Emulating Intermittent Non-Volatile Computing Systems Using Off-the-shelf FPGAs

Kasım Sinan Yıldırım
University of Trento
kasimsinan.yildirim@unitn.it

Davide Brunelli
University of Trento
davide.brunelli@unitn.it

ABSTRACT

Today's intermittent devices operate by relying only on harvested energy accumulated in their tiny energy reservoirs, typically a capacitor. An intermittent device dies when there is no energy in its capacitor and boots again when the harvested energy is sufficient to power its hardware components. This operation leads to frequent loss of computation state and intermediate results—preventing forward progress of computation. In order to mitigate the effects of power failures and ensure forward progress of computation, software and hardware-based solutions are proposed in the literature. In this paper, we focus on hardware-based solutions, mainly non-volatile processors (NVPs) that integrate built-in fast non-volatile memory in their micro-architecture. We emphasize that even though FPGAs are common platforms for fast prototyping and behavioral verification of processor architectures, they do not target NVPs. The main reason is that the logic elements of the current off-the-shelf FPGA platforms comprise only volatile flip-flops and they loose not only their configuration but also the contents of these logic elements whenever power is removed from the system. To remedy this problem, we propose a new FPGA architecture to emulate and verify the behavior of any intermittent computing system that exploits fast non-volatile memories.

KEYWORDS

Intermittent computing, Non-volatile Processors, FPGAs

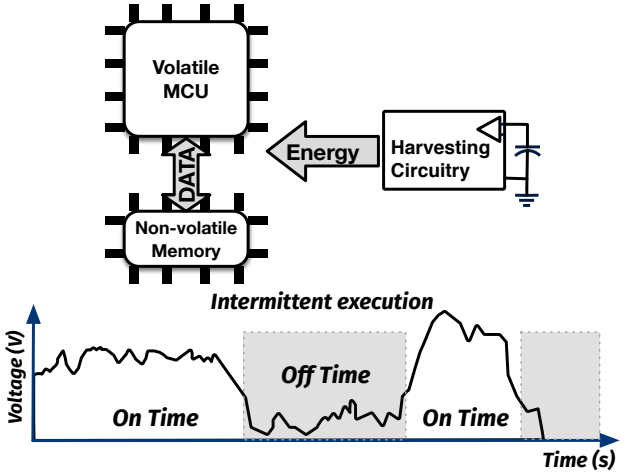

Figure 1: The operation of an energy harvesting battery-less computing system. The harvested energy is stored in a capacitor. The device dies when the voltage level is below a threshold and boots again when the energy is above the threshold—leading to intermittent computation. The volatile microcontroller is equipped with an external non-volatile memory to store intermediate results and state of the computation to recover from power failures.

1 INTRODUCTION

The recent advancements in microelectronics led to the development of batteryless sensors that can operate relying on ambient energy only [16]. This sensing technology opens up new application spaces where small devices that should have eternal lifetimes, completely autonomous operation, and massive deployments in inaccessible locations [6]. Batteryless sensors are equipped with energy harvesting circuits that use several sources such as solar, thermal, and radio waves to accumulate the environmental energy into a small energy storage; typically a tiny capacitor. The sensor operates by relying only on the energy accumulated in this capacitor—the sensor dies when there is no energy and boots again when the capacitor energy is sufficient to power its hardware components. When the sensor dies, the volatile state of its processor; e.g., the contents of stack, program counter, registers, is lost. This leads to the loss of all computational state, and intermediate results [24]—see Figure 1.

Contrary to their volatile counterparts, non-volatile processors (NVPs) integrate built-in non-volatile memory in their architecture—the processor's volatile state is automatically backed up into the internal non-volatile registers when a power failure occurs and restored when the power becomes available [12]. All these backup operations are completely transparent to the programmer. This transparency can be supported via several architectural design decisions that have different pros and cons. A major issue is to decide which state elements of the NVP will be implemented as non-volatile. As an example, all registers can be non-volatile—slower and more energy consuming—or they can be implemented as volatile but counterpart non-volatile registers can be maintained to backup the state at specific points in time. Another major issue is to decide the backup frequency of the volatile state components. For instance, the backup can be performed at every processor cycle or it can be performed on demand backup [12, 13], to decrease backup frequency to save energy.

