Automatic identification and hardware implementation of a resource-constrained power model for embedded systems

Luca Cremona, William Fornaciari and Davide Zoni

DEIB - Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133, Milano, Italy
E-mail: {luca.cremona,william.fornaciari, davide.zoni}@polimi.it

ARTICLE INFO

Keywords:
RTL, Low Power, Run-Time Resource Management, Power Modeling, Embedded Systems

ABSTRACT

In modern embedded systems, the use of hardware-level online power monitors is crucial to support the run-time power optimizations required to meet the ever increasing demand for energy efficiency. To be effective and to deal with the time-to-market pressure, the presence of such requirements must be considered even during the design of the power monitoring infrastructure. This paper presents a power model identification and implementation strategy with two main advantages over the state-of-the-art. First, our solution trades the accuracy of the power model with the amount of resources allocated to the power monitoring infrastructure. Second, the use of an automatic power model instrumentation strategy ensures a timely implementation of the power monitor regardless the complexity of the target computing platforms. Our methodology has been validated against 8 accelerators generated through a High-Level-Synthesis flow and by considering a more complex RISC-V embedded computing platform. Depending on the imposed user-defined constraints and with respect to the unconstrained power monitoring state-of-the-art solutions, our methodology shows a resource saving between 37.3% and 81% while the maximum average accuracy loss stays within 5%, i.e., using the aggressive 20us temporal resolution. However, by varying the temporal resolution closer to the value proposed in the state of the art, i.e. in the range of hundreds of microseconds, the average accuracy loss of our power monitors is lower than 1% with almost the same overheads. In addition, our solution demonstrated the possibility of delivering a resource constrained power monitor employing a 20us temporal resolution, i.e., far higher the one used by current state-of-the-art solutions.

1. Introduction

Considering the large variety of constraints imposed by current applications, the use of specialized hardware accelerators [15, 14] and optimized multi-cores represents the de-facto solution to deliver efficient computing platforms coping with the ever-increasing time-to-market.

However, the complexity of such computing platforms and the variability of the requirements due to the executed applications, motivate the use of run-time power-aware optimization techniques to trade the computational power and the energy consumption. In this scenario, online power monitors are the de-facto solution to estimate the power state of the system at run-time and, thus, to effectively support any optimization technique. Online power monitors deliver a periodic power estimate by leveraging the relationship between the power consumption and the internal switching activity of the target computing platform. The realization of a power monitor encompasses two steps. First, the power model identification stage finds out the mathematical relationship between the power consumption and the switching activity of the computing platform. Second, a power monitor is designed by implementing the mathematical power model into the target platform. The possibility of exploiting such relationship at different abstraction levels motivates the evaluation of both software- and hardware- implemented power monitors, where each of them offers a different trade-off between the accuracy of the power estimates and the implementation complexity. Software power monitors are applications that provides online power monitoring capabilities when the RTL description of the platform is not accessible, at the cost of a non-negligible performance overhead as well as low accuracy and limited temporal resolution for the power estimates. Hardware power monitors rely on dedicated hardware to deliver highly accurate power estimates at high temporal resolution and without performance overhead at the cost of changing the RTL description of the computing platform.

It is important to note that regardless of their software- or hardware-level implementations, the state-of-the-art power monitors are meant to solely minimize the prediction error of the power estimates without accounting for the imposed overheads in terms of area, power, timing and performance.
Contributions - Based on the fact that hardware solutions represent the de-facto choice to provide online power monitoring capabilities to modern embedded systems [6, 5, 11], this paper presents a novel methodology to design efficient power monitors for generic computing platforms. Such monitors minimize the prediction error within a configurable resource budget, thus providing two major contributions to the state-of-the-art:

- *Resource-constrained power monitor* - We designed a set of highly efficient and fully characterized hardware building blocks to implement the power monitor. To this end, our methodology allows the user to accurately configure the resource budget devoted to the power model. Experimental results on 8 High-Level-Synthesis (HLS) generated designs and a RISC-V System-on-Chip, demonstrated that our methodol-

ORCID(s): 0000-0001-6613-5846 (L. Cremona); 0000-0001-8294-730X (W. Fornaciari); 0000-0002-9951-062X (D. Zoni)

ogy always delivers a power model within the specified resource budget with an average accuracy error lower than 5%, thus aligned with state-of-the-art solutions.

- *Automatic power monitor implementation* - To optimize the time-to-market, our methodology allows to automatically augment the Register-Transfer-level (RTL) of a generic target computing platform with the hardware description of the power monitor. In particular, we can easily instrument HLS netlists for which the hand-made implementation has to face with the fact that such descriptions are not meant to be human readable. In addition, the proposed methodology allows to configure the temporal resolution of the power monitor between 20us and 500us without affecting the timing of the monitored design considered in our analysis.

The rest of the paper is organized in four parts. Section 2 presents the background and the state-of-the-art related to the online power monitoring systems. The proposed resource-constrained power monitor design flow is discussed in Section 3, while results considering the accuracy of the power estimates as well as area, power and timing overheads are presented in Section 4. Conclusions are drawn in Section 5.

2. Related works

Considering a generic computing platform, the design of the online power monitor requires *i)* to identify the mathematical formulation of the power model, and *ii)* to implement such model into the computing platform. A summary of the highlights and open issues of the state of the art for both aspects is reported below.

Power model identification - The state-of-the-art contains several solutions to monitor the power consumption of the platform at run-time, either at software- or hardware-level. *Software*-implemented power monitors leverage the relationship between the power consumption of the platform and few coarse grain operations that are monitored by means of existing architectural performance counters [2, 10, 8]. Such schemes offer an online power monitoring solution without the need to change the microarchitecture of the computing platform. However, the corresponding power monitors are executed in the form of software applications, thus negatively affecting the overall performance of the computing platform.

To minimize the performance penalty of software-level power monitors, *hardware*-implemented power models emerged as a viable alternative to offer accurate power estimates at run-time, with no performance overhead for the target computing platform [11, 5, 6]. Typically, they use ad-hoc hardware structures to monitor the switching activity of selected signals at circuit-level and to deliver the power estimates. The absence of performance penalty is balanced by a non-negligible resource overhead to implement the power monitoring infrastructure. We note that all the online power monitors in the state-of-the-art share two key properties. First, the family of the linear models is the most employed one, due to its simplicity and high accuracy in estimating the power consumption of the computing platforms. Second, each proposal aims at improving the accuracy of the power estimates without accounting for the power monitoring cost in terms of resource utilization, performance penalty and power overhead. In contrast with the state-of-the-art, this work proposes a hardware-level design methodology able to trade the accuracy of the power monitor with all its implementation overheads starting from the model identification.

