Transactional simulation approach for modelling performance and energy of a heterogeneous SoC memory system

Amal Ben Ameur

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Approche de simulation transactionnelle pour la modélisation des performances et de l'énergie d'un système mémoire pour SoC hétérogènes

Amal BEN AMEUR

Laboratoire d'Electronique, Antennes et Télécommunications
UCA, UMR CNRS 7248

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur
Dirigée par : François Verdier
Co-encadrée par : Michel Auguin
Soutenue le : 28/06/2019

Devant le jury, composé de :
Pascal Benoit, Maître de conférence, LIRMM
David Navarro, Maître de conférence, ECL
Arnaud Tisserand, Directeur de Recherche, CNRS
François Verdier, Professeur, Université Côte d'Azur
Michel Auguin, Directeur de Recherche, CNRS
Valerio Frascolla, Directeur de Recherche et Innovation, Intel Deutschland GmbH
Patricia Guitton-ouhamou, Architecte Logiciel, Renault Sophia Antipolis

intel

Approche de simulation transactionnelle pour la modélisation des performances et de l'énergie d'un système mémoire pour SoC hétérogènes

Jury :

Rapporteurs :

Pascal Benoit	Maitre de Conférence, LIRMM
David Navarro	Maitre de Conférence, ECL

Examinateurs :

Arnaud Tisserand	Directeur de Recherche, CNRS
Valerio Frascolla	Directeur de Recherche et Innovation, Intel Deutschland GmbH

Invitée :

Patricia Guitton-ouhamou	Architecte Logiciel, Renault Sophia Antipolis

Directeur de thèse :

François Verdier	Professeur, Université Côte d'Azur

Co-directeur de thèse :

Michel Auguin	Directeur de Recherche, CNRS

Résumé

Les appareils mobiles, à chaque nouvelle version des normes et suivant les demandes continues de nouveaux services par les utilisateurs, doivent prendre en charge de plus en plus de fonctionnalités, qui deviennent également de plus en plus exigeantes du point de vue informatique. Par conséquent, être en mesure de répondre aux nouvelles exigences tout en fournissant des puces à faible consommation d'énergie est aujourd'hui le défi le plus important pour les concepteurs de systèmes pour appareils mobiles.

Pour relever ce défi, de nouvelles approches de modélisation de la performance et de la puissance au niveau système ont été proposées, permettant d'explorer les architectures matérielles/ logicielles (HW / SW) dès les toutes premières étapes d'un flot de conception de systèmes sur puce (SoC). Cependant, les solutions existantes prennent en charge de manière limitée l'optimisation de la puissance du système de mémoire (y compris la mémoire SDRAM), qui peut occuper plus de 70% de la surface d'une puce et consommer plus de 30% de l'énergie totale. Dans nos travaux, nous proposons un cadre de simulation basé sur SystemC-TLM au niveau Electronic System Level (ESL), capable de prendre en charge l'exploration commune d'une architecture SoC et de sa configuration mémoire. Ce nouveau cadre permet d'optimiser la consommation d'énergie des SoC tout en faisant correspondre les performances requises en termes de puissance et de performances, de bande passante mémoire et de temps de latence.

Mots-clés—efficacité énergétique, système de mémoire, Electronic System Level, conception du système, co-conception matériel/ logiciel, SystemC.

Abstract

Mobile devices, at each new release of the standards and following users' continuous requests of new services, have to support more and more features, which are also becoming more and more demanding from the computational point of view. As a consequence, being able to fulfil new requirements and at the same time to provide power efficient chips is nowadays the most important challenge for mobile devices system designers.

To tackle this challenge, novel system level performance and power modeling approaches have been proposed allowing hardware/software (HW/SW) architectures to be explored right at the very first steps of a System-on-Chip (SoC) design flow. However, existing solutions have limited support for the power optimization of the memory system (including SDRAM) that may occupy more than 70% of a chip area and consume more than 30% of the total energy. In our work, we propose a SystemC-TLM-based simulation framework at Electronic System Level (ESL), which is able to support the joint exploration of a SoC architecture and its memory configuration. This new framework helps in optimizing the SoC energy consumption while matching the required performance in terms of power and performance, as well as of memory bandwidth and latency.

Keywords—power efficiency, memory system, Electronic System Level, system design, HW/SW co-design, SystemC.

Preface

This PHD is a CIFRE PHD project with Intel. It started at Intel Mobile Communications Sophia Antipolis, and then finished at Intel Munich.

For reasons of confidentiality and security measures, we will be limited on describing Intel environment and detailed information. Thus, the Chapter III represents a compact version of the work actually done.

Table of Contents

Table of Figures

Acknowledgments

Ten years ago, I decided, let’s do Computer Sciences. That was the beginning of a new chapter in my life!

Today, I'm finishing my education journey in Computer Science field.

Writing this PhD has been a truly challenging experience for me and it would not have been possible to do without the support and guidance that I received from many people.

I would like to first say a very big Thank you to my supervisors Pr. Michel Auguin and Pr. François Verdier for all the support, advices and encouragement they gave me during my PHD but also earlier during my two internships with them. Without their guidance and constant feedbacks, this PhD would not have been achievable. Mr. Michel Auguin, despite his retirement, continued to correct the report and work with us, for which I’m truly grateful. I really needed the good mood and the positivity of my supervisors to overcome the obstacles that I faced during the thesis. THANK YOU!

Additionally, I would like to express my very great appreciation to Intel Company for funding my PhD. My sincere gratitude also goes to my manager Juergen English for his encouragement and supervisory role. I also wish to acknowledge the help provided by Valerio Frascolla in writing the conference papers, as well as his guidance in many of my decisions. Many thanks to Patricia Guitton, Didier Martinot, Ralph Hasholzner, Raimar Thudt, and Vikas Patil for their collaboration and help. It was a good experience working with them. I learned a lot from them. I also want to thank my colleagues from Intel Munich, with whom I shared many of my lunch and coffee outings during my travels!

I am deeply grateful to all members of the jury, Mr. Pascal Benoit, Mr. David Navarro and Mr. Arnaud Tisserand for agreeing to read the manuscript and to participate in the defense of this thesis.

I would also like to thank all my friends, you know who you are, for the time we spent together to lower the burden of the thesis and continue to work with a very good mood.

I would also like to thank all my colleagues of the LEAT for sharing a lot of fun with them and, setting a positive work environment.

Last but not least, my biggest thanks goes to my family: my mother Mariem and my Father Nabil, my sisters Rihab and Abir, and my sister's husband Zied. You are my source of life!

I. Introduction

1. Context

The innovation pace of the wireless communication world is breathtaking, not only due to the fierce competition, but also due to the yearly cadence with which standards bodies deliver a new set of functionalities and services to be supported. In this very dynamic context, optimizing products and differentiate against the competitors is key for all those who want to be successful in a make-or-break market.

With the advances in semiconductor technology that have continuously reduced the technological nodes, the problem of power and circuit heat dissipation have tended to worsen at the same time. Such problem can lead to a degradation of the performance of a system, or even endanger its correct functionality. An interesting study on the impact of power consumption on the communication performance of a smartphone is published in [1]. Thereby, it is shown that the reception rate of a 5G smartphone is directly dependent on the proportion of power allocated to the baseband processor (P_{BP}). Results show that R_{max} (the maximum receiving rate of smartphones) can be improved using two ways: applying the latest semiconductor technology to chips (as presented in the part (a) of the Figure 1), and increasing β the proportion of P_{BP} with respect to the total power consumption of the whole chip (P^{chip}_{comp}). However, the power allocated to the baseband processor results from the global power budget authorized of the whole smartphone and the

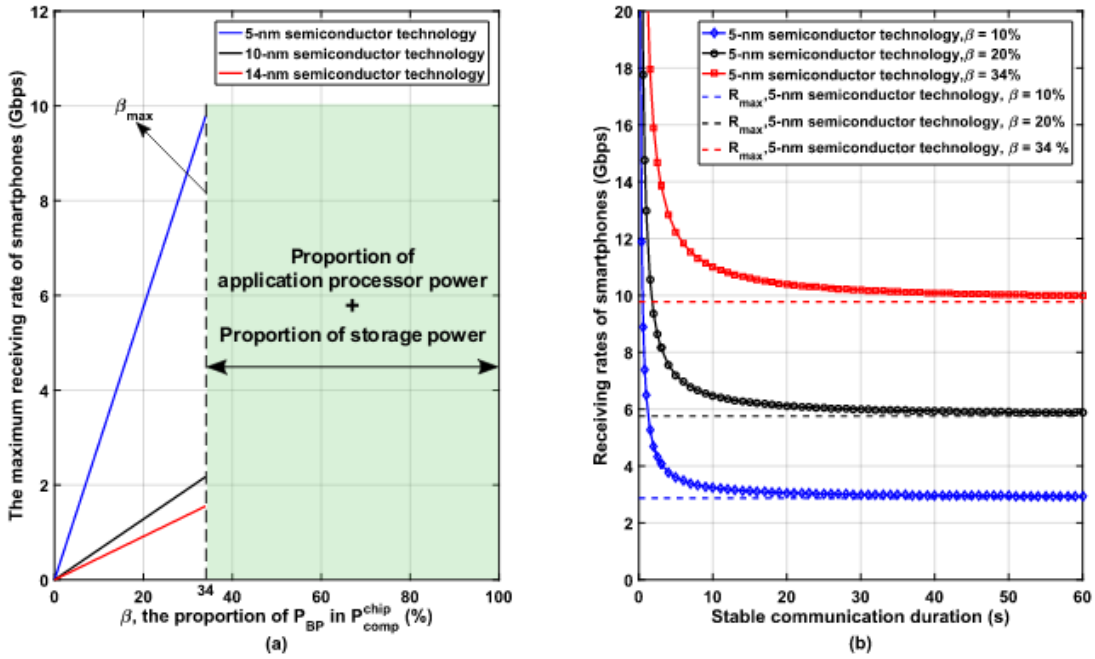


Figure 1. (a) The maximum receiving rate of smartphones; (b) Receiving rates of smartphones VS. stable communication duration

amount of power consumed by the application processor and the memory system (in this case, β_{max}= 34 percent). Thus, reducing the power consumption of the application processor and the memory system makes it possible to increase the reception speed of the smartphone. Consequently, mastering power and performance in order to have the best compromises with the complete system of all hardware blocks is obvious.

In our work, the focus is at platform level, where memory system power optimization, a key optimization in the design of mobile terminals, is taken into consideration. We start with a survey of some broad used memory technology, then we introduce some powerful power-modeling tools and finally describe the System-on-Chip (SoC) modeling approach proposed to simulate SystemC-Transaction Level Modeling (TLM) based platform for power and performance assessment.

The problem of power management in terminals has been broadly discussed in literature [2], [3] and has mainly identified processor power consumption as the most important contributor to the overall system power consumption. However, recent analyses show that the power consumption caused by the memory subsystem represents a significant part of the total power. In [4] authors report that about 35% of the total energy of a Samsung Galaxy S3 I9300 is due to data movements in the memory system on video-playback applications. As a consequence, one can say that the memory hierarchy of a mobile phone platform, and more in general in the overall memory system, do require a higher attention when power optimizations are to be looked for.

The memory system is composed of a controller, potentially some caches, and different kinds of memories, e.g. on-chip RAM and a generic external memory that is used as the main memory of the processor, which could be implemented via Double Data Rate (DDR) DRAM, which is known to be the traditional main memory, or using other more recent technologies like the Low Power DDRx (LPDDRx) family. Memory system performance and power consumption do not depend only on memory design but also on workload (e.g. the cellular protocol stack executed on the communication processor (CP)), the kind of memory controller in use and on the overall system configuration. The interactions between the memory system and the different components of the mobile SoC need to be accurately considered, to achieve a satisfactory estimation of power consumption. Thus, evaluating its power management strategies and performance efficiency requires an approach that includes an appropriate model of the memory, a realistic controller, a high-precision power model, and a representative workload of real use cases. All of the above requires a full system simulation framework able to represent the impact of the different mutual interactions between all the components in the platform and the memory system. Hence, designers of memory system would need easy-to-use and high-precision power models that estimate power and energy consumption of the different operations and state transitions of the memory system. Among those, we use DRAMPower [5] which is an

open source Dynamic Random Access Memory (DRAM) power and energy tool for estimating the power consumption of different DRAM operations at the cycle-accurate level.

The PwClkARCH library (discussed in details in Chapter II, Section 4) presents a framework that helps designers to model the impact of the SoC activities on power consumption by means of Virtual Prototyping (VP), describing the SoC using SystemC-TLM. The library also allows for exploration of different power management strategies, like Dynamic Frequency Scaling (DFS), Dynamic Voltage Frequency Scaling (DVFS), Power Gating and Clock gating. The PwClkARCH library by itself is still not sufficient to give a precise view on main memory power consumption, especially at the refinement level of each operation. In fact, it is such information that is needed to properly explore different configurations and compare different designs of memory systems. Therefore, we need to make work together the PwClkARCH library with a DRAM functional simulator able to evaluate the power consumption as well.

Our approach is to be used at a very early stage of the SoC design flow, as it focuses on a pre-silicon simulation environment at a high level, before the split between SW and HW implementation has been defined. Joint optimization of, e.g., cellular protocol stack SW architecture, mobile SoC HW architecture as well as HW/SW partitioning is enabled by this approach. In fact, our work allows to analyze the impact of different architecture options on mobile SoC power consumption as well as on the performance of cellular protocol stack SW, controlled in power, executed on the CP.

2. Contributions

In this thesis, we propose a framework for power modeling and management at ESL level able to support both CMOS common hardware block and DRAM block. This framework is based on SystemC-TLM, PwClkARCH library which is implemented by the LEAT laboratory and DRAMPower simulator which is an open source tool. Using the framework, designers can explore the architecture of their device or product by measuring the performance and the energy consumption during each simulation. They can then choose the appropriate technology and DRAM configuration for their use case. Our methodology takes into account the interactions between the DRAM and the other blocks, and the memory power model is accurate enough to consider the operations of the DRAM to calculate the power dissipation.

3. Structure of the Dissertation

Chapter II presents the state of the art of this project. We start by defining the level of abstraction used during our work, we introduce PwClkARCH library and we give a survey of the memory system in relation with power consumption (memory requirements and trends).

In Chapter III, we describe the Intel environment used in our work which constitutes the target of our project. This environment is a pre-silicon simulation environment for Intel modem devices.

In Chapter IV, we explain our work on PwClkARCH using Model Driven Engineering approach to facilitate the use of the library with graphical and textual user interfaces.

Chapter V illustrates our contribution around ESL-level memory technology exploration based on power consumption.

The Chapter VI is dedicated for the simulations results. We show our plots and we give our analysis based on the state of the art and the use cases we used.

Chapter VII concludes this dissertation and describes some areas for future works.

II. Power-aware architectures modeling: Background & State of the Art

Following Moore's law, the SoC complexity increases constantly due to the continuous addition of new features that require the integration of a growing number of processing cores and IPs in a single chip (Figure 2). This have a significant impact on implementing new computing paradigms and enhancing computing possibilities. The period between 2000 and 2010 experienced a constant improvement of wireless devices connectivity capabilities, which in turn helped smartphones to offer more and more better services at each new generation. In parallel, connected IoT (Internet of Things) devices and sensors also experienced a proliferation. Between 2006 and 2014 the average number of IPs embedded in a SoC increased from 30 to 120, whereas integrated processor cores increased from 1 to 20 [6]. As a consequence, the SoC design becomes harder along with the underlying power consumption becomes more and more important. In order to face this issue, several means exist in the industry for power estimation and management, but they are still based on low-level CAD (Computer-aided design) tools, take time and require a rather detailed software model of the hardware architecture blocks. In fact, even if cycle-accurate tools give quite accurate power estimates, they require a high amount of simulations imposing to move up to higher abstraction level to address efficiently the first steps of the design flow. Thus, Transaction Level Modeling (TLM) appears as the most effective approach to assess designs with rapid power estimation tools.

Moreover, memory system consumes more than 30% of the overall phones energy when executing applications [7]. Designers prefer to have high memory bandwidth to address the bandwidth needs of new chips and devices. There are several memory options either in terms of technology (DDR, LPDDR, NVM, etc.) or in terms of on-chip/off-chip choices. The possible combinations are numerous, and we have to choose a good tradeoff between bandwidth, power efficiency, cost and reliability. This task is not obvious and need to be addressed at the very beginning of the design flow.

The power estimation existing tools, industrial or academic, are numerous, but unfortunately they treat the memory system separately, that is, without considering the impact of the requests coming from the rest of the system to the memory or the inverse. In the industrial world, they usually use real-world measurements to generate or use the memory device datasheet to estimate memory consumption, and at the higher level, they consider powerful memories of large sizes and high bandwidth to not block or degrade performance. This leads to big problems later in the design cycle. So, treat these problems early in the design flow and define the adequate memory system answering the needs of the product in terms of performance and power consumption is very important.

All this presents the general context of the thesis, which can be divided into two points of interest: the first is the energy consumption analysis and control, and the second is the memory system design. One can imagine then many working hypotheses which vary widely according to the needs of the designer. Already, when we say energy consumption we actually refer to two different aspects: estimation and management. Our work is to develop a framework to help designers for exploring different memory system architectures, in terms of technology and configurations, according to their energy consumption (and when possible by also applying some power management techniques), and also their performance at the beginning of the design flow (ESL level).

This chapter is dedicated to the presentation of basic concepts necessary to the understanding of our research works.

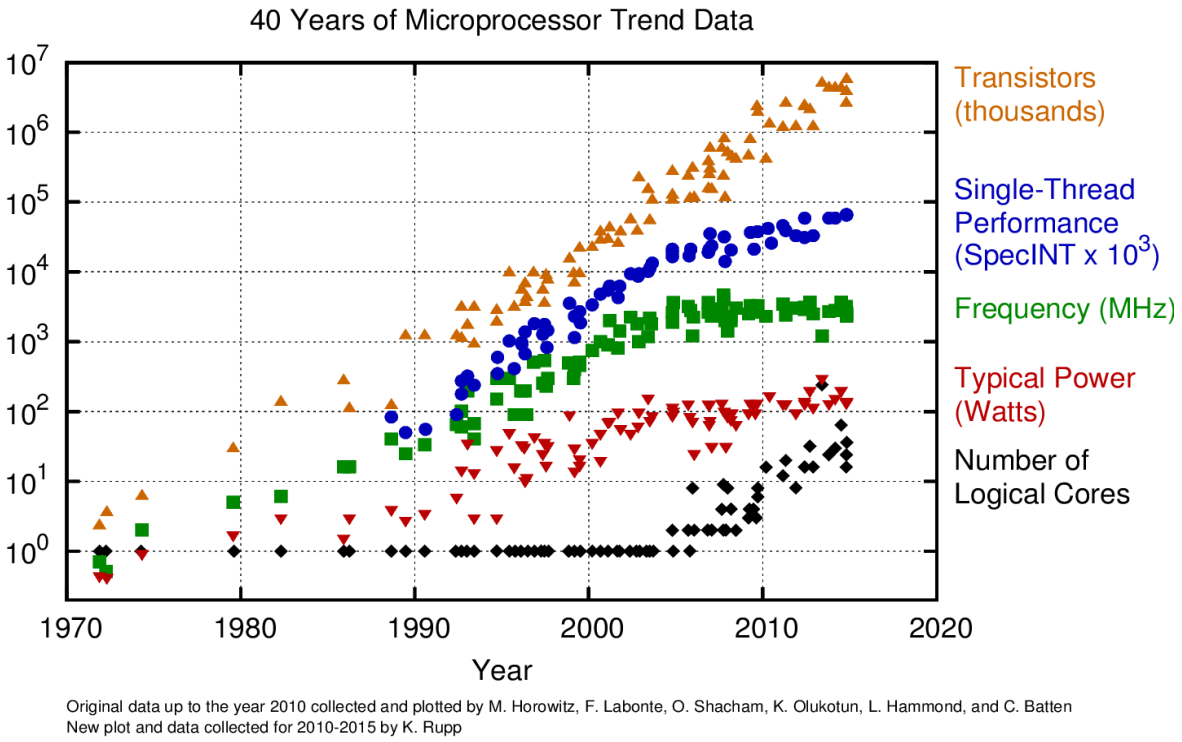


Figure 2. Technology trends according to the Moore's law [8]

1. System-on-Chip Design flow: Level of abstraction

Several design steps are required to implement a system on chip. Current SoCs consist generally of several processors, memories and peripherals. The Figure 3 represents the design process abstraction levels that describe the process of SoC realization from design concept to fabrication. From top to down, the simulation speed decreases and the design becomes more complex as we add in each level certain details. The problem

in focus is to find the best tradeoff between simulation speed and accuracy in order to properly model platforms and IPs correctly so as to mimic the system's behavior and allowing overall system verification.

