The Pennsylvania State University
The Graduate School

SELF-POWERED INTERNET-OF-THINGS NONVOLATILE PROCESSOR AND SYSTEM EXPLORATION AND OPTIMIZATION

A Dissertation in
Computer Science and Engineering
by
Kaisheng Ma

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2018

The dissertation of Kaisheng Ma was reviewed and approved* by the following:

Vijaykrishnan Narayanan
Distinguished Professor in Department of Computer Science and Engineering, The Pennsylvania State University
Dissertation Advisor, Co-Chair of Committee

John (Jack) Sampson
Assistant Professor in Department of Computer Science and Engineering, The Pennsylvania State University
Co-Chair of Committee

Mahmut Taylan Kandemir
Professor in Department of Computer Science and Engineering, The Pennsylvania State University

Soundar Kumara
Allen E. Pearce and Allen M. Pearace Professor in Industrial and Manufacturing Engineering, The Pennsylvania State University

Yuan Xie
Professor in Department of Electrical and Computer Engineering, University of California at Santa Barbara
Special Member

Chita Das
Distinguished Professor of Computer Science and Engineering, Head of Department of Computer Science and Engineering, The Pennsylvania State University

*Signatures are on file in the Graduate School.

Abstract

Every shift in the way our devices are connected or powered brings with it a potential for revolution in the usage and capabilities of the systems built around them. Just as the transition from wired to wireless telephones led to unprecedented changes in our communications and the shift from wall-power to battery-power transformed our expectations for computational systems, the shift from battery-powered systems to self-powered systems promises to fuel the next revolution in the Internet of Things (IoT). The ability to utilize ambient, scavenged energy, such as solar energy, radio-frequency (RF) radiation, piezoelectric effect, thermal gradients, etc., can liberate IoT devices from the lifetime, deployment, and service limitations of a fixed battery. However, the power supplied by energy harvesting sources is highly unreliable and dependent upon ambient environment factors. Hence, it is necessary to develop specialized IoT architectures and systems that are tolerant to this power variation, and also capable of making forward progress on their computation tasks.

This dissertation provides a holistic exploration in applying nonvolatility into the processor and system design in energy harvesting application scenarios. Various techniques are proposed for optimization in nodes level and system level. To begin with, architectural design space is explored in a nonvolatile processor, in which nonvolatility feature is designed within a processor to overcome the unstable power supply through distributed energy and time efficient backup and recovery operations. This dissertation focuses on design space of different architectures, different input power sources, and policies for maximizing forward progress. It is presented that different complexity levels of nonvolatile microarchitectures provide best fit for different power sources and even different trails within same power source. To further overcome this challenge of architectural dependency on application scenario, various techniques are proposed in this dissertation, including frequency scaling and resource allocation to dynamically adjust the microarchitecture to achieve the maximum forward progress. Furthermore, noticing that such nodes usually perform similar operations across each new input record,

I observe optimization opportunities for mining the potential information in buffered historical data, at the potentially lower effort, while processing new data rather than abandoning old inputs due to limited computational energy. This approach is proposed as incidental computing, and synergies between this approach and approximation techniques are explored. After a progress of nonvolatilizing pieces within an energy harvesting node, IoT fog computing in Wireless Sensor Networks (WSN) is re-optimized in this dissertation as one application example to perform system-level optimization. I discuss how nonvolatility features including nonvolatile processors and nonvolatile RF can benefit the system, and how other optimizations like load balance under unstable power, as well as increasing nodes density for the quality of service can be applied to the fog computing system.

This dissertation provides a holistic exploration in designing a battery-less energy harvesting system, from individual parts within a node to an applicable energy harvesting wireless sensor network system optimization. With the increasing demands of maintains-free IoTs with green energy, this dissertation foresees the the vision of self-powered IoTs in the near future.

Table of Contents

List of Figures

List of Tables

Acknowledgments

Foremost, I would like to express my most sincere gratitude to my primary advisor Prof. Vijaykrishnan Narayanan and co-advisor Prof. John (Jack) Sampson for the continuous support of my Ph.D. study and research, for their patience, motivation, enthusiasm, and immense knowledge. The doors to Prof. Vijaykrishnan Narayanan and Prof. John Sampson offices were always open whenever I ran into a trouble spot or had a question about my research or writing. Some important moments last forever: I still remember the night before HPCA 2015 submission. All of you stayed in the lab until 4 AM to guide me for the paper submission, from idea deployment to paper structure, from writing to proof-reading. That was my first-time systematical training to write an architectural paper. I still remember the words from Dr. Vijay N. before HPCA best paper candidate talk: ”People judge you not from where you are from, but from what you do, and how well you are doing it.” I still remember Dr. Sampson's guidance in idea deployment and various supports and suggestions in study, research, and life. He has been being so dedicated to his role as my secondary supervisor.

Special acknowledgment mention goes to Dr. Yuan Xie, not only for his tremendous academic support but also for giving me so many wonderful opportunities and recommendations. He consistently supports my research and life, guiding me in the right direction and encouraging me whenever he thinks I need it.

I am especially grateful to Prof. Mahmut Taylan Kandemir, for being my dissertation committee member, for his consistent help in both research cooperation and my job hunting.

I would also like to thank my thesis committee member Prof. Soundar Kumara, for his encouragement, and insightful comments.

I would like to extend thanks to the many professors within the CSE department including Dr. Mary Jane Irwin, Dr. Chita. R. Das and Dr. Mahmut Taylan Kandemirfor for their help during my Ph.D.

My sincere thanks also go to my collaborators around the world, including Prof. Yongpan Liu, and Prof. Suman Datta, who generously contributed to the work presented in this dissertation.

My thanks also go to Dr. Qiong Cai, and Dr. Cong Xu in HP, Dr. Ren Wu in Novumind, for offering me the summer internship opportunities in their groups and leading me working on diverse exciting projects.

I am also hugely appreciative of my fellow labmates, especially Dr. Xueqing Li, Dr. Huichu Liu, and Dr. Yang Xiao, for guiding me in research and helping me in life.

I would like to thank following fundings which have been supporting my research: NSF ASSIST and NSF Expeditions of Visual Cortex on Silicon.

Finally, but by no means least, I would also like to thank my parents for their wise counsel, sympathetic ears and encouraging words. I dedicate this dissertation to them!

Chapter 1

Introduction

By 2020, 50 billion Internet of Things (IoTs) devices are expected to be connected to the Internet [5, 6]. It is predicted that these connected IoTs will generate more than 11000 billion dollars economic value by 2025 [7]. In fact, IoTs have already exhibited tremendous potential to bring new opportunities in some domains, for instance, the wearable devices [8–10], the smart home [11], and the smart industry [12].

With more application scenarios deployed and more devices connected to the Internet, a vital challenge is observed that can potentially limit the wider application of IoTs – battery life. To begin with, it is inconvenient to charge the batteries in some scenarios. Today, most of the wearable devices need to be charged once a day or several days [13]. In addition, renewing batteries for those IoTs in huge amount in smart industry [14] or in large distributed areas as in the case of environmental and infrastructure monitoring [15–17] can be very costly, and sometimes unrealistic. Furthermore, the existence of batteries significantly frustrates applications with volume and/or safety concerns, as in the case of implantable devices [18].

Energy harvesting emerges as one of the promising techniques to solve problems brought by batteries. By harvesting energy from the ambient environment, including solar, vibration, RF and thermal energy, the lifetime of IoTs can, in principle, be infinite. This can in turn extend the application domain of IoTs to, for example, smart parking [19, 20], environment monitor [16], smart agriculture [21, 22], implantable devices [18], etc.

On the negative side, energy harvesting can provide only unstable power supply, and the magnitude of this instability strongly depends on the application features and execu-

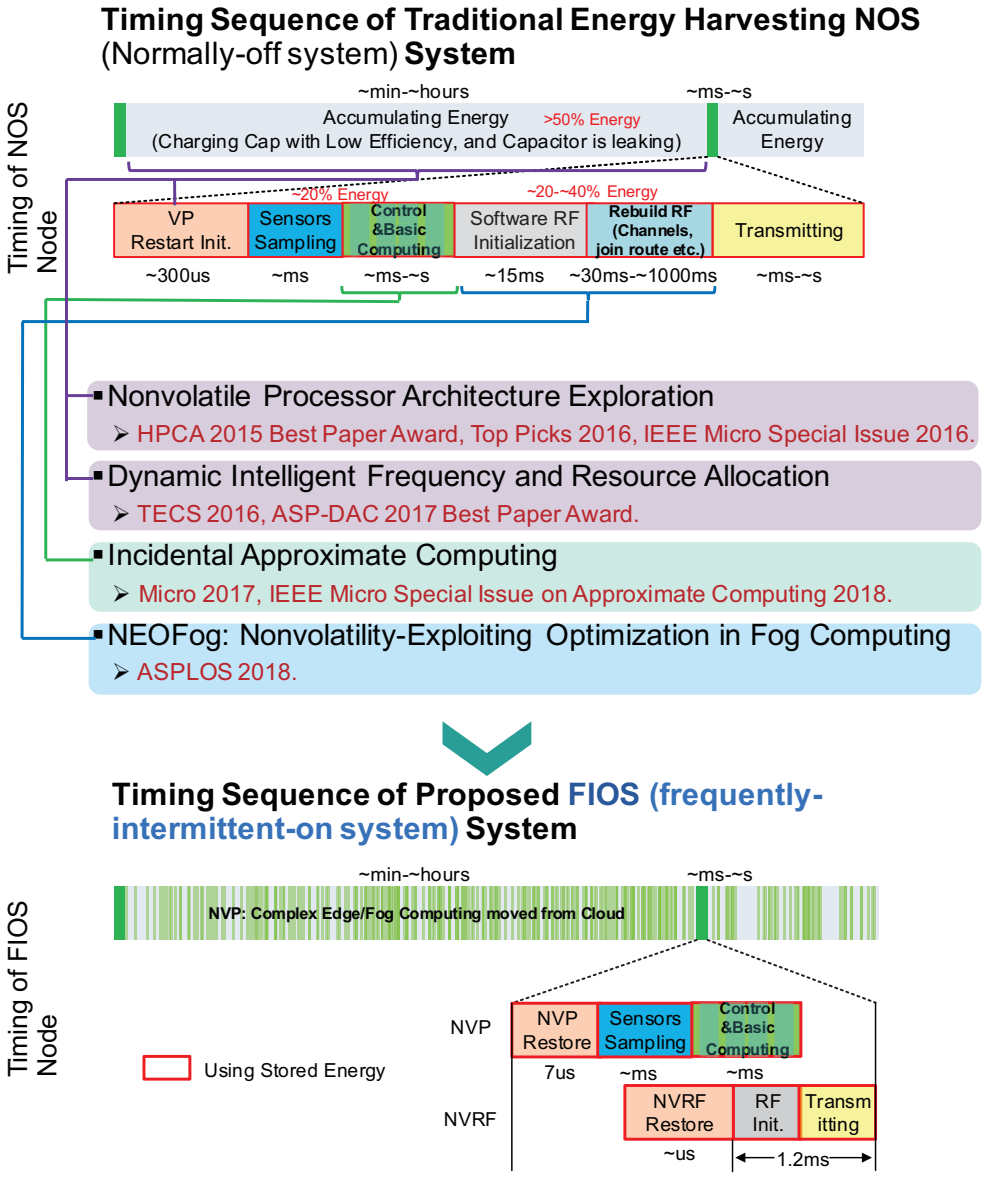


Figure 1.1. Timing squence expansion of a traditional energy harvesting sensor node and according optimization

tion environment characteristics [23]. Traditional solutions either adopt ”Always-on” topology in which the lowest power is the only optimization target or ”Wait-compute” mechanism [24–26] in which a capacitor is employed to accumulate energy for one shot use. Both solutions result in low energy efficiency.

I propose a ”Frequently-intermittent-on” new fog computing topology which uses only a small on-chip capacitor and nonvolatile processors (NVPs) [27]. Most of my Ph.D. work essentially cover architectural level optimizations and exploration of non-

volatility features applied in energy harvesting IoTs, as shown in Figure 1.1. A very first work for architectural exploration of NVPs was published in HPCA 2015 [23], following with some machine learning-based optimizations for dynamic frequency and resource allocation [28]. To further improve the forward progress, which was defined as instructions finished with given power traces, incidental techniques was proposed [29]. Observing that nonvolatility is becoming a prominent feature of many architecture / microarchitecture components, I further proposed novel fog computing infrastructures built upon wireless sensor networks [27].

But still, there are many unexplored domains in the system level and cross-layer optimizations for energy harvesting IoT architectures. Research opportunities also lay in beyond traditional Von Neumann architectures to further reduce the power consumption in extremely low power energy harvesting scenarios.

1.1 Optimization Againt Traditional Stereotypes

My Ph.D. research represents a significant departure from traditional stereotypes of state-of-the-art energy harvesting systems. And at each step of my research, fabrication was followed to prove the architectural exploration findings, and my efforts finally culminated in novel energy harvesting based fog computing systems.

Traditional stereotype 1: Computation on energy harvesting fog node is unreliable.

Proposed approach: Nonvolatile processor (NVP) for reliable and continuous computing, resulting in 2.2X-5X more efficient than wait-compute volatile processor (VP).

Architecture exploration of nonvolatile processors was my first project, focusing primarily on different nonvolatility degrees vs. rollbacks across different complexity micro-processors like non-pipelined, n-stage-pipelined, and out-of-order processors. Through simulation calibrated on a fabricated first version nonvolatile processor from our collaborators [30], some unexpected findings are explored: First, optimizing for low power is not the same as optimizing for maximum forward progress in energy harvesting scenarios, which means a new paradigm of design methodology and tradeoffs should be deployed in designing IoT energy harvesting nonvolatile processors. In addition, intuitively OoO is not considered for energy harvesting scenarios, but I prove that it can be

a preferable option in some cases. The best architecture for NVP is essentially related to power sources and even the power traces in different application scenarios. Realizing this, I further propose a machine learning-based dynamic architecture to select the best microarchitecture in a mix-architecture nonvolatile processor [31]. Of all the proposed microarchitectures, ODSB (On-demand-selected-backup) microarchitecture was adopted in our next version of the nonvolatile processor, and the measured results proved its success [32].

Traditional stereotype 2: Computation on energy harvesting fog node is expensive.

Proposed approaches: Incidental Computing: 4.3X than precise NVP; Frequency and Resource Allocation: 2.1X than NVP.

Due to the weak density of energy in the ambient environment, the energy that can be harvested is usually very unstable/weak. Consequently, approximate computing can be a good fit to still make the system running, rather than letting the energy leaked when accumulating in energy storage devices. Observing that (i) *in many IoT applications, temporal and interactivity requirements can make the quality of partial results, or even the existence of any response at all, more important than the fraction of instructions needed to eventually produce a "best-quality" result*, and (ii) data importance drops over time, I, as the lead author of the work, first introduced the incidental computing concept, wherein older computation is carried out in a best-effort fashion during the execution of newer computations. For the energy-harvesting NVP scenario, this is done through bitwidth-oriented approximation techniques in the data-path, memory, and backup-recovery modules to divide power and resources and provide differential guarantees of output quality between the current and prior computations. I also propose incidental recomputing, wherein the quality of older computations targeted for incidental computing can be gradually improved iteratively if picked up over multiple incidental computing passes. I proposed incidental backup with several retention time matching models and supporting write circuits that can reduce the energy of backup operations through matching the retention time to the combination of the duration of power emergencies and the impacts of reduced fidelity to overall approximation quality.

To further improve the efficiency of nonvolatile processors, a machine learning-based dynamic resource and frequency based microarchitecture for transmitting as much as harvested energy into instructions finished and running the processor at the best

energy efficiency with the most proper resources [28]. The proposed frequency scaling [28, 33] was verified in board level for the next generation NVP.

Collectively, the incidental approximation approaches improved forward progress by 4.28x improvement within tolerable quality loss, and frequency and resource allocation add on another 2.1X. Around 10X forward progress vastly extend the application domain of energy harvesting IoTs, making some applications traditionally impossible due to limited harvested energy but high processor starting threshold, now possible. To my knowledge, this was the first application of approximate computing ideas to the NVP and energy harvesting domain.

Traditional stereotype 3: Communication between NVP and peripherals are critical problems, and RF time and energy dominates in nodes.

Proposed approach: NVRF: 27X faster startup speed, 6.3X packages.

Optimizing forward progress of processors alone cannot solve the system level challenges in energy harvesting systems. The peripheral recovery, which in many systems actually dominates the node energy and time consumption [18, 25, 34], resulting in significant performance degradation, is another critical problem in energy harvesting scenarios that remains unsolved and is an important but last step to bring real energy harvesting systems to life [35, 36].

Noting the different optimization directions between dynamic checkpointing within NVPs and static checkpointing requirements of I/O communication, we (where I supervised a Ph.D. student in Tsinghua, and I was the second author) propose an HW/SW co-optimization solution to achieve reliable and efficient concurrent peripherals' recoveries. By classifying the I/O checkpointing into nonvolatility assisted re-initialization, data re-transmission, and dynamic program checkpointing, the I/O states can be recovered efficiently and reliably. A verification of NVRF chip is fabricated to verify the proposed architecture, which shows 27X faster startup speed, 6.3X packages.

Traditional optimiztion target: a single node.

Beyond single node optimization target: system level optimizations in an energy harvesting wireless sensor network.

After challenging these three stereotypes, I focused on wireless sensor network system-level optimization. Nonvolatile processors have emerged as one of the promising solutions for energy harvesting scenarios, among which Wireless Sensor Networks (WSN) provide some of the most important applications.

In a typical distributed sensing system, due to differences across locations, energy harvester angles, power sources, etc., different nodes may have a different amount of energy ready for use. While prior approaches have examined these challenges, they have not done so in the context of the features offered by nonvolatile computing approaches, which disrupt certain foundational assumptions. I proposed a new set of nonvolatility-exploiting optimizations and embody them in the NEOFog system architecture. I discuss shifts in the tradeoffs in data and program distribution for nonvolatile processing-based WSNs, showing how non-volatile processing and non-volatile RF support alter the benefits of computation and communication-centric approaches. I also proposed a new algorithm specific to nonvolatile sensing systems for balancing load in computation. Collectively, the NV-aware optimizations increase the ability to perform in-fog processing by 4.2X and can increase this to 8X if virtual nodes are 3x multiplexed.

1.2 Preview of the Dissertation

Over the course of this dissertation, I will examine the tradeoffs from architectural level to the system level, focusing on architectural exploration for overcoming unstable power supply, energy efficiency of the energy harvesting system, tradeoffs in quality of service and approximation, and wireless sensor network level optimizations.

Chapter 2 introduces the background including components of a typical energy harvesting system, motivation of nonvolatile processors for intermittent computing, and an energy model for the node system.

Chapter 3 discusses architectural level designs and optimizations for ambient energy harvesting NVPs and provides the design guidelines mapping from power sources to architecture level selection. Aiming at time and energy optimization, I have proposed and simulated various architecture level solutions for non-pipelined, five-stage-pipeline and out-of-order processor architectures, and have discussed the trade-offs between them over the course of this dissertation.

To better optimize the NVP node, bottleneck resource prediction and frequency scaling as explored in Chapter 4, are applied to improve the forward progress. Powering on some resources to mitigate some bottlenecks can actually significantly increase the NVP's parallel density. Through smart DFS, the energy used for forward progress computation increases. I show that the proposed spendthrift solution can achieve 2.08X

forward progress than best case of OoO NVP with static configuration and static frequency.

Chapter 5 describes the exploration of applying approximate techniques to NVPs for tradeoffs between forward progress and quality of service. Adopting approximate computing approaches in NVPs not only improves the forward progress that NVP systems can offer, but it also provides a means for NVPs to optimize for responsiveness as well as overall efficiency, and synergizes with unique features of NVP execution, namely, frequent backup and recovery operations. I introduce the concept of "incidental computing" to address opportunistic responsiveness versus quality tradeoffs under unstable power income, and implement and evaluate an instantiation of the incidental computing approach based on memory and datapath approximation within an NVP. For a prototypical IoT workload, the combination of all of the techniques improves forward progress by an average of 4.28x over a baseline "precise" NVP, of which 1.4x is attributable to NVP-specific backup and restore approximations.

Realizing that the nonvolatile features applied in NVP and NVRF etc. have significantly impacted the node level simulation, which makes the traditional assumptions that edge node level computing is unreliable and expensive in energy not hold. In order to better utilize the new advantages brought by nonvolatility in edges, I propose NEO-Fog - a system-level optimization to better take the advantages of reliable and relatively efficient in fog computing in Chapter 6. Specifically, I optimize the programs from RF energy dominate to computation intensive, from normally-off system to frequently-intermittent-on system, to better utilize the new opportunities brought by nonvolatility. Beyond nodes level, considering the unbalance and time-varying income power and stored energy level of each node and different the energy requirement for workloads, I optimize a distributed load balance algorithm specially optimized for unstable power supply in an intra-chain level to balance the loads and further increase the computation done by edges. I show that it can boost fog computing from 1.9X to 2.1X compared to the system running a traditional load balance. I also propose architectural supports for increasing nodes density for the better quality of service by time-division multiplexing of NVRF. 2X QoS is observed with 3X density increment in a very low energy income situation.

Chapter 7 surveys other approaches in energy harvesting IoTs, including nonvolatility in processors, architectural aspects of energy harvesting, frequency and resource

scaling, active and passive checkpointing, approximate computing and storage, non-volatility in RF control and other low-power RF technologies, load balancing on WSN, and virtualization for the quality of service.

Finally, in Chapter 8 I summarize the dissertation.

This dissertation provides a holistic exploration in applying nonvolatility into the processor and system design in energy harvesting application scenarios. Various techniques are proposed for optimization in nodes level and system level. To validate the simulation infrastructure, I compare and calibrate the simulation results against several fabricated non-volatile processors. The proposed architectures and optimizations, in turn, provide guidance for SOC chip and system design, which further proves the feasibility of the proposed architectures. This dissertation is a first holistic exploration of ambient energy harvesting processor and system design.

Chapter 2

Background

Every shift in the way our devices are connected or powered brings with it a potential for revolution in the usage and capabilities of the systems built around them. Just as the transition from wired to wireless telephones led to unprecedented changes in our communications and the shift from wall-power to battery-power transformed our expectations for computational systems, the shift from battery-powered systems to self-powered systems promises to fuel the next revolution in the internet of things (IoT). The ability to power IoTs using ambient scavenged energy liberates them from the lifetime, deployment and servicing limitations of a fixed battery: Tens of billions of IoT devices are expected to pervade consumer, industrial and public services by the end of the decade [37]. Some of such systems have already been successfully deployed, and some are very reachable in the near futher. The applications that make use of this paradigm are diverse, including 1) area monitoring, eg. the position of the enemy; 2) environmental monitoring; 3) industrial monitoring; 4) medical and healthcare monitoring; 5) traffic control systems; 6) underwater acoustic sensor networks and 7) near-body wearable device networks.

There are, however, several drawbacks in relying on ambient sources of energy for such computing purposes. Most of these energy sources operate at relatively low conversion efficiencies, since only a small fraction of the total transmitted power can be tapped. In addition, they are not reliable energy sources, since external factors could cause a disruption in supply.

This chapter is organized as follows. First it provides a brief overview of typical energy harvesting systems, ambient power sources that could potentially be harvested as well as the factors involved in the designing the processing element. The follow-

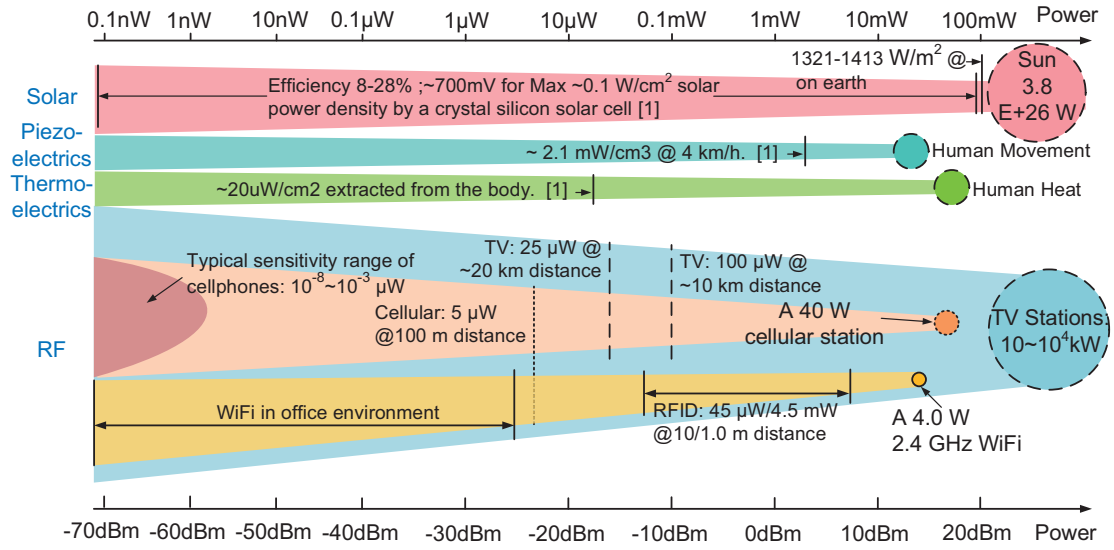


Figure 2.1. Typical power ranges of ambient sources

ing up section examines the various system level background including checkpointing, wait-compute on a volatile processor, front-end circuits etc. for better energy-efficient designs.

2.1 Ambient Energy Sources: Weak and Unstable

Typical ambient energy sources that could be harvested to power an embedded system include solar energy, radio-frequency (RF) radiation, piezoelectric effect and thermal gradients [38], as shown in Figure 2.1. These sources can be classified according to three characteristics: signal magnitude, variability in signal strength, and granularity of variation/intermittency frequency. Research also indicates that implantable biofuel cells (BFCs) are able to generate electric power from sugars found in the body fluid of an insect [39]. A comprehensive comparison among these power sources can be found in reference [40] and reference [41].

Figure 2.2(a) shows the power traces for four typical ambient energy sources. The RF energy is obtained by measuring the power of the frequency spectrum from a TV station, the piezo energy is measured through devices fixed on a bike, the thermal energy is generated from characterizations described inreferences [42–44] and the solar trace is obtained using data from MIDC [45]. It is observed that there is substantial variation in

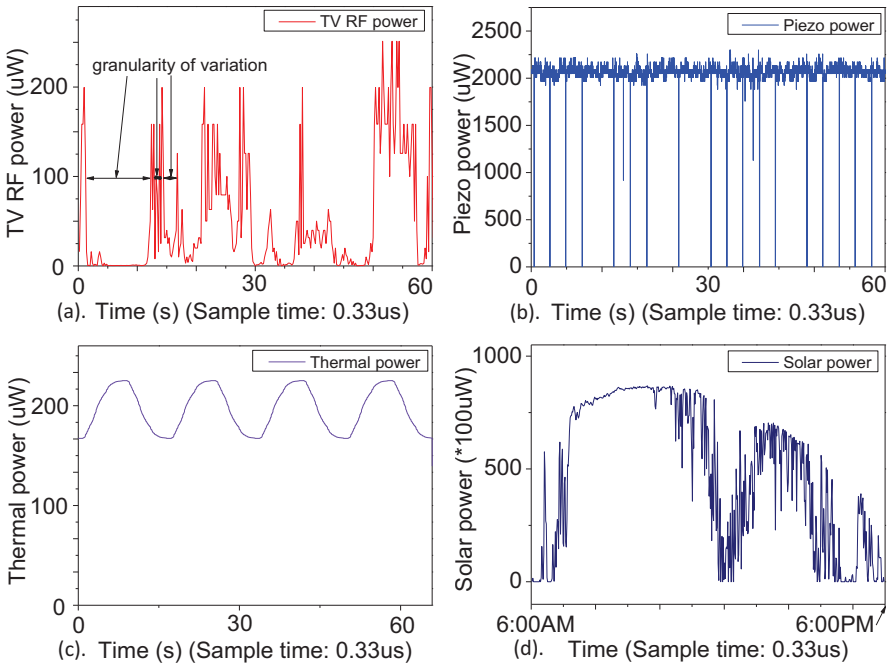


Figure 2.2. Power traces a)TV RF b) Piezo c) Thermal d) Solar

power, even over a few milliseconds for RF in Figure 2.2(a) with the ratio between the maximum and minimum power over this period around 250× [38, 46, 47]. The piezo power is more stable than RF with just some short power loss in Figure 2.2(b). The thermal power, shown in Figure 2.2(c), is even more stable, due to the gradual nature of temperature variation. Variation in solar power, seen in Figure 2.2(d) is contingent on the weather conditions and orientation of the solar cell.

Another feature is the intermittency frequency that influences how soon the power drops below a given threshold as shown in Figure 2.2(a). The intermittency frequency decides the backup and recovery overheads. Sources with periodic behavior, like Figure 2.2(b), facilitate prediction of power loss and enable efficient scheduling of tasks.

2.2 Typical Energy-harvesting System Structures

With the development of the Internet of Things (IoT), smart cities, and implantable and wearable applications, extremely low-power systems powered by ambient energy sources are gaining popularity. Figure 2.3(a) shows an archetypical system structure consisting of (a) an energy harvesting and power management block, (b) a digital signal

processor which is usually implemented using a microcontroller unit (MCU), and (c) the wired or wireless I/O interface. The capacity and implementation of the energy storage medium is also critical to the system design because it directly affects the trade-off between leakage and other overheads and the maximum stored power. In later sections, this trade-off will be discussed in more detail.

It is also noted that different energy sources require different energy harvesters for power conversion. For example, the output of a solar cell is a DC signal, while the RF signal and the output of piezoelectric-based systems are AC signals which require an extra AC-DC rectifier. When the input power is weak, the output voltage may also be low and potentially require an extra DC-DC voltage booster [38].

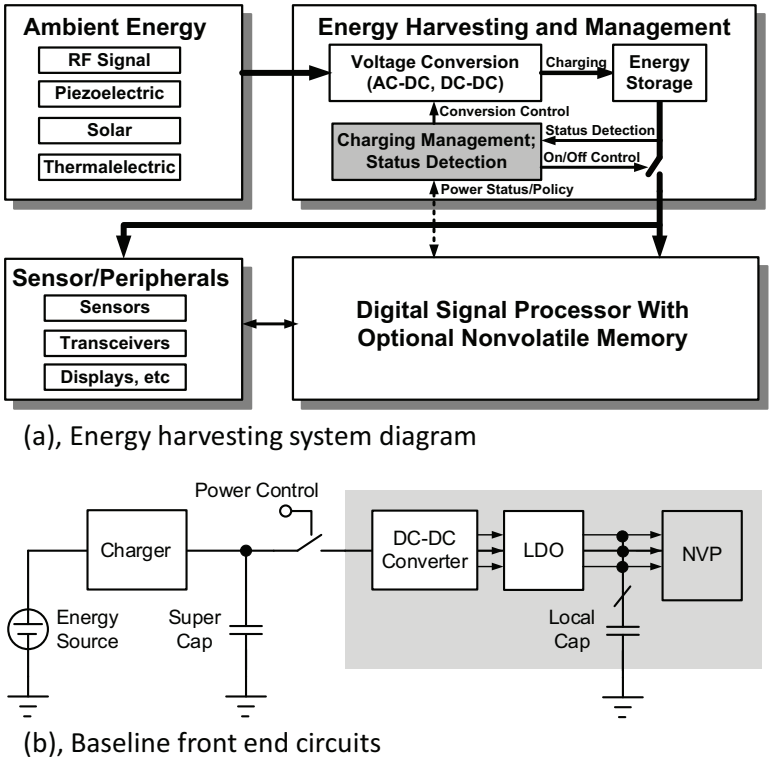


Figure 2.3. Baseline front end circuits

The baseline energy harvesting block is illustrated in Figure 2.3. Subsequent to the AC-DC or DC-DC conversion, an MPP tracking (MPPT) interface is employed to control the charging power for the highest power-conversion efficiency from the energy harvester.

2.3 Why Not Do Wait-compute on A Volatile Processor with A Large Capacitor?

On account of these limitations, most current energy harvesting platforms tend to restrict themselves to applications that require relatively simple signal capturing mechanisms involving minimal computation and processing with a large capacitor for energy stroage. And the system starts only when there are ample energy stored. Such traditional strategy in energy-harvesting systems is to employ a volatile low power MCU or an MCU with checkpointing capability (e.g., FeRAM MSP430 is used in [48]) that waits before starting to execute while charging an energy storage device which must be large enough to store sufficient energy to complete an entire logical work unit, such as an image frame [48]. Systems operating on this paradigm will alternate between periods where they accumulate energy in the energy storage devices (ESD), and ones where powering the system drains the energy. While such a system is able to offer strong guarantees for execution once execution begins, this conventional solution has several limitations, including energy conversion efficiency overheads brought by frequently charging and discharging the capacitor, capacitor leakage [48, 49], minimum charging current(e.g. 20uA for the GZ115 [49]), and slow charging curve [49]). Moreover, if the incoming unit of work is too large, the incoming power may not be sufficient compared to leakage in the ESD [48], or there may be long periods of complete power outage that drain the accumulated charge, and it may take arbitrarily long to reach the threshold for beginning execution.

An alternative execution paradigm is to utilize only a small on-chip capacitor, i.e., one sufficient for backup operations and employ an NVP. This reduces capacitor leakage, and can improve front-end conversion efficiencies by mitigating the overheads of moving charge into and out of a large energy storage device at the cost of additional system complexity for the NVP, more complicated guarantees on the granularity of work accomplished once an execution period begins and the overheads imposed during more frequent backup and restore events during the execution of each logical unit of work. The two approaches can be seen as similar if the logical unit of work is at an instruction or similar granularity, thereby minimizing both charging time and charge lost if a shortfall occurs during a charging period between backup and recovery. Hybrid approaches have also been proposed. For example, Sheng et. al. propose a dual channel front-end

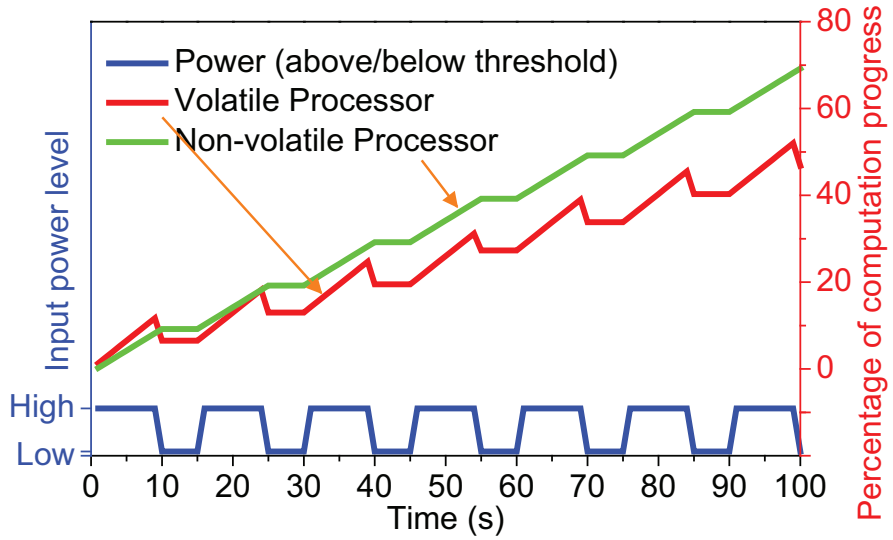


Figure 2.4. VP with checkpointing v.s. NVP processing progress comparison

solution to overcome low charging efficiency [50] in which they design another power channel to bypass the energy storage device and connect directly to the load, and Ma et. al. extend prior NVP models to maintain the capacitor energy level [33] within a bounded range for charging efficiency during execution rather than greedily consuming energy. *Thus, the key energy tradeoff between the two approaches is between the energy wasted on charging and discharging a capacitor with leakage and the backup and recovery overheads of NVP.*

It is observed through simulation and validation that the NVP-based execution approach can outperform the wait-compute scheme by 2.2X-5X.

2.4 Volatile with Checkpointing or Nonvolatile?

Since wait-compute scheme is low efficiency in front-end circuits, a direct path from source directly to load is one possible to improve the efficiency. While this brings another challenge: unstable power supply.

Figure 2.4 illustrates the difference in the behavior of a volatile processor with periodic checkpointing to an external non-volatile memory and a completely nonvolatile processor when working under variable power source conditions. While both processors can only run when the input power exceeds a certain threshold, the volatile processor does not retain the instantaneous state of the system when the power drops below the

threshold, resulting in a forced rollback from previously checkpointed state. This could limit the amount of forward progress from being made.

2.5 An Energy Model for Energy Harvesting System

A key challenge in designing battery-less electronics with energy scavenged power sources is the erratic and unreliable power supply derived from ambient sources. One solution is to add an energy storage capacitor, such as a super-capacitor to smooth the power. Figure 2.6 shows an abstracted charging metaphor that highlights the key components of the systems considered in this paper and Figure 2.5 illustrates an energy trace harnessed from an RF power source stored in a capacitor accounting for its leakage. Since the capacitor leaks, periods during which the power harnessed is less than the leakage power causes the overall stored energy to decrease. Conversely, leakage rates and size/weight restrictions bound the practical scale of capacitor volume, so the capacitor may saturate and be unable to store additional energy during periods of high input power.

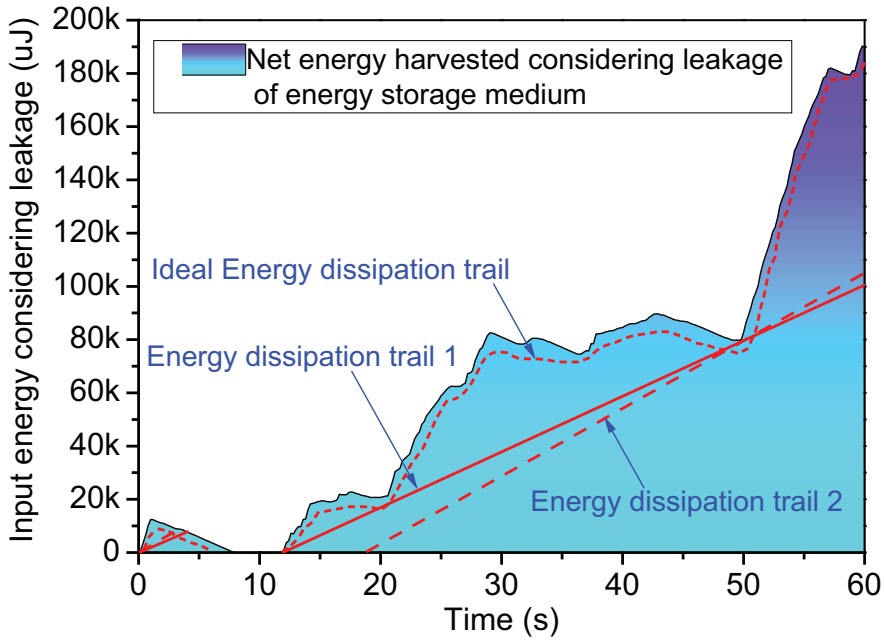


Figure 2.5. A comparison of energy input and consumption trails for a TV RF power trace

Three key insights into the dynamics of energy storage in energy-harvesting systems are that, firstly, over the highly varying input power ranges that these systems must op-

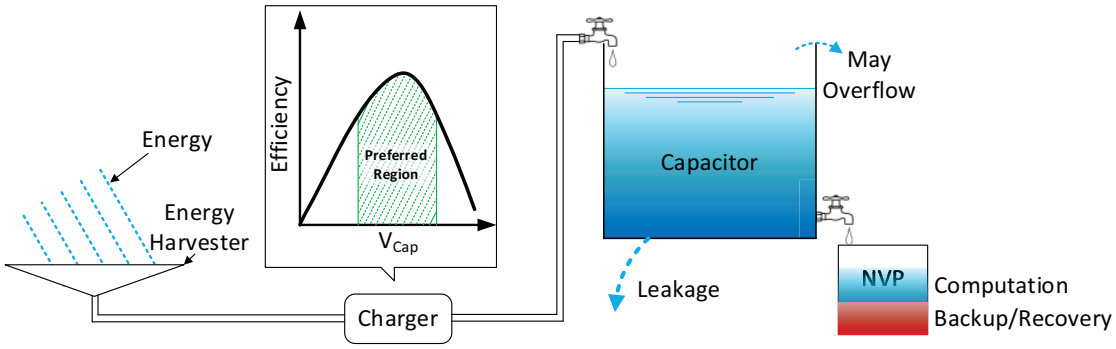


Figure 2.6. Plumbing analogy for energy harvesting and consumption

erate, they will frequently encounter both periods where the energy lost from capacitive storage is greater than that replaced by energy harvesting and periods where short term increases in input power provide more energy than a practical capacitor can store. Second, the effects of capacitive leakage, finite capacitor storage, charging losses and other front-end power components are large enough to require co-optimization when considering processor or other compute-engine optimizations for these platforms. Third, as Figure 2.5 depicts, policy management of the power demand at the processor, the load for this front end, provides substantial leverage in mitigating both capacitive underflow and overflow by changing the slope of energy consumed. Changes in processor policies, such as DVFS can also affect front-end efficiency: For example, since DC-DC conversion losses depend on the difference in voltage, actively depleting or restoring the energy storage capacitor to a particular voltage range will impact the efficiency of the system in harvesting future power.

