Université de Valenciennes et du Hainaut-Cambrésis

Thèse de doctorat

Pour obtenir le grade de Docteur de l'Université de

VALENCIENNES ET DU HAINAUT-CAMBRESIS

Discipline, spécialité selon la liste des spécialités pour lesquelles l'Ecole Doctorale est accréditée :
Informatique

Présentée et soutenue par Santhosh Kumar, RETHINAGIRI.

Le 14/03/2013, à Valenciennes

Ecole doctorale :
Sciences Pour l'Ingénieur (SPI)

Equipe de recherche, Laboratoire :
Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH)

Une approche système pour l'estimation de la consommation de puissance des plateformes MPSoC

JURY

Président du jury
- Valderrama, Carlos. Professeur. Université de Mons.

Rapporteurs
- Bourennane, El-Bay. Professeur. Université de Bourgogne.
- Belleudy, Cécile. Maître de conférences, HDR. Université de Nice-Sophia Antipolis.

Examinateurs
- Senn, Eric. Maître de conférences, HDR- IRISA. Rennes.

Directeur de thèse
- Dekeyser, Jean-Luc. Professeur. Université de Lille1.

Co-directeur de thèse
-Niar, Smail. Professeur. Université de Valenciennes.

Co-encadrant
-Ben Atitallah, Rabie. Maître de conférences. Université de Valenciennes.

System-Level Power Estimation Methodology for MPSoC based Platforms

By

Santhosh Kumar Rethinagiri

A thesis submitted for the degree of

Doctor of Computer Science

At the Université de Valenciennes et du Hainaut-Cambrésis

Ecole Doctorale Sciences Pour l'Ingénieur Université Lille Nord-de-France-072

Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH)

Speciality: Low Power Electronics and Design

Domain : Design Tools, System and Software Design for Embedded Systems

Dissertation

Presented and defended on: **March 2013**

JURY

Reporter	**Prof. El-Bay Bourennane,** Université de Bourgogne, Dijon
Reporter	**MCF HDR Cécile Belleudy,** Université de Nice-Sophia Antipolis
Examiner	**Prof. Carlos Valderrama,** Université de Mons, Mons
Examiner	**MCF HDR Eric Senn,** Université de Bretagne Sud, Lorient
Superviser	**Prof. Jean-Luc Dekeyser,** Université de Lille 1, Villeneuve d'Ascq
Superviser	**Prof. Smail Niar,** Université de Valenciennes, Valenciennes
Co-Superviser	**Assoc. Prof. Rabie Ben Atitallah,** Université de Valenciennes

L'université de Valenciennes et du Hainaut-Cambrésis
Le Mont Houy – LAMIH – 59313 VALENCIENNES Cedex 9
Téléphone (+33) 3 27 51 12 34 – Télécopie (+33) 3 27 51 11 00

Acknowledgements

One of the joys of completion is to look over the journey past and remember all the friends and family who have helped and supported me along this long but fulfilling road. It is a pleasure to thank the many people who have made this thesis possible.

It is difficult to overstate my gratitude to my Ph.D. co-supervisor, Dr. Rabie BEN ATITALLAH. With his enthusiasm, inspiration, and great efforts to explain things clearly and simply, he made working on this thesis fun for me. Throughout my thesis-writing period, he provided encouragement, sound advice, good teaching, good company, and lots of good ideas. I would have been lost without him.

I feel enormously indebt to my directors at INRIA-Lille Nord Europe and at University of Valenciennes, Jean-Luc DEKEYSER and Smail NIAR, for giving me this opportunity. Furthermore, I want to thank all my colleagues of the LAMIH laboratory and the INRIA DaRT team for creating a perfect environment to work and to spend a pleasant time.

I would also like to thank my examiners, who provided encouraging and constructive feedback. It is no easy task, reviewing a thesis, and I am grateful for their thoughtful and detailed comments. To the many anonymous reviewers at the various conferences, thank you for helping to shape and guide the direction of the work with your informative and instructive comments.

This thesis was funded by National Research Agency (ANR) of France in the frame of the OPEN-PEOPLE project. As a member of OPEN-PEOPLE, I have been surrounded by wonderful colleagues who have provided me a rich and fertile environment to study and explore new ideas. At OPEN-PEOPLE, I would like to thank the project leader, Dr. Eric SENN, who has been extremely supportive in allowing me to participate in LAB-STICC laboratory activities while pursuing my PhD studies.

A special thanks to all my friends, who have accompanied me in this wonderful journey of professional and personal growth that started in Chennai and ended in Valenciennes. Thanks for putting up with me, being a support and sharing some unforgettable moments.

Lastly, I would like to thank my family for all their love and encouragement. I wish to thank my parents, Budha Purnima and Late Rethinagiri. They bore me, raised me, supported me, taught me and loved me. And most of all for my loving, supportive, encouraging and patient wife Sushma Rajaram whose faithful support during the final stages of this Ph.D. is so appreciated. I love you all dearly. Thank you.

Finally, I want to dedicate this thesis to my family, specifically to my mother and my wife. Thank you for supporting all my decisions and for your love. I hope you are proud of me and my work.

Santhosh Kumar
Valenciennes

Abstract

Due to the ever increasing constraints on power consumption in embedded systems, this thesis addresses the need for an efficient power modeling and estimation methodology based tool at system-level. On the one hand, todays embedded industries focus more on manufacturing multiprocessor based platforms as they are cost and power effective. On the other hand, modern embedded applications are becoming more and more sophisticated and resource demanding: multimedia (H.264 encoder and decoder), software defined radio, GPS, mobile applications, etc. The most vital corrective measure adopted to tackle the increasing complexity of Multiprocessor System-on-Chip (MPSoC) is to commence designing at the system-level. Decisions taken at the system-level have a greater impact on the tape out of the chip in terms of power and energy but it poses a bigger challenge due to the large architectural solution space. For this reason, efficient system-level power estimation tools are therefore necessary to enable proper Design Space Exploration (DSE) based on power/energy and timing.

In this thesis, we propose a tool based on efficient hybrid system-level power estimation methodology for MPSoC. In this methodology, a combination of Functional Level Power Analysis (FLPA) and system-level simulation technique are used to compute the power of the whole system. Basically, the FLPA concept is proposed for processor architecture in order to obtain parameterized arithmetic power models depending on the consumption of the main functional blocks. In this work, FLPA is extended to set up generic power models for the different parts of the platform. In addition, a simulation framework is developed at the transactional level to evaluate accurately the activities used in the re-

lated power models. The combination of the above two parts leads to a hybrid power estimation that gives a better trade-off between accuracy and speed. The proposed methodology has several benefits: it considers the power consumption of the embedded system in its entirety and leads to accurate estimates without a costly and complex material. The proposed methodology is also scalable for exploring complex embedded architectures. Based on the proposed methodology, our Power Estimation Tool at System-Level (PETS) is developed.

The effectiveness of our PETS tool is validated in terms of accuracy and speed through a typical mono-processor and multiprocessor embedded system designed around the TI OMAP (3530 and 5912) and the Xilinx Virtex II Pro FPGA boards. This methodology is demonstrated and evaluated by using a variety of basic programs to a complete media benchmarks. In order to ensure accuracy and speed, first we compared the estimated power value with the real board measurements for both mono-processor and multiprocessor architectures. Our obtained power estimation results provide less than 3% of error for mono-processor, 3.8% for homogeneous multiprocessor system and 4.3% for heterogeneous multiprocessor system. Second, we compared the estimation speed of our tool with the state-of-the-art power estimation tools and it resulted up to 70x faster.

Résumé

Avec l'essor des nouvelles technologies d'intégration sur silicium submicroniques, la consommation de puissance dans les systèmes sur puce multiprocesseur (MPSoC) est devenue un facteur primordial au niveau du flot de conception. La prise en considération de ce facteur clé dés les premières phases de conception, joue un rôle primordial puisqu'elle permet d'augmenter la fiabilité des composants et de réduire le temps d'arrivée sur le marché du produit final.

Dans cette thèse, nous proposons une méthodologie efficace pour l'estimation de la consommation de puissance des plateformes MPSoC. Cette méthodologie repose sur une combinaison d'une analyse fonctionnelle de la puissance (FLPA) pour l'obtention des modèles de consommation et d'une technique de simulation au niveau transactionnel (TLM) pour calculer la puissance de l'ensemble du système. Fondamentalement, FLPA est proposée pour modéliser le comportement des processeurs en terme de consommation afin d'obtenir des modèles paramétrés de haut niveau. Dans ce travail, FLPA est étendue pour mettre en place des modèles de puissance génériques pour les différentes parties du système (mémoire, logique reconfigurable, etc.). En outre, un environnement de simulation a été développé au niveau transactionnel afin d'évaluer avec précision les activités utilisées dans les modéles de consommation. La combinaison de ces deux parties conduit à une estimation de la puissance hybride qui donne un meilleur compromis entre la précision et la vitesse. La méthodologie proposée a plusieurs avantages: elle estime la consommation du système embarqué dans tous ses éléments et conduit à des estimations précises sans matériel coûteux et complexe. La méthodologie proposée est évolutive pour explorer des

architectures complexes embarquées. Notre outil d'estimation de puissance au niveau système PETS (Power Estimation Tool at System-level) est développé sur la base de la méthodologie proposée. L'efficacité de notre outil PETS en termes de précision et rapidité est validée par des architectures embarquées monoprocesseur et multiprocesseur conçues autour des plateformes OMAP (3530 et 5912) et FPGA Pro Xilinx Virtex II.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

CHAPTER 1

INTRODUCTION

This chapter describes the context, the problem statement, the contribution, and the plan of thesis. The main objective of this thesis work reclines in the scarcity of a fast and accurate power estimation tool at the system-level for complex embedded systems such as: homogeneous and heterogeneous Multi-Processor System-on-Chip (MPSoC). Without such a tool, a reliable Design Space Exploration(DSE) based on power and timing at system-level becomes impossible to achieve in a reasonable time, due to the broadness of the architectural solution space. As a consequence, taking the best architectural decisions becomes very challenging for system designers. However, since decisions taken at the system-level are the most relevant in affecting the quality of the final design, it is very important to take them right from the beginning, in order to avoid costly and time-consuming reiterations. The contribution of this work is therefore the implementation of a fast and accurate system-level power estimation methodology in a tool, which can really help system designers to take the best architectural decisions early in the design cycle.

1.1 Context

Due to the ongoing nano-miniaturisation in chip production, estimation of power consumption is becoming a critical pre-design metric in complex embedded systems such as homogeneous and heterogeneous Multi-Processor System-on-Chip (MPSoC). On the one hand, today's embedded industries focus more on manufacturing RISC processor-based platforms as they are cost and power effective. On the other hand, modern embedded applications are becoming more and more sophisticated and resource demanding. Examples of the concerned applications are numerous such as multimedia (H.264 encoder and decoder), software defined radio, GPS, mobile applications, etc. The computation requirements of such systems are very important in order to meet real-time constraints and high quality of services.

Recently, the ITRS [56] and HiPEAC [1] roadmaps promote *power defines performance* and *power is the wall*. An efficient and fast Design Space Exploration (DSE) of such systems needs a set of tools capable of estimating performance and power at higher abstraction level in the design flow. In current industrial and academic practices, power estimation using low level CAD tools is still widely adopted, which is clearly not suited to manage the complexity of modern embedded systems. Facing this issue, designers should calculate the power consumption as early as possible in the design flow to reduce the time-to-market and the development cost. 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 efficient power modeling methodology, the rapid system prototyping and the accurate power estimates.

Hence, high-level power estimation and optimization in embedded system is the key issue in the early determination of the power budget, being infeasible to synthesize every design solution down to the gate and layout levels in a reasonable time. The goal is to shorten the design turn-around time, by widely exploring the architectural design space and to early re-target the architectural design choices. Accuracy and efficiency of a high-level power analysis should contribute to meet

[1] http://www.hipeac.net/system/files/hipeacvision.pdf

the power requirements and thus avoid costly re-design processes.

To address these design requirements, several industrial and academic institutes are devoting their efforts to facilitate the development of tools for MPSoC design taking into consideration the power metric as an essential parameter. The ANR-08-SEGI-013 OPEN-PEOPLE[1] project was defined in order to answer these requirements, it is funded by the National Research Agency (ANR) of France. OPEN-PEOPLE stands for Open Power and Energy Optimization PLatform and Estimator. Among the target systems, we mention heterogeneous MPSoC such as the TI OMAP 3530 and reconfigurable circuits like the Xilinx Virtex5 FPGA. Our platform allows power estimation using:

- direct access to the hardware execution boards and the measurement equipments. This first alternative enables designers 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 components.
- 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 Fig.1.1 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 the 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. Various research and development works are currently done in the OPEN-PEOPLE project. These works include the definition

[1] www.open-people.fr/

of new methods and tools to model the different components of a heterogeneous system architecture: 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.

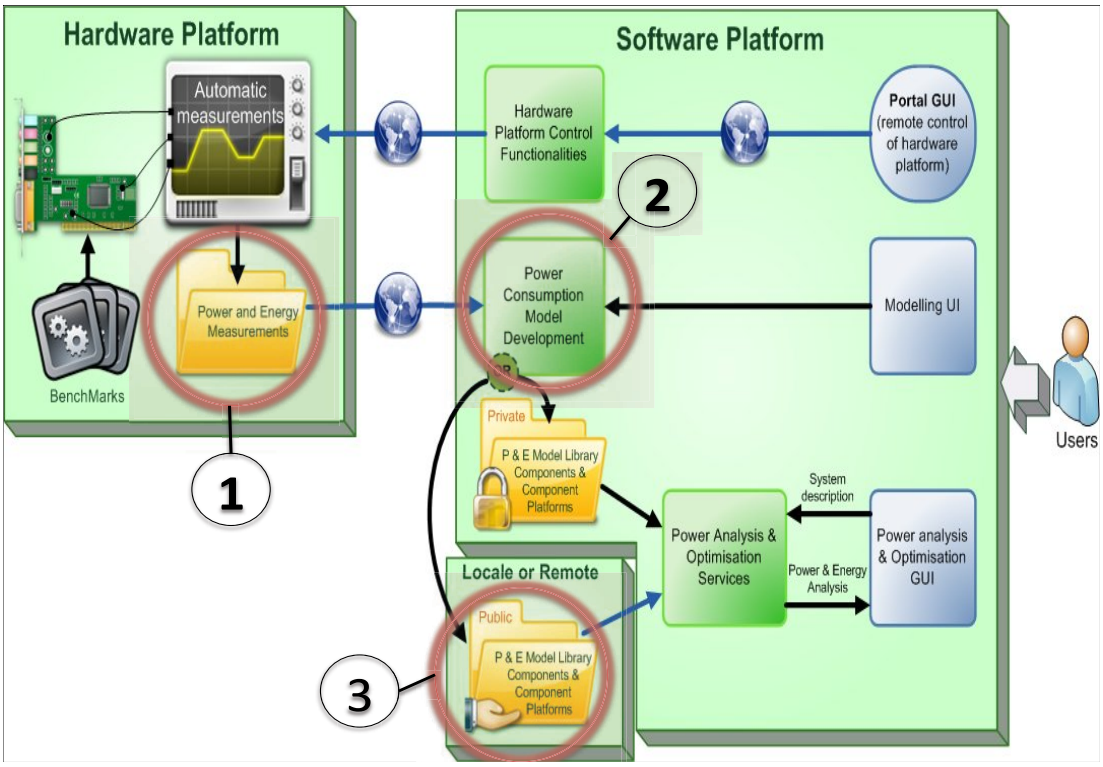

Figure 1.1: Context of the thesis in the OPEN-PEOPLE project

This PhD thesis research is a part of the OPEN-PEOPLE project. The main objectives of this work are encircled in red in the Fig.1.1. The first challenge concerns the development of power models for the selected platforms in the frame of the OPEN-PEOPLE project. The second challenge concerns the Software/Hardware co-simulation of MPSoC at the system-level where we focus on the transactional level (or TLM: Transaction Level Modeling) and virtual platform modeling in order to extract the relevant data for power estimation. The third challenge

concerns the development of a tool based on an efficient power estimation methodology offering a better trade-off between speed and accuracy, hence a reliable and a fast DSE.

1.2 Problem statement

In order to achieve the objective of designing efficient system-level power estimation tools for MPSoC, we started with extracting the main challenges related to the context of this thesis. We have identified three essential factors that define the problem statement and they are mentioned as follows:

- **Power modeling methodology:** The power modeling process is centred around two correlated aspects: the power model granularity and the main activity characterization. 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. In general, fine-grain power estimation results in a 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 activities that cannot be determined easily depending on the complexity of the system. The second aspect involves the characterization of the activities, which requires a huge number of experimental measurements and thus a significant time to 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. For this first challenge, the main question that we have to answer is: *what is the power modeling methodology suitable for MPSoC system-level design that can offer a better trade-off between the time needed to generate the power model and its corresponding accuracy?*
- **Software/Hardware co-simulation of MPSoC:** The second challenge in-

volves the abstraction level on which the system is described. It starts from the usual Register Transfer Level (RTL) and extends upto the algorithmic level. This challenge is tackled by several frameworks by means of the development of Electronic System Level (ESL) tools. The objective is to unify the hardware and software design and to offer a rapid system-level prototyping. In the last years, significant academic and industrial efforts have been deployed to deal with the software/hardware co-simulation issue as the conventional RTL and Cycle-Accurate (CA) tools cannot adequately support the complexity of future MPSoC since they are too slow for a meaningful execution of the software. These efforts led to a taxonomy of tools based on different simulation techniques, description languages, abstraction levels, etc. However, most of these tools don't address the issue of power estimation. In general, going from low to high design level corresponds to more abstract description and then coarser activity granularity. For this second challenge, the main question that we have to answer is: *what are the appropriate simulation technique and the abstraction level suitable for rapid MPSoC prototyping and for extracting accurately the activities for the defined power model (the first challenge)?*

- **Power based design methodology:** Shifting towards higher levels of abstraction has proved to be a winning strategy for dealing with increasing complexity. Indeed, by abstracting away the lower-level details, implementation is faster, which means lower engineering effort, lower cost and lower time to market, as well as higher productivity. Decisions made at the system-level have a very strong impact on the quality of the final product, since the degree of achievable optimization is normally proportional to the abstraction level and, indirectly, to the point in the design flow where decisions are taken: the earlier the better. However, although very important, power based decisions at system-level are very hard to take and this is for two main reasons: the first is that, at the system-level, the design space to consider is extremely broad as a consequence of the limited amount of implementation details available. The second reason is that the impact of the decisions taken at system-level is not known until a very late stage of the design process, which can take months of work. From the second reason mentioned above, it can be con-

cluded that the lack of a quick and accurate System-Level Power Estimation (SLPE) approach is one of the main obstacles to successful system-level design today. In fact, if an efficient system-level methodology for energy and performance estimation was available, it would be possible to carry out a reasonably reliable DSE and thus judge from the beginning of the design flow which architecture is the most suitable for a certain applications domain, in terms of performance and power consumption. For this third challenge, the main question that we have to answer is: *How to provide a tool at system-level in order to guide the designer during the different design choices based on power estimation?*

1.3 Contributions

Our contributions through this thesis in the field of system-level power estimation for MPSoC design is to propose solutions to remove technological barriers presented in the previous section. In summary, our contributions are:

- **Power modeling methodology:** For modeling the power consumption of a MPSoC, we propose the application of the Functional-Level Power Analysis (FLPA), which basically allows to extract the processor power consumption model with a set of high level parameters (i.e., frequency, cache miss rate, etc.). The aim of this first step is to identify the sources of power consumption in the embedded system in its entirety (software tasks, hardware accelerators and the memory system). 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 in order to guarantee the maximum level of accuracy. Afterwards, power models are elaborated by regression functions or simply recorded as multi-entries look up tables. It is important to have a scalable approach to address the complex system power/energy estimation issue. For these reasons, the developed power models are used in the frame of system-level estimation of homogeneous and heterogeneous MPSoC that may contain several processors and hardware accelerators. By extending FLPA

methodology for the overall system, the development of power models are rapid and accurate which answers our first challenge.

- **Co-simulation of MPSoC for power estimation:** To answer the second challenge, we propose an framework for hardware/software co-simulation at system-level for power estimation. A virtual platform based simulation technique is presented. It speeds up the time of the system with a good performance estimation accuracy. Virtual platform technique is implemented at a high abstraction level by using the SystemC/Transaction Level Modeling (TLM) 2.0 kit. It leverages the high-level system specification to provide a hardware/software co-simulation by using a Just In Time (JIT) simulator. The advent of design flows based on SystemC makes it possible that the modeling of a system is done by a tight and efficient coupling relationship between the hardware model and the virtual platform. A co-simulation solution proposed here is based on an architectural template consisting of a common bus and several processors SystemC with wrapper interacting with hardware IP's through the bus. There is a generic wrapper that is put around all CPU models. The wrapper can be seen as an extension of SystemC that makes hardware/software co-simulation more efficient. The SystemC simulation serves as a master that drives the overall simulation. The proposed SystemC simulation environment targets homogeneous and heterogeneous MPSoC with an ability to provide the relevant activities on which the power model relies as stated in the first contribution.
- **Design methodology based on power estimation:** The co-simulation environment for MPSoC has been enriched by models for estimation of power/energy in order to make possible the evaluation of the application. This parameter is the first selection criterion between different architectural solutions possible in order to have a design space exploration. The co-simulation environment also includes flexible tools for assessing the execution time. This metric is a second criterion for the selection of the most appropriate architectural solutions. This proposal was embodied in our work by developing models of performance estimation at system-level. In a detailed view, our work mainly contributes to the area of system-level estimation by proposing a system-level framework for power estimation, energy and performance,

which can indeed help system designers to take the best architectural decisions early in the design flow. The framework presented in this thesis comes as a proof of concept. Further extensions are required to make this framework more general and complete, as it will be discussed in the future works section, at the end of the thesis.

The contributions above were implemented in the environment of OPENPEOPLE developed by the team, especially to complement its design flow.

We can also classify the contribution of this thesis work into three sections :

- **Concept related:** It denotes the definition of the hybrid power estimation methodology. We start with the initial abstract idea of developing power models at the functional level and then integrating it into an independent simulation based framework for power estimation of different platforms.
- **Tool related:** The stated work in the thesis requires scripting, aimed at different processors and automating the operation that could not have been carried out manually. For instance, scripts have been written to automate the processor characterization, power model and also SystemC IP selection for a particular platform. At the software level, scripts have been written to set the application which has to be simulated on the SystemC environment.
- **Experiment related:** Experiments have been conducted throughout the entire development of hybrid power estimation methodology to validate the corresponding estimation accuracy and speed. Power estimation accuracy has been validated against real board measurements for a set of benchmarks. The target platforms used as reference for the experiments are Virtex II Pro, OMAP3530 and OMAP5912.

1.4 Plan of the thesis

This thesis report is organized as follows:

Chapter 2 presents an overview of the most common approaches of system-level design/ power estimation in use today and compares them to the proposed approach in this thesis. Large space is given to the description of simulation-based approaches with particular focus on the Transaction Level Modeling (TLM).

Chapter 3 describes the proposed power modeling methodology and focuses on developing power models for processors (PowerPC, ARM9 and ARM CortexA8), homogeneous and heterogeneous MPSoC based platforms.

Chapter 4 gives the details of the development of the proposed system-level environment which is required for rapid prototyping and fast simulation for power estimation. The simulation support to integrate power models into SystemC/TLM environment is also presented.

Chapter 5 shows the proposed hybrid power estimation methodology and explains the integration of power models into SystemC/TLM environment. It presents the Power Estimation Tool at System-level (PETS) which is based on hybrid power estimation methodology with the help of case-studies and the validation through the experimental results.

Chapter 6 draws the conclusions on the entire work and leaves space for some future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, we present power estimation methodologies available in the literature and we compare them to the proposed methodology in this thesis. There are two types of approaches available: Simulation-based and analytical-based. More insight will be given on power modeling methodologies and simulation-based tools and in particular to the SystemC/TLM.

The conclusion drawn from the previous chapter is that being able to efficiently carry out system-level power estimation is a necessary condition to make design-space exploration (DSE) at system-level possible. Due to their importance and also lack of a defined model, power estimation at system-level in general is a hot research topic today. In this chapter, we will present a survey of the most significant tools for System-Level Design (SLD) with the capability of power estimation and different types of power modeling approaches. Finally, a comparison of our approach to the methodologies available in the literature is presented.

This chapter is organized as follows: Section 2.2, gives a brief description about basics, definition and motivations for system-level design. Section 2.4 presents the available power modeling methodologies in the literature. Sections 2.5 and 2.6 present the power estimation tools available respectively with

simulation-based and analytical-based approaches. Section 2.8 gives more details about the virtual platform tools available in the industries.

2.2 Motivations for system-level design

In 1983, Gajski and Kuhn derived what is today known as a Y-chart [44], which is a representation of the different abstraction levels at which a system can be modeled and estimated. This is shown in Fig. 2.1. Gajski and Kuhn distinguish five different abstraction levels, represented by concentric circles, each of which is classified according to three different domains: behavioral, structural and physical. The innermost circle corresponds to the lowest abstraction level, while the outermost one to the highest.

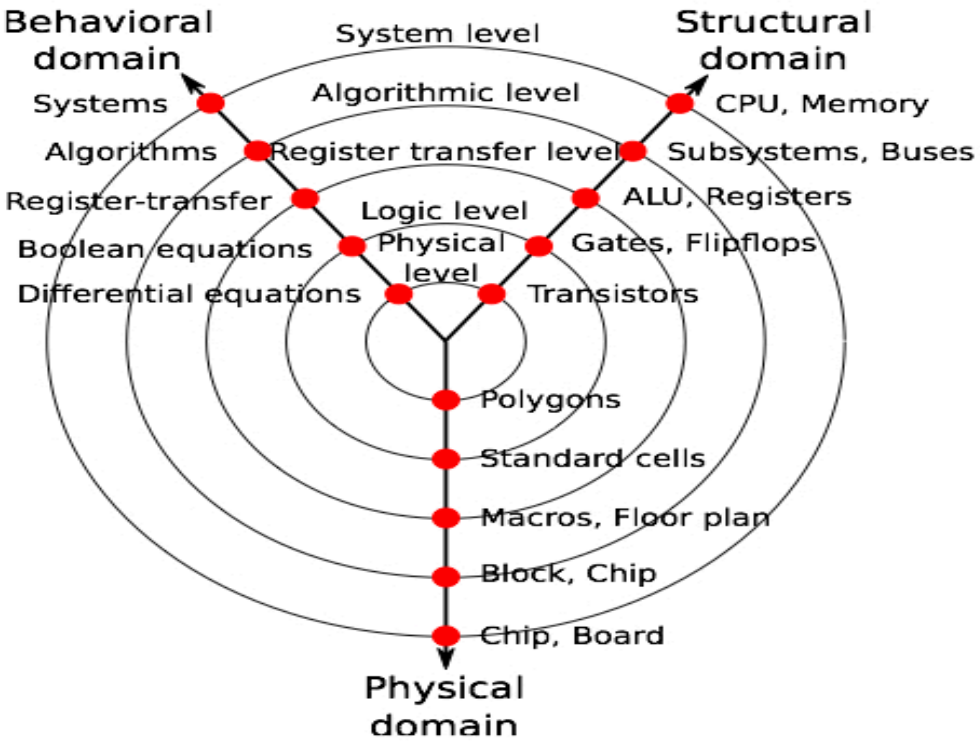

Figure 2.1: Gajski and Kuhn Y chart: source [44]

Until a few decades ago, it was still possible to manually describe a system directly at the physical level, since the amount of complexity was very limited.

When complexity grew, manual description at physical level became impossible and the starting point of the design flow was thus raised to a higher abstraction level, translate (synthesize) a gate-level description into a layout design. Similarly, as complexity kept growing, the starting point of the design flow was further raised up to the Register Transfer Level (RTL) was introduced. Two very well-known examples of languages used for RTL description are VHDL and Verilog. Along with the increase of complexity, todays trend is to further shift the entry level for automatic synthesis up to the system-level.

Thus, shifting towards higher levels of abstraction has proved to be a winning strategy for dealing with the increasing complexity. Indeed, by abstracting away the lower-level details, implementation is faster, which means lower engineering effort, lower cost and lower time to market, as well as higher productivity. Decisions made at the system-level have a very strong impact on the quality of the final product, since the degree of achievable optimization is normally proportional to the abstraction level and indirectly, to the point in the design flow where decisions are taken: the earlier the better. At the system-level, the question that system architects have to answer are the following: given a set of applications and a set of possible architectures, what is the best architecture on which to map this set of applications. The expression best architecture refers to the properties of an architecture in terms of metrics such as performance, power consumption and silicon area, for a given set of applications. For example, what is the power and performance impact of varying the number of levels in the memory hierarchy? What is the best interconnect to use: a bus or a NoC? What is the advantage/disadvantage of implementing part or the whole set of applications in hardware rather than software? These are just examples of the hard choices a designer has to make. Since they are so important, taking the right system-level decisions from the beginning is crucial, especially when complexity grows. Any error at this early stage would lead to annoying design reiterations with a consequent high loss of time, money and probably, a sub-optimal final implementation.

However, although very important, decisions at system-level are very hard to take and this is for two main reasons: the first is that, at the system-level, the design space to consider is extremely broad as a consequence of the limited amount of implementation details available. Fig. 2.2 shows the relation between

the design space width and the abstraction level. The second reason is that the impact of the decisions taken at system-level is not known until a very late stage of the design process, which can take months of work.

From the second reason mentioned above, it can be concluded that the lack of a quick and accurate system-level power estimation approach is one of the main obstacles to a successful system-level design today. In fact, if an efficient system-level methodology for energy and performance estimation was available, it would be possible to carry out a reasonably comprehensive DSE and thus judge from the beginning of the design flow which architecture is the most suitable for a certain application domains, in terms of performance and power consumption. In addition, estimation at any abstraction level is a requirement for the implementation of automatic synthesis tools, since it is only after estimation that the tool can judge what the best solution is.

Efficient estimation at lower abstraction levels has allowed us to have quite mature automatic tools today. Estimation at the physical level requires accounting for the individual capacitance and resistance contributions coming from each transistor and interconnecting wire. Estimation at this level is extremely accurate, but also very slow. Simulation at physical level is also very slow and is thus feasible for only very small designs and for a very short design execution time.

At the gate level, estimation is simplified by the fact that standard cells are used whose physical properties are pre-characterized. Only the impact of cell-to-cell connecting wires has to be estimated separately, which is done using so called wire load models. Estimation at this level is less accurate, although faster, and bigger design sizes can be simulated. At the RTL level, Hardware Descriptive Languages (HDL) are used to describe in words what RTL synthesis translates into logic gates. Simulation is very common at RTL and reasonably fast for medium size designs running very short chunks of application. However, estimation made at this level loses accuracy due to the lack of enough physical details. In general, the increase of the abstraction level is directly proportional to the estimation speed and inversely proportional to the estimation accuracy. When it comes to system-level, the lack of an efficient estimation methodology has been an obstacle to have mature automatic system-level tools available today. In fact, the operation of mapping the system-level functional description to the actual

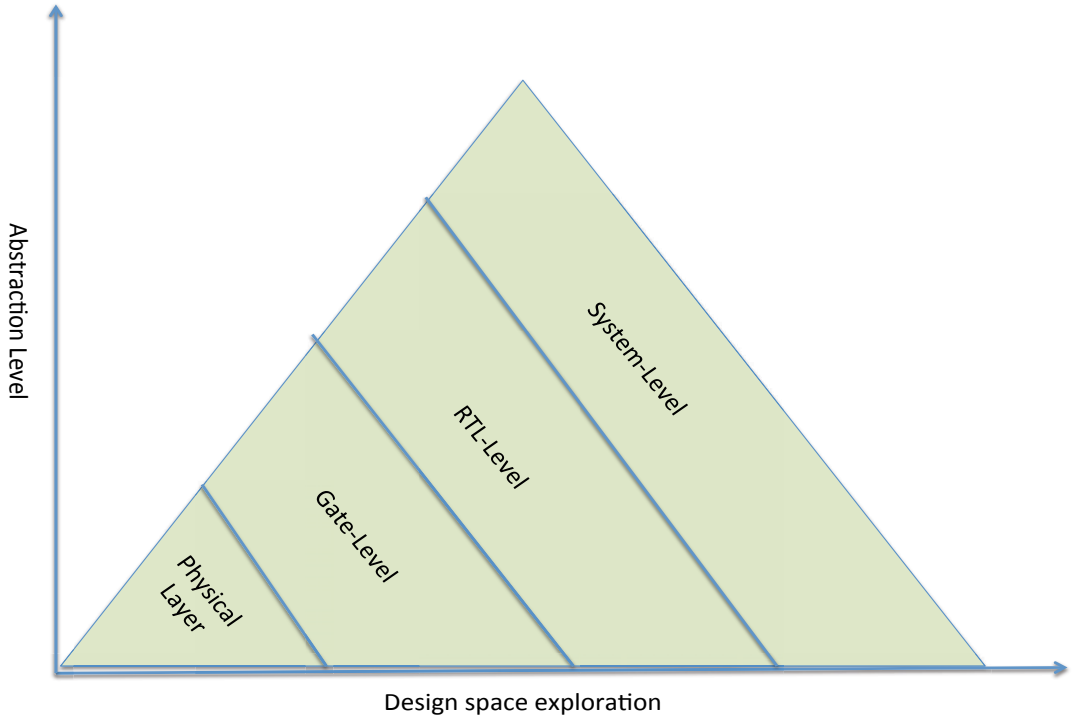

Figure 2.2: Broadness of architectural solutions according to the abstraction level

architecture is still largely done manually. The decision making approach used by system designers has been mostly relying on their acquired experience, on comparison with previous designs and on rules of thumb. However, while this approach can still work with small/medium-size systems, its application to nowadays more and more complex systems has become unrealistic and the need for a more systematic and accurate approach has become a necessity. Cycle-Accurate (CA) and Transaction Level Modeling (TLM) have appeared at the beginning of the last decade as a descriptive based approach raising the abstraction level above RTL in order to reduce simulation time and development effort.

In late 1990s, many large companies started to develop their own models while research institutes and EDA start-ups were proposing a variety of modeling languages. The initial requests were to have cycle-accurate C or C++ models from certain who believed that it was the right way to get simulations running at least one order of magnitude faster than RTL models in VHDL or Verilog. It

soon became obvious that cycle-accurate modeling had several drawbacks [46]. First, the modeling effort was close to the one of creating synthesizable RTL models. It was due to fact that the model complexity was too close to RTL. The only gain was that such models had no synthesis-related constraints [46]. In addition, the RTL was still the reference due to immature synthesis tools. It led to iterations of the C++ model trying to keep in line with the RTL model of the IP under design. Introducing any specification change in the C++ model during the design was almost as long as doing so in the RTL model. The cycle-accurate modeling was actually leading to high costs. These models were not available to architects and were ready for software developers a little too late. Second, the simulation speed for a SoC model was ten times below the original objective. It was simulating at a few kHz compared to the several hundreds of Hz for RTL. Third, using specific languages or modeling optimizations to gain speed was actually locking the modeling team into a specific simulator supplier [46]. Fourth, during final RTL updates before tape-out, it was usually not possible to keep updating the cycle-accurate C++ model due to tight schedule. Thus, the cycle-accurate model was not fully consistent with the reference RTL at tape-out [46]. Normally, modeling engineers would be allocated to another project once the SoC was taped-out. The model would not be usable as a starting point for its next generation design because it was not consistent with the existing RTL and original modeling engineers were unavailable. For all these reasons, we were looking for an higher level of abstraction that would allow much quicker modeling than cycle-accuracy, yet be precise and fast enough for software developers to test the real embedded software using a standard language enabling reuse of models with a variety of simulator suppliers. Ideally, such models should also be usable for performance estimations with enough precision for SoC architects to make decisions.

In essence, TLM abstracts away the RTL details and models functionality and communication among the system modules. Communication is seen as an exchange of transactions between architectural resources. As a result, TLM has proved to be much faster than CA and RTL. The natural question that comes as a conclusion of the above discussion and also the motivation behind this thesis work is as follows: As it has been discussed that system-level estimation is essential

for a successful system-level design, how is it possible to implement a fast and accurate methodology for efficient system-level estimation.

2.3 System-level modeling languages

The terms of TLM defined in the last section can be attained through an appropriate electronic system level (ESL) modeling approach. The right candidate to do this job is a high-level programming language that is capable of developing not only a plain software program, but also of modeling electronic hardware at the conceptual level without describing the real implementation. The potential candidates include SystemC [91], SpecC [65], Hpascal [42], System verilog [121] and Hardware-C [49]. Specification at higher levels of abstraction is possible in environments such as SpecC. A unified and integrated approach to hardware-software co-design is possible if the hardware modeling description is based on the C/C++ languages that are popular in the software community. Hardware-C [49] is an example of such a proposal. SystemC is an emerging standard modeling platform based on C++ that addresses the issues discussed above, and supports design abstraction at the RTL, behavioral and system levels. Consisting of a class library and a simulation kernel, the language is an attempt at standardization of a C/C++ design methodology, and is supported by the Open SystemC Initiative (OSCI), a consortium of a wide range of system houses, semiconductor companies, IP providers, embedded software developers, and design automation tool vendors. Apart from the modeling benefits available in C++ such as data abstraction, modularity, and object orientation, the advantages of SystemC include the establishment of a common design environment consisting of C++ libraries, models and tools, thereby setting up a foundation for hardware-software co-design; the ability to exchange IP easily and efficiently; and the ability to reuse test benches across different levels of modeling abstraction.

In the upcoming sections, we will go through SystemC/TLM used for designing tools at system-level and also advantages of using SystemC/TLM.

2.3.1 SystemC and Transaction Level Modeling 2.0 kit (TLM)

In the recent years, SystemC/TLM [91] has become widely popular among the simulation based approaches in academics and also in industries. This huge popularity is also due to support of the main actors in the domain such as Synopys[9] and Cadence[13].

SystemC is based on a set of C++ classes and provides an event-driven simulation kernel [48]. These facilities enable a designer to simulate concurrent processes, each described using plain C++ syntax. SystemC processes can communicate in a simulated real-time environment, using signals of all the datatypes offered by C++, some additional ones offered by the SystemC library, as well as user defined. In certain respects, SystemC deliberately mimics the hardware description languages VHDL and Verilog, but is more aptly described as a system-level modeling language. In fact, it models hardware at a higher abstraction level than RTL and, in order to do this, it uses C++ as a programming language. Higher abstraction level means higher simulation speed but also less accuracy. The way SystemC trades-off these two important metrics has been characterized in [117]. SystemC is applied to system-level modeling, architectural exploration, performance modeling, software development, functional verification and high-level synthesis. SystemC is often associated with ESL design and with TLM.

Using SystemC as a vehicle to provide the Transaction Level Modeling (TLM) abstraction proved to be the key to the fairly fast deployment of this methodology. There was no issue of proprietary language support by only one CAD vendor or university. There was also no issue of making a purchase decision by the design manager for yet another costly design tool. Eventually, with the collaboration of ARM and Cadence Design Systems, a full-blown proposal was made to the Open SystemC Initiative (OSCI), under the name PV (Programmer View) and PVT (Programmer View Timed). Indeed Programmer View clearly reflects the intent of this new abstraction level, which is to bridge the gap between the embedded software developer and the hardware architect. In 1999, companies like CoWare and Synopsys started to encouraged the Open SystemC Initiative (OSCI) in order to develop an HDL which could model hardware at a higher abstraction level

rather than widely used VHDL and Verilog. The main stimulus was to provide an improvement in the implementation and the simulation efficiency when compared to RTL, which has proved to be a bottle neck in modeling a system-level architecture. TLM tool kit was first introduced in 2000 [76]. TLM tool kit is a set of library function built on the top of an high level language, which is very often SystemC. In TLM, a transaction represents the data exchanged between the different system modules. As we said before similar to SpecC even in SystemC/TLM the computational component is separated from the communication component. For this purpose, TLM provide constructs to efficiently model the inter-module communication component, while the intra-module computational component is generally modeled at the functional/behavioral level. Standard routines have been implemented in TLM which model unidirectional versus bidirectional and blocking versus non-blocking communication. Communication is modeled using channels, interfaces and ports, which are objects provided by the underlying HLL. In the context of this thesis, TLM tool kit version 2.0 is used as a reference.

2.3.2 Advantages to use TLM

Based on what has been said so far, using TLM has some clear benefits:

- First, there is a speed-up in the implementation when compared to the traditional RTL and Cycle-Accurate (CA) approach. This speed-up comes from the fact that most of the RTL and CA details are abstracted away. In use, only the behavioral description is used instead of the computational block details. In [46], the author reports a speed-up of up to 10x and 7x when modeling in TLM compared to RTL and CA as shown in the Fig. 2.3.
- Second, regarding synchronization issues, SystemC and TLM have helped major designers. According to Yi et al. in [131], This issue is due to multiple simulation models may have to run together and synchronize to each other. By using SystemC , there is a homogenous environment for hardware/software co-simulation, thus replacing the considerable Inter-Process Communication (IPC) overhead with a light-weighted thread switch overhead.

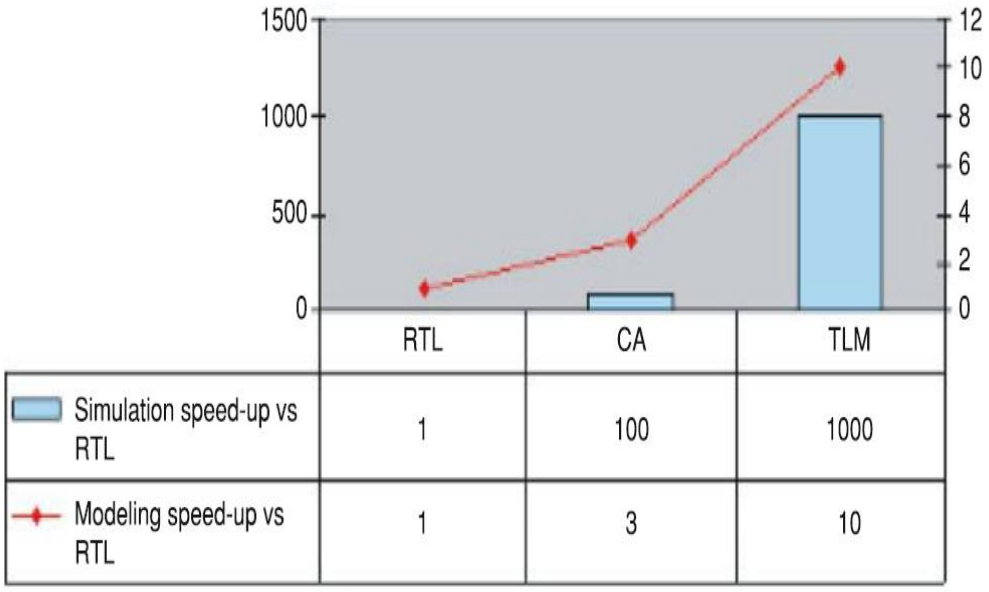

Figure 2.3: TLM speed-up in terms of modeling efforts and simulation speed : source [46]

- Third, compared to the traditional RTL and CA approach, TLM has higher simulation speed of 1000x and 900x respectively [46] as shown in the Fig. 2.3.
- Fourth, F. Ghenassia in [46] states that using the TLM design flow can allow a more efficient HW/SW co-design. This is shown in Fig. 2.4. In essence, the TLM flow would allow a concurrent development of hardware and software: the architectural TLM of the hardware infrastructure enables early software development and verification of hardware software interfaces. This is also a consequence of the fact that, in the classic design flow, software is usually implemented in C/C++, while hardware in VHDL/Verilog. In the TLM flow instead, both the software and hardware models are implemented in C/C++; thus concurrent testing becomes more feasible and the overall design time shorter.

