Dynamic Power and Energy Management for Energy Harvesting Nonvolatile Processor Systems

KAISHENG MA and XUEQING LI, Dept. of Computer Science and Engineering,
The Pennsylvania State University
HUICHU LIU, Intel Labs, Intel Corporation
XIAO SHENG and YIQUN WANG, Dept. of Electronic Engineering, Tsinghua University
KARTHIK SWAMINATHAN, IBM T.J Watson Research Center
YONGPAN LIU, Dept. of Electronic Engineering, Tsinghua University
YUAN XIE, Dept. of Electrical and Computer Engineering, University of California at Santa Barbara
JOHN SAMPSON and VIJAYKRISHNAN NARAYANAN, Dept. of Computer Science and Engineering, The Pennsylvania State University

Self-powered systems running on scavenged energy will be a key enabler for pervasive computing across the Internet of Things. The variability of input power in energy-harvesting systems limits the effectiveness of static optimizations aimed at maximizing the input-energy-to-computation ratio. We show that the resultant gap between available and exploitable energy is significant, and that energy storage optimizations alone do not significantly close the gap. We characterize these effects on a real, fabricated energy-harvesting system based on a nonvolatile processor. We introduce a unified energy-oriented approach to first optimize the number of backups, by more aggressively using the stored energy available when power failure occurs, and then optimize forward progress via improving the rate of input energy to computation via dynamic voltage and frequency scaling and self-learning techniques. We evaluate combining these schemes and show capture of up to 75.5% of all input energy toward processor computation, an average of 1.54× increase over the best static "Forward Progress" baseline system. Notably, our energy-optimizing policy combinations simultaneously improve both the rate of forward progress and the rate of backup events (by up to 60.7% and 79.2% for RF power, respectively, and up to 231.2% and reduced to zero, respectively, for solar power). This contrasts with static frequency optimization approaches in which these two metrics are antagonistic.

CCS Concepts: • **Hardware → Renewable energy**; **Emerging architectures**

Additional Key Words and Phrases: Nonvolatile processor, dynamic power and energy management, energy harvesting, intermittent power supply

107

This project was supported by NSF Award ASSIST 1160483, 1500848, 1461698, and 1533933, Center for Low Energy Systems Technology (LEAST), and China's High-Tech Research and Development (863) Program under contract 2013AA01320.

Authors' addresses: K. Ma and X. Li (Corresponding Authors), Dept. of Computer Science and Engineering, The Pennsylvania State University; emails: {kxm505, lixueq}@cse.psu.edu; H. Liu, Intel Labs, Intel Corporation; email: huichu.liu@intel.com; X. Sheng and Y. Wang, Dept. of Electronic Engineering, Tsinghua University; emails: {x-cheng12, wang-yq05}@mails.tsinghua.edu.cn; K. Swaminathan, IBM T.J Watson Research Center; email: kvswamin@us.ibm.com; Y. Liu, Dept. of Electronic Engineering, Tsinghua University; email: ypliu@tsinghua.edu.cn; Y. Xie, Dept. of Electrical and Computer Engineering, University of California at Santa Barbara; email: yuanxie@ece.ucsb.edu; J. Sampson and V. Narayanan (Corresponding Author), Dept. of Computer Science and Engineering, The Pennsylvania State University; emails: {sampson, vijay}@cse.psu.edu.

DOI: http://dx.doi.org/10.1145/3077575

ACM Reference Format:
Kaisheng Ma, Xueqing Li, Huichu Liu, Xiao Sheng, Yiqun Wang, Karthik Swaminathan, Yongpan Liu, Yuan Xie, John Sampson, and Vijaykrishnan Narayanan. 2017. Dynamic power and energy management for energy harvesting nonvolatile processor systems. ACM Trans. Embed. Comput. Syst. 16, 4, Article 107 (May 2017), 23 pages.
DOI: http://dx.doi.org/10.1145/3077575

1. INTRODUCTION

Improvements in electronics integration, miniaturization, and power efficiency continue to stretch the boundaries of what constitutes a sufficient power envelope within which to operate a computational platform. This trend is now fueling the emergence of self-powered systems that harness ambient energy sources without reliance on batteries. Such batteryless systems are attractive for ubiquitous deployment in the emerging Internet of Things paradigm by eliminating reliance on the presence of, and constant connection to, underlying power infrastructure and easing the restrictions imposed when provisioning batteries and enabling their recharge and/or replacement. These systems are particularly useful in applications where battery weight or biocompatibility is a design limiter. For example, the additional weight of a battery can increase the payload beyond what can be embedded on insects.

A key challenge in designing batteryless electronics with energy-scavenged power sources is the erratic and unreliable power supply derived from ambient sources. One solution is to add an energy storage capacitor, such as a supercapacitor, to smooth the power. Figure 1 shows an abstracted charging metaphor that highlights the key components of the systems considered in this article, and Figure 2 illustrates an energy trace harnessed from an RF power source stored in a capacitor, accounting for its leakage. Since the capacitor leaks, periods during which the power harnessed is less than the leakage power causes the overall stored energy to decrease. Conversely, leakage rates and size/weight restrictions bound the practical scale of capacitor volume, so the capacitor may saturate and be unable to store additional energy during periods of high input power.

Three key insights into the dynamics of energy storage in energy-harvesting systems are as follows: First, over the highly varying input power ranges that these systems must operate, they will frequently encounter both periods where the energy lost from capacitive storage is greater than that replaced by energy harvesting and periods where short-term increases in input power provide more energy than a practical capacitor can store. Second, the effects of capacitive leakage, finite capacitor storage, charging losses, and other front-end power components are large enough to require co-optimization when considering processor or other compute-engine optimizations for these platforms. Third, as Figure 2 depicts, policy management of the power demand at the processor, the load for this front end, provides substantial leverage in mitigating both capacitive underflow and overflow by changing the slope of energy consumed. Changes in processor policies, such as dynamic voltage and frequency scaling (DVFS) [6, 7, 19, 28, 30, 39], can also affect front-end efficiency: For example, since DC-DC conversion losses depend on the difference in voltage, actively depleting or restoring the energy storage capacitor to a particular voltage range will impact the efficiency of the system in harvesting future power.

Recently, nonvolatile processors have been proposed [26, 27] as a means to help insulate tasks executing on energy-harvesting platforms from power-level uncertainty. Such processors provide architectural support in the form of nonvolatile logic elements to back up and recover the compute state instantly instead of having to rely on traditional approaches based on checkpointing and rollback to ensure forward computational progress on a given task. While the instant save and wakeup support in

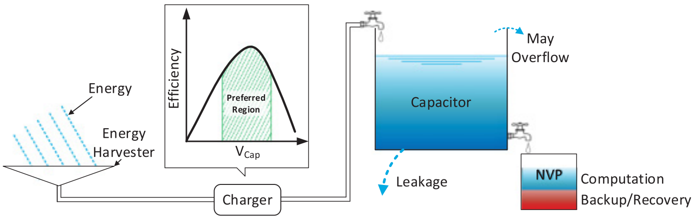


Fig. 1. Plumbing analogy for energy harvesting and consumption.

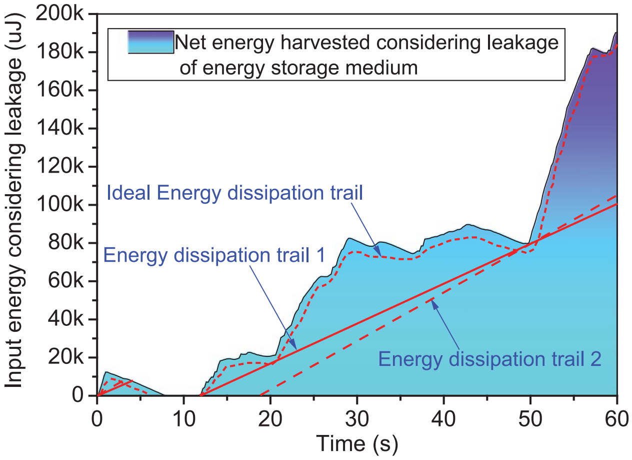


Fig. 2. A comparison of energy input and consumption trails for a TV RF power trace.

nonvolatile processors mitigate the time delays associated with power outages, a significant fraction of the input energy can still be wasted in saving state during outages to the nonvolatile flip-flops. Consequently, techniques such as DVFS that may limit the number of outage-associated backups can redirect the limited energy toward useful computation, altering the observed overheads from nonvolatility. Conversely, these techniques must be adapted to be aware of backup energy costs to properly optimize for forward progress.

The focus of this article is to understand how the limited energy derived from a varying power supply source can be exploited to maximize the computational progress in a nonvolatile processor. Nonvolatile processors offer distinct challenges and opportunities for optimizing forward progress under uncertain power budgets. Nonvolatile backups, if invoked too frequently, incur high energy overheads that must be accounted for in dynamic power policies, but also enable more aggressive speculative optimizations than in volatile processors, such as risking depletion of energy stored in capacitors to perform additional execution during power emergencies when history indicates a likely benefit.