2.6 Processing Quality Tradeoffs

In many IoT applications, temporal and interactivity requirements can make the quality of partial results, or even the existence of any response at all, more important than the fraction of instructions needed to eventually produce a ”best-quality” result.

Adding a ”quality knob” provides flexibility in an NVP, where the need to make conservative decisions regarding energy reserves for backup operations can otherwise impose substantial overheads on execution. In an NVP, if the effort needed to ensure preservation of data is sufficiently reduced, some power emergencies may be avoided

altogether, improving response timeliness. Moreover, in addition to natural synergies with power management, accepting variable quality responses frees a harvesting system to apportion effort with respect to the continued relevance of the data being processed: If an NVP has been without power for some substantial time, resuming work on the input it was processing when power failed may have lower utility, from an application perspective, than moving on to processing the newest input. Discovering the optimal allocation/schedule for an NVP would depend not only on application-specific semantics, but also future knowledge of unpredictable power income.

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

Chapter 3

Nonvolatile Processor Architecture Exploration

This chapter is organized as follows. First it provides a brief overview of configuration assumptions involved in the designing the processing element. The following up sections examine the various architectural considerations that arise when I extrapolate the existing system to use-case scenarios that require more complex, faster and energy-efficient designs. Then the simulation infrastructure is described and validated with a fabricated NVP. The following sections provide the design guideline.

3.1 Architectural Exploration

This section focuses on figuring out which architectural configurations are best suited to optimally utilize the available power and energy by maximizing the processor performance under different energy constraints. Hence, depending on the energy that is harvested, I analyze various parameters such as the number of pipeline stages, the data to be backed up and the frequency of backups.

Some common configuration assumptions for these structures include:

1. ISA MIPS
2. Clock frequency is 8kHz for all configurations. The selection of clock frequency is driven by the limited strength of the WiFi signal used, rather than the limits of the microarchitectures.

3. Instruction Memory and ICache: The instruction memory is assumed to be ROM. The ICache can be SRAM, hybrid [51], or NVM [51, 52]. In this paper, ICache is designed using NVMs.

4. Data memory and DCache: The Data memory is assumed to be non-volatile. An SRAM-based DCache employing a write-through strategy does not require any backing up policy. On the other hand, a write-back strategy necessitates writing the dirty data back to memory. Our system assumes a non-volatile write-back DCache which preserves the dirty data even during periods of power down.

3.1.1 Non-Pipelined Configuration (NP)

In the absence of any pipeline stages, the entire state of the processor can be characterized by a single instruction state. Hence it is sufficient to focus on the following structures for retrieveing architectural state.

1. Program counter (PC): The PC address relates to the instruction being executed and surely needs to be stored.

2. Register file (RegFile): Due to its frequent usage, the RegFile undergoes a large number of writes and hence volatile RegFile is more energy efficient than an NVM based RegFile.

   However, all the volatile RegFiles need to be moved to a non-volatile memory on power failures to save state.

In addition to the architecture, there are also tradeoffs between the energy consumed in backing up and recovering the data and the overall performance. These tradeoffs are explored, by choosing which data to save, and when to save it, as demonstrated by the following policies.

Backup every cycle (BEC):

In spite of the significant energy penalty, this solution employs an NVM register file, or else both the contents of a volatile Regfile and its counterpart non-volatile location need to updated every cycle. As shown in Figure 3.5, only the PC and a few registers are written into the Regfile every cycle. Some instructions such as StoreWord and Jump do

not require any further Regfile write. Consequently, the power increase due to the use of a power hungry non-volatile memory is moderate.

On Demand All backup (ODAB):

This differs from the previous solution in that all RegFile entries need to backed up only in the event of a reduced power state. We develop a control structure shown in Figure 3.1, in which there is an individual NVM backup block to back up the PC and RegFiles. If the input power drops below a preset threshold, a power warning signal is activated. At that instant, the control unit starts to back up the PC and resets the atomic flag for PC to indicate that the PC has been successfully backed up. A similar procedure is carried out for the RegFile. When the power is available again, I first need to accumulate energy in the capacitor to ensure that there is enough energy for the next backup and recovery operation before continuing execution.

On Demand Selective backup (ODSB):

In order to reduce the backup time and energy penalty, I develop an *on demand selective backup* solution, as described below. A synchronous power warning signal is used, which may delay the power warning signal a little bit, but can guarantee that the current PC finishes executing and writing back (if necessary). To avoid re-executing the instruction corresponding to the current PC, $PC+4$ is stored except in case of jump or branch instructions. This solution can save one clock cycle. Since the frequency of this system is very low, even a single clock cycle may be very significant if power down happens frequently. In the volatile RegFile, I add a change flag to each register to identify if a register has been written into between two backup operations. If the register has not been changed during the interval, the control unit will know it from the change flag and would not need to generate addresses for the unchanged data, as shown in Figure 3.3.

Simulation results and comparison:

Figure 3.2 shows the area of each of the components for the schemes described above. It is observed that the total area is similar, since the NVM Cache and Backup Blocks are much larger than the logic components. The critical path delay shown in Figure 3.4 indicates that the BEC has lowest peak frequency due to the frequent backups. However, there are overheads in the other schemes which also prevent them from running at peak performance.

These overheads are illustrated in Figure 3.5, which shows the compute, backup, recovery and off times for each scheme described above. BEC distributes the backup

energy penalty to every cycle. Thus these penalties are the smallest for this case, as shown in Figure 3.6 and Figure 3.7. The recovery time is defined as the time from the activation of the *Energy OK signal* to the time all backup operations are completed. The recovery times in are similar across all schemes, but BEC does not need to accumulate energy for backup. Consequently, this scheme can restore the system the fastest. The ODAB scheme needs to back up the PC and the entire RegFile, thus the time and energy penalty is the largest.

ODSB reduces the number of RegFile entries to be copied, by detecting if the RegFile has changed during two backup intervals, thus requiring less backup time and energy than ODAB.

In order to determine the best NP scheme, optimizing power and energy is more important than timing, on account of the low clock frequency. In BEC, if the interval time (the time of power on during two power loss)is very short, the energy per instruction is low because at most only one RegFile entry is backed up, while ODAB needs to backup all RegFile entries. ODSB backs up only one entry at a time, but it is more complex in design. As the backup interval time is increased, ODAB and ODSB are more energy efficient, as observed in Figure 3.7, on account of backing up only in the event of a power warning.

In order to avoid a large peak power which can result in system instability, I choose to back up and recover data serially. Although a parallel approach can reduce the back up and recovery time, it increases the peak power requirement. From this point of view, the ODSB is better than ODAB.

- *ODSB is most energy efficient strategy when the source is relatively stable like solar energy. Compared to ODAB, ODSB can reduce the backup energy penalty by 69% with only 0.002% area overhead.*
- *While BEC is not the most energy efficient with very weak sources like WiFi, it does not require the time to accumulate energy in the capacitor to ensure sufficient backup energy is available, as shown in Figure 3.5. Hence it is viable when the power failures are extremely frequent (less than 1 in 10 cycles), which rarely happens even in WiFi sources.*

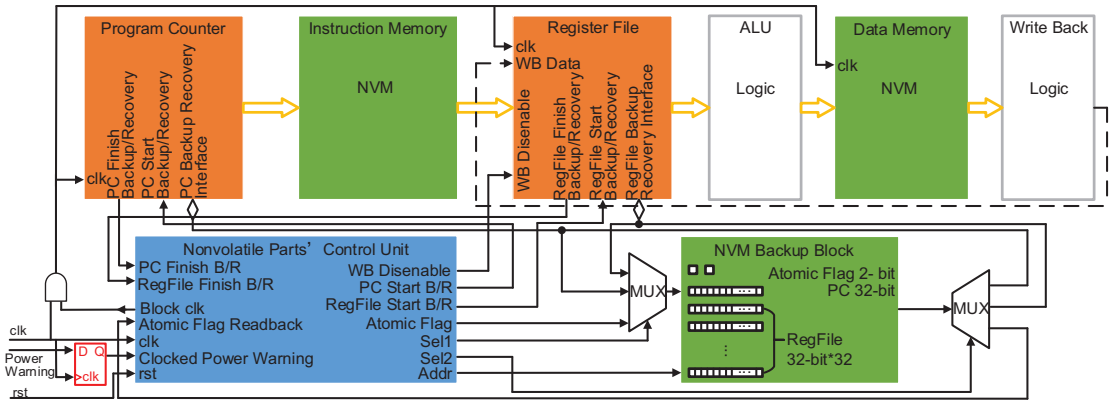


Figure 3.1. NP On Demand All Backup structure

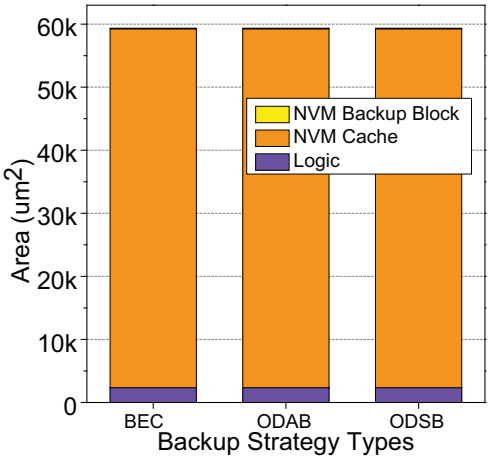


Figure 3.2. Non-Pipelined area

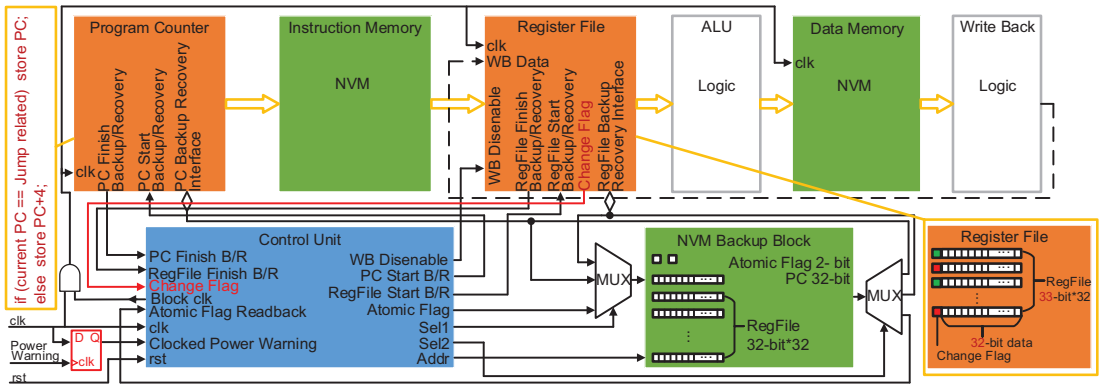


Figure 3.3. Non-Pipelined On Demand Selective Backup structure

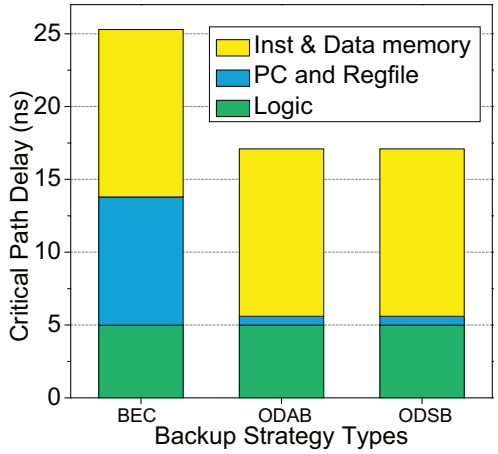


Figure 3.4. Non-Pipelined critical path delay (VDD=0.95V)

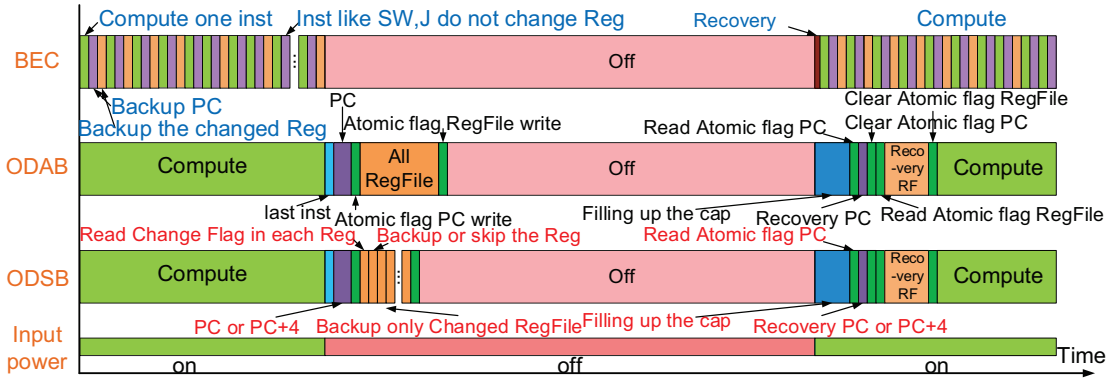


Figure 3.5. Individual runtime components for BEC, ODAB and ODSB schemes

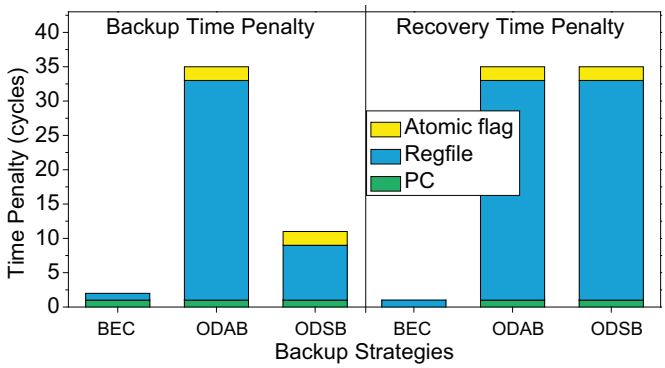


Figure 3.6. NP time penalty comparison

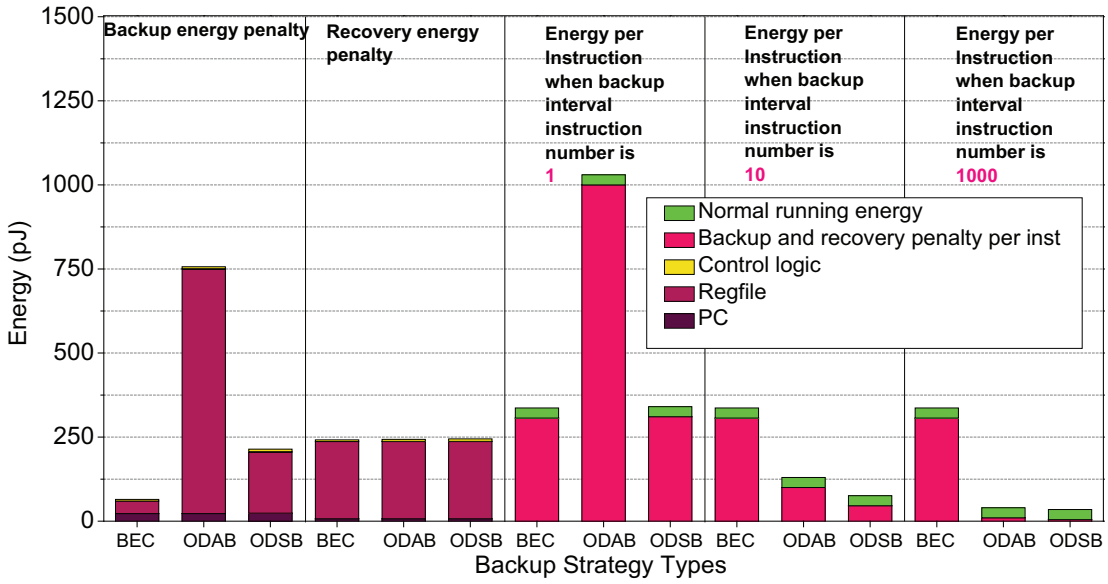


Figure 3.7. Energy overheads for each NP scheme. For high frequency of backups, ODAB has the highest overhead, while BEC consumes maximum energy when the backup interval exceeds 10

3.1.2 N-Stage-Pipeline

In contrast to the MIPS non-pipelined case, a MIPS *N-Stage Pipeline* is traditionally used to improve the clock frequency. Due to the increase in circuit complexity and the activity factor of the processor, the power threshold of this design in energy harvesting systems is higher than that of the Non-Pipelined case. In this subsection, we assume a Five-Stage-Pipeline structure (5SP) and propose two backup schemes.

Shifted PC & volatile flip-flops (SPC/VFF):

The main differences between NP and 5SP configurations are the pipelined data flow with bypass and forward and the complex control flow to handle hazards. In the SPC / VFF scheme, a shifter buffer is designed to remember the PC value in each pipeline stage, as shown in Figure 3.10. This means the PC no longer needs to pass through all pipeline stages to be stored. When the power is down, the clocked power warning signal can guarantee that the PC in the write back stage will be finished. The unfinished PC to be backed up would then be in the data memory stage. The reason that I use a shifter instead of simply rolling back the PC is that if some of the instructions in the pipeline are jump or branch instructions as shown in InstQue2 in Figure 3.10, a different PC would need to be backed up. If the instruction in MEM stage is SW as shown in InstQue1&2 in

Figure 3.10, this SW instruction will guaranteed finished by the clocked power warning signal. We can try to identify if the backed up instruction is SW, if yes, backup the PC in EX stage in the shifter instead of PC in MEM. Instead of dealing with the increased design complexity, we can just backup PC in MEM stage. Once the power is on again, the first instruction will be SW. In this case, we run SW actually twice: the first time is during the back up operation, another one is the first instruction after recovery in case the former one is not properly finished.

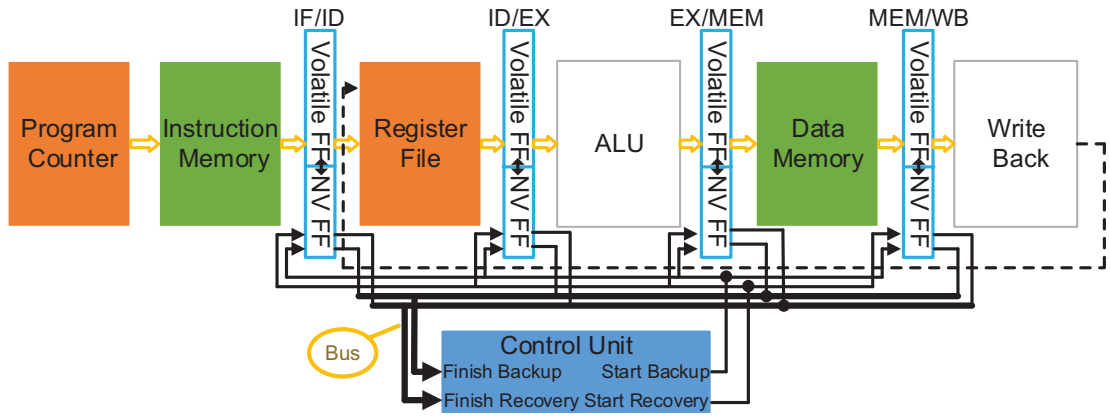


Figure 3.8. Five-Stage-Pipeline NVM Flip-flops backup

Nonvolatile flip-flops solution (NVFF):

This solution involves the use of NVM flip-flops (Figure 3.8). Here, the PC and the Reg-File are automatically backed up through NVM flip-flops in the IF/ID pipeline stages.

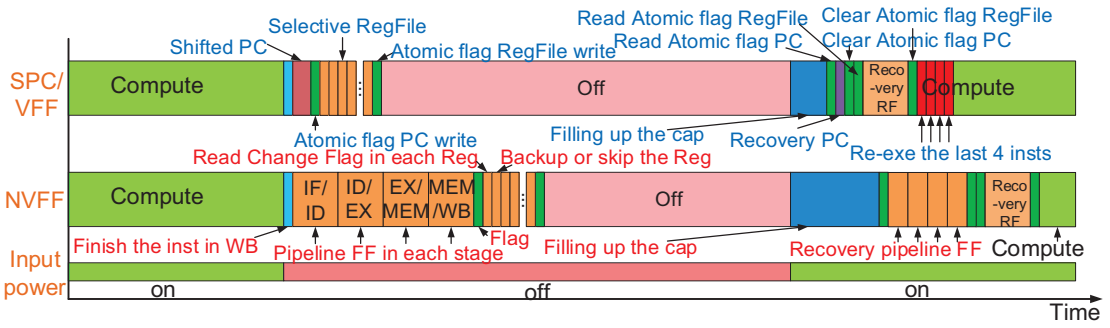


Figure 3.9. Comparison of individual runtime components for SPC/VFF and NVFF

Simulation results and comparison:

SPC/VFF requires 11% less time and 57% less energy than NVFF in Figure 3.11.

Pipeline	IF	RF	EX	MEM	WB
InstQue1	LW	ADD	SUB	SW	ADD
Shifter	PC	PC-4	PC-8	PC-12	
InstQue2	LW	J	SUB	SW	ADD
Shifter	PC2	PC1	PC1-4	PC1-8	

Figure 3.10. Illustration of Shifted PC

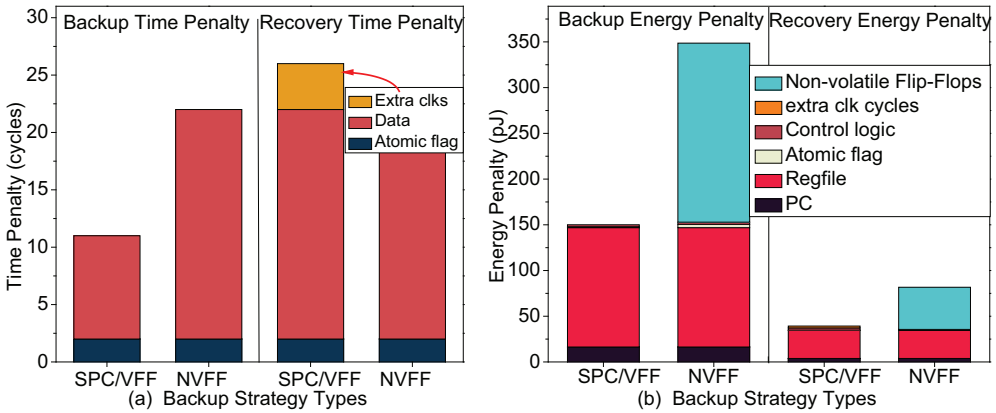


Figure 3.11. Comparison of 5SP a) time & b) energy overheads

However, an extra 4 clock cycles are needed to re-execute the last 4 instructions that are lost from the latter pipeline stages after recovery.

which we regard this as part of the recovery time penalty.

Counter to intuition, I show that SPC/VFF is more energy efficient than NVFF. Instead of backing up all the data in the pipeline latches, SPC/VFF only backs up one PC with a small shifter. Hence, a smaller backup capacitor with lower leakage is sufficient for SPC/VFF, which will in turn affect the power threshold. In this case, SPC/VFF will also be able to outperform NVFF after several repeated instructions.

3.1.3 Out-of-Order Processor (OoO)

Compared to the MIPS 5SP configuration, our MIPS out-of-order (OoO) processor configuration, described in Table 3.1, is much more complex. Figure 3.12 indicates the key blocks we consider in our OoO processor model derived from FabScalar [53]. Conceptually, system state, unlike in the previous two examples, is broadly distributed across several structures such as the PC, ROB, RegFile, Map Table, Issue Queue, Load Store

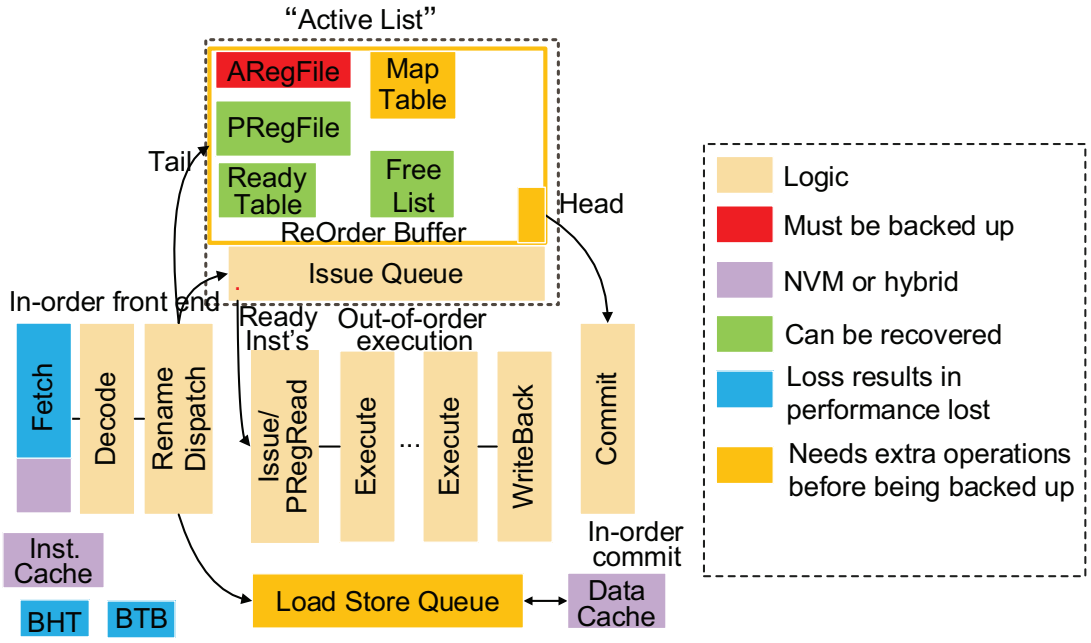


Figure 3.12. OoO volatile/non-volatile structure

Queue as well as the Branch History Table and Branch Target Buffer. Some of these structures are essential to maintain the integrity of the state of the system, while others contribute toward optimizing the performance and/or energy of execution in the presence of frequent backups and recoveries.

Due to the relatively larger power requirements of an OoO processor, there are both fewer periods where the input power exceeds the minimum threshold, as compared to the previous cases, and more state to consider saving during power emergencies. Hence it is imperative to judiciously select the structures to be backed up, in order to ensure a comparable performance to the no-pipeline and n-stage pipeline designs.

Parameter	**OoO**	**Parameter**	**OoO**
Fetch width	4	Map Table	32
Issue width	4	PRegFile	128
ROB size	32	Ready Table	128
IQ size	32	BHT/BTB	128
LSQ size	32/32	ARegFile	32
ICache/DCache	32kB/32kB	Free List	128

Table 3.1. Parameters for OoO processor

We propose several resource selection strategies for this purpose, as illustrated in Figure 3.13.

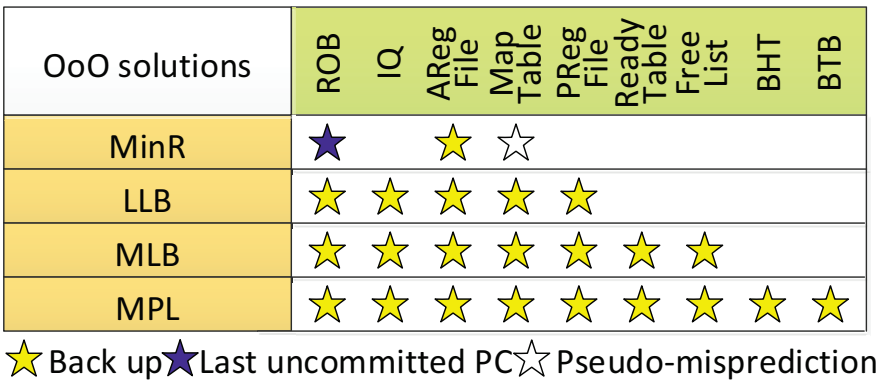


Figure 3.13. Backup schemes for OoO configuration

Minimum State Resource backup solution (MinR):
MinR backs up the minimal number of bits required to preserve functionality across power interruptions, as shown in Figure 3.13 and Figure 3.15. Fundamentally, this approach piggybacks on the branch misprediction mechanism to minimize the number of valid/relevant state bits prior to initiating backup, at the cost of some time and effort being required to enact the misprediction logic prior to checkpointing.

1. ROB and PC: to minimize state storage, I only back up the first uncommitted PC at the head of ROB. This means all the other instructions in the ROB will be abandoned regardless of status.

2. IQ: IQ does not need to be backed up because all the instructions in IQ are uncommitted.

3. ARegFile: We can either choose to backup ARegFile or PRegFile. The ARegFile is preferred since it is designed to be smaller.

4. Map Table:

   It is possible that uncommitted instructions that follow the head of the ROB could have modified the Map Table. However, since we need to restore the state to the instruction at the head of the ROB, the Map Table should also be correspondingly restored. In order to achieve this, we trigger an instruction flush identical to that following a branch misprediction on the ROB head. Since no actual branch prediction occurs, I term this operation *Pseudo-Misprediction*.

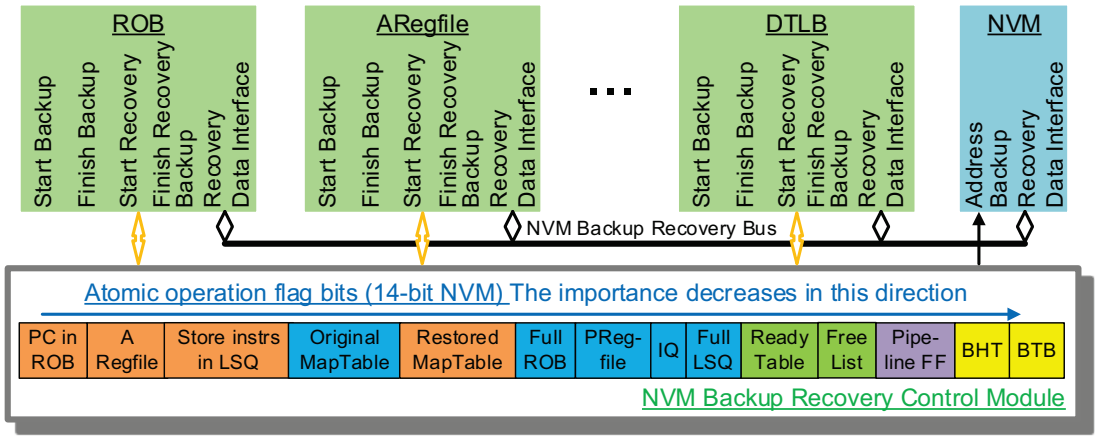


Figure 3.14. OoO integrated flexible atomic backup solution

5. PRegFile, Ready Table, Free List, BHT, and BTB can be recovered.

Low-latency backup solution (LLB):

While the MinR policy minimizes bits pushed to nonvolatile storage, it does so at the expense of requiring additional work before backup can begin. We next consider a backup solution that aims to minimize the number of bits to store if backup begins immediately. Rather than back up only the first uncommitted PC, this solution backs up the entire ROB, IQ, ARegFile, Map Table, and PRegFile. Compared to the MinR policy, structures such as the Ready Table and Free List (Figure 3.17 and Figure 3.18) can be more easily reconstructed, resulting in a penalty of only a few recovery cycles. While LLB stores more state than MinR, it can sometimes nonetheless be more energy-efficient, due to the extra work required of MinR on both backup and recovery.

Middle-level backup solution (MLB):

Instead of using extra recovery time and energy to restore the Ready Table and Free List in the low-level backup solution, MLB backs up Ready Table and Free List as well (Figure 3.13).

Min-state-lost backup solution (MPL):

In this solution, all the structures are backed up including the BHT and BTB as shown in Figure 3.13.

Integrated Flexible Atomic Backup Solution (IFA):

All of the previous solutions save and restore a fixed amount of state determined by the structures in question. However, one key feature of the backup process is that it must necessarily be triggered conservatively: The backup signal must be issued at a

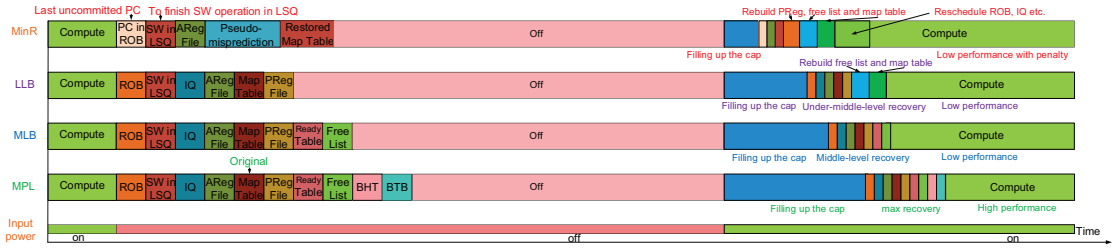


Figure 3.15. Out of order structure backup design trade-off

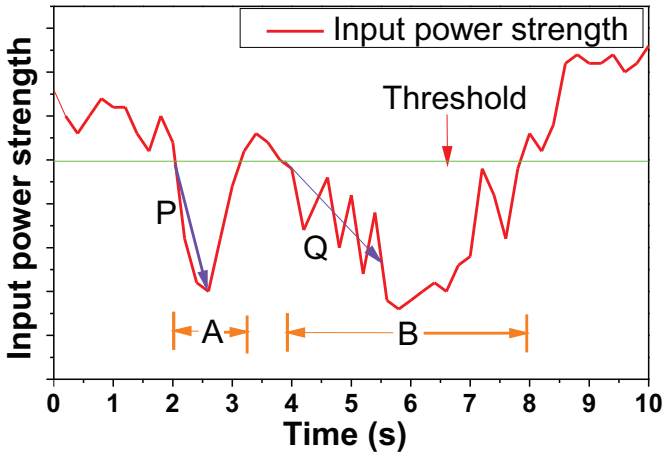


Figure 3.16. Scenarios in which IFA can be applied

point where the processor can guarantee sufficient energy to complete the backup **even assuming zero additional input power during backup**. However, in practice, when a power emergency occurs in an energy-harvesting system, it is not usually because input power has dropped to zero, but becaue it has fallen below some threshold for some period of time. Thus, there may frequently be additional energy available during the backup period that, while insufficient to continue operation, would allow for optional state, such as the BHT, to be subject to optimistic attempts at backup.

We propose a flexible backup mechanism that integrates aspects of the 4 previous solutions together to exploit the conservative nature of the backup trigger. The key idea of the solution is to regard each backup operation as an atomic operation. A backup operation has only two states: success or failure. Figure 3.14 shows the systematic structure of this solution. Figure 3.16 shows how the power may be dropping at different pace to zero and can execute more or less backup.

Simulation results and comparison: For MinR, the pseudo-misprediction operation

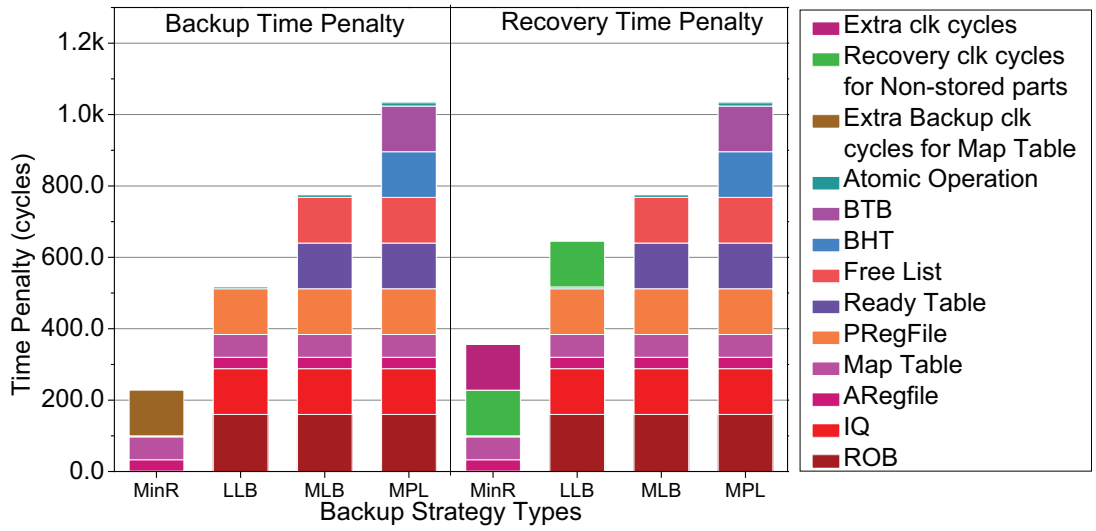


Figure 3.17. OoO time penalty

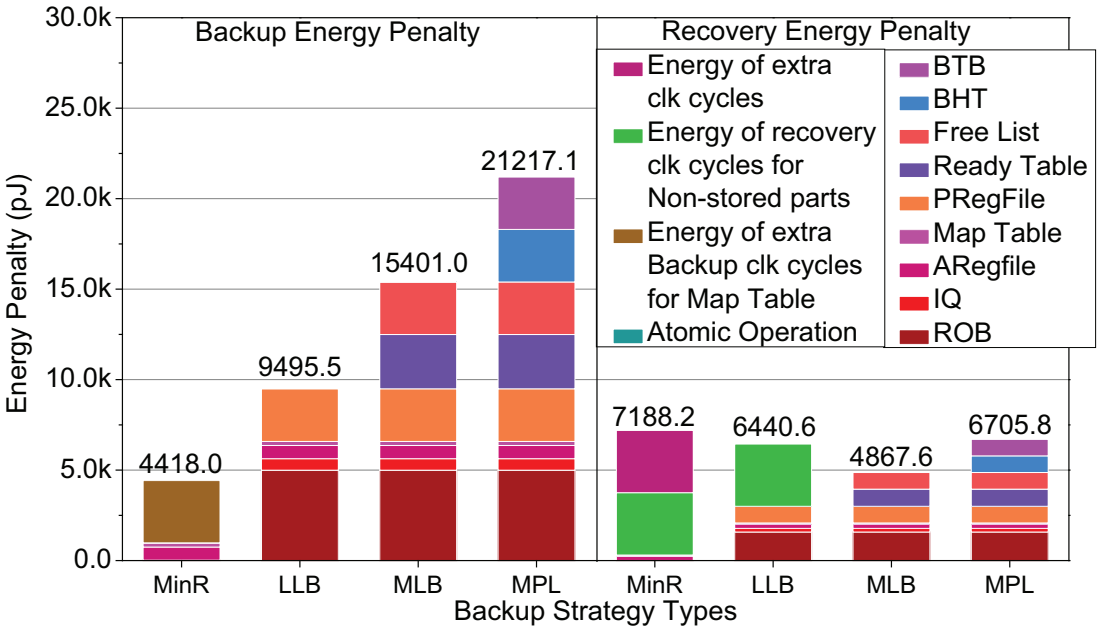


Figure 3.18. OoO energy penalty

for the Map Table requires extra backup clock cycles as shown in Figure 3.15. When recovering, we also need to pay extra clock cycles to restore the PRegFile, Ready Table, and Free List. Further, since we discard all instructions in the ROB following the head, we need to re-execute these instruction, resulting in the timing penalty in Figure 3.17 as well as energy overheads, shown in Figure 3.18.

In the case of LLB, the ROB and PRegFile are relatively large and significantly

increase the backup time and energy in Figure 3.17 and Figure 3.18. On the other hand, the recovery energy penalty is smaller than MinR, because all the instructions and their information in the ROB are backed up, eliminating the need to re-execute these instructions.

The backup time and energy penalty of MLB are are larger than those of LLB, as shown in Figure 3.17 and Figure 3.18, but the recovery energy is lower.

The designer should use this MLB strategy when the system is optimizing the time to resume execution after a power failure.

In MPL, the backup and recovery time penalty and energy penalty are the largest in among all the solutions, but backing up all the additional structures incurs the minimum latency to return to peak performance after a power failure.

Results show a 29 cycle gain for MinR, but not backing up the BHT and BTB negatively affects IPC.

