SimPoint-Based Microarchitectural Hotspot & Energy-Efficiency Analysis of RISC-V OoO CPUs

Odysseas Chatzopoulos[†] Maria Trakosa[†] George Papadimitriou[†] Wing Shek Wong[§] Dimitris Gizopoulos[†]
[†]*University of Athens,* Greece, {od.chatzopoulos | mariatrak | georgepap | dgizop}@di.uoa.gr
[§]*Intel, Austin, Texas*, wing.shek.wong@intel.com

***Abstract*—Building on the flexibility of open-source RISC-V-based CPU designs at the register-transfer level (RTL) we deliver a characterization study that is not feasible on commercial CPUs. We identify the major power-consuming hardware structures by focusing on SonicBOOM's out-of-order (OoO) microarchitecture across three design points of increasing aggressiveness. By introducing and employing the SimPoint methodology on a diverse set of workloads, we shed light on the relationship between microarchitecture and energy efficiency of BOOM, which is the highest-performance CPU design in the public domain. Our analysis highlights the Branch Prediction and the Instruction Scheduler Units as the most power-intensive components. We evaluate the energy efficiency (performance per watt) of the three design configurations of BOOM and conclude that the smallest of the three OoO cores, while being the slowest, prevails. The proposed experimental flow can be used to evaluate any CPU design using arbitrarily large workloads due to the effective use of the SimPoint methodology we introduce in Chipyard - in our case offering a 45-fold reduction of simulation time. Our findings, encompassing 8 key takeaways, can assist microprocessor designers in optimizing energy efficiency by addressing major power contributors.**

I. Introduction

Microarchitecture plays a crucial role in determining the power consumption of a processor or a computer system [1]. Power consumption is a significant concern in modern computing devices due to limitations in battery life, energy efficiency requirements, and the need for thermal management [2]. Power consumption in modern microarchitectures is a multidimensional issue influenced by various factors beyond microarchitecture design, including workload characteristics, system-level design decisions, cooling solutions, and software optimizations [3]–[5]. Among the above factors, microarchitectural decisions profoundly impact power consumption, and striking a balance between power expenditure and performance is a key challenge in modern computing system design. By measuring and optimizing microarchitecture choices, designers can contribute to energy-efficient solutions that align with the growing demand for sustainable and power-conscious technologies [6], [7].

In the era of energy-efficient computing, power modeling, and estimation has emerged as a critical aspect of processor design. With the growing popularity of the RISC-V instruction set architecture (ISA), understanding the combined performance and power implications of microarchitectural choices on RISC-V cores becomes increasingly vital. Having insights into how microarchitecture affects power consumption allows designers to make optimized decisions when designing RISC-V cores (and likely infer the impact of similar decisions on other ISAs and CPUs). Power estimation can be performed at all stages of the design process, from early pre-RTL performance modeling down to post-place-and-route (post-PnR) power estimation. In this study, our focus is RTL power estimation that achieves both high accuracy (compared to pre-RTL modeling on performance simulators) and relatively high speed and flexibility (compared to post-PnR). RTL power estimation encompasses assessing both dynamic and static (leakage) power. Quantifying the power consumption of individual components and their interactions enables designers and computer architects to identify power-hungry parts of the processor design (hot spots), bottlenecks, and opportunities for optimization in the next processor generations.

Previous studies have focused on power consumption in computer systems [8], including studies on power estimation techniques [9], power models [10], and energy-efficient microarchitectural optimizations [6]. However, to the best of our knowledge, a comprehensive understanding of the direct effect of microarchitecture on energy efficiency has not been presented in the literature. Such knowledge is crucial for guiding the design and development of future systems, as it provides insights into the trade-offs between performance and power consumption, enabling informed decision-making during the design process.

This paper builds on top of a publicly available OoO RISC-V CPU (SonicBOOM) [11] and aims to bridge this gap by presenting an in-depth exploration of the impact of microarchitecture on energy efficiency. We delve into the intricate microarchitectural design choices and their effects on power consumption profiles. By employing a diverse set of workloads and leveraging state-of-the-art power measurement techniques and tools, we aim to provide a comprehensive understanding of how various microarchitectural configurations impact performance and power consumption. Through this investigation, we seek to shed light on the relationship between microarchitecture, power consumption and performance and contribute to the ongoing efforts to optimize energy efficiency in modern microarchitectures.

In this paper, we make the following contributions:

1) We present a detailed register-transfer level (RTL) power and performance estimation flow for three distinct BOOM configurations. Using this flow, we analyze the total CPU core power across thirteen microarchitectural

components with fine granularity, accomplished through an end-to-end integration of commercial EDA (Electronic Design Automation) and open-source tools.
2) We assess the impact of microarchitecture on CPU power consumption for the three different configurations of the open-source RISC-V BOOM core.
3) We extend Chipyard's [12] checkpointing infrastructure to support the SimPoint [13] methodology. This allows us to evaluate on average 2 orders of magnitude larger workloads than previous studies [14].
4) We present 8 key takeaways from our analysis of how each of the microarchitectural components we considered contribute to the power consumption of the entire microprocessor core. Future processor designs could exploit these insights to further improve energy efficiency without compromising their performance.
5) Finally, we present the performance-per-watt ratings for all BOOM configurations utilized in this study. Our evaluation results demonstrate that while the smaller BOOM design yields notably lower performance (on average 1.6× lower IPC) compared to the largest BOOM design for the considered workloads, it significantly outperforms in energy efficiency, delivering higher performance per unit of power (on average 52% higher).

II. Background

CPU design engineers employ power estimation and measurement techniques throughout the chip design process, from pre-RTL power modeling[1] to post-PnR power estimation. As we descend to lower levels of abstraction, the accuracy of power estimation increases, while simulation throughput, flexibility, and ease-of-use decrease. This paper emphasizes RTL power estimation, as it offers a good trade-off among accuracy, performance, and flexibility, enabling us to accurately evaluate the impact of microarchitecture on chip power consumption. In this section, we provide a brief description of each power estimation abstraction level.

A. Pre-RTL Power Estimation

There are several pre-RTL power estimation tools targeted at multicore microprocessors [16], [17], memory-arrays [18] and domain-specific accelerators [19], [20]. These tools use software models of varying complexity to estimate the power consumption of these designs. For example, McPAT [16] includes a wide range of models including in-order and out-of-order cores, networks-on-chip, shared caches, integrated memory controllers, and multiple-domain clocking. Such models combined with performance simulators allow engineers to perform quick design space exploration very early in the design process. Although these models are fast and easy to use, their accuracy is quite limited. For example, McPAT reports an average power estimation error of 21% among 4 different processor configurations [16].

[1]Early in the design phase and before the actual RTL design, typically using performance models on microarchitectural simulators, e.g., gem5 [15].

B. RTL Power Estimation

RTL power estimation tools [21] use the actual RTL design source to estimate chip power consumption. All three major EDA companies, Cadence, Synopsys, and Siemens have developed such tools, namely Cadence Joules [22], Synopsys Primepower [23] and Siemens PowerPro [24]. For this study, we use Cadence Joules, although all these tools share similar workflows. Specifically, the workflow of Cadence Joules is enumerated below:

1) As shown in Fig. 1, initially both RTL source (Verilog [25], SystemVerilog [26], VHDL [27]) and Process Design Kit (PDK) library files (Liberty, LEF) [28] are read by the tool. The RTL source is elaborated and saved in a database to be used in the Design Mapping (Synthesis) and Stimulus (Trace File) Processing stages.
2) During the technology mapping stage, a rough synthesized netlist is created by implementing the arbitrary RTL constructs with actual library standard cells (the "Synthesis" box in Fig. 1). This process is much faster than an actual full-chip synthesis step. Joules takes into account the PDK liberty files that contain timing, area, and power specifications of standard cells and constraint files that define the desired clock frequency and other important design constraints.
3) Trace files from RTL simulators are used to determine the toggle rate of each signal in the design. Joules can accept a variety of trace file formats ranging from a few Megabytes to hundreds of Gigabytes of data, depending on workload length and design complexity.
4) Finally, power estimation is performed by first generating an appropriate clock tree and then using the mapped netlist in conjunction with the calculated toggle rate of each signal to determine the Leakage, Internal, and Switching power of the design.