Starting with measurements by perturbing the front end of the energy-scavenging system using a fabricated nonvolatile processor system, we evaluate more complex system-level changes using a validated simulator. Our results indicate that a combination of DC conversion circuitry tuning, capacitor optimizations, and processor frequency and voltage setting policies both improves the rate of forward progress and reduces the number of backup events (by up to 60.7% and 79.2% for RF power, respectively, and up to 213.7% and reduced to zero, respectively, for solar power), in comparison to static frequency optimization approaches.

The frequency scaling approach adopted in the work differs from traditional dynamic frequency scaling (DFS), which is developed for processors with stable power supply. Traditional DFS techniques adjust frequency to improve energy efficiency. They tune the frequency according to workloads, trading off between quality of service and power consumption. In an energy-harvesting NVP system, we target at maximum forward progress, which is the amount of instructions committed while powered with specific power profiles. Constrained by a very limited unstable power supply, instead of saving energy, DFS used in this work is targeted at aggressively improving energy consumption rather than storing it in capacitors to improve the proportion of energy spent on real computation. And as far as we know, this is the first time that DFS is applied to an energy-harvesting NVP system.

The contributions of this article include the following, and can be readily understood using the charging analogy depicted in Figure 1:

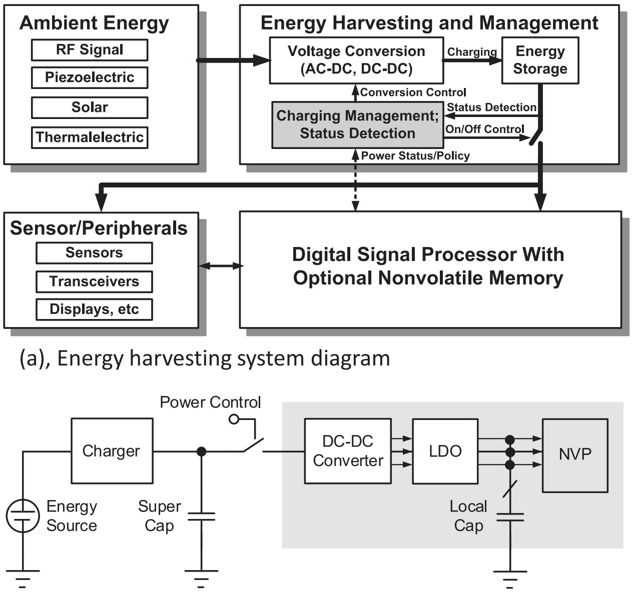
- We tune the front end of the energy-scavenging system to harness the most energy (water) by varying the size of the capacitor (tank) and reduce the DC losses (wastage in connection from source to tank). We leverage nonvolatility aggressively to use the stored energy even if the minimum backup energy threshold has been triggered.
- We show that any static frequency/voltage setting substantially underperforms dynamic approaches. In particular, we show that optimizing for forward progress in an energy-harvested NVP system is distinct from optimizing for least power or least energy per instruction (EPI).
- We dynamically tune the processor frequency and voltage (variable flow from the output tap from the tank) as a function of the incoming power trace (flow of tap feeding the tank) as well as the current capacitor voltage levels (tank storage) to maximize the energy for computations and forward progress. We explore both purely reactive and history-adaptive policies that consider the interplay of the various tradeoffs among system component efficiencies to maximize the forward progress. While DVFS is a well-studied technique, we show why and how a DVFS policy must be tuned for the energy-harvesting NVP environment to account for the substantial impacts of power front-end parameters and the energy costs imposed by both necessary and unnecessary nonvolatile backups.
- We evaluate the combination of our proposed techniques and tunings across a set of kernels and show substantial improvements (1.54×) in forward progress and the correlated metric of the fraction of input energy applied to computation. Our proposed mechanisms and policies increase the fraction of energy consumed by execution to 75.5%, up from only 7.5% in the baseline, statically optimized system, while reducing backup/recovery energy overheads to less than 1% of input energy.

The rest of the article proceeds as follows:

Section 2 provides basic background on energy-harvesting platforms, including the front-end harvesting circuits. Section 3 introduces our prototype NVP system that we use for testing and validation. We then describe our modeling approach and the tuning of front-end parameters. Section 4 proposes speculative optimizations to increase forward progress. Section 5 discusses a set of dynamic power adaptations to better match processor demand with varying supply. Section 6 evaluates our proposed policies across different RF and solar input scenarios and benchmarks. Section 7 discusses related work, and Section 8 concludes the article.

2. BACKGROUND

In this section, a general system structure is introduced to harvest energy from ambient energy sources. In this section, we discuss the characteristics of these energy



(a), Energy harvesting system diagram

(b), Baseline front end circuits

Fig. 3. Baseline front-end circuits.

sources and introduce typical harvesting techniques. The factors that affect the energy conversion efficiency are also discussed.

2.1. Systems Powered by the Ambient Energy Harvesting

With the development of the Internet of Things (IoT), smart cities, and implantable and wearable applications, extremely low-power systems powered by ambient energy sources are gaining popularity. Figure 3(a) shows an archetypical system structure consisting of (1) an energy-harvesting and power management block; (2) a digital signal processor, which is usually implemented using a microcontroller unit (MCU); and (3) the wired or wireless I/O interface. The capacity and implementation of the energy storage medium is also critical to the system design because it directly affects the tradeoff between leakage and other overheads and the maximum stored power. In later sections, this tradeoff will be discussed in more detail.

It is also noted that different energy sources require different energy harvesters for power conversion. For example, the output of a solar cell is a DC signal, while the RF signal and the output of piezoelectric-based systems are AC signals, which require an extra AC-DC rectifier. When the input power is weak, the output voltage may also be low and potentially require an extra DC-DC voltage booster [16].

In this article, the baseline energy-harvesting block is illustrated in Figure 3. Subsequent to the AC-DC or DC-DC conversion, an MPP tracking (MPPT) interface is employed to control the charging power for the highest power-conversion efficiency from the energy harvester.

2.2. Ambient Energy Sources and Harvesting Techniques

Typical ambient energy sources for harvesting include vibration, thermal gradients, and solar and other electromagnetic radiation. Research also indicates that implantable biofuel cells (BFCs) are able to generate electric power from sugars found in the body fluid of an insect [33]. A comprehensive comparison among these power sources can be found in found in articles by Kim et al. [14] and Roundy et al. [31]. In this article, we focus on far-field RF harvesting and solar harvesting as the two key input power environments. The key characteristics of these power sources relevant to this work are that RF energy density varies widely but is generally present to some degree independent of weather and obstacles in a typical range of 0.1nW/cm^2 to 0.1mW/cm^2, while solar power, when present, can provide input power densities over 0.1W/cm^2 harvestable at 10% to 40% efficiency [9].

3. MODELING, VALIDATION, AND TUNING

Our model for the nonvolatile platform is derived from a synthesizable Verilog model for a fabricated, noncommercial research nonvolatile 8051 microprocessor. The Verilog model has been extended with additional connections to control and effect our backup and power management policies, as shown in Figure 5. The model also considers additional features beyond the processor itself, such as the settle-down time for an oscillator from power-on (for clocking after a power failure) and energy per instruction, and includes a front-end circuit-level model that reflects capacitor charging efficiency, chip leakage, and capacitor leakage at different power supply levels. All the parameters in the model have been calibrated using measured results. The key differences between the simulation platform and our fabricated design are the lack of support for the "Energy Policy Unit," "Clock Frequency Unit," and dynamic voltage control mechanisms (see Figure 5) in the fabricated chip.

The model takes as input a power trail and a test bench that drives a benchmark. The outputs of the model are the number of backup operations, the number of clock cycles, the number of finished instructions as forward progress, and the derived statistics from the previous key metrics.

3.1. Baseline Testing Results

We built the PCB prototype shown in Figure 6 to calibrate the model. The NVP chip is designed and fabricated in ROHM 130nm Ferroelectric CMOS technology [26, 38]. While the prototype lacks dynamic frequency and voltage control, we can test it at various fixed voltage/frequency pairs. We tested a frequency range for the processor from 32KHz (min VDD 0.76V) to max frequency 25MHz (min VDD 1.31V) in order to cover both RF-powered and solar-powered operation. The minimum measured VDD is limited by the retention time for ferroelectric nonvolatile memory. For the prototype's technology, dropping VDD further will yield insufficient retention time—only a few milliseconds at 0.5V, which is shorter than some power failures [15, 38]. Due to the limited power income of RF power, we only consider operating frequencies between 32kHz and 1MHz in the model for the 100-minute RF trace as used for Figure 4. For system stability consideration, we ensure that sufficient energy is accumulated in the capacitor for backup before running new instructions when the system starts/recovers.

