A Low Voltage RISC-V Heterogeneous System

Boosted SRAMs, Machine Learning, and Fault Injection on VELOUR

Schuyler Eldridge
IBM T. J. Watson Research Center
schuyler.eldridge@ibm.com

Karthik Swaminathan
IBM T. J. Watson Research Center
kvswamin@us.ibm.com

Nandhini Chandramoorthy
IBM T. J. Watson Research Center
nandhini.chandramoorthy@ibm.com

Alper Buyuktosunoglu
IBM T. J. Watson Research Center
alperb@us.ibm.com

Alec Roelke
University of Virginia
ar4jc@virginia.edu

Xinfei Guo
University of Virginia
xg2dt@virginia.edu

Vaibhav Verma
University of Virginia
vv8dn@virginia.edu

Rajiv Joshi
IBM T. J. Watson Research Center
rvjoshi@ibm.us.com

Mircea Stan
University of Virginia
mircea@virginia.edu

Pradip Bose
IBM T. J. Watson Research Center
pbose@us.ibm.com

ABSTRACT

Architectural research traditionally requires a reliance on abstractions, analytical tools, and models. However, the emerging open source hardware community centered around RISC-V, has enabled new, more intrusive approaches to architectural research characterized by: reuse of existing designs, new languages/tools, and workflows. We provide, by means of a case study, details of ongoing work for the DARPA PERFECT project at IBM and the University of Virginia to develop a heterogeneous RISC-V-based system, VELOUR, capable of operating at very low voltage. VELOUR encompasses an open-source RISC-V microprocessor (Rocket) with a tightly coupled machine learning accelerator (DANA), a power/resiliency management unit (PRIME), low voltage SRAMs, and instrumentation using new tools and languages via critical path monitors (CPMs) and power proxy sensors. Evaluation of low voltage operation occurs with a new instrumentation framework for targeted latch-level fault-injection in FPGA (CHIFFRE).

CCS CONCEPTS

•Hardware → Process, voltage and temperature variations; •Computer systems organization → *Neural networks; Processors and memory architectures;*

KEYWORDS

RISC-V, low voltage operation, accelerators

1 INTRODUCTION

Computer architecture is *expected* to be a quantitative, experimental science. The contribution of specific architectural modifications and design choices should be evaluated on a gamut of benchmark workloads. However, traditional approaches towards achieving sufficient evaluation coverage rely on simulators and tools whose efficacy, ease of misuse, and potential for sources of error has been a point of concern [17, 23]. Similarly, the necessary levels of abstraction miss final implementation details when modeling physical effects, such as physical register faults in actual silicon [4].

The natural, though highest-effort solution, is to evaluate architectural modifications on complete *RTL-capable* system designs—hardware descriptions capable of generating a register transfer level (RTL) circuit and feeding an FPGA or ASIC toolchain. While initially infeasible, or requiring efficient multicore-capable hardware-backed emulation infrastructure [20], this tide is beginning to shift. The current inflection point of the emerging open source hardware community centered around RISC-V has dramatically reduced the friction associated with RTL-capable architectural research. This new paradigm is characterized by:

(1) Myriad RISC-V microprocessor designs using different hardware description languages [9, 11, 18, 21]
(2) New hardware description languages [2, 19] that raise the level of hardware abstraction when describing circuits as well as tools [15] that automate architectural transformations[1]
(3) Open source accelerators [6] and frameworks for working with accelerators [10]

Within this landscape, we investigate augmentations to a heterogeneous system to enable low voltage, resilient operation. The base architecture for this exploration consists of a RISC-V Rocket microprocessor [9] and a deep neural network (DNN) accelerator [3, 6]. We then add a second coprocessor/accelerator to Rocket, a *Power and Reliability Integrated Management Entity* (PRIME), that acts as a single unit for sensing and actuation to achieve low voltage operation. Power proxies (power estimated from performance counter-like events) [7, 12] and critical path monitors (CPMs) [7, 22]

[1] These emerging languages and tools are largely orthogonal to high level synthesis (HLS) approaches in their aim in that they raise the level of abstraction in describing reusable circuits as opposed to talking in terms of algorithms—it is the view of the authors that both approaches are viable in different scenarios and should coexist.