On account of OoO being thought to be too complex for energy harvesting systems, prior work has not considered OoO platforms. Since OoO needs a much higher threshold than NP and NSP, the percentage of time OoO can run is much smaller than NP and NSP. However, OoO remains a favored option in several of the test senarios in subsequent sections because the periods of sufficient power are common enough to sometimes allow superior performance to pay for lost cycles. In summary, storing the minimum number of bits, MinR, does not always provide the least energy backup solution, and MLB has the shortest time to execution after power failure. We also demonstrate that, due to the conservative nature of backup initiation, there is a sizeable potential for opportunistic backup of optional, performance enhancing bits with a flexible backup policy.

3.2 Validation

While the primary focus of this paper has been on an simulation-based exploration, we have explored the non-pipelined on-demand-back up strategy using an actual fabricated processor. In addition to demonstrating the execution of real workloads on the processor, this effort enabled us to gain insights to approximations in initial simulation models and helped refine the simulation model used in this work.

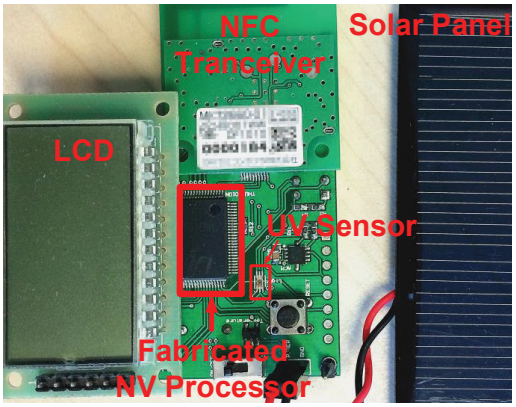


Figure 3.19. System prototype

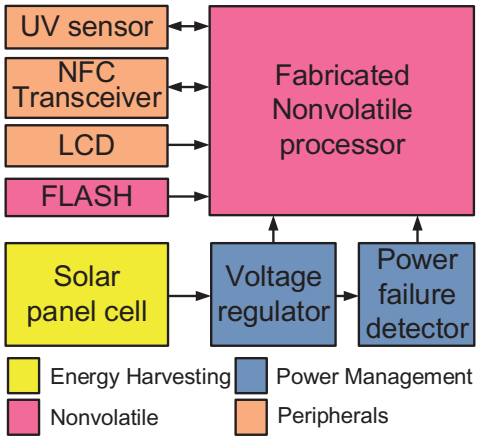


Figure 3.20. Block diagram

3.2.1 System Overview

The nonvolatile processor is based on a Intel 8051 processor and the ISA choice for fabrication was driven by availability of a pre-existing design. This processor supports multi-cycle instructions as compared to the MIPS ISA used in the rest of the paper. Consequently, in this implementation, the saved state includes the state machine that captures the exact cycle in which the instruction was carried out currently. This is the most relevant difference from the perspective of the non-volatile design space explored as compared to the MIPS pipeline in our simulation studies. However, our simulation effort encompassed the 8051 processor as well to gain insight to potential sources of differences between the simulation and real system.

The non-volatile processor based system is interfaced to a solar power panel and a UV sensor as shown in the Figure 3.19. The processor is based on a 0.13 μm ROHM CMOS-ferroelectric hybrid process. The PC and all RegFiles are FeRAM-based Flip-Flops. The Flip-Flops are realized using an additional backup ferroelectric capacitor (FeCap) for each D flip-flop (DFF) used in the design. When a power failure is detected, the NV control logic backs up the DFFs to the FeCaps. When power is resumed, data is restored from FeCaps to DFFs. All FeCaps are distributed and connected close to their own DFFs, thus the data backup and recovery can proceed in parallel to reduce the operation time. Table 3.2 shows the chip specifications. The total power decides the power threshold, the backup energy decides the energy storage capacitor volume. The capacitor used in the system is 470nF.

The design process revealed insight to modeling to key aspects in the simulation

environment. The clocking network is switched to a lower frequency to transition clock generation from an external oscillator to an internal RC circuit. The external oscillator could become unstable or may not have sufficient power to operate. Further, a lower clock frequency increases the reliability of the FeRAM writes and also reduces peak power consumption. The slower clock impacts the overall back-up time as compared to using estimates based on faster operational clock. Similarly, the recovery time should include not only the time required to restore architectural state but the time for the clock generators and power supply grid to be stable.

3.2.2 Simulator Calibration

Once these insights are incorporated in the simulator, a few kernels were executed on both the platform and the simulator (See Table 3.3). To model an intermittent power supply, a 1KHz square waveform power input was fed to the processor and the processor frequency was limited to 3MHz (the maximum frequency at which it could operate based on power supply when connected to the solar panel). Each kernel was executed 1000

Parameter	**Result**	**Parameter**	**Result**
Max. clock	25MHz	Total power	160μW@1MHz
Process Technology	0.13μm	Backup energy	23.1 nJ
V_{DD} for core	0.9V-1.5V	Recovery energy	8.1 nJ
Total area	1.015 mm^2	Backup time	7μs
Energy/Inst	347pJ	Recovery time	3μs

Table 3.2. Measured Parameters

Testbench	**Stable/ms**	**Interrupted/ms**		**error**
	Measured	**Measured**	**Model**	
FIR-11	0.626	1.260	1.209	-1.59%
Sqrt	2.620	5.280	5.190	0.81%
KMP	3.573	7.184	7.059	0.77%
FFT-8	4.207	8.460	8.238	-0.13%
Matrix	5.826	11.740	12.021	2.39%
Bubble sort	27.23	54.705	57.236	4.63%

Table 3.3. Execution Time on simulator and actual platform when using an interrupted power supply generated as a square waveform.

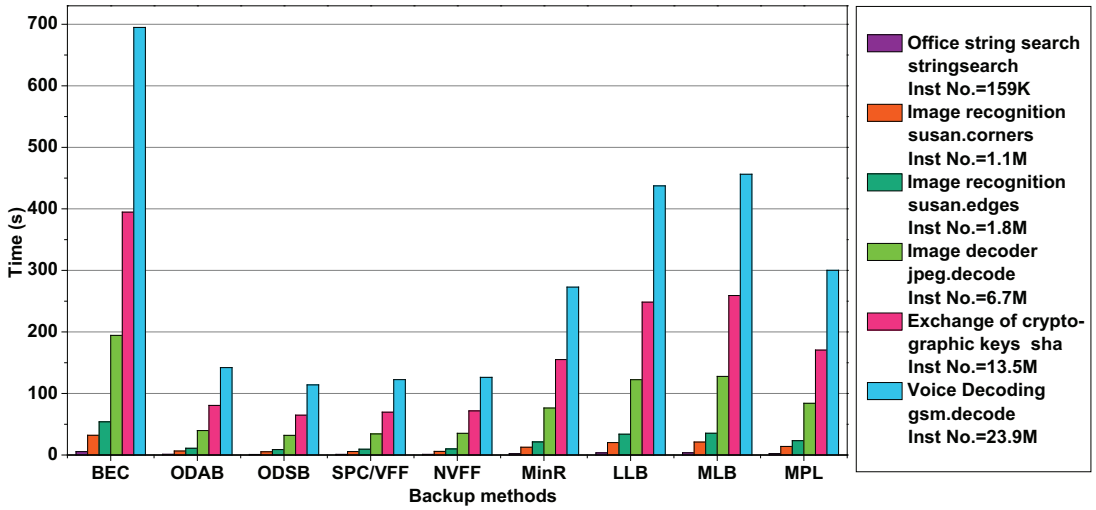


Figure 3.21. Execution time with energy scavenged from WiFi home environment

times to obtain overall completion time shown in Table 3.3. For the stable power case, the simulator and platform mismatch is negligible. For unstable power, the simulator and the platform measurements differ less than 5%. The differences accrue since the simulator averages energy consumed by an instruction to estimate remaining energy for triggers. However, the actual instruction execution exhibit non-uniform activity. Further, the energy storage capacitance models used in the simulation add and decrease in discrete portions unlike the actual design. This validation process for the simulator based on a real design indicates that simulation-based models are fair representation of actual systems.

3.3 Design Guidelines

The complexity of the non-volatile architecture chosen for a particular application scenario depends on a variety of factors. The input power and the stability of the power supply are two key factors that impact the choice. In addition,the computational complexity of the application and its performance requirements is also important.

3.3.1 Dependence on Input Power Characteristics

The input signal characteristics play a major role in determining the optimal design, as is evident from our experiments with Wi-Fi power trails under different environment

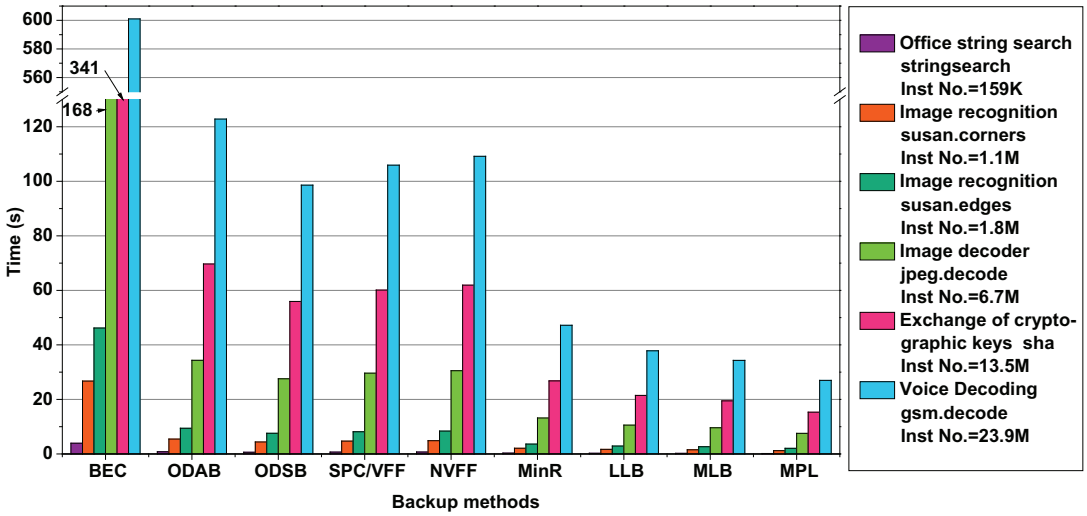


Figure 3.22. Executing time with energy scavenged from WiFi office environment

conditions. Figures 3.21 and 3.22 demonstrate the performance of the various backup schemes when home and office Wifi sources are used for harvesting energy. For the home environment, a non-pipelined ODSB architecture is the best performing. On the other hand, in the office environment, the more complex OoO processor with the minimum performance loss scheme is desirable. The reason for this behavior is that, the home WiFi signal comprises of a single router, while the office environment consists of several routers of similar signal strengths. A disturbance in the signal would result in input power going to almost zero in the home environment, hence the simplest design with the lowest power threshold is preferred. In contrast, in the office environment, the additional routers continue to supply input power at a relatively similar strength in an uniterrupted fashion, thus allowing for more complex architectures.

3.3.2 Dependence on Nature of Input Source

Input energy sources differ both in the magnitude of the input power as well as its variation. Figure 3.23 demonstrates the behavior of the different architectures under these conditions, by testing multiple power traces for each configuration. In each case, the best performing backup policy is adopted. Since the power traces have different ratios between the power on and power off states, the backup/recovery penalties are also different. Consequently, they have different running times. We observe that, for the same input power source, the actual execution time of NP and 5SP are roughly the same.

However, the higher power threshold in the 5SP configuration results in the below-threshold or off-time being much higher. The OoO configuration is nearly 3× faster than NP and 5SP when it executes and hence the overall running time is proportionately smaller. This behavior is consistent across all input sources with the actual execution time determined by the magnitude of the power source.

3.3.3 Quality of Service

Most of the applications that are expected to run on an energy harvesting platform require an output to within a fixed time period. Quality-of-Service (QoS) can be standard to qualify the possiblity to achive that goal. When these systems run on harvesting ambient energy, the unreliable nature of the input source may prevent the QoS demands in some instnaces.

Figure 3.24 shows the percentage of instances that meet the QoS demands specified, for two different applications, measurement of ECG and an edge detection algorithm used in vision sensors.

Figure 3.24 provides an indication of meeting the QoS. For example in Figure 3.24a), the possibility to achieve real-time ECG processing with NP and RF power is 0.92%. Consequently for RF and Thermal sources, real-time processing is not possible. For ECG, most of the solar and Piezo sources can support 100% QoS.

The baseline configrations for Figure 3.23 and Figure 3.24 are listed in Table 3.4. Table 3.4 also indicates several methods to optimize the QoS with efficiency:

- From power input view, there are lots of features that we can improve the input power. The baseline in the QoS simulations for RF is 10km from TV stations, but in some places like New York, the average TV station distance is around 3km [54]. And the RF power strength is in negative square realtionship with distance, so we can 11.1 times larger average power, thus improve the QoS to 100% .

- We can try to improve the efficiency of AC-DC, DC-DC, LDO, reduce the capacitor leakage, etc. to improve QoS.

- From output view, try to shink the techonolgy [55] (baseline is 130nm CMOS) to get lower power consumption, for example, applying 22nm FinFET [56] will achieve 100% QoS for real-time ECG in Manhattan. In addition, various methods

could be applied to reduce the power: new devices like Tunnel-FET [38], low power circuits like sub-threshold circuits, dark silicon [55], gated clock, dynamic-voltage-frequency-scale (DVFS), Dynamic-Adjusting Threshold-Voltage Scheme (DATS) [57] etc. are some examples.

Aspect	Solution	QoS Baseline	Efficiency
RF	Antenna gain	6dBi	α
	Bandwith	539M	α
	Distance	10km	$1/\alpha^2$
Therm	Area	$1cm^2$	α
	ΔT	$20^\circ C$	α^2
Piezo	Volume	$1cm^3$	α
Solar	Area	$4cm^2$	α
	Efficiency	28%	α
Circuit	IP matching, AC-DC, DC-DC, LDO, Cap		
Tech.	Shink Tech.	130nm	α^2
	FinFET, IG-FinFET, TFET, NC-FET	CMOS	
	DVFS, DATS,	Fixed frequency	
	Voltage	0.95V	$1/\alpha^2$

Table 3.4. Baseline and relationship with QoS improvement

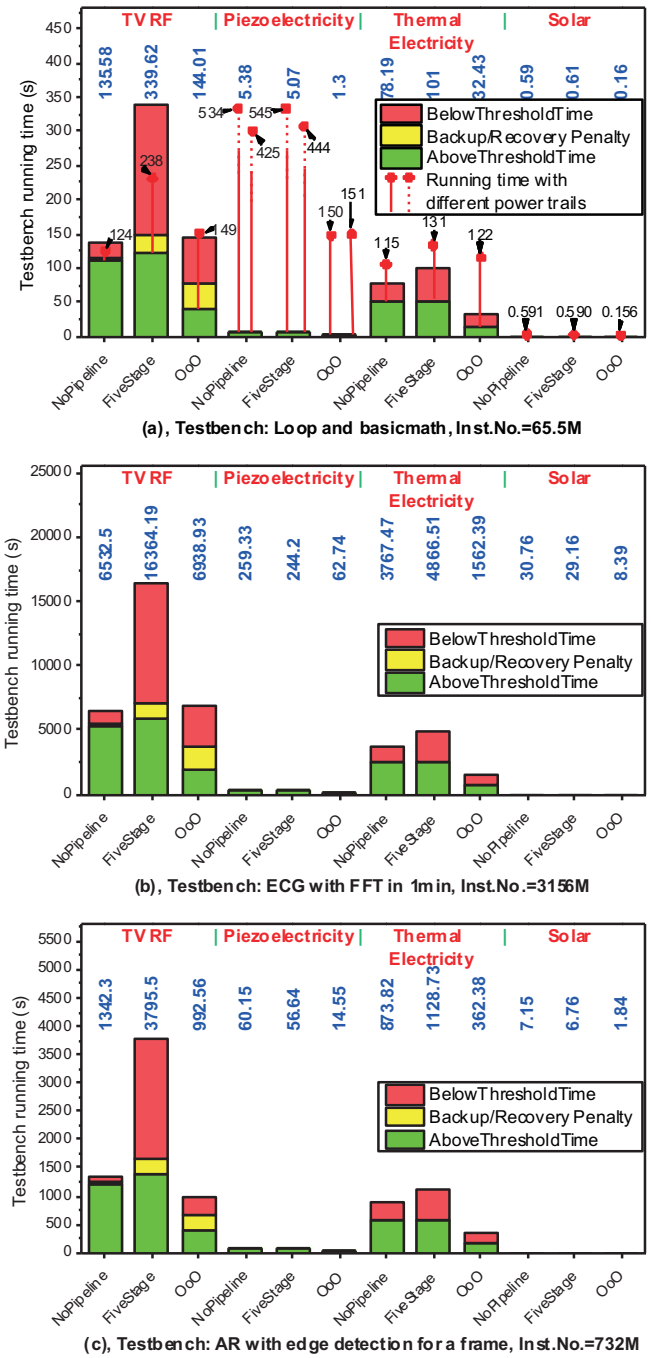
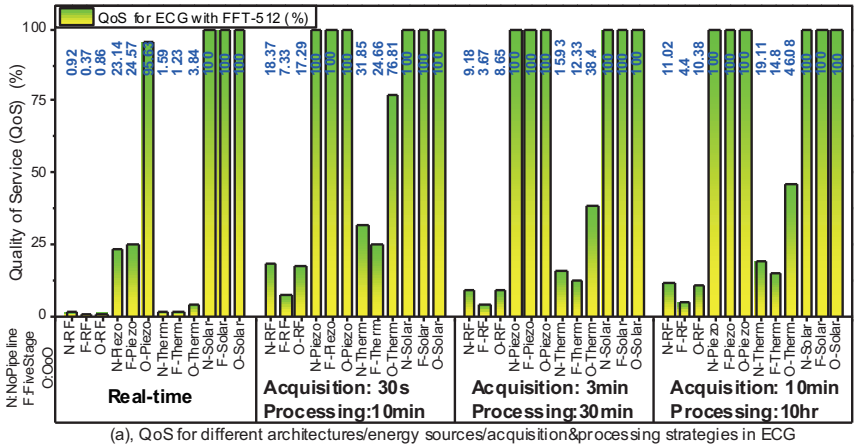
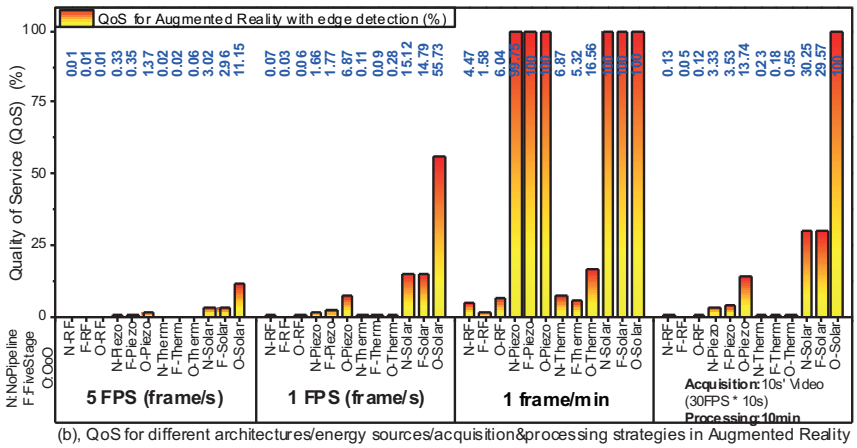


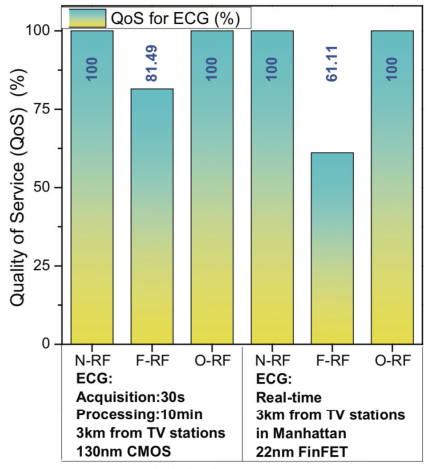
Figure 3.23. Execution time for different testbenches under different power sources and trails



(a), QoS for different architectures/energy sources/acquisition&processing strategies in ECG



(b), QoS for different architectures/energy sources/acquisition&processing strategies in Augmented Reality



(c), QoS improvement

Figure 3.24. QoS for ECG/AR, and QoS optimization

Chapter 4

Dynamic optimization of NVP

With the development of nonvolatile processors (NVPs), energy harvesting is emerging as an increasingly attractive means for powering the internet of things (IoT) [58, 59] NVPs can handle unstable input power by backing up the computation state in distributed nonvolatile flip-flops or integrated memories at very short timescales, allowing systems using these processors to operate without large energy storage devices. In energy-harvesting systems, the local variance in input power is large: The peak available power can be many times larger than average power, and there can be sustained periods where only a minimally active processor can operate at all. Incorporating flexibility into a processor to adapt to changing conditions is a long-studied area. Techniques such as Turbo Boost [60,61] and other dynamic voltage and frequency scaling [62–65] as well as microarchitectural resource adaptation techniques [66, 67] have been proposed by prior work in the context of energy-efficient computing. Prior work on energy harvesting NVPs [58, 59, 68–71] has also indicated that no single microarchitecture best translates input energy into forward progress across varying input power traces. Conceptually, the ideal NVP design is the one that can operate in input power valleys for more on-duty time, and also convert input energy plateaus into more progress rather than let them leak away or overflow.

Both frequency-scaling and microarchitectural adaptation are promising approaches for consuming energy that cannot be otherwise stored in a batteryless system. However, which approach is preferable and how the two approaches can synergize have not been explored in the context of NVPs for the IoT space. In particular, the policy space can be seen as a combination of predicting a) energy income in the next epoch and b) among

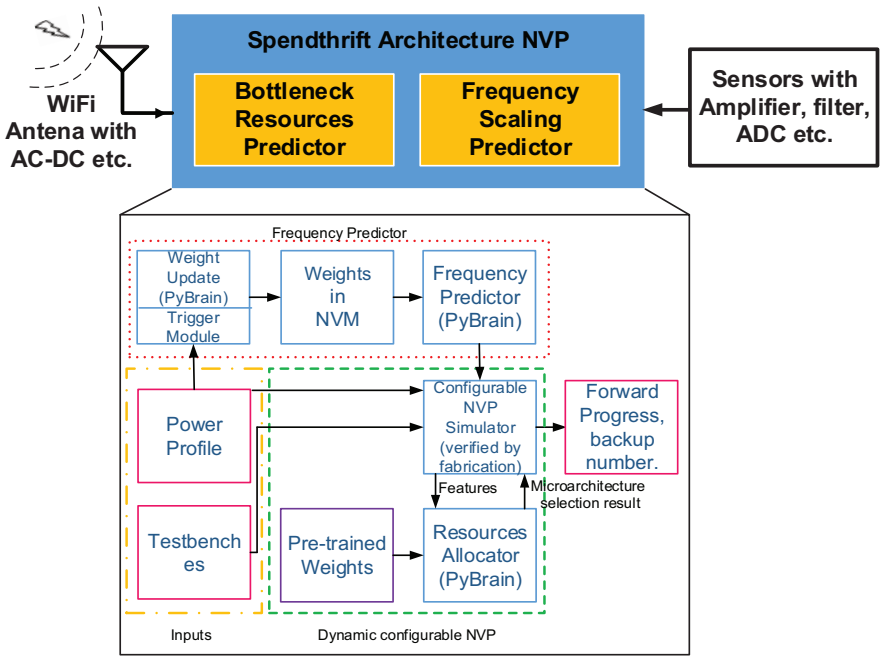


Figure 4.1. Spendthrift architecture and simulation structure.

designs capable of consuming as much of the energy income as possible over the coming epoch, which will offer the best forward progress per unit energy. The aims of this paper are to explore the effectiveness of both frequency and resource scaling techniques in the context of NVPs, and to develop an effective dynamic prediction mechanism to set both frequency and resource parameters efficiently as shown in Figure 4.1. The work makes the following contributions:

- Targeting lower energy per instruction (EPI) for NVP, we propose using an infrequently executed (5Hz) neural network-based predictor to manage bottleneck resources in a reconfigurable out-of-order processor.
- Targeting aggressive leveraging of harvested energy for forward progress, we design a machine learning based dynamic frequency scaling (DFS) module for nonvolatile processors.

4.1 Spendthrift: Bottleneck Resources Prediction

In this section, I propose the hybrid processor microarchitecture - Spendthrift, which incorporates a single-issue In-Order microarchitecture and a multi-issue high performance OoO microarchitecture. The In-Order microarchitecture can operate with the minimum resources powered on for the minimum start-up threshold; The OoO microarchitecture operates with more aggressive power consumption but can achieve the highest throughput. The mechanism of how to find the best configuration for the maximum forward progress is also described in this section.

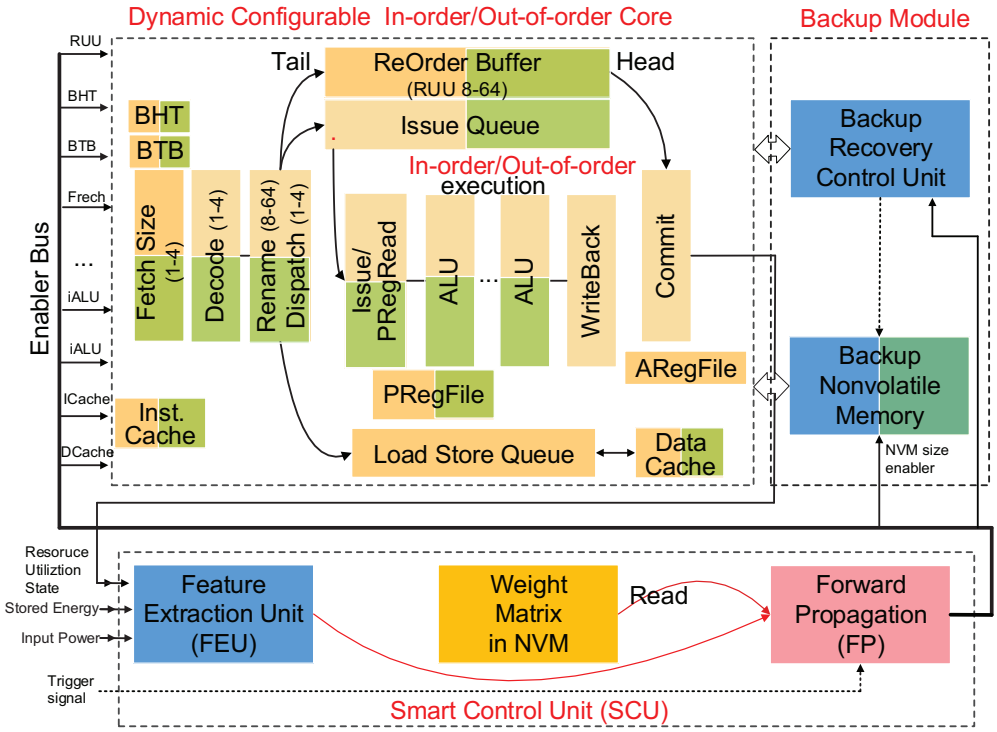


Figure 4.2. The system diagram of configurable resource allocation architecture. Configurations: IO/OO: InOrder/Out-of-Order, low for In-Order, high for Out-of-Order; FT: Fetch Width, low for 1, high for 4; DC/IS: decoder and issue width, low for 1, high for 4; RUU: low for 8, high for 128; ALU: low for 1, high for 4; MP: memory port, low for 2, high for 8; CL1: Instruction and Data Cache: low for -cache:il1 il1:256:32:1:l, high for -cache:il1 il1:256:32:4:l; ICL2: low for -cache:dl1 dl1:256:32:1:l, high for -cache:dl1 dl1:256:32:4:l; DCL2: low for -cache:dl2 ul2:64:64:4:l, high for -cache:dl2 ul2:256:64:4:l; PRE: low for -bpred:bimod 128, high for -bpred:bimod 1024.

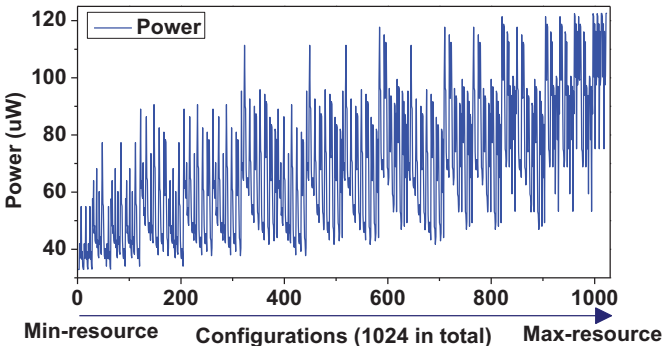


Figure 4.3. Power consumption under different resources, testbench ”susan_corners”.

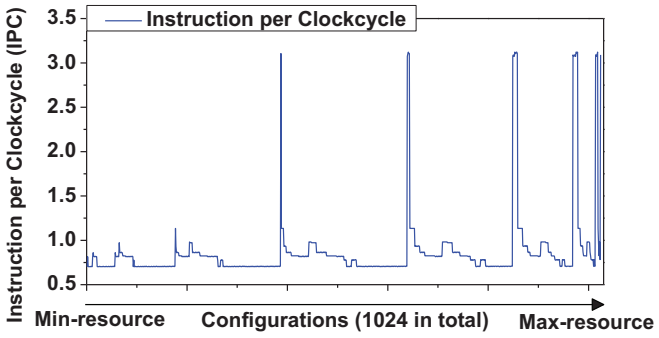


Figure 4.4. IPC V.S. different resource configurations, testbench ”susan_corners”.

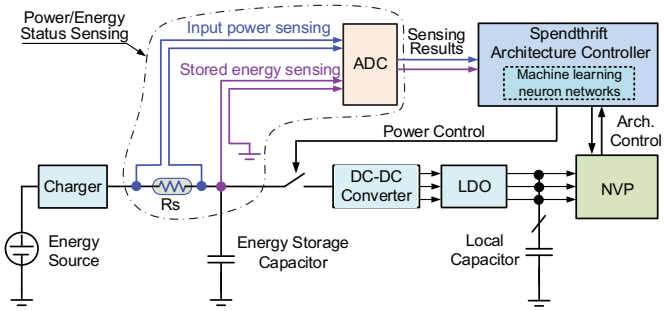


Figure 4.5. System diagram with feature extraction circuits.

4.1.1 Resource Allocation System Structure

Figure 4.2 shows the system diagram. 10 adjustable potential bottleneck resources are selected to balance EPI and performance. The total number of all possible configuration entries is 1024, and each entry uses 10 configuration bits. With limited power income in energy harvesting systems, only bottleneck resources are powered on so as to boost performance with the minimum power penalty. With more resources powered on, the

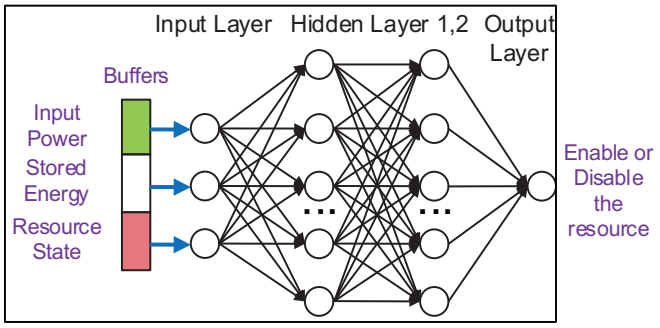


Figure 4.6. Neural network for one resource prediction.

instruction per clock-cycle (IPC) may increase a lot, but the power consumption does not increase as much, thus EPI reduces, as shown in Figure 4.3 and Figure 4.4. It is also noted that turning on and off resources results in switching delay and energy for power-gating control [72]. In addition, in the proposed solution, the need of freeing the resources before turning them off is also considered.

4.1.2 Feature Extraction and Neural Networks

Rather than building a single, large neural network that predicts 10 resources at a time, spendthrift uses 10 small neural networks, one network for one resource prediction. The reason is that multiple small neural networks can significantly reduce the computation amount. Each neural network has four layers as shown in Figure 4.6: one input layer, two hidden layers, and one output layer. These are optimized results following a similar approach considering number of layers and numbers of neurons in each layer, as discussed in [68]. In the input layer, there are three inputs. One input is for the current resource usage conditions: ”0” indicates a not fully utilized resource, and ”1” indicates a fully utilized resource. A condition of ”1” indicates one possible bottleneck resource as it may need extra resources for further performance improvement. The other two inputs are the input power and the stored energy. Both are captured through the front-end circuits shown in Figure 4.5 every 0.2 second by an A/D converter (ADC). A small resistor Rs is used to sense the power delivered through it with negligible voltage drop. Considering 32 required sensed levels, a 5-bit resolution is sufficient while consuming only 1nW power. As for the stored energy sensing, it is equivalent to a measurement and calculation of the voltage across the energy storage capacitor. The two hidden layers

each consists of 10 hidden neurons. The output has 1 node for resource selection result. When the output is higher than 0.5, that resource is treated as a bottleneck resource and will be enabled. The neural networks are triggered every 0.2s. Offline training is used with 10k training set, achieving an accuracy above 90% on Mibench "small inputs" [73] and these initial trained weights are stored in the NVM.

4.2 Dynamic Frequency Scaling

In this section, NVP frequency scaling policies as another effective approach is investigated. For simplicity without losing generality, I use a fixed number of resources, and three power profiles, a typical example of which is seen in Figure 13.

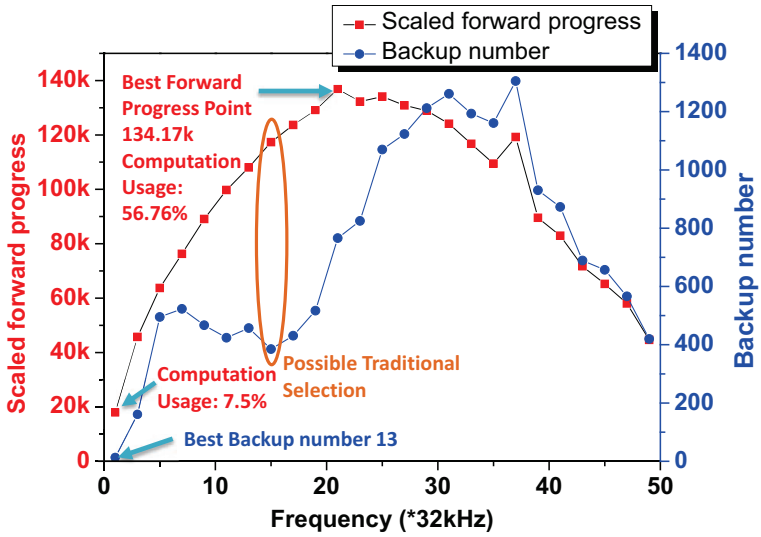


Figure 4.7. NVP performance vs. frequency with minimum resources.

4.2.1 Performance vs. Static Frequency

I scan across frequencies from 32kHz to 1.5MHz with a step of 32kHz. In the simulations, a 672kHz static frequency results in the best forward progress, as shown in Figure 4.7. Compared to the forward progress with minimized 32kHz frequency, the forward progress is accompanied by the penalty of 29X more backup operations. With too low a frequency, a large portion of harvested energy leaks away or cannot be stored in the capacitor because of power consumption lower than input power. With too high

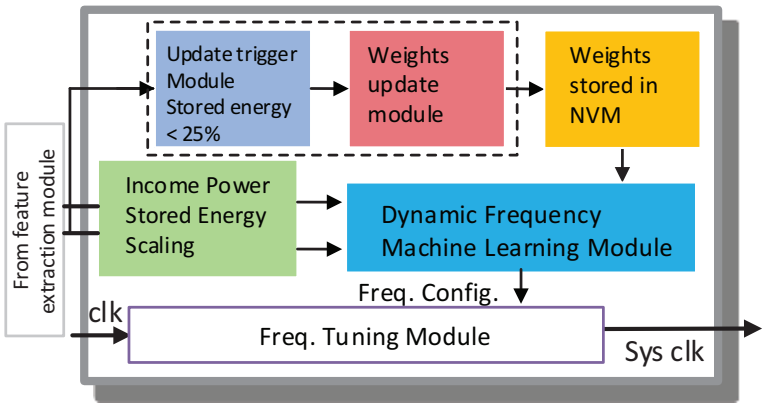


Figure 4.8. Proposed dynamic frequency scaling structure.

a frequency, more power overhead is consumed because of more backup and recovery penalties. Both lead to reduced forward progress. In subsequent sections, the best static frequency 672kHz is used as the comparison baseline for dynamic frequency scaling solution.

4.2.2 Proposed DFS Architecture

Figure 4.8 shows the diagram of the proposed DFS architecture, as a part of the controller integrated in the system clock path. The dynamic frequency machine learning module generates the frequency configuration signal for frequency tuning module. The machine learning module consists of a forward propagation network with scaled income power and stored energy as inputs. The initial weights are trained offline and stored in NVM. As the processor power consumption varies greatly with the amount of resources being used, as shown in Figure 4.3, the proposed DFS architecture adapts to the resources allocation policies. Once the stored energy level is less than 25% of full stored energy while the income power is still able to power a 32kHz processor with minimum resources, the weights update module is triggered to update the weights with a one-step lower frequency than the current one.

4.2.3 DFS Neural Networks

This neural network decides the best frequency that the NVP with fixed resources should run at, based on the input power and the stored energy. The neural network is similar to

the structure in Figure 4.6. It has 2 entries: power income level and stored energy level. There are two hidden layers for 32*2 hidden neurons, and 32 outputs as the predicted possibility to select the frequency. The frequency with the largest possibility will then be selected (Simplified Softmax layer). The initial neural networks achieve above 98% accuracy with 10k training set.

4.3 Methodology

4.3.1 Simulation Infrastructure

The simulation infrastructure consists of several parts as shown in Figure 4.1: the inputs of power profiles and testbenches; a bottleneck resource predictor implemented based on Pybrain [74]; a frequency predictor for NVP; dynamic configurable NVP. The NVP simulator provides features like resource usage, power and energy level, to the resource allocator. By combining these features and pre-trained weights, the allocator gives feedbacks to NVP with microarchitecture selection results. The NVP uses these configurations to reduce energy per instruction and maximize the forward progress. For each given power profile and testbench, the simulation results are forward progress, etc. The frequency predictor generates frequency configuration predicted results for maximum energy used for computation. In order to integrate with the bottleneck resource predictor, the frequency predictor also has an online weights update module to adapt itself to the unstable microarchitecture.

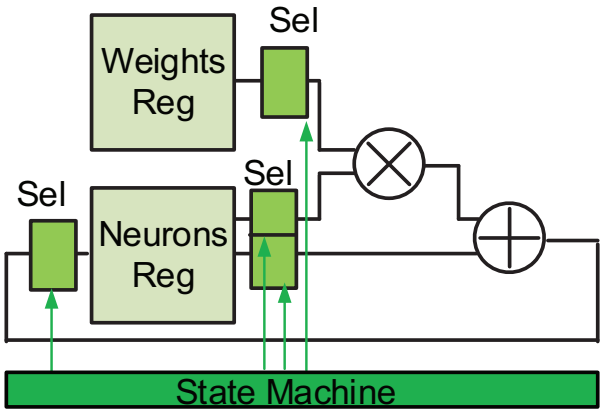


Figure 4.9. Neural networks computation serial architecture.

4.3.2 Testbenches and Power Profiles

MiBench [73] on ”large inputs” is used as the core evaluation suite. In addition to Mibench, some neural network algorithms are also used as testbenches: ADALINE: Adaline network for pattern recognition, classification of digits 0-9 [75]. ART1: Adaptive resonance theory network, brain modeling stability-plasticity demonstration [76]. BAM: Bidirectional associative memory, heteroassociative memory association of names and phone numbers [77]. BOLTZMAN: Boltzmann machine [78]. BPN: Back-propagation, time-series forecasting [79]. CPN: Counter-propagation network, determination of the angle of rotation [80]. HOPFIELD: Hopfield model, associative recall of images [81]. SOM: Self-organizing map, reinforcement learning approach [82]. The power profiles are WiFi home/office profiles, measured in home/office environments.