C. Post-Synthesis Power Estimation

Post-synthesis power estimation tools, instead of using the RTL source as input, rely on an already synthesized netlist. Since the synthesized netlist is the actual netlist used as input for the place-and-route step, the results of power estimation are more accurate. Nevertheless, interconnect and placement information is still missing, thus limiting the fidelity of the results. In our study, the decrease in throughput was not justified by the relatively modest improvement in accuracy.

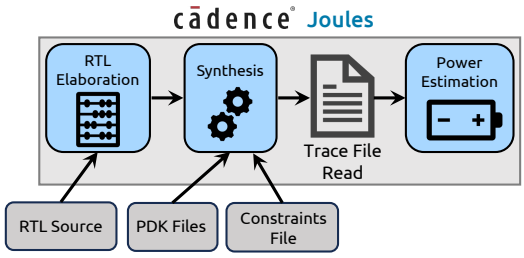

Fig. 1. Cadence Joules power estimation flow.

D. Post-Place-and-Route Power Estimation

The PnR steps transport the chip design from the logical domain to the physical one. A floorplan is used to create a foundation defining the dimensions of the chip and the placement of any hard macros such as SRAMs and special analog cells. Subsequently, all standard cells are placed within the specified floorplan and finally, nets (wires) are routed to connect all the standard cells. Post-PnR tools use this final representation of the chip in the physical domain to calculate power, taking into account the interconnect and actual placement of cells inside the chip. Although this method yields highly accurate results, the significant decrease in throughput along with the need to create a floorplan for each design we want to analyze, renders it unsuitable for our study.

E. Power Dissipation Sources

The power dissipation sources in a digital CMOS integrated circuit [21] are: *Leakage (Static) Power*, *Internal (Short Circuit) Power* and *Switching Power*. *Leakage power* is a consequence of MOSFETs not completely turning off. This is due to a variety of effects including sub-threshold leakage, junction current, and gate tunneling. RTL power estimation tools calculate the system leakage power by summing the leakage power of each system cell as characterized by library vendors in the liberty files. State-based modeling, i.e., cell leakage power being based on cell state, is supported by all major tool vendors. *Internal power* is the power consumed within the boundaries of the cell. It is attributed to momentary short-circuit currents that occur between the power supply and ground, as well as power consumed in internal nets due to charging and discharging of internal capacitance. *Switching power* is consumed when charging or discharging the load capacitance at the output of a cell during logic changes. This load capacitance is the sum of the gate and net capacitance on the driven output. The switching power is directly proportional to the capacitance, the net toggle rate, and the square of operating voltage.

III. Tools, Models, Frameworks

To evaluate the effects of microarchitecture on-chip power consumption, we employ a set of both open-source and commercial tools. These tools enable fast design space exploration of different microprocessor configurations, providing us with metrics such as performance, area, and energy efficiency. In this section, we describe these tools and their interoperability.

A. Chipyard

Chipyard [12] is a widely-used, open-source framework designed at the University of California, Berkeley that enables the design and evaluation of full-system hardware using an agile development approach. Chipyard contains an extensive collection of tools and libraries that facilitate the integration of open-source and commercial tools for the development, simulation, and implementation of complex Systems-on-Chip (SoCs). Multiple configurable components can be combined to create a complete SoC which can be verified through both FPGA-Accelerated RTL simulation (FireSim [29]) and Software RTL simulation, as well as pushed through an automated VLSI design flow that can produce tape-out ready GDSII files. Chipyard's configurable RTL generators rely on the Chisel hardware construction language [30]. Chisel designs are first converted to FIRRTL (Flexible Intermediate Representation for RTL). Since Chipyard supports three main flows namely: Software RTL Simulation, FPGA Accelerated RTL Simulation, and automated VLSI design (HAMMER [31]) it relies on FIRRTL [32] transforms to produce suitable Verilog code for each target. This enables designers to write RTL once, being agnostic of the target platform, and making the entire design process much more streamlined.

B. SonicBOOM

BOOM [33] (Berkeley Out-of-Order Machine) is the state-of-the-art, open-source, OoO RISC-V core developed at the University of California, Berkeley. It is written in Chisel and is highly parameterized, allowing for fast design-space exploration and optimization for a wide range of target applications. In this study, we are using the 3rd generation of BOOM called SonicBOOM [11], which is the latest BOOM release with a variety of performance improvements. Fig. 2 presents a microarchitectural diagram of the BOOM core. The BOOM pipeline is composed of ten logical stages [34]: Fetch, Decode, Register Rename, Dispatch, Issue, Register Read, Execute, Memory, Writeback, and Commit. However, in SonicBOOM, certain stages are combined or overlapped to optimize performance.

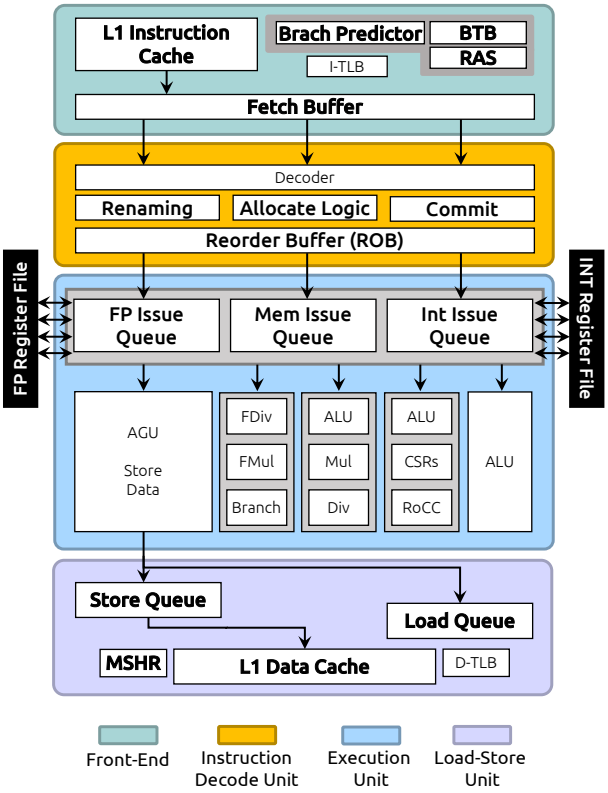

Fig. 2. SonicBOOM microarchitecture. Different background colors show the distinct major units of the core.

C. ASAP7

ASAP7 [28] is a 7-nm predictive process design kit (PDK). It was developed in collaboration with Arm Ltd. for academic use. The kit is based on the assumptions for the 7-nm technology node and is not tied to any specific foundry for manufacturing. The ASAP7 library contains all necessary collateral, including 4x scaled physical pin and blockage views, sample GDSII [28] and Open Access schematic/symbol views, LVS netlists, and parasitic extracted (PEX) cell netlists for circuit simulation. The library includes 106 combinational logic cells, 17 flip-flops (both scan and non-scan), six latches, and three integrated clock gaters with varying drive strengths.

D. Verilator

Verilator [35] is a powerful tool in the domain of hardware description and modeling, specifically a high-performance simulator for the Verilog hardware description language (HDL). It is renowned for its speed, reputed as the fastest open-source simulator for Verilog HDL [36]. It accepts synthesizable Verilog, as well as some system Verilog and synthesis assertions, and compiles them into a standalone C++ or SystemC model that can be incorporated into larger software simulations. As a tool, Verilator is primarily used for synthesizable digital logic designs and has a special emphasis on enabling fast simulation. From an adoption perspective, Verilator has been widely used in both academia and industry. Its ability to handle complex and large designs, combined with its high speed and adaptability, makes it an attractive choice for hardware engineers and researchers. Many significant designs, including open-source processor designs like CVA6 [37], ROCKET [38], and BOOM [11], have utilized Verilator for their simulation and verification tasks.