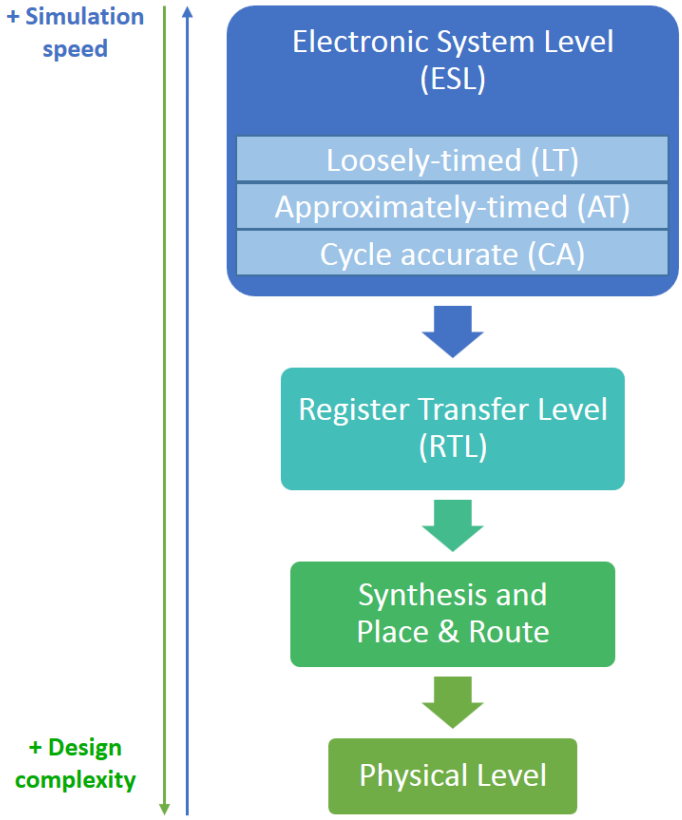


Figure 3. Design process abstraction levels

Efficient computer architecture simulators are available supporting the actual execution of software, including the operating system. For example, Gem5 simulator [9] is a modular platform for system-level architecture and processor microarchitecture. It contains CPU models, GPU models, event-driven memory system including caches and DRAM controller models, and many other features. Gem5 is written primarily in C++. The goal of Gem5 is to model a system with heterogeneous applications running on a set of heterogeneous processing engines, using heterogeneous memories and interconnect. Gem5 provides also a complete platform for research in future energy-efficient systems by enabling DVFS on the Linux kernel for run-time power management. Such simulators are mainly CPU-centric and GPU-centric. They are not well adapted for modelling complex SoCs including communication processors and specific IPs that can generate heavy traffic to and from the memory sub-system, concurrently with the CPU/GPU sub-system, inducing then interferences between execution of all these processes. Note that Gem5 has been connected with SystemC in [10] in order to address the full simulation of MPSoCs. As in our work we consider a task

graph-based modelling of the software part, a CPU/GPU cycle accurate simulator such as Gem5 is not required in this case. This task-graph methodology captures the control flow of the application and provides only the execution workload of each task to the underlying architecture simulation models. Thus, each model simulates the execution latency and the power consumption for an application-task rather than executing real instructions or operations of the software. Therefore, we are able to explore the power and the performance earlier in the design phases of system development process, as task-graph methodology does not require detailed implementation neither of the application, nor of the architecture.

1.1. Electronic System Level (ESL)

Electronic system level (ESL) design and verification is an electronic design methodology focusing on higher abstraction level in a design flow to make as early as possible effective architecture decisions by prototyping, debugging and analyzing complex systems before the RTL (Register Transfer Level) stage. At this stage, VP techniques are used, as they represent in convenient way simulation models of the entire system architecture and the verification environment. VPs consist generally of transaction-level (TL) models, where the simulation speed and the accuracy of the models are balanced. VP are very useful for architecture explorations, performance and power simulations, as well as for preparing system test cases for system verification.

The traditional power management techniques at RTL level or place and route (P&R) are not suitable anymore to perform efficient design space explorations for complex systems, because their simulations are very time consuming due to the multitude of blocks and signals to consider in a SoC. Moreover, to reduce the model complexity, they target power optimization in individual blocks without considering a global view of the system. These tools are efficient at IP-level. However, the speed of design is very important to keep the time-to-market short and to develop competitive products. For that reason, high-level power management on ESL level is desired as it gives ability to the designer to take efficient design decisions at an early design stage.

1.1.1.SystemC

SystemC [11] is basically the standard for ESL design. It is a C++ class-based library for system and hardware design defined by the Open SystemC Initiative (OSCI) that has been merged with Accellera [12]. From December 2005, SystemC has been known also as IEEE-1666 after its standardization from IEEE standards association. Since SystemC is based on C++, it supports all the C++ data types and provides also SystemC data types generally preceded by ‘sc’ like *sc_int* or *sc_signal*.

Design components are defined as modules in SystemC. They are classes that inherit from the *sc_module* base class. Modules may contain methods, ports and channels for connectivity. In SystemC, there are two

kinds of processes namely SC_METHOD and SC_THREAD. SC_METHODs are like functions in C++ or Verilog and they don't consume time i.e. simulated time. However, SC_THREADs can consume time by using the *wait()* function, either for specific time, or for events.

Connecting ports in SystemC is an operation that is basically divided into four steps: declare the port and the channel, instantiate the port and the channel, bind the port to the channel using *bind()* method and finally use the port. Events and sensitivity give SystemC the power to implement hardware prototypes in terms of concurrency and consuming time. Simulated time is managed by the SystemC kernel using *sc_time* datatype.

SystemC simulator has two phases: elaboration and simulation. The former is the phase where design is setup: objects like modules, ports, exports and channels are instantiated. The latter is used to run the resulting system model.

Since 2011, SystemC standard integrates TLM-2.0 transactional modeling which allows functional validation based on SoC simulation.

1.1.2.Transaction Level Modeling (TLM)

The idea behind TLM is to model each component only if it has something to do. The individual components communicate by sending messages requesting data be transferred between each other. The exchange of messages is called a transaction, and the approach is called Transaction Level Modeling (TLM).

The Open SystemC Initiative (OSCI) TLM 2.0 [13] offers a standardized approach to use TLM. It explicitly addresses VP thanks to which SystemC models can easily be organized and communicating within a system. Consider the system as a set of SystemC modules, each has one or more sockets through which the SystemC modules may read and write data. The activity of each module is modeled by a number of parallel threads which communicate with the threads in other modules by passing data through the sockets. This communication is known as a transaction and the data passed as a payload. A module's threads may act as either initiators or targets (Figure 4). An initiator is responsible for creating a payload and calling the transport function to send it. A target receives payloads from the transport function for processing and response. Initiator calls are made through initiator sockets, target calls received through target sockets. A module may implement both target and initiator sockets, allowing its threads to both generate and receive traffic.

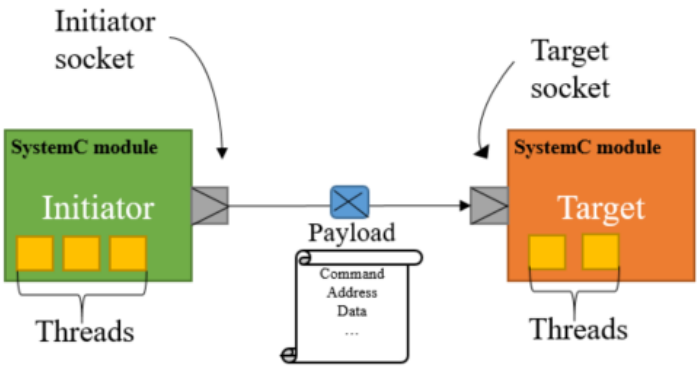


Figure 4. SystemC-TLM Transaction background

The models in the SystemC-TLM Library have been developed in partnership with major IP (Intellectual Property) providers, among which Intel, ARM, Cadence, ST Microelectronics, and Synopsys, giving the designer access to IP vendor reference models and ensuring their correct behavior. Those models are typical of the IP needed by designers to build platforms, debug hardware and software and verify the design before moving to the RTL implementation flow. For instance, Intel uses the SystemC-TLM standard and develops on top of it a productivity layer. The resulting library is Intel property, but it is interoperable with standard SystemC-TLM coding style.

The SystemC-TLM2 standard [13] focuses on use cases such as software performance analysis, hardware architecture analysis but does not provide any intrinsic features for power analysis.

1.2. Register Transfer Level (RTL)

Digital electronic circuits have been mainly designed at the Register Transfer Level (RTL) since the late 1980s. This level expresses the scheduling of operations and data transfers in clock cycle. For the structural aspect, these operations are projected on elementary material resources (registers, arithmetic operators, etc.). Indeed, at this level of description, data transfers and logical operations between the registers are described via signals. An RTL design is usually captured using a hardware description language (HDL) such as Verilog and VHDL. Such languages benefit from many years of development and constant improvements so that the design of a system is almost automatically synthesized down to circuit layout ready for fabrication, provided that efficient command scripts are used in the design flow.

The complexity of hardware design has induced very slow simulations and time-consuming implementation of RTL models. In [14], authors perform test to measure performance increase of design by comparing TLM and RTL level modeling. Figure 5 represents their results of line code comparison. Tiers are general terms of displaying embedded system components, which communicate together, for example, 2-tiers means there

are two components communicating together. Without going into details, results show that the amount of lines needed to model system by using TLM is less than that of using RTL. This is explained by the fact that TLM is used for hiding hardware aspects that are not important at system level and so gain effort, number of code lines, and also time in the simulation.

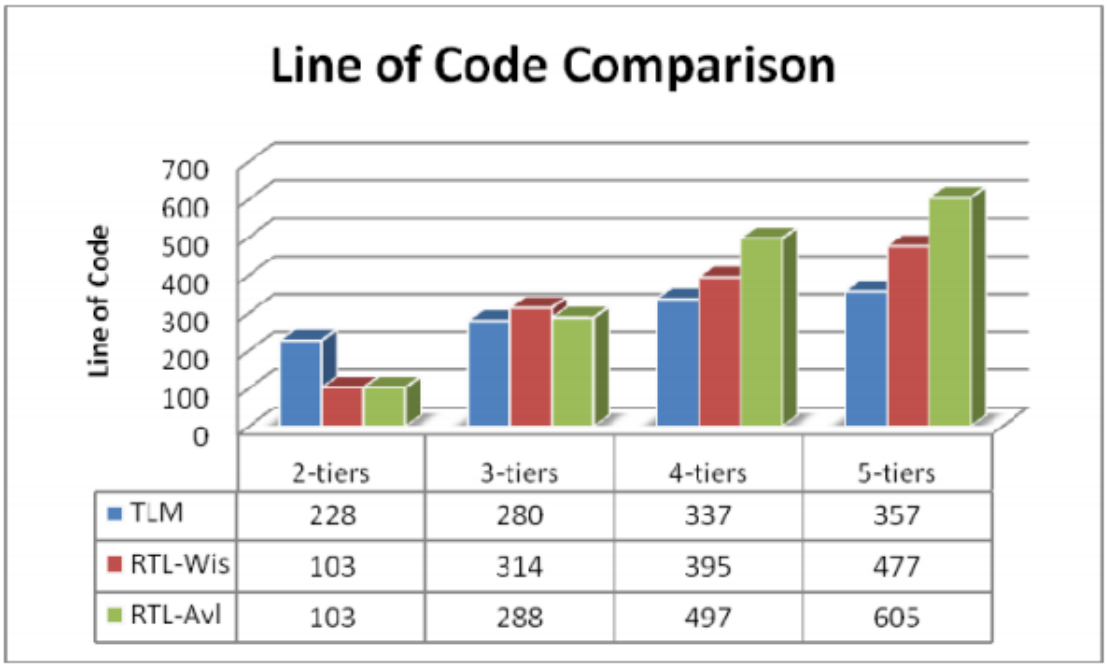


Figure 5. Line of code Comparison [14]

1.3. Synthesis and Chip Fabrication

The transitions between the abstraction levels of the design are achieved through the synthesis processes. The high-level synthesis known as HLS is the transition between ESL level and behavioral RTL level. HLS tools generate a custom architecture (memory banks, data path: registers, multiplexers and buses) that implements a high-level specification [15].

Then, there is an RTL synthesis that target technology cell library to obtain the structural RTL which is the logic synthesis. In this phase, RTL specification of the design is transformed to logic gates implementation, using synthesis tools like Design Compiler from Synopsys or RTL compiler from Cadence.

Finally, we have the layout synthesis or what we call place and route process to obtain a physical representation. Cadence proposes many solutions at different levels like Stratus for high level synthesis, Genus for RTL synthesis and the Encounter® technology.

2. Power modeling: power consumption and power management techniques

2.1. Power Consumption of System-on-Chip

The total power consumed *P* in the CMOS technology consists basically of two factors: dynamic or switching power and static or leakage power. Dynamic power is due to the charging and discharging of capacitances associated to each transistor whose state toggles. So, the dynamic power ($P_{dynamic}$) is the power consumed when the device is active and so the signals are changing values. Static power (P_{static}) is due primarily to gate and channel leakage in each transistor. It represents the power consumed when the device is powered up but the signals are not changing values. Note that in this work, we neglect the short-circuit power which represents the power dissipated by an instantaneous short-circuit connection between the supply voltage and the ground at the time the gate switches state. In the literature, this component has been neglected in comparison to the switching power [16].

The energy consumption of a system *E* can be defined as the summation of both spatial and temporal power consumption of circuits.

$$E = \int_0^t P dt (in\ Joule)$$

$P = P_{dynamic} + P_{static}$

$P_{dynamic} = a*C*V^2*F_{clock}$ *(in Watt)* *(1)*

$P_{static} = V*I_{leakage}$ *(in Watt)* *(2)*

- a is the switching activity
- C is the capacitance
- V is the supply voltage
- F_{clock} is the clock frequency
- $I_{leakage}$ is the leakage current
- t is the execution time of an application

Most of these parameters are technology dependent and may come either from technical datasheets or simulation results at low design stage or from physical measurements on real circuits.

Today, the power reduction is one of the most important design goals of battery-powered devices. Developers are concentrating on reducing power consumption and on gaining better control on power dissipation of their products in order to satisfy the customers' requests.

Designers are used to make tradeoffs between power and performance. Increasing the frequency often will increase the energy consumption since voltage must be adapted to support the frequency change.

Alternatively, the supply voltage can potentially be reduced resulting in lower performance but significantly reducing energy consumption because the quadratic dependence of power on voltage. Designers handle that using several techniques to reduce the power by controlling voltage and frequency values. Some of these techniques are introduced below.

There exist several power management techniques reducing the dynamic power or/and the static power of SoCs. Power management techniques may be applied at various levels of the design flow and so with different granularities. Although each design stage offers an opportunity to save power, higher levels of abstraction have greater impact. In DAC 2012, Shawn McCloud, vice president of marketing at Calypto said: “Today, many optimizations are made at the gate level and lower. But the real impact on power occurs when you move up to the architectural level. You can look at the options for resource sharing and the tradeoffs between parallelism and operating at lower frequency".

According to [17], the best three techniques for reducing energy consumption are Clock Gating, Power Gating and Dynamic Voltage and Frequency Scaling.

2.2. Clock Gating and Power Gating

Clock Gating is one of the most efficient techniques for reducing power consumption in particular the dynamic power. It is based on disabling the clock of the design components when they do not execute any computation. This leads to power savings in the clock tree. Power gating nonetheless, is the most efficient way to reduce leakage power. If a block is not used in some time intervals, it is powered down in these periods.

The basic strategy of both techniques is to provide two power modes: a low power mode and an active mode. The goal is to switch between these modes at the appropriate time and in the appropriate manner to maximize power savings while minimizing the impact to performance.

2.3. Dynamic Voltage and Frequency Scaling

Dynamic Voltage and Frequency Scaling (DVFS) is a very popular technique to reduce the dynamic power. DVFS allows the voltage and the clock frequency to be decreased and increased dynamically according to the processing demand. DVFS works on two fundamental concepts: first, processing units have some “IDLE” time after performing required tasks (at full speed); second, dynamic power consumed by digital circuits is proportional to V^2. The interdependence between voltage and operating frequency should be managed in a good way. The optimal combination between leakage and switching power has to be found as shown in Figure 6.

Figure 7 presents a typical DVFS architecture. Both clock generator and voltage generator are managed by a controller. The controller is responsible of deciding when and how the clock frequency and voltage will change. The control method is usually defined, at least in part, by the software, although it can be coded in the hardware. More generally, the complete control algorithm can be implemented in the software [18].

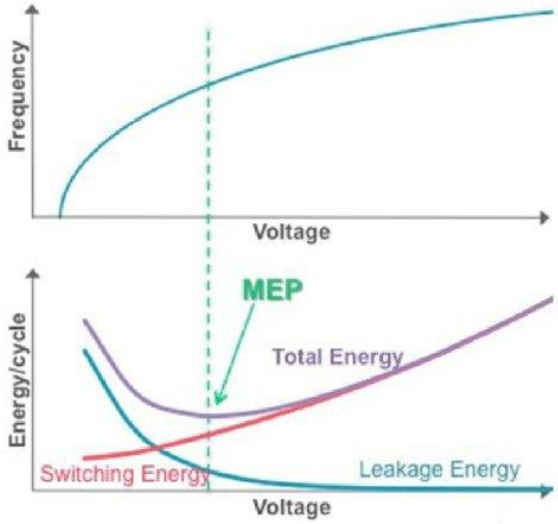


Figure 6. Energy (dynamic and leakage) vs Voltage [19]

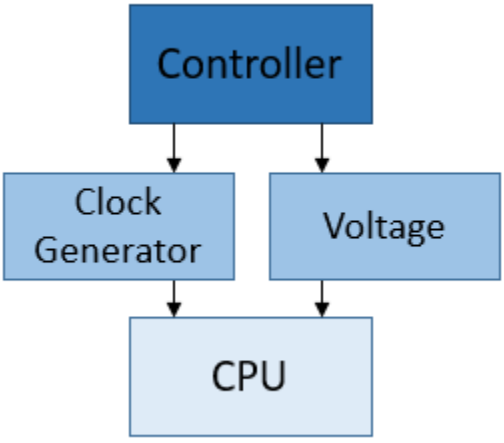


Figure 7. Dynamic voltage and frequency scaling (DVFS) architecture

Next, we present some of the TLM-based power management existing tools and frameworks. We start by introducing the Unified Power Format (UPF) standard which is almost the only standard that exists and recognized by most of design tool providers.

2.4. Unified Power Format (UPF)

UPF has offered a clear design flow utilizing the power management techniques and enabled RTL power-aware verification. The UPF committee was formed by the Accellera organization. The first approved version 1.0 was published in 2007. UPF was used to partition a design into power domains and to apply different power management strategies to control logic values when the domains are being switched off and on (Figure 8). UPF is a Tool Control language (TCL) based syntax. It defines the main features of a power management architecture separately from the functional specification. From version to version, some refinements were included, and several commands were added to remove ambiguities or to fix errors identified in previous versions.

A power domain is a collection of design elements that share a primary supply set. According to UPF terminology, a supply set associates multiple supply nets (such as power and ground much like an electrical plug) as a complete power source for one or more design elements. A supply net so represents the power and the ground rails in the design. UPF allows the creation of a hierarchical structure of power domains. For example, in Figure 8, TOP is a top-level power domain that includes TX_AON, CRC_GEN and RECEIVER power domains. These power domains are created with the following piece of TCL code:

create_power_domain TOP

create_power_domain TX_AON -elements {transmitter power_controller}

create_power_domain RECIEVER -elements receiver

create_power_domain CRC_GEN -elements checker

Power management strategies are implemented through Power State Table where lines correspond to specific system power modes. A system power mode is a legal combination of specific power-domain states. The set of power-oriented components (power domains, supply nets, …) added to a system to control its power consumption is called a power intent.

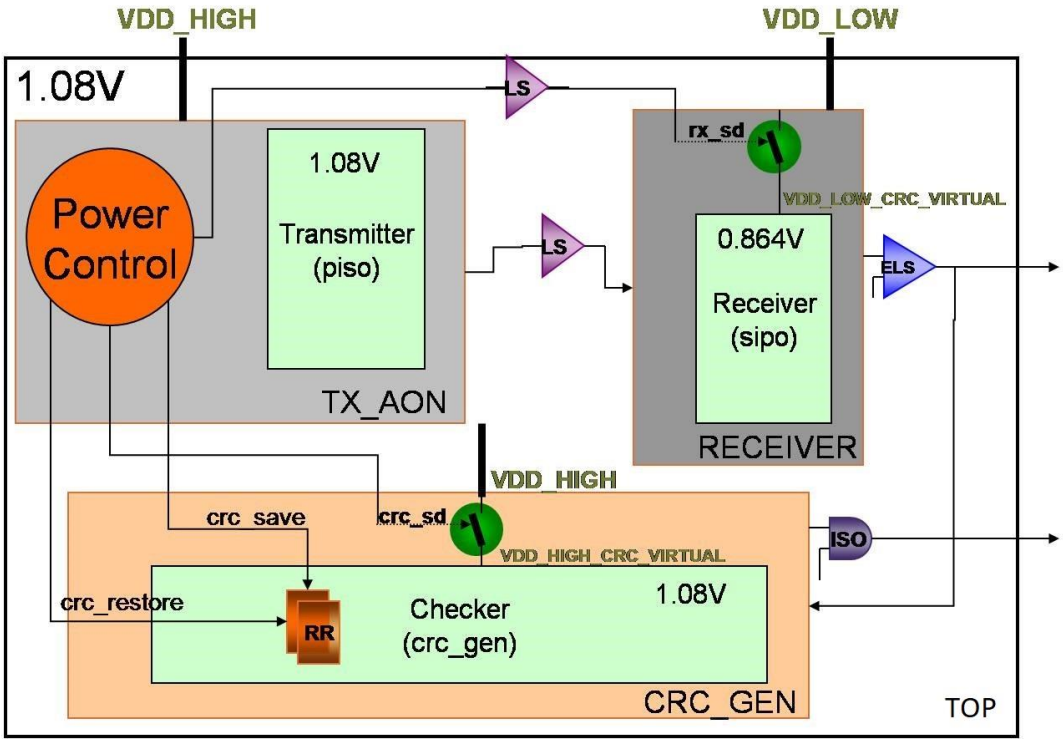


Figure 8. Example of UPF-based power aware design (Source: Magic Blue Smoke)

However, this standard has a scalability issue as it becomes very complex with complex design. As the trend of designing SoCs is to use abstract models above RTL, UPF follows the trend and was updated to version 3.0 [20]. This version boasts support for ESL power modeling and analysis in VP applications. Compared with previous releases of UPF, UPF 3.0 introduces mainly the capability of capturing power states associated with IPs, supply nets and power domains. A power state is a combination of voltage values (and frequency values) of supply nets and logical expressions attached to signals controlling power suppliers. A "functional state" or simstate in UPF (e.g. RUN or DEEP SLEEP) can be associated to a power state to precise the simulation behavior of the cells in this state. Logical expressions can be captured as well to express conditions on transitions between power states. A logical expression can refer to states of supply nets and/or power states of power domains and/or some control conditions issued from the HDL design. At system level, power models as introduced in UPF 3.0 are used in tools such as Platform Architect MCO from Synopsys [21]. But for us even UPF 3.0 does not cover all the needs to support an easy performance/power system design space exploration (DSE) where power/performance tradeoffs are to be investigated, and global power management strategies are to be defined and validated. Moreover, UPF is inflexible and demanding on the tool chain and requires simulators as well as debug environments (Figure 9). As a result, many companies have created their own solutions for power related design. Next, we will present some of these solutions.