We also compare several capacitor volumes to evaluate the efficiency of the front-end circuit. The capacitors are all commercial products including Nichicon (4700uF) and Rukycon (0.47uF, 4.7uF, 47uF, 470uF). Intuitively, increasing the capacitor size would help reduce power interruptions and improve forward progress. In particular, larger capacitors will less frequently be at maximum capacity and will be able to continue to accumulate power during upticks. Simulations show that the percentage of time that capacitors are full decreases from 59.4% with 47uF to 31.5% with 4,700uF. However, there are tradeoffs among physical size, capacity, leakage, and charging efficiency that result in larger capacitors, beyond a certain point, actually reducing the percentage of incoming energy available for computation. Figure 4 shows the simulation results over an RF trace with different capacitor sizes for the processor running at 32kHz. Consequently, a larger capacitor exhibits a smaller capacitor voltage range

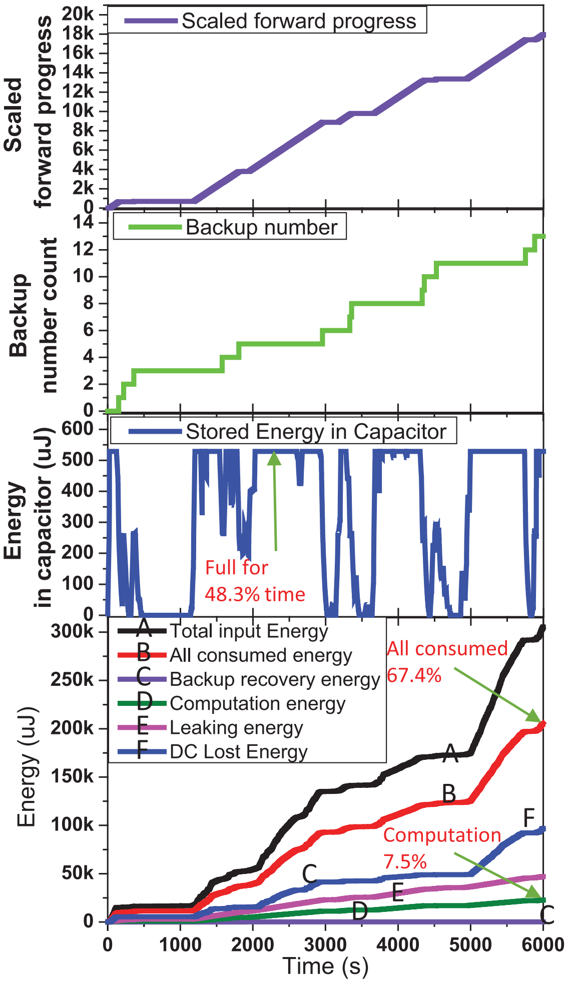


Fig. 4. Simulation results with a 470uF capacitor.

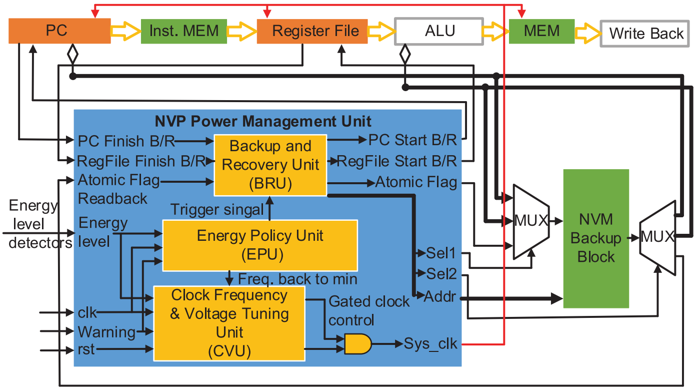


Fig. 5. NVP control block diagram.

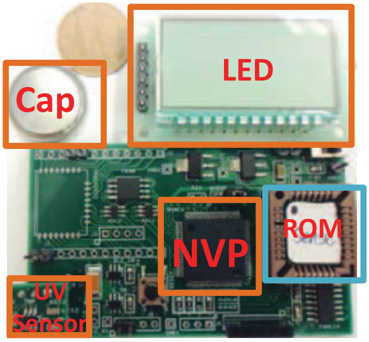


Fig. 6. Testing platform.

but may not profit as efficiently from shorter energy boosts, even though it is expected to reduce the percentage of time that incoming energy is wasted due to the capacitor being full. And charging efficiency varies among capacitors according to start voltage caused by remaining energy and end voltage, which is determined by capacitor size. In summary, the traditional solution of simply increasing the capacitor size is of limited benefit to the NVP system. For optimal operation, the capacitor size can neither be too large nor too small. In our tests, the 470uF capacitor performed the best and is used throughout the rest of this article.

The effects on forward progress stem from the following sources:

(1) Using larger capacitors increases the leakage. Most of the energy is consumed due to the leakage of both the capacitor and the processor.
(2) The DC loss energy represents the energy consumed in the DC-DC charger and the LDO. We assume a constant-voltage charger, which maintains a high voltage to charge the capacitor until the capacitor is full. With a constant-voltage output, the best efficiency of charging the capacitor from 0V to V_{DD} is 50%; hence, at least half of the energy is wasted. Since we also assume a single threshold in our system, the system operates only if the energy detector level is higher than that threshold; otherwise, the system is backed up and stopped. In this case, a large capacitor remains in a higher energy state for a longer period of time. A small capacitor using the same amount of energy has a wider range of voltage drop and results in a higher DC loss, especially in the charger.
(3) The stored energy in a capacitor can be extracted only when the capacitor voltage level reaches a certain threshold. A large capacitor requires a longer time to reach the threshold voltage than the small capacitor. If an input power spike occurs in a short period of time, the voltage of the large capacitor may not be able to charge up to the threshold, and the power boost may be wasted with larger leakage or only partly used. In contrast, a small capacitor can make use of such power boosts better.

3.2. NVP Power Management Unit

The primary focus of this article is on proposed changes to the power management unit, including support for dynamic voltage and frequency scaling. This unit contains three main components: (1) Backup and Recovery Unit (BRU) [26], (2) Energy Policy Unit (EPU), and (3) Clock Frequency and Voltage Tuning Unit (CVU). The BRU is triggered by the EPU to initiate a backup or system restore. The BRU initiates transferring the data from the volatile registers and flip-flops to their nonvolatile counterparts. Specifically, it generates the sequence of state backup/recovery signals like “PC Start

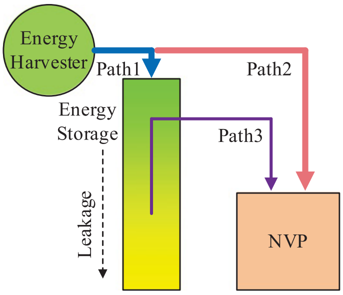


Fig. 7. Key energy paths.

B/R" and "RegFile Start B/R," and sets/resets the atomic flag. The BRU contains a flag to indicate safe storage of the backup state [26]. This flag along with a double buffering of the system state is used to ensure safe recovery in case the backup operation itself is not successful.

The EPU is triggered through a "Warning" signal as long as the input power trace is below a specified threshold. A weak power input is indicative of an imminent power outage. Consequently, the EPU overrides the current CVU clock and voltage setting policy and sets the NVP to the lowest possible frequency. This policy is selected to delay the running out of stored energy in anticipating that the power interruption is transient. The EPU also triggers the BRU if the stored energy reaches a critical threshold to guarantee safe backup in nonvolatile memory.

The CVU controls the system clock and voltage except when the EPU overrides it. The CVU implements the clock frequency policies discussed in Sections 5 and 5.5 based on the input power profile as well as the stored energy. The clock is also gated through the "gated clock control" signal via an AND gate to ensure reliable switching frequency.

The state machines implementing the control policies are triggered at 200ms granularities (for RF power sources) and 1 second (for solar power source) and contribute a negligible power overhead to our system.

4. ENERGY-INCOME SPECULATION

The focus of this section is how to efficiently use the energy stored in the capacitor when the power income is lower than the threshold of the system operation (optimizing *path3* in Figure 7). The optimization target thus balances improving the efficiency of forward progress while maintaining a low number of power emergencies. While the cost of nonvolatile backups is high, their nonvolatility means that speculative approaches where misspeculation results in energy storage capacitor depletion can still be practically considered.

We study the optimization of the storage-to-compute path from an energy storage capacitor design perspective. Based on the results of these explorations, we propose a more aggressive backup policy that can exploit the nonzero level of input power during an impending outage to accomplish additional work prior to a backup operation, and that can elide some backups entirely. This policy comes with the cost of potential rollbacks (requiring double-buffering of checkpoints), but these prove to be very rare in practice.