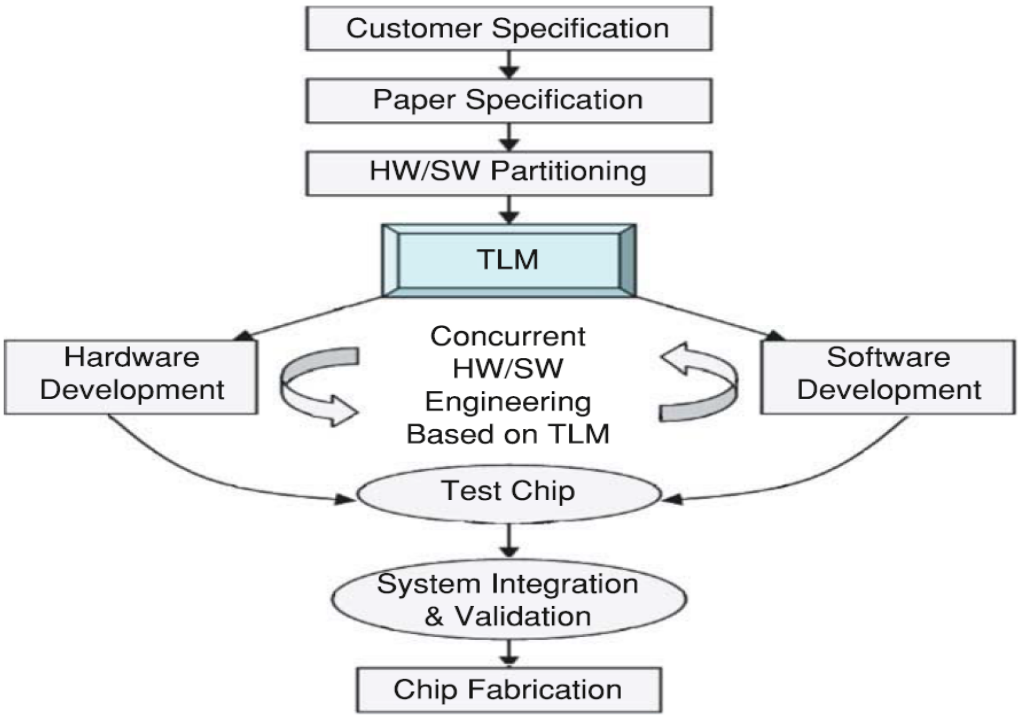

Figure 2.4: TLM design flow : source [46]

- Finally, SystemC/TLM is considered the right modeling language from where High-Level Synthesis (HLS) should start. An example to substantiate this claim is the CtoS [12] tool sold by Cadence. Starting from a SystemC description, this tool outputs RTL code. Inspite of this, HLS is far from being as mature as logic synthesis and therefore requires more research.

2.4 Power modeling approaches and tools

In this section, we give a detail view about the most recent different approaches for power modeling at different abstraction level. The power consumption models can be distinguished into two main categories:

- Low-Level or hardware level models.
- High-Level models.

Low-level or hardware models calculate the power from detailed electrical descriptions: circuit level, gate level and RTL. Among the existing tools for low-levels we can mention, SPICE [88] at the transistor level, Diesel [96] at the gate level and Petrol [95] at the RTL level, which deal with fine-grained activities. The low-level tools require a very long simulation time for large circuits and this makes them inapplicable for complex MPSoC. However, these tools provide a good accuracy, but impractical to implement in the early design flow as they require the knowledge of the circuit details. High-level models deal with instructions and functional units of the programs and without any electrical knowledge of the underlying architecture [35] as shown in Fig. 2.5.

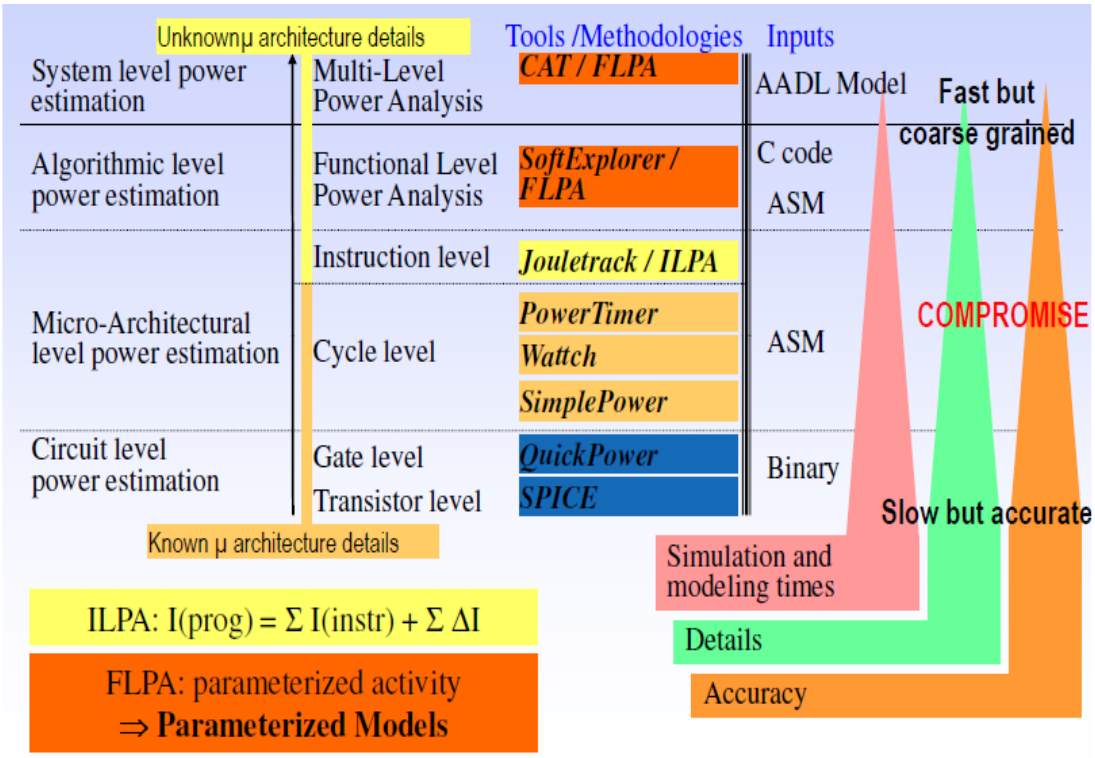

Figure 2.5: Power modeling methodologies and tools according the abstraction levels

2.4.1 Low-level estimation techniques

The efficiency of the power simulator is influenced by both accuracy of the estimation and speed of the simulator. In this section, we will go through few frequently used power modeling techniques at lower levels. The low-level power consumption estimation techniques cover a wide range of abstractions level such as the:

- Circuit/Transistor level
- Logic gate level.
- RTL.
- Architectural level.

2.4.1.1 Circuit-level

Here the microprocessor is represented in terms of transistors and nets which are extremely complex. It also requires to undergo all the steps in the design flow and the layout, routing and parameter extraction. This is not feasible due to the non availability of the technologies and tools. Moreover, the circuit-level of the system uses component models which are based on linear differential equations and works in continuous time domain. This suggest that a simple simulation for a small set of transistors requires a large amount of time which is not always available in this fast moving industry and not practical [31]. In an early attempt to build a low-level power consumption simulator, PowerMil [54] was introduced. This tool is used for simulating current and the power characteristic in VLSI circuits. It is also capable of simulating detailed current behavior in modern deep sub-micron CMOS circuits, including sophisticated circuitries such as sense-amplifiers, with speed and capacity approaching conventional gate level simulators [85], [50].

2.4.1.2 Gate-level

In this subsection, the description about the method to estimate the power consumption based on gate-level description have been given. The main advantage of this methodology compared to the previous methodology is: here the simulation is event-driven and takes place in a discrete time domain which considerably re-

duces the computational complexity, without any significant loss of accuracy [31]. Chou et al. [39] present an accurate estimation of signal activity at the internal nodes of sequential logic circuits. The power consumption estimation in [39]. is a Monte Carlo based approach that take spatial and temporal correlations of logic signals into consideration. These tools [51] generally consist of n-dimensional table to estimate the power consumed in a circuit for a given statistics, where n represents different variables/components capturing the relationship of power and dependent variables such as input probability, transition density etc. In paper [51], authors present an automation methodology, in which, such a tables are automatically generated. Three variants in their model are average input signal probability, average input transition density, and average output zero-delay transition density. For characterization purposes they assume that input nodes are output of latches or flip-flops and make at most one transition per clock cycle. Also, sequential design is single clock system and clock skew is ignored in their analysis hence all the inputs switch simultaneously.

2.4.1.3 RTL

Most of the RTL designs are described as a collection of blocks and interconnections. The blocks, sometimes referred to as macros, are adders, registers, multiplexers and so on, while the interconnections are simply nets or group of nets. Most of the tools presented in the literature follow similar pattern like the power properties of the block can be derived from an analysis of the block isolated from a design, under defined conditions. This gives to the main factor influencing the power consumption model of a macro is the input statistics [31].

Currently, almost all the research in RTL power estimation is based on empirical methods that measure the power consumption of existing implementations and produce models from those measurements. Thus this approach is totally in contrast to the information-theoretic measures of activity to estimate power [77], [104]. There is another approach which is widely used by RTL designers is based on measurement for estimating the power consumption of data-path functional units. Chau et al. [100] introduced a fixed-activity micro-modeling strategy called Power Factor Approximation (PFA) method. The power models are character-

ized in terms of complexity parameters and a PFA proportional constant. Similar works were also done by Kumar et al. [84] and Liu et al. [72]. Above listed approach has a clear problem that the inputs do not affect the switching activity of a hardware block. In order to overcome this problem, activity-sensitive empirical power models were developed. These schemes are based on predictable input signal statistics; an example is the method proposed by Landman and Rabaey [67]. However the separate models built via this technique are quite accurate (10% to 15% error rate), but due to incorrect input statistics or the inability to correctly model the interaction this method is not feasible. The second technique, transition-sensitive power models, is based on input transitions rather than input statistics. This technique was proposed by Mehta et al. [78], assumes a power model is provided for each functional unit a table containing the power consumed for each input transition. Closely related input transitions and power patterns can be concentrated into clusters, thereby reducing the size of the table. Other researchers have also proposed similar macro-model based power estimation approaches [101], [64].

Bogliolo et al. [28] have proposed a methodology for creating power macro-models based on linear regressions but their flow is specific to the structural RTL macros and power estimation is done at the gate-level. Their analysis is restricted to structural RTL representation whose leaf components are combinational logic blocks. This approach is based on: first, off-line characterization in which they compute the power of the RTL macro based on certain tests and second, on-line characterization, in which they do it adaptively for error minimization. Their approach utilizes all the inputs, outputs, transition functions of inputs and outputs on the successive cycles, and then they interpolate the relationship with energy consumption.

Potlapally et al. [99] present a technique in which they do cycle-accurate power macro modeling of the RTL component. This technique is based on the fact that RTL components exhibit different power behavior for different input scenarios. They create power macro model for each of these behaviors also known as power modes. Their framework chooses the appropriate power mode from the input trace in each cycle and then apply power macro-modeling technique discussed by Bogliolo et al. to get an estimate on power numbers. The technique

discussed in [99] is limited to the typical average power estimation scenarios and covers non-trivial scenarios as well but the estimation speed is very slow.

2.4.1.4 Architectural-level estimations

There are many architectural power simulator available in the industry and also in the research faculties. Architectural power simulator employ a combination of lower abstraction power consumption models. These simulators derive power estimates from the analysis of circuit activity induced by the application programmes during each cycle and from detailed capacitive models for the components activated. One of major difference between these simulators are the estimation accuracy and speed. SimplePower tool [127], works with a transition-sensitive power model for the data-path functional unit. The SimplePower core accesses a table containing the switch capacitance for each input transition of the functional unit exercised. Fig. 2.6 and Fig. 2.7 show SimplePower's compilation and simulation framework. The use of a transition-sensitive approach has both design challenges as well as performance concerns during simulation. The first concern is that the construction of these tables is time consuming. Unfortunately, the size of this table grows exponentially with the size of the inputs. The table construction problem can be addressed by partitioning and clustering mechanisms. Further, not all tables grow exponentially with the number of inputs. For example, consider a bit-independent functional unit such as a pipeline register where the operation of each bit slice does not depend on the values of other bit slices. In this case, the only switch capacitance table needed is a small table for a one-bit slice. The total power consumed by the module can be calculated by summing the power consumed by each bit transition. Another concern about this technique is its performance cost of the lookup table for each component access in a cycle.

In order to overcome this concern, simulators like SoftWatt [53] and Skyeye [38] are proposed. These simulators use a simple fixed-activity model for the functional units and only track the number of accesses to a specific component and utilize an average capacity value to estimate the power consumed. In contrast to the datapath components that utilize a transition-sensitive approach, these models estimate the power consumed per access and do not accommodate the

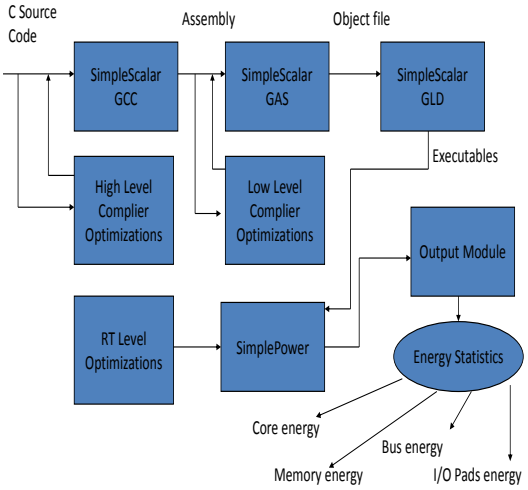

Figure 2.6: Complier of SimplePower tool

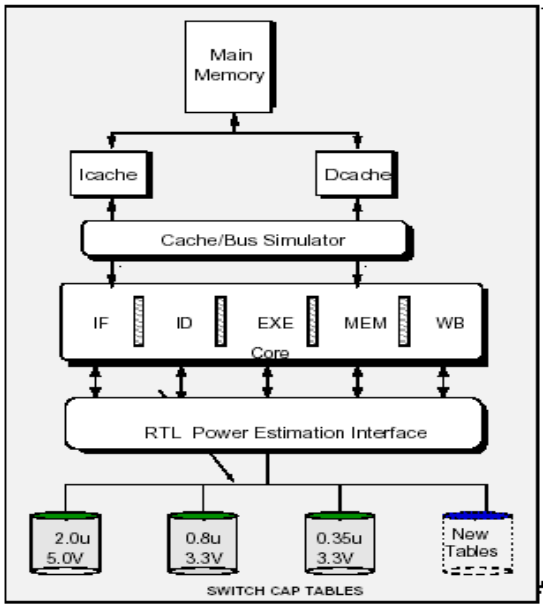

Figure 2.7: SimplePower simulator

power differences found in sequences of accesses.

One of the most widely used another tool in architectural domain is Wattch [32]. Wattch tool is used for superscalar and out-of-order processor. The base infrastructure is offered by SimpleScaler [34] for this tool. SimpleScaler carries out fast, fexible and accurate simulation of modern processors that implement a derivative of MIPS architecture. In addition, it also supports detailed cycle-accurate information for all models, including datapath elements, memory and Content Addressable Memory(CAM) arrays, control logic, and clock distribution network. Wattch uses activity-driven, parametrisable power models, and it displayed an accuracy better than 10% when tested on three different architectures. There are other approaches to evaluate energy estimates at the architectural -level which exploits the correlation between the performance and energy metrics. These methods [103], [60] uses the performance counters present in many processor architectures to provide runtime energy estimation [94]. While providing excellent accuracy; low-level power estimation methodologies are slow and impractical for analyzing the power consumption at an early design stage. Moreover, these methodologies require the availability of lower level circuit details or

a complete HDL design of the targeted processor, which is not available for most of commercial off-the-shelf processors.

SEAS [24] presents a system and framework for analysing SoCs in early design stage. Power analysis in this system works at a granularity of a processor cores, where pre-characterized data for power is utilized based on the power state of the design. Power states of the cores are high level states based on the workload such as active, idle, sleep states in today's processors. By utilizing these high level states of the SoC an early power estimation can be performed which is more efficient and accurate than the traditional spread sheet based approach.

Shin et al. have proposed a methodology [120] for power estimation of operation modes but their analysis is done at logic-level and proposes a way to create power models based on the switching frequencies. [27], [29], [52], [79], [86] utilize the similar approach for power estimation purposes and provide various accuracy and efficiency trade-offs based on the quality of inputs and power modeling. Power estimation accuracy can be significantly increased but generally it impacts the efficiency of power estimation procedure.

2.4.2 High-level estimation techniques

In the recent past, the need for high-level power estimation simulators has been increased which allows an early DSE from the power consumption perspective. The widely used high-level power estimation methodologies are classified into two main categories, Instruction Level Power Analysis (ILPA) and Functional Level Power Analysis (FLPA).

2.4.2.1 Instruction Level Power Analysis (ILPA)

ILPA power modeling methodology for individual processors was proposed by V.Tiwari et al [126]. In this methodology, they propose that by measuring the current drawn by the processor as it repeatedly executes distinct instructions or distinct instruction sequences, it is possible to obtain most of the information that is required to evaluate the power consumption of a program for the processor under test. The processor under test here is Intel DX486. Power model is modeled as base cost of each instruction and its inter-instruction overheads that depend

on next instruction. The base cost of an instruction can be considered as the cost associated with the basic processing needed to execute the instruction. There are some inter-instruction effects which comes into play and are not reflected in the total cost computed solely from base cost. The different effects are:

- Circuit state: switching activity of the system relies on the current inputs and previous state. In other words the difference between the bit pattern of two successive instructions.
- Resource constraints: limitations lead to CPU stalls e.g. pipeline and write buffer stalls.
- Cache misses: This is another inter-instruction effect. The instruction timings listed in manuals give the cycle count assuming a cache hit. For a cache miss, a certain cycle penalty has to be added to the instruction execution time.

By using the similar mechanism power profiler was introduced by Nikoladies et al [114]. The input for this profiler is the trace file of the executed assembly instruction, which is generated by an appropriate processor simulator. In this profiler, they also take into account the base estimates, inter-instruction energy cost of the executed program and also the energy sensitive factors as well as the effect of pipeline stalls and flushes. The complexity in the measurement of the current is one of the biggest drawback in this methodology [115]. There was another approach to reduce the spatial complexity of instruction-level power models presented in [19]. In this approach, the inter-instruction effect has been measured by considering only the additional energy consumption observed when a generic instruction is executed after No-Operation (NOP) instruction. In order to improve ILPA, the instruction level power model was combined with gate level simulator by Sama et al [116]. In this approach, the power cost values were obtained through a power simulator rather than actual measurement; thus modeling is possible at design time and can be part of micro-architecture and/or instruction set architecture exploration. More researchers attempted to enhance the original Tiwari ILPA power consumption modeling technique as in [111], [78]. Eventhough there are plenty research with this ILPA available, there are some drawbacks which were never overcomed by them. One of these drawbacks is that

the number of current measurements is directly related to the number of instructions in the Instruction Set Architecture (ISA), and also the number of parallel instructions composing the very long instruction in the VLIW processor. The problem of instruction level power characterization of K-issue VLIW processor is O(N2K) where N is the number of instructions in the ISA and K is number of parallel instructions composing the VLIW [73]. Also they do not provide any insight on the instantaneous causes of power consumption within the processor core, which is seen as a black-box model. Moreover, the effect of varying data (as well as address) is ignored in the ILPA models, though this effect can be accounted by an additive factor [20].

2.4.2.2 Functional Level Power Analysis (FLPA)

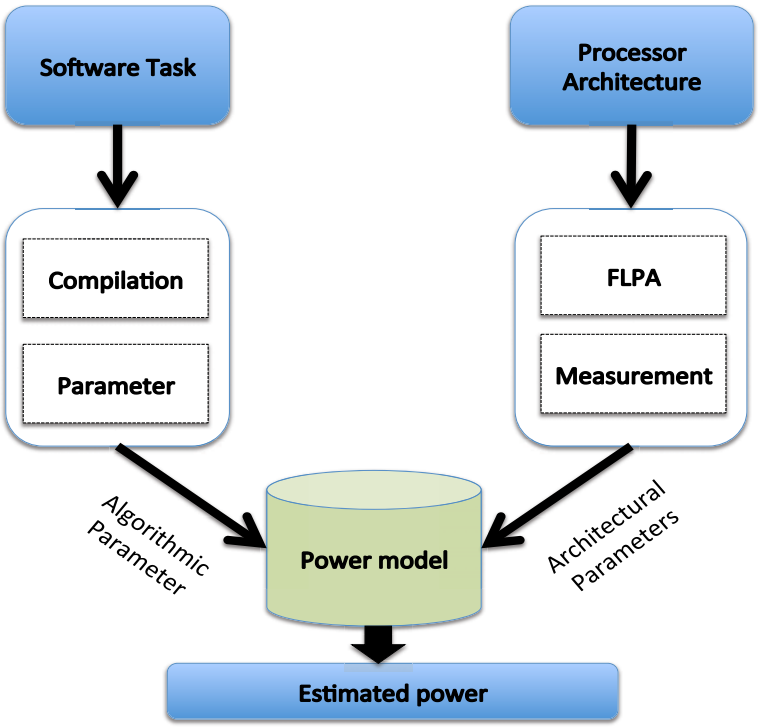

Figure 2.8: FLPA general methodology

To overcome the drawbacks of ILPA, Functional Level Power Analysis (FLPA) was introduced by J.Laurent et al in [57]. Fig. 2.8 gives the working table of power

estimation with the help of FLPA technique. The basic idea of this methodology relies on the identification of a set of functional blocks that influence the power consumption of the target component: like Processing Unit (PU), Instruction Management Unit (IMU), internal memory and others [118]. In addition to these parameters, there are some parameters that affect the power consumption of all functional blocks in the same manner such as operating frequency and word length of input data [68]. 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[40] has been developed and included in the recent toolbox CAT[112].

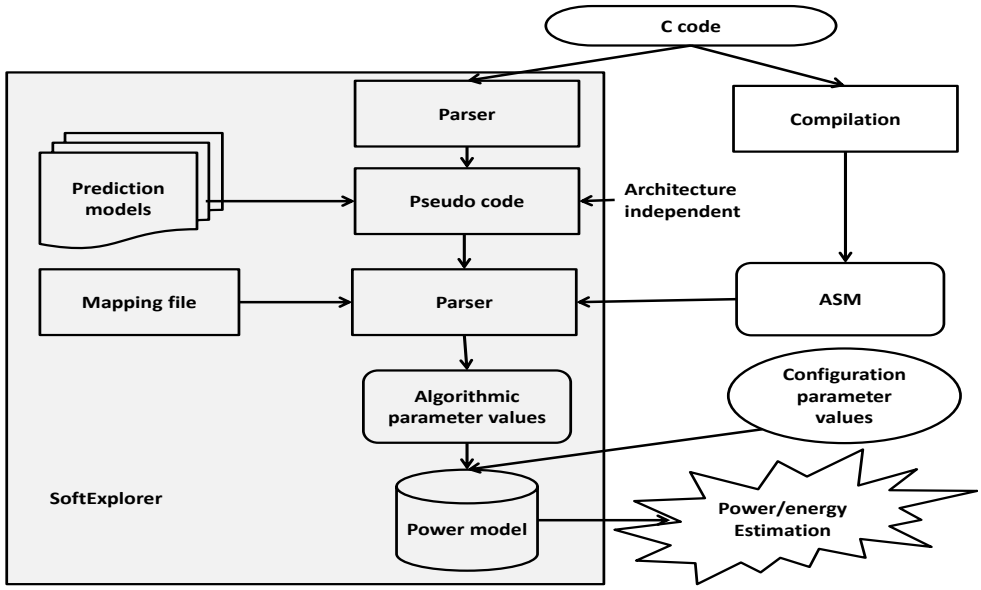

Figure 2.9: SoftExplorer power estimation flow : source [59]

Estimation flow of SoftExplorer is shown in the Fig. 2.9. The entry point of this tool can be both the assembly code generated by the compiler (or directly the ASM code written by the programmer) or directly the C code. This tool gives the global and local (for each loop) consumption and also the consumption repartition. It can also give the energy consumption of the external memory and

the energy consumption repartition between the processor and the external memory (only for the C6x model with the ASM estimation). SoftExplorer analyses the ASM code or the C code and computes the algorithmic parameters of the power model. Then, it uses these parameters to determine the power and energy consumption of the algorithm. The maximum error between the estimation and the physical measures is less than 4% for the ASM estimation (for both the power and energy) and less than 8% for the power about 20% for the energy at the C level. The estimation time is about few seconds (for example, the estimation time for a MPEG-2 encoder is 2 seconds). It also includes a library of power models for simple to complex processors. Only a static analysis of the code, or a rapid profiling is necessary to determine the input parameters for the power models. The functional level power modeling approach is applicable to all types of processor architectures. Furthermore, FLPA modeling can be applied to a processor with moderate effort and no detailed knowledge of the processors architecture is necessary [74]. However, when complex hardware or software components are involved, some parameters may be difficult to determine with precision. This lack of precision may have a non-negligible impact on the final estimation accuracy. In the context of this work, we have used this methodology to generate power models which later fed into the system-level environment and we will compare our tool with the SoftExplorer. Indeed, the FLPA methodology was applied to the building of power-models for different components in an embedded system : the approach have been recently extended by considering, in heterogeneous architectures, the overhead due to operating system services and the use of peripherals, in addition to the intrinsic consumption of applications [112] [90].

2.5 Simulation based estimation tools

From the name, we are able to conclude that these tools rely on simulation for the results. By using simulation, we are able to trace the behaviour of the system and for the given set of input stimuli. Simulation-based approaches are therefore suitable for non-deterministic system behavior and their results are generally representative of the average-case scenario. Significant research efforts have been devoted to develop tools for evaluating power consumption, energy

consumption and DSE at system-level in embedded system design. As a result these research efforts, several enviroments were proposed like SPADE [71], MESH [92], MILAN [81], GRACE++ [61] and Metropolis [21]. The work presented here could be considered as reciprocal to these environments. As discussed in the 2.3.1, SystemC offers a hardware/software co-modeling and co-simulation environment. When compared to traditional co-simulation tools, SystemC/TLM offers faster simulation, as the abstraction level has been elevated from RTL to transaction level. Another benefit of TLM is overheads associated with synchronization and communications between the simulation models can be significantly reduced [30]. There are few simulation approaches which will form a good literature review are presented below.

2.5.1 SPADE

SPADE [71] abbreviation for System-level Performance Analysis and Design-space Exploration and presents an methodology at system-level to explore signal processing processors. Even here the communication happens through channels which are connected to ports. Applications are modeled as a network of concurrent communicating processes, based on the model of computation provided by Kahn Process Networks. The work phenomenon of the spade is as follows, while carrying out the application, each process produces a trace of events which indirectly represents its workload. Then, the traces are transformed into an input for the corresponding architecture power model. Here the accuracy of the estimation is around 20 % when compared to RTL level modeling. Mapping and functional and architecture models are shown in Fig. 2.10. Artemis [98], [97] abbreviation for Architectures and Methods for Embedded Media Systems. The Artemis development environment for power and performance is heavily based on the SPADE framework. Artemis focuses on two challenges: first, developing a modeling and simulation environment for efficient design-space exploration of heterogeneous embedded systems; second, investigating the possibility of using reconfigurable embedded architectures such as FPGAs to give high performance for specific applications and limiting power consumption. Here it is stated that power estimation error are high.

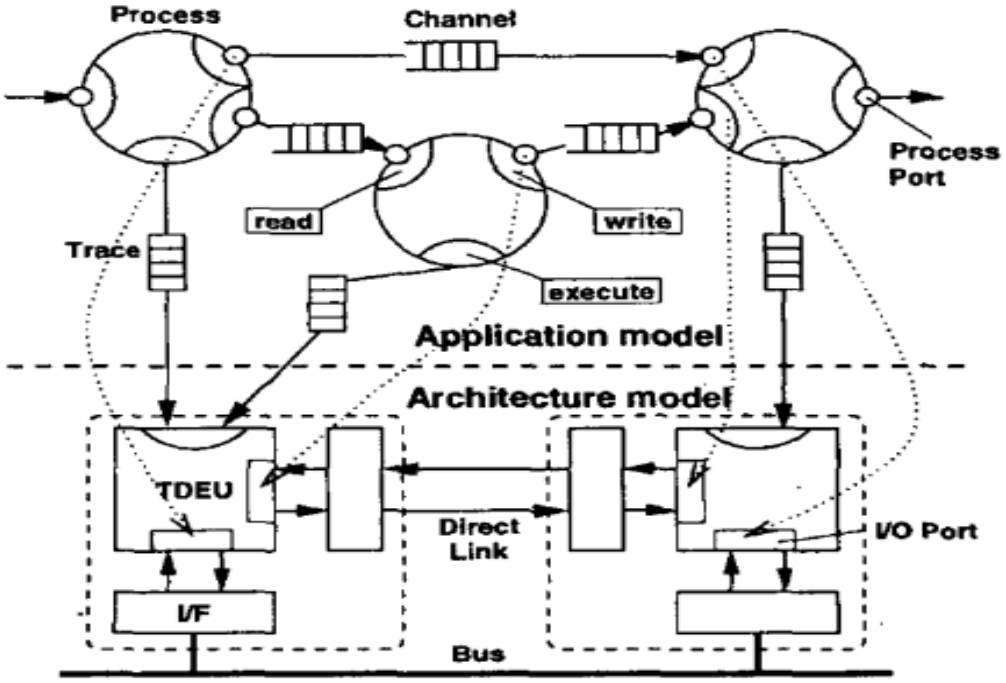

Figure 2.10: SPADE design flow : source [93]

2.5.2 Metropolis

Metropolis [22] gives an unified tool-set with an objective for embedded system development, which supports simulation, formal analysis and synthesis. As its said to be unified tool set, the authors promises that there is an considerable speed up and making it more efficient and also less error prone. Metropolis is designed to provide an infrastructure based on a model with precise semantics that remain general enough to support existing computation models and accommodate new ones. This metamodel can support not only functionality capture and analysis, but also architecture description and the mapping of functionality to architectural elements. Metropolis includes a standard API, which allows feeding the tool with inputs coming from any external tool. The Metropolis metamodel is a language, similar to SystemC, that specifies networks of concurrent objects. This metamodel can be used to represent function, architecture, mapping of the function on the architecture, and platforms.

Set of objects forms a function which is called as processes that concurrently carry out some actions while communicating with each other through ports. The interface methods are actually implemented in other objects called media, which are used to connect ports to each other. Channels are the SystemC equivalent of Metropolis media. The behavior of this network of processes is modeled as a set of executions, where each execution consists of a sequence of events. Events represent a program's entries or exits to some piece of code. Note that Metropolis is in many respects similar to SpecC and SystemC, in that they all rely on the concept of concurrent processes that communicate via channels.

Two aspects distinguish are identified in architectures modeing: the functionality that it has to implement and that implementation's efficiency. In the metamodel, we model functionality as a set of services that an architecture offers to the functional model. Services are just methods, bundled into interfaces. To represent an implementation's efficiency, we must model the cost of each service. We do so by first decomposing each service into a sequence of events, then annotating each event with a value representing the event's cost. This is done by annotating the cost of each atomic event executed within a process. So called quantity managers are used for this purpose. The decomposition of services into event sequences is done by using networks of media and processes, as is also done for the functional model (power). Here the power estimation accuracy varies between 10 to 25 % when compared to RTL models. Architecture networks often match the actual physical structure of the architecture.

There is a network called mapping network which takes care of mapping the functional model to the architectural model. Fig. 2.11 shows the mapping network, on the left side is the functional model and its corresponding architectural model on the right.

2.5.3 MILAN

MILAN [113], [83] stands for Model-based Integrated simuLAtioN. Its an framework to facilitate embedded system design and optimization. It's main focus is to provide a tool for energy-efficient design of signal-processing applications. The following steps will provide the working flow of the framework: application

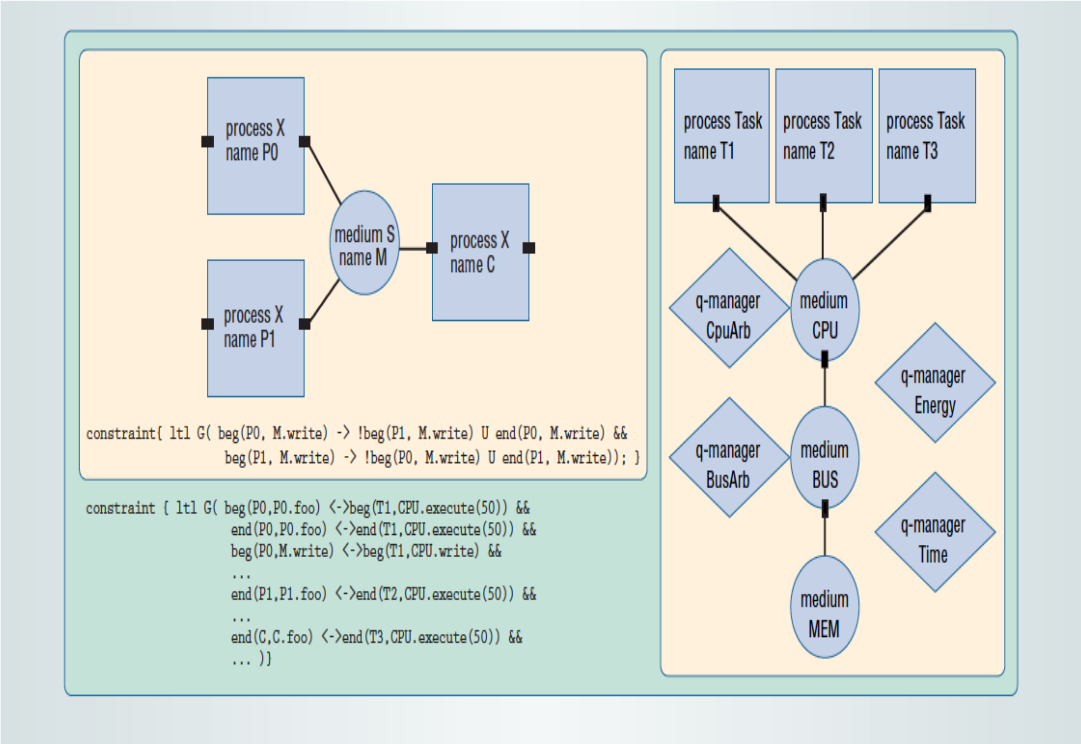

Figure 2.11: Mapping network : source [22]

models are first created using synchronous data flow (SDF) graphs. By using the functional simulator, functional simulation is enabled Functional simulation is enabled. Second, a model for the architectural resources is created, on which the user defines a set of performance constraints, in terms of latency and energy. Once these steps are completed, the user invokes the DSE tools. The predefined DSE tool in MILAN is called DESERT. This tool relies on Ordered Binary Decision Diagrams for constraint satisfaction. The output of DESERT is then passed to and evaluated by the High-level Performance Estimator (HiPerE) tool. This tool estimates system-level energy dissipation and latency and has an average error of 10%. Estimation is carried out at the task level abstraction, which confers the tool high speed. Both this tool and the actual architectural model are based on a so called General Model (GenM) [82]. The designs selected by HiPerE are then passed to lower-level simulator/estimator for the final design selection. Fig. 2.12 shows the complete MILAN flow. From left to right, it is possible to

identify the user's application and architectural resources model, together with the constraints definition; the usage of DESERT and of HiPerE is also shown.

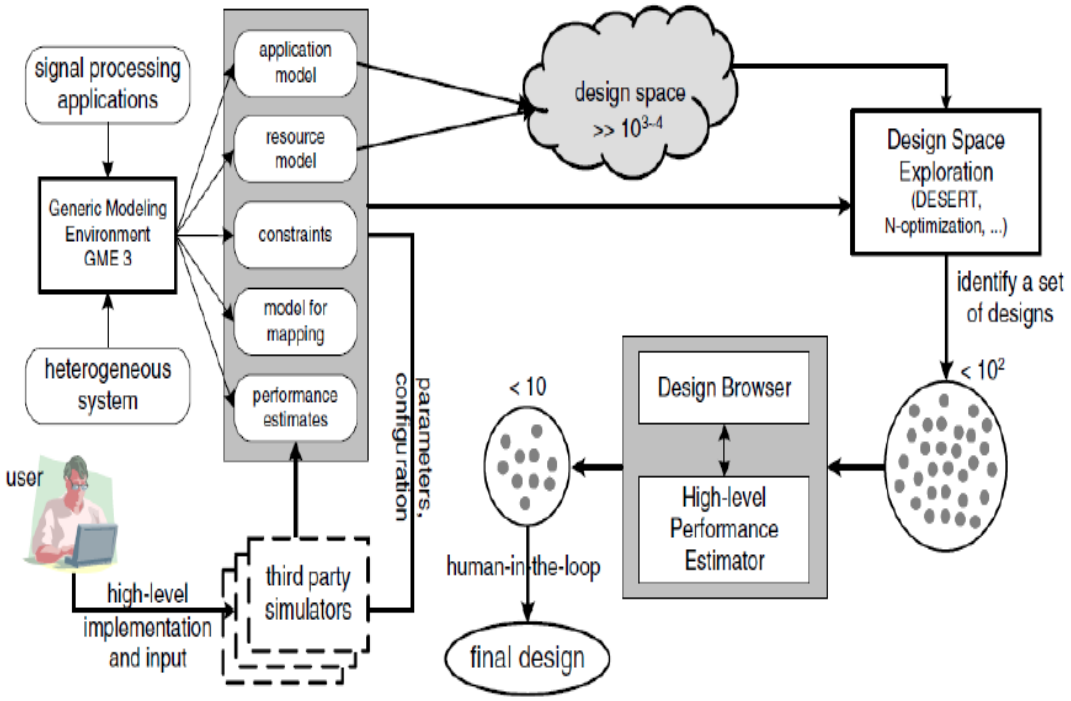

Figure 2.12: Milan framework : source [82]

2.5.4 MESH

MESH [92], [80] stands for Modeling Environment for Software and Hardware. It is defined by the authors as a thread-level simulator, as opposed to traditional instruction-set simulators. In this way, the authors want to emphasize that MESH increases the granularity for which estimation is carried out and thus the simulation speed can be much higher. MESH is a three-layer approach, which considers resources (hardware blocks), software and schedulers. Such layers are modeled by software threads on the evaluation host. Software threads are annotated with time budgets for the corresponding hardware elements. Such time budgets are extracted beforehand by estimation or profiling. Scheduler threads work as arbiters for the software threads. Power estimation capabilities are also implemented in

MESH. As far as microprocessors are concerned, the authors rely on the fact that compilers tend to produce quite regular instructions patterns, from which a power estimate can be extracted, that is representative of the average case.

As part of the ASSET project, Joshi et al. [58] propose a performance evaluation methodology for system-level design exploration. The application behavior is modeled as a set of statistical parameters, which are generated either by static analysis and profiling of the application, or by using some simulation framework like SimpleScalar [18]. Application parameters are independent of the architecture and their extraction is a one-time activity. The target architecture components are instead modeled using SystemC. A set of components on which to map the application models is taken from a library. Building these components also relies on probabilistic models, which are extracted by making a compromise between analytical and cycle-accurate simulation, and which only account for the interaction the component has with the outside, while the internal functionality is modeled as a delay. Note that, during the mapping phase, application parameters must also match with the parameters of the component the application is mapped to. For example, an application parameter could be the distribution of load instructions, whereas the corresponding architectural parameter could be the number of words to be loaded.

J. Kreku et al. in [62] also present a methodology for system-level design and performance evaluation. Their work relies on describing application workloads in UML and platform services in SystemC. The methodology is meant to enable early system-level performance modeling and evaluation through transaction-level simulation, which also allows timing information to be collected. Simulators have also been developed to investigate the properties of some processor micro-architectures. Such simulators are often cycle-accurate. Some examples are SimpleScalar [18] or SimOS [109]. Simics [75] is instead a virtualisation framework that allows entire platforms to be emulated.

2.5.5 StateC

Negri et al. [87] have proposed a power simulation framework of communication protocols (Bluetooth and 802.11) using StateC. StateC is used to model the hi-

erarchical state machines. Their flow is mainly targeted for simulator generation in SystemC. This flow is good for power exploration of protocol modeling but not presented on ASIC/FPGA design flow such as ours. They have mainly targeted wireless protocols in which relevant contribution to the power consumption of a node is due to the communication and not due to the datapath (computation) activity. Their learning phase requires execution of the real chip and can not easily be integrated to any ASIC/FPGA design flow. Also for practical purposes, it is difficult to create power model of the partial design or smaller part of the whole chip because current measured includes a lot of contribution from the other parts of the chip and isolation of the desired unit for power model purposes requires quite a lot of effort.

