PETS: Power and Energy Estimation Tool at System-Level

Santhosh-Kumar RETHINAGIRI*, Oscar PALOMAR*, Osman UNSAL*, Adrian CRISTAL*, Rabie BEN-ATITALLAH† and Smail NIAR†

*Barcelona Supercomputing Center
Barcelona, Spain
†LAMIH, Université de Valenciennes
Valenciennes, France

Abstract—In this paper, we introduce PETS, a simulation based tool to estimate, analyse and optimize power/energy consumption of an application running on complex state-of-the-art heterogeneous embedded processor based platforms. This tool is integrated with power and energy models in order to support comprehensive design space exploration for low power multicore and heterogeneous multiprocessor platforms such as OMAP, CARMA, Zyq 7000 and Virtex II Pro. Moreover, PETS is equipped with power optimization techniques such as dynamic slack reduction and work load balancing. The development of PETS involves two steps. First step: power model generation. For the power model development, functional-level parameters are used to set up generic power models for the different components of the system. So far, seven power models have been developed for different architectures, starting from the simple low power architecture ARM9 to the very complex DSP TI C64x. Second step: a simulation based virtual platform framework is developed using SystemC IP's and JIT/ISS compilers to accurately grab the activities to estimate power.

The accuracy of our proposed tool is evaluated by using a variety of industrial benchmarks. Estimated power and energy values are compared to real board measurements. The power estimation results are less than 4% of error for single core processor, 4.6% for dual-core processor, 5% for quad-core, 6.8% multi-processor based system and effective optimisation of power/energy for the applications

I. INTRODUCTION

Evolution of embedded applications are critical and complex, involving more computing resources therefore indirectly implying the increase of power of the underlying hardware. The follow-through is two fold in power/thermal dissipation due to increase in complexity of the hardware and also decreases the lifetime of battery operated devices (iWatch and google glass). Facing this issue, power and energy constraints are being considered as first order critical pre-design metrics in the embedded community. Due to this issue, designers should calculate and optimize the power dissipation of the embedded system at the earliest in the design flow to reduce the time-to-market cost. In the last decade, various researches have been dedicated to propose power estimation techniques based on different abstraction levels, starting from electrical-level to system-level. Today, system-level power estimation is considered a vital premise to cope with the critical design constraints. However, the development of tools for power estimation at the system-level is in the face of extremely challenging requirements such as the seamless power-aware design methodology, accurate and fast system power modeling approach, and efficient power and energy based Design Space Exploration (DSE). Recently, International Technology Roadmap for Semiconductors (ITRS)\(^1\) predicted that power and energy issues are expected to get worse as we move to the next technology nodes. Similar to ITRS, HiPEAC\(^2\), an European embedded consortium also released their Roadmap-2013 stating that increasing the number of components on a chip and decreasing energy scaling leads to the phenomenon called "dark silicon". This effect leads to propose energy efficient systems with the right combination of different components.

In current practices, power estimation/optimization are done using low level CAD tools, which is unfortunately not applicable for the ever growing complexity of processors supporting complex multimedia applications. This challenge is addressed by several frameworks through the development of Electronic System-Level (ESL) tools. The objective is to develop virtual prototypes which are fully functional software models of the physical hardware system and to unify them with the application design and to offer a rapid system-level designing. Based on the design step and the requirements like timing accuracy and estimation speed, designers could select an appropriate abstraction level to model the software simulating the system. Unfortunately, most of existing system-level simulation based tools don’t consider the power metric for a given abstraction level.

System power modeling, estimation and optimization:
The power modeling at the system-level is centered around two connected aspects: granularity of the power model and characterization of the main activity. First aspect of the power model development depends on the relevant activities granularity. It starts from the fine grain level such as the transistor or gate switching and expands to the coarse grain level like the hardware functional events. In general for system-level designer, fine-grain power estimation need a lot modeling efforts due to the involvement of low level implementation details and technological parameters. Whereas, coarse-grain power models are easy to develop and it depends on least number micro-architectural activities but accuracy decreases as the complexity of system increases. The second aspect involves the characterization of the activities, which requires a huge number of experimental measurements and thus significant time to

\(^{1}\)http://www.itrs.net/Links/2012ITRS/Home2012.htm
\(^{2}\)http://www.hipeac.net/system/files/hipeac_roadmap1_0.pdf
extract the power model. The above-described aspects yield to the definition of the power model that can be represented by a set of analytical functions or a table of consumption values. The selected power model granularity depends on the target abstraction level and the user requirements in terms of estimation accuracy and speed. In this tool, we also address the issue of power/energy optimization of an application based on two well known techniques: Dynamic Power Management (DPM) that switches-off the power supply of a part of the circuit and the Dynamic Voltage and Frequency Scaling (DVFS) that tunes the processor clock speed and its corresponding voltage according to the workload (actual or expected) or the battery charge. There are many implementation of these approaches in the literature [5]. Most of these approaches are not validated with real measurements in terms of accuracy and moreover they don’t rely on realistic power models.

