PowerTap: All-digital Power Meter Modeling for Run-time Power Monitoring

Article *in* Microprocessors and Microsystems · July 2018

CITATIONS
0

READ
1

5 authors, including:

Davide Zoni
Politecnico di Milano
37 PUBLICATIONS **125** CITATIONS
SEE PROFILE

Luca Cremona
Politecnico di Milano
4 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Alessandro Cilardo
University of Naples Federico II
80 PUBLICATIONS **635** CITATIONS
SEE PROFILE

Mirko Gagliardi
University of Naples Federico II
6 PUBLICATIONS **5** CITATIONS
SEE PROFILE

PowerTap: All-digital Power Meter Modeling for Run-time Power Monitoring

Davide Zoni[a,*], Luca Cremona[a], Alessandro Cilardo[b], Mirko Gagliardi[b], William Fornaciari[a]

[a] *Politecnico di Milano - DEIB, 20133 Milan, Italy*
[b] *Università degli Studi di Napoli Federico II - DIETI and CeRICT, 80125 Napoli, Italy*

Abstract

The power consumption is a key metric to design computing platforms. In particular, the variety and complexity of current applications fueled an increasing number of run-time power-aware optimization solutions to dynamically trade the computational power for the power consumption. In this scenario, the online power monitoring methodologies are the core of any power-aware optimization, since the incorrect assessment of the run-time power consumption prevents any effective actuation. This work proposes *PowerTap*, an all-digital power modeling methodology for designing online power monitoring solutions. In contrast with state-of-the-art solutions, *PowerTap* adds domain-specific constraints to the data-driven power modeling problem. *PowerTap* identifies the power model iteratively to balance the accuracy error of the power estimates and the complexity of the final monitoring infrastructure. As a representative use-case, we employed a complex hardware multi-threaded SIMD processor, also considering different operating clock frequencies. The RTL implementation of the identified power model targeting an *Xilinx Artix 7 XC7A200T FPGA* highlights an accuracy error within 1.79% with an area overhead of 9.95% (LUT) and 3.87% (flip flops) and an average power overhead of 12.17 mW regardless of the operating conditions, i.e., number of software threads and operating frequency.

Keywords: Dynamic Power, Power Modeling, Power Monitoring, run-time power optimization, RTL methods, Low power

1. Introduction

The power consumption represents a major obstacle to any advancement in computing technologies, limiting the performance of both embedded and high performance computing (HPC) platforms. On one hand, embedded and portable devices operate within tight power budget constraints to prolong their battery lifetime. On the other hand, HPC platforms, that aim to maximize the performance, are becoming hot-spot limited since the performance increase is restricted by both the maximum junction temperature and the cost of the required cooling systems. While the literature contains several ad-hoc solutions to optimize the power and energy metrics of both the on-chip interconnect [1, 2] and the cache hierarchy [3, 4], the power consumed by the compute unit cores, i.e. microprocessors and accelerators like GPUs, represents a major component of the power budget in such systems, particularly in embedded and mobile platforms. As a consequence, the research community has explored an increasing number of online power monitoring techniques aimed at optimizing the trade-off between power and performance [5, 6, 7, 8]. Unlike special-purpose hardware, general-purpose units like microprocessors and GPUs pose significant challenges both because they are inherently less power efficient and more difficult to characterize by means of closed power models, due to the strong dependence on the software workloads. Furthermore, because general-purpose units are meant to provide the largest degree of flexibility to software applications, they are usually overprovisioned in terms of hardware resources and a significant portion of their subcomponents stay idle, depending on the requirements of the specific application (or application phase) being run [9]. On the other hand, the well-known *dark silicon* problem makes it impossible to concurrently power all the parts of the computing device due to the impossibility of dissipating the full amount of generated heat [10]. In this scenario, online power-aware optimization techniques may play a key role in that they allow the dynamic tuning of the available computing capacity aimed at maximizing the energy efficiency under given thermal constraints.

However, the effectiveness of such optimization techniques is critically subject to the employed power monitoring method as the incorrect assessment of the power state of the system strongly affects the quality of the actuation with a negative impact on the power efficiency on the platform. At run-time, the power consumption can be read out as either a *direct measurement* or an *indirect estimate*. The *direct measurement* is achieved by means of analog sensors providing highly accurate power values at high temporal resolution. However, such solution suffers from a severe scalability issue that limits the deployment of more than few sensors even in complex designs and the use of complex mixed analog-digital design methodologies to implement them. This fact also prevents the identification of the thermal hot-spots at run-time, thus negatively impacting the reliability of the computing platform [11]. In contrast, the *indirect estimate* is achieved by means of a power model of the target architecture that is fed with the platform statistics at run-

*Corresponding Author
Email addresses: davide.zoni@polimi.it (Davide Zoni), luca2.cremona@mail.polimi.it (Luca Cremona), acilardo@unina.it (Alessandro Cilardo), mirko.gagliardi@unina.it (Mirko Gagliardi), william.fornaciari@polimi.it (William Fornaciari)

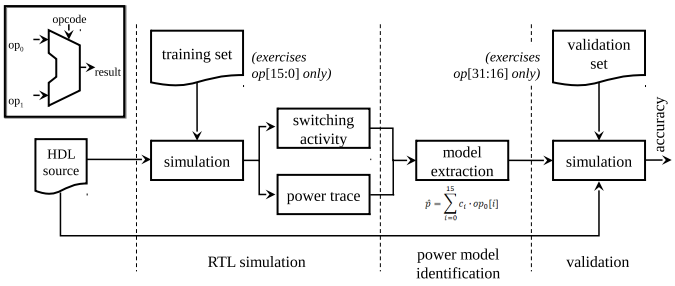


Figure 1: A generic three-stage power model identification flow.

time. Such solutions are usually scalable and cheaper due to the possibility of implementing an all-digital power monitoring infrastructure that can monitor any part of the target. However, in general, they provide a less accurate power estimate compared to direct measurement schemes, thus motivating a huge research effort to bridge such accuracy gap.

All indirect power estimate schemes leverage the relationship between the power consumption and the internal switching activity of the target architecture to build a power model that is later used as part of the online power monitoring infrastructure. The possibility of using such relationship at different abstraction levels highlights a trade-off between the accuracy of the power estimate and the effort required to extract the required information from the target platform. *Power Monitoring Counter* solutions leverage the information obtained from the performance counters that are used as the proxy for the switching activity of the target platform [12, 13]. Such solutions preserve good scalability properties and guarantee a good accuracy of the power estimates and deliver significant flexibility, since they can be implemented *after-shipping*. However, the identified power model is software-implemented, thus the induced system-wide performance overhead, at run-time, is proportional to the temporal resolution of the power estimate time series. Moreover, the performance counter infrastructure represents an optional subsystem that can be either not implemented or limited in the number of the counters that can be concurrently read out. In contrast, the most precise switching information correlating with the power consumption can be extracted by monitoring the toggle activity of each signal in the target platform at Register Transfer Level (RTL) or the corresponding activity of the driving logic cells. While the high number of signals to be monitored makes such solution infeasible, the possibility to remove the drawbacks of *Power Monitoring Counter* schemes motivates the proposal of a number of *all-digital power monitoring* solutions which try to optimally balance the number of monitored RTL signals and the accuracy of the power estimate [11, 14, 15].

Motivational example.

Figure 1 shows a generic three-stage power model identification flow using a fully-combinational 32-bit ALU as an ad-hoc example highlighting possible accuracy issues of existing techniques. The power model is identified starting from the architectural description of the ALU as well as the power trace and the switching activity map generated from the simulation of a generic set of benchmarks (see *Power Model Identification Stage* in Figure 1). To show the limitations of current solutions that exclude or limit the monitoring of data signals, assume a pathological situation where the benchmarks in the training set only exercise the least significant 16 bits of the *op0* data operand of the ALU, while all the remaining signals in the target are kept constant. It is clearly impossible to identify an accurate power model by means of the control signals only as suggested in [16]. On the other hand, we identified the power model according to the methodologies proposed in [11, 15] that allow monitoring a portion of wider signals. In particular, the identified power model makes use of the two least significant bytes of the *op0* operand to produce a power estimate. This fact completely matches the provided switching activity information from the training set, since the two most significant bytes of the *op0* operand are constant, thus not contributing to the power consumption. However, the identified power model fails to accurately predict the power consumption of the target ALU when a different set of benchmarks which only exercises the two most significant bytes of the *op0* operand is used. However, such drawback cannot be addressed by exhaustively extracting the switching activity for the entire space of data input values even for simple targets, i.e., the considered ALU shows a space of the data inputs equal to 2^{32} times two. To this extent, *PowerTap* delivers a fresh approach to the power modeling problem to avoid measuring the switching activity of each combination of data input values in the target.