Power monitor implementation - Concerning the crucial need to provide automatic implementation strategies especially for hardware-level power monitors, the state-of-the-art contains few proposals in this direction. In [4] the authors present a framework that automatically extracts the power model from a generic computing platform. However, the automatic implementation of the corresponding power monitor is left as a future work. The approach presented in [11] describes a methodology to deliver a hardware-level power monitor for a custom multi-core embedded system. Despite the use of a semi-automatic solution to identify the power model, the authors rely on a manual implementation of the corresponding power monitor and it does not account for the power monitor overheads.

In contrast to the state-of-the-art, this work delivers a solid framework leveraging the Yosys open-source synthesizer, to automatically implement a resource-constrained power monitor in any computing platform for which the hardware description is available.

3. Methodology

Figure 1 depicts the proposed toolchain to automatically generate a hardware-level resource-constrained power monitor for generic computing platforms. Starting from the hardware description of the computing platform (`RTL-source`), its corresponding set of design constraints (`designConstr`) and the user-defined constraints (`usrDefConstr`), the flow outputs an hardware description file containing the target design augmented with the power monitor (`netlist power monitor`). We note that the design constraints are expressed in terms of the standard timing, e.g., required operating frequency, and physical, e.g. pinout, requirements for the implemented computing platform. In contrast, the user-defined constraints allow the user to specify the number of allowed resources to implement the power monitoring infrastructure.

The design and implementation of the power monitor is organized in two separate steps: *i)* power model identification (`PwrModel`), and *ii)* power monitor implementation (`PwrMonImpl`). The `PwrModId` stage identifies a power model that ensures the smallest accuracy error within the resource budget, by leveraging: *i)* the switching activity of the signals, *ii)* the power consumption of the computing platform, *iii)* the user imposed resource constraints, and *iv)* the profiled overhead of the basic blocks we use to realize the power model. Finally, the `PwrMonImpl` stage augments the RTL description of the computing platform with a power monitoring infrastructure

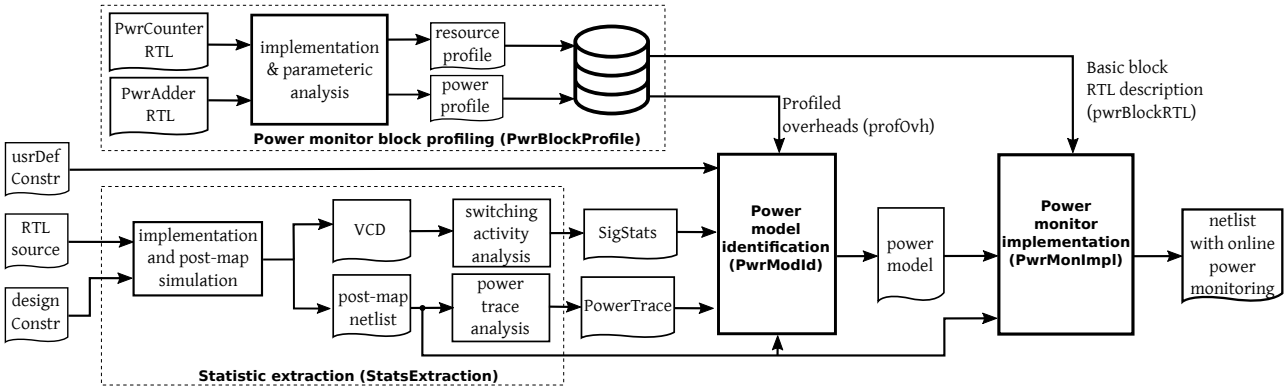


Figure 1: Overview of the proposed resource-constrained automatic power model identification and instrumentation flow.

which implements the identified power model.

The rest of this section is organized in three parts. Section 3.1 discusses the power model identification stage. Section 3.2 presents the architecture and the profiling of the building blocks to deliver a resource-constrained power monitoring infrastructure. Last, Section 3.3 details the process to automatically implement the power monitor.

3.1. Power Model Identification

The power model identification stage produces a linear power model starting from four inputs: *i)* the power consumption traces, *ii)* the corresponding switching activity of the target computing platform, *iii)* the user-defined resource constraints, and *iv)* the profile of the basic blocks of our power modeling infrastructure. In particular, the power model employs the switching activity of a selected subset of input and output wires of the design modules to derive the power estimates.

At first, we perform a multicollinearity analysis to group signals showing a too similar switching activity. For each identified group one signal is kept, while all the others are discarded. Starting from the kept signals, we identify the power model by means of an exhaustive search procedure constrained to the user-defined requirements. In particular, a linear regression procedure returns the model coefficients associated to the corresponding selected signals. In order to obtain the required signal information, the statistics extraction module (`StatsExtraction`) implements a two steps procedure. First, the post-implementation timing simulation produces Value Change Dump (VCD) file. Second, the obtained VCD file is analyzed to obtain the power consumption trace (`PowerTrace`) and the switching activity (`SigStats`) of the each signal in the target computing platform.

Considering the target computing platform, the user-defined resource constraints specifies, for each resource type, the maximum percentage utilization that is allowed to implement the power model. To support the identification of a resource-constrained power model, the identification stage receives in input the profiled information encompassing power consumption, resource utilization and timing for each basic block of our power monitoring infrastructure (see the detailed description in Section 3.2).

Mathematical formulation - To describe the power model, we adopted the formulation proposed in [11]. Its formal definition is expressed by Equation 1.

$$\hat{p}_t = \sum_i c_i \times s_{i,t}^{SVC} + \sum_j c_j \times s_{j,t}^{HWC} \quad (1)$$

At time t, the power estimate ($\hat{p}_t$) is computed as the weighted sum of the switching activity of a carefully identified subset of signals, i.e., $s_{i,t}$ and $s_{j,t}$, where the values of the coefficients c_i and c_j are set by the identification algorithm. To improve the *quality* of the information associated with the switching activity used to calculate the power estimates, our power model formulation can employ the switching activity of each signal in the form of either the Hamming Weight (`HWC`) or the Single Variation Count (`SVC`). For each clock cycle, the `HWC` counts the number of the bits of the signals that flipped with respect to the previous sampled value. In contrast, `SVC` measures the switching activity, as 0 or 1, if the current signal value is changed or not with respect to the previously sampled value. Even if the HWC and SVC statistics of the same physical signal can be hardly employed in the same power model due to correlation issues, the availability of multiple models to measure the switching activity can improve the final accuracy of the power model.

