A Design-Time Method for Building Cost-Effective Run-Time Power Monitoring

Mohamad Najem, Pascal Benoit, Mohamad El Ahmad, Gilles Sassatelli, Lionel Torres

A Design-Time Method for Building Cost-Effective Run-Time Power Monitoring

Mohamad Najem, *Student Member, IEEE*, Pascal Benoit, *Member, IEEE*, Mohamad El Ahmad, Gilles Sassatelli, *Member, IEEE*, and Lionel Torres, *Member, IEEE*

***Abstract*—The emergence of power as a first-class design constraint has fueled the proposal of a growing number of optimization techniques, seeking the best tradeoff to reach the maximum energy efficiency. Effective adaptation strategies depend critically on the monitoring method as an incorrect assessment of the system's state will result in poor decision making. Yet it is indeed a fundamental issue: how to get a precise estimation of the system's state, and especially in a cost-effective way? We address this question for the self-observation of the power consumption. We develop a method that combines several data mining algorithms to monitor the toggling activity on a few relevant signals selected at the register transfer-level. Our approach is based on a generic flow that is able to produce a power model for any register transfer level (RTL) circuit on any technology. This contribution is evaluated on a system on chip RTL model implemented on an field-programmable gate array technology. The experiments demonstrate that the proposed method achieves the accuracy of analog power sensors (error lower than 1%) at a finer granularity and in a cost-effective way.**

***Index Terms*—Data mining, design-time method, field-programmable gate array (FPGA), power modeling, register transfer-level, system on chip (SoC) monitoring.**

I. Introduction

THE CONTINUING trend in applications for ever increasing functionality, performance and integration within systems on chip (SoCs) is leading to circuits with a high power dissipation. Thus, to be able to continue to provide new and improved features, both design and run-time optimization are indispensable in today's systems. Due to time-to-market constraints and uncertainties at design-time, many techniques seek to make run-time tradeoffs with the goal of adapting to an application's or execution environment's need. The adaptation in embedded systems allows the tuning of hardware/software parameters to perform tradeoffs between power consumption and performance during execution. Adaptation techniques can make use of operational parameters of the system at run-time and make decisions that alter the future operation of the system.

Manuscript received December 14, 2015; revised April 28, 2016 and July 22, 2016; accepted September 8, 2016. Date of publication September 28, 2016; date of current version June 16, 2017. This paper was recommended by Associate Editor S. Kim.

The authors are with the Montpellier Laboratory of Informatics, Robotics and Microelectronics, Department of Microelectronics, University of Montpellier, 34090 Montpellier, France (e-mail: mohamad.najem@lirmm.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2016.2614347

However, effective adaptations depend critically on the monitoring method, which should provide accurate estimations about the system state in a cost-effective way. Several techniques for power management have been proposed that require knowledge of how much energy has actually been consumed by the device to make run-time decisions. Among these, the reactive techniques that aim to adapt the system behavior [1], and the proactive approaches that use the information of power to predict future undesired states [2].

The monitoring of the power consumption can be either done by a direct measurement using analog sensors, or by an indirect estimation using information about resource utilization and the system parameters such as chip temperature, process variation, etc. The advantage of the first approach is the high temporal resolution and the high accuracy of the measured power values. But, the scalability is poor for complex systems, as only few sensors can be employed due to their high costs. However, indirect solutions are usually cheaper with a reasonable accuracy.

Monitoring the power consumed every time is crucial for building a fully self-adaptive system and depending on the granularity of the monitoring, the system will be able to react more efficiently. Consider a motivating example: In case of measuring the total power using analog sensors or an additional equipment, thermal hotspots may not be identified. But however, it is possible to predict them more accurately by having a more detailed power information (in time and space). Hence, a proactive adaptation can be performed to avoid a critical temperature before it actually occurs. As a result, reliability can be significantly improved [2].

The internal activity depends on the switching activity of the transistor and consequently affects the power consumption, and can be appraised at different levels of abstraction. It is therefore a question of: how to accurately observe the internal activity in complex SoCs, and especially in a cost-effective way for an efficient monitoring of the power consumption?

This paper addresses this challenge, proposing a systematic and a generic method for the monitoring of the power consumption. The activity is appraised using the toggling information at the hardware level. At this stage, the tracking of fine-grain power variations is more accurate, compared to the information abstracted at higher levels, e.g., using performance counters. In fact, the most precise switching information that directly affects the power consumption would be obtained by counting activities for each transistor. However, this is not realistic as it would be too much expensive and would consume more power to monitor the original circuit itself. A good tradeoff to reduce the cost of the monitoring would be to estimate the activity by observing only the toggling events on

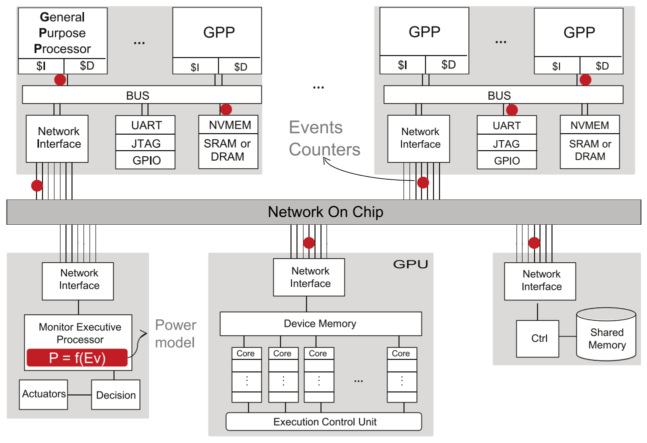


Fig. 1. Envisioned monitoring of the dynamic power for a generic SoC.

few strategic signals from the interfaces between components of the design. In Fig. 1, the envisioned monitoring method is exemplified on a generic SoC. Event counters (ECs) are inserted at design-time to report the activity on some chosen signals by counting switching events; a processor dedicated for the monitoring retrieves the values, and evaluates the dynamic power consumption of the circuit. However, the increase of the SoCs complexity goes hand-in-hand with the growth of the number of wires in a single chip. Therefore a subset of signals has to be selected from a large number of wires, in order to estimate the global activity. To find these correlations, this implies the development of efficient algorithms, which manage large amounts of data. For this purpose, we investigate data mining techniques that are utilized in this paper to extract a subset of strategic signals and to build a cost-effective model in order to estimate online the dynamic power. Hence, our method is as accurate as a power sensor at high temporal resolution, and in addition it has a low area, performance and energy overheads.

II. Background and Related Work

Adaptive integrated systems require a control loop based on a three-step process: 1) monitoring; 2) diagnosis, which analyzes the observed data, e.g., dynamic workloads, to adapt performance and quality-of-service, to optimize resource utilization, or to achieve high energy efficiency; and 3) action, which tunes system parameters accordingly. In this process, the first step is crucial as an incorrect assessment of the system's state will result in poor decision making. This subject is extensively discussed in [3], in which the authors present a survey of existing techniques at all levels of abstraction, aiming to collect system's information, e.g., debugging, performance, quality-of-service, power/energy, temperature, and other metrics. The authors highlight the need of an efficient and generic strategy, which could be used in all SoCs including heterogeneous multicores systems.

The straightforward approach to monitor the power consumption is to online measure it, using analog sensors, such as in [4] that presents a dedicated circuit in a 45 nm SOI silicon technology. The proposed on-chip sensor measures the voltage drop due to a current load and converts it into pulse counts to get a power estimation, and requires 0.02 mm^2 area. Moreover, Bhagavatula and Jung [5] presented an optimized analog power sensor that shows an improvement in time response, but also occupies 0.01 mm^2 area in a 130-nm technology. Although this seems to be a very promising solution, it is not a generic one and the scalability is quite limited. This is the reason why there is a growing interest for indirect approaches.

Therefore, the total power consumption can be expressed as

$$P_{\text{total}} = P_{\text{dyn}} + P_{\text{stat}} = \alpha C V_{\text{dd}}^2 f + V_{\text{dd}} I_{\text{leakage}} \qquad (1)$$

where P_{dyn} is the dynamic power and depends on: the activity factor α, the switching capacitance C, the supply voltage V_{dd}, and the frequency f. The leakage power P_{stat} is estimated as $V_{\text{dd}} I_{\text{leakage}}$, where I_{leakage} is the leakage current. C and I_{leakage} are platform-specific and depend on several parameters such as chip temperature, threshold voltage, etc. The couple (f, V_{dd}) is either constant or tunable such as in case of a dynamic voltage and frequency scaling. However, the activity factor (α) is proportional to the average rate of transistor commutations during the estimation period of the power consumption. And as the monitoring of the global system activity is very expensive at transistor level, research efforts try to estimate it at a higher level of abstraction.

Pathania *et al.* [6] used the CPU utilization at software level as an approximate representation of the application activity factor for a given power mode (f, V_{dd}). Also in [7], a characterization method is proposed in order to approximate α for each application executed by the *ARM big.LITTLE* platform. But these approaches are not suitable for complex systems running multitask programs in parallel, where the activity always varies. On the contrary, an intensive activity in a short period of time may have a significant impact on the system behavior, e.g., by producing a thermal hotspot.

