Blink: Fast Automated Design of Run-Time Power Monitors on FPGA-Based Computing Platforms

Andrea Galimberti
DEIB
Politecnico di Milano
Milano, Italy
andrea.galimberti@polimi.it

Michele Piccoli
DEIB
Politecnico di Milano
Milano, Italy
michele.piccoli@polimi.it

Davide Zoni
DEIB
Politecnico di Milano
Milano, Italy
davide.zoni@polimi.it

***Abstract*—The current over-provisioned heterogeneous multi-cores require effective run-time optimization strategies, and the run-time power monitoring subsystem is paramount for their success. Several state-of-the-art methodologies address the design of a run-time power monitoring infrastructure for generic computing platforms. However, the power model's training requires time-consuming gate-level simulations that, coupled with the ever-increasing complexity of the modern heterogeneous platforms, dramatically hinder the usability of such solutions. This paper introduces *Blink*, a scalable framework for the fast and automated design of run-time power monitoring infrastructures targeting computing platforms implemented on FPGA. *Blink* optimizes the time-to-solution to deliver the run-time power monitoring infrastructure by replacing traditional methodologies' gate-level simulations and power trace computations with behavioral simulations and direct power trace measurements. Applying *Blink* to multiple designs mixing a set of HLS-generated accelerators from a state-of-the-art benchmark suite demonstrates an average time-to-solution speedup of 18 times without affecting the quality of the run-time power estimates.**

***Index Terms*—run-time power monitoring, electronic design automation, FPGA**

I. Introduction

The evolution of computing platforms towards over-provisioned heterogeneous and multi-core architectures has made run-time optimizations crucial to maximize energy efficiency. Knowledge of the dynamic power consumption of the computing platform during its operation is at the core of any run-time optimization methodology, and the collection of such information is generally achieved through an ad-hoc monitoring infrastructure.

The open literature includes a variety of run-time power monitoring frameworks that augment the existing RTL description of the computing platform by instrumenting an on-chip hardware component devoted to the estimation of the dynamic power consumption of the system at run time [1]. State-of-the-art methodologies leverage gate-level simulations of the target computing platform to extract a set of estimated power traces and the corresponding switching activity that are used to identify a power model. The identified power model is then processed to generate an RTL description that meets the accuracy, temporal resolution, and area requirements and is finally instrumented into the computing platform, thus delivering the run-time power monitoring infrastructure.

On the model family side, linear regression models are most commonly used due to their low complexity and overheads while producing estimation errors below 5% [2]. Machine-learning-based solutions, on the contrary, result instead in more significant overheads due to the additional complexity without outpacing linear regression models in accuracy [3].

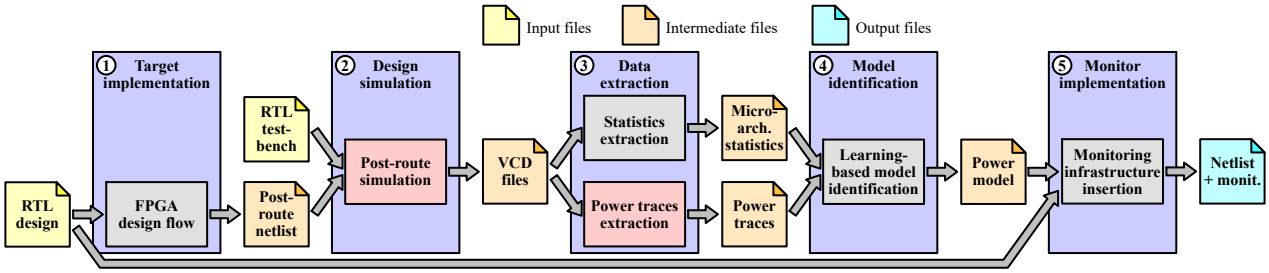
The literature includes several solutions to monitor the platform's power consumption at run time. Software-implemented power monitors leverage the relationship between power consumption and performance monitoring counters and require no microarchitectural modifications, affecting however the monitored system's performance due to monitoring being performed as a software application [4], [5]. In contrast, hardware-based ones offer high accuracy with no performance overhead by employing an ad-hoc hardware infrastructure to monitor selected signals' switching activity and deliver the power estimates, which results in area overhead [3], [6].