Power modeling: algorithmic description - The power model identification algorithm employs a recursive approach to implement the top-down hierarchical visit of the target design (see Algorithm 1). The algorithm takes five inputs: *i)* the top module of the design, *ii)* the user-defined constraints, where each of them is specified as a fraction of the same resource type used by the target design also including an upper bound that specifies the maximum acceptable accuracy error for the identified model, *iii)* profiled information from the `PwrBlockProfile` module, as well as the *iv)* the switching activity and the *v)* power traces of the target computing platform. The mathematical formulation of the identified power model represents the output of the algorithm. We note that the power model is a list of triples, where each triple is defined as the name of a signals of the design to probe, the associated coefficient of the power model, and the employed

Algorithm 1 Top-down hierarchical visiting algorithm.

1: **function** $[model, \mathbf{e}]$ VISIT($M_0, R, \mathbf{e}_{Th}$)
2: $[mId, \mathbf{e}_{M_0}] = ComputePwrModel(M_0, R)$;
3: **if** $\mathbf{e}_{M_0} < \mathbf{e}_{Th}$ **then**
4: $model = mId$; $\mathbf{e} = \mathbf{e}_{M_0}$;
5: **else**
6: $container = sortByPower(M_0.m_0 \ldots M_0.m_N)$;
7: $mIdlList = [\,]$; $rIdList = [\,]$;
8: **for** $i = 1 : container.size$ **do**
9: $m_{Tmp} = container.pop(i)$;
10: $R_{Tmp} = R * mList(i).pwrRel$;
11: $[mId, \mathbf{e}] = VISIT(m_{Tmp}, R_{Tmp}, \mathbf{e}_{Th})$;
12: $[mCont, \mathbf{e}Cont] = VISIT(container, R - rIdList, \mathbf{e}_{Th})$;
13: $mIdList.add(mId)$;
14: $rIdList.add(R_{Tmp})$;
15: **if** $compErr([mIdList\ mCont]) < \mathbf{e}_{Th}$ **then**
16: $model = [mIdList\ mCont]$;
17: $\mathbf{e} = compErr(model)$;
18: $break$;
19: **end if**
20: **end for**
21: **end if**
22: **end function**

Algorithm 2 Power model computation for module m.

1: **function** $[model_{cur}, \mathbf{e}]$ COMPUTEPWRMODEL(m, R)
2: $\mathbf{e} = MAXERR$; $model_{cur} = [\,]$; $res_{cur} = MAXRES$;
3: **for** $i = 1 : size[m.\vec{I}\ m.\vec{O}]$ **do**
4: $C = \binom{[m.\vec{I}\ m.\vec{O}]}{i}$;
5: **for** $j \in C$ **do**
6: $res_{new} = computePwrMonRes(j)$;
7: **if** $res_{new} < R$ **then**
8: $[model_{new}, \mathbf{e}_{new}] = linReg(j, m.pwr)$;
9: $\mathbf{e}_{tmp} = k_1 * \frac{(\mathbf{e}_{new} - \mathbf{e})}{\mathbf{e}}$;
10: $res_{tmp} = k_2 * \frac{(res_{cur} - res_{new})}{res_{cur}}$;
11: **if** $res_{tmp} - \mathbf{e}_{tmp} > 0$ **then**
12: $\mathbf{e} = \mathbf{e}_{new}$; $res_{cur} = res_{new}$;
13: $model_{cur} = model_{new}$;
14: **end if**
15: **end if**
16: **end for**
17: **end for**
18: **end function**

switching activity that is selected as either the Single Variation Count (SVC) or the Hamming Weight Count (HWC).

Starting from the top module of the design, Algorithm 1 performs a top-down visit of the target design hierarchy to find the best power model within both the imposed resource constraint R and the accuracy upper bound limit $\mathbf{e}_{Th}$. The power model of module M_0 (mId_{M_0}) is accepted if the accuracy is within the allowed error $\mathbf{e}_{Th}$ (see lines 2-5 of Algorithm 1). Otherwise, the children modules of M_0 are sorted in a descending order according to their power consumption and an iterative power model identification exploration starts (see lines 8-20 of Algorithm 1). At each iteration in the `for-loop`, the first sub-module in the sorted `container` list is popped out and its power model is identified (see mId at line 11 of Algorithm 1). Moreover, the remaining modules in the `container` list are identified within a single power model. To this extent, a bi-partition of the modules is created. We note that the available resources are split proportionally to the two partitions. The module identified in isolation is added to the $mIdList$ list, i.e., the list containing the power models identified on the modules consuming the majority of the power in the target design. At line 15, Algorithm 1 checks if the aggregate error of the power models in the $mIdList$ plus the power model identified for the *container* modules is lower than the $\mathbf{e}_{Th}$ threshold. This approach allows to optimize the number of the implemented power modules, since the iterative algorithm tries to identify a dedicated power model for the modules than contribute the most to the power consumption, while a single aggregate power model is identified for the remaining ones. The recursive visit of the target design terminates either with a set of identified power models or when the container list is empty and the error is bigger than the imposed $\mathbf{e}_{Th}$ threshold.

