Nonvolatile Processor Architecture Exploration for Energy-Harvesting Applications

Kaisheng Ma
Xueqing Li
Pennsylvania State University

Shuangchen Li
University of California, Santa Barbara

Yongpan Liu
Tsinghua University

John (Jack) Sampson
Pennsylvania State University

Yuan Xie
University of California, Santa Barbara

Vijaykrishnan Narayanan
Pennsylvania State University

This article discusses the design of nonvolatile processors (NVPs) for batteryless applications in the Internet of Things (IoT), in which ambient energy-harvesting techniques provide the power. Achieving reliable, continuous, forward computation with an unstable, intermittent power supply motivates the transition from conventional volatile processors to emerging NVPs. This article provides a guideline for future IoT applications, revealing inherent features of the energy-harvesting NVP design.

The continuous advance of nanoscale circuit technology, improvements in energy-harvesting techniques, and the rising demand for low-power wearable sensors associated with human healthcare signal a promising era of new embedded processors powered by ambient energy sources. The intrinsically unstable, low-power, and intermittent nature of ambient energy sources prevents the direct deployment of conventional digital signal processors designed under stable power assumptions for use in these emerging applications without the risk of frequent loss of progress caused by power failures (see Figure 1). This challenge has driven recent efforts to explore nonvolatile processor (NVP) designs that are not subject to loss of progress.

To deal with power instability, researchers have proposed checkpoint techniques to store the intermediate computation states before power failures occur, through power detecting techniques and external nonvolatile memory storage such as flash and hard disks.[1–5] Still, such processors experience reset and rollbacks when power failures occur. In contrast, an NVP can maintain its intermediate computation states in its on-chip nonvolatile memory during power failures and retrieve them when the power is recovered, thus achieving more forward progress. Given a specific task to both volatile and NVPs under an unstable power supply, as shown in Figure 2, the NVP can finish the task more quickly due to elimination of rollbacks.

NVPs are designed for use in energy-harvested applications that are severely power constrained. However, it is interesting and surprising that design optimization of lowering the power a processor consumes, inherited from traditional processor design, does not guarantee optimized maximum forward

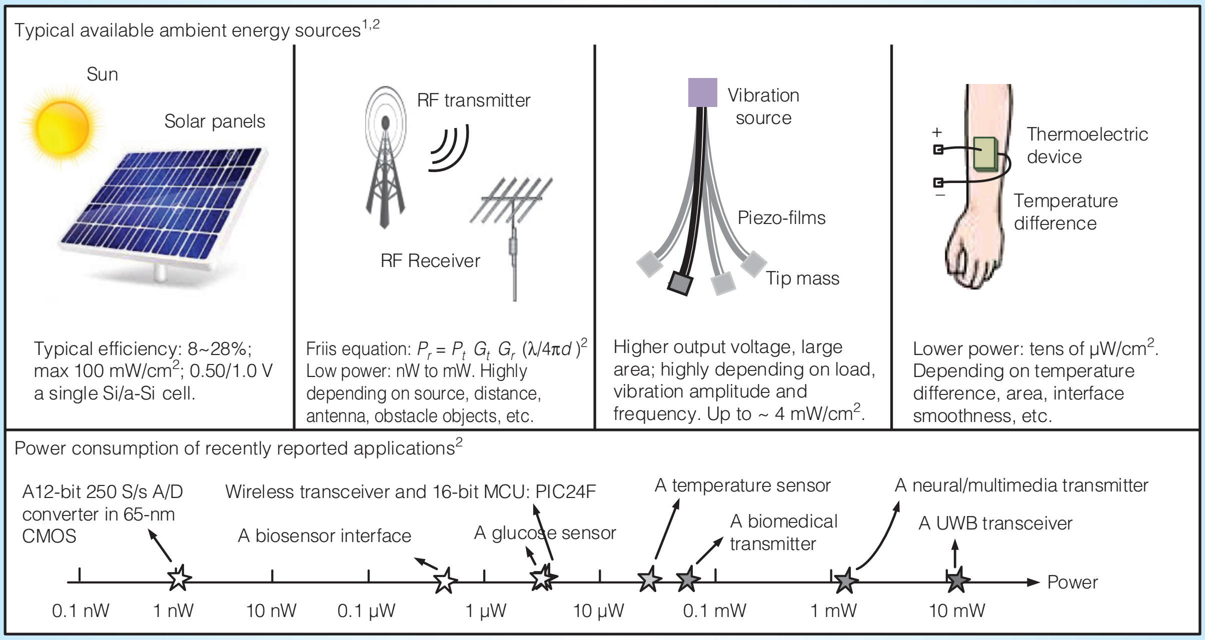
Typical available ambient energy sources[1,2]
Sun
Solar panels
Typical efficiency: 8~28%; max 100 mW/cm^2; 0.50/1.0 V a single Si/a-Si cell.
RF transmitter
RF Receiver
Friis equation: $P_r = P_t\ G_t\ G_r\ (\lambda/4\pi d)^2$ Low power: nW to mW. Highly depending on source, distance, antenna, obstacle objects, etc.
Vibration source
Piezo-films
Tip mass
Higher output voltage, large area; highly depending on load, vibration amplitude and frequency. Up to ~ 4 mW/cm^2.
Thermoelectric device
Temperature difference
Lower power: tens of μW/cm^2. Depending on temperature difference, area, interface smoothness, etc.
Power consumption of recently reported applications[2]
A12-bit 250 S/s A/D converter in 65-nm CMOS
Wireless transceiver and 16-bit MCU: PIC24F
A biosensor interface
A glucose sensor
A temperature sensor
A biomedical transmitter
A neural/multimedia transmitter
A UWB transceiver
Power
0.1 nW
1 nW
10 nW
0.1 μW
1 μW
10 μW
0.1 mW
1 mW
10 mW

