BOBBER: A Prototyping Platform for Batteryless Intermittent Accelerators

Vishak Narayanan
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa, United States
vishakn@iastate.edu

Rohit Sahu
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa, United States
rsahu@iastate.edu

Jidong Sun
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa, United States
jidongs@iastate.edu

Henry Duwe
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa, United States
duwe@iastate.edu

ABSTRACT

Batteryless systems offer promising platforms to support pervasive, near-sensor intelligence in a sustainable manner. These systems solely rely on ambient energy sources that often provide limited power. One common approach to designing batteryless systems is using intermittent execution—a node banks energy into a capacitive store until a threshold voltage is met and the digital components turn on and consume the banked energy until the energy is depleted and they die. The limited amount of available energy demands the development of application- and domain-specific accelerators to achieve energy efficiency and timeliness. Given the extremely close relationship between volatile state and intermittent behavior, performing actual system prototyping has been critical for demonstrating feasibility of intermittent systems. However, no prototyping platform exists for intermittent accelerators. This paper introduces BOBBER, the first implementation of an intermittent FPGA-based accelerator prototyping platform. We demonstrate BOBBER in the optimization and evaluation of a neural network accelerator powered solely by RF energy harvesting.

CCS CONCEPTS

• **Hardware → Hardware accelerators**; **Sensor devices and platforms**; **Reconfigurable logic applications**; • **Computer systems organization → Firmware**; **Embedded hardware**; **Embedded software**.

KEYWORDS

energy-harvesting; batteryless; intermittent computing; accelerators; system prototyping platform; battery-free sensor

ACM Reference Format:
Vishak Narayanan, Rohit Sahu, Jidong Sun, and Henry Duwe. 2023. BOBBER: A Prototyping Platform for Batteryless Intermittent Accelerators . In *Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '23), February 12–14, 2023, Monterey, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3543622.3573046

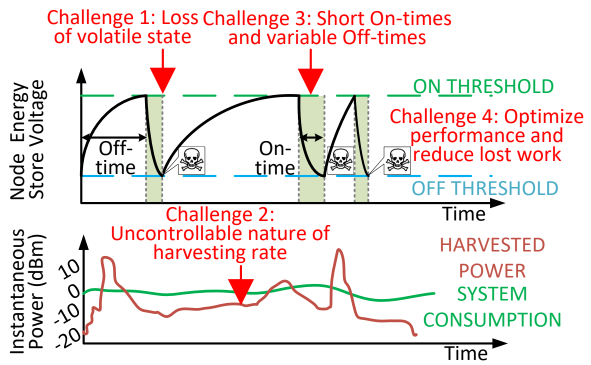

Challenge 1: Loss of volatile state
Challenge 3: Short On-times and variable Off-times
ON THRESHOLD
Challenge 4: Optimize performance and reduce lost work
Node Energy Store Voltage
Off-time
On-time
OFF THRESHOLD
Time
Challenge 2: Uncontrollable nature of harvesting rate
Instantaneous Power (dBm)
10
0
-10
-20
HARVESTED POWER
SYSTEM CONSUMPTION
Time

Figure 1: Intermittent execution and its challenges.

1 INTRODUCTION

As the Internet of Things (IoT) becomes a reality, tens of billions, trillions of smart, sensor-enabled devices—capable of sensing, computing, and communicating—will be deployed in just about every conceivable location in the coming decades. Currently, these systems are powered by conventional energy sources like batteries or wired power. But for such large numbers of devices, continuously powering them with wires is infeasible due to cost and location. On the other hand, batteries require replacement/maintenance due to limited lifetimes and most types are not environmentally friendly [33]. Batteryless systems powered solely from ambient energy sources such as Radio Frequency (RF), thermal, vibration, kinetic, or solar [1, 12, 38] have emerged as a promising paradigm.

Even though batteryless systems show the potential of increased lifetime, near zero maintenance, and environmental friendly materials, designing them poses insidious challenges. Consider the illustration of an RF energy trace shown in Figure 1 where the harvested power is almost always beneath a mW (0 dBm). Most commercial devices (e.g., MSP430FR5994 [18] with 7mW of operational power) would be unusable with this magnitude of harvested power. In response, the intermittent execution paradigm (shown in upper Figure 1) has emerged where the system banks harvested energy into a capacitor until enough energy is collected to turn on and perform sensing, computation, and communication tasks. Once these tasks are completed or the energy has been exhausted, the system dies allowing the capacitor to recharge [7, 9, 10, 13, 19, 26, 30, 39]. However, when there is a power failure, nodes lose all state not stored in a non-volatile memory. This is exacerbated by the sporadic and uncontrollable nature of energy harvesting which makes on-times short and off-times variable.

The MCU-based platforms such as WISP [41] and MOO [46] demonstrated feasible batteryless systems and have been used extensively in intermittent systems research. While many of the past works have used such an MSP430FR5994 MCU system, this system has significant performance limitations (e.g., a 16MHz, 3-stage pipeline with multi-cycle multiply and a conventional dense linear algebra accelerator). While these devices can support intermittent inference [13, 23], common neural network hardware accelerator optimizations such as low-precision, energy-adaptivity, and sparsity cannot be explored in these prototypes. This paper presents a prototyping platform for batteryless intermittent accelerators and makes the following contributions:

- BOBBER[1], an MCU + FPGA-hosted accelerator prototyping platform that can be used to explore heterogeneous acceleration for batteryless intermittent sensor nodes under real energy-harvesting constraints.
- Demonstration of BOBBER's capabilities using an intermittency-aware HW-SW accelerator framework based on TFlite-micro [8]. Resulting accelerators can achieve both correct execution and forward progress under varying energy harvesting conditions despite volatile state loss on the accelerator.
- Experimental evaluation of BOBBER under RF energy harvesting. The evaluations include both platform overhead characterization and end-to-end functional correctness of a CNN under energy-harvesting.