2.5.6 CAFD

In [132], authors attempt to lift power estimation to higher levels than the RTL, and their choice for high-level modeling was Cycle-Accurate Functional Description (CAFD) of the design. They create virtual components for each design block and attach them to the CAFD model of the design block, and compute the power consumption dynamically as the CAFD is simulated. Since this additional overhead to the CAFD simulation causes inefficiency, they also allow periodic turning off of some of the virtual components during some cycles of the simulation. During those cycles, they estimate power based on the history of the power consumption for the turned off components. So even though, the abstraction at which they estimate power is cycle accurate modeling level as ours, their power estimation is not based on regression based technique, and the simulation of CAFD is slower due to the overhead of virtual components. Caldari et al. presented a relative-power analysis methodology [36] for system-level models of the Advanced Micro-controller Bus Architecture (AMBA) and Advanced High-performance Bus (AHB) from ARM. It relies on creating macro-models from the knowledge of the possible implementations. Similarly, Bansal et al. presented a framework in [23], which uses the power-models of the components available at the system-level simulation stage by observing them at run time. It selects the most suitable power-model for each component by making efficiency and accuracy trade-offs.

In [66], the presented framework employs co-simulation techniques for power estimation with the capability of performing accuracy and efficiency trade-offs. They utilize multiple power estimation engines concurrently and have proposed several speed-up techniques to eliminate the overhead of co-simulation.

2.6 Analytical power estimation tools

The advantage of analytical methods over simulation-based methods is that they do not rely on an executable system model and, in general, their power estimation speed is much higher than the speed of their simulation-based counterpart. Nonetheless, analytical approaches often take into account the worst-case scenario and therefore they may be too pessimistic in certain circumstances. For this reason, analytical models are well suited to systems for which it makes sense to assume a deterministic or worst-case behavior, regardless of the input stimuli. Event stream-based models are an example of an analytical approach. In this case, estimation relies on evaluating the task execution on shared resources for event streams, like periodic events. Network calculus [11] and queuing theory are examples of how to use event-stream models for making estimation. The former has been applied to network processor design [122], [123], [47] and embedded real-time systems in general [37]. The latter has been used, among other things, for doing performance and power estimation and modeling contention in MPSoCs [15].

Another analytical method is to use spreadsheet way of prediction. Spreadsheets are very useful in the early stage of design process, when initial planning is going on and a lot of important decisions are being taken [128]. One of the biggest advantages of spreadsheet based analysis is that the user does not really need to learn any complex/sophisticated tool for taking design decisions. One of the basic application of spreadsheet is area estimation. Designers generally have a fair idea of the building blocks for a big design. He/She can easily get an estimate on area by using data sheets from intellectual property (IP) provider, library cell estimates, etc. Spreadsheet provides a capability to capture such information, which can be utilized for quick area estimation. Similarly, some decisions to control power can also be taken using spreadsheet based approach. Power budgeting

approaches using spreadsheets are very helpful for printed circuit board (PCB), power supplies, voltage regulators, heat sink, and cooling systems. Spreadsheet tools vary from utilizing excel sheets, word processors to Unified Modeling Language (UML) [110] etc. In industry, spreadsheet is being advocated by Field Programmable Gate Array (FPGA) vendors such as Xilinx [125], Altera [63]. Power analysis needs to be done very efficiently

2.7 Referenced tools

In this section, we will go through the different tools that have been used in this thesis as a reference with the proposed work.

Table 2.1: Tools used as references in this thesis

References	Name of the tools	Abstraction levels	Power model	Simulation	Speed
Laurent et al. [70]	SoftExplorer	Functional	FLPA	Code profiling	+++
Santhosh et al. [105]	Proposed PETS	Instruction accurate	FLPA	Full simulation	++
Santhosh et al. [108], [107], [106]	HSL	TLM	FLPA	Full simulation	+
Ye. et al. [130]	SimplePower	Cycle-accurate	RTL	Full simulation	-

Table 2.1 shows the different tools referenced in this thesis. SimplePower [129] tool relies on CA simulation technique. The power consumption of the main internal units is estimated using power macro-models, produced from lower-level characterizations in this case at RTL level. The contributions of the unit activities are calculated and added together during the execution of the program on the CA microarchitectural simulator. Though using CA simulators has allowed accurate power estimation, evaluation and simulation time are very significant for the off-the-shelf processor. To overcome this drawback the FLPA was proposed [68], which relies on the identification of a set of functional blocks that 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 [112] has been developed and included in the recent toolbox CAT [112]. It includes a library of power models for simple to complex processors. Only a static analysis of the code, or a rapid profiling is necessary to determine the input parameters for the power models. However, when complex hardware or software components are involved, some parameters may be difficult to determine with precision. This lack of precision may have a non-negligible impact on the final estimation accuracy. To overcome this drawback, we proposed a hybrid power estimation methodology (HSL) [108] by combining interpreted ISS with functional level power model but this methodology suffer in terms of speed as it uses the interpreted ISS for the simulation. In order to refine the value of sensible parameters with a reasonable delay, we propose to couple the OVPSim simulator with the functional level power models which offers us the reasonable trade-off between estimation speed and accuracy in [105].

2.8 Overview of the industrial virtual platform tools available

Today, there is a healthy growth in the market for virtual platform. A number of EDA vendors, such as Synopsys [10] and Carbon Design Systems [1], have gone to market with tools that create such virtual platforms, which comprise transaction-level models of the hardware.

With the recent acquisitions of CoWare and VaST, Synopsys has become an important provider of diverse tools for virtual platforms and software development, consistent with its advocacy role with respect to TLM 2.0. Innovator [6] is an integrated virtual platform development environment provided by Synopsys. It supports virtual platform assembly from SystemC/TLM 2.0 hardware models and software development on top of it. The environment includes the DesignWare System-Level Library, a huge library of transaction-level models for a rich set of components such as processors, memories, and peripherals, that can be extended by user-defined modules.

Platform Architect [14] is another tool used for virtual platform development, initially developed by CoWare and recently acquired by Synopsys. The tool

has some similarities with Synopsys Innovator, in the sense that it represents a graphical environment for platform development from existing TLM hardware components provided in a library. But, compared with Innovator, the components are at a low level of abstraction, and thus are more suitable for performance estimation and architecture exploration.

VaST, another recent acquisition of Synopsys, developed the tools CoMET and METeor [3]. CoMET is a system engineering environment which enables system architects to create and analyze platforms. With cycle-accurate modeling, CoMET produces meaningful quantitative results for both timing and power dissipation. Architects can address the optimum balance of speed, power and cost (size). CoMET is used during chip hardware development for co-verification of RTL along with software and other components modeled at the system-level. METeor is a software development environment which allows embedded software developers to create code using a virtual system prototype that runs at near real-time speeds on an off-the-shelf PC (personal computer). METeor thus forms a pure software implementation of a development board and its in-circuit emulator.

OVP (Open Virtual Platform) developed by Imperas, provides ultra-fast, instruction-accurate virtual platform models [5]. OVP is made up of three main components: APIs that enable modeling in C of a hardware component, a collection of free open-source processor and peripheral models, and the OVPsim simulator which executes these models.

Mentor already has a solid market position with Catapult C; this position has been enhanced with SystemC support [8].

Carbon Design Systems provides solutions to build cycle-accurate IP models for virtual platforms [1]. The generation of the IP models consists of compiling the IPs RTL implementation into a high-speed software model. This illustrates a bottom-up approach, which starts from RTL and goes to a higher level of abstraction as employed by virtual platforms.

The other approach for IP generation is top-down, as proposed by The MathWorks [7]. The EDA Simulator Link tool from The MathWorks provides glue to link the design of high-level algorithms with existing virtual platforms. It automatically generates SystemC components from high-level applications modeled in Simulink. The generated SystemC module has a TLM 2.0 standard interface,

so that it can be incorporated into a virtual platform which supports such an import. The SystemC generation also includes testbenches, so that the IP designer can verify the behavior of the generated TLM component with respect to the modeled functionality in Simulink.

CoFluent Studio is another tool which allows generation of SystemC transactional models from high-level UML (Unified Modeling Language) descriptions or DSL (Domain-Specific Language) descriptions [2]. Platforms are built by assembling generic models of universal components, like processors, integrated circuits, memories, buses, and interfaces. Each generic model provides variable parameters to easily adjust its behavior and performance characteristics. No instruction set simulators are used.

2.9 Positioning of methodologies

As we have seen in the previous section, there is significant research efforts have been devoted to develop power models for evaluating power and energy consumption at different abstraction levels in embedded system design. One of the straight forward power model development approach on processors is the physical-level power analysis methodology [43]. This approach is based on evaluating the switching activity of all circuit nodes of the processor architecture. The basic requirement for this methodology is the availability of the detailed description of the processor architecture on transistor-level, which is rarely available for the state-of-the-art processors. Architecture-level approach was introduced by [33] to reduce the computational effort by abstracted modeling of typical architecture elements such as registers, functional units or load/store queues. This methodology comes very handy in the development of high volume products such as microprocessors. In practical use there is lot of computation time which is the main disadvantage of this methodology. Another possibility for power estimation of processors is the so-called instruction-level power analysis (ILPA) [124]. This methodology is based on low-level simulations or physical measurement of the power consumption of each instruction out of the instruction set of a processor. By analysis of the assembler code of a task it is possible to estimate the specific power consumption of this program performed on a certain processor. The instruction-level power anal-

ysis methodology allows to cover such inter-instruction effects by measuring the energy consumption of groups of processor instructions, but the huge number of possible combinations makes this approach very complex [25]. A more attractive approach for power estimation is the Functional Level Power Analysis (FLPA) methodology. This methodology has been introduced in [102]. Furthermore, in [119] or [26] it could be shown that a good estimation accuracy can be achieved for typical digital signal processor architectures.

In the context of this thesis, first, we propose an efficient power modeling methodology based on functional level parameters. We choose functional level models due to their faster development rate and easy integration at the system-level. This is due to the development rate of the lower level models are slow and it relies on the technological parameters which cannot be retrieved from the system-level environment. Another important issue is to develop power models for complex MPSoC as it needs more effort and longer time. Thus, an extension of FLPA power modeling methodology is used in order to model complex embedded processor core, homogeneous and heterogeneous based platforms which feature a strong dependency of the corresponding power consumption on the performed algorithms. Second, we propose a hardware/software co-simulation environment. The co-simulation environment starts from usual RTL and extends upto the algorithmic level. The objective here is to unify the hardware and software design and to offer a rapid system-level prototyping. In the recent years, there have been lots of researches centred around the software/hardware co-simulation issue as the conventional RTL and CA tools cannot adequately support the complexity of future MPSoC since they are too slow for a meaningful execution of the software. For this reason, we propose a novel approach for hardware/software co-simulation by coupling fast JIT simulation with TLM to tackle this issue. Finally, we combine both functional level power models and fast JIT/TLM approach to propose a hybrid power estimation methodology at system-level [45]. Based on this methodology, a standalone Power Estimation Tool at System-level (PETS) is proposed. PETS consists of two phases. First phase denoted for developing the power model for the system under test and second phase is for power estimation at system-level through co-simulation of hardware/software related to the different platforms. The activities for the power models developed in the first

phase are collected through the simulation. PETS is similar to the majority of the simulation-based approaches reviewed in the past sections, especially from those based on SystemC such as Metropolis, Milan, MESH, StateC and CAFD. In fact, many of these approaches carry out estimation through co-simulation of hardware and software. However, these tools rely on Instruction-Set Simulation (ISS) to simulate their applications where the simulation time is very large for a complex embedded applications. To reduce the simulation time without compromising on the accuracy of activities inside the embedded systems, particularly those incorporating MPSoC, our solution tends towards a virtual platform tool defined with Just-In-Time (JIT) technique, also known as dynamic translation instead of traditional ISS. This abstraction must be sufficient enough to verify the behavior of the system (applications deployed on the architecture) and also allows to measure the execution time and extract activities accurately needed by the power models.

2.10 Conclusion

In this chapter, we review the evolution and state-of-the-art in power consumption modeling and estimation methodologies that rely on the simulation-based and analytical-based. In general two main abstraction levels for power modeling are surveyed in this chapter. The low-level power modeling and estimation techniques cover the circuit-level, gate-level, RTL and the micro-architecture level. The high-level techniques can be divided into two categories, the ILPA and the FLPA. This survey leads us to the appropriate power estimation technique for complex processors based platforms. Second, we go through different abstraction levels software and hardware based co-simulation, with a special focus on the recent attempts to evaluate the impact of different system-level approaches on the power consumption usage of a complex embedded processors based platforms. Finally, we outline the variety of existing research efforts that investigate the effect of applying simulation based estimation of energy and power consumption.

CHAPTER 3

POWER MODELING METHODOLOGY

3.1 Introduction

The typical system design flow has known a paradigm shift with the reuse of Intellectual Properties (IP). An application can be now developed in a very short time with the association of existing MPSoC platforms. Although this design methodology enhances the designer efficiency and reduces the time-to-market, its weak point remains the consideration of the power consumption metric. Current system power estimation is obtained after design place and route or developing power models at the RTL level. At these levels, when the power estimation exceeds the power budget, the designer must backtrack on architecture and algorithm parameters. This operation is time consuming and the power estimation are not always obvious. Moreover, this estimation is not useful to design a new system or extend it for a complex embedded system. To improve the design flow effectiveness, it is necessary to adapt new approaches for considering the power metric in the design flow.

In this chapter, a methodology based on real-board measurements is proposed for modeling the power consumption of nowadays embedded processor based plat-

forms with high-level parameters. This modeling methodology relies on the Functional Level Power Analysis (FLPA), which enables these models to fit into the power model library of a system-level environment provided by the proposed power estimation tool described in the later part of this thesis. This activity is time consuming, but it is justified as being a one-time activity that is used by multiple end-users. The methodology is applied to the Virtex-II Pro and OMAP platforms (3530 and 5912) as a case study and extended to the homogeneous and heterogeneous MPSoC platforms to ensure the scalability of the proposed methodology. There are four main highlights of this proposed methodology and they are: first, the accuracy level is high due to the characterization of the power models using the real-board measurements. Second, the developed power models can be used in the system-level environment in order to enable faster estimation. Third, this power modeling methodology can be extended to be applicable for MPSoC platforms and to have a reliable Design Space Exploration (DSE). Fourth, the development rate (in terms of days and modeling effort) of these power models is faster compared to other available state-of-the-art approaches.

This chapter is organized as follows: Section 3.2, gives a brief description about the modeled embedded platforms in this thesis. In Section 3.3, the FLPA methodology is presented. Section 3.4 shows the power consumption measurement environment. Section 3.5 presents the developed power models of different mono-processor based platforms. Section 3.6 presents the adaptation of our power modeling approach to homogeneous and heterogeneous MPSoC platforms. Section 3.7 validates the proposed power models accuracy against the real-board measurements.

3.2 Modeled platforms

In this section, we will introduce the different embedded processor based platforms chosen by the OPEN-PEOPLE project.

3.2.1 OMAP platforms

The Open Multimedia Applications Platform (OMAP) developed by Texas Instruments is a category of proprietary system on chips (SoCs) for portable and mobile multimedia applications. OMAP devices generally include a general-purpose ARM architecture processor core plus one or more specialized co-processors. In this thesis, we considered using the OMAP3530 with ARM Cortex-A8 processor and the OMAP5912 with ARM9 processor due to the capability of having high-performance processors to run complex embedded applications and intensive signal & media processing applications. The ARM Cortex-A8 provides a 4x performance improvement over ARM9 devices to achieve laptop like performance. Additionally, these new devices provide reduced power consumption for smaller, battery-operated, energy-efficient products.

3.2.1.1 ARM Cortex-A8

The ARM Cortex-A8[1] is 32-bit general purpose processor which is targeted for mobile platform. Both the hardware and instruction set architecture are based on the ARMv7 reference architecture. Like the ARMv7T, the ARM Cortex-A8 processor is based on a Harvard architecture. It consists of a Cortex-A8 Reduced Instruction Set Computer (RISC) processor core and separate instruction and data caches (16kB, 2-way set associative each) and combined L2 cache (256kB each). The Cortex-A8 is dual-issue superscalar, achieving roughly twice the instructions executed per clock cycle.

The block diagram of the ARM Cortex-A8 architecture is shown in the 3.1. The ARM Cortex-A8 (manufactured in a 0.18um technology with a core voltage of VDD = 1.6 V to 3.3 V and an adaptable core clock frequency of 125 to 720 MHz) RISC processor core consists of 13 stages pipeline, which is controlled by a 32-bit instruction word. Normally, all the instruction words are derived from the standard ARM instruction set. As a consequence, the source data of different instructions must be loaded separately into one or two source registers. The result is written back to a target register. Therefore, the instruction set can be divided into load/store and arithmetic instructions. However, branch and control

[1] http://www.arm.com/products/processors/cortex-a/cortex-a8.php

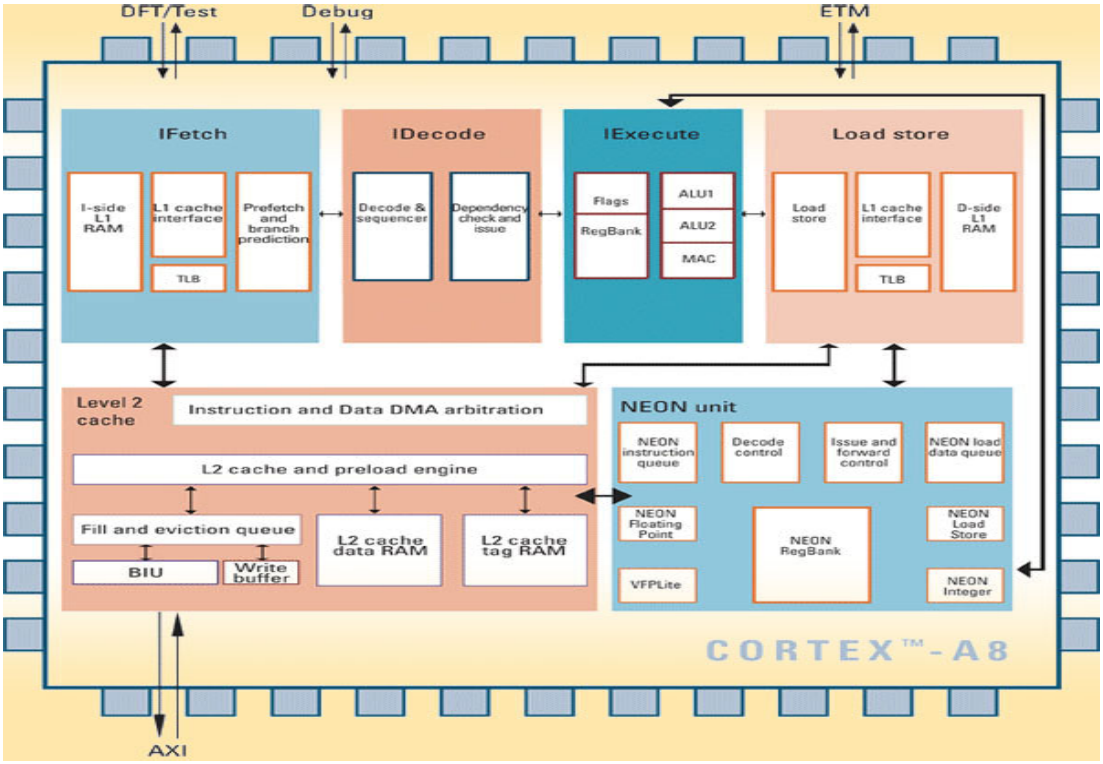

Figure 3.1: Block diagram of ARM Cortex-A8 processor : source [16]

instructions are supplied by the standard ARM instruction set. To improve the code density the Cortex-A8 processor core also features a dynamic instruction set exchange to the Thumb instruction set. These 16-bit instructions are compressed versions of a subset of the standard ARM instructions. The exchange is performed by dynamic decompression in the ARM Cortex-A8 pipeline. The platform used for power modeling purpose is OMAP3530[1]. The OMAP3530 contains an ARM Cortex A8 processor (16kB, 2-way set associative instruction and data caches and 256kB L2 cache).

3.2.1.2 ARM9

The ARM940T is a 32-bit general purpose processor[2] which was introduced to target for mobile platforms such as smart phones. Both hardware and instruction

[1]http://www.ti.com/product/omap3530
[2]http://www.arm.com/products/processors/classic/arm9/

set architecture are based on the ARMv4T reference architecture. The ARM940T consists of an ARM9TDMI reduced instruction set processor core and separate 4kB of instruction cache and data cache respectively.

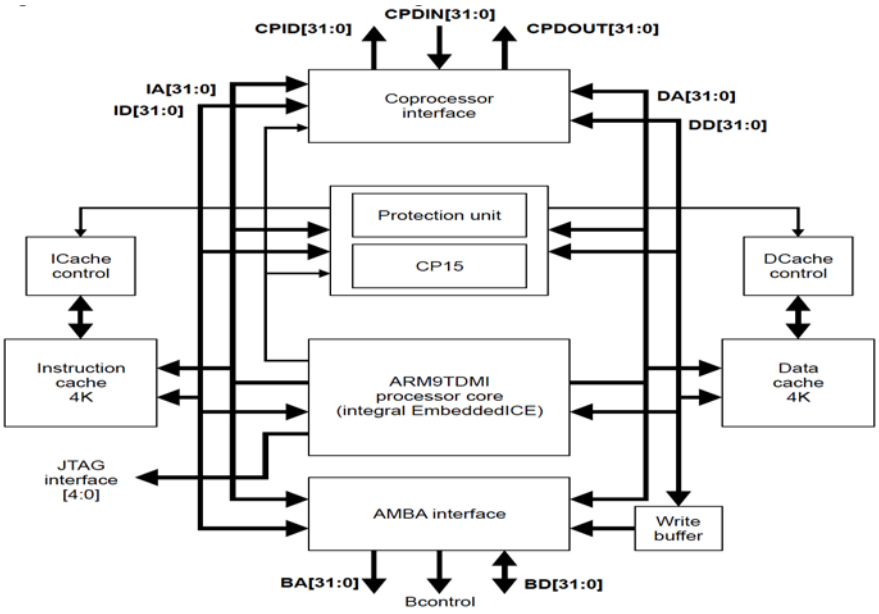

Figure 3.2: Block diagram of ARM9TDMI processor : source [17]

A block diagram of the ARM940T architecture is shown in 3.2. The ARM9TDMI RISC-processor core consists of a five stages pipeline, which is controlled by a 32-bit instruction word. Each instruction word is derived from the standard ARM instruction set. The standard ARM instruction set itself is based on a load/store architecture. For the purpose of power modeling, we used ARM Integrator Core Module featuring an ARM940T (manufactured in a 0.18um technology with a core voltage of VDD = 2.5 V and an adaptable core clock frequency of (12 - 160 MHz) has been deployed as reference platform. The platform used for power modeling purpose is OMAP5912 [1]. The OMAP5912 contains an ARM926EJ- S processor (16kB instruction cache and 8kB data cache).

[1]http://www.ti.com/product/omap5912

3.2.2 Xilinx Virtex-II Pro platform

The Xilinx Virtex-II Pro FPGA platform has features that include two hard-core processors (PowerPC), reconfigurable logic blocks, PCI-Express controllers, Ethernet MAC blocks and high-speed transceivers. In the context of this thesis, we used this platform to extend our power modeling methodology to homogeneous (two PowerPC) and heterogeneous MPSoC (two PowerPC and a hardware accelerator).

3.2.2.1 PowerPC 405

The PowerPC 405[1] CPU core is a 32-bit RISC processor. The PowerPC 405 CPU core is 90nm fourth technology generation. The PowerPC 405 CPU has an optimized interfaces to the CoreConnect bus structure provided for a high bandwidth and low latency system interface solution. It combines the performance and features of standalone microprocessors with the modularity, low power and small die area of embedded CPU cores. This processor has developed for hand-held devices to super computers.

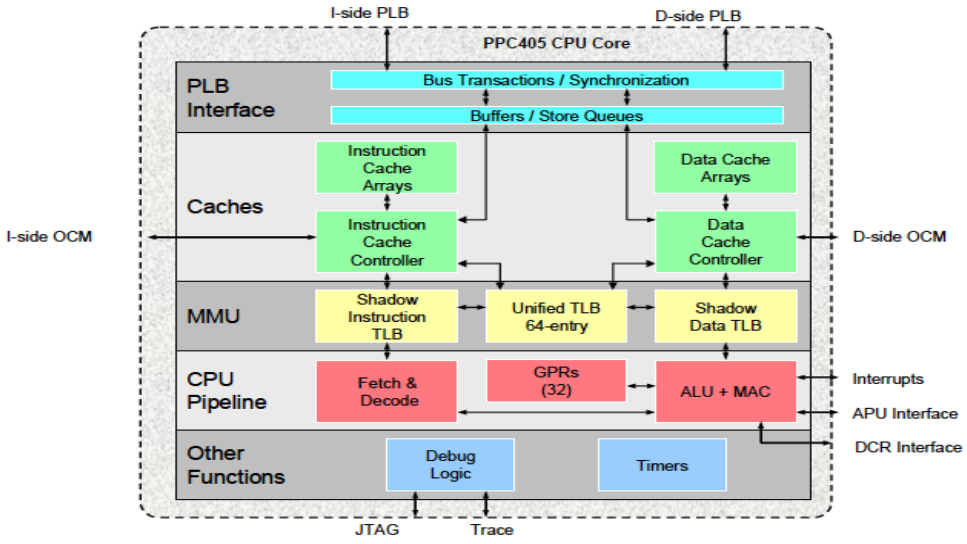

Figure 3.3: Block diagram of PowerPC processor : source [55]

[1]https://www-01.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_405_Embedded_Cores

The PowerPC 405 CPU core consists of a five stage single issue execution pipeline, a 32 element set of 32-bit general purpose registers (GPRs), 64-entry TLB memory management unit, L1 instruction and data caches of 16KB each, architecturally provided set of timers, a program model accessible hardware debug module and several system interfaces. Fig. 3.3 contains a high-level block diagram of the PowerPC 405 CPU core. The PowerPC 405 CPU operates on instructions in a five stages pipeline consisting of a fetch, decode, execute, write-back, and load write-back stage. The platform used for power modeling purpose is Xilinx Virtex-II Pro FPGA. The Virtex-II Pro FPGA contains two PowerPC 405 processors that have a 16kB, 2-way set associative instruction and data caches. Each processor has access to the on-chip memory (BRAM) and the off-chip memory (SDRAM) via the bus.

3.3 FLPA methodology

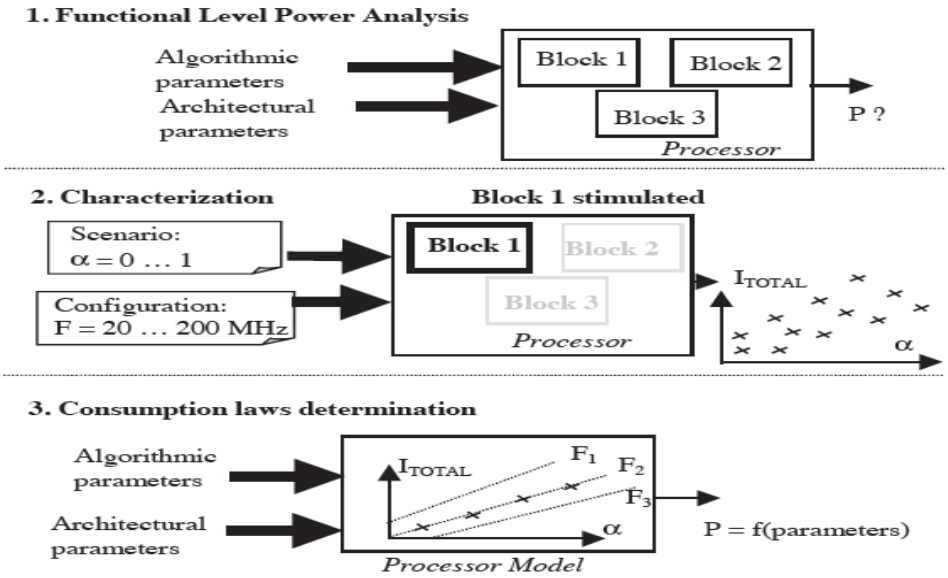

Figure 3.4: Functional Level Power Analysis (FLPA) general methodology : source [69]

As discussed in the previous chapter, creating power models is time-consuming and it is done only once in the flow of our power estimation methodology. This

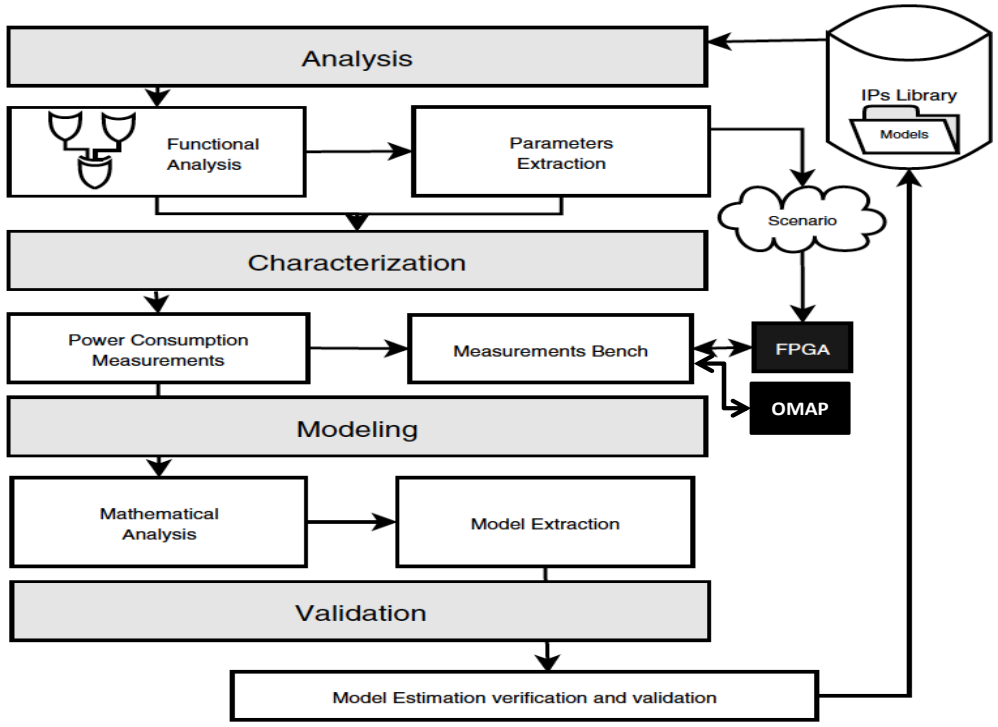

Figure 3.5: Power characterization and modeling methodology framework: source [41]

modeling activity relies on Functional Level Power Analysis (FLPA) methodology. The basic principle of the FLPA methodology is depicted in Fig. 3.4 [69]. The FLPA methodology allows to model the processor and peripheral power consumption with a set of high level parameters. In this approach, processor architecture is divided into different building blocks. Each block is separately stimulated with a particular set of assembler instructions in order to model their power consumption. The FLPA is based on physical measurements which guarantee realistic values with good accuracy. As shown on Fig. 3.5 [41], this methodology has four main parts, which are given below:

1. A functional analysis of the processor allows to determine the parameters which have a significant impact on the power consumption: relevant algorithmic and architectural parameters are then selected.
2. Then, the power characterization step explicits the power consumption be-

havior (obtained by measurements) when each parameter varies independently.

3. After curve fitting, the complete power model is obtained; it expresses the whole power consumption variations related to all the parameters with mathematical laws.
4. Finally, the accuracy of the model obtained is validated using different benchmarking applications comparing to real-board measurement.

In our contribution, this approach is extended for modeling both the memory systems and reconfigurable logic blocks power consumption on the selected platforms (OMAP and Virtex-II pro). Each memory systems and reconfigurable logic blocks will be considered as a black box with a specific granularity level. There are two types of parameters: *algorithmic parameters* that depend on the executed algorithm (typically the cache miss rate for a processor and area utilization for a hardware accelerator) and *architectural parameters* that depend on the component configuration set by the designer (typically the clock frequency). These sets of parameters are defined for a general class of embedded systems. Additional specific parameters can be identified for complex architectures.

Furthermore, the generated power model through the FLPA methodology can be easily adapted to the system-level environment, which is one of our main contribution to the system-level power estimation. In this chapter, the generation of the power model for the OMAP3530, OMAP5912 and Virtex-II Pro platforms are used as case studies.

3.4 Power measurement environment

An automatic power measurement bench was developed in order to reduce the time of each scenario measurements as shown in the Fig. 3.6. In order to measure the block power consumption, we use the Virtex-II Pro FPGA, OMAP3530 and OMAP5912. We successively implement the various block configurations and then placed a stimuli generator. The frequency generator is used to provide the various frequencies to both FPGAs. The logic analyzer allows to test the block functionality on chip. All the measurement equipments communicate with

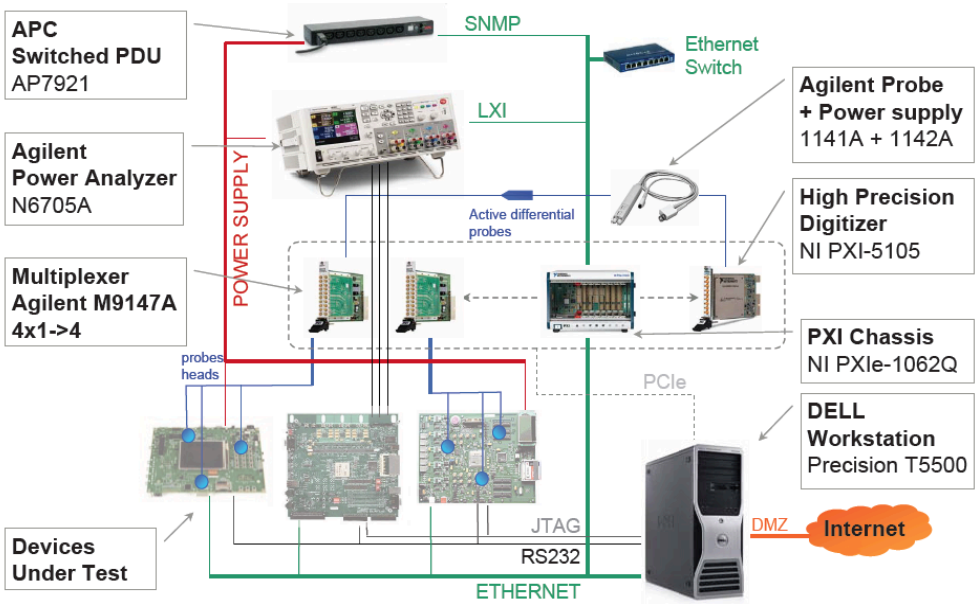

Figure 3.6: Fully automated test bench for current/voltage measurement

the host computer by General Purpose Interface Bus (GPIB). The measurement bench can be used by a distant computer through a LAN connection. The measurement procedure is done as follows: first, we program both FPGA and OMAP board by the host PC using JTAG link. Then, for various parameters values and clock frequencies, we record the supply current consumed by the FPGA core in which block are established. Finally, The power consumed is obtained as the product of the current measured with the FPGA core supply voltage.

3.4.1 Measurement environment for OMAP boards

Fig. 3.7 shows the measurement environment for OMAP3530 and OMAP5912 platforms composed of a power measuring instrument (Agilent LXI digitalizer) in a dedicated private network. The digitalizer accurately measures the static and dynamic current consumption across the resistors place. The Agilent Technologies L4532A is a high-resolution, standalone LXI digitizers. It offers 2 channels of simultaneous sampling at up to 20 MSa/s, with 16 bits of resolution. Inputs are isolated and can measure up to 250 V to handle the most demanding applications.

Fig. 3.8 and Fig. 3.9 show a simple way to take quick power measurements on

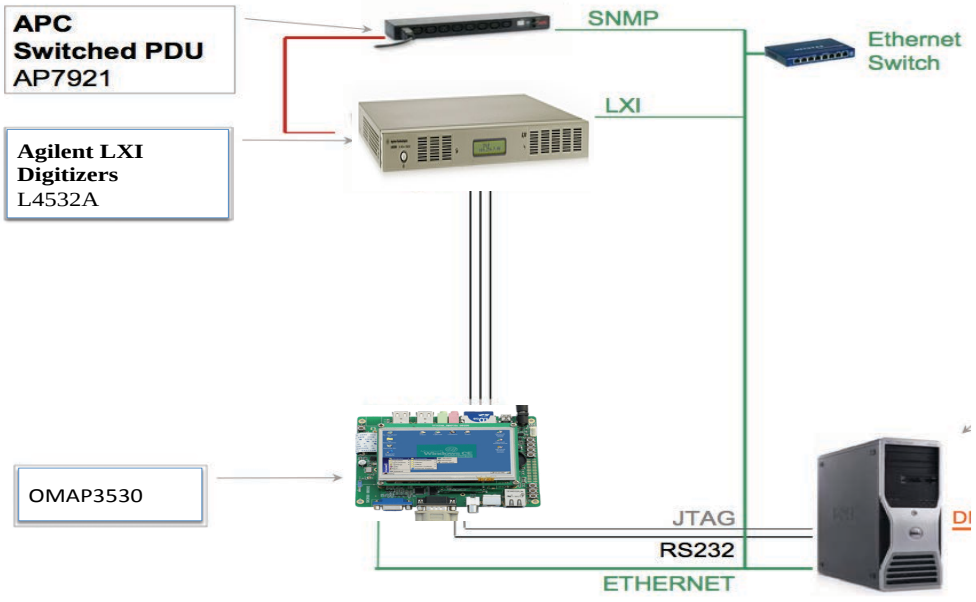

Figure 3.7: Measurement environment for OMAP3530 and OMAP5912

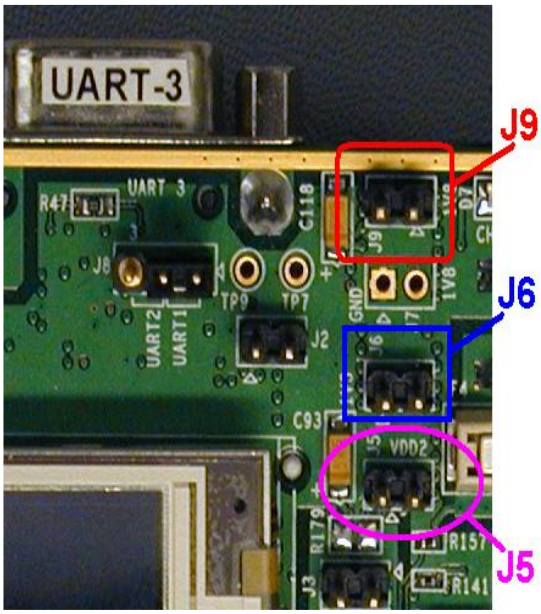

Figure 3.8: Jumpers for OMAP3530

Figure 3.9: Power measurement probes across the jumpers for OMAP3530

the OMAP3530 EValuation Module (EVM) with using a multimeter to measure the voltage drop across the jumpers J5, J6, and J9. Doing this will provide

an instantaneous power measurement and is a good representation of the power consumed in a scenario where the power profile is relatively flat. For scenarios where power changes drastically a multimeter might not present the full power picture. For such cases we will need a more sophisticated tool that can obtain and record several voltage readings over time. For tools with their own built-in current measuring shunt resistors, we could remove the resistors on the EVM.

The EVM has three separate power rails: processor rail, interconnects and peripherals rail, and the EVM 1.8 rail. Each can be measured by measuring the voltage on the specific jumper assigned to that particular rail. Fig. 3.8 and Fig. 3.9 will give an idea about the location of the jumpers for each rail on the board.

The following steps describe how to take the measurements on the board.

1. Measure voltage drop across the jumper pins.
2. Calculate the current being consumed by dividing the measured voltage drop by the resistance in parallel with the jumper pins.
3. Measure the voltage seen by the OMAP pin by measuring the voltage from the jumper pin tied to the OMAP side of the series resistor to ground on the board.
4. Calculate power by multiplying the current from step 2 by the voltage in step 3.

This procedure can be used to calculate the power for the different parts on OMAP3530 platform as shown in the Fig. 3.10:

- VDD1/vdd_{mpuiva} the processors power supply rail. Do this on J6.
- VDD2/vdd_{core} the interconnects and peripherals supply rail. Do this on J5.
- EVM 1.8V - note that this includes not just OMAP3530 1.8V rails, but also other devices on the EVM. Do this on J9.

