Towards a Cross-Layer Framework for Accurate Power Modeling of Microprocessor Designs

Monir Zaman*, Mustafa M. Shihab*, Ayse K. Coskun† and Yiorgos Makris*
*Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, Texas, USA
†Electrical and Computer Engineering, Boston University, Boston, MA, USA
Email: monir.zaman@utdallas.edu, mustafa.shihab@utdallas.edu, acoskun@bu.edu and yiorgos.makris@utdallas.edu

***Abstract*—While state-of-the-art system-level simulators can deliver swift estimation of power dissipation for microprocessor designs, they do so at the expense of reduced accuracy. On the other hand, RTL simulators are typically cycle-accurate but overwhelmingly time consuming for real-life workloads. Consequently, the design community often has to make a compromise between accuracy and speed.**

In this work, we propose a novel cross-layer approach that can enable accurate power estimation by carefully integrating components from system-level and RTL simulation of the target design. We first leverage the concept of simulation points to transform the workload application and isolate its most critical segments. We then profile the highest weighted simulation point (HWSP) with a RTL simulator (AnyCore) for maximum accuracy, while the rest are simulated with a system-level simulator (gem5) for ensuring fast evaluation. Finally, we combine the integrated set of profiling data as input to the power simulator (McPAT). Our evaluation results for three different SPEC2006 benchmark applications demonstrate that our proposed cross-layer framework can improve the power estimation accuracy by up to 15% for individual simulation points and by ∼9% for the full application, compared to that of a conventional system-level simulation scheme.

I. Introduction

In recent years, continuous process scaling has rendered power dissipation a key consideration and figure of merit for microprocessor designs, often superseding the conventional performance parameters. At every stage of development, accurate simulation frameworks are instrumental for exploring the design space and ensuring selection of the most efficient one. Since exact technology libraries are initially unavailable for new architectures, designers typically simulate their design with either high (system) level models or with low (register transfer) level models. In fact, the choice of the simulation framework for estimating performance and power is a trade-off between accuracy and latency [1].

Register-transfer level (RTL) description of designs are written in hardware description languages (HDL) such as VHDL or Verilog. An RTL model can imitate the actual hardware in a cycle-accurate manner, and is significantly more precise than higher level abstractions. However, characterizing a microprocessor requires simulating it with real-life applications, which can be impractically time-consuming with RTL simulators. We illustrate this in Figure 1 by comparing the RTL simulation time with that of a system-level (SL) simulator for three applications from the SPEC CPU2006 benchmark suite [2]. For example, simulating 100 million instructions of

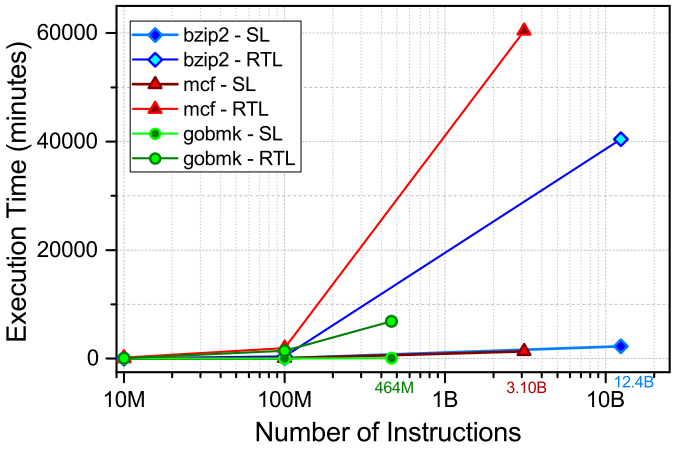

Fig. 1. RISC-V microprocessor simulation: Magnitude of difference in execution time often renders RTL simulation infeasible for designers.

`401.bzip2` with the system-level simulator (gem5) takes only 54 minutes, whereas for the RTL simulator (AnyCore) it takes 377 minutes – which is a 600% increase in simulation time. We also observe that, this trend is common across all three benchmarks, and degrades exponentially for higher number of instructions. It should be noted that, the RTL simulation times for the full benchmarks have been extrapolated from that of their respective first 100M instructions. Furthermore, the latest intellectual properties (IP) are often copyrighted by the commercial vendors, and are unavailable in the public domain. Therefore, the research community often has to depend on dated and less accurate simulation models [3].

On the other hand, system-level simulators model designs at a higher level of abstraction, and are typically written in general-purpose programming languages such as C/C++, Python etc. Consequently, the system-level model of a microprocessor is significantly easier to develop, modify, and parametrize for design space explorations purposes, compared to its RTL counterparts. Most importantly, unlike RTL simulation, system-level simulators can profile large applications within reasonable time (Figure 1). Unfortunately, the significant speedup in simulation comes with a trade-off in accuracy. While the system-level designs attempt to model the real hardware, they often fall short due to cycle inaccuracies and/or other internal design mismatch. Such inaccuracies can be categorized into *modeling*, *specification* and *abstraction*