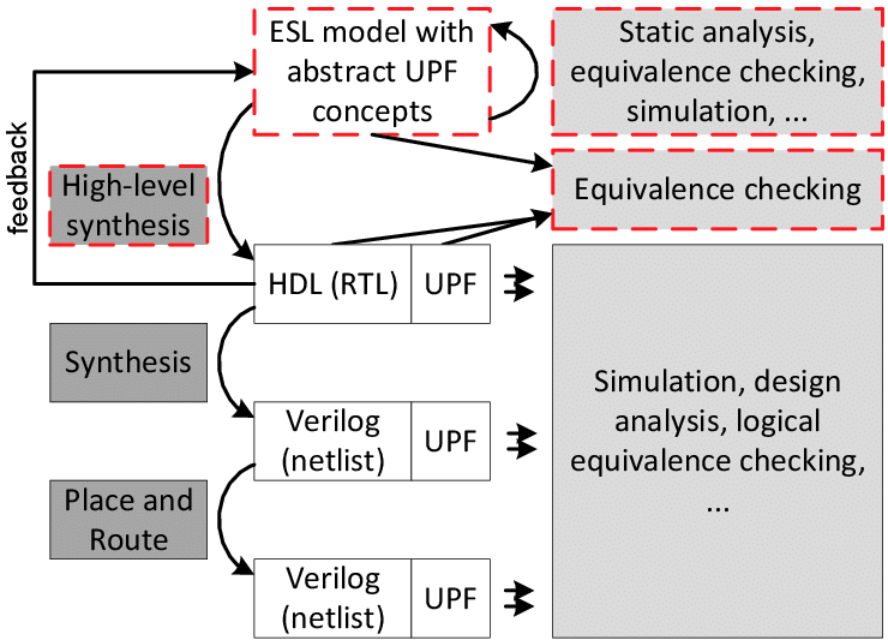


Figure 9. Low-Power systems design flow [22]

2.5. ESL power and performance assessment frameworks

2.5.1. Synopsys Platform Architect

Synopsys Platform Architect (Synopsys PA) MCO (Multicore Optimization Technology) is a SystemC TLM standard-based graphical framework for ESL design, and early power and performance analysis of a multicore SoC architecture [23]. In Synopsys PA the application is modeled as a task-graph. Where each node represents an application task which is instantiated from basic blocks of a Generic Task Library (GTL). The edges in the task-graph represent the dependencies between the nodes. The task nodes use communication token samples for activation and synchronization purposes. Tasks read their input ports, consume or process the input tokens and generate output tokens. The behavior and execution time of each task for actual processing of the incoming tokens is specified by some generic functions, such as for data processing and memory access, which are connected to each task node during instantiation of the corresponding node. The application model, under the control of a default task manager, can be simulated stand-alone to analyze the application behavior and memory traces. The architecture model components are instantiated from a library containing Virtual Processing Unit (VPU) task-driven traffic generators, memory blocks and interconnect subsystems. The mapped task graph on a given architecture can be simulated to analyze the overall power and performance of the system. As in UPF, power consumption in PA framework is described as a high-level state machine. Each state is associated with a certain power consumption and triggered by events detected in the virtual prototype.

2.5.2. *Intel CoFluent Studio*

Intel CoFluent Studio [24] is a commercial model-driven tool for design and exploration of complex embedded systems in the early phases of the design process. In CoFluent the application is modeled, using a GUI tool, as a network of function blocks or process communicating with each other through communication elements such as events, shared variables or message queues. Once the application is modeled, a timed SystemC-TLM model of the application can be automatically generated. The simulation of this timed model can be used for analysis of the application model. Similarly, using the same GUI tool, the system architecture model is also created by connecting different generic architecture resources, such as processors, interrupts, interconnect and memory blocks. The architecture resources are characterized by high level performance attributes. The mapping relation of application model with the architecture model is expressed by creating mapping models of the system. In the mapping models, each function or process in application model is mapped on one of the processing resources in the architecture model. Similarly, the communication path between the processes in the application model is also mapped on the interconnect subsystem in the architecture model. The shared variables and message queues in the application model are mapped on the architecture memory blocks. The SystemC-TLM generated code of the whole system allows the analysis of the power, cost, resources load, memory footprint and dynamic behavior of the application on the architecture. Thus, Intel CoFluent itself doesn't include any power analysis features.

2.5.3. *Intel Docea Power Simulator*

Intel Docea Power Simulator (IDPS) [25] is a software solution for creating power and thermal virtual prototypes of SoCs at system level. Intel Docea Power Analytics (IDPA) [26] is used to collect and structure all the data needed for power analysis in a database. The tool is post-processing power analysis in the way that, from VCD files generated from SystemC-TLM simulation according to the power model defined in the tool, it generates power values needed for analysis. The connection between the two simulators requires to insert specific code in the HDL design model to transmit the relevant events and data from the HDL design to trigger power state transitions in the power model. The power model of the system is a hierarchical set of component power models. A component power model is defined as a Power State Machine (PSM) composed of one or more power states. The user provides power equations, static and dynamic, for each power state. A timed succession of power states defines a static scenario, which helps study power and thermal behaviors. Intel Docea Power Simulator is thus a help for measuring the total power and temperature, but not at all for defining the power optimization strategy.

2.5.4. *Mentor Graphics® Vista™ platform*

Mentor also invests in new generation tools at ESL level that help model performance and power at the early design exploration stage. The Mentor Graphics® Vista™ platform [27] enables modeling power and performance information at TLM level while maintaining accuracy.

In Vista™ platform power models are attached to each IP and assigned to each transaction type that the IP can receive or emit. These power models are dependent from voltage and clock frequency of the IP whose values are defined as parameters of the model.

2.5.5. Comparison of ESL-based tools from EDA providers for power analysis

The table below illustrates that apart Intel CoFluent Studio, the tools mentioned above provides power estimation facilities but are not able to model an explicit power management strategy, i.e. the decision issued from the software part (e.g. the operating system) that set up the power states of power domains according to the functional state of the system.

	Simulation model	Power Estimation	Power Management
Platform Architect MCO	SystemC-TLM	YES	NO
Intel CoFluent Studio	SystemC-TLM	NO	NO
Intel Docea Power Simulator	Internal simulator with possible connection with SystemC-TLM	YES	NO
Mentor Graphics® Vista™	SystemC-TLM	YES	NO

3. Model Driven Engineering approaches

There are also a lot of academic work on the implementation of tools for high-level modeling of energy consumption using SystemC. Without being limited to a few, we cite the work [28], in which a mode/phase-based model is used at IP level where the mode (respectively the phase) informs the power model attached to the IP about the current power state (e.g. ON) (resp. the current functional state, e.g. IDLE).

A Model-Driven Engineering (MDE) approach has been also considered in several works to support the analysis and the exploration of system architectures including both hardware and software. For example, in [29] the application, the architecture and non-functional aspects such as power consumption are modeled using a multi-view approach. But in this framework dedicated for design space exploration no effort is done to connect it with traditional ESL design flow such as the generation of SystemC-TLM code for simulation. In [30], authors develop a MDE approach for estimating power of a hardware architecture. In this work power estimators based on power states are attached to each hardware components and the energy per component is estimated by counting their periods of activity during a system level simulation. As this approach focuses only on power estimation, it is of limited interest for the exploration and the validation of a real power management strategy applied to a functional model. An interchange format XFG with energy-

based annotations is proposed in [31] to support the analysis of energy-aware real-time systems which can be described according to a variety of specification formalisms, each one adapted to a different discipline (e.g. mechanics, software). A formal semantics is defined for XFG in [31] but there is no information provided on the way to use the energy annotations in an analysis or exploration process. The approach developed in [32] proposes a UML profile targeting the design space exploration of embedded systems at a high level of abstraction. A Finite Power State Machine is associated with each CPU in the architecture to model both the power consumption and the operating performance points (OPPs) available for that CPU. A Power Policy Manager is also introduced to select the OPP according to some performance criteria. The framework allows fast abstract simulation for evaluating performance and power. But, as mentioned earlier, the use of power states to model power consumption is not really convenient for verifying that a power/clock intent and the power management strategy are consistent with a functional architecture.

Model Driven Engineering-based (MDE) approaches have been considered for design space exploration of low-power SoCs. In [33] the UML MARTE profile is extended such that Power Finite State Machine (PFSM) can be attached to hardware components. To estimate the system power consumption, power configurations need to be bound manually to the application operational modes. Authors of [34] consider stereotypes to a UML metamodel to associate power-oriented data, a PFSM and a scheduler with HW components. Such ESL specification is far from the low power UPF-style specification which means that a significant effort must be made to provide a specification conformed to the input format of EDA tools.

Systems Modeling Language (SysML) is a graphical modeling language for specifying, analyzing and verifying complex systems. It is defined as an extension of the Unified Modeling Language (UML) using UML's profile procedure [35]. Unlike UML, SysML is not limited to software systems, it is able to model in addition to software, hardware, information, processes and facilities. SysML is considered for example in [31] for capturing non-functional properties such as power consumption models.

4. PwClkARCH library

A recently proposed power modeling library called PwClkARCH implements the high-level modeling approach proposed in [36], and [37] by the laboratory of Electronics, Antennas and Telecommunications (LEAT), which will be explained in details in Section 4.2., so to assist SoC designers with a system level SystemC-TLM framework, inspired from the UPF standard. PwClkARCH is a library of C++/SystemC-TLM classes (Figure 10) allowing a SystemC-TLM architecture to be structured in power domains and clock domains (power/clock intent) such that a power management strategy could be applied to evaluate its impact on power consumption and on performance.

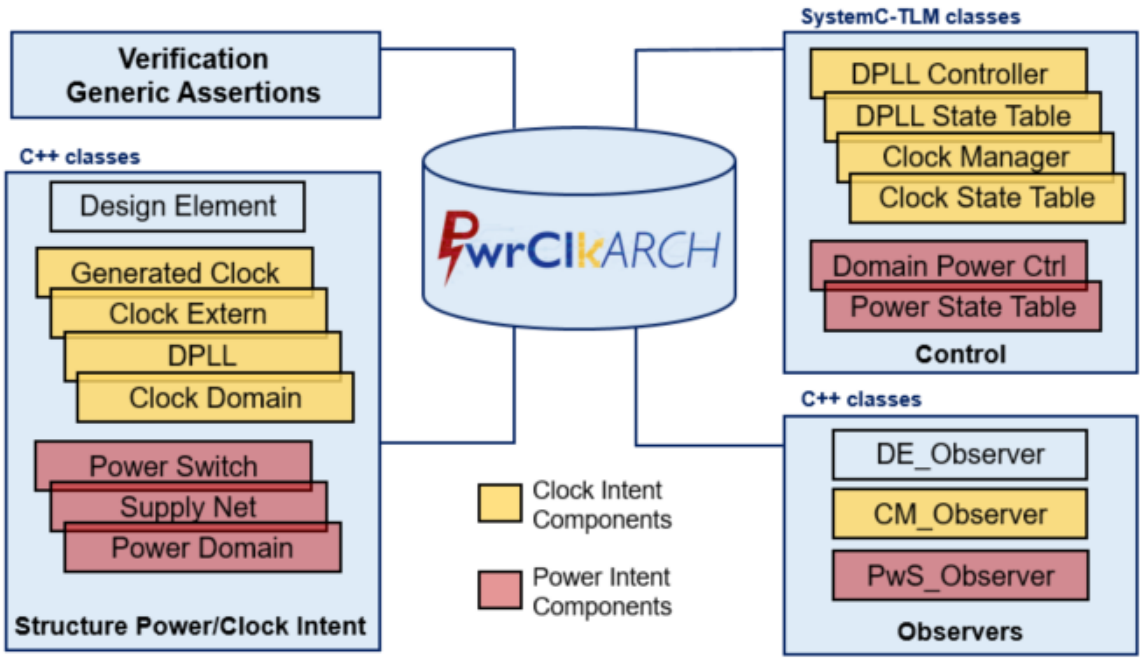


Figure 10. PwClkARCH structure

4.1. Description of the library structure

Before describing the methodology on which the PwClkARCH is based, we should define the aspects/semantics included in this library. Most of them are UPF based semantics that have been abstracted to TLM level. In UPF, a list of components called design elements (DEs) from the HDL design are assigned to a power domain. In this approach, a design element object is defined as a reference or a pointer to the corresponding module in the SystemC-TLM design model. Then a specification of a power/clock intent is built above these DEs thus reinforcing the separation of concerns between HDL design and power/clock intent. The DE class contains the generic power model of a component which is basically the sum of consumptions due to static power ($P_{stat}= VI_{leakage}$) and dynamic power ($P_{dyn}= \alpha CV^2F$). The power model of each instance of a DE is particularized with the values of parameters (e.g. capacitance C) specified with the associated module in the SystemC-TLM model.

A Power Domain (PD) is defined as a subset of system components that receive power from the same Supply Net (SN). SN represents the supply source and can be used to power up the different PD elements. Thus, each PD can be controlled individually. It can be put in different states of operation called Power States (PSs). Concepts of voltage domain and power domain are merged like in UPF, so that a power domain can be either power-gated or voltage-scaled or non-scaled according to its attached supply nets. According to the UPF standard, hierarchical organization of power domains is permitted. Thus, two types of power

domains are included in the library. A power domain of type “container” has at least another power domain of type “nested”. But also, this “nested” power domain can also be of type “container” when it includes other power domain(s).

Correspondingly, a Clock Domain (CD) is a set of system components that have at all times one single clock source on which division operations can be performed but which lead always to synchronous signals or a constant phase at the level of the CD elements. Unlike the Power Domains hierarchy defined by UPF, PwClkARCH do not introduce a hierarchy between Clock Domains.

PwClkARCH associates for a Clock Domain a single Digital Phase Locked Loop (DPLL) which represents its primary clock source and a single Clock Manager (CM) which allows to control the clock distribution to its Design Elements. CM receives a generated clock (i.e. the clock frequency value) from the DPLL in the clock domain and provides individual clock frequencies to each IP blocks in the clock domain. The clock frequency provided by a CM to a IP block results from a simple division on the CM input clock. Each clock generated by a CM can also be gated individually. The CMs are controlled by a Power Manager.

Power Manager (PM) is the intelligent block in the library; meaning that it is the responsible of the power management implementation. PM receives the power requests from the CPU for example or other masters in the platform, then it interprets the request and do the necessary actions. For example, if the CPU requests that the block A go to clock gated state, the PM send a request to the CM responsible of this block to turn off the clock generated to the block A.

The Power State Table (PST) concept is taken from UPF, which constitutes the interface between the power strategy and the power architecture. PST defines a static system power management strategy by defining the power domains’ supply nets states. Columns of a PST represent the states of power domains in terms of their power supply net states, while lines represent the different system power modes. Each line corresponds to one legal combination of specific power domain states.

PwClkARCH, similarly to PST, uses the concept of Clock State Table (ClkST), which defines for each block its possible frequency values during the execution of a scenario. The columns of the ClkST represent local power modes of the hardware blocks. A local power mode sets the clock state of each block by specifying its division factor. This factor is applied by the CM on the source clock of the CD in order to generate the clock needed for a given power mode to the corresponding block. When the division factor is equal to the specific value zero, the clock gating function is applied.

The Domain Power Controller (DPC) is in charge of applying the configuration of controls from a specific row in the PST to the power domains. PM and DPC are included in the PMU (Power Management Unit) which is connected to the SystemC-TLM system bus.

By specifying a line entry in the PST and a line entry in the ClkST, we can define the complete state of PDs and CDs. To facilitate the control of PDs and CDs, an OPP Table (OPPT) is available in PwClkARCH where an entry (an OPP) specifies the index entries is PST and ClkST and the multiplication/division factors of the DPLLs in the system (Figure 11). The PM receives from a master on the bus (e.g. the CPU) a transaction specifying the row index in the OPP Table to be applied to the power architecture. If after a change of row index, a state transition of a power domain (a clock domain resp.) is activated by DPC (CM resp.), the state transition is acknowledged to the PM after a time penalty, which models the time for the supply voltage (DPLL resp.) to reach its new value.

Power Management Unit (PMU) contains instances of the CM, PM, DPC,… as depicted in the Figure 12, it is the only SystemC-TLM power-oriented component that must be added in the transactional model.

The observer classes are used to structure automatically the power and clock intent with monitors to evaluate at different layers (DE, power switch, supply net, and CM layers) the power consumption.

A set of generic assertions are also defined in PwClkARCH for verification purpose. For example, an error is signaled if an IP block without any wake-up facility receives a functional transaction while the IP is in clock gated state or in power gated state.

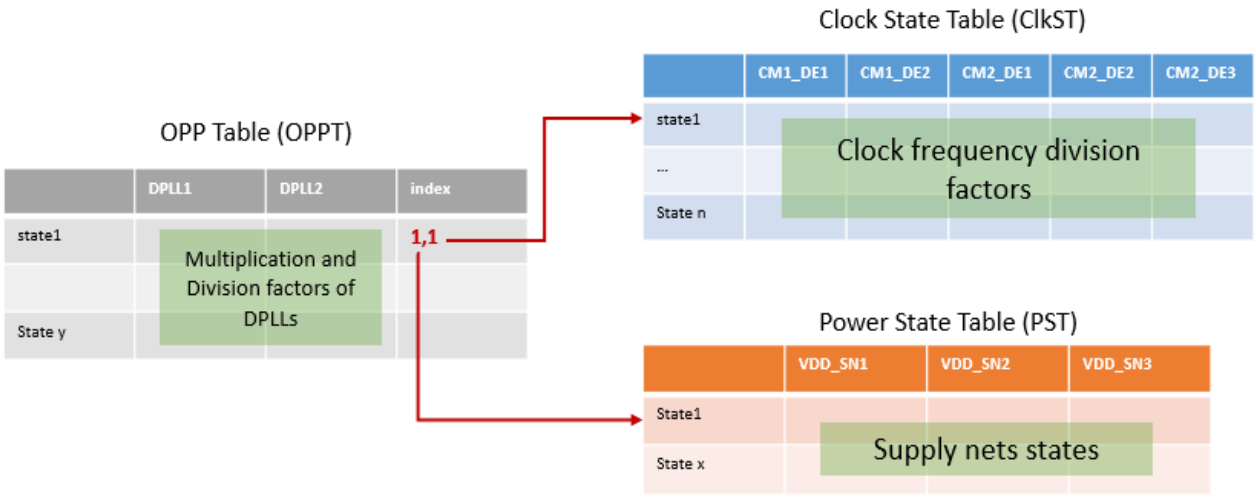


Figure 11. OPPT, ClkST, and PST tables structure

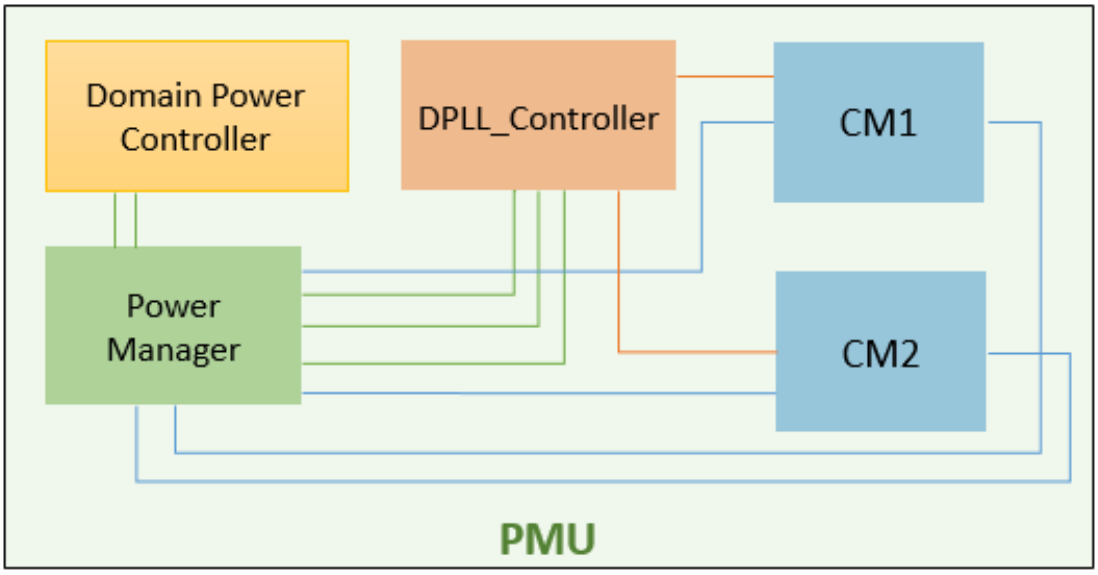


Figure 12. Power Management Unit (PMU) structure

4.2. Methodology behind PwClkARCH

To design high-performance and energy-efficient embedded systems, it is extremely important to address two basic issues. First issue concerns accurate estimation of power consumption of all system components during early design stages. Second issue consists in deriving power optimization solutions that do not negatively impact system performance.

PwClkARCH proposes a methodology based on the modeling of CDs and PDs that allows for clock and voltage distribution management in SystemC-TLM functional models, without any initial power considerations. The overall approach of this methodology consists of the steps shown in the Figure 13 and which will be explained later.