Contributions. This work presents *PowerTap*, a methodology to design an all-digital power monitoring infrastructure for generic RTL descriptions encompassing both the RTL-level power modeling and the feasibility issues. The methodology is especially targeted at microprocessor design and, to the best of our knowledge, it is the first proposal targeting complex hardware multi-threaded architectures with Single Instruction Multiple Data (SIMD) support, representative of current multi/manycore accelerators as well as GPU devices. The power modeling approach turned out to be effective for a large variety of specific validation examples, including computing blocks, e.g., Arithmetic Logic Unit (ALU) and Floating Point Unit (FPU), as well as non computing modules, e.g., Load Store Units (LSUs) and instruction fetch (IF) functions. The three peculiar innovations of *PowerTap* are described in the following:

- *ISA-constrained data-driven RTL power modeling - PowerTap* employs few ad-hoc micro-benchmarks to extract the required RTL-level switching activity used to identify the power model. Each micro-benchmark is tailored to the target architecture and systematically uses a selected subset of the instructions permitted by the Instruction Set Architecture (ISA) specification to stress a precise part of the target platform. Compared to the generic benchmarks used in the power modeling state-of-the-art proposals, the micro-benchmarks employed in *PowerTap* allow to dramatically reduce the computational time to extract the switching activity of the target. In addition, *PowerTap* avoids extracting the complete map of the switching activity due to the operand data input and output signals

due to their huge size. In contrast, we leverage the linear relationship between the power consumption and the switching activity of the operand data input and output to accurately identify the power model.

- *Guidelines for RTL signal selection* - The proposed methodology identifies the crucial signals to be monitored to achieve an accurate power estimate at run-time, with minimal area and power overheads. As an improvement to the state of the art, for each identified signal a set of guidelines are defined to measure its switching activity in a way that minimizes the accuracy error within the range of expected values for such signal.

- *Complete power monitoring design methodology - PowerTap* delivers a complete all-digital power monitoring solution starting from the RTL description of a generic digital architecture. To demonstrate the scalability of our solution, the complete methodology has been validated against a GPU-like SIMD processor endowed with hardware multi-threading support. The results considering a target *Xilinx Artix 7 XC7A200T FPGA* chip highlight an accuracy error within 1.79% with an area overhead of 9.95% (LUT) and 3.87% (flip flops) and an average power overhead of 12.17 mW regardless of the operating conditions, i.e., number of software threads and operating frequency.

Structure of the manuscript. The rest of the manuscript is organized in four parts. Section 2 reports the state-of-the-art on the online power monitoring solutions. Section 3 presents the *PowerTap* methodology while the use case architecture is detailed in Section 4. The results are discussed in Section 5 and some conclusions are drawn in Section 6.

2. Related works

Direct power measurement methods require either external [17] or internal [18] analog meters. Despite their high accuracy and temporal resolution, they suffer from two main limitations. First, they only return the total power consumption, thus making impossible to monitor either thermal hot-spots or power consumption of a specific subsystem of the target platform. Second, scalability issues prevent deploying more than a few of them in complex designs.

The use of the *Power Monitoring Counters* represents a widely adopted solution to design indirect power monitoring schemes. [19] proposed a power monitoring solution that leverages the CPU utilization at the software level as the proxy for the switching activity of the circuit, given a voltage-frequency pair. In contrast, [20, 13, 7] proposed different online power monitoring infrastructures that directly leverage the architectural performance counters to identify a power model for the *ARM big.LITTLE* platform. Other authors in [21] presented a full system power monitoring architecture exploiting the architectural performance counters. A similar approach is proposed in [22] targeting the *Intel Pentium 4* architecture. [23] presented a performance counter-based power model to support an energy-constrained thread scheduling algorithm.
On a different but still related perspective, the investigation in [24] elaborated on the existence of a minimum set of performance counters that allow estimates of the power consumption that stay accurate across different architectures.

Several works proposed *Power Monitoring Counters* schemes for GPUs. [25] presents a linear power model for GPUs leveraging the performance counters exposed through the CUDA software interface. [26] discussed a neural network to predict the average power of a GPU kernel from an empirically derived counter-based performance model of the kernel itself. Starting from the instruction types and low-level architectural counters extracted from a GPU microarchitectural simulator, [27] employed linear regression tree and random forest methods to predict the energy of GPU kernels. Exploiting a closed-form performance model developed in [28], the authors in [29] presented an analytical power model for GPUs. Most of these previous works rely on computationally intensive approaches to solve the power modeling problem, such as neural networks or complex analytical formulations, which are not suitable for online power monitoring implementations. In fact, they highlight the difficulty of capturing in a closed form the power behaviour of the GPU system, particularly the compute cores, only relying on architectural counters.

We note that the above proposals are limited by the availability of the architectural performance counters that, in general, are integrated in complex computing platforms for which their presence does not critically affect both the area and the power budgets of the overall platform. In contrast, ultra-low power embedded systems pose tight area and power budgets for the self-monitoring infrastructures, thus imposing ad-hoc hardware-level solutions.

The all-digital power monitoring proposals represent a recent solution to cope with the constrained area and power budgets that are typical present in low power designs. Moreover, such approaches easily allow minimizing the performance overhead since the power estimate is computed in hardware. The authors in [16] presents a methodology to build an all-digital power monitoring infrastructure starting from the control signals of the target architecture. Differently, [11] proposed a two-step methodology to design an all-digital power monitoring infrastructure without focusing only on the control signals. In particular, starting from the complete RTL description of the target platform, all the signals are virtually organized as multiple single bit signals. Then, a representative set of the single-bit signals is selected to form the power model using the switching activity information extracted from the simulation of a set of benchmarks. A similar approach is discussed in [15], targeting single bit flip-flops instead of the corresponding driven signals. Moreover, [15] accounts for voltage and frequency parameters in the power model identification process to increase the usability of the solution. We note that the current all-digital power monitoring proposals try to minimize the number of observed signals without considering the potential loss of accuracy of the identified power model. In that respect, *PowerTap* presents a set of guidelines that remain generic to any all-digital power modeling solution and are used to define the way the switching

activity is accounted for each selected signal. Finally, the use of generic benchmarks to extract the switching activity represents a second source of problems of any current all-digital power monitoring solution. *PowerTap* proposes a micro-benchmark approach bounded to the allowed and realistic Instruction Set Architecture (ISA) software patterns.

3. PowerTap methodology

PowerTap leverages the RTL-level switching activity to accurately estimate the power consumption of the target platform with two objectives. First, to deliver a low overhead power monitoring infrastructure for generic RTL descriptions regardless of their size and complexity. Second, to ensure a limited accuracy error for the identified power model.

