An efficient Framework for Power-Aware Design of Heterogeneous MPSoC

Rabie Ben Atitallah, IEEE Member, Eric Senn, Daniel Chillet, IEEE Member, Mickael Lanoe, Dominique Blouin

***Abstract*—Currently, designing low power complex embedded systems is a main challenge for corporations in a large number of electronic domains. There are multiple motivations which lead designers to consider low power design such as increasing lifetime, improving battery longevity, limited battery capacity, and temperature constraints, etc. Unfortunately, there is a lack of efficient methodology and accurate tool to obtain power/energy estimation of a complete system at different abstraction levels. This paper presents a global framework for power/energy estimation and optimization of heterogeneous MultiProcessor System on Chip (MPSoC). Within this framework, a power modeling methodology is defined and an open platform is developed. Our methodology takes into account all the embedded system relevant aspects; the software, the hardware, and the operating system. The platform stands for Open Power and Energy Optimization PLatform and Estimator (OPEN-PEOPLE). It includes diverse estimation tools with respect to their abstraction levels in order to cover the overall design flow. Starting from functional estimation and down to real boards measurements, our platform helps designers to develop new power models, to explore new architectures, and to apply optimization techniques in order to reduce energy and power consumption of the system. The usefulness and the effectiveness of the proposed power estimation framework are demonstrated through a typical embedded system conceived around the Xilinx Virtex II Pro FPGA platform.**

***Index Terms*—MPSoC, power modeling, system level estimation**

I. INTRODUCTION

THE increasing complexity of applications and System-on-Chip (SoC) architectures places embedded system designers in front of a very large design space. Exploring the design space to reach an efficient solution becomes very difficult, especially when the design must satisfy a large number of constraints, such as power and energy consumption. These constraints have led to introduce the usage of Multi-Processor System-on-Chip (MPSoC) which allow to integrate very complex systems. These MPSoC are generally heterogeneous and can contain memories (Cache, SRAM, FIFO, etc.), processors (GPP, DSP, etc.), interconnecting elements (Bus, Crossbar, NoC, etc.), I/O peripherals, and reconfigurable logic. To use the tremendous hardware resources available in next generation MPSoC efficiently, rapid and accurate design space exploration (DSE) methods are needed to evaluate the different design alternatives. MPSoCs must be designed with custom architectures to balance the implementation constraints between the application needs (i.e.: high computation rates and low power consumption) and the production cost. Nevertheless, the significant increase of complexity in such systems prevents designers from controlling the complete design flow. To guide the designer during the different design choices, the development of an efficient methodology and associated tools for power estimation and optimization is mandatory.

To be acceptable, the proposed methodology must include all the system-on-chip aspects, i.e. architecture/hardware, application/software, and management/operating system. Furthermore, the associated tools must be able to provide results from several description levels of the in-development system. Indeed, during the first design steps, designers have a very high description granularity of each part of the corresponding system. Nevertheless, first evaluations of power consumption can be necessary to make rapid and reliable design choices. This permits a rapid exploration of a large solution space by eliminating non-interesting regions from the DSE process. Gradually, the possible alternatives will be reduced by refinement of each part of the system. At a lower design step, the designer needs more accurate tools to explore the selected solutions in order to locate the most power-efficient configurations. At each step, different power evaluations can be extracted from a software or a hardware component relying on parametric power consumption models.

In the design flow, the power estimation process is centred around two aspects: *the power model granularity* and *the system abstraction level*. The first aspect concerns the granularity of the relevant activities on which the power model relies. It covers a large spectrum that starts from the fine-grain level such as the logic gate switching and stretches out to the coarse-grain level like the hardware component events. Fine-grain power estimation, in general, yields to more correlated model with data and to handle technological parameters which is tedious for system level designers. On the other hand, coarse-grain power models depend on micro-architectural parameters that cannot be determined easily. Let us highlight that the power estimation accuracy is not altered by the chosen granularity level, however it depends first on the characterization phase of each activity and second on the computing of the related occurrences while carrying out the application. Even we used coarse-grain activities, the characterization in term

Rabie Ben Atitallah is with LAMIH - CNRS UMR 8201, INRIA Lille-Nord Europe, University of Valenciennes, LAMIH DIM-ISTV2 - Le Mont Houy, 59313 Valenciennes Cedex 9, France (e-mail: rabie.benatitallah@univ-valenciennes.fr).

Eric Senn, Mickael Lanoe, and Dominique Blouin are with Lab-STICC - CNRS UMR 3192, University of Bretagne Sud, Centre de recherche, BP 92116, 56321 Lorient Cedex, France (e-mail: eric.senn@univ-ubs.fr, dominique.blouin@univ-ubs.fr, mickael.lanoe@univ-ubs.fr).

Daniel Chillet is with CAIRN - INRIA/IRISA, University of Rennes 1, ENSSAT, 6 rue de kerampont, BP 80518, 22300 Lannion, France (e-mail: daniel.chillet@irisa.fr).

of power or energy cost can be done at a lower level (board measurements, transistor, gate or RTL) and after that these values can be used at a higher abstraction level. The second aspect involves the abstraction level on which the system is described. It starts from the usual Register Transfer Level (RTL) level and extends till the algorithmic level. As we go from higher to lower levels, the power evaluation time increases, which is indirectly proportional to the accuracy. The above presented aspects are correlated. Indeed, different power estimation speed/accuracy trade-offs can be achieved according to the power model granularity and the abstraction level from which the relevant activities should be extracted.

To answer the above challenges, the OPEN-PEOPLE project [1] proposes a complete platform to ease the design of complex systems. It aims at providing a complete platform i) to allow rapid power/energy estimation for complex heterogeneous systems, ii) to test different optimizations in order to significantly reduce the power consumption of the system. Our contributions in this paper can be summarized as follows:

- First, we have defined a power modeling methodology that concerns the software and hardware layers to cover the overall embedded system consumption. Our methodology defines the relevant activities on which the power model relies. These activities are characterized using measurements on real boards. Afterwards, power models are elaborated by regression functions or simply recorded as multi-entries look up tables.
- Second, several abstraction levels are considered for power estimation incorporating the design flow steps. Starting with functional estimation, passing through simulation refinement, up to a native execution on the target platform (GPP, DSP, FPGA, etc.), different speed/accuracy trade-offs are obtained.
- Third, the OPEN-PEOPLE platform has been developed. It provides designers an adequate environment to build up new power models and to make relatively accurate estimates using tools at different abstraction levels.

This paper is organized as follows. After section II which presents the related works, section III exposes an overview of the OPEN-PEOPLE project. Our proposed system power modeling methodology is presented in section IV. In order to evaluate our approach, section V presents the experimental results for a typical MPSoC embedded system designed around the Xilinx Virtex II FPGA board. Finally, Section VI concludes this work and presents some perspectives.

II. RELATED WORKS

Significant research efforts have been devoted to develop tools for power consumption at the different abstraction levels in embedded system design. Among the existing tools for low abstraction levels, we can mention SPICE [1], Diesel [2], and PETROL [2] which operate at the RTL level. These tools are fairly accurate, but require significant amount of simulation time. At such low level, tools are used to optimize power consumption of hardware blocks but not to evaluate entirely complex SoC architectures.

[1] www.open-people.fr

To cope with the evaluation time, several tools have been developed for power consumption estimation at the system level. Among the wide-used approaches, we quote tools based on micro-architectural cycle-level simulation such as Wattch [3] and Simplepower [4]. They define fine-grain power models by characterizing component features such as a set of instructions or functional blocks using analytic power laws. The contributions of the internal unit activities are calculated and added together during the execution of the program on the micro-architectural simulator. This approach needs low-level description of the architecture which is often difficult to obtain for off-the-shelf processors. Though using cycle-level simulators has allowed accurate power estimation, the simulation time of complex MPSoC needed to achieve the results is still significant.

In an attempt to reduce simulation time, recent efforts have been done to build up fast simulators using *Transaction Level Modeling* (TLM) [5] [6]. SystemC [7] and its TLM 2.0 kit have become a de facto standard for the system-level description of SoC. The TLM kit proposes different coding styles to offer concepts for loosely and approximately timed models. However, there is no a standard definition for concepts or methodologies that involves power estimation at the TLM level and this aspect is still under research and is not well established. In [8] and [9], a methodology is presented to generate consumption models for peripheral devices at the TLM level. Relevant activities are identified at different levels and granularities. The characterization phase is however done at the gate level from where the activity and power consumption for the higher level are deduced. Using this approach for recent processors and systems is not realistic. In fact, recent processors have complex architectures; they may contain several pipeline slots, hierarchical memory system (L1 and L2 cache levels), and specific execution units such as the NEON architecture for the ARM Cortex A8. The power characterization phase at the gate level of each activity of these blocks needs a huge number of experiments and significant simulation time. Dhawada et al. [10] proposed a power estimation methodology for PowerPC and CoreConnect-based system at the TLM level. Their power modeling methodology is based on a fine-grain activity (processor instruction, data word transmission via the bus, etc.) characterization at the gate level which needs a huge amount of development time. Such fine characterization leads to a high correlation with data, hence authors announced a quite significant power estimation error. Compared to the previous works, our proposed methodology for power estimation also partially uses SystemC/TLM simulation with coarse grain power models.