One source of inefficiency is the single, conservative, and absolute threshold for initiating backup. While, in the worst case, input power could conceivably immediately drop to zero at the time that minimum required backup energy is reached in the capacitor, in practice, input power is not zero during the backup process: in such situations input power is often merely not quite high enough to keep the capacitor

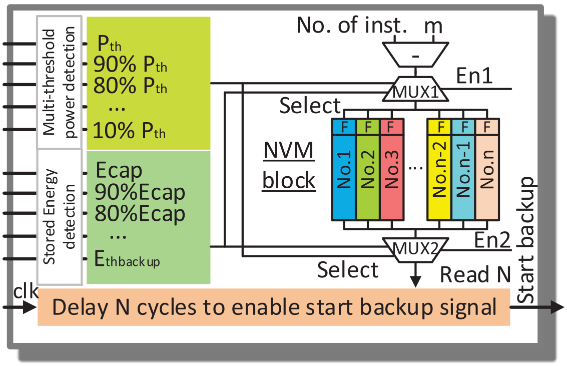


Fig. 8. ALD prediction structure.

adequately charged. We consider a simple estimation strategy, Adaptive Learning Detection (ALD), to capitalize on this difference, and potentially avoid backups altogether in situations where the input power is oscillating around the minimal threshold for operation. In particular, we aim to learn how much additional work we are likely to be able to complete in addition to a backup, given the power input at the time that a back-up would otherwise have been triggered. Using a simple double-buffering checkpointing scheme with an additional valid bit, we can delay the point of backup by the previously learned number of instructions and potentially avoid a backup altogether if power recovers during the delay period.

Figure 8 shows the structure of ALD. We add detectors to both incoming power and stored energy. The main function of the ALD is to decide when to output the "Start backup" signal to the "Backup/Recovery State Machine." Once an energy emergency occurs, it performs a lookup into a 10-entry table of stored instruction counts, indexed by the power income level, discretized into 10 levels. Ten entries are designed because more entries lead to fine-grained instruction numbers. But the energy per instruction varies across test benches, and even within one specific test bench, average energy per instruction varies among different time slots, which may lead to more backup failures. Fewer entries lead to a waste of efficiency because the instruction numbers, if well trained, should be conservative, which means there is larger potential that more instructions can be executed before backup operation. Considering the two boundaries, a 10-entry solution is selected in this design. If no prediction exists (entry invalid), it performs a backup immediately and then uses the amount of energy remaining in the system after backup to estimate the number of additional instructions that could have been executed using a conservative per-instruction cost model and stores this value in the table. If a valid entry exists, the system attempts to delay backup by the indicated number of instructions. To do this, it first marks the prediction as invalid. Three outcomes are possible: First, the delay is conservative, but power is still failing, and both the additional instructions and the backup occur. Second, the delay is conservative, and sufficient energy storage levels are restored by the end of the delay. In this case, no backup actually occurs. In either of the first two cases, the valid bit is reset to true. Third, the delay was too aggressive, and energy is depleted during either instruction execution or midbackup. In either failure, the more recent of the two checkpoints will not have been marked valid at completion, and a rollback to the previous checkpoint will occur. Since the prediction was marked invalid at the beginning in NVM, this will persist across the power failure to remove a bad prediction.

By applying the ALD, we can achieve greater energy usage efficiency of the stored energy. As long as the heuristic for mapping from postbackup energy to additional

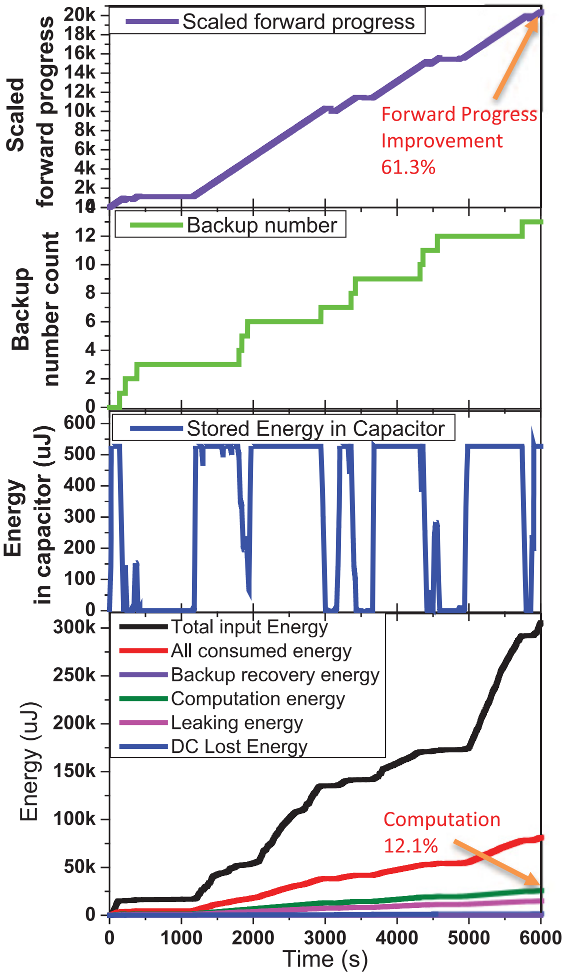


Fig. 9. Evaluation of ALD.

instructions is calibrated conservatively, we can increase energy exploitation with negligible incurred rollbacks. Results in Figure 9 show that, for the RF trace, ALD can improve the computation energy percentage from 7.5% to 12.1%, while maintaining the same backup number of 13.

5. DYNAMIC POWER ADAPTATION

In this section, we propose power optimization policies to improve energy utilization efficiency. For all of these policies, we attempt to adjust both the load and the relative roles of the three paths in Figure 7 in order to maintain the energy level in the capacitor between 70% and 90% of capacity. This range balances the ability to store additional incoming energy during power upticks with front-end charging efficiency: for optimal efficiency, the power consumption in *path2* needs to be kept slightly below the input power, leaving the rest of the power dissipated through *path1* to compensate for capacitor leakage. We examine policies for both dynamic frequency scaling and dynamic voltage and frequency scaling as the primary means to alter processor demand.

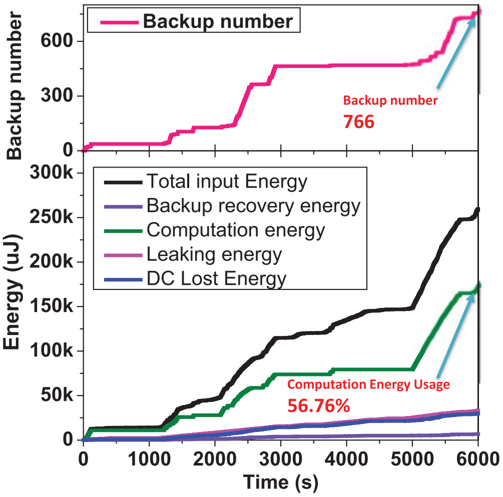


Fig. 10. Forward progress at frequency 672kHz.

In order to modulate the NVP power dissipation rate, the system frequency is set dynamically to tune the dynamic power of the NVP chip (DFS). To enable DFS, multiple oscillators and clock dividers are employed in this work. A simple two-stage process selects and synchronizes across the generated clocks to select the proper frequency for the NVP as the system clock in Figure 13. The latency of this MUX-based selection among multiple clocks has a two-clock-cycle penalty, which is included in the model.

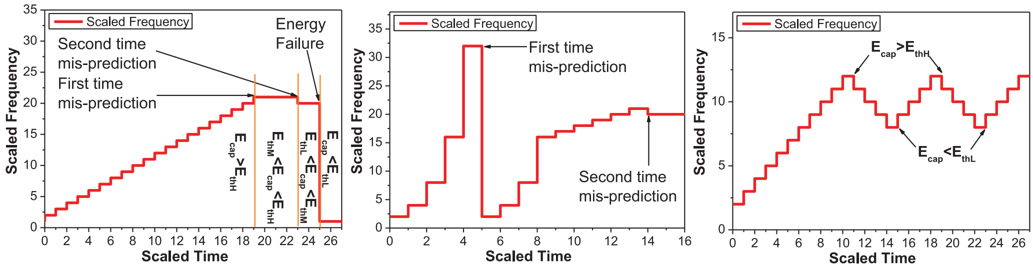
Later, we introduce three reactive polices, operating at coarse temporal granularity, to dynamically set the optimum frequency for the NVP. We then examine improving the decision latency for the policy engines by introducing a learning mechanism that records previous mappings from power and storage levels to preferred operating frequency. Reducing decision latency can improve the degree to which energy consumption can track available energy. We limit ourselves to a coarse (200ms) granularity for our policy engines because this is the finest granularity for some of our input traces. All of the policies are evaluated on the 100-minute RF power trace. Finer granularities of adaptation may provide additional opportunities to avoid waste but would also incur greater overheads due to the more frequent operation of the policy engine itself.

All three of our DFS policies are only employed when the input power levels are above a preferred threshold (2.55uW for RF, and 1.76mW for solar): our default state is our lowest frequency, and DFS is employed primarily for upscaling to exploit additional available energy, rather than downscaling to reduce consumption. At every (200ms) timestep, excluding power emergencies, the policy engine will sample power and storage inputs, will update its state, and may or may not effect a frequency change.

For comparison, the best forward progress with a static frequency for the 100-minute RF trace occurs at 672kHz, as shown in Figure 10. The energy consumption details of this static configuration are shown in Figure 10. Computation consumes 56.76% of the input energy and there are 766 backups.