Figure 2 depicts the four stage flow that implements the *PowerTap* methodology. The *RTL simulation stage* synthesizes, maps and simulates the target design to extract the switching activity of the target platform in the form of a *Value Change Dump* (VCD) file. To reduce the computational time for the power model identification, the *ISA-constrained data-driven micro-benchmarks* method generates a set of micro-benchmarks to selectively stress parts of the target platform. A detailed description of the proposed *ISA-constrained data-driven* approach is discussed in Section 3.1. The VCD information from the *Simulation Stage* are used to compute the power trace and the switching activity for each signal in the target platform (see *Power Trace Extraction Stage* in Figure 2). Depending on the final application of the power monitoring infrastructure, the power trace is generated with different temporal resolutions. The *temporal resolution* of the power trace is defined as the fixed time window, usually expressed in terms of the number of simulated clock cycles, for which the reported switching activity is used to compute a single power value. While the maximum temporal resolution is limited by the employed power analysis tool, the actually employed temporal resolution is subject to the application scenario of the power monitoring infrastructure. For example, a power trace with a 10us temporal resolution on a target that is clocked at 100MHz (10ns period) is made of a power sample time series where each sample is computed from the switching activity within a time window of one thousand clock cycles. In general, the switching activity of a signal is defined as the number of changes in the logic state of the signal itself over a finite amount of time. We extend such definition for multibit signals to support *PowerTap*. The *Single Toggle Count* (STC) of a multibit signal is defined as the transition of the logic state of one of more bits of the signal itself. In contrast, the *Hamming Weight Count* (HWC) of a multibit signal is the number of bits that flip their logic state during the multibit signal transition. In particular, the STC of a multibit signal measures the number of changes of the signal during a fixed amount of time, regardless of the number of actually switching bits for each transition. The HWC measures the number of switching bits for each transition of the same multibit signal over the same fixed amount of time. *PowerTap* constraints the number of considered signals during the power model identification stage by those that represent, in the design hierarchy, a primary input or output for a module. From one hand, this strategy avoids modeling the complex non-linear relationship between the power consumption and the internal logic of an hardware module. On the other hand, this fact positively impacts the computational time for the power model identification due to a strong reduction in the number of considered signals. However, additional techniques are discussed in Section 3.2 and Section 3.3 to achieve an affordable complexity of the power model identification stage even for complex target platforms. The *Power Model Stage* takes the power traces and switching activity in terms of both STC and HWC for each considered signal of the design, to identify the power model of the target platform. Starting from the guidelines to model the switching activity of the selected signals as either STC or HWC (see Section 3.2), an in-depth discussion of the power modeling stage is devoted in Section 3.3. The identified power model is moved from the mathematical to the RTL description and then it is added to the RTL description of the target platform to enable the self-power-monitoring property (see *RTL power model implementation stage* in Figure 2). As shown in Figure 3, the values from the different counters are then combined together as a weighted sum within each module and across the hierarchy for those modules that the methodology considered significant for power estimation. Notice that, in order to reduce the power overhead incurred by the monitoring infrastructure itself, the counters are not immediately fed to the combinatorial multiplier/adder network, but buffered in a register that is enabled only when strictly needed, as required by the desired power resolution.

3.1. ISA-constrained data-driven benchmarks

The *data-driven modeling* (DDM) represents the de-facto solution to identify the power model of a given generic RTL description, within the all-digital power monitoring methodology [11, 15]. In contrast to physical and mathematical approaches, the DDM is based on the data analysis to identify the model without knowing the physical behaviour of the system at hand. On one hand, this fact greatly simplifies the identification of the power model that is not forced to obey to any a-priori physical behavior for which the corresponding mathematical formulation can be arbitrarily complex. On the other hand, the data collection represents a critical design stage to properly extract valuable switching activity information. In particular, the selected dataset has to be representative of all the possible working conditions of the target and also reasonably compact to minimize the computational time to collect the required information. To extract such switching activity information, the state-of-the-art solutions employ a large set of benchmarks, e.g., 150 benchmarks [15] and 60 benchmarks [13], claiming that they are representative of all the possible working conditions of the target. However, such benchmarks are not primarily intended for power modeling and without proper constraints we demonstrated the impossibility of extracting an accurate power model of the target (see Figure 1). In addition, the computational time to extract the required switching activity grows with both the complexity and size of the target architecture as well as the execution time of the benchmarks in the training set, thus motivating the use of small ad-hoc training benchmarks.

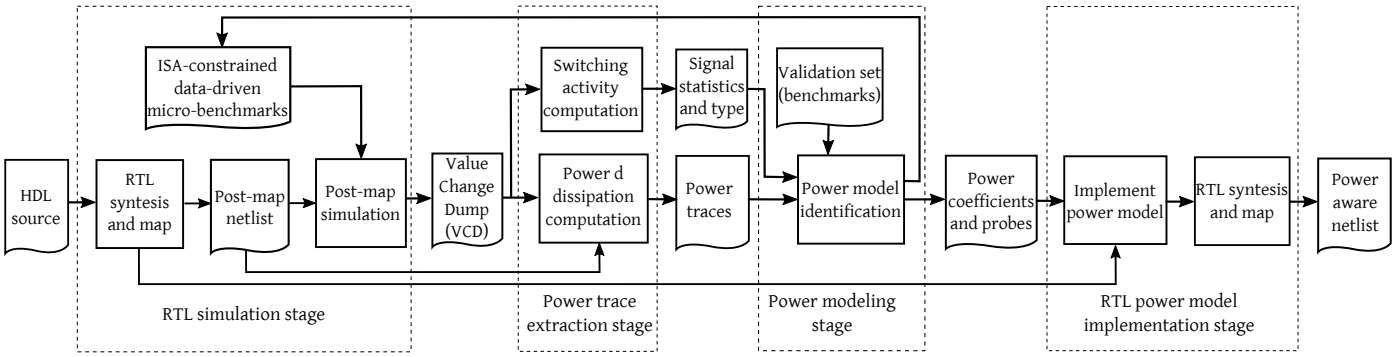


Figure 2: Overview of the simulation flow used within *PowerTap*. The post-map logic simulation provides the power traces and the microarchitectural statistics used to derive the power model. The RTL is instrumented with the derived power model and then synthesized again.

PowerTap introduces a novel ISA-constrained data driven solution to minimize the computational time to complete the switching activity extraction still ensuring good accuracy of the corresponding identified power model. Similar to previous works dealing with counter-based GPU power estimation [8], we classify the different instructions in the ISA by the corresponding architectural functionality they stress in the target platform. In particular, the method classifies the ISA instructions in three partitions, i.e., ALU, FPU and LSU. For each partition, we split the scalar and the SIMD instructions, since depending on the actual microarchitecture they can differently exercise the same functional unit. We note that the micro-benchmarks are designed starting from the architectural view of the target platform thus being suitable regardless the actual microarchitecture that implements the ISA. Depending on the maximum operating frequency of the target and the required temporal resolution of the power trace, we design each micro-benchmark to produce at least hundred power samples to avoid border effects while ensuring a feasible simulation time. A critical design step requires to avoid exploring the space of all input data values. In particular, the data operands of each instruction in the micro-benchmark are fed with random values, since the actual dependency between the power consumption and the data signals is addressed in the *Power modeling stage*. *PowerTap* also implements a feedback loop from the *Power model identification stage* to the *ISA-constrained data-driven micro-benchmarks*. This link allows the designer to manually tune the classes of micro-benchmarks in the case the identification stage highlights the impossibility to accurately model the power consumption of the target.

3.2. Signal selection guidelines

This section presents the five practical guidelines to select the signals to build the power model of the target architecture. For each type of signal, i.e., control or data, it is specified how to measure the switching activity to maximize the information and to avoid over-fit behaviours.

G1 *Take a data-centric approach for defining the power model.* As highlighted by our motivational example, excluding or trimming the data signals used as input to the power model, possibly driven by a certain training set used during the power model identification stage, can jeopardize the resulting model accuracy. Such *benchmark-induced* biases are less critical for performance-oriented models, while for power they might heavily affect the resulting estimate due to the inherent dependence of the switching activity on the data bit patterns. As a consequence, guideline G1 assumes that all bits of the selected data signals are used as inputs to the data model.

G2 *Consider both data and control signals.* Despite their different contribution both control and data signals have to be considered in the process that selects which signals are going to made the power model.

G3 *Hamming Weight Count (HWC) to measure the switching activity of data signals.* The relationship between the power consumption and the switching activity of a data signal depends on the actual number of bits of the signal that switches their state, i.e., hamming weight, since, in general, the higher the hamming weight, the higher the power consumption.

G4 *Single Toggle Count (STC) to measure the switching activity of control signals.* In general, a change in a control signal of the design enforces the execution of a different hardware operation regardless of the hamming weight of the control signal itself. Thus, for each control signal, the guideline imposes to account for a single change, i.e., STC, any time the control signal changes at least one bit.

G5 *Black-box modeling.* The internal state of a generic RTL description is complex and in general highly non-linear, thus imposing the use of complex family of models to accurately capture its dynamic. To avoid such modeling effort, *G5* describes two power model identification strategies. First, identify the power model of a generic module as a function of the input and output signals of the module itself to reduce to number of processed signals and to allow a linear model formulation. Second, identify the power model of a module as the sum of the power estimates of all of its first level children modules plus the power estimate of its glue logic.