In the power estimation/optimization process, the developed power models will interact with system-level platform to grab the strict relevant data depending on the design step. The main challenge is to define a generic power modeling approach that can cover the different abstraction levels, which can guarantee the coherence of the power estimation/optimisation strategy. To answer the above described challenges, we propose **PETS**: Power and Energy estimation Tool at the System-level based on simulation of complex embedded platforms. Our tool focuses on the functional and the transactional levels to deal with the design complexity and the broadness of the architectural solution space. Second, Functional Level Power Analysis (FLPA) is used to elaborate different power models that are plugged afterwards with our tools to evaluate the total consumption of the system. Third, a runtime optimization technique is developed and tested in order to reduce energy and power consumption of the system.

The rest of this paper is organized as follows. After reviewing the related works in Section 2, we elaborate the tool flow in Section 3. In Section 4, we describe PETS power estimation methodology. Section 5 and Section 6 give the detailed explanation about the power modeling methodology and optimization technique used in the this tool respectively. In Section 7, we present the power and energy estimation/optimization results, also validating with the state of the art tools. We conclude in Section 8.

### II. Literature Review

In the last decade, various research efforts have been employed to estimate the processor power dissipation at different abstraction levels in the design flow. To reduce the simulation time, several studies have proposed evaluating system power consumption at higher abstraction levels. These tools use a micro-architectural simulator to evaluate system performance and an analytic power model to estimate consumption for each platform component. Watch [3] and SimplePower [19] are examples of tools available at this level. In general, these tools rely on Cycle-Accurate (CA) simulation technique. Usually, to move from the RTL to the CA level, hardware implementation details are hidden from the processing part of the system, while preserving system behavior at the clock cycle level. The power consumption of the main internal units is estimated using power macro-models, produced from lower-level characterizations. The contributions of the internal unit activities are calculated and added together during the execution of the program on the cycle-accurate micro-architectural simulator. Research works such as in [15] present CA simulation environment for symmetric MPSoC with power estimation. Though using the CA level has allowed accurate power estimation, MPSoC space exploration at this level is not yet sufficiently rapid compared to RTL. In addition, system description at this level is difficult to obtain especially for off-the-shelf processors.

In an attempt to reduce simulation time, recent efforts have been done to build up fast simulators using Transaction Level Modeling (TLM) [1]. SystemC and its TLM 2.0 kit have become a de facto standard for the system-level description of Systems-on-Chip (SoC) by means of offering different coding styles. Nevertheless, power estimation at the TLM level is still under research and is not well established. Lajolo et al. [7] proposed a power estimation methodology for hardware/software co-design based on concurrent and synchronized execution of various power estimators to estimate different parts of the SoC. In this methodology, the authors use RTL/gate-level power model into the SystemC environment to estimate power.

Lately, McPAT [9] was introduced, which is an improved model of Cacti [16] tool set. McPAT supports power, area and timing estimation for multi-core processors. McPAT use its XML interface to interact with the simulator in order to collect the data needed by its power model. In this paper, we will be comparing McPAT tool which is executed with help of the Multi2Sim [17] functional simulator for dual-core ARM Cortex-A9 processor with the proposed tool for the power estimation accuracy. Here the major issue is with the accuracy of power model and its estimation. To overcome this drawback, a hybrid power estimation methodology (HSL) was proposed.
in [13] by combining interpreted ISS with functional level power model but this methodology suffers in terms of speed as it uses the interpreted ISS for the simulation. Moreover, this methodology was proposed for simple processors such as PowerPC405. Similar to the previous work, a signature-based power model for MPSoC based on FPGA [12] was proposed.