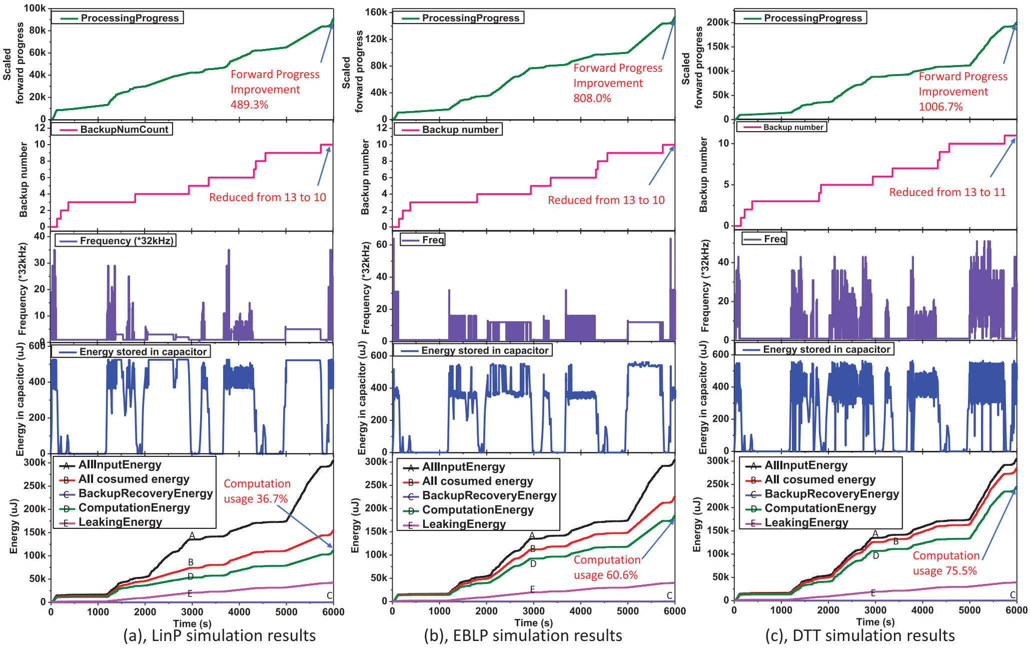
5.1. Linear Policy (LinP)

LinP increases the NVP frequency linearly by one step at a time, as seen in Figure 11(a) (each time step is 200ms). To implement LinP, we need four energy-level detectors, with equal energy gaps between E_{thH} and E_{thM}, E_{thM} and E_{thL}. LinP activates only when the energy level is higher than E_{thH} and continues to increase frequency until the energy



(a), LinP (b), EBLP (c), DTT

Fig. 11. Illustration of the three reactive policies.



(a), LinP simulation results (b), EBLP simulation results (c), DTT simulation results

Fig. 12. Simulation results for LinP, EBLP, and DTT.

storage capacitor energy is no longer greater than E_{thH}, indicating power consumption in excess of harvested power. The appropriate frequency should be near this point, and LinP holds the frequency stable until energy levels pass E_{thM}, at which point the frequency is reduced by another step. If the energy level reaches the backup trigger, $E_{\text{thebackup}}$, the frequency is reset to the minimum for the backup operation.

The results in Figure 12(c) indicate that LinP is able to increase the energy utilization percentage from the (backup-minimizing) baseline of 7.5% to 36.7% with RoO decreasing from 13 to 10. However, because of the large input power range, the LinP sometimes fails to keep the energy stored in the capacitor between E_{thL} and E_{thH}. For example, during the time region from 4,900s to 5,700s, the stored energy is always full in Figure 12(c), which suggests that some input energy is wasted due to the fully charged capacitor. The main reason is that the frequency tuning is not sufficiently fast due to the linear increase and coarse granularity of the policy cycle with respect to the volatility of input power.

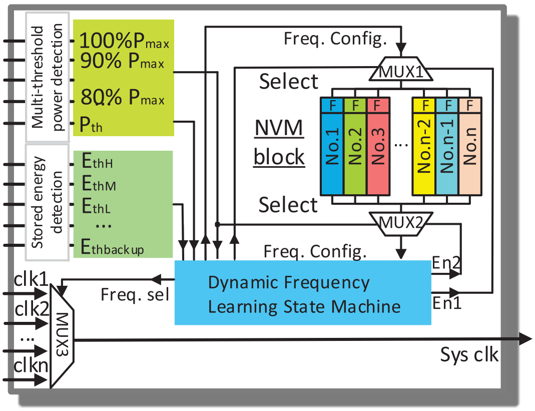


Fig. 13. Dynamic Frequency Learning (DFL) Structure.

5.2. Exponential-bottom-linear Policy (EBLP)

To improve reactivity to input peaks, we consider the combined exponential/linear policy depicted in Figure 11(b). As with LinP, the EBLP only activates when storage levels are above the optimal range. In EBLP, the frequency first increases exponentially. Once it triggers the first indication of overshoot (stored energy decreased below threshold), it will switch to an exponential-and-then-linear policy that transitions from exponential to linear at half the frequency of the first overshoot frequency. The reason that the exponential increase policy is employed after the first misprediction rather than jumping immediately to a new frequency is to restore capacitor energy drained during the frequency overshoot.

The result in Figure 12(b) shows that the computation energy efficiency improves to 60.6%, which is higher than the efficiency of the best static frequency optimization for “Forward Progress,” which yields 56.76%, as seen in Figure 10. At the same time, the number of power outages reduces from 13 (best static frequency optimizing for backup reduction) to 10. Thus, this dynamic approach is superior in both metrics compared to systems statically optimized for either.

5.3. Double Threshold Tracking (DTT)

In DTT, two thresholds are incorporated as shown in Figure 11(c). The scaled frequency first increases from a low value (e.g. “1” in Figure 11(c)), then increases linearly. Once it triggers the high threshold E_{thH}, the frequency decreases linearly. And the frequency increases when E_{thL} is triggered. The drawback is that the frequency is always changing back and forth within a small range.

The results in Figure 12(c) show that the computation energy usage improves to 75.5%, a 1.33× improvement over the 56.76% execution consumption of the best static “Forward Progress” baseline with an improvement in the rate of backups over the best static backup-optimized baseline from 13 to 11.

5.4. Dynamic Frequency Learning (DFL)

LinP, EBLP, and DTT aim to predict the best frequency for NVP operation. Practically, for a given <input power level, stored> energy level tuple, it is reasonable to assume that the previously selected frequency for that pair is a good guess for the right frequency to operate at now. Figure 13 shows a simple lookup-table-based mechanism for predicting operating frequencies given power and energy levels. The 100 entry × 6 bit (5+valid) table encodes the frequency selection value based on an <input power level,

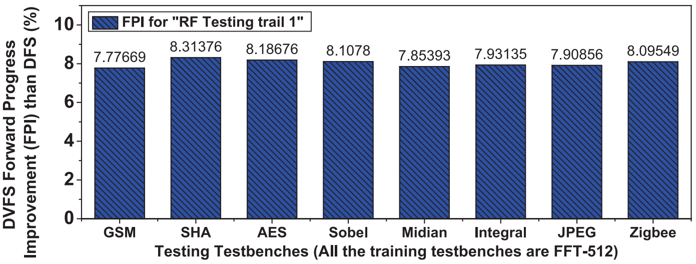


Fig. 14. Forward progress improvement with DVR ranging from 32kHz to 1MHz.

stored> tuple where each of the dimensions has 10 possible values. The goal of DFL is to further reduce the search time required to move the system to a stable frequency when there is additional exploitable energy.

DFL begins in a prediction state. If the tuple indexes an invalid entry, DFL enters a search state, where the reactive policy is employed to locate a stable frequency. After the second overshoot for LinP, or the equivalent points in DTT or EBLP are reached, the selected frequency is set in the lookup table, and DFL returns to the prediction state. If the valid bit is set for the currently indexed entry, the stored frequency is selected directly, and the DFL enters its validation state. In validation mode, the DFL checks, over the next 10s of timesteps (a fixed value from 10 to 20 timesteps, depending on which of the three reactive policies is employed in the search state), whether the input energy remains in the current discretized (10 levels) band; that is, the predicted frequency is neither far too high nor far too low for exploiting the input scenario. If the energy storage band changes in this period, the prediction is marked as invalid and the DFL returns to the prediction state. Otherwise, after leaving the validation state, the DFL returns to the prediction state.

As per the other policies, DFL operation is only triggered when total storage capacitor energy is above the optimal capacitor energy threshold, and the policy engine operates only once per 200ms. We examine the potential of DFL separately for each of LinP, EBLP, and DTT. Compared to reactive policies alone, DFL shows a forward progress improvement (FPI) of 65.4% for LinP, 17.1% for EBLP, and 1.5% for DTT while maintaining the exact same backups as the solely reactive policies. With DFL, any of the three reactive policies can achieve 70+% conversion of input energy to computation energy, whereas previously only DTT achieved this. However, the improvements over baseline DTT are small.

