UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Science
School of Electronics and Computer Science

Energy Allocation and Management for Energy-Harvesting Intermittent Systems

by

Jie Zhan

A thesis for the degree of
Doctor of Philosophy

November 2021

University of Southampton

Abstract

Faculty of Engineering and Physical Science
School of Electronics and Computer Science

Doctor of Philosophy

Energy Allocation and Management for Energy-Harvesting Intermittent Systems

by Jie Zhan

A growing number of autonomous sensor nodes are expected to be deployed in the Internet of Things. Energy harvesting has gained increasing attention for powering these nodes due to its features of long lifetime and energy independence. However, energy harvesting sources manifest intrinsic temporal and spatial variability, which conflicts with the requirement of stable power supply for conventional electronic designs. Energy-driven computing has recently developed to adapt system architecture to only ambient energy sources, with various specifications, such as prolonging active time, increasing energy efficiency, and ensuring forward execution, for different scenarios. While energy storage and energy harvester are critical components in terms of application performance and device dimensions, considerations on how to size energy storage and energy harvester have not been fully studied.

This thesis presents a study on the sizing issue of energy storage and energy harvester in deploying self-powered computing devices. A sizing method is proposed with respect to real-world energy conditions. A configurable system model is built to represent a typical execution process in energy harvesting computing. Based on this model, the sizing method is implemented. This sizing method suggests an efficient range of energy harvester sizes that balance average application performance and harvester dimensions. In this range of harvester sizes, results show that properly sizing energy storage leads to an execution speedup by 5.7-22.2% under real-world deployment. Furthermore, exploration extends to the sizing effect of energy storage on a recent-published power adaptation method for storage-less energy-driven computing.

Contents

List of Figures **ix**

List of Tables **xi**

1 Introduction **1**
1.1 Energy-Harvesting Intermittently-Powered Systems 2
1.1.1 Application Suitability of Storage-less Energy Harvesting Computing 4
1.2 Research Justification 6
1.3 Research Questions 8
1.4 Contributions 9

2 Energy-Harvesting Intermittent Systems **11**
2.1 Energy Harvesting Techniques 11
2.1.1 Light Energy Harvesting 12
2.1.2 Radio Frequency Energy Harvesting 15
2.1.3 Flow Energy Harvesting 15
2.2 Energy Storage for Sensor Nodes 16
2.2.1 Rechargeable Batteries 17
2.2.2 Capacitors 18
2.2.3 Discussion 19
2.3 Energy-Neutral Computing 20
2.4 Intermittent Computing 22
2.4.1 Checkpointing IC 22
2.4.2 Reactive IC 24
2.4.3 Harvest-Store-Use IC 26
2.4.4 Task-based IC 27
2.4.5 Non-Volatile Processors 28
2.5 Power-Neutral Computing 29
2.5.1 Principles of Operations 29
2.5.2 Recent Approaches 30
2.6 Summary 32

3 Effect of Energy Storage Sizing on Intermittent Computing System Performance **35**
3.1 Motivation 35
3.2 Related Work 37
3.3 Reactive ICS Modelling 38

3.3.1 Operating Modes of Reactive ICS . . . 39
3.3.2 Formulating Forward Progress . . . 40
3.4 Exploration of Energy Storage Sizing . . . 41
3.4.1 Model Configuration . . . 42
3.4.1.1 Energy Storage . . . 42
3.4.1.2 Intermittent Computing Load . . . 42
3.4.2 Sizing Energy Storage to Improve Forward Progress . . . 43
3.4.2.1 Impact of Supply Current . . . 43
3.4.2.2 Impact of Volatile State Size . . . 44
3.5 Experimental Validation . . . 45
3.5.1 Model Validation . . . 45
3.5.2 Validation of Sizing Effects . . . 46
3.6 Summary and Discussion . . . 46

4 Energy Storage Sizing Approach for Deploying Intermittent Computing Systems 47
4.1 Motivation . . . 47
4.2 Energy Storage Sizing Approach . . . 47
4.2.1 Input . . . 47
4.2.2 System Model . . . 48
4.2.3 Trade-off . . . 48
4.3 Sizing under Real-World Energy Conditions . . . 49
4.3.1 Simulation Configuration . . . 49
4.3.2 Exploration with Real-World Energy Source Conditions . . . 50
4.3.2.1 Sizing the Energy Harvester . . . 50
4.3.2.2 Sizing the Energy Storage . . . 50
4.3.2.3 Interruption Period . . . 50
4.3.3 Trading Forward Progress, Dimensions, and Interruption Period 52
4.3.3.1 Metric of Dimensions . . . 52
4.3.3.2 Metric of Interruption Periods . . . 52
4.3.3.3 Cost Function . . . 53
4.3.3.4 Results . . . 53
4.4 Summary and Discussion . . . 53

5 Runtime Energy Profiling and Threshold Adaptation 55
5.1 Introduction . . . 55
5.2 Background and Motivation . . . 57
5.2.1 Intermittent Peripheral Operations . . . 57
5.2.2 Variability in Energy Consumption, Storage, and Input . . . 58
5.2.2.1 Capacitor Degradation and Tolerance . . . 58
5.2.2.2 Data size . . . 58
5.2.2.3 Peripheral Configurations . . . 58
5.2.2.4 Device . . . 59
5.2.2.5 Other . . . 59
5.2.2.6 Voltage . . . 59
5.2.2.7 Charging efficiency of energy harveser vs Vcc . . . 60
5.2.3 Forward Progress Improvement with Adaptive Energy Budgeting 60

5.2.3.1 Model Description 60
5.2.3.2 Simulation Setup 60
5.2.3.3 Results 61
5.3 Methodology 62
5.3.1 Online Profiling Method 62
5.3.1.1 Assumptions 62
5.3.1.2 Mathmatical Derivation 63
5.3.1.3 Realistic Considerations 64
5.3.2 Threshold Adaptation 65
5.3.2.1 Profiling Strategy 65
5.3.2.2 Learning Algorithm 65
5.4 Implementation 66
5.5 Experimental Evaluation 67
5.5.1 Benchmark and Comparisons 67
5.5.2 Profiling Accuracy 67
5.5.3 Reliability with Dynamic Energy Consumption 68
5.5.3.1 New devices / operations (once) 69
5.5.3.2 Variability in capacitance due to ageing / tolerance (slowly changing) 69
5.5.3.3 Variability in peripheral configurations (single threshold for a rarely/slightly-changing configuration, multiple thresholds for frequently-changing configurations) . 69
5.5.3.4 Variability in the amount of data to process (fast changing, but perhaps could be an unsuitable test case for reliability as it should violate the API requirement to make it fail) 69
5.5.4 Efficiency 69
5.5.5 Overheads 70
5.5.6 Correctness of computational results (test its intermittent computing functionality, might not be important) 70
5.5.7 Case Study 70
5.6 Conclusions 70
5.7 Summary and Discussion 70

6 Conclusions and Future Work **71**
6.1 Conclusions 71
6.2 Future Work 72

References **73**

List of Figures

1.1 A conceptual architecture of an IPS. 4
1.2 Application Suitability of Storage-less Energy Harvesting Computing. . 6

2.1 Typical current-voltage curve of a solar module (monocrystalline, adapted from [37]) 13
2.2 Daily global horizontal solar energy available in Los Angeles 2012-2014 [44]. 14
2.3 One-day dynamics of global horizontal solar irradiance in Los Angeles 29 April 2016 [45]. 14
2.4 Dynamics of a micro wind harvester (reproduced from [30]). 16
2.5 Voltage trace with hibernation and restoration points in Hibernus (taken from [68]). 25
2.6 Harvest-Store-Use execution (taken from [72]). 26
2.7 Memory architectures of traditional processors and NVPs (taken from [76]). 28
2.8 Architecture of an example power neutral system based on TI MSP430FR platform (taken from [30]). 31
2.9 Board power consumption of ODROID XU4 vs operating frequency and core configurations, running CPU intensive application Raytrace (taken from [31]). 32

3.1 The relationship between energy storage capacitance and ICS forward progress, for various supply currents. 36
3.2 Operating modes of reactive ICSs, and achieved forward progress against supply current. 39
3.3 Operating cycles in the *Intermittent* mode. 40
3.4 Forward progress against energy storage capacitance at different levels of constant supply current. Error bars around optimal points denote the impact of typical $\pm 20\%$ capacitance tolerance. 44
3.5 Maximum forward progress improvement by sizing energy storage given a spectrum of supply current (normalized by the minimum capacitance case), with the corresponding maximum and sub-maximum (95% of maximum) capacitance. 44
3.6 Impact of RAM usage (linear to restore/save overheads) on sizing energy storage with 0.4 mA current supply. Improvement and reduction are normalized by the minimum capacitance case. 45
3.7 Model validation with experimental and modelled forward progress. . 46

4.1 Structure of the proposed system model and sizing approach. 48
4.2 System model of a PV-based ICS. 49

4.3 Improvement of average forward progress by sizing energy storage given different PV panel areas under real-world energy source conditions. The model is able to find the PV panel area required for achieving the target mean forward progress. 51
4.4 Distribution of interruption periods. 51
4.5 Tantalum capacitor volume against capacitance for the six series of capacitors analyzed. 52
4.6 The sizing approach trades off forward progress, capacitor volume, and interruption periods. The results are plotted against a range of PV panel area, given Denver 2018 energy source dataset. 54

5.1 Voltage drop vs data size. 58
5.2 IV curve. 60
5.3 Dynamic execution time. 61
5.4 Dynamic current draw. 61
5.5 Figure for illustrating mathematical derivation. 63
5.6 System circuit. 65
5.7 DMA profiling. 68
5.8 AES profiling. 68
5.9 UART profiling. 68

List of Tables

2.1 Classification of energy sources and energy harvesters in IoT. 12
2.2 Comparison between commercial NiMH and Li-ion rechargeable batteries. 18

3.1 Model parameters of reactive ICS 38
3.2 Profiled MCU parameters 43
3.3 Linear scaling range of volatile state size and restore/save time overheads 45

4.1 PV cell properties under a 1000 W/cm^2, AM-1.5 light source 49

5.1 Peripheral configuration variability, AES Encryption 4KB 59
5.2 Device variability, AES 128-bit Encryption 4KB 59

Chapter 1

Introduction

The promising expansion of the Internet of Things (IoT) has drawn research interests on new design paradigms for deploying tens of billions of electronic devices over a wide geographical range and probably in hard-to-reach places [1, 2]. Such a scenario generates considerations on how to enable the devices in networks to operate independently and effectively and how to construct a long-life, maintenance-free, environmentally friendly, and low-cost IoT.

One of the most significant concerns in deploying IoT devices is how to power numerous low-power devices (tens of billions expected [1, 3, 4]). Traditional wired electricity limits flexibility of deployment and involve expensive wiring costs [5]. Traditional primary batteries (i.e. non-rechargeable batteries) are not suitable for such a large number of devices. A widespread use of primary batteries can cause tedious work of battery replacement due to the limited battery lifetime, and also pose pollution concerns as these batteries are typically made of non-disposable heavy-metal materials [Reference needed]. Therefore, it is necessary to find an alternative powering solution.

A potential power alternative is energy harvesters. Energy harvesters scavenge energy from environmental sources (e.g. solar irradiation, wind flow, radio frequency (RF) signals, and kinetic energy) [6].

Some more information on energy harvesters? e.g. a table of power density of common enegy harvesting sources.

Devices powered by energy harvesters can get rid of power wires and surpass the lifetime limit of primary batteries, enabling a scalable IoT. However, the power generated by energy harvesters in real-world deployment is variable, uncontrollable, and in many cases insufficient for continuous workload operation [7]. Hence, directly using energy harvesters as the power supply without energy buffering may cause a device to keep booting up and shutting down, making little application progress.

Initially, large energy storage, in forms of rechargeable batteries (also known as secondary batteries) or supercapacitors, is allocated with energy harvesters to buffer the temporal variations of energy input and provide reliable power supply. Motivated by such a scenario, energy-neutral (EN) operation was proposed to balance energy input and energy consumption so as to prevent a system from power failures [8]. EN operation intends to sustain systems over a long period of time (e.g. a few days [9] or a year [10]) by adapting system runtime schedules (e.g. duty cycles [9–11] or task schedules [12, 13]) according to the available energy amount.

Rechargeable batteries and supercapacitors are two main choices of energy storage in EN operation. Rechargeable batteries are historically used as energy storage in energy harvesting embedded systems because of their high energy density [14] and stable discharging profile [8]. However, due to electrolyte deterioration, the limited charge-discharge cycles of rechargeable batteries constrain the operating lifetime, causing heavy battery replacement work as well as environmental issues as primary batteries do [15]. To alleviate the problems of rechargeable batteries, supercapacitors are then explored in research. Although the energy density of supercapacitors is several orders of magnitude lower than the energy density of batteries [16], supercapacitors outperform rechargeable batteries in terms of lifetime (e.g. up to 10-20 years for supercapacitors compared to 3-5 years for rechargeable batteries [17]). However, to achieve a comparable energy capacity as batteries, supercapacitors should be designed to tens of farads or one hundred farads [18, 19]. Supercapacitors in such a scale occupy large volume in contrast to small IoT devices.

Numbers?

1.1 Energy-Harvesting Intermittently-Powered Systems

To circumvent the lifetime, pollution, and volume problems in rechargeable batteries and supercapacitors, a research trend in energy-harvesting sensor nodes moved towards eliminating the demand for energy storage and adopting only a minimum amount of energy storage, where the energy storage is only enough for ensuring the most energy-expensive atomic operation[1], typically in the form of a µF-level capacitor.

Capacitors have longer lifetime, smaller volume, blabla... Reference for lifetime of electrolytic capacitors. Seems a bit contradictory to contribution 3.

[1] In this context, an operation is atomic if it should be completed in one consecutive period without power interrupts; otherwise, if interrupted, it should be re-executed from the beginning. Example atomic operations in IoT devices can be peripheral operations and nonvolatile memory read/write operations.

Despite the benefits of small capacitors over batteries and supercapacitors, they considerably limit the buffering capacity for varying harvested power. Thus, the variable harvesting power is almost directly connected to the load, e.g. with only a decoupling capacitor. This violates the demand for stable power supply in conventional computing systems. Without any modifications, a conventional system can only work when input power is higher than system power consumption (which is rare for an energy-harvesting supply), and cannot boot up when input power is lower than system power consumption. Hence, it becomes a major concern that how to guarantee forward execution and functionality of such systems with only minimum energy storage.

Ensuring and improving local processing ability of sensor nodes is crucial for a few reasons. First, to reduce network traffic volume and energy consumption, sensor nodes should be able to process sensing data on-site and transmit only the useful information, typically when the number of sensor nodes increases in orders of magnitude [4]. Second, advanced communications techniques, such as scheduling, routing, coding, and decoding, require local computing ability to ensure timeliness and efficiency in networking [20]. Third, IoT devices are also expected to be able to trigger actions in reaction to the physical world by either receiving commands from other nodes and servers or making a decision based on locally acquired data [21].

Need many citations in the next paragraph.

With an energy-harvesting supply and small energy buffering capacitance, a system is powered up *intermittently* once a small amount of energy is accumulated in the capacitor. The system has to utilize these intermittent power-on cycles to make application progress. To this end, many approaches for energy-harvesting Intermittently-Powered Systems (IPS) have been proposed in the past few years. The majority of these approaches have been addressing how to sustain computational progress throughout intermittent power-on cycles by correctly and efficiently saving and restoring volatile computing state. The volatile computing state includes CPU registers, Static RAM (SRAM) data, and perhaps peripheral configurations and data, i.e. the volatile part that cannot sustain after a power failure. The volatile state is saved into and restored from non-volatile memory (NVM), where most published approaches use Ferroelectric RAM (FRAM). According to the style of saving and restoring state, approaches in IPSs can be categorized as *proactive* and *reactive*. Proactive approaches save and restore state at design-time or compile-time defined points, whilst reactive approaches do that upon an imminent power failure when supply falls below a low threshold ($V_{cc} < V_L$). Details of these approaches are illustrated in Chapter 2.

A conceptual architecture of an typical energy-harvesting IPS is shown in Figure 1.1. The power frontend is an energy harvester, which transduces an ambient energy source into electric power. Then, a power regulator converts the harvested power to a suitable voltage that charges up the energy storage. The type of the power regulator depends

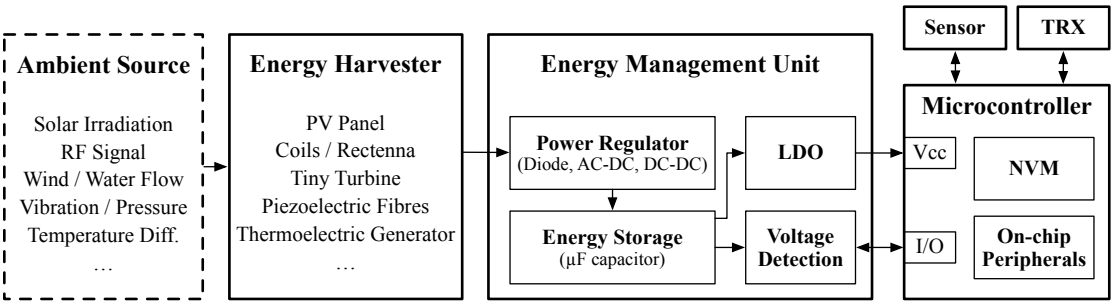


FIGURE 1.1: A conceptual architecture of an IPS.

on the pattern of the power input, which can be an AC-DC converter for AC input (e.g. a rectenna), or a DC-DC converter or a diode for DC input (e.g. a PV panel). The energy storage is in the form of a µF-level capacitor, which buffers a small amount of energy for the load to operate intermittently in short active cycles. The power given to the load is usually conditioned through a low-dropout regulator (LDO) to lower down supply voltage, and hence current consumption. IPSs are usually equipped with a voltage detection circuit so as to wake up or power up the load when the voltage of the energy storage reaches a threshold. In IPSs, the load is typically a microcontroller (MCU) with NVM to sustain computing state, and with many on-chip or external peripherals, e.g. sensors and wireless transceivers (TRX).

1.1.1 Application Suitability of Storage-less Energy Harvesting Computing

An inherent limitation of storage-less energy harvesting computing is that the systems can only execute when there is available power, as opposed to EN computing where the EN systems can still executes with buffered energy if ambient power is not available. This limitation thereby requires that *application operation periods and power availability should be compatible in time*. While there are various needs of application operation periods for various applications scenarios, the power availability is constrained and determined by the availability of the target energy source in the deployed environment. The applications of storage-less energy harvesting computing should be adapted or selected to suit the power availability. Under this consideration, there are two typical categories of application scenarios according to recent publications.

- **Category 1: Applications with flexible time requirements.**

  Applications with flexible requirements on operating periods tolerate the intermittency of energy harvesting sources. In such applications, energy-harvesting devices are allowed to wait for power-available periods to execute.

  Use case 1: Kitchen event detection

  The application aims to capture kitchen events (e.g. dishwasher working, fan on, refrigerator cooling) to record equipment usage. As such events usually last for

from tens of seconds to a few hours, the device does not need to operate immediately after the event occurs or disappears. An implementation of this application is shown by Maeng et al. [22]. The device iterates the following tasks in turn during power-available periods: sampling acoustic information from microphone input, classifying kitchen events with a pre-trained model, and transmitting the results in Bluetooth Low-Energy (BLE) packets to an always-on server. The device harvests RF energy from a dedicated RFID reader, and the packets are transmitted every a few seconds as reported. A specification for this application is to complete program iterations as frequently as possible so as to improve the accuracy of event records.

Use case 2: Temperature monitor for air conditioning

The application aims to monitor indoor temperature for air conditioning. As the temperature does not usually change over a few minutes, the temperature monitor does not need to wake up frequently or periodically. An implementation of this application is shown by Colin et al. [23]. During power-available periods, the device samples temperature by an external analog sensor. If the temperature is detected to be out of a pre-defined range, the device send a BLE packet to alarm the server. The device is also powered by a dedicated RFID reader. Similar to use case 1, the device is expected to maximize sampling frequency in order to capture out-of-range temperature as soon as possible.

- **Category 2: Application activity in correlation with available power.**

  In such applications, the application operations correlates with power-available periods. This correlation is typically linked by an event with harvestable power. When the event occurs, the device is activated by harvesting the power of the event at the same time to start operating. Therefore, the application operation periods and the power availability are inherently simultaneous in such applications.

  Use case 3: Bicycle trip counter

  The bicycle trip counter aims to calculate cycling speed and traveled distance. The wheel rotation brings energy for the device to sense the cycling speed; the device does not need to operate without cycle movement. The trip counter is designed as a nail-sized chip installed on the frame of a bicycle, with a magnet on the wheel for gaining energy [24]. Every wheel rotation activates the trip counter to calculate the current speed and log traveled distance. After collecting enough energy over a few cycling rounds, the trip counter transmits the logged information. This application is also expected to complete computing tasks faster and report results as frequently as possible.

  Use case 4: Power meter

The power meter measures the power flow of a main load wire. The AC power in the wire can be harvested by a coil to activate the power meter. A design is shown in Monjolo [25], where the power meter transmits a plain packet to a server once it collects a preset amount of energy. The server then calculates the elapsed time between the recent two packets to estimate the main load power.

As shown in the above use cases, a common application specification of storage-less energy harvesting computing is to do as much 'work' as possible under the same energy conditions, e.g. completing program iterations as frequently as possible, because the energy cannot be saved for later use. Other 'work' could include improving sensing accuracy or processing offloaded tasks received from other devices. To summarize the application suitability of storage-less energy harvesting computing, a diagram is shown in Figure 1.2. An example unsuitable application can be a periodic sensing task without periodically available power or the period of the energy source does not match the sensing period (left bottom circle in Figure 1.2).

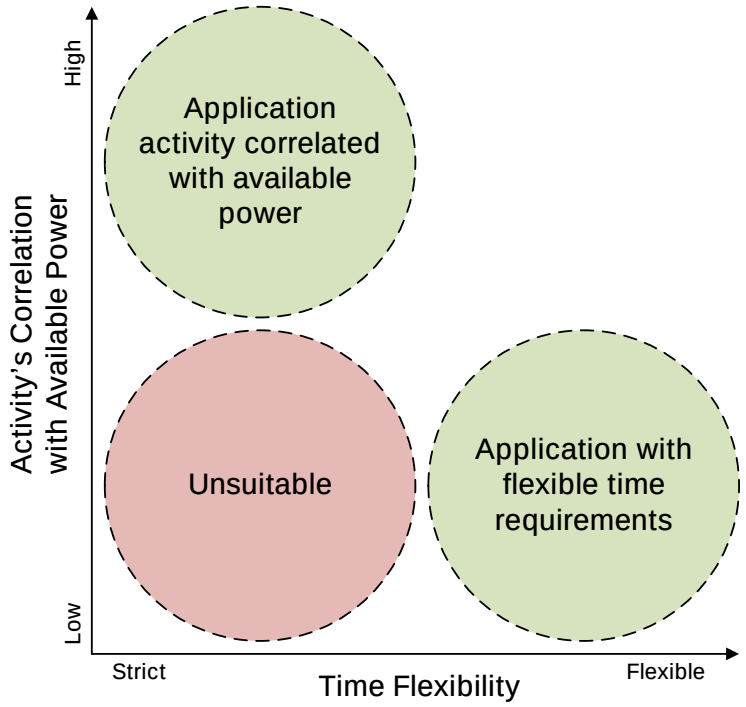


FIGURE 1.2: Application Suitability of Storage-less Energy Harvesting Computing.

