Method for dynamic power monitoring on FPGAs

Mohamad Najem, Pascal Benoit, Florent Bruguier, Gilles Sassatelli, Lionel Torres

Method for Dynamic Power Monitoring on FPGAs

Mohamad Najem, Pascal Benoit, Florent Bruguier, Gilles Sassatelli, Lionel Torres
LIRMM – UMR CNRS 5506 – University of Montpellier 2
Montpellier, France
{mohamad.najem, pascal.benoit, florent.bruguier, sassatelli, lionel.torres}@lirmm.fr

***Abstract*— The ever-increasing integration densities make it possible to configure multi-core systems composed of hundreds of blocks on existing FPGAs that may influence overall consumption differently. Observing total consumption is not sufficient to accurately assess internal circuit activity to be able to deploy effective adaptation strategies. In this case monitoring techniques are required. This paper presents a CAD flow for high-level dynamic power estimation on FPGAs. The method is based on the monitoring of toggling activity for relevant signals by introducing event counters. The appropriate signals are selected using the Greedy Stepwise filter. Our approach is based on a generic method that is able to produce a power model for any block-based circuit. We evaluated our contribution on a SoC RTL model implemented on Spartan3, Virtex5, and Spartan6 FPGAs. A power model and monitors are automatically generated to achieve the best tradeoff between accuracy and overhead.**

Keywords; FPGA; System-on-Chip; Power Modeling; Power Monitoring;

I. INTRODUCTION

Energy efficiency is one of the main challenges facing hardware and software designers. Different techniques ranging from silicon to application abstraction level must be applied to efficiently reduce power consumption. The power consumed is due to switching (dynamic power) and leakage (static power), and therefore depends on many different parameters, including power supply voltage, circuit frequency, load equivalent capacitance, toggling activity, but also temperature and transistor characteristics including threshold voltage. Many run-time techniques can be used to reduce dynamic power, *e.g.* Dynamic Voltage Frequency Scaling (DVFS), task migration, power gating, etc. To efficiency optimize the power-to-performance tradeoff, those adaptations require a monitoring subsystem.

In this paper, the objective is to propose a complete generic method for estimating at run-time the power consumed by any system running on any FPGA. This generally requires external equipment (not applicable for adaptive embedded systems) or dedicated analog sensors (which is area hungry, with a limited configurability and sometimes even not always available). Our approach is completely different: dynamic power is appraised based on toggling activity in the design. In this way, the instantaneous power is periodically evaluated by the system itself. In Fig. 1, the proposed method is exemplified on a System On chip (SoC): Event Counters (EC) report the activity on some chosen nets, the processor retrieves these values, and on the basis of a model generated by the tool, it evaluates the overall consumption of the circuit. Several power-aware techniques can then be applied (task mapping, scheduling, frequency/voltage scaling, power gating), but this part is not in the scope of this paper.

To reach this goal, the main challenge is to find an efficient method able to select a few strategic nets and to build an accurate power model. Our main contribution is a tool for systematic run-time dynamic power monitoring on FPGAs, and includes the following points:

- A generic method to estimate dynamic power at high-level;
- A generic flow to extract events from signals at RTL-level;
- A statistical technique for power modeling and selection of nets;
- An evaluation of accuracy and overhead of the proposed method.

The remainder of this paper is organized as follows. In Section II, the limitations of most relevant power estimation techniques are discussed to highlight the need for this work. Section III is devoted to power modeling method and tools. Section IV describes the experiments and in the Section V we present our conclusion with suggestions for future research.

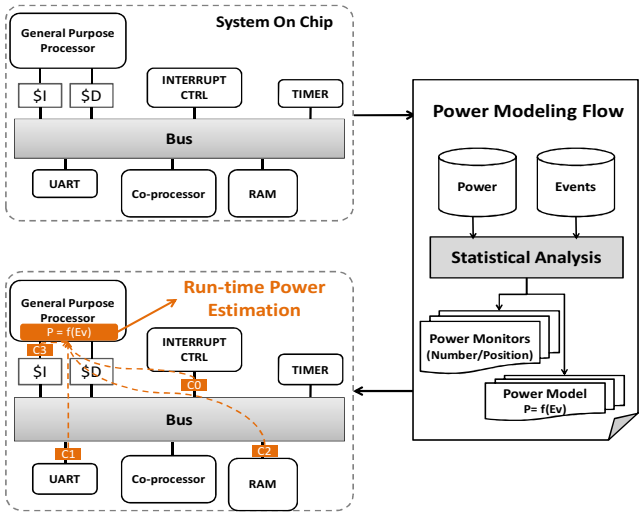

Fig. 1. Example of run-time power estimation using PMF.