In order to optimize the energy/power consumption of an application, there are several studies proposed in the literature such as work load balancing, scheduling policy and dynamic voltage/frequency management. Recently, embedded research community has shifted its focuses on multi-core processors due to the need of high parallelism in the applications. To address this issue, first, work load balancing is one of the prominent way to increase the performance in turn increase the power but reduces the energy. Recently, there is an approach, which is explicitly focuses on loosely timed system and offers the user a set of primitives to express tasks with duration. Their tool exploits this notion of duration to run the simulation in parallel. This allows the user to focus on the performance-critical parts of the program that need to be parallelized [10]. We will be using similar approach in PETS. Another tool that tackles the same issue at the virtual platform level [6], Kreku et al. present a compiler based technique for automatic generation of workload models for performance simulation, while exploiting an overall approach and platform performance capacity models developed previously. Second, there were many researches done in the field of dynamic power management (DPM) [18], [2] and dynamic voltage and frequency scaling (DVFS) [4], [11]. In this tool, we will integrate the inter task DVFS called Dynamic Slack Reclamation (DSR) into the PET. In our previous work [14], we introduced this methodology for simple mono-processor based platforms and in this work, we propose a simulation based power and energy estimation tool for complex multiprocessor and multi-core processor based heterogeneous platform at the system-level. In this tool, we also integrate the DSR to dynamically change the power consumption profile based on the different task of the application for different processors and also implement work load balancing technique to reduce energy.

III. PETS OVERVIEW

PETS is a hardware/software co-simulation based power & energy estimation/optimisation and DSE tool for complex embedded (multiprocessor and multi-core) processors. The overall flow of the PETS tool is given in the Fig. 1. This tool is designed for easy porting of the application and also enables user to choose the processor architecture to perform hardware/software co-simulation. PETS includes eclipse based environment and also supports a customized graphical output interface programmed in C. To automate the process, we used TCL/TK scipts. During the start of PETS, a configuration menu appears to choose the input file (TCL/TK file). The interface is programmed in a way that the power estimator and simulator communicates dynamically to estimate the change in power/energy of an application for the corresponding platform. The power estimator in return gives the power/energy data to the system-level simulator to automatically balance the workload or to use the DVFS technique as per the user input configuration by using a Smart DSE for a reliable optimisation of the application. This approach makes PETS flexible and easy for software programmers to port their application to the state-of-the-art processor architecture available with the tool.

With the flexible interface of this tool, users can modify the SystemC architecture for the future processor and also able to add their own power models into the tool by using a simple extensible interface.

The main units of the PETS are as follows: First, the power models which will be elaborated in the upcoming section. Second, the fast system-level virtual platform based simulator and finally the smart DSE for an effective optimisation and a reliable DSE of the application. The simulator has some fixed counters such as, cache activity counter, instruction per cycle detector, bus access counter and memory access counter.

![PETS simulation framework](image)

IV. PETS ESTIMATION METHODOLOGY

As mentioned earlier, this section exposes our power estimation methodology that is divided into two steps as shown in the Fig. 1. The first step concerns the power model development for the system functional components. In our framework, the FLPA methodology is extended to develop generic power models for different target platforms. The main advantage of this methodology is to obtain power models, which rely on the functional parameters of the system with a reduced number of experiments. As explained in the previous section, FLPA comes with few consumption laws, which are associated with the consumption activity values of the main functional blocks of the system. The generated power models have been adapted to system-level design, as the required activities can be obtained from a system-level environment. For a given platform, the generation of power model is a one time activity. To estimate power dissipation of a platform, first step is to divide the system into different functional blocks based on the activity and then to tune the divided blocks that are simultaneously activated while a micro-benchmarking code is executed.

The functional parameters are of two types: algorithmic parameters which consumes power due to the software running (typically the instruction per cycle (IPC), fetch access rate ($\rho$) for DSP processor and cache miss rate ($\gamma$) for a processor and area utilization ($\alpha$) for a hardware accelerator) and architectural parameters which are set by the designer (typically the clock frequency, bus frequency and number of processor.
TABLE I. GENERIC POWER MODEL PARAMETERS