For the functional level, Tiwari et al. [11] have introduced the concept of Instruction Level Power Analysis (ILPA). They associate a power consumption model with instructions or instruction pairs, which are characterized using measurements on a real chip. The power consumed by a program running on the processor can be estimated using an instruction-set simulator to extract instruction traces, and then adding up the total cost of the instructions. This approach suffers from the high number of experiments required to obtain the model. In addition, it can be applicable only for processors. To overcome

this drawback, Laurent [12] et al. proposed the *Functional Level Power Analysis* (FLPA) methodology that was successfully applied on building high-level power models for different hardware components (processor, memory, I/O peripherals, FPGA, etc.). FLPA relies on the identification of a set of functional blocks which influence the power consumption of the target component. The model is represented by a set of analytical functions or a table of consumption values which depend on functional and architectural parameters. Once the model is build, the estimation process consists of extracting the appropriate parameter values from the design, which will be injected into the model to compute the power consumption. Based on this methodology, the tool SoftExplorer [13] was developed. It includes a library of power models for simple to complex processors. Recently, SoftExplorer has been included as a part of Consumption Analysis Toolbox (CAT) [14]. CAT gives relatively precise power estimation results in a surprisingly small time. Indeed, only a static analysis of the code, or a rapid profiling are necessary to determine the input parameters for the power models. However, when complex hardware or software is involved, some parameters may be difficult to determine with precision. For instance, this is the case of cache miss rates in complex processors. This lack of precision may have a non-negligible impact on the final estimation accuracy, depending on the sensitivity of the parameter. In order to refine the value of sensible parameters in a reasonable delay, we propose in this paper to couple SystemC/TLM simulation with functional power modeling. Thus, a reasonable trade-off between estimation speed and accuracy will be reached.

For the reconfigurable circuits (FPGA), several studies have been done during last years. One of the first modeling proposal has been done in by Garcia et al. in [15], [16]. In these works, the power modeling is measured for the different elements of the circuit (LUT, register, I/O, clock tree, etc). The power consumption measured in this work concerns the active component, but the configurable memory is not considered, and the reconfiguration aspect is not evaluated. In [17], authors explain how the pipeline of some hardware functions can reduce the power consumption by the reduction of the clock frequency. In general, applying pipeline technique leads to increase the area of the hardware block. One important aspect is then to evaluate the trade-off between dynamic and static power. In particular, if the area of one specific hardware block increases the static power of this block will increase too. High-level estimations have also been developed for this type of circuit. For example, the works presented in [18] and [19] propose to model the power by using high level characteristics of the system. In [18], the signal statistics are used to extract the activity and then compute the power consumption. From the signal activity, it is possible to evaluate the activity of each hardware block and then to extract the power/energy consumption of each block. A composition of theses consumptions enables to evaluate the global consumption of the system. In [19], the high level characteristics of the functionality is used to model the power consumption. For example, the frequency of the hardware implementation of a functionality is used to estimate the power/energy consumption, and a sum of all the powers consume in the circuit enables to evaluate the power/energy of the system. When considering operating system level, the service which ensures the task scheduling and the task placement have an impact on the power consumption, and in particular on the static power consumption. The work presented in [20] shows that the reconfiguration must be done as late as possible to prevent leakage current in the reconfiguration memory, but this can be very interesting if and only if it is possible to configure a usable area of the circuit in a very low static power. Even if this technique exists, the trade-off between the configuration of this state and the static power saved by this specific configuration must be evaluated. The mentioned above FLPA approach was also applied to develop consumption models for FPGA at the system, algorithmic [21], and architectural levels [22], and to assess the consumption overhead due to hardware reconfiguration phases [23].

The recent evolution of the FPGA circuits, and in particular their capability to be dynamically and partially reconfigured, leads us to consider specific managements. To ensure this management, we propose to model the power/energy consumption of the reconfiguration step. From this model, we will develop scheduling algorithms which consider the power/energy consumption of the block during its activity and the power/energy consumption of the block during its configuration step. In order to be complete, our work will also include evaluation of the static power of the configuration memory.

The role of an operating system is essential in the context we are discussing here (heterogeneous multi-processors systems) mainly to benefit from a large variety of services to ease the exploitation of embedded platforms (cooperative and pre-emptive multi-tasking, process management, multi-threading, etc.) and to offer abstraction of the hardware that permit to reduce the time to design. Its impact on the energy consumption is however non-negligible. Several studies have studied this impact without actually proposing consumption models. [24] and [25] shown that the energy consumption can rise from 6% to 50% with an OS, depending on the application, and that it increases with the processor frequency and supply voltage. [26] shown that the OS can consume from 1% to 99% of the processor energy depending on the services called. In [27] was evaluated the overhead of using software trusted platform in the context of trusted boot Linux OS. A general study on aspects of OS design to improve energy efficiency was proposed in [28]. An other trend is to analyze the OS energy overhead from simulations at the micro-architectural level like with *Simbed* in [29], [30], or at the instruction level like with *Skyeye* [31], [32]. Such approaches inherit the drawbacks of the simulation level involved (time consuming cycle level simulations, simple processor models, larger errors, etc.) as explained formerly. Actual consumption models are only proposed in a few works [33], [34], or [35]. They however only consider simple systems or only sub parts of the operating systems functionality or services, and furthermore may be again limited by the accuracy of the energy simulators used.

In the frame of the OPEN-PEOPLE project, the particularity of our approach is that it is based on actual measurements on the electronic boards, and that it aims at proposing consumption models for every component in the embedded

systems considered. Following this direction, we propose models to take into account complete real-time embedded systems, including complex processors, reconfigurable components (FPGA), and dedicated OS services such as scheduling, context switching, or inter-process communications [14], [36].

III. OPEN-PEOPLE PLATFORM

OPEN-PEOPLE stands for Open Power and Energy Optimization PLatform and Estimator. The platform is defined for estimation and optimization of the power and energy consumption of complex electronic systems. Among the target systems, we mention heterogeneous MPSoC such as the TI OMAP 3530 [37] and reconfigurable circuits like the Xilinx Virtex5 FPGA [38]. Our platform allows power estimation using:

- direct access to the hardware execution boards and the measurement equipments. This first alternative enables designer to measure the real power dissipation of the target system. To do so, the low level description of the system (C, VHDL, etc.) is carried out natively on the target board. Furthermore, this alternative is used to build new power models for hardware or software component as it will be described in section IV. Several boards have been integrated in our automated bench and equipped with special gear to allow for power consumption measurement. Among those boards, one may find some processor based boards (OMAP 3530, OMAP L138) or some FPGA based boards (Spartan 6, Cyclone 3,LS, Aria 2 GX, Virtex 5, Virtex 2).
- a set of Electronic System Level (ESL) tools coupled with accurate power models elaborated within the first alternative. Mainly, we offer tools at the functional and transactional levels in the context of multilevel exploration of new complex architectures.

The figure 1 presents a global view of the platform which is based on two main parts; the software part and the hardware part. The software user interface ensures the access to the power measurements and helps the designer to define energy models for the hardware and software system components. From the measurements, the designer can build models and compute an estimation of the energy and/or power consumption of its system. In addition, from this software user interface, the hardware platform can be controlled. The hardware part consists of the embedded system boards, the measurement equipments, and the computer that controls these different elements and schedules the list of measurements required by different users.

In the frame of the OPEN-PEOPLE project, new methods and tools to model the different components of an heterogeneous system architecture are proposed including processors, hardware accelerators, memories, reconfigurable circuits, operating system services, IP blocks, etc. For reconfigurable system, the dynamic reconfiguration paradigm will be modeled to estimate how this feature can be used by Operating System (OS) to reduce the energy consumption. Furthermore, this project studies how the complete estimation and validation can be performed for very complex systems with a small simulation time.

IV. SYSTEM POWER MODELING METHODOLOGY

In the following, we may refer indifferently to energy or power models, knowing that passing from one to the other only involves the actual execution time of the object considered. Power and energy consumption are equally important concerns to us: the first is directly linked to the power dissipation and operating temperature of the hardware, the second impacts on the batteries size and lifetime. Also, we will use later the additive property of energy to build consumption models for complete systems.

In order to obtain the global consumption of a complete system, we propose a methodology mainly based on four phases, which are presented in the following paragraphs. The paragraph IV-A explains how the more consuming parts of a system are identified at first. Paragraph IV-B explains how the power and energy consumption of these parts is modeled and estimated. Paragraph IV-C describes the building of power models from consumption measurements, according to the estimation methodology exposed. Paragraph IV-D shows the usage of the developed power models in a power-aware design methodology and the benefit of simulations in refining some input parameters of the models to offer a better accuracy.

A. Consumption sources identification

The aim of this first step is to identify the sources of power consumption in the embedded system. As shown in figure 2, we consider an embedded system in his entirety: the software (i.e. the application code) at the top level, the hardware (the electronic board onto which the code is running) at the lowest level, and between them the operating system and associated services. Our estimation methodology is based on actual measurements on the targeted hardware. Our aim here is to identify the more consuming devices and services, and to make connection between them, and the tasks to be executed. For instance, one task obviously solicits the processor, cache, and memory, but also involves the process manager and the scheduler, which begets context switches and eventually more processing and memory accesses. The same task may also explicitly use Inter Process Communication (IPC) services, or need access to external peripherals. As we have seen in the state of the art section, the operating system energy overhead may take a considerable part of the overall system's consumption. It actually depends on the applications complexity and the number of services called. Our own works have corroborated the fact that the main contribution to be considered is coming from the memory and peripheral, and thus is strongly connected, beside processing, to data transfer and storage activities inside the system.

For the hardware tasks, the sources of consumption are generally different. Indeed, hardware tasks are generally data intensive tasks and the designer normally doesn't use operating system service calls for this type of computation. The source of consumption for one specific task is then not linked to the operating system execution but it is only dependent from the execution of the task on the configurable space.