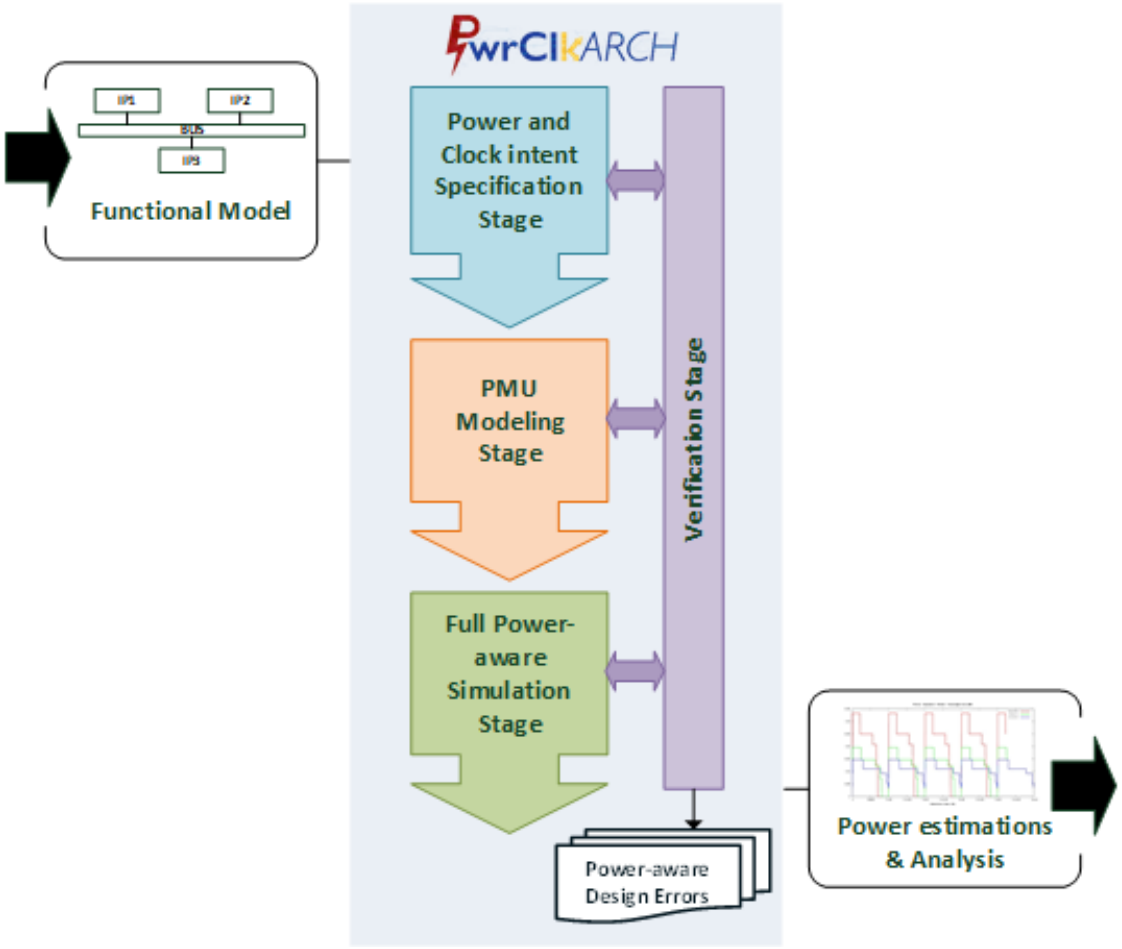


Figure 13. Overall approach of PwClkARCH based methodology

The first step is named Power and Clock Intent Specification Stage. This stage consists first in analyzing how data are exchanged between the components of the functional model to allow the designer to understand when and how often each component has been activated in the different use cases. Then, the designer splits the components in different power domains and clock domains. This is done by instantiating the appropriate objects from the PwClkARCH library: clock sources, generated clocks, power switches, and supply nets. Usually, a top-level power domain and a top-level clock domain that don't contain any logic elements are defined.

The second step is the modeling of the PMU. This unit acts as an interface between the functional model and the power model. It is a SystemC-TLM block connected to the functional model through the PM block' socket. This unit serves as a relay for the implementation of the power management strategy based on the exploitation of CDs and PDs states. This strategy makes it possible to control the power mode of a CD (PD resp.) adapted to the application scenario by adjusting the power modes of its components. Switching from

one power mode to another then corresponds to a power management request represented as a TLM transaction originating from an initiator in the SystemC-TLM system (for example, the CPU on which the embedded software executes).

At the third stage, the complete architecture including its functional model and its power management is simulated. Power consumption value updates are plotted in log files generated automatically during the simulation (Figure 14). These files are useful for analyzing and comparing different power management solutions (for example, different division/multiplication factor configurations in ClkST table) as well as for selecting the power management solution that offers a good compromise between system performance, power consumption and the complexity of the management system of this consumed power (number of CDs and PDs, size of the tables, diversity of frequencies considered for example). In order to verify that the steps of the methodology are completed successfully, a process of verification was added to the proposed system-level design flow.

This verification step is the fourth step of PwClkARCH methodology which is a perpendicular step. The purpose of this step is to check clock and voltage management properties during the simulation. Verification is based on the notion of contract. A contract is defined as a set of assumes (i.e. pre-conditions) and guarantee (i.e. post-conditions) clauses. To verify a contract in simulation, each clause is introduced in PwClkARCH as a specific assertion. This assertion is called at the level of the SystemC-TLM functional code in order to trigger an exception when a contract is violated (Figure 15). For example, a Clock Domain is valid if it contains only one DPLL and only one CM at most. Another example of contract is that during the simulation, there is no functional activity in a CD whose state is clock gated.

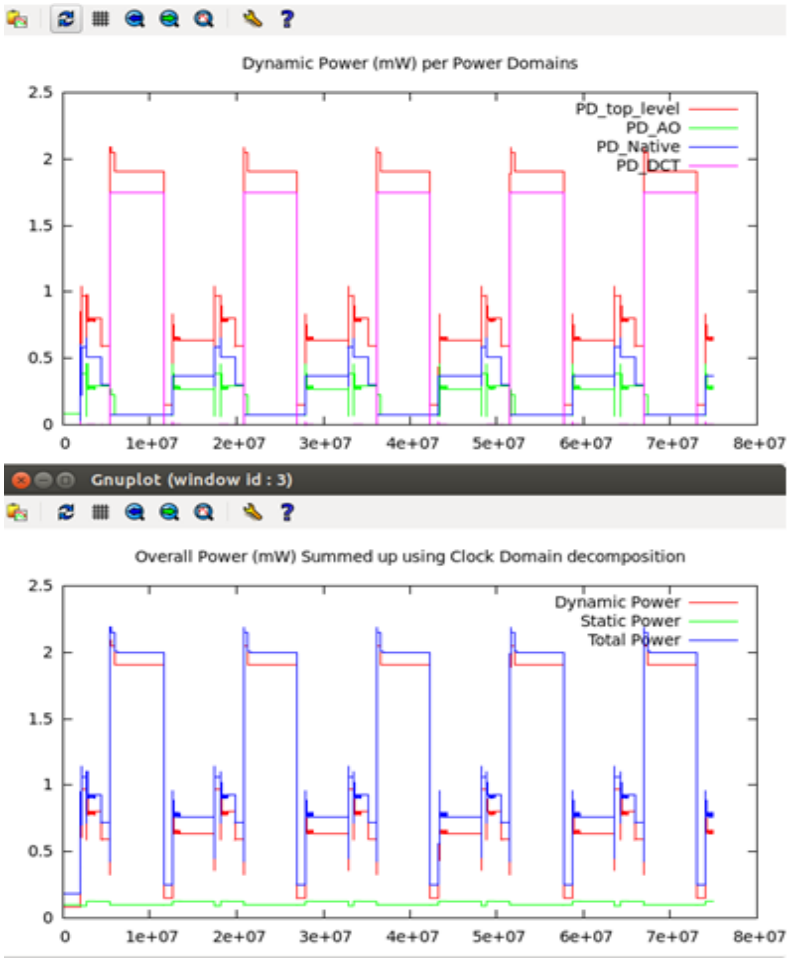


Figure 14. Power consumption curves examples

```
The Number of errors is 5
Specification Error at LINE : 55 at time 9254101 ns FILE : ../src/DE_Observer.cpp
Error label : Assume Assertion Violation : An activity is noticed in design element DCT while its clock is disabled !.
The Number of errors is 6
Specification Error at LINE : 55 at time 9258101 ns FILE : ../src/DE_Observer.cpp
Error label : Assume Assertion Violation : An activity is noticed in design element DCT while its clock is disabled !.
The Number of errors is 7
Specification Error at LINE : 55 at time 9262101 ns FILE : ../src/DE_Observer.cpp
Error label : Assume Assertion Violation : An activity is noticed in design element DCT while its clock is disabled !.
The Number of errors is 8
Specification Error at LINE : 55 at time 9266101 ns FILE : ../src/DE_Observer.cpp
Error label : Assume Assertion Violation : An activity is noticed in design element DCT while its clock is disabled !.
The Number of errors is 9
Specification Error at LINE : 55 at time 9270101 ns FILE : ../src/DE_Observer.cpp
Error label : Assume Assertion Violation : An activity is noticed in design element DCT while its clock is disabled !.
The Number of errors is 10
Specification Error at LINE : 55 at time 9274101 ns FILE : ../src/DE_Observer.cpp
Error label : Assume Assertion Violation : An activity is noticed in design element DCT while its clock is disabled !.
The Number of errors is 11
Specification Error at LINE : 55 at time 9278101 ns FILE : ../src/DE_Observer.cpp
Error label : Assume Assertion Violation : An activity is noticed in design element DCT while its clock is disabled !.
The Number of errors is 12
Specification Error at LINE : 55 at time 9282101 ns FILE : ../src/DE_Observer.cpp
```

Figure 15. Errors Report File example

5. Study of the memory system at ESL level

The design of the memory architecture is becoming more complicated due to the integration of many features in the chips as for example both multimedia applications and communication protocols (LTE) that are particularly memory intensive. Each application often requires different memory behaviors, for example, low latency memory for control-oriented applications, or high-speed memory for multimedia applications. A memory system cannot effectively satisfy at the same time these two requirements. It is then necessary to make compromises, and thus evaluate on representative use cases different memory configurations including cache levels that locally allow to partially absorb the memory transfer requests.

5.1. Memory and storage

In mobile systems, being developed for meeting specific functionality, the computing and memorization resources are necessarily limited. All mobile systems have limited on-chip memory including the caches. However, when running complex and large applications, the amount of memory needed increases, leading to external memory accesses, which is a major source of energy consumption in mobile systems. The reason for that is that external memory accesses incur additional energy dissipation in terms of voltage adaptation, increased memory access time and delay. So, during the development stage of an application, enough measures have to be taken into account to properly handle the memory capacity of the system.

In this section, we first provide necessary DRAM background. Dynamic Random Access Memory (DRAM) is composed of multiple banks arranged in a two-dimensional structure formed by rows (Word Lines) and columns (Bit Lines, as shown in Figure 16, where each cell stores a single bit of data in a capacitor. Each bank contains a row buffer used to cache the data in the most recently accessed row.

The access to DRAM is controlled by the DRAM protocol. It consists traditionally of six commands: activate (ACT), read (RD), write (WR), Precharge (PRE), refresh (REF) and no-operation (NOP) [38]. To start processing, the memory controller issues the ACT command to activate the appropriate row. Then, the controller sends RD and WR commands to the row buffer. Once the RD and WR operations to the row are complete, the controller issues a PRE command in order to prepare the array for commands to a different row. The REF command is one of the key issues w.r.t. power consumption in DRAM. In fact, the controller issues periodically REF commands to avoid loss of data. This makes DRAM consumes power even when not used.

The time execution of each command is very specific and differs from one technology to another. For example, t_{RCD} is the Active to column access delay. It is the time needed to accomplish the ACT command. The value of this parameter may be found in the datasheet of the memory device.

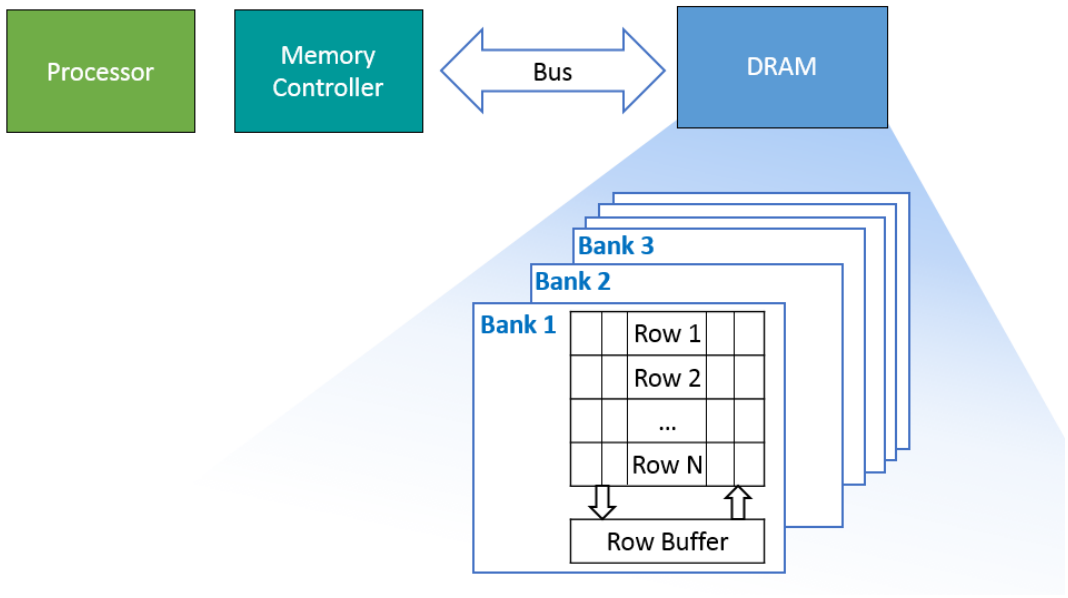


Figure 16. High-level overview of DRAM-based memory system

5.2. Memory requirements: Energy efficiency and performance

5.2.1. Memory requirements

Modern computer systems consist of many kinds of components and their characteristics is different from one another. One of the most important elements is memory. Flip flops are the most basic memory element and contained in almost all logic circuits. Memory modules are obviously common memory element. Usually, performance is the most important criteria for many computer systems. To achieve desired performance, high-speed volatile memory is widely used. However, the volatile memories lose their contents when power supply is cut off. Therefore, it is difficult to utilize low-power technologies such as power gating. Recently, new generation non-volatile memories (NVM) are emerged and have comparable performance to volatile memories (Figure 17). A combination of the NVM and the aggressive power managements present a promising opportunity. A major drawback of the NVMs is a write performance. To write information permanently, a longer latency and larger power are required compared with DRAM.

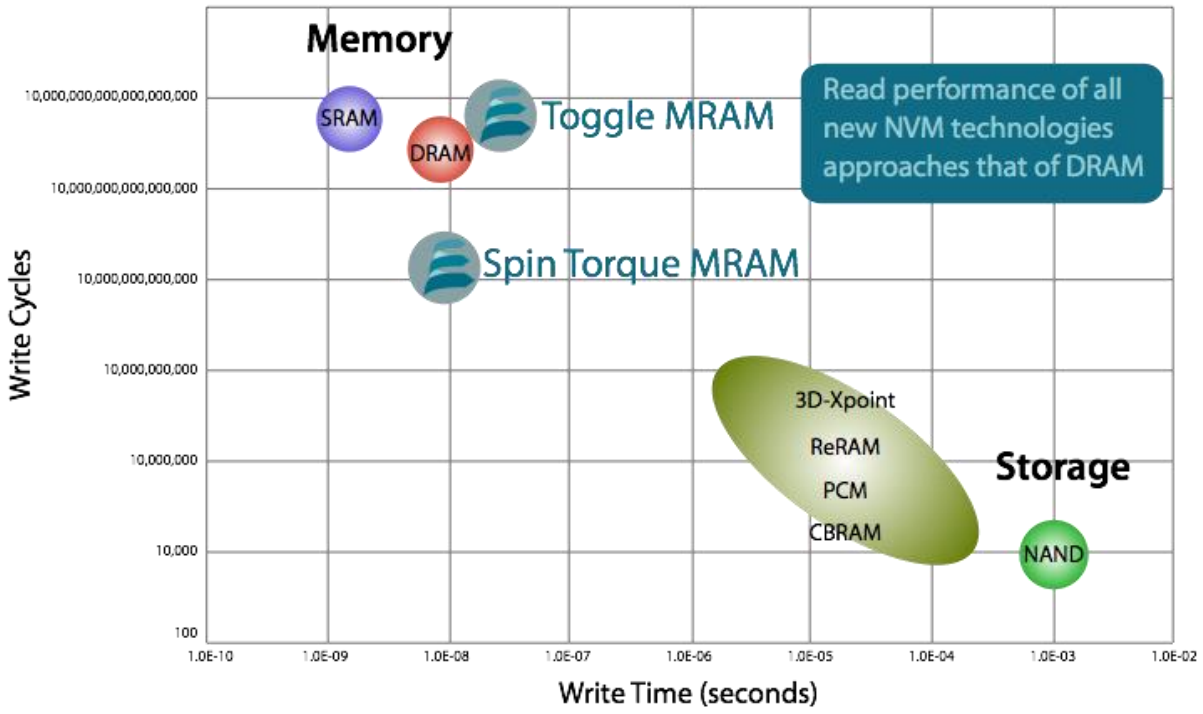


Figure 17. DRAM vs NVM write performance. Source: Everspin Technologies

At circuit level Low Power DRAM (LPDDR) was introduced to reduce power consumption in mobile embedded devices. They provide specific features such as Partial-Array Self-Refresh (PASR) and Deep Power-Down (DPD) which can be activated when the memory part is in idle state. In self-refresh mode, the interface circuitry of the memory controller is disabled and disconnected and refresh operation is done autonomously using an internal counter. The PASR feature enables the controllers to select the amount of memory that will be refreshed during the self-refresh. In DPD mode the power of the memory array is cut allowing to decrease its leakage current. If applications do not require data retention, the DPD mode can be applied. Data in the memory is not retained after the device enters DPD mode.

The problem of optimizing a memory system is widely discussed in the literature. In particular, regarding the higher levels of the memory hierarchy, a multitude of publications exist mainly in the single-core and multi-core contexts of the CMP (Chip MultiProcessor) type. Works like [39] aim to optimize the performance of a memory system by introducing specific "writeback scheduling" mechanisms between the last cache level and the DRAM. This type of approach is of course not to be neglected, but rather we will focus on the problem of reducing interference related to the concurrent execution of applications sharing memory resources. Thus in [40] the focus is on the shared resources in a CMP architecture (last level of cache, shared bus, memory controller) by a set of threads whose scheduling must be aware of the problems of interference. This interference leads to performance degradation that can be high. Thus, degradation by

a factor 10 on DRAM access latency due to interference at the memory controller are observed in [41]. To limit the interference, many works seek to include in each higher level of decision of the memory hierarchy mechanisms that avoid as much as possible the degradation of performance.

The scheduling of access requests to the DRAM is thus a very studied technique to optimize the QoS (Quality of Service). A first widely used technique is the First-Ready-First-Come-First-Serve (FR-FCFS) [42]whose objective is to associate the highest priority to queries that target data in the row-buffer. associated with the DRAM banks, even in relation to older queries. This type of approach gives good results with a single-core, single-thread architecture, but induces significant performance degradations for CMP architectures [43]. Thus, many works have been developed to reduce performance degradations by thread and by core to obtain a fair quality of service. For example, in [44] the scheduler dynamically estimates performance degradation on each thread to be considered for the allocation of priorities to the requests. More recently in [45] is presented an algorithm able to differentiate the requests transmitted by a CPU from those resulting from a GPU, having very different access schemes, in order to distribute performance degradations. Thus, the correct estimate of the interference between applications/threads or performance degradation on a thread or an application induced by the management performed in a memory controller is a significant problem. For example, in [46] and [47] two techniques are proposed for dynamically collecting performance degradations per application and then controlling the speed at which each core emits requests to the memory system. In [47], the memory controller is either capable of associating higher QoS with certain applications than others or minimizing the maximum performance degradation on all applications running in parallel. If the technique proposed in [44] involves inserting substantial hardware resources in the memory controller to dynamically define priorities, a less costly approach to resources is proposed in [48]. This one divides into two categories the applications, those that cause interference and those that are sensitive to interference. Higher priority is associated with the former.

The above techniques require adapting the controllers to include the proposed management mechanisms. Other approaches propose to regulate the emission of the requests to act on the interference rates. This is the case of [46] introduced above but also works presented in [49] which address a mesh manycore architecture where the tasks are assigned and sequenced to the cores according to the pressure they induce on interconnection and on memory.

The design of the memory system architecture is also a vast subject considered in the literature comprising the steps of partitioning, data allocation and their scheduling, these steps being related and sometimes even antagonistic, this subject is also complex.

As a result, work on memory partitioning reveals substantial gains in surface area and consumption under certain conditions. But it is necessary to carry out explorations of memory architectures to find architectural compromises. Indeed, the literature deals with techniques such as banking memory, or hybrid memory architectures.

Let us mention the work of [50] describing the general design of hierarchical memory architectures with its partitioning into memory banks. Their work helps to determine their number and size. For this, they rely on simulations allowing them to extract data such as the most frequently accessed data, access numbers to each part of memories, etc. But obtaining this information by simulation takes a lot of time especially if the simulations are fine grain (e.g. cycle accurate). To overcome these constraints, they propose to estimate an approximation of such numbers thanks to compilation methods, for example by analyzing the source code making detailed studies of the loops to deduce the number of accesses, and finally, this step allows to estimate the size of the memories and the number of banks required.

But this approach seems constraining, because either the application code must be executed in detail, and it is not always available, or this type of fine code analysis may not be effective because it only takes into account features from a compilation analysis on a per-thread basis, without taking into account the entire system, which is increasingly competitive. An example of a dynamic partitioning approach performed at runtime is proposed in [51] where the demand in number of banks by each thread is estimated to make then allocation and partitioning. If performance gains are obtained in simulation, the overhead induced by this type of approach is not considered knowing that it only concerns homogeneous architectures.

Other types of work are particularly interested in generating hybrid memory architectures hoping to take advantage of each type of technology. Let us mention the work of [52] which mentions the interest of non-volatile memories (NVM) with many attractive points, such as its low cost in leakage current, and its high density, which should make it possible to reduce the energy consumption of the system. But the major drawbacks of this type of technology are the high cost of writing in a NVM that increases the dynamic power during this type of operation, as well as latencies that become significant. The authors also note that the reliability of an NVM is less important compared to that of SRAM. Since these drawbacks are mainly due to write operations, the data allocation must be resolved by reducing the number of write operations in the NVMs, and under such conditions, their use could be interesting. The work proposed in [52] then discusses the development of a complex allocation algorithm. It is necessary to obtain a rough initial allocation for which the different costs will be calculated, and from these data, the solution is refined by iterations comparing the different allocation costs until finding an optimal solution or an acceptable compromise.