Figure 1. Ambient energy sources including solar energy, ambient RF energy, vibration energy, and thermal energy. Energy features for power strength, and related parameters are shown along with state-of-the-art application power levels.

progress for an NVP powered by intermittent ambient energy. Although there are similarities to run-to-halt architectures, the root causes in batteryless systems have distinct features. Namely, without energy storage, any power in excess of what the processor can use to compute is rejected and wasted. Thus, it pays to be greedy in harvesting systems.

To optimize the processor's forward progress, various factors should be considered, including the overhead due to different backup and recovery operation policies, the ambient energy characteristics, the selection of processor architectures with different complexities, and the type of nonvolatile memory storage. In this article, we address how to build an NVP, its design tradeoffs, and how different optimization choices lead to varying performance in typical applications, which could guide energy-harvesting processor design explorations.

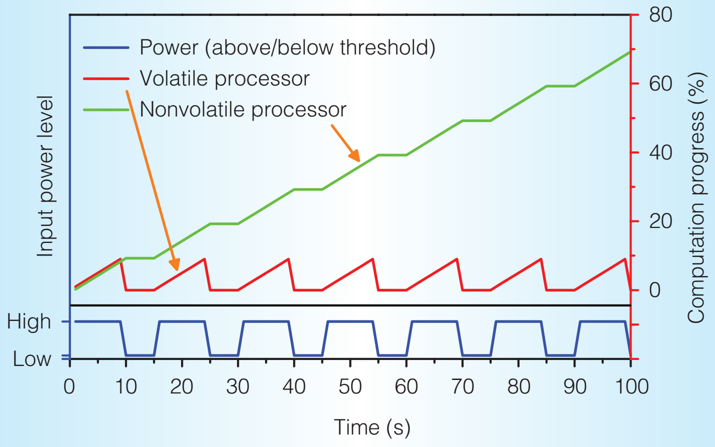

Figure 2. Percentage of computation progress of a volatile processor (VP) and a nonvolatile processor (NVP) under an unstable power condition. Unlike a VP, an NVP's on-chip backup mechanisms ensure monotonic forward progress.

From volatile to nonvolatile processors

As Figure 3 shows, a volatile processor (VP) resets after power interruptions. In contrast, an NVP with built-in nonvolatile memory can back up the intermediate state on the chip when a power failure occurs, and restore the processor state when power comes back. The NVP's backup and recovery operations are at the instruction level and transparent to programmers and compilers.