Nevertheless, each hardware task consumes and/or produces data from/to others blocks (processors, memories, I/O, etc.),

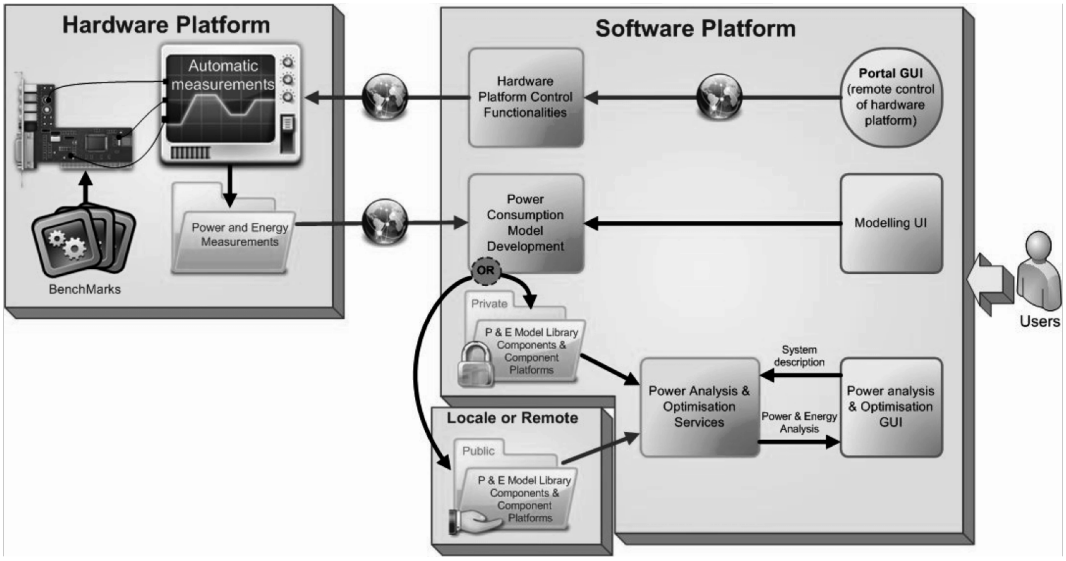

Fig. 1. Global view of the OPEN-PEOPLE platform.

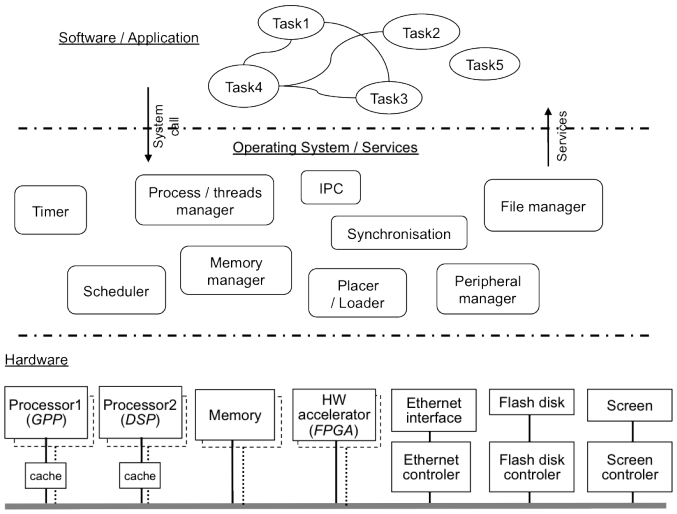

Fig. 2. View of an embedded system and application

so an important overhead due to data transfers can appear for these tasks. For example, when a software task running on a processor sends data to a hardware task, it can be compared as driver function call and it produces an energy overhead during the IPC OS service call of the software task.

B. Estimation methodology

The estimation methodology directly inherits from the previous analysis. In fact, even if the consumption profile may differ from one system to an other, some general rules exist that constitute the basis of our approach.

The different contributions of system parts in the global energy consumption (called here energy levels), come from the experimental modeling of different embedded applications and platforms, which is based on consumption measurements. The class of systems considered, as shown on figure 2, are processor or multiprocessor based, with or without operating systems. Operating systems studied so far are POSIX compliant. The consumption models presented in the next section, which are included in the Open-PEOPLE framework library, rely on the contributions exposed in this section. There is no model of computation really involved at this stage, since the global application model is intended to a static analysis of features related to the consumption. Simulations may be performed as a mean to refine those features in subsequent steps of the design flow (as described in section IV-D). Figures in this section describe the general context of the application class we are considering.

Variations of the power consumption with the time can be modeled as presented on figure 3 for software tasks running on processors, and on figure 4 for hardware tasks running on FPGA. Note that the energy is simply the area of every boxes on these figures. On figure 3, the bigger contribution is P_{ground}, which represents the power consumption of all the components when the system, without OS, is not running any application. This power consumption can be quite important especially for embedded systems on FPGA. Energy overhead of the different tasks and OS services comes in addition to this first one. More or less additional boxes may be considered depending on the actual system and application. To define these boxes, simulations of OS scheduling can help to extract the execution scenario, to define the OS overhead [39], and to propose dynamic power-aware techniques to optimize the consumption at runtime [40]. Indeed, an embedded system may or may not use a virtual memory subsystem, so the ”page fault” box might disappear. Again, dynamic reconfiguration may be used for reconfigurable hardware, and a ”reconfiguration overhead” box should be added for each task reconfiguration. Indeed, some tasks on the figure 3 might be implemented in a FPGA circuit. We then refer to figure 4 to represent the energy contribution of the tasks placed on this circuit. Here, two P_{ground} powers are represented. The first corresponds to the power consumed by the configurable memory plan which maintains the task configurations in place during

the execution, while the second represents the static power consumed by the active elements of the circuit (i.e. the static power for the configurable logic elements, the digital signal processing blocks, BRAM memories, interconnects, etc.). As we show there, for each task configured (or reconfigured) in the configurable space, an additional energy is necessary to load the bitstream within the configurable memory plan. Our experiments show that this energy mainly depends on the size of the bitstream file which has an impact on the configuration time, the position or the file content have a very limited impact on the configuration energy. Note that the figure 4 illustrated the partial and dynamic reconfiguration paradigm with the possibility to configure a specific part of the circuit while the remainder continues to execute the tasks. On this figure, we have illustrated the first configurations for tasks $HTask_i$, $HTask_j$ and $HTask_k$, for these first configurations, additional energies are necessary and this energy directly depends on the configuration file size. We also illustrated a specific scenario where the second execution of the task $HTask_i$ needs a reconfiguration phase, while the third execution doesn't need new reconfiguration because no other task has been configured at the same place. The figure 4 also shows the placer/loader activities to manage the reconfiguration process. For each reconfiguration, the placer/loader service is called, and the first step consists in finding a sufficient area on the reconfigurable area, the placer supports this job. The second step consists in loading the bitstream within the reconfigurable memory, this step is supported by the loader. As illustrated in the figure 4, for each reconfiguration, the placer is always executed, but the loader is optional when the task is already configured in the reconfigurable area.

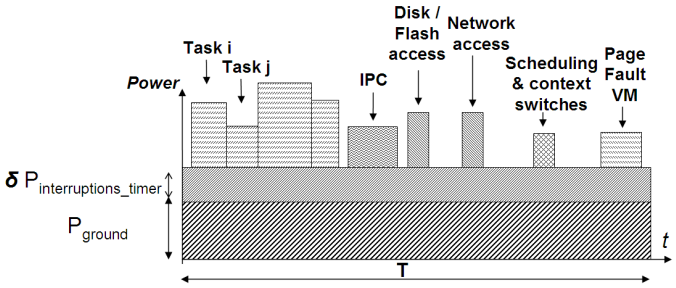

Fig. 3. Energy contribution of different parts of the application running on a hardware platform (software tasks on GPP or DSP) (For a better reading of the figure, the different boxes are not represented on the same scale)

The energy consumption of one software task may be generally modeled as shown on figure 5. It is the addition of the following contributions, or "energy levels" as we use to call them:

- L1: E_{ground} is the "ground" energy consumed during the task τ execution (with execution time $T(\tau)$). It is directly linked to P_{ground}.
- L2: δE_τ is the task's intrinsic contribution, without operating system. δP_τ power consumption includes the consuming resources directly implied in executing the task : the processor, with caches and primary memory δP_τ accesses. Like for the following δP, δP_τ is the difference between the power consumption considered at this point, and P_{ground}. A specific power model of the targeted processor is used to assess δP_τ.

$$\delta E_\tau = \delta P_\tau \times T(\tau)$$

- L3: δE_{tt} is the basic OS energy consumption due to timer ticks interruptions.

$$\delta E_{tt} = \delta P_{tt} \times T(\tau)$$

- L4: $\delta E_{scheduler}$ is the scheduler energy overhead. It includes context switches and scheduling operations.

$$\delta E_{scheduler} = (\delta P_{scheduler} + P_{ground}) \times \delta T_{scheduler}$$

- L5: δE_{IPC} is the energy due to communication and synchronization services.

$$\delta E_{IPC} = (\delta P_{IPC} + P_{ground}) \times \delta T_{IPC}$$

- L6: δE_{device} is the energy overhead incurred by accesses to peripherals (Flash, Ethernet, etc.).

$$\delta E_{device} = (\delta P_{device} + P_{ground}) \times \delta T_{device}$$

- L7: δE_{vm} is the energy overhead due to the OS virtual memory subsystem.

$$\delta E_{vm} = (\delta P_{vm} + P_{ground}) \times \delta T_{vm}$$

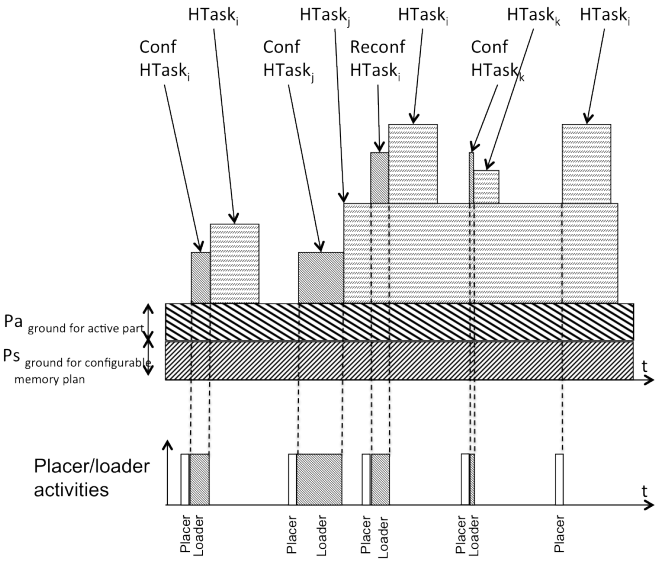

Fig. 4. Energy contribution of different parts of the general application running on a FPGA circuit (hardware tasks on FPGA)

It is remarkable that whereas a specific power model is used for every processor targeted (to estimate δP_τ), dedicated consumption models are defined for any additional OS services considered. We chose here not to use the processor power model for OS services in order first to keep the estimation time low, and second to avoid looking for the service code in the OS complete source code.