These allocation strategies are very complex and require exploration tools. Thus, the works published in [53] and [54] begin by calculating the induced energy and the access time of the various possible allocations, then the allocation choices are made with respect to the energy criterion. In the first work [53], the optimization of the allocation of each block of data (determined before) is carried out. For the second, an implementation of a hardware module in the MMU (Memory Management Unit) that is able to migrate data and copy them from one place to another targets performance criterion.

Turning now to the modeling and simulation stages of non-functional constraints such as dissipated energy, we can notice that some applications, such as those of the modem, lead to lower workload times at the processor level, which can then be put in rest mode. For this, the domains that can be powered by controllable voltage sources, i.e. power domains, must be represented. This point cannot be neglected, as it is actually present in the platforms. The consequences on memory management are also important, since the retention modes, for example, must be able to be modeled.

Thus, the joint work of STMicroelectronics, the Grenoble laboratory Verimag and Doceapower formerly [55] are questioning a high-level model to obtain both performance and power consumed. To do so, they opt for a co-simulation platform allowing, in their case, to simulate the platform modeled in SystemC/ TLM with a consumption estimation tool so that power information can be directly considered in the architectural choices that will be made. Indeed, they start from the observation that usually the power estimation is done after behavioral simulations, the traces are then recorded in a VCD (Value Change Dump) type file, and finally, analyzed offline. But this implies having already benchmarks to execute. Since they are intended to do an architecture exploration in terms of power management, they do not annotate SystemC code directly with numeric values, but instead, they keep the numerical values in a separate model. They have extended the SystemC modules with a set of power parameters that can be of several different types (voltage, frequency, switching activity ...). The value of each of these parameters is set to an adjusted value by the functional model at given time points. These other works highlight the lack of exploitation of the possibilities of the Virtual Prototyping [56] (project in which are involved the universities of Concordia, Costa Rica and California, the companies Qualcomm, Microsoft and Samsung) which usually serves mainly to make functional validation of software during its development. They wish to develop the architectural design possibilities that this type of method allows. For this, they develop a framework which provides an automatic generation of SystemC-TLM models from a graphical modeling. Their Embedded System Environment (ESE) tool also provides an automatic TLM synthesis at the CAM (Cycle Accurate Model) level which consists of RTL interfaces, system software and files ready to prototype the FPGA. The input application is captured as a set of communicating processes concurrently. These processes are, at the sheet level, symbolic representations of functions that can be specified using programming languages such as C

or C ++. They communicate with each other and with memories using well-defined ports. The big benefit of this tool is to be able to make changes and check in a few seconds the impact in terms of performance. The platform can be refined very easily and quickly. However, the presented platform remains very simplistic and far from our concerns, integrating neither the aspects of voltage and power domains nor the aspects of memory access.

5.2.2. Some existing DRAM power models

There is a number of power and performance simulation models for DRAM in the literature. We list some of them which are the most know and used.

CACTI [57] is a cycle accurate power model that enables the modeling of the complete memory hierarchy (SRAM L1 to DRAM). Many works are done either using CACTI as it is or to extend existing versions and enhance them with to support LPDRAM technology for example (CACTI 5.0), or CACTI-P that includes an in-depth model for leakage power management technologies. CACTI primarily focuses on interconnect design for large caches.

DRAMSim2 [58] is a cycle accurate model of a DRAM system including the memory controller and the buses by which they communicate, developed by University of Maryland. It is a C++ model of DDR2/3 memory system, where the timing parameters and specifications taken from manufacturer datasheets are included in a configuration file. DRAMSim2 uses the power model of Micron to compute power consumption given the state transitions of each bank.

DRAMPower [5] is a modeling tool to estimate power and energy for DDRs, LPDDRs and Wide-IO DRAM memories based on JEDEC standards. It basically provides examples of configuration files of memory, timing, and specifications from Micron Technology datasheets. We choose to use DRAMPower tool in our work for many reasons which are explained in Chapter V.

Ramulator [59] presents a performance model simulator for DRAM technology. For some standards, it can provide power consumption by relying on DRAMPower, but it doesn't provide a power model itself.

5.3. Technology trends: Emerging Non-volatile memory

DRAM has been so far the preferred technology for implementing main memory, but challenges with DRAM scaling, like the high cost of refreshes [60], are increasing. To overcome those challenges, some emerging NVM technologies, such as phase change memory (PCM), resistive RAM (RRAM) or spin-transfer torque magnetic memory (STT-MRAM), appear more scalable and their performances are much closer to DRAM than flash memory. Similarly, State-of-the-art showed that NVM technologies read access time is similar to its SRAM equivalent [61] [62]. Emerging NVM memory technologies are explored as potential alternatives

to traditional SRAM/DRAM-based memory architecture. In our work, the focus is on DRAM-based memory architecture.

The first STT-MRAM products from Everspin Technologies, which use Double Data Rate DDR3 interface with some modifications needed to tune timing parameters [63], show that STT_MRAM has matured to a point that it can soon replace traditional DRAM. In [64] authors introduce an optimized MRAM interface, but it is an adaptation of an existing Low Power DDR LPDDR3 specification, originally designed for DRAM, not a dedicated STT-MRAM specific memory controller, therefore providing a sub-optimal solution. Authors in [65] have shown that STT-MRAM main memory provide comparable performance to DRAM using DRAMSim2 simulator. Figure 18 shows an approximate timeline of the DRAM and STT-MRAM capacity growth. The gap between the two technologies is decreasing over the years.

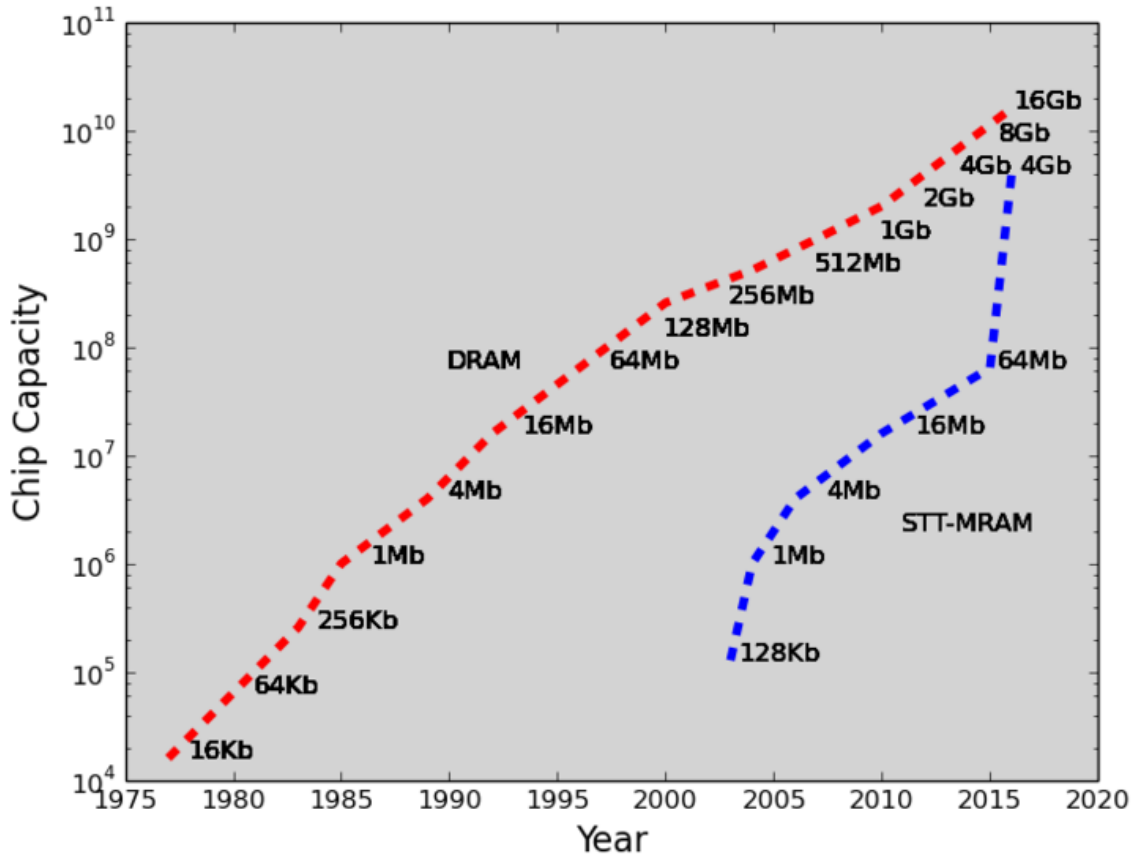


Figure 18. DRAM and STT-MRAM capacity growth in years [65]

The most known MRAM technology is the spin-transfer torque devices (STT-RAM) either with in-plane magnetization or perpendicular-to-the-plane magnetization (Figure 19). MRAM doesn't use electrical

charges, it uses magnetic tunnel junctions (MTJs) to store the data. Thus the storage element is the MTJ. The information is stored in the orientation of the magnetization of the storage layer. Each MTJ consists of two ferromagnetic layers: a free layer (storage layer) with switchable magnetization direction and a fixed layer (pinned layer, reference layer) with fixed direction (Figure 20).

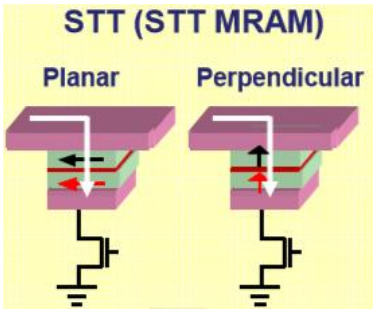


Figure 19. STT MRAM cell structure (spintec inMRAM 2015 by Dieny.B)

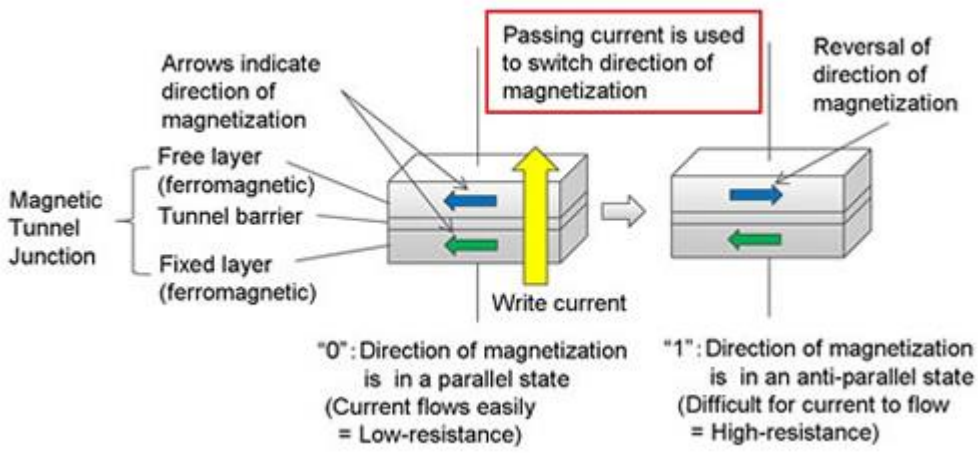


Figure 20. Principle of spin-torque-transfer STT-MRAM [66]

	Cell size (F^2)	Write speed	Power consumption	Endurance
STT-MRAM	6 to 12	**10ns**	Medium	10^{15}
PCM	6 to 12	**75ns**	**Medium**	10^{8}
RRAM	6 to 12	**20ns**	**Low**	10^{10}
SRAM	**100-150**	**2ns**	**Low**	10^{18}
Flash	4	**10µs**	**Very high**	10^{5}

Figure 21. Emerging memories performances (Y. De Charantenay, (Yole) InMRAM 2015)

NVSIM is a simulation tool dedicated to NVM technologies [67]. In the style of CACTI, the NVSIM simulator describes at circuit level various NVM, including STT-RAM, PCRAM, ReRAM, and NAND Flash. The capabilities of this tool are its ability to estimate the internal access time, the access energy, the silicon area of the NVM and to explore internal memory architectures.

6. Conclusion

This chapter allows us to present all the concepts that relate to the context of our work. The state-of-the-art of modeling and verification at ESL for both performance and power management is briefly summarized in this chapter. We introduce also the PwClkARCH library that represents a key basis of our work. We finish the chapter by giving a concise study of the memory system at ESL level. In the context of this project between the LEAT laboratory and Intel, the goal is to join the tools of both parties and therefore the choice of technologies used is almost imposed by the project itself.

Next, we introduce the Intel pre-silicon simulation environment used in our work and we detail the methodology that we propose in this project to provide a framework able to simulate performance and power consumption of a given mobile platform, with a detailed memory model, at the same time.

III. Intel pre-silicon simulation environment

SoC designers face many challenges to improve at the same time performance and energy efficiency, due to the continuous increase of the architecture complexity. Designers have the opportunity to use ESL tools and virtual prototyping to face this complexity in the early step of the system design.

Power consumption and performance are adversely affected by supply voltage and clock frequency. This potential trade-off cannot be studied separately. Our work enhances an existing industrial performance model developed by Intel with the introduction of the PwClkARCH library (Section II.4), which allows a combined early power and performance analysis.

Intel was among the first adopters in the industry of SystemC-TLM-based VP (one of the main contributors of OSCI TLM standard) to design, optimize and verify its products. As SystemC models are reusable, the architecture may be composed from different IP models coming from different owners/teams. During this project, we work on the L2 Coprocessor part of a LTE modem and we add the features needed to estimate its power consumption and to manage the power dissipation. In this chapter, we present the two Intel simulation environments used to validate the proposed methodology.

1. Pre-silicon simulation environment for performance assessment

For our methodology validation, we use an Intel proprietary pre-silicon simulation environment, used for cellular modems performance assessment and analysis. On the one hand, this model does not include any power consideration, neither power estimation features, nor power management techniques. But on the other hand, a lot of performance metrics can be evaluated to asses overall and specific IPs performance.

During the thesis, we worked on two generations of modem and so we used two different simulation environments. The first one was in the Intel Sophia Antipolis France site, and the second one in Intel Munich site. For obvious reasons of safety of the industrial works of Intel, we are not able to detail hugely these architectures that we used during the project. We use abstract models instead to explain our contributions.

1.1. Intel Sophia Antipolis pre-silicon simulation environment

We did our first manipulation on a platform level pre-silicon simulation environment used for architecture performance exploration and validation in Intel Sophia Antipolis. The goal of this environment is to assess whether IPs are properly sized and have sufficient access to external resources, in order to reach the required performance needed to achieve their task on-time and efficiently. The efficiency reflects only performance metrics like internal buffering, parallelism management, memory bandwidth and latency. Thus, the use cases used are high data rate scenarios capable of reaching up to 1 Gbps data throughput. The hardware blocks run with the maximum frequency allowed without taking into account the energy consumption. Our

contribution is to add power consideration to this model, and to minimize power dissipation. Our second goal is to validate the available off-chip memory or to consider a new technology, based on the power indicator and not only on the performance indicator.

Figure 22 shows an abstract architecture of a cellular modem, which includes several IPs like CPU, on-chip and off-chip memories, interconnect and other functional blocks. Among the latter, we focus on the L2 Coprocessor block, which is composed of hardware accelerators needed for the L2 data processing, which includes a set of hardware components that we call Tiles.

The model includes a stochastic model of the processor core to de-correlate its behavior from software availability. Thus, without requiring a software model we run use cases using platform level task scheduler. Hence, the SoC activity is driven by a set of independent task flow. Therefore, it is important that the behavior (traffic patterns and processing duration) of the different IP models are matching accurately what can be observed on real silicon SoC and that the information fed to the task graph like the instruction count, load/instruction ratio, data miss, and address are accurate and correct. All information included in the task graph are collected during the SoC profiling phase on the previous generation silicon. So, the main goal of the model is to produce an accurate traffic pattern, which is necessary for the analysis of the performance of the concurrent execution of the different IPs. This simulation framework fits quite well our need to explore, in a first stage, the impact of various task scheduling strategies on Tiles as well as various power management strategies onto the memory system performance.

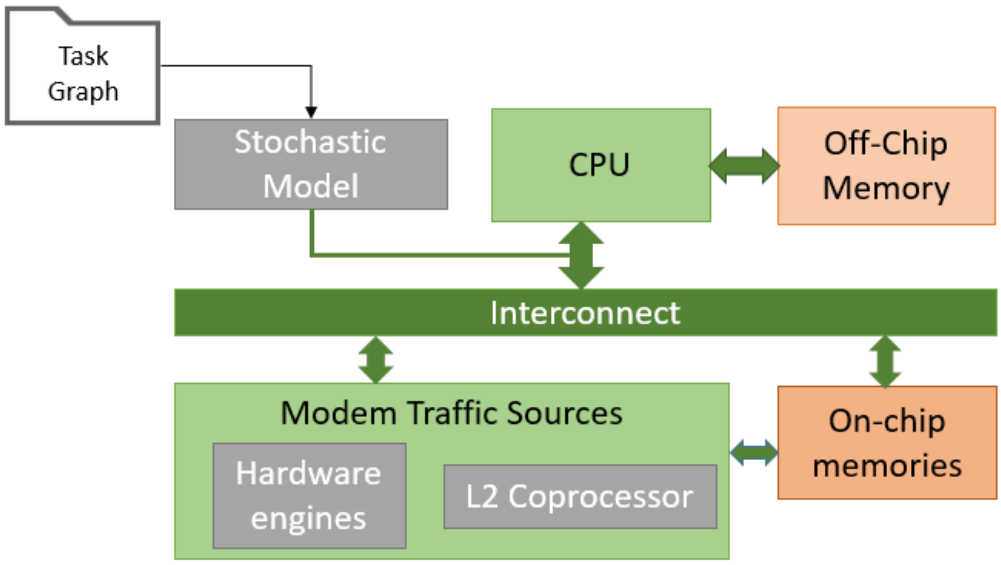


Figure 22. Simplified representation of a cellular modem performance model

The platform model activity is driven by a task scheduler fed by a task graph describing the various processing flows involved in the targeted use case. Figure 23 shows a simplified diagram that maps software tasks to the platform blocks. A single task scheduler is in charge of activating the execution of software tasks on any hardware block, including cores. Thus, the scheduling approach is centralized and this scheduler has the information of all the tasks (Active, Preempted, Pending or Completed). The use case considered for performance analysis are high data rate cellular modem LTE (Long-Term Evolution) scenario capable of reaching up to 1 Gbps data throughput. The analysis focuses on the SoC performance during the worst case concurrent traffic scenario, where interconnection fabric and memory are the main blocks that may cause performance bottleneck.

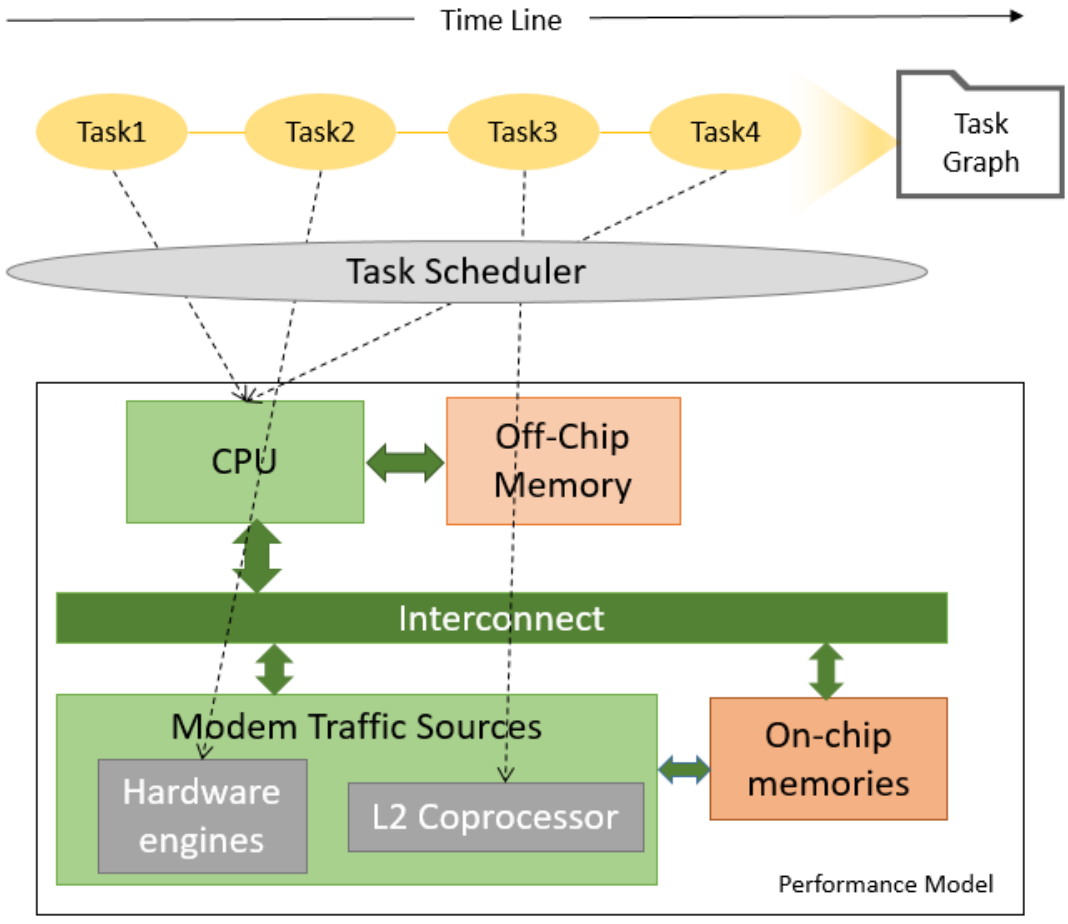


Figure 23. Platform level blocks mapped into simulation tasks

Yet, in order to explore power and performance tradeoffs, it is essential to also model the impact of the power management strategy into the various performance models and to be able to collect power dissipation information out of each key IP model. As described in Section II.4, the PwClkARCH library allows the implementation of a power management strategy on top of a functional SystemC-TLM model, described

within a VP environment. Therefore, our aim is to add a power model using the library to the performance environment introduced above (section 3). By enriching this new environment with the necessary features to study the memory system parameters and behaviors, we provide a new methodology to explore different memory systems, where power and performance are primary constraints.