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau$</td>
<td>External memory access rate</td>
</tr>
<tr>
<td>$\gamma_1$</td>
<td>L1 cache miss rate for a processor</td>
</tr>
<tr>
<td>$\gamma_2$</td>
<td>L2 cache miss rate for a processor</td>
</tr>
<tr>
<td>SCU</td>
<td>Snoop control unit counter for ARM Cortex-A9 multi-core</td>
</tr>
<tr>
<td>$\psi_1$</td>
<td>L1 cache miss rate for a processor</td>
</tr>
<tr>
<td>$\psi_2$</td>
<td>L2 cache miss rate for a processor</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Parallelism rate for DSP processor fetch stage access</td>
</tr>
<tr>
<td>$\beta$</td>
<td>DSP processor processing rate between instruction memory unit and processing unit</td>
</tr>
<tr>
<td>PSR</td>
<td>Pipeline stall rate between instruction memory unit and processing unit</td>
</tr>
<tr>
<td>IPC</td>
<td>Instruction Per Cycle</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Area utilization for a CLB</td>
</tr>
</tbody>
</table>

Algorithmic

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{\text{processor}}$</td>
<td>Frequency of the processor</td>
</tr>
<tr>
<td>$F_{\text{bus}}$</td>
<td>Frequency of the bus</td>
</tr>
<tr>
<td>N</td>
<td>Number of cores</td>
</tr>
</tbody>
</table>

cores). For example, Table I show the common set of generic parameters used in the power model.

The second part is the characterization of the embedded system power consumption by varying the parameters. These variations are obtained by using some elementary assembly programs (called scenario) to stimulate each functional block separately. Characterization is performed by taking measurements on real boards. Finally, a curve fitting of the graphical representation will allow us to determine the power consumption laws by regression. The analytical form or a table of values expresses the obtained power laws. This power modeling approach was proven to be fast and precise [8]. In our work, this approach has been applied to model power consumption for processor, memory, reconfigurable hardware and I/O peripherals.

The second step of the methodology defines the main architecture of our tool that includes the functional level power/energy estimator kernel and fast IA simulator/fast ISS depending on the architecture as shown in Fig. 2. The functional power estimator evaluates the consumption of the target system with the help of the elaborated power models from the first step. 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 kernel communicates with a Just-In-Time (JIT) Open Virtual Platform (OVP) simulator at IA level. The combination of the above two components described at different abstraction levels (functional and IA) leads to a standalone hybrid power estimation tool that gives a better trade-off between accuracy and speed.

The vital function of virtual platform power estimation is to offer a detailed power analysis by means of a complete simulation of the application. This process is initiated by the functional power estimator through the application and data interface (Fig. 2). In this way, the mapping information is transmitted to the OVPsim simulator. Our simulator consists of processors and hardware components which are instantiated from the OVP library and few other models are written in SystemC in to develop a virtual model of the target platform. We would like to emphasis that processors are described using IA level that sequentially executes the instructions and has no notion of concurrency of micro-architecture.

The power estimation step is carried out on the fly while the application is executed on the simulator. During the simulation, the values of the activities such as IPC, cache miss rate, area occupied, bus accesses are given to the power estimator kernel using the activity counter interface to calculate the global power/energy consumption as shown in Fig. 2.

As we discussed before, the following section will elaborate the first step, i.e., the power model development.

V. POWER MODELING METHODOLOGY

In order to prove the effectiveness of the proposed tool, we used different multi-core and heterogeneous architectures implemented into the OMAP (3530, 4430 & L138), CARMA (quad core) and Xilinx (Zynq & Virtex II Pro) platform. This section elaborates the first step of the proposed tool, which is power model development by using functional level power analysis (FLPA) methodology. As discussed before, we adapted the FLPA methodology to develop generic power models for the used platforms. As a first step of the power development, the processor architecture is divided into different functional blocks such as the clock unit, the pipeline stage unit, the memory unit, the bus unit, the peripheral unit and the hardware accelerator unit, etc. Then, the characterization of each component is done by using micro-architectural benchmarks in order to extract the related power consumption models.

A. Power model elaboration

In Table II, we present power consumption models for different processor architectures. The input parameters on which the power models rely are the frequency of the processor ($F_{\text{processor}}$), Instruction Per Cycle ($0 \leq \text{IPC} \leq 2$), parallelism rate for DSP processor fetch stage access ($0 \leq \rho \leq 1$), DSP processor processing rate between instruction memory unit and processing unit ($0 \leq \beta \leq 1$), Pipeline stall rate ($0 \leq \text{PSR} \leq 1$) and the cache miss rate ($0 \leq \gamma_1, \gamma_2 \leq 0.1$). The frequency of the processor and bus are set by the designer, whereas IPC, cache miss rate etc., are considered as an activity of the processor during the execution of the application. The activity of the processor could be collected from the system-level simulator.

