29] perform instruction scheduling to reduce the size written to NVM with fixed checkpoints. Li et al. [30] trim stack size by sharing space between variables and functions so that the stack content for backup can be reduced. Selective backup can be applied. For example, only dirty blocks need to be saved in cache [24, 31], and live-range analysis can be employed to decide which data need backup [32].

The question of when backup should be performed has also been studied. Ransford et al. [33] suggest to back up at loop latch, function return, and positions having a predetermined distance with last backup. Considering that the live content to be backed up varies along the program execution, it provides the flexibility to choose the backup location to reduce the backup cost. Zhao et al. [34] propose to select backup locations with less stack content. The proposed approach leverages the compiler to analyze live data at each instruction so that active stack size can be derived, which is then used for runtime reference to select backup activations. There are also researches targeting maximum forward progress through backup at the farthest position while guaranteeing successful backups. Li et al. [35] discuss how to integrate cache behavior analysis into backup location selection, with the objective of maximizing forward progress. In offline analysis, slots are injected to the disassembled code as backup location candidates. Then the cache behavior can be analyzed to insert backup labels into predefined slots. At runtime, backup is triggered at the farthest label to achieve maximal forward progress. Fan et al. [36] leverage Q-learning to determine where to back up. It takes energy and system status into consideration to guide backup at runtime. A set of states, actions and reward functions are defined to lead the generation of Q-table, which is used for backup guidance. When energy warning occurs at runtime, the proposed controller determines the backup position aiming to achieve maximum forward progress.

Table 2: Summary of checkpointing-centred software techniques

	Correctness	What to back up				When to back up			
Publication	[23]	[21]	[29]	[30]	[24, 31]/[32]	[33]	[34]	[35]	[36]
Key technique	Backup and data versioning	Reduce backup content				Offline			Online
		Compression based backup	Register aware instruction selection	Share space between variables and functions	Selective backup	Reduce backup cost		Maximize forward progress	
						Estimate available energy	Stack size aware	Cache behavior analysis	Q-learning based backup
Hierarchy	Full	Register	Register	Main memory	Cache/Full	Full	Main memory	Cache	

Table 2 presents the overview of representative checkpointing-centred software techniques.

3.2 Non-checkpointing software

In contrast to the design philosophy of checkpointing-centered research, some researchers attempt to provide intermittent execution support from the perspective of operating-system (OS) solutions, in which the system runtime takes care of intermittence with non-checkpointing software techniques.

Modern operation system often allows multitasking to improve CPU utilization. In this situation, the typically adopted runtime checkpointing techniques for an intermittent system may incur quite large runtime overheads, which would offset the computation progress improved by concurrency [40]. To enable low-overhead concurrent task execution on intermittent systems, a non-checkpointing failure-resilient approach [41] has been proposed for intermittent systems by leveraging the characteristics of data accessed in hybrid memory. The main idea is to maintain data consistency by atomically committing data copies modified by finished tasks in volatile memory (VM) to persistently consistent versions in NVM while ensuring the serializability of concurrent task execution through backward validation. Once power is restored, the persistently consistent version also allows instant system recovery.

However, in the above failure-resilient approach, tasks could be aborted and rerun frequently when serializability is violated due to high concurrency. To address this issue, a locking-based concurrency control design [42] has been proposed to ensure that running tasks are serializable from the outset, thus preventing tasks from being aborted and rerun caused by serializability violation.

Moreover, the failure-resilient approach needs to recreate and rerun all unfinished tasks from scratch after each power resumption. Under this context, the progress stagnation problem may arise when lengthy tasks whose execution time is longer than the power-on period are repeatedly recreated and rerun. To combat this, a context switch technique was introduced to preserve the computation progress of lengthy tasks across power cycles without runtime checkpointing [43]. This design enforces the context of the currently executed task to be switched out to NVM if it is lengthy and thus can be preserved during power-off periods. Then, to avoid data inconsistency, not until the next power-on period will lengthy task be scheduled and executed in NVM. This allows lengthy tasks can accumulate their progress across power cycles.

The above discussed efforts have been embedded into an intermittent OS built upon FreeRTOS. The intermittent OS endows energy-harvesting systems with the abilities to (i) run multiple tasks concurrently to improve computation progress, (ii) achieve data consistency without system suspension during runtime, (iii) recover instantly from power failures, and (iv) cumulatively preserve computation progress across power cycles to avoid stagnation. The intermittent OS is released under an open-source license and available at [44].