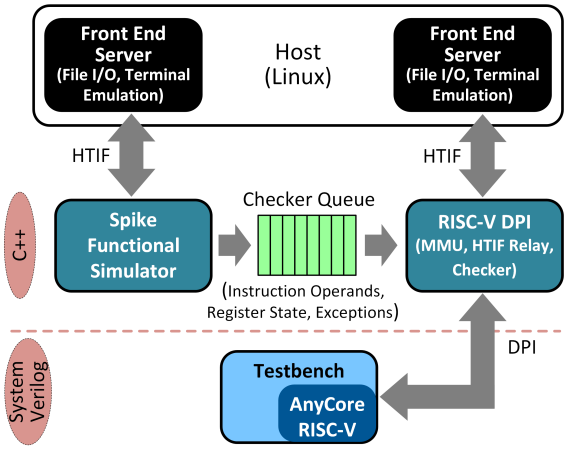

Fig. 2. AnyCore framework: the high-level functional simulator verifies correctness of retired instructions, while DPI calls implement the functions at HDL-level.

errors [1]. While the *modeling* errors tend to improve over time, *specification* and *abstraction* errors are typically more persistent and difficult to fix [4].

While there exists no ideal solution (yet) to avoid the compromise between simulation speed and accuracy, the research community has been rigorously probing this challenge from multiple directions. Sanchez et al. propose a microarchitectural simulator that can reduce the time for detailed simulation by leveraging dynamic binary translation for instruction driven timing models [5]. However, their simulator utilizes system-level description of the design and lacks the accuracy of RT-level information. On the other hand, Oboril et al. propose a detailed simulation framework for simulating and modeling power/area for exploring impact of aging at microarchitectural level [6]. This work also relies on the limited accuracy of the system-level simulator for performance parameters. Instead of using actual RT-level information, the authors introduce different aging models and utilize technology parameters and performance data generated by gem5. As a result, their work does not capture the hardware-level accuracy for performance/power characterization.

In this work, we present a cross-layer scheme that can facilitate accurate power estimation by selectively integrating results from system-level and RTL simulation of the target application. We first leverage the concept of simulation points to transform the workload application and isolate its most critical segments. We then profile the highest weighted simulation point (HWSP) with a RTL simulator for maximum accuracy, while the rest are simulated with a system-level simulator for ensuring fast evaluation. Finally, we use the microarchitectural profile for each simulation point as individual input dataset to the power simulator, and then take a *weighted aggregate* in order to estimate the overall power consumption. In our implementation, we use the AnyCore toolset [7] to perform RTL simulation, the gem5 simulator [8] for detailed system-level simulation, and the McPAT [9] power simulator to generate power estimations for a RISC-V microprocessor [10]. Our evaluation results for three different SPEC2006 benchmark applications demonstrate that our proposed cross-layer framework can improve the power estimation accuracy by up to 15% for individual simulation points and by approximately 9% for the full application, compared to that of a conventional system-level simulation scheme.

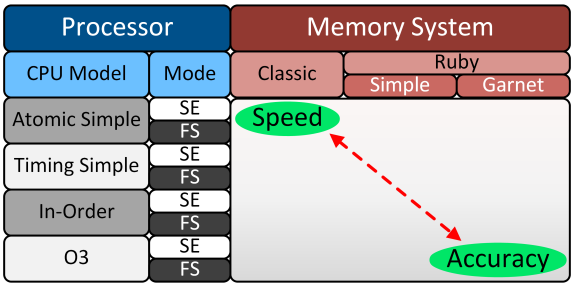

Fig. 3. The gem5 simulator provides multiple CPU models with varying focus on speed and accuracy.

The **main** contributions of this work are as follows:

- We propose a cross-layer simulation platform capable of integrating RTL simulation data with system-level profiling parameters in order to elevate the accuracy of the power simulator.
- We present a comparative analysis of profiling data between system-level and RTL simulation of the HWSP to demonstrate the inaccuracies of the system-level abstraction.
- We apply our proposed methodology on a state-of-the-art RISC-V microprocessor model, and evaluate its performance for multiple SPEC CPU2006 workload applications.

Our evaluation results show that, our proposed cross-layer framework can provide significant improvement in power estimation, compared to existing schemes that leverage data only from a system-level simulator.

II. Background

A. Design simulation at different abstraction levels

In most cases, modern microprocessor designs are evaluated and tuned with either RTL or system-level simulators – particularly in the early stages of development. While the RTL simulators utilize behavioral HDLs for cycle-accurate modeling of the hardware, system-level simulators use high-level models that are faster, albeit less accurate. In the following sections, we briefly discuss the well-established RTL and system-level simulators leveraged in our proposed cross-layer scheme.

1) AnyCore Toolset: The AnyCore toolset is based on a synthesizable, parameterized RTL model of a superscalar, out-of-order microprocessor core. The parameterized description renders it easy to modify various microarchitectural details. Currently the toolset is able to simulate two different instruction sets – PISA [11] and RISC-V [10]. While AnyCore provides the option to choose between a dynamic or a static configuration, we use the static option in this work.