After 15 months of work based on Intel Sophia Antipolis platform, for reasons that didn't depend on us, we had to change the architecture on which we work and so we moved to a new version of the modem platform.

1.2. Intel Munich pre-silicon simulation environment

The second architecture model we used in our work (Figure 24) represents the next generation model of the previous architecture. The differences are mainly in terms of the implementation environment, the technologies used for each block or other added improvements that cannot be detailed and do not bother our work. The architecture is composed of components from Intel's System Architecture and Exploration SAVE library. This library includes some generic models for processing elements, interconnects and memory subsystems, as well as some specific application models, for macro and micro architecture exploration. These models are implemented using Intel SystemC TLM libraries which are compatible with the standard SystemC-TLM but have more facilities for users. This development environment (SAVE) allows us to simulate architecture models to evaluate their performance.

The task scheduling engine interface is modeled as s SystemC module which is used by all the processing elements of the architecture like the Tiles (T0 to T6). This interface schedules tasks according to their priorities. As many tasks may be running in parallel, and may be mapped on the same processing element, the decision of which task to schedule is depending on given priorities. Configuration files are used to specify the values of the different parameters based on Intel SystemC TLM Attributes. In this architecture, the scheduling is distributed. Each hardware engine has its own scheduling interface which schedule the tasks to be run. The new task graph methodology used in this version provides a scheduling layer on top of each processing element in the architecture model, for scheduling the execution of different parallel tasks mapped onto the same processing element. Thus, it is a distributed scheduling approach and not a centralized one as in the previous architecture. This is very important for us since it impact the manner how to collect and send the engine status information (Idle, Active, Preempted or pending) to the PM, which will be explained at the end of section 3.

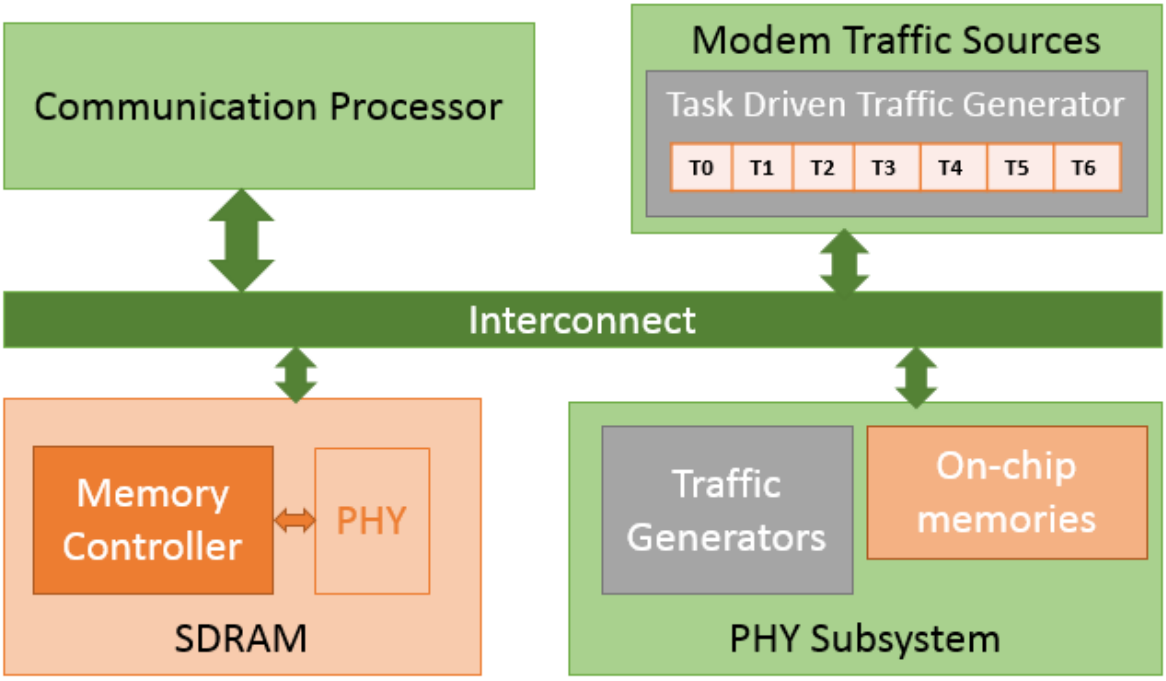


Figure 24. Simplified representation of a cellular modem performance model 2

2. Task Graph

The two previous modem architectures use a task graph to model the modem software application part. Even if differences exist between the task graphs of those two architectures, they describe similar concepts. At system level, on the one side the hardware architecture and the software application models are usually not mature. The architecture resources cannot execute real software applications. On the other side, the simulation results must be accurate enough so that it is possible to make reliable early design decisions. Thus, the application is modeled as a task graph (example Figure 25) where each node represents an abstract application task, and the edges represent dependencies and communication channels between the tasks. Following this approach, it is possible to obtain rapid performance evaluation.

The tasks can be mapped on different resources in the architecture through a configuration file using Intel SystemC Attributes Objects (Figure 26). Each node has a parameter that refer to a resource in the architecture. This file includes other parameters of a task node such as start or release time, deadline, period, priority, successor and predecessor tasks, etc (see Figure 27). A deadline of a task should be smaller or equal to the time interval specified for the transfer. If the transfer is completed before the deadline expires, we say that the deadline is met and the headroom is calculated. If the transfer is completed after the deadline expires, we say that the deadline is missed and a negative headroom is calculated (an example of output log is given in Figure 28). During the simulation, when a task is trigged, it requests starting processing on its mapped

architecture resource and, upon completion, triggers its successor tasks. The behavior of each task is implemented as a finite state machine with four states (IDLE, READY, RUNNING, PREEMTED). The resources only simulate the performance workload of the application tasks without executing real behavior or algorithms of the software tasks. In fact, Intel task graph methodology aims to provide reliable system level performance estimates of a multi-core embedded system architecture for an abstract use case or application. It does not require detailed implementation of the application.

For example, in Figure 25, we have a task graph used on the architecture 2 that includes ten tasks which are mapped on Tiles represented as T0 to T6. The dashed arrows represent the mapping between the task and the hardware block. The green arrow indicates that the task is periodic and therefore it triggers itself. The set of the green and the blue arrows constitute what is called the edges of the task graph.

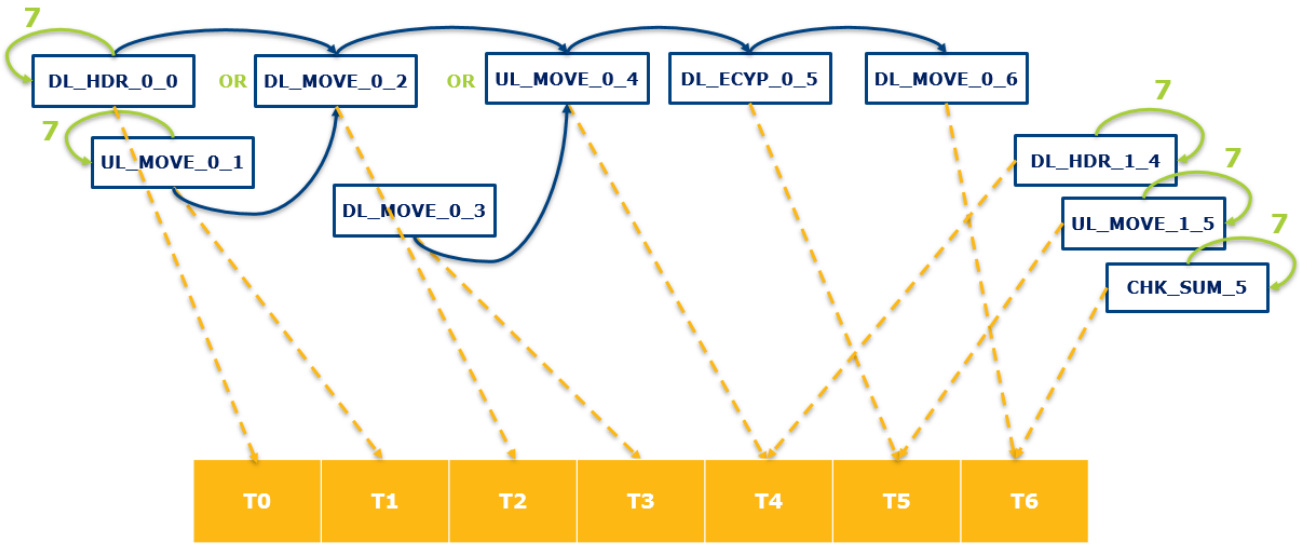


Figure 25. Example of a task graph for communication application

```
top.TG_Constructor.DL_HDR_0_0.task_name          = DL_HDR_0_0
top.TG_Constructor.DL_HDR_0_0.engine_name        = top.DPU0.tile_0
top.TG_Constructor.DL_HDR_0_0.priority           = 0
top.TG_Constructor.DL_HDR_0_0.period             = 1000 us
top.TG_Constructor.DL_HDR_0_0.start_time         = 0 us
top.TG_Constructor.DL_HDR_0_0.deadline           = 223 us
top.TG_Constructor.DL_HDR_0_0.num_itr_trig       = 1
top.TG_Constructor.DL_HDR_0_0.tot_num_itr        = 7
top.TG_Constructor.DL_HDR_0_0.in_trig_mode       = AND
top.TG_Constructor.DL_HDR_0_0.out_trig_mode      = OR
top.TG_Constructor.DL_HDR_0_0.suc_tasks          = DL_MOVE_0_2
top.TG_Constructor.DL_HDR_0_0.tparam_obj         =
top.TG_Constructor.DL_HDR_0_0.dbgLevel           = NOTE
```

Figure 26. Specification of the task DL_HDR_0_0 within a configuration file

Parameter Name	Data Type	Default Value	Description
task_name	string	"?"	Name of the task
state	enum	IDLE	States of the task, *IDLE, PENDING, ACTIVE, PREEMPTED, COMPLETED*
task_id	integer	0	Unique ID of the task (defined during instantiation of the corresponding task-graph node)
engine_name	string	"?"	Name of the engine which this task is mapped on
engineif	reference	NULL	A pointer to the task scheduler interface of the mapped engine
start_time	sc_time	0 ns	Task's release time
period	sc_time	0 ns	Task period (only valid for periodic tasks)
tot_num_itr	integer	0	Total number of iterations (only valid for periodic tasks)
trig_after_itr	integer	0	Total number of executions or iterations of a task before triggering its successor tasks
deadline	sc_time	0 ns	Deadline of the task
priority	integer	0	Priority of the task
workload_obj	string	"_"	Name of the execution workload of the task
workload_obj_ptr	reference	NULL	Pointer to the workload_obj
sel_prob	double	1.0	Execution probability of the task during simulation
suc_tasks	string	"_"	Name of the successor tasks
oTrig_mode	enum	OR	Output-ports triggering mode of the task
iTrig_mode	enum	AND	Input-ports reading mode of the task

Figure 27. General parameters of a task-graph node

```
NOTE: Variable VP_CONFIG_FILE not set. Skipping dynamic configuration.
SC_SOC_MODEL.EMAC_TILE_6 ------------- Processed TTI  # 1   in 243259     ns
SC_SOC_MODEL.EMAC_TILE_6 TTTTTTTTTTTTT Deadline 750 us met        by 506741     ns
SC_SOC_MODEL.EMAC_TILE_6 TTTTTTTTTTTTT Tile headroom          67.5655    %
SC_SOC_MODEL.EMAC_TILE_7 ------------- Processed TTI  # 1   in 242711     ns
SC_SOC_MODEL.EMAC_TILE_7 TTTTTTTTTTTTT Deadline 750 us met        by 507289     ns
SC_SOC_MODEL.EMAC_TILE_7 TTTTTTTTTTTTT Tile headroom          67.6385    %
SC_SOC_MODEL.EMAC_TILE_8 ------------- Processed TTI  # 1   in 633307     ns
SC_SOC_MODEL.EMAC_TILE_8 TTTTTTTTTTTTT Deadline 900 us met        by 266693     ns
SC_SOC_MODEL.EMAC_TILE_8 TTTTTTTTTTTTT Tile headroom          29.6326    %
SC_SOC_MODEL.EMAC_TILE_4 ------------- Processed TTI  # 1   in 759819     ns
SC_SOC_MODEL.EMAC_TILE_4 TTTTTTTTTTTTT Deadline 900 us met        by 140181     ns
SC_SOC_MODEL.EMAC_TILE_4 TTTTTTTTTTTTT Tile headroom          15.5757    %
SC_SOC_MODEL.EMAC_TILE_5 ------------- Processed TTI  # 1   in 759459     ns
SC_SOC_MODEL.EMAC_TILE_5 TTTTTTTTTTTTT Deadline 900 us met        by 140541     ns
SC_SOC_MODEL.EMAC_TILE_5 TTTTTTTTTTTTT Tile headroom          15.6157    %
SC_SOC_MODEL.EMAC_TILE_1 ------------- Processed TTI  # 1   in 802870     ns
SC_SOC_MODEL.EMAC_TILE_1 TTTTTTTTTTTTT Deadline 750 us missed     by 52870      ns
SC_SOC_MODEL.EMAC_TILE_1 TTTTTTTTTTTTT Tile headroom          -7.04933   %
SC_SOC_MODEL.EMAC_TILE_2 ------------- Processed TTI  # 1   in 800091     ns
SC_SOC_MODEL.EMAC_TILE_2 TTTTTTTTTTTTT Deadline 900 us met        by 99909      ns
SC_SOC_MODEL.EMAC_TILE_2 TTTTTTTTTTTTT Tile headroom          11.101     %
SC_SOC_MODEL.EMAC_TILE_3 ------------- Processed TTI  # 1   in 802207     ns
SC_SOC_MODEL.EMAC_TILE_3 TTTTTTTTTTTTT Deadline 900 us met        by 97793      ns
SC_SOC_MODEL.EMAC_TILE_3 TTTTTTTTTTTTT Tile headroom          10.8659    %
SC_SOC_MODEL.EMAC_TILE_6 ------------- Processed TTI  # 2   in 248135     ns
SC_SOC_MODEL.EMAC_TILE_6 TTTTTTTTTTTTT Deadline 750 us met        by 501865     ns
SC_SOC_MODEL.EMAC_TILE_6 TTTTTTTTTTTTT Tile headroom          66.9153    %
SC_SOC_MODEL.EMAC_TILE_7 ------------- Processed TTI  # 2   in 248535     ns
SC_SOC_MODEL.EMAC_TILE_7 TTTTTTTTTTTTT Deadline 750 us met        by 501465     ns
SC_SOC_MODEL.EMAC_TILE_7 TTTTTTTTTTTTT Tile headroom          66.862     %
SC_SOC_MODEL.EMAC_TILE_8 ------------- Processed TTI  # 2   in 636343     ns
SC_SOC_MODEL.EMAC_TILE_8 TTTTTTTTTTTTT Deadline 900 us met        by 263657     ns
SC_SOC_MODEL.EMAC_TILE_8 TTTTTTTTTTTTT Tile headroom          29.2952    %
SC_SOC_MODEL.EMAC_TILE_4 ------------- Processed TTI  # 2   in 760555     ns
SC_SOC_MODEL.EMAC_TILE_4 TTTTTTTTTTTTT Deadline 900 us met        by 139445     ns
SC_SOC_MODEL.EMAC_TILE_4 TTTTTTTTTTTTT Tile headroom          15.4939    %
SC_SOC_MODEL.EMAC_TILE_5 ------------- Processed TTI  # 2   in 760171     ns
SC_SOC_MODEL.EMAC_TILE_5 TTTTTTTTTTTTT Deadline 900 us met        by 139829     ns
SC_SOC_MODEL.EMAC_TILE_5 TTTTTTTTTTTTT Tile headroom          15.5366    %
SC_SOC_MODEL.EMAC_TILE_1 ------------- Processed TTI  # 2   in 804699     ns
SC_SOC_MODEL.EMAC_TILE_1 TTTTTTTTTTTTT Deadline 750 us missed     by 54699      ns
SC_SOC_MODEL.EMAC_TILE_1 TTTTTTTTTTTTT Tile headroom          -7.2932    %
SC_SOC_MODEL.EMAC_TILE_2 ------------- Processed TTI  # 2   in 800695     ns
SC_SOC_MODEL.EMAC_TILE_2 TTTTTTTTTTTTT Deadline 900 us met        by 99305      ns
SC_SOC_MODEL.EMAC_TILE_2 TTTTTTTTTTTTT Tile headroom          11.0339    %
SC_SOC_MODEL.EMAC_TILE_3 ------------- Processed TTI  # 2   in 801631     ns
SC_SOC_MODEL.EMAC_TILE_3 TTTTTTTTTTTTT Deadline 900 us met        by 98369      ns
SC_SOC_MODEL.EMAC_TILE_3 TTTTTTTTTTTTT Tile headroom          10.9299    %
SC_SOC_MODEL.EMAC_TILE_6 ------------- Processed TTI  # 3   in 244223     ns
SC_SOC_MODEL.EMAC_TILE_6 TTTTTTTTTTTTT Deadline 750 us met        by 505777     ns
SC_SOC_MODEL.EMAC_TILE_6 TTTTTTTTTTTTT Tile headroom          67.4369    %
```

Figure 28. Example of output log for deadline examination

3. Adding Power Intent to the simulation environment

The power model used at Intel for those modem platforms is based on a static power model framework based on spreadsheet calculations consisting of calculation algorithms and use cases. An Intel power use case is defined as a combination of system states for which the power consumption must be calculated depending on the percentage of the activity of each block. This power model is developed by a different team to the performance team, so there will probably be gaps between the functional and power models. This encouraged us to work with the performance modeling team to introduce a new method for adding power consideration

using the same simulation environment. Our work is to define the dynamic power model of such platform, at ESL level and linked together with the performance model using the PwClkARCH library.

One of the first aim of this work is to prove that the power-aware PwClkARCH library described in Section II.4, and the performance model described in the previous paragraph, do fit together, and then to show the benefit of applying clock gating on the L2 Coprocessor block in order to reduce dynamic power consumption, while verifying that performance and timing still fulfil the hard real time requirements of the use case in focus.

In our case, we have defined only one PD and one CD including the eight Tiles of the L2 Coprocessor block (architecture 1), enumerated from 1 to 8 (Figure 29). Thus in our PMU, we instantiate one CM and one DPC. We define one design element for each tile, DE1 for Tile 1 and in the same way up to DE8 for Tile8. Each instance of the design element DEx is parametrized with values of capacitance and leakage resistance of the corresponding Tile. There is also a supply net that sets the voltage value for all the PD. The connection between the performance model and the power model is done through the PM of the PMU unit. The scheduler has the knowledge of the task's assignment on the different IPs. When it schedules a task on a Tile, the scheduler sends the information (Tile's number and Tile's state) to the PM. A Tile is inactive when it's not running a task and when there is no task in its pending task queue. The PM takes this information and turns off the clock of the corresponding Tile if the state received is inactive, or turns it on if the received state is active. In this use case, only clock gating is applied because idle periods of the L2 Coprocessor which allow power gating to be applied, are not long enough to save power using power gating.

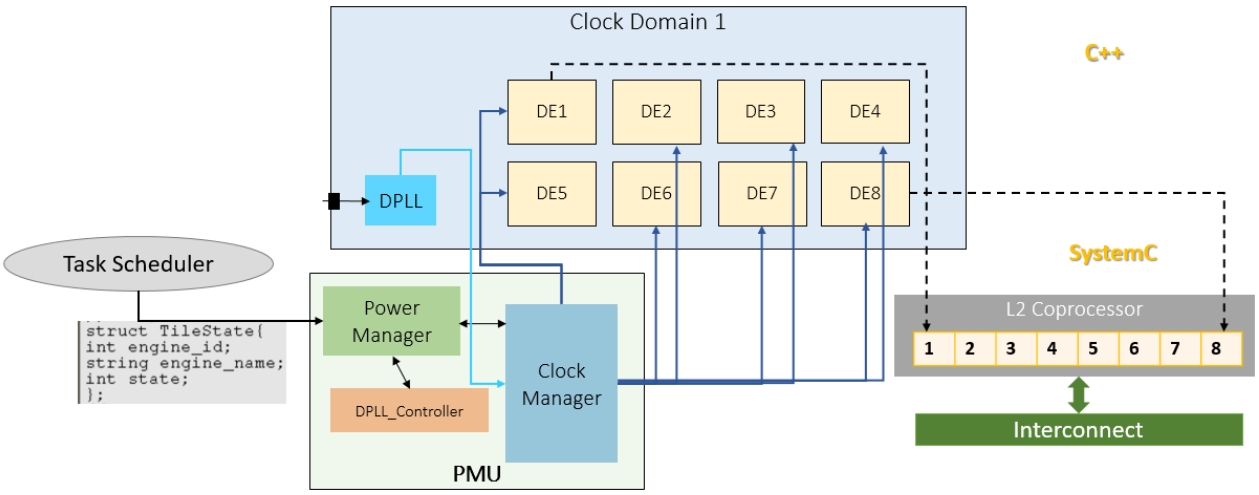


Figure 29. L2 Coprocessor power model definition using PwClkARCH

The objective of this use case is to identify the sufficient clock frequency associated with the scheduling technique developed in the scheduler, which allows to satisfy deadline constraints, and further, to obtain in

parallel the associated power profiles. The activation period of all the tasks is 1ms, the deadline is set to 0.9 ms for some tasks and to 0.75 ms for others.

Regardless of the number of Tiles, the power model definition for the architecture 2 is the same. The difference is at the interface between the PM and the task scheduler. In the first case, since we have only one scheduler, it sends the information to the PM directly. In the second case, the scheduling is distributed through the hardware engines. In fact, for the purpose of task-graph construction and mapping of tasks on different architecture resources, *tg_constructor* module is used. It is an SC MODULE and instantiated with the task-graph configuration file reference, given as a constructor parameter. The functionality of this module is to construct the task-graph and map tasks on the architecture resources. The module tg_constructor has a set of member variables and methods as shown in the Figure 30, which are called while constructing the task-graph. We add a function called *PM_info ()* in the task graph constructor which is called from the scheduler interface of each engine when a task is started or completed.