3.3. Power modeling stage

The power model identification stage represents the core of the *PowerTap* methodology and tries to address two contrasting

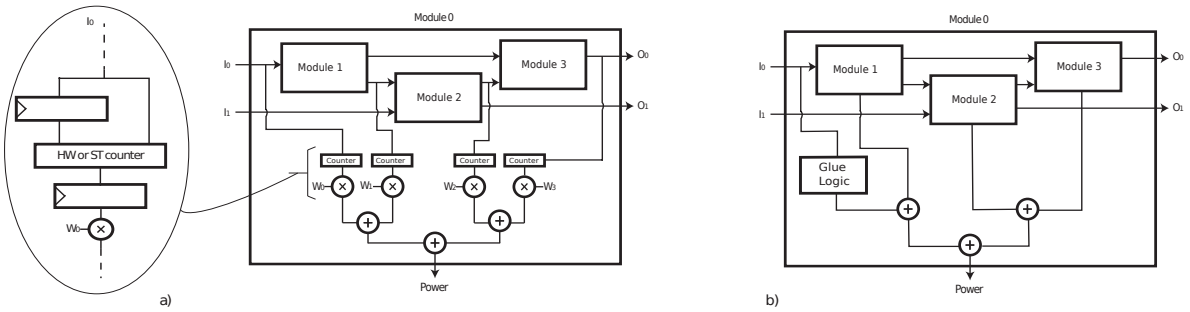


Figure 3: The power model of a module in the hierarchy is built from a weighted combination of its primary inputs and outputs (a) or by adding the power contributions of all of its sub-modules plus the power contribution of its glue logic (b).

requirements. First, the quality of the identified power model strongly affects the accuracy of the power estimates and consequently any power-aware optimization policy. Second, the complexity of the identified power model dominates the power and area overheads of the final power monitoring infrastructure.

PowerTap implements a three-stage power model identification stage to fulfill the two opposite requirements. First, the *top-down visiting* algorithm builds the power model of the target platform as the composition of a set of power models identified in a corresponding number of modules in the target hierarchy (see Algorithm 1). The strategy allows limiting the size of each identification problem to a single module, thus ensuring a scalable solution. This is in contrast with the state-of-the-art solutions, e.g., [11, 15], that try to flatten the target hierarchy to a huge set of signals to identify a global power model that, thus, prevents the scalability of the methodology.
Second, the *validation algorithm* assesses the accuracy of the identified power model against a set of general purpose benchmarks. This step evaluates the quality of the collected switching activity information, since a failure at this stage imposes to augment the set of micro-benchmarks to collect the missing switching activity information (see the link from the *Power model identification* to the *ISA-constrained data-driven micro-benchmarks* in Figure 2). Third, the *reduction algorithm* scans and merges any signal that is common to different identified power models to minimize the power and area overhead of the final power monitoring infrastructure.

The *top-down visiting* algorithm processes the module hierarchy of the target starting from the top module and iterating, top-down, on the levels of the hierarchy (see Algorithm 1). As in [11], we employ the RMSE accuracy metric to drive the visiting (see Algorithm 1) and the model identification (see Algorithm 2) algorithms. The RMSE is defined in Equation 1. At the beginning of the *Power modeling stage*, Algorithm 1 identifies the power module for the top module of the hierarchy. If the RMSE accuracy error of the identified power model does not satisfy the accuracy threshold (*accThreshold*), Algorithm 1 proceeds by visiting the levels of the hierarchy top-down. At each level of the hierarchy, Algorithm 1 computes the power model for each module as well as the power model of the glue logic. The model of the glue logic is obtained by subtracting the power consumption minus the sum of the power consumption of the modules in the considered level. For a level of the hierarchy, the power model of the glue logic accounts for the the power consumption that is not observable from the modules of the considered level. The RMSE accuracy error is evaluated as the sum of the power estimates from all the modules of the actual level of the hierarchy plus the power estimate of the glue. Algorithm 1 terminates either when the RMSE computed for a specific level of the hierarchy satisfies the accuracy threshold (*accThreshold*) or when the threshold on the maximum number of levels is reached (*MaxLevTh*). In the latter case, the vector of power models that collected the best RMSE is returned even if the accuracy threshold is not satisfied.

Algorithm 1 Top-down hierarchical visiting.

1: **function** $[PM, \hat{p}]$ VISIT( )
2: **for** $l \in 1 : maxLevTh$ **do**
3: // compute one power model for each module of this level
4: **for** $m \in submodules(l)$ **do**
5: $[PM_{l,m}, \hat{p}_{l,m}] = ComputePwr(p_{l,m}, \overline{HWC}_{l,m}, \overline{STC}_{l,m})$;
6: **end for**
7: //compute power model for the glue logic
8: $glue\_pwr = p_{l-1} - \sum_m \hat{p}_{l,m}$;
9: $[PM_{l,glue}, \hat{p}_{l,glue}] = ComputePwr(glue\_pwr, \overline{HWC}_{l-1,m}, \overline{STC}_{l-1,m})$;
10: //exit condition
11: **if** $RMSE(\sum m\hat{p}_m + \hat{p}_{l,glue}) < accThreshold$ **then**
12: $return[PM_l, PM_{l,glue}, \sum m\hat{p}_{l,m} + \hat{p}_{l,glue}]$;
13: **end if**
14: //update best power model vector
15: $\hat{p}_{currLev} = \sum_m \hat{p}_{l,m} + \hat{p}_{l,glue}$;
16: $PM_{currLev} = PM_l + PM_{l,glue}$;
17: $[PM_b, \hat{p}_b] = BestRMSE([PM_b, \hat{p}_b], [PM_{currLev}, \hat{p}_{currLev}])$;
18: **end for**
19: $return[PM_b, \hat{p}_b]$;
20: **end function**

Algorithm 2 represents a utility function within Algorithm 1 and computes the power model for a module given its RTL description, the power consumption time series (p) and the switching activity information for each input and output signals of the module itself. The switching activity information is provided in the form of either HWC or STC depending if the corresponding input or output is a data or a control signal, i.e., $\overline{HWC}$ and $\overline{STC}$ in Algorithm 2, respectively. To compose the power model, Algorithm 2 considers all the data signal statistics ($\overline{HWC}$) as well as a subset of the control signals. The latter are

Algorithm 2 Power model computation.

1: **function** $[PM, \hat{p}]$ ComputePwr$(p, \overline{HWC}, \overline{STC})$
2: $\overline{selStats} = \overline{HWC}$;
3: $[PM, RMSE] = Ident(p, \overline{selStats})$;
4: $\overline{STC}_{sort} = SortByPowerCorrelation(\overline{STC}, p)$;
5: **for** $i \in \overline{STC}_{sort}$ **do**
6: $[PM_{tmp}, RMSE_{tmp}] = Ident(p, [\overline{selStats}, \overline{STC}_{sort}(i)])$;
7: **if** $RMSE_{tmp} < RMSE$ **then**
8: $\overline{selStats} = [\overline{selStats}, \overline{STC}_{sort}(i)]$;
9: $RMSE = RMSE_{tmp}$;
10: $PM = PM_{tmp}$;
11: **end if**
12: **end for**
13: return $[PM, PM(\overline{selStats})]$;
14: **end function**

added to the power model following an iterative selection process that terminates when the addition of any control signal to the power model does not improve the RMSE metric (see lines 5-12 in Algorithm 2).

$$RMSE = \sqrt{\frac{\sum_i^N (\hat{p}_i - p_i)^2}{N}} \quad (1)$$

The *validation algorithm* simulates a set of representative benchmarks across different operating conditions to generate, for each module with an associated power model, the time series for both the power estimates and the power consumption. The RMSE accuracy metric defined in Equation 1 is used to assess the quality of the final model. Moreover, the RMSE quantity is also computed between the power consumption and the power estimate of each identified power model. Such quantities allow the methodology to identify the power models and the associated RTL modules for which the collected switching activity information is not sufficient to provide an accurate power estimate against the validation set of benchmarks. In particular, such information is backward propagated to the *ISA-constrained data-driven micro-benchmark* block of the *PowerTap* flow to help generate additional micro-benchmarks that selectively stress such modules. Last, the *reduction* techniques reduce the complexity of the final power monitoring infrastructure by merging the signals used in multiple power models. We note that the minimization of the power and area overheads increases with the width of the merged signals, thus, in general, the data signals represent the best candidates for the merging actions.

4. The *nu+* architecture

The proposed methodology has been validated against *nu+*, a hardware multi-threaded SIMD (vector) processor inspired by the GPU architectural paradigm to allow the execution of both general-purpose [30, 31] and GPU-like [32, 33] applications. The heart of the platform is a RISC in-order core, oriented to highly data-parallel kernels with a lightweight control infrastructure, shown in Figure 4. Most of its resources are dedicated to computation-intensive operations on massive datasets, blending together a hardware multi-threading support with a SIMD paradigm. The *nu+* implementation used in this work is equipped with 4 hardware threads, with a SIMD capability able to compute 16 concurrent operations each cycle, and 64 general purpose registers. Both Data and Instruction caches are 4-way set-associative with 128 sets each and a data width of 512 bits.