The AnyCore RISC-V RTL model implements the RV64G user-level ISA along with monitor-mode (M-mode) and supervisor-mode (S-mode) for the privileged levels. System

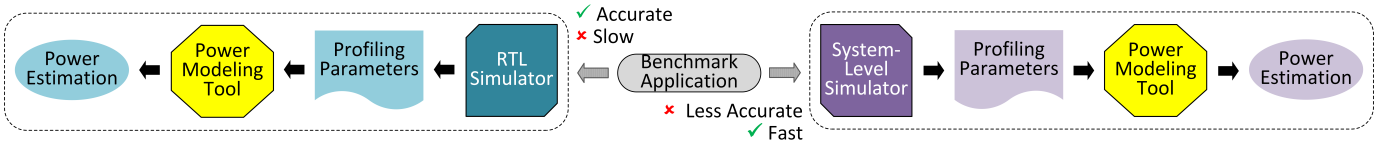

Fig. 4. Conventional power estimation frameworks: Benchmarks are run either using system-level simulator or RTL simulator.

calls in the benchmarks are handled by the RISC-V proxy-kernel (PK) and the front-end server. In addition, the AnyCore design includes a set of L1-caches, where the memory management and address translation tasks are performed by the functional simulator. The functional simulator also emulates the main memory.

Figure 2 presents a high-level view of the AnyCore RISC-V co-simulation framework. The framework guarantees functional correctness of RTL simulation by using "Spike" (RISC-V functional simulator) for each committed instruction. Spike is also used to initialize the registers prior to actual simulation at the RT-level. At the beginning of the simulation, the benchmark is loaded into the *PK* which boots the CPU by setting up the registers, loading the benchmark to the main memory and setting the start program-counter (PC) for the benchmark. Once the desired instruction is reached, the framework starts running detailed simulation with the RTL simulator.

2) gem5 Simulator: gem5 is one of the well-known system-level performance simulator in the open-source domain. As shown in Figure 3, gem5 supports a range of CPU models, simulation modes and memory system hierarchy that corresponds to different levels of simulation speed and accuracy.

The gem5 CPU models are capable of capturing various processor designs and functionality. The *Atomic* CPU is the fastest but least accurate, while the detailed CPU corresponds to most time-consuming but accurate simulations. The detailed CPU model has of two sub-categories – the *In-Order* and the *Out-of-Order* (O3/DerivO3) models. Both of the detailed CPU models are pipelined and highly configurable.

In addition, the gem5 CPU models can run in two simulation *modes* – the system-call emulation (SE) mode and the full-system (FS) mode. In the SE mode, no operating system is loaded by gem5 during the simulation, and system-calls are emulated by the host system. In contrast, the FS mode executes both user-level and kernel-level instructions, and models a complete system by loading an OS in the simulator. The OS boots the machine, simulates all the system-calls, and handles the virtual-to-physical translations.

Also, gem5 is capable of modeling data and instruction caches, memory management unit (MMU), and a unified L2 cache, and supports two types of memory hierarchy. For simpler memory modeling, gem5 uses the *Classic* memory model, where the emphasis is put on the pipeline simulation. The memory uses simple timing model to calculate hits, misses and other memory performance data. On the other hand, the *Ruby* memory model contains various coherence protocols, and can support a more detailed memory hierarchy simulation.

Finally, while the gem5 simulator can simulate different instruction set architectures (ISA), we use recently implemented RISC-V ISA in gem5 in this work [12].

B. SimPoint Toolset

While the most accurate method to profile a workload is to simulate all the instructions, for many real-life applications, such an evaluation can be impractically long. For example, the SPEC CPU2006 benchmark on average contain 2249.75 billion instructions, and executing even a system-level simulation can take days [13, 14]. The SimPoint tool addresses this issue by generating representative phases of a workload, and aggregating the results in order to represent the whole application [15]. The tool identifies and isolates unique phases/regions where the program execution is stable and has a relatively constant CPI. SimPoint starts by generating dynamic execution trace of the given workload and then slices it into user defined sizes. Typically, slices of 1M or 10M instructions can deliver high accuracy with reasonable simulation times [16]. The tool then uses K-means algorithm to form clusters of slices. Towards the end of this stage, a representative slice is chosen from each cluster and set as a *simulation point*. Each simulation point is assigned a weight based on the cluster size it represents, and the sum of the weights is always 1 (i.e., the full application). The weighted simulation points can be simulated in parallel and then aggregated based on weight, in order to generate a fast and accurate characterization profile for the full application. For example, when using the SimPoint tool, Sherwood et al. reported an average IPC error of 3% for SPEC CPU2000 benchmark running Alpha binaries [17].

III. Cross-Layer Framework for Power Estimation

A. Overview

Figure 4 depicts a high-level process flow for conventional performance and power modeling platforms. Typically, profiling data, as well as performance parameters (e.g., IPC) are generated using either a system-level or a RTL simulator. Next, a power simulator utilize such profiling parameters and the activity factor for different microarchitecture modules in order to calculate power consumption.