These previous contributions can be combined to define the energy of tasks and the energy of operating system services called by the task, as presented below

$$\begin{aligned} E_\tau &= E_{ground} + \delta E_\tau \\ E_{OS_\tau} &= \delta E_{tt} + \delta E_{scheduler} + \delta E_{IPC} + \delta E_{device} + \delta E_{vm} \end{aligned} \quad (1)$$

For the reconfigurable space, if we consider data intensive computation tasks which are not preemptable (to ensure a high performance execution) and without operating system service calls, the model of energy consumption can be defined through

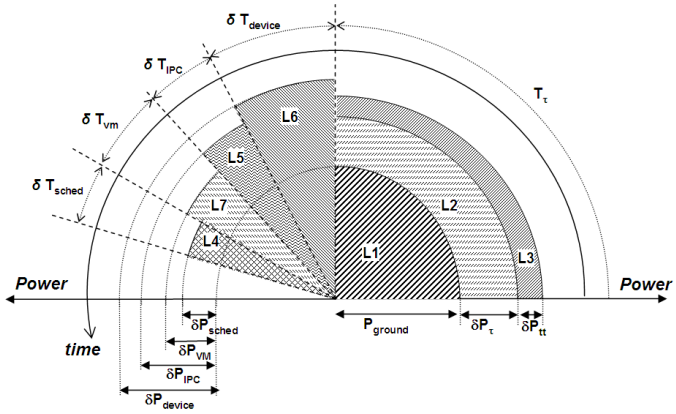

Fig. 5. A representation of the additivity property of energy : every surface represents a contribution in the task's energy consumption

the followings energy contributions

- Es_{ground} is the "ground" energy consumed by the configurable memory plan;
- Ea_{ground} is the "ground" energy consumed by the active elements of the global FPGA circuit. Even if these elements are not configured, a static energy is consumed by this part. We can note that the circuit can provide some mechanisms to configure the circuit with a low power static configuration which can significantly reduce the static power. We don't take this mechanism into account in this study;
- Et_i is the energy consumed by the hardware task $HTask_i$ during its execution. This energy must represent the consumption of the task with the corresponding data transfers; We can note that if we want to propose more flexibility, it is possible to define several implementations for the tasks, more details can be found in [41], [42]
- $Econf_i$ is the energy necessary to configure the task $HTask_i$ for its first execution. For this step, the operating system must manage the configuration, and this is ensured by the operating system service `Placer/Loader` in the figure 2. The execution of this service leads to active the `file manager` service of the operating system and thus leads to consume an important energy to access to the bitstream file ; The figure 4 doesn't shows this energy contribution begeted by the operating system calls, but this energy is included in the operating system execution. The $Econf_i$ presented here corresponds to the effective configuration operation, which is the write operation into the ICAP port (in the case of Xilinx circuit);
- $Ereconf_i$ is the energy necessary to reconfigure the task $HTask_i$ for the other executions. If we consider that the tasks are not preemptable, the context of the tasks doesn't have to be stored. In this case, $Ereconf_i = Econf_i$.

Finally, the global energy is defined as

$$\begin{aligned} E_{fpga} &= Es_{ground} + Ea_{ground} \\ &+ \sum_{i=1}^{Nht} (Econf_i + Et_i) \\ &+ \sum_{i=1}^{Nht} \sum_{j=2}^{Ne_i - 1} (Ereconf_i * \beta_{i,j} + Et_i) \end{aligned} \quad (2)$$

With Nht the number of tasks to execute within the reconfigurable space, Ne_i the number of executions for the task $HTask_i$ and $\beta_{i,j}$ an integer value equal to 1 if the task $HTask_i$ must be reconfigured for the new execution of equal to 0 if the task $HTask_i$ is already configured and just need to be launched. The value $\beta_{i,j}$ depends on the execution order of the tasks, which is dynamically decided (on-line) by the operating system.

C. Consumption modeling / power models

The former Section IV-B presents the estimation process, which can be seen as a global consumption model of the complete application running on the system. The estimation process computes the total consumption for the complete application from the different power models of the system components. The development of those power models is presented in this section.

Our power models come directly from current measurements on the targeted electronic boards. Our modeling approach, the Functional Level Power Analysis (FLPA), was already discussed in former publications [43], [12]. It includes three steps : (i) a decomposition into functional blocks with a strong impact on the power consumption. A functional block gathers some functionalities of the hardware that are interdependent regarding the power consumption. It is important to target here a coarse granularity, in order to keep simple both the consumption models and the estimation process. Parameters with the biggest impact on the power are determined here (the frequency for instance, the cache miss rate for a cache consumption, or the number of instruction per cycle for a superscalar processor). (ii) a set of measurements to characterize the evolution of the consumption with the parameter values. (iii) the determination of the power-models under an analytical form (a mathematical equation) whenever possible; a table of values is used otherwise (a multi-entry look-up table : entries are the model inputs, output is the power or energy).

This modeling approach was proven fast and precise; it produces simple models, even for complex architectures, that can be used at high levels in the design flow.

Models were developed for processors with different architectures, from the simple RISC (ARM7, ARM9) to much more complex architectures (the super scalar VLIW DSP TI-C62, C64, and C67), and also for low-power processors (the TI-C55 and the Xscale) [44], [45]. Important phenomena are taken into account, like cache misses, pipeline stalls, and internal/external memory accesses. The average error, observed between estimations and physical consumption measurements, for a set of various algorithms (FIR filter, LMS filter, Discrete

Wavelet Transform (DWT), Fast Fourier Transform (FFT) 64 to 1024 points, Enhanced Full Rate (EFR) Vocoder for GSM, MPEG1 decoder, MJPEG chain, etc.) is lower than 5% at the assembly level, and lower than 10% from the C-code. Power models were also developed for reconfigurable circuits FPGA, memories, different peripherals and devices, and operating system services [14], [36]. In the case of the application presented in section V, the following power models have been developed : tick timer power model, scheduling power model, inter-process communication power model, Ethernet access power model, compact flash access power model, processor (PowerPC 405) power model, page fault power model.

Concerning FPGA, different power models may be used with different granularity. The choice of one granularity level actually depends on the information at disposal. The system level model is the simplest. At this level, the application code is not known yet. The designer needs to quickly evaluate the application against power constraints, energy constraints and/or thermal constraints. A fast estimation is necessary here, and a much larger error is acceptable. The parameters we can extract from the very high-level specification are the frequency F, the activity rate β, and the occupation ratio α of the targeted FPGA implementation.

The algorithmic level model involves the software specification of the future circuit or the IP component. The architectural resources are not known yet however: they may indeed depend on some configuration parameters of the IP. The parameters of this model are *algorithmic parameters*, in addition to the operating frequency which we rather regard as an *architectural parameter*. Algorithmic parameters describe some features of the algorithm that have a strong impact on its implementation power consumption. For example, we proposed a model for the FIR filter IP with the following input parameters: the filter order, its delay, and its operating frequency [23]. Intuitively, the place and route of the IP should have an influence on its power consumption. However, we observed that this influence was relatively small enough to be neglected in the modeling of such IPs.

It is remarkable that if an IP is not available for a hardware accelerator we need to use, a High-Level Synthesis (HLS) tool can give a good estimate of the amount of resources necessary to implement it, and given the targeted circuit, provide its occupation ratio and activity rate. With those two parameters and the frequency, a system model can be used then.

The architectural level model is the more complex. It takes into account the architecture that will be implemented on the physical target. If the architecture is known with the exact resources involved, a more precise estimation can be achieved. The model will be built from a library of models of the circuit resources. It will take in input the number of resources, the types of resources, the data types, the interconnections, and the operating frequency. Of course, this approach will be used whenever an algorithmic model has not been developed for the application IP.

Our modeling approach was also used to model operating system energy consumption on different platforms. In [14], we present the modeling of the Xillinx Virtex-II Pro XUP platform with a Linux 2.6 operating system. Different services are first modeled like scheduler/timer interrupt, inter-process communications (mqueues, pipes, shared memories, etc.), device accesses (Ethernet, compact flash, etc.). For example, the mqueue energy is here expressed as a set of relations between the processor frequency and the messages size. The operating systems energy overhead is then expressed as the sum of multiple contributions related to services activated during a run of the application. In [46], we present the modeling of the power and energy consumption of virtual memory management mechanisms. The virtual memory subsystem of a complete and recent Linux (patched for real-time) is studied, with its relation with the processors memory management resources (Memory Management Unit and Translation Look-aside Buffer). Our model allows to estimate the time and energy penalties for different page allocation strategies and different categories of page faults. Lately, we have presented in [47] the modeling of the operating system energy overhead on an OMAP3530 EVM board from TEXAS INSTRUMENTS. The influence of the scheduler policy is included there, especially in the context of multiple power domains (voltage and frequency scaling) for multi-task applications.

D. Power-aware design methodology

This subsection details our power-aware design methodology for heterogeneous MPSoC to cover several layers of the design. The objective is to offer for each step a power estimation tool in order to have a gradual refinement of the design space solution basing on the power or energy criteria. In order to cope with the design complexity, we focus specially on the functional and the transactional levels that offer different trade-offs between accuracy and estimation time.