2 BACKGROUND AND MOTIVATION

Commercial-off-the-shelf (COTS) devices are designed to be continuously powered or controllably duty-cycled into low-power states while powered by a battery. Batteryless systems may often harvest insufficient power to even maintain low-power states much less operate continuously. For example, with an Powercast TX91501b RF transmitter [35] (PTX) and corresponding Powercast P2110B Powerharvester [34] (PRX) power at the receiver may vary between 6.56 μW and 5.7 mW based on placement within a research lab, while an MSP430FR5994 [18] consumes 7 mW of average active power and 15 μW when in sleep (retaining all the volatile state). Larger variations may occur based on system deployment, changes in the environment around the system, or system motion. Such intermittent batteryless systems must be carefully designed with system execution and power usage tightly coupled to the small and variable energy harvesting rate.

Typically, batteryless intermittent proposals contain both volatile state (e.g., in SRAM and flip-flops) and non-volatile state (e.g., FRAM memory of the MSP430 that most prototype systems use). Volatile memory is generally used for frequently used data and instructions while non-volatile memory is used for large capacity data and persisting execution progress and code across powerloss. Any volatile state not checkpointed must be recomputed in the next on-time. However, if a program reads and updates a non-volatile state (i.e., has a Write-After-Read (WAR) anti-dependency on non-volatile state), re-execution may produce results not attainable by any correct program execution—the recomputation produces an inconsistent state [24, 29]. Additionally, if execution is continually halted before another checkpoint is created, the application may never complete—i.e., it ceases to make forward progress.

Software solutions for guaranteeing correctness while providing forward progress through an application range from guaranteed checkpointing based on ADC triggers [4] to periodic checkpoints [2, 3, 19, 27] to task-based execution [7, 24, 26, 30, 39]. These run on existing platforms such as WISP[41], MOO[46], Flicker [15], or other custom platforms built around an FRAM-based MSP430. Although the MSP430 provides the low energy accelerator (LEA) [17] which has been leveraged by [13, 20] for DNN inference, it lacks reconfigurability to specifically target an application or domain (e.g., it efficiently handles neither sparsity of convolution layers nor precision lower than 8-bits).

In order to support the design, test, and debug of these prototype systems, intermittency-aware tools were needed. CleanCut [6] guarantees termination under a given set of system assumptions. The energy-interference-free debugger (EDB) [5] enables debugging on MSP430-based prototypes without perturbing the tight coupling between energy harvesting and consumption. EKHO [14] records and replays harvested energy traces for prototyped systems. NORM [40] emulates an intermittent system on an FPGA using an energy-harvesting model while replaying previously recorded energy traces. However, NORM can only model the energy harvesting, power delivery, and power consumption of the design.

Other works have proposed processor microarchitectures for intermittent computing. Clank[16] uses idempotence—a property that a sequence of instructions can be arbitrarily restarted by maintaining the inputs to the sequence [28]—to guarantee correct re-execution. Other approaches include the use of processor heterogeneity in PHASE [11] and wholly non-volatile processors in incidental computing [25]. However, all of these are simulation-based.

Recent works have proposed accelerators specially designed for intermittent systems. Reconfigurable ReRAM crossbar-based accelerators (RCAs) have emerged as a promising technology to efficiently perform the multiply and accumulate (MAC) operations dominant in CNNs [36, 37]. However, RCAs are not commercially

Table 1: Comparison of most closely related works.

Work	Reconfig	Accelerator	EH Prototype
SoC [21]	x	FFT (fixed)	✓
SONIC [13]	x	LEA (fixed)	✓
NEURO-ZERO [23]	x	2nd MSP430	✓
HAWAII [20]	x	LEA (fixed)	✓
PHASE [11]	✓	DNN	x
ResiRCA [36]	✓	ReRAM (RCA)	x
MaxTracker [37]	x	ReRAM (RCA)	x
NORM [40]	✓	x	x
CP-FPGA [45]	✓	x	x
BOBBER (this work)	✓	FPGA (e.g., DNN)	✓

[1]Like a bobber in the water, being pushed and pulled by external, uncontrollable forces, BOBBER intermittently surfaces to compute during on-times and disappears beneath the waves to gather energy to surface again. All PCB hardware, FPGA designs, and software are available at https://zenodo.org/record/7439488

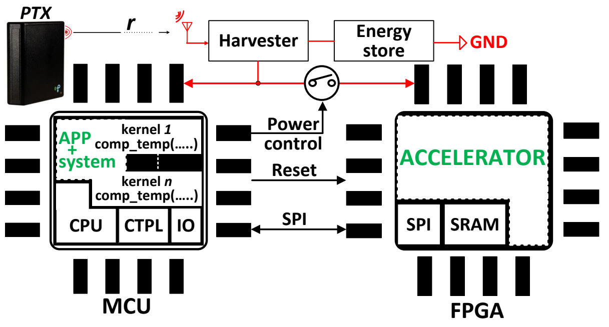


Figure 2: BOBBER platform overview.

available and must be evaluated in simulation. FPGA-based accelerators are also appealing because of their reconfigurability. CP-FPGA [45] and DFT-FPGA [47] represent non-volatile FPGA fabric architectures coupled with checkpointing schemes for volatile computation data. However, the resulting systems are only evaluated in simulation. Currently, while a few commercially available FPGAs have non-volatile logic elements such that their programming is not lost with power [31], they only have volatile state within the circuits (i.e., flip-flops) and configurable memory (i.e., SRAM). Therefore, any accelerator system prototype designed with them will have to address the consistency problem caused by WAR dependencies to external non-volatile memory when recomputing lost volatile state on the FPGA.

In summary, to the best of our knowledge, there have been no platforms that support reconfigurable accelerator prototyping under real-world energy harvesting (see Table 1).