There are two critical takeaways regarding the existing methodology for power estimation. First, while RTL simulators typically possess a more accurate description of a microprocessor, simulating real-life workload applications with them can often be impractically time-consuming, which in turn forces the designers to opt for the less accurate system-level simulators. Second, the accuracy of the power simulator critically depends on the accuracy of the profiling data it receives as input. Based on these two observations, we propose

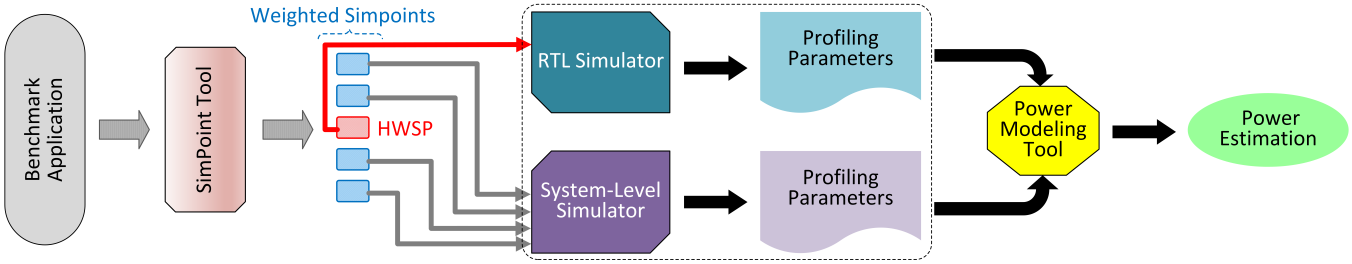

Fig. 5. Proposed cross-layer power estimation framework: (i) the application is transformed into simulation points, (ii) the HWSP is profiled with the RTL simulator, while the rest are profiled in parallel with the system-level simulator, (iii) integrated profiling data is used to generate accurate power estimation.

TABLE I
SIMPOINT DETAILS FOR EVALUATED SPEC CPU2006 BENCHMARKS.

Benchmark	Total Number of Instructions (M)	Number of SimPoints	Instruction Per Simpoint	Simpoint ID	Simpoint Weight	Starting Instruction Number (M)	Ending Instruction Number (M)
401.bzip2	12403	10	10 M	0	0.038	10,890	10,900
				1	0.062	2620	2630
				2	0.027	11800	11810
				3	0.128	11110	11120
				4	0.235	4350	4360
				5	0.188	11830	11840
				6	0.061	8070	8080
				7	0.019	750	760
				8	0.087	9240	9250
				9	0.155	9160	9170
429.mcf	3102	10	10 M	0	0.065	2190	2200
				1	0.065	1170	1180
				2	0.042	90	100
				3	0.187	2290	2300
				4	0.071	1270	1280
				5	0.065	2990	3000
				6	0.006	0	10
				7	0.013	2820	2830
				8	0.013	2860	2870
				9	0.474	1620	1630
445.gobmk	464	6	10 M	0	0.043	10	20
				1	0.152	110	120
				2	0.239	430	440
				3	0.174	120	130
				4	0.239	170	180
				5	0.152	240	250

to profile the most critical segment of a workload with a RTL simulator, while processing the rest of the workload with a fast system-level simulator. We believe that, our proposed cross-layer simulation framework can significantly improve power estimation accuracy, while incurring minimum slowdown in simulation speed, compared to a system-level only, SimPoint-based simulation platform.

As shown in Figure 5, the process flow of our proposed framework can be described in three distinct steps: (i) Using the SimPoint tool, we transform the workload into representative phases. The tool also generates a weight for each phase. (ii) From those phases, we then pick the highest weighted simulation point (HWSP) and profile it with the RT-level simulator, while rest of the phases are simulated using system-level simulator. (iii) Finally, we calculate power dissipation for each simulation point using a power simulator, combine them based on the weights of the corresponding simulation point, and generate estimated power for the complete workload.

It is worth noting that, our framework supports parallel execution of all the simulation points. Therefore, the total simulation time for step (ii) can be represented as following:

$$\textit{Overall workload characterization time} = \textit{max (Profiling time for a simulation point)} \quad (1)$$

Given the same number of instruction simulation, the RT-level takes the maximum amount of time to complete. Thus, based on equation 1, the time needed to characterize performance of a benchmark using simulation points is bound by the time of the RT-level simulation.

B. Implementation

In this section, we detail the step by step implementation for our cross-layer power estimation framework.

1) SimPoint Generation: The first step of our framework is to generate simulation points for each benchmark. In order to generate the simulation points, we use SimPoint toolset v3.2 [15]. We compile three SPEC CPU2006 benchmarks [2] for RISC-V instruction set [10] and generate simulation points each with 10 million instruction interval. The maximum

TABLE II
MICROARCHITECTURE DETAILS FOR ANYCORE CORE-1

Feature	Value	Feature	Value
Fetch-to-Dispatch width	1	L1 Ins. Cache	2 KB
Issue-to-Execute width	3	L1 Data Cache	8 KB
Retire width	1	Active List size	96
Issue Queue	16	Functional units	4
Load/Store Queue	32/32	Physical Register	160
BTB size	1024	RAS	16
BPU entries	1024	Floating-point Pipeline	0

number of simulation point was set to 10. Table I shows the detailed simulation point breakdown generated by the SimPoint tool for the three SPEC benchmark we used for our experiment. Both `401.bzip2` and `429.mcf` benchmarks have 10 simulation points and `445.gobmk` benchmark has 6. The table also shows the start and end point for the detailed simulation of 10 million instructions. In the table, highlighted cells represents the highest weighted simulation point (HWSP) we used for detailed RT-level simulation for each benchmark. It should be noted that, if there exists multiple HWSPs for a benchmark, our current scheme picks the first one. For example, in Table I, the `445.gobmk` benchmark has two HWSPs – SimPoints 2 and 4, and we picked SimPoint 2.