II. RELATED WORK

Our goal is to monitor at run-time the consumption of a system running on an FPGA. Power estimation is one of the challenges to achieve this objective. For this purpose, several studies have

been conducted at different abstraction levels for FPGA circuits. In [1], pre-characterization-based macro-modeling is used to capture average switching power per access to both LUTs and registers, while in [2], switching activities are extracted from the Xpower tool. Instantaneous estimation is addressed in [3], with a cycle-accurate simulator and low-level models for interconnects, logic blocks and LUTs. In [4] and [5], the technique is based on execution profiles called "event signatures" used to estimate the average of total power consumed for a Multi-Processor System on Chip (MPSoC) on FPGA. Modeling is focused on individual components such as processors, bus fabrics, memories, and custom IP blocks. Many similar works have been done outside the context of FPGAs, for instance in [6] where power models are defined for each component from the SoCLib library based on their functional mode. Also an instantaneous linear-based power model was proposed in [7] and [8] based on performance counters such as Miss and CPI. A hybrid approach between circuit and design level power model is proposed in [9], in which the dynamic power is estimated independently for each component in terms of activity. The authors use data read miss and write miss to estimate the dynamic power for an interconnect component. Another approach based on Hidden Markov Models has been proposed in [10] to track at run-time the system power modes.

In the literature, the highest accuracy is achieved with fine grain simulations and low-level models, but these methods are too expensive for run-time monitoring. System-level models can be used for dynamic estimation but they are generally imprecise. In this article, we try with an original approach to bridge the gap between the abstraction level and the estimation accuracy. To the best of our knowledge, this solution is the first one to propose a systematic method for power monitoring for any system running on any FPGA.

III. CAD FLOW FOR POWER MODELING

To achieve accurate and low-cost on-chip dynamic power monitoring, we need to model instantaneous power (referred to as P_{dyn}). The proposed idea is based on an offline simulation from which activities and instantaneous power are recorded. The produced database is then used to identify strategic nets and to produce the predictive power model.

The corresponding Power Modeling Flow (PMF) depicted in the Fig. 2 is explained in detail in the following subsections. It has the three main processes:

- **Power Estimation Flow:** It estimates the $P_{dyn}(t)$ of a placed and routed netlist from a given RTL design and a given FPGA.
- **Event Extraction Flow:** It collects events $Ev(t)$ that occur on internal signals of an RTL design.
- **Power Modeling:** It creates a predictive power model in terms of events.

The offline observation is based on simulation. P_{dyn} is modeled as a function of events counted on specific nets. The challenge is to identify these nets and to produce a power model with the best tradeoff between accuracy and overhead. The PMF is a generic flow that can be applied to any RTL design for any Xilinx FPGA. It can be easily adapted to any vendor to support other technologies.

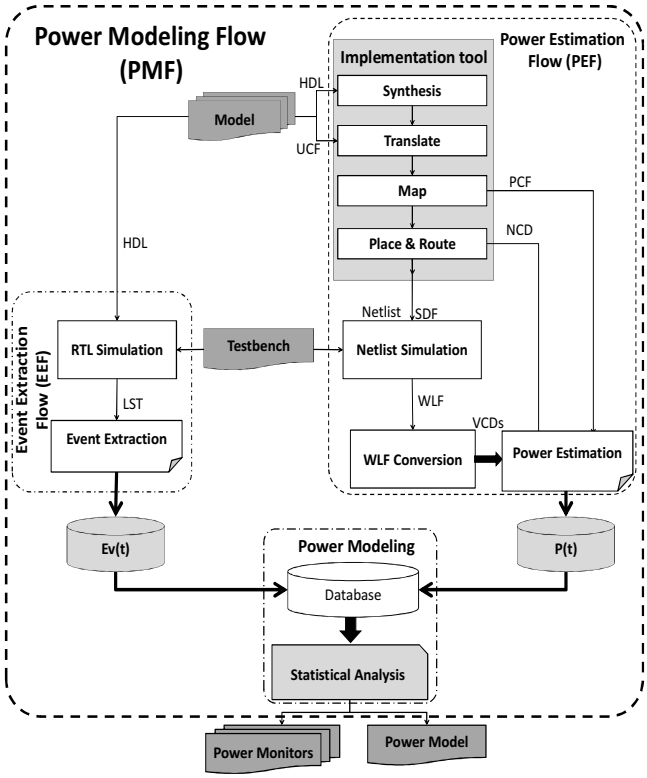

Fig. 2. Proposed Power Modeling Flow.