Each hardware thread has private internal resources such as PC, register file, and status/control registers along with a private memory stack, although all threads share the same compute units and L1 cache. Specific thread information, such as current PC value, thread status, is handled by the *Thread Selection* unit in the first stage, which implements an interleaved multi-threading scheduling in a fine-grain way with a low architectural impact. The Thread Selection unit issues an active thread to the next stage based on its information. At this stage, an internal round robin arbiter forwards to the *Instruction Fetch* the selected thread ID and its current PC value in a fair mode after every cycle. The Thread Selection updates the issued thread PC value according to the feedback signal from the Instruction Fetch, and in case of instruction cache miss the issued thread is stalled becoming no more eligible for scheduling. Its PC value is not increased and a memory request for the requested instruction line is issued to the main memory subsystem, which can require up to *m* cycles depending on the main memory latency. When the data is gathered back from the main memory, the *Thread Selection* is notified and the previous stalled thread is reactivated. In case of instruction hit, the Instruction Fetch retrieves the requested instruction from the cache, which flows along with the scheduled thread ID to the *Decode* unit in the next cycle.

The control system relies on a lightweight scoreboarding mechanism for both data and structural hazard detection. The *Thread Scheduler* is the heart of such logic. It updates the scoreboarding system stalling threads whenever hazards or data cache misses occur. This stage checks potential data hazards for the incoming thread and updates the scoreboard consistently with the issued instruction. At the same time, it checks if the current thread instruction raises a structural hazard on the *Writeback* stage. In the execution datapath different operators have different latencies (such as dividers and multipliers), therefore they can collide in the writeback operation. At each clock cycle, the *Thread Scheduler* issues a schedulable thread to the execution datapath and, whenever hazards occur, it notifies back involved threads IDs to the *Thread Selector* which stalls them until those conditions are no more true.

Execution datapaths and register files are designed to exploit data-level parallelism in line with the SIMD paradigm. The *Register File Manager* fetches data from registers and composes operands. Each thread has both private scalar and vectorial register files. The latter can store up to 16 scalar data, with a total width of 512 bits in order to satisfy the execution pipeline data throughput. Both register files are organized in compact SRAMs with two read and a write ports each, allowing two read and a write operations during each cycle. The total number of registers allocated is proportional to the number of threads supported. In the current implementation each regis-

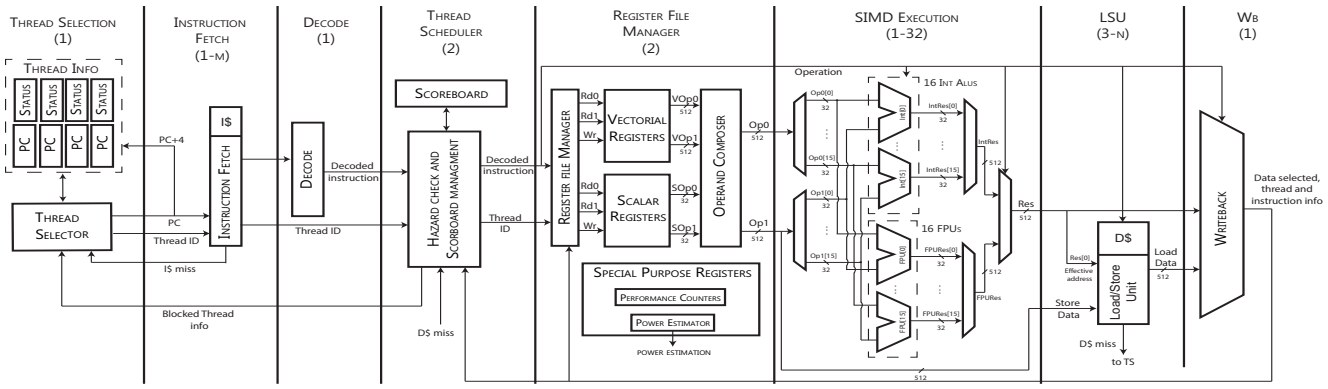


Figure 4: Simplified overview of the target SIMD architecture. This figure highlights both data and thread level parallelism. On the top side of the figure, numbers in brackets highlight the number of cycle for each stage.

ter file has 256 registers, i.e. 64 for each thread. The control logic in this stage retrieves the right register subset based on the requesting thread ID, which feeds the high part of both read address ports. In the next cycle, operands are composed based on the instruction decoded information and issued to the execution pipelines along with thread and decoded information required by the computation units.

The architecture implements an instruction set containing instructions that operate on arrays of data. Computational units are organized in hardware vector lanes, with each scalar operator being instantiated 16 times. The *Integer Execution* unit and the *Floating Point Unit* have a similar organization, they both receive operands composed by the *Register File Manager* organized in vectors, then they internally decompose each vector and feeds all the ALUs and FPUs with scalar operands. In the Integer Execution unit, all the allocated ALUs perform the same operation in one clock cycle. On the other hand, floating point operators have different latencies, up to 32 clock cycles due the divider in this implementation. Such a data parallelism allows each thread to perform SIMD operations on 16 independent data simultaneously. The Integer Execution unit also contains the *Branch Control* module which handles jumps and rollbacks, flushing the control information for the involved thread in the pipeline when they occur and notifying the *Instruction Fetch* to update the thread PC. On the other hand, the computational outputs from both the integer ALUs and floating-point units are gathered and reorganized in a vectorial form, then forwarded to the LSU module along with threads information.

The LSU module is organized in a 4-way set-associative write-back L1 cache strictly coupled with a light cache controller which implements a simple valid/invalid coherence mechanism. Such cache controller handles misses and memory transactions and it also provides both request serialization and merging mechanisms in order to correctly manage concurrent requests from different threads. The cache line width matches the internal hardware lanes capability, thus a read memory request loads 16 scalar data from main memory and stores them into a vector register at once, minimizing requests and exploiting the internal parallelism of the SIMD accelerator. On the other hand, the LSU is organized in three stages. The first stage receives the effective address calculated by the previous stage and, in case of data misses, it notifies the Thread Scheduler unit stalling the thread until the data is retrieved back from the main memory. The other two stages manage respectively coherence information and data.

Finally, the *writeback* receives both integer execution unit and LSU outputs and forwards one of them to the register file write port based on the instruction performed. The right register subset is selected based on the current thread ID. Concurrently, the Thread Scheduler sniffs transactions from the writeback module and updates the scoreboarding system accordingly.

The target architecture comes with a toolchain based on the LLVM project and includes a custom version of the Clang front-end and a native *nu+* back-end. The Clang front-end allows users to compile C/C++ source code in a fast way and with a low memory usage. On the other hand, the toolchain is deeply customized for exploiting the core internal data parallelism and reaching the maximum throughput. The compiler has a complete vision of the SIMD nature of the datapath. It supports custom vector types, thus standard arithmetic and bit-wise operators are available for both scalar and vector operations. Furthermore, the custom version of Clang supports ad-hoc built-in functions that are required to fully exploit target specific features, such as thread synchronization and special SIMD operations.

Control and special purpose registers are visible to the host controller. The architecture provides a dedicated interface which allows the manager to retrieve such registers in real-time during the execution without interfering with the kernel flow. The baseline implementation is equipped with general performance counters, such as cache data misses occurred. In this work, the interface has been extended in order to read the counter registers required by the proposed methodology achieving the capability to read in real-time the current power estimation consumed by the architecture.