3 BOBBER

BOBBER offers a full-stack prototyping platform to quickly develop functionally correct custom intermittent accelerator applications. As shown in Figure 2, BOBBER is powered solely by an RF energy-harvester and computationally comprises an MCU and a discrete FPGA. The MCU runs application and system software on top of an intermittency-aware firmware. The application software can be accelerated by an arbitrary number of computational kernels that are implemented by an accelerator on the FPGA. Kernel calls are simple to implement and templated such that correctness can be guaranteed even when power failures occur during kernel execution and the accelerator's local volatile data is lost.

3.1 BOBBER Hardware Platform

The core BOBBER hardware platform is implemented with COTS components on a 4-layer PCB. Jumper and header support is included to isolate and test individual components such as the energy harvester and extend functionality to include sensors and a radio.

The RF energy harvesting frontend is a Powercast P2110B Powerharvester [34] module with built-in power regulator and supervisor to provide the digital components with a regulated 3.3V during on-times and 0V during off-times. The RF harvester charges a series of dynamically-programmable capacitor banks similar to Capybara [7] in order to support a range of applications, accelerators, and their associated energy costs. BOBBER has four banks of low-leakage ceramic super capacitors, with the default bank having two multi-layer ceramic chip capacitors (MLCC). For an intermittent accelerator, when a larger computation is needed the platform can save-up energy to perform the computation using the accelerator while making simple system control decisions with its MCU. The MCU can also monitor the capacitive storage voltage via an on-board ADC for power control decisions.

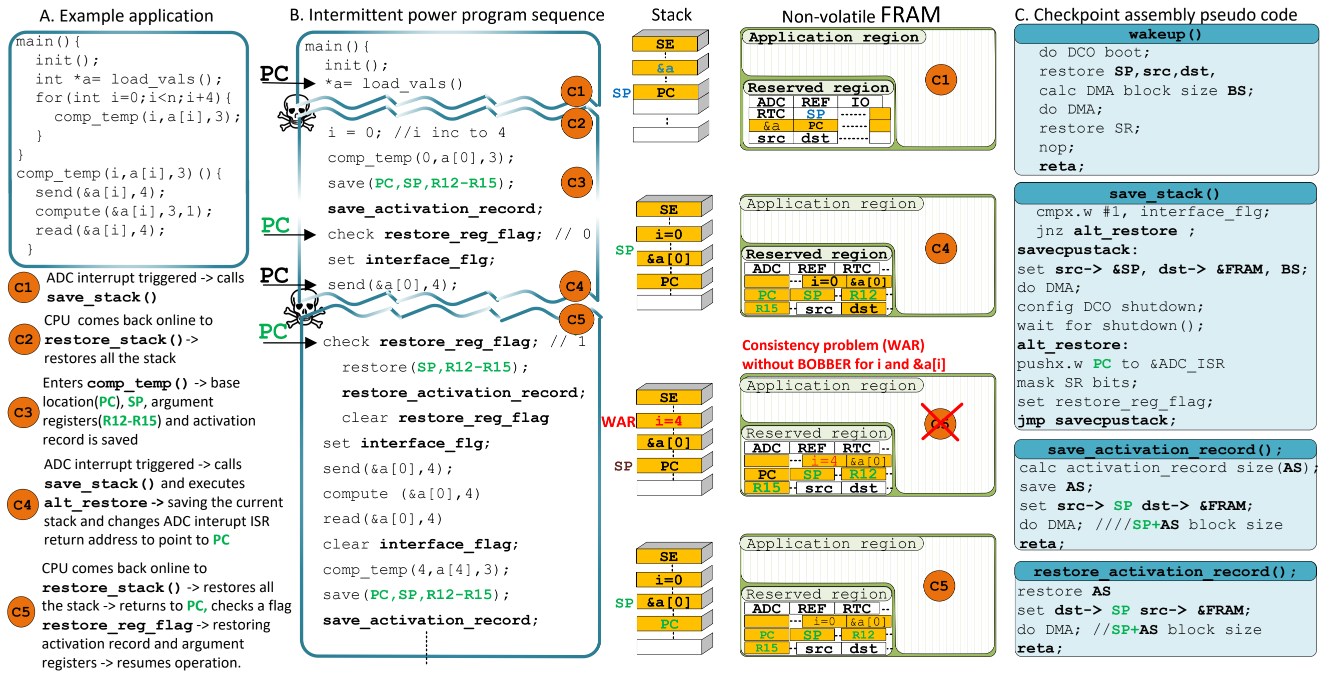
As quantified in Section 2, the rate of RF energy harvesting is low and the energy store is limited. Therefore, we selected MCU and FPGA components for low power and low-energy boot. BOBBER uses an Igloo Nano[31] FPGA which has 3K non-volatile LUT elements, 36 Kbits of SRAM (volatile state) and 1 Kbit of non-runtime-writable ROM. While the Igloo Nano does not need to be reprogrammed after an off-time, any intermediate data stored in its SRAM is lost. BOBBER uses a MSP430 [44] microcontroller, which is commonly used in energy harvesting applications, due to it's byte-updatable FRAM that allows efficient non-volatile updates and checkpoints. It also has wide range of operating modes (7 mW at the most performant active mode and about 15 μW at its lowest-power retention mode). BOBBER utilizes SPI, the highest-bandwidth (2MB/s) communication peripheral supported by the MSP430, to communicate data and instruction set to the FPGA-based accelerator.

3.2 BOBBER Software Platform

Intermittent systems have to deal with frequent power failures and the associated correctness and forward progress issues. BOBBER takes a two-pronged approach: (1) for execution on the MCU we use a full checkpointing-based system to guarantee correctness and forward progress; (2) an idempotent interface to the FPGA accelerator. The flexibility of this interface allows the application to craft sufficiently small kernels to guarantee forward progress.