A. Power Estimation Flow (PEF)

The power estimation tools (*e.g.* Xpower) from FPGA vendors estimate the average power consumed. Since instantaneous power estimation is needed for monitoring, the PEF was established to estimate the instantaneous dynamic power $P_{dyn}(t)$ consumed by the circuit. The principle is to generate N power values after splitting the total time simulated by a time slicing.

The Xilinx tool chain (XST, translation, mapping, etc.) is used to synthesize the placed and routed netlist from HDL files. A simulation description file (WLF: Wave Long Format) is generated with simulation tools. The WLF is then cut into N sub-simulations as described in (1) thanks to *wlfinfo* command from ModelSim. T_0 is the time set to initialize the circuit and T_i 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 power estimation tool such as Xpower, to produce N estimations, which correspond to $P_{dyn}(t)$.

$$T = T_0 + N * T_i \quad (1)$$

B. Event Extraction Flow (EEF)

While PEF is devoted to estimating the $P_{dyn}(t)$, the EEF aims to collect events that occur on nets at the RTL level $Ev_i(t)$. At this stage, the RTL design is simulated using the same testbench and simulation time T as in the PEF flow. From this simulation, an image of signal values and the corresponding time is saved (an example is given in Fig. 3-a). It is then used to count events' occurrences on each single bit $Ev_i(t)$. Fig. 3-b plots the events detected for each change in signal value.

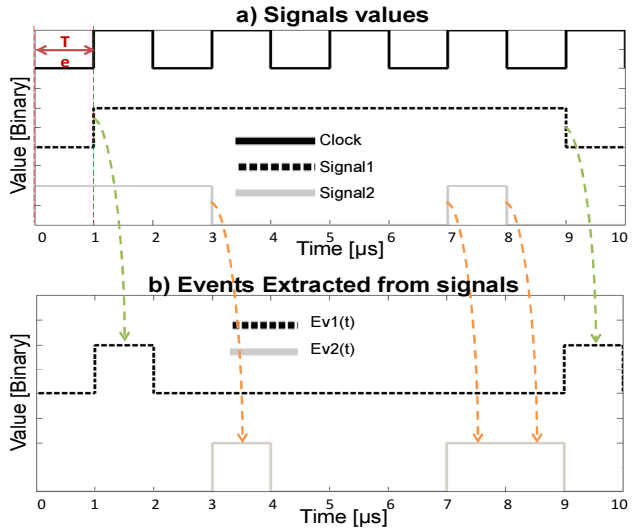

Fig. 3. Example of events extracted for two signals.

C. *Predictive Power Modeling*

In PEF, $P_{dyn}(t)$ is estimated while in EEF, the set of $Ev_i(t)$ is generated. The last step of PMF is (i) to identify the strategic nets in which ECs will be placed and (ii) to model $P_{dyn}(t)$ in terms of $Ev_i(t)$ for the selected signals.

1) Creation of the database

A database with both $P_{dyn}(t)$ and $Ev_i(t)$ is required for the subsequent statistical analysis. This database lists all the signals from the RTL description of the system. For each time interval T_i, the number of occurrences of events on all signals is counted. The $P_{dyn}(t)$ estimated by PEF is then added as a predictive variable. An example of the generated database is given in the Fig. 4: each line (in blue) is a time interval or *item*, and each column (sig_i) is a signal or *attribute*. The corresponding power or *class* for a given item is reported in the green column.

	A	B	C	D	E	F	G	H	I
1	Time	sig1	sig2	sig3	sig4	sig5	sig6	sig7	Power
2	20470	163	0	29	336	1	0	67	22.65486
3	40950	165	0	130	197	0	0	139	19.07762
4	61430	136	0	136	136	0	0	136	19.71793
5	81910	138	0	138	137	0	0	138	19.74456
6	102390	136	0	136	137	0	0	136	19.80862
7	122870	134	0	138	138	0	0	138	19.67234
8	143350	136	0	136	136	0	0	136	19.73729
9	163830	136	0	136	137	0	0	137	19.69253
10	184310	138	0	138	137	0	0	137	19.66534
11	204790	188	0	120	249	0	0	139	18.04136
	[ns]								[mW]

Fig. 4. Example of database generated.