3.4.2 Measurement environment for Virtex-II Pro FPGA

As shown in Fig.3.11, the measurement environment platform for Virtex-II Pro composed of a power measuring instrument (Agilent Power Analyzer N6705A DC) in a dedicated private network. The power analyzer provides power supplies to the devices and accurately measures the static and dynamic consumption. The

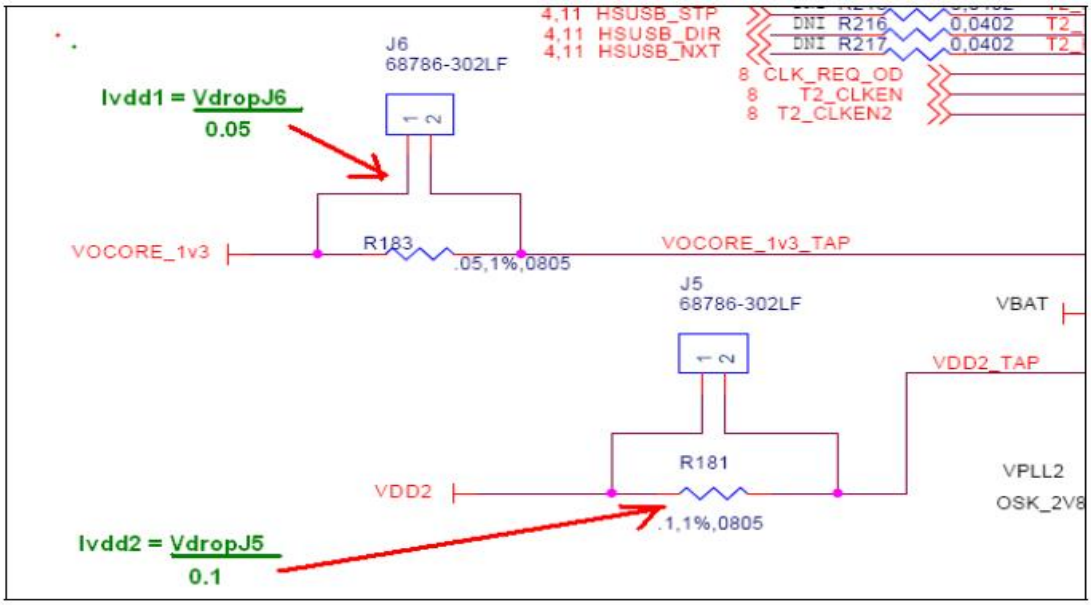

Figure 3.10: Power measurement jumpers for the OMAP3530 platform

Power Analyzer provides unrivalled productivity gains for sourcing and measuring DC voltage and current into the Design Under Test (DUT) by integrating up to 4 advanced power supplies with DMM, Scope, Arb, and Data Logger features. The N6705A eliminates the need to gather multiple pieces of equipment and create complex test setups including transducers (such as current probes and shunts) to measure current into your DUT. Devices configuration and power consumption measurements are automated thanks to a sophisticated benchmark server. For Virtex-II pro, we need three power source for the processor core (1.5 V) and for off-chip memories and peripherals (2.5 V)

3.5 Power models for uniprocessor based embedded platforms

In this section, the generation of the power model for embedded uniprocessor based platforms by using FLPA methodology will be elaborated in detail. This section will also stress on derivation of functional parameters (those parameters which can be retrieved from the system-level environment) affecting the power

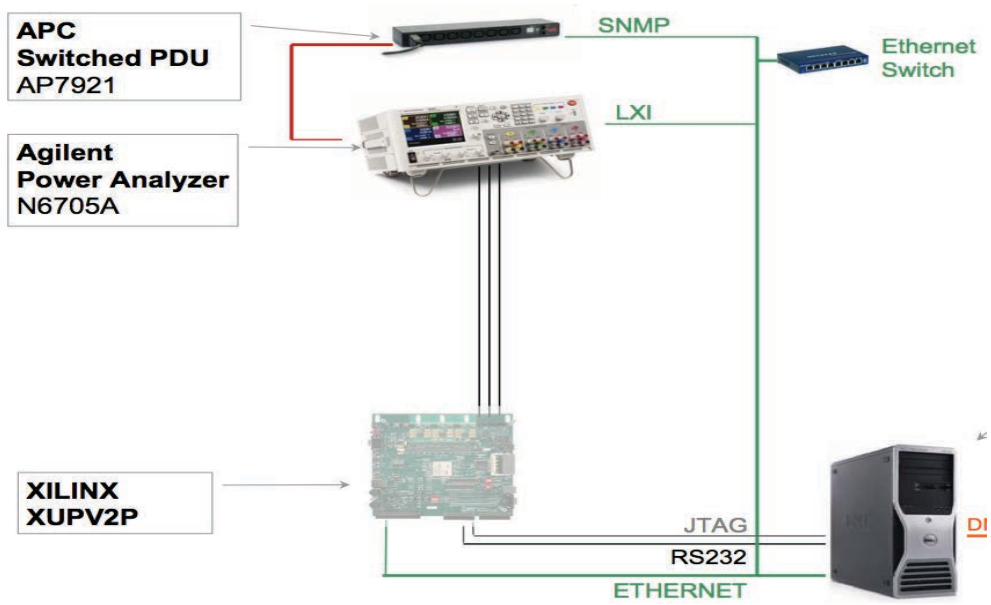

Figure 3.11: Measurement environment for Virtex-II Pro FPGA

consumption and the power model generation for the whole platform.

3.5.1 ARM Cortex-A8

As explained before, the FLPA methodology is used to generate a power model for ARM Cortex-A8. As a first step, the architecture is divided into different functional blocks such as the core clock unit, the memory unit, and the pipeline stage unit as shown in the Fig. 3.12. A parameter is denoted for each functional block of the processor and they are $\gamma 1$ and $\gamma 2$ respectively for L1 and L2 cache miss rates, Instruction Per Cycle (IPC) for the pipeline stage unit and f for the core clock unit. The second step is the characterization of the power model by varying the parameters. These variations are obtained by using some elementary assembly programs (called scenario) or built in test vectors elaborated to stimulate each block separately. In our work, characterization is performed by measurements on OMAP board. For example, we have shown the curve fitting in the Fig. 3.13 presenting the variation of the power consumption according to the IPC parameter using different benchmark programs. In the Fig. 3.13, seq_mul denotes sequential execution of multiply instruction. An example of the

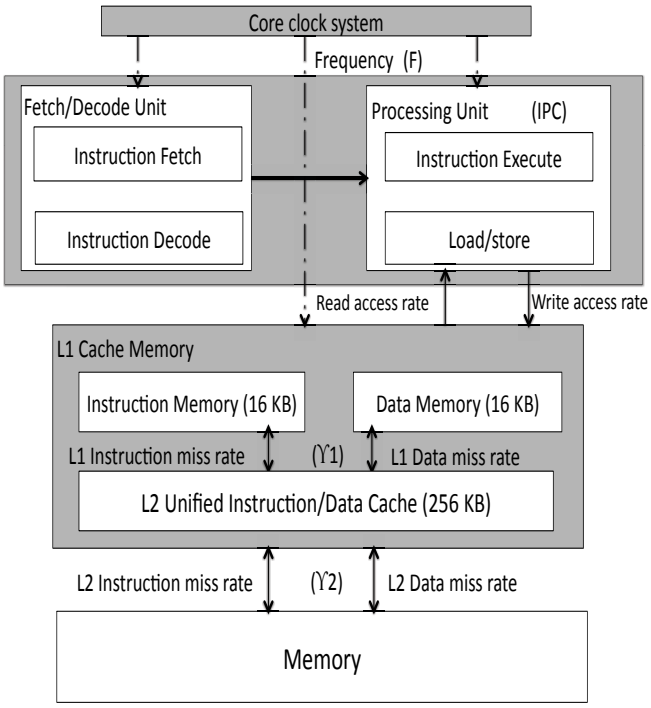

Figure 3.12: Main functional blocks of ARM Cortex-A8 processor

benchmark executing parallel add is shown in the Listing 3.1. Another example regarding frequency is given in the Fig. 3.14. We manipulated the core clock by running the platform on different frequencies starting from 120 to 720 MHz. From the Fig. 3.14, we are able to extract the relationship between the frequency and the power and taking into the consideration the static and dynamic power consumption. The overall operation of a processor can be classified as being in active or in standby mode. Active mode refers to the period of time when the block is actively computing to produce valuable output; the remaining period is called standby mode. In active mode, there are two components of power consumption: dynamic and static power. Dynamic power is consumed while an application is running. The length of time that it runs is usually a small proportion of a clock cycle; for the remaining time, the processor consumes static power. Standby mode, which does not involve any application running, consists of static power alone (assuming that there is also no activity in a clock). It is

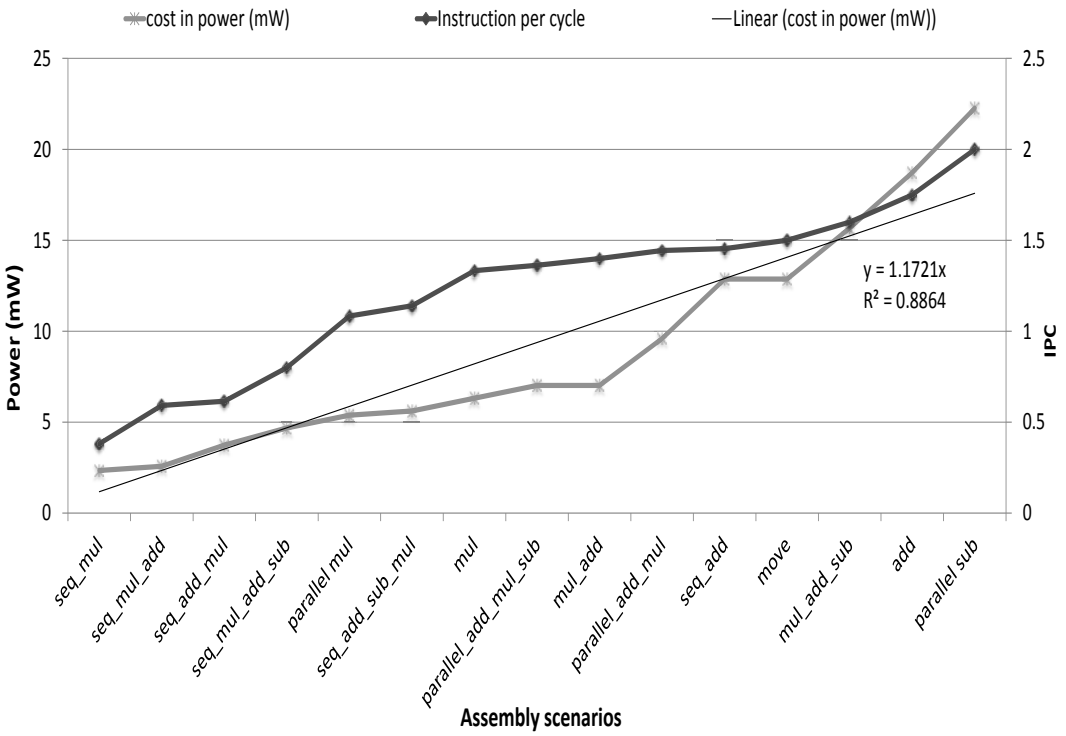

Figure 3.13: Power consumption cost according to the Instruction Per Cycle (IPC)

important to understand that the static power in active mode is a transient one, while that in standby mode is a static one. By using the proposed power modeling methodology, we are able to distinguish between dynamic power and static power and their results as shown in the Fig. 3.14.

Listing 3.1: Benchmark featuring two parallel ADD instructions

```
Loop:   ADD R0, R0, #1
        ADD R1, R1, #1
        Jump Loop               ;infinite loop
```

Listing 3.1 shows that the ARM Cortex-A8 is executing exemplary parallel ADD assembler scenario. The results show that there are significant differences between execution of individual instructions and parallel executions inside the pipeline stage which affects the power consumption. It has to be regarded, that the distribution of basic instructions significantly varies according to the appli-

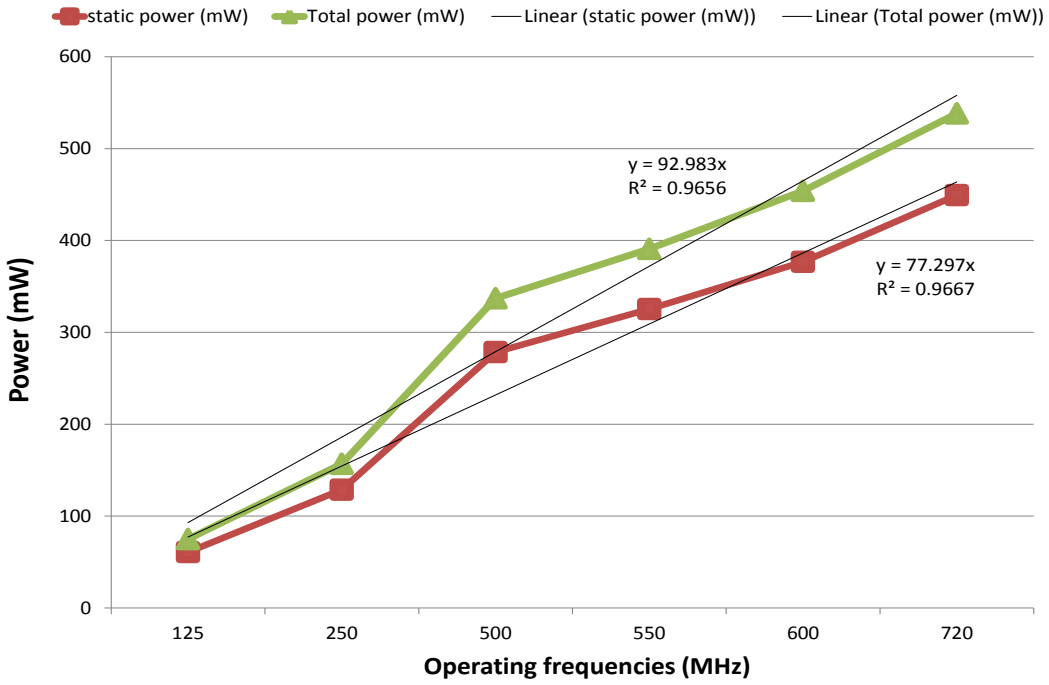

Figure 3.14: Power consumption cost according to the change in frequency

cation.

Table 3.1: Consumption law for the ARM Cortex-A8 platform

Functional parameter	Power model
$F_{processor}$,IPC,γ	P(mW) = $0.79(F_{Processor})$ + 18.65(IPC) + $0.26(\gamma 1 + \gamma 2) + 10.13$

Finally, a curve fitting of the graphical representation will allow us to determine the power consumption laws by regression. The analytical form expresses the obtained power laws. Table 3.1 shows that the power consumption law for the ARM Cortex-A8 processor and its memory system.

3.5.2 ARM9

The ARM9 is the simplest processor compared to the ARM Cortex-A8 processor. As the first step of the power modeling approach, there are three main functional

blocks in ARM940T. They are core clock frequency, the instruction cache and the data cache as shown in the Fig. 3.15. A parameter is denoted for each functional block of the processor and they are γ for L1 cache miss rate and f for core clock frequency. Previous works on the StrongARM have established that the power consumption essentially depends on the clock frequency and the supply voltage. The maximum difference in power consumption between the scenarios is 26%. Increasing the number of instructions in a test scenario (here, more than 4000 instructions) leads to cache misses. Table 3.2 shows that the power consumption law for the ARM9 processor and its memory system.

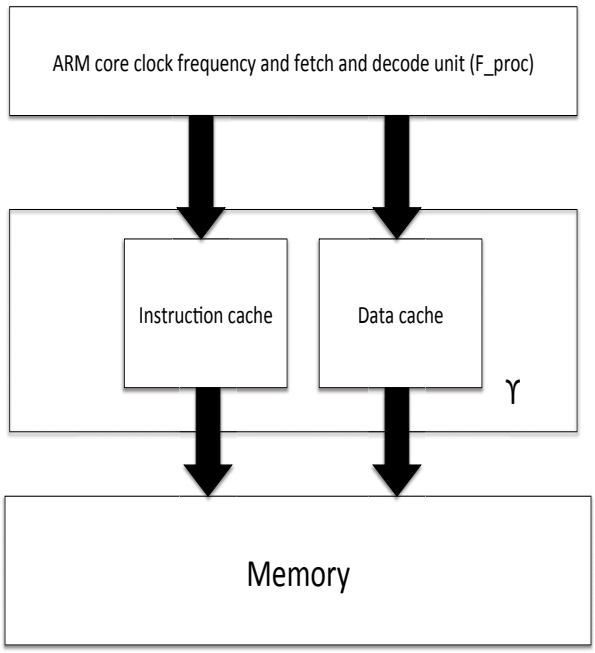

Figure 3.15: Main functional blocks of ARM9 processor

Table 3.2: Consumption laws for the ARM9 platform

Functional parameter	Power model
$F_{processor}$,γ	P(mW) = 1.03($F_{Processor}$) + 0.6 (γ) + 5.3

3.5.3 PowerPC

In a similar fashion to ARM9, the PowerPC 405 is also divided into three functional block. As the first step of the power modeling approach, there are two main functional blocks in PowerPC 405. They are core clock frequency, the instruction cache and the data cache. A parameter is denoted for each functional block of processor and they are γ for L1 cache miss rate and F for core clock frequency.

Table 3.3: Consumption laws for the PowerPC 405 platform

Mapping	Voltage	Power laws
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) = 4.1γ + 6.3F$_{bus}$ + 1599

Table 3.3 shows the power consumption laws for the PowerPC405 processor and its memory system. These models predict consumption of the processor kernel separately, since distinct supplier devices power them with constant voltage: 1.5V for the processor and 2.5V for the SDRAM respectively.

The obtained power models shown in the Table 3.3 depend also on the memory mapping. For this reason, there are different power models for on-chip memory (BRAM) and for external memory (SDRAM). The input parameters on which the power models rely are the frequency of the processor ($F_{processor}$), bus frequency (F_{bus} (MHz)), and the cache miss rate ($0 < \gamma < 100$ (%)). The system designer chooses the frequency of the processor and bus while cache miss rate is considered as an activity of the processor, which could be extracted from the simulation environment. According to these power laws, the static consumption is dominant which is a drawback of the FPGA technology. For this reason, the latest FPGA circuits come with an optimized static power factory setting.

3.6 Power models for multiprocessor platforms

The above developed power models will be used in the framework of system level power estimation of homogeneous and heterogeneous multiprocessor platform that may contain several processors. In the context of this thesis, we have

Table 3.4: Generic power model parameters

Software parameters		
Algorithmic	Name	Description
	τ	External memory access rate
	γ	Cache miss rate for a processor
	α	area utilization for a CLB
Architectural	$F_{processor}$	Frequency of the processor
	F_{bus}	Frequency of the bus
	N	Number of processors

used Virtex-II Pro platform, which contains two PowerPC and reconfigurable logic blocks. This approach is mandatory in the design flow for the generation of power models which can be faster than the other RTL level and another important benefit is that it can be ported into the system-level environment. For instance, Table 3.4 presents the common set of parameters of our generic power model. These sets of parameters are defined for a general class of multiprocessor embedded systems. Additional parameters can be identified for more complex architectures based platforms.

3.6.1 Power model for homogeneous MPSoC

As we have said before that the platform used for the generation of power model for homogeneous multiprocessor system is Xilinx Virtex-II Pro. The above developed power models will be used in the frame of system level estimation of homogeneous MPSoC that may contain several processors. 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 Virtex-II Pro platform. Thus, it is important to have a scalable approach to address the complex system power/energy estimation issue. The equation 3.1 will be considered for the

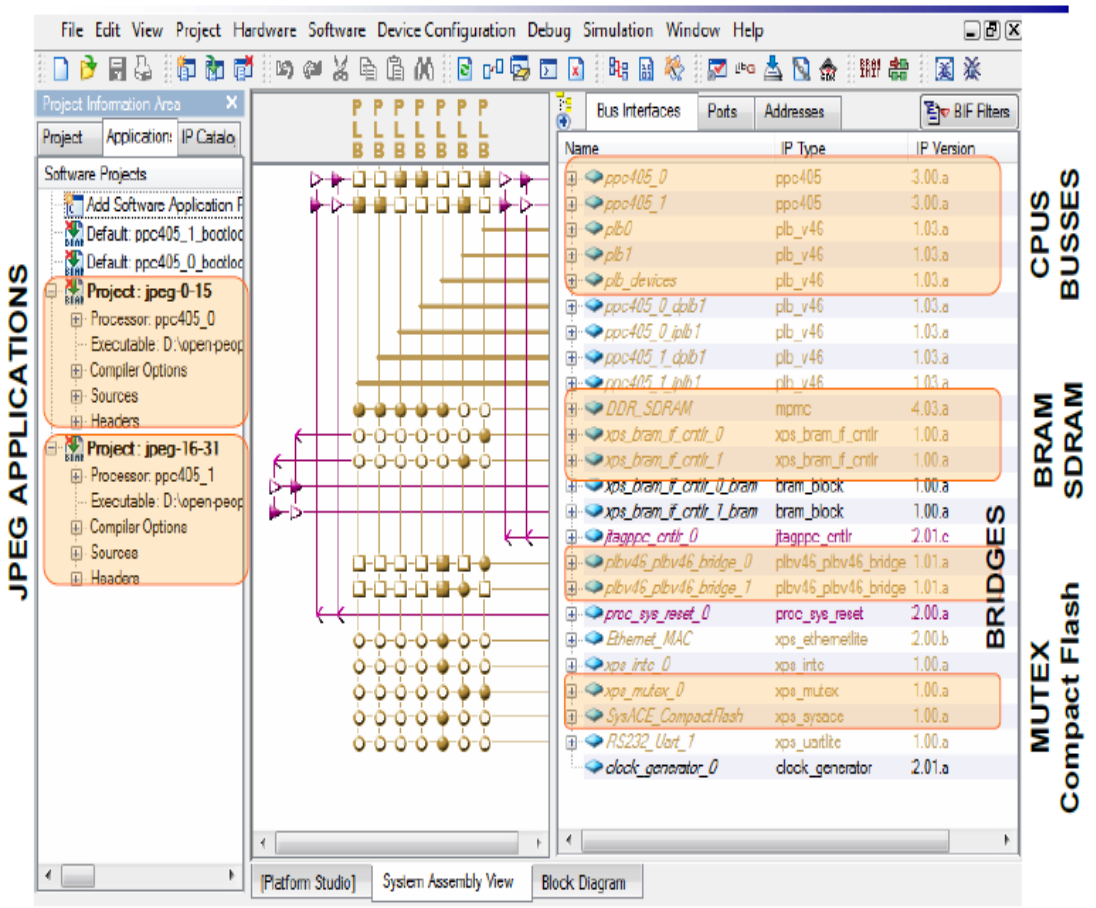

Figure 3.16: Xilinx EDK 10.1 design for two PowerPC processors with shared memory

total system power estimation. We find the sum of the power consumptions of every hardware tasks and the consumption of the synchronization part required to access the shared resources.

Fig. 3.16 shows that the two PowerPC processors are configured inside the Xilinx EDK platform with a shared memory (RAM) and their configuration with the Processor Local Bus (PLB). There are two bridges configured to connect the two PLB buses of the two processors. In order to run the application, we use two methods. First, we split the application into two different parts and then run it on the two different processors, where first processor starts the application and second processor completes the application. Second, As shown in the Fig. 3.16

```
for i = 0 to N
  mutex.lock()
    jpeg.load("image" + i + ".bmp")
  mutex.unlock()

  jpeg.compress()

  mutex.lock()
    jpeg.save("image" + i + ".jpeg")
  mutex.unlock()
```

Figure 3.17: JPEG mutex implementation between the two PowerPC processors

there are two JPEG applications ported on the two processors to run independently. These two processors are synchronized by using the mutex call as shown in the Fig. 3.17.

While running the application on the Virtex-II Pro board, we measured the power across the jumper and its details are shown in the Fig. 3.18. From Fig. 3.18, we can notice that there are three different measurements from left to right. First one from the left denotes the power measured across the Virtex-II Pro running two applications with one processor and it take twice the time (2 x T), Ps denotes the static power and Pj denotes the dynamic power running the board. Middle one denotes two different PowerPC running two different JPEG applications, here static power remains the same Ps, while there is a change in the dynamic power (Pj') and reduced time (T'). In the third one, JPEG application is split to run on both the processor and here mutex power comes into play (Pm). From this figure, we are able to notice that processor increases linearly with a addition of synchronisation power to the power model and its memory and I/O devices if any.

The equation 3.1 will be considered for the total system power (P_{total}) estimation. In addition to the processor (P_{p_i}), the equation involves the power consumption of the synchronization part (P_{sync}) required to access the shared

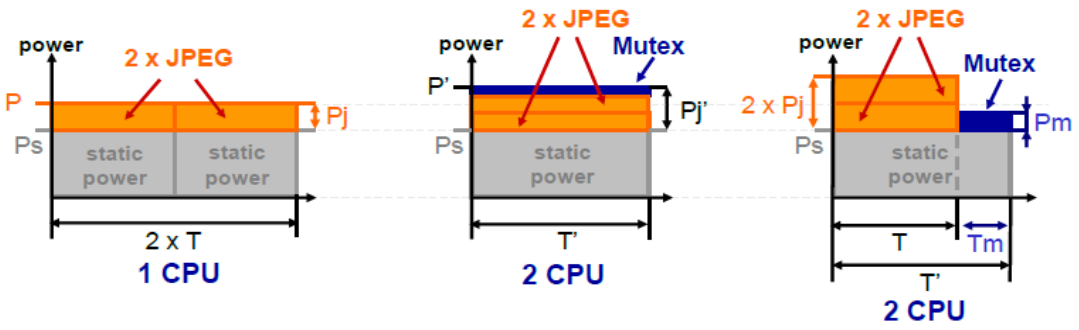

Figure 3.18: Power/time/energy consumption and measurement

memory (P_{mem}) and the shared I/O resources ($P_{I/O}$).

$$P_{total} = \sum_{i} P_{p_i} + P_{mem} + P_{sync} + P_{I/O} \tag{3.1}$$

3.6.2 Power model heterogeneous MPSoC

Nowadays, various techniques are used to improve the performance; hardware acceleration is one of them. Hardware accelerators are designed for computationally intensive software code. Depending upon granularity, hardware acceleration can vary from a small functional unit to a large functional block (like motion estimation in MPEG-2). Many hardware accelerators are built on top of FPGA chips. In this thesis, we use the hardware accelerator built upon a Xilinx Virtex-II Pro FPGA.

A power model has been built for the reconfigurable part of the FPGA component on the Virtex-II Pro board. For a given FPGA, the parameters that can be extracted from the high-level specification are the frequency F (Hz), the toggle rate β (%), and the utilized area α (% slices) of the targeted FPGA. Using a high-level architecture synthesis tool such as GAUT [4], these parameters can be predicted with good estimates. According to the experimental results, the model

does not come as a multi-linear equation of the above-mentioned parameters. For this reason, a 3 entries table of consumption values is used. The power is estimated by interpolation of these 3 input parameters. For instance, Fig 3.19 illustrates the variation of the FPGA power consumption according to area utilization by changing the toggle rate and Fig 3.20 illustrates the variation of the FPGA power consumption according to the switching activity by changing the surface area with an operating frequency set to 100 MHz.

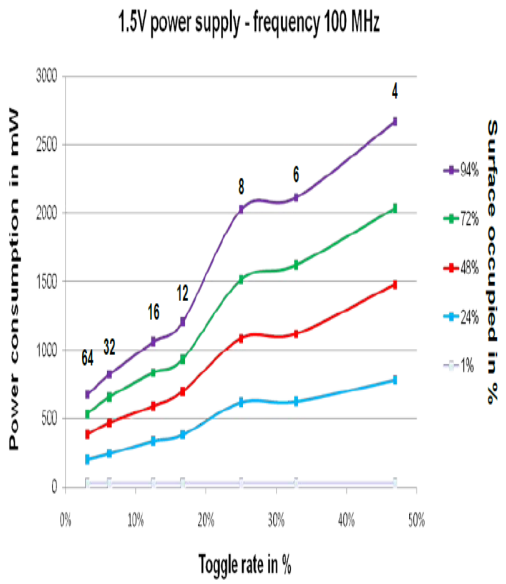

Figure 3.19: FPGA power consumption with 100MHz frequency for different surfaces occupied

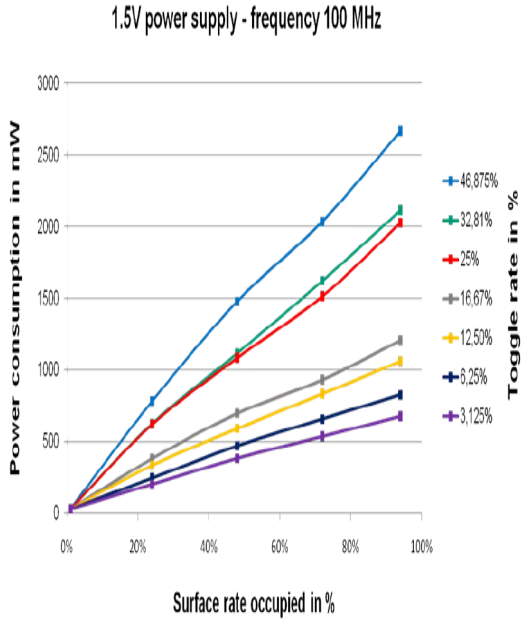

Figure 3.20: FPGA power consumption with 100MHz frequency for different toggle rates

The equation 3.2 will be considered for the total system power (P_{total}) estimation. We mention here that it is necessary to compute the power before the deduction of the total energy consumption. In addition to the processor (P_{p_i}) and the conventional blocks (P_{CLB}), the equation involves the energy consumption of the synchronization part (P_{sync}) required to access the shared memory (P_{mem}) and the shared I/O resources ($P_{I/O}$). Whereas, i and j are the total number of processor's and hardware accelerator's (CLB) respectively.

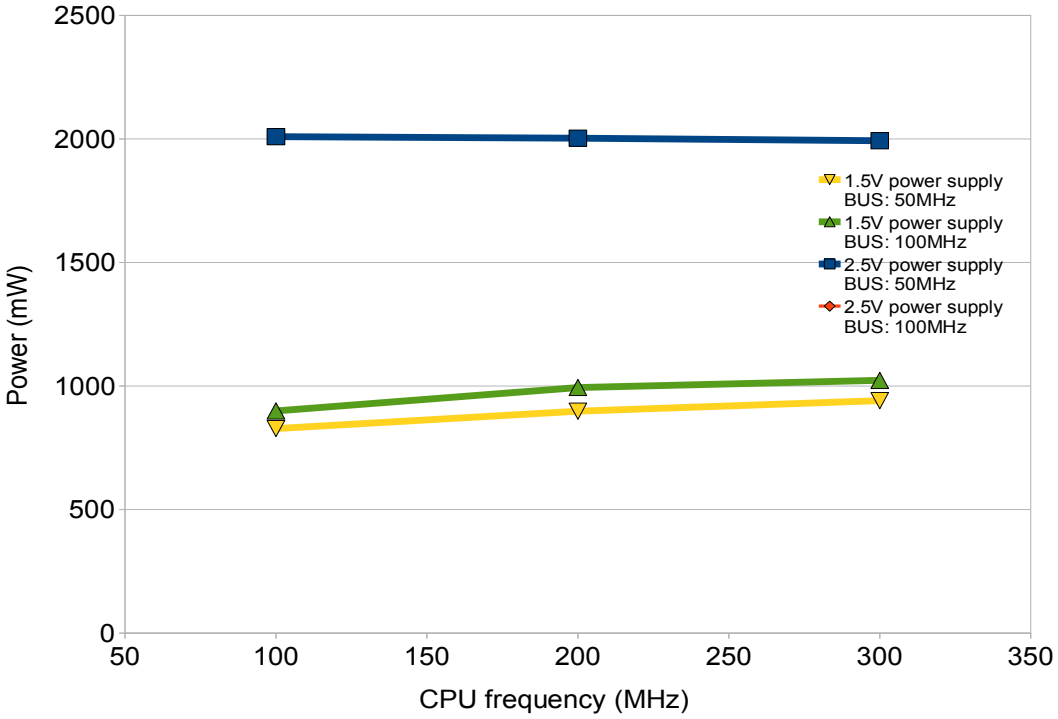

Figure 3.21: Mutex power consumption

$$P_{total} = \sum_{i} P_{p_i} + P_{mem} + P_{sync} + P_{I/O} + \sum_{j} P_{CLB} \tag{3.2}$$

In our Virtex-II Pro platform, synchronization between parallel tasks running on different processors or hardware accelerators is performed by a call to a hardware mutex and and its power consumption in the Fig. 3.21. 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 & bus frequencies. Experimental results show that the mutex power consumption depends mainly on the PLB frequency.

So far, power models for the different processors, homogeneous and heterogeneous multiprocessor based platforms has been developed. Validation of the overall power models will be briefed up with the results in the next section.

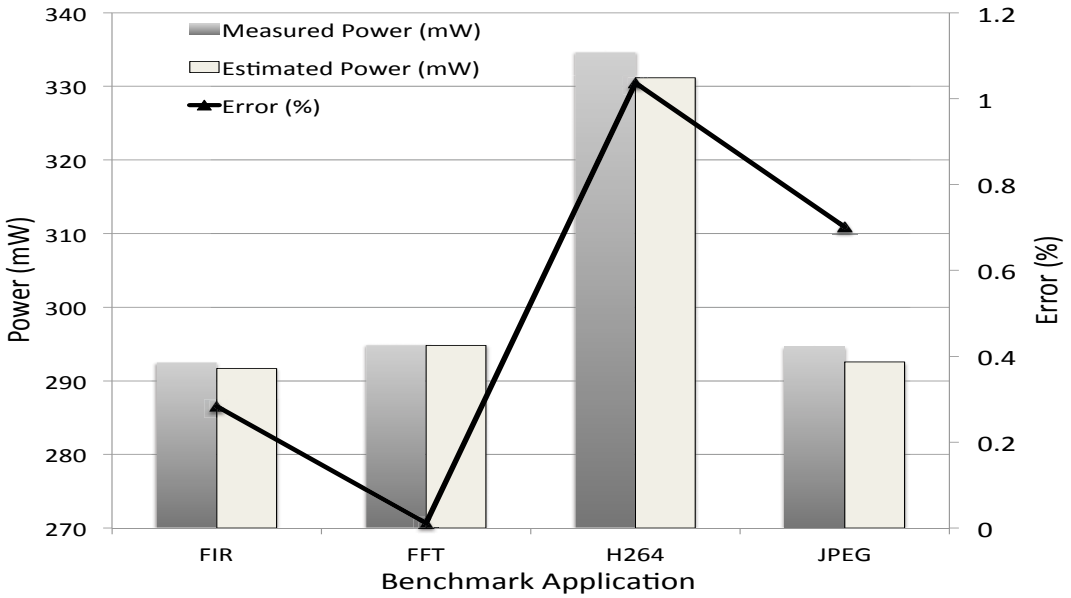

Figure 3.22: Validation of the power model for ARM Cortex-A8

3.7 Validation of the power models

In this section, we will be validating the different processor power models against the real board measurement in order to find the efficiency of FLPA modeling methodology used in this thesis. The flow of the benchmark evaluation is as follows. First, the applications are compiled using respective cross-compilers for the FPGA and OMAP platforms. The binary files are transferred into the platforms (OMAP and Virtex-II pro) and the corresponding power consumption is recorded. Finally, the experimental values measured from the platforms must be compared with the estimations from the power consumption model by extracting the needed activities of the power model from the real-board. The cache miss rate is obtained by executing the valgrind[1] tool on the real-board and IPC is extracted by activating the pipeline counters inside the processor.

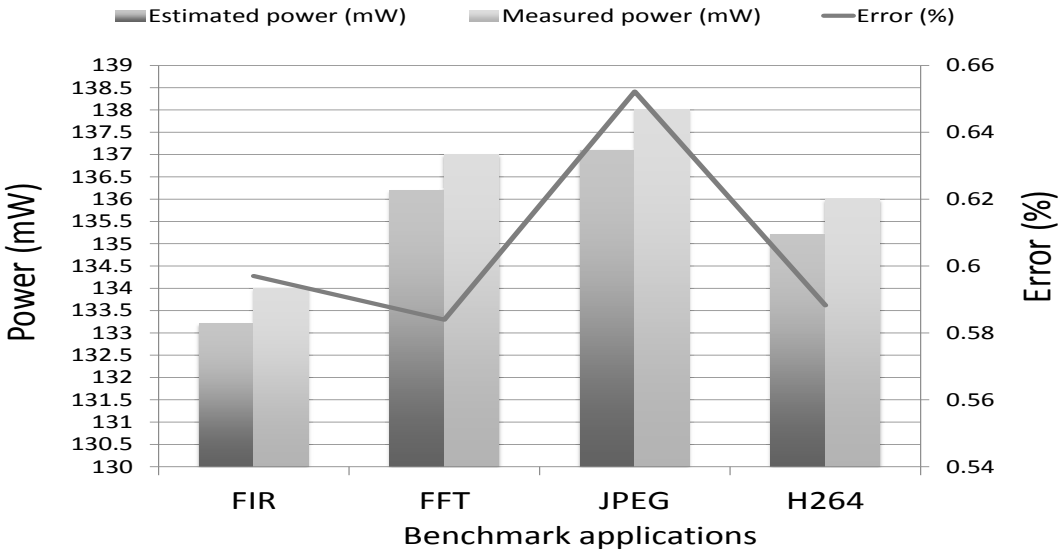

Figure 3.23: Validation of the power model for ARM9

3.7.1 Evaluation of ARM Cortex-A8 power model

Fig. 3.22 represents the estimated total power consumption of each benchmark using the power model shown in Table 3.1 for ARM Cortex-A8 processor. Fig. 3.22 illustrates the results and shows the comparison between the proposed methodology and the real board measurements. Our power modeling approach has a negligible maximum error equal to 1.5%, which offers better accuracy. This is due to better characterization of the power model.

3.7.2 Evaluation of ARM9 power model

Fig. 3.23 exemplarily shows the comparison between measurements and estimation results of the benchmarking programs at 120MHz frequency using the power model shown in Table 3.2. All estimations are lower than the real board measurements. The maximum relative error across all benchmarks is 0.62% at a total variation in all scenarios of up to 2% (depending on the clock frequency) which is a large estimation improvement.

[1]http://valgrind.org/info/tools.html

3.7.3 Evaluation of PowerPC power model

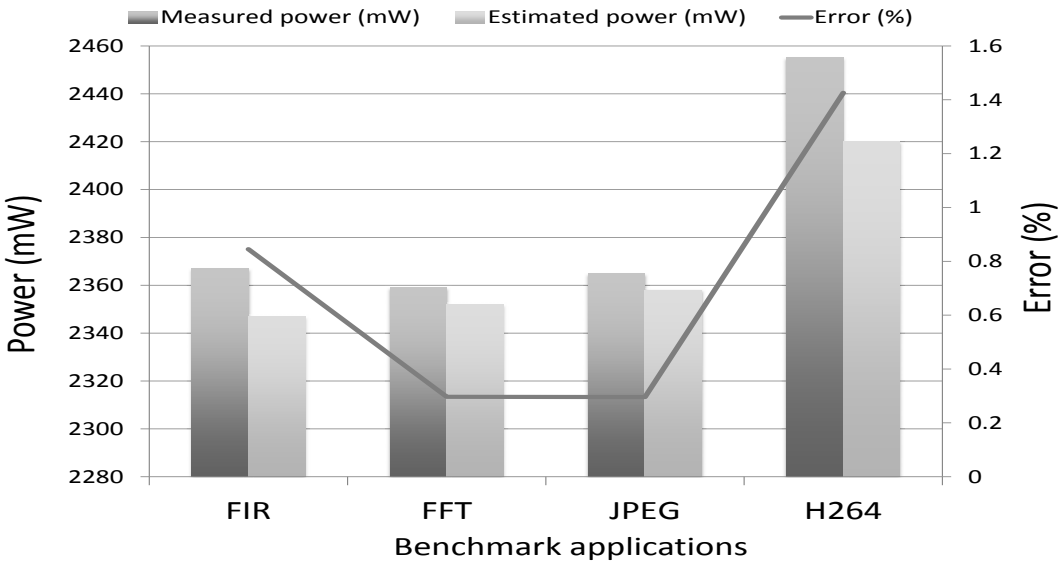

Figure 3.24: Validation of the power model for PowerPC

Fig. 3.24 shows the average error obtained with our high-level power modeling approach against real board measurements for the PowerPC using the power model shown in Table 3.3. The results obtained for the different experiments validate this approach. Thus, a good adequacy between architecture, algorithm, and power consumption can be found with our model at early design stage. Our power modeling approach has a negligible maximum error equal to 1.4%, which offers better accuracy.

3.8 Conclusion

This chapter has described the proposed power modeling methodology for existing different RISC processors, homogeneous and heterogeneous MPSoC based on FLPA. Specifically, the impact of each parameter on the power for different processors has been investigated and modeled. The interest of our approach is to enhance the accuracy of the power models. The time consumption in the design backtrack is strongly reduced and power models can be reused for designing a

new system. Validation has then been done against real board measurements for estimation accuracy, incurring an average error of 0.715% for the models.

Note that, thanks to FLPA approach applied to the power models generation process, this approach is quite general and can therefore be easily adapted and used on different architectures. More important, this modeling exercise has the goal of showing how power models can be implemented in practice. Another example of how the power models developed in this chapter can be adapted for system-level environment will be presented in the upcoming chapters. In this thesis, the models have been expressed using an analytical representations for processors and LUTs for FPGA.

CHAPTER 4

CO-SIMULATION ENVIRONMENT AT SYSTEM-LEVEL FOR POWER ESTIMATION

4.1 Introduction

Before multiprocessor system-on-chip (MPSoC) architectures became so complex, the hardware and software components of an embedded system were designed sequentially. As research pushes for better programming models for multiprocessor and multicore embedded systems, virtual platforms solve one of today's biggest challenges. The benefits of virtual platform are software development, debug, validation before the hardware board is available and rapid prototyping of the hardware boards available in the market, enabling concurrent hardware/software co-design. Virtual platform models the hardware architecture in the form of a simulator, including processors, memories, communication links and peripherals. They enable engineers to start developing and testing the software substantially earlier than it has been possible in the past. Although these platform models enhance the designer efficiency and reduces the time-to-market, its weak point remains the consideration of power consumption metric. Most of the current

system power estimation tools are still working at the RTL or Cycle-Accurate (CA) levels and there is a scarcity of tools enabling power estimation at the system-level.

In the previous chapter, we have proposed an efficient power modeling methodology to cover a large spectrum of embedded systems. In this chapter, we will propose a system-level environment for hardware/software co-simulation to extract the activities needed by the power models developed. This chapter will focus on three main objectives: first, accurate extraction of the activities on which the power model relies. The second is a faster simulation technique and a suitable abstraction level for rapid reuse of IPs. Third, we aimed to provide a framework for homogeneous and heterogeneous MPSoC platforms to ensure the ease in scalability for DSE. Two simulation approaches are used in this thesis. In the earlier phase of this thesis, to build an MPSoC system, we reused IPs (processor, cache, memories etc.,) provided by the SoCLib [1] environment which are described on SystemC/TLM-DT level. In the later phase of this thesis, to have a rapid prototyping and faster simulation, we switched to virtual platform IPs provided by Open Virtual Platform (OVP), whose processors use Just-In-Time (JIT) simulation technique instead of traditional Instruction Set Simulator (ISS).

This chapter is structured as follows: Section 4.2 presents a detailed description of modeling with the help of SoCLib environment and their performance estimation based on extraction of activities need for the power model. Section 4.3 presents a proof of using the JIT/TLM level for simulation and evaluation of MPSoC systems and describes the various hardware models developed and reused from the OVP library to implement and to improve the performance of the proposed virtual system-level framework. Section 4.4 details the implementation of OVP IPs inside the System/TLM environment. In Section 4.6, the effectiveness of our proposal is illustrated through the JPEG encoder application performed on PowerPC based MPSoC and also presents the comparison of our proposed MPSoC environment with ISS/TLM level in terms of estimation speed and modeling efforts.

[1] http://www.soclib.fr/trac/dev

4.2 Virtual prototyping with the help of SoCLib environment

In the earlier phase of this thesis, we used SoCLib library to build our platform. SoCLib is a library of hardware components that enables to build an MPSoC system at SystemC/TLM level. By using the IPs provided by this platform, we instantiated the hardware: first, to emulate the behavior of the embedded platform selected in the frame of the OPEN-PEOPLE project. Second, to explore different architectural solutions based on the activities retrieved from the simulation. In this section, we will start by describing the different components that have been reused to design our experimental platform.