To illustrate this approach, consider the example application in the upper left of Figure 3. The application makes n computational kernel calls to the accelerator using the (`comp_temp`) template. During intermittent execution at point $C1$, the off threshold is reached, the MCU dies, and the program unceremoniously terminates. Under traditional execution, the program would start from the beginning again, loosing any forward progress made. To retain forward progress and to avoid any WAR dependencies, BOBBER uses TI's compute through power loss (CTPL) [43] to save all volatile state to a reserved region in the FRAM as the internal power rail approaches its brownout voltage. The volatile state includes registers (general purpose, stack pointer, and PC), peripheral interrupt settings, stack, and heap. Upon restart, the bootloading process (`wakeup()`) first restores the volatile SRAM state, registers, and interrupts, and then returns from the ADC interrupt to precisely the point at which the checkpoint was initiated.

While CTPL works for the MCU portions of the application, a death during an accelerator template call (e.g., point $C4$) can still cause incorrect execution since the local accelerator state is in volatile SRAM. CTPL will restart the program execution right there and not resend any data to the accelerator before asking the accelerator to perform the computation (`compute()`). The result is that the random startup values are transferred back to the MCU producing incorrect final results. In order to solve this BOBBER treats the template function as atomic—either it fully completes and stores the result back to FRAM or it is re-executed from the beginning. For correctness this function must be idempotent— re-execution of the function with the same inputs will produce the



A. Example application

```
main(){
  init();
  int *a= load_vals();
  for(int i=0;i<n;i+4){
    comp_temp(i,a[i],3);
  }
}
comp_temp(i,a[i],3)(){
  send(&a[i],4);
  compute(&a[i],3,1);
  read(&a[i],4);
 }
```

B. Intermittent power program sequence

```
main(){
  init();
  *a= load_vals()
  i = 0; //i inc to 4
  comp_temp(0,a[0],3);
  save(PC,SP,R12-R15);
  save_activation_record;
  check restore_reg_flag; // 0
  set interface_flg;
  send(&a[0],4);
  check restore_reg_flag; // 1
    restore(SP,R12-R15);
    restore_activation_record;
    clear restore_reg_flag
  set interface_flg;
  send(&a[0],4);
  compute (&a[0],4)
  read(&a[0],4)
  clear interface_flag;
  comp_temp(4,a[4],3);
  save(PC,SP,R12-R15);
  save_activation_record;
```

C. Checkpoint assembly pseudo code

```
wakeup()
  do DCO boot;
  restore SP,src,dst,
  calc DMA block size BS;
  do DMA;
  restore SR;
  nop;
  reta;
```

```
save_stack()
  cmpx.w #1, interface_flg;
  jnz alt_restore ;
savecpustack:
set src-> &SP, dst-> &FRAM, BS;
do DMA;
config DCO shutdown;
wait for shutdown();
alt_restore:
pushx.w PC to &ADC_ISR
mask SR bits;
set restore_reg_flag;
jmp savecpustack;
```

```
save_activation_record();
calc activation_record size(AS);
save AS;
set src-> SP dst-> &FRAM;
do DMA; ////SP+AS block size
reta;
```

```
restore_activation_record();
restore AS
set dst-> SP src-> &FRAM;
do DMA; //SP+AS block size
reta;
```

C1 ADC interrupt triggered -> calls save_stack()
C2 CPU comes back online to restore_stack() -> restores all the stack
C3 Enters comp_temp() -> base location(PC), SP, argument registers(R12-R15) and activation record is saved
C4 ADC interrupt triggered -> calls save_stack() and executes alt_restore -> saving the current stack and changes ADC interupt ISR return address to point to PC
C5 CPU comes back online to restore_stack() -> restores all the stack -> returns to PC, checks a flag restore_reg_flag -> restoring activation record and argument registers -> resumes operation.
PC
Stack
SE
&a
i=0
&a[0]
i=4
WAR
SP
Non-volatile FRAM
Application region
Reserved region
ADC
REF
IO
RTC
src
dst
R12
R15
Consistency problem (WAR) without BOBBER for i and &a[i]

Figure 3: Example BOBBER application and system software.

same outputs. To do this we must guarantee there are no WAR dependencies between variables stored in non-volatile state. While such dependencies between variables can be verified by a compiler, CTPL's checkpointing itself causes a WAR dependency when it saves the stack and the registers just before the MCU dies. Therefore BOBBER saves the registers and stack activation record at the start of the template (e.g., point $C3$). If the `interface_flag` is set on reboot (e.g., point $C5$), the registers and stack activation record from the beginning of the template are restored, effectively restarting the kernel call to the accelerator and guaranteeing correct execution.

4 DEMONSTRATION: INTERMITTENCY-AWARE CNN HW-SW ACCELERATOR FRAMEWORK

To demonstrate BOBBER's capabilities, we designed an intermittent-aware accelerator framework for one of the most computationally-challenging workloads for batteryless nodes—CNNs such as LeNet-5 [22] as used in [13, 32]. As shown in Figure 4, the framework takes a TFlite [42] model as input to a custom 16-bit TFlite-micro [8] runtime implementation. The model and TFlite-micro runtime are hosted on the MCU while 2D and depthwise convolution operators are accelerated by a 2x2 multiply and accumulate (MAC) accelerator on the FPGA.