5. Experimental results

This section discusses the results of the *PowerTap* methodology applied to the *nu+* architecture encompassing a two-fold objective: first, demonstrating the accuracy of the power estimates extracted from the implemented power monitoring in-

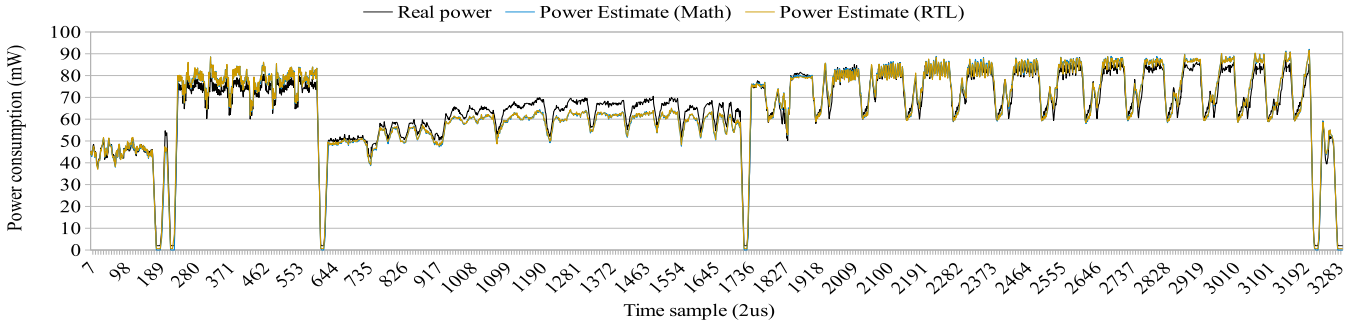


Figure 5: Power estimates over time for the mathematical and RTL implementation of the identified power model. The single thread version of the benchmarks in the validation set are executed on a *nu+* instance that operates at 50MHz and the computed RMSE_{Math} and RMSE_{RTL} are 0.85 and 0.50, respectively. Results are obtained by concatenating the executing of the benchmarks (1)-(14) (see Table 1 for the mapping between benchmark name and numeric identifier).

Table 1: Subset of the WCET benchmarks used as validation set. For each benchmark we provide a SIMD and scalar implementation and, for the former, a three parallel version of the benchmark using 1, 2, and 4 threads of execution. The numeric identifiers associated to each version of the benchmarks are used in Figure 5 – 7 to show the sequence of executed benchmarks.

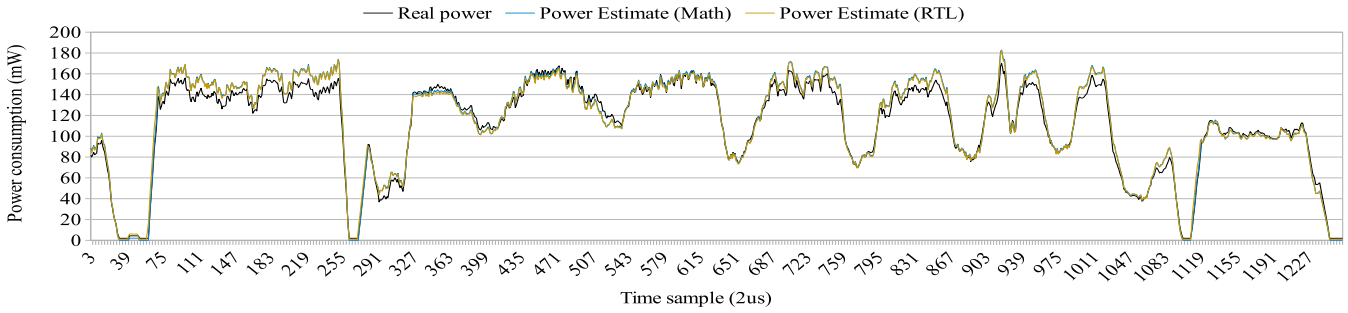
	Scalar	SIMD		
	1	1	2	4
conv_layer	✓(1)	✓(10)	✓(15)	✓(20)
fdct	✓(2)			
fft	✓(3)	✓(11)	✓(16)	✓(21)
fibcall	✓(4)			
fir	✓(5)	✓(12)	✓(17)	✓(22)
gauss	✓(6)	✓(13)	✓(18)	✓(23)
ludcmp	✓(7)			
minver	✓(8)			
mmvet_float	✓(9)	✓(14)	✓(19)	✓(24)

frastructure; second, experimentally validating the robustness of our solution when used against different flavors of the same *nu+* architecture.

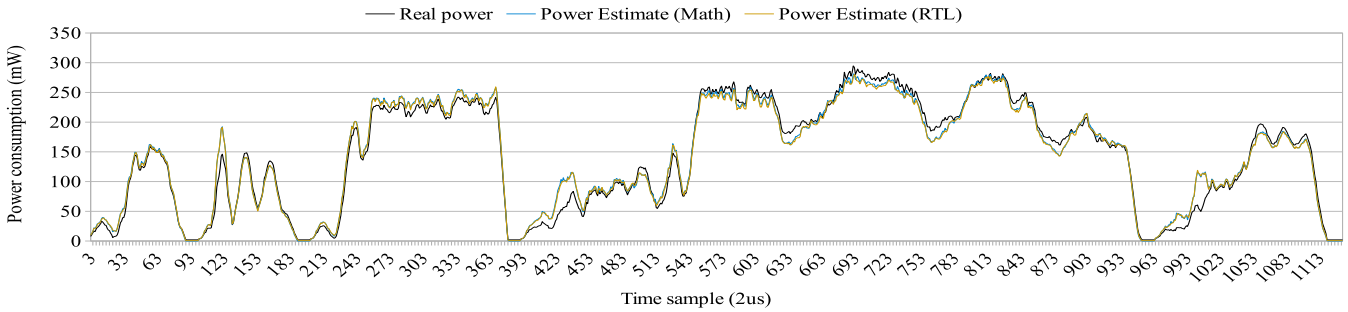
Use case scenarios. *PowerTap* has been validated considering three architectural parameters of the *nu+* target platform: *i)* scalar and vector, i.e., Single Instruction Multiple Data (SIMD), benchmarks, *ii)* multi-threaded benchmarks and *iii)* different operating frequencies, i.e., 20MHz, 50MHz, and 62.5MHz. We note that the power model has been identified considering a single operating frequency and the single-threaded version for all the employed micro-benchmarks while has been validated across different case study scenarios to assess the flexibility of the proposed methodology. We employed *Xilinx Vivado 2017.4* toolchain to synthesize, map, and simulate the *nu+* platform considering the *Nexys4 Video* development board endowed with an *Artix-7 XC7A200T* as target FPGA. The operating frequency constraints for the synthesis has been set to 100MHz without imposing area and power constraints.

Benchmarks. The validation set is made of 9 benchmarks from the WCET suite [30]. Each benchmark provides different implementations to stress different parts of the target architecture (see Table 1). In particular, we provide the scalar version for each workload and the SIMD implementation for five of them. In addition for each SIMD benchmark the parallel versions using one, two, and four threads have been developed. Table 1 associate a numeric identifier to each benchmark. The numeric identifiers are used in the presented results to show the sequence of executed benchmarks within each of the analyzed use-cases. Starting from the guidelines discussed in Section 3.1, a set of 6 micro-benchmarks is used to collect the switching activity information for the power model identification. Starting from the *nu+* Instruction Set Architecture (ISA) we implemented a micro-benchmark for each identified instruction type, i.e., ALU, FPU, and LSU. Moreover, for each micro-benchmark both the scalar and the SIMD implementations are provided. We note that the switching activity is collected considering the single-threaded implementation of each micro-benchmark and the 50MHz operating frequency is used.

Evaluated metrics and power monitoring infrastructure implementation. *PowerTap* has been validated considering the accuracy as the quality metric as well as the power and area overheads. As for the power model identification, the accuracy metric is defined as the *RMSE* (see definition in Equation 1) computed on the benchmarks of the validation set. We employ the *Xilinx Report Power* modeling tool within the *Xilinx Vivado 2017.4* suite to compute the power consumption starting from the VCD information. Unfortunately, *Xilinx Report Power* allows estimating only the average power consumption on the entire simulated time window. To this extent, we implemented a wrapper that splits the VCD in time slices to allow collecting a time series of power values. In particular, we set the temporal resolution of the power trace to 2us, i.e., 100 clock cycles at the operating frequency equal to 50MHz. We note that the temporal resolution is kept constant at 2us regardless the operating frequency of the design. The collected results are discussed in terms of the accuracy of the identified power model with respect to the power consumption extracted from the *Xilinx Power Report* tool when the benchmarks in the validation set are executed. The results are reported considering both the mathematical formulation (*Math*) and the RTL implementation (*RTL*) of the identified power model to highlight the negligible accuracy loss of the implemented power monitoring infrastructure. We employed the strategy proposed in [14], that makes use of fixed point arithmetic, to implement the RTL power monitoring infrastructure starting from the mathematical formulation of the



(a) Two-threaded benchmarks ($RMSE_{Math}$= 0.66 $RMSE_{RTL}$= 0.81). Results are obtained by concatenating the executing of the benchmarks (15)-(19) (see Table 1).



