Automatic Activity-based Power Modeling for Arbitrary RTL

Donggyu Kim, Vighnesh Iyer
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
{dgkim, vighnesh.iyer}@berkeley.edu

***Abstract*—This paper presents a novel methodology to automatically construct power models for any RTL designs, which accurately and quickly reports the power estimation during performance simulation on the FPGA. First, important signals that are most sensitive to power dissipation are selected by analyzing the circuit graph of the target RTL design. Next, design-specific activity-based power models are automatically and accurately constructed and trained with the selected signals, the RTL signal activities, and the power estimates from commercial CAD tools. In addition, the FPGA performance simulators automatically generated from RTL designs are automatically instrumented with signal activity counters. Power dissipation is readily predicted during performance simulation on the FPGA by reading out the activity counter values. Finally, the power models are validated by replaying random RTL state sample snapshots on detailed gate-level simulation, which provides accurate power estimates with statistically bounded errors. Automatic power model construction is demonstrated with simple designs, which can be also used for more complex designs including RocketChip, Hwacha, and BOOM.**

I. Introduction

Power dissipation has been a major design constraint not only for embedded and portable systems but also for high-end servers and datacenters. A plethora of design strategies have been suggested to meet the power constraint while achieving maximum performance. Thus, power estimation methodologies are very important for the evaluation and the validation of novel hardware designs. However, accurate and fast application-specific power estimation for complex hardware design is still a big challenge for system designers and application developers, which will play a more crucial role in the near future as Moore's law is near the end.

Microarchitecture-level power analysis tools calibrated against representative hardware designs are widely used by computer architects [1], [2], [3], [4], [5]. These power models need activity statistics driven from microarchitectural cycle-level software simulators [6], [7], [8]. This approach helps designers study trace-offs with design parameters in early design phase without developing RTL designs. However, this methodology is limited to designs that are similar to what the abstract model is build upon, and requires long simulation times to collect microarchitectural activities from microarchitectural software simulators. Moreover, it is very hard to apply this methodology to non-traditional hardware designs such as application-specific accelerators as their power models should be validated against RTL designs or existing systems.

Once RTL implementations are available, energy efficiency as well as cycle time and area can be evaluated using commercial CAD tools. The existing CAD tools provide very accurate performance and power estimates. However, their simulation time is extremely slow, preventing design space exporation for realistic applications running on complex hardware designs.

This paper describes an automatic activity-based power modeling methodology for arbitrary RTL, which enables fast and accurate power estimation for any RTL designs. First, signals, which determines the cycle behaviors of the target design, are automatically selected by analyzing the circuit graph. Next, a design-specific activity power model is constructed and trained in terms of the selected signal activities. In addition, the target RTL design is automatically instrumented so that the signal activities can be probed during performance simulation on the FPGA. Finally, power dissipation is estimated in real time by pausing the FPGA performance simulation and reading out the signal activities from the FPGA.

II. Related Work

Analytic power modeling [1], [2], [3], [4], [5] combined with microarchitectural cycle-level software simulators [6], [7], [8] is widely used for computer architecture research. This enables early microarchitecture-level design space exploration without necessitating RTL development. However, microarchitectural software simulators run much slower than real systems, making full execution of applications infeasible. Moreover, for different microarchitectures with new technologies, a new analytic power model is constructed and strictly validated against real systems or detailed circuit / gate-level simulations, which is very difficult for non-traditional hardware designs.

Power modeling based on performance-monitoring counters is also popular for power estimation [9], [10], [11], [12], [13], [14]. This method provides a quick power estimate by profiling full execution of applications. However, its application is also limited to well-known microarchitectures. Moreover, it is only practical with existing existing physical systems since standard microarchitectural software simulators are extremely slow.

There are various attempts to build macro models in the boundary of latches [15], [16], [17], [18]. This methodology is less expensive than what the CAD tools do since the power dissipation is expressed in terms of the input and output signals of the macro blocks. However, this is still very expensive for

the FPGA simulation because there are a ton of macro blocks in complex hardware designs.

There are significant efforts to accelerate power estimation using an FPGA [19], [20], [21], [22], [23], [24], [25]. Coburn et al. [19] implement detailed power models directly on the FPGA, which suffers from large FPGA resource overhead. Ghodrat et al. [20] expands the ideas of Coburn et al. [19] by translating part of power computation logic on the FPGA to software to reduce FPGA resource overhead, but introduces communication overhead between FPGA and software. These methods are inherently lack of scalability due to large overhead of power computation logic.