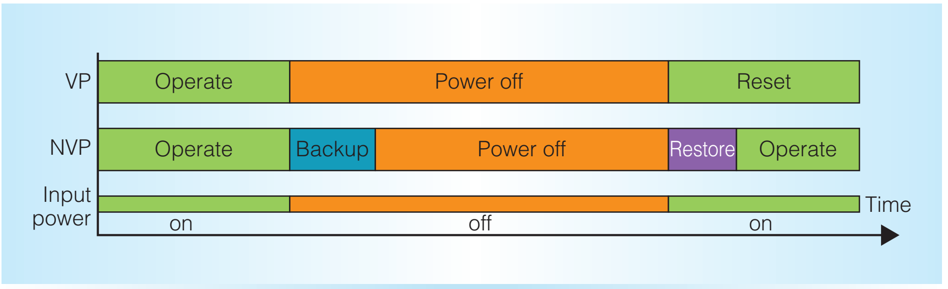

Figure 3. Fine-grained states of VPs and NVPs under unstable power supply. An NVP enters a backup phase after power failure, during which it uses the energy stored in its energy storage capacitor to back up the intermediate computation state. When power is available again, the supporting circuits first make sure that there is sufficient energy stored for the next potential backup, and only then restore the computation state and resume execution. The VP, in contrast, merely resets and resumes.

Backup

A passive capacitor provides the energy for backup. Unlike a battery, capacitors have no lifetime limitation of charging and discharging cycles, which reduces the risk of chemical exposure and provides a lighter weight. The capacitor size is a design knob that impacts the backup strategy and efficiency. When there is no capacitor to provide any guaranteed backup energy, the backup should occur before power loss. Therefore, backup and recovery at a fine granularity are required (for example, backup at every cycle). However, the fraction of harvested energy used for backup becomes high. Consequently, it is prudent to use at least a small capacitor.

What to back up. We need to back up the architecture state, microarchitecture state, and performance enhancers. The architecture state includes the memory, register files, and program counter. The memory includes the main memory, instruction cache, and data cache. All the state elements mentioned earlier should either be designed as nonvolatile memory or backed up to nonvolatile memory. The architecture state is the minimum state required to be backed up for an NVP design.

The microarchitecture state includes pipeline latches, the reorder buffer (RoB), the load/store queue, architectural register files, physical register files, ready table, free list, and map table. Some states do not require backup because they can be recalculated. For example, architectural register files can be recalculated using the physical register and map table. However, the recalculation leads to extra time and energy penalties and requires complex architectural redesign. Moreover, the more a microarchitectural state is backed up, the less instruction reexecution is required. A better solution is dynamically incremental backup of the microarchitecture state, according to the expected energy received during the backup period.

The performance enhancers include the branch history table and branch target buffer. Backing up these prediction tables can improve the prediction accuracy, because there is no need to rebuild these tables. The tradeoff lies between the energy consumed for more backup and that saved for future execution with improved prediction accuracy.

How to back up. We perform a temporal selective backup and data compression before backup. For the former method, instead of backing up all the data, an alternative method is the temporal selective backup, which backs up only the changed data. The penalty is a flag bit to indicate if the data has been overlapped and written. In addition, a complex control logic is required to read the flag bits.

For the compression before backup method, data compression could reduce the amount of data needing backup. The penalty is extra compression circuits. Therefore, we must consider the tradeoff between compression energy and backup energy.

When to back up. There are two main ways to back up data; periodically and on demand.

In the former method, the processor periodically checkpoints the processor state to the nonvolatile part. The more frequent the processor checkpoints, the fewer the rollbacks needed after recovery. The extreme scenario is to back up every clock cycle, where no rollbacks are ever needed. Two methods are proposed to implement the checkpoint solution:

- *Instruction counter* is a small counter to count instruction numbers between two backup intervals. It can support dynamic interval settings to adjust the backup intervals to power profiles. Larger intervals for backup risk a large rollback penalty to reduce wakeup overheads.
- *Compiler-assisted flag* is a method in which the compiler can add one bit to the program counter to indicate the backup operation. This method benefits from simplicity, but different execution paths of the instructions due to branch instructions need consideration. Moreover, the interval is fixed during compilation time, hurting flexibility.