2) Configuration for RTL and System-level Simulation: **AnyCore RTL.** We use static core-1 configuration for AnyCore RISC-V RTL setting. The superscalar, out-of-order microprocessor can fetch, decode, rename one instruction every clock cycle. It issues three instruction each cycle and has four functional units in total in the pipeline. At every clock, one instruction is committed. The pipeline also implements a 2-bit branch predictor unit to predict branch directions in the fetch stage. Table II shows some of the key microarchitectural details for the core-1 setting used in the RTL simulation.

In order to run 10 million detailed instructions starting from the simulation points generated by the SimPoint tool, AnyCore RTL simulator fast forwards until the desired instruction number and starts to run detail simulation from that instruction count. For example, for `401.bzip2` benchmark, we fast forward first 4350 million instructions and then simulate 10 million instructions using the RTL simulator.

gem5 Simulator. We first modify detailed CPU of the gem5 simulator to match the microarchitectural details from Table II. This modification includes changing the branch predictor unit, pipeline width and depths, different parameter sizes etc. We also modify the functional units latency and number of functional unit used by the gem5 out-of-order CPU.

After the modification, we run each simulation points using detailed CPU in gem5. Each simulation points are run in parallel for 10 million instructions. To reduce the effect of cold cache start, we run 100 million warm-up prior to running detailed simulation from the simulation point start (For simulation point instruction start point less than 100 million, we either skip the warm-up (`429.mcf: Simpoint id 6`) or use reduced number of warm-up (`445.gobmk: Simpoint id 0`). At the end of each simulation, detailed data for different performance parameters are generated.

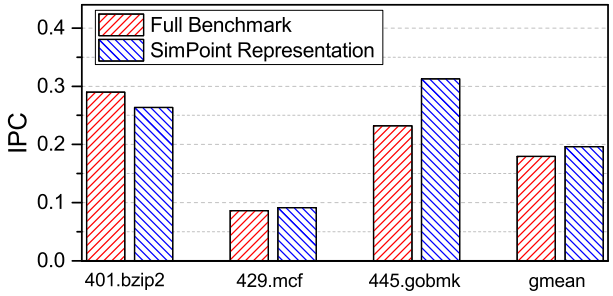

Fig. 6. Variance in IPC for full benchmarks vs. SimPoint representations.

TABLE III
`429.MCF`: FULL BENCHMARK VS. SIMPOINT REPRESENTATION.

Parameter	Unit	Full Benchmark	SimPoint Representation	Variance (%)
Load Count	Per 1000 Ins. (PKI)	368.22	380.98	3.47
Store Count		46.12	45.62	1.08
Branch Count		261.13	268.59	2.86
Branch Mispred		28.95	29.33	1.32
Cache Miss (I)		3.74	3.78	1.00
Cache Miss (D)		320.02	331.47	3.58

3) Power Estimation with McPAT Power Simulator: For the final step in our framework, we use McPAT power simulator to estimate runtime dynamic power consumed by the core [9]. McPAT uses detailed XML file as its input interface. The input file contains architectural details and activity information of various performance parameters generated by the simulators from previous stage. We use 65nm technology node for power estimation. McPAT generates peak, leakage and total runtime power consumption by the core and each of its sub-modules. In this work, we report the runtime dynamic power consumption by the core. In our cross-layer approach, for generating the runtime dynamic power estimation for a benchmark, we first evaluate runtime dynamic power for each of its SimPoints. The input data for each of these SimPoint specific power estimation is generated either from gem5 or from the RTL simulator – depending on the weight of the SimPoint. As mentioned earlier, the HWSP is simulated at the RT-level and rest of the SimPoints are simulated using gem5. Once McPAT generates the runtime dynamic power for each SimPoint, we then multiply each result with its respective SimPoint weight and aggregate them to generate the final representative power for the full benchmark.

IV. EVALUATION

In this section, we discuss our evaluation results and demonstrate the accuracy improvement in power estimation achieved with the proposed cross-layer simulation framework.

A. Experimental Setup

We perform our system-level simulations with the gem5 simulator. For full benchmark simulations with the detailed CPU model, we first run a standard 100 million instructions *warm-up*, and then perform detailed simulation for the rest of the application. Finally, we run the AnyCore RTL simulator using the Cadence NC-Verilog tool (version 15.20).