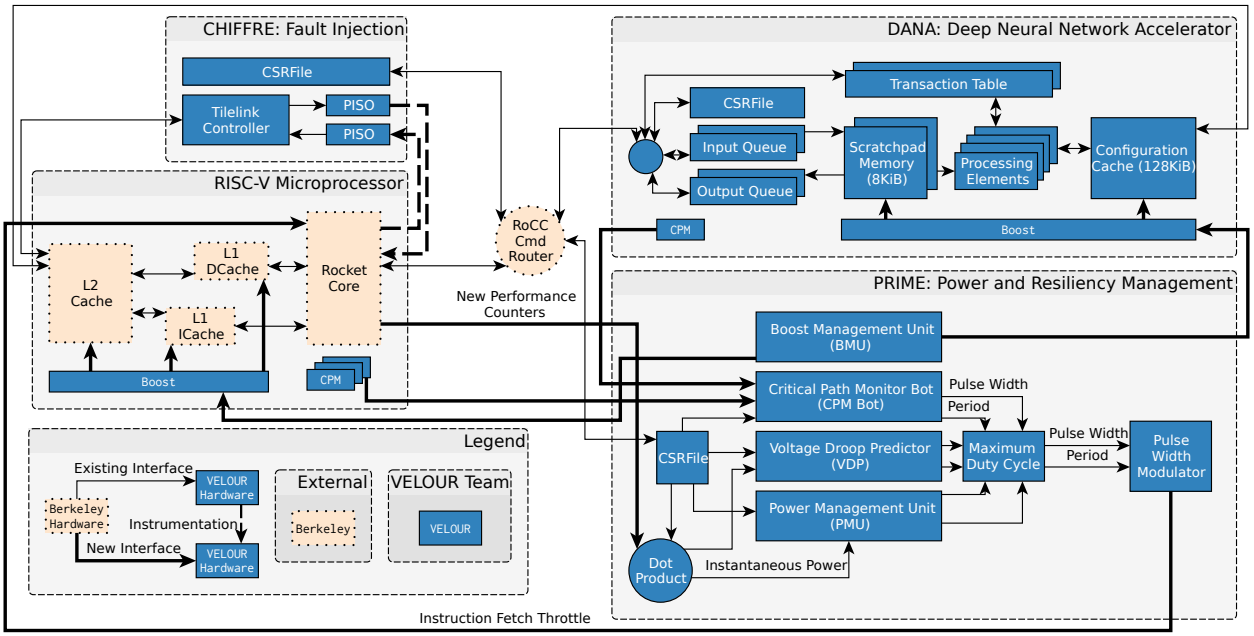

Figure 1: Architecture of the VELOUR design incorporating a RISC-V Rocket microprocessor [9] alongside DANA [3], a deep neural network accelerator, and PRIME, a power/resiliency management unit. Critical Path Monitors and boosted SRAMs are added to the design. Low voltage operation is emulated via RTL-level fault injection using a new instrumentation/injection flow, CHIFFRE.

feed PRIME with real-time data. PRIME then takes steps, via instruction fetch throttling, to prevent incorrect operation resulting from setup time violations.

Pre-silicon estimations of low voltage operation are handled with a new CLEAR-like [4] instrumentation and fault injection flow for Chisel [2] and FIRRTL [15] design environments called CHIFFRE—*Chained Hierarchy Injection for Fault Resiliency Evaluation*. We use CHIFFRE to inject faults representative of failure classes measured from the experimental operation of low voltage SRAMs and registers. CHIFFRE can then be used to identify registers/SRAMs and, when coupled with a CLEAR-like flow, selectively harden only the critical components.