5.5. DVFS

Dynamic Voltage Range (DVR) can also be applied with DFS as DVFS. This is a traditional way to save further power over DFS at the cost of some additional circuit complexity. We can directly extend the previous frequency scaling techniques with commensurate voltage scaling. Nonvolatile logic can add additional challenges to voltage scaling, potentially limiting the dynamic range, as mentioned in Section 3. Figure 14 shows that moving from DFS to DVFS provides an additional 8% forward progress on the RF power trace across most benchmarks.

6. UBIQUITOUS TESTING FOR DIFFERENT POWER PROFILES, TEST BENCHES, AND ENERGY SOURCES

This section integrates all the policies discussed in earlier sections. Specifically, we incorporate the learning from the ALD policy that determines how many instructions can be executed before a backup needs to be triggered when the input power is less

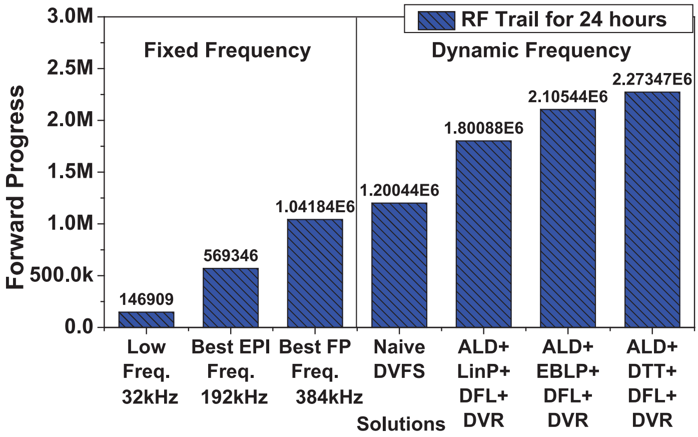


Fig. 15. Comparison of the forward progress for each policy with a 24-hour RF trail.

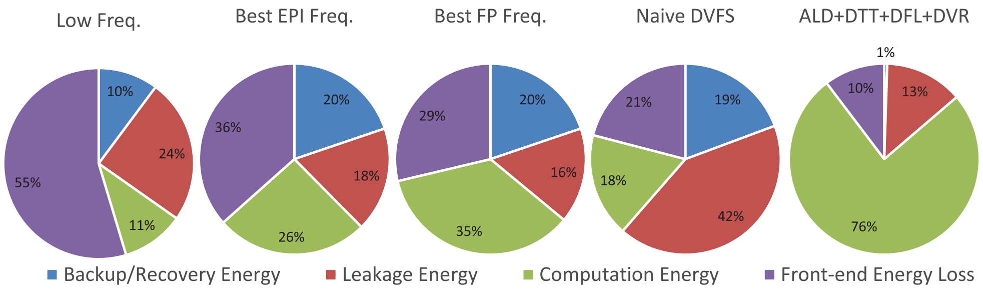


Fig. 16. Energy dissipation for each policy powered by RF.

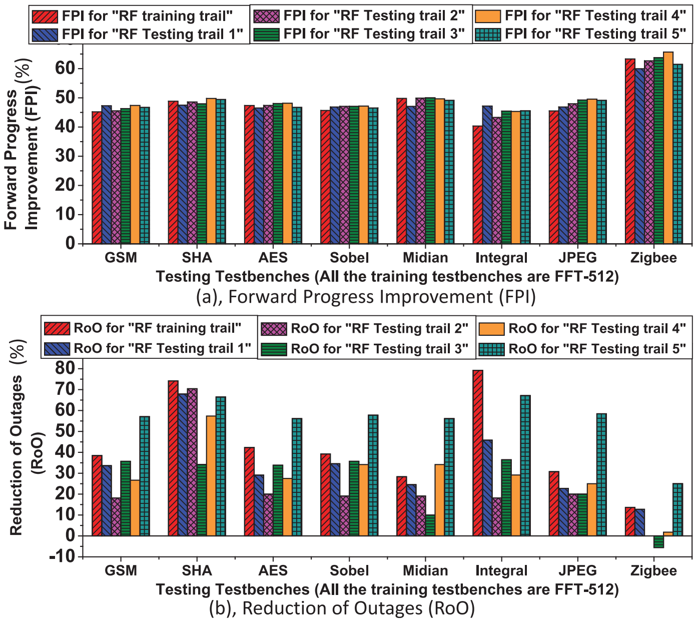
than 2.55uW for RF (1.76mW for solar). We use LinP, EBLP, and DTT to aggressively use the income power, but not to store them. We use DFL to speed up and learn the best frequency setup. We use DVR for DVS to try to reduce the EPI, for a better forward progress. These policies operate and update every 200ms.

6.1. Different RF Trail Results Analysis

In this subsection, we apply our policies to various power trails. In this section, we test a 24-hour RF trail. Different trails can support different max frequencies due to different power levels. For this trail, the max frequency that can be supported is 448kHz.

Figure 15 shows the forward progress comparison for different policies. The naive DVFS policy works like this: there are several energy thresholds for the energy storage capacitor. Once the stored energy level in the capacitor is high from normal, frequency increases at a various step. And frequency reduces when the capacitor energy level is low. If the energy level is very low, backup operation will be triggered. As we can see in Figure 15 and Figure 18, neither low frequency nor high frequency can have the highest forward progress. The forward progress with frequency and voltage at the best energy per instruction (EPI) is definitely not the best. For a static frequency of 384kHz, the best forward progress is achieved.

As shown in Figure 16, the portion of computation energy of “Best FP Freq.” (35%) is lower than that of “High Freq.” (39%). This is because of the complex tradeoffs inside the nonvolatile system, including the EPI.



(a), Forward Progress Improvement (FPI)

(b), Reduction of Outages (RoO)

Fig. 17. (a) FPI over the best statically optimized forward progress baseline, and (b) RoO over the best statically optimized least outage baseline, when warmed with RF training trace and tested on five different power traces across eight benchmarks.

We combine ALD, LinP, EBLP, DTT, DFL, and DVR together and test the forward progress. The solution of ALD+DTT+FL+DVR has the best forward progress, with 89% forward progress improvement over the “Naive DVFS” solution, and the portion of computation energy is as high as 76%, as shown in Figure 16. ALD+LinP+DFL+DVR may work better when the input power level is even lower, because LinP policies step the frequency slowly, which can provide enough time for the energy storage capacitor to be filled. ALD+EBLP+DFL+DVR may work well when the power level is relatively high, but with little variation. ALD+DTT+DFL+DVR may fit to scenarios in which the power variation speed is low.

6.2. Training and Testing with Different RF Trails and Different Test Benches

We used five different RF traces, each 1,000 minutes long, while executing different applications to investigate the effectiveness of these combined learning mechanisms for different training and testing of DFL. The entries in the learning table were initialized based on the training with an independent 100-minute-long RF trace while executing FFT. Figure 17(a) shows the FPI of ALD+DTT+DFL+DVR as compared to a baseline strategy that finds the best fixed frequency setting that maximizes forward progress (by sweeping all possible fixed frequency operations of the processor). Furthermore, the baseline strategy triggers backup conservatively when the energy storage capacitor falls below the minimum energy required for successful backup operation. In contrast, the backup strategy in ALD+DTT+DFL+DVR is aggressive in letting the energy storage capacitor capacity go below the required threshold by predicting a nonzero power input in the future. The aggressive policy uses the double-buffering approach (which is invoked less than 1% in our simulations) as a way to ensure safe forward progress. The ALD+DTT+DFL+DVR provides around 54% improvement in forward progress, as shown in Figure 17(a). There is a variance in the improvement depending on the workload. There are two reasons for this variance. The applications such as Zigbee

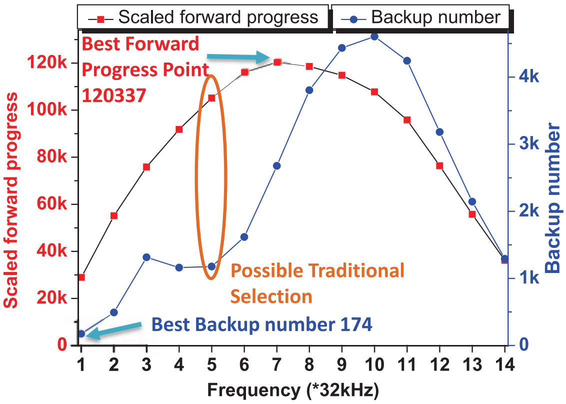


Fig. 18. Sweep over 32 static frequencies showing the tradeoff between maximum forward progress and minimum power outages (solar power trace).

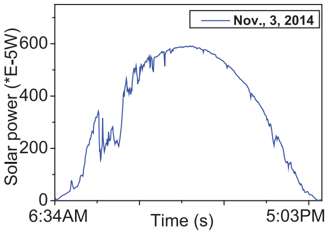


Fig. 19. Training solar trails.