All the run-time power monitoring frameworks from the literature notably rely on time-consuming gate-level simulations to identify their underlying model, thus they are only suited to small computing platforms due to their lengthy execution times [1]. Conversely, the ever-increasing complexity of the computing platforms further widens the gap between the tight time-to-market deadlines and the lengthy design process, thus motivating the investigation of novel techniques.

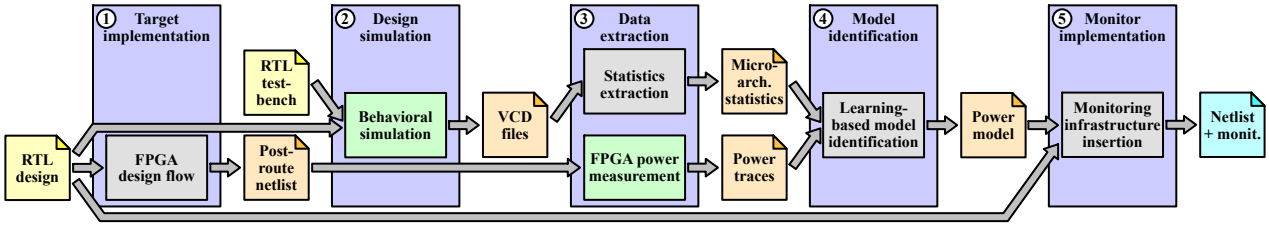
Contributions: This manuscript introduces *Blink*, a novel methodology that leverages a hybrid simulation-measurement approach to speed up the automated instrumentation of an ad-hoc run-time power monitoring infrastructure into generic computing platforms targeting deployment on FPGAs.

Blink dramatically reduces the time-to-solution, compared to state-of-the-art simulation-based methodologies, by replacing their lengthy gate-level simulations to collect the switching activity and estimated power traces with *i)* fast behavioral simulations to collect the switching activity and *ii)* direct measurements of the power traces straight from the FPGA.

Thanks to its speed, *Blink* provides a scalable framework to design and instrument run-time power monitoring infrastructures, outperforming state-of-the-art solutions by 18× on average with comparable area and power overheads and estimation accuracy in our experimental evaluation.



(a) State-of-the-art simulation-based framework [7]



(b) *Blink* framework

Fig. 1: Flowchart diagrams of the (a) state-of-the-art and (b) *Blink* frameworks, with the most time-consuming steps of the state-of-the-art framework in **red** and *Blink*'s improvements in **green**.

II. Methodology

This section describes the *Blink* methodology by highlighting the key changes to the state-of-the-art run-time power monitoring methodologies that allow it to deliver a dramatic time-to-solution speedup with comparable accuracy and overheads.

A. State-of-the-art framework

A state-of-the-art simulation-based framework [7] produces the gate-level netlist of the target computing platform enhanced with run-time power monitoring capabilities from its RTL description and the corresponding testbench. As shown in Fig. 1a, it can be split into five phases:

1) Target implementation: The standard hardware design flow implements the target computing platform by taking its RTL description and performing synthesis, place, and route to produce the corresponding gate-level netlist.

2) Design simulation: A gate-level simulation stresses each part of the post-route netlist of the target computing platform to produce a value change dump (VCD) file that contains the statistics needed to compute the power consumption trace and the corresponding switching activity. The time required by the simulation notably grows as the size of the design increases.

3) Data extraction: This phase processes the VCD file to extract the switching activity and compute the estimated power traces. The latter is notably a lengthy task whose execution time increases as the complexity of the computing platform grows and the temporal resolution of the power traces shrinks.

4) Model identification: A power model, consisting of a subset of the signals in the target computing platform and their corresponding weights, is identified by correlating the switching activity and the power trace, minimizing the distance between the actual power trace and the one produced by the identified power model subject to additional requirements, e.g., the maximum number of signals selected as model inputs.

5) Monitor implementation: A hardware monitoring infrastructure is automatically instrumented into the original target computing platform, wrapping signals identified by the model to periodically read out their switching activity and compute the power estimate and exposing the latter through a register.

