Energy-aware HW/SW Co-modeling of Batteryless Wireless Sensor Nodes

Samuel C.B. Wong
University of Southampton, UK
scbw1g19@ecs.soton.ac.uk

Sivert T. Sliper
University of Southampton, UK
sts1u16@ecs.soton.ac.uk

William Wang
Arm Research Cambridge, UK
william.wang@arm.com

Alex S. Weddell
University of Southampton, UK
asw@ecs.soton.ac.uk

Stephanie Gauthier
University of Southampton, UK
s.gauthier@soton.ac.uk

Geoff V. Merrett
University of Southampton, UK
gvm@ecs.soton.ac.uk

ABSTRACT

Energy harvesting wireless sensor nodes are sensitive to spatial and temporal fluctuations in energy availability. This issue is especially prevalent in batteryless systems, where devices are directly connected to power sources with little or no buffering. The strong coupling of energy supply and demand introduces a new dimension to the problem of designing robust networked sensing systems. We propose a modeling framework for this class of batteryless systems with an emphasis on the interactions between energy and function. The tool models energy harvesters, power management circuitry, energy storage, microcontrollers, sensors, radio modules, environmental models, and is fully extensible. The microcontroller model is based on cycle-accurate instruction set simulators from *Fused*, with various peripheral extensions to enable board-level functionality, such as SPI, DMA, hardware multiplier etc. The tool enables virtual prototyping of self-powered wireless sensor nodes, but is especially useful for studying intermittent operation and developing application specific software, hardware, or combined solutions. The simulator is capable of executing real workloads under realistic conditions and this is demonstrated through a case study where the same compiled binary is executed on a virtual prototype and its corresponding physical wireless sensor system to yield matching digital traces and current profiles.

KEYWORDS

Wireless Sensor Node, Batteryless, Energy Harvesting, Transient Systems, Intermittent Computing, Hardware Software Co-Design, Electronic System Level Modeling, Virtual Prototyping. SystemC

ACM Reference Format:

ENSsys '20, November 16–19, 2020, Virtual Event, Japan
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8129-1/20/11…$15.00
https://doi.org/10.1145/3417308.3430272

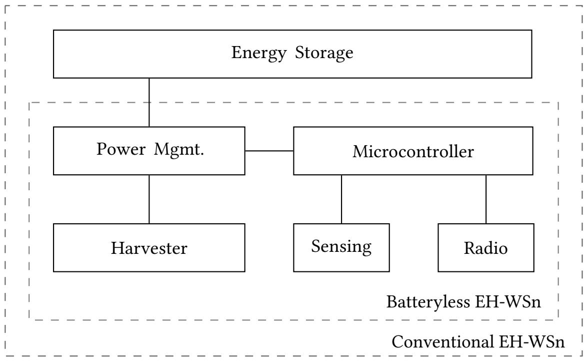


Figure 1: A typical energy harvesting wireless sensor node.

1 INTRODUCTION

Batteryless energy-harvesting wireless sensor nodes (EH-WSn) is a promising technology towards achieving the envisioned vast scale of the Internet of Things (IoT). Powering the tens of billion of devices in the near future [1] with existing battery technologies is neither sustainable nor economical; batteryless devices become a necessity at this scale. Batteryless networked smart sensor systems are, however, difficult to design and an energy driven paradigm is required [2]. These systems are directly exposed to temporal fluctuations in environmental power sources. Further complicating matters is the significant temporal variation in current draw from the various active loads, which dynamically affect the efficiency of the power delivery path.

Existing modeling tools usually assume a stable power supply, or model variations as fixed voltage traces. However, as shown by Hester *et al.*, the harvested power depends strongly on the load [3]. To correctly model batteryless EH-WSns, a comprehensive approach that captures interactions between energy and execution is needed.

The authors of *Fused* postulated that power and performance must be modelled simultaneously for energy-driven computers [4]. For simulating batteryless EH-WSns, we propose the following additional criteria:

(1) *Fine grained* power and functional models of WSn components that capture transient energy demands.
(2) Sufficient *coverage* of WSn components such that no significant current sinks are neglected, and interactions between components are captured.

(3) Realistic and representative *environment models* and workloads to capture the range of possible operating conditions.

More generally, in anticipation of innumerous WSn use cases, a practical modeling tool should be *flexible* and *extensible*, and also promote *code reuse* wherever possible. Another appealing property is full *code congruity* (i.e. that the exact same software can run on the hardware and the virtual prototype of the hardware). This property is crucial in enabling effective *hardware-software co-design*, as well as debugging, of WSn systems, ultimately enabling creative solutions such as those presented in [5] and [6].