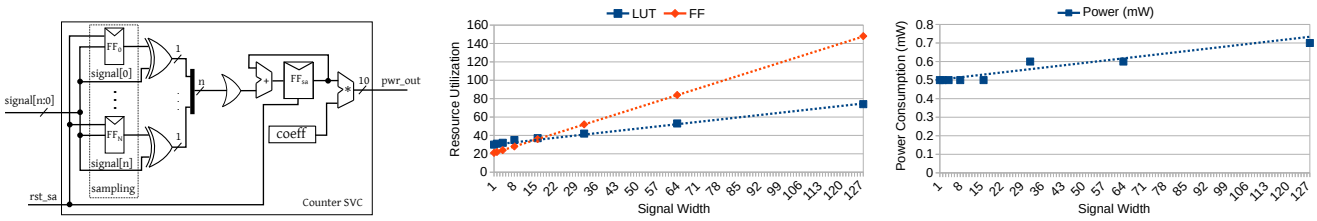
Algorithm 2 describes the power model identification procedure, i.e., the `ComputePwrModel` function, employed in lines 2 of Algorithm 1. Algorithm 2 takes as input a module of the design (`m`) and the maximum amount of resources R that can be employed for its power monitor implementation and outputs the identified power model ($\texttt{model}_{cur}$) and the associated accuracy error ($\mathbf{e}$). It is important to note that both HWC and SVC switching activity measurements are provided to Algorithm 2. It will be the algorithm itself that will chose the one which maximizes the accuracy of the identified model.

To obtain a simple formulation, the algorithm identifies the power model by leveraging the primary inputs and outputs of the target module. Moreover, the two nested for-loops (lines 3 and 5) drives the exploration to favor the power models that require a small number of probed signals. For each iteration of the outer for-loop, the statement at line 4 determines the set of all the combinations of i signals. For each combination j, the inner for-loop (line 5) computes the power model ($model_{new}$) and the accuracy error (e_{new}) by means of a linear regression procedure (see line 8 in Algorithm 2). Moreover, the estimated resource usage (res_{new}) for the identified power model is computed at line 6.

The identified power model ($model_{new}$) becomes the new candidate if the weighted sum of its marginal increment with respect to the current candidate model, i.e. $model_{cur}$, in terms of accuracy (line 9) and resource saving (line 10) is positive. In such a case, the identified power model ($model_{new}$) becomes the current candidate, i.e., $model_{cur}$ (see lines 11-14 in Algorithm 2).
We note that the algorithm allows tuning the weight of the marginal increments in favor of either a more resource-optimized or a more accuracy-optimized power model by means of two parameters, i.e., k_1 and k_2. Without lack of generality, this work used $k_1 = 0.9$ and $k_2 = 0.1$.

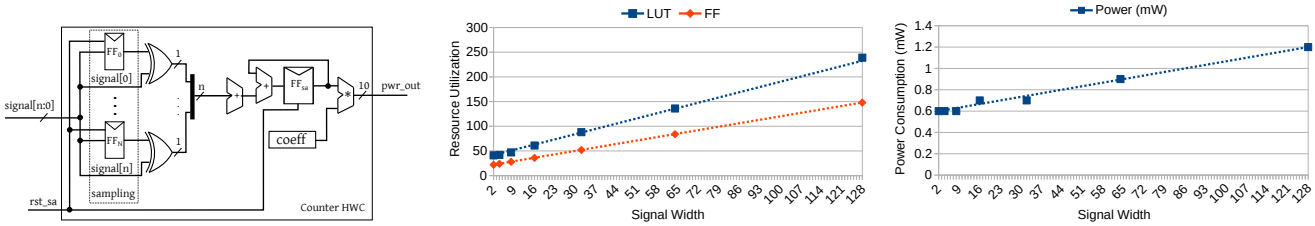
3.2. Profiles of the power monitor building blocks

The proposed power monitoring infrastructure is made of a reduced set of two highly optimized building blocks, i.e., the *power counter* and the *power adder*, for which we provide a complete characterization of the resource utilization and of the power consumption. The use of a small set of



(a) SVC Counter architecture (b) Resource utilization (c) Power consumption

Figure 2: Single Variation Count (SVC) counters architecture.



(a) HWC Counter architecture (b) Resource utilization (c) Power consumption

Figure 3: Hamming Weight Count (HWC) counters architecture.

fully profiled design elements allows to carefully fit the user-specified constraints during the power model identification process.

In the rest of this part, we discuss the characterization of the power counter and of the power adder in terms of resource utilization, timing and power consumption. All the results are extracted from a post-implementation analysis using Vivado 2018.2 and by targeting the Xilinx Artix7-100 FPGA reference technology. In particular, for each block we performed a comprehensive design-space exploration with respect to the temporal resolution of the power monitor and the width of the considered signals for each block of our power monitoring infrastructure.

Starting from the collected results, we observed a similar trend of the overheads across different temporal resolutions. To this end and without loss of generality, the rest of this part discusses the results in terms of the overhead variability due to the width of the probed signal while the temporal resolution is fixed to 20us, i.e., the most-aggressive one offered by our framework. We note that the timing of each building block exceeds the speed of 200MHz, thus no further discussion on such design metric is reported, since it will never create any limitation to the design under consideration.

Power counter - For each signal that is part of the identified power model, we instantiated a power counter module that collects the switching activity of the corresponding physical signal and, periodically, outputs the power contribution as the product between the switching activity and the coefficient associated with the signal in the identified power model. Figure 3 and Figure 2 depict the architectures and the profile of our power counter templates tailored to monitor the switching activities of the signal in terms of either Hamming Weight Count (`HWC`) or Single Variation Count (`SVC`).

The two power counters share a similar structure to store and to measure the switching activity. In particular, the accumulated switching activity of the monitored signal is stored in the FF_{sa} memory, and another dedicated memory element is used to store the corresponding power model coefficient (see `coeff` in both Figure 3a and Figure 2a). In particular, the rst_{sa} signal is used to reset the accumulated switching activity at the beginning of each time period. Moreover, the proposed power counter architectures sample the monitored signal once per clock cycle to measure the switching activity in terms of the signal variations in two consecutive sampled values (see `Sampling` block in Figure 3a and Figure 2a). Such monitoring strategy avoids measuring the glitching activity to provide a strong upper bound to the number of switches for each single-bit signal within a single clock cycle, i.e., 1 at the most. However, we note that the power monitor can still capture the non-negligible power contribution due to the glitching activity. In particular, the glitches are due to the particular microarchitecture of the implemented design. To this end, for each toggle in the monitored signal, the associated power observed by the power model identification strategy is the one of the actual signal toggle plus the power due to the related glitching activity, if any.

The most evident difference between the two power counter architectures is in the way the switching activity is accumulated, since an arithmetic adder is used in the HWC power counter, while an OR gate is used in the SVC counter.