1.2 Research Justification

Self-powered devices solely rely on energy harvesters to power themselves. Energy harvesting power supply varies over time due to uncontrollable environmental conditions. Due to the lifespan, pollution, dimension problems of batteries and supercapacitors, there is a demand for adapting computing architecture to energy harvesting

supply with small storage, e.g. a capacitor. Various approaches in intermittent computing and PN computing emerge to enable and improve energy harvesting computing in the scenario of minimized storage in order to reduce dimensions and cost as well as to increase lifespan of IoT devices. However, a larger storage than the minimum one can reduce the frequency of power outages and increase flexibility in power management, which may benefit the application throughput. Hence, there is a trade-off in sizing energy storage. Apart from energy storage, the size of energy harvesters should also be considered based on a set of factors, such as system power consumption, performance requirements, energy utilization, device dimensions, and cost. A gap of study exists in how to determine the size of energy storage and energy harvesters. The research gap is further described as listed below:

1. A variety of methods in intermittent computing have been proposed to overcome power outages during execution, ensuring forward execution [26], memory consistency [27], peripheral re-configuration [28], etc. Such approaches usually adopt only a minimum amount of energy storage which only allow systems to complete state saving and restoring operations before and after power outages. However, provisioning a certain amount of storage reduces the overhead of intermittent operations. This effect is especially noticeable in reactive intermittent systems (Section 2.4.2), where the energy and time overheads of saving and restoring operations are proportional to the frequency of power outages. On the other hand, increasing energy storage also increases the system leakage power and affects the reactivity (the energy and the charging time to activate devices) of IPSs [23, 29]. The trade-off of scaling energy storage in intermittent computing systems should be fully studied.

2. The size of energy harvesters significantly determines the amount of harvested power. Undersized energy harvesters constrain available power, affecting the maximum performance of devices; oversized energy harvesters provides redundant energy, which cannot be consumed on useful work and is wasted in circuitry or never harvested, affecting the utilization of energy harvesters and unnecessarily increasing device costs. Besides, the size of energy harvesters contributes to a large part in device dimensions, which may violate the size constraint of autonomous devices [10]. Therefore, there is a trade-off between under-provisioning and over-provisioning energy harvesters in energy harvesting devices, and this trade-off should be thoroughly studied.

3. Current power management techniques in energy harvesting computing rely on the feedback of the dynamic detection of available energy in energy storage, whether in EN systems [9, 13] or in PN systems [30, 31]. Scaling the capacity of energy storage may change the responsiveness of such energy detection, and consequently, have an impact on performance adaptation and system application

throughput. This impact exhibits more significant in PN computing than in EN computing as the allocated energy storage in PN systems is minimized and energy detection is more acute.

1.3 Research Questions

Motivated by the previous discussion, the following three research questions are derived:

1. What is the effect of sizing the energy storage capacity on the performance of IPSs?

   Specifically, the energy storage capacity in IPSs is presented as the capacitance between V_{cc} and ground. In IPSs, *forward progress* denotes the effective application progress, excluding re-executed progress, lost progress, and state-saving and -restoring operations [32]. The forward progressing rate directly determines application performance, e.g. program iteration rate or task completion time, and hence is regarded as the performance metric in this study. The goal is to explore whether sizing the energy storage capacity can change the forward progressing rate in IPSs, and if so, to study and quantify the relationship between them.

2. How may the energy storage of IPSs be sized to trade off forward progress against device dimensions and interruption periods?

   While the last question explores the energy storage sizing effect on computational performance, this question encompasses more design factors in IPSs that a capacitor size can affect. Increasing energy storage capacity may benefit forward progress, but may also exhibit side effects. A larger capacitor typically has larger physical dimensions, which are a key design factor that IPSs should minimize in some application scenarios, e.g. wearable and implantable sensors. Also, a larger capacitor also leads to longer charge-discharge cycles, and thus prolongs interruption periods and undermines system reactivity to external events. The goal is to study the trade-off and to propose an approach that recommends an energy storage size for practical deployment.

3. How can an IPS run safely and efficiently with runtime variable energy consumption of tasks?

   Energy consumption of tasks can change at runtime with regards to many factors, where we consider, but not limited to, the variability in data amounts to process, peripheral configurations, devices, and capacitor degradation. A design concept is to allocate just enough energy for each task. This design concept can further break into two aspects – safety and efficiency. The safety aspect means

that the IPS should intend to avoid non-termination by allocating enough energy for tasks. The efficiency aspect means that, while meeting the safety aim, the IPS should minimize the energy budget, such that the system can wait for the least possible energy threshold, maintaining energy efficiency and forward progress. The goal is to devise an approach that can enable IPSs to run with variable energy consumption of tasks, following the above design concept.

1.4 Contributions

The contributions done to the address the research questions in this thesis are:

1. Exploration and analysis of the energy storage sizing effect on reactive intermittent computing system, where we quantify and explain the relationship between energy storage capacity and forward progress.
2. A method and a simulation tool for sizing energy storage in deploying IPSs, where forward progress, dimensions, and interruption periods are traded off in a cost function.
3. An online task profiling and voltage threshold adaptation approach for efficiently performing atomic operations and coping with capacitor degradation.

Chapter 2

Energy-Harvesting Intermittent Systems

The emerging of energy harvesters provides diversified prospects for design paradigms of energy harvesting sensor nodes in IoT applications [21]. This chapter first provides a review on various energy harvesting sources and corresponding energy harvesters in Section 2.1, followed by energy storage techniques that used in energy harvesting computing in Section 2.2. Energy-neutral computing, an early paradigm in energy harvesting computing, is reviewed in Section 2.3. Finally, two recent research topics towards storage-less energy harvesting computing, i.e. intermittent computing and power-neutral computing, are reviewed in Section 2.4 and Section 2.5 respectively with an illustration of proposed methodologies.

2.1 Energy Harvesting Techniques

For all kinds of energy harvesters, although there is only one basic concept — to extract energy from ambient sources, various energy sources and harvesters lead to miscellaneous output characteristics in the amount and wave forms of harvested power, voltage, and current. To select a energy harvester for powering sensor nodes, one important concern is whether the supply power level matches the consumption of the load [33]. For one certain type of energy harvesters, the amount of energy harvesting supply can be scaled within an extent by the amount of energy sources or scaling the energy harvesters. The amount of energy sources is determined by the deploying environment, which cannot be controlled by the harvesting devices, but the size of energy harvesters can be decided at design-time with considerations on systems requirements, such as energy utilization, form factors, performance, etc.

In order to appropriately size and designate energy harvesters for sensor nodes, the power features of different energy harvesters are widely considered by researchers and engineers [34]. A classification of common energy harvesting sources and corresponding energy harvesters used in IoT is presented in Table 2.1. The voltage and current features of different energy harvesters largely differ from each other, due to the intrinsic differences in temporal distributions of the available amount of different energy sources and the physical principles of power conversion. The following part of this section introduces each kind of energy sources and energy harvesting techniques listed in Table 2.1.

Energy source	**Energy harvester**
Light (solar, artificial)	Photovoltaic cells
Radio waves	Radio frequency harvester (antenna)
Flow (air, liquid)	Wind turbine, hydrogenerator
Mechanical (vibrations, pressure, stress-strain)	Electromagnetic, electrostatic, piezoelectric harvester
Heat	Theomo electric generator

TABLE 2.1: Classification of energy sources and energy harvesters in IoT.

2.1.1 Light Energy Harvesting

Due to the abundant energy amount of light, whether from outdoor sunlight or indoor artificial light, light energy becomes a feasible source to powering sensor nodes and is historically treated as a substitute for battery supplies [35, 36]. Light energy can be converted to DC power by photovoltaic (PV) cells, which consist of semiconducting materials, e.g. silicon. When PV cells absorb light, electrons are excited by the photovoltaic effect, producing an electric potential by the separation of electrons and holes.

Given a fixed intensity of light, the output current from PV cells manifests an inverse relationship with the output voltage, as there is a semiconducting bypass within the PV cells. Although the power conversion efficiency may vary among different PV cell techniques (such as monocrystalline, polycrystalline, thin film), the curve shapes of current-voltage relationships are similar. Obviously, higher irradiance leads to higher current output when the output voltage is fixed, because there is more intensive light sources provided. More importantly, the output feature of PV cells can be summarised like ”an inverse semiconductor” —— when the terminal voltage is low, the output

current is almost constant and close to short-circuit current; when the terminal voltage gets close to open-circuit voltage, the output current significantly decreases and finally terminates at the open-circuit voltage.

Typical current-voltage curves of PV cells are shown in Figure 2.1, with an example of a monocrystalline cell given five values of illumination intensity from 200 W/m^2 to 1000 W/m^2. When the voltage is under 15V, the PV cell is similar to a current generator (so when the voltage increases, the power increases almost linearly). When the output voltage increases above 15V, the output current drops significantly and reaches zero at around 22V. According to this phenomenon, there is a voltage point where the cell produces the maximum power, which is named Maximum Power Point (MPP) as indicated by black dots in Figure 2.1.

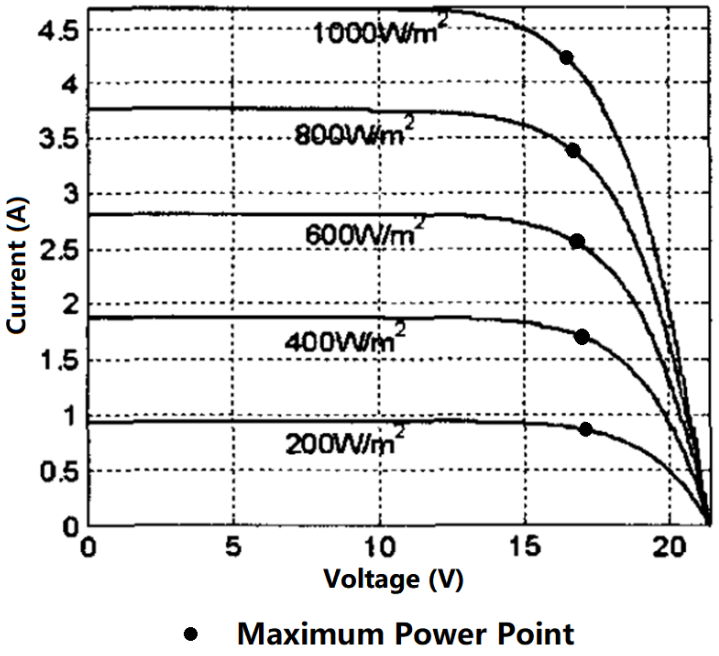


FIGURE 2.1: Typical current-voltage curve of a solar module (monocrystalline, adapted from [37])

In order to extract as much power as possible out of PV panels, most energy harvesting systems with PV modules adopts maximum power point tracking (MPPT) techniques [38–40]. MPPT is achieved by dynamically controlling the output voltage of PV cells around the maximum power point (MPP).

Outdoor sunlight and indoor artificial light are two main sources for light energy harvesting. The illumination intensity of direct sunlight on the earth's surface is typically 1000 W/m^2 [41], while the typical indoor illumination intensity is 10 W/m^2 [33]. Due to this large difference in the power density of these two circumstances, PV modules are more prospective in outdoor applications for harvesting solar energy. Conversion efficiency of PV cells is typically 15% to 25% in outdoor conditions [42].

Solar energy is an uncontrollable but partially predictable source [10, 43]. Solar irradiance demonstrates daily and annual periodicity due to the regularity of celestial movements, as well as irregular variations due to cloud movements, air mass, etc. A 3-year trace of diurnal global horizontal solar energy available measured in Los Angeles from 2012 to 2014 is presented in Figure 2.2, and an example of daily dynamics of global horizontal solar irradiance of the same location is presented in Figure 2.3. As shown in both figures, the predictability is reflected from the roughly annual and daily periodicity, and the uncontrollability and randomness relates to the irregular variations, which include both daily variations in an annual scale and variations over a few seconds and minutes on a daily scale.

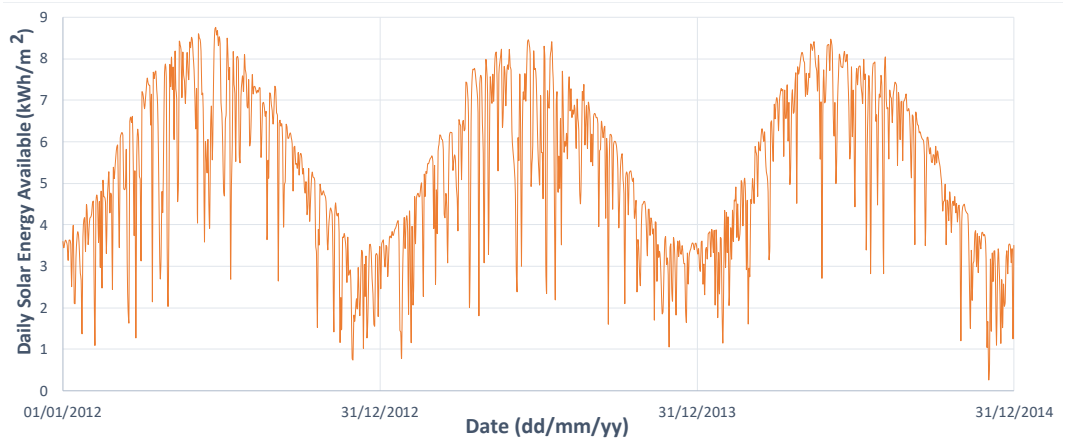


FIGURE 2.2: Daily global horizontal solar energy available in Los Angeles 2012-2014 [44].

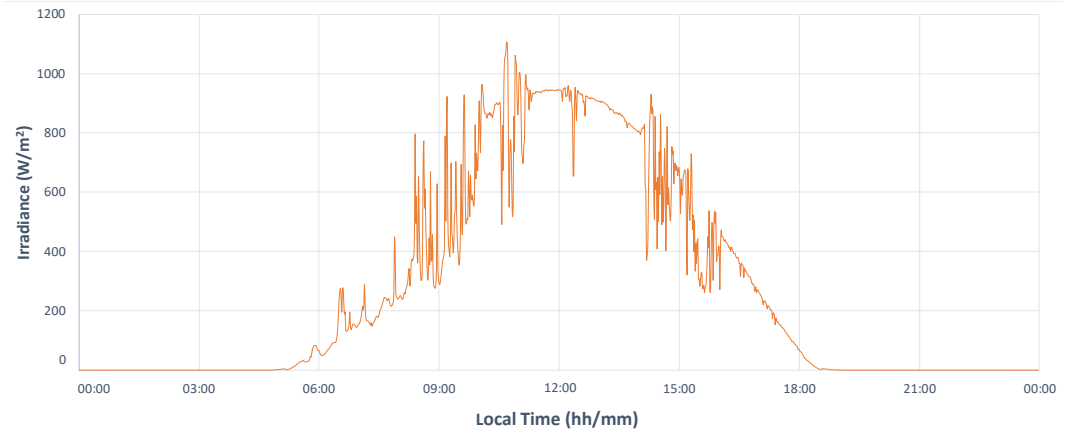


FIGURE 2.3: One-day dynamics of global horizontal solar irradiance in Los Angeles 29 April 2016 [45].

In order to make full use of solar energy, substantial efforts have been made to develop and improve energy harvesting sensor nodes with solar panels [35, 36]. Generally, solar energy harvesting approaches adopt large energy storage, e.g. a rechargeable battery, to smooth out the daily and annual variations. Examples of solar-powered sensor nodes are presented in [9, 35, 46], and a comprehensive review on solar-powered sensor nodes is published in [8].

2.1.2 Radio Frequency Energy Harvesting

Radio Frequency (RF) energy exists in time-varying electromagnetic fields, which widely spread in our environment now due to the propagation of wireless networks, such as Wi-Fi and cellular phone signals [47]. When radio waves pass through an antenna, due to electromagnetic induction, AC voltage is generated. This AC voltage can be rectified and regulated to DC power for sensor nodes. The received RF power is reciprocal to the square of distance from the source to the destination. The maximum conversion efficiency from RF waves to DC power is typically 50-75% given a transmission distance of 100 metres [33].

Due to the widespread deployment of telecommunication networks, RF energy harvesting becomes available in a wide range of locations, both outdoors and indoors. Compared to light energy harvesting, RF harvesting shows its strength in indoor applications as there is often low or no light intensity inside buildings.

A basic and common example of RF harvesting is RF Identification (RFID). In a passive RFID application, an RFID reader transmits RF signals to an RFID tag for asking its tag information. The tag absorbs the signals and energy through its antenna, and then responds the reader with its information. Up to now, Wireless Identification and Sensing Platform (WISP) [48, 49] is presented to show the possibility of the integration of RF energy harvesting in IoT applications.

2.1.3 Flow Energy Harvesting

Flow-based energy harvesting utilizes turbines and rotors to collect the kinetic energy in air flow or liquid flow. Air flow is converted by wind turbines and liquid flow is converted by hydrogenerators. Wind turbines and hydrogenerators are normally in different mechanical structures (shapes), but the fundamental principles of them are the same.

Wind turbines are manufactured in a wide spectrum in terms of dimensions, from a large-scale wind farm (arrays of large turbines) to a portable micro wind turbine. Micro wind turbines are suitable for battery charging and powering autonomous electronic devices.

A raw voltage output trace of a micro wind turbine given a blast of wind is presented in Figure 2.4. Given a constant blow, a wind turbine should produce a sinusoidal voltage signal. Its voltage output vibrates from the positive domain to the negative domain with time, so a rectifier is normally required in order to utilize this AC power for DC load.

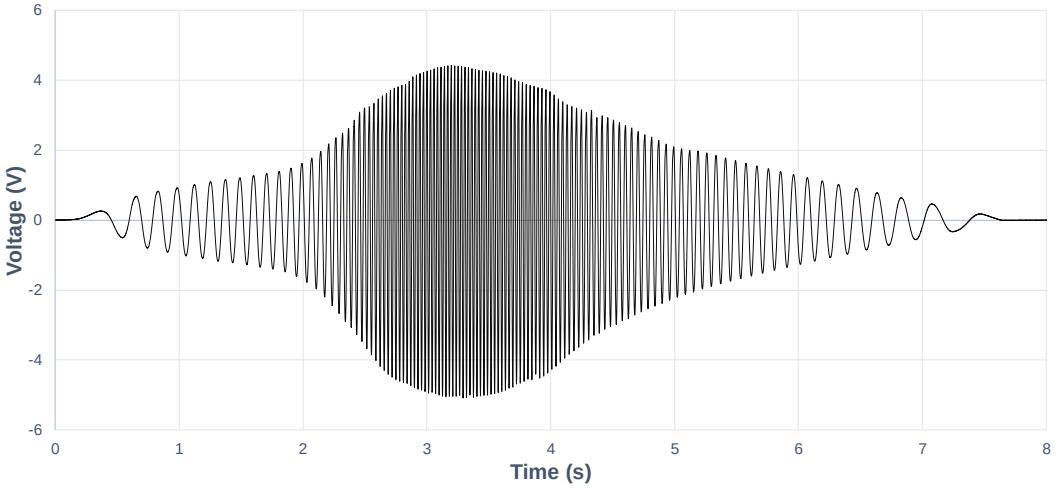


FIGURE 2.4: Dynamics of a micro wind harvester (reproduced from [30]).

Similar to solar energy, wind energy is uncontrollable but partially predictable. Sharma et al. [50] introduce a system that achieves available wind energy predictions based on downloaded weather forecast information within recent 3 days. Also, Cammarano et al. [51] present a wind and solar energy predicting method which dynamically adjusts its time horizon of prediction in order to achieve higher accuracy than its prior methods.

Hydrogenerators harness the energy in moving liquids, such as water or a mix of different liquids. Traditionally, hydrogenerators are used for generating large-scale electricity from rivers and streams. However, since the possible underwater applications in IoT, hydrogenerators can be a suitable alternative for powering sensor nodes. For example, Morais et al. [52] incorporate a small-sized hydrogenerator as a part of energy harvesting supply for sensor nodes.

2.2 Energy Storage for Sensor Nodes

Energy harvesting supply is variable and intermittent over time, causing disparity between power supply and power consumption. In order to deliver stable power output from a varying source, a critical component in an energy harvesting power unit is energy storage, which buffers the harvested energy and powers the load when needed. Besides its ability to buffer energy and its effect on overall efficiency, energy storage has a dominant effect on the size, cost, and lifetime of sensor nodes [14]. Therefore, how to design energy storage is a critical concern in deploying energy harvesting sensor nodes.

Technologies of energy storage used in sensor nodes are generally divided into two categories: rechargeable batteries and capacitors, which are different from each other in

terms of energy density, power density, lifetime, discharging features, leakage, etc. In general, batteries have higher energy density (containing more energy with the same volume/weight), lower leakage, and a more stable discharging curve (a stable voltage output while discharging), while capacitors have higher power density (higher limits for charging/discharging current), and longer lifetime in terms of charge-discharge cycles [14, 35]. The choice of these two forms of energy storage depends on application requirements. These two technologies and their implementations will be briefly reviewed in the following subsections.

2.2.1 Rechargeable Batteries

Batteries are more energy-dense than capacitors and manifest a stable voltage output when discharging. Rechargeable batteries have been widely adopted in mobile devices. Rechargeable batteries are generally made in the following techniques: Sealed Lead Acid (SLA), Nickel Cadmium (NiCd), Nickel Metal Hydride (NiMH), Lithium Ion (Li-ion), and Lithium ion Polymer (Li-Po). Due to the similar techniques and features of Li-ion and Li-Po batteries, Li-ion will be used to represent Li-ion and Li-Po batteries in this subsection. SLA and NiCd batteries are less likely to be implemented in energy harvesting sensor nodes [14, 35]. SLA batteries suffer from low energy density and are normally bulky and heavy, which is unfavorable for sensor nodes. NiCd batteries involve memory effect, i.e. decrease of energy capacity after repeated partially discharging and recharging, which is a common situation in energy harvesting implementations.

Compared to SLA and NiCd batteries, NiMH and Li-ion batteries show a strength in energy density in both weight and volume, and hence, are more suitable for energy harvesting applications [14, 35, 53, 54]. A comparison of two commercial NiMH and Li-ion batteries is listed in Table 2.2 with a variety of perspectives and features. Li-ion batteries are typical lighter than NiMH batteries, with weight energy density 2-3x and volume energy density 1-2x to NiMH batteries. Also, Li-ion batteries significantly outperform NiNH batteries in terms of charging efficiency and self-discharge rate. However, Li-ion batteries are normally more expensive than NiMH batteries, and require more complicated pulse charging circuits [35]. NiMH batteries also provide a relatively constant voltage supply during discharging [9].