2) Signal Selection and Linear Regression

Attributes and items are then analyzed in order to find the correlation between the power and a minimum number of attributes required to achieve good accuracy. For this purpose, data mining algorithms are used. First, we aim to reduce the number of attributes in the database. For this, we use a Greedy method to classify attributes (signals) into two populations: (i) attributes to use in the linear model and (ii) useless attributes. The Greedy Stepwise filter [11] searches greedily through the space of subsets of attributes. It may progress forward from the empty set or backward from the full set. The algorithm selects the attribute that has the highest R-Squared which is a statistical measure representing the correlation with the power. At each step, it selects the attribute that increases R-Squared the most. Then after adding the candidate to the output set, all candidate attributes are checked to see if their significance has been reduced below the specified tolerance level threshold. This algorithm stops adding variables when none of the remaining variables are significant or when the threshold of output set size is reached. The greedy strategy does not guarantee an optimal solution, but it can approximate a global optimal solution in a reasonable time.

Once strategic nets are selected, a predictive model of dynamic power is required to calculate the power with the resulting events extracted from selected signals. Regarding the obtained correlation coefficient value (>0.9), the complexity and the response time, a linear regression is an adequate method for modeling dynamic power (considering that this model needs to be embedded into the system itself, simplicity is a strong criterion). The linear equation (2) correlating the dynamic power with the events on the N selected signals. This equation is composed of a constant value P_0 and the term $w_i * Ev_i$, in which w_i represents the weight of Ev_i on power variations.

$$P_{dyn} = P_0 + \sum_{i=0}^{i=N} w_i * Ev_i \qquad (2)$$

IV. Experimental Evaluation

Our method does not require any external equipment or analog block, and can be applied to any RTL design on FPGA. An open-source system on chip (SoC) [12] is considered in this experiment as a case study. The chosen SoC is fully described in synthesizable VHDL code. It includes a 32-bit processor, an interrupt controller, UART, timer, instruction/data cache memories, and a Wishbone bus as interconnect. As dynamic power depends on the activity which in turn depends on the application executed by the processor, several applications are run in standalone mode (*i.e.* w/o any micro kernel). Two symmetric crypto algorithms are used: AES (Advanced Encryption Standard) and DES (Data Encryption Standard). The idea is to check if even similar algorithms lead to observable power behaviors. Our case study also considers a video decoding application MJPEG (Motion JPEG). A NOP application (with only "nop" instructions) is also used as reference. Execution times of these applications are listed in Table I. The hardware platform was synthesized using Xilinx 13.1 and simulated with ModelSim 10.1d for three different FPGAs (characteristics listed in table II).

TABLE I. Application Execution Time.

Application	**Execution time** (clock cycle)
AES	29910
DES	74313
Nop	51
MJPEG	31000

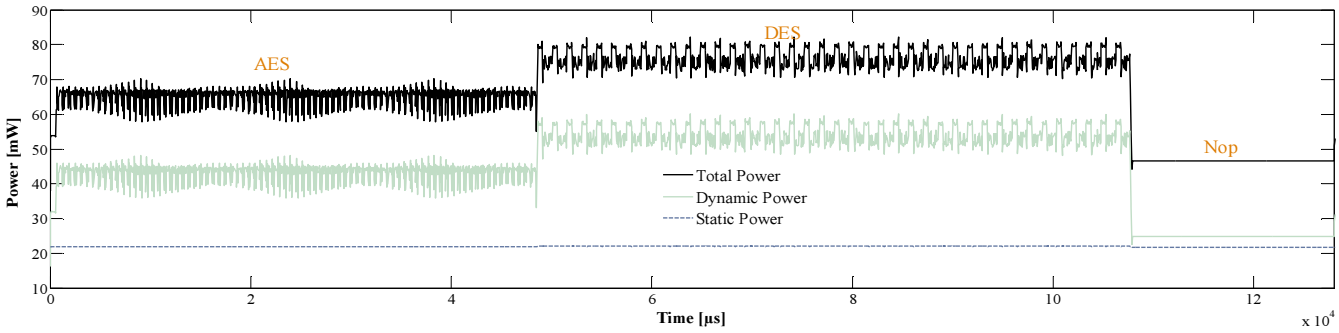

Fig. 5. Power estimated for AES, DES and NOP applications for xc6slx16 FPGA board.