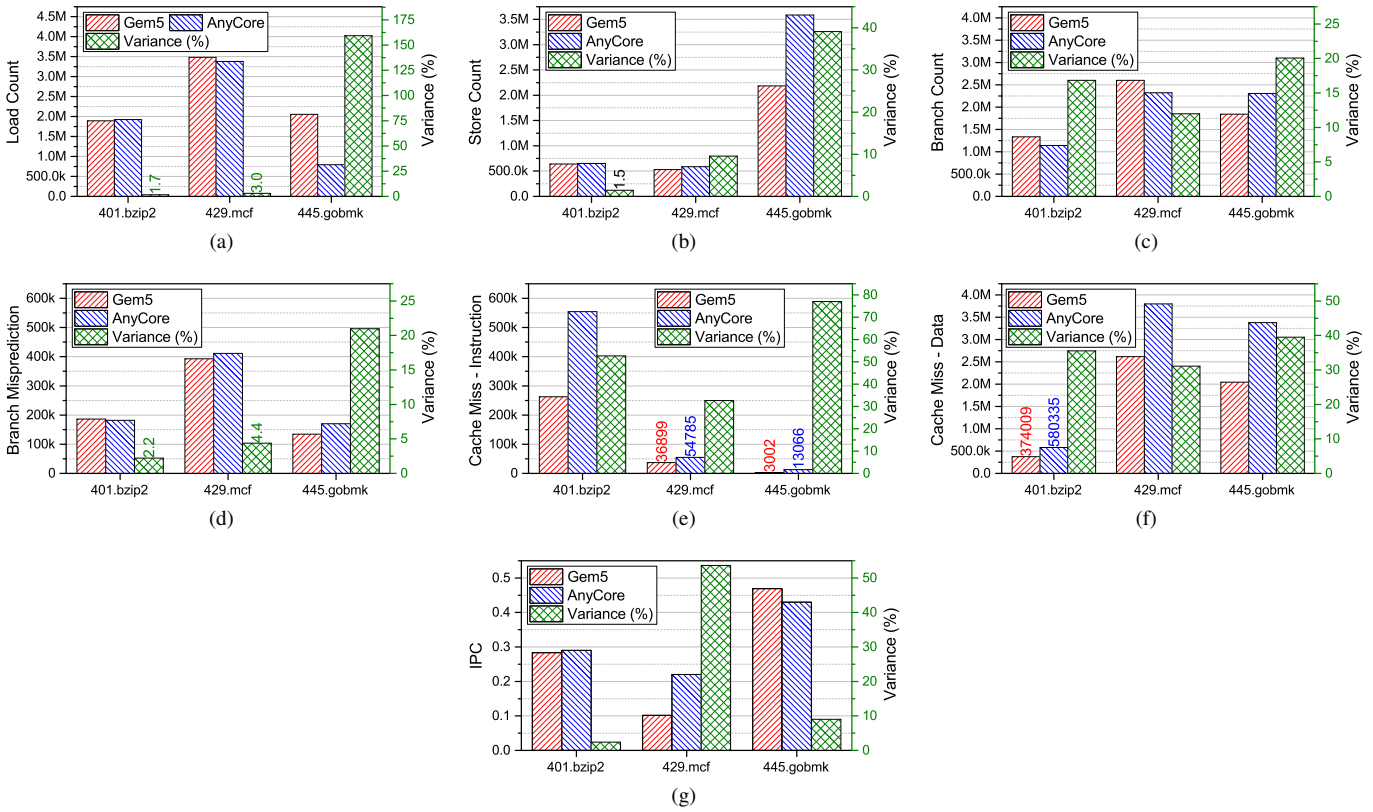

Fig. 7. Profiling accuracy improvement with the RTL (AnyCore) simulator over system-level (gem5) simulator. We show the improvement for various micro-architectural parameters achieved for the highest-weighted SimPoint between the two abstraction levels.

B. Evaluation Results

1) Verification of SimPoint-based representation: Our framework is set upon the idea of utilizing representative phases in lieu of a complete workload application. Therefore, it is critical that, in aggregate, the representative phases (i.e., simulation points) actually mimic the behavior of the original application they are supposed to 'represent'. In order to verify this stipulation, we first perform a comparative analysis on the characterization parameters collected from the benchmark applications and their SimPoint-based representations.

Figure 6 shows the comparison of instruction per cycle (IPC) between the benchmark applications and their respective SimPoint-based representations. We can observe that, the IPC for full benchmark runs are 0.290, 0.086 and 0.232 for `401.bzip2`, `429.mcf` and `445.gobmk`, respectively. Also, when using SimPoint-based simulation to represent the same benchmarks, the IPC results are 0.264, 0.091 and 0.313, respectively. Consequently, we can confirm that the average (gmean) variance between the IPC from the full benchmarks and their representative SimPoints is only 1.64%.

In addition, we present a detailed comparison of the critical characterization parameters for the `429.mcf` benchmark and its SimPoint-based representation in Table III. We can observe that, the variance for the number of *load* instruction count is 3.47%, while for the *store* instructions it is 1.08%. On the other hand, the variances for *branch* instruction count and *branch mispredictions* are 2.86% and 1.32%, respectively. Finally, our evaluation shows the variance for *instruction cache misses* is 1%, and for *data cache misses* it is 3.58%.

From the above discussion, we can reasonably conclude that a set of carefully generated SimPoints can accurately represent the characterization behavior of the original application.

2) Improved profiling with cross-layer framework: As mentioned earlier, RTL simulations provide significantly more accurate profiling compared to their system-level counterparts. In order to explore the amount of discrepancies in the critical parameters, we simulate the highest-weighted SimPoints (HWSP) in both the RTL (AnyCore) and the system-level (gem5) simulator. The HWSP for each benchmark is run for 10 million instructions, starting at the stated instruction number (Table I). Figure 7 presents the result of this evaluation.

Load count. Figure 7a shows the variance in number of load instructions for the three benchmarks. We can see from the figure that, the HWSP of `401.bzip2` exhibits the lowest variance of 1.7% between the RTL and system-level simulation, while the variance is highest for `445.gobmk` at 159%.