Backup on demand means that the processor backs up the state only if there is a power failure. This method can significantly reduce unnecessary backup operations. This method's overheads include constantly monitoring the power supply and identifying the trigger threshold for backup initiation. The other penalty is that it requires a capacitor to store energy to ensure successful backup operation after power failure occurs. The larger energy store required increases the system recovery latency. Before resuming computation, the system must first accumulate enough energy to guarantee sufficient energy for the next backup operation.

Where to back up. The two primary methods for where to perform backup are the all-in-parallel method and via a centralized nonvolatile block. The former method requires distributed nonvolatile storage structures, such as nonvolatile SRAM[4] or nonvolatile flip-flops.[5] The nonvolatile flip-flops are all distributed alongside their volatile counterparts, storing architecture and microarchitecture states and providing high performance and long duration. The all-in-parallel structure is fast in backup time, but suffers from system instability caused by a larger temporary power peak. Another penalty is the large area of distributed nonvolatile units. Because they are distributed, the peripheral circuit needs to be duplicated for each part. Hence, a compromise is to make the all-in-parallel into part-in-parallel, or use a multistep approach to finish the backup operation.

The centralized nonvolatile block method can use various kinds of available nonvolatile blocks, such as spin-torque-transfer RAM,[6] phase-change RAM,[7] and memristors.[8] In this structure, the backup operations are carried out serially, which reduces the peak power. However, the structure requires additional interconnections to connect each faraway register and RAM to the nonvolatile block far away, and it requires a complex control module to generate the address for the backup data.

Recovery

Depending on the chosen backup policy, designers must craft a corresponding recovery policy that ensures all state is correctly and atomically restored or recomputed. Additionally, recovery policies must be robust against power failures occurring during the recovery phase.

On-demand backup. For the on-demand backup strategy, the processor cannot start recovery unless guaranteed enough energy for the next possible backup operation. This leads to longer wakeup latencies. However, for the periodic checkpoint solution, recovery can start immediately when power is regained.

Microarchitecture state. Two kinds of penalty exist for microarchitecture state recovery. The first is the time and energy penalty for instruction reexecution due to missing instruction information. For example, we can back up only the last uncommitted program counter in the RoB; all the other instructions in the RoB are discarded, even if they have already been analyzed and assigned resources to allocate. The second kind of recovery penalty is the microarchitecture state recalculation penalty (for instance, restoring the free list with the help of the map table, architectural register files, and physical register files).

The tradeoff lies between recomputation and restoring.

On-demand recovery. The states are not all recovered from nonvolatile parts as soon as power resumes. We restore parts of data only when we are going to use them. For example, the register files are not all recovered after power resumes. The control module analyzes the next program counter, finds the register files that are going to be used in this program counter, and recovers them. This solution avoids restoring some potentially useless data and might use the otherwise consumed energy for more forward progress, but it also requires additional control logic.

Normal operation

A traditional power-efficient processor focuses on maximizing the computations performed for a given amount of power without considering power supply variability. In an NVP, the goal is to maximize the computations performed considering a varying power supply, with little provision to back up the excess instantaneous power due to the absence of a battery. There are two main power consumption considerations in the design of an NVP.

Minimum turn-on power. The processor configuration determines the smallest amount of power required to run the processor. If the processor's configuration is more complex, the probability of harnessing more power than the minimum required for turning on the processor is reduced. In contrast, a less-complex processor limits the maximum power that can be used. Consequently, any power that is harvested that is greater than the peak power of the processor is wasted. Hence, an NVP should attempt to use as much of the instantaneous power as possible for maximum forward progress.

Power that is not used for computation. Backup and recovery costs, and leakage from both the NVP chip and the capacitor for temporary energy storage constitute waste, that is, energy that was not used for computation. The tradeoff between capacitor volume and leakage plays an important role in the system. A large capacitor stores more energy but also suffers from more leakage due to the larger area. Furthermore, a longer time to accumulate the voltage could also lead to more leakage. A small capacitor provides low leakage but increases the potential that the capacitor is full, limiting the ability to store any excess power not used by the processor.

Figure 4 shows the simulation results of some of the tradeoffs associated with the different power consumption aspects discussed.