4.2.1 Available models at TLM-DT level for MPSoC design

As we said before, SoCLib provides different hardware models such as cache, bus protocol, processor, memory models, interconnects etc., Understanding the behavior of each component is an essential step in order to extract the needed activities for the power models. In this section, we will describe the various hardware models used to built an MPSoC.

- Virtual Component Interface (VCI) protocol: In general, most of the components are connected by an on-chip bus. It can be either by a Network on Chip (NoC), a simple bus or a crossbar. In this thesis, components are connected mainly using the VCI on-chip-bus protocol. This makes the components to easily interoperable. Moreover, VCI is simple enough to integrate new components, without forbidding translation of VCI to other protocols.
- Xcache: This hardware component is a generic cache controller, fully compliant with the VCI advanced protocol. It can be used to interface a single instruction issue - 32 bits RISC processor (in our case PowerPC 405) to a VCI based multi-processor system. They act directly as a wrapper for any Instruction Set Simulator (ISS) respecting the generic cache/processor interface defined in the ISS header file. We used this component to inject counters to extract the cache activities such as: Instruction cache read hit/miss, Data

cache read/write hit/miss. The description about the configuration of Instruction and Data cache are given below:

- Instruction Cache
  - It is a read-only cache.
  - It uses the mapping table to support uncached segments.
  - In case of read miss or read uncached, the processor is stalled until the missing instruction is available.
  - The two VCI transactions generated by the Instruction cache are given below:
    - *read burst* corresponding to a missing cache line,
    - *one word read*, when the corresponding address is uncached.
- Data Cache
  - The write policy is write-through (the data is immediately written in memory and the cache is updated only in case of hit).
  - The data cache contains a write buffer and builds a burst when there are successive write requests in the same cache line.
  - It uses the mapping table to support uncached segments.
  - The data Cache supports the following requests : read, write, linked load and store Conditional
  - The data cache accepts a line invalidate command.
  - Three types of VCI transactions can be generated by the data cache and they are given below:
    - *read burst* corresponding to a missing cache line.
    - *one word transaction* corresponding to an uncached read, a linked load or a store conditional.
    - *write burst* of variable length (no larger than a cache line)
  - The processor is stalled in case of cached read MISS, in case of uncached read or in case of write, if the write buffer is full.

- Processor: There are wide range of processors available in the SoCLib library. In this thesis, we used PowerPC405 ISS and ARM ISS. This ISS uses the

ISS2 API and should be wrapped with a VciXcacheWrapper. The simulation model is actually an ISS, organised as a two-stage pipeline:

  - *First stage:* instruction fetch execute with a possible access to the external data cache.
  - *Second stage:* read memory access is written back to registers.

  The main functional specifications are as follows:

  - The floating point instructions are not supported.
  - There is no TLB, and no hardware support for virtual memory directly in the ISS. Nevertheless, Memory Management Unit(MMU) may be supported through the cache.

- Interconnection network: This hardware component is a VCI compliant full crossbar, it contains two independant crossbars for VCI commands and VCI responses. It must only be used in clusterized architecture, to interconnect a limited number of VCI initiators and targets in a local sub-system. The associated sub-system is identified by a global index.
  - The number of VCI initiators and VCI targets are parameters that should not be larger than 4 each.
  - The VciLocalCrossbar component has a dedicated VCI interface (both initiator and target) to connect the local subsystem to the global VCI interconnect.

  When several initiators try to reach the same target, the arbitration policy is round-robin (RR). As any VCI advanced compliant interconnect, this component uses the MSB bits of the VCI ADDRESS field to route the command packets to the proper target, thanks to a routing table implemented as a ROM. This routing table is build by the constructor from the informations stored in the mapping table. It uses the VCI RSRCID field to route the response packet to the initiator.

- Memory module: This VCI target is an embedded SRAM controller. It is actually a simplified version of the VciRam component and provides the same services. It handles one or several independent memory segments. Each segment is defined by a base address and a size (number of bytes). Both the base and the size parameters must be a multiple of 4. The segments allocated to

a given instance of this component must be defined in the Mapping Table. The segments are implemented as dynamically allocated arrays in the constructor. This component supports an optional latency parameter, defining the RAM access latency.

4.2.2 Estimating performance with Soclib environment

In order to estimate and evaluate performance, we used Virtex II Pro as reference platform to built our simulation framework. We choose this platform because of its availability of two PowerPC processors and configurable logic blocks. Hence, we can able to built and evaluate our mono-processor, homogeneous and heterogeneous multiprocessor architectures.

Using the components described in the previous sub-section 4.2.1, different multiprocessor architectures can be modeled, simulated and evaluated. These can contain a variable number of processors, hardware accelerators, peripherals, I/O, etc.

Figure 4.1 shows an simulation framework emulating the Virtex II Pro FPGA based MPSoC. These architectures are built for running intensive signal processing application requiring high computation rate. To run these applications with the simulator, different tasks are set and then compiled to the target processor. Manipulated data are located on the memory component. To ensure correct sequencing of tasks between processors, synchronization variables are set. These synchronization variables are stored in memory. They are read and modified by different processors. The estimated performance of the application is given by the simulator in cycles. Each processor is programmed with a timer counter register. The timer counter is incremented at each clock cycle. Thus, to estimate the execution time of a task on a processor, we have to read the register value before and after the execution of the task. Furthermore, the execution time of the entire application is given by the global clock of the SystemC simulator. As we have pointed out, the description of an architecture at TLM-DT provides accurate performance estimates. But our objective here is to accurately extract the activities needed for the power model. For this purpose, we have injected several activity counters for the hardware components such cache, memory and

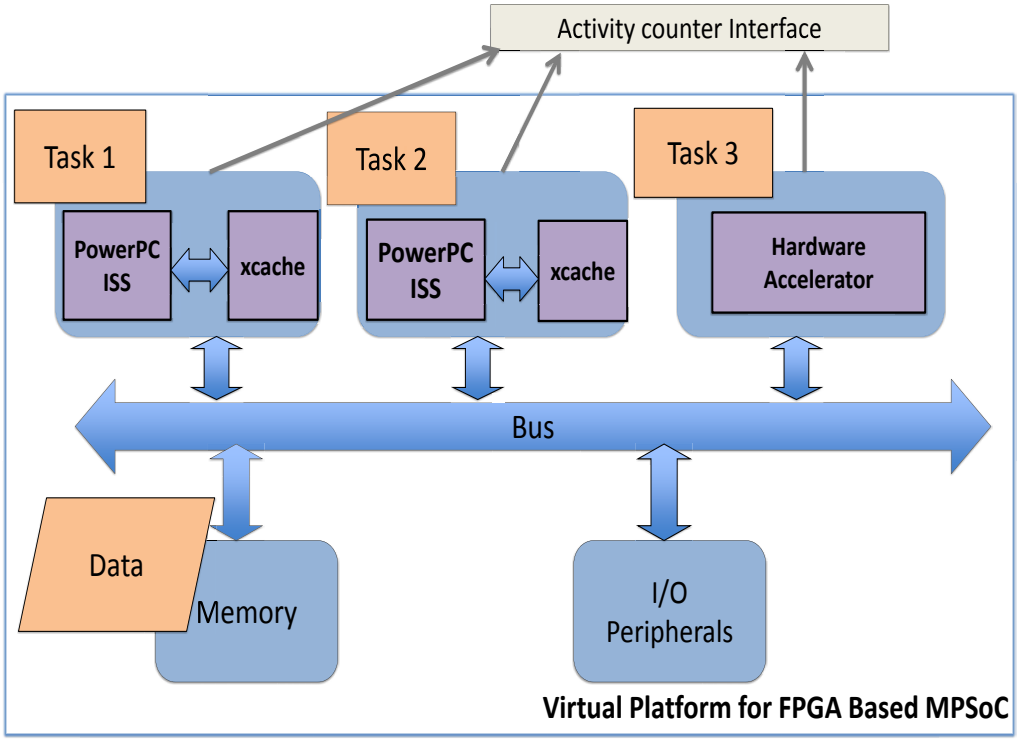

Figure 4.1: Simulation Environment with SoCLib

the processor. For each and every access to the cache or memory, the counters are incremented during the simulation. The activity results obtained at this level will enable our framework to provide an accurate power estimation and also yield a reliable design space exploration. For hardware accelerator power model, the GAUT[1] tool is used to estimate the activities such as surface occupied and toggle rate. GAUT is an academic high-level synthesis tool dedicated to signal processing applications. Starting from a pure C function GAUT extracts the potential parallelism before selecting/allocating operators, scheduling and binding operations. The mandatory design constraints are the throughput (the initiation interval), the clock period and the target technology. The optional design constraints are I/O timing diagram and the memory mapping. GAUT synthesizes a potentially pipelined architecture composed of a processing unit, a memory unit, a communication and multiplexing unit and a GALS/LIS interface.

[1] http://hls-labsticc.univ-ubs.fr/

Table 4.1: Application miss rates for PowerPC based ISS/TLM

Program	Instruction miss rate	Read Miss rate	Write Miss Rate	Total Miss Rate
acquisition	0.004386	3.56	37.73	0.10
rgb2yuv	0.001228	3.73	99.01	6.00
dct y	0.002983	4.89	45.72	4.00
dct u	0.000365	4.99	45.72	4.00
dct v	0.000354	4.39	45.72	4.00
qt y	0.000852	2.66	99.90	6.00
qt u	0.000476	2.66	99.90	6.00
qy v	0.000476	2.66	99.90	6.00
huff y	0.004435	4.98	20.11	1.00
huff u	0.000575	4.97	20.8	1.00
huff v	0.000693	4.96	20.61	1.00
rebuild image	0.298380	3.95	30.19	3.00
complete application	0.000342	0.129	0.59	0.22

Table 4.1 shows an example of extracting activities (cache miss rates) at TLM-DT. The results are reported for the JPEG decoder application which will be detailed in the section 4.6.1. The extracted activities are compared with the values generated by the valgrind [1] tool executed on the real-board for the cache miss rate and the results shows an average error of 1%. The objective here is to show that, through this environment it is possible to accurately extract the activities needed for the power model and performance. Despite the accuracy of these estimates, the major problem is the time required to obtain the results at the TLM-DT level. The simulation time at this level for JPEG encoder application is around 300 sec for decoding a JPEG file image size of 512x443 pixels using a machine Pentium M (2.2 GHz). The simulation speed is slow due to the use of interpreted ISS, which executes the target binary instruction by instruction. In each iteration, the interpreted ISS fetches, decodes and executes the instruction. The interpreted ISS's are slow due to their interpreted nature, but provide very detailed estimation of cache, pipeline and memory model. The interpreted ISS's are very flexible. For this reason, in the next section we will be focusing on fast estimation aspect without compromising the accuracy of the activities using virtual platforms.

[1] http://valgrind.org/

4.3 Virtual prototyping with the help of OVP platform

To reduce the simulation time without compromising on the accuracy of activities inside the embedded systems, particularly those incorporating MPSoC, our solution tends towards a virtual platform defined at JIT/TLM level. This abstraction must be sufficient to verify the behavior of the system (applications deployed on the architecture) and also to measure the execution time and to extract activities accurately. Our case study is built around a family of MPSoC on which an application is deployed. From the deployment of this architecture, we evaluate the performance of our system which allows us to extract the most appropriate solution. In this section, we limit ourselves to performance criteria such as execution time and accuracy of the activities in comparison with ISS/TLM level. In our work, we have set the following objectives:

- The functional verification of this platform is to run the target application on a defined architecture and to ensure the accuracy of the simulation result.
- The analysis of system performance without excessively penalizing the simulation time.
- The analysis of the power consumption activities of the system for architectural exploration.
- Reduced development effort compared to the lower levels.

4.3.1 OVP platform models for MPSoC design

In this section, we present the component models that have been used in our virtual MPSoC prototype. These components are generic and allow us to evaluate the performance for a class of systems with shared memory MPSoC. Component models used in our work are: processor, memory caches and interconnection networks (bus).

4.3.2 OVPsim

OVPsim[1] is equipped with an infrastructure to program multiprocessor platforms. The OVPsim simulator can simulate arbitrary multiprocessor shared memory configurations and heterogeneous multiprocessor platforms. OVPsim platform models can be compiled as shared objects, they can be encapsulated in any simulation environment that is able to load shared objects. This includes C, C++, and SystemC simulation environments. In the context of this thesis, OVPsim has been wrapped inside a SystemC wrapper. OVP simulator is built as a slave and thus callable from other environments such as SystemC. The reverse is however not true. OVPsim cannot call a SystemC model. This is quite natural since the calling of SystemC would bring the entire simulator performance back down to the very performance it is trying to replace. On the other hand, substituting part of the system which is a SystemC based platform with an OVP model may bring about a large performance gain in relative terms. Putting OVP models in SystemC environment therefore requires careful scheduling.

OVP models and subsystems can be encapsulated in SystemC platforms and harnessed using:

- sc_clock(), i.e. at the detailed instruction or clock level
- TLM 2.0, i.e. the new OSCI transaction level approach

Since modeling in pure ISS/SystemC brings down the simulation speed and hence the rate of power estimation at system-level, we emphasize on integrating OVP models at the transaction level.

The several key points due to which the OVPsim is faster are explained in the sections below:

- Just-In-Time (JIT) code morphing technology:
  Traditional processor models are written in HDL or similar modeling languages, which are activated by a clock signal and implemented by a loop. When the clock is activated, the model will fetch the next instruction, decode it and calls the specific function to execute the instruction. The conventional style may be accurate and straightforward in structure but they are not fast. Instead of the traditional style, the OVP tool provides Just-In-Time

[1] http://www.ovpworld.org/dlp/

(JIT) code morphing technology. This style is slightly similar to dynamically compled ISS. The working mechanism of this technology is given below:

  - Whenever there is a new instruction encountered during the program execution is morphed into its native machine code. The exact translations to be made are specified by the processor modeler using the Virtual Machine Interface (VMI) API.
  - Adjacent sections of translated processor instructions are gathered into code blocks, which are held in a library for the processor. Separate libraries are held for supervisor mode code fragments and user mode code fragments.
  - There is no need to morph the code which is already done once, the simulator simply re-executes the existing code block from the library. OVP technology handles the generation of native machine code and the efficient management of code blocks and libraries to give extremely fast simulation. This is possible because, as simulation proceeds, run time (execution of translated code blocks) dominates morph time (JIT compilation). It may be possible that not all instructions map closely to the JIT code morphing opcode set. Such a simulation method is capable of providing speed improvements if the application under test has a portion of code used repeatedly, which in general, all the real time applications do.

- Program counter modeling

  The simulator always knows the address of the current instruction. Instead of maintaining the program counter value each time in the processor model, it is fetched directly from the simulator when required. Thus the processor models do not explicitly model the register values that are infrequently referenced and can be created easily on demand. The same is the case very often for processor status registers. This makes processor models execute at a faster rate.

4.3.3 Interfaces of OVPsim

In order to model an MPSoC, several components have to be modeled such as peripherals, processor, bus architecture etc. OVP is thus made of four interfaces.

- Innovative CPU Manager (ICM): The ICM interface is used to create the platform netlist of the system to use with OVPsim simulator. It allows instantiation of multiple processors, buses, memories and peripherals that can further be connected together and application programs executables can be loaded in simulated memories. This interface is written in C.
- Virtual Machine Interface (VMI): The VMI interface allows the processor model to communicate with the simulation kernel and also the other components of the system. Processors developed by OVP uses an code morphing approach which is coupled with a Just-In-Time (JIT) complier to map the processor instructions. Few of the abilities of VMI is listed below: First, The VMI can be used for any type instruction set of the processor (CISC and RISC). Second, VMI allows modeling of several peripheral in a short time such as L2 cache. Third, VMI allows a form of virtualization for capabilities such as file I/O. This allows direct execution on the host using the standard libraries provided. Fourth, encapsulating existing ISS models within OVPsim helps to export some basic features (for example, the existing ISS model should be available as a shared object, provide an API to allow it to be run instruction-by-instruction or for a number of instructions, and provide an API allowing memory to be modeled externally) through VMI interface. Finally, VMI enables modeling of the mode dependent behavior (kernel/user mode) of an instruction. Using the VMI, OVPsim can implement arbitrary multiprocessor systems.
- Behavioral Hardware Modeling (BHM) and Peripheral Programming Model (PPM): They are used to write behavioral models of hardware/software systems which are peripheral to the processors in the platform being developed. Each instance of a peripheral model runs on its own virtual machine with an address space large enough for the model. This processor and its memory are separate from any processors, memories and buses in the platform being simulated; they exist only to execute the code of the peripheral model.

This processor is called a Peripheral Simulation Engine(PSE). The difference between PPM and BHM is:

- BHM: This API gives access to behavioral modeling processes (threads), simulated delays, events diagnostic control and simulator message stream. This API can support more general forms of communication and provides the piece that TLM is missing.
- PPM: This API gives access to connectivity of peripherals in platforms, creation and control of ports, nets, address spaces and windows into memory address space. Thus, this API understands about buses and networks and is similar in terms of functionality with the OSCI TLM interface proposal.

The BHM/PPM has similar concepts to SystemC, but each instance of each model exists in its own private address space. It is normally pretty easy and simple to wrap existing C functions in a BHM/PPM peripheral model.

To build a whole platform, the combination of above mentioned interfaces plays a major role. The Fig.4.2 shows the interaction between the interface. Each instance of a peripheral model runs on its own virtual machine with an address space large enough for the model.

OVP provides with processor models like ARM processors, MIPS processors, PowerPC processors, Tensilica and OpenRISC OR1K. A number of standard embedded devices to allow assembly of a complete platform, including various types of memories, traps, bridges, DMA engines and UARTs, to name a few are also modeled. OVP processor models are instruction accurate in purely functional space and not in the behavioral space as compared to the commonly used approximately timed models. To make it clear, the functional model does not consider the timing, however it includes the sequence while the behavioral model includes timing but the detail of timing is not defined. OVP models are functional models. Instruction accuracy in terms of OVP means that the registers hold the correct values at the end of each instruction and create the right effects from executing that instruction. Now, they have added another feature to this model called multi-execution pipelines and out of order execution. Listing 4.1, shows the instantiation of the processor model and in our case, it is ARM Cortex-A8

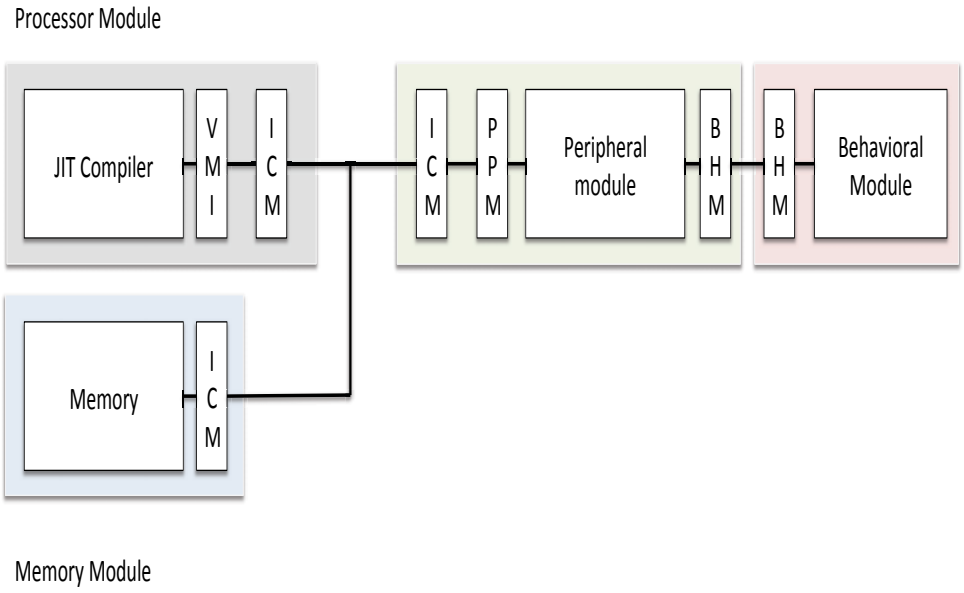

Figure 4.2: Communication of OVP interfaces

processor with a 32 bits address mode.

Listing 4.1: Instantiation of the processor

```
int main(int argc, char ** argv) {

        // initialize CpuManager
        icmInit(0, 0, 0);


        // create a processor
        icmProcessorP processor = icmNewProcessor(
                "cpu1",                 // CPU name
\               "Cortex-A8",    // CPU type
                0,                              // CPU cpuId
                0,                              // CPU model flags
                32,                             // address bits
                model,                  // model file
                "modelAttrs",   // morpher attributes
                0,                              // enable tracing etc
                0,                              // user-defined attributes
                semihosting,    // semi-hosting file
```

```
                "modelAttrs"    // semi-hosting attributes
        );

        // load the processor object file
        icmLoadProcessorMemory(processor, argv[1], False, False, True);

        // run simulation
        icmSimulatePlatform();

        // terminate simulation
        icmTerminate();

        return 0;
}
```

4.3.4 VMI Memory Model

The functions in this interface are used to build Memory Model Components (MMC), such as instruction and data caches, that supplement OVP processor models. An MMC fits between a bus master such as a processor or a peripheral with bus mastership capability, and a bus slave such as a RAM, ROM or peripheral with a bus slave port. MMCs can also be cascaded to model, for example, multi-level caches.

There are two distinct kinds of memory component models: full and transparent. Full models implement storage and so can be used to accurately model components such as caches that are incoherent with main memory. Transparent models do not implement storage (so cannot be incoherent) but can be used to create very fast performance monitors. As an example, a transparent cache model would model only the cache tags and use this information to count hits and misses.

Listing 4.2: Instantiation of the full cache model

```
        const char *vlnvRoot = 0; // when null use default library
        const char *model = icmGetVlnvString(
                vlnvRoot, "ovpworld.org", "processor", "or1k", "1.0", "model"
        );
        const char *semihosting = icmGetVlnvString(
                vlnvRoot, "ovpworld.org", "semihosting", "or1kNewlib", "1.0",
                "model");
        const char *mmc_model = icmGetVlnvString(
```

```
            vlnvRoot, "ovpworld.org", "mmc", "wb_1way_32byteline_2048tags",
            "1.0", "model");


    // initialize CpuManager
    icmInit(0, 0, 0);


    // create a processor
    icmProcessorP cpu1h = icmNewProcessor(
    "cpu1", // CPU name
    "Cortex-A8", // CPU type
    0, // CPU cpuId
    0, // CPU model flags
    32, // address bits
    model, // model file
    "modelAttrs", // morpher attributes
    0, // simulation attributes
    0, // user-defined attributes
    semihosting, // semi-hosting file
    "modelAttrs" // semi-hosting attributes
    );

    // create transparent MMCs
    icmMmcP mmcL1I = icmNewMMC("mmcL1I", mmc_model, "modelAttrs", 0, 0, True);
    icmMmcP mmcL1D = icmNewMMC("mmcL1D", mmc_model, "modelAttrs", 0, 0, True);
    icmMmcP mmcL2 = icmNewMMC("mmcL2", mmc_model, "modelAttrs", 0, 0, True);

    // connect level-1 mmcs direct to processor ports
    icmConnectTransparentMMC(cpu1h, mmcL1I, "sp1", mmcL1D, "sp1");

    // cascade level-1 mmcs to level-2
    icmChainTransparentMMC(mmcL1I, "mp1", mmcL2, "sp1");
    icmChainTransparentMMC(mmcL1D, "mp1", mmcL2, "sp1");


    // create the processor bus
    icmBusP bus = icmNewBus("bus", 32);


    // connect master port of level-2 MMC to bus
    icmConnectMMCBus(mmcL2, bus, "mp1", True);

    // create and connect memories...
```

In order to speed-up the simulation without compromising accuracy, first, we decided to tune the data pattern granularity of the application and to analyse its effect in the accuracy and speed.

In the context of this thesis, we use both transparent and full cache models depending on their need. Transparent cache is used whenever there is a need for performance and full cache models are used to accurately extract the cache activities for the processor with a slight reduction in the performance. Implementation of the cascaded transparent cache model is given in Listing 4.2. Mostly, full cache models are used for multiprocessor architecture and it is connected to a VMI interface of a processor by a ICM interface of a bus. Cache model settings are given in the Listing 4.3. From this listing, we are able to configure different cache models for different processors by changing the values of the lines, ways and size. Cache Ratio Monitor (CRM) is an essential component of this work. With the help of CRM, we are able to extract the activities of the cache models. Full implementation of cache model and CRM are given in the Listing 4.4. Cache readInfo gives the details about the read activities inside the cache and cache writeInfo for the write activities.

Listing 4.3: Cache model settings

```
#include    vmi/vmiMmc.h
#include    vmi/vmiMmcAttrs.h
#include    vmi/vmiMessage.h

typedef struct cacheObjectS {
        Uns32 lines;
        Uns32 ways;
        Uns32 size;
} cacheObject, *cacheObjectP;


// Cache object constructor
static VMIMMC_CONSTRUCTOR_FN(cacheConstructor) {

cacheObjectP cache = (cacheObjectP)component;

cache->lines = (Uns32)vmimmcGetUns64Attribute(component,   lines   );
cache->ways = (Uns32)vmimmcGetUns64Attribute(component,    ways    );
cache->size = (Uns32)vmimmcGetUns64Attribute(component,    lines   );
```

Listing 4.4: Cache Ratio Monitor (CRM)

```
// Cache object
typedef struct cacheObjectS {
```

```
        // MODELLING ARTIFACTS
        vmimmcPortP nextPort; // next port (TRANSPARENT)
        memDomainP nextDomain; // next domain (FULL)
        memRegionP lastRegion; // last accessed (FULL)
        Uns32 mruKey; // access optimization
        cacheLineP mruLine; // access optimization
        cacheAccessInfo readInfo; // read access recording
        cacheAccessInfo writeInfo; // write access recording


        // TRUE CACHE CONTENTS
        Uns32 keys[CACHE_TAGS][CACHE_WAYS]; // set of keys for cache
        cacheLineP index[CACHE_TAGS][CACHE_WAYS]; // index into cache lines
        cacheLine lines[CACHE_TAGS][CACHE_WAYS]; // set of lines for cache
} cacheObject, *cacheObjectP;


        // Cache object link
        static VMIMMC_LINK_FN(cacheLink) {

                vmiPrintf(
                        "\n\%s called for \%s\n",
                        FUNC_NAME,
                        vmimmcGetHierarchicalName(component)
                );


        cacheObjectP cache = (cacheObjectP)component;
        vmimmcPortP nextPort = vmimmcGetNextPort(component, "mp1");
        memDomainP nextDomain = vmimmcGetNextDomain(component, "mp1");

        // sanity check that we know whether we are in transparent or full
        // mode

        VMI_ASSERT(
                !(nextPort && nextDomain),
                "\%s: expected either nextPort (transparent)"
                "or nextDomain (full), not both",
                FUNC_NAME
        );


        // set the next connected MMC model port
        cache->nextPort = nextPort;
        cache->nextDomain = nextDomain;

        if(nextPort) {
```

```
                vmimmcAttrCP attrs = vmimmcGetPortAttrs(nextPort);


        // set transparent functions to call on a miss
        if(attrs) {
                cache->readInfo.missCB = attrs->readNTransparentCB;
                cache->writeInfo.missCB = attrs->writeNTransparentCB;
                }
        }
}
```

4.3.5 Memory models

The ICM API allows the OVP models to be exported into a SystemC environment. There are two levels at which the ICM API can be used: C and C++. It is the C++ API that is utilized in the SystemC environment. Once exported, an OVP model can be controlled from the SystemC interface by, for example, allowing it to be clocked one instruction at a time. Listing 4.5 gives the details about memory model implementation. Memory models are connected via its ICM interface to the ICM interface of the processor. In Listing 4.5, the configuration of the memory model and its connectivity to bus and processor is also given.

Listing 4.5: Implementation of the memory models

```
int main(int argc, char ** argv) {
        // initialize CpuManager
        icmInit(0, 0, 0);

        // create a processor
        icmProcessorP processor = icmNewProcessor(
                "cpu1", // CPU name
                "Cortex-A8", // CPU type
                0, // CPU cpuId
                0, // CPU model flags
                32, // address bits
                model, // model file
                "modelAttrs", // morpher attributes
                0, // simulation attributes
                0, // user-defined attributes
                semihosting, // semi-hosting file
                "modelAttrs" // semi-hosting attributes
        );

        // create the processor bus
```

```
        icmBusP bus = icmNewBus("bus", 32);

        // connect the processor busses
        icmConnectProcessorBusses(processor, bus, bus);

        // create two simulated memories for low and high regions
        icmMemoryP memory1 = icmNewMemory("mem1", ICM_PRIV_RWX, 0x003fffff);
        icmMemoryP memory2 = icmNewMemory("mem2", ICM_PRIV_RWX,
        0xffffffff-0x00401000);

        // map the address range 0x00400000:0x00400fff externally to the processor,
        // read only

        icmMapExternalMemory(bus, "external", ICM_PRIV_R, 0x00400000, 0x00400fff,
        extMemReadCB, extMemWriteCB, 0
        );

        // connect memories to bus
        icmConnectMemoryToBus(bus, "mp1", memory1, 0);
        icmConnectMemoryToBus(bus, "mp2", memory2, 0x00401000);

        // show the bus connections
        icmPrintBusConnections(bus);

        // load the processor object file
        icmLoadProcessorMemory(processor, argv[1], False, False, True);
        Bool done = False;
        while(!done) {
        Uns32 currentPC = (Uns32)icmGetPC(processor);


        // disassemble instruction at current PC
        icmPrintf(
        "0x\%08x:\%s\n", currentPC,
        icmDisassemble(processor, currentPC)
        );


        // execute one instruction
        done = (icmSimulate(processor, 1) != ICM_SR_SCHED);
        // dump registers
        icmDumpRegisters(processor);
        }

        // free simulation data structures
        icmTerminate();
        return 0;
}
```

Fig. 4.3 gives a detail view about the connectivity between ARM Cortex-A8 processor and its cache models and the main memory. Processor model is connected to its cache models via VMI interface. L1 to L2 cache connection is also through VMI interface, whereas memory model is connected via its ICM interface to the VMI interface of the L2 cache.

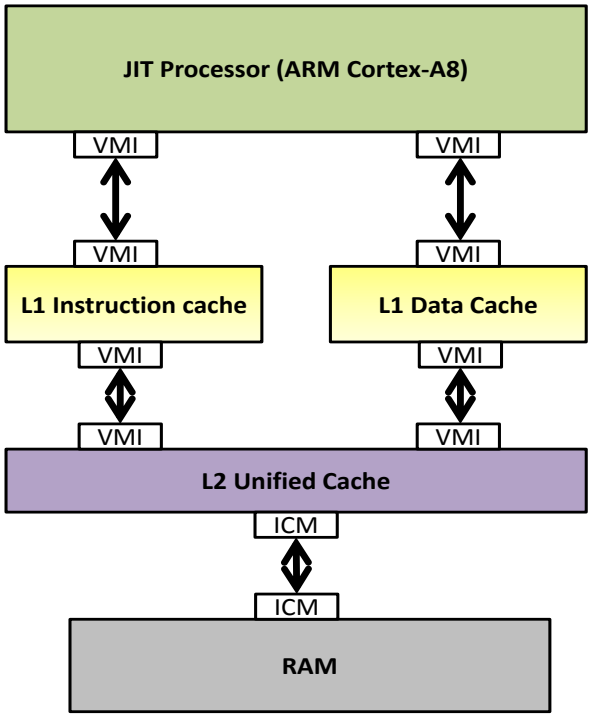

Figure 4.3: Cortex-A8 interface implementation

4.4 OVP in SystemC/TLM environment

Components reuse becomes mandatory in order to design challenge and design time. This requires design methodologies for inter IP communication and implementation. TLM2.0 provides new level of performance and interoperability. With TLM2.0 it is possible to enable models from different vendors to work together in a virtual platform. The OVP provides C++ interface to encapsulate their models in the SystemC environment. New developments have been made

to make OVP models work in TLM2.0 compliant SystemC platforms. The availability of SystemC TLM2.0 technology to use with OVP CPU models allows the encapsulation of OVP models in existing TLM2.0 compliant SystemC platforms, thereby solving the model interoperability issue and enabling fast solutions for successful deployment of virtual platforms by hybrid simulation of OVP and SystemC. To integrate the existing OVP model, SystemC wrappers are written for communication.The conventional APIs in OVP are built in C. To make TLM2.0 compliant, SystemC wrappers are used by several new classes in which the conventional C routines for the models are called. These classes build the wrapper around the binaries of the OVP processor, peripheral, memories and bus models allowing them to be exported to an outer simulation environment other than OVP. Once exported to SystemC environment, these models can then be controlled from the SystemC interfaces. Of the various abstraction levels provided by TLM2.0, it is the loosely timed modeling that gives a higher performance. It enables processes to run ahead of simulation time (temporal decoupling) and uses a quantum keeper. It is this abstraction level on which wrappers have been built so that the models could be run as fast as possible. Features like Direct Memory Interface (DMI) are used to provide direct pointer to the memory in the target bypassing the sockets in the transport calls enabling a faster simulation needed by the application. The processor has the option to invalidate DMI in which the transport call goes over the bus. The wrappers are supported for TLM2.0 blocking transport interface with timing annotation.

4.4.1 OVP inside TLM2.0

The wrapper used around the OVP processor model in the TLM2.0 environment are generic in nature. These wrapper can be manipulated for any kind of processor under test. These wrappers gives free running of each processor for a large number of instruction rather than putting them all in a bottleneck. SC_MODULE defines the class of each generic wrapper for the processor model. Fig.4.4 shows the detail of the wrapper. The implementation of the bus wrapper is shown in Fig.4.5. In order to mask the IP at TLM level, first C++ wrapper are built which can collect every move of a processor, bus etc., inside separate classes, as shown in Fig.4.4.

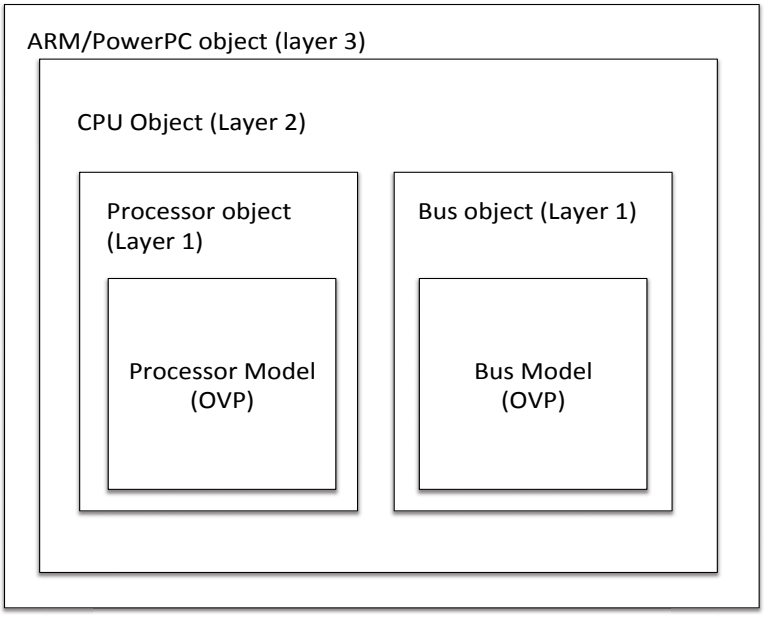

Figure 4.4: Wrapper configuration

These classes (wrappers) assess the OVPsim core for the functionality of the respective model. The outer SC_MODULE calls objects of these processor and bus classes. Based on this hierarchy of wrappers the module of processor shown in the Fig.4.5 has objects of the bus instantiated inside it. This allows mapping of the OVP processor address space to a local OVP memory/peripheral (through OVP Bus) as well as an external memory or peripheral with TLM2.0 target socket. In order to connect to an external memory/peripheral, a portion of address space of the local OVP bus, directly connected to the OVP processor is bridged to another bus (TLM Bus shown in figure) over which read/write callbacks are registered. Initiator sockets are opened on the processor model. Any access to this TLM bus address space which is mapped to an external memory/peripheral will trigger these read/write callback functions on the TLM bus, indirectly connected to the processor. The callback functions then create the appropriate transaction request and forward the transport call with its generic payload over the initiator

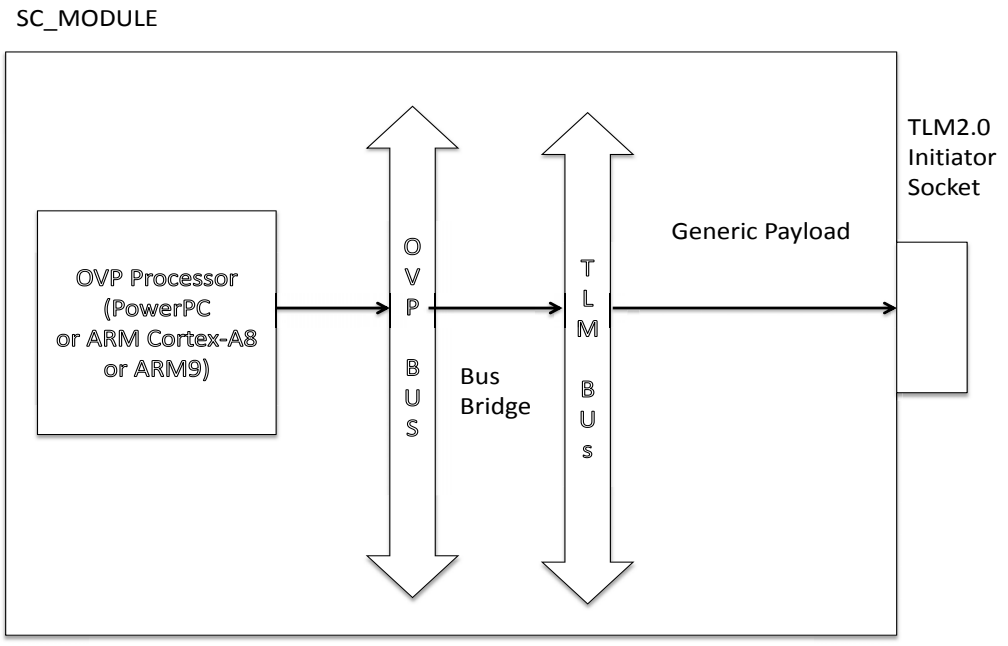

Figure 4.5: Wrapper implementation

sockets. Fig.4.6 gives the full system of the OVP and its wrappers.

This is a generic wrapper put around CPU models and is used in a processor configuration specific layer to create specific processor wrappers like that for ARM, PowerPC etc., which is then instantiated into the SystemC platform. The processor thus, on encountering an instruction that do a load/store to/from memory location on the bus, will call a function in the wrapper code which in turn issues the necessary blocking transactions on the bus. Wrappers for the peripheral model are also constructed in a similar fashion using the read/write callbacks registered on the bus connected to the peripheral model within an SC_MODULE. The TLM2.0 wrapper also provides a bus decoder with a configurable number of initiator and target sockets which are used to forward the transaction arriving on its target port to the proper initiator port based on the bus address map. The SystemC environment then calls the OVP simulator through this wrapper. Proper synchronization between the two simulators needs to be maintained to

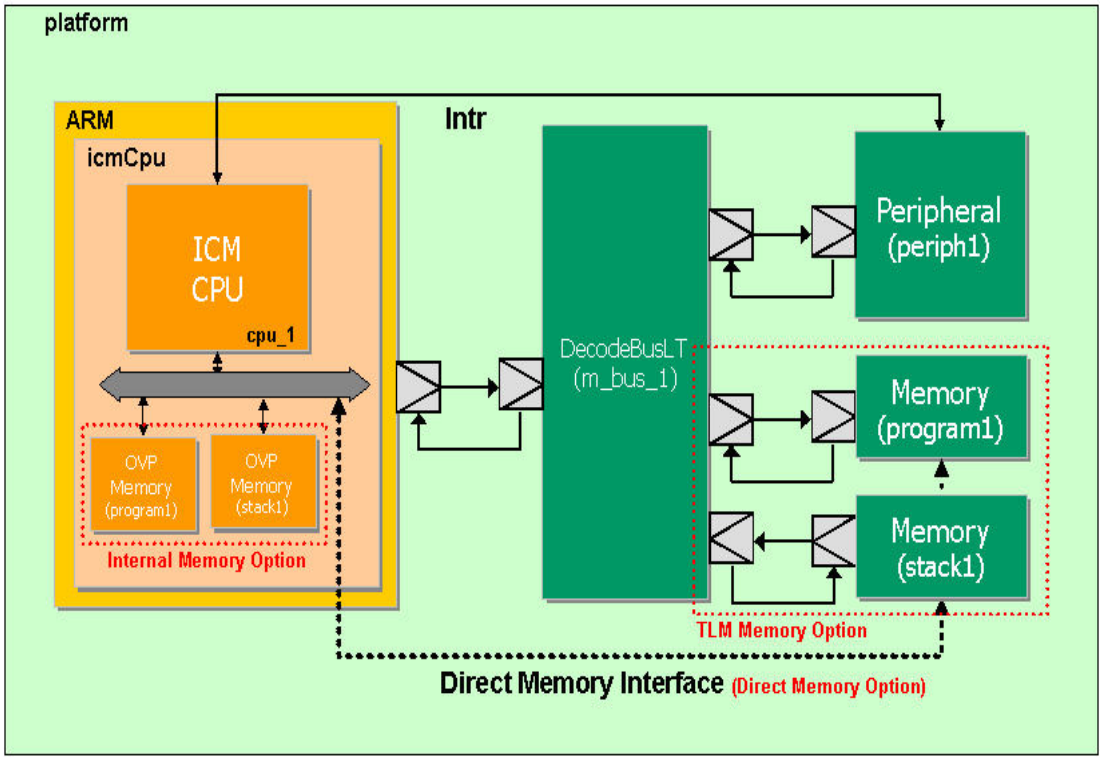

Figure 4.6: Full system implementation: Source [89]

achieve correct working of the models in the platform. As the simulation starts, each processor runs from a SystemC thread. The thread executes IPQ instructions on the processor without advancing SystemC time, where the function call asking the processor to simulate for IPQ instructions is from OVP environment through the wrapper. When the allotted instructions have completed, the thread calls SystemC wait() to advance time. The OVP simulator synchronizes with the SystemC simulation kernel every time the quantum is over. Thus each processor executes a number of instructions at a time in a round-robin schedule. Based on this background, a wrapper is prepared to enable OVP models to communicate with Open SCML based models. Listing 4.6 shows an view about the OVPSim implemenation inside SystemC/TLM.