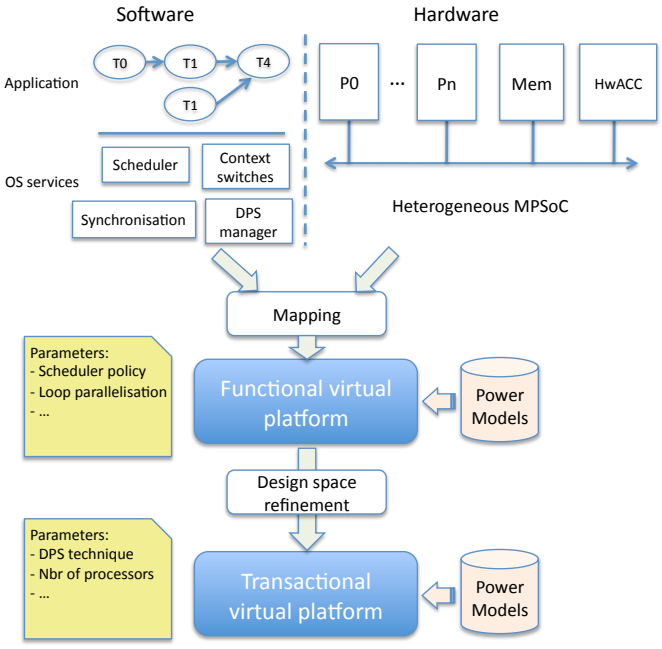

Fig. 6. Power-aware design methodology

1) Functional level estimation: As explained in the introduction, our approach is based on the use of coarse grain power models, built upon physical measurements, which give a good trade-off between accuracy and estimation time. This approach allows the modeling of even complex components in a reasonable time. From there, different abstraction levels of the specification may be used to refine the input of the models.

In the frame of the former European project SPICES [48], our energy estimation methodology and power models have been integrated in a Computer Aided Design (CAD) Consumption Analysis Toolbox (CAT) [49]. The aim of SPICES was to provide avionic system builders with reliable analysis tools for the design of critical embedded systems. The CAT toolbox combines a set of power estimation models with a system architecture model to provide system-level power consumption analysis. CAT has been used on our case study to compute the estimated power consumption. CAT runs on the Windows and Linux platforms and is deployed on the Eclipse Integrated Development Environment (IDE). The central part of CAT is a Domain Specific Language (DSL) that was defined to describe system architectures from a power analysis perspective. It also serves as a communication layer between the CAT application tiers to exchange the modeled system data. CAT can also be used in conjunction with the Open Source AADL Tool Environment (OSATE) [50], and the Toolkit in Open source for Critical Aeronautic Systems Design (TOPCASED) [51]. CAT may be downloaded with related documentation on [52].

2) Transactional level estimation: In order to offer a detailed power analysis by the means of a complete simulation of the application, we used a fast SystemC simulator at the transactional level. Our simulator consists of several hardware components which are instantiated from the SoCLib [2] library in order to build a prototype of the target system. We highlight that processors are described using Instruction Set Simulator (ISS) that sequentially executes the instructions and has no notion of concurrency of micro-architecture. Such technique is currently in use industrially and is expected to provide good performance accuracy. Today, heterogeneous MPSoC can be evaluated using simulation technique [53] at different abstraction levels. Fig 7 shows our developed system level power estimation tool that includes the *functional power estimator* and *fast SystemC simulator*. The functional power estimator evaluates the consumption of the target system with the help of the elaborated power models. It takes into account the architectural parameters (e.g. the frequency, the number of processors, the processor cache configuration, etc.) and the application mapping. It also requires the different activity values on which the power models rely. In order to collect accurately the needed activity values, the functional power estimator communicates with a fast SystemC simulator at a TLM level. Combination of the above two components described at different abstraction levels (functional and TLM) leads to a hybrid power estimation that gives a better trade-off between accuracy and speed.

[2]https://www.soclib.fr/

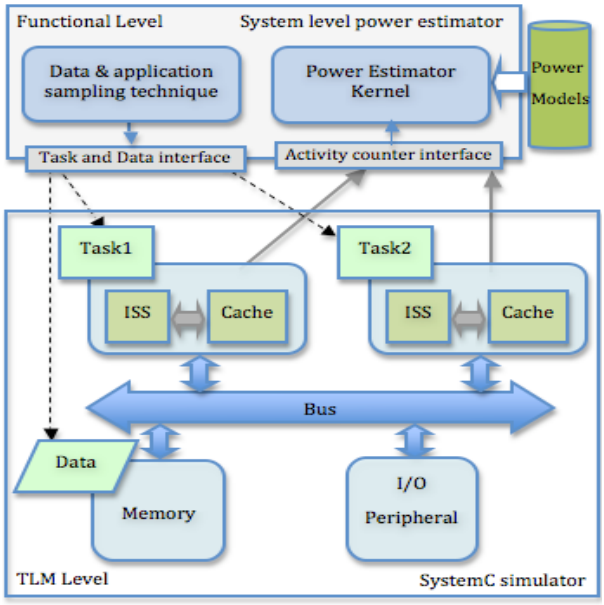

Fig. 7. Power estimator functioning

V. The Case Study

This section describes the usefulness and the effectiveness of our power estimation framework for a PowerPC 405-based SoC implemented into the Xilinx Virtex II Pro FPGA (XupV2Pro) platform. The Virtex II Pro FPGA contains two hardware PowerPC 405 processors that have a 16KB, 2-way set associative instruction and data caches. In addition, a large number of configurable logic blocks (CLB) are available for implementing hardware accelerators. Each processor has the access to the on-chip memory (BRAM) and the off-chip memory (SDRAM) via Processor Local Bus (PLB). We used the JPEG (Joint Photographic Experts Group) application as a benchmark. The JPEG application consists of 6 main tasks: acquisition of the input image, conversion RGB (Red, Green and Blue) to YUV (luminance, blue chrominance, and red chrominance components), Discrete Cosine Transform (DCT), Quantization, Huffman coding, and rebuild of the output image.

A. Power model elaboration

The power models presented below are the result of the consumption modeling of the targeted system. For every component identified as taking a large part of the platform overall consumption (it is the initial step of our modeling approach presented in section IV-A), the Functional Level Power Analysis is applied. As explained in section IV-C, the first step consists in dividing the component architecture into different functional blocks, and then to group the blocks that are simultaneously activated when a code or an application is running. Next comes the characterization of the component power consumption which is achieved by varying the parameters which were identified as having a strong impact on the power consumption (Frequency, Voltage, etc.). These variations are produced with some basic assembly programs. The power consumption is measured for every of

those programs, and a power model is derived from the set of measures : either an analytical model, i.e. a mathematical function, is computed by regression, or the set of consumption values is stored in a multiple entries table.

1) Processor power model: The FLPA methodology allows to identify the architectural parameters and system activities that have the strongest influence on the processor power consumption. Table I shows the power consumption laws for the PowerPC 405-based SoC implemented into the XUP FPGA. These models predict consumption of the kernel and the I/Os parts separately, since they are powered by distinct power supplies (2.5V for the kernel and 1.5V for the I/Os). The input parameters of the power models are the processor frequency ($F_{processor}$), the bus frequency (F_{bus}), and the cache miss rate (γ). The frequencies of the processor and bus are fixed by the system designer while the cache miss rate is considered as an activity of the processor which could be extracted from the simulation environment. Either internal BRAM or external SDRAM may be used with the PowerPC, which gives two different set of models.

TABLE I
CONSUMPTION LAWS FOR THE TARGET PLATFORM

Power models for PowerPC 405		
BRAM	1.5V	P(mW)=0.40 $F_{processor}$ + 3.24 F_{bus} + 74
	2.5V	P(mW) = 5.37 F_{bus} + 1588
SDRAM	1.5V	P(mW) = 0.38 $F_{processor}$ + 3.45 F_{bus} + 79
	2.5V	P(mW) = 4.1 γ + 6.3 F_{bus} + 1599

2) FPGA power model: A power model has been built for the reconfigurable part of the FPGA component on the XUP board. This model has been built with the coarser granularity, hence at the system level, as described in section IV-B. This model does not come as a multi-linear equation of the frequency F, switching activity β and area utilization α. For this reason, a 3 entries table of consumption values is used. The power is estimated by interpolation of these 3 input parameters. Fig 8 illustrates the variation of the FPGA power consumption according to area utilization and the switching activity with an operating frequency set to 100 MHz.

3) OS power model: In order to estimate the energy overhead due to the operating system while running our application on the Xilinx Virtex II board, we have developed power and energy models for the following Linux 2.6 services: timer interrupt, Inter Process Communication (IPC), Ethernet and Compact Flash accesses, and virtual memory. A detailed description of those models may be found in [14], [36] and [46]. We will here only sketch their major features.

Tick timer model: Every timer tick, the `scheduler_tick()` function is called to evaluate the runnable processes. To estimate the energy overhead incurred by timer interrupts, we have executed several computing intensive programs with and without OS. Roughly the same power overhead was observed for all programs, which appeared to be a small proportion of the system's global consumption. The energy overhead is obtained buy multiplying the power overhead with the program execution time. The parameters of the power model are the CPU frequency and the tick timer frequency.

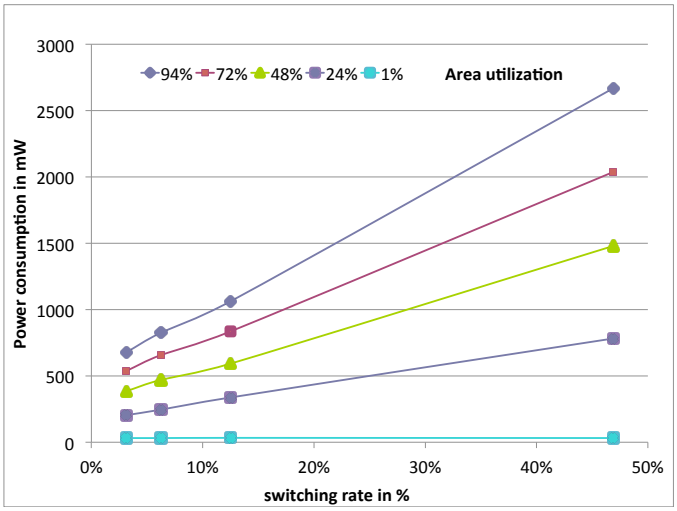

Fig. 8. FPGA power consumption with 100 MHz frequency

IPC model: Inter-process communications allow threads in one process to share information with threads in other processes, even on different hardware platforms: the OS explicitly copies information from a sending process address space into a distinct receiving process address space. We have developed models for the following IPC mechanisms: pipes, mqueue, shared memory and sockets (for remote IPC). To model IPC power consumption, we have executed programs that repeatedly use an IPC mechanism, with different values for parameters such as the amount of data sent and received, the OS tick timer frequency and the processor frequency. Finally, the parameters that impact on the energy consumption are the messages size and the CPU frequency, plus the protocol chosen in case of a socket (Ethernet access).