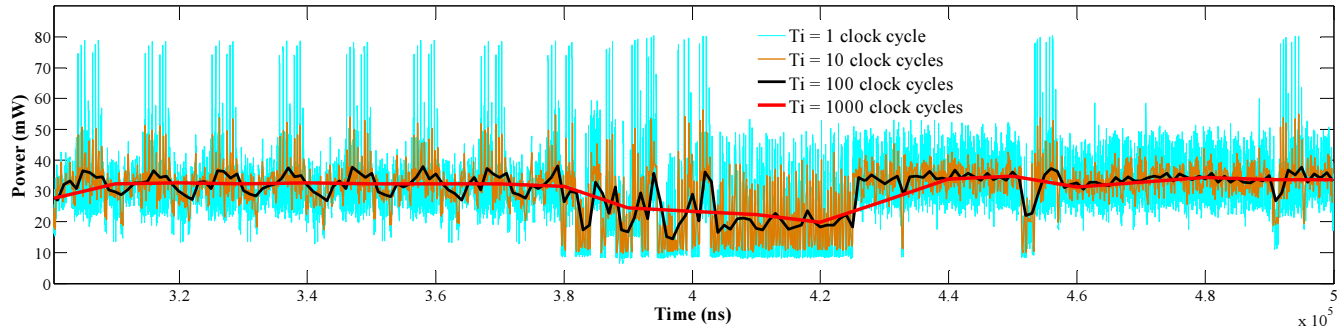

Fig. 6. High frequency power estimation.

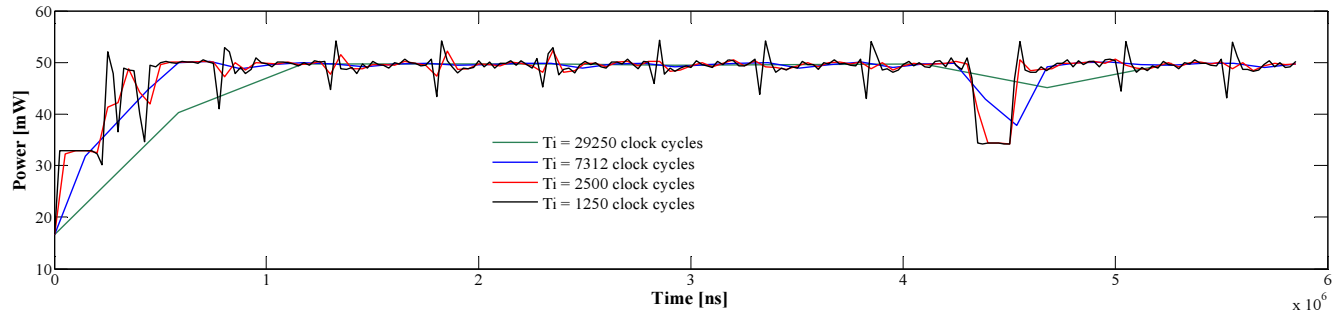

Fig. 7. Low frequency power estimation.

A. *Power Estimations*

The SoC was simulated while the processor executed the applications. In the testbench, we ran 80 iterations of the AES, 40 iterations of the DES, and 200 iterations of the NOP. Fig. 5 illustrates the total power $P_{total}(t)$, the dynamic power $P_{dyn}(t)$ and the static power $P_{static}(t)$ for the xc6slx16 Spartan6 FPGA. We notice that the DES application consumes more compared to AES, which in turn consume more than the NOP application. In addition, it is possible to identify the 40 iterations of DES due to its two levels of power unlike AES, for which we observe an aliasing effect due to the value of T_i (the impact of T_i is discussed further).

TABLE II. TARGET FPGAS.

FPGA	*Technology (nm)*	*Frequency (MHz)*	*LUT type*	*Vdd (v)*	*Speed grade*
Spartan3	90	25	LUT4	1.2	-4
Virtex5	65	50	LUT6	1	-1
Spartan6	45	50	LUT6	1.2	-3

TABLE III. AVERAGE ESTIMATED POWER.