The main contributions of the work reported in this paper are:

(1) A modeling framework for batteryless EH-WSn that accurately capture timing, energy, and functionality of the HW/SW components, as well as their interactions.
(2) Code contribution to the open source *Fused* simulator for models of the various on-chip/off-chip peripherals, power conditioning analog circuitry, and energy harvesters, as well as drivers and validation effort to enable virtual prototyping of complete batteryless EH-WSn.
(3) A case study illustrating the effective and accurate virtual prototyping of a batteryless EH-WSn executing a realistic workload consisting of sensing, compute, and radio transmission, which is validated against its physical counterpart.

2 BACKGROUND & RELATED WORKS

In this section, background information on batteryless EH-WSn and its design is presented. We also make the case for needing a new framework for batteryless EH-WSn by reviewing existing simulation tools and critically evaluating their suitability.

2.1 Batteryless Wireless Sensor Nodes

A typical WSn consists of a compute core, sensing elements, a radio, power management circuitry, and energy storage, as illustrated in Figure 1. A batteryless WSn minimizes or completely removes the energy storage element to achieve a smaller footprint, reduce Bill of Material (BOM) cost, and ease maintenance. As the dominant battery technologies today are environmentally hazardous, costly tracking of the entire WSn product life cycle is required, whereas batteryless WSns enable true deploy and forget scenarios.

Challenges from going Batteryless Due to the lack of a sufficiently large energy buffer, the system is exposed to temporal variation in the power source, including unpredictable periods of power outage. The microcontroller needs to sustain computation across these power cycles. It will also need to keep track of the state of the many peripherals and their workloads, both on- and off-chip. Furthermore, the device loses track of time during periods of complete power outage. The implication is that most duty-cycling-based schedulers or networking protocols simply cannot work on batteryless WSns.

Enabling Research Many ingenious solutions addressing the aforementioned issues have been proposed. For example, there are solutions to sustain computing through intermittent power cycles, e.g. *Ratchet* [7], *Hibernus* [8], *INK* [9], and *TotalRecall* [10], tracking peripherals, e.g. *Karma* [11] and *RESTOP* [12], timekeeping through power loss, e.g. *[Cus]TARDIS* [13] and *CHRT/Botoks* [14]. However, the custom hardware required in some of these work, and the absence of a common platform, hinders effective collaboration and comparison.

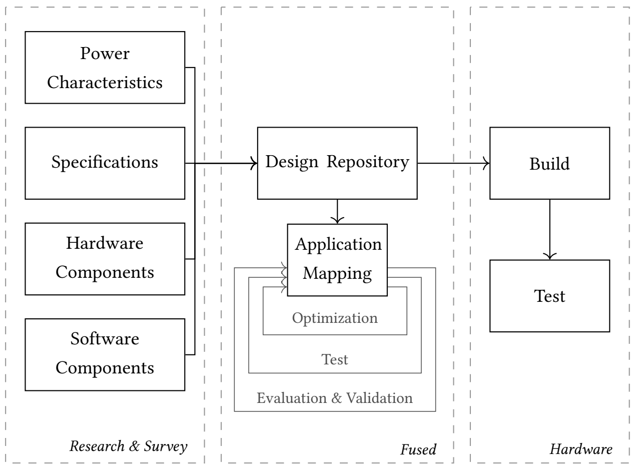


Figure 2: Design flow of batteryless wireless sensor nodes. The power characteristics, including power supply and consumption, is made explicit in the design repository.

2.2 Model-based Batteryless EH-WSn Design

Batteryless EH-WSns are energy-driven, i.e. their performance varies with energy availability. This variability requires the WSn to be adaptive, e.g. scale performance in an energy-aware fashion and be resilient to loss of power supply. To facilitate effective batteryless EH-WSn design, we propose the systematic flow shown in Figure 2. Power supply and consumption characteristics are made explicit as a major component in the design repository, on par with the application specifications, hardware components, and software components. Multiple iterations of application mapping, optimization, evaluation, validation, and tests can then be carried out in the modeling framework before building the physical device/prototype. Ideally, the framework acts as a virtual prototyping platform and an emulator, allowing the exploration of different architectural, hardware,and software solutions.

Nonetheless, equally important are the work before and after the modeling stage. Early on, analytical models like *EH-Model* [15] can be used to rapidly explore the design space. Tools like *Ekho* [3] and *Shepherd* [16] aid the designer throughout the design flow by providing a means for accurate power profiling (research and survey) and replaying the traces during modeling and/or hardware validation. Other important tools include *EDB* [17], which is an energy-aware debugging utility, and hardware prototyping platforms such as *Flicker* [18]. Together these tools, with an effecting virtual prototype framework, enables the effective study and design of batteryless EH-WSns.

2.3 Short-listed Simulators

A summary of various candidate simulators is given in Table 1. We next discuss why these tools are ill-suited to be used in our virtual prototyping framework for batteryless EH-WSn.