During the workshop presentation, we will demonstrate (via live demo) the VELOUR system running MNIST handwritten digit classification [14] on an FPGA with faults simulated from operating at reduced voltage.

2 ARCHITECTURAL OVERVIEW

Figure 1 shows the architecture of the VELOUR chip. This consists of:

(1) A RISC-V Rocket Microprocessor
(2) A deep neural network accelerator (DANA)
(3) A power and resiliency management unit (PRIME)

Both DANA and PRIME are implemented as Rocket Custom Coprocessors (RoCCs) using the RoCC interface for core to coprocessor communication. All design is done using Chisel3 [8], a domain specific language, embedded in Scala, for describing hardware.

Rocket and DANA are augmented with three circuit and microarchitectur-level features to enable low voltage operation:

(1) Critical Path Monitor (CPM) sensors
(2) *Boosted SRAMs [13] that operate at low voltage nominally but at a higher voltage for reads and writes.* The supply voltage boosting techniques demonstrated by Joshi *et al.* have made it possible to operate at previously un-attainable voltage levels of 0.3V and below. Our proposed chip design uses an implementation of the capacitive coupling-based supply voltage technique proposed in this work. By selectively applying the boost voltage only at the time of a read or write, to the corresponding SRAM array, the overheads of such a technique can be reduced substantially.
(3) New performance counters for "power proxy" events

Instrumentation associated with these augmentations occurs using the *Chisel Annotations API*—a way for specific parts of a design (e.g., modules, ports) to be linked to downstream FIRRTL passes. The existing `BlackBoxSource` annotation class enables seamless integration of blackboxed CPM simulation models with downstream tools. The Boosted SRAM logic is designed using custom ASIC design tools such as Cadence Virtuoso and combined with design compiler-generated memories. This integrated macro is then exported to the ASIC design flow in the form of a `.lib` file during ASIC synthesis and as `.lef` files during the the post-synthesis Place-and-Route flow. Boosted SRAMs do, however, require augmentations of the underlying VELOUR design to connect boost control signals to their controlling unit located in PRIME. FIRRTL, a language for

describing circuits, and the FIRRTL compiler, an LLVM-like circuit compiler that repeatedly transforms circuits using a sequence of provided or user-defined *passes*, provide a natural environment for these augmentations. Boosted SRAMs are connected to PRIME via a FIRRTL pass that adds IO and wires along a path from a Boosted SRAM to PRIME through their lowest common ancestor module.

Comments related to the efficacy of these approaches for performing computer architecture research are discussed in Section 6.

3 DANA: AN SMT DNN ACCELERATOR

DANA, a deep neural network[2] accelerator, provides support for both learning and inference *transactions*. We define a transaction as a request to interact with a specific neural network stored in an operating system-managed physical memory data structure. DANA supports simultaneous multi-threading (SMT) of transactions enabling new accelerator uses, e.g., a foreground inference transaction whose network description is periodically updated by a background learning transaction.

DANA operates directly on a binary data structure describing a neural network and its weighted connections—*a neural network configuration*. We modified this data structure to support 32-bit internal pointers, enabling a theoretical maximum network size of 4GiB. Practically, however, configurations must fit within DANA's on-chip SRAMs (shown in Figure 1 as 128KiB for a forthcoming VELOUR ASIC). DANA's tooling supports network configurations described in the Fast Artificial Neural Network (FANN) library [16] format as well as, via our modifications, a fully-connected subset of Tensorflow [1] with a limited set of activation functions (sigmoid, arctangent, linear).

DANA has been transparently modified to use boosted SRAMs for its Configuration Cache (store of neural network configurations) and scratchpad memories (for storing intermediate computations). Additionally, DANA now includes limited infrastructure for throttling the available pool of multiply-accumulate processing elements (PEs). Three types of PE "governors" are introduced:

Hard Throttling: The available pool of PEs can be decreased by a supervisor.

Cooldown: All PEs are available to be allocated, but only one can allocated every N cycles.

Linear Backoff: Only one PE is available initially. Whenever all PEs are in use for a programmable number of cycles, the available pool of PEs is increased by one.