Moreover, there also exist research efforts that use the performance events from the preintegrated performance counters to appraise the global activity such as in [8]. Choi *et al.* [9] estimated the power for a dynamic thermal and power management on the *ARM11* MPCore platform using a simple linear combination of five performance events: 1) the number of instructions executed; 2) the number of L1 data cache access; 3) the number of L2 cache access; 4) the number of stall cycles due to data dependency; and 5) the number of coherence transactions. Moreover, Bellosa *et al.* [10] estimated the power as a linear combination of processor-internal event energy, such as the energy consumed while the execution of a branch misprediction or memory retired, etc. Another approach was proposed in [11] and [12], in which authors compute a weighting factor of maximum power for each component in the system. These factors were approximated using the access rate for each component based on the performance events, for example, the bus control access rates were obtained by counting all bus transactions (all reads, writes and prefetches). Furthermore in [13], analytic power models were provided for each component of the *Pentium IV* system after a selection of some relevant performance events. Another method is addressed in [14], where power models are represented by an analytical function or by a table of consumption values, which depend on a set of parameters such as cache miss rate and pipeline stall rate. These models are extended in [15] to address the *OMAP* (*ARM* + DSP) platform. These approaches may have a good accuracy in some cases, but are quite design specific. Although there is a high number of available events for few configurable hardware counters. This leads to a limitation in the number of simultaneous events that can be counted in a single execution. Additionally, these counters are only included on high performance processors, so they may not be suitable for other embedded systems. The methodology described in this paper is applicable to any type or size of embedded system, and attempts to minimize the additional hardware required for the monitoring of the power consumption to efficiently allow the decision making for power optimizations.

In previous methods, run-time estimations of the power consumption are based on the information available at the system and software levels, which limits the preciseness or requires additional characterizations. However, there exist also several works that use the toggling information at hardware level to estimate the activity factor α. A technique based on hidden Markov models is proposed in [16] to track at run-time the system power modes. Power variations are represented as a fix number of power modes depending on the activity of strategic nets chosen at the register transfer (RT)-level. This approach is interesting, but assumes a set of power modes that is suitable for components such as memories, but too limited for complex components such as processors. Moreover, Peddersen and Parameswaran [17] proposed a method that identifies the control signals for each component of the processor, on which events counters are then added to monitor the activity that contribute to power variations. This method seems to be similar in the concept to the one proposed in this paper, but it is not a systematic one and authors do not address complex systems. The selection of events was based on the study of the correlation between signals and power, and also a linear regression was used to estimate the power.

The selection of relevant events for power modeling was only addressed in [18], in which authors present a statistical analysis to achieve their purposes. They first computed the correlation coefficient between each performance event and the power consumption for the *Dell PowerEdge R805 SMP* platform for each application separately. Then, they ranked events for the selection, based on the median of the computed correlation for all applications. But in complex systems, the processors execute many applications in parallel and their proposed statistical analysis is no longer applicable.

From the variety of approaches in the literature, it is clear that current solutions are either imprecise or too design and technology specific. Our estimation method relies on the information at the hardware level, which compared to existing hardware-level methods [16], [17], is a systematic and generic approach addressing complex systems and can be applied to any circuit and any technology. In our method, the global activity is appraised from the toggling activity on few strategic signals at the RT-level. Compared to software approximations [6], [7] and performance counters estimations [9], [10], our approach does not require any preintegrated counters or characterizations and is more accurate at high temporal resolution. Furthermore, the originality comes from the developed selection method inspired from data mining, and also from the proposed models that accurately track fine-grain variations in the power consumption.

III. General Methodology and Database Generation

The general idea behind our estimations is to use the information of toggling activity from the hardware level to accurately appraise the global system activity that affects the power consumption. Our aim is to be as generic as possible

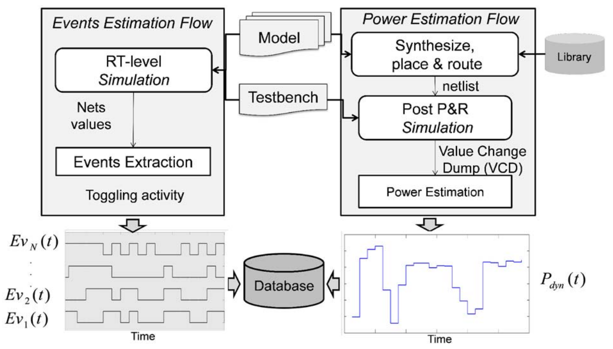


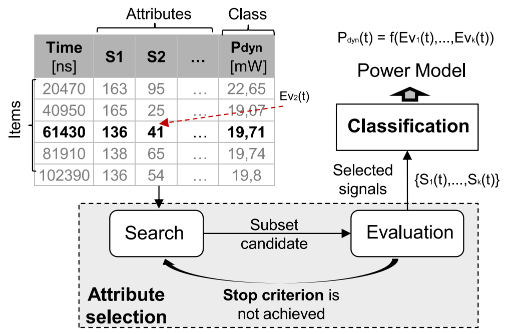
Fig. 2. Generic flow for the generation of the databases.

and, at the same time, to have precise estimations in a cost-effective way. One way to achieve this is to search for correlations with the power consumption among the wires from the communication interfaces connecting the components at RT-level of the design. Our method tries to identify at this stage the strategic signals, on which ECs are attached at design time to observe at run-time the global system activity. Indeed, the estimation of the power consumption by considering the switching information from a lower abstraction layer may be more accurate, but also the number of required counters will be much higher. In all cases, our method can be applied at any hardware level and at any stage from the design hierarchy. However, it is always a question of the tradeoff between the accuracy and the cost of the monitoring.

Furthermore, a significant increase in the number of wires is expected due to the continuing trend for the increase in designs complexity. In order to tackle this challenge, we developed a method based on data mining techniques. Data mining encompasses several competences from different domains, including statistical analysis, databases, machine learning and artificial intelligence. In this paper, we investigate the features selection methods from data mining to identify relevant and strategic signals among all signals from the interfaces between components. The regression techniques and nonlinear models are also compared to model the power in terms of the toggling activity of the selected signals.

In Fig. 2, a complete flow is proposed that is able to generate the database for the data mining analysis. It is based on recording: 1) the toggling activity on RT-level simulations and 2) the power consumption on a placed and routed netlist. The dynamic power consumption is considered for the data analysis, as far as it is directly affected by the internal activity. The power estimation flow (PEF) is developed to collect the power trace. A post place-and-route simulation is performed to generate a value change dump (VCD) file containing net transitions over time. The generated file is then used by an estimation tool (e.g., PrimeTime-PX from SYNOPSYS) to estimate the instantaneous power values. The obtained power trace is a discrete set of average values over an adjustable time interval T. Since T is the sampling period of the power, it must be noticed that it is likely to have some impact on model's preciseness. This is one way to gather the dynamic power. In parallel, the events extraction flow (EEF) performs an register transfer level (RTL) simulation to track the toggling activity over time, which can be obtained by counting the occurrences of rising and falling edges on each single bit signal during the same adjustable time interval T.

This offline flow leads to a database composed of items recorded at specified time intervals, which are then analyzed



Time [ns]	S1	S2	…	Pdyn [mW]
20470	163	95	…	22,65
40950	165	25	…	19,07
61430	**136**	**41**	…	**19,71**
81910	138	65	…	19,74
102390	136	54	…	19,8

Fig. 3. Proposed data mining method for a lightweight monitoring of the dynamic power.

with a Data Mining tool. An example of the generated database is given in Fig. 3: one line, called item or *instance*, is added for each time interval T. It encompasses the counted events over the last period for each individual signal (one column refers to one RTL signal), also called *attribute*. The corresponding power or *class* for a given item is reported in the last column. The general method for the data mining analysis is also depicted in Fig. 3, a two-step process in which *Attribute Selection* (also known as feature selection) preprocessing is first performed to select a subset of signals, and then *Classification* techniques are applied to build the model. Algorithms and metrics involved are detailed in the following sections.

IV. Attributes Selection

At this stage, the application of the flow proposed in the previous section for a complex circuit, leads on constructing a big database with many signals and items. Let N_{att} be the total number of attributes and equal to the total number of available signals at the RT-level. Let N_{item} be the total number of items, that corresponds to the counted events and the equivalent power consumption. The challenge is to find a minimum number of attributes (k, $k \ll N_{\text{att}}$) that is enough for power modeling. The classical statistical correlations and hypothesis tests operate on each attribute separately and compute the relationship between each attribute and the power. However, perfectly correlated attributes are truly redundant in the sense that no additional information is gained by keeping all, and also poorly correlated attributes with power can sometimes be useful together for power modeling.

The *Attribute selection* methods from data mining encompasses the power of statistics and computer science, and offers several search algorithms that tries to remove as much irrelevant and redundant information as possible. For this purpose, we explore different type of selection method in order to select a subset of attributes for power modeling. In general, the attribute selection performs a search through the space of attribute subsets, and, as a consequence, must address four basic issues affecting the nature of the search (as shown in Fig. 3) [19].

A. Starting Point

The search can start from a point in the attribute subset space from which to begin the search can affect the direction of the search. One option is to begin with no attribute and successively add attributes (forward search). Conversely, the search can begin with all attributes and successively remove them (backward search). Another alternative is to begin somewhere in the middle and move outwards from this point.

B. Search Procedure

In artificial intelligence, a search procedure is an algorithm for finding an element with specified properties among a collection of elements. They are divided into three main categories: 1) *exhaustive*; 2) *heuristic*; and 3) *meta-heuristic*. With N_{att} initial attributes, there exists $2^{N_{\text{att}}}$ possible subsets. The exhaustive search consists of systematically enumerating all possible candidates and checking whether each candidate satisfies the criterion for the selection.

However, heuristic approaches are techniques designed to be faster than exhaustive. Their objective is to produce a solution in polynomial time that is enough for solving the problem. The basic search strategy, called greedy search (also known as greedy hill-climbing) [20], considers local changes to the current attribute subset. Often, a local change is simply the addition or deletion of a single attribute from the subset depending on the search direction (forward or backward). At each iteration, it selects the attributes that improves more the current solution. Therefore, the evaluation function produces a metric to compare subsets, which is defined later in the evaluation procedure.