Table 1: Comparison of various simulators for batteryless EH-WSn.

Simulator	Timing Accuracy	Energy Model	Code Congruity	Ext. Peripherals	AMS[1]	SW Debug	Language
MSPSim-based **[19] [20] [21]**	indiscriminate instruction level, no wall-clock time	indiscriminate instruction level, limited to MCU	yes (MSP430 ISA)	numerous USARTs devices	no	limited support for GDB, custom CLI	Java
gem5, QEMU **[22] [23]**	some accurate cycle-level processor models in gem5	events/activity-mapped power estimation [24]	yes (numerous ISAs)	gem5-no, QEMU used for developing HW drivers	no	yes	C/C++, Python
RTL, e.g. NVsim **[25]**	RTL, cycle accurate simulation	allows for very accurate power profiling	yes	no	no	no	HDL
WSN Simulators [26][27][28]	instruction level modeling support lacking	yes but lacks models of processors	no, uses models of software	yes	yes	no	MATLAB, Python, SystemC etc.
Fused **[4]**	cycle-accurate execution, loosely timed simulation	event- & state-based active current draw	yes (ARMv6-m, MSP430)	no (extended in this work)	limited (extended in this work)	GDB, debug lock-step with power trace	C/C++, SystemC-TLM w/ AMS

[1] Analog & Mixed-Signal simulation, i.e. capability to properly simulate analog circuits together with digital execution.

MSPsim-based. A popular simulator for WSn is *MSPsim* [19], which is a Java-based instruction level emulator of the MSP430 series microprocessor and supports loading of binaries. However, *MSPsim* does not have an accurate power modeling framework and lacks proper simulation interfaces and abstractions, such as an event queue, a global clock, communication channels etc. While *Siren* [21] and *MSPsim++* [20] extended *MSPSim* to include basic energy simulation capabilities and, in the case of *Siren*, extensions for modeling nonvolatile memory, they fall short because they assume static energy consumption per instruction and only accommodate very simple energy calculations.

Processor/SoC and RTL Simulators. Another popular simulation framework is *gem5* [22], which provides fast simulation of processor-memory systems and provides a power consumption estimation utility through *McPAT* [24]. However, *gem5* mainly targets high-performance-computing (HPC) systems with deep pipelines, multiple levels of caches, memory management units etc. Another similar tool is QEMU, which utilizes dynamic instruction set translation [23]. It is mainly used for hardware-assisted virtualization due to its speed but cycle-accurate information is lost as a result. These simulators are immensely powerful but target application-class processors. Then there are highly targeted and detailed RTL simulations such as those tailored for nonvolatile processors (NVP), i.e. processors that can sustain computation through power outages like *NVsim* [25]. RTL simulation demands too much low-level details and complexities to be an effective virtual prototyping tool.

Wireless Sensor Network Node Simulators. These are simulators that target wireless sensor nodes with explicit intent to use these nodes in a network simulation context. MATLAB is a popular tool for these simulators [27] [28] but SystemC-based models also exist [26]. This class of simulators fall short of cycle-accurate modeling and are therefore unable to meet state-of-the-art batteryless WSn research needs in areas like intermittent computing.

3 PROPOSED MODELING FRAMEWORK

From our review, *Fused* is most suitable to be the basis for our batteryless EH-WSn prototyping framework. It is written in C++using the SystemC-TLM library [29] and the Analog Mixed Signalextension (SystemC-AMS) [30], i.e. it uses a standardized simulation interfaces and has options for analog/mixed-signal modeling. Fused implements a cycle-accurate microcontroller CPU and supports emulation of the MSP430 and ARMv6-m ISAs, at the time of writing. For power and functional models of memory and peripherals, Fused uses a mixture of TLM and discrete event simulation as the model of computation (MoC).

We extend this framework to include radio modules and other on- and off-chip peripherals, which we also model using TLM and discrete event simulation. An off-chip peripheral, e.g. the NRF24 radio shown in Figure 7, will usually have a communication interface (SPI in the case of the NRF24), a register file containing the control/status registers, and a state machine. For the radio module, transmit/receive buffers are also implemented and a PHY packet class object is generated for each transmit event. Apart from the packet content (e.g. address, payload, CRC etc.), the packet is also tagged with metadata such as transmit duration. This packet is currently printed to standard output when generated; in the future it will be passed through a wireless channel model to another WSn.