4.3.3 Overhead Analysis

Making predictions and effecting the changes in resource configurations and frequency imposes some overheads. I use one neural network prediction module to predict bottleneck resources one by one, and then the frequency prediction. I implement the neural network predictor using dedicated hardware, as the software overhead would be untenable. I evaluate the overhead of the predictor by synthesizing the prediction module using a 32nm library with VDD=0.85V. The neural network serial architecture shown in Figure 4.9 has only one multiply accumulate (MAC) module, and a state machine is developed to select the weights, source neurons, and target neurons from the ROM or register files. The neural network predictor can run at a maximum of 156MHz, but we run it at 10MHz, considering the low frequency of the NVP. The power is 3.36uW, and it costs 141 cycles to finish one prediction for one bottleneck resource prediction. The energy cost of making one resource prediction is 47.36pJ. Similarly, the frequency predictor costs 711pJ per prediction. These overheads in energy correspond to 5.9% of average WiFi energy income during 0.2s interval (0.2s*10uW=2uJ). The additional power and energy sampling circuits also impose overheads, but we consider these negligible due to only being employed once per 0.2s. The area is 23744um^2, 2.3% of a Non-pipelined processor. A single prediction takes 3.5uS on average to complete. When no prediction is being made, the circuit is power-gated. These overheads are included in all predictor results.

4.4 Evaluation and Dissicussion

In this section, I first examine the efficacy of the bottleneck resource predictor and smart frequency predictor, each in isolation, and then consider a system employing both techniques.

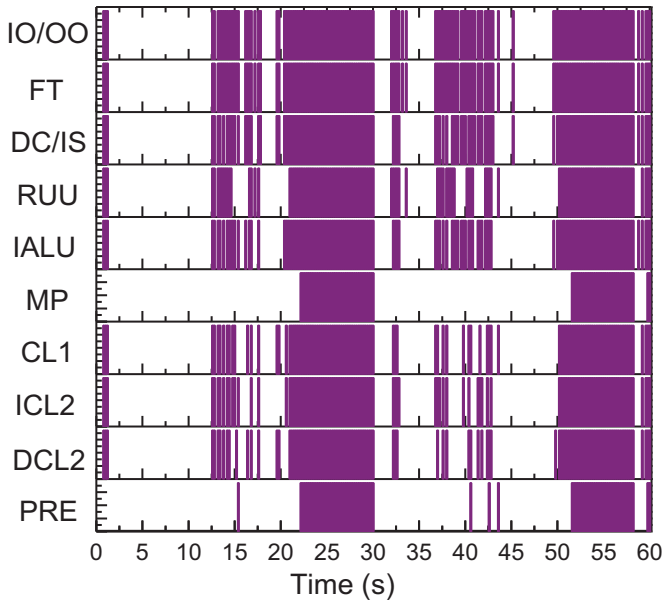


Figure 4.10. Testbench ”susan_corners” resource allocation.

4.4.1 Bottleneck Resource Prediction Results

In the test, a home WiFi power profile is used as inputs. The testbenches are Mibench ”large inputs”. Figure 4.10 and Figure 4.11 show the resource prediction results for two different testbenches on the same home WiFi power profile. The processor runs only in a portion of the total time. With more power available, the controller predicts to power on more resources to reduce the energy per instruction. When the input power is high, and the energy storage capacitor is full, the neural network controller predicts to power all the resources for the maximum forward progress, regardless of lower energy per instruction. Different testbenches may require different configurations for best forward progress. If we compare testbench ”rijndael_encoder” in Figure 4.11 to ”susan_corners” in Figure 4.10, the ”PRE” branch prediction is more likely to be the bottleneck resource, while the ”ICL2” instruction cache level 2 is not. The system provides a relatively high

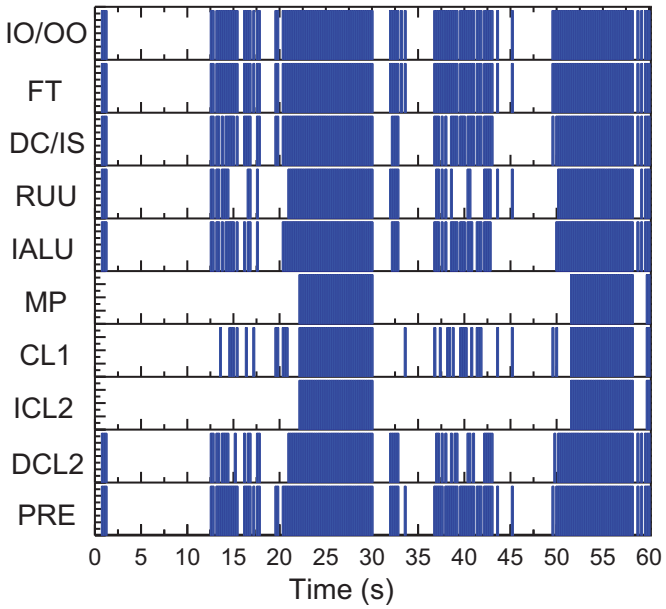


Figure 4.11. Testbench "rijndael_encoder" resource allocation.

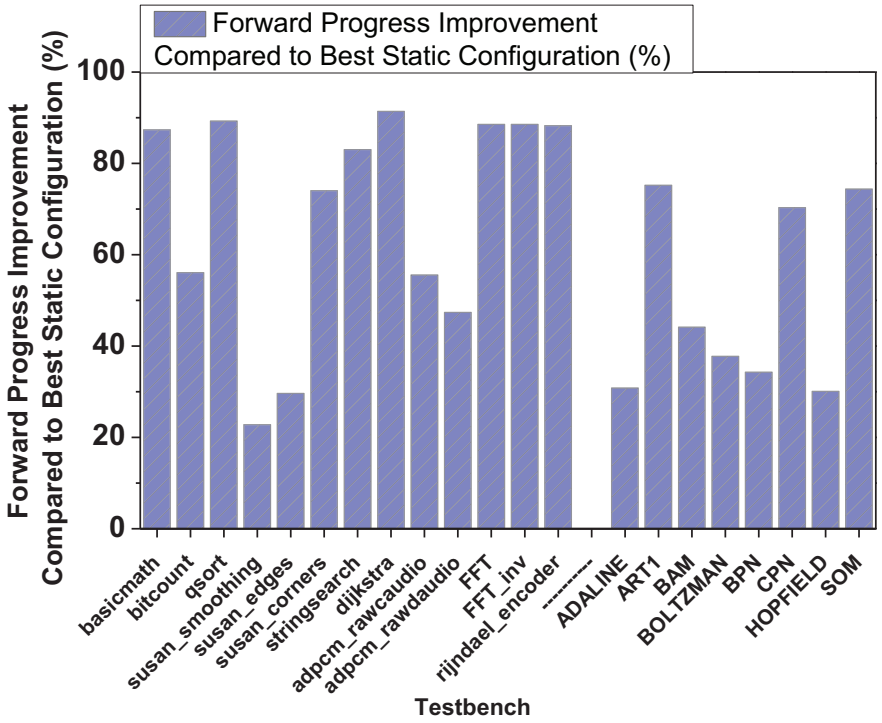


Figure 4.12. Bottleneck resource prediction: An average of 61.8% forward progress improvement.

tolerance for prediction errors. Moreover, it is difficult to define an ”absolute” error. For example, one good prediction is: utilizing the power income aggressively, then running with the minimum resources configuration at the next cycle. In the experiments, the predictor selects one configuration with the minimum resources at first and the rest of the energy is saved in the capacitor, then the predictor selects one configuration with the best energy per instruction for the next prediction cycle even if part of the energy has been leaked. Thus, the energy storage device provides a tolerance for prediction errors. As long as the forward progress is maximized, the predictor is still a good one. Figure 4.12 shows the maximum forward progress improvement for different testbenches. Both Mibench and some neural network programs are tested. This method provides an average of 61.8% forward progress improvement. Forward progress improves for the following reasons: To begin with, when the input power is low, the predictor generates the minimum resources configuration for NVP to guarantee computation and to reduce the chance of backing up data to save power. Secondly, when the power supply exceeds a predefined power threshold, only the bottleneck resources are powered on. Thirdly, when the power is high and the stored energy level is full, all possible resources are powered on even if the energy per instruction is not the lowest.

4.4.2 Smart DFS Results and Analysis

When smart DFS is applied to the NVP, the system frequency dramatically changes with the input profile. As shown in Figure 4.13, the frequency almost follows the trends of the input power profiles. For a low input power and low stored energy, the smart DFS predictor uses a low frequency to reduce the activation threshold so as to aggressively use incoming energy, rather than storing it. For high power income scenarios, it bursts the frequency to a proper level that can just match the income power. The capacitor does provide some buffering for the energy. Figure 4.13 shows two frequencies for two different testbenches. They have different energy per instruction when running on OoO NVP with fixed resources. This difference results in different frequency profiles. In the time section between 30s and 40s, the frequency for testbench HOPFIELD is higher than that of SOM. But between 40s and 50s, the frequency of SOM is higher than that of HOPFIELD. This indicates that current frequency and dissipated energy has influence on later prediction results through energy stored in capacitor. The capacitor is a cushion

for improper frequency prediction because some of the energy can be saved to be used later, although with some leakage penalty.

Figure 4.14 is the forward progress improvement compared a best static frequency 672kHz, showing an average of 43.0% improvement. The variation among different test-benches is very small because the frequency changes only the energy used for forward progress computation, the EPI factor is offset during the computation for percentage improvement.

4.4.3 Resources Reallocation or DFS?

In its simplest form, we can think about forward progress via the following equation: Forward Progress = Energy used for computation / Energy per Instruction (EPI)

While backups and other overheads affect both terms on the right hand side, there is a clear intuitive mapping from each of our mechanisms to each of the right hand side terms: smart DFS primarily influences energy used for computation, and bottleneck resource prediction targets EPI. However, both approaches compete for the same income power to affect their benefits, and it is not immediately clear how best to apportion power between the two mechanisms.

Powering on all resources is rarely the most energy efficient way to use income energy. However, from the bottleneck resource predictor's perspective, in the absence of frequency scaling, aggressively using a large amount of temporary power income when the energy storage is full or almost full is still a good solution, even if the energy per instruction is not maintained at the best efficiency. When frequency scaling is also merged into the system, this situation changes: Targeting the best EPI point while bursting the frequency to aggressively use the income energy is the better solution at timescales large enough to support frequency boosting.

To support combined operation, I add two more modules to the Figure 4.8 dashed line box. These are needed because the changing resources make frequency prediction more complicated since different consumption levels occur for the same frequency setting. To avoid increased backups, once the stored energy level is less than 25%, the training module is triggered to compute the new weights, and update the weight in NVM, using one step lower frequency.

Figure 4.15 shows the results of the combined approach. One key behavior seen

in Figure 4.15 is that, if the stored energy level is not full, the bottleneck resources predictor is unlikely to predict powering on all resources, and will continue at a more EPI-efficient point. The aggressiveness of the combined solution results in a smaller percentage of time that the stored energy is full compared to baseline, which helps keep the resource predictor operating in an EPI-sensitive region.

It is observed that the EPI drops when more power is available due to powering on bottleneck resources, as shown in Figure 4.15. In contrast, when the input power is very small, the predictor generates minimum resource configurations with higher EPI to ensure the NVP continues running but avoids backup operations. The forward progress of the combined prediction scheme does fall short of what would be expected if the two techniques were orthogonal (i.e. 2.30X over baseline). This is because the bottleneck predictor still can either predict full resources or minimum resources, leading to a deviated EPI from the most efficient one.

Similar gains are seen across WiFi power traces. Across our benchmark suite, Spendthrift shows average forward progress improvements of, 2.09X, 1.94X, and 1.99X for each of 3 power profiles. Minimum and maximum improvements are 2.66X and, 1.76X, 2.16X and 1.84X, and 2.33X and 1.82X, respectively, for each of the three traces. Thus, the average improvement across all three traces for our suite is 2X with an observed minimum of 1.76X over the best static baseline.

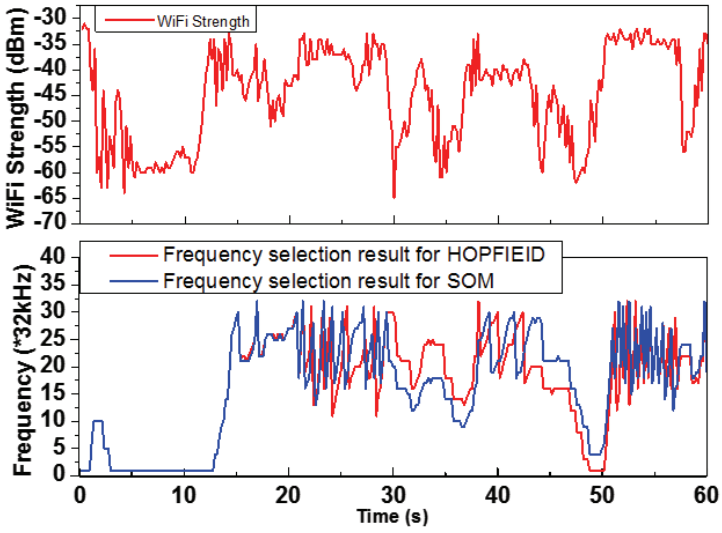


Figure 4.13. DFS frequency selection results.

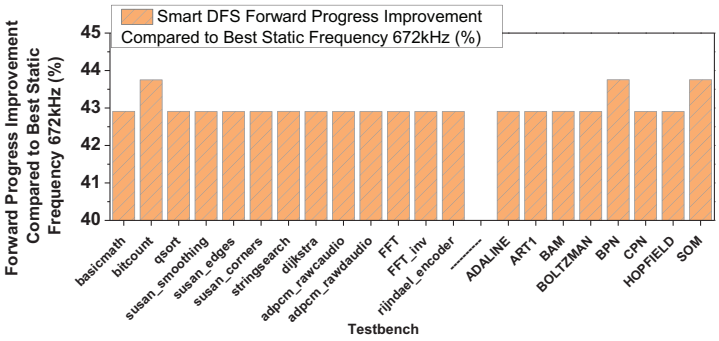


Figure 4.14. DFS frequency forward progress improvement compared to the best static frequency 672kHz.

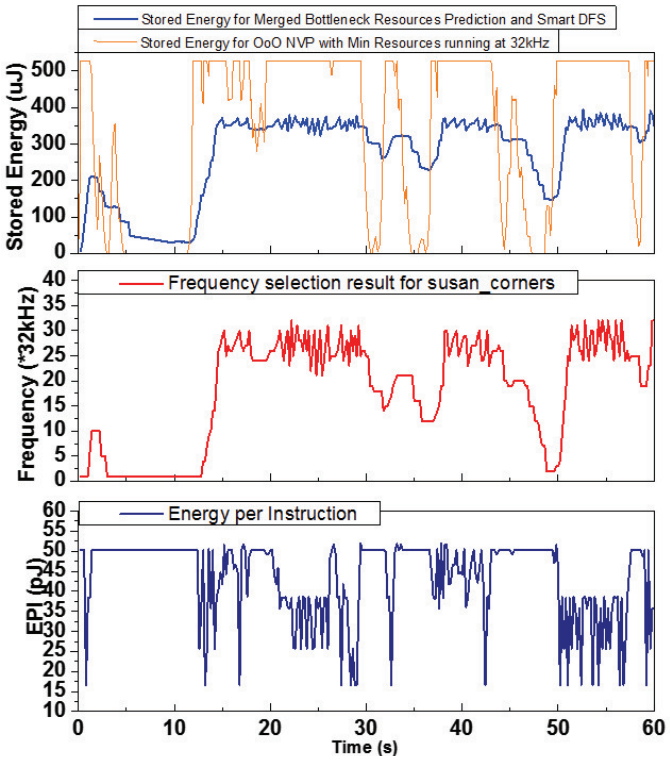


Figure 4.15. Fine-grained simulation results for merged bottleneck resources predictor and smart DFS.

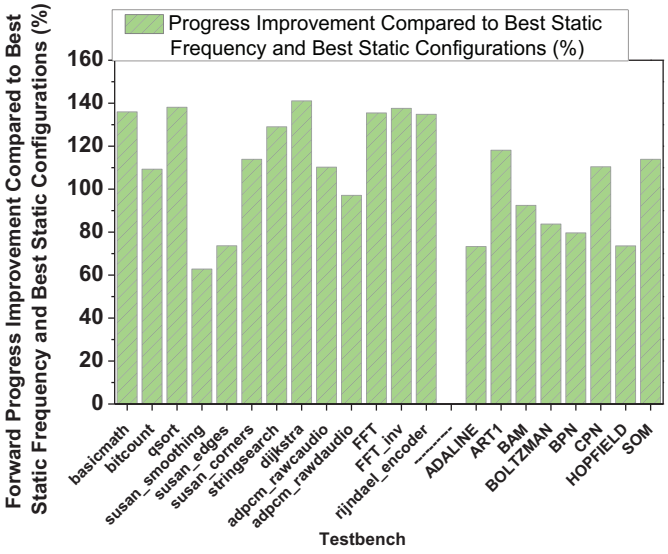


Figure 4.16. Forward progress improvement compared to OoO best static configuration and best static frequency. Average improvement of 2.08X.

Chapter 5

Incidental computing on NVP

Every shift in the way our devices are connected or powered brings with it a potential for revolution in the usage and capabilities of the systems built around them. Just as the transition from wired to wireless telephones led to unprecedented changes in our communications and the shift from wall-power to battery-power transformed our expectations for computational systems, the shift from battery-powered systems to self-powered systems promises to fuel the next revolution in the *Internet of Things* (IoT). The ability to power IoT devices using ambient, scavenged energy liberates them from the lifetime, deployment, and servicing limitations of a fixed battery: Tens of billions of IoT devices are expected to pervade consumer, industrial and public services by the end of the decade [37], and, for many of the target applications in these areas, the replacement of batteries or creating infrastructure to provide a wired power supply makes an IoT-scale approach impractical from a cost perspective.

While ambient energy sources are notoriously fickle, concurrent advances in energy harvesting, ultra-low power computation, and non-volatile memory have enabled a new generation of processors, known as *non-volatile processors* (NVPs), that can withstand the significant temporal variations and even short spurts of "no power" that are common in such power profiles. NVPs tightly integrate non-volatile memory elements into the logic fabric of the processor, thereby enabling almost instantaneous stopping and starting of execution via distributed backup and restore functionality for processor state. Recent device and circuits design exploration in emerging nonvolatile logic and embedded memory has made NVPs faster and more energy-efficient with lower overheads in handling *in situ* parallel distributed backup and restore operations [83–85]. For

NVPs with microarchitectural hardware-managed backup, systems can make persistent progress even if only one instruction successfully completes between power interruptions, and software approaches to manage the semantics of intermittent computation have emerged [86] that can leverage non-volatile memory elements to provide correct execution despite power interruptions. Advances in energy harvesting efficiency has enabled the powering of NVPs from RF energy [26], motion harvesters [87, 88], ambient lighting [89] and thermal variations [90], all of which exhibit significant instability.

Prior efforts on hardware-managed NVPs that perform local computation (in contrast to sense-and-transmit only IoT models) have focused on enhancing the efficiency of converting harvested energy into persistently executed instructions [32,91–94]. These techniques primarily focused on (1) reducing the number and overheads of backups and restores and (2) adapting the compute architecture to exploit dynamic variations in incoming power which would otherwise be wasted due to limited energy storage capability. However, the forward progress metric used in these works does *not* directly capture higher level application semantics regarding the ”utility” of the work performed: *In many IoT applications, temporal and interactivity requirements can make the quality of partial results, or even the existence of any response at all, more important than the fraction of instructions needed to eventually produce a ”best-quality” result.*

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

To capture these notions, I introduce the concept of **incidental approximate computation**, wherein communicated application tolerance for approximate outputs is used to maintain timeliness of responses from an NVP while taking advantage of repetitive

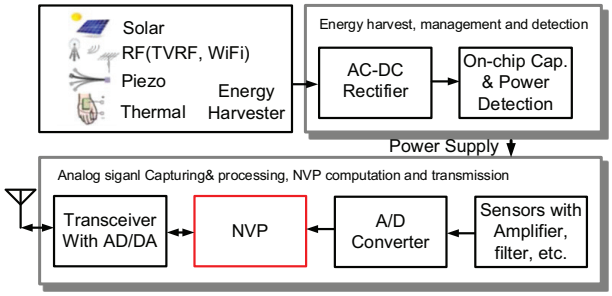


Figure 5.1. NVP-based energy-harvesting system.

behaviors within the NVP's workload to apply any power surplus, when available, to improve the quality of abandoned older work. The paper makes the following contributions:

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

- I propose incidental backup with several retention time matching models and supporting write circuits that can reduce the energy of backup operations through matching the retention time to the combination of the duration of power emergencies and the impacts of reduced fidelity to overall approximation quality.

- Collectively, the incidental approximation approaches improve forward progress by 4.28x improvement within tolerable quality loss.

5.1 System Model

Figure 5.1 provides a block diagram for a general batteryless IoT system powered by ambient energy-harvesting techniques [25, 48, 95–99]. Such systems consist of a set

of energy-harvesting, management, and detection components that comprise the power-provisioning front-end, as well as analog signal capturing and processing, NVP computation, and transmission units that implement the IoT tasks. Note that the front-end modules may vary depending on the energy sources. For the running example, we consider an NVP IoT platform in a ”wristwatch form factor” using an unbalanced ring to harvest energy. The system uses an AC-DC rectifier following the rotational energy harvester. A capacitor is used to capture enough energy to ensure the NVP backup operation and to stabilize cycle-level execution voltages. As a result, the NVP is also in charge of system-level power policies that start, back up, or recover system state.

5.1.1 System Energy Distribution

I measured several prototype platforms in order to quantify the energy distribution of a typical wearable NVP system. The NVP system consists of an NVP running at 1MHz, costing 0.209mW, various sensors, and RF module with data rate 250kbps, costing 89.1mW. The distribution of energy needs between computation and communication varies significantly by application domain. For simple temperature sensing wireless sensing networks, NVP computation consumes 2.4% of total energy, for UV exposure metering, computation consumes 16.8% of energy, and, for more complex data processing like pattern matching and image processing, computation can consume 59.5% to 95% of energy depending on the data processing algorithms. These image signal processing algorithms are surprisingly common in many sensors, including gas sensing (spectrum analysis), water quality monitor (spectrum analysis and image processing). Motivated by this, I focus the evaluation efforts on a collection of image signal processing kernels widely used in post-sensing data analysis as testbenches. One feature I observed is that the input data for these applications are usually buffered frame-by-frame, with no data dependencies between them. On the other hand, for the power levels available through an unbalanced ring class of harvester, more than 80% of the captured data may have to be abandoned in order to meet output deadlines due to weak data processing capability in the NVP. Processing the historical buffered data with incidental computing is one solution to get at least some low quality results, which can be enhanced later by ”incidental recomputation”, rather than abandoning these inputs entirely.

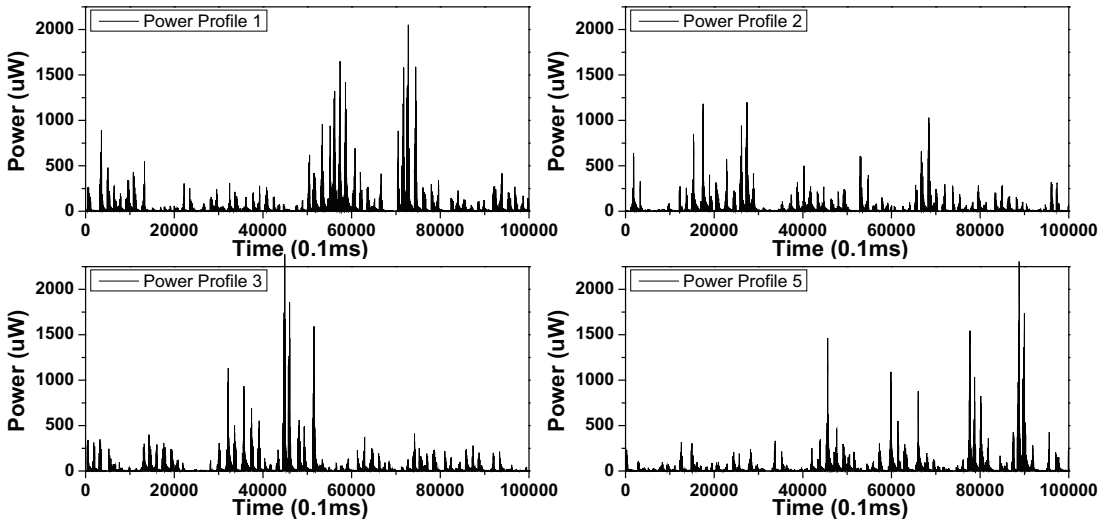


Figure 5.2. Power profiles of ”watch” in daily life use

5.1.2 Turning Energy to Forward Progress

A wristwatch harvester can generate an average of 10uW to 40uW power in daily activities [87, 88]. However, these profiles are unstable, varying from 0 to 2000uW at a fine temporal granularity as shown in Figure 5.2. Assuming a processor operation threshold of 33uW, the system can experience 1000 to 2000 power emergencies in a 10s time window. Due to rollbacks and lost work in such an environment, many prior works propose **forward progress** (the number of instructions that have *persistently* committed) as a key ”execution metric” for comparing the work done by processors.

A traditional strategy in energy-harvesting systems is to employ a volatile low power MCU or an MCU with checkpointing capability (e.g., FeRAM MSP430 is used in [48]) that waits before starting to execute while charging an energy storage device which must be large enough to store sufficient energy to complete an entire logical work unit, such as an image frame [48]. Systems operating on this paradigm will alternate between periods where they accumulate energy in the energy storage devices (ESD), and ones where powering the system drains the energy. While such a system is able to offer strong guarantees for execution once execution begins, this conventional solution has several limitations, including energy conversion efficiency overheads brought by frequently charging and discharging the capacitor, capacitor leakage [48, 49], minimum charging current(e.g. 20uA for the GZ115 [49]), and slow charging curve [49]). Moreover, if the incoming unit of work is too large, the incoming power may not be sufficient

compared to leakage in the ESD [48], or there may be long periods of complete power outage that drain the accumulated charge, and it may take arbitrarily long to reach the threshold for beginning execution.

An alternative execution paradigm is to utilize only a small on-chip capacitor, i.e., one sufficient for backup operations and employ an NVP. This reduces capacitor leakage, and can improve front-end conversion efficiencies by mitigating the overheads of moving charge into and out of a large energy storage device at the cost of additional system complexity for the NVP, more complicated guarantees on the granularity of work accomplished once an execution period begins and the overheads imposed during more frequent backup and restore events during the execution of each logical unit of work. The two approaches can be seen as similar if the logical unit of work is at an instruction or similar granularity, thereby minimizing both charging time and charge lost if a shortfall occurs during a charging period between backup and recovery. Hybrid approaches have also been proposed. For example, Sheng *et al.* propose a dual channel front-end solution to overcome low charging efficiency [50] in which they design another power channel to bypass the energy storage device and connect directly to the load, and Ma *et al.* extend prior NVP models to maintain the capacitor energy level [33] within a bounded range for charging efficiency during execution rather than greedily consuming energy. *Thus, the key energy tradeoff between the two approaches is between the energy wasted on charging and discharging a capacitor with leakage and the backup and recovery overheads of NVP.*

It is observed that the NVP-based execution approach can outperform the wait-compute scheme by 2.2X-5X for the power traces shown in Figure 5.2.

5.2 Incidental Computing

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

5.2.1 Incidental Computing

Roll-forward Instead of Roll-back. I make the following key observations for IoT applications. In many deployment scenarios, catching up quickly after a power failure may take priority over the quality of response. Furthermore, such applications often contain kernels with independent loop iterations that could conceivably be skipped over in their entirety. However, skipping represents in a sense a maximum quality reduction, especially considering that each iteration, especially in image / signal processing kernels, performs the same essential computation on different data. Finally, while average power, even during periods of sufficient power to allow uninterrupted execution, is low in harvested systems, peak power can be substantially higher than average.

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

Below, I discuss the details of incidental approximate SIMD computing. When a power failure happens, the computation states are backed up with the stored energy, and the data that are marked as incidental, like variable ”src” in Figure 5.8, are backed up using the assigned unreliable storage policy in the NVM. When the power recovers, if a roll-forward is indicated (”$incidental_recover_from$”), instead of recovering from the backed-up PC, the PC is set to the place marked by ”$incidental_recover_from$”. From the application’s perspective, the program rolls forward to process newer data from the buffer (in the example of program in Figure 5.8, it’s a new frame of data). As a result, the newest captured data are always processed as the first priority.

During processing of the new data, the microarchitecture controller compares the current computation state to the state backed up using the old data. If there is a match, an SIMD strategy is applied to the old state and data. Note that, the computation precision of the newly-added SIMD for old data depends on the income power level under the control of the ”incidental” pragma’s ”minbits” and ”maxbits”. In this way, a minimum

quality can be guaranteed by ”minbits”, and the energy beyond the amount necessary for full precision processing of new data are instead applied to the old data for incidental SIMD computing. If the computation is interrupted again, both the new data and SIMDed old data become incidental, and a newest data computation begins from ”$incidental_recover_from$”. Note that multiple old data can be SIMDed. In the current implementation, at most 4-way SIMD can be achieved.

Recompute and Combine (RAC). I assume that, in general, the importance of data drops over time. If some old data are later found to be ”interesting”, and demand high precision output to validate ”uncommon results”, an *incidental recomputing* can be performed. Instead of inserting an interrupt into the current program, incidental recomputing employs incidental SIMD to recompute the old data, and tracks the precision of sub-component outputs. These two versions of the outputs can then be merged by combining the best precision sub-components from each run. After multiple recomputations and merges, I expect much better quality outputs as demonstrated in Section 5.7.5. It is important to emphasize that, in this incidental recompute method, a better quality result can be achieved without affecting the current data processing loop.

5.2.2 Incidental Backup

For the three power profiles shown in Figure 5.2, power supply unreliability would cause an NVP to perform as many as 1400 to 1700 backups per minute, costing 20.1% to 33% of the total income energy (simulated and measured with running testbenches shown in Figure 5.28). Approximate computing provides an opportunity to substantially mitigate these overheads by relaxing the reliability (i.e., write energy reduction brings the probability of flipped data storage beyond expected retention time) of the ”lower order” NVM bits used to back up data during power emergencies, and using commensurately less energy for backup and recovery operations. Moreover, if the energy reserves needed for backup are reduced, fewer power emergencies may occur.

Retention Time Shaping (RTA). Current NVPs [32, 91–94] utilize nonvolatile technologies with maximum retention times on the order of a decade or more, and parameters tuned to maximize both retention and reliability. However, most power emergencies in wearable harvesting devices last just a few ms, and are rarely more than a fraction of a second. Figure 5.3 plots the duration (left) and frequency of power emergencies (right)

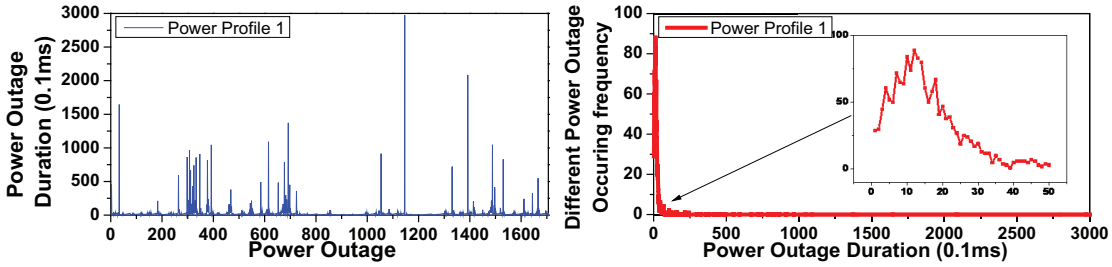


Figure 5.3. Power outage duration (left) and statistics (right)

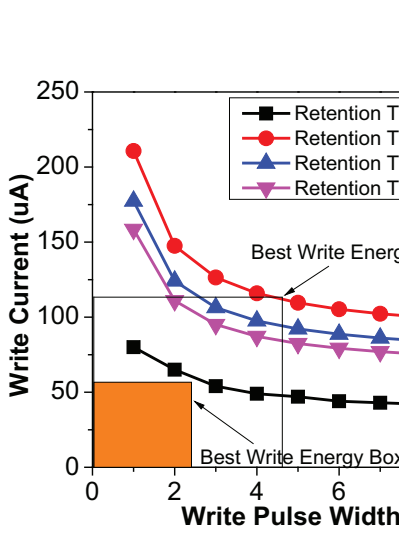


Figure 5.4. STT-RAM Write energy & retention time [1–3]

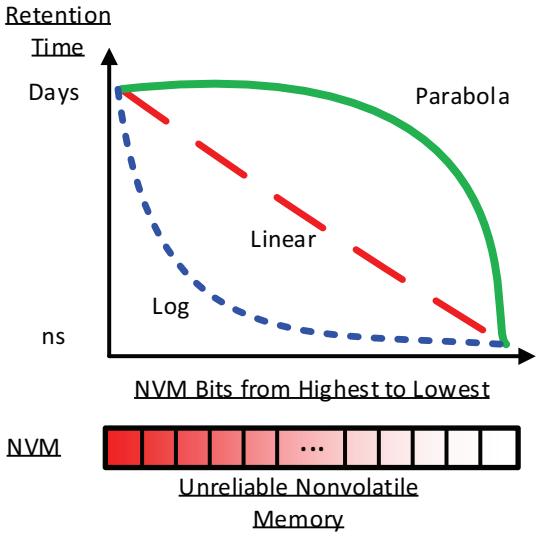


Figure 5.5. Retention Time Shaping (RTA)

in the examined traces.

By matching the retention time to the power interval profile, the write energy can be significantly reduced. From the perspective of write energy for the backup operation, Figure 5.4 shows the relation between STT-RAM[1] write current and write pulse width for different retention times. I note that 77% of write energy can be saved, for instance, by reducing the retention time from 1 day to 10 ms. However, applying a retention time reduction uniformly is very difficult to implement profitably for two main reasons: First, future power income is, in general, very difficult to predict, and, second, the cost of prediction failures can be very high.

Approximate computing eases the practical adoption of such an approach. Higher

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

order bits are retained with longer duration, preventing catastrophic quality loss, while lower order bits can be unreliably persisted, saving energy. I consider three retention time reduction functions to shape the retention time in a way that reduces from the most significant bit to the least significant bit, as shown in Figure 5.5.

Our three retention time reduction policies are: linear (Equation 5.1), log (Equation 5.2), and parabola (Equation 5.3). B stands for bit index; for this example, it is from 1 to 8, and the T is the retention time, whose unit is 0.1ms.

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

I propose these three policies based on observations of the relationship between bitwidth precision and final result quality, considering both program features as well as power source profiles [100]. Through a quantitative analysis of the relationship, with MATLAB as a tool for regression and expression, I provide three retention time policies to trade off between energy and qualities.

$$T = 427B - 426 \tag{5.1}$$

$$T = \pi^{(B-1)} + 9 \tag{5.2}$$

$$T = -61B^2 + 976B - 905 \tag{5.3}$$

5.3 Architectural Support

In this section, I introduce the architectural support needed to implement the incidental approximate computing concept outlined in Section 5.2.

Microarchitecture Support for Incidental Computing. The high-level design modifications and microarchitectural support needed for incidental computing are illustrated in Figure 5.6. A control unit (Figure 5.6, bottom) dynamically controls *whether ap-*

proximation should be used and, if so, how. The main task of this unit is to set the number of precise and approximate bits for SIMD for different hardware components based on the available power level. One of the outputs of this unit is approximation control bits, which are used to control approximation in the register files, ALU, pipeline flip-flops, and data memory. Approximation can also be globally disabled by a running program via unsetting the AC_EN register, overriding the decisions of the control module and forcing "full precision" execution.

Another important functionality is to manage the control transfers of incidental computing, namely *from which point the SIMD execution should begin*. This is decided by comparing the current PC and the values of key loop variables (e.g. induction variables analyzed by the compiler at compile time, such as variable "n" in Figure 5.8) against buffered values. To implement this, an additional circular nonvolatile buffer within the controller records the PC of the last N (four, in the implementation) resume-points from which the SIMD operation can begin. Once a power failure happens, the system states are backed up: the resume-point PC is backed up in a 2Byte*4 size buffer made of nonvolatile flip-flops in the controller; the register files are stored in multi-version nonvolatile registers. The oldest value is overwritten (discarded in FIFO order). When incidental SIMD is enabled, the current PC is compared against stored resume-point PCs. If the current PC matches one of the stored PCs, the controller has the modified register file generate a bit-vector indicating which register values associated with the matching resume-point PC have values identical to the current register values. This vector is then combined with a compiler-generated mask. Once matches in both PC and the mask-indicated variables are observed, SIMD width is increased and the buffer storing the SIMDed resume-point PC is cleared.

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

Configurable approximate ALU: Direct extension to 4 versions. The ALU also performs approximate computation using the techniques proposed by [101, 102]. Our ALU can also work on packed SIMD operands.

Power-gated register files: Each register is designed with nonvolatile logic, has an AC bit, and is extended from 8 bits to 32 bits (4 versions) for incidental computing. These extensions can be powered off when incidental computing is not employed. Comparison circuits are also added to indicate an identical match between the current register

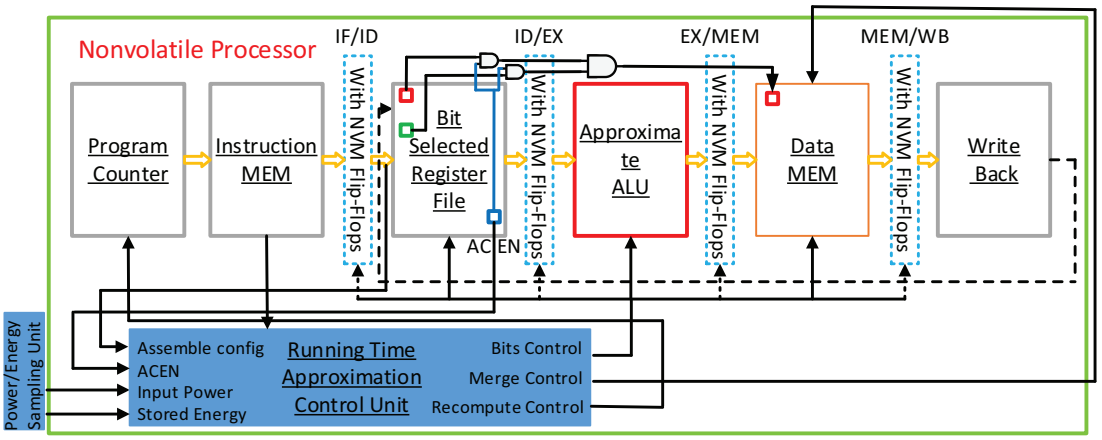


Figure 5.6. NVP execution approximation scheme. Global AC enable bit "AC EN" can be configured via writing to specific register.

value and the values of prior versions. Comparison is governed by the controller, which both enables the comparison circuits and specifies the version to compare against.

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

Recomputation is implemented through an instruction marks a pragma-indicated control point into the nonvolatile PC buffer for comparison. The difference between incidental recomputation and normal incidental SIMD is that the programmer can set up the PC to be the one marked with "incidental recover from" rather than the one before power fails. After the recomputation finishes, an instruction requests the controller to use the multi-version memory to combine the new results with previous ones according to a specified policy conveyed by the programmer. No further execution can occur until the controller has completed the combination operation, using a state machine to iterate over the specified memory region one pair of memory values at a time. The combination options supported by the memory are max-precision (metadata max), max-value, min-value, and sum.

Hardware Support for Incidental Backup. Considering that the write time and write current both affect retention time, the NVM write circuits can be redesigned as shown in Figure 5.7. As shown in the cross bar STT-RAM array part (green background color), one STT-RAM has three nodes, "Bit", "BitB" and "Write Enable" signals for one bit

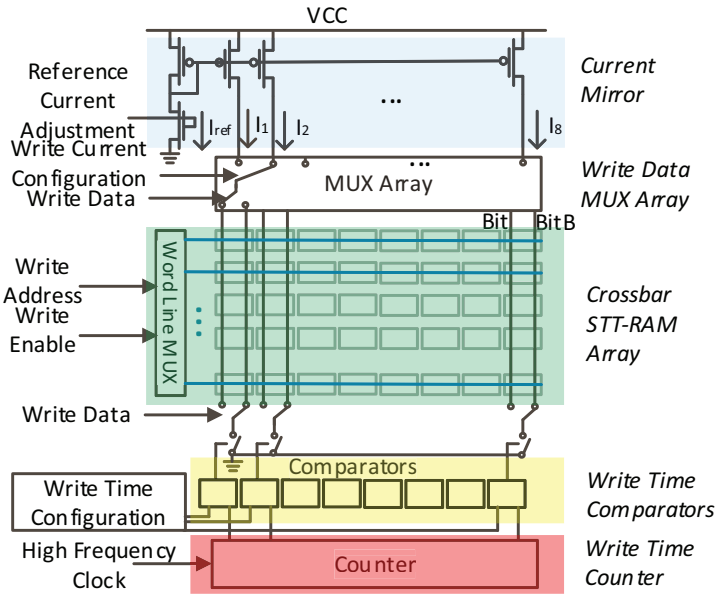


Figure 5.7. Proposed dynamic retention control scheme

cell. The write data can be changed through flipping the current direction of ”Bit” and ”BitB”, under the control of the ”Write Enable” transistor. The big idea of implementing such write operation supporting dynamic retention time is to apply write current control on one line in either ”Bit” or ”BitB”, and to use the other line to control the write duration time. Similar retention time tradeoffs can also be observed from ReRAM, PCRAM [103, 104], and FeRAM [105], and the dynamic retention time control scheme can be extended to these devices.