4.1 BOBBER-based Convolution Accelerator

Figure 4 shows the 2x2 multiply and accumulate (MAC) accelerator hosted on the FPGA. The MAC is a 3-stage pipeline and has a variable-width instruction set architecture (ISA). Each instruction starts with a 4-bit opcode. The ISA comprises three instructions:

(1) Load data from MCU: 11-bit address, 10-bit count, and up to 2^{10} bytes
(2) 2x2 MAC: 11-bit addresses, 6-bit count, and up to 2^{6} bytes

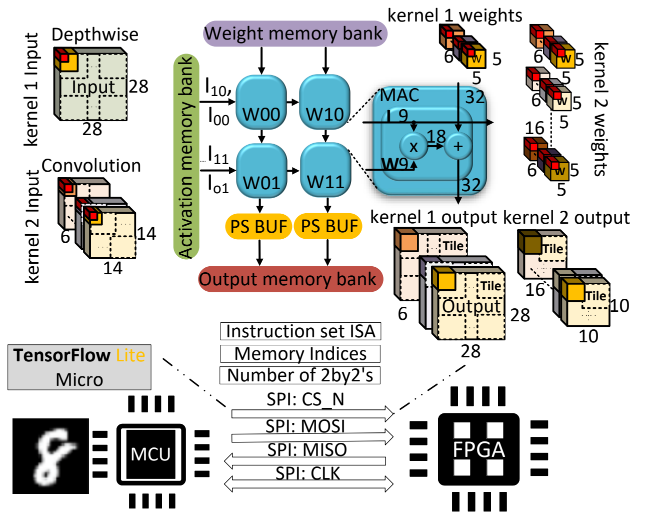


Figure 4: BOBBER-based intermittency-aware CNN accelerator framework.

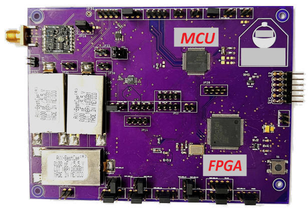
(3) Store data to MCU: 11-bit address and a single byte

The FPGA uses four of the Igloo Nano's eight 4608-bit SRAM banks for activations and weights, allowing double-buffering to overlap computation and communication. This state is lost when the FPGA is not powered.

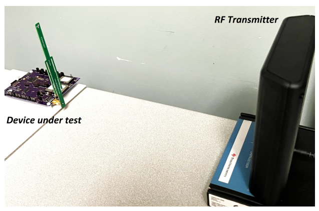
4.2 BOBBER-based CNN Application

The application software has two jobs: (1) Offloading computation to the accelerator in idempotent kernel calls and (2) providing knobs to enable forward progress under intermittent execution.

In order to map the TFlite-based CNNs to kernel calls, we partitioned layerwise TFlite operators to work with 2x2 MACs. Note that



(a) BOBBER platform (PCB front).



(b) BOBBER experimental setup.

Figure 5: BOBBER evaluation.

this is done once per operator and can be reused across the CNN or any other model using the operator. For example, consider the LeNet-5 [22] CNN shown in Figure 4. It has one depthwise and two convolution layers. For the depthwise layer, the input feature maps have height and width of 28 which uses a 5x5 filter along 6 output channel to give a 28*28*6 output. The layerwise 5x5 computations are first broken down to 2x2, padded where necessary, and accumulated at the end to get a single output pixel of 28*28*6. Once the operator computation is partitioned, we map the core computation to an inlined and idempotent BOBBER template function.

Even though BOBBER's software platform guarantees idempotence, the execution time of a single invocation of the compute kernel function may exceed the available energy buffer's on-time causing the computation to never complete. To enable forward progress, we dynamically tile the TFlite-micro operator template functions with respect to the output height p and width q. For a tile of width l and height l, each output channel has $\frac{p}{l} * \frac{q}{l}$ tiles. Each tile represents one invocation of the compute kernel. This results in a tradeoff: as the number of tiles increases, the overall performance decreases due to redundant computation and data movement. On the other hand, larger tile sizes also risk losing more computed data per node death that must be recomputed in the next on-time. To mitigate this risk an application can turn off early when it predicts the next tile cannot complete. The application uses off-line profiling and an estimate of available remaining on-time since boot to predict if the next tile can complete. Although not used by the application, BOBBER supports charge state sensing and monitoring (using an on chip ADC), which could be used to explore intelligent runtime tile selection algorithms.

5 METHODOLOGY

To demonstrate execution on RF energy-harvesting alone we placed BOBBER at 0.5m from a Powercast TX91501b RF transmitter [35] as shown in Figure 5b. Images were stored on the FRAM, wires disconnected, harvesting jumper set, and, after several minutes, wires were reattached and LeNet-5 inference results collected. Only the 50 mF capacitor is used as an energy store. Measurement data reported in Section 6 was collected using an RF signal generator feeding 9dBm (equivalent to 0.5m) of RF power into BOBBER's antenna SMA connector in order to repeatably compare tile sizes and design points. For latency measurements in continuous and intermittent power, we compare 4 design points, *MSP430*—execution using solely the MSP430, *MSP430 w-CTPL*—execution using MSP430 with only CTPL. *Baseline Accel*—execution using the FPGA custom accelerator with only CTPL and no tiling, and *BOBBER*—execution using BOBBER. A sniffer node is used to timestamp all the computational progress of the application, timestamping total inference time, number of computations, data movement, different operator times. Using this information, measured average system power, and the number of on-times per layer operator, we calculate the energy breakdown under intermittent power.

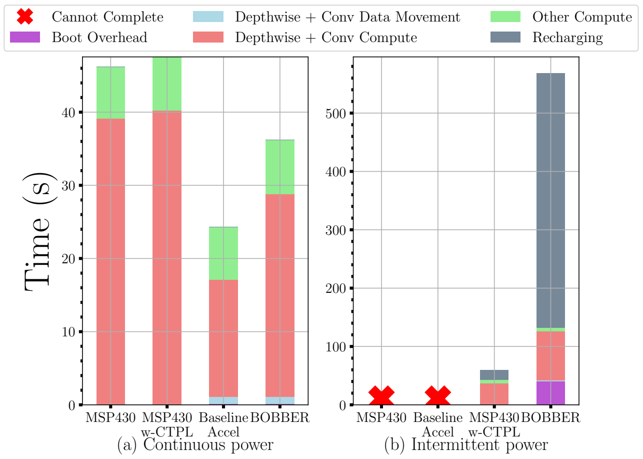


Figure 6: LeNet-5 total inference latency.