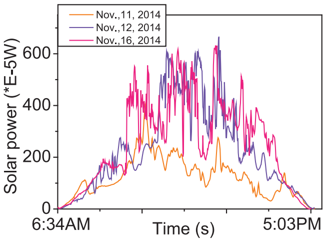


Fig. 20. Testing solar trails.

that had a smaller energy per instruction benefited more from the aggressive adaptive thresholds for backup energy. Further, the applications that exhibited similar energy consumption as FFT benefited from the initial setting better than others. However, due to the DFL, this difference is not large.

We also observe that the number of power outages using ALD+DTT+DFL+DVR are lower than that obtained by the best fixed frequency setting that minimized the number of outages (by sweeping all possible 32 fixed frequency operations of the processor). It is observed from Figure 17(b) that the number of outages can be reduced due to the aggressive backup.

Different test benches have different energy per instruction. This will lead to variations in ALD; for example, if the ALD is preheated by a test bench with small energy per instruction, once a different test bench is applied in execution, there is a larger potential that the backup operation will be a failure, which may bring huge rollback penalties to the system. If ALD is preheated by a large energy per instruction test bench while testing on a small energy per instruction test bench, the prediction is too conservative, leading to energy waste. In the later evaluation section, the FFT-512 test bench is used for preheating while testing on other test benches. This is to evaluate the influence of energy per instruction crossing different test benches. For frequency scaling, similar influence is observed.

6.3. Solar Trails Simulation Results and Discussion

We also use four different solar traces, each 1 day long, while executing different applications as shown in Figure 20. The entries in the learning table were initialized based on the training with an independent solar trace, 1 day long, using FFT as workload as

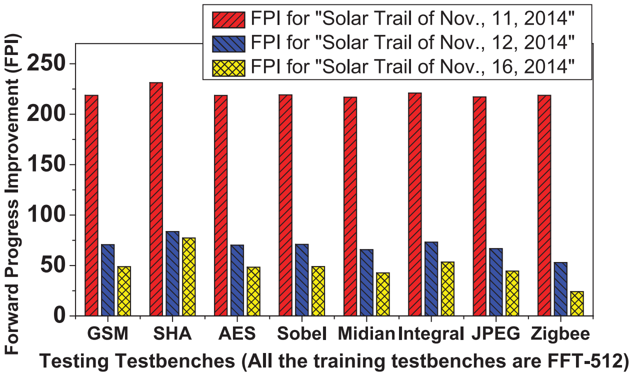


Fig. 21. FPI over the best statically optimized forward progress baseline for solar power trails tested on three different power traces across eight benchmarks.

shown in Figure 19. In the solar trace, the training was based on an input profile from a sunny day, while testing was based on a cloudy day where power supply was more erratic and variant. However, we observe that the proposed technique is still efficient in enhancing FPI (see Figure 21). Further, we observed that due to the higher input power of the solar power scavenging, the number of outages was drastically reduced using the aggressive policy of ALD+DTT+DFL+DVR (in most cases to no outages).

6.4. Limitations and Future Work

The power and energy solution proposed by this work can improve the forward progress remarkably with small area penalty. But more details need to be considered when implementing the system; for instance, the glitches caused by the switching frequency need to be handled by waiting some cycles. About future improvement, we are considering applying machine-learning algorithms to the control logic to help predict the frequency.

7. RELATED WORK

7.1. Power Management Technique in Energy-Harvesting Systems

Power management is critical in energy-harvesting systems, not only due to the limited maximum energy but also due to the spatiotemporal energy availability. The environmental energy-aware task scheduling is first proposed by Kansal and Srivastava [13] by using a distributed framework to adaptively learn its energy environment. Kansal et al. [12] further develops a general model for achievable performance based on the environmental energy resources. These techniques rely on a compiler or software interface for task allocation across the sensor nodes. Recent circuit designs adopt the maximum power tracking techniques to maximize the energy utilization efficiency for different energy harvesters [3, 37]. Architectural power management has also been investigated by Baglio et al. [1], Christmann et al. [4], Ma et al. [26], Vanhecke et al. [36], and Yaqub et al. [42] with respect to the harvester circuit, the processor complexity, the source of harvested energy, and energy storage devices. Different from the previous works, our article focuses on the power/energy control path design for the dynamic frequency setting. The innovation of the work is that it further pushed the traditional DVFS optimization methods to new energy-harvesting applications with unstable power supply. The target of the work concentrates on maximum forward progress, backup numbers, and new evaluation parameters that are more meaningful from application and user levels. Rather than just using logic, this work takes advantage of nonolatile memory within the processor to design lookup tables, learning, and preheating techniques.

7.2. Energy Storage and Power System Design

Various energy storage components have been explored in various energy-harvesting platforms [4, 29, 32] for enhancing system performance. Battery-assisted energy storage is discussed by Christmann et al. [4] and Porcarelli et al. [29], and the supercapacitor is used in works by Sheng et al. [32]. Because the supercapacitor offers higher energy efficiency compared to the battery [2], our work uses capacitance-based energy storage. Our work also applies an adaptive capacitor sizing technique (multi-small-cap) to mitigate higher leakage associated with the supercapacitor as compared to the batteries. The new counter-intuitive insight brought by this work is that increasing the capacitor volume alone does not help for maximum forward progress; on the contrary, it may harm the forward progress due to large leakage. Another significant reason to avoid a large capacitor is that it reduces the chances that the processor can run because it always requires a large power burst to charge the capacitor to a minimum start-up voltage threshold. These insights are novel and counter-intuitive for the energy-harvesting system.

7.3. Application and Resource Scheduling

Resource management and application scheduling serve as another design aspect in energy-harvesting systems to ensure the quality of service and system performance. By using the workload management framework and runtime scheduling scheme, Siegel et al. [34] and Xiang and Pasricha [40-41] investigate the intertask dependency, reliability-to-performance optimization, and runtime variations. Our work focuses on the hardware-oriented implementations of energy policies to enhance the forward progress, which isolates the programmers and operation system.

7.4. Dynamic Voltage and Frequency Scaling

The DVFS technique is well known in power-aware system design [6, 7, 19, 28, 30, 39]. Our DVFS policies must contend with more rapid power fluctuations and frequent outright power failures than traditional systems employing DVFS, and be able to optimize against backup overheads in a nonvolatile processor. In practice, our DVFS policies are effectively used to boost performance in times of high-power availability to more aggressively leverage the income power, rather than for improving EPI efficiency, which has parallels with Intel's TurboBoost 2.0 approach [10].

7.5. Most Recent Research Progress

During the review progress of this work, more research works have been proposed in the nonvolatile processor domain. New devices [17] and corresponding circuits [8] have been proposed to be applied to NVP [18]. Novel architectures have been explored for NVP [20–25]. Ma et al. propose dynamic architecture to dynamically switch among different architectures [22]. Beyond that, a machine-learning-based frequency and resource scaling method is proposed to manage the power and energy of the whole system [24]. A machine learning based dynamic frequency scaling and resource allocation method is proposed [24]. Higher-level supports in compiler level [11], programming language [5], and operating system level [35] are also explored.

8. CONCLUSION

The variability of input power in energy-harvesting systems limits the effectiveness of static optimizations aiming to maximize the input-energy-to-computation ratio. We apply some of the traditional schemes like DVFS to these new application scenarios under energy harvesting an unstable power supply. In order to fit into the new applications, preheating, dynamic learning, and lookup table methods are introduced to overcome the gap between traditional techniques and unstable power supply. Aiming at maximum forward progress, we introduce a unified energy-oriented approach to first optimize the number of power failures, by more aggressively using the stored energy available when power failure occurs, and then optimize forward progress via improving the rate of input energy to computation via dynamic voltage and frequency scaling. We evaluate combining these two schemes and show capture of up to 75.5% of all input energy toward processor computation, an average of 1.54× forward progress increase over the best static frequency case.

REFERENCES

[1] S. Baglio, C. Trigona, B. Ando, F. Maiorca, G. L'Episcopo, and A. Beninato. 2012. Energy harvesting from weak random vibrations: Bistable strategies and architectures for MEMS devices. In *Proceedings of the 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS'12)*. 154–157. DOI:http://dx.doi.org/10.1109/MWSCAS.2012.6291980

[2] V. A. Boicea. 2014. Energy storage technologies: The past and the present. *Proc. IEEE* 102, 11 (Nov. 2014), 1777–1794. DOI:http://dx.doi.org/10.1109/JPROC.2014.2359545

[3] D. Brunelli, L. Benini, C. Moser, and L. Thiele. 2008. An efficient solar energy harvester for wireless sensor nodes. In *Proceedings of the Design, Automation and Test in Europe, 2008 (DATE'08)*. 104–109.

[4] J. F. Christmann, E. Beigne, C. Condemine, and J. Willemin. 2010. An innovative and efficient energy harvesting platform architecture for autonomous microsystems. In *Proceedings of the 2010 8th IEEE International NEWCAS Conference (NEWCAS'10)*. 173–176. DOI:http://dx.doi.org/10.1109/NEWCAS.2010.5603747

[5] A. Colin and B. Lucia. 2016. Chain: Tasks and channels for reliable intermittent programs. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, 514–530.