Listing 4.6: Instantiation of the memory models

```
class simple : public sc_core::sc_module {
```

```
public:
        simple (sc_module_name name);

        icmTLMPlatform Platform;
        decoder <2,3> bus1;
        ram     ram1;
        ram     ram2;
        Cortex-A8       cpu1;
}; /* simple */

simple::simple (sc_module_name name)
        : sc_core::sc_module (name)
        , Platform ("icm", ICM_VERBOSE | ICM_STOP_ON_CTRLC
                                | ICM_ENABLE_IMPERAS_INTERCEPTS
                                | ICM_WALLCLOCK)
        , bus1("bus1")
        , ram1 ("ram1", "sp1", 0x100000)
        , ram2 ("ram2", "sp1", 0x10000)
        , cpu1 ( "cpu1", 0)
{

        // bus1 masters
        cpu1.INSTRUCTION.socket(bus1.target_socket[0]);
        cpu1.DATA.socket(bus1.target_socket[1]);

        // bus1 slaves
        bus1.initiator_socket[0](uart1.bport1.socket); // Peripheral
        bus1.setDecode(0, 0x90000000, 0x90000007);

        bus1.initiator_socket[1](ram1.sp1); // Memory
        bus1.setDecode(1, 0x0, 0xfffff);

        bus1.initiator_socket[2](ram2.sp1); // Memory
        bus1.setDecode(2, 0xffff0000, 0xffffffff);


int sc_main (int argc, char *argv[] ) {

        simple top("top");
        top.cpu1.setIPS(100000); // 1MHz


        sc_core::sc_start();
}
```

4.5 The simulation environment

In this section, we will show graphical representation of our simulation platform and also eclipse based environment for system-level co-simulation with detailed pictures.

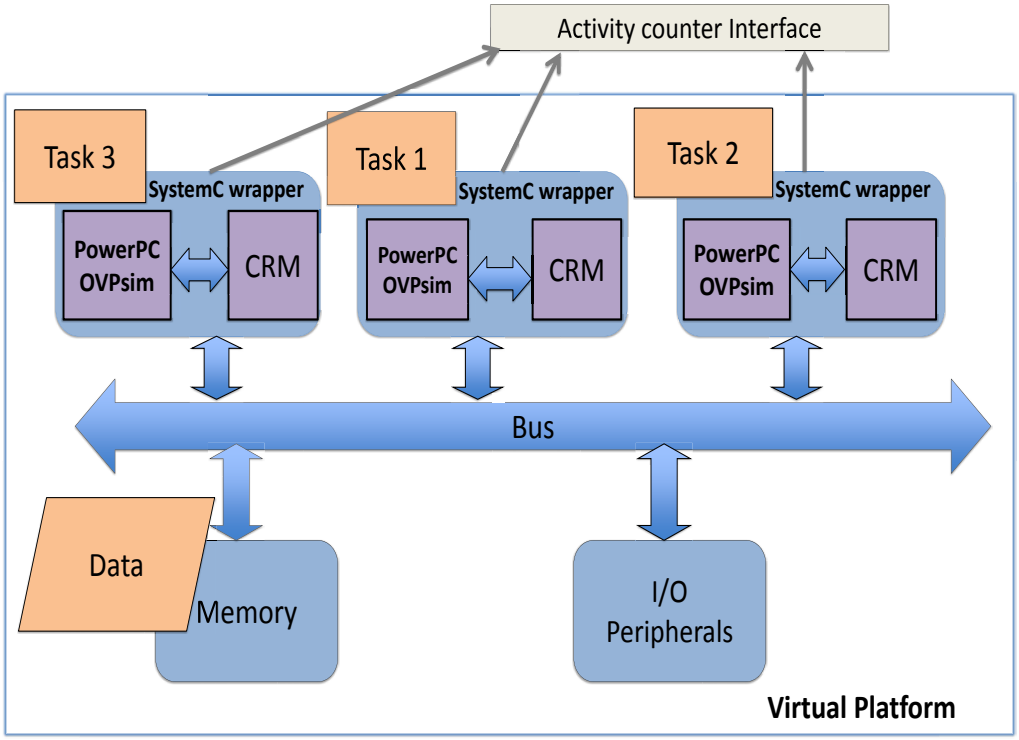

Figure 4.7: Proposed virtual platform prototype

Our proposed virtual platform prototype of a ARM Cortex-A8, ARM9, and PowerPC405 based architecture has been developed. This prototype uses different virtual hardware models as discussed in the previous section, a cache ratio monitor (CRM) provided with the virtual platform for cache miss rate, virtual memory model and the JIT for the target processor as shown in Fig. 4.7. From the CRM, we are able to determine the occurrences of the main activities which are recorded into activity counter interface such as cache miss rates. For all the three processors the following counters are used for different cache miss rates: read data miss, write data miss and read instruction miss. In a similar fashion, we extracted the activities for the homogeneous and heterogeneous MPSoC.

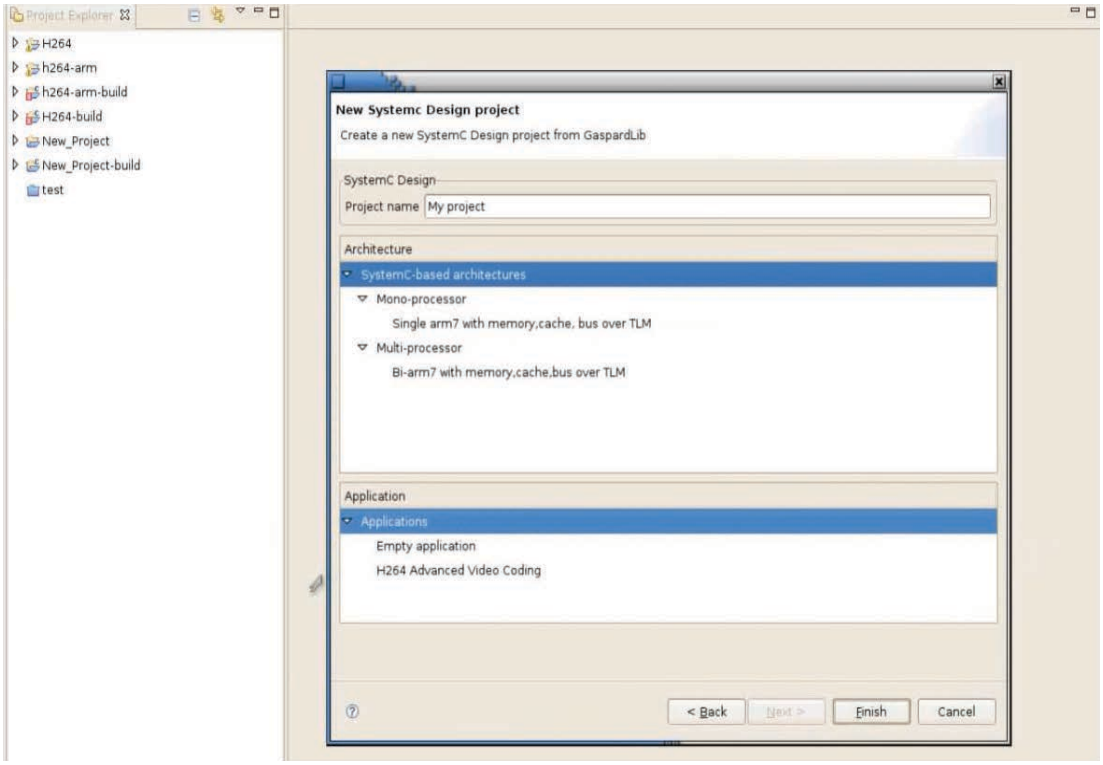

Figure 4.8: Preferences for processor and application setting

Fig. 4.8 shows the new SystemC project creation wizard in order to guide the hardware model developer in the initial instantiation of an IP and build settings. In this wizard, we are able to create a SystemC project with mono-processor and multiprocessor architecture and at the same time, we are able to port the use case application into the eclipse framework for hardware/software co-simulation. Sample projects are provided (e.g. ARM IP's, memory and multimedia applications) to quickly show how our eclipse environment looks like in Fig. 4.9. Fig. 4.9 also shows how to run the application upon the simulated platform like the real platform and also to debug the embedded application. Fig. 4.10, Fig. 4.11 and Fig. 4.12 show the different activity counters simulation results for a mono-processor architecture, which are needed for the power models to estimate the power consumption and to optimize the application for a better architectural solution. In the Fig. 4.13, we present the results of multiprocessor architecture running an application and displaying its timing details and total number of instructions.

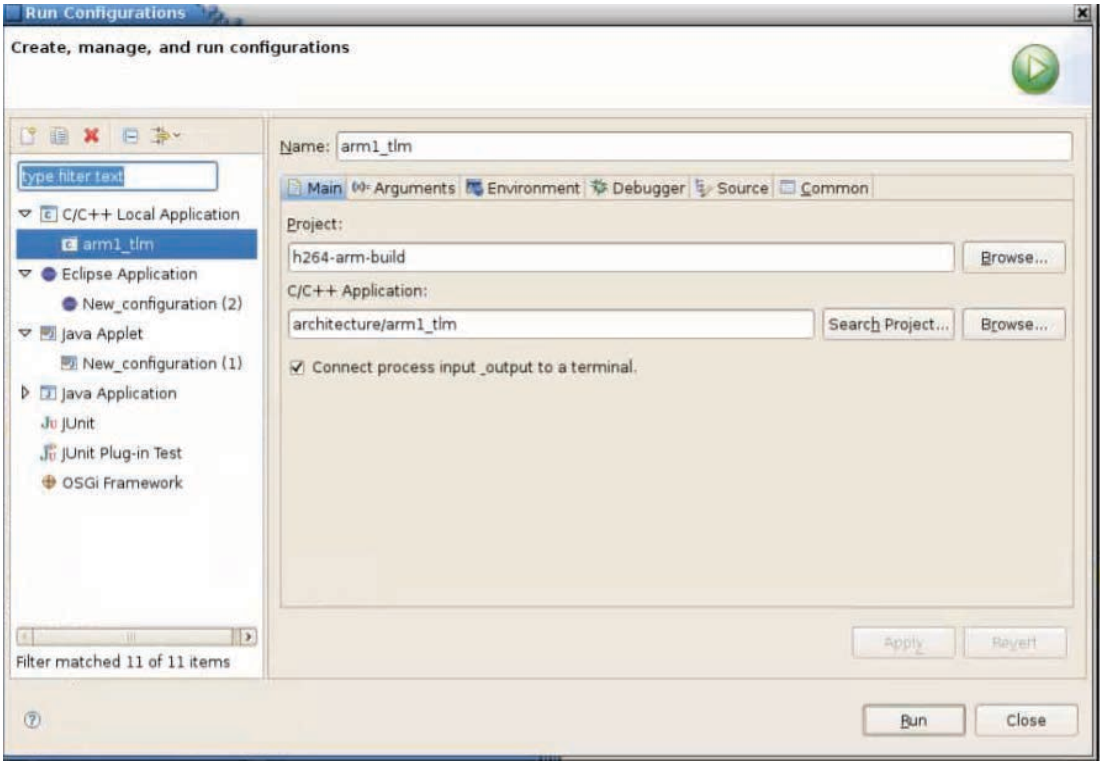

Figure 4.9: Hardware/Software co-simulation

4.6 Experimental results

In this section, the effectiveness of our proposal is illustrated through the JPEG encoder application performed on PowerPC based MPSoC and we also presents the comparison of proposed MPSoC environment with JIT/TLM and ISS/TLM level in terms of estimation speed and modeling efforts.

4.6.1 JPEG algorithm

In order to simulate the platforms, there is a need to choose proper application which could be executed on the processor. The choice of application should be such that the workload on the processor is quite high. Baseline JPEG Decoder is chosen as a benchmark application for our current simulation framework. Joint Photographic Experts Group or in short, JPEG is a widely used image compression technique. It is used in image processing systems such as copiers, scanners

```
cacheDestructor called for mmcL1I

READ ACCESSES:
  HITS      :     630,715,900
  MISSES    :         885,066
  1-byte    :               0
  2-byte    :               0
  4-byte    :     631,600,966
  8-byte    :               0
  N-byte    :               0 (0 bytes, average size=0.0 bytes)
  TOTAL READ :    631,600,966
  TOTAL BYTES:  2,526,403,864

cacheDestructor called for mmcL1D

READ ACCESSES:
  HITS      :     137,899,372
  MISSES    :          30,741
  1-byte    :       3,863,283
  2-byte    :       8,375,004
  4-byte    :     125,691,826
  8-byte    :               0
  N-byte    :               0 (0 bytes, average size=0.0 bytes)
  TOTAL READ :    137,930,113
  TOTAL BYTES:    523,380,595

WRITE ACCESSES:
  HITS      :     103,954,070
  MISSES    :          38,618
  1-byte    :       3,363,959
  2-byte    :       2,294,128
  4-byte    :      98,334,601
  8-byte    :               0
  N-byte    :               0 (0 bytes, average size=0.0 bytes)
  TOTAL WRITE:    103,992,688
  TOTAL BYTES:    401,290,619
```

Figure 4.10: L1 cache result after the simulation

```
cacheDestructor called for mmcL2

READ ACCESSES:
  HITS      :           1,328
  MISSES    :         953,097
  1-byte    :               0
  2-byte    :               0
  4-byte    :               0
  8-byte    :               0
  N-byte    :         954,425 (30,541,600 bytes, average size=32.0 bytes)
  TOTAL READ :        954,425
  TOTAL BYTES:     30,541,600

WRITE ACCESSES:
  HITS      :          39,476
  MISSES    :           3,244
  1-byte    :               0
  2-byte    :               0
  4-byte    :               0
  8-byte    :               0
  N-byte    :          42,720 (1,367,040 bytes, average size=32.0 bytes)
  TOTAL WRITE:        42,720
  TOTAL BYTES:     1,367,040
```

Figure 4.11: L2 cache result after the simulation

and digital cameras. A JPEG decoder is capable of reconstructing image data from a stream of compressed image data. This requires that some transformations be applied to the compressed image data. This results in the reconstruction of the image data. The fact that this coding method forms the basis for all DCT-based JPEG decoders makes it an interesting. For that reason it was selected to be implemented in this project. JPEG decoder is a streaming multimedia application which has a degree of parallelism and consists of 5-6 tasks. The JPEG decoding process is graphically depicted in Fig.4.14. Before the operations performed by the decoder are explained, we look at the encoder. The JPEG encoder divides an image in blocks of 8 by 8 pixels. The encoder then has a number of blocks, which when placed in the right order, form the original image. The encoder applies a number of operations on each of these blocks. These operations include a discrete cosine transform, quantization, zigzag scan and variable length encoding. The result of these operations, and of the encoder, is a compressed image.

The decoder reverts the transformations applied by the encoder to the image data. The decoder takes the compressed image data as its input. It then subsequently applies following operations to the compressed image.

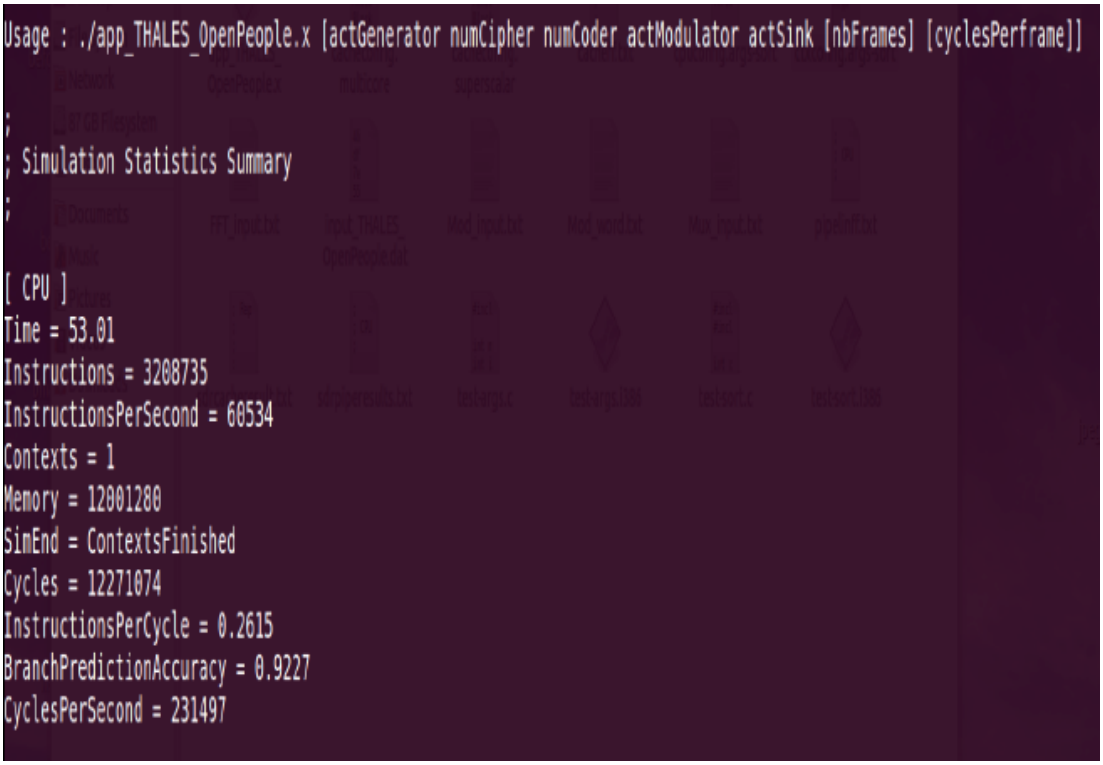

Figure 4.12: IPC simulation results

- Variable Length Decoding (VLD)
- Zigzag scan (ZZ)
- De-quantization (DQ)
- Inverse Discrete Cosine Transform (IDCT)
- Color Conversion
- Reordering

The decoder then obtains the reconstructed bitmap image. The compressed image data forms a byte stream input for the decoder. This byte stream contains so called markers. A marker is a two-byte combination, which identifies a structural part of the compressed image data. The incoming bit stream is parsed to get header information and image data based on the markers and various transformations are then applied.

```
Info
Info ------------------------------------------------
Info CPU '/cpu1' STATISTICS
Info   Type                  : arm
Info   Nominal MIPS          : 100
Info   Final program counter : 0x49a30
Info   Simulated instructions: 674,028,340
Info   Simulated MIPS        : 292.1
Info ------------------------------------------------
Info
Info ------------------------------------------------
Info CPU '/cpu0' STATISTICS
Info   Type                  : arm
Info   Nominal MIPS          : 100
Info   Final program counter : 0x49a30
Info   Simulated instructions: 477,122,391
Info   Simulated MIPS        : 206.8
Info ------------------------------------------------
Info
Info ------------------------------------------------
Info TOTAL
Info   Simulated instructions: 1,151,150,731
Info   Simulated MIPS        : 498.9
Info ------------------------------------------------
Info
Info ------------------------------------------------
Info SIMULATION TIME STATISTICS
Info   Simulated time        : 6.74 seconds
Info   User time             : 2.26 seconds
Info   System time           : 0.04 seconds
Info   Elapsed time          : 2.31 seconds
Info   Real time ratio       : 2.92x faster
Info ------------------------------------------------
```

Figure 4.13: Simulation results for multiprocessor architecture

4.6.2 Application task graph mapping for dual processor platform

The JPEG decoder application is made to run on the developed monoprocessor, homogeneous and heterogeneous multiprocessor platforms based on PowerPC405 architecture and then tested. The case is limited to two processor systems but could be extended to several cores depending on the workload of the application. In order to execute the same application on two processors, we need to partition the total tasks among two processors in such a way that each processor has almost equal computation and communication load. As seen from Fig.4.14, the various tasks in the decoder are performed one after the other. Thus the platform will be having processors which are active one after the other.

To select proper task partitioning for the application under experimentation, careful study of the application is done to find the match between JPEG decoder, the two processors platform and MPSoC platform with hardware accelerators.

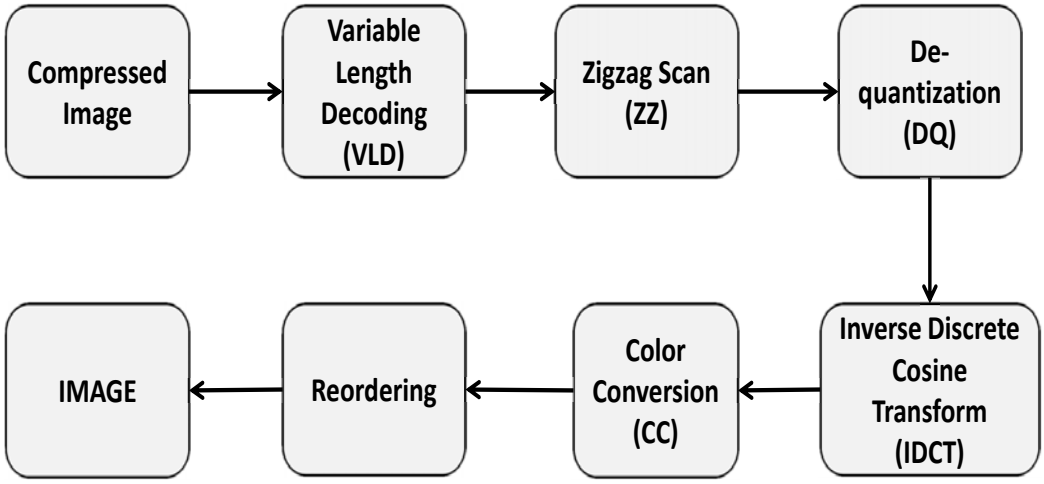

Figure 4.14: JPEG decoder flow

The compressed image data is connected to the VLD in the JPEG decoder. Therefore, the VLD must be incorporated in the first processor. As seen in Fig.4.14, re-ordering is connected to the output. In order to divide the ZZ, DQ, IDCT and color conversion over the two processors, the data consumption and production rate of the various parts of the system are looked upon. The VLD consumes data from the outside world and produces data in blocks. The zigzag scan, de-quantization and IDCT also consume and produce one block at a time. The color conversion and re-ordering requires one or more (up to 10) blocks before they can run. The color conversion however produces data in a block-by-block basis and sends this to the re-ordering unit which then produces output data. This implies that the communication over connection 2 of our two processor system is always in blocks. Thus every division of the JPEG decoder in two processors require the same data rate. The subdivision of the JPEG decoder does not influence the communication load of the system. Similarly, when we

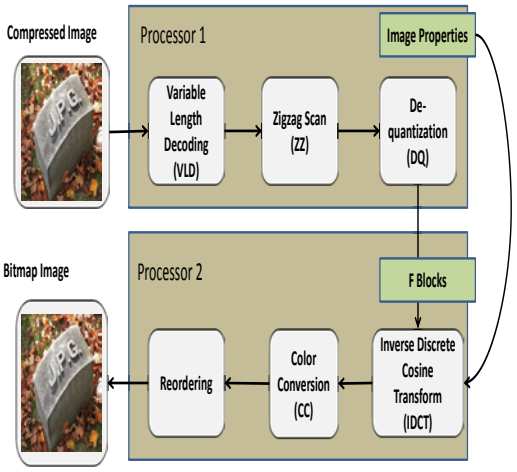

Figure 4.15: JPEG decoding process with 2 processors

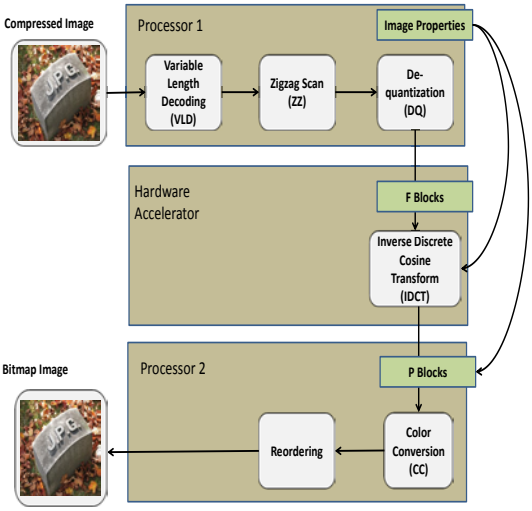

Figure 4.16: JPEG decoding process with 2 processors and hardware accelerator

use the JPEG application with a hardware accelerator, we port the IDCT on the hardware accelerator.

For the proper partitioning of decoder, first 2 processors platform and second 2 processors with a hardware accelerator platform, the computation load on two processors must be more or less same. This enables processor2 or hardware accelerator to start as soon as processor1 has produced one block. The survey result of the system load for various parts of the JPEG decoder is shown in Tables 4.2 and 4.3. The Tables 4.2 & 4.3 show that partitioning just before and after IDCT-function is the easiest to realize.

This choice enables almost 50-50% of load sharing among two processor platform and 50-20-30% with a hardware accelerator. It also has the advantage that the Huffman decoding and de-quantization tables required by the VLD and DQ units respectively do not need to be shared by both processors. Based on this task partitioning, the data flow among the two processors in the system is shown in Fig. 4.15 and Fig. 4.16.

Table 4.2: JPEG workload on 2 processors

Processor 1	Task Name	workload (%)
	Variable Length Decoding (VLD)	35
	Zigzag scan	5
	De-quantization (DQ)	10
Processor 2	Inverse Discrete Cosine Transform (IDCT)	20
	Color Conversion	15
	Reordering	15

Table 4.3: JPEG workload on 2 processors and hardware accelerator

Processor 1	Task Name	workload (%)
	Variable Length Decoding (VLD)	35
	Zigzag scan	5
	De-quantization (DQ)	10
Hardware Accelerator	Inverse Discrete Cosine Transform (IDCT)	20
Processor 2	Color Conversion	15
	Reordering	15

4.6.3 Performance estimation and simulation results of our proposed virtual platform

Validation of the proposed PowerPC based virtual platform is performed against PowerPC based ISS/TLM for the accuracy of the activities and the timing measurement. Table 4.4 summarizes the results for the cache activity inside the platform. The table shows JPEG application performing on a PowerPC with a cache. Note that, for this validation, the actual applications chosen are not so important, since the main goal is to verify the ability to accurately capture the activities when compared to ISS/TLM simulation. On comparison with Table 4.1, we are able to confirm that there is no accuracy loss incurred while changing to Instruction Accurate simulation from ISS/TLM. Besides, validation based on more and larger applications like audio/video encoding would be not feasible, due to the very long time necessary to perform on ISS/TLM simulation and extraction of the activities. From Table 4.4, it can be observed that the generalized platform shows a very good degree of accuracy, with an error within 0.23% of the cycle accurate level.

For speed comparison, ISS/TLM has been chosen as a reference. The reason

Table 4.4: Application miss rates for PowerPC based virtual platform

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

is that this represents the high-level methodology for system-level estimation commonly used at present. For this purpose, a dedicated ISS/TLM level model in SystemC has been implemented for the reference MPSoC architecture. The implementation has been as abstract as possible, since it exclusively represents the transactions occurring across the platform among the different IPs. In addition, communication is handled using bidirectional blocking interfaces.

Table 4.5: Timing comparison between proposed single processor environment and SoCLib environment

Application	JIT/TLM	ISS/TLM
JPEG 2000	3.22 sec	900 sec
H.264	56 sec	3600 sec
FFT	1.2 sec	300 sec
Downscaler	7.2 sec	2100 sec

For this experiment, a set of very large application examples have been chosen. The reason was to show that our system-level environment can handle very large examples and also to make the difference between ISS/TLM simulation and IA/TLM simulation execution speed more evident. The applications chosen are the image JPEG2000, the video compression codec H.264, FFT and Downscalar. The JPEG has been applied to a image size of 256x256. The H.264 has been

Table 4.6: Timing comparison between proposed two processors environment and SoCLib simulation environment

	JIT/TLM	ISS/TLM
Simulated Instruction/cycles	411322456 Instructions	342446113 cycles
Simulation time	3.66 sec	396 sec
Speedup	106.91	1

used for a video with a resolution of 176x144. The results of this comparison are reported in Table 4.5 and show an average speed improvement of 300x for JIT/TLM compared to ISS/TLM simulation. Besides, note that the advantage of JIT/TLM over ISS/TLM increases as the number of instructions increases. This confirms the capacity of JIT/TLM to be used for complex and real use-case scenarios.

Table 4.6 shows the results for two PowerPC processors running with JPEG application with an equal workload. From Table 4.6, we are able study that the proposed platform is 100x faster than the traditional ISS/TLM simulation.

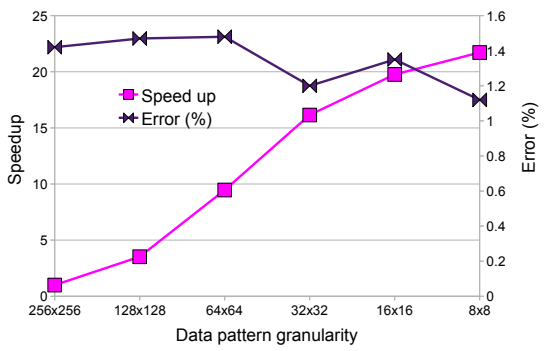

Figure 4.17: Power estimation error and speedup according to the data pattern granularity

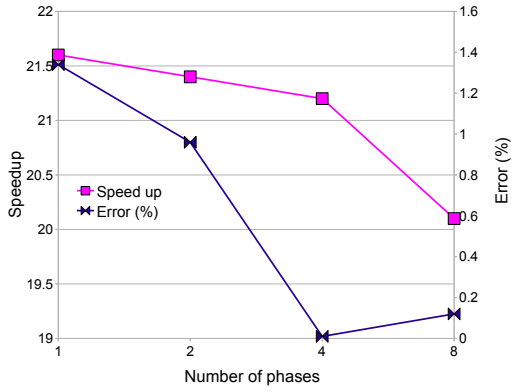

Figure 4.18: Power estimation error and speedup according to the number of phases

To simulate the full JPEG2000 application, it took 4 sec. per image. In order to speed-up the simulation without compromising accuracy, first, we decided to tune the data pattern granularity of the application and to analyse its effect in the accuracy and speed. To do so, we alter the input data size from 256×256 to 8×8 and we re-perform the simulation of the entire application. Fig. 4.17

summarizes the experimental results. We achieved a maximum speed-up of 21x with 8×8 data pattern compared to a full image simulation and the accuracy remained under 1.5% of error. Let us note that this approach is made possible by several specificities of the context, first by the fact we are targeting intensive parallel application that presents a low dependency between the data patterns. This is not a specific characteristic of the JPEG application, however a common feature of embedded signal processing applications. The second specificity is the parameters on which the elaborated functional power model relies. Mainly in our case, we bank on the *cache miss rate* which hide the handled data size while keeping the general behavior of the application. This is one of the main advantages of using FLPA methodology for power modeling.

As a second attempt to make power activities extraction faster, we used a sampling approach in order to reduce the total number of executed instructions. We vary the number of phases between 1 to 8 and we measured the simulation speed and the activity accuracy as shown in Fig. 4.18. In general, increasing the number of sampling phases will increase the accuracy while reducing the speed. This scenario is observed also in our case. We reached a maximum speedup of 21.5x with 1 phase compared to an entire application simulation and the accuracy remained under 1.4% of error. Due to the high task parallelism inside the JPEG application, few number of phases are sufficient for generating the pertinent activities for the power model. As all the JPEG tasks handle the same pattern data (blocks 8×8) in a repetitive fashion, the results of Fig. 4.17 and Fig. 4.18 were correlated. In other words, reducing the data pattern size will decrease the number of iterations inside the tasks. Thus, considering a 8×8 pattern size is equivalent to 1 sampling phase. For this reason, the simulation speed-ups in the two cases were similar. Nevertheless, for other applications the above two aspects can have an additional impact to increase the simulation speed.

4.6.4 Modeling efforts

Until now we have shown the usefulness of our approach in terms of simulation speed and accurately extracting the activities. However, this approach has proven its effectiveness also in terms of modeling effort. It allows designers to

develop, reuse and validate MPSoC systems in a short time. Table 4.7 presents the modeling effort needed to express in terms of Lines Of Code (LOC) to design an MPSoC system at JIT/TLM compared to ISS/TLM level. According to the results, the modeling effort with JIT/TLM is reduced by a factor of 28%.

Table 4.7: Modeling efforts

	JIT/TLM (LOC)	ISS/TLM (LOC)
Processor	220	1486
Cache	112	244
Memory	89	183
Interconnect	75	177
Total	571	2090
Reduction (%)	28	

4.7 Conclusion

This chapter has presented the proposed system-level virtual platform framework for co-simulation of hardware/software, reuse of platforms used for power estimation and to accurately extract the activities for the power models. This is done with the intent of maintaining system-level environment regardless of the platforms used. A PowerPC processor has been taken as a reference processor, while MPSoC has been taken as reference platform and JPEG application has been used as a benchmark application to validate the framework. Here, we emphasis again that the main contribution of this chapter is that we have proposed a fast JIT/TLM simulation framework to obtain the needed micro-architectural activities for the power models, which allows us to reach accurate estimates. With such proposed framework, the designer can explore several implementation choices: monoprocessor, homogeneous and heterogeneous multiprocessor systems. After proposing the system-level power estimation framework, we have validated the simulation speed against ISS/TLM simulation technique.

CHAPTER 5

POWER ESTIMATION TOOL AT SYSTEM-LEVEL (PETS) AND EXPERIMENTAL RESULTS

5.1 Introduction

As mentioned in the earlier chapters, the contribution of this thesis is to propose a tool based on fast and accurate power estimation methodology that operates at system-level. This chapter presents the proposed hybrid power estimation methodology, which is a combination of the power modeling approach proposed in Chapter 3 and the system-level framework proposed in Chapter 4. This methodology is proposed to work as a proof-of-concept for power estimation at system-level for a given MPSoC platform. Hence, we introduce a Power Estimation Tool at System-level (PETS) based on the hybrid power estimation methodology. This chapter also presents and analyses results of experiments carried out using PETS tool and compares them to other tools described in the literature such as Soft Explorer[1] and SimplePower[2]. This chapter also explains the power estimation flow

[1]http://www.softexplorer.fr/
[2]http://www.cse.psu.edu/ mdl/software.htm

for monoprocessor, homogenous and heterogeneous multiprocessor system for system designers. This comparison is performed using different types of benchmark applications in terms of accuracy and speed.

Section 5.2 starts with the flow of the hybrid power estimation methodology making profit from our power modeling methodology and our system-level co-simulation environment. Section 5.3 presents the flow of the PETS design for power estimation. Section 5.4 presents the experimental results and also offers a comparison of the power estimation speed of the proposed tool against the other existing tools.

5.2 Hybrid power estimation methodology

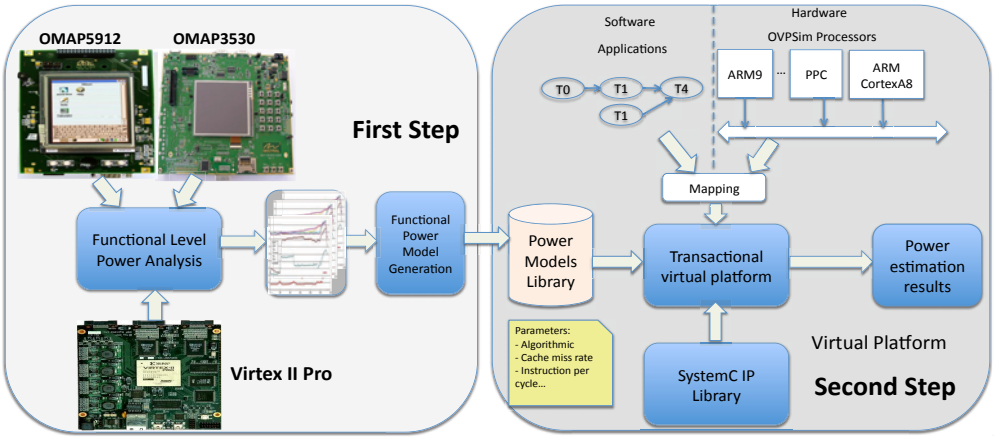

Figure 5.1: Power estimation flow

Fig. 5.1 is used as a reference throughout the chapter in order to ease the approach description.

We propose hybrid system-level power estimation methodology, which is divided into two parts as shown in Fig. 5.1. The **first part** concerns the power model elaboration for the system hardware components at functional level. In our framework, the FLPA methodology is used 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. The **second part** of the methodology defines the architecture of our hybrid power estimation methodology that includes the *Power Estimator Kernal* and *fast Just In Time (JIT) system-level simulator provided by OVPSim* as shown.

5.2.1 Part 1: Power model generation

As explained in the previous chapters, FLPA comes with few consumption laws, which are associated to 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 done at once. To estimate the power consumption of an MPSoC system, the first step is to divide the architecture into different functional blocks and then to cluster the components that are concurrently activated when the code is running. The second step is the characterization of the processor power consumption when the parameters vary. These variations are obtained by using some elementary assembly programs (called scenario) or built in test vectors elaborated to stimulate each block separately. Characterization are performed by measurements on real boards. Finally, a curve fitting of the graphical representation will allow us to determine the power consumption models by regression. The analytical form or a table of values expresses the obtained power models. This power modeling approach was proven to be fast and precise. In our work, this approach has been applied to model power consumption for processor and its memory systems.

5.2.2 Part 2: System-level environment development

As explained in the previous section, the second part contains a functional power estimator and a system-level simulator. The functional power estimator evaluates the consumption of the target system with the help of the elaborated power models from the first part. 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 JIT simulator. The combination of the above two components described at different abstraction levels (functional and virtual platform) leads to a hybrid power estimation that gives a better trade-off between accuracy and speed. The vital function of the proposed PETS tool is to offer a detailed power analysis by the means of a complete simulation of the application. This process is initiated by the functional power estimator through *the processor and application mapping interface* (Fig. 5.2). In this way, the mapping information is transmitted to the fast JIT simulator at system-level. In our previous framework with SoCLib [108], we presented an accurate TLM simulation technique that allows to evaluate the MPSoC performances. In this work, the processor is a JIT simulator provided by OVPSim wrapped inside a SystemC interface to communicate with the shared memory and the peripherals. In the power estimation step, the simulator collects the activities that are influenced by the application and the input data. At the end of the simulation, the values of the activities are transmitted to the power consumption models or power estimator kernal using *the activity counter interface* in order to calculate the global power consumption as illustrated in Fig. 5.1.

Hybrid power estimation methodology expects to receive two pieces of information from the user: a data (architecture) specification and a task (use case) specification. The data specification tells about the processor description (IPs), connection and their configuration. The IPs that can be used in the data specification must have a corresponding model in the SystemC IP library. The task specification contains information on how to use the design described in the data specification to be used. For example: first, the applications used for the spe-

cific test case; two, the way such applications have been mapped to the resources specified in the data specification; three, the time offset at which each application gets triggered; fourth, the input data; fifth, the I/O from which the data is received, and sixth, the I/O through which the data is sent out. As for the data specification case, the applications mentioned in the task specification must be present in the applications database. In simple words, the data specification contain the information about the architecture of the system under test and the task specification contains the application to be simulated on system. As of today, data specification and task specification are in the form of a file.

5.2.3 Engineering efforts

The main aim of this section is to quantify the engineering effort required by the hybrid power estimation methodology. For this purpose, the hybrid power estimation methodology is divided into three phases: a characterization phase, a SystemC IP development phase and an estimation phase.

The characterization phase is laborious and thus it requires a large amount of engineering effort. This phase includes generation of power models (Part 1) and development of SystemC IPs. As summarized in 5.2.1, the creation of power model implies characterizing each parameter for power. As summarized in 5.2.2, SystemC IPs phase consist of development and reuse of the IP's for the system-level environment where the impact of certain elements like caches, bus and transaction between the IPs are modulated. Although time-consuming, this characterization phase and SystemC IP phase are justified for the following three reasons: first, it is done only once; second, it gives high accuracy to the whole hybrid power estimation methodology since it is carried out by direct measurement; third, it is a shared activity, which helps to bring the engineering effort down. As opposed to the characterization phase, the estimation phase is very fast and interactive, and the engineering efforts are minimal as we reuse most of the IPs available in the open source libraries. In the estimation phase, power values are collected from the simulation environment and given to the power estimation kernel. Where, the estimation kernel calculates the total power. Based on these considerations, the overall engineering effort required by our methodology is low.

The reason is that JIT/TLM simulation relies on the availability of IPs.

5.3 PETS tool design flow

In order to verify power requirements during the design phase of a given system, power estimation has become an essential part of the design process. Due to the advances in process technology scaling as well as rising demands for computational performance and functionality, increasingly complex designs have to be handled in the power estimation process. Systems-on-chips (SoC) are typically composed of a large number of sub-components, each contributing to the overall power consumption. Furthermore, it can be observed that the power consumption of these devices is progressively more dependent on software applications, determining the utilization of system components as well as actuating available on-chip power management features. It is therefore favourable, to provide automated and reprogrammable power estimation resources not only to system architects and hardware designers but also to power-aware software application and operating system developers.

For the purpose of fast yet accurate software power estimation, system-level power estimation tool (PETS) are considered promising. With proposed tool, we achieve a considerable power estimation speed-up compared to Low-level-based methods. However, the time-consuming task of power model generation and required SystemC models adaptation for these complex platform power estimation has been best of our knowledge. In the context of system-level power estimation, the novel contributions of the proposed PETS tool are as follows:

- We propose a systematic, automated power estimation and modeling tool that automatically determines power of a given system under test and an application (provided in the library).
- We develop an automated technique for implementing this power model in system-level environment .
- A case study on a several state-of-the-art benchmarks are used to illustrates the benefits of our automated power estimation tool.