However, the lack of design tools makes the exploration of architectural design space as well as fast prototyping of NVPs difficult. The main reason is that the logic elements of the current off-the-shelf FPGA (field programmable gate array) platforms comprise only volatile registers. Therefore, FPGAs loose not only their configuration but also the contents of these logic elements whenever power is removed from the system. Moreover, as of now, hardware description languages such as VHDL or Verilog do not provide any keyword to describe non-volatile architectural elements and logic.

To the best of our knowledge, the state-of-the-art does not propose a solution to emulate and validate non-volatile architectures using off-the-shelf FPGAs.

In this paper, we provide a new FPGA architecture for emulating any intermittent computing system that exploits fast non-volatile memories in their micro-architecture; e.g. NVPs. The proposed architecture is composed of auxiliary blocks that simulate (i) the behavior of irregular power supply typical to energy-harvesting intermittent systems; (ii) the persistence of the non-volatile micro-architectural elements as well as the longer delay of read/write operations (as compared to those of volatile memory). We also introduce another block that approximates the power consumption of the emulated technology. The detailed description of our procedure is presented in the next sections.

2 BACKGROUND

A new class of embedded devices that can sense, compute and communicate without batteries emerged—e.g. RF-powered battery-less sensors [4] solely rely on the harvested energy of ambient radio frequency waves in the air. These sensors comprise ultra low-power micro-controllers; e.g. MSP430FR5969 [20], whose main architectural components; e.g. registers and main memory, are volatile. These micro-controllers are also equipped with a non-volatile secondary memory components; e.g. Ferroelectric RAM (FRAM) [21], to store information that will persist upon power failures. To mitigate the effects of unpredictable power failures, several *software-aided* solutions have been proposed [2, 3, 7, 8, 11, 14, 15, 22]. Generally speaking, these solutions backup the volatile state of the processor into non-volatile memory with software techniques so that the computation can be recovered from where it left upon a power failure. Moreover, they ensure the memory consistency so that the backed-up state in the non-volatile memory will not be different than the volatile state, or vice versa. However, software-aided recovery solutions require transmitting data from built-in volatile components of the processor; e.g. registers, to non-volatile memory—suffers from low speed; e.g. 200 micro-secs [17], and large energy penalty that grows with the size of volatile elements [17, 19]. Also, these solutions require programmers to structure their software by considering programming models designed for intermittent systems; e.g. task-based programming [3, 24]

On the other side, NVPs bring non-volatile memory into the micro-architecture of the processor so that backup and recovery from a power failure are transparent to the programmer as well as introduce less overhead as compared to software-aided solutions; e.g. only on the order of a few microsconds [9, 23]. Since backup and retention operations are fast as compared to the software-aided solutions, NVPs reduce leakage power by allowing system to completely shut down when the device idle [1].

Due to the higher power required for non-volatile memory read/write operations, NVPs might also consume more power as compared to volatile processors [13]—therefore they require micro-architecture-level optimizations. To decrease the energy consumption of NVPs, recent works proposed i) using more efficient memory technologies; e.g. ReRAM [10] and hybrid CMOS/ferroelectric non-volatile flipflop [18]; ii) embedding non-volatility into the computing logic; e.g. transistor level, using NCFET [9] so that logic gates could also store their states intrinsically in a non-volatile fashion; iii) using new backup strategies; e.g. backup at every processor cycle or on demand backup [12, 13], to decrease backup frequency in order to save energy. These efforts provide implementation technology-level energy optimizations.