Compact Flash accesses: The Compact Flash (CF) is seen here as the standard mass storage device in the system. Two different Linux system calls were modeled. The first model concerns buffered I/O which are the default Linux I/O operations. When the I/O is buffered the compact flash does direct memory access from/to the kernel cache, and not from/to the user space source/destination buffer allocated by the user application. The second model concerns self-caching I/O. In this case, the application will keep its own I/O cache in user space (often in shared memory), so it does not need any additional lower level system cache. Again, the amount of data transferred and the CPU frequency were kept as inputs to the models.

Virtual memory: The power and energy consumption modeling measurements have been conducted on the XUP Virtex-II pro development board with a 256MB SDRAM and a compact Flash for the root file system. The operating system analysed is the Xilinx Open Source Linux which is based on the 2.6.29 Linux kernel, to which we applied the RT-Preempt patch to make it fully preemptable. The memory management unit (MMU) of the processor performs address translation and protection functions. The translation look-aside buffer (TLB) is used by the MMU for address translation. Each valid entry contains the virtual page number and its translation into a

physical page number. If a virtual address does not match an entry in the TLB, the CPU raises a TLB miss exception. The operating system provides the physical address from its page global directory (PGD) and updates the TLB. Whenever the address is not in the PGD, a minor or major page fault occurs. Our measures show that the energy cost of TLB misses is negligible in front of the cost of page faults. The energy overhead model for page fault is finally a function of the number of TLB misses and the processor frequency.

B. System level power estimation

1) Monoprocessor architecture: In the next step, the developed power models are integrated into system level design tools as explained in Section IV. As a first scenario, we used the JPEG application with a PowerPC monoprocessor based architecture. To do so, we developed a system level prototype of the PowerPC based SoC, with the help of SystemC models including ISS for the target processor, with the cache parameters and bus latencies set to emulate the real platform behaviour. A set of counters are injected into the simulator to determine the values of different miss rates: read data miss, write data miss and read instruction miss of the corresponding caches. The fast SystemC simulator takes the full JPEG application with the real standard frame size of 256×256 pixels and simulates it entirely in order to collect the required activities.

Table II shows the detailed activities of each task in the application, as a result of the SystemC simulation. From these results several remarks can be drawn. First, we can notice that instruction cache miss rates and read data miss rates are very low when compared with write data miss rates. This is due to the fact that the task kernel is small (a small number of instructions) and that the volume of data accessed is also small compared to the cache size (16 KB). With the new submicron technologies however, static power consumption cannot be neglected. For this reason, some software processors, such as the Microblaze, come with reconfigurable cache sizes to fit the application requirements. Secondly, we observe that the data write miss rates have a high impact on the total power consumption. This is due to the algorithm structure which does not favour the reuse of data output arrays, and to the usage of write-through cache policy. As we can see, the statistics collected in Table II can help in tuning the application structure for a better optimization of the system power consumption.

In the next step, using the obtained results and the power models shown in Table I, we estimated the total power consumption of each task. Fig. 9 illustrates the results and shows the comparison between the proposed hybrid estimator, the SoftExplorer tool introduced in Section II, and the real board measurements. Negative and positive errors correspond respectively to under and over estimation of the consumption. The average error is the mean value of the absolute values of those errors. First, the hybrid power estimator has a negligible average error equal to 0.02% which offers better accuracy than SoftExplorer with its average error of (3.32%). Indeed, the activities captured in the SystemC simulator are more accurate than the static analysis or rapid profiling of the code

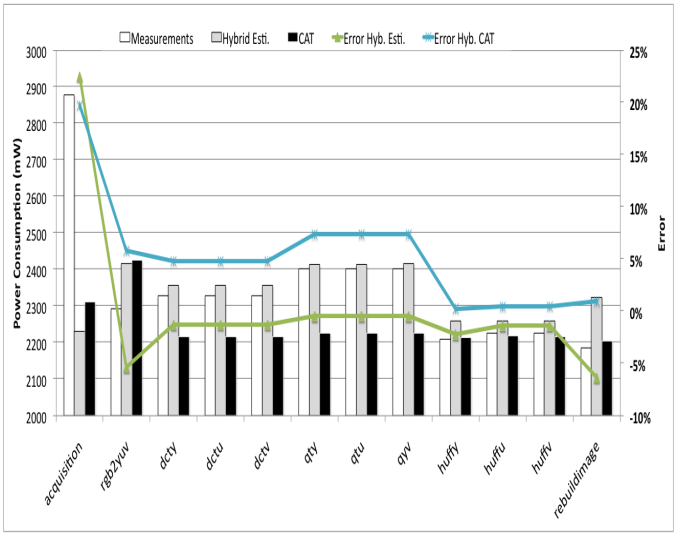

Fig. 9. Power Estimation and Comparison with SoftExplorer

performed by SoftExplorer. The maximum of error is reported for the acquisition and rebuild tasks, respectively 22.5% and 6.36%. In fact, these two tasks use operating system calls to read and write from/to files. Those system calls are however only executed by the system level simulator by means of a virtual file system, which does not reflect precisely the real operating system's behaviour. Finally, without considering the acquisition and rebuild tasks, the hybrid estimator gives an average error of 1.32% while SoftExplorer's is 3.17%.

2) Homogeneous multiprocessor architecture: The second case study involves an homogeneous architecture with identical processors to run the JPEG application. To evaluate the impact of the number of processors on the execution time and total energy consumption, we executed the JPEG on systems with 1 to 8 processors. The PowerPC frequency was set to 300 MHz and the PLB frequency to 100 MHz. All the processors execute the same workload but on different image macroblocks. Figure 10 reports the execution time in ms and the total energy consumption in mJ.

Given these results, we see that adding processors to the system decreases the execution time, which improves the system performance. This variation is not linear because the processors share resources, which generates conflicts at some times, and reduces the speedup as waiting cycles are added to the processors execution. In terms of energy consumption, we observe that until a certain number of processors, the total system energy consumption decreases as the execution time is reduced. Adding more processors increases the power consumption, however with not the same slope as the time decreases. As we are using only the ASIC PowerPC processors integrated in the Xilinx Virtex II FPGA and the processors are executing the same workload in parallel, the static power is not influencing significantly the total consumption. But increasing the number of processors over a certain limit tends to be ineffective, as it just adds new conflicts at the PLB level, leading to more waiting cycles.

3) Hardware accelerators architecture: In this part, we emphasize the benefit of our estimation methodology in the

TABLE II
APPLICATION MISS RATES

Program	Instruction miss rate (%)	Read Miss rate (%)	Write Miss Rate (%)	Total Miss Rate (%)
acquisition	0.003386	3.56	31.73	0.02
rgb2yuv	0.001128	3.03	99.91	5.64
dct y	0.002283	4.49	40.72	3.88
dct u	0.000315	4.49	40.72	3.88
dct v	0.000314	4.49	40.72	3.88
qt y	0.000812	2.06	99.88	5.58
qt u	0.000406	2.06	99.93	5.58
qy v	0.000406	2.06	99.94	5.58
huff y	0.004375	4.58	20.11	0.85
huff u	0.000515	4.57	19.8	0.84
huff v	0.000643	4.56	19.61	0.84
rebuild image	0.298380	3.05	25.19	2.87
complete application	0.000012	0.029	0.09	0.012

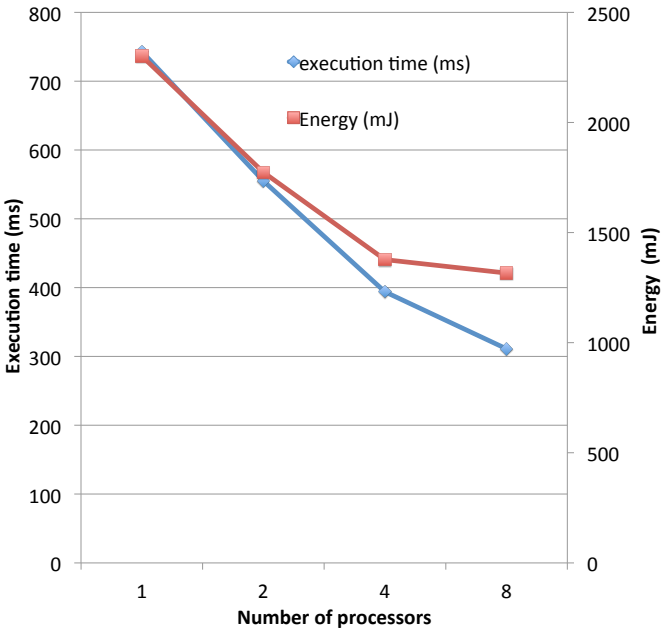

Fig. 10. Execution time and energy variation in terms of the number of processors

context of heterogeneous architecture. In general, the choice of a hardware accelerator is driven principally by the performance requirements of the application and the processor usage of each task. For the JPEG application, the DCT task is the most time consuming task. Thus, it is selected to be implemented as a hardware accelerator. Various trade-offs can be done between the amount of consumed hardware resources (i.e.: the area utilization), the execution time, and the power consumption. The DCT task is highly regular and has large repetition spaces in its multiple hierarchical levels. Such large repetition spaces allow us to fully exploit the existing partitioning in VHDL (i.e. hardware-software and parallel-sequential hardware). System-level architecture synthesis tool such as GAUT [54] or ROCCC [55] can be used to obtain several implementations of the hardware accelerator with different trade-offs between the execution time or the number of resources [56]. Certainly, more accurate estimation of these parameters can be obtained at lower levels using the commercial RTL tools but at the price of significant evaluation time. We selected a configuration which is about 200 times faster than a software execution with a PowerPC processor running at 100 MHz. A hardware synthesis of this configuration occupies 18% of the XupV2Pro. According to the FPGA power model, the power consumption of the chosen DCT hardware accelerator is around 300mW offering 40% of power saving compared to the software execution.