The use and programming of parameters affecting these governors are processed via supervisor writes to a new Control and Status Register (CSR) file modeled after the design of Rocket's CSR file.

Further discussion of DANA and DANA's SMT transactional model is provided in the original thesis [5].

4 PRIME: VOLTAGE/POWER MANAGEMENT

In order to facilitate robust low voltage operation, we designed PRIME (Power and Reliability Integrated Management Entity) as a Rocket Custom Coprocessor. The standard RoCC interface has been extended with performance counter power proxy connections via new PRIME-specific Chisel traits.

PRIME consists of three sub-units used to monitor power and voltage noise and provides actuation techniques for their mitigation by means of execution throttling. These include three *bots*:[3]

(1) A Power Management Unit (PMU)
(2) A Power history-based Voltage Droop Predictor (VDP)
(3) A Critical Path Monitor (CPM) bot

While VDP and CPM approaches have overlapping coverage, we include both to better understand the benefits and drawbacks of each approach in the final chip.

PRIME additionally acts as the central configuration point for setting specific SRAM boost parameters at run-time through its Boost Management Unit (BMU).

4.1 Throttling/relaxation mechanisms for power and noise management

The PMU, VDP, and CPM bot all work together to determine if any transient situation in the Rocket core or any of its accelerators will cause incorrect operation. These three units all output a throttling duty cycle that can throttle the instruction fetch rate of Rocket to reduce power and voltage noise.

The PMU estimates the processor power at a cycle-level granularity. The power is computed as a linear sum of individual events over a programmable time window. Each event is a user-defined binary signal driven by combinational or sequential logic added to the Rocket core, i.e., additional performance-counter like signals. Each event is assigned a power estimate using Cadence Joules on a placed-and-routed Rocket core running microbenchmarks crafted to repeatedly execute the specific event. We use a the state-of-the-art 14nm Global Foundries LPP process. When the windowed power estimate computed by the PMU exceeds certain programmable thresholds, the PMU generates a throttle signal. For instance, in case of a power emergency where the operating power exceeds the highest threshold, the PMU indicates that the throttling should be set to the highest level, possibly stalling every alternate cycle, or even stopping execution altogether for a short period of time.

The VDP estimates differential power (ΔP) as a proxy for voltage noise—a large rise in power over a short period of time intuitively corresponds to a large $^{dI}/_{dt}$ droop. It estimates the power difference across successively increasing epoch granularities and indicates if one or more of these computed ΔP values crosses a pre-specified threshold. Each subsequent threshold, like with the PMU, corresponds to increasingly more aggressive throttling.

The CPM bot determines the likelihood of a timing error due to $^{dI}/_{dt}$ droop, by tracking the available slack on the critical paths during execution. The VELOUR chip contains several CPMs distributed across various parts of the chip. The outputs from these CPMs are in the form of a thermometer code indicating the amount of measured available slack in each critical path. When any of these thermometer codes violates run-time programmable thresholds the core is determined to be undergoing a CPM emergency and to have no available slack. Instruction fetch is immediately stopped and brought back up slowly.

[2]The authors clarify that "deep" means neural networks with all neurons in one layer connected to all neurons in the next (fully-connected) of any depth (number of layers) and any number of neurons per layer.

[3]State machines that determine a throttling level

The output of any of these bots is the duty cycle for which the core should be throttled (specified as a `[pulse width, period]` tuple). The maximum duty cycle is computed and sent to a pulse width modulation (PWM) unit that actuates the requested worst case measured throttling of all the bots in PRIME.

After throttling for a specified epoch, it is essential that the execution is not immediately restored to its original rate. A sudden increase in activity, due to changing state from stopped to full bore, can cause a significant $^{dI}/_{dt}$ droop (which is exactly what PRIME is designed to avoid!). Hence, full bore execution is restored gradually by separate state machines inside each of PMU, VDP, and CPM bots. The current implementation uses both linear and exponential reductions in throttle duty cycle.