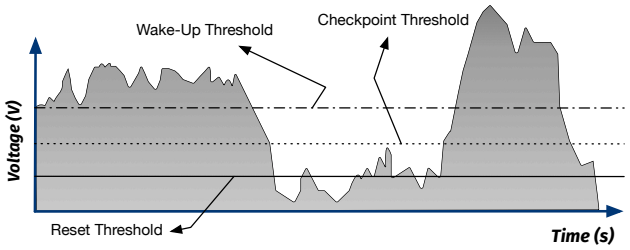

Figure 2: Identified thresholds during intermittent operation. The computing system dies when the voltage level in the capacitor is below reset threshold. The system starts operation when the voltage level is above wake-up threshold. The software running on the computing system can also be notified; via the checkpoint threshold so that some volatile state can be manually copied to the non-volatile memory; as in [2].

FPGAs are used in many applications due to the increased cost and time associated with the custom ASIC (application specific integrated circuit) design. They are extremely useful for fast prototyping custom processor architectures and their behavioral verification. There are several implementations of popular volatile processor architectures, such as RISC-V, using popular hardware description langugaes (HDLs). These implementations can be used to implement and execute these processors on FPGAs. However, it is not defined how to use FPGAs to prototype NVP architectures since their logic elements are volatile. To the best of our knowledge, none of the existing HDLs support description of non-volatile logic elements.

3 PROPOSED EMULATION ARCHITECTURE

We propose an emulation strategy that mimics non-volatile memory elements and power failures. Our procedure is composed of three main auxiliary blocks: (i) intermittency emulation that emulates irregular power supply typical to energy-harvesting systems; (ii) non-volatile register emulation that emulates the persistence of the non-volatile registers in the micro-architecture as well as the delays of the read/write operations; and (iii) power consumption that approximates the power consumption of the emulated technology. Figure 3 presents an overview of the proposed methodology.

3.1 Intermittency Emulation

In order to emulate a power failure, a `RESET` line is triggered by the intermittency emulation block. This block can generate random triggers as well as can follow an energy trace of a realistic energy harvesting scenario; as presented in [5]. This block can be composed of a memory that can hold a pre-collected voltage trace. The voltage trace can also be given by external signals to this block. At each clock pulse, a new voltage value can be picked and compared