NiMH and Li-ion batteries have been widely implemented in energy harvesting sensor nodes. Heliomote [35] uses two NiMH batteries in series to match the charging voltage (2.2-2.8V) with the MPP of the solar panel. HydroSolar [53] also adopts two NiMH batteries to avoid the Li-ion charging hardware. Jiang et al. [18] design a hybrid storage system including a lithium based rechargeable battery as the secondary buffer, due to its high efficiency and charge density.

	NiMH (Panasonic BK150AA)	Li-ion (EEMB LIR14500)
Nominal voltage	1.2 V	3.7 V
Charge capacity	1500 mAh	750 mAh
Energy capacity	1.80 Wh	2.775 Wh
Weight	26 g	20 g
Dimensions	⌀14.5mm × 50.5mm	⌀14.1mm × 48.5mm
Weight energy density	69 Wh/Kg	139 Wh/Kg
Volume energy density	216 Wh/L	366 Wh/L
Operating temperature	-20°C to 65°C	-20°C to 60°C
Charging cycles (until 80% capacity)	>500	>300
Reference price	£2.91	£3.25
Charging efficiency [54]	66%	99.9%
Self-discharge [54]	30% per month	10% per month
Charging Method [54]	Trickle	Pulse

TABLE 2.2: Comparison between commercial NiMH and Li-ion rechargeable batteries.

Despite the high energy density and stable discharging voltage, batteries still show a typical drawback at short lifetime (less than 5 years [17]), which involves manual replacement of batteries or devices after the battery lifetime expires. Also, batteries raise environmental concerns due to the heavy metals and toxic chemicals within. If not properly charged, Li-ion batteries can cause safety issues, i.e. explosion and fire, which are problematic when deployed in distant and wild places. In addition, rechargeable batteries are susceptible to temperature. Most batteries only exhibit their rated characteristics around 20°C, and lose their efficiency and capacity when operating at extreme temperatures (around their rated limits) [54].

2.2.2 Capacitors

Due to the lifetime limits and pollution issues of batteries, capacitors, typically supercapacitors, are considered as an alternative to replace rechargeable batteries as energy storage. Supercapacitor (also known as ultracapacitors or electrostatic double-layer capacitors) are capacitors with higher energy density than electrolytic capacitors. Unlike conventional capacitors, where charges are stored and separated by solid dielectric, supercapacitors maintain charges based on double-layer or pseudo-capacitive charging phenomena [55]. Supercapacitors are still much less energy-dense than batteries, but act as a transition from capacitors to batteries.

Compared to rechargeable batteries, supercapacitors exhibit strengths in a large number of charge/discharge cycles, long lifetime (20 years), high charge/discharge efficiency (98%). The self-discharge rate of supercapacitors is higher than batteries, with 5.9-11% of maximum capacity per day [56, 57], but this leakage is insignificant compared to the small capacity and the total energy gained per day. The main constraint

of supercapacitors is still the low energy density, which results in large storage dimensions if the aim were to achieve a comparable capacity with batteries. In order to maintain the same form factors of sensor nodes, designers have to adapt system architecture to a small storage (compared to batteries).

Prometheus [18] introduces supercapacitors into energy storage for sensor nodes whereby two 22F supercapacitors are used in combine with a Li-Po battery. AmbiMax [58] also proposes a hybrid storage design similar to Prometheus, but with two more 10F supercapacitors for wind energy harvesters. To achieve longer lifetime than battery-based sensor nodes, Everlast [19] demonstrates the feasibility of replacing batteries with supercapacitors in energy harvesting sensor nodes, designing a power system that adopts a 100F supercapacitor as the only energy reservoir.

However, farad-level supercapacitors occupy a significant part of device volume. The advent of energy-driven computing [59] introduces the application and design scenario where execution happens only if there is energy available. Within this scenario, energy storage using millifarad-level supercapacitors are investigated in energy harvesting sensing applications [49, 60]. Furthermore, intermittent computing, which will be illustrated in the next section, enables computation given intermittent power, making progress with electrolytic capacitors or even without dedicated storage (only microfarad-level parasitic capacitance).

2.2.3 Discussion

To summarise, due to the requirements on lifetime, environmental-friendliness, and form factors in energy harvesting sensor nodes, the energy storage designs have transformed from batteries to supercapacitors, and eventually eliminated the need for dedicated storage.

Batteries have been the preferable choice for buffering harvested energy and powering sensor nodes because they make sensor nodes easy to program and operate reliably until the battery lifetime expires. However, the environmental issues and short lifetime of batteries limit the deployment of ubiquitous sensors. Supercapacitors avoid the problems of batteries and have been used to replace batteries, but the low energy density of supercapacitors also makes sensor nodes bulky and heavy in order to achieve sufficient capacity for uninterrupted operations. Recent development of intermittent computing enables forward execution over power outages and encourages storage-less designs in energy harvesting sensor nodes.

Although the minimum need for storage capacity to operate sensor nodes decreases with the evolution of computing techniques, decreased storage does limit the flexibility of energy usage. A storage-less system has to consume the incoming power immediately, otherwise the energy is wasted. This fact consequently restricts the application

scenarios of storage-less systems to energy-driven applications, where execution needs to run only when there is available energy sources to harvest. However, energy-driven applications do not cover all the demands in IoT, so simply reducing the storage need is not always desirable. A wider spectrum of storage designs should be explored to suit and optimise for different application scenarios.

2.3 Energy-Neutral Computing

Energy-neutral (EN) computing aims to operate sensor nodes with at least a certain performance level over a period of time. Energy-neutrality can be described as the following equation:

$$E_{min} \leq E_{t_0} + \int_{t_0}^{t_0+\Delta t} [P_h(t) - P_c(t)]dt \leq E_{max} \tag{2.1}$$

where $P_h(t)$ and $P_c(t)$ are the harvested and consumed power at time t, t_0 is the time when EN computing is meant to start, Δt is the length of period during which EN conditions are achieved, E_{t_0} is the initial available energy in energy storage at time t_0, E_{min} is the minimum amount of stored energy below which the system cannot sustain (typically due to insufficient supply voltage), and E_{max} is the maximum capacity of energy storage beyond which the harvested energy is wasted. $P_c(t)$ includes the power consumption of the whole system, such as the MCU, peripherals, power conversion circuit, and the power leakage of energy storage. $P_h(t)$ is the harvested power after conversion.

EN devices are typically powered by solar cells [61], and Δt is typically 24 hours or one year to suit the period of the solar energy source. In order to achieve energy neutrality over such a long term, sufficient amount of energy storage, typically in the form of rechargeable batteries, is required to smooth out the large temporal variations of harvested power. The capacity of the energy storage is determined by how long the system tries to maintain a stable performance as larger energy storage tolerates more energy differences. In general, the length of Δt and the difference between P_h and P_c determine how much storage is required, and on the other hand, the capacity of energy storage limits how long Δt can be.

In order to ensure that the system works uninterruptedly by managing the stored energy (the middle term in Equation 2.1) between E_{min} and E_{max}, EN computing dynamically adapts system performance and power consumption over the period Δt. Typical adapting techniques include adjusting workload duty cycles and participation in network activity [59].

Kansal et al. [9] illustrate a preliminary power management algorithm by which the incoming energy is estimated by an Exponentially Weighted Moving Average (EWMA) of the past recorded slots of harvested energy, and the system tries to exploit the harvested energy by scaling its duty cycles. Vigorito et al. [62] introduce a Linear-Quadratic Tracking (LQT) approach to scale duty cycles based on the current battery level, and as evaluated in its datasets, mean duty cycle is improved by 6-32% and duty-cycle variation is reduced by 6-69% compared to [9], which means the system works with a more a stable performance. In [11], a Proportional-Integral-Derivative (PID) controller is used for monitoring and stabilizing the voltage of a supercapacitor-based energy storage, and hence, the storage level of this system. While these approaches achieve satisfactory energy neutrality for the magnitude of hours, they all show a latency when responding to the harvested power, and high variance of duty cycles when adapting to a new power trace. Additionally, approaches in [62] and [9] rely on an accurate estimating algorithm to detect the remaining battery energy, which is vulnerable to deployed time and temperature.

In [63], a prediction algorithm for solar energy named Weather-Conditioned Moving Average (WCMA) is presented, in which both the current and the past weather data are taken into account to achieve higher accuracy than EWMA methods. It is reported by the authors that the average prediction error is improved from 28.6% in EWMA to 9.8% in WCMA over a test duration of 45 days, but it is unclear in the article that how to harness this prediction to improve the system performance. Similarly, weather forecast is adopted in [50], by which the authors build a model to approximate the available solar and wind energy. Although these two methods based on weather data provide high prediction accuracy, the network overheads for receiving these data are not presented, and how to fully utilize this daily prediction is still a problem.

Different from the aforementioned daily EN operations, a long-term annual power management based on duty-cycling is presented in [10] to achieve annual energy neutrality. The authors use an adjustment factor, which is dynamically calculated from the historical windows, to modify the design-time energy prediction model to a more realistic model, and determine its performance level accordingly. However, this algorithm is only tested in simulation instead of practical experiments. Moreover, for such a long-term EN operation, a large battery is required, but the battery deterioration is ignored in their analysis.

In [12], a task scheduling algorithm for optimising the performance of an energy harvesting system (typically based on PV harvesters) is exhibited. Given a predicted power trace, storage bounds, energy consumption of tasks and quality of tasks, the proposed scheduling algorithm is proved to be able to find the optimal scheduling in a pseudo-polynomial time which leads to the maximum sensing quality. While this algorithm provides an ideal solution for power management, it requires that the energy source should be predictable with high accuracy, and the energy cost and quality

of each task should be defined at design time. The first requirement almost constrains this algorithm within the cooperation of solar energy. The second requirement is hard to achieve since a) in practice the energy consumption of tasks may change due to temperature and dynamic data amount [64] and b) the energy cost of a system includes many elements other than the energy consumption of computing tasks.

EN computing efficiently utilizes energy and maintains system performance, ensure reliability and periodic task execution despite variable harvesting power input. However, in almost all energy neutral approaches reviewed above, a large energy storage, i.e. a rechargeable battery, is in need in order to buffer temporal energy variations. The usage of batteries poses sustainability challenges due to the limited lifetime and pollution issues. Recent research develops intermittent computing and power-neutral computing, which minimise the need for energy storage. The next two sections (Section 2.4 and Section 2.5) review the methodologies of these two research topics.

2.4 Intermittent Computing

Energy harvesting provides an autonomous power supply for wireless sensor nodes as an alternative of battery power. However, with small storage, energy harvesting systems inevitably suffer from frequent power outages, which affect forward execution of programs. Intermittent computing (IC, also known as transient computing) aims to maintain forward execution and computation correctness through power failures [26]. Intermittent execution spans its execution and intermittently computes over power outages, while conventional execution restarts after power interrupts. A typical characteristic of an IC system is that it starts executing whenever there is power available and suspends during power outages; after power recovery, it can continue its prior task correctly instead of restarting from the beginning of a program.

Due to different design considerations, the methodologies in IC varies in a wide spectrum [65]. These methodologies include saving snapshots of system state to non-volatile memory (NVM), breaking down execution into small tasks, hardware circuits for suspend and restore operations, etc. The existing IC approaches can be classified into four types: checkpointing, reactive IC, task-based IC, and non-volatile processors (NVP). The following part of this section explains the each methodology one by one, as well as their works and current research progress.

2.4.1 Checkpointing IC

Checkpointing IC inserts checkpoints into code at compile time. When a checkpoint is called, the system checks the current available energy amount. If this amount is less

than a predefined threshold, which indicates the available energy may not be enough to sustain execution, a snapshot saving function is called at this checkpoint. To save a snapshot of the system computing state, the system copy current stacks and heaps, local and global variables, general registers, the stack pointer, and the program counter, into the NVM. A checkpointing system continuously operates until it encounters power outages, where the supply voltage is less than the minimum operating voltage of the systems. After the supply voltage recovers, the system restore its state from the last checkpoint, and hence, continue its execution from that checkpoint.

Mementos [26] first provides a checkpointing solution in which checkpoints are planned at compile time. Mementos includes three strategies of placing checkpoints, which are placing at every loop, placing at every function call, and an auxiliary timer delay to determine the minimum cycle between two adjacent checkpoints. Additionally, programmers can also insert or delete checkpoints manually as a custom option. Two NVM blocks are used and snapshots are saved to the two blocks alternately, so there is always at least one available and complete snapshot even if the energy is depleted during saving a new snapshot. One significant shortcoming of Mementos is the instrumenting strategy: with the different sizes of loops and functions, the granularity of checkpoints can be either too small, which introduces high run-time overheads, or too large, which leads to non-termination where the execution can never get to the next checkpoint. It is a concern in Mementos that how to set the voltage threshold which triggers saving snapshot. Setting this too high leads to redundant snapshots, while setting this too low leads to the failure of saving snapshots and cannot guarantee forward progress.

HarvOS [66] is proposed to improve the strategies of inserting checkpoints in Mementos. HarvOS analyses the control-flow graph of a program and splits it into sub-graphs with a checkpoint inserted for each sub-graph. To reduce the number of checkpoints compared to Mementos, the size of sub-graphs is set close to the worst-case number of useful cycles the MCU can execute until the next checkpoint. To reduce the size of snapshots, the RAM usage in each sub-graph is analysed and the checkpoint is placed at the point with the least RAM usage. HarvOS claims to reduce 68% checkpoints on average compared to Mementos.

Chinchilla [67] proposes a checkpointing tool which automatically overprovisions checkpoints at compile time and adaptively eliminates unnecessary checkpoints at run time. Compared to Mementos and HarvOS, Chinchilla relieves the programming efforts on manually inserting checkpoints while still achieves an efficient number of checkpoints at run time.

An advantage of the checkpointing method is the size of a specific snapshot can be estimated from the program execution flow to find a smaller snapshot [66]. However,

there are still two significant challenges remaining unsolved in checkpointing methods: idempotency violation and non-termination.

An execution is idempotent if it can be repeated while maintaining the same result [27]. Non-idempotent actions include I/O operations and NVM writes, which are fairly common in IC applications. Repeating non-idempotent actions can leads to undesired results, so these non-idempotent actions should be executed only once. Checkpointing systems repeat executing the code between two adjacent checkpoints, and hence, cause non-idempotency. Current compile-time checkpointing methods as listed above are not able to ensure idempotency.

Non-termination in checkpointing systems exhibits when the energy consumption of execution between two checkpoints is too much that the system energy buffer and harvested supply cannot sustain. Non-termination typically happens when the instrumentation strategy of checkpoints ignores the size of the energy buffer, as in Mementos. HarvOS and Chinchilla manage to mitigate non-termination, but they cannot eliminate this problem as they cannot dynamically insert checkpoints at run time according to varying environmental sources.

2.4.2 Reactive IC

Instead of instrumenting checkpoints at compile time, reactive IC does not set predefined checkpoints but save snapshots at run time when the supply voltage is detected to be lower than a threshold that indicates an imminent power failure. Therefore, the snapshot saving operations is only invoked when there is an indication of an imminent power outage, i.e. a low supply voltage. Also, after saving a snapshot, a reactive IC system suspends its execution and enter a low-power mode, rather than continues execution until a power outage as checkpointing systems do. When the voltage supply recovers above a restore threshold, the system either restores the last snapshot if the system reboots, or just continues execution if the system comes back from the low-power mode.

Hibernus [68] saves only one snapshot before a power interruption and then enter the sleep mode. Two fixed voltage thresholds, V_H and V_R, are predefined for hibernation (save a snapshot and sleep) and restoring a snapshot. An on-chip voltage comparator and an on-chip voltage reference generator are used for monitoring the supply voltage and triggering hibernation when the supply voltage drops to V_H or restoration when the supply voltage recovers to V_R. A voltage trace is shown in Figure 2.5 to explain Hibernus behaviours, and this trace is representative for a reactive IC system behaviours. To adapt thresholds to variable energy sources, Hibernus++ [69] implements dynamic self-calibration for suspend and restore thresholds by executing a hibernation test. By using adaptive thresholds instead of fixed thresholds as in Hibernus,

Hibernus++ makes itself compatible with a variety of energy sources. Compared to Hibernus, Hibernus++ improves application execution time by reducing the overheads of suspend and restore operations.

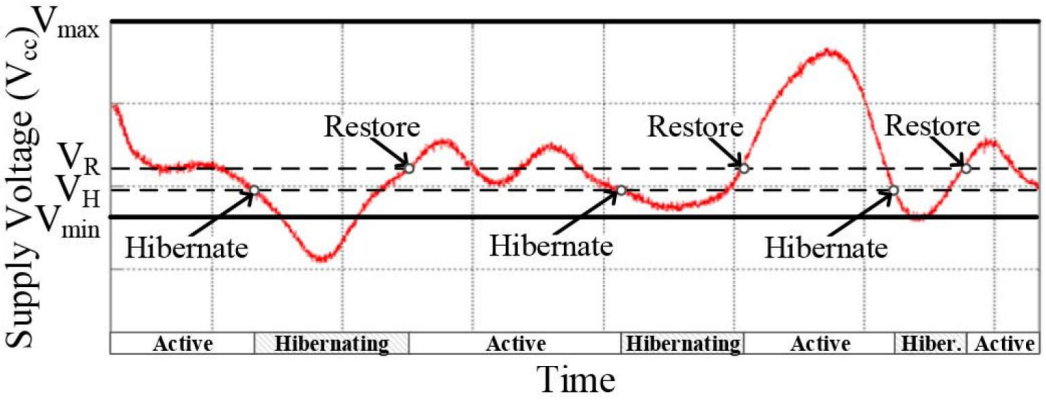


FIGURE 2.5: Voltage trace with hibernation and restoration points in Hibernus (taken from [68]).

Quickrecall [70] is a similar approach to Hibernus except replacing RAM with NVM, so that all the run-time volatile data become non-volatile and only registers are necessary to be saved in a snapshot. An external voltage comparator detects a triggering voltage V_{trig} to back up only peripherals and registers. Compared to the voltage thresholds in Mementos and Hibernus, V_{trig} in Quickrecall is lower since the reduced energy and time overheads for saving and restoring a snapshot. However, using NVM as RAM may lead to the higher cost of NVM accesses. A comparison between Hibernus and Quickrecall is presented in [71], showing that Quickrecall performs worse when the frequency of power interrupts is low as the NVM consumes more than volatile RAM, and performs better when the frequency of power interrupts is high as the overheads of saving snapshots are much lower.

Reactive IC methods only save snapshots when power failure is imminent, and hence, reduce the number of snapshots compared to checkpointing methods. Also, reactive IC avoids code re-execution by suspending execution after saving a snapshot, and hence, ensures idempotency.

The RAM usage varies at run time, so the size of snapshots in reactive IC also varies throughout code execution. To circumvent this issue, Hibernus saves the entire RAM in each snapshot while Quickrecall does not use RAM at all. Comparing Hibernus to checkpointing methods, the overheads of saving snapshots in Hibernus is larger as checkpointing methods can avoid saving large snapshots by analysing the program. Such high saving overheads becomes significant when the frequency of power outages increases. Increasing the size of energy storage in reactive IC should be helpful to mitigate frequent snapshot taking because the increased energy storage can filter the variations of supply voltage and avoid frequent voltage drops.

2.4.3 Harvest-Store-Use IC

Harvest-store-use IC systems perform a complete task in one consecutive period when the harvested energy in energy storage is enough. A complete task typically includes sensing, processing, and transmitting actions. In order to sustain a successful task execution, the required capacity of energy storage is larger than the minimum required storage in other IC methodologies. Harvest-store-use systems need to calibrate the energy consumption of the task at design time and set an energy threshold to trigger execution based on that energy consumption. When the threshold is reached, which means there is enough energy for a task, the system performs one task and sleeps until the next threshold trigger. As shown in Figure 2.6, the behaviours of the amount of stored energy can be seen as alternating in turn between two states: the collecting state and the executing state.

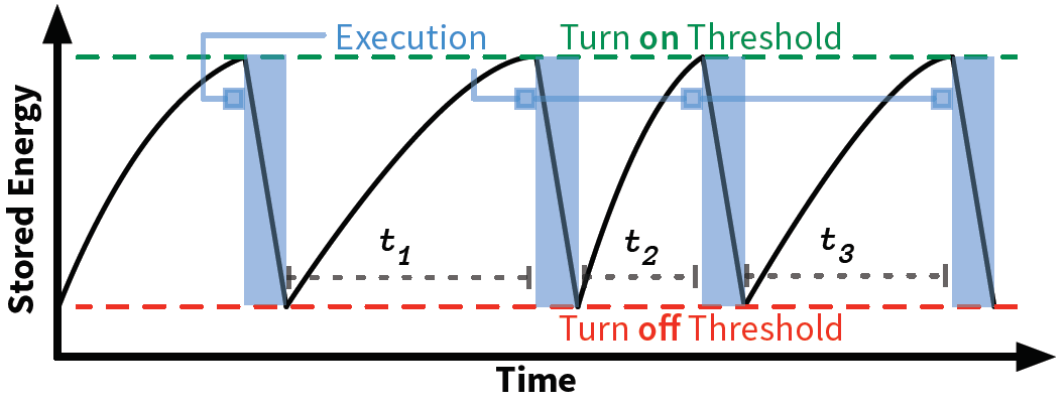


FIGURE 2.6: Harvest-Store-Use execution (taken from [72]).

Monjolo [25] is an early design following the harvest-store-use pattern. Monjolo presents a home power meter, whereby a current transformer is installed around the main power cable and provides the energy for this metering system. When the energy stored in a 500μF capacitor reaches a predefined amount, the system transmits a data packet. Another wireless receiver keeps collecting these packets and approximates the power of the main cable based on the receiving frequency of packets. Such a system contains little sensing and processing work on the transmitting node, and instead, it treats the intensity of energy sources as the sensing data, and processes this translation of data on the receiving node which is powered stably.

WISPCam [49] is a wireless camera that obtains energy from an RF harvester. The harvested energy is stored in a 6mF supercapacitor and the data (photos) are saved in NVM. Once the energy is sufficient for taking one photo, the system starts execution and depletes the energy for taking a picture and data transmission.

Similarly, Dynamic Energy Burst Scaling (DEBS) [60] also wakes up and executes tasks when there is enough energy in the 80μF capacitor. The major difference between DEBS and the above two approaches is DEBS can adjust the energy thresholds dynamically

for a set of different tasks and generates energy bursts according to which task is in need.

Harvest-store-use paradigms are suitable for occasions where the harvested power is too weak to support the power consumption of any normal execution (other IC methodologies may quickly deplete energy storage and make little progress). Also, harvest-store-use methods circumvent the idempotency issues by complete tasks in one burst. However, this pattern is task-based, which means its operation is limited to one or several fixed energy-defined tasks and also relies on high-quality design-time profiling of tasks.