The decoupling of the bots duty cycle generation and the actual throttling by the PWM unit enables the addition of new bots to meet the needs of the designer. New bot development is further facilitated by the object-oriented features of the Chisel (and Scala) languages.

5 CHIFFRE: INSTRUMENTATION/INJECTION

VELOUR, being a demonstration chip for low voltage techniques, requires a framework for performing automated and what-if analyses related to low voltage operation before empirical measurement using an actual ASIC. Unfortunately, efficient abstractions involving pre-RTL architectural level fault injection have been shown to be inaccurate for resiliency analysis—RTL-level fault injection is required for accurate results [4].

We are currently developing a framework for RTL-level fault injection within the RISC-V Rocket/Chisel/FIRRTL environment. This framework, CHIFFRE, follows a two-step process of circuit *instrumentation* and *injection*.

First, an RTL-level design is augmented with fault injection capabilities. This consists of four specific additions to any module tagged for instrumentation:

- **Controller Addition:** A programmable injection unit controller capable of injecting faults during specific events (e.g., a particular cycle count after being enabled) is added to all tagged modules.
- **Combinational Logic:** Combinational logic is added after every register to allow the use of correct or faulty values (as will be determined by the injection unit controller).
- **Reference Replacement:** References to all registers are replaced with those of the output of the added combinational logic.
- **IO Additions:** IO is added to every module to allow the injection unit controller to be programmed at run-time. While this can be a network, we opt for a scan chain due to its simplicity.

A global controller is then added to design in the form of a CHIFFRE RoCC unit.

Second, based on hierarchy information gathered from the instrumentation procedure, a small configuration compiler is used to generate a binary bitstream that describes the types and locations of faults to be injected. This bitstream can then be included in a RISC-V program and used to configure all fault injection logic before a program is run.

The instrumentation framework currently uses `Verilog::Perl` operating directly on FIRRTL-emitted Verilog, but is in the process of being moved over to FIRRTL. The eventual flow is shown in Figure 2. Instrumentation steps are provided entirely by added FIRRTL passes[4]

6 COMMENTS ON RISC-V ECOSYSTEM

Figure 2 shows the ultimate, complete VELOUR flow targeting an FPGA evaluation platform. Notably, this flow requires minimal modifications to the Berkeley-specific RISC-V ecosystem.

Berkeley-driven designs (Rocket) and tooling (Chisel/FIRRTL) provide, generally, sufficient infrastructure for conducting RISC-V-based architectural experimentation. Rocket, by providing the RoCC socket (though with intermittent support in upstream repositories), enables the direct addition of accelerators of both tightly coupled (communicated with via direct register reads and writes) and loosely coupled (communicated with via memory interfaces) varieties.[5] Further accelerator decoupling can happen via direct connection to the TileLink cache-coherency interface outside of the core or as an actual device via an interface of the designer's choosing.

Chisel provides a similar substrate for building highly parameterized, reusable hardware designs not currently possible with Verilog, SystemVerilog, or VHDL. The benefits come from the fact that Chisel (and FIRRTL) are restricted languages for describing circuits. This is a stark contrast to other HDLs (Verilog) that are languages for describing parallel events and their relationships. Relatedly, the adoption of FIRRTL brings an LLVM-like approach to architectural research in the sense that direct circuit transformations can be tested empirically.

The learning curve of Chisel (specifically Scala) does prove to be a stumbling block for both broader architectural research using Rocket (as well as broader industry adoption of Rocket). Nevertheless, this is an orthogonal point as *the authors fully realize that RISC-V is not Rocket and bears no connection with Chisel or FIRRTL.* This distinction is not fully understood by the broader community and makes for a point of confusion about what is RISC-V (an ISA specification) and what is not RISC-V (implementations of the RISC-V ISA and tooling that supports development of RISC-V implementations).