In the analog domain, *Fused* includes a model of an ideal capacitor as energy storage, a simple energy management circuit, and a voltage-limited current source, all modelled using the Timed Data Flow (TDF) MoC of *SystemC-AMS*. We model more complex analog devices along the power delivery path, and to that end use a combination of *SystemC AMS*'s TDF MoC, the Linear Signal Flow (LSF) MoC, and the Electrical Linear Networks (ELN) MoC. For instance, we use this approach to implement a model for the buck-boost converter and the solar cell shown in Figure 4 and Figure 5 respectively. The analogue and digital domains have a shared view of simulation time thus allowing for time-accurate simulation results. However, modules are allowed to advance local simulation time if there are no dependencies, thus enabling fast simulations, which greatly benefits virtual prototyping workflows. Figure 3 illustrates the proposed modeling framework.

Apart from hardware components, the WSn firmware is also an important part of the virtual prototype. In our framework, we ensure full *code congruity*, i.e. we make sure that the code executing on the virtual prototype can be used without modification on the physical target. Where available, we use standard libraries/drivers

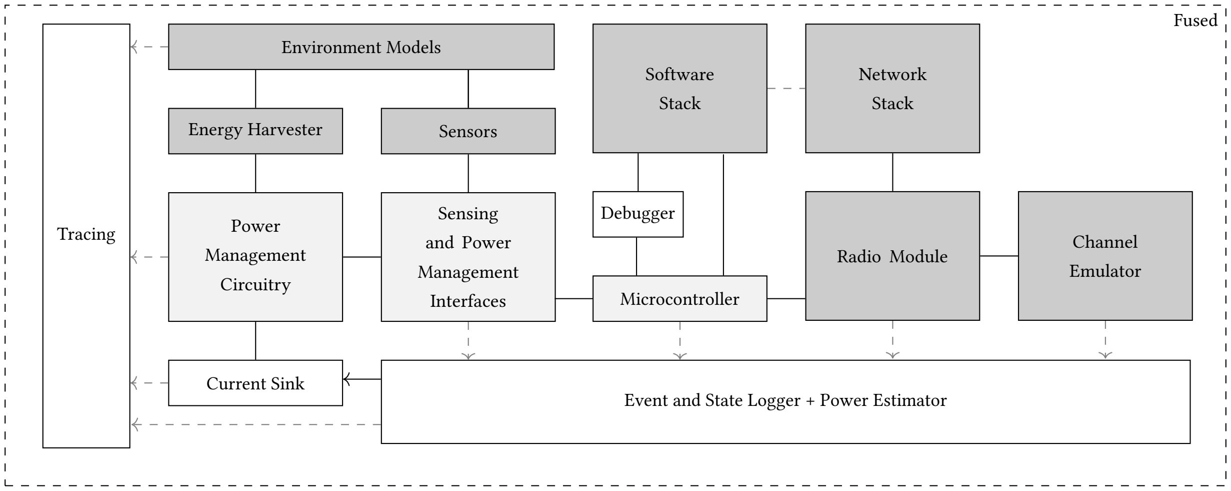


Figure 3: Overview of proposed virtual prototyping framework for batteryless EH-WSn. Shaded blocks indicate extensions or modifications (lighter shade) to the original *Fused* simulator (white).

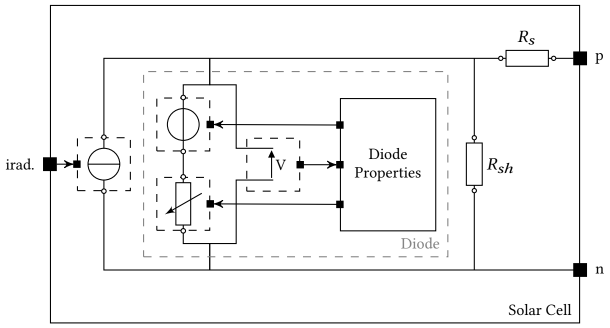


Figure 4: Single diode solar cell model. The diode properties are specified with the TDF MoC whereas the linear components uses the ELN MoC.

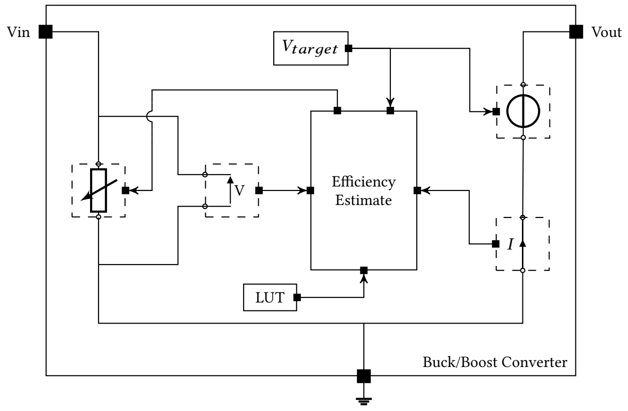


Figure 5: TDF model of buck/boost converter at equilibrium, which allows for faster simulation.

provided by the vendors to further enforce this requirement. Building virtual prototype amounts to connecting up hardware components in a 'board-level' abstraction and providing the necessary stimuli, e.g. loading executable binaries into memory. The building blocks are simulated together as a single Design Under Test (DUT) or separately by using unit-test style stimuli and monitors in a testbench.