The write current for different retention time is generated by a current mirror, shown in blue background in Figure 5.7. I_{ref} is a baseline current. Different write currents, from I_1 to I_8, can be generated with little overhead by provisioning multiple output current mirror circuits with different W/L ratio of PMOS transistors, because the maximum current variation ratio is less than 3X from 1 day to 10ms, requiring only a small number of variant W/L ratios. The STT-RAM process variation [106] can be adjusted with the ”Current Adjustment” signal during test in fabrication. Different currents can be selected in the MUX array according to different configurations (Log, linear etc). The write current is connected to either ”Bit” or ”BitB”, depending on ”Write Data”.

The other line of ”Bit” or ”BitB” controls the write time. A high frequency 4-bit counter (sub ns per cycle) is implemented, and a comparator for each column of cells is designed to compare the counted time with the pre-set threshold stored in the nonvolatile

”Write Time Configuration” module. Once the counter time reaches the threshold, the write operation is terminated by breaking the connection to GND.

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

5.4 Software Support

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

Table 5.1. Semantics for supporting incidental computing

Pragma support: In order to support incidental computing, four pragmas are provided, as listed in Table5.1:

- incidental (src, minbits, maxbits, policy). This pragma indicates the allowed range of bitwise precision (with which a variable can be approximated) as well as its storage approximation policy. The first line in Figure 5.8 indicates that variable ”src” can be approximated within a range of [minbits, maxbits] and the retention time policy is ”linear”.

- incidental_recover_from (variable). This pragma is used only with induction variables in looping constructs. The example line 4 in Figure 5.8 means that variable ”frame” is used to indicate a fixed recovery restart point. Instead of recovering fromthe last instruction, the NVP skips over the remainder of a partially completed loop iteration to begin a new iteration.

- recompute(buf, minbits). Once some data are found to be ”interesting”, and we want to perform a recomputation to further improve the output quality with ”minbits”. When we obtain the new results, we use the the next pragma to merge the new results with the previous ones.

- assemble(buf, assemble_mode). This merges the new data with the previous, using one of the following strategies: sum, max, min, or higherbits. The ”higherbits” option means that the results computed with higher bits cover the results of the lower bits.

Programmer's role: The programmer can use the pragmas to provide the compiler with three pieces of information: (i) to guarantee a minimum output quality, the programmer needs to mark the data that can be approximated, and how they can be approximated; (ii) the programmer also needs to indicate where the program should recover from; and finally (iii) in cases where some data are deemed ”interesting” (as opposed to ”incidental”), the programmer needs to indicate how recomputation will be performed and how the results of this recomputation will be combined with old data to generate results with higher precision.

Compiler's role: The compiler uses the parameters provided by these pragmas to set some of the bits/values in the architecture: (i) the ”AC” bit for variables like ”src”; (ii) a recovery program counter; (iii) key variables within a loop, like ”n” in the code example in Figure 5.8, in order to accurately match the break points; (iv) if the target code cannot be incidentally SIMDized, it is replicated multiple times, each with different inputs; and (v) we generate an instruction that configures a register to trigger hardware-based data merging.

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

Note that, while prior research [107] proposed various directives for approximate computing, some of the actions taken by the compiler when processing the directives are entirely different from prior approaches, as we target an NVP based environment. Other approximation techniques [107–109] target ”active” approximation, while our approach is ”passive” due to limited energy in energy harvesting scenarios. That is, in our model, approximation is exogenously induced by insufficient power on a computation that is precise both in default and preference.

5.5 Putting Incidental Computing All Together

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

Figure 5.8. Example program with annotations

I now go over an example program fragment (Figure 5.8) to show how pragmas are used and how they interact with the underlying hardware. This sample program implements a portion of the median kernel.

I mainly focus on how to setup the first two types of pragmas and parameters, since the re-computation pragma is easier to understand. The first line marked with (a1) indicates that the programmer wants variable ”src”, which is a frame of data from an image sensor, to be incidentally computed between 2 bits and 8 bits, and the unreliable memory policy is linear. Line 4 marked with (b) indicates that the recovery program counter should be marked to the instruction that begins the update of the induction variable ”frame”.

The bottom part of Figure 5.9 shows a portion of the detailed power profile of power profile 2 in Figure 5.2. The grayscale power profile shows that, although there are some power spikes around 700*100us and 2700*100us, they are not ”dark” (in color) enough (maintaining high power) to generate ample energy. The baseline 8-bit NVP has lowest threshold individually, leading to 42% system-on time. And the 4-SIMD has highest threshold, resulting in 3% system-on time (regarded as 3%*4=12%). Both of them can not achieve best forward progress. The system start threshold for incidental pragmas (a1,b) - [2 bits to 8 bits] is lower than that of (a2,b) - [6 bits to 8 bits]. The system performs recovery, computation or recomputation, and backup as shown in the top and middle portions of Figure 5.9. Due to different threshold levels, the NVP running pragmas (a2,b) has at least 6 bits, to guarantee a minimum output quality. However, the system is on in 16% of the total time period due to higher threshold. In comparison, the NVP running pragmas (a1,b) runs more instructions than all other solutions (although system-on is 38.7%, lower than 8-bit NVP 42%, but FP is 3.7X if incidental results are also considered, compared to 8-bit NVP), with more backup and recovery operations, and generates lower quality incidental outputs.

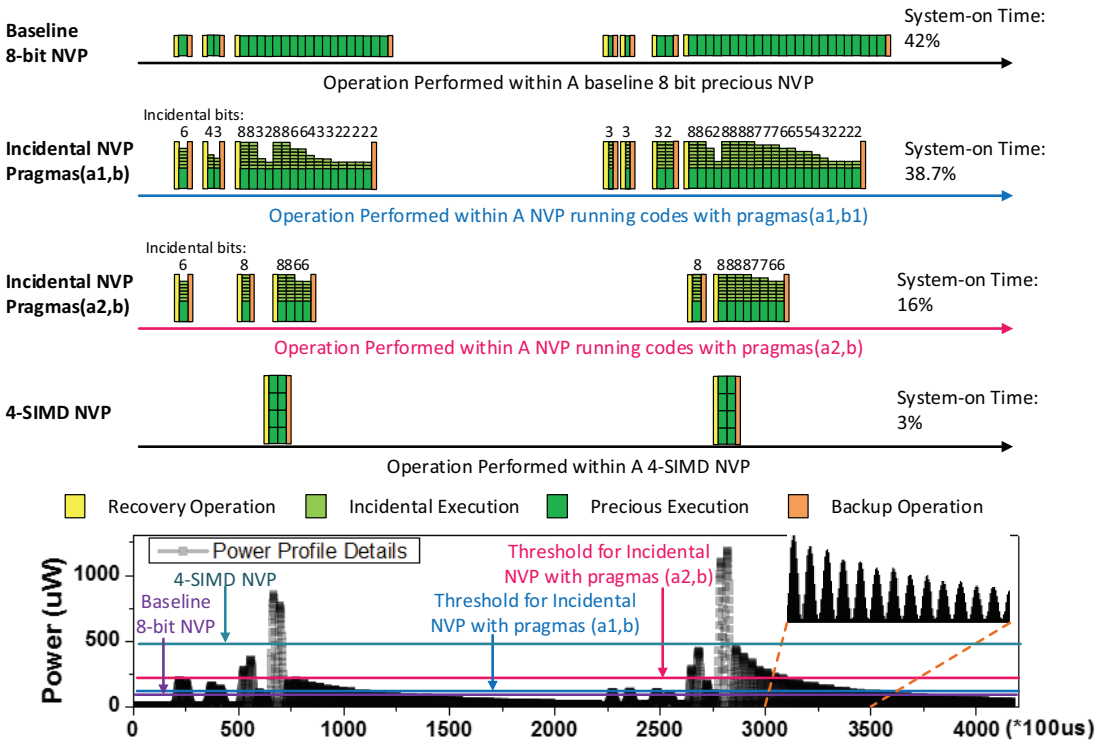


Figure 5.9. Timing based behavior analysis

The programmer should set the minbits lower if the application is to be run faster, but with low quality incidental outputs. If, however, the low quality outputs turn out to be ”interesting”, the programmer may choose to recompute to obtain better quality outputs. Note that, if we look into the power profiles, we can see that in tiny scales (Figure 5.9 bottom right), there are still lots of glitches in the power, meaning different computation bitwidths for different elements in the ”src” array. Note that this is more aggressive than the always-high-quality (a2,b) configuration.

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

For the recomputation, the programmer is in charge of checking whether there are

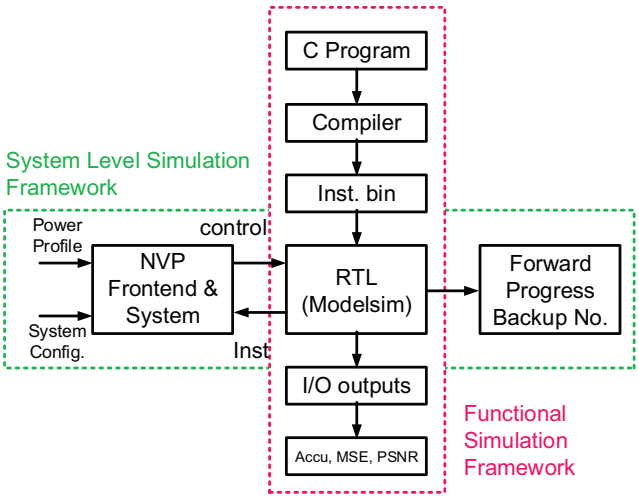


Figure 5.10. NVP framework with both functional and system-level simulation.

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

5.6 Simulation and Validation

The simulation framework consists of two parts as shown in Figure 5.10. The first part is a functional simulator, the core of which is a modified 8051 RTL [32], which I further modified with support for incidental computing logic and approximate memory. For framework compatibility, the inputs are generated as ROM arrays, and the outputs are generated through GPIO P2 and P3. I compiled the sthece code, and modified it to embed the ”AC” bits. The RTL running in Modelsim initializes the ROM, RAM, etc. The quality analysis for image outputs is performed by computing PSNR and MSE in MATLAB.

The second part of our framework is a system-level simulator. This system level simulation implemented in Matlab, and Python handles the system-level components including parameters and features of analog front-end circuits, capacitor etc., which cannot be implemented in RTL. The inputs to this simulator are the power profiles sampled every 0.1ms and the system configuration parameters such as the system capacitor size, capacitor leakage, chip leakage, front-end circuit efficiency, system start threshold,

backup energy threshold, and recovery threshold. This system-level simulator controls the RTL simulator steps and gets the decoded instructions in order to decide various polices that dictate energy consumption. The system-level simulator, together with the functional simulator, generate important output metrics such as the *amount of forward progress* and the *number of backups*.

To test the effectiveness of the approach, I used several image processing and pattern matching kernels from MiBench [110] for the following reasons: Firstly, as discussed in Section 5.1.1, the computation in NVP dominates the energy consumption in the whole system for systems where sensing tasks utilize computationally intensive post-sensing algorithms, as are common in image processing and pattern matching. Secondly, these kernel are widely used in many post-sensor processing algorithms including gas sensing (spectrum analysis), water quality monitor (spectrum analysis and image processing), and power spectrum analysis of heart-rate variability. Finally, image processing kernels have also been employed on other successfully prototyped energy harvesting platforms in the literature [25, 48, 111].

To validate the choice of NVP execution rather than wait-compute for an energy-harvesting image-processing platform, I simulate frame rates for a volatile wait-compute platform (33uW power volatile MCU, the same model adapted in the NVP. Sensor power is not included) powered by the same ”watch” energy harvester. For a 256*256 image size, similar to that used in other prototyped platforms [25, 48], susan.corners, susan.edges, and jpeg.encode are 1.65s, 4.9s, 12.55s per frame individually. An NVP solution without approximation [28] can improve this to 0.97s, 2.28s, 5.22s per frame individually with no quality loss. Incidental techniques can further improve to 0.3s, 0.59s, 1.2s per frame individually with minimum preset tolerable quality loss as shown in Table 5.2. Based on these observations, I believe that image/signal processing kernels running direclt on IoT devices can be an important workload in the energy-harvesting domain.

5.7 Results and Disscussion

In this section, I first evaluate the impact of the individual approximation schemes on quality of output (adding noise or losing detailed information), and then quantify the forward progress contributions due to individual techniques employed. Finally, I provide

results from the holistic evaluation.

5.7.1 Bitwidth vs. Quality

I use two key metrics to quantify the output quality compared against an 8-bit non-approximate baseline, namely, mean squared error (MSE) and peak signal-to-noise ratio (PSNR). To establish quality baselines, I first investigate MSE and PSNR for fixed-known-correct bit approaches, varying the bitwidth. This will allow us to ground the exploration of approximation in the NVP context from an output quality viability perspective. The N-bit reduced-quality ALU preserves the upper N bits and produces random outputs for the lower $8-N$ bits, whereas the non-preserved bits in the reduced quality memory are truncated, and the operations using their values are treated as shifted N-bit operations. The approximate ALU models gradient VDD for different bits in ALU [101, 102], hence the addition of noisy bits rather than truncation. Figures 5.11 and 5.13 show the image output from three testbenches, namely, sobel, median, and integral, for ALU and memory bitwidth reductions, respectively.

I first consider ALU bit-quality reduction. As seen in Figure 5.12(a), the MSE for median and integral increases significantly when using less than 3 bits in the ALU. In contrast, for sobel, the MSE increases dramatically when there are fewer than 6 bits, indicating that sobel is not as amenable to fixed-width approximation as median. Figure 5.12(b) shows PSNR for reduced bit widths; traditionally, above 20-40 dB is considered a good PSNR response. For median and integral, even operating at a bitwidth of 1 can provide quality above 20 db, and 40 dB is achieved at 4-6 bits whereas sobel cannot achieve even 20 dB with anything less than full precision.

Bit reduction in memory produces somewhat distinct outputs from ALU precision reduction. As can be seen in Figure 5.14, the MSE measure of quality drops further than with ALU bit reduction. PSNR in Figure 5.14 (b) is similar to the approximate ALU solution. The PSNR metric is more similarly affected by either adding noise or losing detail compared to MSE, which is more sensitive to loss than noise.

5.7.2 Progress vs. Quality

Two other key evaluation parameters are forward progress (in the NVP sense of *persistent* forward progress), represented by number of instructions committed with a given

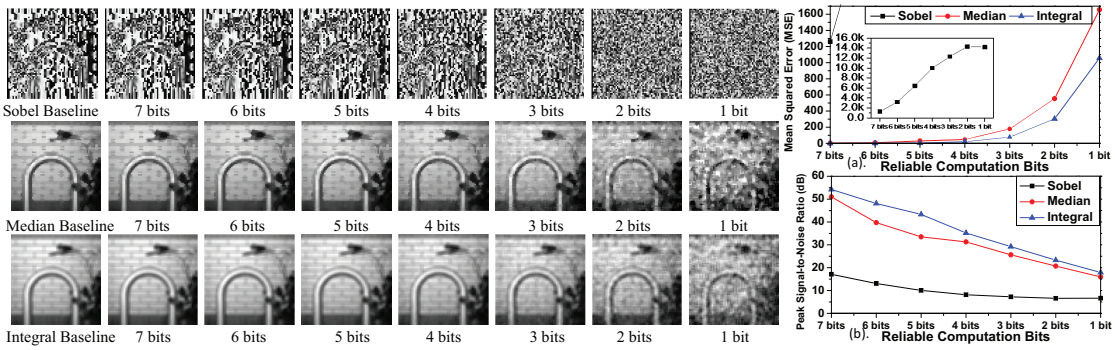


Figure 5.11. Impact of approximate ALU on image quality.

Figure 5.12. AC ALU MSE PSNR

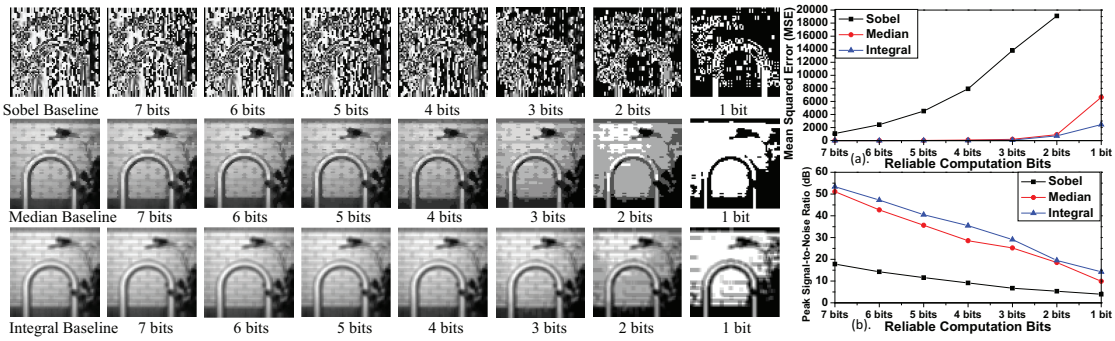


Figure 5.13. Impact of approximate memory on image quality.

Figure 5.14. Unreliable Memory MSE PSNR

power profile, and the number of backups. Figure 5.15 shows the forward progress achievable when the number of reliable bits in both the ALU and memory are reduced in tandem. By reducing the bits from full precision (8 bits in the 8051 NVP) to 1 bit, the forward progress doubles. The number of instructions executed increases for two reasons: not only is the power per operation reduced, but the lower power consumption and reduced local state to back up combine to trigger fewer power emergencies and to provide a lower activation threshold, leading to a higher duty cycle. As can be observed in Figure 5.16, the number of backups reduces by an average of 45% when the number of bits is reduced from 8 to 1.

5.7.3 Dynamic Bitwidth Approximation

In dynamic bitwidth, the computation of incidental approximation and memory bits changes along with the power profile. While this approach does not provide a bitwidth guarantee stronger than the above 1-bit solution, it will, in practice, execute at a much higher average bitwidth. Figure 5.18, shows the change in precision over time for three

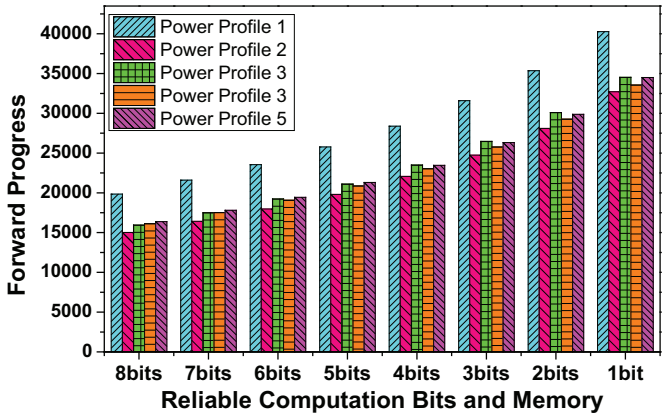


Figure 5.15. Forward progress on different bitwidths.

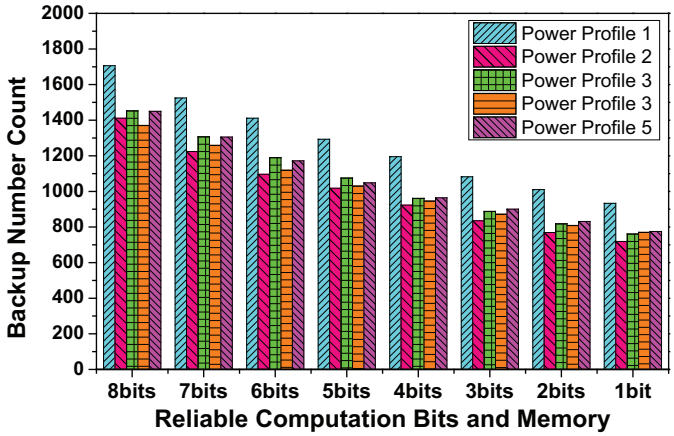


Figure 5.16. Number of backups on different bitwidths.

of the power profiles, and summarizes the distribution across bitwidths at the right of the figure. Figure 5.17 shows the output for the median testbench. Examining the MSE and PSNR in Figure 5.19 and the forward progress depicted in Figure 5.20 reveals that the execution quality of the dynamic bitwidth approximation is roughly comparable to a 2-bit solution, but the dynamic approach achieves an additional 20% forward progress

(a), Median with power profile 1

(b), Median with power profile 2

(c), Median with power profile 3

Figure 5.17. Impact of dynamic bitwidth on median.

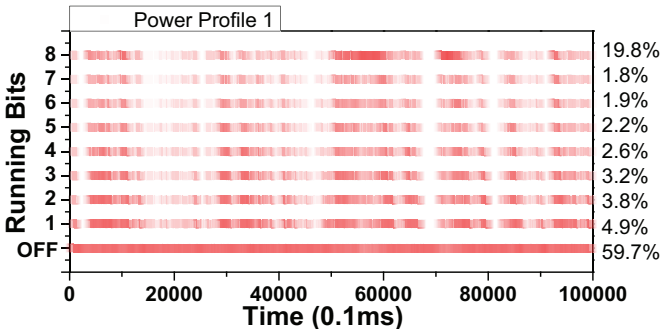


Figure 5.18. Utilization of bitwidthsof profile 1 for Median

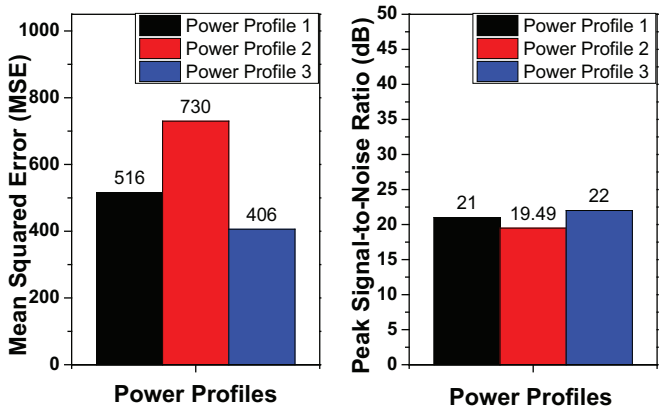


Figure 5.19. QoS of dynamic bitwidth on Median.

over the similar quality fixed-bit approximation approach.

As previously mentioned, some kernels, such as the sobel testbench, are not as amenable to approximation as others. Rather than allowing dynamic bitwidth to be entirely determined by power profiles, it may sometimes be necessary to guarantee a higher minimum quality of results. For median, I find that the 4-bit-minimum-dynamic approach achieves similar MSE and PSNR results, across three power profiles, (MSE = 1.46, 1.72, and 1.72. PSNR = 46.5dB, 45.7dB, and 45.8dB) to a 7-bit fixed bitwidth solution, while achieving 22% more forward progress in Figure 5.21.

5.7.4 Backup and Recovery Approximation

The quality of outputs, both visually, as seen in Figure 5.26 left part, and by PSNR, in Figure 5.24, is similar for different retention time polices, although both retention time failures and MSE scores vary more widely. As seen in Figure 5.22, the retention failure counts for each bit vary significantly across both power profiles and policies, ranging

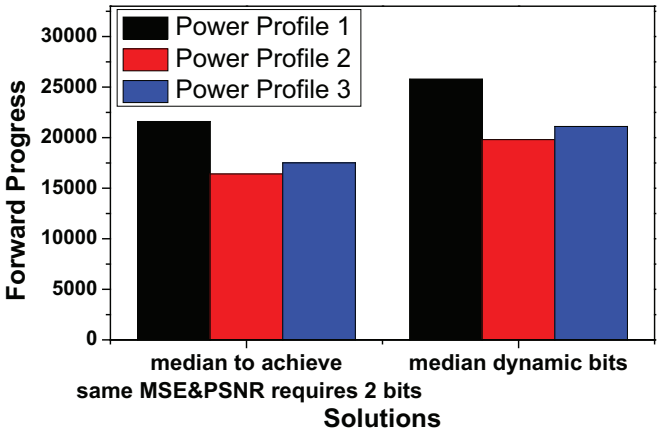


Figure 5.20. FP of dynamic bitwidth for Median.

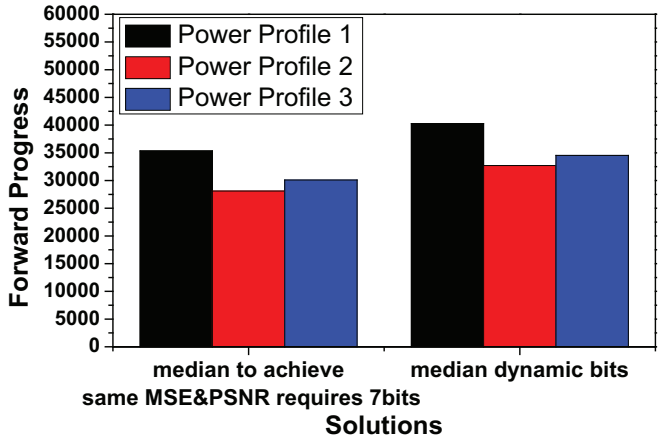


Figure 5.21. FP of the "MinBits=4" for Median.

from 15 to 1200 retention time violations. Surprisingly, the log retention policy has the best MSE (Figure 5.23), as well as the best PSNR (Figure 5.24), among the three policies tested. From this, I conclude that either the relatively low total number of bit errors on higher bits, while higher for the log policy, is still well within the tolerance of the approximable algorithms and the result reflects random variations in output quality (i.e. noise in the measurement of noise sensitivity), or some amount of noise is preferable for the applications examined. While the latter is possible, it seems the less likely of the two for the majority of the kernels examined in this work.

All the retention reduction policies reduce the backup energy requirements. Since the energy used to perform backups is reduced, it is available for increased computation, resulting in an average of 50% forward progress improvement, as shown in Figure 5.25. The log policy frees the greatest amount of energy and the parabola policy the least,

with consistent trends in the forward progress benefits from these policies.

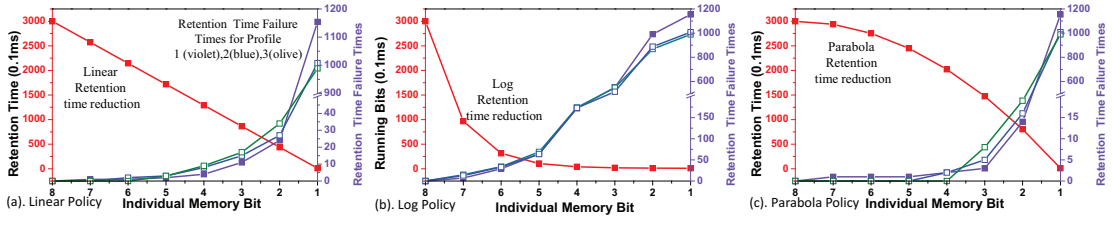


Figure 5.22. Backup, retention, and failure times with different bitwidths for (a) Linear, (b) Log, and (c) Parabola

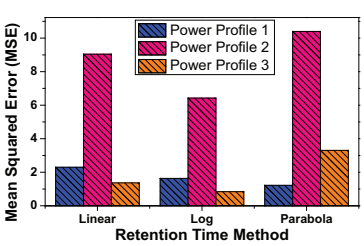


Figure 5.23. MSE vs. retention.

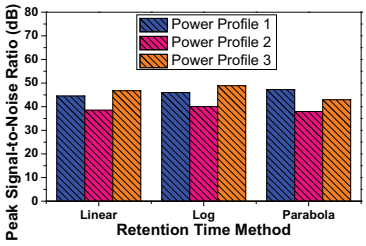


Figure 5.24. PSNR vs. retention.

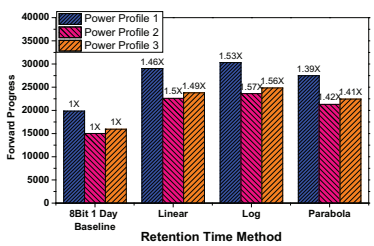
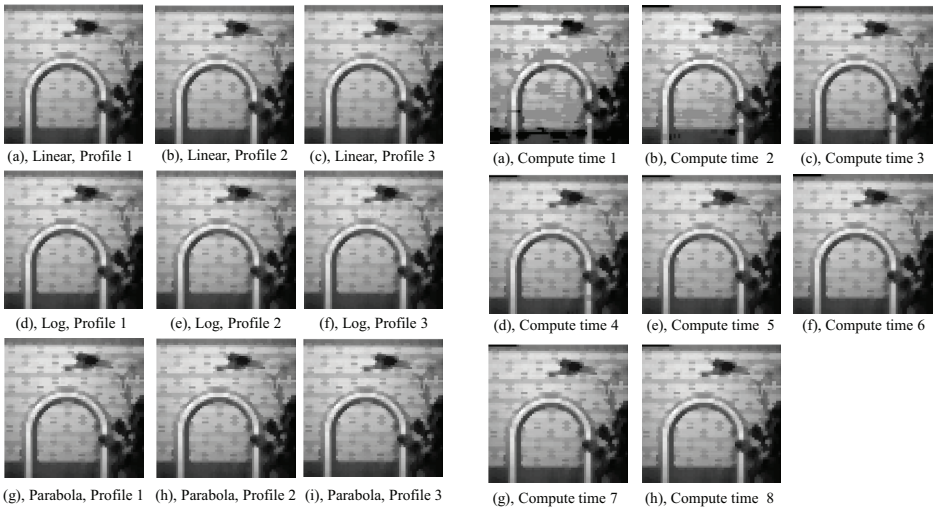


Figure 5.25. FP improvement.

5.7.5 Recomputation

As previously shown, reducing the bitwidth in an NVP can improve forward progress, ***producing an earlier output at the cost of quality.*** For instance, as seen in Figure 5.15, an initial, 4-bit result can be produced roughly 1.5x faster than an 8-bit result. For an application with a real-time deadline, any remaining slack time could be used to improve image quality, whereas, because of power-uncertainty, always waiting for the 8-bit full precision result will more frequently miss deadlines. For algorithms whose outputs are derived from highly independent computations, such as in many image processing kernels, recomputation can be used to replace output elements that were computed with low precision with higher precision recomputed outputs.

Below, I present an exploration of the potential benefits of the incidental recomputation using a *model* that always performs entire output passes with dynamic precision and then takes the highest precision output pixel from each and merges them. Figure 5.26 right part shows the outputs of recomputation with varying minimum bitwidths and Figure 5.27 shows the quality improvement as a function of the *additional recomputation*



Results with different retention time policies. Results with recomputation.

Figure 5.26. Visual results on different retention time policies(left), and recomputation(right).

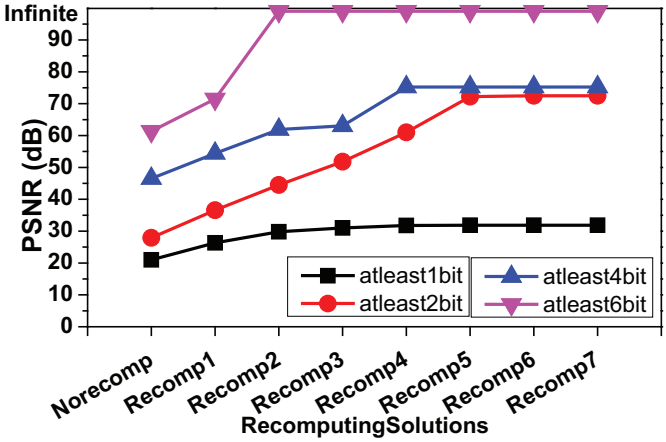


Figure 5.27. Impact of recomputation on quality.

passes. Note that there is little value in recomputation beyond four to five passes. However, the approach employed is able to capitalize on random variation in the input power profile to perform iterative improvement.

5.7.6 Putting It All Together

All these incidental techniques together can provide adjustable tradeoffs between QoS (quality of service) and forward progress, which provides programmers with a design space to play with through a debug-test-modify loop until the QoS reaches the minimum requirements (this is akin to experimenting with various parallelization options in

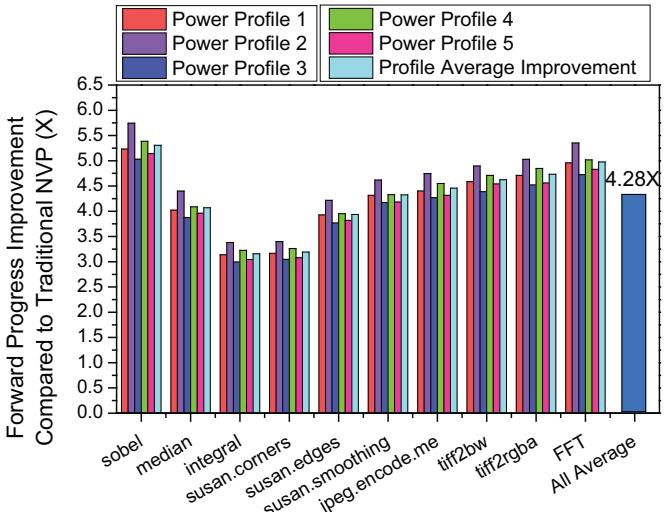


Figure 5.28. FP gain of incidental computing & backup

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

Table 5.2. Targeting at QoS, fine-tuned incidental policies.

OpenMP before picking up the right one). Table 5.2 shows an example of policies that I have fine-tuned to target QoS. For all testbenches except JPEG, I define the QoS target in terms of PSNR. For all the power profiles tested, the listed targets are always achieved except for the JPEG testbench. In the JPEG encoding testbench I apply incidental computing only on motion estimation, wherein approximation-induced error affects only the size of the compressed output. I define our QoS target for this kernel to be an output size that is no more than 50% larger than the full-precision compressed output. Across all power profiles, 97% of the 25000 compressed output frames from JPEG met QoS. For all kernels, incidental approximation is applied with full precision in the current iteration and dynamic bitwidth for incidental loop executions.

Based on the observation and experiences, the programmers should first decide the ”minbits” to make the QoS above the QoS threshold, then reduce the ”minbits”, and try to fine-tune the incidental backup policy and the recompute times to compensate the QoS

loss. I also suggest to employ linear incidental backup when average power is expected to be higher (e.g. scenarios akin to profiles 1, 4) and parabola when average power is low (e.g. profiles 2, 3, 5); preference for the logarithmic policy over linear/parabola is strongly kernel-specific. If the expected power characteristics are unknown, a lookup table or machine learning based mapping from the sampled power to configurations can be applied.

Figure 5.28 plots the gains of an incidental NVP with fine-tuned policies in Table 5.2. I observe a 4.3X improvement over precise NVPs [32]. Several factors contribute to the gains of incidental approximation: (1) omitting execution of some instructions, replaced with incidental computing, (2) dynamic approximation reduces power consumption, and (3) incidental computing provides the SIMD benefits of reduced instruction fetch energy. Overall, the gains vary substantially from testbench to testbench, due to, in large part, the different predefined pragma loop lengths. The variation among different power profiles is largely due to slight variation of the energy per instruction within these testbenches.

5.8 Summary

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

Chapter 6

Nonvolatility-exploiting Fog Computing - A System Level Perspective

As the integration of a new technology percolates through systems of increasing complexity, new optimization opportunities are consistently found in its wake. For example, research developments in non-volatile elements, such as ReRAM [112], STT-RAM [113], and PCRAM [114], have led to use patterns for nonvolatile memories [103] distinct from their volatile counterparts and enabled nonvolatile logic-compatible elements such as NV-DFFs [115, 116] and nonvolatile SRAM [117] that can support distributed on-chip backup and restore operations. These developments, in turn, have led to the exploration of nonvolatile processor architectures (NVPs) that rely on these integrated nonvolatile circuits to provide new guarantees for intermittently powered execution. As an increasing number of NVPs have been proposed and several have now been fabricated [32, 92–94, 118] with varying feature sets, new opportunities will arise as components of existing multi-node systems are replaced by their nonvolatile counterparts.

Energy-harvesting wireless sensor networks are a major sub-domain of the *Internet of Things* (IoT), and many such systems have already been successfully deployed. The applications that make use of this paradigm are diverse, including area monitoring, e.g., the position of the enemy; environmental monitoring; industrial monitoring; medical and health-care monitoring; traffic control systems; underwater acoustic sen-

sor networks; and near-body wearable device networks. Despite their diversity, most systems operating on energy harvested from ambient power sources share a common core design pattern in their operation as *normally-off systems* (NOS). First, each node is designed with a large super-capacitor or rechargeable battery capable of storing enough energy to perform at least one complete unit of work. Then, in deployment, the node waits, harvesting energy, until this energy storage device is sufficiently charged before starting a simple micro controller (MCU) to oversee the collection of a data sample from the sensor, and, after very limited processing, transmits the sampled data. While there is increasing research interest in moving more computation to sensing platforms [24–26, 119–122], these systems have traditionally avoided relying on complex local computing in energy harvesting nodes based on the following two assumptions: 1) energy harvesting power is unreliable, so shorter work periods are preferable for reliability and cost/form-factor reductions in necessary energy-storage capacity, and 2) the net effect of redundancy and recomputation mechanisms for circumventing the unreliability of energy income is that both computation and communication at the sensor node are energy expensive, constraining both work done at data collection and the topologies of the deployed networks. Table 6.1 demonstrates examples of currently deployed energy-harvesting sensor applications built on this model.

Existing System	**Energy Source**	**Sensors**	**Network Topology**	**Transmitted Data**
Bridge Health Monitor [121, 122]	Solar, Piezoelectric	Accelerometers, piezo-sensors	Zigbee Chain Mesh	Raw sampled data
Wearable UV Meter [119, 120]	Solar	UV sensor	Star	Raw data
Joint-less Railway Temp. Monitor [123]	Solar	Multiple temperature sensors	Zigbee Chain Mesh, GPRS	Raw uncompressed data
Machine Health Monitor [124, 125]	Piezoelectric, thermal, RF	3-axis accelerometer,vibration, temperature	Star, bus or tree	Raw data
RF Powered Camera [24–26]	RF Source, WiFi	Image sensor	Point-to-point backscatter	Raw image pixels

Table 6.1. Functionality and components of current energy harvesting WSN system

However, advances in both hardware [28, 33] and software [24, 86, 126–130] management of integrated nonvolatile resources to more effectively compute and communicate [4] in intermittent power environments warrant reconsidering these high-level assumptions. Computation under unreliable power supply is now relatively reliable on

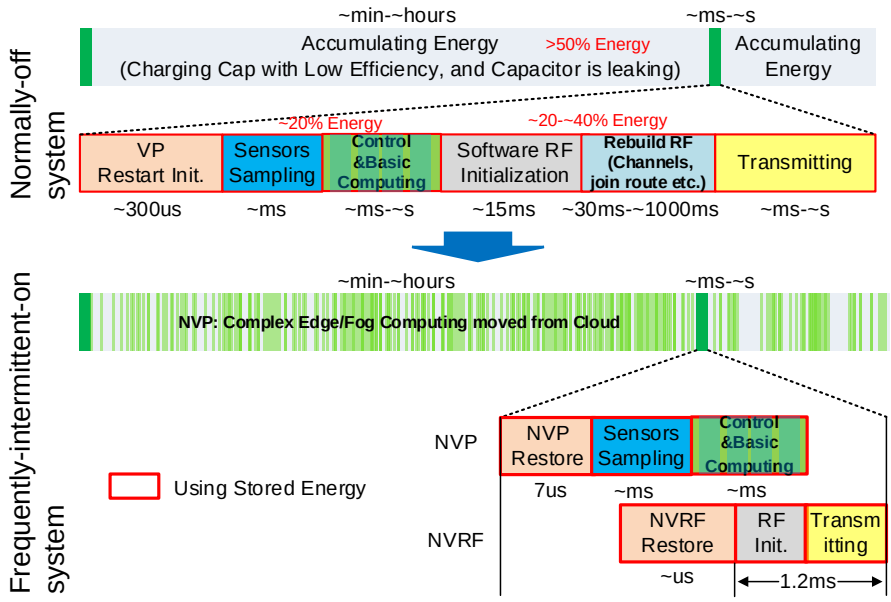


Figure 6.1. Optimizing from normally-off system (NOS) to frequently-intermittent-on system (FIOS)

processors with managed, integrated non-volatile storage (e.g. NVPs). In addition, NVPs have been shown to make better forward progress than their volatile counterparts given the same incoming power, i.e., NVPs are not only more reliable, but are also more efficient than their predecessors (if only in unstable power environments). Recent works that look beyond the processor to leverage nonvolatility in the communication path of these platforms have shown that acceleration [4] of the initialization of the RF module through the introduction of a nonvolatile RF controller (NVRF) can substantially reduce the time and energy cost for data transmission. Collectively, these advances have weakened prevailing assumptions about the cost and reliability of computation and communication at the sensor node level. Thus, it is likely that the traditional optimizations for collections of such nodes that aim to limit on-node computation as a first-order goal may no longer be effectively capitalizing on the current opportunities within these distributed systems, and new algorithmic and system level redesigns should be explored for the next generation energy harvesting based wireless sensor network systems.