For each power counter architecture, Figure 3b and Figure 2b) report the resource utilization, i.e., flip-flops (FF) and Look-Up-Tables (LUT), with respect to the width of the probed signal. Considering the SVC power counter architecture in Figure 2a, both the number of required FFs and LUTs is linear with the width of the monitored signal even if the number of FFs grows faster (see Figure 2b). We note that while the number of FFs is dominated by the width of

the signal showing a linear coefficient equal to 1, the number of LUTs required to perform the SVC computation, i.e., the boolean-OR, grows slower. In contrast, the number of LUTs dominates the resources required to implement the HWC power counter due to increasing complexity of performing the arithmetic addition as the width of the probed signal increases. In particular, the number of FFs still grows linearly with the width of the probed signal and it is comparable with the one required for the SVC power counter. We note that, given the width of the probed signal, the marginal difference in the number of required FFs between the two power counter architectures is due to the memory where the switching activity is accumulated during the time epoch. As expected, the power consumption of the two power architectures slightly grows with the increase of the width of the probed signal, while the HWC power counter shows an higher power consumption than the SVC one due to the additional design complexity of the arithmetic addition.

Power adder - The power adder is a combinatorial block that is meant to sum up all the power contributions from the instantiated power counters to deliver the periodic power estimate.

We note that the impact of the power adder in terms of resource and power overheads is very limited with respect to the rest of the power monitor architecture. To minimize the number of dimensions of our design-time exploration without any flexibility loss, each input to the power adder, i.e., the power counter output, is 10-bit signal while a 12-bit signal is used to size the output of the power adder. By trading the precision and the dynamic range of power measures our solution can address computing platforms ranging from embedded systems up to High-Performance Computing (HPC) platforms. Such configuration can be achieved by changing the unit employing to report the power consumption in the power traces (see `PowerTrace` in Figure 1).

When targeting the embedded system domain, the use of a mW precision to report the power consumption in the power traces allows each power counter to deliver a probed power consumption between 0 and 1024mW while the output of the power adder ranges between 0 and 4096mW. In contrast, the use of a Watt precision to report the power consumption in the power traces allows each power counter to report a probed power between 1 and 1024W while the output of the power adder can top up to 4096W.

While this paper is focused on the embedded system domain, we note that our benchmarks, and in general all state-of-the-art power monitoring solutions in the area of embedded systems, show that the number of signals employed to identify the power model is usually lower that 16. Moreover, the power contribution due to a single power counter probing a 128-bit physical signal using the HWC power counter, i.e., the most complex one, is always far below 1W. To this end we employed the mW precision to report the power consumption within the power traces. Starting from such design-time configuration, we observed a number of LUTs due to the power adder that ranges from 32 to 88 when the number of 10-bit input signals ranges from 2 to 32. Moreover, the power consumption of the power adder is between 0.4 mW and 0.5mW moving from 2 to 32 inputs.

3.3. Power Monitor Implementation

Starting from the mathematical formulation of the identified power model, the `PwrMonImpl` stage delivers the final netlist of the target design augmented with the power monitor. In particular, the RTL description of the power monitor, that is target independent, is added to the netlist of the computing platform and then a last incremental implementation pass is required to deliver the final implementation netlist. Such aspect of technology independence is crucial and it is enabling the use of our methodology in any hardware design flow.

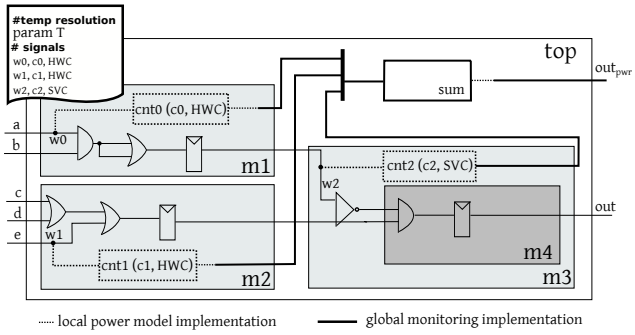


Figure 4: Example of a power monitor that will be added to a target design.

The power monitor implementation stage realizes the identified power model in two steps, moving from the local to the global power monitor. For each triple of values in the mathematical formulation of the identified power model, the local power monitor stage implements a customized power counter. The power counter is customized in terms of the width of the monitored signal, the coefficient associated with the signal, and the type of switching activity and corresponding counter selected for the signal. In contrast, the global power monitor stage implements a single power adder in the top module of the target design and connects its inputs with the output of each implemented power counter. Figure 4 details the instrumentation of a three-counter power model into a target design where the top module (`top`) implements three sub-modules (`m1`, `m2` and `m3`). The three power counters specified in the power model are implemented to monitor the `w1`, `w2`, and `w3` signals in the target. For each power counter the associated coefficient (`c0`, `c1`, and `c2`) and the type of switching activities are specified in the power model. Last, the power adder instance (`sum`) delivers the periodic power estimate. We note that, in addition to the width of each probed signal and the type between either HWC or SVC to account for the switching activity, the power monitor implementation stage makes use of the temporal resolution (see `param` T in Figure 4) to correctly size each power counter.

Table 1
Resource utilization for the considered benchmarks, without the implementation of any power monitor.

Benchmark	DSP	LUT	FF
fibonacci	0	185	101
crc	0	438	297
aes-Enc	0	491	233
aes-Dec	0	1037	137
expint	2	1361	1334
sqrt	2	2299	1682
qsort	0	2775	1323
fft	78	14974	10776
RISC-V	10	7868	5606
average	10.2	3492	2378.7

4. Results

This section reports the assessment of the proposed methodology focusing on the accuracy and the resource utilization of the hardware-level power monitoring infrastructure. The experimental setup is discussed in Section 4.1, while the accuracy of the power model estimates and the resource utilization results are detailed in Section 4.2. Section 4.3 will analyze the trend of the design metrics, by varying the temporal resolution of the power estimates.

4.1. Experimental setup

To demonstrate the value and the feasibility of our solution we employed a set of HLS-generated hardware accelerators and a complete embedded RISC-V-based system-on-chip computing platform.