Besides the above idea line of keeping the atomic execution of critical tasks at the OS level, other works have achieved this at the compiler or programmer levels co-operated with specific execution strategies. Kiwan Maeng et al. propose a low-overhead programming model, *Alpaca* [45] for intermittent computing on energy-harvesting devices. Alpaca eliminates checkpoints because it can runtime preserve execution progress at the granularity of a user-defined task. The key insight is that data originally shared between tasks are copied into a buffer private to each task. Data consistency are maintained at the end of a task. Further, Kiwan Maeng et al. propose *CatNap* [46], an event-driven energy-harvesting system where a new embedded programming model allows a programmer to express time-critical code subsets. At runtime, the special time-critical code has higher priority than the rest of the code to be reliably executed on schedule with the reserved energy. CatNap can well meet timing requirements of the widespread periodic and reactive execution modes.

4 POWER-ADAPTIVE DESIGN

4.1 Adaptive hardware reconfiguring

Due to the instability of scaveneged ambient energy, it is found that ambient energy often cannot be sufficiently translated into forward progress. For instance, peak power tends to be many times the average power and lasts for a while, but during this period, the processor may still be operating at the minimal consumption level. In this scenario, the energy efficiency will be very low. So besides the efforts on power failure resilience, the other research category focuses on how to adapt an energy harvesting system to the power fluctuation property.

From the perspective of a co-designed hardware and software system with a reconfigurable power system, A. Colin et al. propose *Capybara* [47]. They define the energy demands of capacity-constrained tasks and temporally-constrained tasks. Capacity-constrained tasks must be executed atomically without interruption, while temporally-constrained tasks are reactive, thus requiring energy to be available on-demand. The proposed Capybara supports declarative specification of tasks' energy demands or mixed demands with a reconfigurable storage capacity mechanism. The responsiveness of Capybara can be improved by 2×-4× over a static energy capacity.

From the perspective of computing architectures, Kaisheng Ma et al. propose a machine learning-based integrated architecture of NVP [48] on the basis of architecture exploration in [26]. The architecture integrates three micro-architectures of NP, NSP and OoO, and their power requirements are $P^{NP} < P^{NSP} < P^{OoO}$. In [48], energy-dependent data are fed into a lightweight neural network for future power level prediction, and then the most appropriate

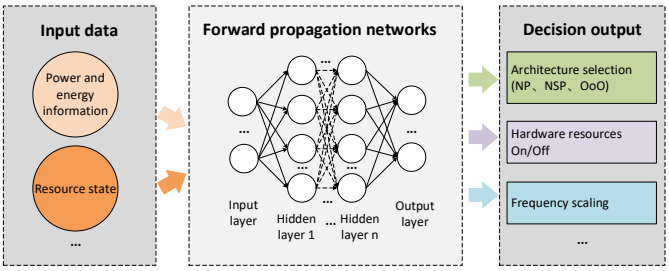

Figure 2: Several adaptive resource scheduling strategies.

architecture which maximizes progress of an application for the next power level can be chosen.

Expanding the above idea further, two techniques to improve energy efficiency are proposed in [49]. One is resource scaling which manages bottleneck resources in a reconfigurable OoO processor targeting lower energy per instruction (EPI). The other is dynamic frequency scaling which aggressively leverages harvested energy for NVPs. Both approaches are directed by an infrequently called machine learning algorithm. The proposed *Spendthrift* that combines the two techniques by setting both frequency and resource parameters efficiently outperforms either technique in isolation.

Figure 2 depicts the process of how to activate appropriate hardware resources directed by an NN-based power predictor.

Aside from the NN-based resource scaling techniques, a different idea has been explored from the viewpoint of opportunistic responsiveness. The works [50, 51] introduce a novel concept of *incidental computing* to address opportunistic responsiveness versus quality tradeoffs under unstable power availability. The very starting point is that the quality of older computations targeted for incidental computing can be gradually improved iteratively, if picked up over multiple incidental computing passes. The ultimate optimization of energy efficiency is achieved by reducing the energy of backup operations through a well matched retention time. It is shown that this approximate computing improves forward progress by an average of 4.28x over a baseline "precise" NVP.