2.4.4 Task-based IC

Task-based IC decomposes a program into a series of atomic tasks, which only deliver non-volatile results after all operations in a task are completed [27]. Task-based IC is achieved by programming and execution models, which aim to ensure NVM consistency and idempotency. In such models, accesses to NVM and I/O operations are carefully managed to prevent idempotent violations. To ensure idempotency, the program control flow is divided by task boundaries, and the communication between tasks is enabled by reading or writing NVM data on those boundaries. To avoid non-termination, the maximum size of one task is limited by the capacity of energy storage. Therefore, task-based IC can be seen as a rigorously-organized and fine-grained checkpointing method, which eliminates the the non-termination and idempotency problems in checkpointing IC. Task-based systems feature with fast suspend and restore operations because only the runtime and the current task should be versioned and restored through power outages [65].

DINO [27] proposes the first task-based IC programming and execution model, illustrating the task-based idea and providing a basic groundwork. DINO implements the programming and execution model on the LLVM compiler for C code, with program libraries and compiler passes. Chain [73] improves DINO data flows with ”Channels”, which is dedicated to manage non-volatile data, guaranteeing the correctness on applications with both idempotent and non-idempotent code. Alpaca [74] introduces data privatization which reduces memory usage compared to Chain.

A main drawback of DINO, Chain, and Alpaca is they require great programming efforts for programmers to understand the implemented libraries and redesign a program according to the task-based concept. A recent work, CleanCut [75], proposes an auxiliary tool to check and automatically decompose the non-terminating tasks (the energy consumption of which exceeds the capacity of system energy storage).

Also, like checkpointing IC, task-based IC inevitably involves re-execution. Alpaca, the state-of-the-art task-based approach, reports a run time overhead of 1.3-3.6x compared to plain C code given constant power supply.

2.4.5 Non-Volatile Processors

Non-Volatile Processors (NVPs) incorporate automatic backup and restore hardware within the chips. A comparison of memory architecture between traditional processors and NVPs is shown in Figure 2.7. The traditional volatile elements are replaced with non-volatile elements to achieve efficient backup and restore operations with a faster speed and lower energy consumption than the conventional memory architecture. To be specific, the registers and cache are equipped with built-in additional non-volatile backup and restore circuits, so that when the supply power is going to disappear, the computing state can be saved locally just beside the elements, rather than being copied out into an external NVM. It is reported that the backup and restore speed of NVPs can be 2-4$\times$ magnitudes faster than the state-of-the-art NVM based commercial processors [76].

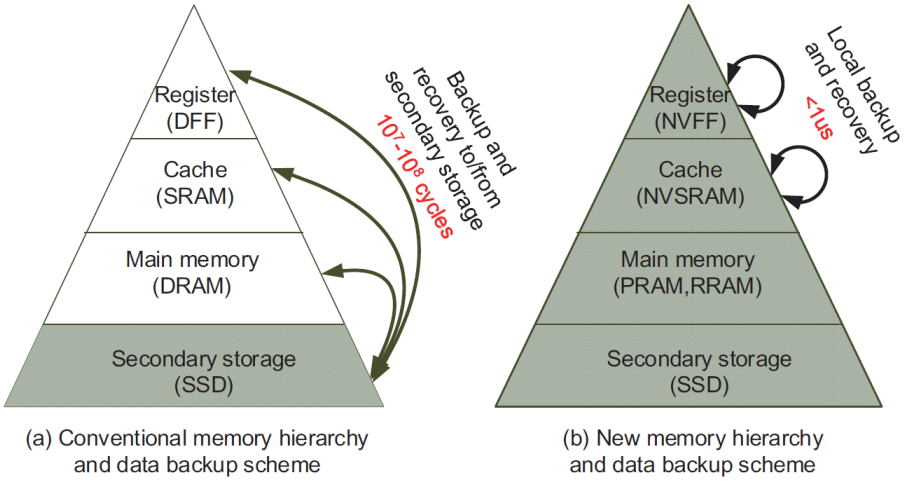


FIGURE 2.7: Memory architectures of traditional processors and NVPs (taken from [76]).

Wang et al. [77] present a preliminary NVP with 3μs backup time and 7μs restore time, which enables the processor to operate safely under a 20 kHz square wave of power. As a comparison, the existing MCUs in TI MSP430 family can only achieve 212μs and 310μs for saving and restoring states respectively. Su et al. [78] extend the backup and restore time overheads to a system level, presenting a NVP with 46μs system-level wake-up time and 14μs system-level sleep time. Liu et al. [28] integrate a NVP into a system-on-chip with independent backup and restore circuits for peripherals.

NVP-based research follows with the development of NVP hardware. Ma et al. [79] examine the performance and energy consumption of several types of NVPs with different ambient sources, providing a guideline for NVPs selection. The concept of "Incidental Computing" based on NVPs is proposed in [80] to improve the forward progress under unstable power supply, and also provides an evaluation of performance on NVPs. Essentially, it pays more attention to processing forward data in need than the buffered historical data from recovery, but an incidental recomputing on the historical data is performed when there is abundant energy. It is reported that this approach outperforms the existing save-and-use computing scheme by 2.2-5$\times$ in the simulation with respect to an image processing speed, and also the forward progress is improved by 4.28$\times$ on average over a basic NVP.

NVPs perform well in terms of the response to power intermittency, but the research on how to deliver better forward progress with NVPs is limited. Dynamic Voltage and Frequency Scaling (DVFS) can be a potentially applicable solution [81]. In a traditional NVP, the small buffering capacitor tends to be either charged to be full or depleted rapidly and frequently [82]. This behaviour accounts for a large part of backup and recovery overheads, so power management based on NVP is in need.

2.5 Power-Neutral Computing

While IC aims to ensure forward execution despite frequent power outages, energy harvesters may also generate more power than systems can consume when ambient sources are sufficient. Such excessive energy is wasted if not stored for later usage or consumed immediately.

2.5.1 Principles of Operations

Power-neutral (PN) computing aims to manage power without additional storage or with only a very limited amount of storage which can only sustain its system for milliseconds. In principle, power-neutral computing is a special case of energy neutral computing when Δt in Equation 2.1 is equal (or close) to zero. Technically, PN computing scales the instantaneous system power consumption to match the instantaneous harvested power with theoretically zero storage (in other words, energy neutrality is met instantaneously). PN operations can be translated into the following expressions:

$$P_h(t) = P_c(t) \tag{2.2}$$

$$where \quad t \in \{t | V_{cc}(t) \geq V_{min}\} \tag{2.3}$$

where V_{cc} is the input voltage of the computing load, and V_{min} is the minimum voltage required for the system to operate. Equation 2.2 describes the methodology of power neutrality (dynamic and instantaneous power adaptation). Equation 2.3 limits the requirement for power neutrality that the system should be powered and active to make reactions of performance scaling. This requirement may change according to different system designs, but for contemporary computing and sensing loads, this is determined by the supply voltage.

Given a very limited amount of storage and a range of scalable performance and power consumption, PN computing scales down performance if P_h is lower than P_c, such that V_{cc} remains stable, which extends execution time and avoids suspend and restore operations. On the other hand, PN computing scales up performance if P_h is higher than P_c, such that the excessive harvested energy is immediately consumed on useful work rather than wasted.

In practice, however, there does not exist a system that can adjust its power consumption instantaneously to the harvested power without any overheads. Any performance scaling costs a small amount of time and energy overheads, which a system cannot afford without any energy storage. Therefore, a minimum storage is still required, normally in the form of decoupling or parasitic capacitance, to provide a small but sufficient amount of energy for scaling performance and adapting power consumption.

In order to achieve power neutrality, a system has to adapt its performance and hence power consumption. Performance scaling can be achieved by hardware controlling, such as Dynamic Frequency Scaling (DFS) [30], Dynamic Voltage and Frequency Scaling (DVFS) [31], or switching on/off load elements [31, 83] (also known as Dynamic Power Management, DPM [84] or hot-plugging). Apart from these achieved methods, duty-cycle scaling and task scheduling are also choices for changing performance and consumption, though they have not been implemented in current research yet.

2.5.2 Recent Approaches

The concept of PN computing is proposed in [30] and implemented on a Texas Instrument MSP430FR5739 MCU without an external energy buffer. As shown in Figure 2.8, the executing load is directly connected to a regulated energy harvesting source. The control scheme in [30] utilizes DFS with a voltage feedback. Specifically, two voltage thresholds, V_{dec} and V_{inc}, are set for detecting voltage variance caused by power inequality and then scaling performance accordingly. In order to respond fast to power difference, the capacitance is reduced to 19μF, which is only the parasitic and on-board decoupling capacitance. When $P_h(t) > P_c(t)$ and the operating voltage V_{cc} increases rapidly due to the small capacitance and reaches V_{inc}, the MCU increases its operating frequency resulting in faster computing speed and higher power consumption, and

also increases the thresholds between which the new voltage value is contained; and vice versa, a reverse procedure is executed for $P_h(t) < P_c(t)$. In a word, this control scheme is trying to make the operating voltage stable around a desired value so that $P_h(t)$ equals $P_c(t)$ approximately.

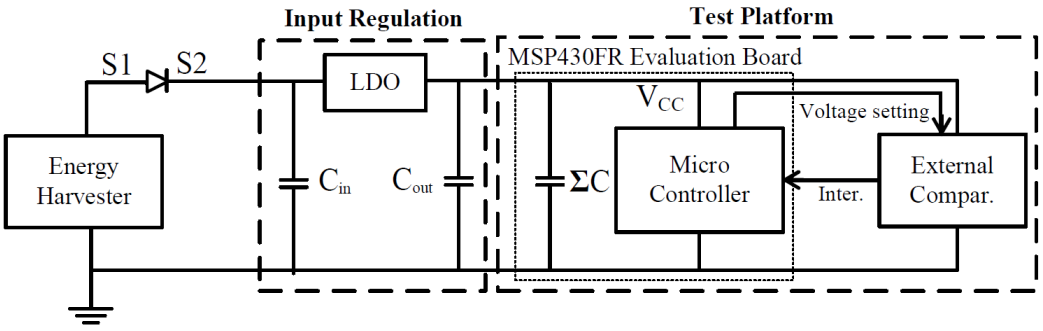


FIGURE 2.8: Architecture of an example power neutral system based on TI MSP430FR platform (taken from [30]).

A similar control scheme is adopted in [31] where the platform is an MP-SoC adopting DVFS and DPM, which leads to higher performance, higher power consumption, and more operating points than the MCU in [30]. A 47mF supercapacitor is used for safely overcoming performance switching where the power consumption of the board is normally above 2W. As an illustration for how to scale performance by DVFS and DPM, Figure 2.9 presents an example application profile of 'power consumption vs performance' when DVFS and DPM applied on a heterogeneous multi-processor system-on-chip (MP-SoC) platform. The SoC used in this platform is the Samsung Exynos5422 big.LITTLE SoC with four 'big' high-performance A15 cores and four 'LITTLE' low-power A7 cores. In this case, the performance refers to the speed of executing this application for one time and is proportional to the operating frequency under a certain core configuration. As shown in the figure, each performance level (a pair of frequency and core status, also named as an operating point) requires a certain power consumption. At run-time, the system dynamically switch its performance among these operating points so as to timely match $P_c(t)$ with $P_h(t)$.

There are three advantages in this kind of PN control scheme. First, the voltage is stabilized so it can offset ephemeral power drops which cause insufficient voltage supply and power failures, and therefore the lifetime increases (e.g. reported by 4-88% in [30]). Second, as power neutral computing eliminates many elements that required in EN systems, such as large energy storage, power converters and MPPT units, the size and cost of devices is reduced and the number of power consumption components also decreases. Third, if powered by a solar panel and the operating voltage range encompasses the MPP of the given solar panel, the system embraces an intrinsic MPPT characteristic as it can stabilize the voltage around a target value.

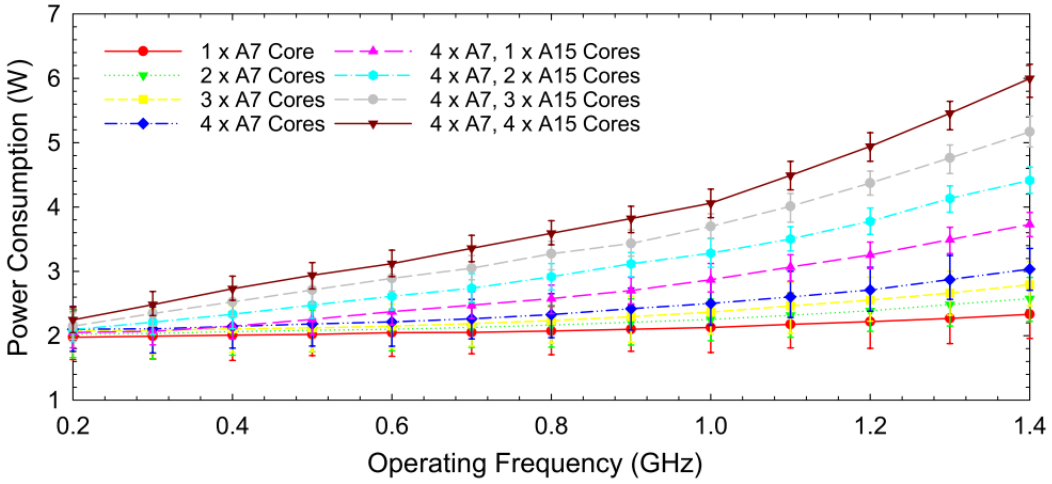


FIGURE 2.9: Board power consumption of ODROID XU4 vs operating frequency and core configurations, running CPU intensive application Raytrace (taken from [31]).

Similarly, Wang et al. [83] propose a storage-less and converter-less approach which can be classified as a power neutral system. In this design, a 47μF bulk capacitor is equipped with a 3.29mW non-volatile MCU and up to 16.5mW peripherals. This capacitor is also small enough compared to the 19μF capacitance operating with an up to 3 mW MCU in [30]. An external MPPT controlling element dynamically adjusts the power duty-cycle for the non-volatile load in order to match the harvested current and the consumed current, and hence power neutrality is met.

One disadvantage of power neutral computing is that it has to passively scale its power consumption as well as its performance, causing large variations in performance. However, this might not be good in terms of the overall forward progress. In the next chapter, a preliminary analyse is explained about how the forward progress is improved when the capacitor size is increased, while not violating the merits of PN computing.

2.6 Summary

This chapter introduces a background of energy harvesting techniques, summarises the evolution of energy storage used in energy harvesting computing, and reviews the existing methodologies of battery-less energy harvesting computing.

EN computing emphasizes the continuous activity of devices over a long-term duration (e.g. several days, one year) by buffering harvested energy in large energy storage and adapting energy consumption "reluctantly". However, large energy buffers, usually in the form of batteries or large supercapacitors, are demanded for EN operations, whereas such large energy storage limits device lifespans, increases the cost, mass, dimensions of devices, and bring pollution and maintenance issues. This contradicts the design requirements of ubiquitous sensor deployments.

To circumvent the limitations in EN computing, intermittent computing is recently developed. Intermittent computing continues computation after the supply fails rather than restarts from the beginning of programs. Hence, intermittent computing devices can achieve forward execution despite frequent power failures with only minimum storage (e.g. a decoupling capacitor) to secure successful saving and restoring operations of computing states between volatile and non-volatile components. Based on intermittent computing, PN computing introduces run-time performance adaptation to match power consumption with harvested power, such that the number of saving and restoring operations can be reduced and application execution speed is increased.

However, with minimised storage, an intermittent computing device has to frequently wake up, execute shortly, and halt when the harvested power is less than the load power consumption, consuming much energy in managing system states. As for PN computing, volatile power from environment results in significant performance variations, which then cause performance loss. The remaining part of this thesis reports a study on how to mitigate these two problems and improve system execution speed by adding a small amount of energy storage without significantly affect device dimensions.

Chapter 3

Effect of Energy Storage Sizing on Intermittent Computing System Performance

3.1 Motivation

Internet of Things (IoT) devices are becoming ubiquitous, with forecasts of hundreds of billions being installed in the near future [85]. They are conventionally battery-powered, thus have constrained lifespans, necessitating inconvenient periodic battery replacement. Energy-harvesting is a potential solution. Environmentally harvested power is, however, intrinsically variable and intermittent [86]. Traditionally, large energy storage devices such as rechargeable batteries or supercapacitors are used to smooth out supply variability [87]. Unfortunately, these increase cost and device dimensions [88], raise pollution concerns [89], and still limit lifespans [90].

Recently, *intermittent computing systems* (ICSs) have been proposed as an alternative [91]. Instead of using large energy storage devices to sustain execution, they tolerate power interruptions by saving the state of the system into non-volatile memory (NVM) so that computation can continue when power is restored. They may save this state (e.g. CPU registers and RAM contents) either *statically* at pre-defined points, or *reactively* by detecting when the supply is about to fail [91].

Static approaches save state at points determined at design or compile time, either by inserting checkpoints [92, 93] or decomposing a program into atomic tasks[1] [94, 95].

[1] Atomic operations in ICSs denote operations that should be completed in one continuous period. If an atomic operation is interrupted by a power failure, it should be re-executed rather than resumed. Examples of atomic operations include saving and restoring volatile state, transmitting and receiving packets, and sampling sequences of data from sensors.

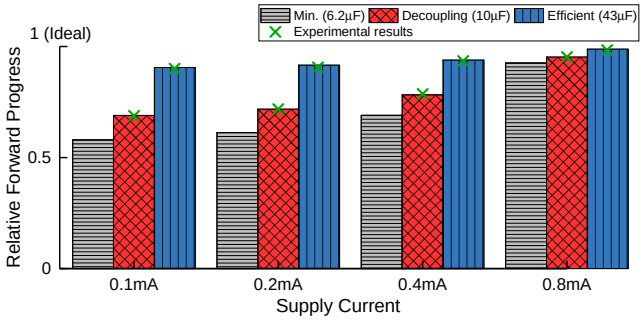


FIGURE 3.1: The relationship between energy storage capacitance and ICS forward progress, for various supply currents.

After a power interruption, progress rolls back and resumes from the last saved checkpoint or task boundary. This can introduce issues such as violation of data memory consistency, along with wasting energy on lost and re-executed progress.

Conversely, reactive approaches monitor the supply voltage and only save state when it falls below a threshold [96–98], which is set high enough to reliably save state even with a total and immediate drop-off in harvested energy. They then enter a low-power mode, in many cases preserving their volatile memory and avoiding re-execution. These typically make more forward progress than static approaches, e.g. a 2.5× mean computational speedup [99].

In ICSs, *forward progress* denotes the effective application progress, excluding re-executed progress, lost progress, and state-saving and -restoring operations [32]. The amount of forward progress directly determines application performance, e.g. program iteration rate or task completion time. In this paper, to allow fair comparison, we define normalized forward progress as *the ratio of the effective execution time to the total elapsed time*, without being restricted to a specific workload.

With the goal of minimizing device dimensions and interruption periods, most ICS approaches adopt a minimum amount of energy storage [96, 100–103]. This is typically just sufficient for the most energy-expensive atomic operation. However, our assertion is that this can be *inherently inefficient in terms of time and energy*. We show that a system with minimum energy storage frequently goes through a cycle of: wake up; restore state; execute program; save state; halt.

We propose that provisioning *slightly more* energy storage can prolong the operating cycles, reduce the frequency of interruptions, and hence improve forward progress. We show with modelled and experimental results that (Figure 3.1) using efficiently-sized energy storage capacitance (43 µF) achieves up to a 55% improvement in forward progress compared against using the theoretical minimum amount of capacitance (6.2 µF). This improvement is more significant with a weaker supply. However, the relationship between ICS energy storage capacitance and forward progress has not previously been defined. Also, current tools for ICSs (Section 3.2) are not practical for fast

estimation of forward progress in a long-term deployment, and lack a method of sizing energy storage to improve forward progress while moderating the physical size and interruption periods.

This paper presents an approach for sizing energy storage in ICSs, quantifying and trading off forward progress, capacitor volume, and interruption periods. The main contributions are:

- A reactive ICS model which accurately estimates forward progress; experimental validation shows a 0.5% mean error (Section 3.3).
- A model-based sizing approach that recommends appropriate energy storage capacitance in ICSs (Section 4.2).
- An exploration based on the model, where we analyze the energy storage sizing effect on forward progress, showing up to 65% forward progress improvement (Section 3.4).
- An evaluation of the impact of sizing in real-world conditions using real energy availability data (Section 4.3). This includes a cost function-based method for trading off parameters. In an example, this reduced capacitor volume and interruption periods by 83% and 91% respectively, while sacrificing 7% of forward progress.

The associated simulation tool, coded in C, is available open-source at *(link to be provided on publication)*.

3.2 Related Work

To explore forward progress of ICSs, simulation tools need to represent transient operation (timescales of µs-ms) as well as long-term overall performance (from days up to years).

Su *et al.* [104] modelled a dual-channel solar-powered nonvolatile sensor node, and Jackson *et al.* [105] provided a model to explore battery usage in ICSs. Both were configured for long-term simulations and large energy storage (from mF-scale supercapacitors to batteries), thus cannot respond to frequent power interruptions and accurately estimate forward progress when using minimized energy storage (e.g. 4.7 µF [102]).

In contrast, a set of fine-grained models have been proposed to accurately simulate the frequent micro-operations in ICSs. NVPsim [106] is a gem5-based simulator for nonvolatile processors. Fused [107] is a closed-loop simulator which allows interaction between power consumption, power supply, and forward progress. EH model [108]

TABLE 3.1: Model parameters of reactive ICS

Input Parameters	
I_{harv}	Energy harvester current supply
C	Energy storage capacitance
Configuration Parameters	
I_{exe}	Execution current draw
I_{lpm}	Low-power mode current draw
I_r	Restore current draw
I_s	Save current draw
I_{leak}	Leakage current draw
V_r	Restore voltage threshold
V_s	Save voltage threshold
T_r	Restore time overhead
T_s	Save time overhead
Output Parameter	
α_{exe}	Normalized forward progress

can compare a range of ICS approaches in a single active period with the same energy budget, quantifying forward progress by the energy spent on effective execution. These fine-grained models are inefficient for processing long-term energy data, especially when iterative tests are needed for various system configurations.

Besides models and simulators, hardware emulators of energy harvesters [109, 110] can provide repeatable power profiles recorded from energy harvesters for experimental comparisons. Though they provide practical results, hardware emulations are limited by hardware options and are generally impractical for performing long-term trials.

To address the above problem, we provide a reactive ICS model to estimate forward progress, as well as a simulation tool that enables fast exploration with long-term real-world environmental conditions. Further, we provide a sizing approach which recommends appropriate energy storage capacitance for deploying ICSs.

3.3 Reactive ICS Modelling

To facilitate the understanding and exploration of reactive ICSs, we present a model which outputs the normalized forward progress α_{exe} when powered from a constant current supply I_{harv}. Parameters of this model are listed in Table 3.1. The model assumes that all configuration parameters remain constant.