**FPGA power model:** A power model of the hardware accelerator/reconfigurable part is developed for the Zynq & Virtex II Pro board by high-level synthesis tool. For any given FPGA, the high level parameters which can be extracted from the specification are the frequency $F$, the switching rate $\beta$ and the utilized area $\alpha$. In this power model development, we used GAUT \(^3\). With the help of GAUT, we predicted the power with good estimates. An automated 3 way entries LUT of power dissipation values written in C file is used. This is due to the results does not come as a linear equation of the before mentioned parameters. The power estimation is done based on interpolation of the mentioned 3 input parameters.

\(^3\)http://www – labsticc.univ – ubs.fr/www – gaut/
Extrapolation for heterogeneous multiprocessor architectures: The developed power models are used in the system-level environment for consumption estimation of heterogeneous multi core/processor architectures which may contain several different processors and hardware accelerators. This approach allows architecture scalability which cannot be implemented due to the hardware limitations or unavailability of the target component. For instance, we cannot exceed dual-core ARM Cortex-A9 based architecture using our Zynq platform. We mention here that it is necessary to compute the energy before the deduction of the total power consumption. Equation 1 gives the total energy consumption of the platform. The parameters used are the processor \((E_p)\) and the conventional blocks \((E_{CLB})\). The equation also involves the energy consumption of the synchronization part \((E_{sync})\) required to access the shared memory \((E_{mem})\) and the shared I/O resources \((E_{I/O})\). Where, \(i\) and \(j\) are the total number of processor cores and hardware accelerator’s (CLB) respectively.

\[
E_{total} = \sum_{i=1}^{n} E_p + E_{mem} + E_{sync} + E_{I/O} + \sum_{j=1}^{n} E_{CLB} \tag{1}
\]

VI. Power optimization technique

DSR & work loading balancing are one of the promising techniques to reduce power/energy consumption of modern processors. The pseudo frequencies are built at startup based on the available frequency in the real board and then at runtime.

The developed power models are used in the system-level environment for consumption estimation of heterogeneous multiprocessor architectures which may contain several different processors and hardware accelerators. This approach allows architecture scalability which cannot be implemented due to the hardware limitations or unavailability of the target component. For instance, we cannot exceed dual-core ARM Cortex-A9 based architecture using our Zynq platform. We mention here that it is necessary to compute the energy before the deduction of the total power consumption. Equation 1 gives the total energy consumption of the platform. The parameters used are the processor \((E_p)\) and the conventional blocks \((E_{CLB})\). The equation also involves the energy consumption of the synchronization part \((E_{sync})\) required to access the shared memory \((E_{mem})\) and the shared I/O resources \((E_{I/O})\). Where, \(i\) and \(j\) are the total number of processor cores and hardware accelerator’s (CLB) respectively.

\[
E_{total} = \sum_{i=1}^{n} E_p + E_{mem} + E_{sync} + E_{I/O} + \sum_{j=1}^{n} E_{CLB} \tag{1}
\]

**TABLE II. GENERIC POWER MODELS**

<table>
<thead>
<tr>
<th>Processors</th>
<th>Power models</th>
</tr>
</thead>
<tbody>
<tr>
<td>PowerPC</td>
<td>(P(mW) = 1.03 \cdot \gamma + 0.12 + 0.54)</td>
</tr>
<tr>
<td>ARM9</td>
<td>(P(mW) = 0.79 \cdot \gamma + 38.65 \cdot IPC + 0.26 (\gamma + 0.26) + 10.13)</td>
</tr>
<tr>
<td>ARM Cortex-A8 (single core)</td>
<td>(P(mW) = 0.63 \cdot \gamma + 66.54 \cdot IPC + 0.67 (\gamma + 1.4 (\gamma + 10.45))</td>
</tr>
<tr>
<td>ARM Cortex-A9 (dual core)</td>
<td>(P(mW) = 0.63 \cdot \gamma + b \sum_{i=1}^{2} (IPC_{c1-c2}) + c \sum_{i=1}^{2} (\gamma + 1.4 (\gamma + 12.45))</td>
</tr>
<tr>
<td>ARM Cortex-A9 (quad core)</td>
<td>(P(mW) = 0.59 \cdot \gamma + b \sum_{i=1}^{4} (IPC_{c1-c4}) + c \sum_{i=1}^{4} (\gamma + 1.4 (\gamma + 2.5 (SCU) + 16.45)</td>
</tr>
<tr>
<td>DSP</td>
<td>(P(mW) = 0.66 \cdot \gamma + 0.138) (1 - PSR) + 0.12 (\gamma + 0.0257 (\gamma (shared + unified) + 0.0345 (PSR) + 8.45)</td>
</tr>
<tr>
<td>TMS320C64x</td>
<td>(P(mW) = 0.66 \cdot \gamma + 0.138) (1 - PSR) + 0.12 (\gamma + 0.0257 (\gamma (shared + unified) + 0.0345 (PSR) + 8.45)</td>
</tr>
</tbody>
</table>