Power monitor generation for the use-case scenarios - The Bambu open-source HLS tool from the Panda project [7] has been selected to translate eight WCET kernels [3] into their corresponding Verilog RTL descriptions. Moreover, we employed the System-on-Chip (SoC) from [9] to further assess the effectiveness of our solution against complex hand-coded hardware designs. The SoC features a 32-bit bus-based architectures and an in-order, five-stage RISC-V CPU using an Harvard memory architecture. The CPU offers the hardware support to single-precision floating point as well as to integer divisions and multiplications. A UART peripheral and the SoC debugger completes the SoC architecture.

In particular, Table 1 reports the resource utilization of each considered design in terms of Digital Signal Processing (DSP) tiles, LUTs, and flip-flops (FFs) before the power monitor instrumentation. For each hardware design, we implemented the methodology presented in Section 3 in five steps:

1. Vivado 2018.2 has been employed to generate both the post-synthesis and the post-implementation netlist of each considered hardware design.
2. The switching activities and the power traces to identify the power models have been collected from the post-implementation simulations.
3. Starting from the switching activities, the netlist, the power traces and the user-imposed constraints, we employed Matlab-2019a to implement the power model identification algorithms.
4. The identified power model as well as the post-synthesis netlist of the corresponding hardware design, have been fed to the Yosys open-source synthesizer which delivered the netlist augmented with the power monitor.
5. Vivado 2018.2 post-implementation has been executed on the netlist generated by Yosys, to deliver the final design implementation.

The functional validation has been assessed by prototyping each design on the Digilent Nexys4-DDR board featuring a Xilinx Artix7 XC7A100TCSG324-1 FPGA targeting a 100MHz implementation frequency. For the RISC-V SoC, we employed the already available prototyping setup [9]. In contrast, we developed a novel hardware controller employing a simple UART-based request-response protocol to interface the HLS-generated designs under test with the host computer. By means of such architecture for debugging, we were able to assess the functionality of the hardware accelerators instrumented with the power monitors. It is worth noticing that the operating frequency of the power monitor exceeds 200MHz, hence the maximum clock speed is limited by the rest of the computing platform. In fact, the power monitor is working in parallel to the monitored design, thus leaving unchanged the original critical path.

Quality metrics - The Root Mean Square Error (RMSE) quality metric has been employed to measure the accuracy of the proposed power monitor. In particular, we measure the distance between each of the n samples of the power estimates trace ($\hat{p}_i$) and the corresponding power consumption sample (p_i), provided by Xilinx technology libraries (see Equation (2)).

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}\left(p_i - \hat{p}_i\right)^2}{n}} = \sqrt{\frac{\Sigma_{i=1}^{n}\left(e_i\right)^2}{n}} \quad (2)$$

By employing the formulation in Equation (3), the pointwise real power consumption (p_i) is defined as the addition between the power estimate ($\hat{p}_i$) and an error (e_i). Considering the error as a random variable with Gaussian distribution, zero-mean and with a σ standard-deviation, the RMSE actually defines the σ quantity, i.e., the variability of the power estimates over the actual power consumption. To this end, Equation (4) defines the percentage RMSE normalized to the average power consumption ($RMSE_{norm}$) that we use as our quality metric for the accuracy of the power estimates.

$$p_i = \hat{p}_i + e_i \quad (3)$$

$$RMSE_{norm}[\%] = \frac{RMSE}{\mathbb{E}(p)} \times 100 \quad (4)$$

To demonstrate the effectiveness of the resource constraints, we discuss different power monitors implemented by fixing

Table 2
Experimental results for the benchmarks considering three constraints on the use of resources (5%, 10%, 20%) for the power monitoring infrastructure and without any boundary as in the state-of-the-art. A dashed gray cell means that the cost of the power monitor exceeds the boundary and it is discarded. The overheads on the use of resources and power are relative % values expressed over the size of the original unmonitored designs reported in Table 1.

Name	Unconstrained					Constraint 20%					Constraint 10%					Constraint 5%				
	# HWC	# SVC	% of used LUT	% of used FF	Pwr Ovh	# HWC	# SVC	% of used LUT	% of used FF	Pwr Ovh	# HWC	# SVC	% of used LUT	% of used FF	Pwr Ovh	# HWC	# SVC	% of used LUT	% of used FF	Pwr Ovh
fibonacci	1	2	38.2	20.4	33.5	0	1	16.2	8.3	13.6	-	-	-	-	-	-	-	-	-	-
crc	2	3	33.3	12.8	28.4	0	2	15.3	7.9	13.6	0	1	6.8	3.1	6.1	-	-	-	-	-
aes-Enc	1	2	20.2	10.3	17.1	0	2	12.2	7.1	9.6	0	1	6.1	2.9	5.4	-	-	-	-	-
aes-Dec	1	3	15.9	9.8	12.7	0	3	12.9	7.1	9.7	0	3	8.7	3.8	7.7	0	1	2.9	1.1	2.3
expint	1	2	8.9	4.8	6.9	1	2	8.9	4.8	6.9	1	2	8.9	4.8	6.9	0	1	3.1	1.7	2.8
sqrt	2	1	8.9	4.9	6.6	2	1	8.9	4.9	6.6	2	1	8.9	4.9	6.6	0	1	1.3	0.4	1.1
qsort	0	1	1.1	0.6	0.8	0	1	1.1	0.6	0.8	0	1	1.1	0.6	0.8	0	1	1.1	0.6	0.8
fft	1	2	0.7	0.5	0.7	1	2	0.7	0.5	0.7	1	2	0.7	0.5	0.7	1	2	0.7	0.5	0.7
RISC-V	19	10	18.4	10.2	15.9	18	10	17.9	9.9	15.2	10	4	9.1	4.1	8.2	4	4	4.6	2.2	4.1
average	3.11	2.9	16.2	8.2	13.6	2.4	2.7	10.1	6.0	7.4	2.1	2.6	5.7	3.6	4.3	1	2	2	1.3	1.9

the maximum Look-Up-Tables (LUT) budget. For each design, the LUT boundary is specified as a percentage of the total number of LUTs. We note that the proposed methodology allows to constrain any type of resource while we only discuss the LUT constraint, since the LUTs are the most critical resources in FPGA designs.