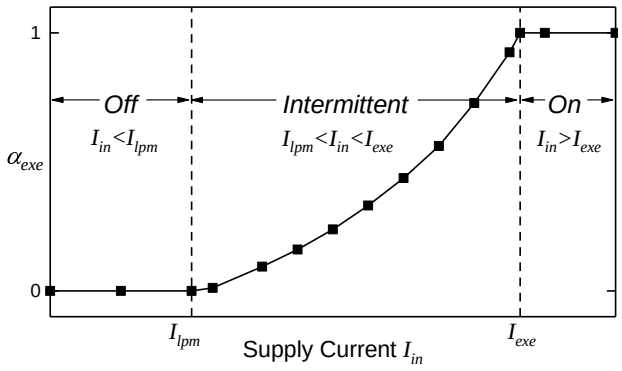


FIGURE 3.2: Operating modes of reactive ICSs, and achieved forward progress against supply current.

For brevity, I_{in} denotes the usable input current as expressed in (3.1). The effect of capacitor leakage current, I_{leak}, is discussed at the end of Section 3.3.2.

$$I_{in} = I_{harv} - I_{leak} \tag{3.1}$$

3.3.1 Operating Modes of Reactive ICS

The behavior of reactive ICSs can be classified into three operating modes depending on the supply current, as shown in Figure 3.2. These are differentiated by the relationship between input current I_{in} and the system's current draw in its low-power mode (LPM) or active modes, i.e. I_{lpm} and I_{exe}. We define the three modes as:

- *Off* mode: When $I_{in} < I_{lpm}$, the system stays inactive. The supply voltage V_{cc} cannot rise above the restore threshold V_r to wake the system and start execution. The LPM current I_{lpm} includes the consumption of voltage monitoring circuits and system idle current.
- *On* mode: When $I_{in} > I_{exe}$, the system executes constantly as the supply voltage V_{cc} never drops below V_s. V_{cc} grows until I_{in} and I_{exe} are in equilibrium, which may result from I_{in} decreasing due to poor impedance matching, or I_{exe} increasing due to either greater current draw at higher voltage or dissipation through overvoltage protection circuits.
- *Intermittent* mode: When $I_{lpm} < I_{in} < I_{exe}$, the system executes intermittently after $V_{cc} > V_r$ and before $V_{cc} < V_s$. V_{cc} can rise above V_r and the system starts execution. However, the stored energy is then consumed by the load as $I_{in} < I_{exe}$, causing V_{cc} to eventually drop below the save threshold V_s, where the system saves its state and enters LPM. The system stays in LPM until V_{cc} rises to V_r again and then resumes execution. In general, a higher I_{in} leads to more forward progress in this mode, but the exact relationship between I_{in} and forward progress requires further analysis.

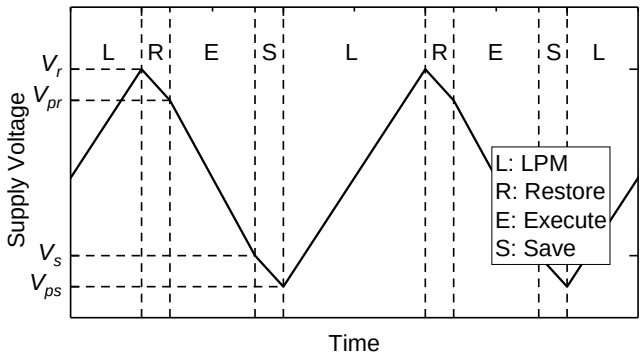


FIGURE 3.3: Operating cycles in the *Intermittent* mode.

3.3.2 Formulating Forward Progress

Next, we derive formulations to calculate α_{exe} from I_{in} and energy storage capacitance C. We then explore the effect of capacitor leakage on maximum forward progress.

In the *On* and *Off* modes, the normalized forward progress is trivial to find (simply 1 and 0 respectively). In the *Intermittent* mode, as shown in Figure 3.3, the system goes through four intervals in turn, i.e. charging, restoring, executing, and saving, with current consumption of I_{lpm}, I_r, I_{exe}, and I_s in each interval respectively. The normalized forward progress, i.e. effective execution time ratio, is indicated as T_{exe}/T_{cycle}, where T_{exe} is the time spent on effective execution in one operating cycle and T_{cycle} is the period of operating cycles. Hence, the forward progress given all supply levels is expressed as:

$$\alpha_{exe} = \begin{cases} 0 & , \ \mathit{Off}\,(I_{in} < I_{lpm}) \\ \dfrac{T_{exe}}{T_{cycle}} & , \ \mathit{Intermittent}\,(I_{lpm} < I_{in} < I_{exe}) \\ 1 & , \ \mathit{On}\,(I_{in} > I_{exe}) \end{cases} \tag{3.2}$$

In the following analysis, we focus on deriving T_{exe}/T_{cycle} in the *Intermittent* mode. Let V_{pr} (post-restore) and V_{ps} (post-save) denote the voltage after restoring and saving operations. V_{pr} and V_{ps} can be calculated as:

$$V_{pr} = V_r + \frac{T_r(I_{in} - I_r)}{C} \tag{3.3}$$

$$V_{ps} = V_s + \frac{T_s(I_{in} - I_s)}{C} \tag{3.4}$$

With (3.3), the time spent on effective execution T_{exe} in one operating cycle can be expressed as:

$$T_{exe} = \frac{C(V_{pr} - V_s)}{I_{exe} - I_{in}} \tag{3.5}$$

Analogously, with (3.4), the charging interval can be described as:

$$T_{charge} = \frac{C(V_r - V_{ps})}{I_{in} - I_{lpm}} \tag{3.6}$$

With (3.5) and (3.6), the period of an operating cycle is:

$$T_{cycle} = \quad T_{charge} + T_r + T_{exe} + T_s \tag{3.7}$$

Finally, combining (3.3)–(3.7), we obtain normalized forward progress α_{exe} in the *Intermittent* mode as:

$$\begin{aligned} \alpha_{exe} &= \frac{T_{exe}}{T_{cycle}} \\ &= \frac{\frac{C(V_r - V_s) + T_r(I_{in} - I_r)}{I_{exe} - I_{in}}}{\frac{C(V_r - V_s) + T_s(I_s - I_{lpm})}{I_{in} - I_{lpm}} + \frac{C(V_r - V_s) + T_r(I_{exe} - I_r)}{I_{exe} - I_{in}}} \end{aligned} \tag{3.8}$$

In the numerator T_{exe}, $C(V_r - V_s)$ represents the amount of charge in the capacitor available for restoring and executing. $T_r(I_{in} - I_r)$ represents the charge used by a restore operation. $I_{exe} - I_{in}$ is the rate of charge consumption from the energy storage during execution.

To explore the effect of energy storage on forward progress, we need to analyze $d\alpha_{exe}/dC$. Here, if we assume that I_{leak} remains constant, α_{exe} keeps increasing and approaches $(I_{in} - I_{lpm})/(I_{exe} - I_{lpm})$ when energy storage capacitance C increases. Defining $(I_{in} - I_{lpm})/(I_{exe} - I_{lpm})$ as α_{exe_ideal}, $\alpha_{exe} = \alpha_{exe\_ideal}$ is an ideal case, where restore and save overheads are absent.

In an electrolytic capacitor, however, I_{leak} typically increases with C with the following relationship [111]:

$$I_{leak} = kCV_{cc} \tag{3.9}$$

where k is a constant normally in a range 0.01 to 0.03 ($\frac{A}{F \cdot V}$). Combining (3.9) with (3.1), dI_{in}/dC is $-kV_{cc}$, meaning I_{in} decreases linearly as C increases. Thus, when C increases, α_{exe} keeps approaching α_{exe_ideal} while α_{exe_ideal} decreases. Hence, we believe that there is a capacitance value that leads to the maximum α_{exe} considering I_{leak} increases with C.

3.4 Exploration of Energy Storage Sizing

In this section, we configure the reactive ICS model presented in Section 3.3 to approximate a real ICS platform, and then present an exploration of the relationship between α_{exe} and C with respect to I_{harv} and volatile state size.

3.4.1 Model Configuration

3.4.1.1 Energy Storage

The energy storage is represented as an ideal capacitor with leakage current. Its terminal voltage is directly applied to the load, so is modelled as:

$$C\frac{dV_{cc}}{dt} = I_{harv} - I_{load} - I_{leak} \tag{3.10}$$

where I_{load} is the current consumption of the load. In this exploration, we refer to the empirical I_{leak} of AVX TAJ low-profile series tantalum capacitors, which depends on capacitance C, rated voltage V_{rated}, and terminal voltage V_{cc} [111]:

$$I_{leak} = 0.01\lambda C V_{rated} \quad (A) \tag{3.11}$$

where λ denotes the ratio of the actual current leakage at V_{cc} to the current leakage at V_{rated}, and λ is approximated as:

$$\lambda = 0.05 \times 20^{\frac{V_{cc}}{V_{rated}}} \tag{3.12}$$

We assume a typical load of $< 4.0\,\text{V}$ so, to minimize leakage, we select a device with $V_{rated} = 10\,\text{V}$ so as to operate between 25-40% of its rated voltage [111].

3.4.1.2 Intermittent Computing Load

The load parameters of current draws and time overheads, as listed in Table 3.2, were profiled with the experimental settings explained in Section 3.5.1. The current draw was profiled with experimental measurements at a range of supply voltages. The variation of I_{lpm} between V_{off} (1.8 V) and V_r (2.4 V) is 2%, and for I_{exe} between V_s (2.1 V) and 3.3 V is 1.5%. I_{exe} also has a run-time variation of 2.8% due to a variable memory access rate. We omit these minor variations and use the mean of I_{exe} and I_{lpm} in the model. I_r and I_s are measured at V_r and V_s respectively. Given the voltage thresholds and the current consumption, the minimum energy storage capacitance is 6.2 µF. This guarantees that a save and restore operation can complete even if the incoming supply current drops instantaneously to zero. The model parameters in Table 3.2 are given as an example, and can be changed for different load characteristics. For example, T_r and T_s can be tuned for different volatile state sizes.

TABLE 3.2: Profiled MCU parameters

Parameter	Value
I_{exe}	887 µA
I_{lpm}	26 µA
I_r	971 µA
I_s	811 µA
T_r	1.903 ms
T_s	1.880 ms

3.4.2 Sizing Energy Storage to Improve Forward Progress

3.4.2.1 Impact of Supply Current

Increasing energy storage capacitance above the minimum can improve forward progress by reducing the frequency of power interruptions, but this improvement may be offset by increased leakage. Figure 3.4 shows the relationship between forward progress and energy storage capacitance for a range of constant supply currents. Optimal capacitance values are shown for each current value.

The minimum capacitance (dashed line in Figure 3.4) is calculated to deliver correct operation even if the supply current instantaneously drops to zero. If it does not drop to zero, this means that correct operation could have continued even with a smaller capacitance, though designing a system in this way would be inadvisable owing to unpredictability of the supply. This property is illustrated in Figure 3.4, in the area on the left of the dashed line. It may be observed that, for each of the current values, there is a sudden drop-off towards zero forward progress. This illustrates the hazard of setting the capacitance too small: the stored energy is too low to allow a restore and save to be undertaken.

Typically, commercially-available capacitors have a $\pm$20% tolerance. The effect of this variation on maximum forward progress is shown to be negligible ($< 0.23\%$) in Figure 3.4. However, it must be pointed out that the effect would be much more pronounced if operating at the minimum capacitance as the variation of forward progress is larger with smaller capacitance values. Thus, it is recommended that a tolerance is considered when designing ICSs with minimum capacitance.

Figure 3.5 shows that an improvement in forward progress of up to 65% can be achieved when using the optimal capacitance instead of the minimum. However, it may not be desirable to set the capacitance solely for maximizing forward progress, because there are often trade-offs with other factors including increased interruption periods and dimensions. While a large improvement can be delivered with the optimal capacitance, as shown in Figure 3.5, 95% of this gain can still be obtained with significantly smaller

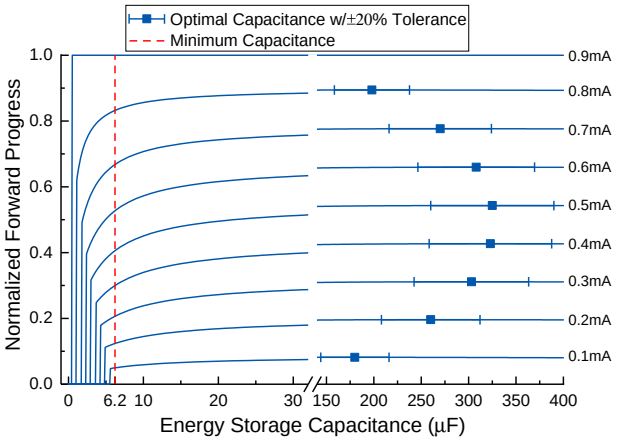


FIGURE 3.4: Forward progress against energy storage capacitance at different levels of constant supply current. Error bars around optimal points denote the impact of typical ±20% capacitance tolerance.

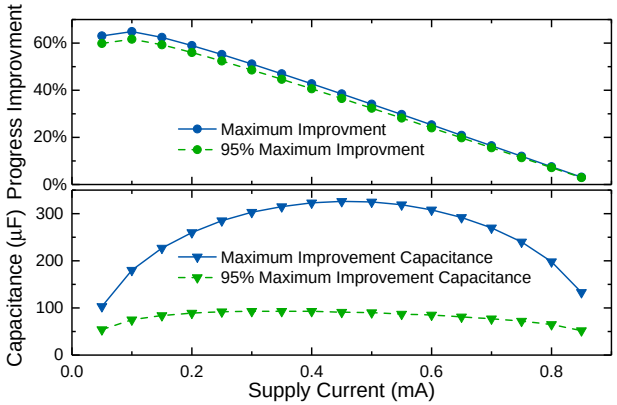


FIGURE 3.5: Maximum forward progress improvement by sizing energy storage given a spectrum of supply current (normalized by the minimum capacitance case), with the corresponding maximum and sub-maximum (95% of maximum) capacitance.

capacitances (mean 31% of the optimal value). For example, reducing from 325 µF to 90 µF gives 95% of the maximum improvement with a 0.5 mA supply.

3.4.2.2 Impact of Volatile State Size

The size of volatile state differs across applications with different amounts of RAM usage, and hence incurs varying time and energy overheads for restore and save operations. We measured time overheads of restore and save operations in the minimum case (64B register data and a 160B stack) and the maximum case (64B register data and a full 2048B RAM) respectively as shown in Table 3.3. As these time overheads are expected to be linear to the state size [112], the model can be tuned for various volatile state sizes by linearly scaling the profiled values.

An example of this is plotted in Figure 3.6. The forward progress improvement by sizing energy storage increases with the volatile state size, and the optimal capacitance

TABLE 3.3: Linear scaling range of volatile state size and restore/save time overheads

State Size (Registers + SRAM)	Restore Time	Save Time
64B + 160B (lower bound)	232 µs	208 µs
64B + 2048B (upper bound)	2.298 ms	2.274 ms

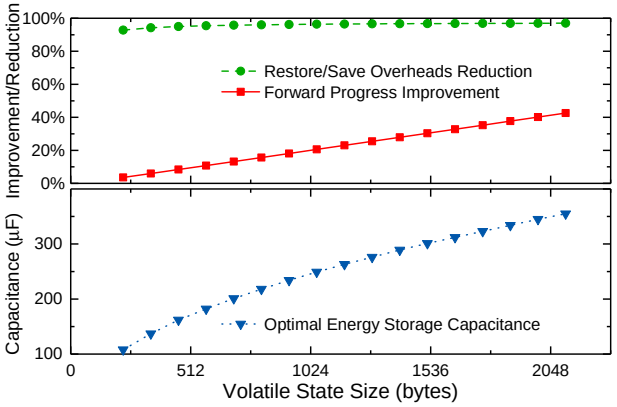


FIGURE 3.6: Impact of RAM usage (linear to restore/save overheads) on sizing energy storage with 0.4 mA current supply. Improvement and reduction are normalized by the minimum capacitance case.

grows accordingly. The improvement becomes insignificant when the volatile state size is small because the restore and save overheads are already negligible. For example, when the workload uses the least volatile state (the leftmost point), the maximum progress improvement is only 3.6% although the restore and save overheads are reduced by 93%.

3.5 Experimental Validation

3.5.1 Model Validation

We implemented and parameterized a reactive ICS [113] on a TI MSP430FR6989 microcontroller to validate our model. The on-board decoupling capacitance was measured as 10.0 µF, and hence was the minimum capacitance that could be tested. Further capacitance was added to provide extra energy storage. The parameters are profiled with the MCU running a Dijkstra path finding algorithm with 1696 B RAM usage at 8 MHz. The supply voltage monitoring circuits use the MCU's internal comparator and an external 3 MΩ voltage divider. The restore and save voltage thresholds are set as V_r = 2.4 V and V_s = 2.1 V respectively. The MCU shutdown voltage V_{off} is 1.8 V.

To validate the accuracy of our model, we powered the device with a range of supply currents ([0.1, 0.2, ..., 0.8]mA) to operate the device in *Intermittent* mode, and repeated

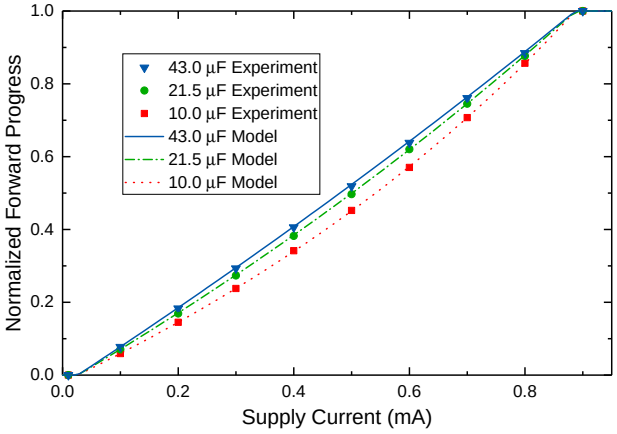


FIGURE 3.7: Model validation with experimental and modelled forward progress.

the tests with three energy storage capacities: a) 10.0 µF decoupling capacitance; b) 21.5 µF (11.5 µF added); c) 43.0 µF (33.0 µF added). We compared the actual forward progress against predictions generated from our model. As shown in Figure 3.7, the model-generated output matches closely with the experimental results with only 0.5% mean absolute percentage error.

The reactive ICS model presented in this section will be used to explore energy storage sizing effects in Section 3.4. In the next section, we extend this approach with a cost function for trading off forward progress against various design factors.

3.5.2 Validation of Sizing Effects

As previously shown in Fig.3.1, the efficiently-sized energy storage capacitance (43 µF) improves forward progress by up to 55% and 30% compared to the minimum and decoupling capacitance respectively. We notice that this improvement becomes significant when the supply current attenuates because the save and restore overheads consume a larger proportion of the available energy. Also, this achieves at least 90% of the ideal forward progress mentioned in Section 3.3.2. These results illustrate the importance of this technique, in particular for conditions where the supply current is low.

3.6 Summary and Discussion

Chapter 4

Energy Storage Sizing Approach for Deploying Intermittent Computing Systems

4.1 Motivation

4.2 Energy Storage Sizing Approach

We propose a sizing approach which recommends appropriate energy storage capacitance for an ICS, trading off forward progress against capacitor volume and interruption periods. We present a system model which accepts real long-term data on environmental energy conditions. The three inputs can be swept for design exploration, but we focus on energy storage in this paper. The model outputs forward progress, capacitor volume, and interruption periods (defined in Section 4.3.2). These are subsequently traded off in a cost function to obtain the appropriate energy storage capacitance. This process is summarized in Figure 4.1 with details explained as follows.

4.2.1 Input

A time trace of representative environmental energy conditions in the intended deployment location is provided as an input, along with the energy harvester size; for design exploration, these can optionally be changed to explore variations and scales of harvested power. A pre-defined set of energy storage capacitance values are swept through.

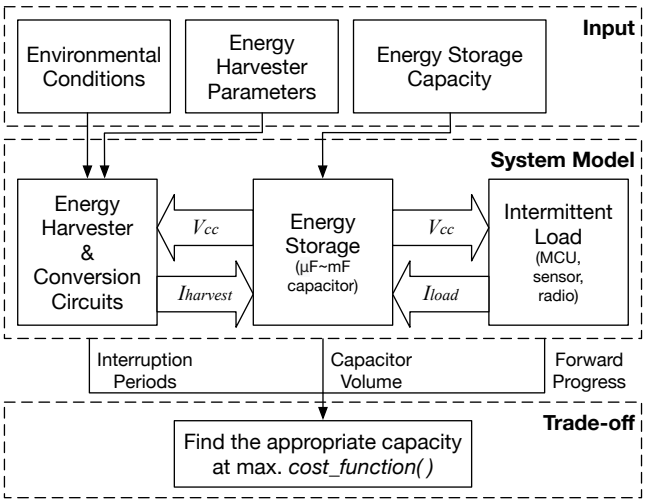


FIGURE 4.1: Structure of the proposed system model and sizing approach.

4.2.2 System Model

This contains three modules:

- *Energy Harvester and Conversion Circuits*: The energy harvester module transduces environmental energy into electricity. In ICSs, conversion circuits may simply be a diode to inhibit backflow of current. The energy harvester and conversion circuits can be modelled together as a module because they are usually coupled or integrated.
- *Energy Storage*: Energy storage in ICSs is usually in the form of a µF- to mF-scale capacitor. It must be sufficient to complete the most energy-expensive atomic operation, and may be formed only of the decoupling capacitor(s).
- *Intermittent Load*: Includes all the power consumers in an ICS, such as a microcontroller, sensors, and a radio.

The outputs from the system model are the interruption periods, capacitor volume, and forward progress.

4.2.3 Trade-off

The appropriate capacitance is then found through a cost function. This may trade off forward progress against capacitor volume and interruption periods.

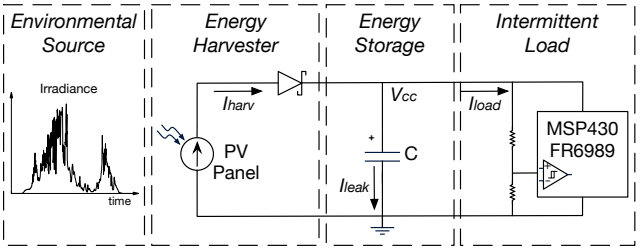


FIGURE 4.2: System model of a PV-based ICS.

TABLE 4.1: PV cell properties under a 1000 W/cm^2, AM-1.5 light source

Parameter	Value
Open-Circuit Voltage	0.89 V/cell
Short-Circuit Current	14.8 mA/cm^2
Maximum Power Voltage	0.65 V/cell
Maximum Power Current	12.1 mA/cm^2

4.3 Sizing under Real-World Energy Conditions

In this section, we model an ICS with a photovoltaic (PV) energy harvester to explore the energy storage sizing effect in real-world energy conditions, and demonstrate use of the proposed sizing approach.