7 CONCLUSION

We provide an overview of a forthcoming design, VELOUR, that uses the Berkeley RISC-V and hardware development ecosystem as a substrate for low voltage architectural experiments. VELOUR incorporates a machine learning accelerator, DANA, alongside a RISC-V Rocket microprocessor. Low voltage features are added in the form of power proxy and CPM-derived throttling in a newly designed coprocessor, PRIME, as well as throttling augmentations to DANA. A new framework, CHIFFRE, for pre-ASIC low-voltage experimentation is discussed.

[4] A FIRRTL pass takes a FIRRTL circuit as input and returns a circuit (usually with some modifications).
[5] BOOM [21] does not, at present support RoCC.

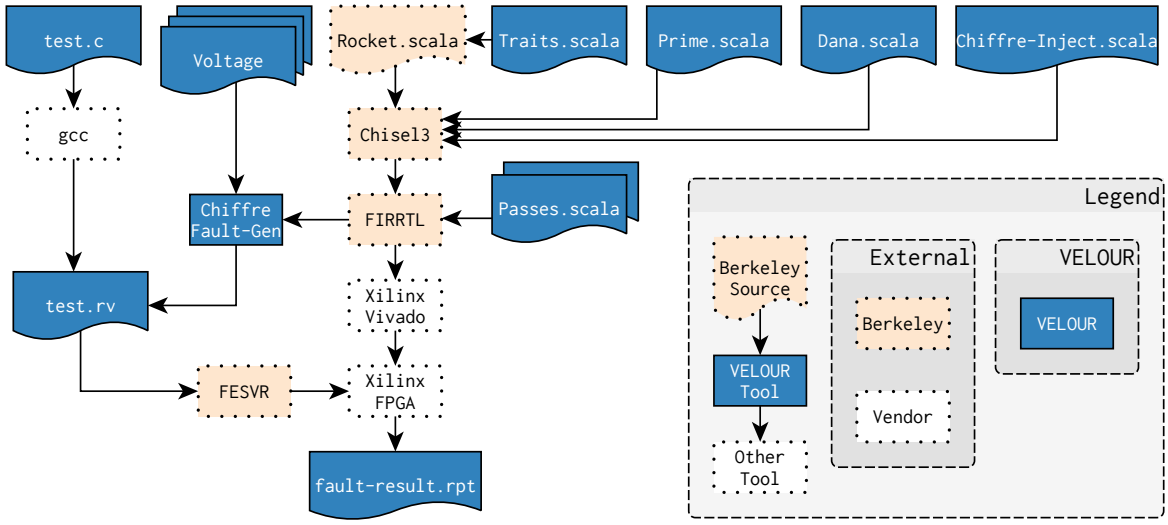

Figure 2: Overview of the VELOUR chip and its associated toolflow

All ongoing DANA development is done on the existing open source repository while both PRIME as well as the CHIFFRE framework are expected to be open sourced in Fall 2017.

ACKNOWLEDGMENTS

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. This document is Approved for Public Release, Distribution Unlimited.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović. 2012. Chisel: Constructing hardware in a Scala embedded language. In *49th ACM/EDAC/IEEE Design Automation Conference (DAC), 2012*. 1212–1221. https://doi.org/10.1145/2228360.2228584

[3] Boston University Integrated Circuits and Systems Group. 2017. Dynamically Allocated Neural Network (DANA) RISC-V RoCC Accelerator GitHub Repository. Online: https://github.com/bu-icsg/dana. (2017).

[4] E. Cheng, S. Mirkhani, L. G. Szafaryn, C. Y. Cher, H. Cho, K. Skadron, M. R. Stan, K. Lilja, J. A. Abraham, P. Bose, and S. Mitra. 2016. CLEAR: Cross-layer exploration for architecting resilience: Combining hardware and software techniques to tolerate soft errors in processor cores. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/2897937.2897996

[5] Schuyler Eldridge. 2016. *Neural Network Computing Using On-Chip Accelerators*. Ph.D. Dissertation. Boston University.