Members	Description
`void hw_engine_register (task_scheduler_engine_if *eng)`	Stores the references of processing element in `tg_constructor`.
`void register_all_tasks()`	Instantiates task nodes and invokes other function members for construction of the task-graph step by step.
`void map_task(node* tg)`	Maps tasks on their corresponding processing elements or engines.
`std::vector<std::pair <int,int>> build_graph()`	Instantiates fifo channels and establishes the connection between nodes through these fifo channels.
`void PM_info (task_scheduler_engine_if *eng, node*)`	Sends necessary information to a power estimating module during simulation.
`void print_graph (bool write)`	Prints the task-graph in text format in the activity report.
`std::vector <task_scheduler_engine_if*> hw_engine_lst`	List of instantiated architecture processing elements.
`std::vector <tg_node*> task_lst`	List of all instantiated nodes in the task-graph.

Figure 30. Main function and variable members of tg_constructor

At the end of the simulation, an activity report is generated which gives detailed information about the run-time statistics of each tasks, such as, minimum, maximum and average execution time, completion time,

deadline misses, etc. The total execution and completion time of a task depends on the load of the engine (hardware resource), load and latency of the interconnect and memory subsystems.

4. Conclusion

In this Chapter, we present Intel pre-silicon simulation environment used for performance evaluation and architecture exploration. We introduce then our work on providing a joint performance-energy consumption simulation environment. This model helps making early decisions on the choice of the most suitable architecture like memories size, bandwidth, frequency, etc., and of the scheduling strategies in communications SoCs.

The next step of our work will focus on the introduction of a memory model, within our power model, which allows a global analysis of the system, in term of performance and power, knowing that the power strategy influences the frequency of the memory requests.

IV. Model Driven Engineering for PwClkARCH-based design: Graphical and textual user interfaces for code generation

1. The Model Driven Engineering approach

Models are more and more important in the field of hardware design. Nowadays, the difference between mobile devices on the market is no longer based on the performance of the processor alone, on the contrary, it is based on consumption in terms of power and battery life. This motivates us to go further from TLM-level modeling and develop a model-based layer that can be considered our starting point. This model-based layer allows us first to separate the non-functional properties (in our case power intent and clock intent) from the functional ones, i.e. the hardware architecture, in a separation of concerns based-approach to simplify power modeling and, second to use model transformations to map the specific technology platforms and generate the SystemC-TLM code of the overall system including both power and functional models in a single simulation model. This layer is based on Model Driven Engineering (MDE) approach.

A preliminary work based on UML for specifying a power intent including a SystemC-TLM code generator was introduced in [68]. In this work, we propose to create a graphical modeling tool to simplify the integration of power management features using PwClkARCH library, MDE and the two Eclipse plugins Sirius and Acceleo. The advantage of Sirius is to simplify and facilitate the use of this library through a graphical interface and with the help of Acceleo to correctly generate the simulation code SystemC-TLM/C++, which then makes it possible to evaluate both performance and power consumption of the system.

Figure 31 illustrates the flow we propose to automatically generate a power/ performance SystemC-TLM simulation code resulting from the merging of a HDL design and the code produced from a graphical entry or a textual entry of a power/clock intent.

This Chapter describes our work around the MDE approach used to facilitate the designer ‘s task to take into consideration power aspects right at the beginning of the design flow. First, we describe our Metamodel, then we explain how this Metamodel helps us to generate the code needed to have the joint power and performance simulation. Finally, we close this part with the presentation of the graphical interface implemented for the use of the PwClkARCH library.

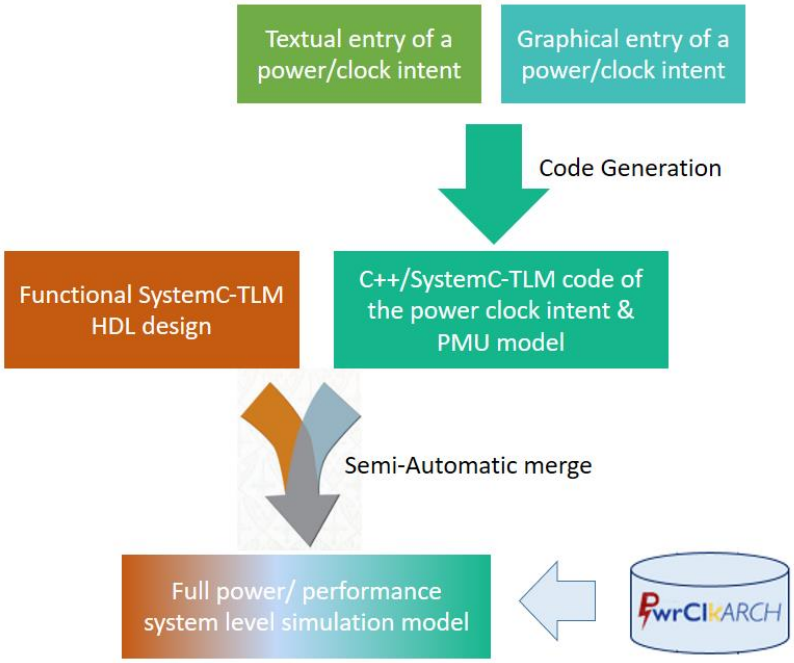


Figure 31. Power/ Performance simulation model code generation

2. PwClkARCH' Metamodel

Using MDE approach and its own transformation techniques, executable models (simulation code) can be generated from a high-level model. To do that, the first step is to define the metamodel of the system and then define the transformation rules that allow generating the code needed for the simulation of a given system. A metamodel is a model of a model. It is a simplified model of an actual model of a circuit, system, or software.

In the MDE methodology, and according to the traditional Object Management Group (OMG) metamodeling infrastructure, we have basically four layers, from the meta-metamodel layer M3, which is the top layer, to the real-world layer M0. These layers are presented in Figure 32. The Meta Object Facility MOF is an OMG standard designed to provide a means in order to define the structure or an abstract syntax of a language or data. The most popular example of layer M2 MOF is the Unified Modeling Language (UML). A model (M1) must conform to the metamodel and it represents a real system of layer M0. MOF defines a model interchange standard XML Metadata Interchange (XMI) for serializing MOF-based models.

In practice, we use the Eclipse Modeling Framework (EMF) [69] to model the PwClkARCH' metamodel based on the Ecore format [70], which is basically a subset of UML class diagrams. EMF is a modeling framework and code generation facility based on structured data model. A metamodel can be considered as a Class Diagram on the Meta level. Thus, basically we have the same semantics except that we add an "E" to refer to Ecore. EMF defines the types of the meta classes as EClasses. EClasses can have EAttributes to

describe the properties, EReferences to describe associations and relations, and EOperation to represent the modeled operations local to a specific class. EMF is easy to use for creating and editing models. In addition, it allows the validation of created models, which is essential to keep compliance with their metamodels. Defining the PwClkARCH metamodel is a very sensitive operation, as a valid model is needed to run the transformations. In fact, we must specify the containment relationship between the classes that represents the basic relation for the instantiation of the model. We cannot for example instantiate a power domain in our model if we have not defined a reference of type containment from the *Model* class to the *Top* class and then the *Power_Domain* class. EMF provides also the possibility to enrich the created metamodels by adding Object Constraint Language (OCL) constraints. We will explain those constraints later.

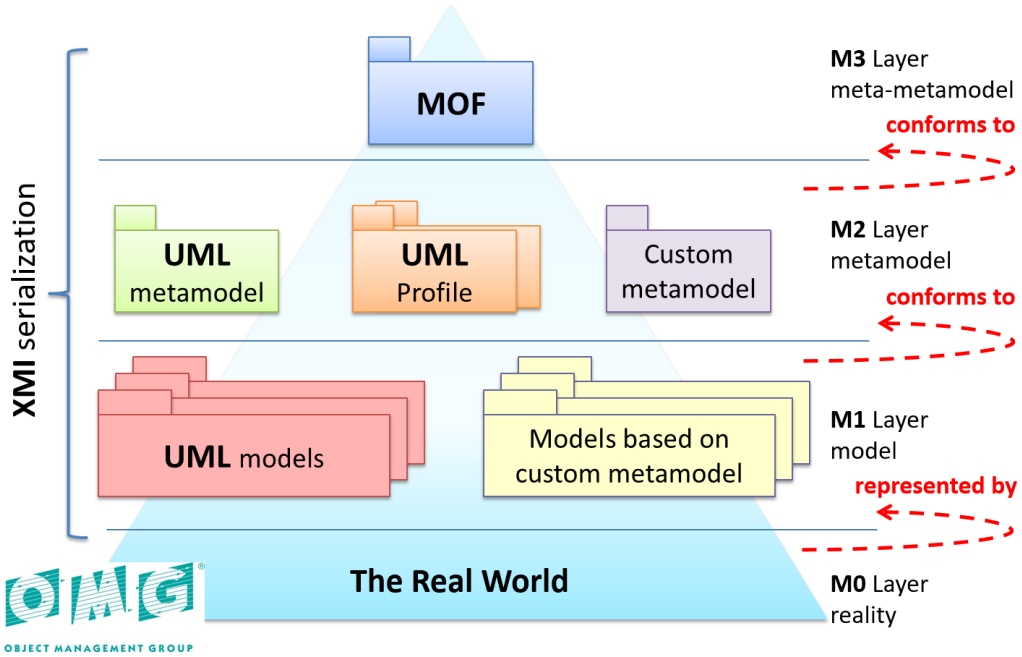


Figure 32. OMG' flow infrastructure

Figure 33 represents the metamodel of the PwClkARCH library described with the EMF framework. We define different EClasses for the different classes implemented in the library which have been introduced in Section II.4 like *Power_Domain*, *Clock_Domain*, *Design_Element*, *DPLL*, etc. *Model* class declares the Power Model of the system.

Taking the example of a design element in the architecture, it has to belong to one clock domain and to one power domain. To model this relationship in our metamodel, we consider the *DesignElem* class that represents the components of the architecture, the *ClockDomain* class to represent the clock domains and the *PowerDomain* class for the power domains representation. We add then two references from the

DesignElem class to the *ClockDomain* and from the *ClockDomain* to the *PowerDomain* classes called respectively *belongsCD* and *belongsPD*. As one component can't belong at the same time to more than one clock domain or one power domain, we set the upper bound of these references to one (more information are given in the Appendix).

All structural constraints imposed by cardinality constraints, composition relationships and OCL constraints defined in the metamodel are validated and have to be respected when building a model. Usually a model is simply obtained using EMF dynamic instance creation option, but in our case, the aim is to create the model graphically using drag and drop in an editor to facilitate the use of the library.

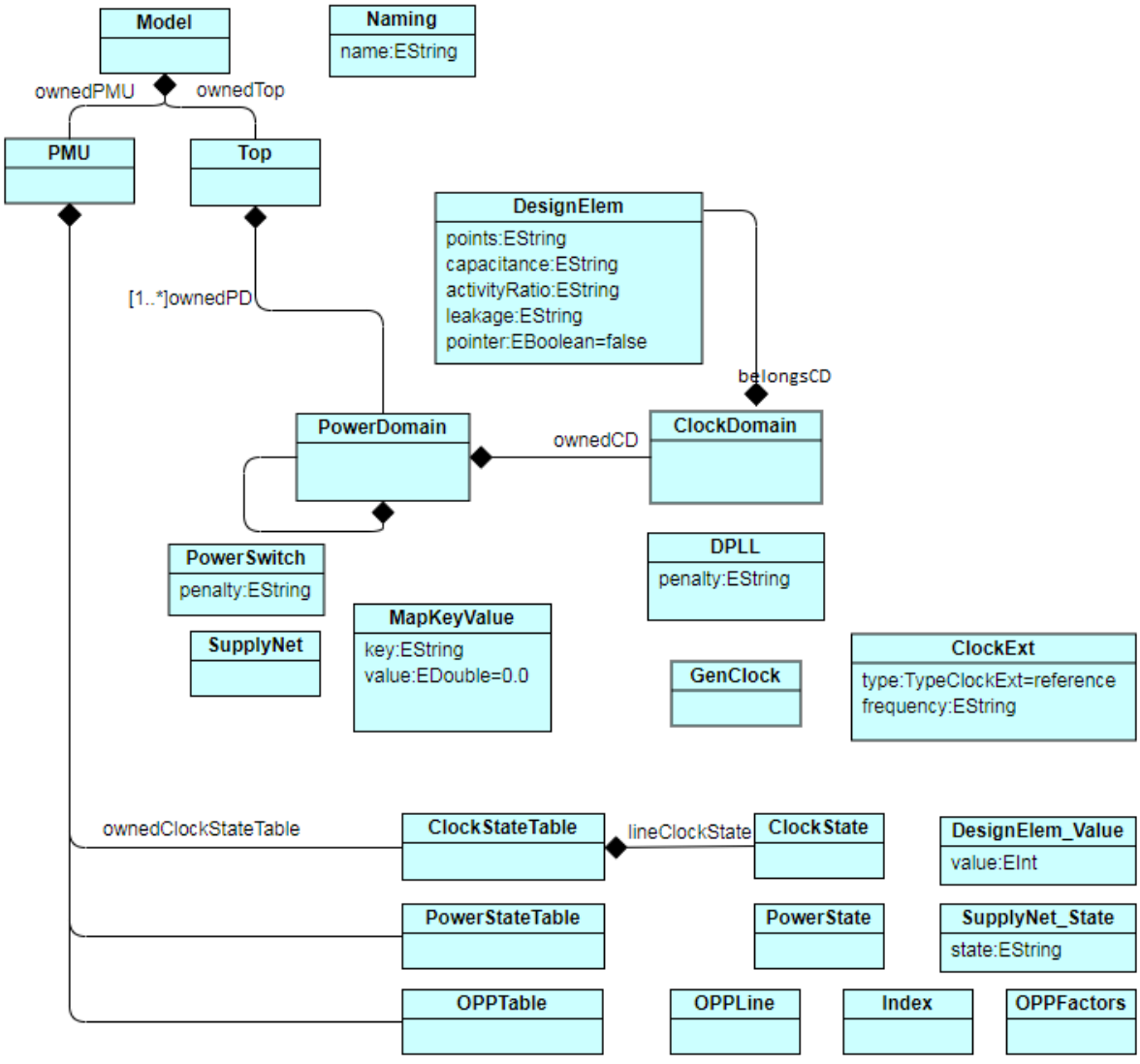


Figure 33. PwClkARCH Metamodel

3. Code generation

The MDE approach is a well-suited solution for automating a model transformation which in our case consists in the code generation in C++ and SystemC-TLM representing the structure of a power/clock intent. A model transformation (MT) [71] is a compilation process that allows moving from an abstract model to a more detailed target model. So, before performing any MT, a set of transformation rules must be specified first. Each MT is performed using a transformation engine which applies transformation rules to a source model so as to generate a target model. Model transformations used in our work are of type Model-to-Text (M2T): executable models are generated from a specific high-level model using Acceleo template(s). Thus, most of the C++ and SystemC-TLM code needed to run power simulation is generated automatically.

To do that, we use the plugin Acceleo. Acceleo [72] is an open-source code generator from the Eclipse foundation. It is a pragmatic implementation of the MOF Model to Text (M2T) standard. Acceleo ensures code/model synchronization and ensures incremental generation of the code. Figure 34 represents the four steps needed to generate codes using the Acceleo plugin, from the EMF modeling to the code generation. The inputs of the M2T acceleo transformation process are: the metamodel, the model and the template. The output text (codes) may have several forms/types (e.g. SystemC, C++, Java, text…).

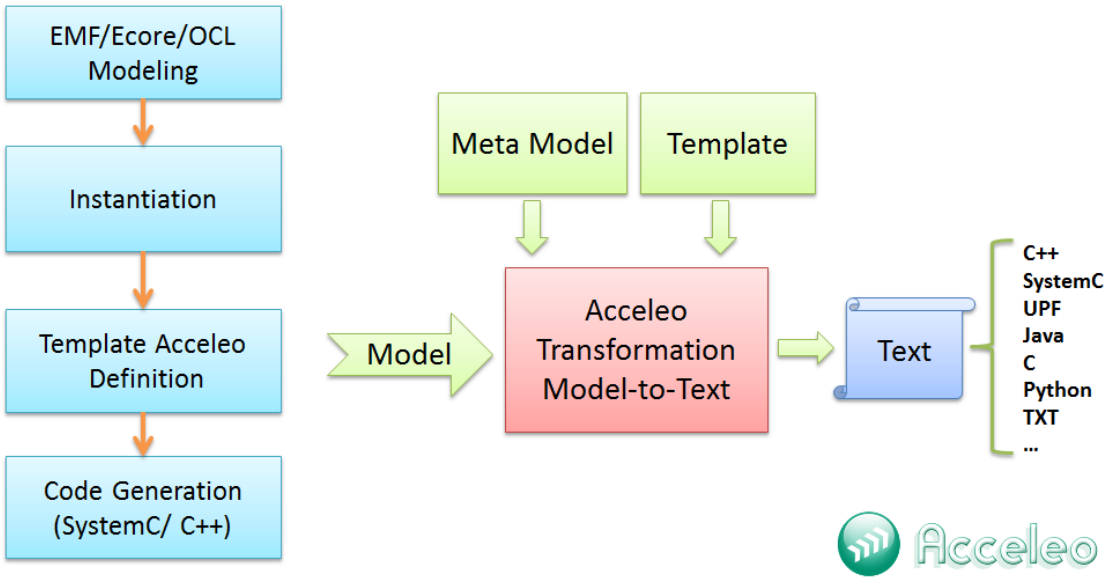


Figure 34. Model to Text Transformation schema

For code generation, Acceleo provides a pseudo-language for navigating models and extracting the data needed to generate code based on EMF's EClass, EAttribute and EOperation elements. Using this language, we implement one or more templates to define the syntax of the output text.

We must specify the metamodel that will be used for retrieving the information needed for the generation code when creating the Acceleo project. Thus, the metamodel information section allows specifying the input metamodel of our new module file, either from the list of registered metamodels or directly by URI. In this way, we have connected the different parts of the MDE approach, in particular the metamodel, the model and the Acceleo template, to build the full chain of the code generation.

We may have one or many templates; it's not the number of templates that defines the number of the output files. In fact, in the code of the template *Generate.mtl*, the tag "[file]" indicates the beginning of the generated text. It takes as input a few parameters including the name of the file and the character encoding. Since we define one metamodel that models the whole power and clock intent of the platform, we define also one template that generates three files which are the "PMU.h" and "PMU.cpp" SystemC-TLM codes, and the "sc_main.cpp" C++ code. The files "PMU.h" and "PMU.cpp" include the creation of Clock manager units, the PM unit and the interconnection. More precisely, they define the signals and the binding between the different components of the PMU unit.

The C++ generated code "sc_main.cpp" includes the definitions of the different clock and power domains of the platform defined in the model. It instantiates also the components (design elements), the DPLLs, power switches, the different clocks needed for the definition of the clock tree, and the three tables OPPT, CLKST, PST that allow to specify the values of the different parameters used to apply a given power management technique. This code must be included later in the SystemC-TLM code defining the hardware architecture of the platform (top level main code for example).

For complex hardware architecture including a large number of IPs, it could be error prone to develop manually a model of a power/clock intent directly using dynamic instances from classes of the metamodel. To address this problem, a designer should create a model with more advanced features. To do that, we have developed two types of tools: a graphical entry tool (next section) able to perform verification dynamically on the structure of the power/clock intent while it is created, and a textual entry tool (section 5) based on UPF-inspired syntax.

4. Graphical workbench

After preparing the basis of our tool including metamodeling and code generation, we start the implementation of a graphical editor to represent the PwClkARCH interface to the user. This tool should executes all the facilities that have been explained in Section II.4. Like every graphical editor, we introduce a palette. In our case, the palette helps the user to:

- Add, edit, and delete library elements (Power Domain, Clock Domain, Design Element, Supply Net, Power Switch, Clock Extern, Generated Clock, and DPLL).

- Modify the properties of the different elements of the library:
  - ✓ The names associated with the instances of the different elements.
  - ✓ Time penalty values for the Power Switch and the DPLL.
  - ✓ Capacitance, leakage resistance and activity factor values of the Design Element.
  - ✓ Clock Extern type and frequency value.
- Enter graphically the different tables needed for power management strategy implementation:
  - ✓ Enter the Power State Table (PST).
  - ✓ Enter the Clock State Table (CST).
  - ✓ Enter the OPP Table (OPPT).
- Generate SystemC-TLM code from the graphical view needed to define the power model of the architecture. This code is divided into 3 codes: *PMU.h*, *PMU.cpp* and the code need to be inserted in the main class of the model (*sc_main.cpp*).
- Import previously defined power models and their graphical visualization so they can be easily modified.

In addition to these functions related mainly to the definition of the power model, we have introduced some features to facilitate the use of the tool to the user. For that, two points have been studied:

- Ergonomics: the graphical interface of the tool should be simple and practical to ensure the comfort for the user without referring to specific prerequisites. For example:
  - ✓ Automate the creation of elements when using the graphical editor. For example, when a nested power domain is created inside a parent power domain, automatically a power switch is inserted inside that domain and its input is connected to the internal supply net of the power domain.
- Reliability: The created tool must be reliable and produce correct results regardless of the manipulation of the user. Reliability is ensured thanks to:
  - ✓ A comprehensive study of the structure of the PwClkARCH library to define the most appropriate metamodel.
  - ✓ OCL-based preconditions to check if the actions of the user are feasible and can prevent most kinds of misuse. OCL is a formal language that can be used to specify invariants, pre- and post- conditions as well as describe guards and constraints on operations. The creation of OCL was a very interesting effort to bring some formality to UML. OCL expressions do not have side effects. That is, when they are evaluated the system state does not change.

4.1. Main Interface

The main interface of our graphical tool is depicted in Figure 35. It is designed with a Drag and Drop palette (1) to execute the functionalities described previously in order to obtain a coherent graphical presentation of a power model of a given system.