- The VP always resets and experiences rollbacks to the beginning of execution under power interruption. The VP cannot finish tasks longer than the length of uninterrupted power failures.
- The NVP processors all make forward progress at different rates determined by their turn-on power requirements and the amount of instantaneous power that can be put to use for computations.
- Although the out-of-order (OoO) processor seems to be a nonintuitive choice for low-power scenarios, despite the low fraction of turn-on time, the OoO configuration makes the best use of the available instantaneous power to achieve maximum forward progress for this specific power trace.

Mapping from architecture to application and energy profiles

Most of the applications that are expected to run on an energy-harvesting platform have either hard or soft real-time requirements. Quality of service (QoS) is thus an appropriate metric to gauge the possibility of practical deployment. When these systems run on harvested ambient energy, the unreliable nature of the input source can degrade the QoS. QoS serves as a standard and target for mapping from architectures to energy profiles.

Previous works have proposed many nonvolatile architectures with different complexity levels.[3] Because many energy sources are available in an ambient environment, depending on the application requirement for QoS, we can apply different kinds of architectures to different energy sources to satisfy the application's QoS as much as possible.

We consider several traditional features to evaluate the input power profiles, including

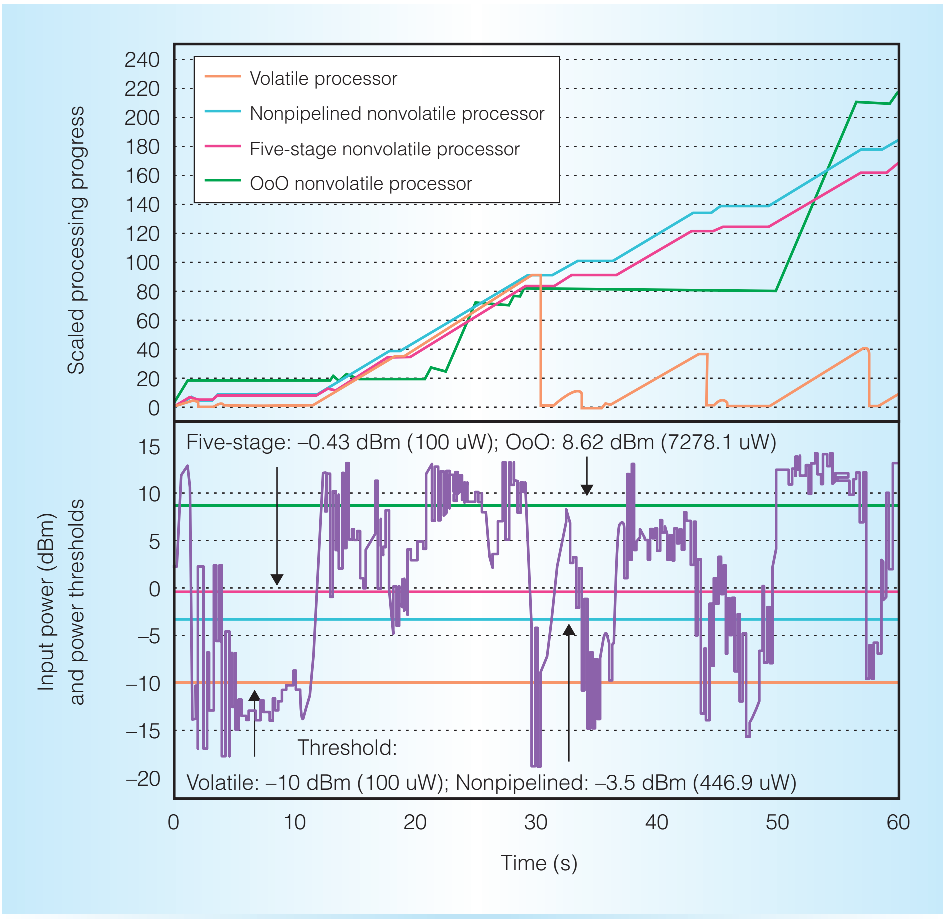

Figure 4. Forward progress comparison of volatile, nonpipelined, five-stage-pipeline, and out-of-order NVPs under a TV RF power profile. For this power profile, the OoO processor can achieve maximum forward progress because of more aggressively using the input energy.

average power, power variation, and power granularity. We define the power variation as the ratio between the maximum and minimum power income. The power granularity describes how long the power can exceed certain thresholds, and the times for power outages. We use these three parameters to evaluate the features of power sources, and we map architectures to specific power sources to satisfy the QoS requirement.