On the other hand, Atienza et al. [21] manually attached activity monitoring units to the target design on the FPGA. Bhattacharjee et al. [22] manually implement event counters on top of RTL designs to estimate power dissipation from FPGA emulation. PrEsto [23] generates power models from signals manually selected by users. In addition, This technique also requires additional manual efforts to instrument existing FPGA simulators with power models. These methods have less FPGA resource overhead, but require significant manual efforts as well as designers' intuition.

Yang et al. [24] proposes to construct cycle-by-cycle power models by pruning probed signals with machine learning without requiring designers' intuition. This methodology is built upon the assumption that power dissipation is a function of some primary registers' toggles. However, it lacks of the justification that the truncated SVD properly selects the primary registers for complex hardware designs. Also, it instruments the RTL designs with the power computing logic as in PrEsto [23], which may be automated by commercial CAD tools.

Strober [25] automatically transforms RTL designs into FPGA performance simulators and also automatically instrument the simulators with scan chains to capture RTL state snapshots. Then, the RTL state snapshots are randomly sampled from the FPGA simulation and replayed on detailed gate-level simulation to compute the average power. Although, Strober quickly provides accurate power estimates with statistically bounded errors, the average power should be computed after the FPGA simulation finishes, preventing online power computation, which is the motivation of this paper.

III. Methodology

There are five major steps in our methodology(Figure 1). First, the *signal selection* step walks the circuit graph in FIRRTL[26] and picks important signal candidates according to the circuit topology. Next, the *power model construct and training* step builds a power model using selected signals and trains the model using preliminary waveforms from small workloads. The *RTL instrumentation* step automatically instrument the FPGA performance simulators generated by Strober [25] with the signal activity counters. The *power prediction* step computes the power dissipation in real time during FPGA performance simulation. Finally, the *power validation* step validates the power models by comparing against the power estimation from Strober, and further trains the power models if necessary.

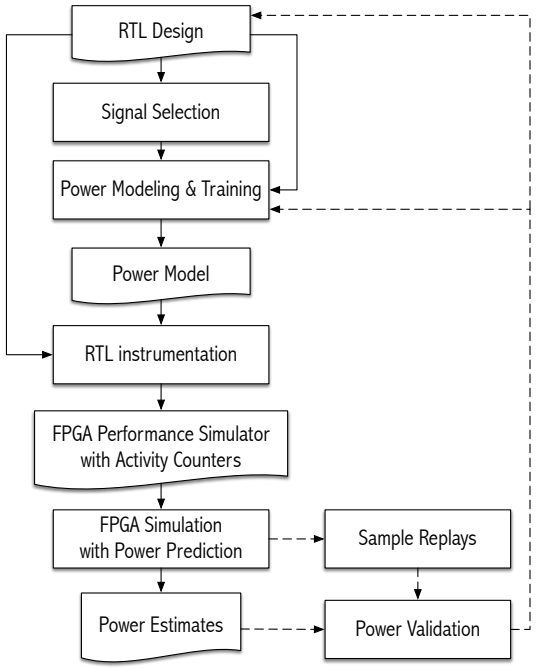

Fig. 1. Overall Tool Flow

A. Signal Selection

To construct and train a power model, we should select a subset of signals because it is infeasible to keep track of all signal activities on the FPGA.

There are several observations to select power-sensitive signals. First of all, *the signal activities of the whole design are driven by the inputs and the state of the design.* Thus, we may be able to build an accurate power model in terms of the register and input toggles. However, there are a large number of registers in complex hardware designs, and thus, this is still infeasible for the FPGA.

Fortunately, it seems that *there are only a small number of signal activities or/and high-level events that drive the state transitions of the design*. Otherwise, all power estimation methods based on event counters [9], [10], [11], [12], [13], [14] do not make sense. Therefore, we may be able to construct an accurate power model by using only a small number of signals that approximate high-level performance / power events.

Figure 2 describes how power sensitive signals are selected. First of all, a random RTL design is fed into the FIRRTL compiler. In this paper, we assume this design is written with Chisel [27]. However, the target designs are not necessarily developed with Chisel because the FIRRTL compiler will support various languages including Verilog in the near future.