(b) Four-threaded benchmarks ($RMSE_{Math}$= 1.22 $RMSE_{RTL}$= 1.79). Results are obtained by concatenating the executing of the benchmarks (20)-(24) (see Table 1).

Figure 6: Power estimates over time for the mathematical and RTL implementation of the identified power model. The two and four thread versions of the benchmarks in the validation set are executed on a *nu+* instance that operates at 50MHz.

power model. Each switching activity counter (see Figure 3) is implemented as an RTL register with a number of bits that allows counting the maximum switching activity of the monitored signal for the entire duration of the 2us time frame with no overflow. The maximum switching activity is intended as the flip of each bit of a multibit signal for each clock cycle. The area overhead is computed as the percentage increase of the occupation of the target due to the power monitoring infrastructure. Such overhead is reported in terms of LUT and registers. We also reported the average power consumption due to the power monitoring infrastructure considering the *nu+* platform and the *Xilinx Artix 7 XC7A200T FPGA* chip. We note that *PowerTap* delivers an all-digital power monitoring solution that introduces no *performance* overheads, since the power estimate is computed by means of dedicated hardware resources that stay separated from the computational resources of the target platform.

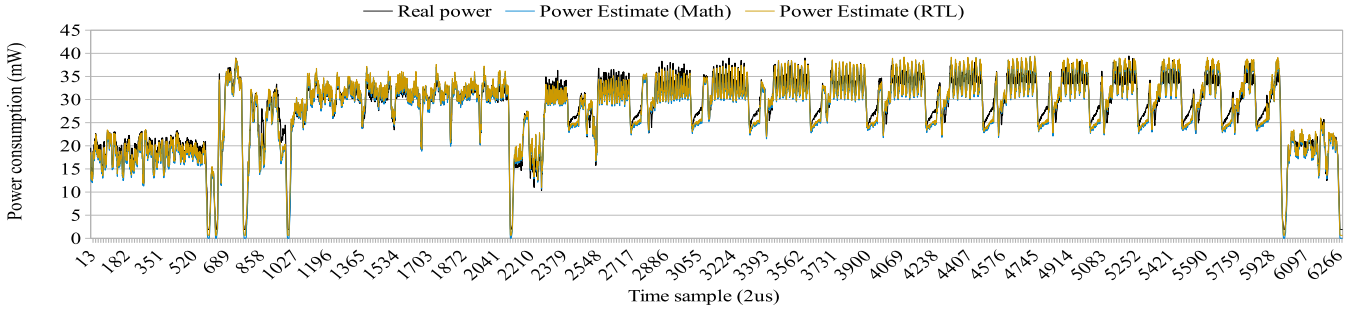
5.1. PowerTap accuracy results

Figure 5 reports the results considering a *nu+* instance operating at 50 MHz while executing the single thread version of each benchmark in the validation set. We note that the scalar and the SIMD versions of each benchmark have been executed. To enhance the readability of the results, the time-based power trace for each benchmark has been concatenated to provide a single trace. The y-axis report the total power consumption of the *nu+* in *mW*. The RMSE accuracy error is limited to 0.85 (*Math*) and 0.50 (*RTL*).

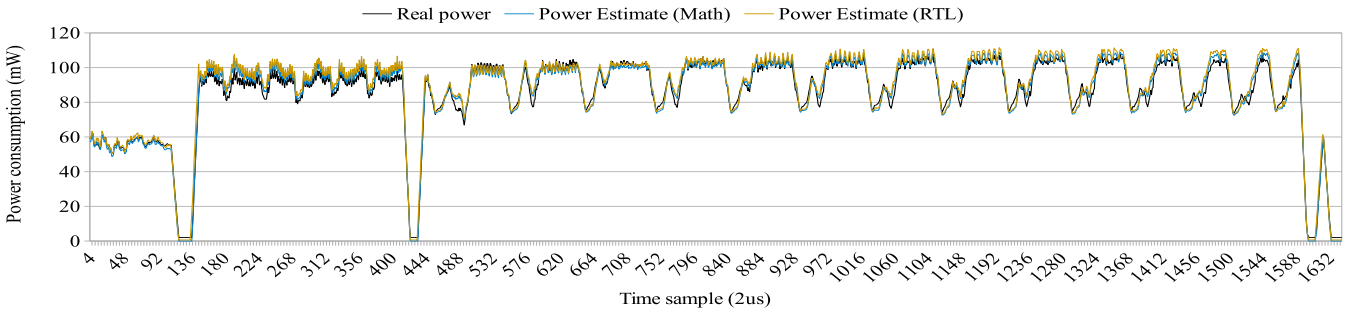
Figure 6 reports the power estimates for both *Math* and *RTL* as well as the real power consumption of *nu+* while executing the subset of the benchmarks in the validation set that employ two (Figure 6a) and four (Figure 6b) hardware threads. We note that the maximum RMSE accuracy error is limited to 1.22 (*Math*) and 1.79 (*RTL*) for the multi-threaded version of the benchmarks even if such configuration differs from the identification scenario where single-threaded micro-benchmarks have been used. Moreover, Figure 7 reports the power estimates, both the *Math* and the *RTL*, and the real power consumption of *nu+* while executing the single thread version of the benchmarks in the validation set on a *nu+* instance operating at 20MHz (Figure 7a) and 62.5MHz(Figure 7b). Similar to the multi-threaded scenario, we report the maximum RMSE accuracy error that is limited to 1.17 (*Math*) and 1.19 (*RTL*). The flexibility of the identified power model in providing accurate power estimates for a broad range of use-case scenarios confirms the quality of the identified power model itself and the effectiveness of the methodology to design the micro-benchmarks that have been used to collect the switching activity information.

5.2. PowerTap power and area overheads

We evaluated the area and power overheads due to the final RTL implementation of the power monitoring infrastructure of the *nu+* platform targeting the *Xilinx Artix 7 XC7A200T FPGA*. Table 2 reports the area and the power overheads due to a single power counter that is monitoring a signal of the target. In particular, we considered different sizes for the monitored signal to show the scalability property of the power counters. The reported power values are extracted considering the *maximum*



(a) 20MHz operating frequency ($RMSE_{Math}$= 0.46 $RMSE_{RTL}$= 0.44).Results are obtained by concatenating the executing of the benchmarks (1)-(14) (see Table 1).



(b) 62.5MHz operating frequency ($RMSE_{Math}$= 1.17 $RMSE_{RTL}$= 1.19).Results are obtained by concatenating the executing of the benchmarks (1)-(14) (see Table 1).

Figure 7: Power estimates over time for the mathematical and RTL implementation of the identified power model. The single thread version of the benchmarks in the validation are executed on a *nu+* instance that operates at different frequencies.

Table 2: Power and area overheads of a *PowerTap* counter that monitors an architectural signal. Different sizes of the signal are considered. Both HWC and STC switching activity counters are considered. The power consumption is calculated assuming the maximum switching activity of the monitored signal.

Monitored signal width (bits)	Counter type	Area		Power with max switch activity (mW)
		LUTs	Flip flops	
32	HWC	98	68	4.7
	STC	28	53	2.3
64	HWC	146	103	9.3
	STC	39	85	4.1
128	HWC	239	170	17.7
	STC	60	149	7.7

switching activity, that is defined as the toggle of the logic state of each bit of the monitored signal at each clock cycle. Moreover, we set a temporal resolution of 2us and an operating frequency of 50 MHz, thus collecting the switching activity across 100 clock cycles to compute the power value. We note that the number of used flip flops of the HWC power counter increases with the size of the monitored signal and the number of cycles in the time window. In contrast, the number of used flip flops for an STC power counter is function of the number of clock cycles in the time window, since for each clock cycle at most a unit increment is added to the counter regardless of the number of toggled bits in the monitored signal.

As expected, the power consumption is greatly affected by the counter type, i.e., HWC or STC, since the same switching activity in the monitored signal induces a switching activity that is higher for the HWC counter.

The final results are obtained from the implementation of the power monitoring infrastructure within the *nu+* platform targeting the *Xilinx Artix 7 XC7A200T FPGA*. The area overhead is limited to 9.95% LUT and 3.87% flip flops. Moreover, the average power overhead due to the power monitoring infrastructure is 12.17 mW.