A mechanism for unit testing is implemented for testing individual modules. Unit tests are executable specifications that are written in C++ according to the vendors' datasheets. It may also be used to check high-level target specifications for a bespoke hardware module. A full-system test usually means loading firmware into the emulated memory and debugging using the integrated GDB server. Similarly to when debugging code on a prototyping board, code can thus be stepped through one instruction/source line at a time. However, debugging the virtual prototype is often much more effective due to the availability of traces and logs of the internal signals, power rails, state machines, and bus traffic, etc.

3.1 Power Model

Active Current Draw. Active components, e.g. the microcontroller, the radio, and sensors, determine the active current draw. We use the existing modeling methodology in *Fused*, where high-level events/states registered from the active components map to a certain energy/power contribution, respectively, which are then combined into the total load current draw. In general, events are used to model detailed dynamic power consumption, whereas states are used for modelling static power (i.e. leakage) or rough dynamic power consumption (e.g. the average active/standby/sleep power of a module). Together, events and states of components determine the total current draw of the WSn within a time-slice. Direct playback of active current traces associated with special events is used occasionally, e.g. for the hardware startup of the microcontroller where Fused replays the current and time consumed before the first instruction after reset is fetched. The current draw in external peripherals are mostly state-based as annotated in Figure 7.

Passive Operating Points. Active load current determines the operating points of other components and the corresponding power

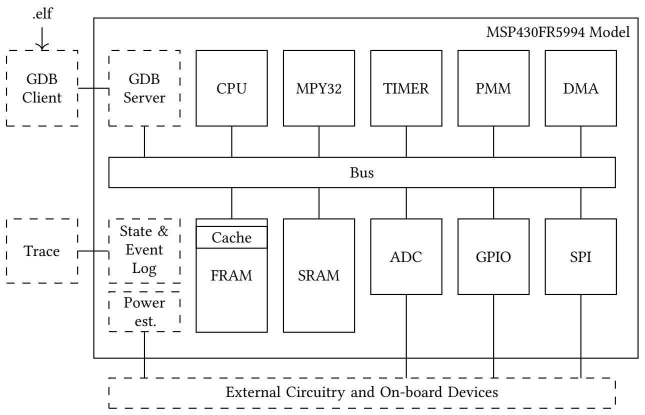
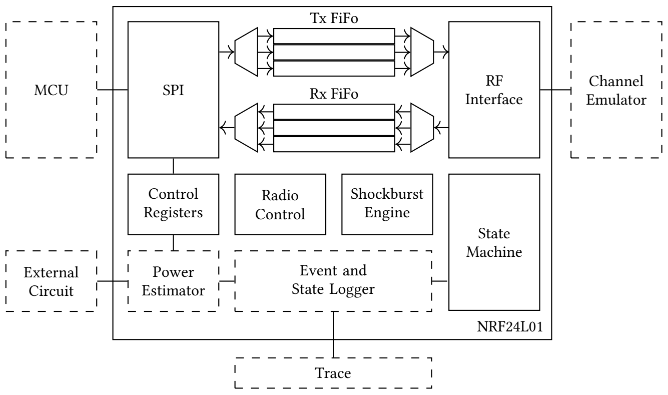
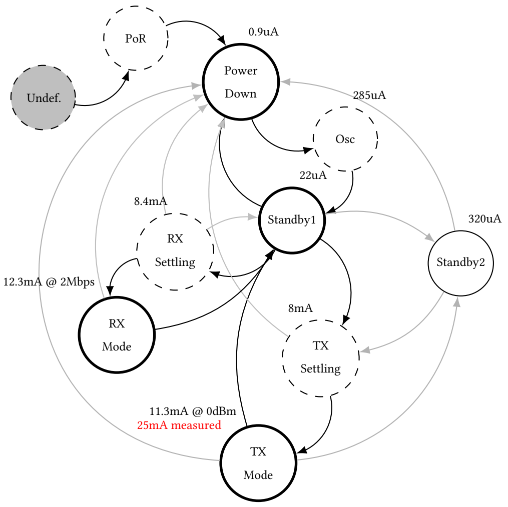


Figure 6: Implemented functional units in the MSP430FR5994 microcontroller model.



(a) NRF24 Short Range Radio Model Architecture.



(b) NRF24 Radio State Diagram.

Figure 7: NRF24 module with state-based power-model.

delivery efficiency from sources and the dissipation in the power conditioning circuitry. For instance, the efficiency of a typical buck converter falls sharply as the load current decreases and this is captured in our framework. Device behaviour such as this necessitates the closed-loop approach employed. The ELN MoC used in our framework, e.g. in Figure 4 and 5, captures these dynamics/interactions. Where appropriate, a TDF MoC is used, e.g. for diode properties, to balance between accuracy and simulation speed.