The architectures we consider are the nonpipelined (NP), *n*-stage pipeline (NSP), and OoO design points. The NP architecture requires a low start-up voltage threshold, and the backup strategies are relatively simple and straightforward, which requires little energy. The NSP architecture requires the lowest voltage threshold, but it can run faster (at higher clock frequency) than NP. The additional complexity of NSP compared to NP incurs an area overhead and thus larger leakage power. However, because of the lower V_{DD} required by NSP, the front-end circuits' efficiency can be higher than NP. From this perspective, NSP fits energy sources with low V_{DD}. NSP's backup and recovery strategies are at a relatively similar level of energy requirement to those of NP. The OoO NVP requires the highest power threshold, limiting its duty cycle relative to NP and NSP for many power profiles. When the OoO processor does run, however, it can be much faster than NP and NSP, potentially resulting in better forward progress. The backup and recovery energy of OoO are high, meaning that frequent backup and recovery should be avoided.

Figure 5a shows an example of the simulated running time of a 1-minute electrocardiogram

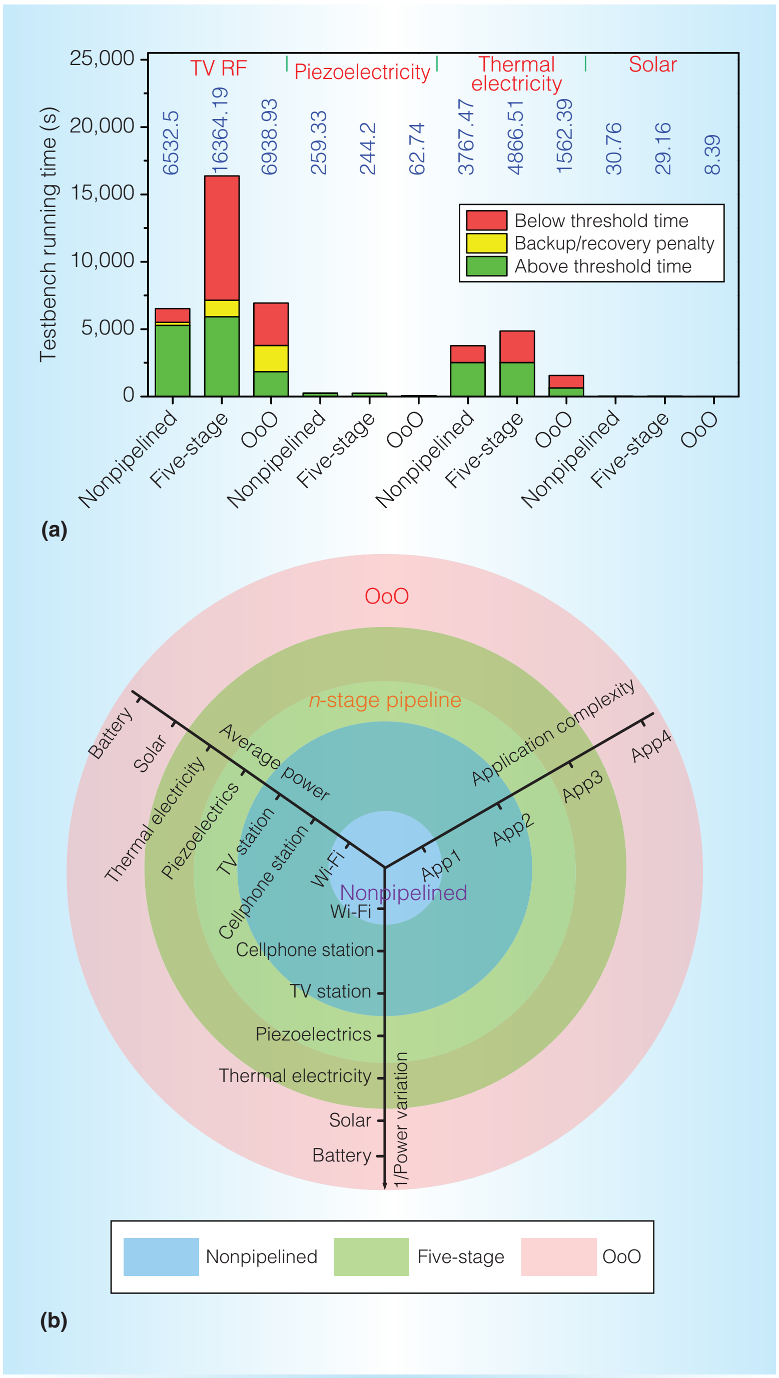

Figure 5. Testbench running time simulation results and mapping to architectures. (a) 1-minute electrocardiogram testbench running time with different architectures and different power sources. (b) Mapping architectures to application complexity and power features.

(ECG) signal processing. The *y*-axis is the running time and the *x*-axis is the architecture of the NVP. For ECG data within 1 minute, the NSP, OoO for solar, and OoO for piezoelectricity can achieve real-time data processing. However, for other energy sources with other architectures, the running time is long, meaning that real-time ECG processing is not accessible. We can also see that the TV RF power profile has large variation, and imposes large backup and recovery-time penalties. This is because the running time depends mainly on the above-threshold time, and different power profiles can have different proportions of what is above and below the threshold time, resulting in different running times even with the same architecture and testbench.

Figure 5b illustrates the architecture suggestions identified with application complexity and power profile features such as average power and inverse power variation. The figure shows that the nonpipelined architecture is suitable for low average power with large variation, and relatively simple application. The NSP architecture occupies the middle level among design points; it has a large overlap with the NP architecture because they both focus on low threshold design. The OoO architecture adapts the most complex applications and performs best with relatively stable power sources like thermal and solar energy sources, with low power variation.