6. Conclusions

This work presented *PowerTap*, a novel power modeling methodology to design an all-digital online power monitoring infrastructure for complex GPU-like processor architectures. In contrast with the state of the art, *PowerTap* exploits the ISA of the target to extract few power-aware micro-benchmarks each able to selectively stress a part of the target. Such approach minimizes the computation time to extract the switching activity to identify the power model. Moreover, the novel top-down visiting approach iteratively identifies the power model for the target trading the accuracy for the area and power overheads.

We assessed *PowerTap* against a hardware multi-threaded SIMD processor considering a validation set made of 9 benchmarks for which the scalar and SIMD implementations as well as their software parallelization using one, two and four threads generated 24 software scenarios. Moreover, we assess the validity of *PowerTap* considering three operating frequencies for the target platform. The results confirm an $RMSE_{RTL}$ accuracy error limited to 1.79. We note that such accuracy errors are collected by using for all the architectural and benchmark configurations the same power model that has been identified once starting from the switching activity information extracted from

the execution of single-thread benchmarks and a single operating frequency. This fact qualifies the experimental robustness of our solution that can deliver a flexible RTL power monitoring infrastructure remaining accurate against highly different operating conditions that are not-observed during the power model identification to minimize the computational time. Considering the implementation of the power monitoring solution for the *nu+* SIMD platform on a *Xilinx Artix 7 XC7A200T FPGA* the area overhead is limited to 9.95% LUT and 3.87% flip flops. Moreover, the average power overhead is within 12.17 mW, thus making our solution applicable for self-adaptive low-power target platforms.

7. Acknowledgments

This work was partially supported by the EU H2020 Research and Innovation Programme under agreement no. 671668 ("MANGO" project http://www.mango-project.eu/).

References

[1] D. Zoni, A. Canidio, W. Fornaciari, P. Englezakis, C. Nicopoulos, Y. Sazeides, Blackout, J. Parallel Distrib. Comput. 104 (C) (2017) 130–145.

[2] D. Zoni, J. Flich, W. Fornaciari, Cutbuf: Buffer management and router design for traffic mixing in vnet-based nocs, IEEE Transactions on Parallel and Distributed Systems 27 (6) (2016) 1603–1616.

[3] S. Bartolini, P. Foglia, C. A. Prete, Exploring the relationship between architectures and management policies in the design of nuca-based chip multicore systems, Future Generation Computer Systems 78 (2018) 481–501.

[4] D. Zoni, L. Colombo, W. Fornaciari, Darkcache: Energy-performance optimization of tiled multi-cores by adaptively power-gating llc banks, ACM Trans. Archit. Code Optim. 15 (2) (2018) 21:1–21:26.

[5] J. Krzywda, A. Ali-Eldin, T. E. Carlson, P.-O. stberg, E. Elmroth, Power-performance tradeoffs in data center servers: DVFS, CPU pinning, horizontal, and vertical scaling, Future Generation Computer Systems 81 (2018) 114 – 128.

[6] D. Zoni, F. Terraneo, W. Fornaciari, A control-based methodology for power-performance optimization in NoCs exploiting dvfs, Journal of Systems Architecture 61 (5) (2015) 197 – 209.

[7] M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra, S. Vishin, Power-performance modeling on asymmetric multi-cores, in: Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems, CASES '13, IEEE Press, Piscataway, NJ, USA, 2013, pp. 15:1–15:10.

[8] R. A. Bridges, N. Imam, T. M. Mintz, Understanding GPU power: A survey of profiling, modeling, and simulation methods, ACM Comput. Surv. 49 (3) (2016) 41:1–41:27.

[9] D. Zoni, L. Colombo, W. Fornaciari, DarkCache: Energy-performance optimization of tiled multi-cores by adaptively power gating LLC banks, ACM Trans. Archit. Code Optim. (2018) 1:25.

[10] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, D. Burger, Dark silicon and the end of multicore scaling, in: Proceedings of the 38th Annual International Symposium on Computer Architecture, ISCA '11, ACM, New York, NY, USA, 2011, pp. 365–376.

[11] M. Najem, P. Benoit, M. E. Ahmad, G. Sassatelli, 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 36 (7) (2017) 1153–1166.

[12] W. L. Bircher, L. K. John, Complete system power estimation using processor performance events, IEEE Transactions on Computers 61 (4) (2012) 563–577.

[13] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, 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 36 (1) (2017) 106–119.

[14] D. Zoni, L. Cremona, W. Fornaciari, PowerProbe: Run-time power modeling through automatic RTL instrumentation, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2018, 2018, pp. 749–754.

[15] D. J. Pagliari, V. Peluso, Y. Chen, A. Calimera, E. Macii, M. Poncino, All-digital embedded meters for on-line power estimation, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2018, 2018, pp. 743–748.

[16] J. Peddersen, S. Parameswaran, CLIPPER: Counter-based low impact processor power estimation at run-time, in: 2007 Asia and South Pacific Design Automation Conference, 2007, pp. 890–895.

[17] X. Jiang, P. Dutta, D. Culler, I. Stoica, Micro power meter for energy monitoring of wireless sensor networks at scale, in: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN '07, ACM, New York, NY, USA, 2007, pp. 186–195.

[18] S. Bhagavatula, B. Jung, A power sensor with 80ns response time for power management in microprocessors, in: Proceedings of the IEEE 2013 Custom Integrated Circuits Conference, 2013, pp. 1–4.

[19] A. Pathania, A. E. Irimiea, A. Prakash, T. Mitra, Power-performance modelling of mobile gaming workloads on heterogeneous MPSoCs, in: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), 2015, pp. 1–6.

[20] G. Singla, G. Kaur, A. K. Unver, U. Y. Ogras, Predictive dynamic thermal and power management for heterogeneous mobile platforms, in: DATE, 2015, pp. 960–965.

[21] W. L. Bircher, L. K. John, Complete system power estimation using processor performance events, IEEE Transactions on Computers 61 (4) (2012) 563–577.

[22] C. Isci, M. Martonosi, Runtime power monitoring in high-end processors: Methodology and empirical data, MICRO 36, IEEE Computer Society, Washington, DC, USA, 2003, pp. 93–.

[23] K. Singh, M. Bhadauria, S. A. McKee, Real time power estimation and thread scheduling via performance counters, SIGARCH Comput. Archit. News 37 (2) (2009) 46–55.

[24] R. Rodrigues, A. Annamalai, I. Koren, S. Kundu, A study on the use of performance counters to estimate power in microprocessors, IEEE Transactions on Circuits and Systems II: Express Briefs 60 (12) (2013) 882–886.

[25] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, S. Matsuoka, Statistical power modeling of GPU kernels using performance counters, in: International Conference on Green Computing, 2010, pp. 115–122.

[26] S. Song, C. Su, B. Rountree, K. W. Cameron, A simplified and accurate model of power-performance efficiency on emergent GPU architectures, in: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, 2013, pp. 673–686.

[27] J. Chen, B. Li, Y. Zhang, L. Peng, J. k. Peir, Statistical GPU power analysis using tree-based methods, in: 2011 International Green Computing Conference and Workshops, 2011, pp. 1–6.

[28] S. Hong, H. Kim, An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness, SIGARCH Comput. Archit. News 37 (3) (2009) 152–163.

[29] S. Hong, H. Kim, An integrated GPU power and performance model, SIGARCH Comput. Archit. News 38 (3) (2010) 280–289.

[30] J. Gustafsson, A. Betts, A. Ermedahl, B. Lisper, The malardalen wcet benchmarks: Past, present and future., in: B. Lisper (Ed.), WCET, Vol. 15 of OASICS, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2010, pp. 136–146.
URL `http://dblp.uni-trier.de/db/conf/wcet/wcet2010.html\#GustafssonBEL10`

[31] C. Bienia, S. Kumar, K. Li, Parsec vs. splash-2: A quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors, in: Proceedings of the 2008 International Symposium on Workload Characterization, 2008.

[32] A. Romanoni, A. Delaunoy, M. Pollefeys, M. Matteucci, Automatic 3d reconstruction of manifold meshes via delaunay triangulation and mesh sweeping, in: Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), 2016, p. 7477650.

[33] A. Romanoni, M. Ciccone, F. Visin, M. Matteucci, Multi-view stereo with single-view semantic mesh refinement, in: Computer Vision Workshop (ICCVW), IEEE International Conference on, 2017, pp. 706–715.