To understand the potential changes stemming from adopting nonvolatile nodes, I have deployed and measured various real NVP-based systems. To explore how utilizing the node level nonvolatile features affects system tasks, I introduce optimizations that leverage NVP and NVRF efficiency and reliability to trade increased com-

putation for reduced transmission, changing the system from a *normally-off system* (NOS) to a frequently-intermittently-on system (FIOS), as shown in Figure 6.1. In addition, I propose an efficient load balancing approach for NV-mote chains under certain wireless protocols. I then explore new optimizations, driven by realistic user requirements [131–133], that leverage the features of the platforms with both NVPs and NVRF (NV-motes), specifically NVRF state-share among nodes, to allow adding node virtualization to improve quality-of-service (QoS). Analyzing the benefits of the new approaches, I combine these discussed features to propose the **NEOFog** architecture for the next generation fog computing.

This chapter demonstrates that, by exploiting node-level nonvolatility, the NEOFog approach can provide new optimization opportunities and improve system performance and efficiency due to the following contributions:

- Introducing nonvolatility into nodes improves the computation efficiency and reduces the energy for data transmission. Accordingly, I optimize the programs from RF-energy-dominating to computation-intensive, and from normally-off to frequently-intermittently-on, so as to better utilize the new opportunities brought by nonvolatility.

- Considering (i) the imbalance and time-varying income power and stored energy level of each node and (ii) different energy requirements of different workloads, I propose a distributed load balancing algorithm specially optimized for ”unstable power supply” in an intra-chain level to balance the loads and further increase the computation done in the fog. I show that it can increase the amount of data processed in-fog by 1.9X to 2.1X compared to a system that employs a traditional load balancing algorithm.

- I propose architectural support for improving QoS via increasing nodes virtualization and slotted time-division multiplexing enabled by the NVRF in a fashion not previously viable. A nearly 2X QoS improvement is observed for a 3X multiplexing in low energy income situations.

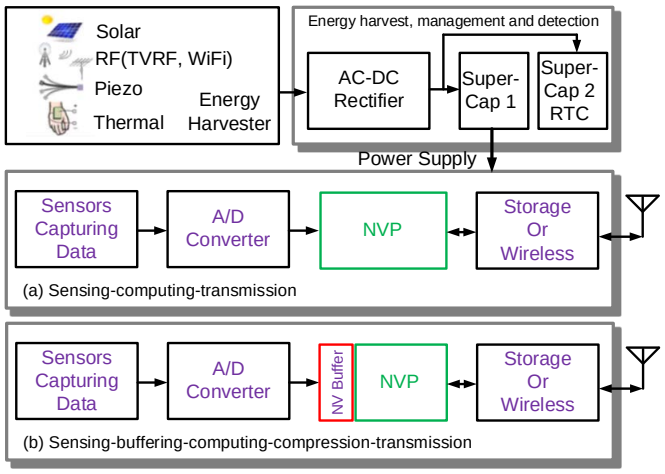


Figure 6.2. NVP-based energy-harvesting system organization.

6.1 System Model

In this section, I first introduce a typical energy harvesting system. I then explain, step by step, how nonvolatility changes the processor, buffer, and RF. Lastly, I discuss network topology and construction.

6.1.1 Traditional Wait-compute Systems

System components. Figure 6.2(a) illustrates a block diagram for a typical NVP-based WSN node powered by energy harvesting. There are four types of ambient energy that are widely available and relatively easy for commodity systems to harvest: 1) solar, via photovoltaic cells; 2) RF via antennas; 3) piezoelectronics, via vibrations of either the substrate or entity to which the harvester is attached; and 4) thermal energy, via thermal gradients across a thermoelectric. All these energy sources lack stability, varying with different locations, angle, etc. of the node's dynamic environment. Front-end circuit design is specific to the AC or DC characteristics of the input source. Capacitors are adopted to temporarily store the harvested energy. The rest of the node consists of data sensing, ADC, NVP, storage, as well as transceiver.

In our platform, two super-capacitors are employed, one for powering the real-time clock chip, which is utilized to synchronize the wireless communication, and another one for powering the rest of the node. The real-time clock super-capacitor has a higher charging priority because if it loses power entirely, then, when the system recovers,

resynchronizing with the logical time slots in the network imposes large overheads compared to normal state restoration.

Wait-compute system timing and energy. Figure 6.1 (upper) shows the typical execution pattern for a NOS, in which most of the time is spent accumulating energy into the capacitor. The system starts only when there is sufficient energy stored in the capacitor. System activation begins with processor initialization for about 300us, followed by sensor sampling under the control of the processor, then activation of the RF parts, whose initialization penalty is large, and finally raw data are sent out.

One version of the WispCam project [25], powered by RF, is a typical system example using this approach. In the described deployment, the system first accumulates energy for 15 minutes (5m away from the RF source), and then starts the system for three seconds. Of the three seconds system-on time, only 115ms is spent for data sampling, and the rest is for data transmission under the control of the processor. Due to long charging time with capacitor leakage, as well as low charging efficiency, more than half of the energy income is wasted. Sensing consumes around 20% energy, and data transmission and computation consume 20-40%, even though WispCam uses the backscatter wireless technology, which is extremely energy efficient. Similar properties have been observed in our own observations on the systems shown in Table 6.1, wherein the RF parts, even using Zigbee, dominate the energy consumption. The dominance of communication motivates our study of leveraging the improved computation efficiency of an NV-mote to shift the costs from communication domain to computation domain and adopt a FIOS rather than NOS paradigm.

6.1.2 Nonvolatility in NV-motes

NVP. Nonvolatile processors [32,134–136] have emerged as a promising solution for intermittent unstable power under various energy scenarios. Through distributed fast and efficient nonvolatile logic including NVSRAM, NVFF, etc. [115–117], the computation state within the processors are backed up with an on-chip capacitor and restored when power recovers. NVPs can still achieve forward progress under power failure frequencies which are as high as 100kHz [32], belying the notion that an unstable power supply produces unreliable task completion, even if ensuring timeliness remains challenging.

The FIOS approach can improve system efficiency over NOS by reducing the ineffi-

ciencies involved with charging and discharging an energy storage device at the cost of the overheads of supporting backup and recovery in an NVP. Prior research [29] has indicated that replacing a volatile processor operating as a NOS with an NVP and a FIOS approach can increase forward progress by 2.2X to 5X (depending on the power profile at hand), making this an appealing set of tradeoffs.

Beyond the basic NVP operations seen in fabricated designs, dynamic policies have been proposed to further improve the forward progress of NVPs [28, 33]. In this chapter, I assume that the NVP implements the Spendthrift [28] frequency scaling and resource allocation architecture to provide efficient conversion of incoming energy into completed work.

NVBuffer. To ensure consistent data transmission between the sensors and the NVP, Figure 6.2(b) shows the modification to the nodes to incorporate a nonvolatile buffer (NVBuffer) to guarantee asynchronous data transmission. NVBuffer, which is usually implemented as an NV FIFO, also provides a raw data buffer for load balance between nodes.

NVRF. Traditional nodes operating as NOS require reinitializing the radio frequency (RF) transceiver before transmission, as all configuration and data in the transceiver are lost when a power failure occurs. Traditional software-based re-initialization leads to large overheads due to a long initialization delay between the host processor (whether VP or NVP) and the RF module, which consists of a high performance baseband core (about 10mW in ML7266 or ML7396 zigbee chipset) for computing wireless protocols and logic and analog parts exhibiting high power draw in TX or RX mode (between 30-100mW). Figure 6.3(a) shows the delay path in conventional software RF based initialization. The data stored in NVM (note that, in a traditional VP, this is a separate flash module) are taken out through the bus to the processor that, after some processing of the data, sends the processed initialization data back through the bus to the public SPI interface, and then to the RF transceiver. Measurement shows that the whole initialization process can take as much as 33ms, during which the RF module dissipates enormous energy.

Nonvolatile radio frequency controllers (NVRF) [4] represent an attempt to employ a hardware-controlled nonvolatile IO interface to support fast and efficient initialization of a specified peripheral. By offloading the RF initialization task to the NVRF controller to speed up the long delay between processor and peripheral, 27X speedup is observed

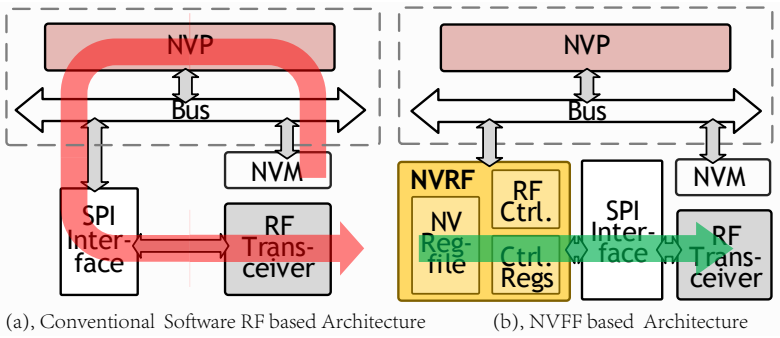


Figure 6.3. Software RF vs. NVRF [4]

(1.2ms) [4]. In fabricated NVRF, the NVRF controller stores the configuration of RF chips in a nonvolatile register file and initialize the RF chip under the control of a finite state machine following special protocols of RF chips; both ML7266 and ML7396 are supported. Figure 6.3(b) diagrams NVRF initialization. Data from the NV registers where the configuration information and the latest transmission data are stored are sent through direct memory access with special designed SPI interface to the RF transceiver. In addition, NVRFs can self-reinitialize once configured by the processor, which totally frees the processor from controlling the RF during many tasks. Even during the periods where the RF is not activated in FIOS, the processor can store the data to be sent and any associated configuration information into the NV data buffer and control registers. Once the NVRF is triggered by a timer or through a control signal from the processor, it sends data out without further intervention.

In summary, the presence of an NVRF controller speeds up initialization by 27X through quick and processor-independent response in a direct nonvolatile memory access (DNVMA) fashion. Reducing the stand-by time of the RF module, whose stand-by power is enormous, saves substantial energy, and prior measurements [4] indicate that NVRF achieves a 6.2X throughput advantage over software-based RF control.

6.1.3 Network Topology and Construction

A problem inherent with unstable power supplies is coordinating both sending and receiving nodes to be active at the same time. To address this, I employ a real-time clock (RTC) powered by a super-capacitor as shown in Figure 6.2(a). The RTC coordinates a common notion of time such that the synchronized sets of sending and receiving nodes can be co-active. The RTC wakes up once in every predefined interval, and as a result,

once synchronized, all the nodes in the network with sufficient energy would wake up at the same time to transmit and receive data, and to again synchronize the RTC. For those nodes without sufficient energy to wake up at the RTC-indicated time, they will wakeup at a multiple of the RTC-indicated time rather than when they first accumulate sufficient energy. If, however, a node exhausts all its stored energy, meaning that even the RTC is desynchronized, the node will wake up whenever it has sufficient power in order to attempt to re-connect and synchronize with the whole cluster network. Another possible solution for the root problem would be to use an RF wakeup sensor [137, 138], but this has not been implemented in our work. In this chapter, I focus on load imbalance of energy income among nodes and the computation requirements of the whole system, which, while in some cases changing how and when the network is used, do not introduce new network architectures.

With respect to network topologies, there are many prior works discussing various self-organized cluster networks [139–147]. It has been shown that nodes preferring to communicate primarily with others in physical proximity can save transmission energy, and relay transmission can also save energy. I regard this as the MAC layer, which is transparent to programmers and schedulers. The most often used network topologies include star, bus, tree and mesh, as can be seen in the systems in Table 6.1. Although a mesh topology is adopted in the bridge monitoring and joint-less railway temperature monitoring systems, the network works like a chain mesh [121] due to the physical locations of the nodes along the railway or bridge. Our proposed intra-chain load balancing and inter-chain node virtualization algorithms specifically optimize these chain mesh systems.

6.2 Distributed Fog Computing

In this section, I explore various approaches for supporting and enhancing distributed fog computing on NVP-based systems powered by energy-harvesting. I group the approaches into (1) *Node* level optimizations for increasing compute capability, (2) *Intra-chain* level distributed fog computing load balancing schemes, and (3) *Inter-chain* level optimizations to enable leveraging slotted time-multiplexing node virtualization for QoS.

6.2.1 Node Level - Reoptimizing for an NV-mote

Integrating nonvolatility into nodes or their individual components, and managing hardware resources accordingly, has been introduced in a wide range of prior efforts [148–155]. In this chapter, I focus on the system-level integration of NV-motes and, at the single node level, consider optimizations on the execution of system work sequences to better match the FIOS paradigm rather than the previous NOS paradigm – thereby, making better use of the features offered by an NV-motes nonvolatile components. Specifically, traditional programing in energy-harvesting WSN deployments tries to limit reliance on local computation. However, recent advances in handling intermittency make computation functionally reliable [23, 126, 127, 151, 156–164] and more efficient [28, 33], while NVRF [4] and backscatter [165, 166] techniques significantly reduces the portion of energy contributed by the RF module. These shifts can be exploited by altering the work sequences performed during the active periods and by increasing the amount of local processing performed, such as tasks off-loaded from cloud, data compression and merging, in order to limit the communication costs. An effective node-level design for NV-motes should follow the rules of clearly splitting the computation energy source and policy to improve the efficiency of the computation.

Take an actually deployed "Bridge Health Monitor" system in Table 6.1 for example, a naive sensing-computing-transmission bridge cable node samples and transmits raw acceleration, etc. data of about 300-500MB per day. Then, these data are analyzed in-cloud, performing noise removal, FFT, etc. to produce strength models in order to monitor the strength of bridge cable, which can be calculated through harmonic and vibration. Each step is called a "task" in the paper, and all bridge cable nodes run homogeneous tasks in the actual scenario.

To improve the efficiency of the computation, we can shift computation from the cloud to the fog that includes combination of 3-direction acceleration into one cable-vertical direction vibration, noise removal, FFT, strength calculation in three different bridge structure-specialized models, temperature and humidity noise removal, temperature and humidity compensation of each model's results, and calculation of the average strengths and data compression. Because the processed data are only the strength data with less variation than original acceleration etc. raw data, the compression has a good compression ratio. So the local-computing can dominate the computing time and energy rather than compression.

Figure 6.4 shows differences in the sequences of tasks performed in a completely volatile node, a node with a non-volatile processor, and a NEOFog NV-mote. The volatile node (VP) and nonvolatile node operate as NOS whose supporting front-end circuits are designed with a single super capacitor for energy storage, as shown in Figure 6.5(a). The system is active only when there exists sufficient stored energy to perform all the tasks in the sequence at the top of Figure 6.4. This includes using the processor to initialize the volatile RF using software. In the VP, this takes about 15-100ms and building the connection requires 30ms-1s before data can be transmitted. Replacing a VP with NVP can bring 2.2X-5X forward progress benefits [29]. The NOS NVP's startup time is much shorter, 32us, compared to VP. Due to the direct restore of the RF states with the help of NVP, the data transmission time reduces to 33ms. However, the NVP benefits were not maximized in this wait-compute scheme [29].

To support the FIOS operation, the front-end circuits need to be enhanced, as shown in Figure 6.5(b). More specifically, adding SW1 allows a direct source-to-load unstable power channel to the NVP. These front-end circuit design concepts were originally proposed by Wang et al. [167], improving the front-end efficiency to 90%, and further developed by Sheng et al. [168] taking into account node-level considerations. Further investigations along similar directions include optimizations for rectifier [169] and mix-source design [170]. Our work leverages, but does not substantially advance, the front-end design literature. However, we do shift task sequencing and power source dependency to better utilize the direct source-to-load advantages of Figure 6.5(b). To optimize for the FIOS operation, I propose the sequence shown at the bottom of Figure 6.4. Application computations off-loaded from the cloud and other complex local programs are executed in an intermittent fashion (dashed-line boxes), increasing their effective computation efficiency via a leaner front-end conversion ratio, while the other system activities (red-line boxes) are still powered from the capacitor. NVBuffers are designed to sit between the sensors and the NVP, as well as within the NVRF, to guarantee reliable asynchronous data transfers. By adopting more computation locally, like local processing and complex data compression, lower data transmission can be expected. By this means, the proposed FIOS clearly divides data sample/transmission and computing, and sufficiently utilizes the high efficiency feature brought by NVP via the support of direct-dual-channel front-end circuits. To the best of my knowledge, this is the first work that addresses system-level rearchitecting to benefit from NVP+NVRF.

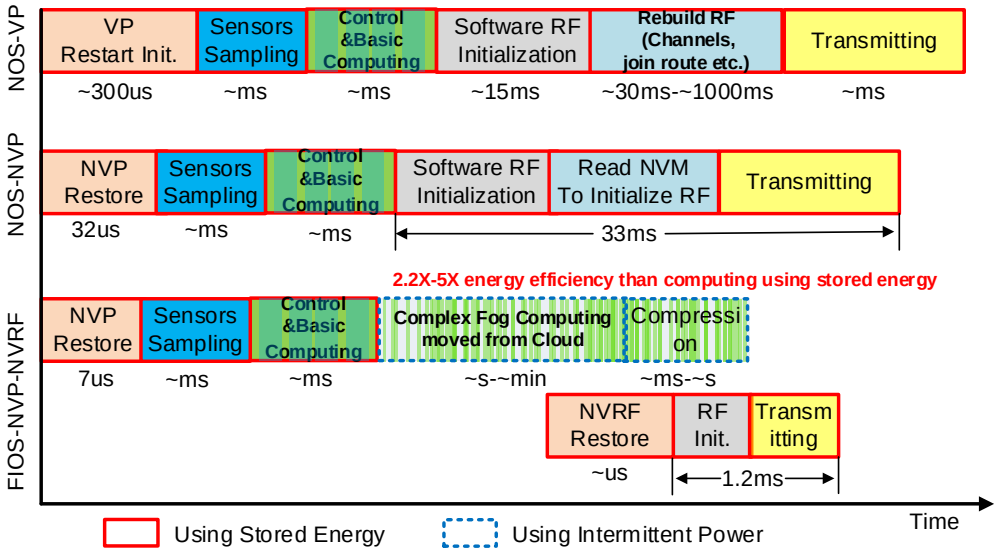


Figure 6.4. Time details of node level.

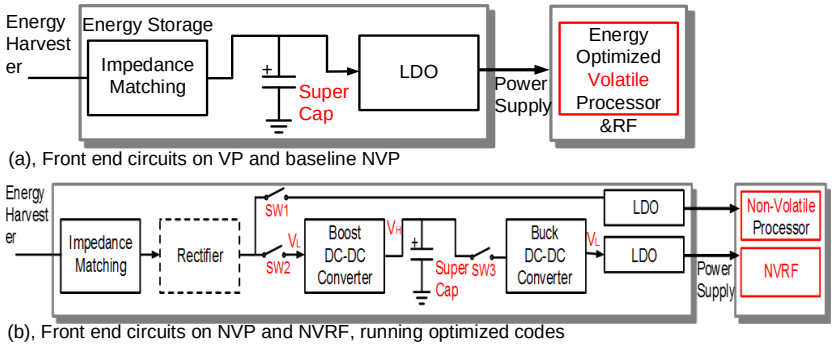


Figure 6.5. Front-end circuits for wireless nodes.

6.2.2 Intra-chain Level - Load Balancing in FIOS

At any given time, there can be substantial variation in both energy income and reserves among the nodes in the network due to variations in the physical deployment and environment, as well as history effects. Local efficiency optimizations that perform frequency and resource scaling in the NVP [28] can further exacerbate these variations. Similarly, any task or resource heterogeneity at the node level (e.g., differences in node-level provisioning across generations in a network with nodes of different deployment ages or data-dependent or position-dependent software execution) would also amplify variations in energy reserve levels over a given time period. Considering the imbalances stemming from both unbalanced load distribution and varying node-level energy-constrained computation capabilities, *I desire to design a network balance method to*

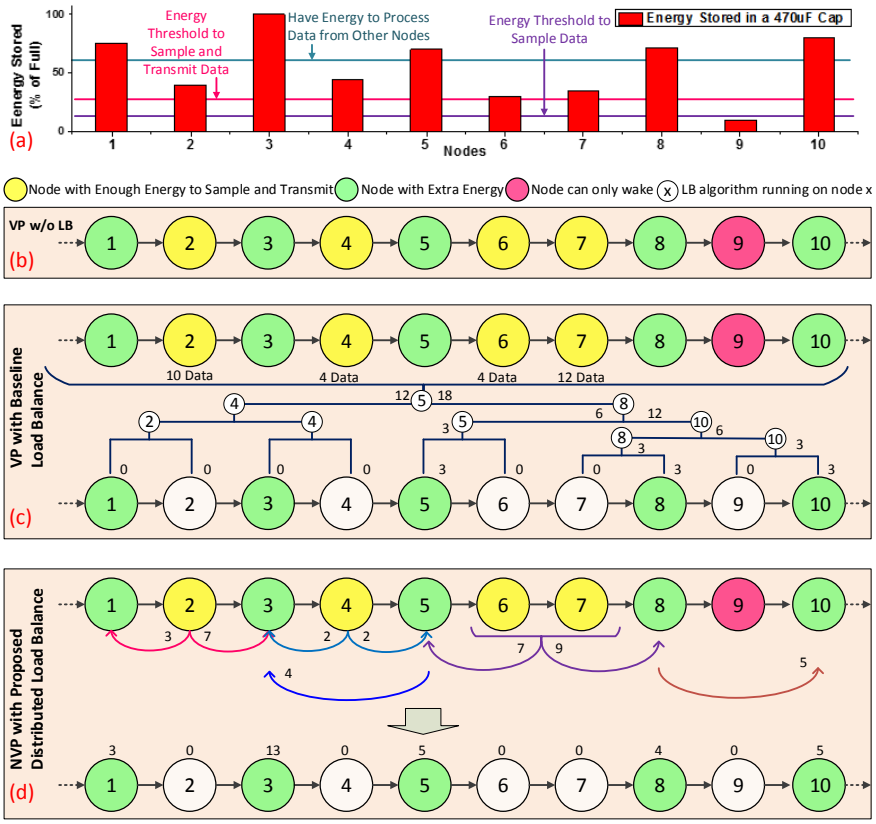


Figure 6.6. Illustration of distributed load balance algorithms

make the nodes that harvest ample energy do more computation than nodes with limited energy, and allocate the tasks to the most efficient rather than inefficient nodes.

Efficiency-Oriented Load Balance for FIOS. An unstable power supply presents challenges for load balancing in an energy harvesting system. Since the nodes may or may not have sufficient energy to start, traditional top-down load-balancing approaches can fail to reach regions of the network. Specific to energy harvesting systems, the computational efficiency varies substantially depending on local energy income, and thus, even when scheduling uniform tasks to nodes with uniform hardware, the dynamic computational efficiency of the target nodes must be an optimization goal for energy harvesting networks (in addition to load balance). Moreover, the capabilities of a given node are time-varying at fine granularities and may shift after scheduling. Distinct from similar scheduling scenarios in data centers, each node generates its own inputs through distributed sensing of raw data and application code is often small enough to be predistributed to all nodes. Due to these different distributed system level requirements, a

new scheduling algorithm and supporting architecture is deployed to fit into the specific needs of FIOS.

Distributed Load Balancing Algorithm. Figure 6.6 demonstrates several approaches to load balancing. Figure 6.6(a) shows the stored energy thresholds that separate the different classes of potential actions a node could take in the next interaction phase: Yellow stands for nodes with enough energy to sample and transmit data; Green stands for nodes with extra energy beyond yellow ones; Red nodes do not have sufficient energy to communicate. Nodes with stored energy level lower than "energy_threshold_to_sample_data" are effectively dead for the current sampling period, so they are not shown in Figure 6.6(b). Absent load balancing, efficiency is very low, as can be observed in Figure 6.6(b) – the available energy and energy required are unbalanced. Volatile nodes with baseline up-down multi-level tree load balance may not fully alleviate energy imbalances: Figure 6.6(c) shows an up-down binary scheduling that is only partly achieved (left 12 tasks are all missed) when the assigned node 4 running parts of the load balance is low on stored energy.

I propose a distributed load balancing algorithm tailored for energy harvesting scenarios. Based on available energy at each node, a node determines how much extra energy it has available for any tasks beyond what it expects processing its own raw data will require. The available energy as well as NVP configuration (frequency and resource state for the Spendthrift policy [28]) are shared with other nearby nodes in the local network chain. For example, node 4 can know states of its left node 3 before touching another energy hungry node 2, and node 5 on the right in the first round. Based on the energy available to left or right nodes, node 2 estimates the execution time of its tasks to be distributed, and builds an array $\langle a_1, a_2, \ldots, a_n \rangle$ and array $\langle b_1, b_2, \ldots, b_n \rangle$ to stand for shortest time running the same tasks on either best efficiency nodes on the left side or right individually. Then, Algorithm 1 is called to compute an "assignment result". Based on the result, for instance, two tasks from node 4 are assigned to node 3, and another two to node 5. A second call to Algorithm 1 may happen when the assigned tasks require more energy than one node has already stored or beyond $MAXTIME$ - load balance call interval. In the example, node 8 is over assigned, a second call distributes 5 tasks to node 10. While, in the worst-case, several distribution rounds may be required to achieve a balanced load and optimality is not guaranteed, this distributed bottom-up algorithm is expected, in practice, to produce fewer, and more local, data transmissions.

Note that if load balance algorithm is interrupted, no load balance will take place at that region. This failure affects performance, but not functionality of the network, and is modeled in the simulation framework. After balancing, the data transmission begins, tasks are computed, and results are transmitted at during the next power-on period.

Node Implementation of Load Balancing.

Let us define $OPT(i,k)$ as the first k tasks that can be finished within time i by the most efficient node on the left and within time $OPT(i,k)$ on the right. The specific task indexed as k can be finished by either left or right. If the k_{th} task is finished by left, then the right's total time does not change; thus, we have:

$$OPT(i,k) = OPT(i - left[k], k-1) \tag{6.1}$$

On the other hand, if the k_{th} task is finished by right, then the left's total time does not change; thus, we have:

$$OPT(i,k) = OPT(i,k-1) + right[k] \tag{6.2}$$

By combining both the situations, we have:

$$\begin{aligned} OPT(i,k) = min(&OPT(i - left[k], k-1), \\ &OPT(i,k-1) + right[k]) \end{aligned} \tag{6.3}$$

Algorithm 1 is implemented as an interrupt-driven program in the node software. It is a dynamic-programing approach with three steps: build the table, find the minimum time required to finish all the tasks, and generate the assignment. The algorithmic complexity of the approach is $O(n * MAXTIME)$, which is task number*load balance call interval.

6.2.3 Inter-chain Level - Node Virtualization for Enhanced QoS

Naively increasing node count and density will not improve QoS. Figure 6.7 shows one example of zigbee protocol. When there are only 10 nodes (Node 11, 21, ... , 101) in the system, the network topology is as the red line. When the node density in the area increases 4x, naive zigbee protocol opts for locality in transmission distance and

Algorithm 1: DISTRIBUTED LOAD BALANCING

Input: Time cost if n task running on the nodes on the left $\langle a_1, a_2, \ldots, a_n \rangle$; Time cost if n task running on the nodes on the right $\langle b_1, b_2, \ldots, b_n \rangle$; $MAXTIME$: load balance call interval;

Output: Out $\langle o_1, o_2, \ldots, o_n \rangle$: task assignment to either nodes on the left or right

```
1  n ← Sizeof(a)
2  p ← Zeros(MAXTIME, n)
3  sa ← 0
4  #build the table
5  for k = 1 → n do
6      sa+ = a[k]
7      for i = 1 → sa do
8          p[i,k] = p[i,k−1] + b[k]
9          if i ≥ a[k] then
10             if p[i,k] < p[i−a[k],k−1] then
11                 p[i,k] = p[i,k]
12             else
13                 p[i,k] = p[i−a[k],k−1]
14 #find the minimum time
15 minTime ← ∞
16 for i = 1 → sa do
17     if p[i,k] ≥ i then
18         temp = p[i,k]
19     else
20         temp = i
21     if minTime > temp then
22         minTime = temp
23         AtimeFinal = i
24         BtimeFinal = p[i,k]
25 #generate the assignment output
26 i ← AtimeFinal
27 for k = n → 1 do
28     if i ≥ a[k] then
29         if p[i,k−1] + b[k] < p[i−a[k],k−1] then
30             Out[k] = "right"
31         else
32             Out[k] = "left"
33     else
34         Out[k] = "right"
35 return Out
```

increases the number of jumps from 9 to 25 when transmitting data from node 11 to 101.

In order to improve the system performance while using the existing RF protocol

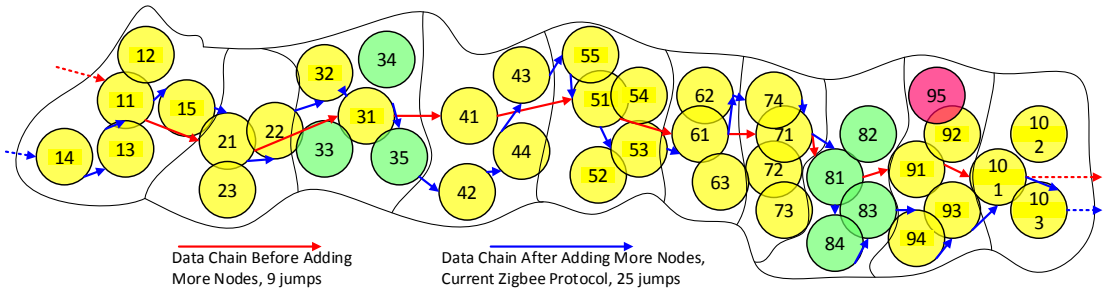


Figure 6.7. Naive density increase does not boost Zigbee QoS

(zigbee in this work), I propose a slotted time-multiplexing node virtualization for QoS algorithm (NVD4Q), shown in Algorithm 2, that leverages the capabilities of the NVRF to support new behaviors. When a new node is added to an existing network, it will first open its NVRF to search for the closest node and then clone that node's NVRF state, as shown in Figure 6.3(b). It will then synchronize its timer with the network. However, rather than waking up every tick of the RTC timer, it will receive an initial (phase) offset in ticks (unique among the clones of the same node) and a pre-set tick count between activations (common among all the clones) which are only updated when requested by software. Software can thereby manage the effective sampling frequency of the collection of the cloned nodes in order to match the sampling frequency needed by the particular application for the single logical node that the cloned nodes implement via time-multiplexing based on the number of clones for a given node. Membership to a set of clones is updated at a programmer-defined frequency specific to both the application and deployment scenario. For instance, a bridge monitor system may have nearly permanent clone set memberships, while a mountain sliding monitoring system may update at a low frequency and sensor nodes on moving objects would frequently request network reconstruction, including re-association of clones.

With this NVD4Q, we expect to achieve the behaviors shown in Figure 6.8. Collectively, these behaviors are expected to increase the number of samples processed by each logical node. These include: At each wake-up period, only nodes with a common phase (10 in the example) wake up. Nodes in chain 1 to 5 wake up consecutively, but chain 6's nodes are totally different from chains 1 to 5. From the network's perspective, the network structure and information does not change during power off period; so, because of the NVRF, there is no reconstruction penalty required for the network. Note that each node continuously accumulates energy in its own super capacitor, and as a re-

Algorithm 2: NODE VIRTUALIZATION FOR QOS ALGORITHM

```
1  Always open the NVRF
2  Find the closest node through NVRF
3  Copy its states of NVFF in NVRF controller and NVM
4  Synchronize the timer, then turn off NVRF
5  while Timer reaches a pre-set time do
6      Update or not update wake-up interval time
7      Start NVRF to send or receive data
8      if Received special command
9      then
10         Go to line 2
11     Shut down NVRF w/o backup of NVRF control states
```

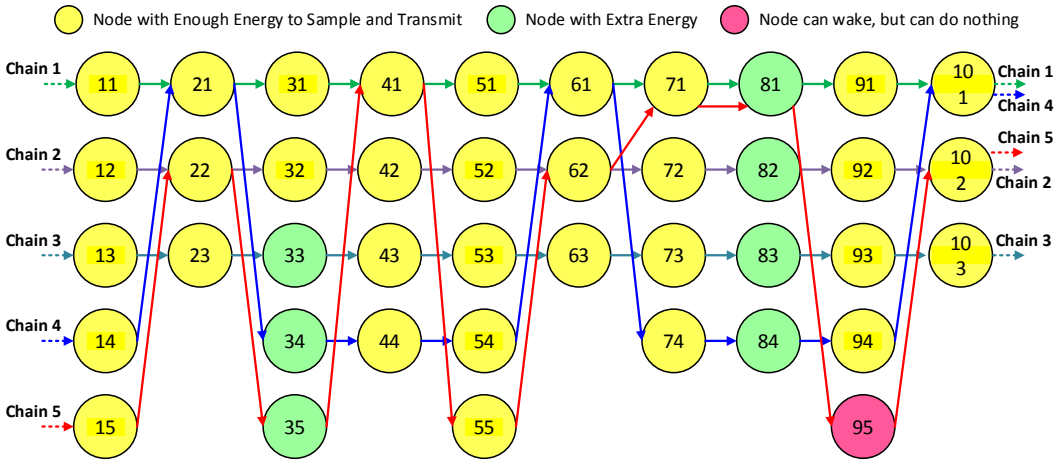


Figure 6.8. NVD4Q Node virtualization for QoS expected effects

sult, chances that one node runs out of energy become lower. Each node in a collection of clones activates less frequently than a baseline node (by a factor equal to the number of clones), and hence, the energy to be dissipated is smaller.

6.3 Simulation Methodology

I built and measured real WSN prototype platforms to quantify the energy distribution of energy harvesting and NVP-based WSN systems. Each node consists of an NVP running at 1MHz, consuming 0.209mW, sensors specific to different applications, and a Zigbee based RF module with a data rate of 250kbps, consuming an average of 89.1mW when transmitting. The power varies depending on the configuration of the TX strength, ranging from 72mW to 102mW. The RF module, when working in the RX mode, typi-

cally consumes 72mW.

Our simulation framework consists of two parts. The first part is a node-level functional simulator, at the core of which is a modified 8051 RTL [32], which has been calibrated with physically fabricated nodes [23]. The node-level functional simulator also includes two parts, one matlab/python part starts the nodes' simulation, calculating the input power, energy, stored energy, efficiency etc. and calls the Modelsim Verilog part (step by cycle) to run function simulation. The node-level simulation is cycle-accurate, and our modeling of NVRF is calibrated against a fabricated device [4]. Capacitor size selection and according policy are modeled in work [33]. Power and stored energy sampling supporting circuits (including ADC's power) and penalty are also modeled [28] with more features in sensors such as accelerometer LIS331DLH, image sensor LUPA1399, temperature sensor TMP101, etc. For example, for TMP101, the initialization costs 566ms, and one time sampling costs 0.283ms. For Zigbee chip with volatile RF and traditional method, ML7266 initialization costs 531ms (Host MCU@1MHz), and data transmission of N Bytes costs $(255+1.44*N+0.032*N)$ms, while zigbee chip ML7266 with NVRF module only costs 28ms as initialization and $(1.74(NVRF\ start)+0.156+0.216*N+0.032*N)$ms as data transmission. The measured node was built with the FIOS mode, not directly replacing WISP with NVP.

The second part of our framework is a WSN system-level simulator. It starts thousands of node simulators at a time, and provides WSN system-level status. The front-end circuit efficiencies, stored energy level, etc. at system level are modeled in previous work [29]. The time running one specific program is simulated in Verilog to determine clock cycle counts. RF energy is computed as the integration of power over that specific time period considering active, idle, and OFF modalities (89.1mW at TX and RX, 14.93mW at IDLE).

Various failures are modeled in our simulation framework. First, at a node level, two kinds of failures are modeled: node failures caused by depletion of energy, and packet failures caused by wireless transmission affected by weather and channel interference. Transmission success parameters are from an experiment with constant-light powered 3-mote point-hop-router transmission (A$\rightarrow$B$\rightarrow$C) mounted on top of a building, with a distance of 10-15m and 10 days consecutive data collection. Packet loss rate 0.75% was observed (totally 14400 A$\rightarrow$C packets were expected), mainly affected by weather, especially rain. Packet transmission success rate between two sufficiently powered nodes

was therefore modeled as 99.25%.

Second, at intra-chain network level, for a 3-mote transmission example (A→B→C), when B fails to start due to energy shortage, orphan_scan function in Zigbee stack is called in A to broadcast, C sends unicast to A to confirm ("Scan_confirm" status is success), following with an update of "AssociatedDevList". So, A→C. When B recovers, B broadcasts, A adds B in its "AssociatedDevList" and removes C, C join B, and finally A→B→C. At inter-chain level, the "AssociatedDevList" as well as other info are duplicated within virtual nodes to avoid rebuilding the network structure. "RSSI" which exists in every data packet, is the signal strength, which is used to find the closest neighbors. All these latencies and energy are measured in real ML7266 NVRF based nodes, and are then modeled.

To test the effectiveness of the whole simulation framework, the deployed "bridge health monitor system" and "wearable UV meter system" in Table 6.1 are used to validate the baseline NVP simulations. Other systems with different sensors and functions are also built, whose measured results are presented in Table 6.2.

Due to node count limitations in currently deployed systems using fabricated NVPs and NVRF chips, as well as the fact that the proposed load balancing and virtualization for QoS (NVD4Q) policies need some architectural-level support that is not yet present in the fabricated devices, the load balance and the NVD4Q results are simulated. Our simulator runs thousands of single-node simulators simultaneously (1000 for intra-chain simulation, and 1000 to 5000 for inter-chain simulation). Each node has different power inputs. The communication is mimicked by direct data transmission under a certain successful transmission possibility through virtual buffers among nodes. Of the simulated thousands of nodes, 10 consecutive nodes' information is shown as the presented example in the paper for simplicity.

6.4 Results and Discussion

In this section, I evaluate the impact of the individual schemes on WSN quality and then quantify the contributions due to individual techniques employed. Further, I combine them together and evaluate under both dependent (time and location correlated) and independent power trace scenarios. To quantify WSN output quality, I employ the following metrics: counts of node wakeups, successfully processed samples, and sam-

ples processed in the fog. For total data processed in the fog, the data compression, decompression and other post-optimized operations not present in the baseline system are not included. Only tasks offloaded from the cloud are considered as data processing (i.e., work that would otherwise have been done at the cloud).

6.4.1 Single Node Energy Distribution

Method	*Naive sensing-computing-transmission*				*Sensing-buffering-complex local computing-compression-transmission*			*Comparison*
App.	**Inst. NO.**	**Compute energy(nJ)**	**TX energy(nJ)**	**Compute ratio**	**Compute energy(mJ)**	**TX energy(mJ)**	**Comp. ratio**	**Energy saved**
Bridge Health	545	1366.86	22809.6	5.65%	81.7	6.95	92.2%	-55.2%
UV Meter	460	1153.68	5702.4	16.8%	108.3	6.8	94.1%	-48.8%
WSN-Temp.	56	140.448	5702.4	2.4%	75	6.99	91.5%	-57.1%
WSN-Accel.	477	1196.316	17107.2	6.53%	83.6	6.59	92.7%	-54.9%
Pattern Matching	1670	4188.36	2851.2	59.5%	345.1	5.39	98.5%	-24.1%

Table 6.2. Measured energy distribution on different platforms using two different strategies. Parameters in the table stand for parameters and energy during two-time data transmission interval.