Power [mW]		*Xc5vlx110t*	*Xc6slx16*	*Xc3s1000*
AES	*Static*	1048.50	21.94	99.44
	Dynamic	108.17	43.19	59.99
DES	*Static*	1048.65	22.05	99.63
	Dynamic	117.66	54.21	70.82
NOP	*Static*	1048.13	21.76	98.91
	Dynamic	84	24.85	26.31
MJPEG	*Static*	N/A	22.04	N/A
	Dynamic	N/A	53.14	N/A

The same testbench was applied for spartan3-1000 and Virtex5 FPGAs. Table III compares the average P_{total}, P_{dyn}, and P_{static} computed for the different FPGAs. Accordingly, applications rank is conserved according to their dynamic power consumption. While operating temperature, technology and process variations largely determine the static power for a specific device [13] (as shown in Table III, it can be approximated as constant for a given device), dynamic power consumption is completely design-dependent, and is

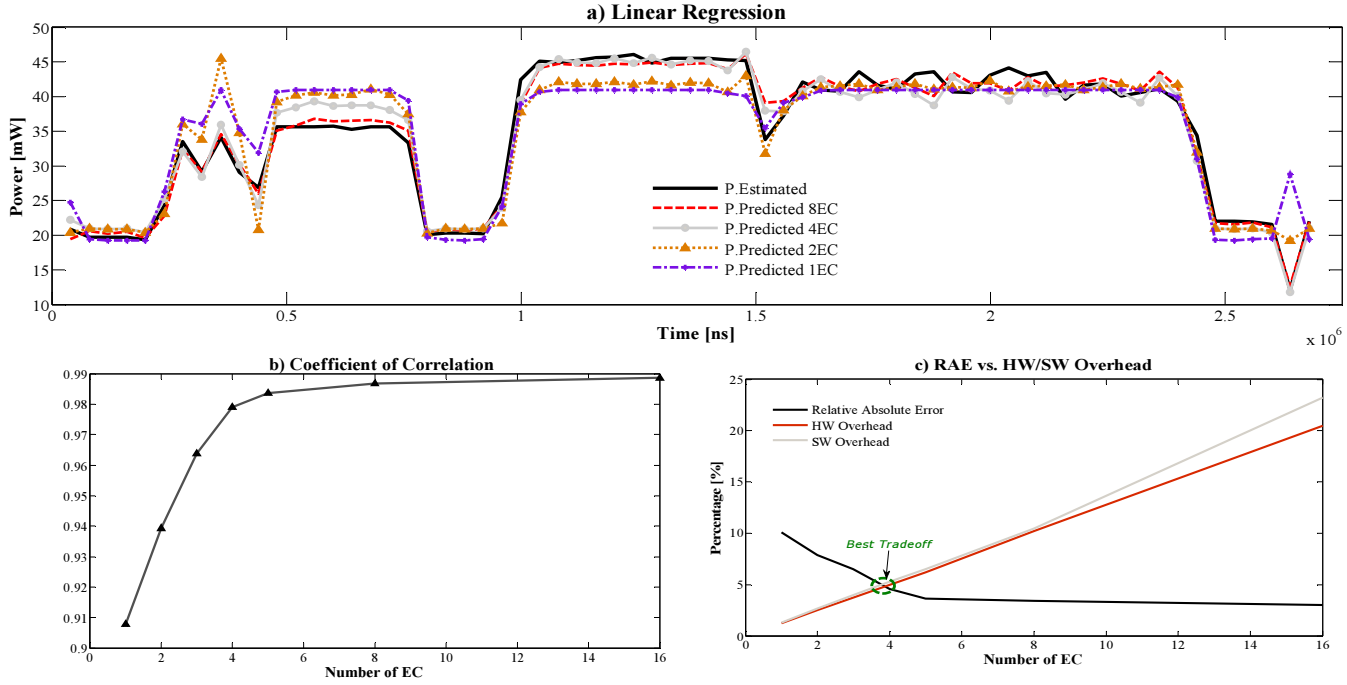

Fig. 8. Linear Models with several number of counters used.

determined by many factors including resource utilization, low-level features such as logic partition, mapping, placement, and routing, but also core frequencies (listed in Table II).

B. Sampling the Power estimation (T_i)

As observed in Fig. 5, the sampling period T_i may introduce aliasing effect in the estimation of $P_{dyn}(t)$. We applied the power estimation flow to the architecture while varying the value of T_i to observe the impact of it. In this testbench, the processor was executing 8 iterations of the AES before transmitting the result on the UART. The temperature was maintained at 25 °C with a 50 MHz clock frequency. Two sets of sampling periods are analyzed: (i) High frequency and (ii) Low frequency evaluation. In Fig. 6, the estimated $P_{dyn}(t)$ is shown for a part of the AES simulation with T_i equal to 1, 10, 100, and 1000 clock cycles, respectively. The dynamic power varies each clock cycle with high amplitude. By increasing T_i, the power comes closer to the average power variation. In the low frequency evaluations, the chosen values of T_i were respectively 29250 cycles, 7312 cycles, 2500 cycles and 1250 cycles. The logic behind the choice of T_i is to start sampling power for T_i equal to the execution time of the AES and then decrease it. Fig. 7 clearly shows that $P_{dyn}(t)$ remains average while missing out critical variations as the fall of power consumption due to the communication with the UART (print_uart) for high values of T_i.