4) Extrapolation for complete MPSoC architecture: The above developed power models will be used in the frame of system level estimation of heterogeneous MPSoC that may contain several processors and hardware accelerators. This approach is mandatory in the design flow for two reasons, even if the corresponding estimates are less accurate than those provided by real board measurements. First, system level estimation can be achieved with acceptable accuracy 10-1000x faster than the physical level taking into account the required design time. Second, it allows exploring architectures that cannot be implemented due to the hardware resource limitation or the unavailability of the target component. For instance, we cannot exceed two PowerPC based architecture using our XupV2Pro platform. Thus, it is important to have a scalable approach to address the complex system power/energy estimation issue. The equation 3 will be considered for the total system energy estimation. We find there the sum of the energy consumptions of every software tasks with the related operating system energy overhead (see equation 1) and the sum of the energy consumptions of every hardware tasks (see equation 2). The consumption of the synchronization part required to access the shared resources is included in E_{OS_τ}.

$$E_{total} = \sum (E_\tau + E_{OS_\tau}) + \sum E_{fpga} \qquad (3)$$

In our XupV2Pro platform, a software synchronization between several tasks running on different processors or hardware accelerators will call for a hardware mutex through an OS service. Several experiments have been conducted to evaluate the additional power cost of this hardware component. This study includes three parameters which are the number of masters, and the processor and bus frequencies. Fig 11 shows

that the mutex power consumption depends mainly on the PLB frequency.

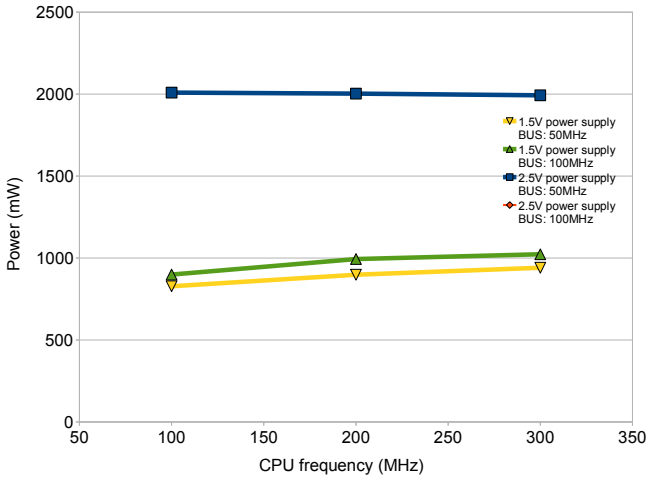

Fig. 11. Mutex power consumption

VI. Conclusion

This paper presents a complete estimation and optimization framework for power aware design of heterogeneous MPSoC. Indeed, energy and power constraints are considered as major challenge when the system runs on batteries. Thus, designers must take these constraints into account earlier in the design steps. Furthermore, the complexity of the systems leads to develop high level methods and tools to help the designer to make good decisions at each design step.

The goal of the Open-People project consists in proposing a global framework that helps designer to obtain earlier power and energy consumption estimation. First, a power modeling methodology has been defined to address the global system consumption that includes processors, memories, reconfigurable circuits, operating system services, etc. Secondly, these power models are integrated in a global power component library in the context of multi-level design space exploration: to refine power and energy estimations, they are used in conjunction with simulation tools at different abstraction levels [57], [58].

This paper presents the framework and illustrates a specific case study for the JPEG application implemented in the XUP Virtex II Pro circuit. Our approach was used to define the power and energy consumption models for every hardware and software component in our application. A global estimation methodology was proposed, to finally provide the system's global power and energy estimation.

With those different estimations, the designer can explore several implementation choices (monoprocessor, homogeneous and heterogeneous multiprocessor). Thanks to fast SystemC simulations, it is also possible to quickly evaluate the application implementation on new custom hardware architectures.

The future works for this project will focus on more complex heterogeneous platforms, for example OMAP 3530 which corresponds to the actual MPSoC platform used by the industrials. Furthermore, in order to obtain more accurate power estimations, some power model refinements must be realized. This is the case for the data exchanges between hardware and software tasks respectively executed on hardware resource and on processor which are currently estimated at high level of abstraction.

Acknowledgment

The Open-PEOPLE project is funded by the French Agence Nationale de la Recherche (ANR-08-SEGI-013). We would like to gratefully thank all members of Open-PEOPLE, who actively participate to the development of this framework and contributed to the present work.

References

[1] "SPICE manual," University of Berkeley (USA), URL: http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/.

[2] Philips Electronic Design and Tools Group, "DIESEL User Manual," Philips Research, Tech. Rep., Jun. 2001.

[3] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proceedings of the 27th annual international symposium on Computer architecture*, 2000, pp. 83–94.

[4] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. Irwin, "The Design and Use of SimplePower: A Cycle Accurate Energy Estimation Tool," in *Design Automation Conf*, Jun. 2000.

[5] G. Beltrame, L. Fossati, and D. Sciuto, "ReSP: A Nonintrusive Transaction-Level Reflective MPSoC Simulation Platform for Design Space Exploration," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 12, pp. 1857–1869, Dec. 2009.

[6] "The SoCLib project: An open modelling and simulation platform for system on chip design," 2009, http://soclib.lip6.fr/.

[7] Open SystemC Initiative, "Systemc," 2008, world Wide Web document, URL: http://www.systemc.org/.

[8] N.Dhanwada, I. Lin, and V.Narayanan, "A power estimation methodology for systemc transaction level models," in *International conference on Hardware/software codesign and system synthesis*, 2005.

[9] I. Lee, H. Kim, P. Yang, S. Yoo, E. Chung, K.Choi, J.Kong, and S.Eo, "Powervip: Soc power estimation framework at transaction level," in *Proc. ASP-DAC*, 2006.

[10] N. Dhanwada, R. A. Bergamaschi, W. W. Dungan, I. Nair, P. Gramann, W. E. Dougherty, and I.-C. Lin, "Transaction-level modeling for architectural and power analysis of powerpc and coreconnect-based systems."

[11] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," in *Transactions on VLSI Systems*, 1994.

[12] J. Laurent, N. Julien, E. Senn, and E. Martin, "Functional Level Power Analysis: An efficient approach for modeling the power consumption of complex processors," in *Proc. Design Automation and Test in Europe DATE*, Paris, France, march 2004.

[13] 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 *Fourteenth International Workshop on Power and Timing Modeling (PATMOS 2004)*, Santorini, Greece, September 2004, pp. 15–17.

[14] S. Dhouib, E. Senn, J.-P. Diguet, D. Blouin, and J. Laurent, "Energy and power consumption estimation for embedded applications and operating systems," *Journal of Low Power Electronics (JOLPE)*, vol. 5, no. 3, 2009.

[15] A. Garcia, W. Burleson, and J. Danger, "Power modelling in field programmable gate arrays (FPGA)," *Lecture notes in computer science*, pp. 396–404, 1999.

[16] ——, "Power modelling in field programmable gate arrays (FPGA)," in *Field Programmable Logic and Applications*. Springer, 2004, pp. 396–404.

[17] A. D. G. G, J. L. Danger, and W. P. Burleson, "Reducing the power consumption in fpgas with keeping a high performance level," *VLSI, IEEE Computer Society Workshop on*, vol. 0, p. 47, 2000.

[18] L. Shang and N. Jha, "High-level power modeling of CPLDs and FPGAs," in *Computer Design, 2001. ICCD 2001. Proceedings. 2001 International Conference on*. IEEE, 2002, pp. 46–51.

[19] N. Abdelli, A. Fouilliart, N. Mien, and E. Senn, "High-Level Power Estimation of FPGA," in *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*. IEEE, 2007, pp. 925–930.

[20] P. Yuh, C. Yang, C. Li, and C. Lin, "Leakage-aware task scheduling for partially dynamically reconfigurable FPGAs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 4, pp. 1–26, 2009.

[21] D. Elleouet, N. Julien, and D. Houzet, "A high level soc power estimation based on ip modeling," in *Proceedings of the 13th Reconfigurable Architectures Workshop*, Rhodes Island, Greece, April 2006.

[22] D. Elleouet, Y. Savary, and N. Julien, "A fpga power aware design flow," in *Power and Timing Modeling, Optimization and Simulation, PATMOS*, Montpellier, France, September 2006.

[23] E. Senn, N. Julien, N. Abdelli, D. Elleouet, and Y. Savary, "Building and using system, algorithmic, and architectural power and energy models in the fpga design-flow," in *Intl. Conf. on Reconfigurable Communication-centric SoCs 2006*, Montpellier, France, July 2006.

[24] A. Acquaviva, L. Benini, and B. Ricco, "Energy characterization of embedded real-time operating systems," in *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP01)*, September 2001.

[25] K. Baynes, C. Collins, E. Fiterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, and B. Jacob, "The performance and energy consumption of three embedded real-time operating systems," in *Proceedings of CASES01*, Atlanta, Georgia, USA, November 2001.

[26] R. P. Dick, G. Lakshminarayana, A. Raghunathan, and N. K. Jha, "Power analysis of embedded operating systems," in *Proceedings of the IEEE 37th Design Automation Conference (DAC-00)*, NY, June 2000, pp. 312–315.

[27] N. Aaraj, A. Raghunathan, S. Ravi, and N. Jha, "Energy and execution time analysis of a software-based trusted platform module," in *Proc. of the IEEE Conf. on Design, Automation and Test in Europe*, 2007, pp. 1128–1133.