E. SimPoint

SimPoint [39] is a sophisticated performance analysis tool designed for evaluating and understanding the behavior of computer programs, particularly those executed on large-scale computing systems. Developed to handle the complexity of modern processors and applications, SimPoint aids researchers and engineers in pinpointing representative "simulation points" within the execution of a program. These simulation points capture the essential characteristics of the program's behavior, enabling a more efficient and manageable analysis of its performance. By intelligently selecting a subset of these simulation points, SimPoint significantly reduces the computational resources required for detailed simulations while maintaining a high degree of accuracy. This makes it a valuable tool for optimizing software and hardware systems, facilitating the identification of critical bottlenecks, and ultimately enhancing the overall efficiency of complex computing environments. In Section IV-A, we delve into the challenges associated with integrating SimPoints into RTL-power analysis and detail the strategies employed to overcome them.

IV. Experimental Results & Analysis

Our experimental flow is aligned with previous work [14] and is shown in Fig. 3 and Fig. 4.

A. Experimental Setup

We first select the target Chipyard System-on-Chip (SoC) configurations ❶. These, feature three distinct BOOM setups: MediumBOOM (2-wide core), LargeBOOM (3-wide core), and MegaBOOM (4-wide core), as detailed in Table I. The configurations encompass the entire chip, including the BOOM core, caches, and peripherals, designed as customizable, Chisel-based generator IP blocks ❷. Through a systematic build process, these Chisel-based designs are transformed into Verilog representations ❸. Subsequently, Verilator ❹ converts the target Verilog into a fast, multithreaded, cycle-accurate simulator ❺, capable of generating cycle-by-cycle signal traces for the entire chip.

Executing the benchmarks/SimPoint checkpoints (the generation of SimPoint checkpoints is explained below) ❻ on our fast multithreaded simulator results in the generation of a trace file for each one ❼. These trace files, along with the target Verilog RTL code produced in a previous step, are provided to Cadence Joules ❽. Additionally, the ASAP7 PDK library files ❾ and the HAMMER-generated TCL script ❿—which orchestrates the Joules power-estimation flow as outlined in Section II-B—are included in the input. To investigate the impact of microarchitecture on power consumption, a consistent clock rate is employed across all three BOOM config-

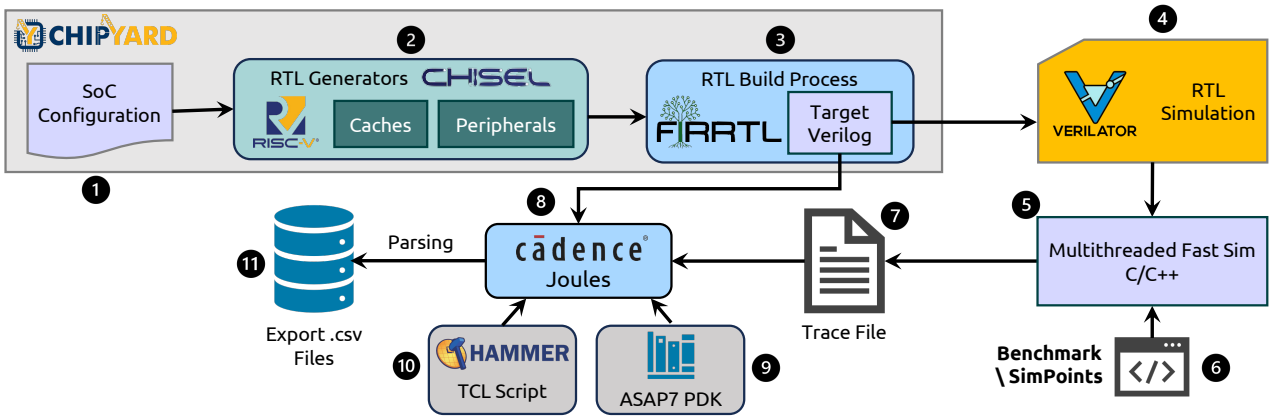

Fig. 3. The experimental flow used in this study for RTL power modeling and evaluation.

urations; consequently, the microarchitectures differ in their IPC (instructions per cycle). The designated clock frequency for our designs is *500MHz*, which is consistent with previous studies [14], [40] that use the ASAP7 PDK. The power consumption results from Joules are parsed and processed, allowing for fine-grained analysis of the power consumption of the target BOOM configurations ⓫.

We employ eleven diverse benchmarks derived from the MiBench [41] and Embench [42] suites, with *hundreds of millions of dynamic instructions* each, as shown in Table II. To facilitate the analysis of extensive workloads, particularly in the context of RTL evaluation studies, which typically focus on workloads of less than a few million instructions due to prohibitively slow simulation times and trace file sizes (as observed in studies like [14]), we introduce and employ the SimPoint methodology in Chipyard. By employing SimPoints, we completed our experiments in slightly over 2 days, achieving a 45× speedup compared to the over 3-month execution time that would have been necessary without considering trace file size and processing time constraints.

As shown in Fig. 4 integrating SimPoints into our workflow involves utilizing the gem5 [15] simulator. This simulator is leveraged to generate the essential basic block vectors (BBVs) needed by the SimPoint algorithm. These BBVs characterize each program interval (i.e., SimPoint size chunks of the program) containing information about the frequency of execution of all the static basic blocks. This program profile produced by gem5 is then input into the SimPoint executable, which calculates the SimPoints representing the different phases of the workload. For each chosen SimPoint, we create an appropriate architectural checkpoint using Spike [43], a RISC-V ISA simulator.

To load and execute the derived SimPoints, we utilize Chipyard's recently added feature of architectural checkpointing. Employing a custom version of the testbench driver, we execute the SimPoint checkpoints in the target BOOM configuration. Before executing the SimPoint code, a warm-up period is allowed to mitigate inaccuracies resulting from the cold cache memories and branch predictor. This entire process is automated and executed using a Python script. Table II provides a breakdown of information, including instruction count, SimPoint interval size, and the number of SimPoints

TABLE I
BOOM CONFIGURATIONS USED IN THIS STUDY.

uArch Parameter	Medium	Large	Mega
Fetch/Decode/Issue Width	4/2/4	8/3/5	8/4/8
Int / Mem / FP IQ Entries	20/12/16	32/16/24	40/24/32
Int / Mem / FP EXUs	2/1/1	3/1/1	4/2/2
Int Regs	80	100	128
Int RF Rd Ports	6	8	12
Int RF Wr Ports	3	4	6
FP Regs	64	96	128
FP RF Rd Ports	3	3	6
FP RF Wr Ports	2	2	4
ROB Entries	64	96	128
L1 I$ Size / Associativity	16KB / 4	32KB / 8	32KB / 8
L1 D$ Size / Associativity	16KB / 4	32KB / 8	32KB / 8
Load / Store Queue Size	16 / 16	24 / 24	32 / 32
Branch Predictor	TAGE	TAGE	TAGE
BTB Entries	256	512	512

Fig. 4. Steps to generate checkpoints based on the SimPoint algorithm results.

considered. In this study, all the results presented rely on the execution of the highest-ranked SimPoints (the specific count for each benchmark is detailed in Table II). These SimPoints provide at least 90% program coverage ensuring high accuracy. The SimPoint size to program dynamic instructions ratio we use in this study is on average 1:300 compared to 1:20000 in prior studies analyzing the SPEC2006 benchmarks [44]–[46]. The larger the ratio, the less SimPoints are needed to reach adequate coverage. However, our workflow is entirely configurable and capable of accommodating any quantity and scale of SimPoints.