Addressing Wireless Sensor Network (WSN) composed of multiple energy harvesting devices, the work *NEOFog* [52] provides new optimization opportunities to improve system performance and efficiency by exploiting the benefit of node-level nonvolatility. The implementation of Spendthrift [49] and non-volatile radio frequency controllers (NVRF) can shift nodes' paradigm to computation-intensive or frequently-intermittently-on, so as to improve the computation efficiency and reduce the energy of data transmission. NEOFog also takes advantage of node virtualization and slotted time-division multiplexing to improve computing QoS.

4.2 Adaptive task scheduling

Besides allocating adaptive hardware resources to match the power input, there are works exploring finer-grained strategies to accommodate the changing harvesting power through adaptive task scheduling. Amjad Yousef Majid et al. propose *Coala* [53], an adaptive task-based execution model. By means of task coalescing and splitting, Coala allows efficient execution on a sub-task scale so as to preserve computation progress. The challenging issues of task transition and task termination are also well handled by Coala.

Recently, more researches target NN applications in energy harvesting scenarios. As the power requirement of the modern practical inference tasks such as DNN (hundreds of mW) are several orders-of-magnitude than the current energy-harvesting systems (a few mW), DNN inferences can hardly fit the intermittent computing in an extremely short burst fashion. To address this issue, G. Gobieski et al. propose an intermittent DNN inference framework, *SONIC* [54, 55], to flexibly size tasks to grow or shrink to fit the energy budget. SONIC also uses Alpaca [45] tasks to avoid checkpointing and thus impose very low overheads. K. Qiu et al. propose a low power, reconfigurable ReRAM crossbar architecture, *ResiRCA* to be able to dynamically activate different scaled convolutional computations [11]. At the software layer, a resilient computation scheduling strategy, *ResiShcedule*, provides the optimal tuning combination of loop tiling, ReRAM duplication and pipelining techniques to cope with different power inputs. Moreover, directed by a multi-level power predictor, smooth transition between different computation schedule policies is proposed to maintain partial results instead of discarding them.

5 FUTURE TRENDS

Continuous improvements in the energy harvesting domain can be expected with advances in NVP/NVA circuits, architecture, software and cross-layer co-design. Potential advances are anticipated in the following aspects:

- Circuit designs: emphasizing delivery of higher energy conversion efficiency;
- Specific NVA prototypes: bringing about fundamental change on NVP architecture to meet a large variety of scenario demands (e.g. SNN accelerator);
- Software tools: being further evolved to take full advantage of the new architectural features at compiler and/or OS levels;
- Compatible I/O operations: achieving a fast I/O operation recovery at minimum cost by novel checkpointing or non-interruptable strategies through power intermittence sensing; and
- Simulator systems: being able to simulate variable system-level architectures as well as microarchitectures of NVPs and NVAs with smart energy harvesting power management modules.

6 CONCLUSION

Energy harvesting is a promising power technology for IoT edge devices. With advances in enabling smart nodes in IoTs, NVP designs have evolved to encompass NVA. Integrated with software support, both NVP and NVA should guarantee system correctness and high energy efficiency. This paper provides an overview of the recent studies on the energy harvesting-friendly NVP technology, including NVP and NVA architectures as well as their circuits, correctness-aware checkpointing and non-checkpointing software strategies, power efficiency-driven HW/SW co-designs. We believe this paper will motivate further evolution and optimizations of energy harvesting systems.

7 ACKNOWLEDGEMENT

This work was supported in part by NSF #1822923, NSFC #61872251 and Beijing Advanced Innovation Center for Imaging Technology.