6 RESULTS

We evaluate the BOBBER platform in terms of total latency and energy for running correct and complete LeNet-5 inference under both continuous and intermittent power. Importantly, BOBBER enables correct and complete execution while powered solely by the RF energy harvester.

Figure 6a shows the overall inference latency measured under continuous power. In continuous power the Baseline Accel is 2.7x faster for the depthwise layer, 2.2x faster for the convolution layer, and 1.9x faster overall compared to the raw application running only on the MSP430. In continuous power, BOBBER's overall latency reduces to 1.3x better than the MSP430. The degradation arises from BOBBER's tiling—49 tiles for depthwise and 25 tiles for convolution. This tiling level guarantees reliable intermittent execution. However compared to the Baseline Accel, BOBBER has 1.4% more data movement and 73% more redundant computation.

Figure 6b shows the overall inference latency measured under intermittent power with a 50 mF capacitance. The MSP430 and Baseline Accel do not complete as their execution time exceeds one on-time and they have no checkpoint mechanism, while BOBBER and MSP430 w-CTPL complete and guarantee correctness. Since execution spans multiple on-times, BOBBER's latency is increased to 567s for a single inference. The added time comes from booting after powerloss (41s) and recharging the capacitor (477s). Under the current configuration an MSP430 + CTPL implementation outperforms BOBBER under intermittent operation. Figure 7 shows the reason why—BOBBER consumes 32.6mW of total average system power compared to 4.7mW for MSP430 w-CTPL. A significant combined overhead of 17 mW is due to the choice of Igloo Nano's 3.3V 20 MHz CMOS oscillator with a crystal base resonator and the 41% efficient 3.3 to 1.5V LDO regulator used from the Igloo Nano reference design. Using straightforward improvements in part selection, the total system power could be drastically reduced. Further optimizing the accelerator and interface (e.g., variable/reduced precision or

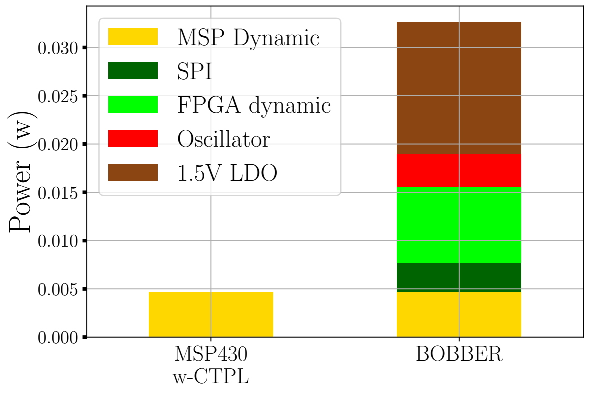


Figure 7: Characterization of average power consumption of different components in BOBBER for Depthwise and Conv layers.

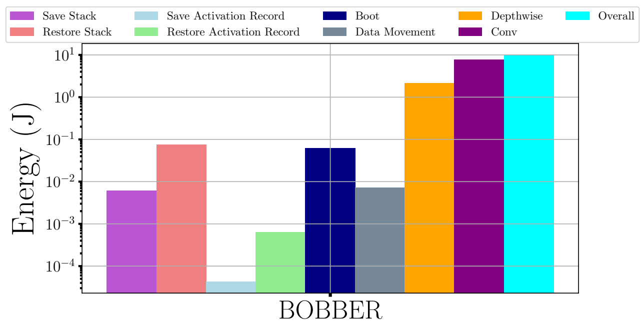
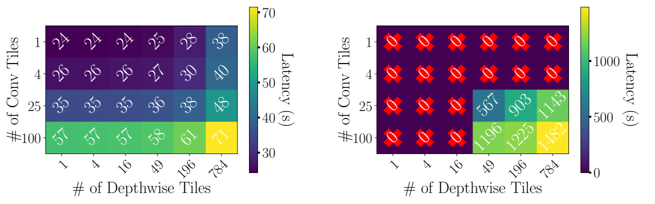


Figure 8: Energy breakdown for a full intermittent inference.



(a) BOBBER inference latency under continuous power. **(b) BOBBER inference latency under intermittent power.**

Figure 9: Comparison of LeNet-5 inference latency for different tile sizes in continuous power and intermittent power. Fewer tiles require less redundant computation, but largest tile's execution may exceed maximum on-time.

customizing based on layer filter size) would reduce the average power of the FPGA, SPI, and, as a result, the regulator overhead.

The time spent recharging the capacitor is dependent both on the total energy of the baseline computation, but also added energy costs from intermittent execution. Figure 8 shows the breakdown of energy costs for running a full inference with a 50 mF capacitor. The largest intermittency-related factors are the energy from FPGA boot and stack restoration at 61.2 mJ and 74.2 mJ, respectively. Combined these are still less than 1.4% of the total energy of the inference demonstrating that BOBBER imposes minimal overheads to guarantee correctness.

Figure 9a shows the inference latency under continuous power for different tile configurations in the convolution and depthwise

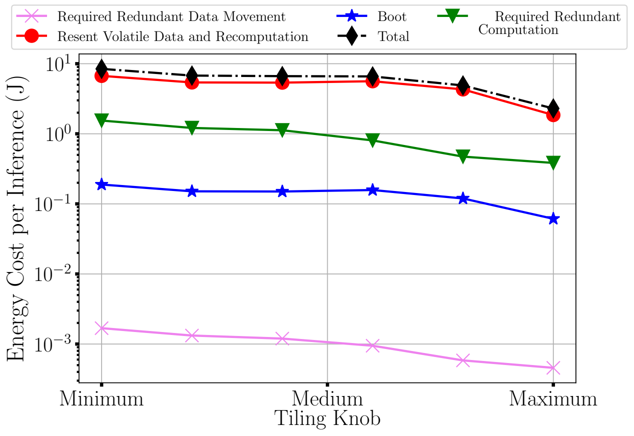


Figure 10: Tiling exploration for 50 mF capacitor.

convolution operators. Under continuous power, the best configuration has no tiling (# tiles = 1 for both operators) the worst performance is where the layers are tiled to the maximum possible extent. Under intermittent power, as shown in Figure 9b, decreasing the number of tiles still allows a lower inference latency until a single tile can no longer execute one kernel in a single on-time.