[6] Schuyler Eldridge, Amos Waterland, Margo Seltzer, Jonathan Appavoo, and Ajay Joshi. 2015. Towards General-Purpose Neural Network Computing. In *2015 International Conference on Parallel Architecture and Compilation, PACT 2015, San Francisco, CA, USA, October 18-21, 2015*. 99–112. https://doi.org/10.1109/PACT.2015.21

[7] Michael Floyd, Malcolm Ware, Karthick Rajamani, Tilman Gloekler, Bishop Brock, Pradip Bose, Alper Buyuktosunoglu, Juan C Rubio, Birgit Schubert, Bruno Spruth, et al. 2011. Adaptive energy-management features of the IBM POWER7 chip. *IBM Journal of Research and Development* 55, 3 (2011), 8–1.

[8] Freechips Project. 2017. Chisel. Online: https://github.com/freechipsproject/chisel3. (2017).

[9] Freechips Project. 2017. Rocket Chip GitHub Repository. Online: https://github.com/freechipsproject/rocket-chip. (2017).

[10] Bespoke Silicon Group. 2017. rocc-template. Onlline https://bitbucket.org/taylor-bsg/bsg_riscv_rocc. (2017).

[11] MIT CSAIL's Computation Structures Group. 2017. Riscy Processors - Open-Sourced RISC-V Processors. Online: https://github.com/csail-csg/riscy. (2017).

[12] Wei Huang, Charles Lefurgy, William Kuk, Alper Buyuktosunoglu, Michael Floyd, Karthick Rajamani, Malcolm Allen-Ware, and Bishop Brock. 2012. Accurate fine-grained processor power proxies. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 224–234.

[13] R. V. Joshi, M. M. Ziegler, and H. Wetter. 2017. A Low Voltage SRAM Using Resonant Supply Boosting. *IEEE Journal of Solid-State Circuits* 52, 3 (March 2017), 634–644. https://doi.org/10.1109/JSSC.2016.2628772

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (Nov 1998), 2278–2324. https://doi.org/10.1109/5.726791

[15] Patrick S. Li, Adam M. Izraelevitz, and Jonathan Bachrach. 2016. *Specification for the FIRRTL Language*. Technical Report UCB/EECS-2016-9. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-9.html

[16] Steffen Nissen. 2003. *Implementation of a fast artificial neural network library (fann)*. Technical Report. Department of Computer Science University of Copenhagen (DIKU). http://fann.sf.net.

[17] Tony Nowatzki, Jaikrishnan Menon, Chen han Ho, and Karthikeyan Sankaralingam. 2014. gem5, GPGPUSim, McPAT, GPUWattch, "Your favorite simulator here" Considered Harmful. (2014).

[18] Pulp Platform Project. 2017. Pulpino. Online: https://github.com/pulp-platform/pulpino. (2017).

[19] SpinalHDL. 2017. SpinalHDL. Online: https://github.com/SpinalHDL/SpinalHDL. (2017).

[20] Z. Tan, A. Waterman, R. Avizienis, Y. Lee, H. Cook, D. Patterson, and K. Asanovic. 2010. RAMP gold: An FPGA-based architecture simulator for multiprocessors. In *Design Automation Conference*. 463–468.

[21] UC Berkeley Architecture Research Group. 2017. Berkeley Out-of-Order Machine. Online: https://github.com/ucb-bar/rocket-chip. (2017).

[22] Tobias Webel, PM Lobo, Ramon Bertran, GM Salem, Malcolm Allen-Ware, R Rizzolo, Sean M Carey, Thomas Strach, Alper Buyuktosunoglu, Charles Lefurgy, et al. 2015. Robust power management in the ibm z13. *IBM Journal of Research and Development* 59, 4/5 (2015), 16–1.

[23] S. L. Xi, H. Jacobson, P. Bose, G. Y. Wei, and D. Brooks. 2015. Quantifying sources of error in McPAT and potential impacts on architectural studies. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. 577–589. https://doi.org/10.1109/HPCA.2015.7056064