Store count. The comparative result for number of store instructions is shown in Figure 7b. We can see that `445.gobmk` again manifests the maximum variance of 39%, whereas for `401.bzip2` the variance is the minimum at 1.5%.

Branch count. Figure 7c shows that, the variance in branch instruction count is the highest for `445.gobmk` at 20%. In contrast, `429.mcf` shows the least variance of 12%.

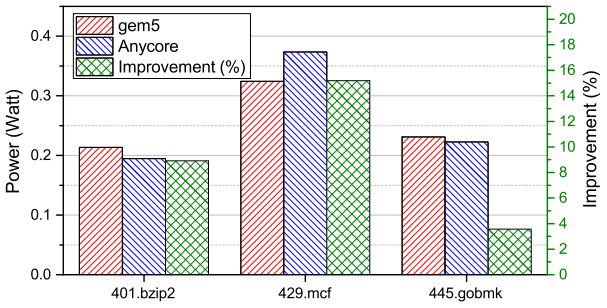

Fig. 8. Accuracy improvement in power estimation for the highest-weighted SimPoint (HWSP).

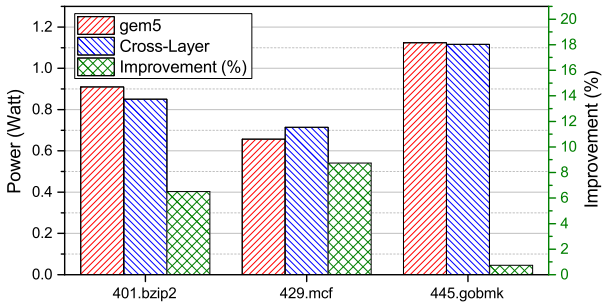

Fig. 9. Accuracy improvement in power estimation for full benchmark applications.

Branch misprediction. As shown in Figure 7d, for branch misprediction count, `401.bzip2` exhibits the lowest 2% variation, while for `445.gobmmk` it stands at 21%.

Cache miss. Figure 7e portrays the fluctuation in instruction cache misses between the two simulation platforms. We note that, `401.bzip2`, `429.mcf`, and `445.gobmk` report variances of 53%, 33%, and 77%, respectively. In a similar fashion, Figure 7f shows the variances in data cache misses to be 36%, 31%, and 39%, for `401.bzip2`, `429.mcf`, and `445.gobmk`, respectively.

IPC. As our final point of comparison, we present variance in IPC values attained from the gem5 and the AnyCore simulator in Fiugre 7g. One can note that, `429.mcf` exhibits the highest amount of variation in IPC at 54%, while `401.bzip2` shows the lowest variation of 2%.

It is worth noting that, the variations in result for the microarchitectural parameters are primarily due to the inherent simplification and inaccuracy in gem5 modeling compared to its RTL counterpart [18]. This inaccuracy can be overcome by integrating the RTL simulation results for each of these parameters. Since RTL simulation gives more accurate result for such performance parameters, we can concur from the above analysis that, our proposed cross-layer simulation framework enables a significantly improved profiling of the highest-weighted simulation points from each benchmark application. This in turn should enable us to achieve more accurate power estimations, as these profiling parameters are directly utilized by the power simulator.

3) Power Estimation Results: Figure 8 portrays the improvement in power estimation accuracy by integrating profiling data for the highest weighted simulation point (HWSP) with the RTL simulator. Specifically, the figure compares the runtime dynamic power from system-level (gem5) simulation with that of the RTL simulation (AnyCore). We can observe that, with profiling data from gem5, McPAT estimates the power dissipation to be 0.21W, 0.32W and 0.23W for `401.bzip2, 429.mcf and 445.gobmk`, respectively. However, when McPAT is fed with the profiling data from AnyCore, the power estimation changes by 8.91%, 15.18% and 3.56%, for `401.bzip2, 429.mcf and 445.gobmk`, respectively.

Finally, in Figure 9, we evaluate the impact of our proposed cross-layer scheme on the power estimation accuracy for the full benchmark applications. Specifically, we compare McPAT's power estimation numbers for the gem5-only data against that with our cross-layer approach where the power simulator leverages data from both gem5 and the RTL simulator. We can observe that, the improvement in overall accuracy of power estimation for `401.bzip2` is 6.5%, for `429.mcf` the improvement is 8.7%, and for `445.gobmk` the accuracy is improved by 0.73%.

4) Simulation time: As stated in Equation 1, for a SimPoint-based framework like ours, the characterization time for a benchmark is bound by the time of the RT-level simulation. In our evaluation, simulating the HWSP with the RTL simulator takes 66, 172 and 36 minutes, for `401.bzip2, 429.mcf` and `445.gobmk`, respectively. On the other hand, a full benchmark simulation with the gem5 (detailed CPU model) simulator takes 2270, 1338, and 71 minutes for `401.bzip2, 429.mcf`, and `445.gobmk`, respectively. These results confirm that, our framework can be, on average, ∼78% faster than a conventional full benchmark simulation with a system-level simulator.