Larger tile sizes also risk an increased amount of wasted energy due to lost partial computations that must be recomputed and the corresponding input data that must be resent. Figure 10 plots the energy overheads due to tiling for tile sizes in decreasing order of required data movement (i.e., increasing effective tile size). As the tile size increases the energy cost of required (i.e., required under continuous power) redundant data and data movement decreases. Boot energy under intermittent execution also decreases as fewer on-times are required. Somewhat surprisingly, energy due to recomputation and resent data from lost volatile state also decreases. This occurs because our application chooses to die early before starting a tile if it estimates it cannot complete resulting in all tile sizes having a similar average data loss per on-time. Since larger tile sizes have fewer on-times, they actually have less recomputation.

7 CONCLUSION

This paper introduces BOBBER, the first batteryless intermittent FPGA platform, laying a foundation for quickly prototyping intermittent accelerators. BOBBER provides a full stack solution including PCB hardware design and system software provide correctness and forward progress on accelerated applications. Evaluations demonstrate BOBBER can correctly complete LeNet-5 inference while being solely powered by an RF energy harvester. The low overheads and flexibility of BOBBER can enable many future directions such as prototyping energy-adaptive bit-precision accelerators, intermittency-aware on-FPGA operator fusion, intermittent security co-processors, and other as yet unknown ideas.

8 ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation under Grant 2144757. Thanks to Dr Nathan Neihart and Mahmoud Gshash for their assistance and expertise during the evaluation of BOBBER. Thanks to Jake Meiss, Justin Sung, Zixuan Guo, Andrew Vogler, and Jake Tener who worked on a preliminary design of BOBBER.

REFERENCES

[1] Abu Bakar and Josiah Hester. 2018. Making sense of intermittent energy harvesting. In *Proceedings of the 6th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*. 32–37.

[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. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 12 (2016), 1968–1980.

[3] Domenico Balsamo, Alex S Weddell, Geoff V Merrett, Bashir M Al-Hashimi, Davide Brunelli, and Luca Benini. 2014. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embedded Systems Letters* 7, 1 (2014), 15–18.

[4] Wei-Ming Chen, Tei-Wei Kuo, and Pi-Cheng Hsiu. 2020. Enabling failure-resilient intermittent systems without runtime checkpointing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 12 (2020), 4399–4412.

[5] Alexei Colin, Graham Harvey, Brandon Lucia, and Alanson P Sample. 2016. An energy-interference-free hardware-software debugger for intermittent energy-harvesting systems. *ACM SIGARCH Computer Architecture News* 44, 2 (2016), 577–589.

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

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

[8] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Tiezhen Wang, et al. 2021. Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proceedings of Machine Learning and Systems* 3 (2021), 800–811.

[9] Jasper de Winkel, Carlo Delle Donne, Kasim Sinan Yildirim, Przemysław Pawełczak, and Josiah Hester. 2020. Reliable timekeeping for intermittent computing. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. 53–67.

[10] Vishal Deep, Vishak Narayanan, Mathew Wymore, Daji Qiao, and Henry Duwe. 2020. HARC: A Heterogeneous Array of Redundant Persistent Clocks for Batteryless, Intermittently-Powered Systems. In *2020 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 270–282.

[11] Harsh Desai and Brandon Lucia. 2020. A power-aware heterogeneous architecture scaling model for energy-harvesting computers. *IEEE Computer Architecture Letters* 19, 1 (2020), 68–71.

[12] Conrad Donovan, Alim Dewan, Deukhyoun Heo, and Haluk Beyenal. 2008. Batteryless, wireless sensor powered by a sediment microbial fuel cell. *Environmental science & technology* 42, 22 (2008), 8591–8596.

[13] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence beyond the edge: Inference on intermittent embedded systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*.

[14] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. 2015. Tragedy of the coulombs: Federating energy storage for tiny, intermittently-powered sensors. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 5–16.

[15] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 1–13.

[16] Matthew Hicks. 2017. Clank: Architectural support for intermittent computation. *ACM SIGARCH Computer Architecture News* 45, 2 (2017), 228–240.

[17] Texas Instruments. 2022. *Low-Energy Accelerator (LEA)*. Retrieved Sep 23, 2022 from https://www.ti.com/lit/an/slaa720/slaa720.pdf

[18] Texas Instruments. 2022. *MSP430FR599x, MSP430FR596x Mixed-Signal Microcontrollers*. Retrieved Sep 23, 2022 from https://www.ti.com/lit/ds/symlink/msp430fr5994.pdf

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

[20] Chih-Kai Kang, Hashan Roshantha Mendis, Chun-Han Lin, Ming-Syan Chen, and Pi-Cheng Hsiu. 2020. Everything leaves footprints: Hardware accelerated intermittent deep inference. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 3479–3491.

[21] Alicia Klinefelter, Nathan E Roberts, Yousef Shakhsheer, Patricia Gonzalez, Aatmesh Shrivastava, Abhishek Roy, Kyle Craig, Muhammad Faisal, James Boley, Seunghyun Oh, et al. 2015. 21.3 A 6.45 μW self-powered IoT SoC with integrated energy-harvesting power management and ULP asymmetric radios. In *2015 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*. IEEE, 1–3.

[22] Yann LeCun et al. 2015. LeNet-5, convolutional neural networks. *URL: http://yann.lecun.com/exdb/lenet* 20, 5 (2015), 14.