REFERENCES

[1] https://www.ubuntupit.com/top-20-emerging-iot-trends-that-will-shape-your-future-soon/.
[2] A. S. H. Mayue Shi, Eric M Yeatman, "Energy harvesting piezoelectric wind speed sensor," *Journal of Physics: Conference Series*, vol. 1407, pp. 012–044, Nov 2019.
[3] https://assistcenter.org/inertial-energy-harvesting/.
[4] Z. J. Chew and M. Zhu, "Adaptive self-configurable rectifier for extended operating range of piezoelectric energy harvesting," *IEEE Transactions on Industrial Electronics (TIE)*, vol. 67, no. 4, pp. 3267–3276, 2020.
[5] M. Mangrulkar and S. G. Akojwar, "A simple and efficient solar energy harvesting for wireless sensor node," in *ICRCICN'16*, pp. 95–99.
[6] A. Tobola, H. Leutheuser, M. Pollak, P. Spies, C. Hofmann, C. Weigand, B. M. Eskofier, and G. Fischer, "Self-powered multiparameter health sensor," *IEEE journal of biomedical and health informatics*, vol. 22, no. 1, pp. 15–22, 2018.
[7] R. Shigeta, T. Sasaki, D. M. Quan, Y. Kawahara, R. J. Vyas, M. M. Tentzeris, and T. Asami, "Ambient rf energy harvesting sensor device with capacitor-leakage-aware duty cycle control," *IEEE Sensors Journal*, vol. 13, no. 8, pp. 2973–2983, 2013.
[8] M. Gao, P. Wang, Y. Wang, and L. Yao, "Self-powered zigbee wireless sensor nodes for railway condition monitoring," *IEEE TITS'18*, vol. 19, no. 3, pp. 900–909.
[9] M. Xie, C. Pan, J. Hu, C. Yang, and Y. Chen, "Checkpoint-aware instruction scheduling for nonvolatile processor with multiple functional units," in *ASP-DAC'15*, pp. 316–321.
[10] J. Yue, R. Liu, W. Sun, Z. Yuan, Z. Wang, Y. Tu, Y. Chen, A. Ren, Y. Wang, M. Chang, X. Li, H. Yang, and Y. Liu, "7.5 a 65nm 0.39-to-140.3tops/w 1-to-12b unified neural network processor using block-circulant-enabled transpose-domain acceleration with 8.1 × higher tops/mm2and 6t hbst-tram-based 2d data-reuse architecture," in *ISSCC'19*, pp. 138–140.
[11] K. Qiu, N. Jao, M. Zhao, C. S. Mishra, G. Gudukbay, S. Jose, J. Sampson, M. T. Kandemir, and V. Narayanan, "Resirca: A resilient energy harvesting reram crossbar-based accelerator for intelligent embedded processors," in *HPCA'20*, pp. 315–327.
[12] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Zhao, M.-F. Chiang, Y. Yan, B. Sai, and H. Yang, "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," in *ESSCIRC'12*, pp. 149–152.
[13] X. Li, S. George, K. Ma, W. Tsai, A. Aziz, J. Sampson, S. K. Gupta, M. Chang, Y. Liu, S. Datta, and V. Narayanan, "Advancing nonvolatile computing with nonvolatile NCFET latches and flip-flops," *TCAS-I'17*, vol. 64, no. 11, pp. 2907–2919.
[14] S. K. Thirumala, A. Raha, H. Jayakumar, K. Ma, V. Narayanan, V. Raghunathan, and S. K. Gupta, "Dual mode ferroelectric transistor based non-volatile flip-flops for intermittently-powered systems," in *ISLPED'18*, pp. 1–6.
[15] Y. Liu, Z. Wang, A. Lee, F. Su, C. Lo, Z. Yuan, C. Lin, Q. Wei, Y. Wang, Y. King, C. Lin, P. Khalili, K. Wang, M. Chang, and H. Yang, "4.7 a 65nm reram-enabled nonvolatile processor with 6× reduction in restore time and 4× higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic," in *ISSCC'16*, pp. 84–86.
[16] A. Lee, C. Lo, C. Lin, W. Chen, K. Hsu, Z. Wang, F. Su, Z. Yuan, Q. Wei, Y. King, C. Lin, H. Lee, P. Khalili Amiri, K. Wang, Y. Wang, H. Yang, Y. Liu, and M. Chang, "A reram-based nonvolatile flip-flop with self-write-termination scheme for frequent-off fast-wake-up nonvolatile processors," *JSSC'17*, vol. 52, no. 8, pp. 2194–2207.