4.3.1 Simulation Configuration

We integrate the validated reactive ICS model into a system model with a PV energy-harvesting supply as shown in Figure 4.2. The energy storage model and the intermittent load model are as presented in Section 3.4.

We use a converter-less supply circuit where only a Schottky diode is connected to the energy harvester output in order to prevent current backflow. The energy source conditions are imported from NREL outdoor solar irradiance data [114] and EnHANTs indoor irradiance data [115]. Four sets of light conditions are used to encompass different energy environments. To convert irradiance into harvested power, we adopt a PV cell model [116] which uses the parameters available in common datasheets, so it can easily be reconfigured to suit various devices. We refer to Panasonic Amorton glass type solar cells [117] for PV cell properties as shown in Table 4.1. We set four cells in series (with V_{oc} = 3.56V) to match the operating voltage of the MCU (maximum 3.6V), and model energy harvester sizing by scaling the cell area.

4.3.2 Exploration with Real-World Energy Source Conditions

In real-world deployments, ambient energy source conditions are dependent on time and location. The energy harvester and storage need to be sized to achieve the desired forward progress across the range of expected conditions.

4.3.2.1 Sizing the Energy Harvester

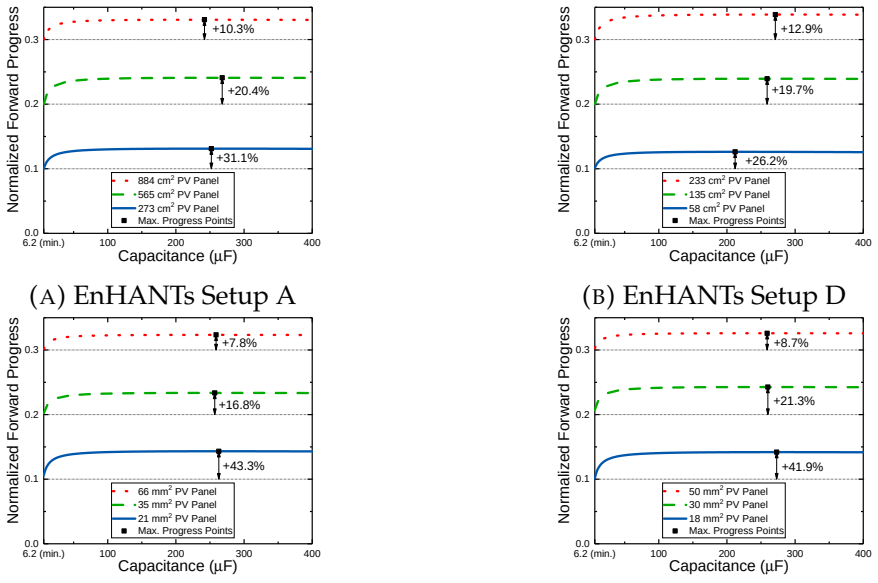
For the purposes of this exploration, three levels of baseline mean forward progress (α_{exe}) are set as 0.1, 0.2, and 0.3. We use the system model to find the PV panel area that achieves the expected forward progress under the different energy source conditions with minimum energy storage. We scale the PV panel area to find that which achieves each baseline α_{exe}. As shown in Figure 4.3, the energy harvester sizes that achieve the desired α_{exe} may span orders of magnitude given different energy source conditions from mm^2 for outdoor sources ((c) and (d)) to cm^2 for indoor sources ((a) and (b)).

4.3.2.2 Sizing the Energy Storage

Having obtained the energy harvester sizes for the baseline forward progress, we then use the modelling approach to size energy storage. We analyze the sizing effect of energy storage on forward progress given real-world energy conditions. Figure 4.3 shows a 7.8-43.3% improvement in forward progress by sizing energy storage under the given real-world energy conditions and baseline energy harvester sizes. It can also be inferred that optimizing energy storage can either improve forward progress for a given energy harvester size, or reduce the energy harvester size that achieves the target forward progress. Given higher-power energy sources (e.g. Denver 2018 and Hawaii 2018 outdoor solar), increasing the harvester size efficiently improves forward progress with minor dimensional overheads, e.g. tens of mm^2; however, given lower-power sources (e.g. EnHANTs Setup A and Setup D indoor light), optimizing energy storage capacitance can save tens of cm^2 of PV panel area to achieve the same forward progress.

4.3.2.3 Interruption Period

Besides forward progress, we also explore how the capacitance can change the interruption periods. When interrupted by insufficient power supply, an ICS enters an interruption period where it saves its volatile state, waits for supply voltage to recover, and restores the state to resume execution, without making any forward progress. Applications that require frequent sensing may be negatively affected by long interruption



(A) EnHANTs Setup A (B) EnHANTs Setup D

(C) NREL Denver 2018 (D) NREL Hawaii 2018

FIGURE 4.3: Improvement of average forward progress by sizing energy storage given different PV panel areas under real-world energy source conditions. The model is able to find the PV panel area required for achieving the target mean forward progress.

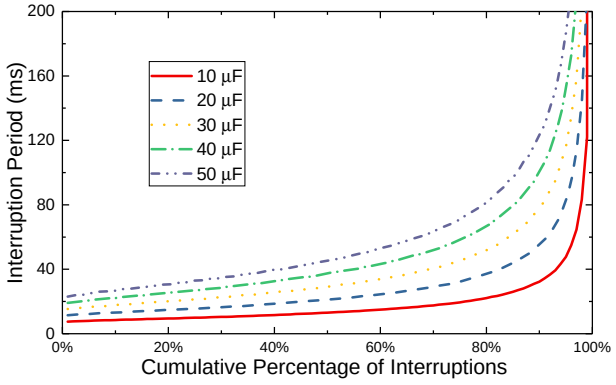


FIGURE 4.4: Distribution of interruption periods.

periods. We measure an interruption period as *the period between two successive execution periods*, e.g. a consecutive 'SLR' period in Figure 3.3 forms an interruption period. We record all the interruption periods during a one-year simulation with 10–50 µF capacitors, the Denver 2018 dataset, and an 80 mm^2 PV panel. Figure 4.4 presents the distribution of all the interruption periods. With increased energy storage, the interruption period is prolonged. For example, the 90th percentile of interruption periods increases from 32.2 ms at 10 µF to 123.4 ms at 50 µF at an approximate rate of 23 ms per 10 µF. Facilitated by the simulator, developers are enabled to estimate whether the distribution of interruption periods meet their application requirement.

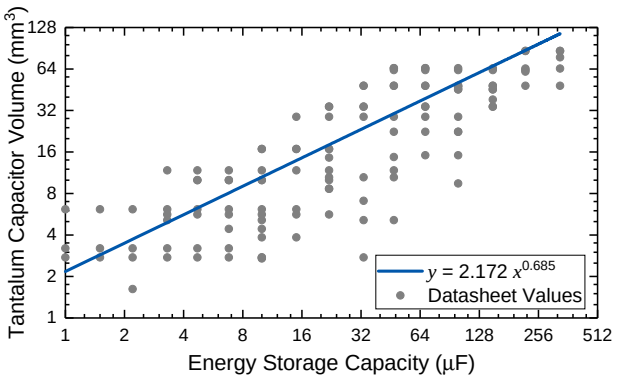


FIGURE 4.5: Tantalum capacitor volume against capacitance for the six series of capacitors analyzed.

4.3.3 Trading Forward Progress, Dimensions, and Interruption Period

Although increasing energy storage capacitance improves forward progress, larger capacitance increases both dimensions and interruption periods. We evaluate the overheads of increased capacitor dimensions and interruption periods, and then trade them off against forward progress using a cost function to suggest an optimal capacitance value.

4.3.3.1 Metric of Dimensions

The overhead of capacitor dimensions is evaluated by characteristics of off-the-shelf tantalum capacitors. We narrow down the range of sample capacitors within a set of characteristics: low-profile, 10V rated voltage, and surface-mount package, and select six series of capacitors[1]. The volume and capacitance of these devices are plotted in Figure 4.5. We use the regression of these data to approximate a capacitance-volume relationship.

4.3.3.2 Metric of Interruption Periods

Applications may have various requirements on interruption periods. To demonstrate the usage of our sizing approach, we consider a designer requests the 90th percentile of all interruption periods as an example metric of interruption periods, denoted as T_{int}. This metric indicates 90% of interruption periods are shorter than T_{int}. This metric can be adapted for particular application requirements.

[1]The series of capacitor considered were: AVX TAJ, AVX TACmicrochip, AVX F92, Vishay 572D, Vishay 591D, and Vishay 592D.

4.3.3.3 Cost Function

From the previous observations (Figure 3.5) we can see that achieving the optimal progress improvement costs much more capacitance (mean 3.2×) than to achieve 95% improvement. A trade-off is necessary to improve forward progress while restricting the overheads of increased capacitor volume and interruption periods. We use the cost function in (4.1) to trade off forward progress, capacitor volume, and interruption periods:

$$f = \frac{\alpha_{exe}}{k_1} - \left(\frac{v_{cap}}{k_2}\right)^2 - \left(\frac{T_{int}}{k_3}\right)^2 \tag{4.1}$$

where v_{cap} denotes capacitor volume and T_{int} denotes interruption periods. α_{exe}, v_{cap}, T_{int} can be described as functions of C. k_1, k_2, and k_3 are independent factors used for normalizing each metric, and they are empirically determined according to applications. In this example, the undesirable parameters are expressed as quadratics to give an increasing cost to higher values. While only three parameters are considered here, others (such as the energy harvester size) could be included for a system-wise sizing scenario. As an example, we configure the function by setting $k_1 = 0.2$, $k_2 = 200\,\text{mm}^3$, and $k_3 = 500\,\text{ms}$.

4.3.3.4 Results

The effect of the trade-off is plotted in Figure 4.6 using the Denver 2018 energy source dataset. Compared to the capacitor size that solely maximizes forward progress, on average, an appropriately-sized capacitor achieves 93% of the maximum forward progress, while saving 83% of capacitor volume and 91% of interruption periods. Compared to the minimum storage case, the appropriately-sized capacitor improves forward progress by 12-124% with energy storage increased from 6.2 µF to 30 µF.

As shown in Figure 4.5, the closest available capacitance that satisfies the 6.2 µF minimum capacitance is 6.8 µF, whereas the closest available capacitance to the appropriate 30 µF is 33 µF. The minimum volumes of 6.8 µF and 33 µF capacitors are both $2.75\,\text{mm}^3$, which means using the appropriate capacitance, instead of the minimum one, may not incur dimensional overhead. The regressed volume of the above two capacitance values are $8.1\,\text{mm}^3$ and $23.8\,\text{mm}^3$ respectively. However, the selection of capacitors can be dependent on factors other than physical volume, such as reliability, operation temperature, and more specific application needs. These factors can also be added into the cost function if necessary.

4.4 Summary and Discussion

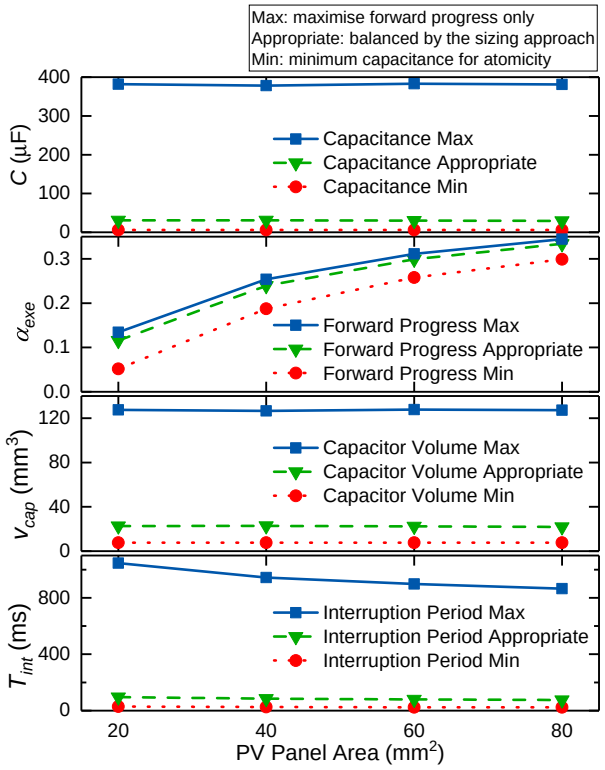


FIGURE 4.6: The sizing approach trades off forward progress, capacitor volume, and interruption periods. The results are plotted against a range of PV panel area, given Denver 2018 energy source dataset.

Chapter 5

Runtime Energy Profiling and Threshold Adaptation

5.1 Introduction

Energy harvesting has become a promising power solution for the Internet of Things, liberating wireless sensors from batteries and the power grid. Batteryless devices harvest ambient energy, such as light, radio-frequency, and mechanical movement, which is then buffered in a capacitor. As the harvested power is typically insufficient for continuous operation, such devices operate in an intermittent way – when a certain amount of energy is collected, the processor wakes up, executes program until the amount of energy falls below a threshold, where it sleeps or dies, and waits for the next energy cycle [ref]. Prior work in *intermittent systems* has developed sophisticated methods to preserve forward progress across frequent power interruptions by carefully *checkpointing* the volatile computing state in CPU registers and volatile memory into non-volatile memory (NVM), and restoring it on reboot [ref].

Apart from computing, embedded sensor systems need to utilize peripherals, such as sensors, computational accelerators, and radios, which typically require *atomicity*. In the context of intermittent systems, an atomic operation should not be checkpointed during execution; if interrupted by power failures, it should restart rather than checkpoint and resume. A peripheral operation is considered atomic because it is infeasible to completely read, save, and restore the intermediate internal state of peripherals, and even if possible, could produce unwanted results. (Example) Prior works on intermittent peripheral operations either individually dedicate an design-time calibrated energy budget for each peripheral operation [60], or allocate a universal energy budget that ensure the most energy hungry operation can finish in one energy cycle [22].

However, in this paper we propose that manually profiling each peripheral operation and setting fixed thresholds is impractical due to variability in capacitance and energy consumption, where we have considered the following cases:

1. *Variability in capacitance:* As a component for buffering energy in intermittent systems, capacitors typically present a $\pm$10-20% tolerance on rated capacitance. Capacitors also age over time. It is shown that capacitance can decrease by 7.2% in 3000 hours under a 25 °C ambient temperature in experiments [**?**], and by 50% within 10 years under 40 °C as manufacturers stated [**?**]. A degraded capacitor does not change the load consumption, but can increase the voltage difference before and after an operation, and hence makes the pre-defined voltage threshold unsafe.

2. *Variable amount of data to process:* A peripheral function can accept a runtime variable amount of data, e.g. sending different lengths of packets or encrypting different amount of data. Statistically, this linearly scales the charge consumption.

3. *Variability in peripheral configurations:* A peripheral can run with variable configurations at runtime, and demonstrate variable performance and energy consumption. For example, an AES accelerator can encrypt or decrypt data with 128-, 192-, or 256-bit keys. A longer key provides more security, but also takes more time and energy to complete.

4. *Device variability:* Devices have their variation in power consumption, even with the same part number. A threshold profiled on one device can be inadequate on another.

A fixed threshold can be violated if any of the above cases happen, and lead to non-termination. In practical deployment, profiling every atomic operation for every device with all runtime scenarios could an extremely tedious work. Considering this complexity, it is unrealistic to profile the energy budget in every scenario and customize a voltage threshold accordingly.

On the other hand, using only one high voltage threshold, though probably avoids non-termination, can affect system efficiency. Microcontrollers and peripheral devices typically draw more current at a higher supply voltage. The high voltage in the buffering capacitor can also decrease the charging efficiency of energy harvesters and increase system leakage. Hence, setting a high voltage threshold for the processor to wake up results in a superlinear long charging time, which therefore slows down the system execution or even leave the system in a infinite wait at low input power.

To address the above issue, we propose OPTA (online profiling and threshold adaptation), a methodology that profiles energy consumption of operations at runtime and

allocate adaptive thresholds based on newly profiled consumption and user-defined parameters. OPTA profiles the voltage decrease that an operation can cause without any energy input during the operation, while the energy harvesting supply is connected. The profiling strategy is to measure the input current in the charging cycle so as to calculate the maximum drop of supply voltage in the discharging cycle. The profiled energy budget can therefore guarantee the completion of an atomic operation. Based on the profiling results, OPTA dynamically adapts the threshold for each atomic operation, with an option of scaling threshold by user-defined parameters or peripheral configurations.

*** Contributions and paper structure ***

5.2 Background and Motivation

5.2.1 Intermittent Peripheral Operations

Several papers have been published on handling atomic peripheral operations in intermittent systems, where energy profiling of workloads is an inherent part of their methodologies.

DEBS [60] experimentally profiles the energy consumption of each task at design time, and designates a threshold to each task individually. DEBS enters a low-power mode (LPM) after completing an operation, and waits for energy replenishment until the next threshold.

Samoyed [22] utilizes a custom design-time *energy profiler* [**?**] to identify whether the adopted energy storage size suffices to run an adequate number (hundreds, as suggested) of peripheral operations in one energy cycle. At runtime, Samoyed starts execution when energy is refilled to a certain threshold, and keeps executing until energy is depleted. Samoyed differs from most static approaches, e.g. Alpaca, on handling computational workloads where it reactively checkpoints when energy falls below a low threshold, and supports user-customized subdivision of peripheral operations when the operation cannot complete in one energy cycle.

RESTOP [**?**] provides programmer-configurable rules that track the instructions issued to peripherals through serial interfaces in a history table. On power recovery, RESTOP re-issues instructions saved in the history table and then resumes the interrupted operation. At design time, RESTOP needs to profile the worst-case energy consumption for restoring peripheral state to identify the minimum (most-efficient) restore threshold.

Prior work profiles the energy consumption of each atomic operation at design time to determine a voltage threshold or a capacitor size that ensures forward progress. However, this does not guarantee the completion of every atomic operation because energy

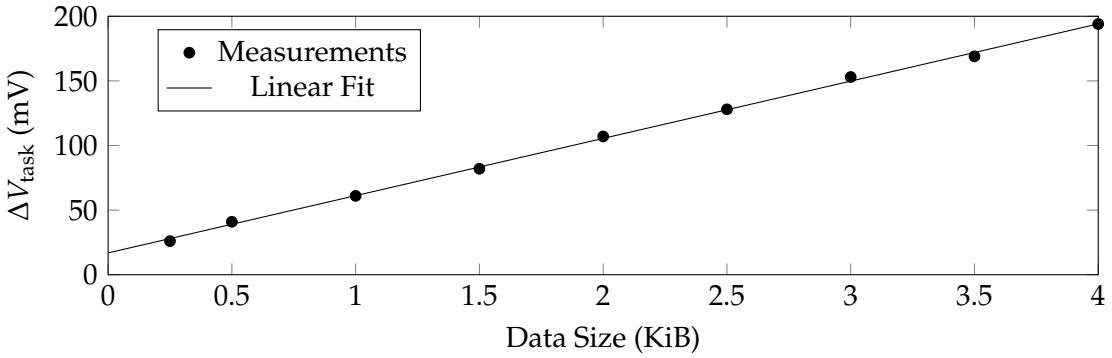


FIGURE 5.1: Voltage drop vs data size.

consumption can change with any runtime conditions different to the profiling setup (detailed in Section 5.2.2). Hence, previous designs typically achieve atomicity by carefully managing the system state, such that the atomic operation does not output valid data until completion, and, if a power failure happens during the operation, the system state rolls back to the start of the operation upon power recovery (rather than checkpoint and resume). Also, previous designs should usually leave an inefficiently safe margin between the profiled energy consumption and the energy to be allocated, such that dynamic variation can be largely tolerated.

5.2.2 Variability in Energy Consumption, Storage, and Input

We took an example of atomic operations to study the dynamic variation of energy consumption. We chose the AES accelerator on TI MSP430FR5994, running a data encryption peripheral function.

Explain each kind of dynamic variation of energy consumption, as listed in Section I-A.

5.2.2.1 Capacitor Degradation and Tolerance

5.2.2.2 Data size

Figure 5.1

5.2.2.3 Peripheral Configurations

Table 5.1

TABLE 5.1: Peripheral configuration variability, AES Encryption 4KB

Configuration	ΔV_{task}
128-bit key	194 mV
192-bit key	230 mV
256-bit key	245 mV

TABLE 5.2: Device variability, AES 128-bit Encryption 4KB

Device No.	ΔV_{task}	System Capacitance	Time
1	185 mV	29 µF	6.444 ms
2	194 mV	35 µF	6.479 ms
3	179 mV	29 µF	6.462 ms

5.2.2.4 Device

Table 5.2

5.2.2.5 Other

Clock frequency (Tentative). 1MHz: 757uA (This should be checked later as it takes the same time to complete).

5.2.2.6 Voltage

(This should be relevant as you mention it's better to operate at lower voltage)

— Start V — End V — Delta V —

— 2.020V — 1.836V — 184mV —

— 2.102V — 1.917V — 185mV —

— 2.214V — 2.030V — 184mV —

— 2.301V — 2.117V — 184mV —

— 2.424V — 2.240V — 184mV —

— 2.542V — 2.352V — 190mV —

— 2.634V — 2.444V — 190mV —

— 2.710V — 2.521V — 189mV —

— 3.360V — 3.166V — 194mV —

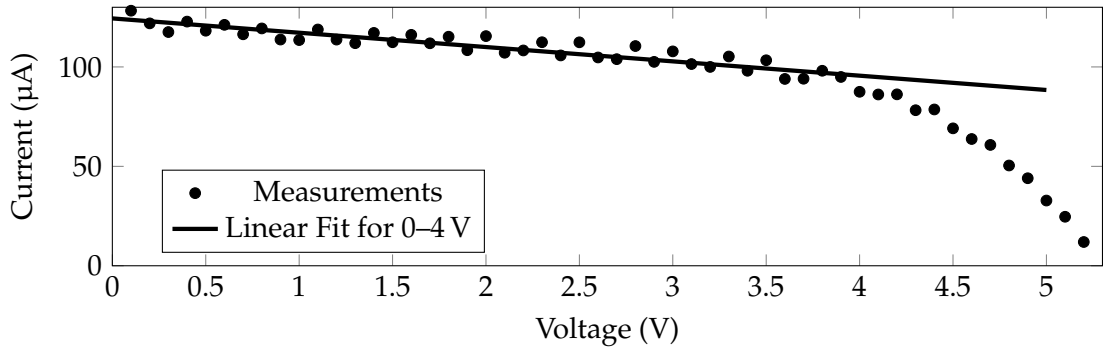


FIGURE 5.2: IV curve.

5.2.2.7 Charging efficiency of energy harveser vs Vcc

5.2.3 Forward Progress Improvement with Adaptive Energy Budgeting

We took a modelling process to demonstrate the potential of improving forward progress with adaptive energy budgeting, given dynamic execution time and dynamic current draw.