In the FIRRTL compiler, a circuit graph is expressed with an intermediate representation, FIRRTL [26], so that a designer can write custom compiler passes for their design. A signal

analysis pass is implemented, which examines all write enable signals for registers. For SRAMs, their write enable signals and read data busses are selected. The selected signals and busses are dumped to the file used for power model training in Section III-B.

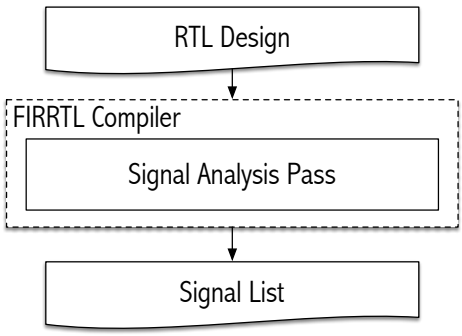

Fig. 2. Signal Selection

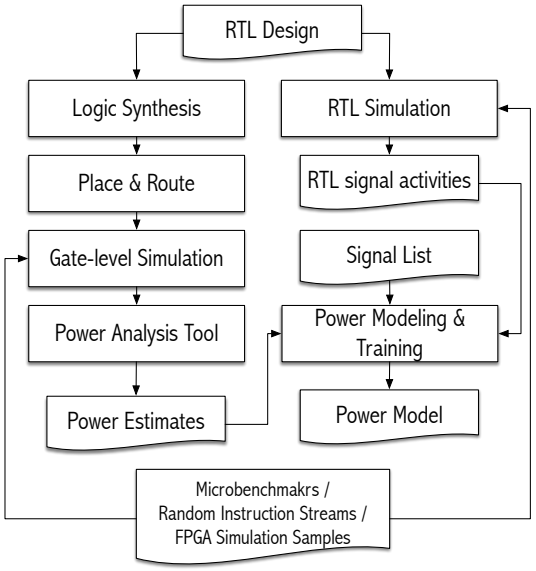

Fig. 3. Power Modeling and Training

B. Power Modeling and Training

Once the signals and busses are selected from the previous step, a design-specific activity-based power model can be constructed. Figure 4 explains how power models are built and trained. First, an RTL design is fed into the logic synthesis tool (e.g. Synopsys Design Compiler ®) and the place-and-route tool (e.g. Synopsys IC Compiler ®) to obtain a gate-level design, which is simulated in SDF back-annotated gate-level simulation (e.g. Synopsys VCS ®) for accurate power estimates by the power analysis tool(e.g. Synopsys PrimeTime PX ®).

RTL signal activities are also computed from RTL simulation(e.g. Verilator, Synopsys VCS ®). By providing the signal list, the RTL signal activities, and the detailed power estimates to the power modeling and training algorithm, a design-specific power model can be constructed expressed with the signal toggle activities.

For RTL simulation and gate-level simulation, microbenchmarks or random instruction streams are employed for initial power model training. In addition, there can be random execution sample snapshots from long-running FPGA performance simulation [25], which are used not only for power model validation but also for further training the power model for more accurate power estimates.

1) Macromodels for Complex Combinational Logic: Previous work [23] suggests that linear power models for complex combinational logic such as ALUs, SEC/DED circuits, interrupt controllers, and multipliers are inaccurate. To overcome this issue, we turn to macromodels which attempt to estimate the power of complex logic by only looking at the input and output switching activity of the combinational block. In this paper we will measure the accuracy of a previously proposed power macromodel that uses 4 statistics $P_{in}, D_{in}, SC_{in}, D_{out}$. [15][16]

This macromodel considers sequences of input vectors into a combinational block. Each sequence may contain N input vectors. Then we can define for a given sequence,

- P_{in} = average number of input bits that are logic 1 across a sequence
- D_{in} = average number of input bit transitions across a sequence (average input transition density)
- SC_{in} = average spacial correlation coefficient across a sequence and across all input bit pairs
- D_{out} = average number of output bit transitions across a sequence (average output transition density)

In the context of power estimation for a large digital design: if a complex combinational block is detected in the design, the power macromodel will be used in place of a linear model. To feed the macromodel with data, the combinational block's inputs and outputs will be instrumented. A small logic block on the FPGA will map the raw input and output signals into a 4-tuple that represents the above statistics, and the tuple will be fed out of the FPGA emulator to be plugged into the power macromodel.