4.2. Accuracy and overhead results

Area and power overheads - For each benchmark design, Table 2 reports the occupation of the implemented power monitors considering different constraints for the LUTs and by assuming a temporal resolution of 20us, i.e., the highest our methodology can provide. In particular, we report results for the unconstrained power monitor (`Unconstrained`) and for three LUT-constrained power monitors (`Constr-20%`, `-10%`, `-5%`). Results demonstrate the effectiveness of our resource-constrained methodology where all the implemented power monitors respect the specified resource constraints. As expected, the use of an aggressive resource constrain on small designs, i.e., using less than 500 LUTs, prevents the implementation of the power monitor (it is still feasible but exceeding the user-defined constraints). For example, the methodology fails to implement the power monitor for some targets, such as `fibonacci`, `crc` and `aes-Enc` designs when the power monitor overhead is limited to 5% (see `Constraint-5%` results in Table 2). This is not a drawback of our methodology, but, on the contrary, it is a proof that our solution ensures a robust control over the resource overheads due to the power monitor. The power monitor for small designs can anyway be implemented simply by relaxing the resource constraint, as it is evident moving from 5% to 10% and then 20% overhead, where all the power monitors are implemented for all the designs. In particular, our investigation confirms that the smallest single-counter power monitor implementation requires a minimum of 30 LUTs, thus the target computing platform must use 600 LUTs at least to allow a power monitor constrained to 5%. We note that the smallest Xilinx Artix7 FPGA, i.e., Artix7-12, features 8000 LUTs thus confirming that a 600 LUT design is actually a tiny one.

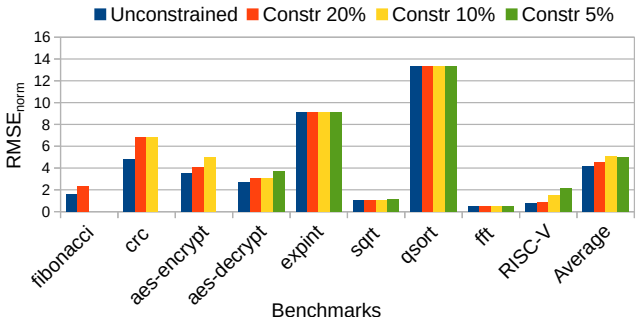


Figure 5: Accuracy loss of the power estimates with respect to the power traces extracted in clean room (Vivado 2018.2). Our policy considers acceptable only the solutions with a RMSE_{norm} error below 15.

As expected, the power overhead has a trend aligned to the resource overhead, hence it is decreasing moving from the unconstrained implementation to our constrained power models. The average power overhead is lower than 5% for a resource constraint of 10% showing an average of 1.9% when LUTs are constrained to 5%.

Accuracy - For each evaluated design, Figure 5 reports the accuracy of the implemented power monitors employing the 4 LUT-constraints, i.e., unconstrained, 5%, 10%, and 20%. The RMSE degrades, i.e., higher values, with the resource constraint, although it is always lower than 5% on average regardless of the imposed resource constraint. We notice that the obtained accuracy is aligned with other state of the art solutions [5, 6] for which it is not possible however to constraint the resources, and to select the temporal resolution during the automatic instrumentation process.

For example, by comparing the unconstrained solution to the one using a shallow LAT constraints, i.e., `Constr-20%`, the average resource saving is 37.3% while the average RMSE_{norm} degradation is limited to 0.4%. As expected, the benefit in-

creases with the severity of the imposed LUT-constraint, i.e. a negligible accuracy loss allows to sensibly drop down the used LUTs.

4.3. Exploring different time resolutions

To provide a wider evaluation of our methodology, this section discusses the trend of the considered design metrics by varying the temporal resolution of the power estimates. In particular, we analyze area, i.e., LUTs and FFs (see Figure 6a and 6b), and power overheads (see Figure 6c) as well as the accuracy loss by considering different temporal resolutions, i.e., 20us, 100us, 200us, 300us, 400us, and 500us (see Figure 6d). We note that the investigated temporal resolutions are aligned with the ones employed in state-of-the-art solutions to optimize the energy-performance trade-off in single- [13] and multi- [12] cores.

By lowering the temporal resolution, we observe a marginal increase for both the resource utilization and the power overhead. Such increase is due to the need to store more statistics, i.e., for a longer time period, related to the switching activity of the probed signals in the circuit. In particular, more FFs and LUTs are required to perform larger multiplications and additions to deliver the power estimates. However, considering each implemented power monitor, the area and power overhead increase is always lower than 5% across the entire range of the analyzed temporal resolutions.

In contrast, the accuracy of the power estimates improves by lowering the temporal resolution of the power monitor (see Figure 6d). Lower temporal resolutions act as smoothing factors for the power spikes that are the primary cause of the accuracy loss. In particular, the power consumption trace appears more regular at lower temporal resolutions, thus allowing the power monitor to better track it.

5. Conclusions

This paper presented a methodology to automatically instrument a resource-constrained power monitor into generic hardware designs. Results have been validated considering both HLS-generated hardware accelerators as well as a RISC-V based SoC across a wide set of temporal resolutions ranging from 20us to 500us.

Depending on the imposed user-defined constraints and with respect to the unconstrained power monitoring state-of-the-art solutions, our methodology shows a resource saving between 37.3% and 81% while the maximum average accuracy loss stays within 5%, i.e., using the aggressive 20us temporal resolution. However, by varying the temporal resolution closer to the value proposed in the state of the art, i.e. in the range of hundreds of microseconds, the average accuracy loss of our power monitors is lower than 1% with almost the same overheads.

In summary, our methodology allows to optimize the time-to-market by allowing the automatic integration of a power monitor into a target design, while allowing to flexibly trade between the accuracy of the estimate and the overheads.

6. Acknowledgments

This work was partially supported by the H2020 FET-HPC project “RECIPE” [1]. G. A. no. 801137.

References

[1] Agosta, G., Fornaciari, W., Atienza, D., Canal, R., Cilardo, A., Flich Cardo, J., Hernandez Luz, C., Kulczewski, M., Massari, G., Tornero GavilÃą, R., Zapater, M., 2020. The recipe approach to challenges in deeply heterogeneous high performance systems. Microprocessors and Microsystems 77, 103185. URL: http://www.sciencedirect.com/science/article/pii/S0141933120303525, doi:https://doi.org/10.1016/j.micpro.2020.103185.

[2] Bircher, W.L., John, L.K., 2012. Complete system power estimation using processor performance events. IEEE TC 61, 563–577.