V. Related Work

gem5 simulator. gem5 is a widely used system-level simulator for performance characterization, design modelling and design space exploration [8]. Fernando et al. used gem5 simulator to model both in-order and out-of-order arm microprocessors [3]. Their design modeled the microarchitectural details based on published and estimated data. Yang et al. extends gem5 to build a VLIW simulation platform [19]. They also modeled their design based on cycle accurate simulator and finally validates against the RTL simulator. Note that, our scheme is different from the prior works because of the incorporation of RT-level information for accuracy improvement. Moreover, instead of running the full workloads, we leverage smart usage of SimPoint generated phases of the workload to reduce overall simulation time.

SimPoint-based benchmark simulation. Simpoint-based simulations create represtative points/phases for a workload, and simulates those points only. Maximilien et al. uses Simpoint technique to profile benchmarks for different performance parameters and predict the performance of the bench-

mark [20]. Their model is solely dependent on the SimPoint accuracy and the hardware model used by the system-level simulator. Coskun et al. used the SimPoint tool to create a database for benchmarks and use that for dynamic thermal management [21]. However, their work did not include any RT-level data for improving accuracy.

To the best of our knowledge, our proposed scheme is the first to utilize SimPoints for integrating high-level and RTL simulation in order to achieve accurate power estimation.

VI. Conclusion and Future Work

In this work, we presented a cross-layer scheme that enables accurate power estimation for microprocessor designs. Our proposed scheme first utilizes SimPoints to locate critical segments of an application. We then selectively run system-level (gem5) and RT-level (AnyCore) simulation on such segments for collecting more input for the power simulator (McPAT). Our evaluation results show that, the proposed scheme can improve power estimation accuracy by more than 15% for individual SimPoints, and by $\sim$9% for full benchmark applications – compared to the existing system-level simulation based frameworks.

In future, we plan to extend this work by incorporating a detailed analysis on isolating the modeling mismatches that can cause unwanted variances in profiling results from gem5 and the RTL simulator. We are also working on expanding our evaluation and analyses by adding a wider range of benchmarks. Finally, we will explore the microarchitecrtural characteristics of individual SimPoints for all the benchmarks, which in turn may allow us to select the SimPoint to be simulated on the RTL Simulator on a case-by-case basis (rather than just the HWSP), and improve the performance of our cross-layer framework thereby.

References

[1] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy evaluation of gem5 simulator system," *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, 2012.

[2] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.

[3] F. A. Endo, D. Courouss, and H. P. Charles, "Micro-architectural simulation of in-order and out-of-order arm microprocessors with gem5," in *2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV)*, 2014.

[8] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[4] B. Black and J. P. Shen, "Calibration of microprocessor performance models," *Computer*, vol. 31, no. 5, pp. 59–65, May 1998.

[5] D. Sanchez and C. Kozyrakis, "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," *ACM SIGARCH Computer architecture news*, vol. 41, no. 3, pp. 475–486, 2013.

[6] F. Oboril and M. B. Tahoori, "Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level," *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, 2012.

[7] R. B. R. Chowdhury, A. K. Kannepalli, S. Ku, and E. Rotenberg, "Anycore: A synthesizable rtl model for exploring and fabricating adaptive superscalar cores," *Performance Analysis of Systems and Software (ISPASS), 2016 IEEE International Symposium on*, 2016.

[9] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *MICRO 42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 469–480.

[10] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The risc-v instruction set manual, volume i: Base user-level isa. eecs department," *University of California*, 2011.

[11] D. Burger, T. M. Austin, and S. Bennett, "Evaluating future microprocessors: the simplescalar tool set," University of Wisconsin-Madison, Tech. Rep., 1996.

[12] A. Roelke and M. Stan, "Risc5: Implementing the RISC-V ISA in gem5," *First Workshop on Computer Architecture Research with RISC-V (CARRV)*, 2017.

[13] A. A. Nair and L. K. John, "Simulation points for spec cpu 2006," in *2008 IEEE International Conference on Computer Design*, 2008.

[14] K. Ganesan, D. Panwar, and L. K. John, "Generation, validation and analysis of spec cpu2006 simulation points based on branch, memory and tlb characteristics," in *SPEC Benchmark Workshop*. Springer, 2009.

[15] G. Hamerly, E. Perelman, J. Lau, and B. Calder, "Simpoint 3.0: Faster and more flexible program phase analysis," *Journal of Instruction Level Parallelism*, vol. 7, no. 4, pp. 1–28, 2005.

[16] E. Perelman, G. Hamerly, and B. Calder, "Picking statistically valid and early simulation points," in *Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '03, 2003.

[17] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 5, pp. 45–57, Oct. 2002.

[18] A. Gutierrez, J. Pusdesris, R. G. Dreslinski, T. Mudge, C. Sudanthi, C. D. Emmons, M. Hayenga, and N. Paver, "Sources of error in full-system simulation," in *Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on*, 2014.

[19] L. Yang, L. Wang, X. Zhang, and D. Wang, "An approach to build cycle accurate full system vliw simulation platform," *Simulation Modelling Practice and Theory*, vol. 67, pp. 14 – 28, 2016.

[20] M. B. Breughe, S. Eyerman, and L. Eeckhout, "Mechanistic analytical modeling of superscalar in-order processor performance," *ACM Trans. Archit. Code Optim.*, vol. 11, no. 4, pp. 50:1–50:26, Jan. 2015.

[21] A. K. Coskun, R. Strong, D. M. Tullsen, and T. Simunic Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 169–180, Jun. 2009.