According to different application and power profile scenarios, we provide the rough architecture selection guidelines in Table 1.

In this article, we have shown that, for energy-harvesting systems, traditional low-power design optimizations do not always yield the optimal forward progress for a system that must contend with high dynamic power variability and for which backup and restore costs can consume significant portions of the total energy expended. Put simply, in a deployment without significant energy storage, power austerity is prodigal, and greed is good if forward progress is the metric of merit. NVPs enable more aggressive designs in power-constrained environments and represent a significant step forward on this front. However, our investigations have also shown that current energy-harvesting systems are bad at being greedy. Key aspects of future work in the area will be to continuously maximize the fraction of incoming energy that is

Table 1. NVP architecture selection suggestions for different power sources and scenarios.

Power sources and scenarios	Power features	Architecture
Solar power inside a room	Low power income level with infrequent power boost	Nonpipelined (NP)
Solar power while riding a bike in the forest	Low power income level with frequent power boost	Out of order (OoO)
Solar power on cloudy days or in a shadow	Intermediate power	NP or *n*-stage pipeline (NSP)
TV RF fixed antenna	Extremely low power with little variation	NSP
TV RF antenna worn by human	Extremely low power with large variation	NP
Office Wi-Fi RF with little movement	Extremely low power with little variation	NSP
Office Wi-Fi RF with frequent movement	Extremely low power with large variation	NP
Home Wi-Fi RF	Extremely low power with very large variation	NP
Thermal band fixed on arms, little movement	Relatively stable intermediate power	OoO
Thermal band fixed on legs, running	Periodical wave with almost no power failure, but with some variation	NSP or OoO
Piezo device fixed on a bike	Near stable power but with frequent short-time power outage	OoO
Piezo device under the shoes or fixed on knees and the leg	Periodical power boost with large power outage	OoO

turned into useful computation and to develop approaches that employ dynamic microarchitectural and architectural adaptations to successfully capitalize on periods with high incoming power. MICRO

Acknowledgments

This work was supported in part by the Center for Low Energy Systems Technology (LEAST), sponsored by MARCO and DARPA, Shannon Lab Huawei Technologies, High-Tech Research and Development (863) Program under contract 2013AA01320, the Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions under contract YETP0102, and by the NSF awards 1160483 (ASSIST), 1205618, 1213052, 1461698, and 1500848.