Another approach, called BestFirst (extended to LinearForwardSelection) [21], allows the backtracking along the search path. Like greedy hill climbing, best first moves through the search space by making local changes to the current attribute subset. However, unlike hill climbing, if the path being explored begins to look less promising, the best first search can backtrack to a more promising previous subset and continue the search from there. Given enough time, a best first search will explore the entire search space, so it is common to use a stopping criterion. Normally this involves limiting the number of backtracking that result in no improvement.

On the other hand, meta-heuristic methods are based on stochastic and iterative optimizations. Genetic algorithms are adaptive meta-heuristic search based on the principles of natural selection in biology. They employ a population of competing solutions, evolved over time, to converge to an optimal solution. Effectively, the solution space is searched in parallel, which helps in avoiding local optima. For attribute selection, a solution is typically a fixed length binary string representing an attribute subset, where the value of each position in the string represents the presence or absence of a particular attribute. It is an iterative process where each successive generation is produced by applying genetic operators such as *crossover* and *mutation* to the members of the current generation. Mutation changes some of the values (thus adding or deleting attributes) in a subset randomly. Crossover combines different attributes from a pair of subsets into a new subset. The application of genetic operators to population members is determined by their fitness (how good an attribute subset is with respect to an evaluation strategy). Better attribute subsets have a greater chance of being selected to form a new subset through crossover or mutation. In this manner, good subsets are "evolved" over time.

C. Evaluation Procedure

The evaluation procedure defines a metric (called merit) to evaluate the current candidate subset selected by the search procedure. There are two main categories: 1) *wrapper* and 2) *filter* [19]. *Wrappers* are algorithms that use feedback from a classification algorithm in order to determine which attribute(s) are needed in order to build the most accurate model. The merit can then be one of the preciseness metrics such as the root mean squared error (RMSE), mean absolute error (MAE), etc. The correlation subset evaluation (CSE) is one of the implementations of the wrapper evaluator. It measures the RMSE for a given model built with the entire attributes, and then compares it to the error for each candidate subset. In the end, the final subset is the one that produces the most accurate model (lowest RMSE).

On the contrary, the *Filter* approach only uses statistical dependencies, e.g., correlation, to evaluate an attribute without any involvement of a classification model. In the literature, there are many filter evaluators, but most of them address the nominal and binary databases such as information gain, gain ratio, relief-F, one-R, chi-squared, etc. The correlation feature selection (CFS) is widely used for numerical database problems [22]. It quantifies the fitness of a candidate subset using the merit measurement shown in (2), where A is the number of attributes in the candidate subset, $\bar{r}_{\text{ac}}$ is the average correlation between each attribute and the class, and $\bar{r}_{\text{aa}}$ is the average correlation among attributes in the candidate subset. The subset having the highest merit is the outcome of the CFS and corresponds to a set of attributes, which highly correlate with the power and uncorrelated with each other

$$M_s = \frac{A\bar{r}_{\text{ac}}}{\sqrt{A + A(A-1)\bar{r}_{\text{aa}}}}. \tag{2}$$

D. Stopping Criterion

Depending on the search procedure, an attribute selector might stop adding or removing attributes when none of the resulting attributes improves the merit of the current solution, e.g., BestFirst and LinearForwardSelection. Alternatively, the algorithm might continue to revise the attribute subset as long as the merit does not degrade, e.g., GreedyStepwise, or the size of the output subset reached a predefined parameter.

V. Classification

In the previous section, we have presented several attribute selection methods, which allow the selection of a subset with k signals among the N_{att} available. The best selection method is the one that quickly produces a solution with the fewest number of attributes (k), which are enough to accurately appraise the global system activity. The objective is then to build a model that is able to estimate the power consumption in terms of the toggling activity counted on the k selected signals. This is the role of the three regression processes described in the following paragraphs.

A. Linear Regression

Regarding the complexity and the response time, the linear regression is the simplest model to be used in order to produce a resourceful estimation of the consumption. Equation (3) relates the power with the counted events on the k selected signals. It is composed of a constant term P_0, which corresponds to the idle power consumption, and the sum of terms $w_i Ev_i$, in which w_i are the coefficients of Ev_i that weigh the contribution of each attribute on power variations

$$P_{\text{dyn}} = P_0 + \sum_{i=1}^{i=k} w_i Ev_i. \tag{3}$$

B. Multivariate Adaptive Regression Splines

In complex systems, the toggling activity may have a nonlinear behavior that would have a negative impact on the linear model accuracy. Therefore, a multilinear based model is also considered in this paper. Multivariate adaptive regression splines (MARS) [23] is a nonparametric spline-based method. It is based on a combination of linear truncated basis functions to approximate the model. The MARS based power model can be written as an additive function of the product basis functions as shown in (4) where β_0 is the coefficient of the constant basis function $B_0 = 1$, B_m is the mth basis function, β_m is the coefficient of the basis function, and M is the number of basis functions in the model. Each basis function is a Hinge function shown in (5) and depends on the events for one selected signal where c is a constant compared to the counted events

$$P_{\text{dyn}} = \beta_0 + \sum_{j=1}^{j=M} \beta_j B_j \tag{4}$$

$$B_j = \max(0, c - Ev). \tag{5}$$

C. Neural Network

Artificial neural networks are used in data mining for regression analysis, especially for supervised learning to model nonlinear hypothesis. For this purpose, we aim to evaluate the multilayer feed forward neural network. It consists of neurons, that are arranged in layers. The first layer (input) has k neurons (total of selected signals) and the output layer contains one neuron that produces the estimation of the power. The layers between have a configurable number of neurons, and each neuron has direct connections to all neurons of the subsequent layer. The connection between the ith and jth neuron is characterized by the weight coefficient w_{ij} and the ith neuron by the threshold coefficient r_i. The weight coefficient reflects the degree of importance of the given connection in the neural network. The output value x_i of a neuron is determined by (6), where φ_i is the input of the transfer function, which is also determined in (7), where N_C is the number of connections to the ith neuron. The back-propagation algorithm used to train the neural network, varies the threshold coefficients r_i and weight coefficients w_{ij} to minimize the sum of the squared differences between the computed and required output values

$$x_i = \frac{1}{1 + e^{-\varphi_i}} \tag{6}$$

$$\varphi_i = r_i + \sum_{j=1}^{j=N_C} w_{ij} x_j. \tag{7}$$

VI. Experimental Setup

The generic method presented in this paper is applied for a demonstration on an field-programmable gate array (FPGA) technology. Power efficiency has become a major concern for all electronic devices, including reconfigurable circuits, which

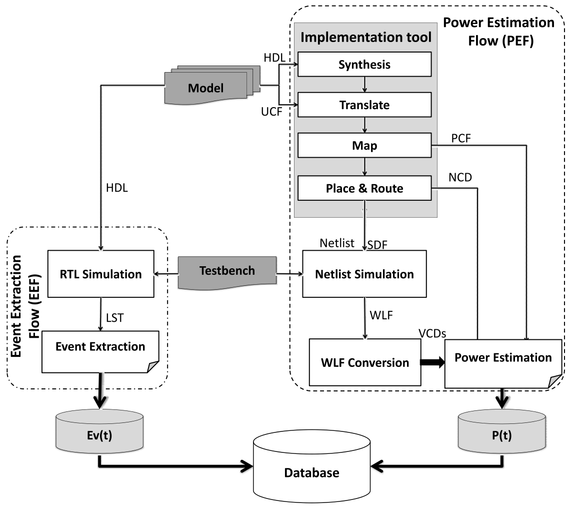


Fig. 4. Proposed flow that generates the database for the FPGA implementations.

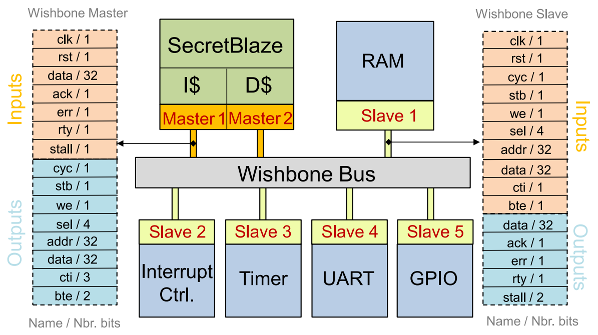


Fig. 5. SecretBlaze SoC.

TABLE I
APPLICATIONS EXECUTED BY THE PROCESSOR

Applications	Execution Time (clock cycle)	Description
Nschineu	2762500	Simulator of petri net
MJPEG	87500	Video compression
MXM	5682500	matrix to matrix multiplication
compress	25000	SPEC95 data compression
statemat	97500	Simulation of a car window state machine
qurt	37500	Square root operations
whetstone	4250000	Performance test program

has led to a growing body of research on power management. Estimation of the power consumption for FPGAs is a topic which is discussed in [24], where precharacterization-based macro-modeling is used to capture average switching power per access to both look-up tables (LUTs) and registers. Another linear-based model for the dynamic power is proposed in [25], in which the switching activity from the RT-level is utilized to estimate component power breakdowns for the FIR filter application with a fair error (10% of average error). Power models for multiprocessor SoC (MPSoC) circuit on FPGA is conducted in [26]. The technique is based on the offline extraction of the abstract profile of an application called event signatures used to estimate the average of the total power consumption. The authors achieve 10% of average error estimating the average power. It is worth noting that there is no real contribution addressing the power monitoring issue, and comparing tradeoffs on model accuracy against the overhead.