B. Blink framework

The *Blink* methodology, depicted in Fig. 1b, improves the state-of-the-art frameworks by optimizing their most time-consuming phases, i.e., *design simulation* and *data extraction*. In particular, it replaces the post-route simulation and power traces extraction steps, in red in Fig. 1a, with the corresponding and significantly faster behavioral simulation and FPGA power measurement steps, in green in Fig. 1b.

FPGA power measurement: Blink substitutes the lengthy extraction of the estimated power traces in the *data extraction* phase, that takes the longest in state-of-the-art frameworks, with the direct measurement of power consumption from an FPGA prototype board that hosts the target computing platform. A bitstream is first obtained from the post-route netlist of the latter and flashed on the FPGA, then a power trace is collected directly from the FPGA protoype board through an oscilloscope while the computing platform is running. The accurate temporal alignment between the VCD obtained from the *design simulation* and the measured power trace measured is notably paramount to enabling an effective identification of the run-time power model of the target computing platform. A hardware trigger pin enables such alignment and the adoption of a delay before the actual measurement minimizes power distortion due to ringing phenomena.

Behavioral simulation: The direct measurement of the power traces, in place of their extraction from VCD files, makes it possible to also optimize the *design simulation* phase. The *Blink*

TABLE I: Accelerator designs and corresponding monitoring infrastructure. Legend: **Freq** clock frequency (MHz), **Pwr** power consumption (W), **HW** Hamming-weight and **ST** single-toggle counters, **Acc** accuracy, ✓yes, ×no.

Accelerator design configuration					FPGA resource utilization						Counters		Overhead			Acc
ID	**AES**	**Blowfish**	**GSM**	**MIPS**	**LUT**	**FF**	**BRAM**	**DSP**	**Freq**	**Pwr**	**HW**	**ST**	**LUT**	**FF**	**Pwr**	**RMSE**
A1	✓	×	×	×	13764	14674	0	0	150	0.22	4	0	2.8%	1.9%	1.6%	4.3%
A2	✓	×	×	×	24412	26170	0	0	150	0.47	5	0	2.1%	1.7%	1.2%	4.5%
A3	✓	✓	×	×	50408	72307	66.5	0	100	0.50	6	0	1.0%	0.9%	1.0%	4.1%
A4	✓	×	×	✓	41067	29802	30	160	130	0.37	7	0	2.1%	2.1%	1.8%	3.6%
A5	×	✓	×	×	45793	72098	92.5	0	100	0.32	4	2	0.5%	0.4%	0.8%	4.3%
A6	×	×	×	✓	43325	23814	30	240	120	0.24	8	0	1.2%	2.6%	2.9%	4.9%
A7	×	✓	×	✓	32931	36907	49.5	80	100	0.18	8	0	0.8%	1.0%	0.4%	4.9%
A8	×	×	✓	✓	25216	23572	0	240	100	0.19	3	1	0.7%	0.9%	0.7%	3.4%
A9	×	✓	✓	✓	48226	60573	75	240	100	0.29	5	0	0.7%	0.7%	0.6%	4.4%
A10	✓	✓	✓	✓	53500	66373	75	240	100	0.40	9	1	1.9%	1.4%	0.1%	3.9%

methodology does not require indeed a gate-level simulation of the post-route netlist of the target computing platform, but it can instead run a simpler and faster behavioral simulation of the RTL design to obtain a VCD that can be used solely for the extraction of the microarchitectural statistics, i.e., the switching activity.

The other three phases are instead the same as the state-of-the-art simulation-based framework ones, with only few minor changes to support *Blink*'s improvements. In particular, the *model identification* step takes the measured power consumption trace as an input rather than the estimated power trace extracted from the gate-level switching activity, which is equivalent from the execution time standpoint.

III. Experimental evaluation

The experimental campaign evaluates the area and power overheads and accuracy of run-time monitoring infrastructures obtained by applying *Blink* as well as the time-to-solution speedup compared to state-of-the-art simulation-based frameworks. The experiments target a benchmarking platform, depicted in Fig. 2, that instantiates a set of accelerator designs. The host PC drives the computation by interacting via UART (Rx and Tx in Fig. 2), while a trigger signal (Trg) marks the start and end of the computation for the oscilloscopes and is emulated when simulating the target computing platform so that the VCDs accurately match the measured power traces.