B. Power Consumption Analysis per Component

Fig. 5, Fig. 6 and Fig. 7 show the power consumption of each of the 13 hardware components considered in this study across all eleven workloads for the three BOOM configurations, as described in Section IV-A. As shown in Fig. 7, the *Integer Register File* is the second largest contributor of power in the MegaBOOM configuration, having the highest consumption of any component in the *Sha* benchmark. Its average power consumption across all workloads is 4.83mW, which accounts for 12% of the total BOOM tile power (the BOOM tile contains the core and the L1 cache memories). However, as shown in Fig. 5, Fig. 6 (i.e., the MediumBOOM and the LargeBOOM, respectively), the power of the Integer Register File is significantly lower than the MegaBOOM configuration. In the LargeBOOM configuration (see Fig. 6), the power usage is 0.72mW on average, which is only 2.95% of the total BOOM tile power. Similarly, in the MediumBOOM configuration (see Fig. 5), the power consumption of the Integer Register File is further reduced to nearly 0.27mW or 2% of the BOOM tile power. The higher power consumption in the MegaBOOM configuration can be attributed to two main factors: (1) the higher number of physical integer registers in the MegaBOOM configuration and (2) the bypass network, which exhibits non-linear growth based on the number of read and write ports. In the MegaBOOM configuration, there are 12 read ports and 6 write ports, resulting in a larger bypass network. In comparison, the LargeBOOM configuration has 8 read ports and 4 write ports, while the MediumBOOM configuration has 6 read ports and 3 write ports (see Table I).

TABLE II
BENCHMARK INSTRUCTIONS, SIZE & NUMBER OF SIMPOINTS.

Benchmark	Suite	Interval	# Simpoints	Instructions
Basicmath (Bmath)	MiBench	1M	2	364,758,047
Stringsearch (StSear)		1M	2	136,360,766
FFT		1M	1	266,217,322
iFFT		1M	1	266,643,273
Bitcount (BC)		1M	3	495,204,057
Qsort		1M	1	22,868,929
Dijkstra (Dijk)		1M	3	227,879,044
Patricia (Patr)		2M	2	154,589,629
Matmult (MxM)	Embench	1M	1	516,885,284
Sha		1M	3	111,029,722
Tarfind		2M	1	1,220,430,895

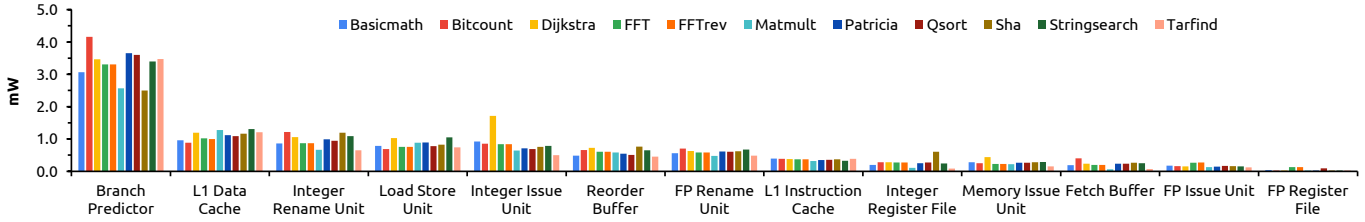

Fig. 5. Power consumption per hardware structure for the MediumBoom across all eleven benchmarks.

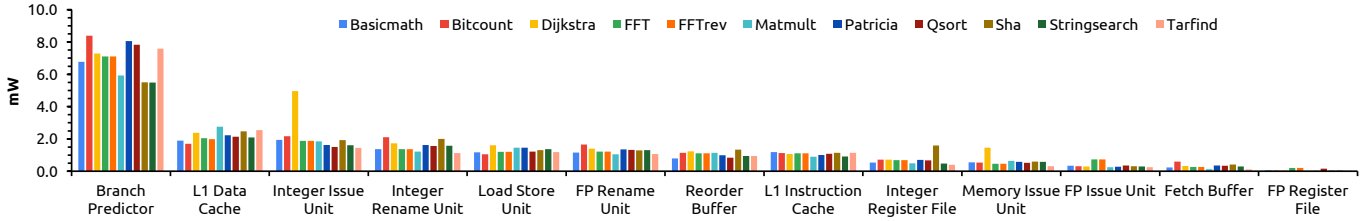

Fig. 6. Power consumption per hardware structure for the LargeBoom across all eleven benchmarks.

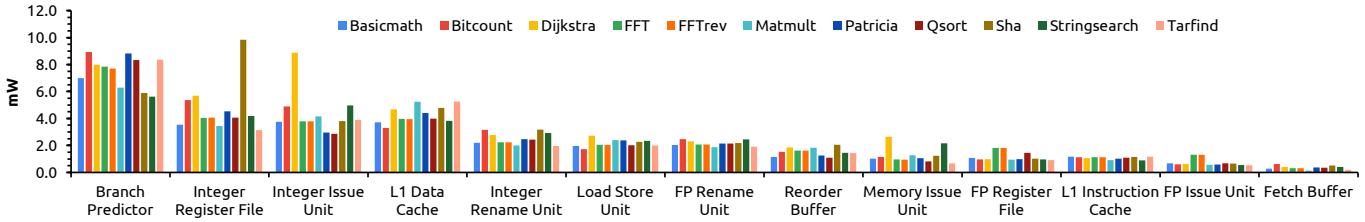

Fig. 7. Power consumption per hardware structure for the MegaBoom across all eleven benchmarks.

Key Takeaway #1: The power consumption of the Integer Register File varies significantly across different BOOM configurations (MediumBOOM, LargeBOOM, and MegaBOOM). This is attributed to the varying number of registers but *more importantly* the varying number of read and write ports that non-linearly affect the size of the bypass network.

Regarding workload sensitivity, it is evident that the power consumption of the integer register file exhibits considerable variation across the eleven benchmarks. The IRF power consumption is highly correlated with IPC. The Sha benchmark that achieves the highest IPC across the 3 configurations has the highest IRF power consumption by a large margin.

The *Floating Point Register File* demonstrates a deficient power consumption across all benchmarks, except for *FFT*, *iFFT* and *Qsort* in both the MediumBOOM and LargeBOOM configurations. On average, it consumes 0.05mW and 0.08mW, accounting for only 0.4% and 0.3% of overall power consumption respectively. This is naturally expected since no other benchmarks utilize FP registers. However, in the case of MegaBOOM, as illustrated in Fig. 7, substantial power consumption in the floating point register file can be observed across all benchmarks (1.18mW on average), with an expected surge at the aforementioned FP-intensive workloads. This power, on FP-free workloads, almost entirely stems from static logic power and can be attributed to the large bypass network resulting from the increased access ports in the MegaBOOM configuration, as shown in Table I.

Key Takeaway #2: The FP register file has very low power consumption in the two smaller BOOM configurations with expected spikes in FP-heavy workloads. In MegaBOOM, in contrast, static power consumption significantly increases. This is due to the extended ($2\times$) number of read and write ports.

The *Integer Rename Unit* ranks as the fifth-largest contributor to power consumption in the MegaBOOM configuration, the fourth-largest in LargeBOOM and the third-largest in MediumBOOM. On average, across all the workloads, it consumes 0.95mW in MediumBOOM, 1.57mW in LargeBOOM, and 2.5mW in MegaBOOM. These values correspond to 7.2%, 6.4%, and 6.2% of the total power consumed by the BOOM tile, respectively. Its power consumption highly depends on the workload being executed, with the *Sha* and *Bitcount* consistently exhibiting the highest power consumption across all three BOOM configurations. This is anticipated because these two benchmarks exhibit the highest IPC, thereby exerting considerable stress on the integer pipeline.

The *Floating Point Rename Unit* plays a significant role in power consumption across all BOOM configurations. On average, it consumes 0.6mW in MediumBOOM, 1.29mW in LargeBOOM, and 2.16mW in MegaBOOM. These values represent 4.5%, 5.3%, and 5.4% of the total power consumed by the BOOM tile, respectively. Surprisingly, although only three benchmarks utilize FP registers, the Floating Point Rename Unit does not follow the same power consumption pattern as the FP Register File, described earlier. This may be attributed to the creation of a new allocation list initialized to zero for

every executed branch instruction [34]. The allocation list is a new copy of the free list, so upon a branch misprediction, the processor can revert the state of the free list. Consequently, constant register writing occurs throughout program execution, even when no floating-point instructions are being executed.