Table 6.2 shows the measured parameters of each system. Five diverse deployments are examined, including bridge health monitoring, UV meter (data capturing only), temperature, acceleration sensing, and heartbeat signal pattern matching. Two strategies are deployed: naive sensing-computing-transmission and sensing-buffering-computing-compression-transmission. In the former strategy, the single node starts itself to sense the data, which are normally of relatively small size. Since the computation involves mostly sensing and simple data processing, the instruction counts in Table 6.2 are also light. After processing the captured data, the node performs the transmission sequence, including starting the RF module, initialization, and channel setup. The computation ratio indicates the percentage of NVP energy consumption among all node components. I observe that, for some applications, computation does not play an essential part in energy consumption as the transmission energy dominates. For some other applications; however, especially in the pattern matching, the NVP computation is substantial, consuming up to 59.5% of node energy.

The other approach considered employs sensing-buffering-computing-compression-transmission, except when there is a real-time request from a control node. Instead of immediately processing and transmitting the captured data, it is buffered in a 64kB nonvolatile memory buffer designed to support this policy, as shown in Figure 5.1(B).

Once the buffer is full, it triggers an interrupt of the NVP to process the buffered data. If the node lacks energy to process or send the buffered data out, the sampled data are discarded. "Incidental Computing" techniques [29] have been proposed to mitigate this. Two observations are noticed. Firstly, I find that, for some applications, more complex data processing requires data from neighboring samples in time – e.g., the pattern matching in heartbeat monitoring. Thus, buffering and operating on the local sequence as a batch reduces data to be transmitted. Secondly, compression (bzip or jpeg depending on application) before data transmission can reduce the data size to $3\% - 14.5\%$ of its original. The many repeated patterns in data, especially in that sensed by WSNs, foster high data compression rates. However, compression is not free; it requires a large amount of computation energy as shown in Table 6.2. To compute the total energy savings, I use the following formulas:

$$E_{naive} = (E_{naive}^{comp} + E_{naive}^{trans}) * 64k \tag{6.4}$$

$$E_{new} = E_{new}^{comp} + E_{new}^{trans} \tag{6.5}$$

$$Energy\ saved\ ratio = (E_{new} - E_{naive})/E_{naive} \tag{6.6}$$

Note that the naive method repeatedly samples a single data item and sends it out, while the new method samples and buffers until 64k data, then processes them together. Although compression costs energy, total energy is saved through less energy spent in RF initialization and smaller data size to transmit. Through the optimization of the buffered strategy, we observe that computation energy dominates across applications, ranging from 91.5% to 98.5%. Compared to the naive strategy, the buffered strategy can save $24.1\% - 57.1\%$ total energy. In the actual deployed systems, physical buffers are present, but both strategies are utilized to balance between energy savings and fast response times, although the buffered strategy dominates running times.

6.4.2 Performance of Distributed Load Balancing

Figure 6.9 shows the stored energy level of three consecutive nodes in a chain when powered with a solar panel in the daytime. As can be seen in Figure 6.9(top), the stored energy level of VP node 1 without load balancing during the 0-50 min period indicates that the capacitor was frequently full, meaning further energy was rejected by the sys-

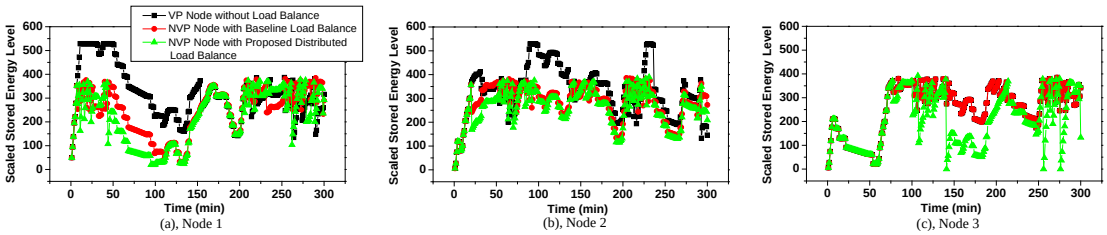


Figure 6.9. Stored energy level of 3 consecutive nodes in a chain

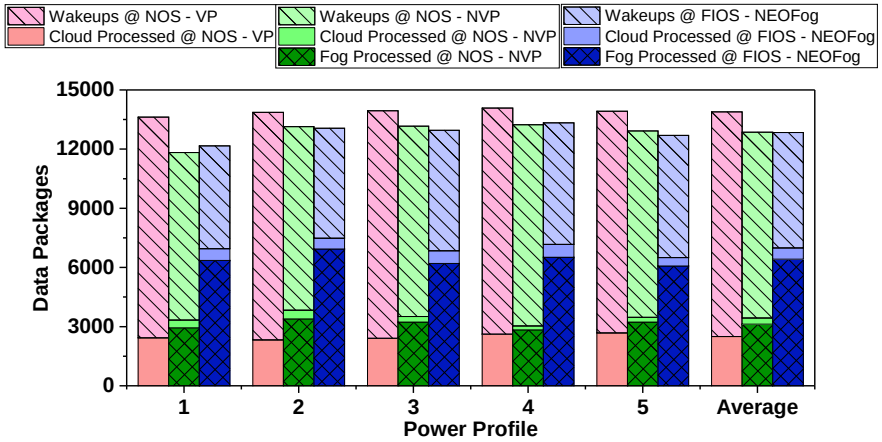


Figure 6.10. Total data packages captured by the nodes in the network, and total data packages processed by the fog, when the power traces are ample and independent

tem. Using an NVP node with baseline load balancing reduces the stored energy by balancing the loads among nodes. Employing the proposed distributed load balance further reduces the stored energy level. A similar situation can also be observed for Nodes 2 and 3, indicating that both load balancing approaches succeed in avoiding the capacitor overflow scenarios seen without load balancing by moving work to neighboring nodes.

Performance with Independent Power Income

Since the load balance is intra-chain, the power income level among nodes plays an important role for the intra-chain load balance. The more stored energy variation among the nodes, the more impact from the proposed algorithm. I use solar power traces from a forest to mimic deployment scenarios. With winds, depending on whether the leaves are moving, the power variation among nodes is large, and effectively independent.

I consider a forest fire monitoring system. Such a system may have as many as 1k to 1M nodes, but I present data for only the 10 nodes of one chain. Given uninterrupted power and no failures, these 10 nodes could ideally deliver 15000 data packages in 5

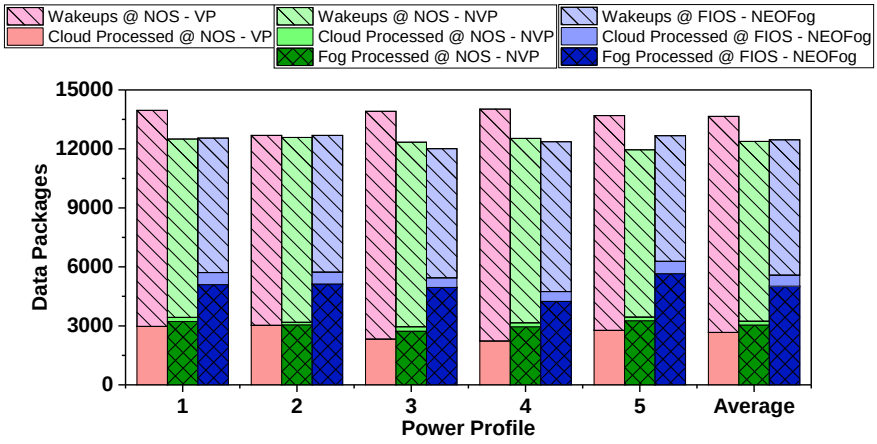


Figure 6.11. Total data packages captured by the nodes in the network, and total data packages processed by the fog, when the power traces are ample and dependent

hours. Each power trace is a synthetic trace generated from data collected by a single energy-harvesting node in both tree-covered and open environments under different time and weather conditions. Individual node power incomes are generated by concatenating sequences from the measured traces in random order. The computation performed when performing in-fog offload is a reconstruction kernel for a volumetric map based on point samples.

Figure 6.10 shows the simulated number of wakeups, total processed data packets, and in-fog processed packets for five different power traces over three different systems. The VP nodes do not perform fog-offloading, and without load balancing, despite 13656 node wakeups (1344 node failures due to energy depletions), only capture and transmit 2664 packets. With a higher activation threshold, NVP nodes running the baseline load balance only exhibit 12383 wakeups, but improve the total packets processed to 3236 and increase in-fog processing to 3045. NEOFog wakeups are similar to the baseline NVP, but substantially improve the total packets processed to 5582, 37% of the ideal of 15000, and increase in-fog processing to 5018 packets.

Performance with Dependent Power Income

To explore situations where the power observed by each node is strongly correlated with its neighbors, I consider the power observed from daytime solar traces at fixed locations on bridges in a bridge monitoring system. Each power trace is a synthetic trace generated from the same 5-hour period from 5 different days of measured bridge

data [45]. Individual node power incomes are generated by applying 30% random variance to the measured data traces. The computation performed when performing in-fog offload is based on the structural health monitoring algorithms in [171, 172]. Again, under an ideal power and communication scenario, the maximum number of processed packets would be 15000.

Figure 6.11 shows the simulated wakeups, processed packets, and in-fog processed packets for bridge monitoring under a set of highly dependent power traces. In a system with dependent power profiles, stored energies exhibit lower variance, and thus the load balance scheme is activated less frequently. The amount of energy required to transmit data from one node to another decreases, partially compensating for reductions in actual load-balancing. This is why the dependent values are within 10% of those for the independent power traces.

For the dependent power traces in the bridge monitoring scenario, similar to the prior forest scenario, the VP nodes do not perform fog offloading, and without load balancing, despite still performing 13886 node wakeups, only captures and transmits 2494 packets. With a higher activation threshold, NVP nodes running the baseline load balance again exhibits reduced wakeup counts of around 12859, but improves the total packets processed to 3439 and increases in-fog processing to 3126. NEOFog wakeups remain similar to the baseline NVP, but substantially improve the total packets processed to 6990, 46.6% of the ideal of 15000, and increases in-fog processing to 6418 packets.

If we compare the NEOFog system with nodes running distributed load balance to VP nodes without load balance and NVP nodes with baseline load balance, the average gains of 2.8X and 2.0X gains are seen in the total network output, respectively, for average gains for a high variance, independent deployment and 2.1X and 1.7X gains are seen for a dependent power scenario. While this indicates that the proposed load balance is less effective in dependent power conditions, it still vastly outperforms the baseline and sans-balancing systems.

6.4.3 NVD4Q - Virtualization for QoS Results

I consider a mountain sliding monitoring system as the running example for NV4DQ, in which solar-powered nodes are randomly distributed in the area to be monitored via aerial dispersion. Some of them will have excellent sun exposure, while others may

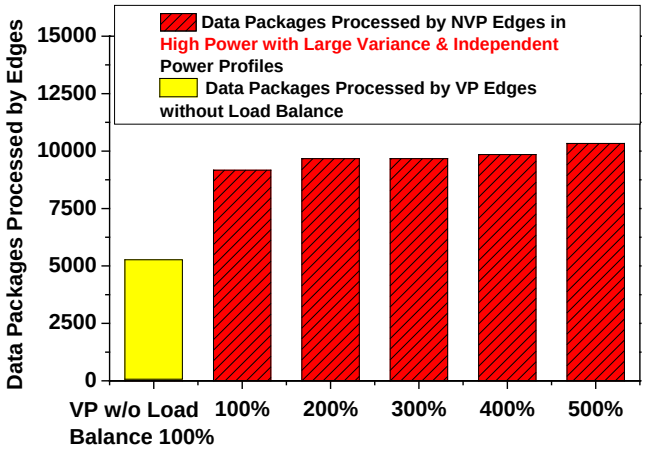


Figure 6.12. Increasing node multiplexing in an environment with high power with large independent variance

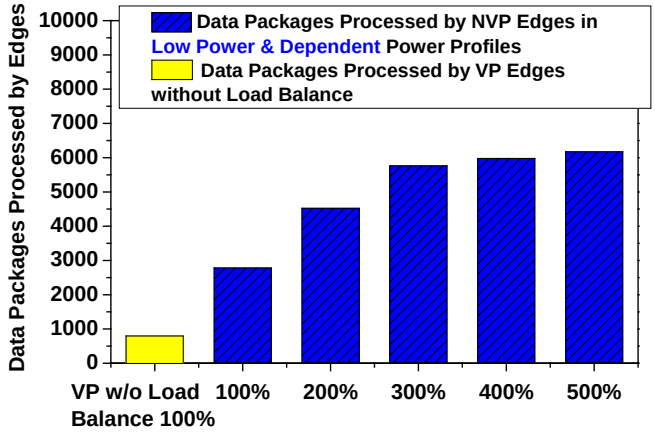


Figure 6.13. Increasing node multiplexing in an environment with very low power and dependent variation

fall into grass or shrubs or be poorly oriented. On a sunny day (which can be regarded as high power with large variance), the network collects around 12000 samples, and a traditional VP system without load balancing can process about 5000 data packages in-fog, as shown in Figure 6.12. The NVP nodes with the proposed load balancing can process around 9500 data packages, almost double the baseline. Since the in-fog processing rate is already very high, the NV4DQ approach provides minimal gains.

However, the highest concern for the events the system is designed to monitor, slides, occurs during heavy rains when solar energy is very limited. I show the results for operation during inclement weather in Figure 6.13. A traditional VP system without load balancing can process only 725 data packages in-fog, while the NEOFog system

with identical node count can process 2800 packages in-fog. Increasing the node count and multiplexing rate when using NV4DQ substantially improves the in-fog processing rate, up until 300% multiplexing is reached, by allowing longer times for each node to accumulate the less-available energy and from less frequent startup overheads for each node. Benefits saturate at 3X multiplexing for this experiment, because total successful sampling under the reduced power conditions reduces to 8000.

6.5 Summary

The maturation of NVPs and other integrated nonvolatile elements brings new optimizing opportunities at system level design as prior assumptions regarding the key tradeoffs and design principles for energy-harvesting systems are challenged. To address this, in this chapter, I revisit the core operating paradigm of normally-off systems, and instead, reoptimize for the frequently-intermittently-on systems enabled by NVPs and NVRFs. I show that integrating nonvolatility into nodes can improve the amount of computation offloaded to the fog, rather than performed in the cloud, and that applying NVP-specific distributed load balancing can further increase fog computing capability. I also provide a node virtualization technique that exploits NVRF advantages to increase fog computing QoS in lower power-income conditions. Collectively, these optimizations increase fog-offload capabilities by 4.2X at baseline deployment node count and up to 8X at 3X multiplexing.

Chapter 7

Related work

7.1 Nonvolatility in Processors

By designing distributed nonvolatile logic or elements at a microarchitectural level, computation state can be checkpointed before power outages occur with only on-chip energy storage and without programmer intervention [97, 98, 173–175]. Various materials can implement the nonvolatile features, for example, FeRAM based NVPs [92, 93, 118], ReRAM based NVPs [32], and MRAM (magnetoresistive random access memory) based NVPs [94]. Beyond these, there are other types of NVPs [96, 135, 176], and NV associative processors [177] as well. NVP efficiency has been improved by frequency scaling [33] and dynamic resource control [28] for NVPs. Many cross-layer works leveraging integrated non-volatility to address intermittency have also been explored, including, OS and high-level synthesis approaches [160, 161], programming language and compiler approaches [126, 149], HW/SW approaches [155], and software-based approaches [162–164]. While this dissertation explores a wider domain of NVP and then relies on the existence of NVPs, it focuses on how to benefit from their use at system level.

7.2 Architectural Aspects of Energy Harvesting

Computing under unreliable power supply conditions leads to several interesting architecture and system-level issues, many of which have been dealt with in this disserta-

tion. B. Lucia *et al.* [178] have explored the possibility of concurrent programming under intermittent energy and the various efforts required to maintain program consistency. These issues are addressed by means of atomic instructions allied with an on-chip capacitance to ensure that the processor has sufficient power to complete the ongoing instruction. Jayakumar, H *et al.* [179] use an FeRAM for quickly checkpointing the system state in case of power loss in transiently powered computers. My work also explores in detail various micro-architectures by varying the power-on threshold, thus being able to optimally run for a whole range of application complexities. In reference [180], the authors propose a power-management technique for a solar-powered multicore architecture. My dissertation, on the other hand, extends our analysis to different energy sources with a detailed micro-architectural evaluation.

7.3 Frequency and Resource Scaling

Noting the trade-offs among different NVP microarchitecures, a machine learning based methodology is proposed by Ma, *et al.* [68] to dynamically switch between three distinct design points. To the best of my knowledge, there exist no works that attempt to predict bottleneck resources and DFS for NVP IoT platforms.

DFS is a traditional technique targeting either boosted performance or reduced energy [62–65] Distinct from these works, I apply DFS to NVP for energy harvesting scenarios to adjust to variations in power-income rather than variations in the workload or thermal headroom. Traditional methods use lookup tables for DFS, while in more complex energy harvesting scenarios with unstable power, I apply machine learning to handle it.

Kontorinis, *et al.* [66] propose table-driven resource allocation in a configurable datapath to conservatively enforce peak power reductions with minimal performance degradation. I employ a similar scheme in this work to tune EPI to the current power income, but are not constrained to conservative solutions and utilize neural networks rather than tables to predict the bottleneck resources for unstable power incomes.

7.4 Active Checkpointing

Energy harvesting is attractive for IoT. To support operation under an intermittent power supply, many works across the computation hierarchy have been explored, including OS and high-level synthesis approaches [160, 161], programming language and compiler approaches [149, 181], HW/ SW approaches [179], and software based approaches [162–164]. I classify these approaches as active checkpointing [182], in which the programmer needs to identify the essential data and software invokes a backup before energy runs out.

7.5 Passive Checkpointing

Another alternative is passive checkpointing. By designing distributed or centralized nonvolatile logic or storage elements managed at a microarchitectural level, computation state can be checkpointed *transparently to software* before power outages using stored energy. This is called a nonvolatile processor [23, 28, 31, 97, 98, 173–175, 183–185]. Various materials and circuits with nonvolatility can be adopted to design different types of NVPs, e.g., FeRAM-based NVPs [93, 118], ReRAM-based NVPs [32], and MRAM-based NVPs [91, 94].

Many of the active checkpointing works [25, 48, 149, 162–164] are validated on a TI MSP430 with on-chip FeRAM, which can be considered an embryonic form of NVP, with software-based methods for active check-pointing. The different approaches for achieving continuous computation under unstable power supply have differing tradeoffs. The active method is modest in cost, but it is bounded by the backup speed and energy. Passive checkpointing can save system initialization time and energy when powered up, but is difficult to design and can potentially impose operational overheads.

7.6 Approximate Computing

There is a substantial body of work focusing on approximate computing in the general purpose computing domain. A statistical guarantee method in controlling quality has recently been proposed for an approximate accelerator [186]. Quality detection and error correction by exact recomputing on host processor is proposed by Khudia *et*

al. [187]. A pipeline-parallel approach for producing progressively higher quality output across multi-kernel execution chains via iterative recomputation is described by Miguel *et al.* [188]. Tang *et al.* improve bank-level parallelism for irregular applications [189]. A self-tuning approximation with quality feedback control for graphics engines is proposed by Samadi *et al.* [190]. Hardware support for approximate operations includes voltage scaling and speculation [191] and multi-voltage setups [107].

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

7.7 Approximate Storage

Another form of approximation is approximate storage [192] . Approximation in DRAM based on data criticality is explored and optimized [193]. Approximate storage in solid-state memories is employed by Sampson *et al.* [194] and load-value approximation is developed by Miguel *et al.* [195]. Approximation is often a system-level approach, requiring support at multiple layers, cross-layer optimization and co-design. Recent work [196, 197] continue to examine compiler and programming level support for approximation, and a pure software based solution [198] targeting GPU approximation has been explored. An architecture using signal significance to vary approximation levels in an inter-frame motion estimator is presented [199]. Code acceleration with limited-precision analog computation is developed [200]. Configurable trade-offs between precision and energy are explored [201]. A "Rely" programming model for

verifying unreliable hardware is developed [202], but random power failures are not modeled. Approximation in energy-harvesting is explored in software by Sampson *et al.* [203], but not targeted on nonvolatile processors.

7.8 Nonvolatility in RF Control and Other Low-power RF Technologies.

Nonvolatile RF control was first proposed by Wang *et al.* [4], and it boosts the startup speed of a commercial Zigbee module by 27X. Other point-to-point techniques like backscatter [165, 166] are also promising techniques to reduce RF energy in nodes, and there are many protocol implementations that aim to implement lower-power RF communication [204–207]. Any of these protocols would reap further benefits from the NVRF approach. In this dissertation, I emphasize how to develop new network-level strategies that exploit the ability to clone NVRF states and its superior latency and throughput for cheap node virtualization via time-multiplexing without changing the network protocols.

7.9 Load Balancing on WSN

Load balancing on WSNs, aims to match the work to be done with the nodes that have sufficient energy to perform it. Many prior load balance methods have been proposed, including weight-based approaches [208, 209], package forwarding [141], pseudo-sink protocol [142], and dynamic route [143] balancing. Some works use partitioned clusters for load balance [144–146]. A decentralized routing algorithm, known as a game theoretic energy balance routing protocol is proposed by Abd *et al.* [210]. All these approaches target battery-powered devices rather than energy-harvesting nodes with highly unstable power supply. The proposed balancing algorithm is specifically designed for high failure rates, even during the balancing algorithm itself. To this end, it eschews global communication and optimization, and aims instead to reduce the scope of the shared information and the associated transmission costs.

7.10 Virtualization for Quality of Service

Wajgi *et al.* [147] proposes backup nodes for the cluster head after it triggers a threshold. Other works also propose inter-cluster level optimizations by optimizing the protocol and network topology [139, 140]. In contrast, the proposed NVD4Q policy shares states in NV elements in NVRF, and thus, the (virtual) network topology does not change. Note that NVD4Q is not a load balance at inter-chain level. Rather, it enhances the QoS via each (physical) node having more time to accumulate energy before communicating, enhancing the success rate of each virtual node.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

Technology trends leading to the proliferation of IoT devices operating on harvested energy demand a corresponding revolution of the abilities of processors to adapt to unstable power supplies. This dissertation discusses architectural level designs and optimizations for ambient energy harvesting NVPs and provides the design guidelines mapping from power sources to architecture level selection. Aiming at time and energy optimization, I have proposed and simulated various architecture level solutions for non-pipelined, five-stage-pipeline and out-of-order processor architectures, and have discussed the trade-offs between them over the course of this manuscript. To better optimize the NVP node, bottleneck resource prediction and frequency scaling can both be applied to improve the forward progress. Powering on some resources to mitigate some bottlenecks can actually significantly increase the NVP's parallel density. Through smart DFS, the energy used for forward progress computation increases. I show that the proposed spendthrift solution can achieve 2.08X forward progress than best case of OoO NVP with static configuration and static frequency. Furthermore, adopting approximate computing approaches in NVPs not only improves the forward progress that NVP systems can offer, but it also provides a means for NVPs to optimize for responsiveness as well as overall efficiency, and synergizes with unique features of NVP execution, namely, frequent backup and recovery operations. I introduce the concept of "incidental computing" to address opportunistic responsiveness versus quality tradeoffs under unstable power income, and implement and evaluate an instantiation of the incidental

computing approach based on memory and datapath approximation within an NVP. For a prototypical IoT workload, the combination of all of the techniques improves forward progress by an average of 4.28x over a baseline ”precise” NVP, of which 1.4x is attributable to NVP-specific backup and restore approximations. Realizing that the nonvolatile features applied in NVP and NVRF etc. have significantly impacted the node level simulation, which makes the traditional assumptions that edge node level computing is unreliable and expensive in energy not hold. In order to better utilize the new advantages brought by nonvolatility in edges, I propose NEOFog - a system-level optimization to better take the advantages of reliable and relatively efficient in fog computing. Specifically, I optimize the programs from RF energy dominate to computation intensive, from normally-off system to frequently-intermittent-on system, to better utilize the new opportunities brought by nonvolatility. Beyond nodes level, considering the unbalance and time-varying income power and stored energy level of each node and different the energy requirement for workloads, I optimize a distributed load balance algorithm specially optimized for unstable power supply in an intra-chain level to balance the loads and further increase the computation done by edges. I show that it can boost fog computing from 1.9X to 2.1X compared to the system running a traditional load balance. I also propose architectural supports for increasing nodes density for the better quality of service by time-division multiplexing of NVRF. 2X QoS is observed with 3X density increment in a very low energy income situation.

In conclusion, this dissertation provides a holistic exploration in applying nonvolatility into the processor and system design in energy harvesting application scenarios. Various techniques are proposed for optimization in nodes level and system level. To validate the simulation infrastructure, I compare and calibrate the simulation results against several fabricated non-volatile processors. The proposed architectures and optimizations, in turn, provide guidance for SOC chip and system design, which further proves the feasibility of the proposed architectures. This dissertation is a first holistic exploration of ambient energy harvesting processor and system design.

8.2 Future Opportunities and Directions

Cross-layer support for NVPs

With the emergence of the lower-layers device and circuits level innovations, e.g.,

new nonvolatile devices like FeFET [211], NCFET [212], according circuits, or devices with intrinsic approximate features (e.g. ReRAM), the new architectures and more achievable application scenarios can be further explored.

In the upper layers, opportunities lay in optimizing the OS and high-level synthesis support [160, 161], programming language and compiler support [126, 149], HW/SW co-design support [155], and application/software level support [163, 164] etc. to further optimize the NVPs. At this day, the optimization of upper layers for supports of nonvolatile processors remains only partly explored.

For other system components beyond NVPs, applying nonvolatility into WiFi, Bluetooth, 4G LTE, LoRA, sensors, and other peripherals may reduce the energy consumption of the specific modules through mitigating or even elevating the re-initialization of the peripherals.

Application specific optimizations

In higher application system level, with more nonvolatility integrated into the system, the traditional optimizations in system level need to be rethought. As a first step, I have explored system-level optimization of load balance in energy harvesting wireless sensor networks [27]. Beyond this, plenty of opportunities exist, including new wireless protocols for supporting unstable and unre-transferable networks, block-chain based decentralized networks, implantable medical systems, etc.

Beyond optimizations for lower power in extremely energy hungry energy harvesting scenarios, nonvolatility are also expected to reduce power consumption in high performance computing elements, for instance, applying nonvolatility into CPUs or GPUs used in data centers for utilizing the fast and efficient backups and recoveries to more frequently turn on and off of unused microarchitectures show promising benefits for reducing total power consumption.

Beyond Von Neumann architectures

During the optimizations of designing energy harvesting IoTs' architectures, I found that the data movement including backup recovery energy takes an essential part in the whole energy consumption. The energy cost by data movement between computation and storage, which is normal in Von Neumann architectures, limits further optimization towards lower power. To build extremely power efficient cognitive energy harvesting systems that can understand and interpret complex visual scenes for applications like self-powered contact glasses, architecture that can go beyond the conventional Von Neu-

mann model, in particular, those that can promise better energy efficiencies, are desired. Since nonvolatile memories and logic are already utilized in NVPs for backups, they are also expected for processing in memory with approximate techniques, and/or neuromorphic computing. Exploration of the mapping of vision algorithms etc. onto such fabrics and integrating within sensors is promising as further opportunities.

Bibliography

[1] SWAMINATHAN, K., R. PISOLKAR, C. XU, and V. NARAYANAN (2012) "When to forget: A system-level perspective on STT-RAMs," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, IEEE, pp. 311–316.

[2] JOG, A., A. K. MISHRA, C. XU, Y. XIE, V. NARAYANAN, R. IYER, and C. R. DAS (2012) "Cache revive: architecting volatile STT-RAM caches for enhanced performance in CMPs," in *Proceedings of the 49th Annual Design Automation Conference*, ACM, pp. 243–252.

[3] SMULLEN, C. W., V. MOHAN, A. NIGAM, S. GURUMURTHI, and M. R. STAN (2011) "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, IEEE, pp. 50–61.

[4] WANG, Z., F. SU, Y. WANG, Z. LI, X. LI, R. YOSHIMURA, T. NAIKI, T. TSUWA, T. SAITO, Z. WANG, K. TANIUCHI, M.-F. CHANG, H. YANG, and Y. LIU (2017) "A 130nm FeRAM-Based Parallel Recovery Nonvolatile SOC for Normally-OFF Operations with 3.9 Faster Running Speed and 11 Higher Energy Efficiency Using Fast Power-On Detection and Nonvolatile Radio Controlle," in *Proc. Symp. VLSI Circuits (VLSI Circuits)*, pp. C336–C337.

[5] NORDRUM, A. (2016) "Popular internet of things forecast of 50 billion devices by 2020 is outdated," *IEEE Spectrum*, **18**.
URL `https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated`

[6] OF EVERYTHING CONNECTIONS COUNTER (CISCO), C. I., "How Many Internet Connections are in the World? Right Now." .
URL `http://blogs.cisco.com/news/cisco-connections-counter`

[7] MANYIKA, J., M. CHUI, P. BISSON, J. WOETZEL, R. DOBBS, J. BUGHIN, and D. AHARON (2015) "Unlocking the Potential of the Internet of Things," *McKinsey Global Institute*.

[8] METCALF, D., S. T. MILLIARD, M. GOMEZ, and M. SCHWARTZ (2016) "Wearables and the internet of things for health: wearable, interconnected devices promise more efficient and comprehensive health care," *IEEE pulse*, **7**(5), pp. 35–39.

[9] AZARIADI, D., V. TSOUTSOURAS, S. XYDIS, and D. SOUDRIS (2016) "ECG signal analysis and arrhythmia detection on IoT wearable medical devices," in *Modern Circuits and Systems Technologies (MOCAST), 2016 5th International Conference on*, IEEE, pp. 1–4.

[10] HAGHI, M., K. THUROW, and R. STOLL (2017) "Wearable Devices in Medical Internet of Things: Scientific Research and Commercially Available Devices," *Healthcare informatics research*, **23**(1), pp. 4–15.

[11] HO, G., D. LEUNG, P. MISHRA, A. HOSSEINI, D. SONG, and D. WAGNER (2016) "Smart locks: Lessons for securing commodity internet of things devices," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ACM, pp. 461–472.

[12] SHARIATZADEH, N., T. LUNDHOLM, L. LINDBERG, and G. SIVARD (2016) "Integration of digital factory with smart factory based on Internet of Things," *Procedia CIRP*, **50**, pp. 512–517.

[13] WAHBAH, M., M. ALHAWARI, B. MOHAMMAD, H. SALEH, and M. ISMAIL (2014) "Characterization of human body-based thermal and vibration energy harvesting for wearable devices," *IEEE Journal on emerging and selected topics in circuits and systems*, **4**(3), pp. 354–363.

[14] WANG, S., J. WAN, D. ZHANG, D. LI, and C. ZHANG (2016) "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination," *Computer Networks*, **101**, pp. 158–168.

[15] PERERA, C., A. ZASLAVSKY, P. CHRISTEN, and D. GEORGAKOPOULOS (2014) "Sensing as a service model for smart cities supported by internet of things," *Transactions on Emerging Telecommunications Technologies*, **25**(1), pp. 81–93.

[16] LAZARESCU, M. T. (2013) "Design of a WSN platform for long-term environmental monitoring for IoT applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, **3**(1), pp. 45–54.

[17] BLAAUW, D., D. SYLVESTER, P. DUTTA, Y. LEE, I. LEE, S. BANG, Y. KIM, G. KIM, P. PANNUTO, Y.-S. KUO, ET AL. (2014) "IoT design space challenges: Circuits and systems," in *VLSI Technology (VLSI-Technology): Digest of Technical Papers, 2014 Symposium on*, IEEE, pp. 1–2.

[18] QIAN, X.-H., Y.-C. WU, T.-Y. YANG, C.-H. CHENG, H.-C. CHU, W.-H. CHENG, T.-Y. YEN, T.-H. LIN, Y.-J. LIN, Y.-C. LEE, ET AL. (2017) "A bone-guided cochlear implant CMOS microsystem preserving acoustic hearing," in *VLSI Circuits, 2017 Symposium on*, IEEE, pp. C46–C47.

[19] GENG, Y. and C. G. CASSANDRAS (2013) "New 'smart parking' system based on resource allocation and reservations," *IEEE Transactions on Intelligent Transportation Systems*, **14**(3), pp. 1129–1139.

[20] AKHAVAN-REZAI, E., M. F. SHAABAN, E. F. EL-SAADANY, and F. KARRAY (2016) "Online intelligent demand management of plug-in electric vehicles in future smart parking lots," *IEEE Systems Journal*, **10**(2), pp. 483–494.

[21] RAY, P. P. (2017) "Internet of things for smart agriculture: Technologies, practices and future direction," *Journal of Ambient Intelligence and Smart Environments*, **9**(4), pp. 395–420.

[22] JAYARAMAN, P. P., A. YAVARI, D. GEORGAKOPOULOS, A. MORSHED, and A. ZASLAVSKY (2016) "Internet of things platform for smart farming: Experiences and lessons learnt," *Sensors*, **16**(11), p. 1884.

[23] MA, K., Y. ZHENG, S. LI, K. SWAMINATHAN, X. LI, Y. LIU, J. SAMPSON, Y. XIE, and V. NARAYANAN (2015) "Architecture exploration for ambient energy harvesting nonvolatile processors," *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pp. 526–537.

[24] NADERIPARIZI, S., Z. KAPETANOVIC, and J. R. SMITH (2016) "Battery-free connected machine vision with wispcam," *GetMobile: Mobile Computing and Communications*, **20**(1), pp. 10–13.

[25] NADERIPARIZI, S., A. N. PARKS, Z. KAPETANOVIC, B. RANSFORD, and J. R. SMITH (2015) "Wispcam: A battery-free rfid camera," in *RFID (RFID), 2015 IEEE International Conference on*, IEEE, pp. 166–173.

[26] TALLA, V., B. KELLOGG, B. RANSFORD, S. NADERIPARIZI, S. GOLLAKOTA, and J. R. SMITH (2015) "Powering the next billion devices with Wi-Fi," *arXiv preprint arXiv:1505.06815*.

[27] MA, K., J. LI, T. WU, Z. WANG, X. LI, Y. LIU, Y. XIE, M. T. KANDEMIR, J. SAMPSON, and V. NARAYANAN (2018) "NEOFog: Nonvolatility-Exploiting

Optimizations for Fog Computinge," *International Conference on Architectural Support for Programming Languages and Operating Systems*.

[28] MA, K., X. LI, S. R. SRINIVASA, Y. LIU, J. SAMPSON, Y. XIE, and V. NARAYANAN (2017) "Spendthrift: Machine learning based resource and frequency scaling for ambient energy harvesting nonvolatile processors," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, IEEE, pp. 678–683.

[29] MA, K., X. LI, J. LI, Y. LIU, Y. XIE, J. SAMPSON, M. T. KANDEMIR, and V. NARAYANAN (2017) "Incidental computing on IoT nonvolatile processors," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ACM, pp. 204–218.

[30] WANG, Y., Y. LIU, S. LI, D. ZHANG, B. ZHAO, M. F. CHIANG, Y. YAN, B. SAI, and H. YANG (2012) "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," *ESSCIRC (ESSCIRC), 2012 Proceedings of the*, pp. 149–152.

[31] MA, K., X. LI, Y. LIU, J. SAMPSON, Y. XIE, and V. NARAYANAN (2015) "Dynamic Machine Learning Based Matching of Nonvolatile Processor Microarchitecture to Harvested Energy Profile," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 670–675.

[32] LIU, Y., Z. WANG, A. LEE, F. SU, C. P. LO, Z. YUAN, C. C. LIN, Q. WEI, Y. WANG, Y. C. KING, C. J. LIN, P. KHALILI, K. L. WANG, M. F. CHANG, and H. YANG (2016) "A 65nm ReRAM-enabled nonvolatile processor with 6X reduction in restore time and 4X higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic," *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 84–86.

[33] MA, K., X. LI, H. LIU, X. SHENG, Y. WANG, K. SWAMINATHAN, Y. LIU, Y. XIE, J. SAMPSON, and V. NARAYANAN (2017) "Dynamic power and energy management for energy harvesting nonvolatile processor systems," *ACM Transactions on Embedded Computing Systems (TECS)*, **16**(4), p. 107.

[34] WANG, Z., F. SU, Z. LI, X. LI, R. YOSHIMURA, T. NAIKI, T. TSUWA, T. SAITO, Z. WANG, K. TANIUCHI, M.-F. CHANG, H. YANG, and Y. LIU (2017) "A 130nm FeRAM-based parallel recovery nonvolatile SOC for normally-OFF operations with 3.9x faster running speed and 11x higher energy efficiency using fast power-on detection and nonvolatile radio controller," in *Symposia on VLSI Technology and Circuits*, IEEE, pp. C336–C337.

[35] LUCIA, B., V. BALAJI, A. COLIN, K. MAENG, and E. RUPPEL (2017) "Intermittent Computing: Challenges and Opportunities," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 71, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[36] MAENG, K., A. COLIN, and B. LUCIA (2017) "Alpaca : Intermittent Execution without Checkpoints," *OOPSLA*, **1**(October).

[37] JANKOWSKI, S., J. COVELLO, H. BELLINI, J. RITCHIE, and D. COSTA (2014) "The Internet of Things: Making sense of the next mega-trend," *IoT primer*.

[38] LI, X., U. D. HEO, K. MA, H. LIU, V. NARAYANAN, and S. DATTA (2014) "RF-Powered Systems Using Steep-Slope Devices," *IEEE International New Circuits and Systems Conference*.

[39] SHOJI, K., Y. AKIYAMA, M. SUZUKI, N. NAKAMURA, H. OHNO, and K. MORISHIMA (2014) "Diffusion refueling biofuel cell mountable on insect," in *Micro Electro Mechanical Systems (MEMS), 2014 IEEE 27th International Conference on*, IEEE, pp. 163–166.

[40] KIM, S., R. VYAS, J. BITO, K. NIOTAKI, A. COLLADO, A. GEORGIADIS, and M. TENTZERIS (2014) "Ambient RF Energy-Harvesting Technologies for Self-Sustainable Standalone Wireless Sensor Platforms," *Proceedings of the IEEE*, **102**(11), pp. 1649–1666.

[41] ROUNDY, S., D. STEINGART, L. FRECHETTE, P. WRIGHT, and J. RABAEY (2004) "Power sources for wireless sensor networks," in *Wireless sensor networks*, Springer, pp. 1–17.

[42] GREZAUD, R. and J. WILLEMIN (2013) "A self-starting fully integrated auto-adaptive converter for battery-less thermal energy harvesting," in *New Circuits and Systems Conference (NEWCAS), 2013 IEEE 11th International*, IEEE, pp. 1–4.

[43] LEONOV, V., T. TORFS, P. FIORINI, and C. VAN HOOF (2007) "Thermoelectric converters of human warmth for self-powered wireless sensor nodes," *Sensors Journal, IEEE*, **7**(5), pp. 650–657.

[44] LEONOV, V., T. TORFS, R. J. VULLERS, J. SU, and C. VAN HOOF (2010) "Renewable energy microsystems integrated in maintenance-free wearable and textile-based devices: the capabilities and challenges," in *Industrial Technology (ICIT), 2010 IEEE International Conference on*, IEEE, pp. 967–972.

[45] MEASUREMENT and I. D. C. (MIDC) "http://www.nrel.gov/midc/," .

[46] HARPE, P., E. CANTATORE, and A. VAN ROERMUND (2013) "A 10b/12b 40 kS/s SAR ADC With Data-Driven Noise Reduction Achieving up to 10.1b ENOB at 2.2 fJ/Conversion-Step," *IEEE Journal of Solid-State Circuits*, **48**(12), pp. 3011–3018.

[47] XIAODAN, Z., X. XIAOYUAN, Y. LIBIN, and L. YONG (2009) "A 1-V 450-nW Fully Integrated Programmable Biomedical Sensor Interface Chip," *IEEE Journal of Solid-State Circuits*, **44**(4), pp. 1067–1077.

[48] NADERIPARIZI, S., Z. KAPETANOVIC, and J. R. SMITH (2016) "Battery-free connected machine vision with wispcam," *GetMobile: Mobile Computing and Communications*, **20**(1), pp. 10–13.

[49] SERIS DATASHEET, C.-X. G. "https://www.cap-xx.com/resource/cap-xx-g-series-datasheets/," .