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

[4] Jianlei Yang, Liwei Ma, Kang Zhao, Yici Cai, Tin-Fook Ngai, 2015. Early stage real-time soc power estimation using rtl instrumentation, in: The 20th ASPDAC, pp. 779–784. doi:10.1109/ASPDAC.2015.7059105.

[5] Najem, M., Benoit, P., Ahmad, M.E., Sassatelli, G., Torres, L., 2017. A design-time method for building cost-effective run-time power monitoring. IEEE TCAD 36, 1153–1166.

[6] Pagliari, D.J., Peluso, V., Chen, Y., Calimera, A., Macii, E., Poncino, M., 2018. All-digital embedded meters for on-line power estimation, in: DATE 2018, pp. 743–748.

[7] Pilato, C., Ferrandi, F., 2013. Bambu: A modular framework for the high level synthesis of memory-intensive applications, in: 23rd Int. Conf. on Field programmable Logic and Applications, pp. 1–4.

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

[9] Scotti, G., Zoni, D., 2019. A fresh view on the microarchitectural design of fpga-based risc cpus in the iot era. Journal of Low Power Electronics and Applications 9, 19. doi:10.3390/jlpea9010009.

[10] Walker, M.J., Diestelhorst, S., Hansson, A., Das, A.K., Yang, S., Al-Hashimi, B.M., Merrett, G.V., 2017. Accurate and stable run-time power modeling for mobile and embedded CPUs. IEEE TCAD 36, 106–119.

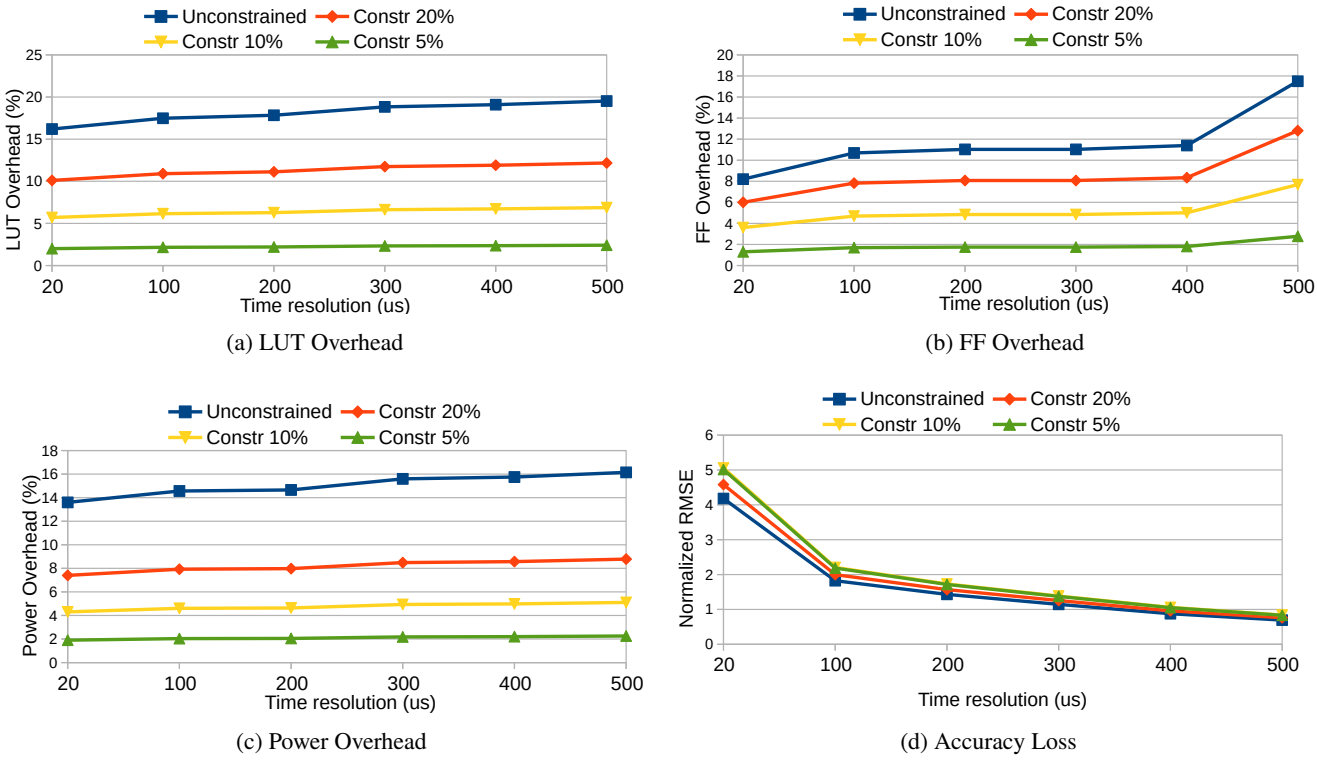
[11] Zoni, D., Cremona, L., Cilardo, A., Gagliardi, M., Fornaciari, W., 2018. Powertap: All-digital power meter modeling for run-time power monitoring. Microprocessors and Microsystems 63, 12. doi:https://doi.org/10.1016/j.micpro.2018.07.007.

[12] Zoni, D., Cremona, L., Fornaciari, W., 2020a. All-digital control-theoretic scheme to optimize energy budget and allocation in multi-cores. IEEE Transactions on Computers 69, 706–721. doi:10.1109/TC.2019.2963859.

[13] Zoni, D., Cremona, L., Fornaciari, W., 2020b. All-digital energy-constrained controller for general-purpose accelerators and cpus. IEEE Embedded Systems Letters 12, 17–20. doi:10.1109/LES.2019.2914136.

[14] Zoni, D., Galimberti, A., Fornaciari, W., 2020c. Efficient and scalable fpga-oriented design of qc-ldpc bit-flipping decoders for post-quantum cryptography. IEEE Access 8, 163419–163433. doi:10.1109/ACCESS.2020.3020262.

[15] Zoni, D., Galimberti, A., Fornaciari, W., 2020d. Flexible and scalable fpga-oriented design of multipliers for large binary polynomials. IEEE Access 8, 75809–75821. doi:10.1109/ACCESS.2020.2989423.



(a) LUT Overhead

(b) FF Overhead

(c) Power Overhead

(d) Accuracy Loss

Figure 6: Design Space Exploration by varying the time resolution of the implemented power monitor. The picture reports the trend of the average values.