The accelerator architecture instantiates accelerator clusters that are enabled by a scheduler to have a variable number of active resources at a time. Each cluster contains a number of accelerators of the same type that share a memory for their input and output data. Multiple clusters in the same design can include different types of accelerators. The latter are obtained through high-level synthesis (HLS) of the *AES*, *Blowfish*, *GSM*, and *MIPS* applications from the *CHStone* benchmark suite [8].

The hardware flow targets an *AMD Artix-7 100* FPGA mounted on a *NewAE Technology CW305 Artix FPGA Target* board and the *AMD Vitis 2023.1* toolchain is employed for HLS, RTL synthesis, place-and-route, bitstream generation, and simulation. Power consumption is computed from the voltage drop measured by two *Pico Technology PicoScope 5244D* oscilloscopes across a 100mΩ shunt resistor.

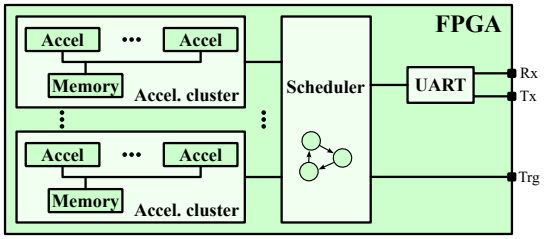


Fig. 2: Architecture of the benchmarking platform for the accelerator designs targeted by the experimental campaign.

The identification of the power model and the evaluation of power estimates obtained by the monitoring infrastructure are carried out by splitting the collected dataset according to a 80:20 ratio and targeting a 10µs temporal resolution. Model identification and monitoring implementation are performed according to the state-of-the-art simulation-based methodology selected as a reference [7] to provide the fairest comparison.

The first-order linear model takes as its inputs the switching activity of a selected subset of input and output signals of the modules in the design hierarchy and outputs the estimated power consumption. The hardware counters that monitor the selected signals are either single-toggle ones, counting any change in the target multi-bit signal, or Hamming-weight counters, that count the number of bits that toggled their values.

A. Overhead and accuracy results

We evaluate the area overhead of the monitoring infrastructures instantiated by *Blink* as the number of additional FPGA resources required to implement the hardware counters and compute the power estimate, and their accuracy as the root-mean-square error (RMSE) of the estimation. *Blink*'s measures of the power consumption of the actual target computing platform allow us to provide accurate data also with respect to the power overhead of the monitoring infrastructure, that is often neglected or inaccurate in the literature [1].

The experiments target designs with various combinations of the accelerators and clusters containing different numbers of the latter. TABLE I details the instantiated accelerators, the FPGA resource utilization, the clock frequency targeted by synthesis and place-and-route, and the maximum power consumption of

TABLE II: Execution time (in minutes) of the reference state-of-the-art and *Blink* run-time power monitoring frameworks, applied to the accelerator designs described in TABLE I. Legend: **DUT** design under test, **SOTA** state-of-the-art.

DUT	**Target impl.**		**Design sim.**		**Data extr.**		**Model ident.**		**Monitor impl.**		**Total**	
ID	**SOTA**	***Blink***	**SOTA**	***Blink***	**SOTA**	***Blink***	**SOTA**	***Blink***	**SOTA**	***Blink***	**SOTA**	***Blink***
A1	11	11	781	34	912	11	4	5	12	13	1720	74
A2	16	16	804	50	1189	16	5	4	18	17	2032	103
A3	28	28	1474	116	2461	14	3	4	29	30	3995	192
A4	33	33	1411	92	1879	16	5	5	34	34	3362	180
A5	26	26	1474	80	1911	8	4	4	27	28	3442	146
A6	19	19	268	16	305	4	5	5	21	21	618	65
A7	21	21	1131	86	1196	6	5	4	22	23	2375	140
A8	15	15	370	37	501	6	4	5	17	18	907	81
A9	27	27	1519	120	2025	8	5	4	29	28	3605	187
A10	32	32	1702	131	2061	10	5	5	34	33	3834	211
Speedup		1.00×		15.11×		150.76×		1.01×		0.99×		18.12×

ten accelerator designs. The latter target clock frequencies in a 100-150MHz range and show a maximum power consumption between 0.18W and 0.50W. TABLE I also lists the quality metrics related to the monitoring infrastructure, reporting the number of single-toggle and Hamming-weight counters that monitor the toggling activity of selected signals as well as the LUT and FF area overhead, the power overhead, and the RMSE estimation accuracy metric. The results highlight area and power overheads below 3% and an RMSE below 5%, on par with the state-of-the-art methodologies [1].