**Fig. 3.** PETS power estimation of different processor architecture based platform

**Fig. 4.** Cache miss rate activity
Dynamic slack are produced due to the early completion of the application. For this reason, we consider the worst case completion time of each and every task in an application. This is also known as Longest Processing Time (LPT). By this way, we reduce the dynamic power consumption of the processor by increasing the execution time of early completion and by decreasing the frequency of the processor for the tasks. The identification of dynamic slack in a task is retrieved by comparing the actual execution time and worst case execution time.

Workload balancing: As mentioned earlier, work load balancing is one of best way to increase the performance and reduce the energy consumption of the processor. The objective is to exploit the parallelism in both architecture and also in the algorithm. To do that, all the processor core are equally loaded with the task in an application. The weight of each task is retrieved by static profiling of the application. From the retrieval of weight, the tool will compute the following steps, computational complexity and feasibility of the task mapping of the given application to the processor core or on a FPGA. This technique doesn’t consider the communication overhead between the processor cores, which is a critical factor of communication intensive applications. Similar type of work has been proposed to achieve load balancing of a JPEG application for MPSoC synthesis [20].

So far, power model for the single core and dual core processor, FPGA and heterogeneous multiprocessor architecture has been developed. Estimation of the overall power consumption for single core processor, dual-core processor and heterogeneous multiprocessors at system-level will be briefed up with the results in the next section as the second step of power estimation methodology with the help of Multimedia and SPEC2006 benchmarks.

VII. POWER ESTIMATION ENVIRONMENT

In the second step, a system-level prototype of an PowerPC405, ARM9, ARM Cortex-A8, multi-core ARM Cortex-A9 based architectures has been programmed. The developed prototype uses different SystemC models especially the JIT & interpreted ISS simulator provided by OVP and SoCLib for the target processor. Furthermore, the cache parameters, pipeline stage unit and bus latencies are set to emulate the real platform behaviour. A set of counters are injected into the simulator to determine the values of different IPC, cache miss rates: read data miss, write data miss and read instruction miss access rate, memory access and bus access.

A. Power & energy estimation results

1) Power estimation for single core architecture: First study is to estimate the power consumption of single core processor architectures. For example, we used the MPEG 2 decoder simple profile application as a benchmark to show the data gather by the activity counters in Fig. 4. The MPEG application consists of 4 main tasks: VLD, IQ, IDCT and IMC. In Fig. 4, we show the results of the activities fetched by the counter interface of the fast IA simulator for each task of the MPEG-2 Part 2 decoder application. The results collected in Fig. 4 will help designers to tune the application for a better power optimization. Detailed results from multimedia and single processing benchmarks show a similar behaviour like the MPEG-2 Part 2 decoder application. In the following step, we estimated the total power dissipation of each tasks using the power models shown in Table II (single core). Fig. 3 illustrates the results and shows the comparison between the proposed tool and the real board measurements. In order to prove that our tool is accurate, we used several other benchmarking programs available in the industry as shown in the Fig. 3. Our tool has a negligible maximum error of 4% compared to real board measurement.

2) Multi-core & heterogeneous multiprocessor architecture energy estimation: The second study involves a processor architecture with identical cores and heterogeneous processors to run the multimedia benchmarks. All the cores/processors execute the same workload. Fig. 5 & Fig. 6 report the total energy consumption in mJ.