[23] Seulki Lee and Shahriar Nirjon. 2019. Neuro.ZERO: A Zero-Energy Neural Network Accelerator for Embedded Sensing and Inference Systems. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems* (New York, New York) *(SenSys '19)*. Association for Computing Machinery, New York, NY, USA, 138–152. https://doi.org/10.1145/3356250.3360030

[24] Brandon Lucia and Benjamin Ransford. 2015. A Simpler, Safer Programming and Execution Model for Intermittent Systems. *SIGPLAN Not.* 50, 6 (jun 2015), 575–585. https://doi.org/10.1145/2813885.2737978

[25] Kaisheng Ma, Xueqing Li, Jinyang Li, Yongpan Liu, Yuan Xie, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2017. Incidental computing on IoT nonvolatile processors. In *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 204–218.

[26] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2019. Alpaca: Intermittent execution without checkpoints. *arXiv preprint arXiv:1909.06951* (2019).

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

[28] Scott A. Mahlke, William Y. Chen, Roger A. Bringmann, Richard E. Hank, Wen-Mei W. Hwu, B. Ramakrishna Rau, and Michael S. Schlansker. 1993. Sentinel Scheduling: A Model for Compiler-Controlled Speculative Execution. *ACM Trans. Comput. Syst.* 11, 4 (nov 1993), 376–408. https://doi.org/10.1145/161541.159765

[29] Andrea Maioli, Luca Mottola, Muhammad Hamad Alizai, and Junaid Haroon Siddiqui. 2019. On Intermittence Bugs in the Battery-Less Internet of Things (WIP Paper). In *Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems* (Phoenix, AZ, USA) *(LCTES 2019)*. Association for Computing Machinery, New York, NY, USA, 203–207. https://doi.org/10.1145/3316482.3326346

[30] Amjad Yousef Majid, Carlo Delle Donne, Kiwan Maeng, Alexei Colin, Kasim Sinan Yildirim, Brandon Lucia, and Przemysław Pawełczak. 2020. Dynamic Task-Based Intermittent Execution for Energy-Harvesting Devices. *ACM Trans. Sen. Netw.* 16, 1, Article 5 (feb 2020), 24 pages. https://doi.org/10.1145/3360285

[31] Microsemi 2020. *IGLOO nano Low Power Flash FPGAs Datasheet*. Microsemi.

[32] Matteo Nardello, Harsh Desai, Davide Brunelli, and Brandon Lucia. 2019. Camaroptera: A Batteryless Long-Range Remote Visual Sensing System. In *Proceedings of the 7th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems* (New York, NY, USA) *(ENSsys'19)*. Association for Computing Machinery, New York, NY, USA, 8–14. https://doi.org/10.1145/3362053.3363491

[33] Jens F Peters, Manuel Baumann, Benedikt Zimmermann, Jessica Braun, and Marcel Weil. 2017. The environmental impact of Li-Ion batteries and the role of key parameters–A review. *Renewable and Sustainable Energy Reviews* 67 (2017).

[34] Powercast 2016. *P2110B 915 MHz RF Powerharvester Receiver*. Powercast.

[35] Powercast 2018. *TX91501B–915 MHz Powercaster Transmitter*. Powercast.

[36] Keni Qiu, Nicholas Jao, Mengying Zhao, Cyan Subhra Mishra, Gulsum Gudukbay, Sethu Jose, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2020. ResiRCA: A resilient energy harvesting ReRAM crossbar-based accelerator for intelligent embedded processors. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE.

[37] Keni Qiu, Nicholas Jao, Kunyu Zhou, Yongpan Liu, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2021. MaxTracker: Continuously Tracking the Maximum Computation Progress for Energy Harvesting ReRAM-Based CNN Accelerators. 20, 5s, Article 78 (sep 2021), 23 pages. https://doi.org/10.1145/3477009

[38] D. C. Ranasinghe, R. L. Shinmoto Torres, A. P. Sample, J. R. Smith, K. Hill, and R. Visvanathan. 2012. Towards falls prevention: A wearable wireless and batteryless sensing and automatic identification tag for real time monitoring of human movements. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*.

[39] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: System Support for Long-Running Computation on RFID-Scale Devices. *SIGPLAN Not.* 46, 3 (mar 2011), 159–170. https://doi.org/10.1145/1961296.1950386

[40] Simone Ruffini, Luca Caronti, Kasım Sinan Yıldırım, and Davide Brunelli. 2022. NORM: An FPGA-Based Non-Volatile Memory Emulation Framework for Intermittent Computing. *J. Emerg. Technol. Comput. Syst.* 18, 4, Article 73 (oct 2022), 18 pages. https://doi.org/10.1145/3517812

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

[42] TensorFlow. 2022. *Deploy machine learning models on mobile and edge devices*. Retrieved Sep 23, 2022 from https://www.tensorflow.org/lite

[43] Texas Instruments 2015. *Intelligent System State Restoration After Power Failure With Compute Through Power Loss Utility*. Texas Instruments.

[44] Texas Instruments 2017. *MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family User's guide*. Texas Instruments.

[45] Z. Yuan, Y. Liu, J. Li, J. Hu, C. J. Xue, and H. Yang. 2017. CP-FPGA: Energy-Efficient Nonvolatile FPGA With Offline/Online Checkpointing Optimization.

IEEE Transactions on Very Large Scale Integration (VLSI) Systems 25, 7 (2017).

[46] Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. 2011. Moo: A batteryless computational RFID and sensing platform. *University of Massachusetts Computer Science Technical Report UM-CS-2011-020* (2011).

[47] Xinyi Zhang, Clay Patterson, Yongpan Liu, Chengmo Yang, Chun Jason Xue, and Jingtong Hu. 2020. Low Overhead Online Data Flow Tracking for Intermittently Powered Non-Volatile FPGAs. *J. Emerg. Technol. Comput. Syst.* 16, 3, Article 26 (jul 2020), 20 pages. https://doi.org/10.1145/3371392