[50] SHENG, X., C. WANG, Y. LIU, H. G. LEE, N. CHANG, and H. YANG (2014) "A high-efficiency dual-channel photovoltaic power system for nonvolatile sensor nodes," in *Non-Volatile Memory Systems and Applications Symposium (NVMSA), 2014 IEEE*, IEEE, pp. 1–2.

[51] NATSUI, M., D. SUZUKI, N. SAKIMURA, R. NEBASHI, Y. TSUJI, A. MORIOKA, T. SUGIBAYASHI, S. MIURA, H. HONJO, K. KINOSHITA, S. IKEDA, T. ENDOH, H. OHNO, and T. HANYU (2013) "Nonvolatile logic-in-memory array processor in 90nm MTJ/MOS achieving 75reduction using cycle-based power gating," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 194–195.

[52] GUO, X., E. IPEK, and T. SOYATA (2010) "Resistive Computation: Avoiding the Power Wall with Low-Leakage, STT-MRAM Based Computing," *2010 Proceedings of the 37th annual international symposium on computer architecture*.

[53] CHOUDHARY, N. K., S. V. WADHAVKAR, T. A. SHAH, H. MAYUKH, J. GANDHI, B. H. DWIEL, S. NAVADA, H. H. NAJAF-ABADI, and E. ROTENBERG (2011) "Fabscalar: Composing synthesizable rtl designs of arbitrary cores within a canonical superscalar template," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, pp. 11–22.

[54] FOOL, T. "http://www.tvfool.com/index.php," .

[55] ESMAEILZADEH, H., E. BLEM, R. ST. AMANT, K. SANKARALINGAM, and D. BURGER (2012) "Dark Silicon and the End of Multicore Scaling," *Micro, IEEE*, **32**(3), pp. 122–134.

[56] MA, K., X. CUI, K. LIAO, N. LIAO, D. WU, and D. YU (2014) "Key characterization factors of accurate power modeling for FinFET circuits," *Science China Information Sciences*, pp. 1–13.

[57] XIAOXIN, C., M. KAISHENG, L. KAI, L. NAN, W. DI, W. WEI, L. RUI, and Y. DUNSHAN (2013) "A Dynamic-Adjusting Threshold-Voltage Scheme for FinFETs low power designs," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pp. 129–132.

[58] MA, K., X. LI, S. LI, Y. LIU, J. J. SAMPSON, Y. XIE, and V. NARAYANAN (2015) "Nonvolatile Processor Architecture Exploration for Energy-Harvesting Applications," *IEEE Micro*, **35**(5), pp. 32–40.

[59] LIU, Y., Z. LI, H. LI, Y. WANG, X. LI, K. MA, S. LI, M.-F. CHANG, S. JOHN, Y. XIE, ET AL. (2015) "Ambient energy harvesting nonvolatile processors: from circuit to system," in *Proceedings of the 52nd Annual Design Automation Conference*, ACM, p. 150.

[60] CHARLES, J., P. JASSI, N. S. ANANTH, A. SADAT, and A. FEDOROVA (2009) "Evaluation of the Intel Core i7 Turbo Boost feature," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, IEEE, pp. 188–197.

[61] ROTEM, E., A. NAVEH, A. ANANTHAKRISHNAN, E. WEISSMANN, and D. RAJWAN (2012) "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, **2**(32), pp. 20–27.

[62] BHAT, M. S., V. V. RAO, B. V. PAI, ET AL. (2014) "Implementation of dynamic voltage and frequency scaling for system level power reduction," in *Circuits, Communication, Control and Computing (I4C), 2014 International Conference on*, IEEE, pp. 425–430.

[63] LIN, B., A. MALLIK, P. DINDA, G. MEMIK, and R. DICK (2009) "User-and process-driven dynamic voltage and frequency scaling," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, IEEE, pp. 11–22.

[64] PARK, S., J. PARK, D. SHIN, Y. WANG, Q. XIE, M. PEDRAM, and N. CHANG (2013) "Accurate modeling of the delay and energy overhead of dynamic voltage and frequency scaling in modern microprocessors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **32**(5), pp. 695–708.

[65] CHOI, K., R. SOMA, and M. PEDRAM (2005) "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *IEEE transactions on computer-aided design of integrated circuits and systems*, **24**(1), pp. 18–28.

[66] KONTORINIS, V., A. SHAYAN, D. M. TULLSEN, and R. KUMAR (2009) "Reducing peak power with a table-driven adaptive processor core," in *Proceedings of the 42nd annual IEEE/ACM international symposium on microarchitecture*, ACM, pp. 189–200.

[67] CAPALIJA, D. and T. S. ABDELRAHMAN (2013) "Microarchitecture of a coarse-grain out-of-order superscalar processor," *IEEE Transactions on Parallel and Distributed Systems*, **24**(2), pp. 392–405.

[68] MA, K., X. LI, Y. LIU, J. SAMPSON, Y. XIE, and V. NARAYANAN (2015) "Dynamic Machine Learning Based Matching of Nonvolatile Processor Microarchitecture to Harvested Energy Profile," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, IEEE Press, pp. 670–675.

[69] LI, X., H. LIU, U. D. HEO, K. MA, S. DATTA, and V. NARAYANAN (2014) "RF-powered systems using steep-slope devices," in *New Circuits and Systems Conference (NEWCAS)*, pp. 73–76.

[70] LIU, H., X. LI, R. VADDI, K. MA, S. DATTA, and V. NARAYANAN (2014) "Tunnel FET RF rectifier design for energy harvesting applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, **4**(4), pp. 400–411.

[71] MA, K., X. LI, J. SAMPSON, Y. LIU, Y. XIE, and V. NARAYANAN (2015) "Nonvolatile processor optimization for ambient energy harvesting scenarios," in *Non-volatile Memory Technology Symposium (NVMTS)*.

[72] ROYANNEZ, P., H. MAIR, F. DAHAN, M. WAGNER, M. STREETER, L. BOUETEL, J. BLASQUEZ, H. CLASEN, G. SEMINO, J. DONG, ET AL. (2005) "90nm low leakage SoC design techniques for wireless applications," in *IEEE International Solid-State Circuits Conference*, pp. 138–140.

[73] GUTHAUS, M. R., J. S. RINGENBERG, D. ERNST, T. M. AUSTIN, T. MUDGE, and R. B. BROWN (2001) "MiBench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, IEEE, pp. 3–14.

[74] SCHAUL, T., J. BAYER, D. WIERSTRA, Y. SUN, M. FELDER, F. SEHNKE, T. RĆ1/4CKSTIEĆ, and J. SCHMIDHUBER (2010) "PyBrain," *Journal of Machine Learning Research*, **11**(Feb), pp. 743–746.

[75] WIDROW, B. and M. E. HOFF (1960) *Adaptive switching circuits*, *Tech. rep.*, STANFORD UNIV CA STANFORD ELECTRONICS LABS.

[76] CARPENTER, G. A. and S. GROSSBERG (1987) "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer vision, graphics, and image processing*, **37**(1), pp. 54–115.

[77] KOSKO, B. (1988) "Bidirectional associative memories," *IEEE Transactions on Systems, Man, and Cybernetics*, **18**(1), pp. 49–60.

[78] ACKLEY, D. H., G. E. HINTON, and T. J. SEJNOWSKI (1985) "A learning algorithm for Boltzmann machines," *Cognitive science*, **9**(1), pp. 147–169.

[79] RUMELHART, D. E., G. E. HINTON, and R. J. WILLIAMS (1985) *Learning internal representations by error propagation*, *Tech. rep.*, DTIC Document.

[80] HECHT-NIELSEN, R. (1987) "Counterpropagation networks," *Applied optics*, **26**(23), pp. 4979–4984.

[81] HOPFIELD, J. J. (1982) "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, **79**(8), pp. 2554–2558.

[82] KOHONEN, T. (1982) "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, **43**(1), pp. 59–69.

[83] LI, X., J. SAMPSON, A. KHAN, K. MA, S. GEORGE, A. AZIZ, S. K. GUPTA, S. SALAHUDDIN, M. F. CHANG, S. DATTA, and V. NARAYANAN (2017) "Enabling Energy-Efficient Nonvolatile Computing With Negative Capacitance FET," *IEEE Transactions on Electron Devices*, **64**(8), pp. 3452–3458.

[84] LI, X., S. GEORGE, K. MA, W. Y. TSAI, A. AZIZ, J. SAMPSON, S. K. GUPTA, M. F. CHANG, Y. LIU, S. DATTA, and V. NARAYANAN (2017) "Advancing Nonvolatile Computing With Nonvolatile NCFET Latches and Flip-Flops," *IEEE Transactions on Circuits and Systems I: Regular Papers*, **PP**(99), pp. 1–13.

[85] LI, X., K. MA, S. GEORGE, W. S. KHWA, J. SAMPSON, S. GUPTA, Y. LIU, M. F. CHANG, S. DATTA, and V. NARAYANAN (2017) "Design of Nonvolatile SRAM with Ferroelectric FETs for Energy-Efficient Backup and Restore," *IEEE Transactions on Electron Devices*, **64**(7), pp. 3037–3040.

[86] LUCIA, B. and B. RANSFORD (2015) "A Simpler, Safer Programming and Execution Model for Intermittent Systems," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '15, pp. 575–585.

[87] XUE, T. and S. ROUNDY (2015) "Analysis of Magnetic Plucking Configurations for Frequency Up-Converting Harvesters," *Journal of Physics Conference Series*, **660**(1), 012098.

[88] XUE, T., X. MA, C. RAHN, and S. ROUNDY (2014) "Analysis of Upper Bound Power Output for a Wrist-Worn Rotational Energy Harvester from Real-World Measured Inputs," *Journal of Physics: Conference Series*, **557**(1), p. 012090.

[89] TAN, Y. K. and S. K. PANDA (2011) "Energy Harvesting From Hybrid Indoor Ambient Light and Thermal Energy Sources for Enhanced Performance of Wireless Sensor Nodes," *IEEE Transactions on Industrial Electronics*, **58**(9), pp. 4424–4435.

[90] LEONOV, V., T. TORFS, P. FIORINI, and C. V. HOOF (2007) "Thermoelectric Converters of Human Warmth for Self-Powered Wireless Sensor Nodes," *IEEE Sensors Journal*, **7**(5), pp. 650–657.

[91] SENNI, S., L. TORRES, A. GAMATIÉ, and G. SASSATELLI (2017) "Non-volatile processor based on MRAM for ultra-low-power IoT devices," *ACM Journal of Emerging Technologies in Computing (JETC)*, **00**(00).

[92] ZWERG, M., A. BAUMANN, R. KUHN, M. ARNOLD, R. NERLICH, M. HERZOG, R. LEDWA, C. SICHERT, V. RZEHAK, P. THANIGAI, and B. O. EVERSMANN (2011) "An 82 uA/MHz microcontroller with embedded FeRAM for energy-harvesting applications," in *2011 IEEE International Solid-State Circuits Conference*, pp. 334–336.

[93] KHANNA, S., S. C. BARTLING, M. CLINTON, S. SUMMERFELT, J. A. RODRIGUEZ, and H. P. MCADAMS (2014) "An FRAM-Based Nonvolatile Logic MCU SoC Exhibiting 100Digital State Retention at VDD= 0 V Achieving Zero Leakage With ¡ 400-ns Wakeup Time for ULP Applications," *IEEE Journal of Solid-State Circuits*, **49**(1), pp. 95–106.

[94] SENNI, S., L. TORRES, G. SASSATELLI, and A. GAMATIE (2016) "Non-Volatile Processor Based on MRAM for Ultra-Low-Power IoT Devices," *J. Emerg. Technol. Comput. Syst.*, **13**(2), pp. 17:1–17:23.

[95] LI, X., H. LIU, U. D. HEO, K. MA, S. DATTA, and V. NARAYANAN (2014) "RF-powered systems using steep-slope devices," *New Circuits and Systems Conference (NEWCAS)*, pp. 73–76.

[96] SU, F., Y. LIU, Y. WANG, and H. YANG (2017) "A Ferroelectric Nonvolatile Processor with 46 us System-Level Wake-up Time and 14 us Sleep Time for Energy Harvesting Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, **64**(3), pp. 596–607.

[97] MERRETT, G. V. and B. M. AL-HASHIMI (2017) "Energy-driven computing: Rethinking the design of energy harvesting systems," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp. 960–965.

[98] RODRIGUEZ ARREOLA, A., D. BALSAMO, A. K. DAS, A. S. WEDDELL, D. BRUNELLI, B. M. AL-HASHIMI, and G. V. MERRETT (2015) "Approaches to transient computing for energy harvesting systems: A quantitative evaluation,"

in *Proceedings of the 3rd International Workshop on Energy Harvesting & Energy Neutral Sensing Systems*, ACM, pp. 3–8.

[99] NADERIPARIZI, S., Z. KAPETANOVIC, and J. R. SMITH (2017) "RF-powered, backscatter-based cameras," in *Antennas and Propagation (EUCAP), 2017 11th European Conference on*, IEEE, pp. 346–349.

[100] MA, K., M. LIAO, X. LI, Z. HUAN, and J. SAMPSON (2017) "Evaluating trade-offs in granularity and overheads in supporting nonvolatile execution semantics," *ISQED*.

[101] YE, R., T. WANG, F. YUAN, R. KUMAR, and Q. XU (2013) "On reconfiguration-oriented approximate adder design and its application," *Proceedings of the International Conference on Computer-Aided Design*, pp. 48–54.

[102] GUPTA, V., D. MOHAPATRA, S. P. PARK, A. RAGHUNATHAN, and K. ROY (2011) "IMPACT: imprecise adders for low-power approximate computing," *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, pp. 409–414.

[103] XIE, Y. (2013) *Emerging Memory Technologies: Design, Architecture, and Applications*, Springer Science & Business Media.

[104] PAN, C., M. XIE, C. YANG, Y. CHEN, and J. HU (2017) "Exploiting Multiple Write Modes of Nonvolatile Main Memory in Embedded Systems," *ACM Transactions on Embedded Computing Systems (TECS)*, **16**(4), p. 110.

[105] SHEIKHOLESLAMI, A. and P. G. GULAK (2000) "A survey of circuit innovations in ferroelectric random-access memories," *Proceedings of the IEEE*, **88**(5), pp. 667–689.

[106] ZHAO, W., X. ZHAO, B. ZHANG, K. CAO, L. WANG, W. KANG, Q. SHI, M. WANG, Y. ZHANG, Y. WANG, ET AL. (2016) "Failure analysis in magnetic tunnel junction nanopillar with interfacial perpendicular magnetic anisotropy," *Materials*, **9**(1), p. 41.

[107] ESMAEILZADEH, H., A. SAMPSON, L. CEZE, and D. BURGER (2012) "Architecture support for disciplined approximate programming," *ACM SIGPLAN Notices*, **47**(4), pp. 301–312.

[108] SAMPSON, A., W. DIETL, E. FORTUNA, D. GNANAPRAGASAM, L. CEZE, and D. GROSSMAN (2011) "EnerJ: Approximate data types for safe and general low-power computation," in *ACM SIGPLAN Notices*, vol. 46, ACM, pp. 164–174.

[109] PARK, J., X. ZHANG, K. NI, H. ESMAEILZADEH, and M. NAIK (2014) *Expax: A framework for automating approximate programming*, *Tech. rep.*, Georgia Institute of Technology.

[110] GUTHAUS, M. ET AL. (2001) "MiBench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pp. 3–14.

[111] WEDDELL, A. S., M. MAGNO, G. V. MERRETT, D. BRUNELLI, B. M. AL-HASHIMI, and L. BENINI (2013) "A survey of multi-source energy harvesting systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*, EDA Consortium, pp. 905–908.

[112] AKINAGA, H. and H. SHIMA (2010) "Resistive random access memory (ReRAM) based on metal oxides," *Proceedings of the IEEE*, **98**(12), pp. 2237–2251.

[113] WANG, K., J. ALZATE, and P. K. AMIRI (2013) "Low-power non-volatile spintronic memory: STT-RAM and beyond," *Journal of Physics D: Applied Physics*, **46**(7), p. 074003.

[114] DONG, X., N. MURALIMANOHAR, N. JOUPPI, R. KAUFMANN, and Y. XIE (2009) "Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future exascale systems," in *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on*, IEEE, pp. 1–12.

[115] PORTAL, J.-M., M. BOCQUET, M. MOREAU, H. AZIZA, D. DELERUYELLE, Y. ZHANG, W. KANG, J.-O. KLEIN, Y.-G. ZHANG, C. CHAPPERT, and W.-S. ZHAO (2014) "An overview of non-volatile flip-flops based on emerging memory technologies," *Journal of Electronic Science and Technology*, **12**(2), pp. 173–181.

[116] LI, X., S. GEORGE, K. MA, W.-Y. TSAI, A. AZIZ, J. SAMPSON, S. K. GUPTA, M.-F. CHANG, Y. LIU, S. DATTA, and V. NARAYANAN (2017) "Advancing Nonvolatile Computing With Nonvolatile NCFET Latches and Flip-Flops," *IEEE Transactions on Circuits and Systems I: Regular Papers*.

[117] CHIU, P.-F., M.-F. CHANG, S.-S. SHEU, K.-F. LIN, P.-C. CHIANG, C.-W. WU, W.-P. LIN, C.-H. LIN, C.-C. HSU, F. T. CHEN, K.-L. SU, M.-J. KAO, and M.-J. TSAI (2010) "A low store energy, low VDDmin, nonvolatile 8T2R SRAM with 3D stacked RRAM devices for low power mobile applications," in *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*, IEEE, pp. 229–230.

[118] WANG, Y., Y. LIU, S. LI, D. ZHANG, B. ZHAO, M.-F. CHIANG, Y. YAN, B. SAI, and H. YANG (2012) "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," in *ESSCIRC (ESSCIRC), 2012 Proceedings of the*, IEEE, pp. 149–152.

[119] LI, J., Y. LIU, H. LI, R. HUA, C. J. XUE, H. G. LEE, and H. YANG (2016) "Accurate personal ultraviolet dose estimation with multiple wearable sensors,"

in *Wearable and Implantable Body Sensor Networks (BSN), 2016 IEEE 13th International Conference on*, IEEE, pp. 347–352.

[120] KYUNG, C.-M., H. YASUURA, Y. LIU, and Y.-L. LIN (2016) *Smart Sensors and Systems: Innovations for Medical, Environmental, and IoT Applications*, Springer.

[121] GASTINEAU, A., T. JOHNSON, and A. SCHULTZ (2009) "Bridge Health Monitoring and Inspections–A Survey of Methods," .

[122] FARRAR, C. R. and K. WORDEN (2012) *Structural health monitoring: a machine learning perspective*, John Wiley & Sons.

[123] HUIFANG, G., M. KAISHENG, and Z. WENCHAO (2011) "The real-time temperature measuring system for the jointless rail," in *Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on*, vol. 3, IEEE, pp. 902–906.

[124] YAN, R. and R. X. GAO (2006) "Hilbert–Huang transform-based vibration signal analysis for machine health monitoring," *IEEE Transactions on Instrumentation and measurement*, **55**(6), pp. 2320–2329.

[125] LEE, J., F. WU, W. ZHAO, M. GHAFFARI, L. LIAO, and D. SIEGEL (2014) "Prognostics and health management design for rotary machinery systemsReviews, methodology and applications," *Mechanical systems and signal processing*, **42**(1), pp. 314–334.

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

[127] LUCIA, B., V. BALAJI, A. COLIN, K. MAENG, and E. RUPPEL (2017) "Intermittent Computing: Challenges and Opportunities," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 71, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[128] ZHAO, M., Q. LI, M. XIE, Y. LIU, J. HU, and C. J. XUE (2015) "Software assisted non-volatile register reduction for energy harvesting based cyber-physical system," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, EDA Consortium, pp. 567–572.

[129] LI, H., Y. LIU, C. FU, C. J. XUE, D. XIANG, J. YUE, J. LI, D. ZHANG, J. HU, and H. YANG (2016) "Performance-aware task scheduling for energy harvesting nonvolatile processors considering power switching overhead," in *Proceedings of the 53rd Annual Design Automation Conference*, ACM, p. 156.

[130] PAN, C., M. XIE, Y. LIU, Y. WANG, C. J. XUE, Y. WANG, Y. CHEN, and J. HU (2017) "A lightweight progress maximization scheduler for non-volatile processor under unstable energy harvesting," in *Proceedings of the 18th ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, ACM, pp. 101–110.

[131] COLLINS, B. D. and R. W. JIBSON (2015) *Assessment of existing and potential landslide hazards resulting from the April 25, 2015 Gorkha, Nepal earthquake sequence*, *Tech. rep.*, US Geological Survey.

[132] HONG, H., W. CHEN, C. XU, A. M. YOUSSEF, B. PRADHAN, and D. TIEN BUI (2017) "Rainfall-induced landslide susceptibility assessment at the Chongren area (China) using frequency ratio, certainty factor, and index of entropy," *Geocarto International*, **32**(2), pp. 139–154.

[133] GUZZETTI, F., A. CARRARA, M. CARDINALI, and P. REICHENBACH (1999) "Landslide hazard evaluation: a review of current techniques and their application in a multi-scale study, Central Italy," *Geomorphology*, **31**(1), pp. 181–216.

[134] PAN, X. and R. TEODORESCU (2014) "Nvsleep: Using non-volatile memory to enable fast sleep/wakeup of idle cores," in *Computer Design (ICCD), 2014 32nd IEEE International Conference on*, IEEE, pp. 400–407.

[135] ONIZAWA, N., A. MOCHIZUKI, A. TAMAKOSHI, and T. HANYU (2017) "Sudden Power-Outage Resilient In-Processor Checkpointing for Energy-Harvesting Nonvolatile Processors," *IEEE Transactions on Emerging Topics in Computing*, **5**(2), pp. 151–163.

[136] YU, W.-K., S. RAJWADE, S.-E. WANG, B. LIAN, G. E. SUH, and E. KAN (2011) "A non-volatile microcontroller with integrated floating-gate transistors," in *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*, IEEE, pp. 75–80.

[137] LEE, S. H., Y. S. BAE, and L. CHOI (2016) "The design of a ultra-low power RF wakeup sensor for wireless sensor networks," *Journal of Communications and Networks*, **18**(2), pp. 201–209.

[138] JIANG, H., P.-H. P. WANG, L. GAO, P. SEN, Y.-H. KIM, G. M. REBEIZ, D. A. HALL, and P. P. MERCIER (2017) "24.5 A 4.5 nW wake-up radio with-69dBm sensitivity," in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*, IEEE, pp. 416–417.

[139] ISRAR, N. and I. AWAN (2006) "Multi-hop clustering algo. For load balancing in WSN," *International Journal of SIMULATION*, **8**(1).

[140] KIM, N., J. HEO, H. S. KIM, and W. H. KWON (2008) "Reconfiguration of clusterheads for load balancing in wireless sensor networks," *Computer Communications*, **31**(1), pp. 153–159.

[141] LÁSZLÓ, E., K. TORNAI, G. TREPLÁN, and J. LEVENDOVSZKY (2011) "Novel load balancing scheduling algorithms for wireless sensor networks," in *The Fourth Int. Conf. on Communication Theory, Reliability, and Quality of Service, Budapest*, pp. 54–49.

[142] OZDEMIR, S. (2009) "Secure Load Balancing via Hierarchical Data Aggregation in Heterogeneous Sensor Networks." *Journal of Information Science & Engineering*, **25**(6).

[143] DENG, Y. and Y. HU (2010) "A load balance clustering algorithm for heterogeneous wireless sensor networks," in *E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference on*, IEEE, pp. 1–4.

[144] SIAVOSHI, S., Y. S. KAVIAN, and H. SHARIF (2016) "Load-balanced energy efficient clustering protocol for wireless sensor networks," *IET Wireless Sensor Systems*, **6**(3), pp. 67–73.

[145] KIM, H.-Y. (2016) "An energy-efficient load balancing scheme to extend lifetime in wireless sensor networks," *Cluster Computing*, **19**(1), pp. 279–283.

[146] PAL, V., G. SINGH, and R. YADAV (2015) "Balanced cluster size solution to extend lifetime of wireless sensor networks," *IEEE Internet of Things Journal*, **2**(5), pp. 399–401.

[147] WAJGI, D. and N. V. THAKUR (2012) "Load balancing based approach to improve lifetime of wireless sensor network," *International Journal of Wireless & Mobile Networks*, **4**(4), p. 155.

[148] ZHANG, D., Y. LIU, J. LI, C. J. XUE, X. LI, Y. WANG, and H. YANG (2016) "Solar power prediction assisted intra-task scheduling for nonvolatile sensor nodes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **35**(5), pp. 724–737.

[149] RANSFORD, B., J. SORBER, and K. FU (2011) "Mementos: System Support for Long-running Computation on RFID-scale Devices," in *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, ACM, New York, NY, USA, pp. 159–170.

[150] XIE, M., M. ZHAO, C. PAN, J. HU, Y. LIU, and C. J. XUE (2015) "Fixing the broken time machine: Consistency-aware checkpointing for energy harvesting powered non-volatile processor," in *Proceedings of the 52nd Annual Design Automation Conference*, ACM, p. 184.

[151] LUCIA, B. and B. RANSFORD (2015) "A simpler, safer programming and execution model for intermittent systems," *ACM SIGPLAN Notices*, **50**(6), pp. 575–585.

[152] LI, Q., M. ZHAO, J. HU, Y. LIU, Y. HE, and C. J. XUE (2015) "Compiler directed automatic stack trimming for efficient non-volatile processors," in *Proceedings of the 52nd Annual Design Automation Conference*, ACM, p. 183.

[153] ZHANG, D., S. LI, A. LI, Y. LIU, X. S. HU, and H. YANG (2014) "Intra-task scheduling for storage-less and converter-less solar-powered nonvolatile sensor nodes," in *Computer Design (ICCD), 2014 32nd IEEE International Conference on*, IEEE, pp. 348–354.

[154] ZHANG, D., Y. LIU, X. SHENG, J. LI, T. WU, C. J. XUE, and H. YANG (2015) "Deadline-aware task scheduling for solar-powered nonvolatile sensor nodes with global energy migration," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, IEEE, pp. 1–6.

[155] JAYAKUMAR, H., A. RAHA, and V. RAGHUNATHAN (2014) "QuickRecall: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers," in *2014 13th International Conference on Embedded Systems*, IEEE, pp. 330–335.

[156] COLIN, A., G. HARVEY, B. LUCIA, and A. P. SAMPLE (2016) "An energy-interference-free hardware-software debugger for intermittent energy-harvesting systems," *ACM SIGPLAN Notices*, **51**(4), pp. 577–589.

[157] COLIN, A., G. HARVEY, A. P. SAMPLE, and B. LUCIA (2017) "An Energy-Aware Debugger for Intermittently Powered Systems," *IEEE Micro*, **37**(3), pp. 116–125.

[158] COLIN, A., A. P. SAMPLE, and B. LUCIA (2015) "Energy-interference-free system and toolchain support for energy-harvesting devices," in *Proceedings of the 2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, IEEE Press, pp. 35–36.

[159] LIU, Y., Z. LI, H. LI, Y. WANG, X. LI, K. MA, S. LI, M.-F. CHANG, S. JOHN, Y. XIE, ET AL. (2015) "Ambient energy harvesting nonvolatile processors: from circuit to system," *Proceedings of the 52nd Annual Design Automation Conference*, p. 150.

[160] VAN DER WOUDE, J. and M. HICKS (2016) "Intermittent computation without hardware support or programmer intervention," in *Proceedings of OSDI16: 12th USENIX Symposium on Operating Systems Design and Implementation*, p. 17.

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

[162] TI "CTPL "Compute Through Power Loss" software utility," .

[163] BALSAMO, D., A. S. WEDDELL, G. V. MERRETT, B. M. AL-HASHIMI, D. BRUNELLI, and L. BENINI (2015) "Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems," *IEEE Embedded Systems Letters*, **7**(1), pp. 15–18.

[164] BALSAMO, D., A. S. WEDDELL, A. DAS, A. R. ARREOLA, D. BRUNELLI, B. M. AL-HASHIMI, G. V. MERRETT, and L. BENINI (2016) "Hibernus++: a self-calibrating and adaptive system for transiently-powered embedded devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **35**(12), pp. 1968–1980.

[165] LIU, V., A. PARKS, V. TALLA, S. GOLLAKOTA, D. WETHERALL, and J. R. SMITH (2013) "Ambient backscatter: wireless communication out of thin air," *ACM SIGCOMM Computer Communication Review*, **43**(4), pp. 39–50.

[166] KELLOGG, B., A. PARKS, S. GOLLAKOTA, J. R. SMITH, and D. WETHERALL (2014) "Wi-Fi backscatter: Internet connectivity for RF-powered devices," in *ACM SIGCOMM Computer Communication Review*, vol. 44, ACM, pp. 607–618.

[167] WANG, C., N. CHANG, Y. KIM, S. PARK, Y. LIU, H. G. LEE, R. LUO, and H. YANG (2014) "Storage-less and converter-less maximum power point tracking of photovoltaic cells for a nonvolatile microprocessor," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, IEEE, pp. 379–384.

[168] SHENG, X., C. WANG, Y. LIU, H. G. LEE, N. CHANG, and H. YANG (2014) "A high-efficiency dual-channel photovoltaic power system for nonvolatile sensor nodes," in *2014 IEEE Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pp. 1–2.

[169] CHAOUR, I., S. BDIRI, A. FAKHFAKH, and O. KANOUN (2016) "Modified rectifier circuit for high efficiency and low power RF energy harvester," in *2016 13th International Multi-Conference on Systems, Signals Devices (SSD)*, pp. 619–623.

[170] MASOTTI, D. and A. COSTANZO (2015) "Start-up solutions for ultra-low power RF harvesting scenarios," in *2015 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO)*, pp. 1–3.

[171] YAO, R. and S. N. PAKZAD (2012) "Autoregressive statistical pattern recognition algorithms for damage detection in civil structures," *Mechanical Systems and Signal Processing*, **31**, pp. 355–368.

[172] CERDA, F., S. CHEN, J. BIELAK, J. H. GARRETT, P. RIZZO, and J. KOVACEVIC (2014) "Indirect structural health monitoring of a simplified laboratory-scale bridge model," *Smart Structures and Systems*, **13**(5), pp. 849–868.

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

[174] BOGDAN, P., M. PAJIC, P. P. PANDE, and V. RAGHUNATHAN (2016) "Making the Internet-of-things a Reality: From Smart Models, Sensing and Actuation to Energy-efficient Architectures," in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES '16, ACM, New York, NY, USA, pp. 25:1–25:10.

[175] PERRICONE, R., I. AHMED, Z. LIANG, M. G. MANKALALE, X. S. HU, C. H. KIM, M. NIEMIER, S. S. SAPATNEKAR, and J.-P. WANG (2017) "Advanced spintronic memory and logic for non-volatile processors," in *Proceedings of the Conference on Design, Automation & Test in Europe*, European Design and Automation Association, pp. 972–977.

[176] K. YU, W., S. RAJWADE, S. E. WANG, B. LIAN, G. E. SUH, and E. KAN (2011) "A non-volatile microcontroller with integrated floating-gate transistors," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 75–80.

[177] MA, Y., S. MIURA, H. HONJO, S. IKEDA, T. HANYU, H. OHNO, and T. ENDOH (2016) "A 600-μW ultra-low-power associative processor for image pattern recognition employing magnetic tunnel junction-based nonvolatile memories with autonomic intelligent power-gating scheme," *Japanese Journal of Applied Physics*, **55**(4S), p. 04EF15.

[178] RANSFORD, B. and B. LUCIA (2014) "Nonvolatile Memory is a Broken Time Machine," in *Proceedings of the Workshop on Memory Systems Performance and Correctness*, MSPC '14, ACM, New York, NY, USA, pp. 5:1–5:3.

[179] JAYAKUMAR, H., A. RAHA, and V. RAGHUNATHAN (2014) "QUICKRECALL: A Low Overhead HW/SW Approach for Enabling Computations across Power Cycles in Transiently Powered Computers," in *VLSI Design and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference on*, pp. 330–335.

[180] LI, C., W. ZHANG, C.-B. CHO, and T. LI (2011) "SolarCore: Solar energy driven multi-core architecture power management," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pp. 205–216.

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

[182] MIRHOSEINI, A., B. D. ROUHANI, E. SONGHORI, and F. KOUSHANFAR (2016) "Chime: Checkpointing Long Computations on Interm ittently Energized IoT Devices," *IEEE Transactions on Multi-Scale Computing Systems*, **2**(4), pp. 277–290.

[183] MA, K., X. LI, S. LI, Y. LIU, J. J. SAMPSON, Y. XIE, and V. NARAYANAN (2015) "Nonvolatile Processor Architecture Exploration for Energy-Harvesting Applications," *IEEE Micro*, **35**(5), pp. 32–40.

[184] MA, K., X. LI, K. SWAMINATHAN, Y. ZHENG, S. LI, Y. LIU, Y. XIE, J. J. M. SAMPSON, and V. NARAYANAN (2016) "Nonvolatile Processor Architectures: Efficient, Reliable Progress with Unstable Power." *IEEE Micro*, **36**(3), pp. 72–83.

[185] SU, F., K. MA, X. LI, T. WU, Y. LIU, and V. NARAYANAN (2017) "Nonvolatile processors: Why is it trending?" in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp. 966–971.

[186] MAHAJAN, D., A. YAZDANBAKHSH, J. PARK, B. THWAITES, and H. ESMAEILZADEH (2015) "Towards Statistical Guarantees in Controlling Quality Tradeoffs for Approximate Acceleration," *ACM SIGARCH Computer Architecture News*, **43**(3), pp. 554–566.

[187] KHUDIA, D. S., B. ZAMIRAI, M. SAMADI, and S. MAHLKE (2015) "Rumba: an online quality management system for approximate computing," *ACM SIGARCH Computer Architecture News*.

[188] SAN MIGUEL, J. and N. E. JERGER (2016) "The Anytime Automaton," *International Symposium on Computer Architecture*.

[189] TANG, X., M. KANDEMIR, P. YEDLAPALLI, and J. KOTRA (2016) "Improving bank-level parallelism for irregular applications," in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, IEEE, pp. 1–12.

[190] SAMADI, M., J. LEE, D. A. JAMSHIDI, A. HORMATI, and S. MAHLKE (2013) "Sage: Self-tuning approximation for graphics engines," *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 13–24.

[191] BACHA, A. and R. TEODORESCU (2014) "Using ECC feedback to guide voltage speculation in low-voltage processors," *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 306–318.

[192] MA, K., X. LI, J. SAMPSON, Y. LIU, Y. XIE, and V. NARAYANAN (2015) "Nonvolatile Processor Optimization for Ambient Energy Harvesting Scenarios," *15th Non-volatile Memory Technology Symposium (NVMTS 2015)*.

[193] LIU, S., K. PATTABIRAMAN, T. MOSCIBRODA, and B. G. ZORN (2012) "Flikker: saving DRAM refresh-power through critical data partitioning," *ACM SIGPLAN Notices*, **47**(4), pp. 213–224.

[194] SAMPSON, A., J. NELSON, K. STRAUSS, and L. CEZE (2014) "Approximate storage in solid-state memories," *ACM Transactions on Computer Systems (TOCS)*, **32**(3), p. 9.

[195] MIGUEL, J. S., M. BADR, and N. E. JERGER (2014) "Load value approximation," *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 127–139.

[196] SUI, X., A. LENHARTH, D. S. FUSSELL, and K. PINGALI (2016) "Proactive Control of Approximate Programs," *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 607–621.

[197] WONG, D., N. S. KIM, and M. ANNAVARAM (2016) "Approximating warps with intra-warp operand value similarity," *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 176–187.

[198] SAMADI, M., D. A. JAMSHIDI, J. LEE, and S. MAHLKE (2014) "Paraprox: Pattern-based approximation for data parallel applications," *ACM SIGARCH Computer Architecture News*, **42**(1), pp. 35–50.

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

[200] ST AMANT, R., A. YAZDANBAKHSH, J. PARK, B. THWAITES, H. ESMAEILZADEH, A. HASSIBI, L. CEZE, and D. BURGER (2014) "General-purpose code acceleration with limited-precision analog computation," *ACM SIGARCH Computer Architecture News*, **42**(3), pp. 505–516.

[201] MISHRA, N., H. ZHANG, J. D. LAFFERTY, and H. HOFFMANN (2015) "A probabilistic graphical model-based approach for minimizing energy under performance constraints," *ACM SIGPLAN Notices*, **50**(4), pp. 267–281.

[202] CARBIN, M., S. MISAILOVIC, and M. C. RINARD (2013) "Verifying quantitative reliability for programs that execute on unreliable hardware," in *ACM SIGPLAN Notices*, vol. 48, ACM, pp. 33–52.

[203] SAMPSON, A., A. BAIXO, B. RANSFORD, T. MOREAU, J. YIP, L. CEZE, and M. OSKIN (2015) "ACCEPT: A programmer-guided compiler framework for practical approximate computing," *University of Washington Technical Report UW-CSE-15-01*, **1**.

[204] KIM, Y. J., H. S. BHAMRA, J. JOSEPH, and P. P. IRAZOQUI (2015) "An Ultra-Low-Power RF Energy-Harvesting Transceiver for Multiple-Node Sensor Application," *IEEE Transactions on Circuits and Systems II: Express Briefs*, **62**(11), pp. 1028–1032.

[205] PRUMMEL, J., M. PAPAMICHAIL, J. WILLMS, R. TODI, W. AARTSEN, W. KRUISKAMP, J. HAANSTRA, E. OPBROEK, S. RIEVERS, P. SEESINK, J. VAN GORSEL, H. WOERING, and C. SMIT (2015) "A 10 mW Bluetooth low-energy transceiver with on-chip matching," *IEEE Journal of Solid-State Circuits*, **50**(12), pp. 3077–3088.

[206] SELVAKUMAR, A., M. ZARGHAM, and A. LISCIDINI (2015) "13.6 A 600uW Bluetooth low-energy front-end receiver in 0.13 um CMOS technology," in *Solid-State Circuits Conference-(ISSCC), 2015 IEEE International*, IEEE, pp. 1–3.

[207] LIN, Z., P.-I. MAK, and R. MARTINS (2014) "9.4 A 0.5 V 1.15 mW 0.2 mm 2 Sub-GHz ZigBee receiver supporting 433/860/915/960MHz ISM bands with zero external components," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, IEEE, pp. 164–165.

[208] GUPTA, G. and M. YOUNIS (2003) "Load-balanced clustering of wireless sensor networks," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 3, IEEE, pp. 1848–1852.

[209] ZHANG, H., L. LI, X.-F. YAN, and X. LI (2011) "A load-balancing clustering algorithm of WSN for data gathering," in *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on*, IEEE, pp. 915–918.

[210] ABD, M. A., S. F. M. AL-RUBEAAI, B. K. SINGH, K. E. TEPE, and R. BENLAMRI (2015) "Extending wireless sensor network lifetime with global energy balance," *IEEE Sensors Journal*, **15**(9), pp. 5053–5063.

[211] LI, X., K. MA, S. GEORGE, W.-S. KHWA, J. SAMPSON, S. GUPTA, Y. LIU, M.-F. CHANG, S. DATTA, and V. NARAYANAN (2017) "Design of Nonvolatile SRAM with Ferroelectric FETs for Energy-Efficient Backup and Restore," *IEEE Transactions on Electron Devices*.

[212] LI, X., J. SAMPSON, A. KHAN, K. MA, S. GEORGE, A. AZIZ, S. K. GUPTA, S. SALAHUDDIN, M.-F. CHANG, S. DATTA, ET AL. (2017) "Enabling energy-efficient nonvolatile computing with negative capacitance fet," *IEEE Transactions on Electron Devices*, **64**(8), pp. 3452–3458.

Vita

Kaisheng Ma

Kaisheng Ma received his Ph.D. degree in Department of Computer Science and Engineering, The Pennsylvania State University. He received the Masters degree in Electronics and Communication Engineering from Peking University, Beijing, in 2013.

In the past 5 years, He has published 36 papers (18 first author papers), and 377 google citations (till March 2018), and has several patents in the US. As the first author, he has won many awards, including 2015 HPCA Best Paper Award, 2016 IEEE MICRO Top Picks, 2017 ASP-DAC Best Paper Award. He has many honors, including 2016 Penn State CSE Department Best Graduate Research Award, 2016 Cover Feature of NSF ASSIST Engineering Research Center Newsletter (Among 40 graduate students across four participating universities.), 2011 Yang Fuqing & Wang Yangyuan Academician Scholarship (1/126, Peking University.).

His research interests include Non-volatile processor architecture and Neural Networks Accelerator Design.