Key Takeaway #3: Even with minimal use of FP registers, the Floating Point Rename Unit shows a surprisingly high power consumption trend across various BOOM configurations. This mismatch between structure usage and power identifies an opportunity for redesigning the allocation and initialization mechanisms within the unit. This might involve exploring more efficient ways to manage the allocation list, possibly by minimizing the constant register writing when no floating-point instructions are executed.

BOOM employs a *three-way distributed scheduler* design, where different units are responsible for issuing different types of instructions. The Integer Issue Unit handles integer instructions, the Memory Issue Unit handles memory instructions, and the Floating Point Issue Unit handles FP instructions.

Among these units, the *Integer Issue Unit* exhibits the highest power consumption across all BOOM configurations, as shown in Fig. 5, Fig. 6 and Fig. 7. On average for all benchmarks, the Integer Issue Unit consumes 0.83mW in MediumBOOM, 2.08mW in LargeBOOM, and 4.4mW in MegaBOOM. These values correspond to 6.3%, 8.6%, and 10.9% of the total power consumed by the BOOM tile, respectively. The Integer Issue Unit demonstrates significant workload sensitivity, as its power consumption is highly correlated with queue occupancy. Programs with high integer ILP (i.e., many integer instructions that do not have dependencies with each other) will result in lower issue queue occupancy since integer instructions won't have to wait to be executed and thus issued immediately to the execution units. While the issue rate will be higher most of the issue queue entries will be empty minimizing power consumption of the issue select and wake-up logic.

A clear illustration of this behavior is observed when comparing the *Dijkstra* and *Sha* workloads. *Dijkstra*, despite having lower IPC, exhibits much higher power consumption in the Integer Issue Unit compared to *Sha*. Conversely, *Sha* achieves the highest IPC out of all workloads but demonstrates lower power consumption in the Integer Issue Unit. To better demonstrate queue occupancy, we present fine-grained power measurements for each integer issue queue slot for the two workloads in Figure 8. In *Dijkstra*, all issue slots show notable power consumption, indicating high queue occupancy, whereas in *Sha*, only a few slots exhibit significant power consumption (i.e., the queue remains mostly empty with few slots being utilized systematically).

The *Memory Issue Unit* represents the component with the second-highest power consumption among the distributed scheduler units. On average for all benchmarks, it consumes 0.26mW in MediumBOOM, 0.62mW in LargeBOOM, and 1.3mW in MegaBOOM. This accounts for 2%, 2.5%, and 3.2% of the total BOOM tile power consumption, respectively. For the same reason as the Integer Issue Unit, the Memory Issue Unit also exhibits high sensitivity to different workloads. *Dijkstra* and *Stringsearch* consistently demonstrate the highest power consumption across all three BOOM configurations. These observations suggest that due to a higher proportion of memory instructions, these two benchmarks significantly drive the power consumption of the Memory Issue Unit.

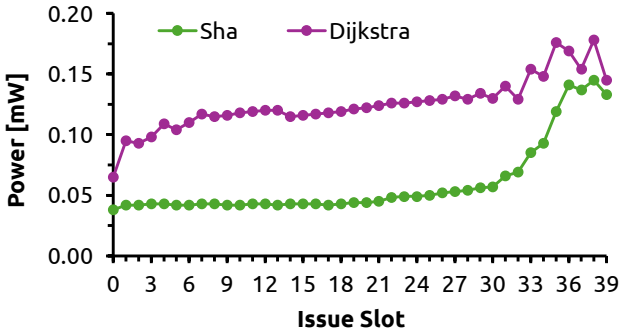

Fig. 8. Power consumption per issue slot of *Dijkstra* and *Sha* benchmarks for the MegaBOOM configuration. Each dot represents one of the 40 integer issue slots of the integer issue unit.

Key Takeaway #4: Among the three distributed scheduler units, the Integer Issue Unit stands out as the highest power consumer. Collectively, these scheduler units represent the *second-highest power consumption*, trailing only behind the branch predictor across all three configurations. This positions them as a primary focus for optimization efforts.

The *Floating Point Issue Unit* consumes the lowest power among the distributed scheduler units. On average for all benchmarks, it consumes 0.17mW in MediumBOOM, 0.39mW in LargeBOOM, and 0.74mW in MegaBOOM configurations. These values represent a small portion of the overall power consumed by the BOOM tile. However, for the *FFT*, *iFFT* workloads, in which there is significant floating-point activity, we observe higher power consumption for the Floating Point Issue Unit compared to all other benchmarks. Specifically, in both the *FFT* and *iFFT* workloads, the power consumption is as high as 0.27mW in MediumBOOM, 0.74mW in LargeBOOM, and 1.3mW in MegaBOOM.

Key Takeaway #5: The deployment of collapsing queues in the BOOM issue units [47] enhances queue utilization but sacrifices energy efficiency due to frequent register writes per cycle (as presented in [48]). Analyzing the performance-power trade-offs across different implementations could potentially enhance the energy efficiency of the BOOM distributed scheduler.

The *Reorder Buffer (ROB)* plays a crucial role in out-of-order CPUs, including BOOM. BOOM implements register renaming with a merged register file [49], which is the typical

implementation in most modern CPUs [50]–[52]. Therefore, instruction data is stored in the register file rather than in the ROB, which makes the ROB size significantly smaller. This architectural decision is reflected in the power consumption of this reduced-size ROB. On average for all benchmarks, the ROB consumes 0.61mW in MediumBOOM, 1.08mW in LargeBOOM, and 1.57mW in MegaBOOM configurations. These values account for 4.6%, 4.4%, and 3.9% of the total power consumed by the BOOM tile, respectively. In terms of workload sensitivity, the power consumption of the ROB varies significantly across different benchmarks, although not to the same extent as the issue units. The workload-specific behavior remains consistent across all three BOOM configurations.

Key Takeaway #6: Introducing adaptive ROB sizing changes (along the lines of [48], [53]) based on workload characteristics can help optimize power consumption. For instance, workloads with more or longer instruction dependencies might benefit from a slightly larger ROB size to avoid stalls, thereby improving performance without substantially increasing power consumption.

The *Branch Predictor* is the primary contributor to power consumption across all three BOOM configurations. It averages 3.34mW in MediumBOOM, 7mW in LargeBOOM, and 7.6mW in MegaBOOM. This accounts for 25.3%, 28.8%, and 18.8% of the total power consumption per BOOM tile, respectively. As shown in Fig. 6 and Fig. 7, the branch predictor exhibits similar average power consumption across benchmarks in these two BOOM configurations. This similarity arises from both configurations having Branch Target Buffers (BTBs) of the same size, while MediumBOOM utilizes a BTB half the size (see Table I). Notably, the predictor's power consumption significantly varies depending on the executed workload, resulting in substantial differences among benchmarks. Compared to a prior study utilizing the Gshare branch predictor [14], TAGE consumes an average of 2.5× more power across all BOOM configurations.

Key Takeaway #7: The branch predictor stands out as the primary contributor to power consumption across all three BOOM configurations. Compared to previous studies [14] using the Gshare predictor, TAGE consumes on average *2.5*× more power. This makes the BP a prime target for optimization and warrants a detailed study of the performance-power trade-off between TAGE and GShare in BOOM.

The power consumption of the *Fetch Buffer* is relatively low. On average for all benchmarks, it consumes 0.22mW in MediumBOOM, 0.31mW in LargeBOOM, and 0.36mW in MegaBOOM. The Fetch Buffer serves as a container for instructions retrieved from the instruction cache. During the Decode Stage, instructions are retrieved from the Fetch Buffer and undergo decoding. Subsequently, the necessary resources are allocated for each instruction. If certain required resources are unavailable, a stall occurs, temporarily halting the process until the required resources become available. We believe that the power consumption of the Fetch Buffer is sensitive to the workload being executed due to these stalls.