[17] M. Qazi, A. Amerasekera, and A. P. Chandrakasan, "A 3.4-pj feram-enabled d flip-flop in 0.13-μm cmos for nonvolatile processing in digital systems," *JSSC'14*, vol. 49, no. 1, pp. 202–211.
[18] "Texas instruments. [online]. available: www.ti.com/product/cc1101,"
[19] F. Su, W. Chen, L. Xia, C. Lo, T. Tang, Z. Wang, K. Hsu, M. Cheng, J. Li, Y. Xie, Y. Wang, M. Chang, H. Yang, and Y. Liu, "A 462gops/j rram-based nonvolatile intelligent processor for energy harvesting ioe system featuring nonvolatile logics and processing-in-memory," in *2017 Symposium on VLSI Circuits*, pp. C260–C261.
[20] Y. Zha, E. Nowak, and J. Li, "Liquid silicon: A nonvolatile fully programmable processing-in-memory processor with monolithically integrated reram for big data/machine learning applications," in *2019 Symposium on VLSI Circuits*, pp. C206–C207.
[21] Y. Wang, Y. Liu, S. Li, X. Sheng, D. Zhang, M.-F. Chiang, B. Sai, X. Hu, and H. Yang, "PaCC: A parallel compare and compress codec for area reduction in nonvolatile processors," *TVLSI'13*, vol. PP, no. 99, pp. 1491–1505.
[22] A. Colin and B. Lucia, "Chain: tasks and channels for reliable intermittent programs," in *Proc. ACM Program. Lang. (OOPSLA)*, vol. 51, pp. 514–530, 2016.
[23] B. Lucia and B. Ransford, "A simpler, safer programming and execution model for intermittent systems," *PLDI'15*, vol. 50, no. 6, pp. 575–585.
[24] M. Xie, M. Zhao, C. Pan, H. Li, Y. Liu, Y. Zhang, C. J. Xue, and J. Hu, "Checkpoint aware hybrid cache architecture for nv processor in energy harvesting powered systems," in *CODES+ISSS'16*, pp. 1–10, Oct.
[25] H. Jayakumar, A. Raha, W. S. Lee, and V. Raghunathan, "Quickrecall: A hw/sw approach for computing across power cycles in transiently powered computers," *JETC'15*, vol. 12, no. 1.
[26] K. Ma, Y. Zheng, S. Li, K. Swaminathan, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Architecture exploration for ambient energy harvesting nonvolatile processors," in *HPCA'15*, pp. 526–537.
[27] T. Onuki, W. Uesugi, A. Isobe, Y. Ando, S. Okamoto, K. Kato, T. R. Yew, J. Y. Wu, C. C. Shuai, S. H. Wu, J. Myers, K. Doppler, M. Fujita, and S. Yamazaki, "Embedded memory and arm cortex-m0 core using 60-nm c-axis aligned crystalline indium–gallium–zinc oxide fet integrated with 65-nm si cmos," *JSSC'17*, vol. 52, no. 4, pp. 925–932.
[28] C. Ding, S. Liao, Y. Wang, Z. Li, N. Liu, Y. Zhuo, C. Wang, X. Qian, Y. Bai, G. Yuan, X. Ma, Y. Zhang, J. Tang, Q. Qiu, X. Lin, and B. Yuan, "CirCNN: Accelerating and compressing deep neural networks using block-circulant weight matrices," in *MICRO'17*, p. 395–408.
[29] M. Xie, C. Pan, J. Hu, C. J. Xue, and Q. Zhuge, "Non-volatile registers aware instruction selection for embedded systems," in *RTCSA'14*, pp. 1–9.
[30] Q. Li, M. Zhao, J. Hu, Y. Liu, Y. He, and C. J. Xue, "Compiler directed automatic stack trimming for efficient non-volatile processors," in *DAC'15*, pp. 1–6.
[31] W. Song, Y. Zhou, M. Zhao, L. Ju, C. J. Xue, and Z. Jia, "EMC: Energy-aware morphable cache design for non-volatile processors," *TC'19*, vol. 68, no. 4, pp. 498–509.
[32] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *OSDI'18*, pp. 129–144.
[33] B. Ransford, S. S. Clark, M. Salajegheh, and K. Fu, "Getting things done on computational rfids with energy-aware checkpointing and voltage-aware scheduling," in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, pp. 5–5.
[34] M. Zhao, C. Fu, Z. Li, Q. Li, M. Xie, Y. Liu, J. Hu, Z. Jia, and C. J. Xue, "Stack-size sensitive on-chip memory backup for self-powered nonvolatile processors," *TCAD'17*, vol. 36, no. 11, pp. 1804–1816.