C. RTL Instrumentation

The RTL instrumentation is built upon the Chisel3 and FIRRTL port of the Strober's flow to generate FPGA performance simulators. Figure 4 shows how the FPGA performance simulators are instrumented with the signal activity counters.

First, the target RTL design is FAME1-transformed to create the FPGA performance simulator [25] in the FIRRTL compiler. Then, a custom transform is executed to attach activity counters to the FPGA performance simulator by using the signal list from Section III-A. In addition, scan chains are inserted not only to read out the activity counter values but also to capture RTL state snapshots to validate the power

model. Finally, simulation mapping and platform mapping are conducted for FPGA simulation [25].

Note that signal selection in Section III-A can be integrated to this flow if signals are not pruned during power modeling and training in Section III-B. In this case, FPGA simulation can run in parallel while training power models, which uses slow CAD tools extensively.

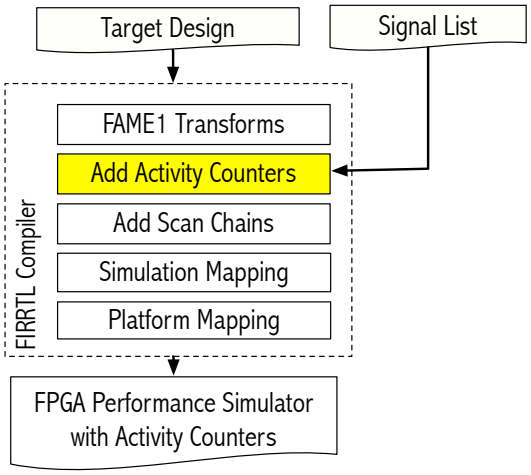

Fig. 4. Power Modeling and Training

D. Power Prediction during FPGA Simulation

Once the FPGA performance simulator instrumented with activity counters is obtained, power estimates are quickly available during FPGA simulation. Since the target RTL is FAME1-transformed, FPGA simulation can be easily paused, and activity counter values are read out from the FPGA in the same way random RTL state sample snapnots are taken in Strober [25]. If the power model is not available during FPGA simulation because it is still trained, the activity counter values are saved to estimate the power dissipation later.

E. Power Validation

Designers may want to validate the power estimates from Section III-D after FPGA simulation. Since this methodology is combined with Strober, an accurate power estimates with statistically bounded errors are available by replaying random RTL state snapshots, which are used to further train the power model for more accurate power estimates (Figure 4).

IV. Experimental Setup and Results

A. Power Macromodel for Combinational Logic

To validate the accuracy of the power macromodel proposed in [16], we will generate multiple sequences of input training vectors, derive an accurate power estimate from PrimeTime, and use the statistics $(P_{in}, D_{in}, SC_{in}, D_{out})$ associated with each sequence to train the macromodel. We will then generate a separate set of sequences which will be used to test the macromodel's accuracy as compared to the PrimeTime power estimate.

The combinational circuits used in this study were the following ISCAS 85 benchmark circuits:

- *c17*: 6 NAND gates
- *c432*: 27-channel interrupt controller
- *c499*: 32-bit SEC
- *c880*: 8-bit ALU
- *c1908*: 16-bit SEC/DED
- *c2670*: 12-bit ALU
- *c6288*: 16x16 multiplier
- *c7552*: 32-bit adder/comparator

The training and testing datasets consisted of 300 sequences of 30 vectors. They were run through the DUT using RTL simulation and post-PAR gate-level simulation. The gate-level simulations were used to derive switching activity files which when passed to PrimeTime PX gave a power estimate for a given sequence. Using the input and output vectors, and the power estimate from PrimeTime the 4 following power macromodels were constructed:

- *4D Table*: A simple map from a $P_{in}, D_{in}, SC_{in}, D_{out}$ vector to a power estimate.
- *Linear Model*: Coefficients were derived for an equation of the form $c_0 + c_1 P_{in} + c_2 D_{in} + c_3 SC_{in} + c_4 D_{out}$ using least squares regression. (5 coefficients)
- *Quadratic Model*: Coefficients were derived for an equation consisting of the linear terms, but also cross terms and self-squared terms using least squares regression. (15 coefficients)
- *Cubic Model*: Added triplet cross terms and self-cubed terms. (35 coefficients)

Once the models were constructed and trained using the training dataset, they were used to make predictions on the testing dataset.