The *Load Store Unit (LSU)* is a significant contributor to the total BOOM tile power consumption. In MediumBOOM, on average across all benchmarks, it consumes 0.84mW, in LargeBOOM 1.3mW and MegaBOOM 2.2mW. This accounts for 6.3% of the total BOOM tile power consumption in MediumBOOM and 5.4% in both LargeBOOM and MegaBOOM. The LSU exhibits very high workload sensitivity similar to that of the issue units, with the workload-dependent behavior being consistent across the three BOOM configurations.

The *L1 Data Cache* emerges as a significant contributor to power consumption in all three BOOM configurations, ranking as the second-highest contributor. It averages 1.13mW in MediumBOOM, 2.24mW in LargeBOOM, and 4.34mW in MegaBOOM, accounting for 8.5%, 9.2%, and 10.8% of the total power consumed by the BOOM tile, respectively. Notably, despite the identical data cache size and associativity in the LargeBOOM (Fig. 6) and MegaBOOM (Fig. 7) configurations, they exhibit distinct power consumption levels. This difference is attributed to the MegaBOOM configuration's inclusion of two memory execution units, facilitating more concurrent data cache accesses and doubling the Miss Status Handling Registers (MSHRs) compared to LargeBOOM. The data cache's power consumption strongly correlates with the executed workload, directly influenced by the access pattern specific to each benchmark. This workload-driven behavior remains consistent across all three BOOM configurations. Remarkably, the *Matmult* and *Tarfind* benchmarks demonstrate the highest power consumption in relation to the data cache.

Key Takeaway #8: While additional memory execution units and expanded MSHRs may enhance performance by enabling a larger number of concurrent cache accesses, they come with a trade-off in increased power. Architects might need to explore ways to fine-tune or dynamically manage the size of these resources based on workload characteristics to reduce power consumption without compromising performance.

The *L1 Instruction Cache*, on average across the benchmarks, consumes 0.36mW in MediumBOOM, while for LargeBOOM and MegaBOOM configurations, it consumes 1.06mW. These values represent 2.7%, 4.3%, and 2.6% of the total power consumption of the BOOM tile, respectively. Interestingly, the power consumption of the instruction cache remains almost identical in both the LargeBOOM and MediumBOOM configurations. This similarity may be attributed to the caches having the same size, associativity, and number of MSHRs (Miss Status Handling Registers). The L1 instruction cache is the least impacted component by the workload being executed. Across all configurations, the variation in power consumption between different benchmarks is negligible. This may be attributed to the cache's consistent and regular access pattern,

receiving reads every cycle of program execution. The fetch buffer plays a significant role in this by decoupling instruction fetching from execution. This contributes to a constant flow of instructions and maintains the cache's regular access pattern.

Highlighting the specific components that consume the most power in a CPU (like Branch Prediction and Integer Rename Units) provides clear targets for optimization. Future designs could focus on these areas to improve energy efficiency without compromising performance. Understanding which components significantly contribute to power consumption enables designers to make informed trade-offs. For instance, they might choose to optimize specific units while accepting a marginal increase in power consumption in others, depending on the overall performance gains achieved.

C. Contribution of Microarchitectural Components

Fig. 9 shows the power consumption percentage of the components we considered for each of the three configurations of BOOM that were analyzed. In the case of MegaBOOM, the considered components consume 85% of the total power of the BOOM tile. The BOOM tile comprises the BOOM core and L1 caches. For the LargeBOOM and the MediumBOOM configurations, this percentage decreases to 81% and 73%, respectively. This decrease can be attributed to the fact that most of the components we considered, such as the register files, issue units, and ROB, undergo size variation in these configurations. In contrast, the remaining components (Execution Units, Decoding logic, Fetch Target Queue, etc.) of the tile experience considerably smaller variation. The observation we made provides valuable insight: the BOOM tile predominantly consumes power through 13 specific components. Enhancements in the power efficiency of these components can have a significant positive impact on its overall power efficiency.

D. Instructions per Cycle (IPC)

IPC denotes the number of instructions executed or completed per clock cycle, directly impacting CPU efficiency and speed. Enhancing IPC is pivotal for CPU designers, enhancing overall performance by executing more instructions in parallel. It crucially impacts single-threaded task performance by enabling faster software execution. IPC varies based on workload and applications, impacted by diverse instruction latencies and dependencies. The complexity of instructions and microarchitecture efficiency determine their impact on achieved IPC. To this end, in this subsection, we show the IPC, calculated using the SimPoint methodology, for all benchmarks and BOOM configurations used in this study. Fig. 10 shows the IPC

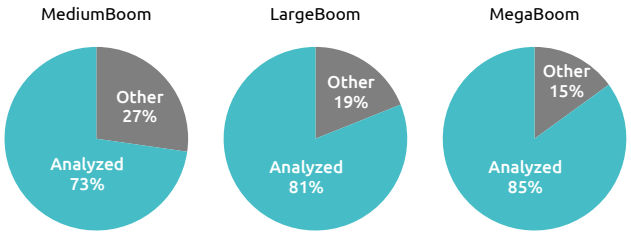

Fig. 9. Analyzed component contribution to the full chip for all configurations.

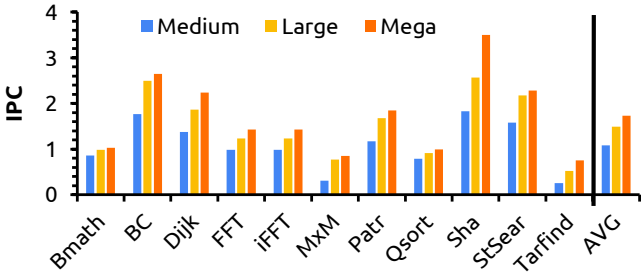

Fig. 10. Instructions per Cycle (IPC) for each benchmark (x-axis) and all three BOOM configurations (MegaBoom, LargeBOOM, MediumBOOM).

we measured across all 11 of our benchmarks for the three different BOOM configurations we considered. *Tarfind* has the lowest IPC in all three configurations, and *Sha* has the highest. The *Sha* benchmark has a lot of inherent ILP and can utilize the issue width of all three designs efficiently. We can see that the IPC is 1.83, 2.6, and 3.5 in MediumBOOM, LargeBOOM and MegaBOOM configurations. This is close to the theoretical maximums these BOOM configurations could reach since they have a decode width of 2, 3 and 4 respectively. While IPC is a good metric of performance, nowadays performance per watt (i.e., energy efficiency) is becoming much more important.

E. Performance per Watt

Performance per watt is a metric used to measure the efficiency of a system or device by assessing its performance in relation to the power it consumes. It is commonly used in the context of computer systems, processors, graphics cards, and other electronic devices where energy efficiency is important. Typically expressed as a ratio or a figure of merit, such as instructions per cycle per watt (IPC/W), these measurements illustrate the amount of work or computational capability achievable per unit of electrical power consumed. The importance of performance per watt has increased significantly in recent years due to the amplified demand for energy-efficient computing systems. With power consumption emerging as a limiting factor in numerous applications, enhancing performance per watt has become a pivotal objective for hardware manufacturers. Fig. 11 shows the performance-per-watt values for all BOOM configurations and benchmarks. Notably, in 8 out of 11 benchmarks, the MediumBOOM configuration excels. However, in the *Matmult*, *Stringsearch*, and *Tarfind* benchmarks, LargeBOOM takes precedence. While boasting the best absolute performance, MegaBOOM significantly sacrifices more power to achieve it.