5.2.3.1 Model Description

We modelled an ideal case of adaptive energy budgeting, where the system exactly knows how much time the next atomic operation will take and how much the current draw is. The system sets a voltage threshold that exactly guarantees the charge consumption even if the supply current immediately drops to zero after the start of an operation.

5.2.3.2 Simulation Setup

The system has a 10 µF capacitor for buffering energy with 1.5 V charge at the beginning. The system consumes 20 µA in the LPM and 0 µA when off. The shutdown threshold is 1.8 V, which is also the end voltage that all approaches are profiled for or adapted to. We set a dummy workload which draws 750 µA and takes up to 5 ms to complete. Two sets of simulation were conducted. One set was run with a randomized dynamic execution time that is uniformly distributed between 0 and 5 ms, with 750 µA current draw unchanged. The other was run with a dynamic current draw which is increased by 0-20% uniformly distributed, with 5 ms execution time. We ran each set of simulation for 10 times and took the mean, maximum, and minimum of the number of completed operations as a metric of forward progress.

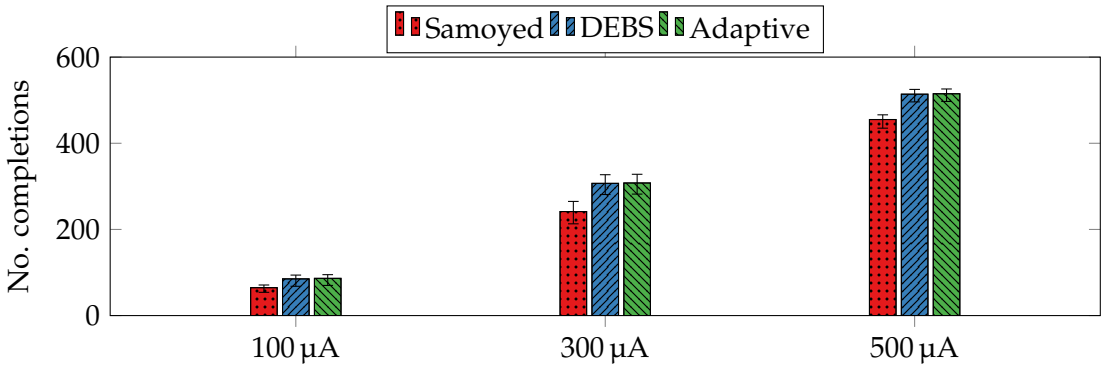


FIGURE 5.3: Dynamic execution time.

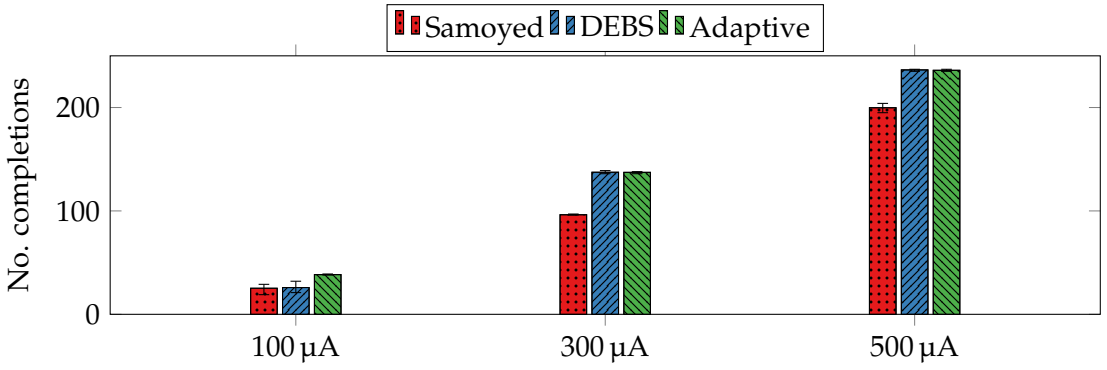


FIGURE 5.4: Dynamic current draw.

5.2.3.3 Results

Figure 5.3.

Figure 5.4.

Analyse where the improvement comes from (probably with a voltage trace).

5.3 Methodology

Motivated by the previous examples, we propose OPTA, a new methodology for profiling energy consumption of tasks at runtime and dynamically adapting the energy budget to the varying energy consumption of tasks.

5.3.1 Online Profiling Method

Here, we present an efficient online profiling method. OPTA dynamically profiles the actual voltage droop that a task consumes while the supply is connected. The actual voltage droop means the voltage droop in V_{cc} by running a task if the supply current immediately drops to zero after the task starts. With the supply connected, the supply current keeps charging the capacitor during execution, so the actual voltage droop cannot be measured simply by two voltage measurements at the beginning and the end of a task. A naive solution will be disconnecting or short-circuiting the supply during profiling, but this can waste energy. In contrast, while keeping the supply connected, our method analyzes the supply current in the charge cycle, and uses it to derive the actual voltage droop in the discharge cycle. We will explain this method with its mathematical reasoning first, and then discuss possible concerns in its implementation.

5.3.1.1 Assumptions

We have made the following assumptions for the mathematical reasoning. Later, we will discuss the situations where these assumptions do not hold.

- *The supply current remains constant across a charge-discharge cycle.* In intermittently-powered systems, the energy buffering capacitor is typically small, e.g. tens to hundreds of µF, and hence the charge-discharge cycle is typically short, e.g. tens to hundreds of ms. The supply current is unlikely to change in such a short period. Hence, in the following reasoning, we presume that the supply current is constant across a charge-discharge cycle.
- *The system is able to measure the supply voltage and time when active.* Off-the-shelf MCUs, e.g. MSP430 series, are equipped with many low-power peripherals, including ADC for measuring voltage and RTC for time-keeping. In the following reasoning, we assume that these two modules are available and free of any overheads.

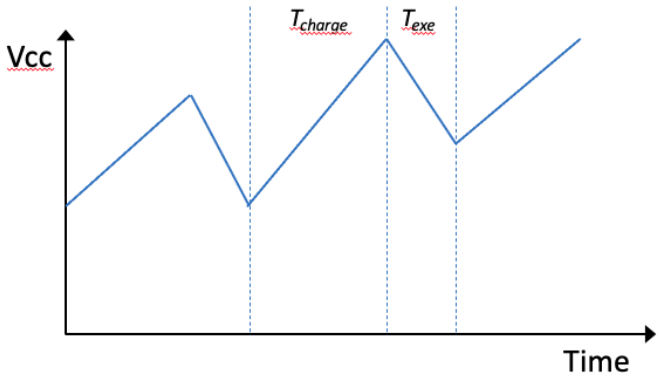


FIGURE 5.5: Figure for illustrating mathematical derivation.

5.3.1.2 Mathmatical Derivation

We show an illustrative trace of V_{cc} across charge-discharge cycles in Fig. 5.5.

(A list that explains math symbols to be inserted)

The charging cycle can be described as

$$\Delta V_{charge} C = (I_{in} - I_{sleep}) T_{charge} \tag{5.1}$$

where

$$I_{sleep} = I_{comp} + I_{rtc} + I_{lpm} \tag{5.2}$$

The discharging cycle can be described as

$$\Delta V_{discharge} C = (I_{in} - I_{exe}) T_{discharge} \tag{5.3}$$

where

$$I_{exe} = I_{comp} + I_{rtc} + I_{task} \tag{5.4}$$

The actual charge consumption of a task is

$$\Delta V_{task} C = I_{exe} T_{discharge} \tag{5.5}$$

Combining (5.1), (5.2), (5.3), (5.4), and (5.5), we can get the expression of ΔV_{task} as

$$\Delta V_{task} = \Delta V_{charge} \frac{T_{exe}}{T_{charge}} - \Delta V_{exe} + \frac{I_{sleep} T_{exe}}{C} \tag{5.6}$$

Here, ΔV_{task} is the actual voltage droop that we should allocate before the execution of a task. ΔV_{charge}, ΔV_{exe}, T_{exe}, and T_{charge} are the values that can be measured by the ADC

and RTC at runtime. If $\frac{I_{sleep}T_{exe}}{C}$ is negligible, ΔV_{task} can be derived at runtime with all perceivable values.

5.3.1.3 Realistic Considerations

Although the above equation is straightforward, there are some practical concerns that may affect the accuracy or efficiency.

- Minimizing $\frac{I_{sleep}T_{exe}}{C}$.

  As the profiling method ignores $\frac{I_{sleep}T_{exe}}{C}$, the profiled value can be theoretically smaller than the actual one. However, if we look at the empirical values of I_{sleep}, T_{exe}, and C in some typical IPSs, this is relatively small. The sleep current I_{sleep} is a key property that should be minimized in IPSs. The execution time of a task T_{exe} is typically within 20 ms as the energy storage capacitor cannot afford a long, energy-hungry task. The capacitance of energy storage C is usually from 10 µF to hundreds of µF. Hence, $\frac{I_{sleep}T_{exe}}{C}$ is typically a few mV. This is insignificant compared to the voltage droop of a task (potentially hundreds of mV), and is easily offset by a margin in implementation, e.g. the granularity of voltage comparator.

- How fast can supply current change in a charge-discharge cycle? What can it cause?

  The rationale behind this method indicates that we use the average current input in the charge cycle as the current input in the discharge cycle. Although this is unlikely to change largely considering the short execution time of a task, we still analyze the effect it can bring when the two current values are different. We denote the two current values as $I_{in-charge}$ and $I_{in-discharge}$. When $I_{in-charge} > I_{in-discharge}$, the system over-profiles the voltage droop. This is typically fine, as the task can still safely finish and the over-profiled value would be corrected in the following measurements. When $I_{in-charge} < I_{in-discharge}$, the system under-profiles the voltage droop. This can result in a lower energy budget than the safe one. However, as $I_{in-discharge}$ is increasing, the task can usually finish with the additional energy. The extreme case is when the system profiles a task with an increasing supply current, while executes the task with a decreasing supply current next time. This can result in the failure of a task as it runs out of energy. To maintain atomicity, our approach restarts the task from the beginning if it fails by disabling checkpoints during the task execution.

- ADC, RTC, and processing overheads.

  The overheads of this profiling method in implementation come from ADC, RTC and processing.

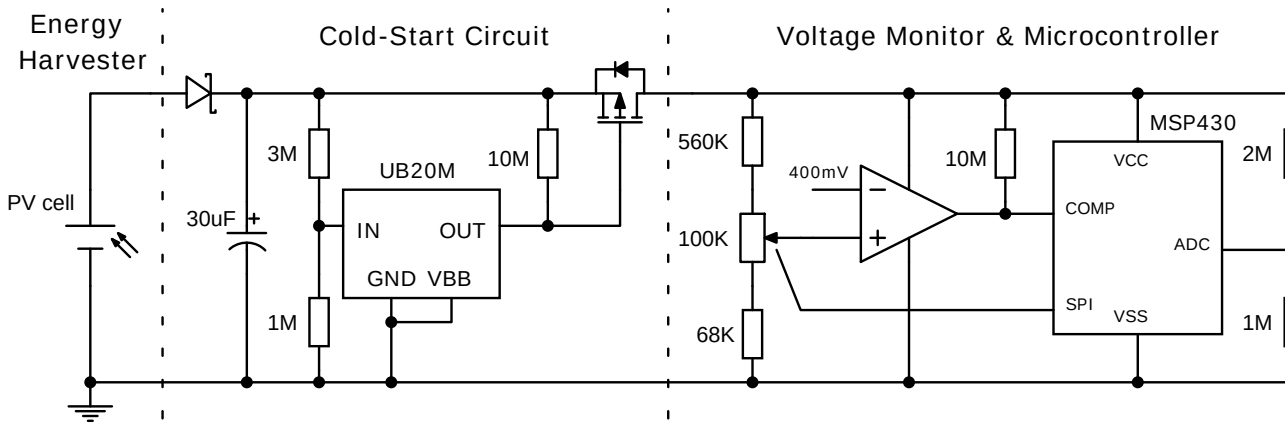


FIGURE 5.6: System circuit.

5.3.2 Threshold Adaptation

5.3.2.1 Profiling Strategy

When should we take a profiling measurement?

Goal: Reduce unnecessary/redundant measurements and take necessary measurements.

5.3.2.2 Learning Algorithm

How do we use the measurements to update the voltage threshold?

Enable voltage threshold adaptation against runtime variation of energy consumption due to unforeseeable operating conditions, and also enable linear adaptation to function knobs that are known to the system.

- Without function knobs: use the latest profiled threshold.
- With function knobs: do a linear regression based on a number of recent measurements.

5.4 Implementation

- Overview
- Circuit
- Modules used in MSP430
- State Saving and Restoring Algorithm
- Usage

5.5 Experimental Evaluation

5.5.1 Benchmark and Comparisons

Benchmark:

- AES encryption
- AES decryption
- RF transmission
- UART send
- DMA (a very lightweight/efficient task, probably not ideal for the reliability test)

Comparisons:

DEBS, Samoyed (later), Plain C (for evaluating overheads)

We may temporarily overlook Samoyed as its implementation is a bit more complex and its threshold setting is unclear. Samoyed scales down the atomic task if it fails to complete (but never scale back). It uses an ”energy profiler” in previous work to test the whether the smallest scale of all peripheral tasks and randomized inputs can successfully complete. They suggested it is appropriate to set an energy capacity that can run the smallest scale of an operation for hundreds of times in one energy cycle. Hence, it does not look for a threshold for a task with a specific configuration, but instead its aim is to minimize the chance of non-termination in practice by giving a high margin.

5.5.2 Profiling Accuracy

Question: How does our profiling approach perform in terms of accuracy?

Accuracy compared to manual measurement.

Figure shows:

- Profiling results (average of 10)
- Each profiling reading (range)
- Dynamic threshold (Perhaps add horizontal lines as configurable thresholds in figures)
- Actual voltage drop (average of 5 manual readings)

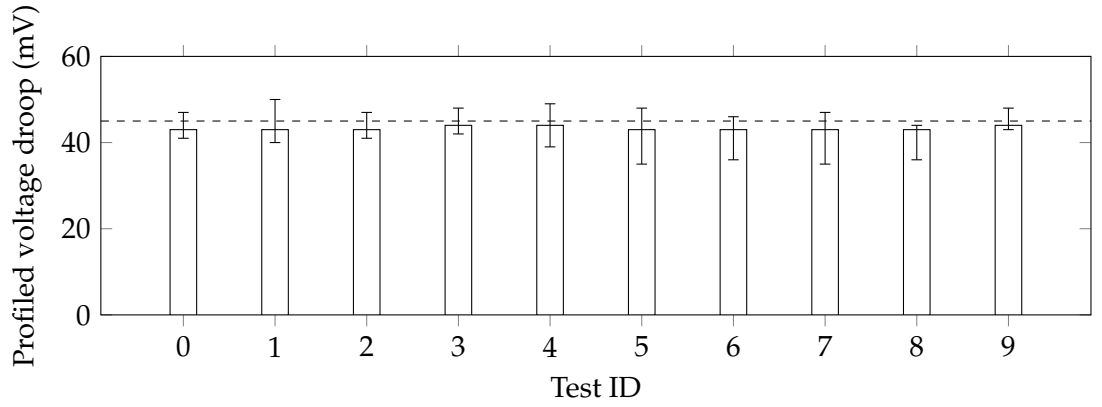


FIGURE 5.7: DMA profiling.

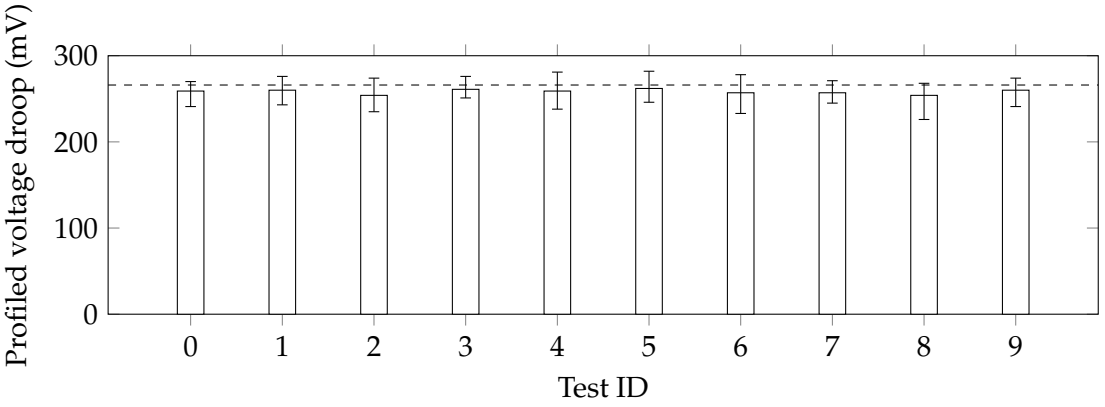


FIGURE 5.8: AES profiling.

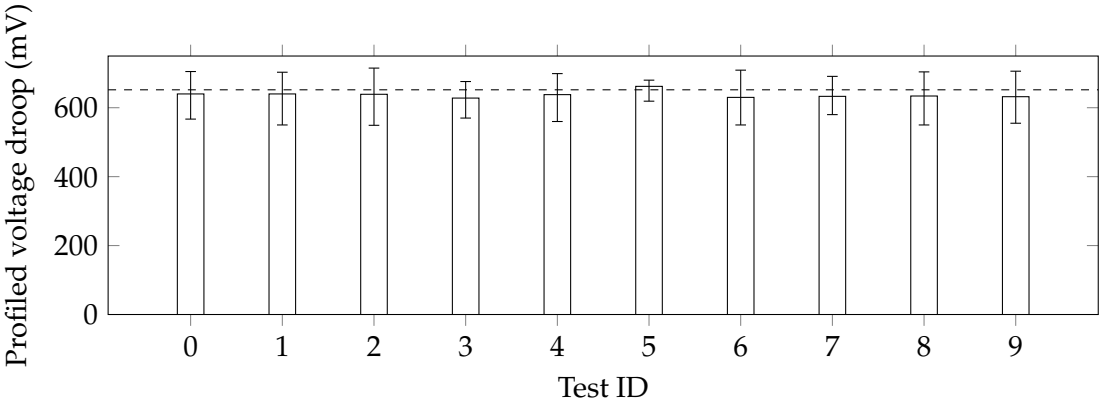


FIGURE 5.9: UART profiling.

5.5.3 Reliability with Dynamic Energy Consumption

Question: Can it still make forward progress correctly with changes (as listed below) while other SoA approaches can't?

Number of failures?

Cases of changes and experiments:

5.5.3.1 New devices / operations (once)

- Show the voltage trace that illustrates how it profiles and adapts on new devices or new operations.

5.5.3.2 Variability in capacitance due to ageing / tolerance (slowly changing)

- Profile the tasks for DEBS with the target end voltage at 1.8V (need explanation on this) and 30uF capacitance.

- Build a capacitor bank with a better granularity. The potential testing range of capacitance should be 20-30uF, so we probably use a combination of 2x10uF + 10x1uF capacitors, with some 0.1uF capacitors where necessary.

- Decrease the capacitance step by step. Record the capacitance where DEBS fails, the adaptive thresholds, and a voltage trace that shows what happens.

5.5.3.3 Variability in peripheral configurations (single threshold for a rarely/slightly-changing configuration, multiple thresholds for frequently-changing configurations)

- Profile the tasks for DEBS with the target end voltage at 1.8V and a ”default” configuration.

- Presumably DEBS can only complete the tasks with configurations that consumes the same or less energy as it was profiled, while OPTA adapts the threshold.

5.5.3.4 Variability in the amount of data to process (fast changing, but perhaps could be an unsuitable test case for reliability as it should violate the API requirement to make it fail)

- This would be similar to the capacitance test but with a less granularity needed.

5.5.4 Efficiency

Question: Does it run faster than other SoA approaches (make more progress under the same energy condition) under conditions that all approaches can make forward progress?

Comparisons: DEBS, Samoyed.

Test conditions:

- (1) A constant data size (2) Randomized data sizes (DEBS threshold configured for the largest data size)

- A few levels of input current

5.5.5 Overheads

- Current & time overheads of profiling and adaptation (with a further breakdown according to sub-operations) compared to Plain C.

Time is measured by GPIO signals, and current is calculated by measuring voltage droops.

The energy/charge consumption can also be calculated.

- Memory overhead. Check the section sizes of the compiled code. Compared it to a PlainC version and a Hibernus-like IC version.

5.5.6 Correctness of computational results (test its intermittent computing functionality, might not be important)

Question: does it produce correct results from atomic functions across power failures?

Compare the output of our approach with intermittent supply vs Plain C solution with continuous supply. Use a computational workload probably, as an atomic function should be guaranteed to finish.

5.5.7 Case Study

Apply the proposed approach on an application that includes multiple atomic operations and the device runs with dynamic energy consumption due to operating conditions.

5.6 Conclusions

5.7 Summary and Discussion

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Intermittent computing enables continuity of computation on energy harvesting sensor nodes despite frequent power failures by managing system volatile and non-volatile states. With a goal of minimising device dimensions and cost, intermittent computing systems only adopt energy storage (e.g. a decoupling capacitor) at the minimum requirement of ensuring computing correctness. However, given power production is less than power consumption, such systems have to frequently wake up, execute shortly, and halt, wasting energy on state managing operations. A method of sizing energy storage and energy harvester for intermittent computing devices under real-world deployment is proposed. This sizing method provides a suggestion on how to balance performance and dimensions in sizing energy harvester. In a suggested range of energy harvester sizes, increasing storage from 20µF to 80µF and improve 5.7-22.2% on application execution speed under real-world energy source conditions while not affect device dimensions.

A power-neutral system has to match the power consumption with the available power instantaneously through DVFS since it has almost no energy storage. However, this affects the overall performance, since a system performing DVFS can obtain more computation when operating within a constrained range of frequency instead of scaling frequency according to the harvested power. Based on this, a study on the effect of energy storage capacity on the performance of power-neutral systems is proposed. In this study, the analysis shows that with the increase in capacitance, the system is able to operate within a more stable range of frequency, and hence more forward progress achieved given the same power input and within the same time. It is reported in the simulation that when properly selecting the size of the capacitor, the forward progress can be improved by 14.6% given a sinusoidal power input.

6.2 Future Work

References

[1] Oliver Hahm, Emmanuel Baccelli, Hauke Petersen, and Nicolas Tsiftes. Operating systems for low-end devices in the internet of things: a survey. *IEEE Internet of Things Journal*, 3(5):720–734, 2016.

[2] Luca Mainetti, Luigi Patrono, and Antonio Vilei. Evolution of wireless sensor networks towards the internet of things: A survey. In *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, pages 1–6. IEEE, 2011.

[3] Tosiron Adegbija, Anita Rogacs, Chandrakant Patel, and Ann Gordon-Ross. Microprocessor optimizations for the internet of things: A survey. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):7–20, 2017.