The dedicated flow is depicted in Fig. 4. PEF is adapted to estimate the $P_{\text{dyn}}(t)$ for designs implemented on FPGAs. For this purpose, the Xpower tool from *Xilinx v13.1* is used: unfortunately, it only estimates the average of the power consumed during the simulation. Since instantaneous estimation is required for monitoring purposes, the simulation is split into periodic time slices to provide N_{item} estimations. The *Xilinx* tool-chain (XST, translation, mapping, etc.) provides the synthesized, then placed and routed netlist from the HDL files. A simulation description file [wave long format (WLF)] is generated along with *ModelSim v10.1d*. The WLF is then cut into N_{item} subsimulations as described in (8) thanks to wlfinfo command from *ModelSim*. T_0 is the time set to initialize the circuit and T is the time interval for each produced WLF file. The last step of the PEF flow is to convert the WLF to VCD files and then run the Xpower tool to deliver N_{item} estimations, which correspond to $P_{\text{dyn}}(t)$. While PEF produces the estimation of $P_{\text{dyn}}(t)$, the role of the EEF is to build a toggling activity report at the RT-level. *ModelSim* simulation is run to simulate the RTL description of the system, from which the activity is extracted

$$T_{\text{total}} = T_0 + N_{\text{item}} \times T. \tag{8}$$

The hardware platform considered in our experiment is the open-source configurable SoC SecretBlaze [27]. This SoC is a micro-controller addressed for high performance computing applications, fully described in VHSIC hardware description language. The architecture is represented in Fig. 5, and includes a 32-bit processor, an interrupt controller, universal asynchronous receiver transmitter, timer, instruction/data cache memories, and a bus as interconnect. All the components of the SoC communicates through master–slave Wishbone interfaces, which are also illustrated in this figure. However, going further in details about the communication protocol is out of the scope of this paper, but interested readers can refer to [28]. However, the flattened top level architecture has 1531 single bit signals, including all master–slave Wishbone connections, which will be considered as attributes in the generated database ($N_{\text{att}} = 1531$).

The Atlys board which is based on a *Xilinx Spartan-6 LX45* FPGA has been chosen for this paper; in particular, it has integrated real-time sensors on its power rails which provide an interesting feedback to verify design-time and run-time estimations. *WEKA* [29] was also chosen to put into practice the search and classification methods on the generated database. This tool offers several advantages: it is open-source and it contains a large number of configurable heuristics, parameters, evaluators and classification methods. It was completed by an open-source toolbox for MATLAB/Octave Areslab [30].

Since this is a programmable system, the internal activity depends on data and instructions. It is therefore necessary to set up a test bench that has a large coverage. To this end, several benchmark applications were compiled and then executed by the processor one after the other in a standalone mode (i.e., without any micro kernel). These programs are listed in Table I with their corresponding duration in clock cycles, and a short description of their content. The obtained power profile is depicted in Fig. 10(b) for the smallest sampling period T

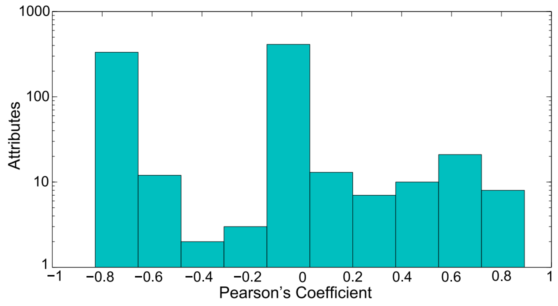


Fig. 6. Histogram of the computed correlations between attributes and power.

TABLE II
EVALUATED ATTRIBUTES SELECTION METHODS

Att. Selection Methods	Search	Evaluation
M1	Greedy Stepwise	CFS
M2	BestFirst	CFS
M3	Genetic	CFS
M4	Greedy Stepwise	CSE + LM
M5	BestFirst	CSE + LM
M6	Genetic	CSE + LM
M7	Greedy Stepwise	CSE + NN
M8	BestFirst	CSE + NN
M9	Genetic	CSE + NN
M10	Greedy Stepwise	CSE + MARS

TABLE III
METRICS MEASURED FOR THE ATTRIBUTES SELECTION METHODS

	Filter Approach			**Wrapper Approach**		
	M1	*M2*	*M3*	*M4*	*M5*	*M6*
Subset size	72	53	262	97	84	273
Eval. Merit	0.985	0.985	0.931	0.934	0.934	0.95
Time [min]	0.56	0.4	11.76	930.15	660.2	5040.86

(equal to 100 μs). The first observation that can be made is the great diversity of behaviors: the power of *nschineu* maintains a pretty constant value while *MJPEG* produces a clear discontinuity. The power of *MXM* fluctuates around an average value with small variations, while the following benchmarks have much higher disparity in power such as *compress*, *statemat* and *qurt*. Significant variations in power consumption of *Whetstone* are due to the execution of different types of functions such as array initialization, conditional jumps, procedure calls, arithmetic functions, and others. All applications are executed by the processor at the maximum operating frequency 25 MHz. The sampling period of power T is one parameter that can be adjusted. It will be tailored to analyze the impact on accuracy and overhead of the generated models.

VII. EXPERIMENTAL RESULTS

The objective of the experiments is to evaluate and compare different methods from data mining for power monitoring purposes. The challenge is to find the best combination of algorithms and parameters to achieve the highest accuracy in a cost-effective way. We first provide an evaluation of the different search and classification methods. Then, we study the overhead of the proposed monitoring method.

A. Attributes Selection

At this stage, the generated database has initially 1531 attributes (N_{att}) and 6500 items (N_{item}) for the smallest sampling period T (equal to 100 μs), which corresponds to 37.19 MB of data memory size. The standard method that statisticians use to select a subset of attributes is to measure the correlation between each attribute and the power. The significance of any correlation is computed using p-value, which should be lower than 0.05 for having strong correlation. The application of this method results by selecting 821 attributes that have a significant correlations with the power with a p-value lower than 0.05. The histogram of the correlation coefficient of the selected signals is shown in Fig. 6, where various significant correlations can be concluded. This shows the weakness of the classical statistic method for resolving the problem of the selection.

The first step consists of evaluating the attribute selection methods presented in Section IV. For this purpose, the evaluated methods are shown in Table II, in which we considered both Greedy and BestFirst from heuristic search and the genetic search from meta-heuristic. In the literature, the population size parameter for the genetic algorithm is recommended to be between 50 and 200 for a better selection in the genetic search [31]. Whereas, the number of iterations (or generations) should be large enough to ensure the convergence toward a solution. For this reason, we set the population size and the maximum generation to 100 and 200, respectively. Moreover, the number of neurons at the hidden layer in NN model has a significant impact on model precision and training time. According to our experiments, three neurons in the hidden layer in the NN model are enough to produce an accurate and resourceful model. However, the exhaustive search was eliminated from this evaluation as it is very time consuming and does not complete the search. Moreover, the search algorithms are completed by both evaluations: the wrapper using CSE with both LM and NN models and the Filter using CFS. Furthermore, the selection method in Areslab toolbox that implements the MARS model, uses the greedy search with the wrapper approach for the selection of attributes.

The application of the possible selection methods shown in Table II to the initial database, does not all produce a solution in a reasonable time. For this reason, the only compared methods are those completed within three days, while the eliminated methods are $M7$–$M10$. This is due to the high training time of both NN and MARS models that highly increases the time needed to complete the search by the wrapper approach, which train several times the model at each iteration. The time needed for the selection, the evaluation merit and the cardinal of the selected subset are compared in Table III. It must be noticed that the merit in the filter methods is based on (2), while in the wrapper methods is equal to the RMSE of the trained model. For this reason, the comparison between wrappers and filters *should not be based on this metric*.

First, the filter based methods ($M1$–$M3$) are much faster than wrappers ($M4$–$M6$) according to Table III. They almost eliminate redundant and useless attributes. Moreover, the heuristic methods with the filter evaluation is much more useful than the classical statistic method. For example, $M1$ and $M2$ have selected only 72 and 53 attributes, respectively, compared to 821 selected by the p-value selection method. Comparing $M3$ to $M1$ and $M6$ to $M4$ in Table III, the genetic

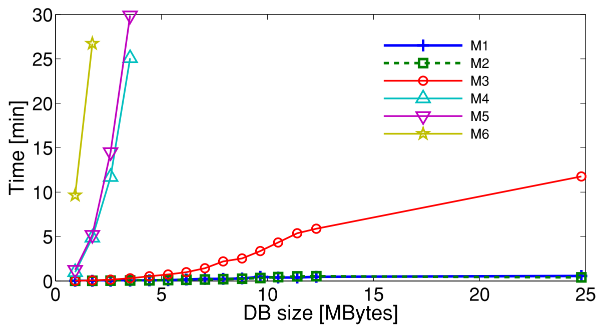


Fig. 7. Time taken by the selection methods for different sizes of databases.

search is much slower than heuristic for both filter and wrapper based methods. It also selects a higher number of attributes with a lower merit. Both heuristic search produce a comparable solution (merit: $M1 = M2$ and $M4 = M5$), while the greedy search tends to select a higher number of attributes. This is due to the stop criterion detailed in Section IV. The Greedy continue adding attributes as long as the merit does not degrade, while the BestFirst stops adding attributes when none of the resulting attributes improves the merit.

B. Scalability

In order to evaluate the complexity of the selection methods, we apply all remaining methods ($M1$–$M6$) on different sizes of databases. Each database was created by randomly removing attributes from the initial set. Indeed, the complexity of these methods depends also on several parameters, including the number of instances simulated, the order of the attributes in the database, etc. But, such evaluation allows us to make a primary approximation about the scalability of these techniques to handle complex systems generating big quantity of data.

Fig. 7 plots the time needed to complete the selection by all techniques for all created databases. It is clear that filter methods ($M1$–$M3$) are faster than the wrappers that uses the feedback of the linear model to select relevant attributes ($M4$–$M6$). With a simple linear approximation, we can estimate the time needed to complete the search in case of hundred thousand of signals. $M1$ and $M2$ will take around 3 min, while $M4$ needs two days to complete the search. However, the approximated time taken by the wrappers $M4$ and $M5$ is about 150 days, and 2.5 years for $M6$. We remind that wrapper approaches with both NN and MARS models were not able to produce a solution in our case with thousands of attributes, and were previously eliminated. From this paper, we conclude that selection methods in the literature must be efficiently used to address the challenge of managing high quantity of data.