[35] J. Li, M. Zhao, L. Ju, C. J. Xue, and Z. Jia, "Maximizing forward progress with cache-aware backup for self-powered non-volatile processors," in *DAC'17*, pp. 1–6.
[36] W. Fan, Y. Zhang, W. Song, M. Zhao, Z. Shen, and Z. Jia, "Q-learning based backup for energy harvesting powered embedded systems," in *DATE'20*, pp. 1247–1252.
[37] B. Ransford and B. Lucia, "Nonvolatile memory is a broken time machine," in *Proceedings of the workshop on MMSPC'14*, pp. 1–3.
[38] M. Xie, M. Zhao, C. Pan, J. Hu, Y. Liu, and C. J. Xue, "Fixing the broken time machine: consistency-aware checkpointing for energy harvesting powered non-volatile processor," in *DAC'15*, pp. 184:1–184:6.
[39] M. Surbatovich, L. Jia, and B. Lucia, "I/O dependent idempotence bugs in intermittent systems," *Proc. ACM Program. Lang. (OOPSLA)*, vol. 3, 2019.
[40] W.-M. Chen, T.-S. Cheng, P.-C. Hsiu, and T.-W. Kuo, "Value-Based Task Scheduling for Nonvolatile Processor-Based Embedded Devices," in *RTSS'16*, pp. 247–256.
[41] W.-M. Chen, P.-C. Hsiu, and T.-W. Kuo, "Enabling Failure-resilient Intermittently-powered Systems Without Runtime Checkpointing," in *DAC'19*, pp. 104:1–6.
[42] W.-M. Chen, Y.-T. Chen, P.-C. Hsiu, and T.-W. Kuo, "Multiversion Concurrency Control on Intermittent Systems," in *ICCAD'19*, pp. 1–8.
[43] W.-M. Chen, T.-W. Kuo, and P.-C. Hsiu, "Enabling failure-resilient intermittent systems without runtime checkpointing," *TCAD'20, Early Access*.
[44] https://github.com/EMCLab-Sinica/Intermittent-OS/.
[45] K. Maeng, A. Colin, and B. Lucia, "Alpaca: Intermittent execution without checkpoints," *Proc. ACM Program. Lang. (OOPSLA)*, vol. 1, 2017.
[46] K. Maeng and B. Lucia, "Adaptive low-overhead scheduling for periodic and reactive intermittent execution," in *PLDI'20*, p. 1005–1021.
[47] A. Colin, E. Ruppel, and B. Lucia, "A reconfigurable energy storage architecture for energy-harvesting devices," in *ASPLOS'18*, pp. 767–781.
[48] K. Ma, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Dynamic machine learning based matching of nonvolatile processor microarchitecture to harvested energy profile," in *ICCAD'15*, pp. 670–675.
[49] K. Ma, X. Li, S. R. Srinivasa, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Spendthrift: Machine learning based resource and frequency scaling for ambient energy harvesting nonvolatile processors," in *ASP-DAC'17*, pp. 678–683.
[50] K. Ma, X. Li, J. Li, Y. Liu, Y. Xie, J. Sampson, M. T. Kandemir, and V. Narayanan, "Incidental computing on iot nonvolatile processors," in *MICRO'17*, pp. 204–218.
[51] K. Ma, J. Li, X. Li, Y. Liu, Y. Xie, M. Kandemir, J. Sampson, and V. Narayanan, "Iaa: Incidental approximate architectures for extremely energy-constrained energy harvesting scenarios using iot nonvolatile processors," *IEEE Micro'18*, vol. 38, no. 4, pp. 11–19.
[52] K. Ma, X. Li, M. T. Kandemir, J. Sampson, V. Narayanan, J. Li, T. Wu, Z. Wang, Y. Liu, and Y. Xie, "NEOFog: Nonvolatility-exploiting optimizations for fog computing," in *ASPLOS'18*, pp. 782–796.
[53] A. Y. Majid, C. D. Donne, K. Maeng, A. Colin, K. S. Yildirim, B. Lucia, and P. Pawełczak, "Dynamic task-based intermittent execution for energy-harvesting devices," *TSN'20*, vol. 16, no. 1.
[54] B. L. Graham Gobieski, Nathan Beckmann, "Intermittent deep neural network inference," in *SysML Conference*, pp. 1–3, 2018.
[55] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *ASPLOS '19*, p. 199–213, 2019.