4D table prediction involves interpolating new $(P_{in}, D_{in}, SC_{in}, D_{out})$ vectors on the training dataset: if interpolation wasn't possible due to the testing vector being outside the convex hull of the training vectors, then the nearest neighbor's power estimate was used. Prediction on the linear, quadratic, and cubic models was performed with simple matrix multiplication of the testing vectors and the previously calculated coefficients.

Errors in the the macromodel were measured and are summarized in Figure 5 and Figure 6.

B. Sequential Circuit Power Modeling

To demonstrate the capability of the automatic signal section methodology for sequential logic, we evaluate our methodology using simple Chisel designs described as follows:

- *GCD*: hardware implementation of Euclidean algorithm
- *Parity*: computation for the parity of inputs
- *Stack*: hardware implementation of stack
- *Risc*: very simple single-cycle pipeline
- *RiscSRAM*: very simple multi-cycle pipeline with SRAMs

We run 20 sets of simulation to train the power models and use different 20 sets of simulation to test the power models. Quadratic models are used for *GCD* and *Parity*, and cubic

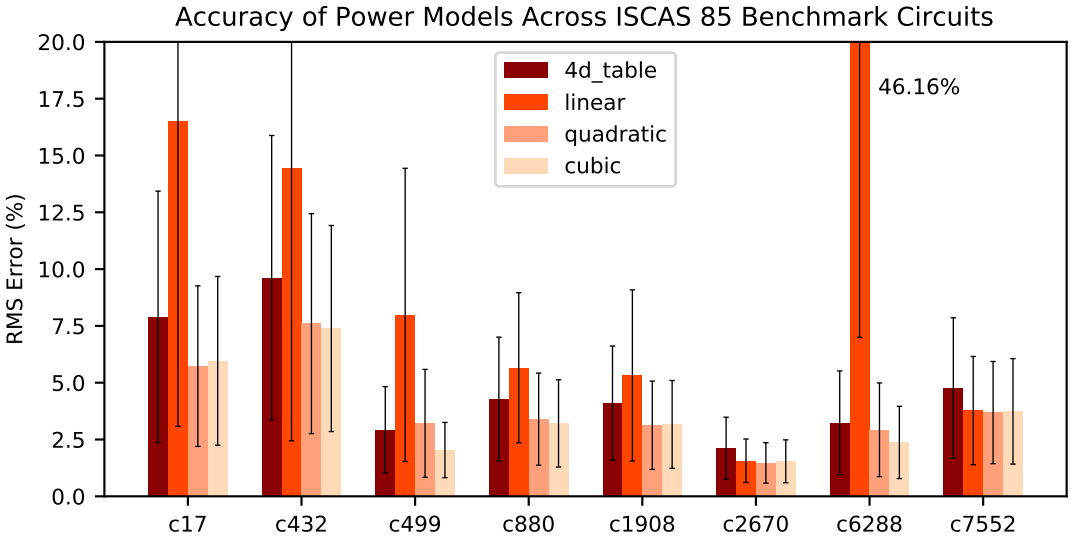

Fig. 5. RMS errors from macromodel power estimation, error bars indicate 1 SD

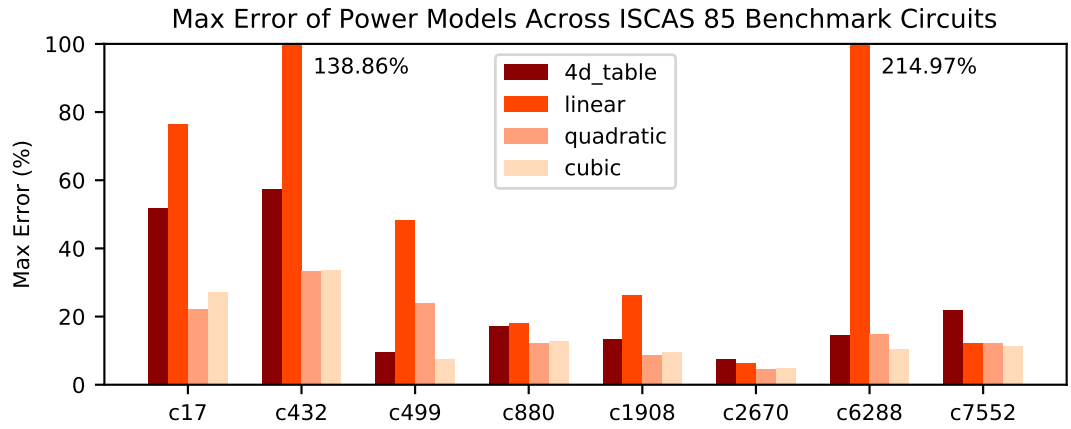

Fig. 6. Max errors from macromodel power estimation, worst case accuracy

models are used for *Stack*, *Risc*, and *RiscSRAM*. Table I shows the power model errors for the designs. The maximum error of *Stack* is somewhat warrisome, but we may address it by having more training sets. Impressively, this methodology works well for more complex designs like *Risc* and *RiscSRAM*.

To demonstrate how well signal selection works, we use a more complex design, RISCV-mini [28]. RISCV-mini implements RV32I of the User-level ISA Version 2.0 [29] and the Machine-level ISA of the Privileged Architecture Version 1.7 [30]. RISCV-mini includes a 3-stage pipeline as well as instruction and data caches as shown in Figure 7, which is neither too simple nor too complex. From RISCV-mini, 108 signals are selected. However, about 50 % of theses signals are due to lack of inter-module analysis, which will be cut out from pass optimization.