C. Relevant Signals

From the previous study, we conclude that the wrapper approaches are not suitable in most of selection methods for complex systems, while the filter methods are able to operate on big database sizes within a reasonable time. Second, the heuristic search is able to produce a solution in a reasonable time with fewer number of attributes compared to the genetic search. Hereof, we propose to use a combination of both filter and wrapper methods with an iterative search, to select the subset that produces the most accurate model for the three classifier LM, NN, and MARS. For this paper, we use the Greedy search instead of BestFirst, because it provides a ranking parameter for the selected attributes. The utilization of BestFirst will also lead to a comparable solution.

$M1$ selects 72 from 1531 initial attributes that have almost a significant relation with the power. Then, the application of $M4$, $M7$, or $M10$ to the resulting database produces a solution that leads the best for power model's precision. The selected attributes are shown in Table IV, where $M4$ has selected eight attributes while $M7$ and $M10$ have selected six to build the most accurate model. Most of these signals are related to the data or address buses from the wishbone interfaces shown in Fig. 5. It must be noticed that all data and address buses are connected inside the wishbone and a dedicated controller manages the control signals to select one target to conduct a request. In other words, the activity on the data or address bus connected to the ICache (master 1) is almost the same for the DCache (master 2). Consequently, snooping the toggling activity on few bits of these buses is enough to appraise the global activity, which corresponds to the instruction and data cache misses, the uncachable write access, the transaction to the TIMER, etc. At the end, this information could not be retrieved by a simple observation of previous knowledge of the hardware. With our method, we are able to extract this knowledge from the high quantity of data with a reasonable time.

TABLE IV
SIGNALS SELECTED BY THE THREE WRAPPER METHODS $M4$, $M7$, AND $M10$

Sig. Nbr.	Selection Method	M-S interface	Description	Corr Coef.
1	M4	Master 2	addr(22): bit 22 from the address bus in the output interface.	0.8921
2	M4, M7, M10	Master 1	data(29): bit 29 from the data bus in the input interface.	0.805
3	M4	Master 1	data(15): bit 15 from the data bus in the input interface.	-0.8021
4	M4,M7, M10	Master 1	data(28): bit 28 from the data bus in the output interface.	-0.0289
5	M4, M7 M10	Master 1	addr(23): bit 23 from the address bus in the output interface.	0.7579
6	M4	Slave 1	data(10): bit 10 from the data bus in the input interface.	-0.0364
7	M4	Master 2	data(10): bit 10 from the data bus in the output interface.	0.7055
8	M4	Master 1	data(23): bit 23 from the data bus in the output interface.	-0.0289
9	M7,M10	Master 1	data(10): bit 10 from the data bus in the input interface.	-0.8132
10	M7,M10	Master 2	addr(14): bit 14 from the address bus in the output interface.	0.6411
11	M7	Slave 2	clk: wishbone clock from the input interface.	-0.0283
12	M10	Master 2	addr(26): bit 26 from the address bus in the output interface.	-0.0407

D. Power Tracking and Models Accuracy

In the previous section, we have evaluated different attribute selection methods, and selected few signals that relates the best for power modeling. In this experiment, we aim to evaluate the accuracy of the three models. For this reason, the generated set of data is randomly split over items into two parts: 1) *training* set to build the model and 2) *test* set to measure its accuracy. This methodology has been applied for the subsequent described experiments. The LM model achieves an average error 3.62% on the test set using the events counted on eight signals, while the error for NN and MARS is 4.02%

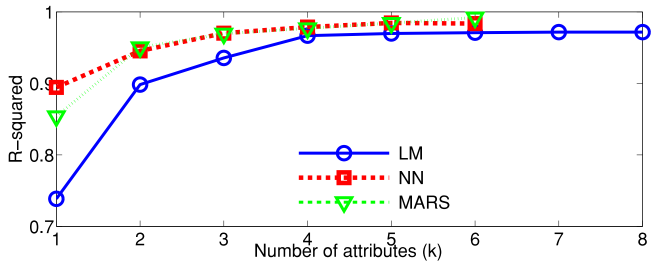


Fig. 8. Coefficient of determination (R-squared) computed for the test databases.

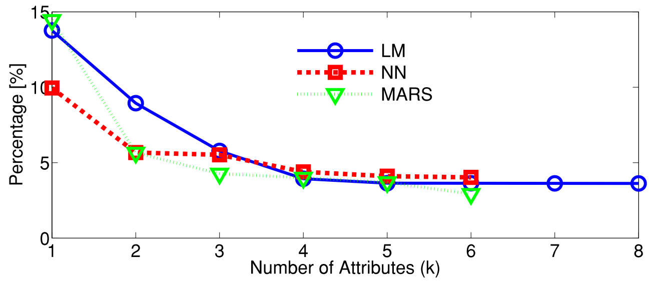


Fig. 9. Average of error computed for the test databases.

and 2.92%, respectively, with only six signals. The selected subset illustrated in Table IV corresponds to the maximum model precision. However, each selected signal requires the integration on an EC to monitor the occurrence of the toggling activities, which is an extra overhead. For this purpose, we aim at evaluating the impact of each selected signal on model precision to find the best tradeoff between the accuracy and the overhead.

Hereof, we investigate the impact of two parameters on model accuracy: the number of selected attributes (k) and the sampling period (T). Let us assume first T is fixed to 100 μs, while varying the number of attributes. At each iteration, we remove the attribute that has the lowest impact on model precision. The coefficients of determination (denoted R^2) are reported in Fig. 8; it measures how well the model predicts the original values (close to 1 is better). The average of the error is also used to ascertain the model precision in Fig. 9; it is the MAE divided by the average of the reference power values. The three models track very well the power variations beyond three attributes, where R^2 becomes higher than 0.9 and the average error is lower than 6%.

Also, the gain in accuracy with k higher than 3 is less than 0.5% for each additional attribute. It is interesting to note that the MARS model outperforms the others, except for $k = 1$. We also observe that NN is better than LM for k smaller than 4, and the error is near for the other cases. We finally conclude that it is possible to reach a relative good accuracy over time with a small number of signals, which means that our methodology is able to provide trustworthy energy estimations. In the next experiments, we consider two cases: 1) k equal to 3, which is the first value of k for which the three models achieve a good accuracy and 2) k equal to 8 for LM and 6 for both NN and MARS, which is the most accurate models achieved.

Fig. 10 shows the reference power profile and the three studied classification models for the highest precision ($k = 8$ for LM and $k = 6$ for NN and MARS). Three areas of interest are highlighted on the power trace extracted for the smallest sampling period T equal to 100 μs. On the first one, when MJPEG begins, a sudden discontinuity occurs: both LM and NN are able to follow the reference but with a non-negligible difference. MARS is the only model that is able to track the best these discontinuities. In the second situation (MXM), it can be seen that despite the small variations, it is interesting to observe that the three models are able to track the consumption. The third case further supports this tendency as linear, MARS and NN are able to follow the reference on rapid and significant variations. Fig. 10 is a good illustration, but is not sufficient enough to appraise the accuracy of the models.

Let us assume now k be the smallest value producing an accurate model (k equal to 3). The sampling period T should be considered according to the whole adaptation process. As T may also have a repercussion on the accuracy and the overhead, it is therefore appropriate to specify the average error for different periods. The corresponding values are plotted in Fig. 11 for the three models. As expected, the error reduces while increasing T. It is, however, important to remark that the decrease is faster for MARS and LM models.

To summarize, all the models are able to achieve a good accuracy with few attributes on various activities, and the MARS model provides the best results compared to the other approaches. There is also a tradeoff to find between the required precision and timings of the adaptation loop. Moreover, as these two parameters have a direct effect on the area and power overhead, we must consider this before drawing the final conclusions.

E. Overhead

The last step of these experiments aims at evaluating the additional costs required for the implementation of the models generated by the classification methods. It must be recalled at this point that the final objective is to monitor the power consumption at run-time: the estimation should be as precise as possible, but the area, performance and energy overheads should be at the same time as small as possible. Power sampling period T and the number of attributes k are the two parameters that can be tuned to study different accuracy/overhead tradeoffs. k is set to 3, which corresponds to the minimal number of attributes required for a sufficient accuracy, and to 8 and 6 for LM and both NN and MARS, respectively, which corresponds to the highest preciseness. As a reminder, each attribute corresponds to one selected signal on which EC should be attached. T is varied from 100 μs to 1 ms.

We designed a configurable monitoring unit based on modules counting the number of rising and falling edges on selected signals. Each counter can be reset, enabled or disabled when required. The monitoring unit is controlled by the main processor, which allows managing and synchronizing the different modules at the software level. Counter's size depends on T, 12- to 15-bit here, which leads to the same area on the *SC6LX45* FPGA: each EC occupies eight slice registers and eight slice LUTs. Compared to the resources required by the SoC SecretBlaze (1574 slice registers and 2466 slice LUTs), each counter implies a 0.51% slice register and 0.32% slice LUTs overheads. Therefore, the area overhead varies between 1.5% for $k = 3$ and 4% for $k = 8$.

Once events have been counted over the period T, values are collected by a monitoring management unit. Its role is to compute the estimation of the power consumed during T.