To efficiently monitor dynamic power variations, we need to strike the balance between overhead and accuracy. The Nyquist-Shannon sampling theorem is used to generalize the choice of T_i [14]. As a result, the signal can only be reconstituted without aliasing if the sampling frequency F_s is higher than twice the highest frequency ($F_{max} = 1/T_{min}$). In the case of a programmable SoC, T_{min} should be set to the smallest execution time of a pattern of an application. In this example, we set T_{min} to 7500 clock cycles which is the execution time of print_uart. To better sample power variation without missing significant sons, the value of T_i should be lower than 3750 clock cycles. This phenomenon is clearly visible in Fig. 7.

C. Power Predictions

Once T_i was fixed as specified in the previous section, a database with 213 attributes and 131 items was generated for the considered testbench (AES, DES, NOP). We used the WEKA tool [11] for the statistical analysis of the database. WEKA, which is open-source software, contains a large number of configurable tools for data mining purposes (including the Greedy Stepwise and linear regression). In this work, it is used to select signals and produce the power model. The WEKA tool offers several heuristics, parameters, evaluators and classification methods that could be explored: however this topic is not in the scope of the paper. As explained before, Greedy Stepwise was chosen for the attribute selection and Linear Regression to generate the power model, with the set of parameters providing the best results.

TABLE IV. SIGNAL SELECTED BY GREEDY STEPWISE.

Signal name	Description
Wb_master_i_s_2	MSB bit for data bus from **ICache** to Wishbone Bus.
Wb_master_o_s_13	MSB 7th bit for address bus Wishbone Bus to **ICache**.
Wb_master_o_s_4	2nd bit of byte selects from Wishbone bus connected to **ICache**.
Wb_slave_o_s_0	MSB bit for data bus from the Wishbone bus to the **RAM**.

Table IV shows the four selected signals. The most representative signals are those related to the Instruction Cache (ICache). These four signals have the highest correlation with the dynamic power variation and are sufficient to accurately track system activity, as confirmed by the coefficient of correlation shown in Fig. 8-b. Fig. 8-a shows the dynamic

power estimated by the PEF flow along with the ones predicted by the linear models with 1 to 8 signals. Decreasing the number of ECs to less than 4 produces a model incapable of detecting all power modes among applications (*e.g.* the NOP power level). To examine the overhead of this approach, we assumed first a basic EC based on two 12-bit counters and one 12-bit register. They occupy 36 Slice Registers and 33 Slice LUTs on Spartan6, which correspond to the HW overhead. We also evaluated the computation cost of the linear power model by measuring the execution time. The processor needs 26 clock cycles to execute 1 multiplication and 1 addition, which correspond to the SW overhead introduced by one counter. The accuracy of the model (relative absolute error) vs HW/SW overhead is finally represented in Fig. 8-c. This confirms the ability of the proposed method to build an accurate power monitoring system with low overhead. For instance, the linear model based on 4 EC has 4% of average error, 7% of hardware overhead and 5.3% of additional load on the CPU.

D. Calibration of the predictive models

The power prediction models depend on the Xpower estimations. As stated in the literature [15], there are always some differences between the tools and the real values, which are due to approximations in power models, variability or device aging. We implemented on a XUP Virtex5 board the SoC architecture executing our tesbench (AES, DES, and NOP). We used the N6715B power analyzer from Agilent to supply the Virtex5 board and the N6781A Source Measure Unit (SMU) allowed us to capture the instantaneous current flow with a precision of 250 µA. The Fig. 9 shows the power measured by these instruments. This experiment reveals a 15% of average difference between estimations and measurements. The nets selection is not affected by this difference, so the proposed model can be calibrated with this information to estimate the real consumption. However, a fully adaptive system would require embedded PVT sensors (*e.g.* Ring Oscillator) to recalibrate the model itself, which will be part of the future works.