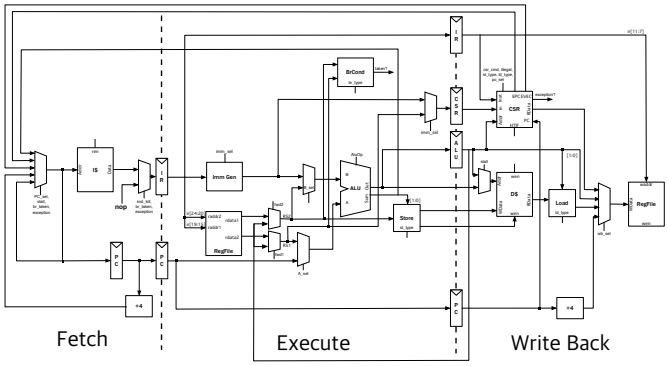

Fig. 7. RISCV-mini Pipeline

V. Conclusion

In this report, a novel approach to accurately and quickly predict cycle-level power estimates for arbitrary RTL was

Design	GCD	Parity	Stack	Risc	RiscSRAM
Number of Selected Signals and Busses	7	3	8	11	15
Normalized RMS error(%)	29.3	1.9	31.9	4.0	10.0
Average error(%)	5.1	0.9	26.9	2.3	6.5
Maximum error(%)	16.5	2.4	153.7	7.4	20.5

TABLE I
EVALUATION WITH EXAMPLES

presented. Import signals sensitive to power dissipation were automatically selected using a custom analysis pass in the FIRRTL compiler. Also, power models were built and trained using the signal list, the RTL signal activities, and the accurate power estimates from detailed gate-level simulation. The RTL instrumentation was done on top of the Strober's flow for the FPGA performance simulators to automatically add activity counters using the signal list. During FPGA simulation, online power prediction was readily obtained by pausing the simulation and reading out the activity counters. Finally, the power model was validated and further trained by replaying random RTL state snapshots.

In this project, we evaluated automatic signal selection and power modeling with simple designs. In the future, we will complete the FPGA instrumentation, power prediction, and power validation flows and eventually evaluate this methodology using more complex designs like Rocket-chip, Hwacha, and BOOM.

One possible application of this methodology is fast DVFS modeling using FPGA simulation. To enable DVFS in simulation, online power prediction must be available for the power management unit model to control voltage and frequency. By using this methodology, it is expected we will simulate DVFS with FPGA performance simulators in the near future.

REFERENCES

[1] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *ISCA*, 2000.

[2] N. Vijaykrishnan, M. Kandemir, M. J. Irwin, H. S. Kim, and W. Ye, "Energy-driven integrated hardware-software optimizations using SimplePower," in *ISCA*, 2000.

[3] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009.

[4] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "GPUWattch: enabling energy optimizations in GPGPUs," in *ISCA*, 2013.