Power Profiling. Information regarding timing and power levels for operational states are mostly obtainable from the manufacturer's datasheets for the device/component. The implemented models are later validated with measurements on the physical device. For use cases where information from the manufacturer is inadequate, power consumption is measured on physical hardware and regressed onto time-accurate events and states. This had been done in *Fused* to profile the current consumption of the MSP430FR5994 microcontroller based mostly on memory accesses. In this work, the radio module used in our experiments turned out to be a counterfeit, and does not conform to the power specifications in the datasheets. We updated the power model to reflect the measurements on the device we have on hand and these values are recorded alongside the datasheets values in Figure 7.

4 BATTERYLESS ENVIRONMENTAL SENSOR CASE STUDY AND RESULTS

We built a virtual prototype of a full batteryless WSn, including all integrated circuits on the printed circuit board, as shown in Figure 8. In this setup a solar cell trickle charges a small capacitor, which is being monitored by a comparator. When the voltage across the capacitor exceeds the ON threshold, the switch to the active components is closed and the microcontroller turns on. The MCU then takes over the source monitoring duty and keeps the power switch closed. The closed-loop energy modeling in *Fused* means that as the active current load varies, the solar cell's operating point on its IV curve shifts accordingly. The same is true for the boost converter model used.

The microcontroller used in this system is the MSP430FR5994, as shown in Figure 6. The MSP430 processor, memory, bus, and various internal functional units/peripherals are implemented. Part of the MSP430 processor model is the instruction set simulator which allows the execution of real application binaries. In the simulated system, the application code in this case performs a sense operation on the BME280 environmental sensor, a compute block to process the raw readings, and finally sends the payload to the NRF24 radio modules to be transmitted.

4.1 Results and Discussion

Figure 9 shows the output traces from the virtual prototype and also includes the active current trace measured from the corresponding hardware. The results demonstrate the high accuracy of our simulation, which captures various power states of the batteryless EH-WSn and timing of events such as sensing and radio transmission. The model can also be further tuned to account for device-to-device variation. Furthermore, our framework captures the tight coupling/interactions in a EH-WSn, e.g. SPI packets triggering external peripheral functionality, radio transmission causing an immediate voltage drop across the capacitor etc. This enables

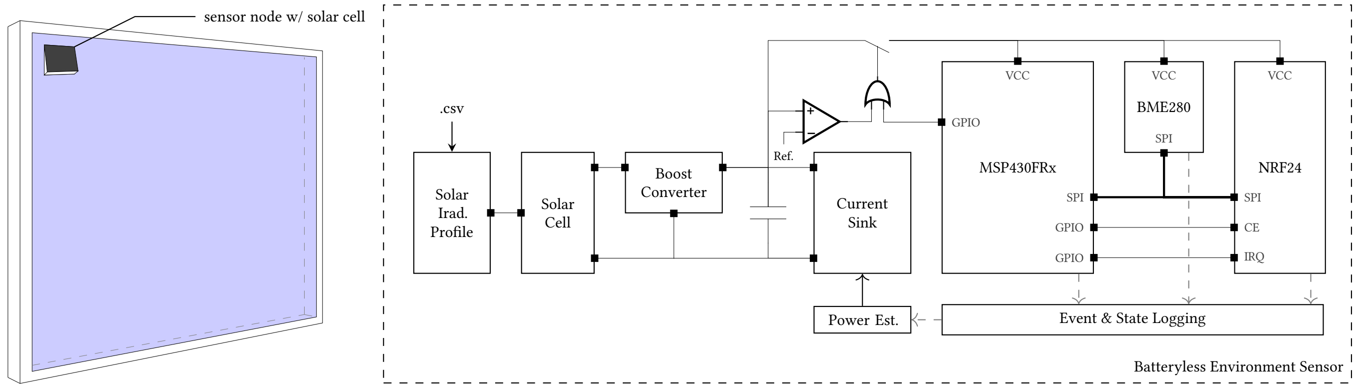


Figure 8: Virtual prototype of batteryless energy harvesting environmental sensor node.