Compared to real board measurements, our tool achieved a maximum error of 4.5% for dual core architecture while McPAT has a maximum error of 28%. This accuracy is obtained because of two main reasons. First, power laws are extracted from real board measurements. Second, additional activities that are intrinsic to parallel processing such as synchronization and communication overheads are accurately evaluated by using our PETST build-in simulator. To retrieve activities for OMAP3530 platform, we used C64x+ cycle accurate simulator with a special interface from PETST. The before mentioned explanations encourage us to consider architectures with a higher number of cores/processor in the context of exploring new complex heterogeneous processor architectures. Fig. 5 & Fig. 6 show that, for the implemented multimedia parallel application, adding cores/processors/hardware accelerator to the system decreases the execution time, which improves the system performance.

B. Power optimization for multi-core & multiprocessor architecture

Power optimization based on DSR: In order to evaluate the described DSR technique implemented in our PETST tool, we used a H.264 (4 slices 50 frames) application running on an ARM Cortex-A9 (quad core) processor based platform. The platform is configured with 4 processor cores at 500MHz. Based on the longest processing time, we determine the parameters of each task (release time, actual-case and worst-case execution time, relative deadline and periodicity). These parameters are given as an input to the tool during the execution of H.264 application. In the start-up of the tool, we had set up some pseudo frequencies based on the available frequency on the board in order to dynamically change the frequency during the application simulation. To optimize the power consumption of the overall system, we change dynamically the frequency of each processor core according to these pseudo frequencies (250 MHz, 500 MHz and 720 MHz).

In order to evaluate our tool with this technique, we executed an modified H.264 with different processor configu-
Energy optimization based on DSR of H.264 application:
The results of power consumption during the execution of the H.264 is given in Table III with different configurations of the multi-core processor based platform. We have different configurations of the number of active cores to verify the impacts on power consumption with and without the optimization algorithm. This analysis takes into account the CPU load by varying the number of threads with different configurations decoder (2, 4 slices). The Table III shows the power dissipation and execution time of the different configurations of H.264 for active processor cores.

<table>
<thead>
<tr>
<th>Processor core</th>
<th>Power (mW)</th>
<th>Execution time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 processor core without DSR</td>
<td>374</td>
<td>4.5</td>
</tr>
<tr>
<td>2 processor core with DSR</td>
<td>231</td>
<td>4.8</td>
</tr>
<tr>
<td>4 processor core without DSR</td>
<td>391</td>
<td>3.7</td>
</tr>
<tr>
<td>4 processor core with DSR</td>
<td>314</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Energy optimization based on workload balancing:
To implement the workload based energy optimisation in our tool, we used a similar setup like the DSR (H.264 on an ARM Cortex-A9 quad core processor). First, static profiling of the sequential H.264 application is done. Based on the profiling results, we get the dependency graph of the application and also the computational intensive task of H.264...
(i.e., motion estimation/intra prediction). We further divided motion estimation and intra prediction to have a four way graph to be implemented on the quad core processor. Table IV presents the result of energy optimization based on work load balancing technique by PETS. In the table, it also gives the relation between without load balancing. 2 processor core and 4 processor core energy optimisation result. From the table IV, we are able to see that using this technique will reduce the energy consumption of the processor.

VIII. CONCLUSION

This paper proposes a simulation based Power Estimation and optimisation Tool for complex processor based platforms at System-level. PETS is the first tool to integrate power models and optimization technique at the simulation level for the state-of-the-art processors. PETS tool includes an eclipse environment and a scripting interface to instantiate the processor and the application. Moreover, it contains a graphical interface for displaying the results of the simulation, which is coupled with the eclipse environment. First, a power modeling methodology for system-level has been proposed to address the global system consumption that includes heterogeneous processors, reconfigurable hardware, etc. Secondly, the developed power models are seamlessly coupled with a fast simulation technique to grab the needed micro-architectural activities for the power models with a better trade-off between accuracy and speed. In addition, using functional power models brings transparency regarding the low level implementation, reduces the number of dependent parameters and ease out the extraction of the required data. Our proposed system-level power estimation tool explores these two aspects and offers an accurate and fast system-level power prediction. Experimental results show that our tool exhibits less than 4% average error compared with the real measurements and more accurate compared to other tools. With the proposed tool, the designer can explore several implementation choices: single core, multi core and heterogeneous multiprocessor architecture based platforms.

As part of future work, we will focus more on integration of thermal and reliability models based on energy and power at the system-level. Furthermore, in order to obtain more accurate power estimations, some power model refinements must be realized.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community’s Seventh Framework Programme [FP7/2007-2013] under the ParaDIME Project (www.paradime-project.eu), grant agreement no 318693.

REFERENCES