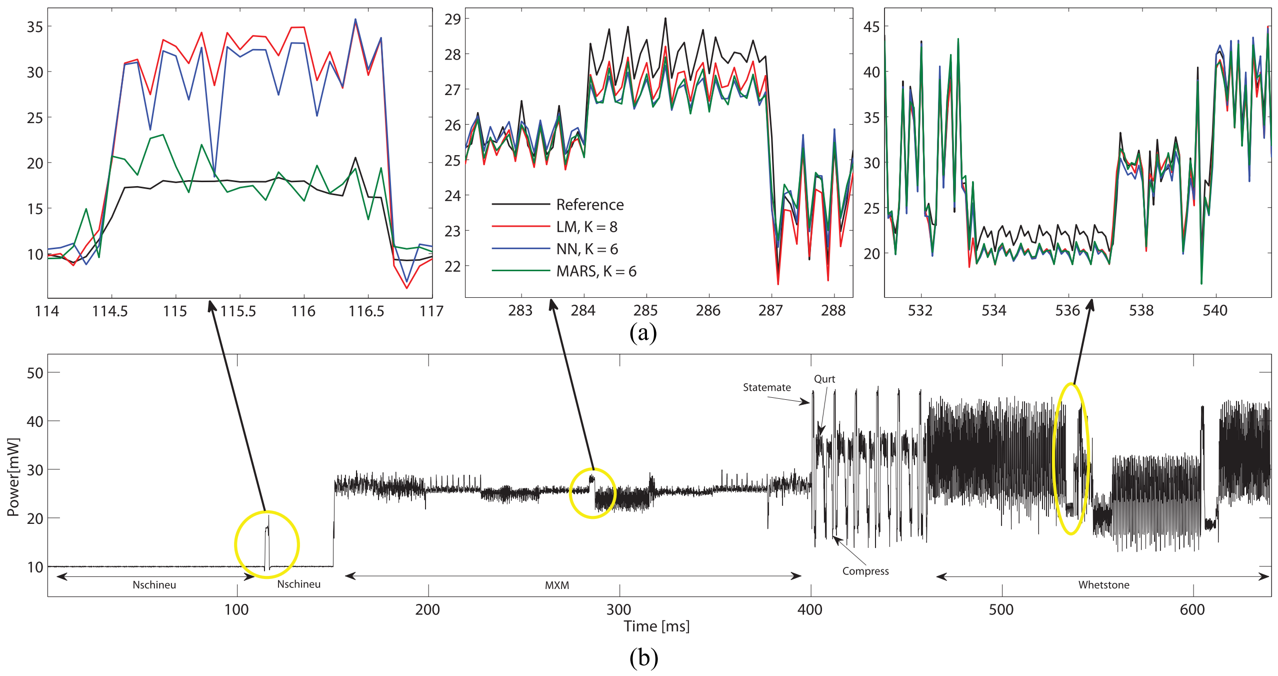


Fig. 10. Profile of power and the fitting of the three models with the maximum precision (eight attributes for LM and six attributes for both NN and MARS) for a sampling period equal to 100 μs. (a) Models fitting. (b) Power profile.

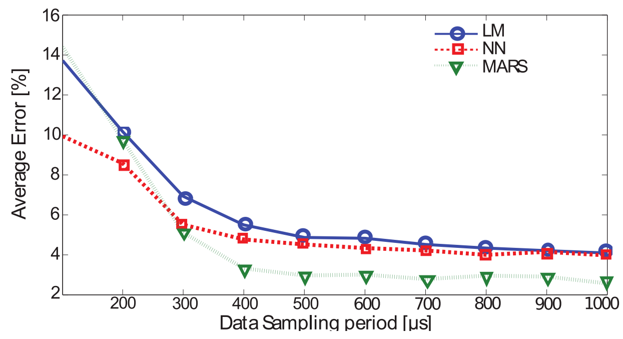


Fig. 11. Average of error of the three model built with three attributes computed for the test database with different sampling periods.

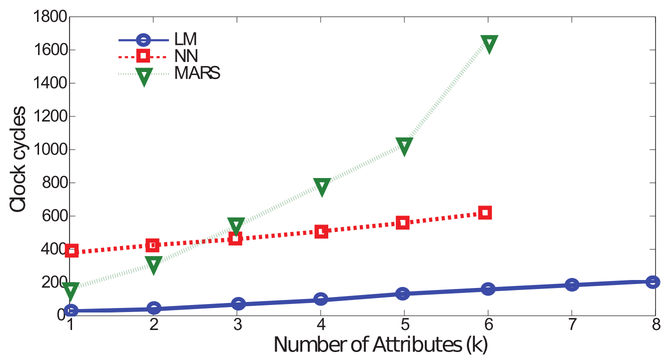


Fig. 12. Computation time of the three models while varying the number of attributes.

This calculation is shown in Fig. 12 and depends on the chosen classification method: LM, NN, or MARS. The three models have different complexity, which impacts the time needed to produce the estimations. For the LM model, it only requires one multiplication and addition for each EC. As mentioned in Section V, the MARS model is the sum of the product of M basis functions (4) and (5): M multiplications and additions are required, as well as a conditional structure for the max calculus.

The time needed for the neural network based estimation depends on the number of neurons in the hidden layer. We approximate the complexity of this model by (9), where N_{HL} is the number of neurons at the hidden layer, k is the number of selected attributes, C_{Mult} is the number of cycles needed to execute 1 multiplication, C_{Add} is the number of cycles needed to execute 1 addition, and C_t is the number of cycles needed to execute the sigmoid transfer function shown in (6). Equation (9) shows the high dependence between the complexity of the neural network and the number of neurons at the hidden layer (N_{HL}), which justifies the need to minimize N_{HL} once a steady and accurate model is achieved (in our case equal to 3). On the other hand, it is clear that the computation of an exponential function is too much expensive: that is the reason why the sigmoid function was replaced by a faster transfer function shown in (10). The faster transfer function was only considered for the computation of the overhead in this experiment and does not affect the model precision. In all cases, the transfer function mainly influences the computing and training times of the neural network, and has a low impact on model precision as discussed in [32]

$$C_{\mathrm{NN}} = N_{\mathrm{HL}} \cdot \big(k \cdot C_{\mathrm{Mult}} + k \cdot C_{\mathrm{Add}} + C_t + C_{\mathrm{Mult}} + C_{\mathrm{Add}}\big) + C_t \quad (9)$$

$$\mathrm{FastSigmoid}(x) = \frac{0.5 \times x}{1 + |x|} + 0.5. \quad (10)$$

We considered in the scope of this paper that the management of the monitoring was handled by the CPU. The computation overheads for each model have been obtained by

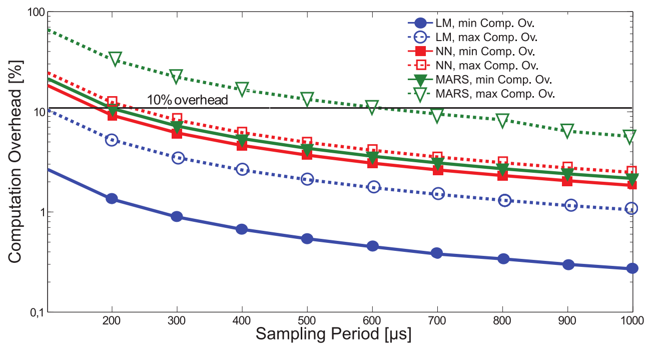


Fig. 13. Minimum and maximum computation overhead of the three models for different sampling periods.

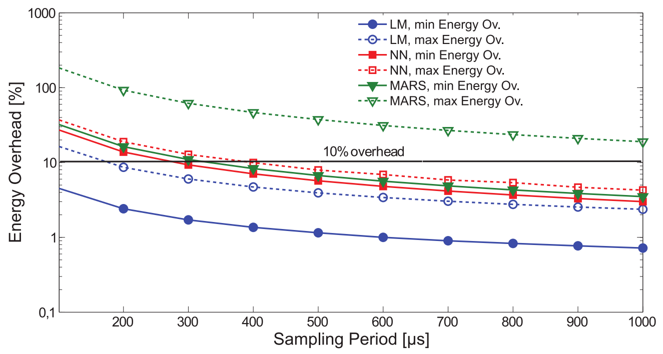


Fig. 14. Worst-case energy overhead corresponding to the energy consumed by ECs and the energy consumed by the processor to compute the power by the three models.

programming and running classification models on the processor. Results are shown in Fig. 13: the overhead is calculated as the ratio between the number of cycles needed to complete the computation of the power by the processor for a given model (shown in Fig. 12), and the sampling period T. The minimum computation overhead corresponds to the cost of the three models for k equal to 3 (lowest sufficient accuracy), while the maximum overhead is the cost of the three models at higher precision ($k = 8$ for LM and $k = 6$ for both NN and MARS). Not surprisingly, the linear model has the lowest overhead compared to the other methods. We also observe that the maximum computation overhead of MARS is not sustainable: this is due to the selection method on MARS, which associates several basis functions for a unique signal that rapidly increases the complexity of the model. The maximum overhead of the LM (dotted blue line) and NN (dotted red line) is lower than 10% for a period greater than 200 and 300 μs, respectively. This overhead reduces significantly by reducing the number of signals (k) in the models and by increasing the sampling period; it can achieve 0.3% for LM (blue line) and 2% for NN models (red line) with three signals at 1 ms. Moreover, the minimum MARS computation overhead is about 3% at 1 ms (green line).