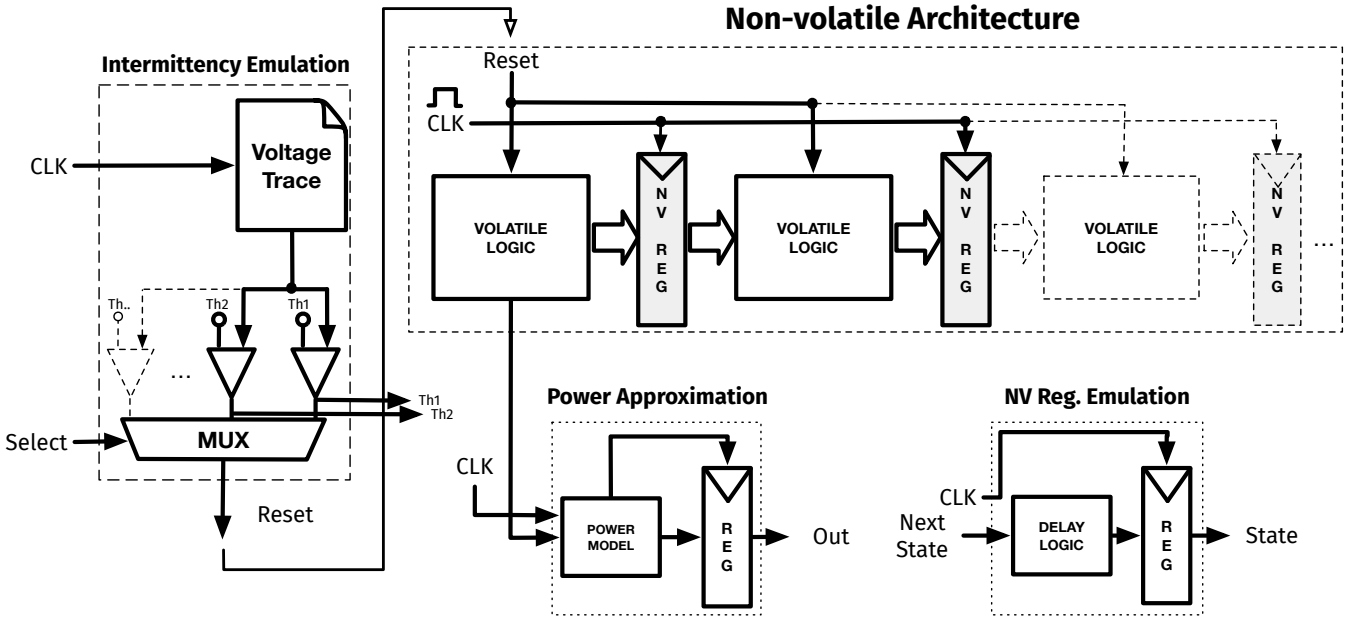

Figure 3: Non-volatile Register Emulation. Our architecture is composed of three auxiliary blocks. Intermittency Emulation block emulates intermittent power supply. Non-volatile Register Emulation block emulates the persistence of the non-volatile registers as well as the delays of the read/write operations. Power Approximation block approximates the power consumption of the emulated technology.

against several pre-determined threshold values using comparators. If the current voltage value in the trace is smaller than the corresponding threshold value, the corresponding comparator outputs a high signal. A multiplexer placed in front of the comparators is used to select the output of the desired threshold comparison operation.

This block can also output signals that indicate if a threshold condition is satisfied in order to be used by some other logic in the architecture, as presented in Figure 2. In particular, a threshold value can be set to trigger a software routine just before a power failure, as in [2]. Or a dedicated hardware block can be triggered via one of these signals to perform back-up operation so that volatile state elements of the micro-architecture is copied to their non-volatile counterparts.

3.2 Non-volatile Register Emulation.

Since the registers in FPGA logic elements are volatile, one cannot directly implement non-volatile registers. To this end, we propose to implement non-volatile registers using volatile registers in FPGAs. Our solution is simple: we connect the RESET line to all volatile registers and combinational logic other than non-volatile registers in the architecture. When the reset signal is triggered, the contents of the volatile registers will be cleared. Since the non-volatile register blocks (implemented as volatile registers) are not connected to the reset line, their contents will remain in course of resetting other logic elements. Another issue is that non-volatile read/write operations are slower than volatile ones. To emulate this behaviour, i.e. for the delays introduced by NVM circuits, we add parametric delays as well. To this end, we place a combinational logic following the inputs of the volatile register to simulate the delay of non-volatile register operations.

3.3 Power Consumption Approximation

To provide a reliable emulation and assessment of the intermittent micro-architecture, we accelerate the power estimation by mapping the power model related circuit onto the prototyping platform. We defined a power model for each volatile logic block, as depicted in Figure 3. The power model, fed by activity counters, is configured to measure the power consumption of the programmed logic into each block. It takes into account the technology of the emulated chip and the type of activity requested by each volatile logic block. The counters provide in-depth and distributed information about the power performance, while the parametric delays introduced in the read/write NVMs realize the proper latency of the used memory technology (i.e., ReRAM, FRAM, etc.).

4 CONCLUSIONS AND FUTURE WORK

In this paper, we described an FPGA architecture for emulating any intermittent computing system that exploits fast non-volatile memories, to store temporary status in case of supply failures. The architecture is equipped with some auxiliary parts that simulate the behavior when the system is powered by an unregular power supply, typical of any batteryless transient computing system powered by energy harvesters, and take into account the delay of the NVMs and the measure the power consumption of the emulated technology. The proposed architecture is appropriate for verifying the behavior

of such new types of systems over long time scales, typical of duty-cycling energy-neutral Internet of Things (IoT) applications.

Future studies can target the real implementation of the proposed architecture on popular FPGA platforms in the market. In particular, a non-volatile processor architecture can be implemented, and its behavioral verification can be done using the proposed FPGA architecture. Moreover, the accuracy of the power approximation block can be observed by comparing the actual ASIC implementation of the processor with respect to its implementation in the proposed emulation architecture.