[6] X. Cui, K. Ma, K. Liao, N. Liao, D. Wu, W. Wei, R. Li, and D. Yu. 2013. A dynamic-adjusting threshold-voltage scheme for FinFETs low power designs. In *Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS'13)*. 129–132.

[7] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini. 2012. CoScale: Coordinating CPU and memory system DVFS in server systems. In *45th Annual IEEE/ACM International Symposium on Microarchitecture*. 143–154.

[8] S. George, K. Ma, A. Aziz, X. Li, A. Khan, S. Salahuddin, M.-F. Chang, S. Datta, J. Sampson, S. Gupta, and V. Narayanan. 2016. Nonvolatile memory design based on ferroelectric FETs. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 118.

[9] M. A. Green, K. Emery, Y. Hishikawa, W. Warta, and E. D. Dunlop. 2014. Solar cell efficiency tables (version 43). *Progress Photovoltaics Res. Appl.* 22 (2014), 1–9.

[10] Intel. Intel®turbo boost technology 2.0. http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html.

[11] M. Kaisheng, L. M. Julie, L. Xueqing, H. Zhixuan, and S. Jack. 2017. Evaluating tradeoffs in granularity and overheads in supporting nonvolatile execution semantics. In *The 18th International Symposium on Quality Electronic Design (ISQED'17)*. Santa Clara.

[12] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. 2007. Power management in energy harvesting sensor networks. *ACM Trans. Embed. Comput. Syst.* 6, 4, Article 32 (Sept. 2007).

[13] A. Kansal and M. B. Srivastava. 2003. An environmental energy harvesting framework for sensor networks. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design, 2003 (ISLPED'03)*. 481–486.

[14] S. Kim, R. Vyas, J. Bito, K. Niotaki, A. Collado, A. Georgiadis, and M. M. Tentzeris. 2014. Ambient RF energy-harvesting technologies for self-sustainable standalone wireless sensor platforms. *Proc. IEEE* 102, 11 (2014), 1649–1666.

[15] H. Kimura, Z. Zhong, Y. Mizuochi, N. Kinouchi, Y. Ichida, and Y. Fujimori. 2013. Highly reliable non-volatile logic circuit technology and its application. In *Proceedings of the 2013 IEEE 43rd International Symposium on Multiple-Valued Logic (ISMVL'13)*. 212–218.

[16] X. Li, U. D. Heo, K. Ma, H. Liu, V. Narayanan, and S. Datta. 2014a. RF-powered systems using steep-slope devices. In *Proceedings of the IEEE International New Circuits and Systems Conference*.

[17] X. Li, H. Liu, U. D. Heo, K. Ma, S. Datta, and V. Narayanan. 2014b. RF-powered systems using steep-slope devices. In *Proceedings of the New Circuits and Systems Conference (NEWCAS'14)*. 73–76.

[18] X. Li, K. Ma, S. George, J. Sampson, and V. Narayanan. 2016. Enabling internet-of-things: Opportunities brought by emerging devices, circuits, and architectures. In *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. 1–6.

[19] X. Lin, Y. Wang, S. Yue, N. Chang, and M. Pedram. 2013. A framework of concurrent task scheduling and dynamic voltage and frequency scaling in real-time embedded systems with energy harvesting. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design (ISLPED'13)*. IEEE Press, 70–75.

[20] Y. Liu, Z. Li, H. Li, Y. Wang, X. Li, K. Ma, S. Li, M.-F. Chang, S. John, Y. Xie, J. Shu, and H. Yang. 2015. Ambient energy harvesting nonvolatile processors: From circuit to system. In *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 150.

[21] K. Ma, X. Li, S. Li, Y. Liu, J. J. Sampson, Y. Xie, and V. Narayanan. 2015a. Nonvolatile processor architecture exploration for energy-harvesting applications. *IEEE Micro* 35, 5 (2015), 32–40.

[22] K. Ma, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan. 2015b. Dynamic machine learning based matching of nonvolatile processor microarchitecture to harvested energy profile. In *Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'15)*. IEEE, 670–675.

[23] K. Ma, X. Li, J. Sampson, Y. Liu, Y. Xie, and V. Narayanan. 2015c. Nonvolatile processor optimization for ambient energy harvesting scenarios. In *Proceedings of the 15th Non-Volatile Memory Technology Symposium.*

[24] K. Ma, X. Li, S. R. Srinivasa, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan. 2017. Spendthrift: Machine learning based resource and frequency scaling for ambient energy harvesting nonvolatile processors. In *Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 678–683.

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

[26] K. Ma, Y. Zheng, S. Li, K. Swaminathan, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan. 2015. Architecture exploration for ambient energy harvesting nonvolatile processors. In *Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA'15)*. 526–537.

[27] P. P. Mercier, S. Bandyopadhyay, A. C. Lysaght, K. M. Stankovic, and A. P. Chandrakasan. 2013. A 78 pW 1 b/s 2.4 GHz radio transmitter for near-zero-power sensing applications. In *2013 Proceedings of the ESSCIRC*. 133–136. DOI:http://dx.doi.org/10.1109/ESSCIRC.2013.6649090

[28] R. Miftakhutdinov, E. Ebrahimi, and Y. N. Patt. 2012. Predicting performance impact of DVFS for realistic memory systems. In *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture*. 155–165.

[29] D. Porcarelli, D. Brunelli, M. Magno, and L. Benini. 2012. A multi-harvester architecture with hybrid storage devices and smart capabilities for low power systems. In *Proceedings of the 2012 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM'12)*. 946–951. DOI:http://dx.doi.org/10.1109/SPEEDAM.2012.6264533

[30] N. B. Rizv and A. Y. Zomaya. 2013. A Primarily Survey on Energy Efficiency in Cloud and Distributed Computing Systems. arXiv preprint (Oct 2013).

[31] S. Roundy, D. Steingart, L. Frechette, P. Wright, and J. Rabaey. 2004. Power sources for wireless sensor networks. In *Wireless Sensor Networks*. Springer, 1–17.

[32] X. Sheng, C. Wang, Y. Liu, H. G. Lee, N. Chang, and H. Yang. 2014. A high-efficiency dual-channel photovoltaic power system for nonvolatile sensor nodes. In *Proceedings of the 2014 IEEE Non-Volatile Memory Systems and Applications Symposium (NVMSA'14)*. 1–2.

[33] K. Shoji, Y. Akiyama, M. Suzuki, N. Nakamura, H. Ohno, and K. Morishima. 2014. Diffusion refueling biofuel cell mountable on insect. In *Proceedings of the 2014 IEEE 27th International Conference on Micro Electro Mechanical Systems (MEMS'14)*. IEEE, 163–166.

[34] H. J. Siegel, B. Khemka, R. Friese, S. Pasricha, A. A. Maciejewski, G. A. Koenig, S. Powers, M. Hilton, R. Rambharos, G. Okonski, and S. Poole. 2014. Energy-aware resource management for computing systems. In *Proceedings of the 2014 Seventh International Conference on Contemporary Computing (IC3'14)*. 7–12.

[35] J. Van Der Woude and M. Hicks. 2016. Intermittent computation without hardware support or programmer intervention. In *Proceedings of 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*. 17.

[36] C. Vanhecke, L. Assouere, A. Wang, P. Durand-Estebe, F. Caignet, J.-M. Dilhac, and M. Bafleur. 2015. Multisource and battery-free energy harvesting architecture for aeronautics applications. *IEEE Transactions on Power Electronics* 30, 6 (2015), 3215–3227.

[37] C. Wang, N. Chang, Y. Kim, S. Park, Y. Liu, H. G. Lee, R. Luo, and H. Yang. 2014. Storage-less and converter-less maximum power point tracking of photovoltaic cells for a nonvolatile microprocessor. In *Proceedings of the 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC'14)*. 379–384.

[38] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Zhao, M.-F. Chiang, Y. Yan, B. Sai, and H. Yang. 2012. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *ESSCIRC (ESSCIRC'12)*. 149–152.

[39] Q. Wu, V. J. Reddi, Y. Wu, J. Lee, D. Connors, D. Brooks, M. Martonosi, and D. W. Clark. 2005. A dynamic compilation framework for controlling microprocessor energy and performance. In *Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture*. 271–282.

[40] Y. Xiang and S. Pasricha. 2014a. Fault-aware application scheduling in low-power embedded systems with energy harvesting. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES'14)*. Article 32, 10 pages.

[41] Y. Xiang and S. Pasricha. 2014b. A hybrid framework for application allocation and scheduling in multicore systems with energy harvesting. In *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*. ACM, 163–168.

[42] R. Yaqub, H. Ahmad, N. A. Boakye-Boateng, and Y. Wang. 2012. System architecture for ride portfolio reporting employing energy harvesting scheme. In *Proceedings of the 2012 International Conference on Connected Vehicles and Expo (ICCVE'12)*. 241–245. DOI:http://dx.doi.org/10.1109/ICCVE.2012.54

Received September 2015; revised November 2016; accepted January 2017