Finally, we analyze the energy overhead E_{ov}, which depends on the power consumed by the ECs over the period T and the energy consumed by the processor during the model computation (which is related to the power consumed by the processor and the time required to produce the estimation). E_{ov} can be easily expressed as a function of the period T and the number of attributes k as depicted in (11). E_{EC} corresponds to the energy consumed by the ECs, and is equal to $k * P_{\text{EC}} * T$. The maximal power consumed by EC is 0.01 mW. E_{min} is the minimal dynamic energy of the system without any monitoring, $P_{\text{min}} * T : P_{\text{min}}$ is 10 mW, which corresponds to the minimal dynamic power consumed by the system. P_{model} is the energy needed to estimate the power by a given model, which is equal to $P_{\text{model}} * C_{\text{model}}$, where C_{model} is the required computation time that is shown in Fig. 12 for several values of k

$$E_{\text{ov}} = \frac{E_{\text{extra}}}{E_{\text{min}}} = \frac{k \times E_{\text{EC}} + E_{\text{model}}}{E_{\text{min}}} = \frac{k \times P_{\text{EC}} \times T + P_{\text{model}} \times C_{\text{model}}}{P_{\text{min}} * T}. \quad (11)$$

In all cases, we considered the worst case scenario, e.g., we assumed that ECs were always activated. Fig. 14 reports the minimum energy overhead that corresponds to the cost in energy of the three models for k equal to 3 (lowest sufficient accuracy), and the maximum energy overhead that is the cost of the three models at higher precision ($k = 8$ for LM and $k = 6$ for both NN and MARS). In the context of adaptive systems targeting energy efficiency, it is clear that the preferred method will be the ones involving the lowest energy overheads. Regarding this criterion, the maximum energy overhead of LM (dotted blue line) varies between 11% at 100 μs and 2% at 1 ms, while the minimum energy (blue line) is between 4.2% at 100 μs and 0.8% at 1 ms. NN and MARS are more expensive: the minimum energy of NN (red line) is between 12% at 100 μs and 2% at 1 ms, while for MARS (green line) varies between 20% at 100 μs and 3% at 1 ms. We remind that the worst-case energy is considered in this experiment, where ECs are designed without any optimization and the overhead is compared to the minimum dynamic energy consumed by the system without any monitoring. However, this overhead is much lower against the total energy consumption [adding $P_{\text{static}} * T$ to the denominator in (11)].

F. Result Analysis

The previous experiments have evaluated the precision, the computation cost and the energy overhead of the proposed method. The best monitoring method is the one that has the lowest cost and the highest precision. According to the previous study, the number of signals k and the sampling period T, have a direct impact on the model precision, the performance and the energy overheads. Therefore, several tradeoffs between the accuracy and the overhead can be concluded. The first solution might be the linear model with three signals, where the models fit most of power variations for a fine-grained adaptation. The average error is about 4%, and the computation and the energy overheads are equal to 2.7% and 4%, respectively, at the finest granularity 100 μs. The second solution may use the MARS model with only three (or four attributes), where the models tracks very well all power variations and the average error reaches 2.5%. But, it requires a higher period of the control loop compared to the first solution: this is because of the overheads that achieves less than 5% for a sampling period greater than 700 μs. However, the neural network based solutions can be eliminated, because they achieve a precision equal to the first solution with a highest computation and energy overheads.

Previously, in Section II, we presented the main factors that affect the power consumption, and we discussed the existing works in the literature. We remind that the internal activity is

TABLE V
AVERAGE ERROR OBTAINED IN EXISTING WORKS ARE COMPARED TO OUR METHOD WITH THE THREE MODELS USING SIX SIGNALS ($k = 6$)

Publication	Activity	Model	Time resolution	Average Error
[16]	switching events	Linear	100 μs	7%
[17]	switching events	Linear	80 μs	2.5%
[10]	performance events	Linear	1 ms	7%
[11]	performance events	Linear	440 ms	6%
[12]	performance events	Linear	500 ms	8%
Our method	switching events	Linear	100 μs	3.8%
		NN		4%
		MARS		2.8%

one factor that can be appraised at different levels of abstraction. Table V compares the error of the three models using six signals (the highest accuracy achieved) with the previously presented works that only consider the variation of the activity in their models. The most similar approaches are the ones that appraise the system activity using the toggling information with a linear model at the hardware level [16], [17]. The obtained average error in [16] for the small circuits (RAM and DSP) is about 7% at the resolution of 100 μs, while in [17] 2.5% of average error was achieved using nine signals for also small components such as a divider, a multiplier and an instruction/data caches. Our obtained average error is much lower compared to [16] at the same granularity, and is comparable to [17] with a lower overhead as the number of signals used in our linear model is lower (6 in this case). Also, our proposed approach was experimented on a more complex SoC.

On the other hand, several methods presented in the related works use the performance events from the PMU to estimate the system activity at higher levels [10]–[12]. The linear model constructed in [10] for the Pentium IV board has achieved 7% of average error with a fine grain resolution equal to 1 ms, while comparable errors were achieved in [11], and in [12] for the big core of the *ARM big.LITTLE* processor at a much higher granularity. Although our method was tested on a smaller circuit, but it is more accurate at finer timing resolutions; it is equal to 3.8% at 100 μs compared to 6% at 440 ms in [11] and 8% at 500 ms in [12]. To summarize, the studied existing and most similar approaches use a linear combination to estimate the power in terms of the system activity. Our solution shows a better accuracy with the same regression, and also goes further in modeling the power by introducing both NN and MARS models, which can be interesting for more complex systems such as MPSoCs.

The overhead of the monitoring is poorly addressed in the literature, where most of the techniques have evaluated only the accuracy of the estimations. In order to compare our results with existing works, we compared our monitoring method with the PMU, which is widely used for power estimation purposes. Ho *et al.* [33] have presented an architecture of a PMU for the LEON3 system with six hardware counters. Obtained area overhead on the *Xilinx ML605 Virtex-6* FPGA is about 3.2% of flip-flops and 6% of LUTs. However, our proposed monitoring method with three ECs requires only 1.53% and 0.96% of additional flip-flops and LUTs, respectively.

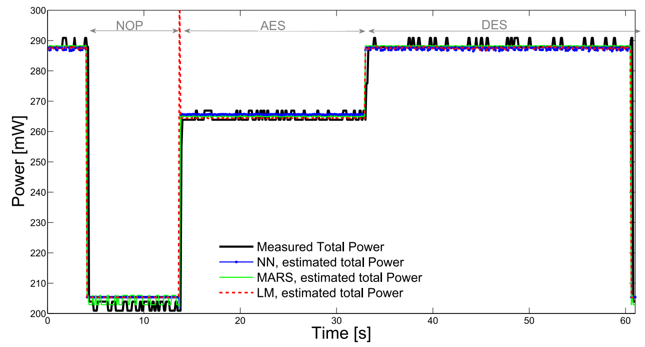


Fig. 15. Comparison between the monitored total power and the one measured by the off-chip power monitor on the Atlys FPGA at 100 ms timing granularity.

G. Validation

In all previous experiments, the reference power values were obtained from the power estimation tool XPower. In this section, the objective is to implement the proposed monitoring method on the *Atlys* FPGA that offers an off-chip power monitor based on linear technology's *LTC2481C* sigma-delta analog-to-digital converters. Similar to the simulations, we also considered the SecretBlaze SoC, but unlike the previous experiments, it executes three new applications: 1) two symmetric crypto algorithms advanced encryption standard and data encryption standard and 2) NOP application that has only "nop" instructions. Three ECs were added to the SoC to count the toggling activity on the previously selected signals: $S2$, $S4$, and $S5$ illustrated in Table IV. The power estimated by our on-chip monitoring was compared to the power measured by the off-chip power monitor for a timing resolution equal to 100 ms.

The tracking of power is shown in Fig. 15, where the black corresponds to the total power measured by the power monitor and the red, blue and green are the estimated values by LM, NN, and MARS, respectively. Our method achieves the accuracy of a power sensor, where the obtained average error is about 0.9% compared to 1% the error of the off-chip *LTC2481C*. It therefore becomes possible to monitor the power consumption at fine-grained timing resolution using an on-chip lightweight monitoring subsystem. Finally, this experiment highlight the reliability of the analysis done about the signals selection and power modeling.

VIII. CONCLUSION

We have proposed a new approach for the run-time monitoring of the dynamic power in SoCs, that is achieved by the insertion of ECs to monitor the relevant toggling activity on some strategic signals at the RT-level and using a dedicated power model. We proposed a method inspired from data mining algorithms to produce a lightweight and accurate monitoring subsystem. First, we introduced the power monitoring flow, which generates the databases from the offline simulations. We then presented a combination of algorithms from the attributes selection methods that select the relevant signals from the databases. Three classifiers were also explained and compared in this paper to model the dynamic power: the linear model, the neural network and the MARS model. This approach was demonstrated on a basic SoC architecture implemented on the *Xilinx Spartan-6 LX45* FPGA. A useful study was done in the experiments, from which we compared the accuracy and the overhead of all models.

Our experimental results indicate that our approach is able to automatically identify and select a small number of signals correlating with the power without any previous knowledge. The useful study of the different tradeoffs between the accuracy of models and the overhead in terms of area, energy and computation, revealed the originality of our approach that helps designers to easily set up a lightweight and accurate monitoring subsystem for the modern SoCs. The validation of our method by a real power measurement shows the efficiency of the proposed monitoring that is able to achieve the accuracy of power sensors at finer granularity. In future works, we will investigate additional parameters in order to have a model capable of taking temperature, process variations, frequency, etc., into account.

REFERENCES

[1] H. Sasaki, S. Imamura, and K. Inoue, "Coordinated power-performance optimization in manycores," in *Proc. 22nd Int. Conf. Parallel Architect. Compilation Tech. (PACT)*, Edinburgh, U.K., Sep. 2013, pp. 51–61.

[2] T. Ebi, M. A. Faruque, and J. Henkel, "TAPE: Thermal-aware agent-based power econom multi/many-core architectures," in *IEEE/ACM Int. Conf. Comput.-Aided Design Dig. Tech. Papers*, 2009, pp. 302–309.

[3] G. Kornaros and D. Pnevmatikatos, "A survey and taxonomy of on-chip monitoring of multicore systems-on-chip," *ACM Trans. Design Autom. Electron. Syst.*, vol. 18, no. 2, p. 17, 2013.

[4] S. Bhagavatula and B. Jung, "A low power real-time on-chip power sensor in 45-nm SOI," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 7, pp. 1577–1587, Jul. 2012.