Flow of our standalone power estimation tool is given in the Fig.5.2. The

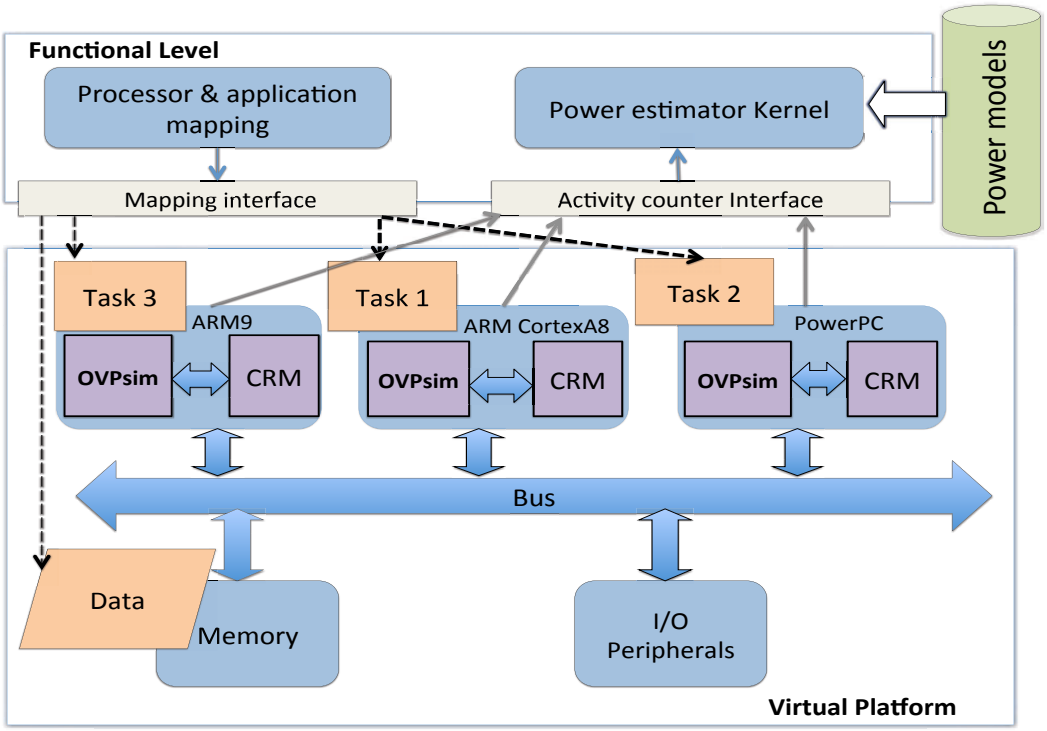

Figure 5.2: PETS tool

developed power models in Chapter 3 are loaded into power model library in the form of C code and by similar way the developed SystemC IP's, OVPsim module and benchmark application are loaded into their IP and application libraries. The user by using a script file can load the system, where in the script the description of the processor, type of power model and type of application to be ran are given. The virtual platform module runs the needed platform and application. The counter interface which collects the needed architecture data for the power and gives it to the power data which gives out the total power estimation of the particular benchmark application.

5.4 Experimental results

5.4.1 Power estimation accuracy of monoprocessor based platform

5.4.1.1 ARM Cortex-A8 based platform (OMAP3530)

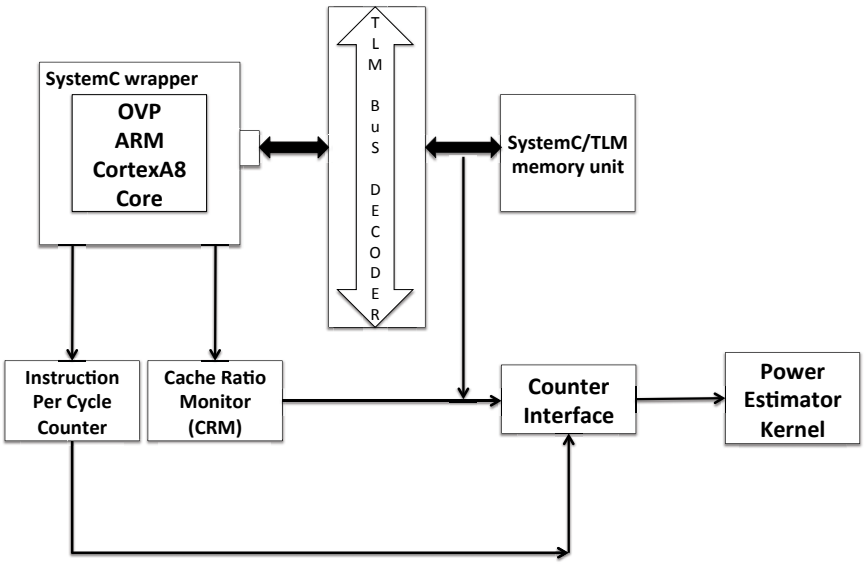

Figure 5.3: Mono-processor platform of ARM Cortex-A8

A system level prototype of a ARM Cortex-A8 based architecture has been developed. This prototype uses different virtual hardware models, a Cache Ratio Monitor (CRM) provided with the virtual platform for cache miss rate, and the JIT for the target processor and SystemC wrapper around the processor in order to capture the activities. In such prototype, OVP processor with a TLM 2.0 compliant SystemC wrapper is made to interact with simple TLM 2.0 target memories and other peripherals. This is done by connecting a processor model to a SystemC based bus decoder which has TLM 2.0 target and initiator sockets as shown in the Fig. 5.3. The bus decodes the incoming address and based on the address, forwards the transaction to one of its initiator port which is connected to TLM 2.0 target socket of the memory. Furthermore, the cache parameters and

the bus latencies are set to emulate the real platform behaviour. From the CRM, we are able to determine the occurrences of the main activities. This prototype uses different SystemC models especially the OVPsim for the target processor. A set of counters are injected into the simulator to determine the values of different cache miss rates: read data miss, write data miss and read instruction miss and Instruction Per Cycle (IPC) counter.

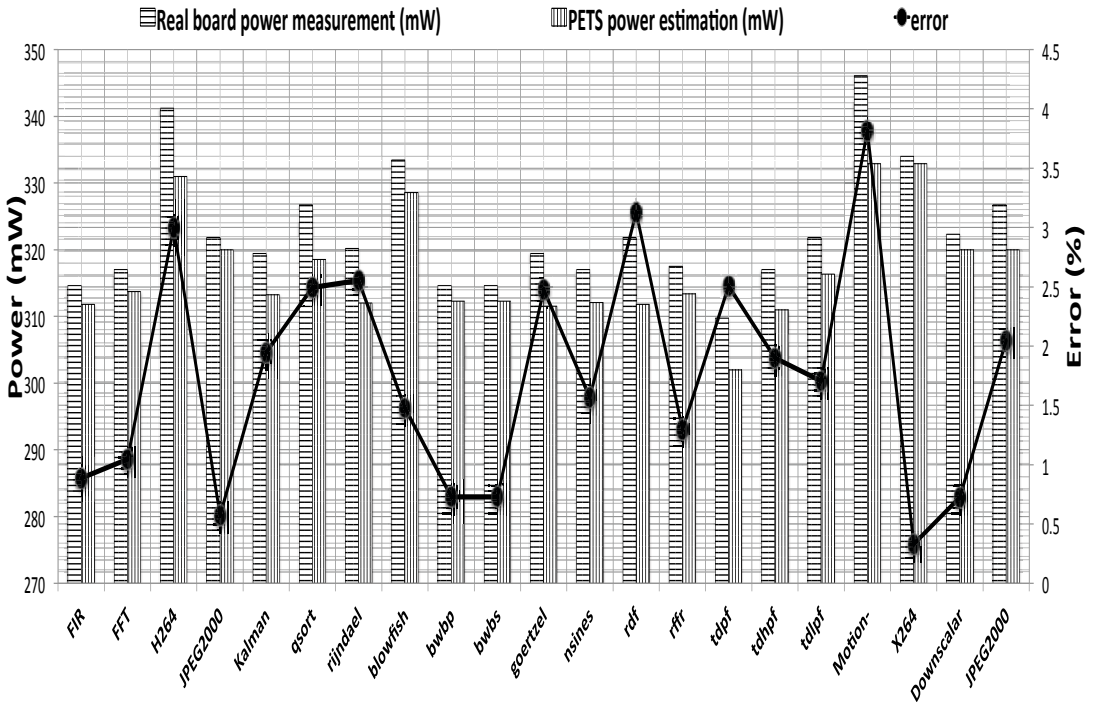

Figure 5.4: Power estimation accuracy vs real board measurement using ARM Cortex-A8 at 500 MHz

In the next step, we estimated the total power consumption of each task using the power model shown in Table 3.1. Fig. 5.4 illustrates the results and shows the comparison between the proposed PETS, and the real board measurements. First, our power estimator has a negligible average error equal to 1%, which offers better accuracy. This is due to better accuracy of the captured activities in the simulator. For this reason, we calculate again the average error without taking into account the static power. Our methodology produces a maximum error of 4.3% and the SoftExplorer gives around 9.4% in comparison with real

board measurement.

5.4.1.2 ARM9 based platform (OMAP5912)

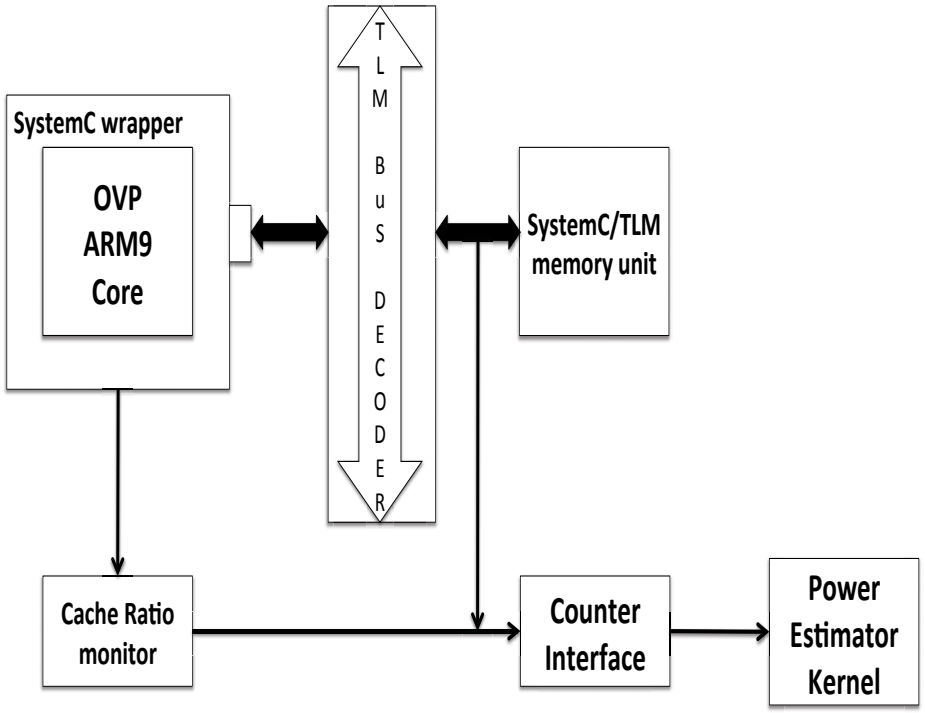

Figure 5.5: Mono-processor platform (ARM9)

Similar to the ARM Cortex-A8, PETS tool contains a virtual platform prototype of an ARM9 architecture, which has been developed with the help of SystemC and OVPsim processor. This prototype uses different virtual hardware models, a Cache Ratio Monitor (CRM) provided with the virtual platform for cache miss rate, and the JIT for the target processor and SystemC wrapper around the processor in order to capture the activities. In such prototype, OVP processor with a TLM 2.0 compliant SystemC wrapper is made to interact with simple TLM 2.0 target memories and other peripherals. This is done by connecting a processor model to a SystemC based bus decoder which has TLM 2.0 target and initiator sockets as shown in the Fig.5.5. Furthermore, the cache parameters and the bus latencies are set to emulate the real platform behaviour. From the

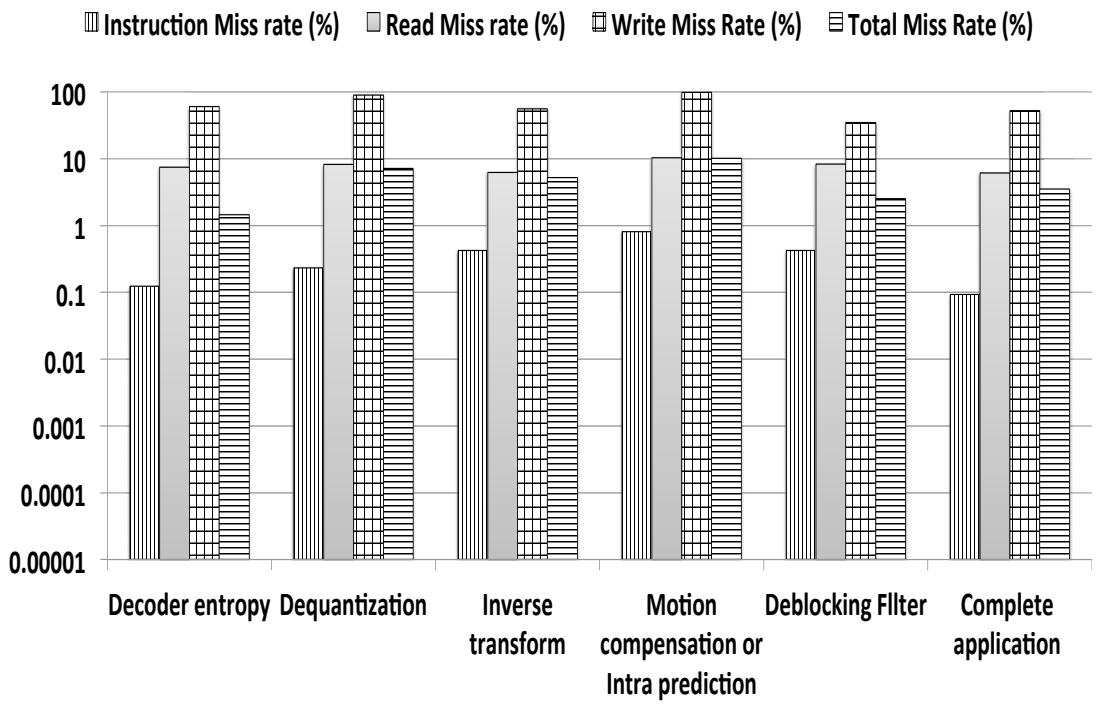

Figure 5.6: Cache miss rate for the H.264 application (ARM9 at 120 MHz)

CRM, we are able to determine the occurrences of the main activities. For the ARM9 processor the following counters are used for different cache miss rates: read data miss, write data miss and read instruction miss. In some of the cases, simulations have been carried out in backdoor mode. This is a way to access memory/peripheral in which the transaction request does not actually goes over the bus. As a main application, we used the H.264/AVC baseline profile decoder that supports intra and inter-coding, and entropy coding with Context-Adaptive Variable-Length Coding (CAVLC) as a benchmark for ARM9 processor. The H.264 decoder application consists of 5 main tasks: decoder entropy, dequantization, inverse transform, motion compensation or intra prediction and de-blocking filter.

Fig. 5.6 shows the detailed results of the activities fetched by the fast JIT-SystemC simulator for each task of the H.264 application for ARM9 processor. From these results several remarks can be drawn. First, we can notice that

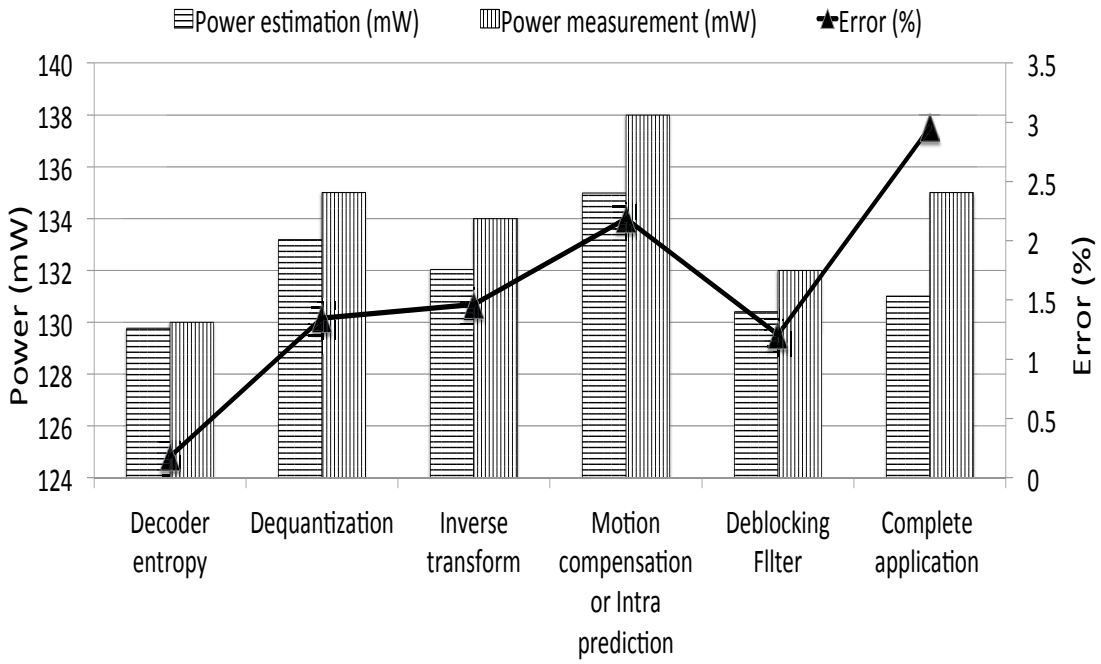

Figure 5.7: Power estimation accuracy for the H.264 application using ARM9 at 120 MHz

instruction cache miss rates and read data miss rates are very low when compared with write data miss rates. This is due to the reduced task kernel and data pattern sizes that are very low compared to the cache size (4 KB of instruction cache and data cache respectively), which decreases the access to the external memory and thus having a minimal effect on the dynamic power consumption. Second, the data write miss rates have a high impact on the total power consumption of the system. This is because of the algorithm's structure, which does not favour the reuse of data output arrays and the usage of write through cache policy. Therefore, the statistics collected in Fig. 5.6 could help in tuning the application structure for a better optimization of the system power consumption.

In the next step, we estimated the total power consumption of each task using the power model shown in Table 3.2. Fig. 5.7 illustrates the results and shows the comparison between the proposed methodology and the real board measurements. First, our power estimator has a negligible average error of 2%. This

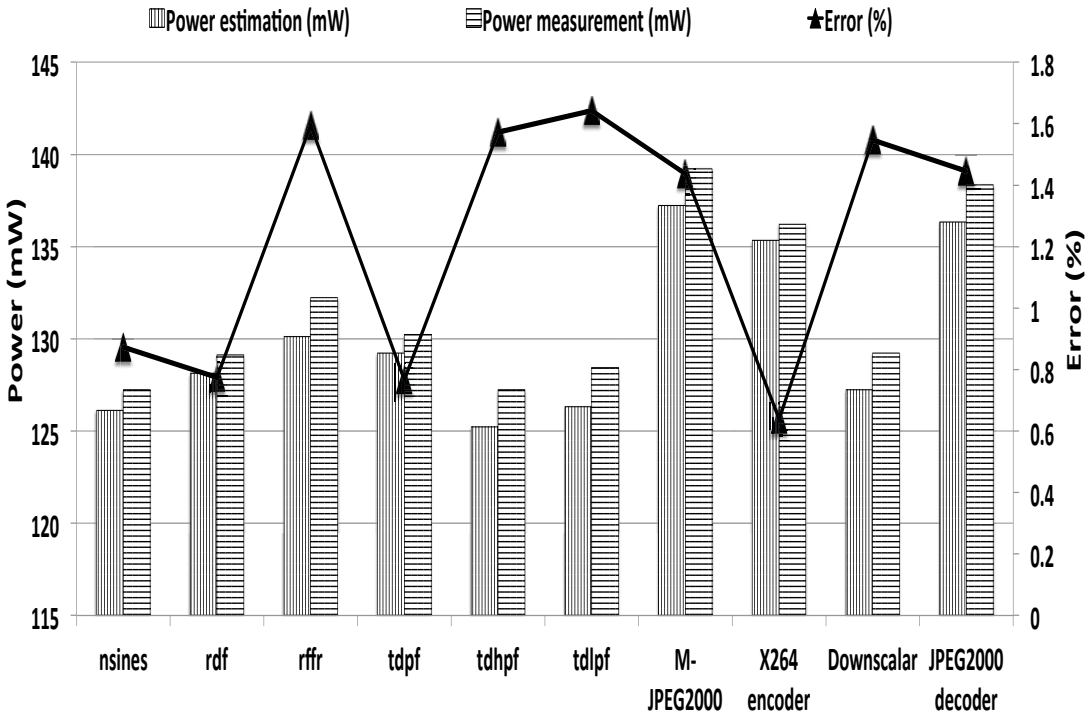

Figure 5.8: Power estimation accuracy vs real board measurement using ARM9 at 120 MHz

study offers a detailed power analysis for each task in order to help designers to detect peaks of consumption and thus to propose efficient mapping or optimization techniques. In order to evaluate the accuracy of our methodology, we carried out power estimation on several image & signal processing benchmarks. Fig. 5.8 illustrates the power results by showing the estimation accuracy between the proposed power estimation methodology and the real board measurements. Our proposed methodology tool has an negligible average error of 1.24% for ARM9, which is considered as a high accuracy level when compared to SoftExplorer's average error of 8% for ARM9 processor.

5.4.1.3 PowerPC based platform (Virtex II Pro)

A system level prototype of a PowerPC based architecture has been developed. This prototype uses different SystemC models especially the OVPsim for the

target processor. Similar to ARM9 model, a set of counters are injected into the simulator to determine the values of different cache miss rates: read data miss, write data miss and read instruction miss.

Fig. 5.9 shows the detailed results of the activities fetched by the OVP enabled fast SystemC simulator for each task of the JPEG application. 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 reduced task kernel and data pattern sizes that are very low compared to the cache size (16 KB), which decreases the access to the external memory and thus having a minimal effect on the dynamic power consumption. However, with the new sub-micron technologies the effect of the static consumption cannot be neglected. For this reason, a softcore processor such as the Microblaze comes with reconfigurable cache size to fit with the application requirements. Second, the data write miss rates have a high impact on the total power consumption of the system. This is because of the algorithm structure, which does not favour the reuse of data output arrays and the usage of write-through cache policy. Therefore, the statistics collected in Fig. 5.9 could help in tuning the application structure for a better optimization of the system power consumption. Detailed results from H.264/AVC decoder and several other multimedia benchmarks show a similar behaviour as like the JPEG application as shown in the Fig. 5.11 .

In the next step, we estimated the total power consumption of each task using the power models shown in Table 3.3 (SDRAM mapping). Fig. 5.11 illustrates the results and shows the comparison between the proposed PETS, SoftExplorer tool, and the real board measurements. First, our power estimator has a negligible average error equal to 1.59%, which offers better accuracy than SoftExplorer's with its average error of 4.32%. This is due to better accuracy of the captured activities in the simulator than the static analysis or rapid profiling of the code. The average error obtained here is negligible due to the dominance of static power. For this reason, we calculate again the average error without taking into account the static power. Our methodology produces an average error of 4.3% and the SoftExplorer gives around 9.4% in comparison with real board measurement.

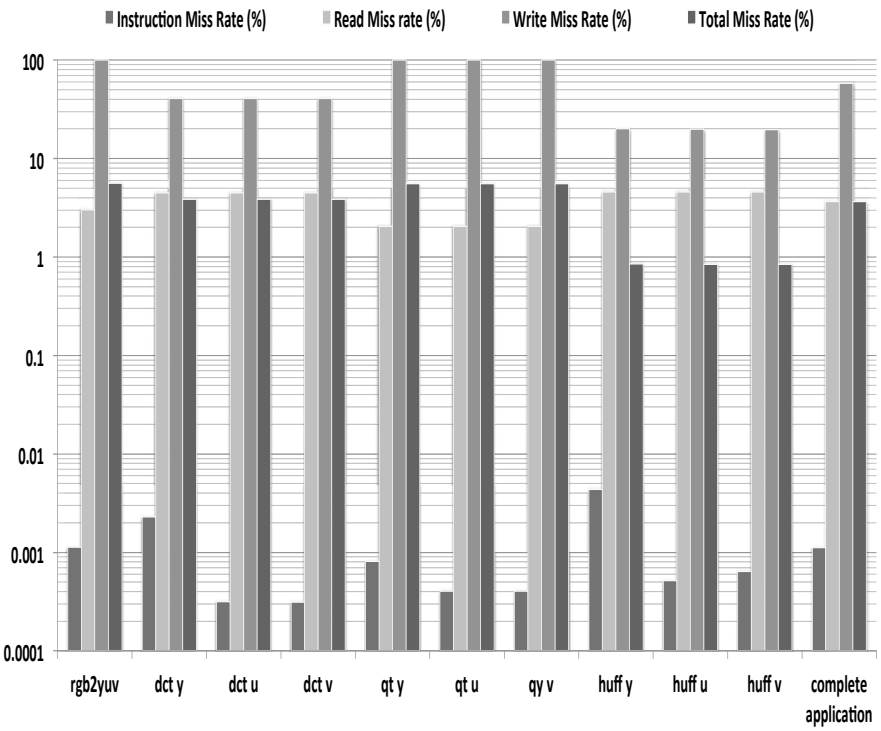

Figure 5.9: JPEG application cache miss rates

5.4.2 Homogeneous multiprocessor based platform

This study involves a multiprocessor architecture with two identical PowerPC processors to run several image & signal processing applications. Based on the available task partitioning of the JPEG, the application was split into two parts, each part being executed on a separate processor. The platform constructed to simulate such a system is illustrated in Fig.5.12. Each processor has its own program memory which contains the executable in .elf format. In the current framework, the cores communicate with each other via shared memory. This shared memory is used to transfer necessary information among the processors like the image properties consisting parameters as image size, number of components, sampling rate and the necessary blocks from core 1 to core 2 for computation purpose. The two processors synchronize via polling mechanism in which the semaphore present in shared memory is constantly polled by both processors. Thus processor 1 reads the input image from the memory. Each time processor

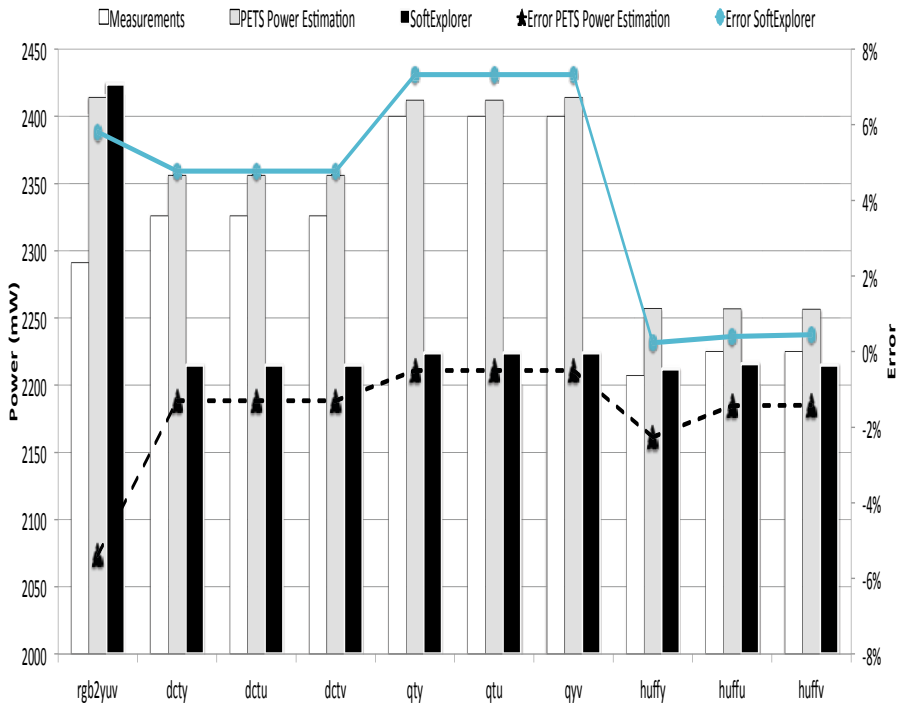

Figure 5.10: Power estimation accuracy

1 generates an 8x8 block after quantization step, it places the block into the shared memory and sets the semaphore to high. It then waits for this semaphore to set back to a low value, which is done by Processor 2. Processor 1 writes the block to the shared memory only when the semaphore is low. Processor 2 on the other hand, waits for the high value of semaphore. When semaphore is found high, it reads the block from the shared memory, and reset the semaphore to a low value so that next block could be written by Processor 1. Processor 2 then performs DCT on the block. When sufficient numbers of blocks are obtained, color-conversion and re-ordering is performed and the reconstructed data is stored back in the output memory. Similarly, we executed the JPEG application for 1 to 8 processors. To go further, we would like to address another important issues, i.e., the energy and execution time of the application while increasing the number of processors. To evaluate the impact of the number of processors on the execution time and total energy/power consumption, we executed the JPEG on

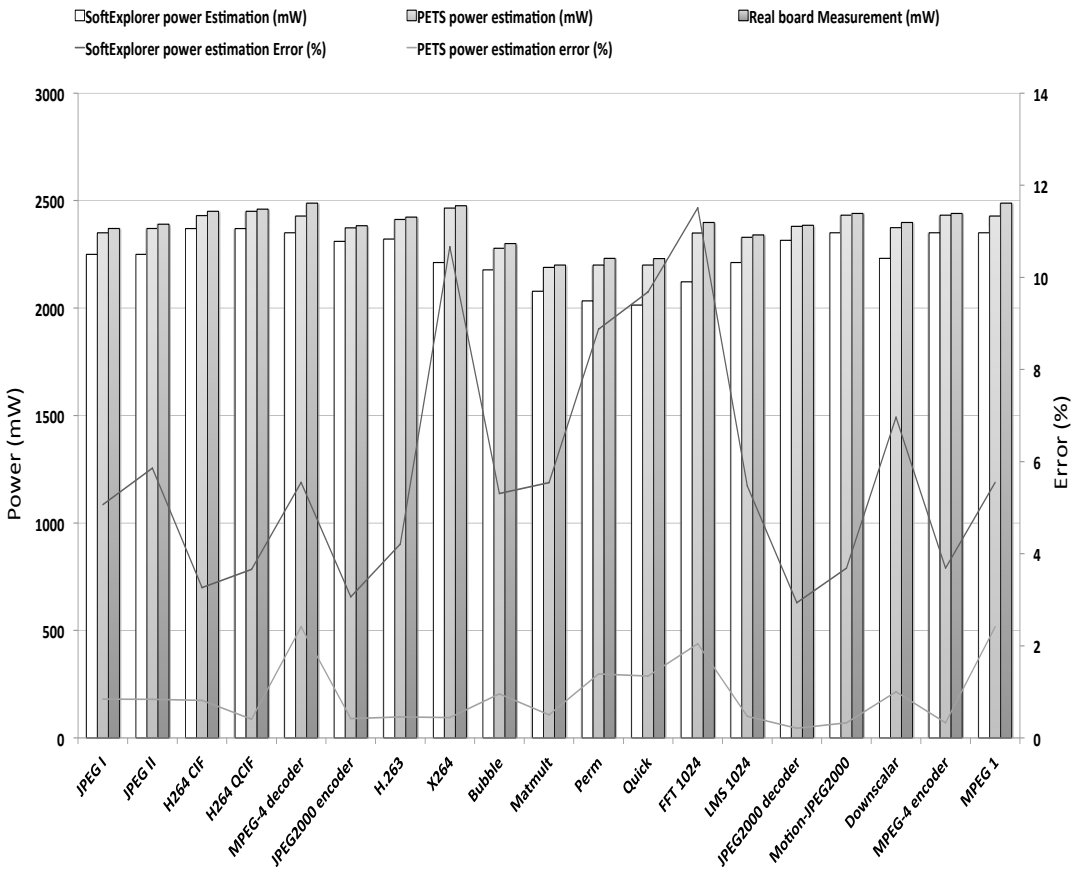

Figure 5.11: Comparison of power estimation accuracy for PowerPC based architecture)

systems with 1 upto 8 processors. The PowerPC frequency was set to 300MHz and the PLB frequency to 100MHz. All the processors execute the same workload but on different image macroblocs. Fig. 5.13 reports the execution time in ms and the total energy consumption in mJ for JPEG application. Fig. 5.13 shows that, for the implemented parallel JPEG application, 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 speed-up as waiting cycles are added to the execution time. In terms of energy consumption, we observed that until a certain number of processors, the total system energy consumption decreases

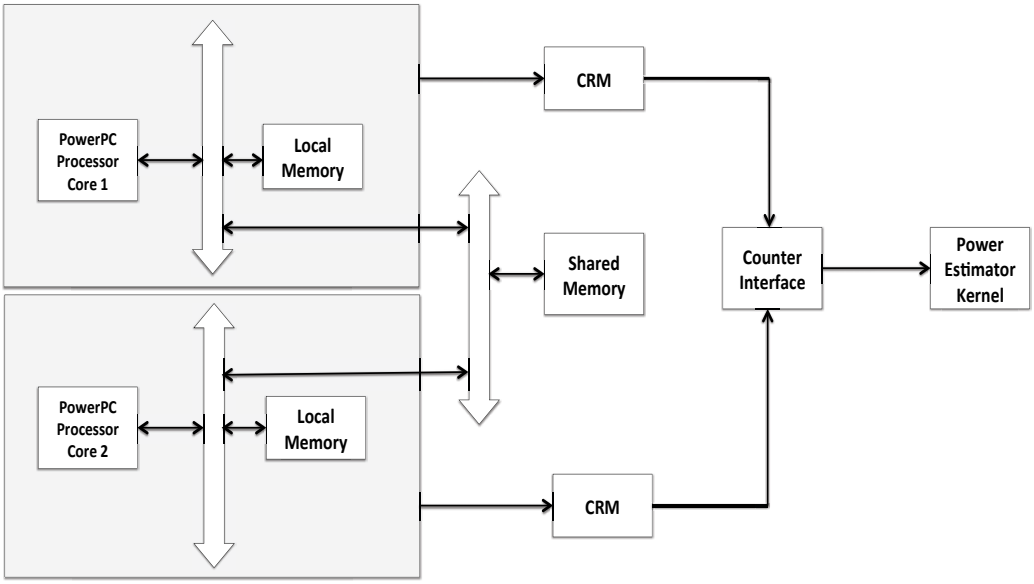

Figure 5.12: Dual core PowerPC platform

as the execution time is reduced, and then it tends to stabilize as the system performance improves. But increasing the number of processors over a certain limit tends to be ineffective, as it just adds new conflicts at the bus level, leading to more waiting cycles.

From Fig. 5.14, we are able to conclude that proposed PETS tool is accurate and efficient for a dual processor system with its negligible average error of 0.89% when compared to the real board measurements. Compared to real board energy measurements, PETS tool achieved an error of 0.79% and 3.49% respectively for one and two processors. This accuracy is obtained because of three main reasons. First, power models are extracted from real board measurements. Second, our tool considers the synchronization part while using multiprocessor system. Finally, additional activities that are intrinsic in parallel processing such as shared data communication overheads are accurately evaluated by using our JIT simulator. The above mentioned reasons encourage us to consider architectures with

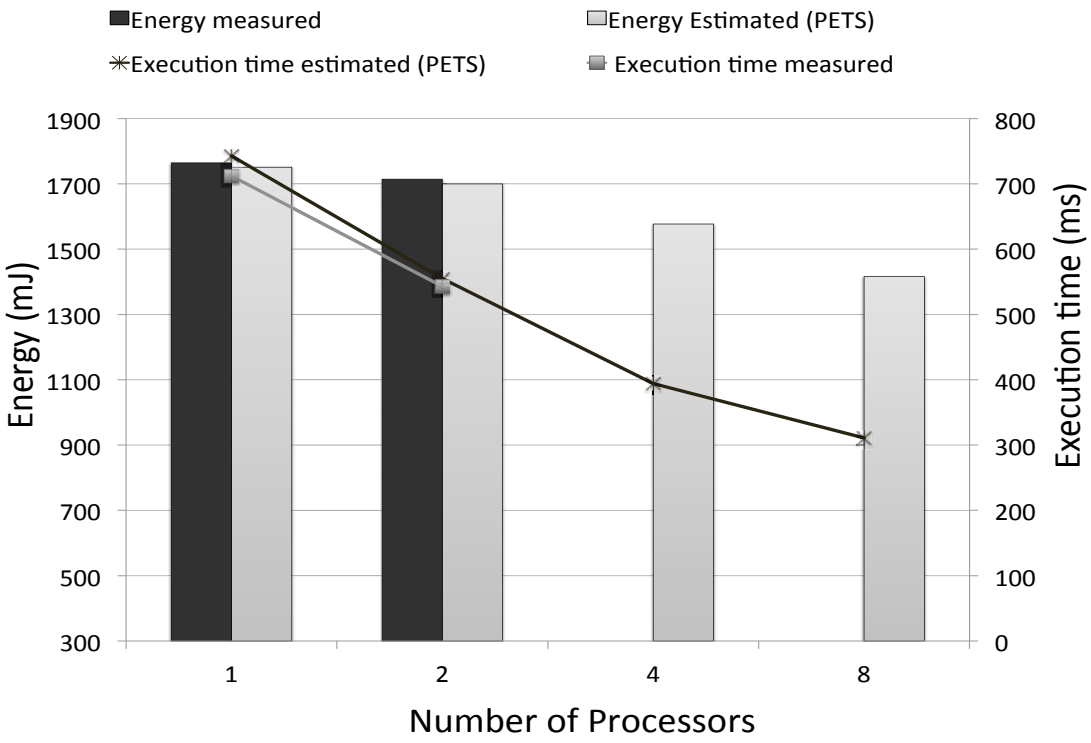

Figure 5.13: Execution time and energy variation according to the number of processors

a higher number of processors in the context of exploring new complex MPSoC.

At present, the exploration phase in the design flow of an embedded system focuses more on multi-objective optimisation problems, which tries to identify a solution with the optimal function cost involving criteria such as time, area, and power. In order to find the best implementation solution, a set of experiments have to be considered and evaluated. For the above mentioned reason, we tried to estimate the energy for different types of multimedia applications with 1 to 8 processors PowerPC based-architecture. We used parallel algorithm to run the multimedia benchmarks on the different processors. Fig. 5.15 gives the detailed energy estimation results and the corresponding execution time for different configurations. From this figure, we are able to extract several conclusions. First, depending on the execution time, we are able to find the optimal architecture that could satisfy the application requirement in term of computation rate. For instance, depending on the frame rate to achieve such as 15 or 30 frames per

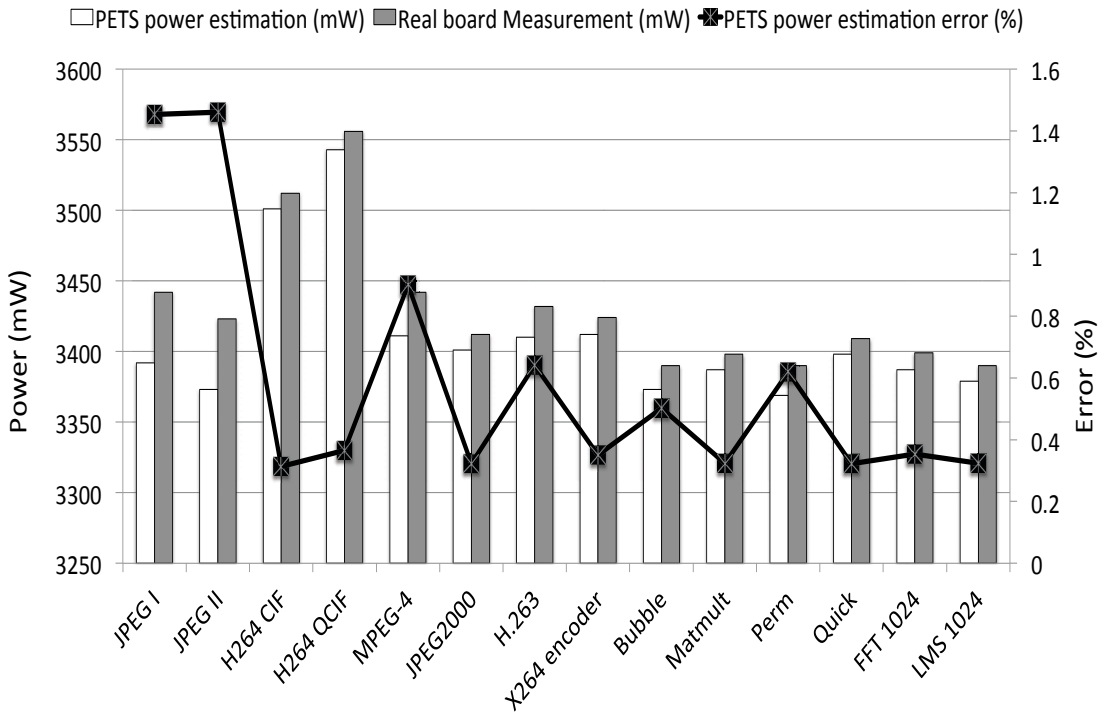

Figure 5.14: Power estimation of homogeneous two PowerPC multiprocessor architecture

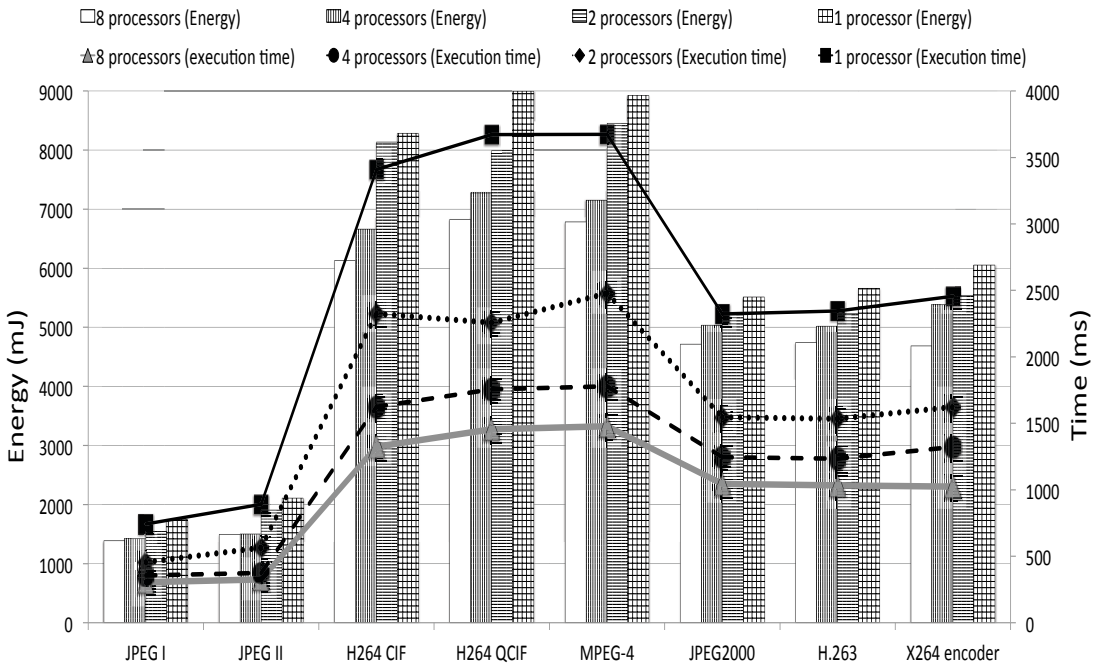

Figure 5.15: Energy estimation according to the number of processors using PowerPC

second (f/s), the H.264 decoder could be executed more or less efficiently. From the simulation results, 4 and 8 processors based-architectures offer respectively a total performance of 15 f/s and 19 f/s. We conclude that using 4 processors is sufficient to reach a rate of 15 f/s, however the 8 processors based-architecture offers a more energy-efficient solution. Indeed, with the price of area cost, the total energy is decreased by 9% while moving from 4 to 8 processors. Nevertheless, this behavior is not the same for all the applications. As an example for the JPEG application, using 8 processors instead of 4 leads to a reduced the total energy by a factor of 2.2%. Consequently, the corresponding additional area cost cannot be justified.