[5] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, "Aladdin: A pre-RTL, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *ISCA*, 2014.

[6] N. Binkert, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, D. A. Wood, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, and T. Krishna, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, Aug 2011.

[7] A. Patel, F. Afram, and S. Chen, "MARSSx86: A full system simulator for x86 CPUs," in *DAC*, 2011.

[8] T. F. T. Wenisch, R. R. E. R. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. J. Hoe, "SimFlex: Statistical Sampling of Computer System Simulation," *IEEE Micro*, vol. 26, pp. 18–31, Jul 2006.

[9] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *The 9th ACM SIGOPS European workshop*, 2000.

[10] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *SIGMETRICS*, 2003.

[11] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *MICRO*, 2003.

[12] W. Bircher, M. Valluri, J. Law, and L. John, "Runtime identification of microprocessor energy saving opportunities," in *ISLPED*, 2005.

[13] W. L. Bircher and L. K. John, "Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events," in *ISPASS*, 2007.

[14] R. Bertran, M. Gonzelez, X. Martorell, N. Navarro, and E. Ayguade, "A Systematic Methodology to Generate Decomposable and Responsive Power Models for CMPs," *IEEE Transactions on Computers*, vol. 62, pp. 1289–1302, Jul 2013.

[15] F. Najm and S. Gupta, "Power Modeling for High-Level Power Estimation," in *VLSI*, 2000.

[16] F. Najm and S. Gupta, "Analytical models for RTL Power Estimation of Combinational and Sequential Circuits," in *CADICS*, 2000.

[17] Q. Wu, Q. Qiu, M. Pedram, and C. S. Ding, "Cycle-accurate macro-models for RT-level power analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 4, pp. 520–528, 1998.

[18] A. Bogliolo, L. Benini, and G. De Micheli, "Regression-based RTL power modeling," *TODAES*, vol. 5, pp. 337–372, jul 2000.

[19] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: A new paradigm for power estimation," in *DAC*, 2005.

[20] M. A. M. Ghodrat, K. Lahiri, and A. Raghunathan, "Accelerating system-on-chip power analysis using hybrid power estimation," in *DAC*, 2007.

[21] D. Atienza, P. G. Del Valle, G. Paci, F. Poletti, L. Benini, G. De Micheli, and J. M. Mendias, "A fast HW/SW FPGA-based thermal emulation framework for multi-processor system-on-chip," in *DAC*, 2006.

[22] A. Bhattacharjee, G. Contreras, and M. Martonosi, "Full-system chip multiprocessor power evaluations using FPGA-based emulation," in *ISLPED*, 2008.

[23] D. Sunwoo, G. Y. Wu, N. a. Patil, and D. Chiou, "PrEsto: An FPGA-accelerated Power Estimation Methodology for Complex Systems," in *FPGA*, 2010.

[24] J. Yang, L. Ma, K. Zhao, Y. Cai, and T. F. Ngai, "Early stage real-time SoC power estimation using RTL instrumentation," in *ASP-DAC*, 2015.

[25] D. Kim, A. Izraelevitz, C. Celio, H. Kim, B. Zimmer, Y. Lee, J. Bachrach, and K. Asanović, "Strober : Fast and Accurate Sample-Based Energy Simulation for Arbitrary RTL," in *ISCA*, 2016.

[26] P. S. Li, A. M. Izraelevitz, and J. Bachrach, "Specification for the FIRRTL Language," Tech. Rep. UCB/EECS-2016-9, EECS Department, University of California, Berkeley, 2016.

[27] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, "Chisel: constructing hardware in a scala embedded language," in *DAC*, 2012.

[28] D. Kim, "RISCV-mini: A Simple RISC-V 3-state Pipeline in Chisel." https://github.com/ucb-bar/riscv-mini.git.

[29] A. Waterman, Y. Lee, K. Asanović, and D. Patterson, "The RISC-V Instruction Set Manual: User-Level ISA Version 2.0," Tech. Rep. UCB/EECS-2014-54, EECS Department, University of California, Berkeley, 2014.

[30] A. Waterman, Y. Lee, R. Avizienis, D. Patterson, and K. Asanovi, "The RISC-V Instruction Set Manual: Privileged Architecture Version 1.7," Tech. Rep. EECS-2015-157, EECS Department, University of California, Berkeley, 2015.