[28] A. Vahdat, A. Lebeck, and C. Ellis, "Every joule is precious: A case for revisiting operating system design for energy efficiency," in *Proceedings of the Ninth ACM SIGOPS European Workshop*, September 2000.

[29] K. Baynes, C. Collins, E. Fiterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, and B. Jacob, "The performance and energy consumption of embedded real-time operating systems," *IEEE Transactions on Computers*, vol. 11, no. 52, pp. 1454–1469, November 2003.

[30] X. Zhao, Y. Guo, H. Wang, and X. Chen, "Fine-grained energy estimation and optimization of embedded operating systems," in *International Conference on Embedded Software and Systems Symposia, ICESS Symposia '08*, July 2008, pp. 90–95.

[31] Y. Chen, *The Analysis and Practice on Open Source Embedded System Software–Based on SkyEye and ARM Developing Platform*. Beihang University Press, 2004.

[32] F. Nicolas, F. Antoine, and F. Paul, "esimu: a fast and accurate energy consumption simulator for real embedded system," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2007*, June 2007, pp. 1–6.

[33] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2003, pp. 160–171.

[34] A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li, and L. K. John, "Using complete machine simulation for software power estimation: The softwatt approach," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2002.

[35] T. Tan, A. Raghunathan, and N. Jha, "Energy macromodelling of embedded operating systems," *ACM Transactions on Embedded Computing Systems*, vol. 4, no. 1, pp. 231–254, February 2005.

[36] J. D. S. Douhib, E. Senn, "Model driven high-level power estimation of embedded operating systems communication and synchronization services," in *Proceedings of the 6th IEEE International Conference on Embedded Software and Systems*, China, May 25-27 2009.

[37] Texas Instruments, "OMAP3530/25 Applications Processor," http://focus.ti.com/lit/ds/sprs507f/sprs507f.pdf, 2009.

[38] Xilinx, "Virtex-5 FPGA User Guide," http://www.xilinx.com.

[39] R. Urunuela, A. Dplanche, and Y. Trinquet, "Storm a simulation tool for real-time multiprocessor scheduling evaluation," *gdr soc sip*, vol. 2009, no. 1, pp. 1–2. [Online]. Available: http://hal.archives-ouvertes.fr/docs/00/49/57/47/PDF/gdr-09.pdf

[40] A. Schranzhofer, J.-J. Chen, and L. Thiele, "Dynamic power-aware mapping of applications onto heterogeneous mpsoc platforms," *Industrial Informatics, IEEE Transactions on*, vol. 6, no. 4, pp. 692 –707, nov. 2010.

[41] R. Bonamy, D. Chillet, O. Sentieys, and S. Bilavarn, "Towards a power and energy efficient use of partial dynamic reconfiguration," in *Proc. 6th Int Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC) Workshop*, Montpellier, France, Juin 2011, pp. 1–4.

[42] R.Bonamy, D.Chillet, S.Bilavarn, and O.Sentieys, "Parallelism level impact on energy consumption in reconfigurable devices," in *HEART (International Workshop on Highly-Efficient Accelerators and Reconfigurable Technologies)*, Londres, Mai 2011.

[43] N. Julien, J. Laurent, E. Senn, and E. Martin, "Power consumption modeling of the TI C6201 and characterization of its architectural complexity," *IEEE Micro, Special Issue on Power- and Complexity-Aware Design*, Sep./Oct. 2003.

[44] N. Julien, S. Gailhard, and E. Martin, "Low power synthesis methodology with data format optimization applied on a DWT," *Journal of VLSI Signal Processing*, pp. 195–211, 2003.

[45] E. Senn, J. Laurent, N. Julien, and E. Martin, "SoftExplorer: Estimating and optimizing the power and energy consumption of a C program for DSP applications," *the EURASIP Journal on Applied Signal Processing, Special Issue on DSP-Enabled Radio*, vol. 2005, no. 16, September 2005.

[46] M. Lano and E. Senn, "Energy modeling of the virtual memory subsystem for real-time embedded systems," in *Conference on Design and Architectures for Signal and Image Processing*, Edinburgh, Scotland, October 2010.

[47] B. Ouni, C. Belleudy, S. Bilavarn, and E. Senn, "Embedded operating systems energy overhead," in *Conference on Design & Architectures for Signal & Image Processing*, Tampere, Finland, November 2-4 2011.

[48] The SPICES ITEA Project Website. [Online]. Available: http://www.spices-itea.org/

[49] D. Blouin and E. Senn, "An extensible system-level power consumption analysis toolbox for model-driven design," in *Proceedings of the 8th IEEE International NEWCAS Conference*, Montreal, Canada, June 2010.

[50] The Open Source AADL Tool Environment (OSATE). http://aadl.sei.cmu.edu/aadlinfosite/OpenSourceAADLToolEnvironment.html.

[51] TOPCASED Home. [Online]. Available: http://topcased-mm.gforge.enseeiht.fr/

[52] CAT - Consumption Analysis Toolbox. http://sourceforge.net/apps/trac/lab-sticc/.

[53] R. Ben Atitallah, S. Niar, S. Meftali, and J.-L. Dekeyser, "An MPSoC performance estimation framework using transaction level modeling," in *The 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Daegu, Korea, Aug. 2007.

[54] Chavet Cyrille, Andriamisaina Caaliph, Coussy Philippe, Casseau Emmanuel, Juin Emmanuel, Urard Pascal and Martin Eric, "A design flow dedicated to multi-mode architectures for DSP applications," in *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design (ICCAD '07)*, San Jose, California, 2007.

[55] N. W. Villarreal Jason, Park Adrian and H. Robert, "Designing Modular Hardware Accelerators in C with ROCCC 2.0," in *Proceedings of the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'10)*, Washington, DC, USA, May 2010.

[56] A. Gamatié, S. Le Beux, É. Piel, R. Ben Atitallah, A. Etien, P. Marquet, and J.-L. Dekeyser, "A Model Driven Design Framework for Massively Parallel Embedded Systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 10, no. 4, 2011.

[57] D.Blouin, D.Chillet, E.Senn, S.Bilavarn, R.Bonamy, and C.Samoyeau, "Aadl extension to model classical fpga and fpga embedded within a soc," *International Journal of Reconfigurable Computing*, vol. 2011, p. 15 pages, 2011.

[58] D.Blouin, E.Senn, R.Bonamy, D.Chillet, S.Bilavarn, and C.Samoyeau, "Fpga modeling for soc design exploration," in *HEART (International Workshop on Highly-Efficient Accelerators and Reconfigurable Technologies)*, Londres, Mai 2011.

Rabie Ben Atitallah is currently an Associate Professor in Computer Science at the University of Valenciennes and member of LAMIH laboratory within the DIM (Decision, Interaction, and Mobility) team. He is also an associated member of DaRT project at the INRIA Lille-Nord Europe research institute. He is an IEEE member and a member of High Performance and Embedded Architecture and Compilation (HiPEAC) European Network of Excellence. Previously, he received his PhD degree in Computer Science from the University of Lille1 in March 2008. Between March 2008 and August 2009, he had a post-doctoral position at INRIA Lille-Nord Europe and the University of Valenciennes. His research interests include Embedded system design, MultiProcessor System-on-Chip (MPSoC), Low power-aware design, Virtual prototyping, Simulation, and Dynamic reconfigurable computing.

Eric Senn received the B.S. degree in Electrical Engineering from the University of Paris VI in 1991. He was student of the Ecole Normale Supérieure de Cachan from 1991 to 1992 and he succeed the Agrgation of Electrical Engineering in 1992. In 1993, he received the M.S. Degree in Electronics and Computer Sciences from the University of Paris XI. He was Professor of the French Ministry of Defense in the GIP (Geography Image and Perception) Laboratory for the DGA (Déléguation Générale de l'Armement) from 1995 to 1999. He received his Ph.D. degree in Electronics from the University of Paris XI in 1998. He is currently an Associate Professor in the University of South Brittany (UBS-France) and member of the Lab-STICC since 1999. He received the HDR (Habilitation a Diriger des Recherches) Degree in 2008 from the UBS. His current works include research on high-level methods and tools for embedded systems design (encompassing hardware and software architectures, as well as real time operating systems), power and energy modeling, analysis and optimization, and Model Driven Engineering (Meta-models definition, and model transformations).

Daniel Chillet is member of the Cairn Team, which is an Inria Team located between Lannion and Rennes in France. Daniel CHILLET received the Engineering degree and the M.S. degree in electronics and signal processing engineering from University of Rennes 1, respectively, in 1992 and in 1994, the Ph.D. degree in signal processing and telecommunications from the University of Rennes 1 in 1997, and the habilitation to supervise PhD in 2010. He is currently an Associate Professor of electrical engineering at the Enssat, engineering school of University of Rennes 1. Since 2010, he is the Head of the Electronics Engineering department of Enssat. His research interests include memory hierarchy, reconfigurable resources, real-time systems, and middleware. All these topics are studied in the context of MPSoC design for embedded systems. Low power design based on reconfigurable systems is one important topic and spatio-temporal scheduling, memory organization and operating system services have been previously addressed on several projects.

Mickael Lanoe received the graduated engineer degree in Electrical Engineering from Ecole Supérieure dElectricité of Rennes in 2001. He was a software architect at Rennes until 2009 after which he joined the Lab-STICC as a research engineer. His research interests are methods and tool for consumption estimation and optimization of embedded systems.

Dominique Blouin received a B.Sc. in physics from University of Sherbrooke (Sherbrooke, CANADA) in 1989, and a M.Sc. in physics (astrophysics) from the University of British Columbia (Vancouver, CANADA) in 1994. He was a software architect at Cassiopae (www.cassiopae.com) until 2008 when he joined the Lab-STICC at University of Bretagne-Sud as a research engineer. His research interests are Model Based Engineering, Requirements Engineering, Computer Aided Design tools, model transformations and Domain-Specific Languages (DSL) for embedded systems.