5.4.3 Heterogeneous multiprocessor based platform

In this part, we emphasize the benefit of our estimation methodology in the 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 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, 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. We selected a configuration, which is about 200 times faster than a software execution with a PowerPC processor running at 100 MHz. The synthesized hardware occupies 18% of the XupV2Pro. According to the FPGA power model, the power consumption of the chosen hardware DCT is around 300mW offering 40% of power saving compared to the software execution and 25% of reduction in execution time.

5.4.4 Estimation speed comparison

In this section, first, we will present our simulation speed results of hybrid power estimation approach and other different approaches available at the system-level. Second, we go ahead in comparing our proposed PETS tool with other power estimation tools available in the industries and academics.

5.4.4.1 Estimation speed comparison of different approaches

We will compare the efficiency of the proposed hybrid power estimation methodology approach in term of simulation speed with other approaches OVP standalone, ISS+TLM2.0 and Cycle-Accurate (CA). This comparison is for the quantification of our proposed tool to the state-of-art power approaches used in current practices. All the approaches re executed on a PC (Intel, 1.8 GHz, 2 Go of RAM). In order to compare the results, computer benchmarking has been done. Power estimation has been carried out with a JPEG application.

Table 5.1: JPEG application power estimation speed

	OVP	Hybrid power estimation methodology	ISS+TLM2.0	CA
Single processor	1.22 sec	1.22 sec	190.34 sec	891 sec
multiprocessor	0.8 sec	0.89 sec	160.54 sec	821 sec

From the table 5.1 it is clear that OVP provides a simulation speed improvement. Also, when switching from pure OVP environment to the OVP-TLM2.0 environment, there is no drop in simulation speed. For multiprocessor, the application is programmed in a way with strict data dependency among processor cores in which both the processors continuously polls the memory to get the pixel block. A tight synchronization is maintained among them. Table 5.1 shows that the speed improvements are not as high as for standalone single core systems when compared to the standalone OVP processor. Also when working with OVP models at TLM2.0 environment, a slight drop in simulation performance is observed for multiprocessor platforms. This drop generally falls in the range of 3-4%. Reduced speed improvements are because of the synchronization needed between the two processors.

5.4.4.2 Estimation speed comparison of different tools

Now, we will compare the efficiency of the proposed PETS tool in term of estimation speed with the state-of-the art tools such as, SimplePower, TLM with ISS based simulation and SoftExplorer (functional level simulator) approaches. This comparison is for the quantification of our proposed tool to the state-of-art power estimation tools used in current industrial and academic practices. SoftExplorer, TLM with ISS based simulation (HSL), and our proposed tool are executed on a PC (Intel, 1.8 GHz, 2 Go of RAM), whereas SimplePower on a Workstation (Ultra Sparc T2+, 1.6 GHz, 2 Go of RAM). In order to compare the results, computer benchmarking has been done to confirm that the workstation is always faster compared to the PC for all kind of applications. Power estimation has been carried out with a set of image & signal processing applications and also with SPEC 2008 [1] benchmarks.

[1] http://http://www.spec.org/benchmarks.htmlpower/

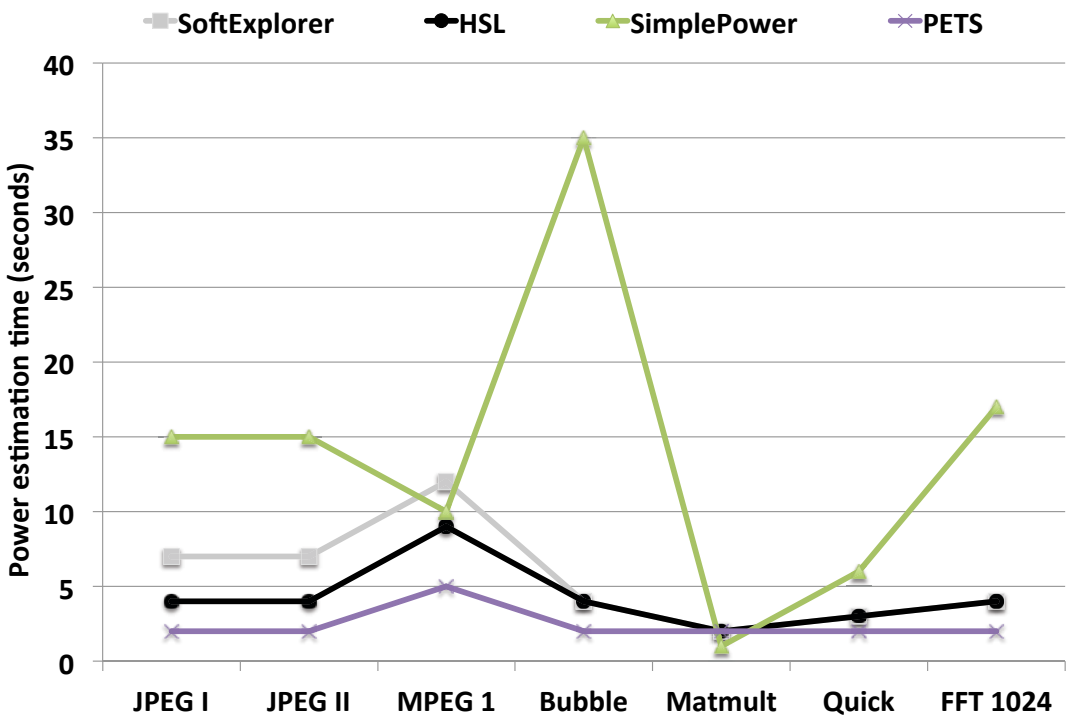

Figure 5.16: Comparison of estimation time for the different tools

From Fig. 5.16, we can notice that SoftExplorer and TLM with ISS based simulation have an average estimation time of 5 seconds, which is faster when compared to the SimplePower's average estimation time of 20 seconds. Our proposed tool has an average estimation time of 2.45 seconds, which is faster compared to the other tools. Our tool works by running the application on the virtual platform thereby collecting the dynamic activities. SimplePower uses cycle accurate specifications to collect the necessary power data, whereas SoftExplorer realizes a static profiling of the code, which results in reduced execution time and thus resulting in a low power consumption estimation time. Static profiling of the C code is not sufficient to determine the average execution time and the global energy consumption, for this reason we need to run the application on the virtual platform in order to collect the activities accurately and efficiently. Experimental results prove that our proposed tool is efficient, fast and accurate.

5.5 Conclusion

A hybrid power estimation methodology has been presented for rapid and accurate estimation of power in MPSoCs. The proposed approach consists of two steps: first, a one-time activity performed for power modeling, aimed at identifying and characterizing the main parameters affecting the power consumption; second, the development of a estimation simulation environment at system-level, performed to account for the occurrence of such parameter activities for a number of processors hosting different types of applications. As a result, a high estimation accuracy is achieved, which is within 5.7% of the real board measurement. This chapter has also presented our proposed PETS tool based on hybrid power estimation methodology. This study was conducted with different RISC processors, homogeneous and heterogeneous MPSoC running different benchmark applications. It was an opportunity to show the use of proposals described in chapter 3 and chapter 4. This chapter also shows that with this tool it is easy to explore the space of architectural solutions based on power and timing. Finally, we compared the simulation results of the PETS tool to different the state-of-the-art tools. Significant acceleration factors were measured and accurate comparison of the power has been validated against the real board measurements with an average error of less than 6%.

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

This chapter summarizes the main contributions of the thesis that have been discussed in the previous chapters and proposes extensions to the present PETS as part of the future works.

6.1 Summary

Power estimation at system-level has become primary concern for an efficient hardware design. Increasing adoption of system-level tools in design flow is necessitating the proposal of power estimation methodologies at higher levels of abstraction. This thesis developed a system-level tool (PETS tool) cum framework for power and energy estimation, which is based on hybrid power estimation methodology. In most of our approaches, we put an effort in providing a rationale behind the idea and then backed it with experimental results. The need of having efficient system-level power estimation tools is justified as the power becoming a critical pre-design metric in complex embedded systems and the increasing broadness of the design space at the system-level, which makes it very difficult to take the best architectural decisions. Decisions taken at the system-level, early in the design phase, are however essential in affecting the quality of the final design.

Taking such decisions manually, based on the experience matured from previous designs and on rules of thumb, as it has been done so far, is not a feasible approach any more. Efficient system-level power estimation tools are therefore necessary to automate the activities performed at system-level. Specifically, such tools can be used to perform power and timing based design-space exploration. The objective of this thesis was to design an efficient system-level power estimation tools for MPSoC based platforms for that we have extracted the three main challenges related to this thesis and they are:

1. what is the power modeling methodology suitable for MPSoC system-level design that can offer a better trade-off between the time needed to generate the power model and its corresponding accuracy?
2. what are the appropriate simulation technique and the abstraction level suitable for rapid MPSoC prototyping and for extracting accurately the activities for the defined power model (the first challenge)?
3. How to provide a tool at system-level in order to guide the designer during the different design choices based on power estimation?

To answer the above challenges, we proposed a power estimation tool (PETS) which rely on system-level simulation of the target to carry out the power estimation. Its estimation activity is based on the following two steps. First, a power model generation step which create the power models based on FLPA. These are not executable models, since they come in the form of either look-up tables or analytical expressions. Although the power model development is time-consuming, since it is generated on a real board measurements for each parameter. This step still makes sense since it is a one-time activity performed by the user and it gives high accuracy to the whole estimation tool. A case study has been presented in Chapter 3, where different power models have been developed for different RISC processors. The accuracy of the model has then been validated against direct measurements from the real board and has shown to be within 5%. Chapter 4 presented different case studies about system-level environment used in this framework. The accuracy of the system-level models has been measured to be within 2% of the TLM model estimation. A combination of power modeling methodology and system-level framework gives a hybrid power estimation

methodology which works as a standalone irrespective of the platforms under test as described in the Chapter 5.

Different case studies have been presented in Chapter 5, where PETS has been used to estimate power and execution time for a set of benchmark applications. PETS estimation accuracy has been validated against real board measurement and proved to be within 6%. PETS estimation speed has also been validated against ISS based TLM and various other state-of-the-art tools, showing an average speedup of 70X. It is emphasized that, because of the very large size of real use-cases and the large amount of factors to be considered when doing system-level power estimation, it was not possible to deal with all these aspects within this work. Instead, smaller use-cases have been used as case studies and sensible simplifications have been made. As a consequence, the PETS tool presented in this thesis comes as a proof of concept, with the goal of showing the overall feasibility of the PETS approach as a system level power estimation tool. Further extensions are required to make this tool more general and complete, as is discussed in the next section.

6.2 Future works

Despite the detailed presentation that was given in this thesis work, further development is needed for PETS to be used as a complete, stand-alone tool. Among the aspects that can be improved, the following ones are considered to be the most relevant:

- Power estimation tool presented in this thesis can be extended in many ways. One of the possible extensions is to provide model for leakage power at system-level. Regression model presented in this thesis considers constant leakage power. However, for lower technology nodes this relationship is not constant, thus an effort is required to improve the developed model. Second possibility is to extend the regression model for multiple IPs, that is one model to measure power consumption of multiple IPs. Such a model will make the estimation task very easy and requires minimal changes for new architectures and technology nodes.

- One of the possible extension of this work can be done for battery life predication at system-level. Predicting the residual energy of the battery source that powers a portable electronic device is imperative in designing at system-level and applying an effective dynamic power management policy for the device. This work can be done by closed-form analytical expression for predicting the remaining capacity of a lithium-ion battery. Then proposing a high level model, which relies on power, correctly accounts for the temperature and cycle ageing effects. Then integrating the high level model into the system-level framework. Power optimization will become another key motivation adopting system-level for this work as 32 nm technology ushers in dramatic increases in power density, affecting battery life as well as thermal and supply integrity.
- Another important extension would be to consider thermal models at system-level, by using infrared measurement setup to capture run-time power consumption and thermal characteristics of modern chips. It can be done by using infrared cameras with high spatial resolution (10x10m) and high frame rate (125fps) to capture thermal maps and then to generate a detailed power breakdown (leakage and dynamic) for each processor floorplan. Then proposing a genetic algorithm to find a power equation for each floorplan block that produces the measured temperature for a given thermal package. The difference between the predicted power and the externally measured power consumption for the processor and enabling this at the system-level.
- Next future scope would to provide system-level modeling for reliability and device degradation by constructing a hierarchical power modeling tree and augment the transaction level models with power estimation functions. In the system-level reliability modeling, we can able to propose a transaction-based error susceptibility model for a bus-based System-on-Chip system. This reliability model provides a detailed analysis of different kinds of errors and the susceptibility of such systems to such errors on various components that comprise the bus. By injecting single and multi-bit error during the execution of various transactions and examine the effect of the errors. At the system-level modeling for device degradation can be explored any Negative Bias Temperature Instability (NBTI) and Hot Carrier Effects (HCE) that

cause device degradation in the system. There are tools such as HCE and NBTI Incorporated Tool for ASICs (HANITA), for the complete analysis of circuit degradation. These tools analyze the degradation impact on bus systems and the vulnerability of buses to such circuit degradation. By using this information, we can propose a hardware-based mechanism to detect the timing degradation and we can further propose a PROactive BUS (PROBUS) architecture that dynamically adapts to retain the system functionality even after the system timing degrades.

REFERENCES

[1] Carbon design systems. http://carbondesignsystems.com. 42, 43

[2] Cofluent design. http://www.cofluentdesign.com. 44

[3] Comet, meteor. http://www.synopsys.com/Systems/VirtualPrototyping/VPModels/Pages/CoMET-METeor.aspx. 43

[4] The Gaut Website. http://www-labsticc.univ-ubs.fr/www-gaut/. 71

[5] Imperas inc. http://www.ovpworld.org/. 43

[6] Innovator. https://www.synopsys.com/ARCHIVE/VIRTUALPLATFORMS/Pages/Innovator.aspx. 42

[7] The mathworks, eda simulator link, simulink. http://www.mathworks.com. 43

[8] Mentor graphics. http://www.mentor.com. 43

[9] Synopsys. www.synopsys.com. 18

[10] Synopsys, innovator, platform architect, comet, meteor. http://www.synopsys,com. 42

[11] Network calculus. In Jean-Yves Boudec and Patrick Thiran, editors, *Network Calculus, volume-2050*, Lecture Notes in Computer Science, pages 3–81. Springer Berlin Heidelberg, 2001. 40

[12] Ctos. http://www.cadence.com/Community/tags/CTOS/default.aspx, jun 2005. 21

[13] Cadence design systems. www.cadence.com, jun 2010. 18

[14] Platform architect. http://www.synopsys.com/Systems/ArchitectureDesign/Pages/PlatformArchitect.aspx, 2011. 42

[15] Vikram S. Adve and Mary K. Vernon. Parallel program performance prediction using deterministic task graph analysis. *ACM Trans. Comput. Syst.*, **22**[1]:94–136, February 2004. 40

[16] ARM Cortex-A8. Cortex-a8 core runs faster, sips power. http://www.linuxfordevices.com/c/a/News/Samsung-CortexA8/, 2009. xiv, 52

[17] ARM9 TDMI. Arm940t technical reference manual. http://infocenter.arm.com/, 2012. xiv, 53

[18] Todd Austin, Eric Larson, and Dan Ernst. Simplescalar: An infrastructure for computer system modeling. *Computer*, **35**[2]:59–67, February 2002. 38

[19] H. Schmit B. Klass, D. E. Thomas and D. F. Nagle. Modeling inter-instruction energy effects in a digital signal processor. *in Power Driven Microarchitecture Workshop in conjunction with International Symposism Computer Architecture*, **1**, June 1998. 29

[20] M. Balakrishnan. Low power design. *http://embedded.cse.iitd.ernet.in/homepage/course/low power/index.shtml.*, 2008. 30

[21] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli. Metropolis: an integrated electronic system design environment. *Computer*, **36**[4]:45 – 52, april 2003. 33

[22] Felice Balarin, Yosinori Watanabe, Harry Hsieh, Luciano Lavagno, Claudio Passerone, and Alberto Sangiovanni-Vincentelli. Metropolis: An integrated electronic system design environment. *Computer*, **36**[4]:45–52, April 2003. xiv, 34, 36

[23] Nikhil Bansal, Kanishka Lahiri, Anand Raghunathan, and Srimat T. Chakradhar. Power monitors: A framework for system-level power estimation using heterogeneous power models. In *Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*, VLSID '05, pages 579–585, Washington, DC, USA, 2005. IEEE Computer Society. 39

[24] Reinaldo A. Bergamaschi, Youngsoo Shin, Nagu Dhanwada, Subhrajit Bhattacharya, William E. Dougherty, Indira Nair, John Darringer, and Sarala Paliwal. Seas: a system for early analysis of socs. In *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, CODES+ISSS '03, pages 150–155, New York, NY, USA, 2003. ACM. 28

[25] H. Blume, D. Becker, L. Rotenberg, M. Botteck, J. Brakensiek, and T. G. Noll. Hybrid functional- and instruction-level power modeling for embedded and heterogeneous processor architectures. *J. Syst. Archit.*, **53**[10]:689–702, October 2007. 45

[26] H. Blume, M. Schneider, and T. G. Noll. Power estimation on functional level for programmable processors. In *Advances in Radio Science*, 2004. 45

[27] A. Bogliolo, L. Benini, and G. De Micheli. Adaptive least mean square behavioral power modeling. In *Proceedings of the 1997 European conference on Design and Test*, EDTC '97, pages 404–, Washington, DC, USA, 1997. IEEE Computer Society. 28

[28] Alessandro Bogliolo, Luca Benini, and Giovanni De Micheli. Regression-based rtl power modeling. *ACM Trans. Des. Autom. Electron. Syst.*, **5**[3]:337–372, July 2000. 25

[29] Alessandro Bogliolo, Luca Benini, and Giovanni De Micheli. Characterization-free behavioral power modeling. In *In Proceedings of the Design Automation and Test in Europe*, pages 767–773, 1998. 28

[30] S. Boukhechem, E.-B. Bourennane, and H. Samahi. Co-simulation platform based on systemc for multiprocessor system on chip architecture exploration. In *Microelectronics, 2007. ICM 2007. Internatonal Conference on*, pages 105 –110, dec. 2007. 33

[31] C. Brandolese̊c. *A Codesign Approach to Software Power Estimation for Embedded Systems*. PhD thesis, Politecnico di Milano, Institute of Electronics and Information, 2000. 23, 24

[32] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. *SIGARCH Comput. Archit. News*, **28**[2]:83–94, May 2000. 27

[33] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. *SIGARCH Comput. Archit. News*, **28**[2]:83–94, May 2000. 44

[34] D. Burger and T. M. Austin. The simplescalar tool set, version 2.0. *SIGARCH Computer Architecture News*, **25**:13–25, 1997. 27

[35] M. Casas-Sanchez C. J. Bleakley and J. Rizo-Morente. Software level power consumption models and power saving techniques for embedded dsp processors. *Journal of Low Power Electronics*, **2**:281290, 2006. 22

[36] M. Caldari, M. Conti, M. Coppola, P. Crippa, S. Orcioni, L. Pieralisi, and C. Turchetti. System-level power analysis methodology applied to the amba ahb bus. In *Proceedings of the conference on Design, Automation and Test in Europe: Designers' Forum - Volume 2*, DATE '03, pages 20032–, Washington, DC, USA, 2003. IEEE Computer Society. 39

[37] Samarjit Chakraborty, Simon Kunzli, and Lothar Thiele. A general framework for analysing system properties in platform-based em-

bedded system designs. In *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1*, DATE '03, pages 10190–, Washington, DC, USA, 2003. IEEE Computer Society. 40

[38] Y. Chen. *The Analysis and Practice on Open Source Embedded System Software–Based on SkyEye and ARM Developing Platform.* Beihang University Press, 2004. 26

[39] T. Chou and K. Roy. Accurate estimation of power dissipation in cmos sequential circuits. *IEEE Transaction VLSI Systems*, **4**:369380, 1996. 24

[40] Saadia Dhouib, Eric Senn, Jean-Philippe Diguet, Dominique Blouin, and Johann Laurent. Energy and power consumption estimation for embedded applications and operating systems. *J. Low Power Electronics*, **5**[4]:416–428, 2009. 31

[41] David Ellouet, Yannig Savary, and Nathalie Julien. An fpga power aware design flow. In *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, **4148** of *Lecture Notes in Computer Science*, pages 415–424. Springer Berlin Heidelberg, 2006. xv, 56

[42] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, **88**[2]:303–338, June 2010. 17

[43] Jerry Frenkil. Tools and methodologies for low power design. In *Proceedings of the 34th annual Design Automation Conference*, DAC '97, pages 76–81, New York, NY, USA, 1997. ACM. 44

[44] Daniel Gajski and Robert H. Kuhn. New vlsi tools - guest editors introduction. *IEEE Computer*, **16**[12]:11–14, 1983. xiv, 12

[45] A. Gerstlauer, C. Haubelt, A.D. Pimentel, T.P. Stefanov, D.D. Gajski, and J. Teich. Electronic system-level synthesis methodologies. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, **28**[10]:1517 –1530, oct. 2009. 45

[46] Frank Ghenassia. Transaction-level modeling with systemc: Tlm concepts and applications for embedded systems. *Springer*, 2005. xiv, 16, 19, 20, 21

[47] Matthias Gries, Chidamber Kulkarni, Christian Sauer, and Kurt Keutzer. Exploring trade-offs in performance and programmability of processing element topologies for network processors. In *in Proc. of Network Processor Workshop in conjunction with Ninth International Symposium on High Performance Computer Architecture (HPCA-9*, pages 75–87. Morgan Kaufmann, 2003. 40

[48] Thorsten Grotker. System design with systemc. *Kluwer Academic Publishers*, **Norwell, MA, USA**. 18

[49] Rajesh Kumar Gupta. *Co-synthesis of hardware and software for digital embedded systems*. PhD thesis, Stanford, CA, USA, 1994. UMI Order No. GAX94-14572. 17

[50] Subodh Gupta and Farid N. Najm. Power macromodeling for high level power estimation. In *Proceedings of the 34th annual Design Automation Conference*, DAC '97, pages 365–370, New York, NY, USA, 1997. ACM. 23

[51] Subodh Gupta and Farid N. Najm. Power macromodeling for high level power estimation. In *Proceedings of the 34th annual Design Automation Conference*, DAC '97, pages 365–370, New York, NY, USA, 1997. ACM. 24

[52] Subodh Gupta and Farid N. Najm. Energy and peak-current per-cycle estimation at rtl. *IEEE Trans. Very Large Scale Integr. Syst.*, **11**[4]:525–537, August 2003. 28

[53] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mary Jane Irwin, N. Vijaykrishnan, Mahmut Kandemir, Tao Li, and Lizy Kurian John. Using complete machine simulation for software power estimation: The softwatt approach. In *Proceedings of the 8th International*

Symposium on High-Performance Computer Architecture, HPCA '02, pages 141–, Washington, DC, USA, 2002. IEEE Computer Society. 26

[54] Charlie X. Huang, Bill Zhang, An-Chang Deng, and Burkhard Swirski. The design and implementation of powermill. In *Proceedings of the 1995 international symposium on Low power design*, ISLPED '95, pages 105–110, New York, NY, USA, 1995. ACM. 23

[55] IBM. Ppc405 product overview. https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/3D7489A3704570C0872571DD0065934E/, 2006. xiv, 54

[56] ITRS. Design, 2010 edition. http://public.itrs.net/, 2010. 2

[57] N. Julien J. Laurent, E. Senn and E. Martin. High level energy estimation for dsp systems. *in proceedings International Workshop on Power And Timing Modeling and Optimization and Simulation PATMOS01*, pages 311–316, September 2001. 30

[58] C. P. Joshi, Anshul Kumar, and M. Balakrishnan. A new performance evaluation approach for system level design space exploration. In *Proceedings of the 15th international symposium on System Synthesis*, ISSS '02, pages 180–185, New York, NY, USA, 2002. ACM. 38

[59] Nathalie Julien, Johann Laurent, Eric Senn, and Eric Martin. Power consumption modeling and characterization of the ti c6201. *IEEE Micro*, **23**[5]:40–49, September 2003. xiv, 31

[60] I. Kadayif, T. Chinoda, M. Kandemir, N. Vijaykirsnan, M. J. Irwin, and A. Sivasubramaniam. vec: virtual energy counters. In *Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering*, PASTE '01, pages 28–31, New York, NY, USA, 2001. ACM. 27

[61] Torsten Kempf, Malte Doerper, R. Leupers, G. Ascheid, H. Meyr, Tim Kogel, and Bart Vanthournout. A modular simulation framework for spatial and temporal task mapping onto multi-processor

soc platforms. In *Proceedings of the conference on Design, Automation and Test in Europe - Volume 2*, DATE '05, pages 876–881, Washington, DC, USA, 2005. IEEE Computer Society. 33

[62] Jari Kreku, Mika Hoppari, Tuomo Kestil, Yang Qu, Juha-Pekka Soininen, and Kari Tiensyrj. Application workload and systemc platform modeling for performance evaluation. In Martin Radetzki, editor, *Languages for Embedded Systems and their Applications*, **36** of *Lecture Notes in Electrical Engineering*, pages 131–147. Springer Netherlands, 2009. 38

[63] Thomas H. Krodel. Powerplay-fast dynamic power estimation based on logic simulation. In *Proceedings of the 1991 IEEE International Conference on Computer Design on VLSI in Computer & Processors*, ICCD '91, pages 96–100, Washington, DC, USA, 1991. IEEE Computer Society. 41

[64] M. Favalli L. Benini, A. Bogliolo and G. De Micheli. Regression models for behavioral power estimation. *Integrated Computer-Aided Enggineering*, **5**:95106, 1998. 25

[65] D. Gajski L. Cai and M. Olivarez. Introduction of system level architecture exploration using the specc methodology. In *In Circuits and Systems*, The 2001 IEEE International Symposium on, volume 5, pages 9, May 25-27 2001. 17

[66] Marcello Lajolo, Anand Raghunathan, and Sujit Dey. Efficient power co-estimation techniques for system-on-chip design. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '00, pages 27–34, New York, NY, USA, 2000. ACM. 40

[67] Paul E. Landman and Jan M. Rabaey. Activity-sensitive architectural power analysis for the control path. In *Proceedings of the 1995 international symposium on Low power design*, ISLPED '95, pages 93–98, New York, NY, USA, 1995. ACM. 25

[68] J. Laurent, N. Julien, and E. Martin. Functional level power analysis: An efficient approach for modeling the power consumption of complex

processors. In *Proceedings of the Design, Automation and Test in Europe Conference*, Munich, 2004. 31, 41

[69] 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. xv, 55, 56

[70] 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 *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, **1**, pages 666 – 667 Vol.1, feb. 2004. 41

[71] Paul Lieverse, Todor Stefanov, Pieter van der Wolf, and Ed Deprettere. System level design with spade: an m-jpeg case study. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, ICCAD '01, pages 31–38, Piscataway, NJ, USA, 2001. IEEE Press. 33

[72] D. Liu and C. Svensson. Power consumption estimation in cmos vlsis chips. *IEEE Journal of Solid-State Circuits*, **29**:663670, 1994. 25

[73] C. Silvano M. Sami, D. Sciuto and V. Zaccaria. An instruction-level energy model for embedded vliw architectures. *IEEE Transaction on CAD of Integrated Circuits and Systems*, **21**:998–1010, 2002. 30

[74] H. Blume M. Schneider and T. G. Noll. Power estimation on functional level for programmable processors. *in journal of Advances in Radio Science*, **2**:215–219, 2005. 32

[75] Peter S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hållberg, Johan Högberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner. Simics: A full system simulation platform. *Computer*, **35**[2]:50–58, February 2002. 38

[76] Laurent Maillet-Contoz and Frank Ghenassia. Transaction level modeling. In Frank Ghenassia, editor, *Transaction Level Modeling with SystemC*, pages 23–55. Springer US, 2005. 10.1007/0-387-26233-4_2. 19

[77] Diana Marculescu, Radu Marculescu, and Massoud Pedram. Information theoretic measures of energy consumption at register transfer level. In *Proceedings of the 1995 international symposium on Low power design*, ISLPED '95, pages 81–86, New York, NY, USA, 1995. ACM. 24

[78] H. Mehta, R. M. Owens, and M. J. Irwin. Instruction level power profiling. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1996. on Conference Proceedings., 1996 IEEE International Conference - Volume 06*, ICASSP '96, pages 3326–3329, Washington, DC, USA, 1996. IEEE Computer Society. 25, 29

[79] Huzefa Mehta, Robert Michael Owens, and Mary Jane Irwin. Energy characterization based on clustering. In *Proceedings of the 33rd annual Design Automation Conference*, DAC '96, pages 702–707, New York, NY, USA, 1996. ACM. 28

[80] Brett H. Meyer, Joshua J. Pieper, JoAnn M. Paul, Jeffrey E. Nelson, Sean M. Pieper, and Anthony G. Rowe. Power-performance simulation and design strategies for single-chip heterogeneous multiprocessors. *IEEE Trans. Comput.*, **54**[6]:684–697, June 2005. 37

[81] S. Mohanty, V. K. Prasanna, S. Neema, and J. Davis. Rapid design space exploration of heterogeneous embedded systems using symbolic search and multi-granular simulation. *SIGPLAN Not.*, **37**[7]:18–27, June 2002. 33

[82] S. Mohanty and V.K. Prasanna. Rapid system-level performance evaluation and optimization for application mapping onto soc architectures. In *ASIC/SOC Conference, 2002. 15th Annual IEEE International*, pages 160 – 167, sept. 2002. xiv, 36, 37

[83] Sumit Mohanty and Viktor K. Prasanna. A hierarchical approach for energy efficient application design using heterogeneous embedded systems. In

Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems, CASES '03, pages 243–254, New York, NY, USA, 2003. ACM. 35

[84] L. Rader N. Kumar, S. Katkoori and R. Vemuri. Profile-driven behavioral synthesis for low-power vlsi systems. *IEEE Design and Test*, **12**:7084, 1995. 25

[85] F. N. Najm. A survey of power estimation techniques in vlsi circuits. *IEEE Transactions on VLSI Systems*, **2**:446455, 1994. 23

[86] Farid N. Najm. Towards a high-level power estimation capability. In *Proceedings of the 1995 international symposium on Low power design*, ISLPED '95, pages 87–92, New York, NY, USA, 1995. ACM. 28

[87] Luca Negri and Andrea Chiarini. Power simulation of communication protocols with statec. In A. Vachoux, editor, *Applications of Specification and Design Languages for SoCs*, pages 277–294. Springer Netherlands, 2006. 38

[88] University of Berkeley. The University of Berkeley website: Spice manual. http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/. 22

[89] Open Virtual Platform. Ovpsim, 2012. World Wide Web document, URL: http://www.ovpworld.org/technology_ovpsim.php. xvi, 103

[90] Bassem Ouni, Cecile Belleudy, and Eric Senn. Energy characterization and classification of embedded operating system services. In *Digital System Design (DSD), 2012 15th Euromicro Conference on*, pages 684 –691, sept. 2012. 32

[91] Preeti Ranjan Panda. Systemc: a modeling platform supporting multiple design abstractions. In *Proceedings of the 14th international symposium on Systems synthesis*, ISSS '01, pages 75–80, New York, NY, USA, 2001. ACM. 17, 18

[92] Joann M. Paul, Donald E. Thomas, and Andrew S. Cassidy. High-level modeling and simulation of single-chip programmable heterogeneous multiprocessors. *ACM Trans. Des. Autom. Electron. Syst.*, **10**[3]:431–461, July 2005. 33, 37

[93] Kees Vissers Paul Lieverse, Pieter Van Der Wolf and Ed Deprettere. A methodology for architecture exploration of heterogeneous signal processing systems. *J. VLSI Signal Process. System*, **5**[3]:103–108, 2001. xiv, 34

[94] M. Pedram. Power aware design methodologies. *J. M. Rabaey, Ed. Norwel*, **MA, USA: Kluwer Academic Publishers**, 2002. 27

[95] R. Peset-Lopis and K. Goossens. The petrol approach to high-level power estimation. In *Proc. of the ISLPED*, Monterey, California, USA, August 1998. 22

[96] Philips Research. *Philips Electronic Design and Tools Group. DIESEL User Manual*, 2001. 22

[97] A. D. Pimentel, S. Polstra, F. Terpstra, A. W. van Halderen, J. E. Coffland, and L. O. Hertzberger. Embedded processor design challenges. chapter Towards efficient design space exploration of heterogeneous embedded media systems, pages 57–73. Springer-Verlag New York, Inc., New York, NY, USA, 2002. 33

[98] Andy D. Pimentel, Louis O. Hertzberger, Paul Lieverse, Pieter van der Wolf, and Ed F. Deprettere. Exploring embedded-systems architectures with artemis. *Computer*, **34**[11]:57–63, November 2001. 33

[99] Nachiketh R. Potlapally, Anand Raghunathan, Ganesh Lakshminarayana, Michael S. Hsiao, and Srimat T. Chakradhar. Accurate power macro-modeling techniques for complex rtl circuits. In *in Proc. Int. Conf. VLSI Design*, pages 235–241, 2001. 25, 26

[100] S. Powell and E. M. Chau. Estimating power dissipation of vlsi signal processing chips: the pfa technique. *in VLSI Signal Processing IV*, pages 250–259, 1990. 24

[101] M. Pedram Q. Wu, Q. Qiu and C.-S. Ding. Cycle-accurate macro-models for rtlevel power analysis. *IEEE Transaction VLSI Systems*, **6**:520–528, 1998. 25

[102] Gang Qu, Naoyuki Kawabe, Kimiyoshi Usami, and Miodrag Potkonjak. Function-level power estimation methodology for microprocessors. In *Proceedings of the 37th Annual Design Automation Conference*, DAC '00, pages 810–813, New York, NY, USA, 2000. ACM. 45

[103] D. Brooks R. Joseph and M. Martonosi. Runtime power measurements as a foundation for evaluating power/performance tradeoffs. *in proceedings of the Workshop on Complexity Effectice Design WCED, held in conjunction with ISCA01*, pages 13–25, 2001. 27

[104] J. N. Rabaey and M. Pedram. Low power design methodologies. *The Springer International Series in Engineering and Computer Science*, **4**:366, 1996. 24

[105] Santhosh Kumar Rethinagiri, Rabie Ben Atitallah, Jean-Luc Dekeyser, Eric Senn, and Smail Niar. An efficient power estimation methodology for complex risc processor-based platforms. In *Proceedings of the great lakes symposium on VLSI*, GLSVLSI '12, pages 239–244, New York, NY, USA, 2012. ACM. 41, 42

[106] S.K. Rethinagiri, R.B. Atitallah, and J. Dekeyser. A system level power consumption estimation for mpsoc. In *System on Chip (SoC), 2011 International Symposium on*, pages 56 –61, 31 2011-nov. 2 2011. 41

[107] S.K. Rethinagiri, R. Ben Atitallah, S. Niar, E. Senn, and J. Dekeyser. Fast and accurate hybrid power estimation methodology for embedded systems. In *Design and Architectures for Signal and Image Processing (DASIP), 2011 Conference on*, pages 1 –7, nov. 2011. 41

[108] S.K. Rethinagiri, R. Ben Atitallah, S. Niar, E. Senn, and J.-C. Dekeyser. Hybrid system level power consumption estimation for fpga-based mpsoc. In *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pages 239 –246, oct. 2011. 41, 42, 122

[109] Mendel Rosenblum, Edouard Bugnion, Scott Devine, and Stephen A. Herrod. Using the simos machine simulator to study complex

computer systems. *ACM Trans. Model. Comput. Simul.*, **7**[1]:78–103, January 1997. 38

[110] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004. 41

[111] J.T. Russell and M.F. Jacome. Software power estimation and optimization for high performance, 32-bit embedded processors. In *Computer Design: VLSI in Computers and Processors, 1998. ICCD '98. Proceedings. International Conference on*, pages 328 –333, oct 1998. 29

[112] J.P. Diguet S. Douhib. 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. 31, 32, 42

[113] S. Neema S. Mohanty, V. K. Prasanna and J. Davis. Rapid design space exploration of heterogeneous embedded systems using symbolic search and multi-granular simulation. *In Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems, LCTES/SCOPES 02*, **New York, NY, USA, 2002. ACM. ISBN 1-58113-527-0. URL http://doi.acm.org/10.1145/513829.513835**:1827, 2001. 35

[114] P. Neofotistos K. Kosmatopoulos T. Laopoulos S. Nikolaidis, N. Kavvadias and L. Bisdounis. Instrumentation set-up for instruction level power modeling. *in proceedings of the 12th International Workshop on Power and Timing Modeling, Optimization and Simulation PATMOS02*, **London, UK: Springer-Verlag**:71–80, 2002. 29

[115] T. Laopoulos L. Bisdounis S. Nikolaidis, N. Kavvadias and S. Blionas. Instruction level energy modeling for pipelined processors. *Journal of Embedded Computing*, **1**:317324, 2005. 29

[116] Akshaye Sama, J. F. M. Theeuwen, and M. Balakrishnan. Speeding up power estimation of embedded software. In *Proceedings of the 2000 international*

symposium on Low power electronics and design, ISLPED '00, pages 191–196, New York, NY, USA, 2000. ACM. 29

[117] Gunar Schirner and Rainer Dmer. Quantitative analysis of the speed/accuracy trade-off in transaction level modeling. *ACM Trans. Embed. Comput. Syst.*, **8(1. ISSN 1539-9087**:1–29, 2008. 18

[118] Eric Senn, Nathalie Julien, Johann Laurent, and Eric Martin. Power consumption estimation of a c program for data-intensive applications. In *Proceedings of the 12th International Workshop on Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation*, PATMOS '02, pages 332–341, London, UK, UK, 2002. Springer-Verlag. 31

[119] Eric Senn, Nathalie Julien, Johann Laurent, and Eric Martin. Power consumption estimation of a c program for data-intensive applications. In *Proceedings of the 12th International Workshop on Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation*, PATMOS '02, pages 332–341, London, UK, UK, 2002. Springer-Verlag. 45

[120] Hyunchul SHIN and Changhee LEE. Operation mode based high-level switching activity analysis for power estimation of digital circuits(energy in electronics communications). *IEICE transactions on communications*, **90**[7]:1826–1834, jul 2007. 28

[121] Chris Spear. *SystemVerilog for Verification, Second Edition: A Guide to Learning the Testbench Language Features*. Springer Publishing Company, Incorporated, 2nd edition, 2008. 17

[122] Lothar Thiele, Samarjit Chakraborty, Matthias Gries, and Simon Knzli. Design space exploration of network processor architectures. In *In Network Processor Design: Issues and Practices, Volume 1*, pages 30–41. Morgan Kaufmann Publishers, 2002. 40

[123] Lothar Thiele, Samarjit Chakraborty, Matthias Gries, Alexander Maxiaguine, Er Maxiaguine, and Jonas Greutert. Embedded software in network processors - models and algorithms. In *In First Workshop on Embedded Software, LNCS 2211*, pages 416–434. Springer Verlag, 2001. 40

[124] V. Tiwari, S. Malik, A. Wolfe, and M.T.-C. Lee. Instruction level power analysis and optimization of software. In *VLSI Design, 1996. Proceedings., Ninth International Conference on*, pages 326 –328, jan 1996. 44

[125] Elias Todorovich, Eduardo I. Boemo, F. Cardells, and Javier Valls. Power analysis and estimation tool integrated with xpower. In Russell Tessier and Herman Schmit, editors, *Proceedings of the ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, FPGA 2004, Monterey, California, USA, February 22-24, 2004*, page 259. ACM, 2004. 41

[126] S. Malik V. Tiwari and A. Wolfe. Power analysis of embedded software a first step towards software power minimization. *IEEE Transaction on VLSI Systems*, page 437445, 1994. 28

[127] M. Kandemir W. Ye, N. Vijaykrishnan and M. J. Irwin. The design and use of simplepower: A cycle-accurate energy estimation tool. *in proceedings of the 37th conference on Design automation DAC2000*, **New York, NY, USA: ACM**:340–345, 2000. 26

[128] Wayne Wolf. Household hints for embedded systems designers. *Computer*, **35**[5]:106–108, May 2002. 40

[129] W. Ye, N. Vijaykrishnam, M. Kandemir, and M.J. Irwin. The design and use of simplepower: a cycle accurate energy estimation tool. In *Proc. Design Automation Conference DAC'00*, June 2000. 41

[130] W. Ye, N. Vijaykrishnan, M. Kandemir, and M.J. Irwin. The design and use of simplepower: a cycle-accurate energy estimation tool. In *Design Automation Conference, 2000. Proceedings 2000*, pages 340 –345, 2000. 41

[131] Dohyung Kim Youngmin Yi and Soonhoi Ha. Fast and accurate cosimulation of mpsoc using trace-driven virtual synchronization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **26(12)**:21862200, 2007. 19

[132] Lin Zhong, S. Ravi, A. Raghunathan, and N. K. Jha. Power estimation for cycle-accurate functional descriptions of hardware. In *Proceedings of the*

2004 IEEE/ACM International conference on Computer-aided design, ICCAD '04, pages 668–675, Washington, DC, USA, 2004. IEEE Computer Society. 39