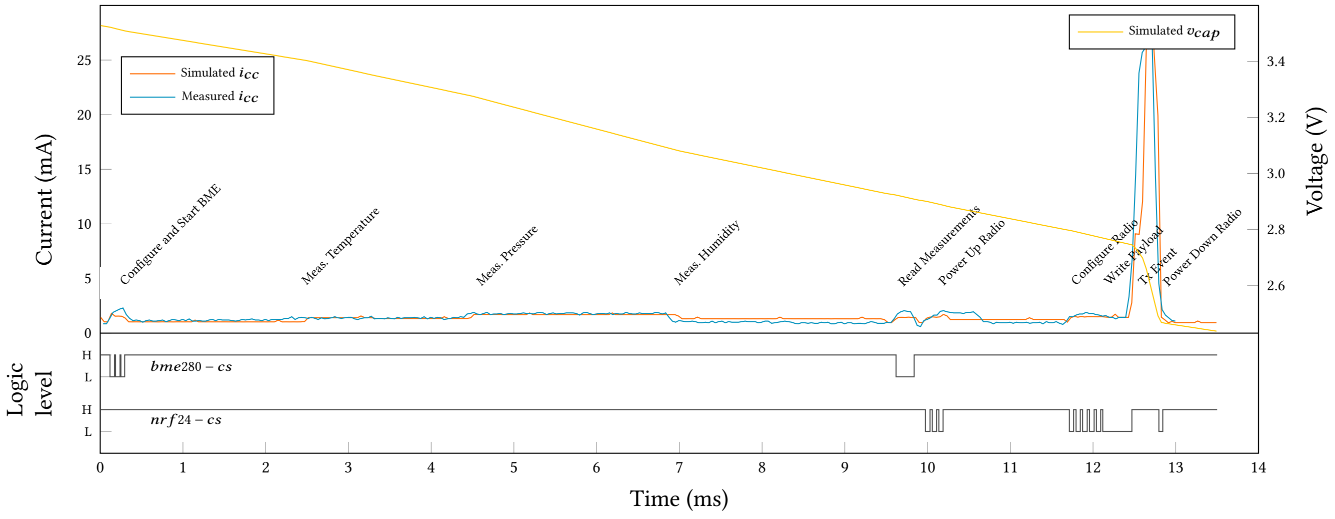


Figure 9: Analogue and digital traces from the virtual prototype of a batteryless EH-WSn performing a simple sense and transmit workload. The measured active current draw from the physical hardware is shown for comparison. The active-low chip-select signals indicate SPI communication windows. After booting the microcontroller, the supply current to the 22μF capacitor is cut-off and the voltage is allowed to decay over the period of this workload.

virtual prototyping HW/SW solutions that rely on timely reaction to changes in the supply rail or other cyber-physical events.

In addition, from the virtual prototype, variables and internal signals, e.g. state machines and on-chip bus traffic can be traced. This highlights the utility of the virtual prototyping approach in accelerating the design and development process. The output traces and power delivery circuit simulation is in lock step with code execution, hence pausing code execution pauses the entire system, including e.g. the position of the sun, a radio packet halfway being transmitted etc. The events leading up to that point can be investigated and the entire system can then be resumed without affecting device operation.

5 CONCLUSION

This work extends the original *Fused* simulation framework for device-level full system simulation of batteryless EH-WSn. While tailored for energy-driven systems, this tool will also benefit WSn device modeling and development in general. Effective virtual prototyping can be done due to the fast and accurate emulation of systems. Development and validation effort for the framework is ongoing as we integrate more components, explore different use cases, and experiment with emerging technologies.

Future work. (1) Case studies and demonstrators of various batteryless EH-WSn use cases. (2) Architectural exploration and experimenting with emerging technologies.

ACKNOWLEDGMENTS

This work was supported in part by the Engineering and Physical Sciences Research Council UK, under Platform grant EP/P010164/1. Source code is available at https://github.com/UoS-EEC. Experimental data associated with the paper is available at https:// doi.org/ 10.5258/SOTON/D1593.

REFERENCES

[1] Peter Jonsson, Stephen Carson, et al. Ericsson mobility report. *Ericsson, Sweden, Tech. Rep. EAB-19*, 2019.

[2] Geoff V Merrett and Bashir B M Al-Hashimi. Energy-driven computing: Rethinking the design of energy harvesting systems. In *Design, Automation &amp; Test in Europe Conference &amp; Exhibition (DATE), 2017*. IEEE, May 2017.

[3] Josiah Hester, Timothy Scott, and Jacob Sorber. Ekho: Realistic and repeatable experimentation for tiny energy-harvesting sensors. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, page 330–331, New York, NY, USA, 2014. Association for Computing Machinery.

[4] Sivert T. Sliper, William Wang, Nikos Nikoleris, Alexander Weddell, and Geoff Merrett. Fused: closed-loop performance and energy simulation of embedded systems. In *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (23/08/20 - 25/08/20)*, page 121, January 2020.

[5] Domenico Balsamo, Oktay Cetinkaya, Alberto Rodriguez Arreola, Samuel Chang Bing Wong, Geoff Merrett, and Alexander Weddell. A control flow for transiently-powered energy harvesting sensor systems. *IEEE Sensors Journal*, April 2020.