References

1. S. Kim et al., "Ambient RF Energy-Harvesting Technologies for Self-Sustainable Standalone Wireless Sensor Platforms," *Proc. IEEE*, vol. 102, no. 11, 2014, pp. 1649–1666.
2. X. Li et al., "RF-Powered Systems Using Steep-Slope Devices," *Proc. IEEE 12th Int'l New Circuits and Systems Conf.*, 2014, pp. 73–76.
3. K. Ma et al., "Architecture Exploration for Ambient Energy Harvesting Nonvolatile Processors," *Proc. Int'l Symp. High-Performance Computer Architecture*, 2015, pp. 526–537.
4. T. Miwa et al., "NV-SRAM: A Nonvolatile SRAM with Backup Ferroelectric Capacitors," *IEEE J. Solid-State Circuits*, vol. 36, no. 3, 2001, pp. 522–527.
5. Y. Wang et al., "A 3us Wake-up Time Nonvolatile Processor Based on Ferroelectric Flip-Flops," *Proc. 4th European Solid-State Circuits Conf.*, 2012, pp. 149–152.
6. H. Noguchi et al., "7.5 A 3.3 ns-Access-Time 71.2μW/MHz 1Mb Embedded STT-MRAM Using Physically Eliminated Read-Disturb Scheme and Normally-Off Memory Architecture," *Proc. IEEE Int'l Solid-State Circuits Conf.*, 2015, pp. 1–3.
7. G. De Sandre et al., "A 90nm 4Mb Embedded Phase-Change Memory with 1.2 V 12ns Read Access Time and 1MB/s Write Throughput," *Proc. IEEE Int'l Solid-State Circuits Conf.*, 2010, pp. 268–269.
8. M.-F. Chang et al., "19.4 Embedded 1Mb ReRAM in 28nm CMOS with 0.27-to-1V Read Using Swing-Sample-and-Couple Sense Amplifier and Self-Boost-Write-Termination Scheme," *Proc. IEEE Int'l Solid-State Circuits Conf.*, 2014, pp. 332–333.

Kaisheng Ma is a PhD student in the Department of Computer Science and

Engineering at Pennsylvania State University. His research interests include energy-harvesting architectures, machine learning, and neuromorphic computing. Ma has an ME in microelectronics from Peking University. Contact him at kxm505@cse.psu.edu.

Xueqing Li is a postdoctoral researcher in the Department of Computer Science and Engineering at Pennsylvania State University. His research interests include wireless transceivers, high-performance data converters, self-powered nonvolatile systems, and circuits and systems using emerging devices. Li has a PhD in electronics engineering from Tsinghua University. He is a member of IEEE. Contact him at lixueq@cse.psu.edu.

Shuangchen Li is a PhD student in the Department of Electrical and Computer Engineering at the University of California, Santa Barbara. His research focuses on computer architecture, especially emerging nonvolatile memory. Li has an MS in electronic engineering from Tsinghua University. Contact him at shuangchenli@ece.ucsb.edu.

Yongpan Liu is an associate professor in the Department of Electronic Engineering at Tsinghua University. His research interests include nonvolatile computation, low-power VLSI design, emerging circuits and systems, and design automation. Liu has a PhD in electronics engineering from Tsinghua University. He is a member of IEEE; the ACM; and the Institute of Electronics, Information, and Communication Engineers. Contact him at ypliu@tsinghua.edu.cn.

John (Jack) Sampson is an assistant professor in the Department of Computer Science and Engineering at Pennsylvania State University. His research interests include energy-efficient computing, architectural adaptations to exploit emerging technologies, and mitigating the impact of dark silicon. Sampson has a PhD in computer engineering from the University of California, San Diego. He is a member of IEEE. Contact him at sampson@cse.psu.edu.

Yuan Xie is a professor in the Department of Electrical and Computer Engineering at the University of California, Santa Barbara. His research interests include computer architecture, EDA, and VLSI design. Xie has a PhD in electrical engineering from Princeton University. He is a fellow of IEEE. Contact him at yuanxie@ece.ucsb.edu.

Vijaykrishnan Narayanan is a Distinguished Professor of Computer Science and Engineering and Electrical Engineering at Pennsylvania State University. His research focuses on energy-efficient computing. Narayanan has a PhD in computer science from the University of South Florida. He is a fellow of IEEE and the ACM. Contact him at vijay@cse.psu.edu.