[4] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[5] Jan M Rabaey, M Josie Ammer, Julio L Da Silva, Danny Patel, and Shad Roundy. Picoradio supports ad hoc ultra-low power wireless networking. *Computer*, 33(7):42–48, 2000.

[6] Paul D Mitcheson, Eric M Yeatman, G Kondala Rao, Andrew S Holmes, and Tim C Green. Energy harvesting from human and machine motion for wireless electronic devices. *Proceedings of the IEEE*, 96(9):1457–1486, 2008.

[7] Sravanthi Chalasani and James M Conrad. A survey of energy harvesting sources for embedded systems. In *Southeastcon, 2008. IEEE*, pages 442–447. IEEE, 2008.

[8] Sujesha Sudevalayam and Purushottam Kulkarni. Energy harvesting sensor nodes: Survey and implications. *IEEE Communications Surveys & Tutorials*, 13(3):443–461, 2011.

[9] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(4):32, 2007.

[10] Bernhard Buchli, Felix Sutton, Jan Beutel, and Lothar Thiele. Dynamic power management for long-term energy neutral operation of solar energy harvesting systems. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pages 31–45. ACM, 2014.

[11] Trong Nhan Le, Olivier Sentieys, Olivier Berder, Alain Pegatoquet, and Cecile Belleudy. Power manager with pid controller in energy harvesting wireless sensor networks. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 668–670. IEEE, 2012.

[12] Antonio Caruso, Stefano Chessa, Soledad Escolar, Xavier del Toro, and Juan Carlos López. A dynamic programming algorithm for high-level task scheduling in energy harvesting iot. *IEEE Internet of Things Journal*, 2018.

[13] Peter Wägemann, Tobias Distler, Heiko Janker, Phillip Raffeck, Volkmar Sieh, and Wolfgang SchröDer-Preikschat. Operating energy-neutral real-time systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 17(1):11, 2018.

[14] Fayaz Akhtar and Mubashir Husain Rehmani. Energy replenishment using renewable and traditional energy resources for sustainable wireless sensor networks: A review. *Renewable and Sustainable Energy Reviews*, 45:769–784, 2015.

[15] Daler Rakhmatov, Sarma Vrudhula, and Deborah A Wallach. Battery lifetime prediction for energy-aware computing. In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 154–159. ACM, 2002.

[16] Geoff V Merrett and Alex S Weddell. Supercapacitor leakage in energy-harvesting sensor nodes: Fact or fiction? In *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, pages 1–5. IEEE, 2012.

[17] Farhan I Simjee and Pai H Chou. Efficient charging of supercapacitors for extended lifetime of wireless sensor nodes. *IEEE Transactions on power electronics*, 23(3):1526–1536, 2008.

[18] Xiaofan Jiang, Joseph Polastre, and David Culler. Perpetual environmentally powered sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 65. IEEE Press, 2005.

[19] Farhan Simjee and Pai H Chou. Everlast: long-life, supercapacitor-operated wireless sensor node. In *Proceedings of the 2006 international symposium on Low power electronics and design*, pages 197–202. ACM, 2006.

[20] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[21] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad hoc networks*, 10(7):1497–1516, 2012.

[22] Kiwan Maeng and Brandon Lucia. Supporting peripherals in intermittent systems with just-in-time checkpoints. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1101–1116. ACM, 2019.

[23] Alexei Colin, Emily Ruppel, and Brandon Lucia. A reconfigurable energy storage architecture for energy-harvesting devices. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 767–781. ACM, 2018.

[24] Samuel Wong Chang Bing, Domenico Balsamo, and Geoff V Merrett. An energy-driven wireless bicycle trip counter with zero energy storage. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 404–405. ACM, 2018.

[25] Samuel DeBruin, Bradford Campbell, and Prabal Dutta. Monjolo: An energy-harvesting energy meter architecture. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 18. ACM, 2013.

[26] Benjamin Ransford, Jacob Sorber, and Kevin Fu. Mementos: System support for long-running computation on rfid-scale devices. *Acm Sigplan Notices*, 47(4):159–170, 2012.

[27] Brandon Lucia and Benjamin Ransford. A simpler, safer programming and execution model for intermittent systems. *ACM SIGPLAN Notices*, 50(6):575–585, 2015.

[28] Yongpan Liu, Fang Su, Yixiong Yang, Zhibo Wang, Yiqun Wang, Zewei Li, Xueqing Li, Ryuji Yoshimura, Takashi Naiki, Takashi Tsuwa, et al. A 130-nm ferroelectric nonvolatile system-on-chip with direct peripheral restore architecture for transient computing system. *IEEE Journal of Solid-State Circuits*, 2019.

[29] Tongda Wu, Lefan Zhang, Huazhong Yang, and Yongpan Liu. An extensible system simulator for intermittently-powered multiple-peripheral iot devices. In *Proceedings of the 6th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*, pages 1–6. ACM, 2018.

[30] Domenico Balsamo, Anup Das, Alex S Weddell, Davide Brunelli, Bashir M Al-Hashimi, Geoff V Merrett, and Luca Benini. Graceful performance modulation for power-neutral transient computing systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(5):738–749, 2016.

[31] Benjamin J Fletcher, Domenico Balsamo, and Geoff V Merrett. Power neutral performance scaling for energy harvesting mp-socs. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 1520–1525. European Design and Automation Association, 2017.

[32] K. Ma, X. Li, K. Swaminathan, Y. Zheng, S. Li, Y. Liu, Y. Xie, J. J. Sampson, and V. Narayanan. Nonvolatile processor architectures: Efficient, reliable progress with unstable power. *IEEE Micro*, 36(3):72–83, May 2016.

[33] Faisal Karim Shaikh and Sherali Zeadally. Energy harvesting in wireless sensor networks: A comprehensive review. *Renewable and Sustainable Energy Reviews*, 55:1041–1054, 2016.

[34] Scott D Moss, Owen R Payne, Genevieve A Hart, and Chandarin Ung. Scaling and power density metrics of electromagnetic vibration energy harvesting devices. *Smart Materials and Structures*, 24(2):023001, 2015.

[35] Vijay Raghunathan, Aman Kansal, Jason Hsu, Jonathan Friedman, and Mani Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 457–462. IEEE, 2005.

[36] Winston KG Seah, Zhi Ang Eu, and Hwee-Pink Tan. Wireless sensor networks powered by ambient energy harvesting (wsn-heap)-survey and challenges. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 1–5. IEEE, 2009.

[37] Weidong Xiao, William G Dunford, and Antoine Capel. A novel modeling method for photovoltaic cells. In *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, volume 3, pages 1950–1956. IEEE, 2004.

[38] Oscar López-Lapeña, Maria Teresa Penella, and Manel Gasulla. A new mppt method for low-power solar energy harvesting. *IEEE Transactions on Industrial Electronics*, 57(9):3129–3138, 2010.

[39] Francisco Paz and Martin Ordonez. High-performance solar mppt using switching ripple identification based on a lock-in amplifier. *IEEE Transactions on Industrial Electronics*, 63(6):3595–3604, 2016.

[40] Deepak Verma, Savita Nema, AM Shandilya, and Soubhagya K Dash. Maximum power point tracking (mppt) techniques: Recapitulation in solar photovoltaic systems. *Renewable and Sustainable Energy Reviews*, 54:1018–1034, 2016.

[41] Shad Roundy, Dan Steingart, Luc Frechette, Paul Wright, and Jan Rabaey. Power sources for wireless sensor networks. In *European workshop on wireless sensor networks*, pages 1–17. Springer, 2004.

[42] Cian O Mathuna, Terence O'Donnell, Rafael V Martinez-Catala, James Rohan, and Brendan O'Flynn. Energy scavenging for long-term deployable wireless sensor networks. *Talanta*, 75(3):613–623, 2008.

[43] Detlev Heinemann, Elke Lorenz, and Marco Girodo. Forecasting of solar radiation. *Solar energy resource management for electricity generation from local level to global scale. Nova Science Publishers, New York*, 2006.

[44] National solar radiation database data viewer. https://maps.nrel.gov/nsrdb-viewer/.

[45] Solrmap loyola marymount university (rsr). http://midcdmz.nrel.gov/apps/daily.pl?site=LMU&start=20100406&yr=2016&mo=05&dy=05.

[46] Peter Corke, Philip Valencia, Pavan Sikka, Tim Wark, and Les Overs. Long-duration solar-powered wireless sensor networks. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 33–37. ACM, 2007.

[47] Aaron N Parks, Alanson P Sample, Yi Zhao, and Joshua R Smith. A wireless sensing platform utilizing ambient rf energy. In *Power Amplifiers for Wireless and Radio Applications (PAWR), 2013 IEEE Topical Conference on*, pages 160–162. IEEE, 2013.

[48] Alanson P Sample, Daniel J Yeager, Pauline S Powledge, Alexander V Mamishev, and Joshua R Smith. Design of an rfid-based battery-free programmable sensing platform. *IEEE transactions on instrumentation and measurement*, 57(11):2608–2615, 2008.

[49] Saman Naderiparizi, Aaron N Parks, Zerina Kapetanovic, Benjamin Ransford, and Joshua R Smith. Wispcam: A battery-free rfid camera. In *RFID (RFID), 2015 IEEE International Conference on*, pages 166–173. IEEE, 2015.

[50] Navin Sharma, Jeremy Gummeson, David Irwin, and Prashant Shenoy. Cloudy computing: Leveraging weather forecasts in energy harvesting sensor systems. In *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on*, pages 1–9. IEEE, 2010.

[51] Alessandro Cammarano, Chiara Petrioli, and Dora Spenza. Pro-energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks. In *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pages 75–83. IEEE, 2012.

[52] Raul Morais, Samuel G Matos, Miguel A Fernandes, António LG Valente, Salviano FSP Soares, PJSG Ferreira, and MJCS Reis. Sun, wind and water flow as energy supply for small stationary data acquisition platforms. *Computers and electronics in agriculture*, 64(2):120–132, 2008.

[53] Jay Taneja, Jaein Jeong, and David Culler. Design, modeling, and capacity planning for micro-solar power sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 407–418. IEEE Computer Society, 2008.

[54] Michal Prauzek, Jaromir Konecny, Monika Borova, Karolina Janosova, Jakub Hlavica, and Petr Musilek. Energy harvesting sources, storage devices and system topologies for environmental wireless sensor networks: A review. *Sensors*, 18(8):2446, 2018.

[55] Paulo R Bueno. Nanoscale origins of super-capacitance phenomena. *Journal of Power Sources*, 414:420–434, 2019.

[56] Jiří Libich, Josef Máca, Jiří Vondrák, Ondřej Čech, and Marie Sedlaříková. Supercapacitors: Properties and applications. *Journal of Energy Storage*, 17:224–227, 2018.

[57] Christian Renner, Jürgen Jessen, and Volker Turau. Lifetime prediction for supercapacitor-powered wireless sensor nodes. *Proceedings of FGSN*, pages 1–6, 2009.

[58] Chulsung Park and Pai H Chou. Ambimax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes. In *2006 3rd annual IEEE communications society on sensor and ad hoc communications and networks*, volume 1, pages 168–177. IEEE, 2006.

[59] Geoff V Merrett and Bashir M Al-Hashimi. Energy-driven computing: Rethinking the design of energy harvesting systems. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 960–965. European Design and Automation Association, 2017.

[60] Andres Gomez, Lukas Sigrist, Michele Magno, Luca Benini, and Lothar Thiele. Dynamic energy burst scaling for transiently powered systems. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pages 349–354. EDA Consortium, 2016.

[61] Soledad Escolar, Stefano Chessa, and Jesús Carretero. Energy-neutral networked wireless sensors. *Simulation Modelling Practice and Theory*, 43:1–15, 2014.

[62] Christopher M Vigorito, Deepak Ganesan, and Andrew G Barto. Adaptive control of duty cycling in energy-harvesting wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*, pages 21–30. IEEE, 2007.

[63] Joaquin Recas Piorno, Carlo Bergonzini, David Atienza, and Tajana Simunic Rosing. Prediction and management in energy harvested wireless sensor nodes. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 6–10. IEEE, 2009.

[64] Matthew J Walker, Stephan Diestelhorst, Andreas Hansson, Domenico Balsamo, Geoff V Merrett, and Bashir M Al-Hashimi. Thermally-aware composite run-time cpu power models. In *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2016 26th International Workshop on*, pages 17–24. IEEE, 2016.

[65] Sivert T Sliper, Domenico Balsamo, Alex S Weddell, and Geoff V Merrett. Enabling intermittent computing on high-performance out-of-order processors. In *Proceedings of the 6th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*, pages 19–25. ACM, 2018.

[66] Naveed Anwar Bhatti and Luca Mottola. Harvos: Efficient code instrumentation for transiently-powered embedded sensing. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 209–219. ACM, 2017.

[67] Kiwan Maeng and Brandon Lucia. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 129–144, 2018.

[68] Domenico Balsamo, Alex S Weddell, Geoff V Merrett, Bashir M Al-Hashimi, Davide Brunelli, and Luca Benini. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embedded Systems Letters*, 7(1):15–18, 2015.

[69] Domenico Balsamo, Alex S Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M Al-Hashimi, Geoff V Merrett, and Luca Benini. 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):1968–1980, 2016.

[70] Hrishikesh Jayakumar, Arnab Raha, and Vijay Raghunathan. Quickrecall: A low overhead hw/sw approach for enabling computations across power cycles in transiently powered computers. In *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*, pages 330–335. IEEE, 2014.

[71] Alberto Rodriguez Arreola, Domenico Balsamo, Anup K Das, Alex S Weddell, Davide Brunelli, Bashir M Al-Hashimi, and Geoff V Merrett. 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*, pages 3–8. ACM, 2015.

[72] Josiah David Hester and Jacob Sorber. New directions: The future of sensing is batteryless, intermittent, and awesome. In *15th ACM Conference on Embedded Networked Sensor Systems (SenSys' 17)*, 2017.

[73] Alexei Colin and Brandon Lucia. Chain: tasks and channels for reliable intermittent programs. *ACM SIGPLAN Notices*, 51(10):514–530, 2016.

[74] Kiwan Maeng, Alexei Colin, and Brandon Lucia. Alpaca: intermittent execution without checkpoints. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):96, 2017.

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

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

[77] Yiqun Wang, Yongpan Liu, Shuangchen Li, Daming Zhang, Bo Zhao, Mei-Fang Chiang, Yanxin Yan, Baiko Sai, and Huazhong Yang. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *ESSCIRC (ESSCIRC), 2012 Proceedings of the*, pages 149–152. IEEE, 2012.

[78] Fang Su, Yongpan Liu, Yiqun Wang, and Huazhong Yang. A ferroelectric nonvolatile processor with 46μs system-level wake-up time and 14μs sleep time for energy harvesting applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(3):596–607, 2017.

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

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

[81] Kaisheng Ma, Xueqing Li, Karthik Swaminathan, Yang Zheng, Shuangchen Li, Yongpan Liu, Yuan Xie, John Jack Sampson, and Vijaykrishnan Narayanan. Nonvolatile processor architectures: Efficient, reliable progress with unstable power. *IEEE Micro*, 36(3):72–83, 2016.

[82] Fang Su, Kaisheng Ma, Xueqing Li, Tongda Wu, Yongpan Liu, and Vijaykrishnan Narayanan. Nonvolatile processors: Why is it trending? In *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 966–971. IEEE, 2017.

[83] Yiqun Wang, Yongpan Liu, Cong Wang, Zewei Li, Xiao Sheng, Hyung Gyu Lee, Naehyuck Chang, and Huazhong Yang. Storage-less and converter-less photovoltaic energy harvesting with maximum power point tracking for internet of things. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(2):173–186, 2016.

[84] Yung-Hsiang Lu, Luca Benini, and Giovanni De Micheli. Low-power task scheduling for multiple devices. In *Hardware/Software Codesign, 2000. CODES 2000. Proceedings of the Eighth International Workshop on*, pages 39–43. IEEE, 2000.

[85] Philip Sparks. The route to a trillion devices. Technical report, ARM Ltd., 2017.

[86] Sravanthi Chalasani and J. M. Conrad. A survey of energy harvesting sources for embedded systems. In *IEEE SoutheastCon*, pages 442–447, April 2008.

[87] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. Power management in energy harvesting sensor networks. *ACM Trans. Embed. Comput. Syst.*, 6(4), September 2007.

[88] F. I. Simjee and P. H. Chou. Efficient charging of supercapacitors for extended lifetime of wireless sensor nodes. *IEEE Trans. Power Electron.*, 23(3):1526–1536, May 2008.

[89] Guannan Liu, Yanjun Yu, Jing Hou, Wei Xue, Xinhui Liu, Yanzhen Liu, Wanhua Wang, Ahmed Alsaedi, Tasawar Hayat, and Zhengtao Liu. An ecological risk assessment of heavy metal pollution of the agricultural ecosystem near a lead-acid battery factory. *Ecological Indicators*, 47:210 – 218, 2014.

[90] Fayaz Akhtar and Mubashir Husain Rehmani. Energy replenishment using renewable and traditional energy resources for sustainable wireless sensor networks: A review. *Renewable Sustainable Energy Rev.*, 45:769 – 784, 2015.

[91] Sivert T. Sliper, Oktay Cetinkaya, Alex S. Weddell, Bashir Al-Hashimi, and Geoff V. Merrett. Energy-driven computing. *Philosophical Transactions of the Royal Society A*, 378(2164), 2020.

[92] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 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, pages 159–170, New York, NY, USA, 2011. ACM.

[93] N. A. Bhatti and L. Mottola. HarvOS: Efficient code instrumentation for transiently-powered embedded sensing. In *Proc. 16th IPSN*, pages 209–220, Pittsburgh, PA, USA, April 2017.

[94] Amjad Yousef Majid, Carlo Delle Donne, Kiwan Maeng, Alexei Colin, Kasim Sinan Yildirim, Brandon Lucia, and Przemysław Pawełczak. Dynamic

task-based intermittent execution for energy-harvesting devices. *ACM Trans. Sen. Netw.*, 16(1), February 2020.

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

[96] D. Balsamo, A. S. Weddell, A. Das, A. R. Arreola, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini. Hibernus++: A self-calibrating and adaptive system for transiently-powered embedded devices. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 35(12):1968–1980, 2016.

[97] M. Zhao, C. Fu, Z. Li, Q. Li, M. Xie, Y. Liu, J. Hu, Z. Jia, and C. J. Xue. Stack-size sensitive on-chip memory backup for self-powered nonvolatile processors. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 36(11):1804–1816, 2017.

[98] Y. Liu, J. Yue, H. Li, Q. Zhao, M. Zhao, C. J. Xue, G. Sun, M. Chang, and H. Yang. Data backup optimization for nonvolatile sram in energy harvesting sensor nodes. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 36(10):1660–1673, 2017.

[99] Kiwan Maeng and Brandon Lucia. Supporting peripherals in intermittent systems with just-in-time checkpoints. In *Proc. 40th PLDI*, pages 1101–1116, Phoenix, AZ, USA, 2019.

[100] Hrishikesh Jayakumar, Arnab Raha, Woo Suk Lee, and Vijay Raghunathan. Quickrecall: A HW/SW approach for computing across power cycles in transiently powered computers. *J. Emerg. Technol. Comput. Syst.*, 12(1), August 2015.

[101] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. Tragedy of the coulombs: Federating energy storage for tiny, intermittently-powered sensors. In *Proc. 13th SenSys*, page 5–16, Seoul, South Korea, 2015.

[102] Domenico Balsamo, Benjamin J. Fletcher, Alex S. Weddell, Giorgos Karatziolas, Bashir M. Al-Hashimi, and Geoff V. Merrett. Momentum: Power-neutral performance scaling with intrinsic mppt for energy harvesting computing systems. *ACM Trans. Embed. Comput. Syst.*, 17(6), January 2019.

[103] Kiwan Maeng and Brandon Lucia. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *Proc. 13th OSDI*, pages 129–144, Carlsbad, CA, October 2018.

[104] Fang Su, Yongpan Liu, Xiao Sheng, Hyung Gyu Lee, Naehyuck Chang, and Huazhong Yang. A task failure rate aware dual-channel solar power system for nonvolatile sensor nodes. *ACM Trans. Embed. Comput. Syst.*, 18(4), July 2019.

[105] Neal Jackson, Joshua Adkins, and Prabal Dutta. Capacity over capacitance for reliable energy harvesting sensors. In *Proceedings of the 18th International Conference*

on Information Processing in Sensor Networks, IPSN '19, pages 193–204, New York, NY, USA, 2019. ACM.

[106] Yizi Gu, Y. Liu, Y. Wang, H. Li, and H. Yang. NVPsim: A simulator for architecture explorations of nonvolatile processors. In *Proc. 21st ASP-DAC*, pages 147–152, January 2016.

[107] Sivert T Sliper, William Wang, Nikos Nikoleris, Alexander Weddell, and Geoff Merrett. Fused: closed-loop performance and energy simulation of embedded systems. In *Proc. ISPASS*, Boston, MA, USA, 2020.

[108] J. San Miguel, K. Ganesan, M. Badr, C. Xia, R. Li, H. Hsiao, and N. Enright Jerger. The EH model: Early design space exploration of intermittent processor architectures. In *Proc. 51st IEEE/ACM MICRO*, pages 600–612, Fukuoka, Japan, Oct 2018.

[109] Josiah Hester, Timothy Scott, and Jacob Sorber. Ekho: Realistic and repeatable experimentation for tiny energy-harvesting sensors. In *Proc. 12th SenSys*, page 1–15, Memphis, Tennessee, 2014.

[110] Kai Geissdoerfer, Mikołaj Chwalisz, and Marco Zimmerling. Shepherd: A portable testbed for the batteryless iot. In *Proc. 17th SenSys*, page 83–95, New York, NY, USA, 2019.

[111] Radovan Faltus. Low leakage current aspect of designing with tantalum and niobium oxide capacitors. Technical report, AVX, 2012.

[112] Sivert T. Sliper, Domenico Balsamo, Nikos Nikoleris, William Wang, Alex S. Weddell, and Geoff V. Merrett. Efficient state retention through paged memory management for reactive transient computing. In *Proc. 56th DAC*, pages 26:1–26:6, June 2019.

[113] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embedded Systems Letters*, 7(1):15–18, March 2015.

[114] A Andreas and T Stoffel. Nrel solar radiation research laboratory (srrl): Baseline measurement system (bms); golden, colorado (data). *NREL Report NO.DA-5500-56488*, 1981.

[115] M. Gorlatova, A. Wallwater, and G. Zussman. Networking low-power energy harvesting devices: Measurements and algorithms. *IEEE Trans. Mobile Comput.*, 12(9):1853–1865, September 2013.

[116] Silvano Vergura. A complete and simplified datasheet-based model of pv cells in variable environmental conditions for circuit simulation. *Energies*, 9(5), 2016.

[117] Panasonic. Amorphous silicon solar cells. https://www.panasonic-electric-works.com/cps/rde/xbcr/pew_eu_en/ca_amorton_solar_cells_2018_en.pdf. Last accessed: 23 June, 2019.