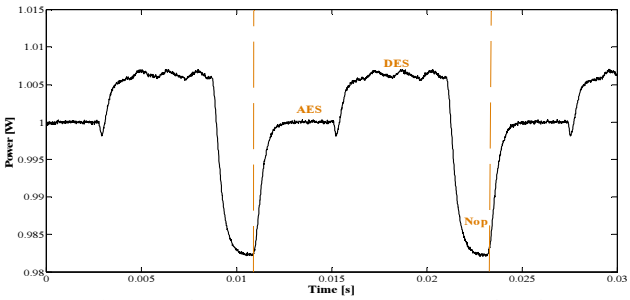

Fig. 9. Total power measured for xupv5 FPGA board.

V. CONCLUSION

The Power Modeling Flow (PMF) is a design tool that can systematically produce predictive models for any block-based architecture on FPGA, thus allowing the system to monitor at run-time its power consumption. First, we introduced the Power Estimation Flow (PEF) aimed at estimating the instantaneous power consumption. We then showed how to extract events from signals with the Event Extraction Flow (EEF). A linear equation correlating dynamic power with the appropriate event values after signal selection performed by the Greedy Stepwise was then introduced. This approach was demonstrated on a basic SoC architecture. Results obtained showed 4% of error between power estimated by the model and power estimated by Xpower with 7% area overhead by introducing four 12-bit counters. ECs are only one element in assessing system state. It was shown for instance that the estimations needed to be calibrated (because of process variation or aging for instance). 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] D. Chen, J. Cong, and Y. Fan, "Low-power high-level synthesis for FPGA architectures," in *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, 2003, pp. 134–139.

[2] C. Najoua, B. Mohamed, and B. M. Hedi, "Accurate dynamic power model for FPGA based implementations," *IJCSI International Journal of Computer Science*, vol. 9, no. 2, pp. 84–89, 2012.

[3] F. Li, D. Chen, L. He, and J. Cong, "Architecture evaluation for power-efficient FPGAs," *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays - FPGA '03*, p. 175, 2003.

[4] R. Piscitelli and A. D. Pimentel, "A High-Level Power Model for MPSoC on FPGA," in *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, 2011, no. ii, pp. 128–135.

[5] R. Piscitelli and A. D. Pimentel, "A Signature-Based Power Model for MPSoC on FPGA," *VLSI Design*, vol. 2012, pp. 1–13, 2012.

[6] R. Ben Atitallah, S. Niar, A. Greiner, S. Meftali, and J. L. Dekeyser, "Estimating energy consumption for an MPSoC architectural exploration," in *Proceedings of the 19th international conference on Architecture of Computing Systems*, 2006, pp. 298–310.

[7] M. Pricopi, T. S. Muthukaruppan, and V. Venkataramani, "Power-Performance Modeling on Asymmetric Multi-Cores," in *ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2013.

[8] I. Lee, H. Kim, P. Yang, S. Yoo, E.-Y. Chung, K.-M. Choi, J.-T. Kong, and S.-K. Eo, "PowerViP: Soc power estimation framework at transaction level," in *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, 2006, pp. 551–558.

[9] R. Nath and D. Carmean, "Power Modeling and Thermal Management Techniques for Manycores," in *Proceedings of International Symposium Computers and Communications of Low Power Design (ISCC)*, 2013.

[10] I. Mansouri, P. Benoit, L. Torres, and F. Clermidy, "Fine-grain dynamic energy tracking for system-on-chip," *IEEE Transactions on Circuits and Systems. Part II, Express Briefs*, vol. 60, no. 6, p. 4, 2013.

[11] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Burlington, USA: Elsevier Inc., 2011, p. 629.

[12] L. Barthe, L. V Cargnini, P. Benoit, and L. Torres, "The SecretBlaze: A Configurable and Cost-Effective Open-Source Soft-Core Processor," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, 2011, pp. 310–313.

[13] J. H. Choi, A. Bansal, M. Meterelliyoz, J. Murthy, and K. Roy, "Leakage power dependent temperature estimation to predict thermal runaway in FinFET circuits," in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, 2006, pp. 583–586.

[14] E. J. Candes and M. B. Wakin, "An Introduction To Compressive Sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21–30, 2008.

[15] J. P. Oliver and E. Boemo, "Power estimations vs. power measurements in Cyclone III devices," in *Programmable Logic (SPL), 2011 VII Southern Conference on*, 2011, pp. 87–90.