REFERENCES

[1] Tosiron Adegbija, Anita Rogacs, Chandrakant Patel, and Ann Gordon-Ross. 2018. Microprocessor optimizations for the Internet of Things: a survey. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 1 (2018), 7–20.

[2] Domenico Balsamo, Alex S. Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M. Al-Hashimi, Geoff V. Merrett, and Luca Benini. 2016. Hibernus++: a Self-calibrating and Adaptive System for Transiently-powered Embedded Devices. 35, 12 (2016), 1968–1980.

[3] Alexei Colin and Brandon Lucia. 2016. Chain: Tasks and Channels for Reliable Intermittent Programs. In *Proc. OOPSLA*. ACM, Amsterdam, Netherlands, 514–530.

[4] Shyamnath Gollakota, Matthew S Reynolds, Joshua R Smith, and David J Wetherall. 2013. The emergence of RF-powered computing. *Computer* 47, 1 (2013), 32–39.

[5] Josiah Hester, Timothy Scott, and Jacob Sorber. 2014. Ekho: realistic and repeatable experimentation for tiny energy-harvesting sensors. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. 330–331.

[6] Josiah Hester and Jacob Sorber. 2017. The Future of Sensing is Batteryless, Intermittent, and Awesome. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 21.

[7] Josiah Hester, Kevin Storer, and Jacob Sorber. 2017. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 17.

[8] Hrishikesh Jayakumar, Arnab Raha, Woo Suk Lee, and Vijay Raghunathan. 2015. Quickrecall: A HW/SW Approach for Computing Across Power Cycles in Transiently Powered Computers. *ACM J. Emerg. Technol. Comput. Syst.* 12, 1 (July 2015), 8:1–8:19.

[9] Xueqing Li, Sumitha George, Kaisheng Ma, Wei-Yu Tsai, Ahmedullah Aziz, John Sampson, Sumeet Kumar Gupta, Meng-Fan Chang, Yongpan Liu, Suman Datta, et al. 2017. Advancing nonvolatile computing with nonvolatile NCFET latches and flip-flops. *IEEE Transactions on Circuits and Systems I: Regular Papers* 64, 11 (2017), 2907–2919.

[10] Yongpan Liu, Zhibo Wang, Albert Lee, Fang Su, Chieh-Pu Lo, Zhe Yuan, Chien-Chen Lin, Qi Wei, Yu Wang, Ya-Chin King, et al. 2016. 4.7 A 65nm ReRAM-enabled nonvolatile processor with 6× reduction in restore time and 4× higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic. In *Solid-State Circuits Conference (ISSCC), 2016 IEEE International*. IEEE, 84–86.

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

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

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

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

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

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

[17] Xin Shi, Tongda Wu, Keni Qiu, Huazhong Yang, and Yongpan Liu. 2018. Time Stamp Based Scheduling for Energy Harvesting Systems with Hybrid Nonvolatile Hardware Support. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 339–344.

[18] Fang Su, Yongpan Liu, Yiqun Wang, and Huazhong Yang. 2017. A Ferroelectric Nonvolatile Processor with 46mus System-Level Wake-up Time and 14mus Sleep Time for Energy Harvesting Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers* 64, 3 (2017), 596–607.

[19] Fang Su, Kaisheng Ma, Xueqing Li, Tongda Wu, Yongpan Liu, and Vijaykrishnan Narayanan. 2017. Nonvolatile processors: Why is it trending?. In *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 966–971.

[20] Texas Instruments. 2018. MSP430FR5969 LaunchPad Development Kit. http://www.ti.com/tool/MSP-EXP430FR5969

[21] Texas Instruments, Inc. 2014. FRAM FAQs. http://www.ti.com/lit/ml/slat151/slat151.pdf. Last accessed: 2018.

[22] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent Computation Without Hardware Support or Programmer Intervention. In *Proc. OSDI*. ACM, Savannah, GA, USA, 17–32.

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

[24] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemysław Pawełczak, and Josiah David Hester. [n.d.]. InK: Reactive Kernel for Tiny Batteryless Sensors. In *The 16th ACM Conference on Embedded Networked Sensor Systems (SenSys 2018)*.