V. Related Work

Power consumption studies: Natarajan *et al.* in [54] employ a performance and power model specific to the Alpha 21264 processor, aiming to evaluate the energy and identify any inefficiencies caused by misspeculation and over-provisioning. They found that flushed instructions contribute to around 6% of the overall energy consumption, whereas over-provisioning imposes an average tax of 17%. Tiwari *et al.* in [55] present an overview of power consumption

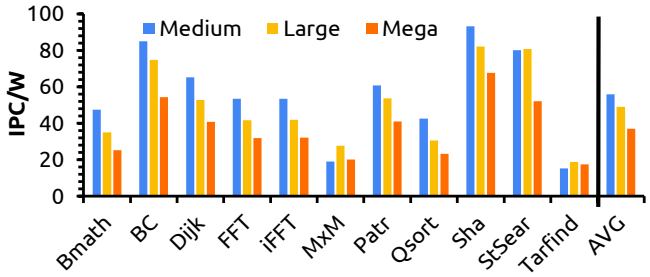

Fig. 11. Performance per Watt for each benchmark (x-axis) and all three BOOM configurations (MegaBoom, LargeBOOM, MediumBOOM).

concerns specifically related to Intel CPUs. They outline the primary factors contributing to the growing emphasis on low-power design and briefly discuss system and benchmarking considerations, as well as the sources of power consumption in high-performance CPUs. They also highlight previously implemented techniques on real designs. Brooks *et al.* in [1] present the utilization of energy-enabled performance simulators during the initial stages of design and explore the emerging paradigms in processor design to provide insights into their inherent power-performance characteristics.

Power modeling and simulation tools: Power modeling and simulation tools play a crucial role in understanding and evaluating the power consumption of microarchitectures. Zhai *et al.* in [17] present McPAT-Calib, a microarchitecture power modeling framework for modern CPUs, which combines internal improvements and ML calibration to McPAT [56] to enhance (1) modeling accuracy and (2) the lack of support for advanced technology nodes of McPAT. Varma *et al.* in [57] developed a power estimation tool that is efficient, precise, and applicable to various performance estimation frameworks. They propose a methodology designed for system designers who are constructing systems using established components such as processors, caches, buses, peripherals, etc. Zaman *et al.* in [58] use SimPoints in a cross-layer power-estimation framework. They do not perform design space exploration and do not use highly advanced power-estimation tools like Cadence Joules but rely on McPat for their power measurements. Finally, Xu *et al.* in [59] use SimPoints in RTL simulations but for the purposes of performance analysis.

VI. Conclusion

We presented an in-depth exploration of the impact of microarchitectural configurations on energy efficiency (i.e., power consumption and performance) using the state-of-the-art RISC-V based out-of-order CPU design (SonicBOOM). Through our RT-level investigation, we demonstrate how microarchitectural design choices significantly influence power consumption profiles. By assessing the power consumption of three different configurations of the SonicBOOM design and evaluating the effects of thirteen major microarchitectural components, we gained insights into the relationship between microarchitecture and power consumption. Furthermore, our performance per watt evaluations showed that, despite providing lower performance, smaller RISC-V designs exhibited higher performance per watt ratios (i.e., are more energy efficient). These findings have important implications for system designers and computer architects, as they can now make informed decisions to optimize energy efficiency in modern microarchitectures. Our extension of Chipyard's infrastructure to support SimPoints can benefit the entire community by enabling the performance and power analysis of vastly larger workloads than ever before.

Acknowledgments

We want to thank the anonymous reviewers for their useful comments. This work was supported by research gifts from Intel and AMD, as well as the European Union's Horizon Europe research and innovation programme under grant agreements No 101097224 (REBECCA), No 101093062 (Vitamin-V), and No 101070238 (NEUROPULS). Views and opinions expressed are however, those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. This work was also supported by the Hellenic Foundation Research and Innovation (HFRI) project VEMER.

References

[1] D. Brooks, P. Bose, S. Schuster, H. Jacobson, P. Kudva, A. Buyuktosunoglu, J. Wellman, V. Zyuban, M. Gupta, and P. Cook, "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors," *IEEE Micro*, vol. 20, no. 6, pp. 26–44, 2000.

[2] P. Koutsovasilis, C. D. Antonopoulos, N. Bellas, S. Lalis, G. Papadimitriou, A. Chatzidimitriou, and D. Gizopoulos, "The impact of cpu voltage margins on power-constrained execution," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 1, pp. 221–234, 2022.

[3] G. Papadimitriou and D. Gizopoulos, "Characterizing soft error vulnerability of cpus across compiler optimizations and microarchitectures," in *2021 IEEE International Symposium on Workload Characterization (IISWC)*, 2021, pp. 113–124.

[4] D. Gizopoulos, G. Papadimitriou, A. Chatzidimitriou, V. J. Reddi, B. Salami, O. S. Unsal, A. C. Kestelman, and J. Leng, "Modern hardware margins: Cpus, gpus, fpgas recent system-level studies," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2019, pp. 129–134.

[5] G. Papadimitriou, A. Chatzidimitriou, D. Gizopoulos, V. J. Reddi, J. Leng, B. Salami, O. S. Unsal, and A. C. Kestelman, "Exceeding conservative limits: A consolidated analysis on modern hardware margins," *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 2, pp. 341–350, 2020.

[6] Khubaib, M. A. Suleman, M. Hashemi, C. Wilkerson, and Y. N. Patt, "Morphcore: An energy-efficient microarchitecture for high performance ilp and high throughput tlp," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, 2012, pp. 305–316.

[7] A. Gamatie, G. Devic, G. Sassatelli, S. Bernabovi, P. Naudin, and M. Chapman, "Towards energy-efficient heterogeneous multicore architectures for edge computing," *IEEE Access*, vol. 7, pp. 49 474–49 491, 2019.

[8] S. Shankar and A. Reuther, "Trends in energy estimates for computing in ai/machine learning accelerators, supercomputers, and compute-intensive applications," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, 2022, pp. 1–8.

[9] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: a new paradigm for power estimation," in *Proceedings. 42nd Design Automation Conference, 2005.*, 2005, pp. 700–705.

[10] A. Radovanovic, B. Chen, S. Talukdar, B. Roy, A. Duarte, and M. Shahbazi, "Power modeling for effective datacenter planning and compute management," *IEEE Transactions on Smart Grid*, vol. 13, no. 2, pp. 1611–1621, 2022.

[11] J. Zhao, B. Korpan, A. Gonzalez, and K. Asanovic, "Sonicboom: The 3rd generation berkeley out-of-order machine," in *Fourth Workshop on Computer Architecture Research with RISC-V*, vol. 5, 2020.

[12] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton *et al.*, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *IEEE Micro*, vol. 40, no. 4, pp. 10–21, 2020.

[13] 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.

[14] O. Chatzopoulos, G. Papadimitriou, W. S. Wong, and D. Gizopoulos, "Energy efficiency of out-of-order cpus: Comparative study and microarchitectural hotspot characterization of risc-v designs," in *2023 IEEE International Symposium on Workload Characterization (IISWC)*, 2023, pp. 216–220.

[15] J. Lowe-Power, A. M. Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen, B. Beckmann, S. Bharadwaj, G. Black, G. Bloom, B. R. Bruce, D. R. Carvalho, J. Castrillon, L. Chen, N. Derumigny, S. Diestelhorst, W. Elsasser, C. Escuin, M. Fariborz, A. Farmahini-Farahani, P. Fotouhi, R. Gambord, J. Gandhi, D. Gope, T. Grass, A. Gutierrez, B. Hanindhito, A. Hansson, S. Haria, A. Harris, T. Hayes, A. Herrera, M. Horsnell, S. A. R. Jafri, R. Jagtap, H. Jang, R. Jeyapaul, T. M. Jones, M. Jung, S. Kannoth, H. Khaleghzadeh, Y. Kodama, T. Krishna, T. Marinelli, C. Menard, A. Mondelli, M. Moreto, T. Mück, O. Naji, K. Nathella, H. Nguyen, N. Nikoleris, L. E. Olson, M. Orr, B. Pham, P. Prieto, T. Reddy, A. Roelke, M. Samani, A. Sandberg, J. Setoain, B. Shingarov, M. D. Sinclair, T. Ta, R. Thakur, G. Travaglini, M. Upton, N. Vaish, I. Vougioukas, W. Wang, Z. Wang, N. Wehn, C. Weis, D. A. Wood, H. Yoon, and Éder F. Zulian, "The gem5 simulator: Version 20.0+," 2020. [Online]. Available: https://arxiv.org/abs/2007.03152

[16] 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 *Proceedings of the 42nd annual ieee/acm international symposium on microarchitecture*, 2009, pp. 469–480.