B. Time-to-solution speedup

TABLE II provides a breakdown of the execution times of the five framework phases for the application of the state-of-the-art [7] and *Blink* methodologies to the ten accelerator design instances, and its bottom row lists the average speedup for each phase as well as for their total.

Blink's execution times are notably in the order of few hours, ranging from around 1 to less than 4 hours, while the state-of-the-art methodology requires tens of hours, with execution times ranging instead from 10 to more than 66 hours. Whereas it took almost 18 days overall to obtain a monitoring-enhanced netlist for each of the accelerator designs by using the state-of-the-art flow, applying *Blink* required slightly less than 1 day, with an overall time-to-solution speedup in a range comprised between 9× and 23×, and more than 18× on average. Design simulation and data extraction are the phases with the largest time savings, with speedups of 15× and 151×, respectively.

Employing a different commercial simulator might reduce the execution time of the post-route simulation in the state-of-the-art framework, but would remarkably not avoid the need to perform the computation of the power trace, which mandates instead the usage of the *AMD Vitis* toolchain to obtain a power trace that is accurate enough for modeling purposes.

IV. Conclusions

This paper introduced *Blink*, a scalable framework for the fast and automated design of run-time power monitoring infrastructures targeting FPGA-based computing platforms that optimizes time-to-solution by replacing the lengthy gate-level simulations and power trace computations used in traditional methodologies with faster behavioral simulations and direct power trace measurements. Applying *Blink* to various realistic accelerator-based designs demonstrated an average time-to-solution speedup of 18× with comparable overheads and accuracy. Future extensions foresee its application to RISC-V-based systems-on-chip [9] and cryptography accelerators [10].

References

[1] D. Zoni, A. Galimberti, and W. Fornaciari, "A survey on run-time power monitors at the edge," *ACM Comput. Surv.*, vol. 55, no. 14s, Jul 2023. [Online]. Available: https://doi.org/10.1145/3593044

[2] M. Najem, P. Benoit, M. El Ahmad, G. Sassatelli, and L. Torres, "A design-time method for building cost-effective run-time power monitoring," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 7, pp. 1153–1166, July 2017.

[3] Z. Lin, W. Zhang, and S. Sharad, "Decision tree based hardware power monitoring for run time dynamic power management in fpga," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–8.

[4] R. Rodrigues, A. Annamalai, I. Koren, and S. Kundu, "A study on the use of performance counters to estimate power in microprocessors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 12, pp. 882–886, Dec 2013.

[5] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and stable run-time power modeling for mobile and embedded cpus," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 106–119, Jan 2017.

[6] Z. Xie, S. Li, M. Ma, C.-C. Chang, J. Pan, Y. Chen, and J. Hu, "Deep: Developing extremely efficient runtime on-chip power meters," in *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2022, pp. 1–9.

[7] L. Cremona, W. Fornaciari, and D. Zoni, "Automatic identification and hardware implementation of a resource-constrained power model for embedded systems," *Sustainable Computing: Informatics and Systems*, p. 100467, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537920301918

[8] Y. Hara, H. Tomiyama, S. Honda, H. Takada, and K. Ishii, "Chstone: A benchmark program suite for practical c-based high-level synthesis," in *2008 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2008, pp. 1192–1195.

[9] D. Zoni and A. Galimberti, "Cost-effective fixed-point hardware support for risc-v embedded systems," *Journal of Systems Architecture*, vol. 126, p. 102476, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762122000595

[10] A. Galimberti, D. Galli, G. Montanaro, W. Fornaciari, and D. Zoni, "Fpga implementation of bike for quantum-resistant tls," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, 2022, pp. 539–547.