[5] S. Bhagavatula and B. Jung, "A power sensor with 80ns response time for power management in microprocessors," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, San Jose, CA, USA, Sep. 2013, pp. 1–4.

[6] A. Pathania, A. E. Irimiea, A. Prakash, and T. Mitra, "Power-performance modelling of mobile gaming workloads on heterogeneous MPSoCs," in *Proc. 52nd Annu. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2015, pp. 1–6.

[7] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras, "Predictive dynamic thermal and power management for heterogeneous mobile platforms," in *Proc. Design Autom. Test Europe Conf. Exhibit.*, Grenoble, France, 2015, pp. 960–965.

[8] S. Eyerman, K. Hoste, and L. Eeckhout, "Mechanistic-empirical processor performance modeling for constructing CPI stacks on real hardware," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Austin, TX, USA, 2011, pp. 216–226.

[9] W. Choi, H. Kim, W. Song, J. Song, and J. Kim, "ePRO-MP: A tool for profiling and optimizing energy and performance of mobile multiprocessor applications," *Sci. Program.*, vol. 17, no. 4, pp. 285–294, 2009.

[10] F. Bellosa, A. Weissel, M. Waitz, and S. Kellner, "Event-driven energy accounting for dynamic thermal management," in *Proc. Workshop Compilers Operating Syst. Low Power (COLP)*, vol. 22. New Orleans, LA, USA, 2003, pp. 1–10.

[11] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchitect.*, San Diego, CA, USA, 2013, pp. 93–104.

[12] M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra, and S. Vishin, "Power-performance modeling on asymmetric multi-cores," in *Proc. Int. Conf. Compilers Architect. Synth. Embedded Syst. (CASES)*, Montreal, QC, Canada, 2013, pp. 1–10.

[13] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *IEEE Trans. Comput.*, vol. 61, no. 4, pp. 563–577, Apr. 2012.

[14] E. Senn, J. Laurent, N. Julien, and E. Martin, "SoftExplorer: Estimation, characterization, and optimization of the power and energy consumption at the algorithmic level," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation* (LNCS 3254), E. Macii, V. Paliouras, and O. Koufopavlou, Eds. Heidelberg, Germany: Springer, 2004, pp. 342–351.

[15] S. K. Rethinagiri, R. B. Atitallah, S. Niar, E. Senn, and J.-L. Dekeyser, "Hybrid system level power consumption estimation for FPGA-based MPSoC," in *Proc. IEEE Int. Conf. Comput. Design VLSI Comput. Process.*, Amherst, MA, USA, 2011, pp. 239–246.

[16] I. Mansouri, P. Benoit, L. Torres, and F. Clermidy, "Fine-grain dynamic energy tracking for system on chip," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 6, pp. 356–360, Jun. 2013.

[17] J. Peddersen and S. Parameswaran, "CLIPPER: Counter-based low impact processor power estimation at run-time," in *Proc. Asia South Pac. Design Autom. Conf. (ASP-DAC)*, Yokohama, Japan, Jan. 2007, pp. 890–895.

[18] R. Zamani and A. Afsahi, "A study of hardware performance monitoring counter selection in power modeling of computing systems," in *Proc. Int. Green Comput. Conf. (IGCC)*, 2012, pp. 1–10.

[19] P. Langley, "Selection of relevant features in machine learning," in *Proc. AAAI Fall Symp. Relevance*, vol. 184. 1994, pp. 245–271.

[20] R. Caruana and D. Freitag, "Greedy attribute selection," in *Proc. 11th Int. Conf. Mach. Learn.*, New Brunswick, NJ, USA, 1994, pp. 28–36.

[21] L. Xu, P. Yan, and T. Chang, "Best first strategy for feature selection," in *Proc. 9th Int. Conf. Pattern Recognit.*, Rome, Italy, Nov. 1988, pp. 706–708.

[22] Q. Zhu, L. Lin, M.-L. Shyu, and S.-C. Chen, "Feature selection using correlation and reliability based scoring metric for video semantic detection," in *Proc. IEEE 4th Int. Conf. Semantic Comput. (ICSC)*, Pittsburgh, PA, USA, Sep. 2010, pp. 462–469.

[23] J. H. Friedman, "Multivariate adaptive regression splines," *Ann. Stat.*, vol. 19, no. 1, pp. 1–67, 1991.

[24] D. Chen, J. Cong, and Y. Fan, "Low-power high-level synthesis for FPGA architectures," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Seoul, South Korea, 2003, pp. 134–139.

[25] C. Najoua, B. Mohamed, and B. M. Hedi, "Accurate dynamic power model for FPGA based implementations," *IJCSI Int. J. Comput. Sci.*, vol. 9, no. 2, pp. 84–89, 2012.

[26] R. Piscitelli and A. D. Pimentel, "A signature-based power model for MPSoC on FPGA," *VLSI Design*, vol. 2012, pp. 1–13, Jan. 2012.

[27] L. Barthe, L. V. Cargnini, P. Benoit, and L. Torres, "The secretBlaze: A configurable and cost-effective open-source soft-core processor," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Workshops Phd Forum (IPDPSW)*, Anchorage, AK, USA, 2011, pp. 310–313.

[28] R. Herveille *et al.*, "WISHBONE system-on-chip (SoC) interconnection architecture for portable IP cores," *Revision B*, vol. 3, pp. 4–32, Sep. 2002.

[29] M. Hall *et al.*, "The WEKA data mining software: An update," *SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[30] G. Jekabsons. (2011). *Areslab: Adaptive Regression Splines Toolbox for MATLAB/Octave*. [Online]. Available: http://www.cs.rtu.lv/jekabsons/

[31] O. Roeva, S. Fidanova, and M. Paprzycki, "Influence of the population size on the genetic algorithm performance in case of cultivation process modelling," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Kraków, Poland, Sep. 2013, pp. 371–376.

[32] P. Sibi, S. A. Jones, and P. Siddarth, "Analysis of different activation functions using back propagation neural networks," *J. Theor. Appl. Inf. Technol.*, vol. 47, no. 3, Jan. 2013.

[33] N. Ho, P. Kaufmann, and M. Platzner, "A hardware/software infrastructure for performance monitoring on LEON3 multicore platforms," in *Proc. 24th Int. Conf. Field Program. Logic Appl. (FPL)*, Munich, Germany, Sep. 2014, pp. 1–4.

Mohamad Najem (S'14) received the B.S. and M.S. degrees in electronic and informatics systems from the University of Pierre and Marie Curie, Paris, France, in 2008 and 2012, respectively, and the Ph.D. degree in automatic and microelectronic systems from the University of Montpellier, Montpellier, France, in 2015.

He is currently a Post-Doctoral Fellow with ENSTA-Bretagne Engineering School, Brest, France, and LabSTICC Research Institute, Brest. His current research interests include the monitoring of self-adaptive systems, statistical analysis, and the virtualization of field-programmable gate arrays.

Pascal Benoit (M'06) received the M.S. and Ph.D. degrees in microelectronics from the University of Montpellier, Montpellier, France, in 2001 and 2004, respectively.

He joined the Karlsruhe Institute of Technology, University of Karlsruhe, Karlsruhe, Germany, where he was a Scientific Assistant. Since 2005, he has been a permanent Associate Professor with the Montpellier Laboratory of Informatics, Robotics and Microelectronics, University of Montpellier. He has co-authored over 130 publications in books, journals, and conference proceedings, and holds five patents. His current research interests include self-adaptive and secured approaches for embedded systems.

Mohamad El Ahmad received the license degree from Lebanese University, Beirut, Lebanon, in 2012, and the master's degree in microelectronics system from the University of Montpellier, Montpellier, France, in 2014, where he is currently pursuing the Ph.D. degree with the Montpellier Laboratory of Informatics, Robotics and Microelectronics, Department of Computer Engineering.

His current research interests include performance, power and thermal management of multiprocessor system-on-chip, and reconfigurable/embedded processors and systems.

Gilles Sassatelli (M'03) received the M.S and Ph.D degrees in microelectronics and control automation from the University of Montpellier, Montpellier, France, in 1999 and 2002, respectively.

He is a Senior Research Scientist with the CNRS Montpellier Laboratory of Informatics, Robotics and Microelectronics, University of Montpellier. He is currently the leader of the Adaptive Computing Group composed of seven permanent staff researchers and 20 Ph.D. candidates. He has published over 200 publications in a number of peer-reviewed renowned international conferences and journals. His current research interests include reconfigurable and multiprocessor architectures for embedded and high performance computing.

Lionel Torres (M'97) received the master's and Ph.D. degrees from the University of Montpellier, Montpellier, France, in 1993 and 1996, respectively.

From 1996 to 1997, he was an IP Core Methodology Research and Development Engineer with ATMEL, San Jose, CA, USA. From 1997 to 2004, he was an Assistant Professor with the Polytech Montpellier Engineering School (Microelectronic Design) and the Montpellier Laboratory of Informatics, Robotics and Microelectronics, University of Montpellier, where he has been a Full Professor, since 2004, and was the Head of the Microelectronic Department, from 2007 to 2010. He is currently the Deputy Head of Polytech Montpellier (Engineering School of Montpellier) in charge of research, international, and industrial relationship. He leads several national, and European projects concerning microelectronic design and applications, and he is the Co-Founder of ALGODONE, Montpellier, France. He is involved in different major conference as Design, Automation and Test in Europe Conference, Very Large Scale Integration Conference (VLSI), Field-Programmable Logic and Applications Conference, IEEE Computer Society Annual Symposium on VLSI, and design automation conference, and has co-authored over 40 journal papers and 150 conference publications. He holds around ten patents. His current research interests include adaptive very large-scale integration architecture.