[6] Michele Magno, Philipp Mayer, and Luca Benini. A self-sustaining micro-watt programmable smart audio sensor for always-on sensing. In *2018 Ninth International Green and Sustainable Computing Conference (IGSC)*, pages 1–8. IEEE, 2018.

[7] Joel Van Der Woude and Matthew Hicks. Intermittent computation without hardware support or programmer intervention. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 17–32, Savannah, GA, November 2016. USENIX Association.

[8] Domenico Balsamo, Alex Weddell, Geoff Merrett, Bashir Al-Hashimi, Davide Brunelli, and Luca Benini. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *Embedded Systems Letters, IEEE*, 7:15–18, 03 2015.

[9] Kasim Sinan, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. Ink: Reactive kernel for tiny batteryless sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, page 41–53, New York, NY, USA, 2018. Association for Computing Machinery.

[10] Harrison Williams, Xun Jian, and Matthew Hicks. Forget failure: Exploiting sram data remanence for low-overhead intermittent computation. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 69–84, New York, NY, USA, 2020. Association for Computing Machinery.

[11] Adriano Branco, Luca Mottola, Muhammad Hamad Alizai, and Junaid Haroon Siddiqui. Intermittent asynchronous peripheral operations. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, SenSys '19, page 55–67, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Alberto Rodriguez Arreola, Domenico Balsamo, Geoff Merrett, and Alexander Weddell. Restop: retaining external peripheral state in intermittently-powered sensor systems. *Sensors*, 18(1):1–19, January 2018.

[13] Josiah Hester, Nicole Tobias, Amir Rahmati, Lanny Sitanayah, Daniel Holcomb, Kevin Fu, Wayne P. Burleson, and Jacob Sorber. Persistent clocks for batteryless sensing devices. *ACM Trans. Embed. Comput. Syst.*, 15(4), August 2016.

[14] Jasper de Winkel, Carlo Delle Donne, Kasim Sinan Yildirim, Przemysław Pawełczak, and Josiah Hester. Reliable timekeeping for intermittent computing. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 53–67, New York, NY, USA, 2020. Association for Computing Machinery.

[15] Joshua San Miguel, Karthik Ganesan, Mario Badr, and Natalie Enright Jerger. The EH Model: Analytical Exploration of Energy-Harvesting Architectures. *IEEE Computer Architecture Letters*, 17(1):76–79, January 2018.

[16] Kai Geissdoerfer, Mikołaj Chwalisz, and Marco Zimmerling. Shepherd: A portable testbed for the batteryless iot. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, SenSys '19, page 83–95, New York, NY, USA, 2019. Association for Computing Machinery.

[17] A. Colin, G. Harvey, A. P. Sample, and B. Lucia. An energy-aware debugger for intermittently powered systems. *IEEE Micro*, 37(3):116–125, 2017.

[18] Josiah Hester and Jacob Sorber. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, New York, NY, USA, 2017. Association for Computing Machinery.

[19] Joakim Eriksson, Adam Dunkels, Niclas Finne, Fredrik Osterlind, and Thiemo Voigt. Mspsim–an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, volume 118, 2007.

[20] Saad Ahmed, Muhammad Nawaz, Abu Bakar, Naveed Anwar Bhatti, Muhammad Hamad Alizai, JUNAID HAROON SIDDIQUI, and Luca Mottola. Demystifying energy consumption dynamics in transiently-powered computers. *ACM Transactions on Embedded Computing Systems (TECS).*

[21] Matthew Furlong, Josiah Hester, Kevin Storer, and Jacob Sorber. Realistic simulation for tiny batteryless sensors. In *Proceedings of the 4th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*, pages 23–26, 2016.

[22] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011.

[23] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *USENIX Annual Technical Conference, FREENIX Track*, volume 41, page 46, 2005.

[24] M. Walker, S. Bischoff, S. Diestelhorst, G. Merrett, and B. Al-Hashimi. Hardware-validated cpu performance and energy modelling. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 44–53, 2018.

[25] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):994–1007, 2012.

[26] Leander B. Hörmann, Philipp M. Glatz, Karima B. Hein, Christian Steger, and Reinhold Weiss. A hardware/software simulation environment for energy harvesting wireless sensor networks. In *Proceedings of the 9th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, PE-WASUN '12, page 61–68, New York, NY, USA, 2012. Association for Computing Machinery.

[27] Habibu Hussaini, Murtala Adamu, Ajagun Susan, and Gerald Ijemaru. Energy harvesting wireless sensor networks: Design and modeling. *International Journal of Wireless & Mobile Networks*, 6, 11 2014.

[28] Himanshu Sharma, Ahteshamul Haque, and Zainul Jaffery. Modeling and optimisation of a solar energy harvesting system for wireless sensor network nodes. *Journal of Sensor and Actuator Networks*, 7:40, 09 2018.