[17] J. Zhai, C. Bai, B. Zhu, Y. Cai, Q. Zhou, and B. Yu, "Mcpat-calib: A risc-v boom microarchitecture power modeling framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 1, pp. 243–256, 2022.

[18] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP laboratories*, vol. 27, p. 28, 2009.

[19] S. Rogers, J. Slycord, M. Baharani, and H. Tabkhi, "gem5-salam: A system architecture for llvm-based accelerator modeling," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 471–482.

[20] O. Chatzopoulos, G. Papadimitriou, V. Karakostas, and D. Gizopoulos, "Enabling design space exploration of risc-v accelerator-rich computing systems on gem5," *RISC-V Summit Europr 2023*, June 2023. [Online]. Available: https://riscv-europe.org/media/proceedings/posters/2023-06-08-Odysseas-CHATZOPOULOS-abstract.pdf

[21] H. Ranjitha, S. Hiremath, and S. G. Langadi, "Rtl power estimation: Early design cycle approach for soc power sign-off," in *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2018, pp. 480–484.

[22] Cadence, "Cadence joules," https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/power-analysis/joules-rtl-power-solution.html, accessed: 2023-12-06.

[23] Synopsys, "Synopsys primepower," https://www.synopsys.com/implementation-and-signoff/signoff/primepower.html, accessed: 2023-12-06.

[24] Siemens, "Siemens powerpro," https://eda.sw.siemens.com/en-US/ic/powerpro/, accessed: 2023-12-06.

[25] D. Thomas and P. Moorby, *The Verilog® hardware description language*. Springer Science & Business Media, 2008.

[26] S. Sutherland, S. Davidmann, and P. Flake, *SystemVerilog for Design Second Edition: A Guide to Using SystemVerilog for Hardware Design and Modeling*. Springer Science & Business Media, 2006.

[27] P. J. Ashenden, *The designer's guide to VHDL*. Morgan Kaufmann, 2010.

[28] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.

[29] S. Karandikar, H. Mao, D. Kim, D. Biancolin, A. Amid, D. Lee, N. Pemberton, E. Amaro, C. Schmidt, A. Chopra *et al.*, "Firesim: Fpga-accelerated cycle-exact scale-out system simulation in the public cloud," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 29–42.

[30] 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 *Proceedings of the 49th Annual Design Automation Conference*, 2012, pp. 1216–1225.

[31] E. Wang, "Hammer: A platform for agile physical design," *Master's thesis, EECS Dept, UC Berkeley*, 2018.

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

[33] K. Asanovic, D. A. Patterson, and C. Celio, "The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor," University of California at Berkeley Berkeley United States, Tech. Rep., 2015.

[34] T. R. of the University of California, "RISCV-BOOM Core Documentation," https://docs.boom-core.org/en/latest/index.html, accessed: 2023-12-06.

[35] W. Snyder, "Verilator: Open simulation-growing up," *DVClub Bristol*, 2013.

[36] ——, "Verilator 4.0: open simulation goes multithreaded," in *Open Source Digital Design Conference (ORConf)*, 2018.

[37] J. Balkind, K. Lim, F. Gao, J. Tu, D. Wentzlaff, M. Schaffner, F. Zaruba, and L. Benini, "Openpiton+ ariane: The first open-source, smp linux-booting risc-v system scaling from one to many cores," in *Workshop on Computer Architecture Research with RISC-V (CARRV)*, 2019, pp. 1–6.

[38] K. Asanovic, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz *et al.*, "The rocket chip generator," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, vol. 4, 2016.

[39] E. Perelman, G. Hamerly, M. Van Biesbrouck, T. Sherwood, and B. Calder, "Using simpoint for accurate and efficient simulation," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, pp. 318–319, 2003.

[40] J. Zhai, C. Bai, B. Zhu, Y. Cai, Q. Zhou, and B. Yu, "Mcpat-calib: A microarchitecture power modeling framework for modern cpus," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–9.

[41] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538)*. IEEE, 2001, pp. 3–14.

[42] [Online]. Available: https://www.embench.org/

[43] B. Keller, "Risc-v, spike, and the rocket core," *Berkeley Architecture Group*, 2013.

[44] A. A. Nair and L. K. John, "Simulation points for spec cpu 2006," in *2008 IEEE International Conference on Computer Design*, 2008, pp. 397–403.

[45] 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 *Computer Performance Evaluation and Benchmarking: SPEC Benchmark Workshop 2009, Austin, TX, USA, January 25, 2009. Proceedings*. Springer, 2009, pp. 121–137.

[46] M. Choi, T. Fukuda, M. Goshima, and S. Sakai, "An inductive method to select simulation points," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 12, pp. 2891–2900, 2016.

[47] T. R. of the University of California, "RISCV-BOOM Core Documentation: The Issue Unit," https://docs.boom-core.org/en/latest/sections/issue-units.html, accessed: 2023-12-06.

[48] D. Folegnani and A. Gonzalez, "Energy-effective issue logic," in *Proceedings 28th Annual International Symposium on Computer Architecture*, 2001, pp. 230–239.

[49] A. Gonzalez, F. Latorre, and G. Magklis, *Processor Microarchitecture: An Implementation Perspective*. Morgan & Claypool Publishers, 2010.

[50] J. Doweck, W.-F. Kao, A. K.-y. Lu, J. Mandelblat, A. Rahatekar, L. Rappoport, E. Rotem, A. Yasin, and A. Yoaz, "Inside 6th-generation intel core: New microarchitecture code-named skylake," *IEEE Micro*, vol. 37, no. 2, pp. 52–62, 2017.

[51] K. Yeager, "The mips r10000 superscalar microprocessor," *IEEE Micro*, vol. 16, no. 2, pp. 28–41, 1996.

[52] M. Clark, "A new ×86 core architecture for the next generation of computing," in *2016 IEEE Hot Chips 28 Symposium (HCS)*, 2016, pp. 1–19.

[53] M. Choi, J. H. Park, and Y.-S. Jeong, "Revisiting reorder buffer architecture for next generation high performance computing," *J. Supercomput.*, vol. 65, no. 2, p. 484–495, aug 2013. [Online]. Available: https://doi.org/10.1007/s11227-011-0734-x

[54] K. Natarajan, H. Hanson, S. Keckler, C. Moore, and D. Burger, "Microprocessor pipeline energy analysis," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design, 2003. ISLPED '03.*, 2003, pp. 282–287.

[55] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, "Reducing power in high-performance microprocessors," in *Proceedings of the 35th Annual Design Automation Conference*, ser. DAC '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 732–737. [Online]. Available: https://doi.org/10.1145/277044.277227

[56] 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 *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42. New York, NY, USA: Association for Computing Machinery, 2009, p. 469–480. [Online]. Available: https://doi.org/10.1145/1669112.1669172

[57] A. Varma, E. Debes, I. Kozintsev, P. Klein, and B. Jacob, "Accurate and fast system-level power modeling: An xscale-based case study," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, may 2008. [Online]. Available: https://doi.org/10.1145/1347375.1347378

[58] M. Zaman, M. M. Shihab, A. K. Coskun, and Y. Makris, "Cape: A cross-layer framework for accurate microprocessor power estimation," *Integration*, vol. 68, pp. 87–98, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926018305376

[59] Y. Xu, Z. Yu, D. Tang, G. Chen, L. Chen, L. Gou, Y. Jin, Q. Li, X. Li, Z. Li, J. Lin, T. Liu, Z. Liu, J. Tan, H. Wang, H. Wang, K. Wang, C. Zhang, F. Zhang, L. Zhang, Z. Zhang, Y. Zhao, Y. Zhou, Y. Zhou, J. Zou, Y. Cai, D. Huan, Z. Li, J. Zhao, Z. Chen, W. He, Q. Quan, X. Liu, S. Wang, K. Shi, N. Sun, and Y. Bao, "Towards developing high performance risc-v processors using agile methodology," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2022, pp. 1178–1199.