The workspace (2) presents the editor in which we drop elements from the palette. In Figure 35, we show the different clock and power domains, the design elements, the supply nets etc. used to model the power Intent and the clock intent of a given architecture.

The third part of the graphical interface of our tool is the Properties window (3) that helps the user to enter the properties of the different elements. “Belongs PD” and “Belongs CD” properties, as the names indicate, allow to specify respectively the PD and the CD, which the selected element in the workspace belongs to. We also remark that the values of the three parameters necessary for the calculation of the power and the energy, capacitance, activity ratio, and leakage resistance, can be specified and modified easily without going into the code.

The menu bar (4) gathers tools acting on graphic elements to ensure the following features:

- Change the colors through a color palette.
- Export the graphic of the power model in image format.
- Enlarge or reduce the size of the chart.
- Filter the display to perform the desired items only.

In this part, we were interested in the stage of development of the graphical workbench using the Sirius plugin to build a model conforming to the metamodel PwClkARCH. For that we started by creating a project of type "Viewpoint Specification Project" and we handled mainly the file ".odesign" and Java classes to reach our needs. First, we defined the graphical aspect associated with the different EMF classes and describe the structure, appearance and behavior of the modelers that represent the elements of the palette. Second, we set preconditions in OCL to prevent the user from making errors when using the graphical editor. For example, we check the PD existence before we can create a Supply Net. That is, we allow the addition of a SN only for PDs located in the "Top" element. Third, we automated the creation of the different elements of the library to reduce the risk of error when presenting the power management methods and facilitate the task to the user. This can be done for example in the case of the power model's structural situations like the creation of the CD; By structure, a CD has a DPLL and a DPLL has two input clocks and one output clock. As a result, during graphic manipulation, a creation of a CD generates an automatic creation of a DPLL, two Clock Externs as input and one Generated Clock element as output.

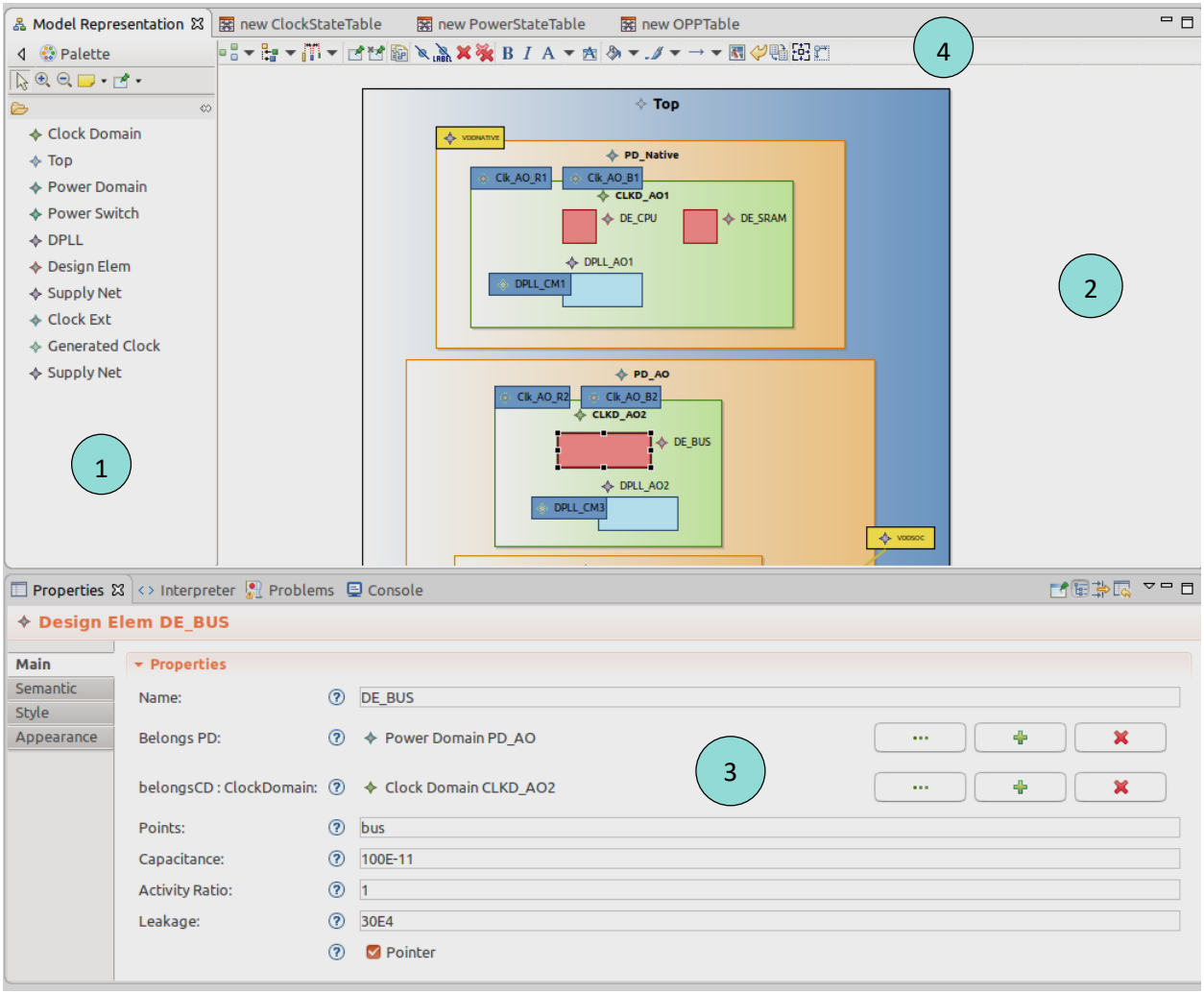
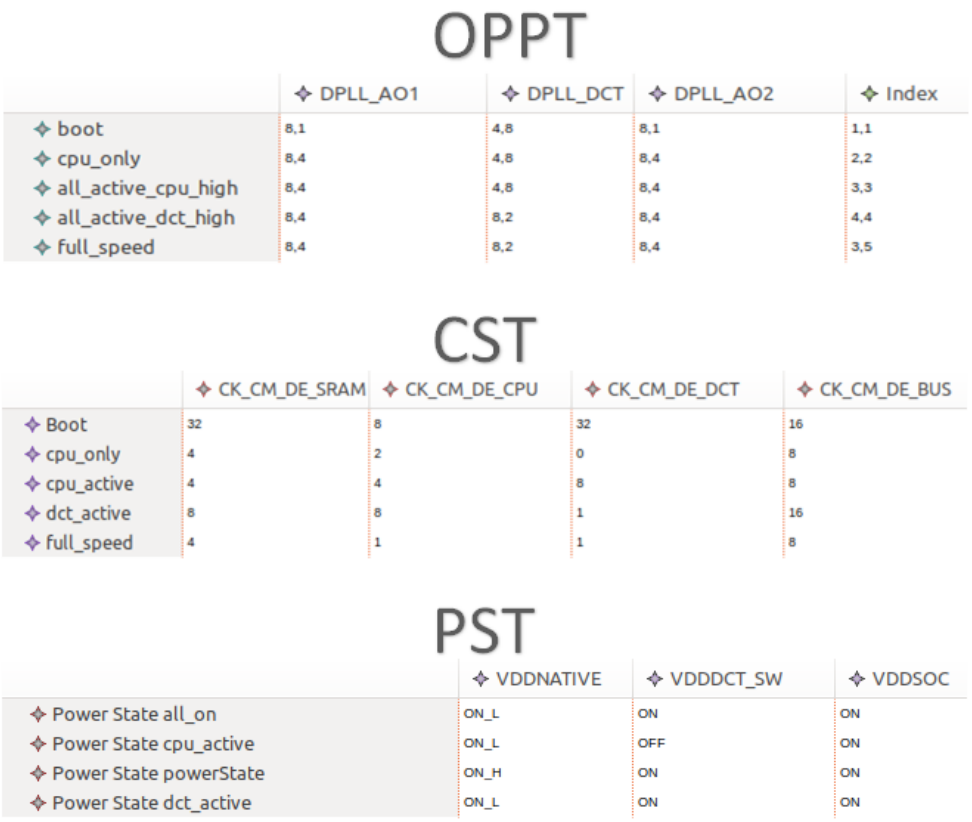


Figure 35. Main Interface of the graphical entry tool

4.2. Power and clock states tables ' interface

To manage the CST, PST and OPPT tables from the graphical tool, we use the Sirius plugin. Sirius is a project that makes it easy to create a graphical modeling workbench using Eclipse modeling technologies, including EMF and GMF (Graphical Modeling Framework).

To accomplish this task, we translated the structure of the tables represented in Figure 36 as class instances internally to an XML file illustrating the hierarchy of the power management strategy (Figure 37). It is this XML file that will be used later in the code generation by Acceleo.



OPPT

	DPLL_AO1	DPLL_DCT	DPLL_AO2	Index
boot	8,1	4,8	8,1	1,1
cpu_only	8,4	4,8	8,4	2,2
all_active_cpu_high	8,4	4,8	8,4	3,3
all_active_dct_high	8,4	8,2	8,4	4,4
full_speed	8,4	8,2	8,4	3,5

CST

	CK_CM_DE_SRAM	CK_CM_DE_CPU	CK_CM_DE_DCT	CK_CM_DE_BUS
Boot	32	8	32	16
cpu_only	4	2	0	8
cpu_active	4	4	8	8
dct_active	8	8	1	16
full_speed	4	1	1	8

PST

	VDDNATIVE	VDDDCT_SW	VDDSOC
Power State all_on	ON_L	ON	ON
Power State cpu_active	ON_L	OFF	ON
Power State powerState	ON_H	ON	ON
Power State dct_active	ON_L	ON	ON

Figure 36. Example of Tables Views

Figure 37. XML representations of CLKST, PST and OPPT

5. UPF-like Integration

EDA tools addressing low power design at register transfer level (RTL) down to GDSII level use standards such as UPF (Unified Power Format, IEEE standard 1801, chapter II section 2.4). A power architecture (power intent) specified using UPF includes a partitioning of the functional IP blocks (e.g. see Figure 38) of the architecture into power domains (e.g. PD1), supply nets (e.g. VDD1) and power switches (e.g. PS1) for power gating facility. Low power design using UPF supports a strong separation between the HDL description of the architecture and the UPF description of the power intent. Unfortunately, such low power design flow is not available at ESL and UPF does not provide any facility to describe a clock tree, whereas clock tree modeling is critical to capture and control the switching power.

In this context, we have developed also a parser of a simple UPF-like language to add more facilities to the creation of a model on the one hand, and to get closer to the UPF standard on the other hand. Usually SoC designers are used to capture their design with script-based entry tools rather than graphical tools. Thus, the idea is to introduce an UPF-like file in our generation code flow. In this file, the power model of the platform is defined using commands inspired from UPF syntax (especially in the case of clock domains). Let's explain this generation flow using another platform example. Considering the following power model that we want to simulate (Figure 38).

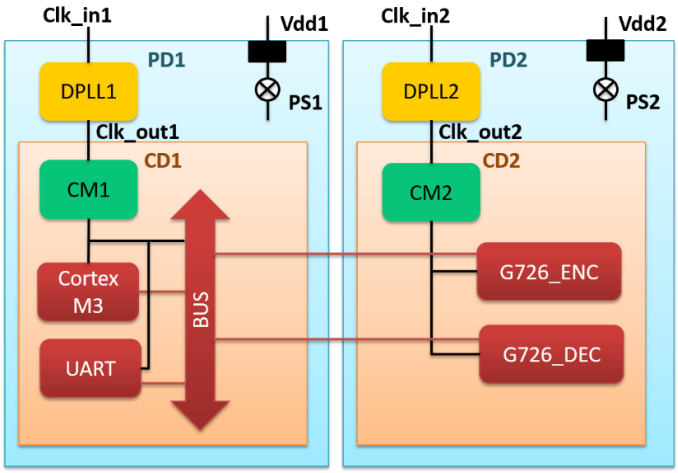


Figure 38. Power Model Example

To create a power domain called PD1 that includes a *CortexM3*, an UART and a Bus, we use the following command: ***create_power_domain*** *PD1 –elements {CortexM3/Bus/UART}*

To create a clock domain called CD1 that includes a *CortexM3*, an UART and a Bus, we use the following command: ***create_clock_domain*** *CD1 –elements {CortexM3/Bus/UART}*

We also use commands to define Supply Nets, input and output clocks of the DPLLs and all the elements needed to define our power model (Figure 39).

```
create_power_domain PD1 -elements {CortexM3/Bus/UART}
create_power_domain PD2 -elements {G726_ENC/G726_DEC}

create_clock_domain CD2 -elements {G726_ENC/G726_DEC}
create_clock_domain CD1 -elements {CortexM3/Bus/UART}

create_supply_clock Clk_in1 -direction in -domain {CD1}
create_supply_clock Clk_out1 -direction out -domain {CD1}
create_supply_clock Clk_in2 -direction in -domain {CD2}
create_supply_clock Clk_out2 -direction out -domain {CD2}

create_power_switch PS1 -domain PD1
create_power_switch PS2 -domain PD2

create_dpll DPLL1 -domain CD1  -penalty_delay 10 -output_supply_clock Clk_out1 -input_supply_clock {Clk_in1}
create_dpll DPLL2 -domain CD2 -penalty_delay 5 -output_supply_clock Clk_out2 -input_supply_clock {Clk_in2}

end;
```

Figure 39. UPF-like input file

In order to be comprehensible by eclipse, this file should be transformed into XML syntax. For that, we use JFlex which is a lexical analyzer generator for Java written in Java. The main design goals of JFlex are: full unicode support, fast generated scanners and platform independence. We develop our lexical UPF-like parser that contains a set of rules which correspond to a set of actions. For example: for the first command in the Figure 39, the actions to do are first to recover everything written after create_power_domain, then extract the name of the power domain, and later the list of the elements included in this PD. The output of the parser should be the same as the XML syntax of a model created using the Eclipse editor, which has been validated and tested on this example (Figure 40).

Thus, from an UPF-like script that defines the power model of a given architecture, we generate automatically the XML model that could be used in our EMF project described earlier to generate the simulation code (Figure 41). This UPF-like-based approach illustrates the possibility of defining at the ESL level a scripting language inspired by UPF in order to reinforce the separation between functional model and power model at this level of ESL abstraction. This type of scripting language should be standardized to facilitate the development of power/performance simulation tools addressing this ESL level.

```
<myPd name="PD1"/>
<myPd name="PD2"/>
<myDe name="CortexM3" belongsPD="//@myPd.0" belongsCD="//@myCd.1" points="//@mySo.0"/>
<myDe name="Bus" belongsPD="//@myPd.0" belongsCD="//@myCd.1" points="//@mySo.1"/>
<myDe name="UART" belongsPD="//@myPd.0" belongsCD="//@myCd.1" points="//@mySo.2"/>
<myDe name="G726_ENC" belongsPD="//@myPd.1" belongsCD="//@myCd.0" points="//@mySo.3"/>
<myDe name="G726_DEC" belongsPD="//@myPd.1" belongsCD="//@myCd.0" points="//@mySo.4"/>
<myCd name="CD2"/>
<myCd name="CD1"/>
<mySo name="CortexM3"/>
<mySo name="Bus"/>
<mySo name="UART"/>
<mySo name="G726_ENC"/>
<mySo name="G726_DEC"/>
<myDplls name="DPLL1" supply="//@myCd.1" has="//@myClkExt.0" has_one="//@myGenClk.0" penalty_delay="10.0"/>
<myDplls name="DPLL2" supply="//@myCd.0" has="//@myClkExt.1" has_one="//@myGenClk.1" penalty_delay="5.0"/>
<myClkExt name="Clk_in1"/>
<myClkExt name="Clk_in2"/>
<myGenClk name="Clk_out1"/>
<myGenClk name="Clk_out2"/>
<myPS name="PS1" Supply="//@myPd.0"/>
<myPS name="PS2" Supply="//@myPd.1"/>
```

Figure 40. XML output syntax

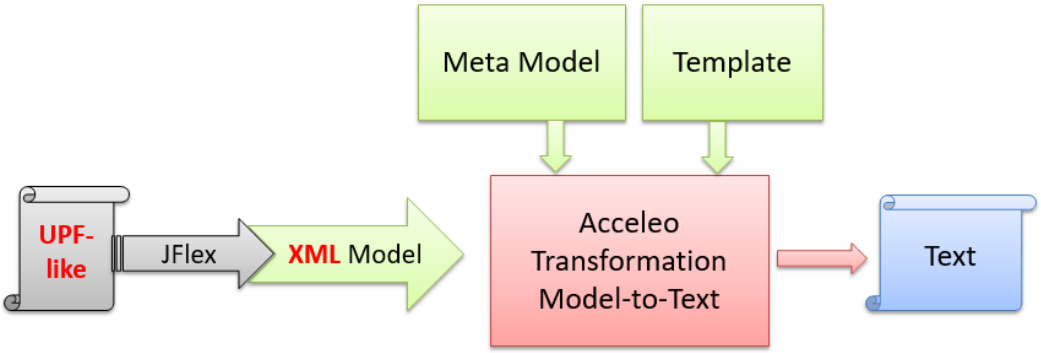


Figure 41. UPF-like generation flow

6. Conclusion

Figure 42 represents the construction chain of our graphical tool, needed to apply the methodology implemented in PwClkARCH. We distinguish a first phase of development of the tool, the upper line, and a second phase of exploitation of this tool which is the lower line. The development of the tool requires first of all to develop the metamodel under EMF and then to associate with this metamodel all the graphic elements under Sirius that refer to the metamodel. Once the development of the graphical tool is realized, we can exploit it by creating a graphic model (the box on the bottom right of the figure), which makes it possible to build a model of a Power Intent of any platform which is conform to the PwClkARCH

metamodel. This work makes the library more attractive to the user as he manipulates a graphic editor rather than lines of codes.

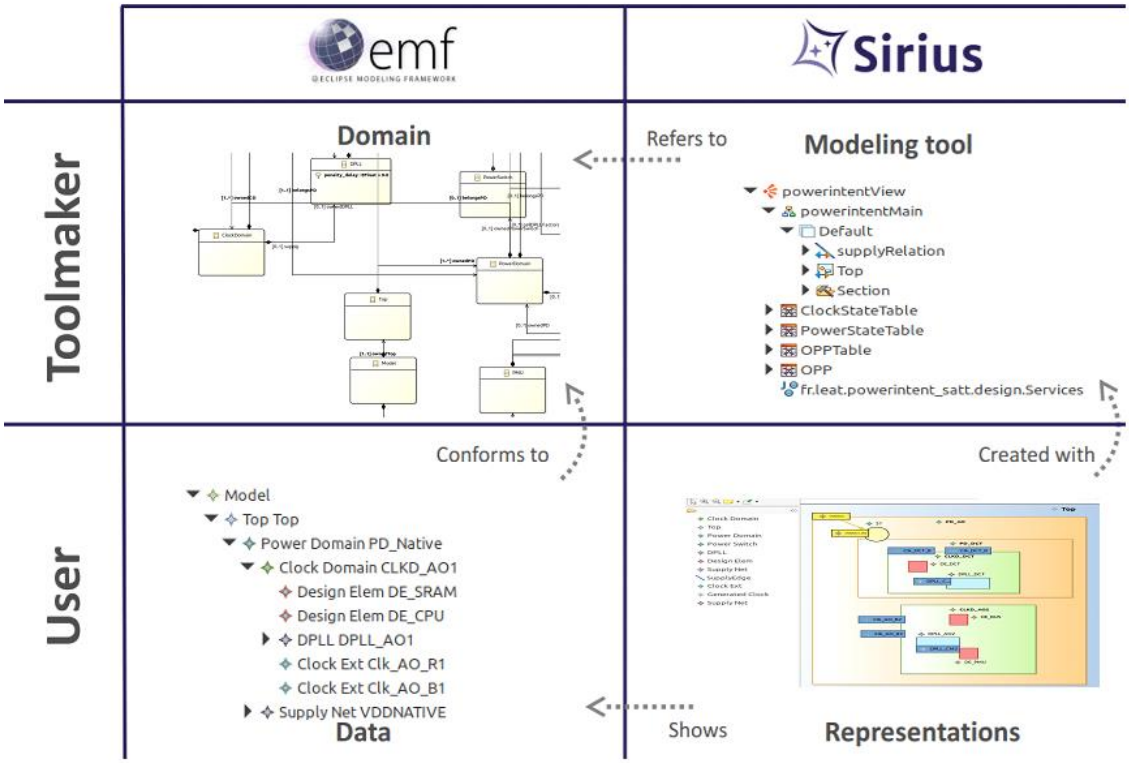


Figure 42. Steps of the graphical tool development

All the developments introduced in this chapter concern a power/performance simulation environment for hardware architecture composed of a set of interconnected IPs. But as illustrated in chapter II, the memory sub-system could impact significantly performance and power of a SoC because its behavior depends on the type of memory accesses transmitted by the SoC, on its internal state, on decisions from the memory controller, and in return the timing of completion of read/write operations by the memory sub-system impacts the behavior of the SoC. To capture these behaviors, simulation models of the memory sub-system should be developed in conjunction with the power/performance model of the SoC. This point is addressed in the next chapter.

V. Exploration of memory technology based on simulation at TLM level

In most computing systems, and especially in mobile terminals, the memory system represents a major performance and energy bottleneck. The features and therefore the requirements on the memory system are increasing rapidly [73] and need to be addressed as early as possible in the design flow. This is even more important since the trend of memory consumption is of the same order of magnitude as that of the logic part as stated by the International Technology Roadmap for Semiconductors (ITRS) (Figure 43).

In fact, the multi-core design that nowadays underpins most of the mobile device architectures, made of both homogeneous and heterogeneous cores, and the presence of components that perform specific processing, e.g. cellular communication [74] or video capture, lead to concurrent memory access streams across the communication subsystem of the platform. Added to this, power management decisions that vary the frequency of the components, for example when using DVFS techniques, modify dynamically the usage of memory system. It is therefore necessary to consider an environment capable of describing all these aspects in order to effectively evaluate the performance and power of the entire system.

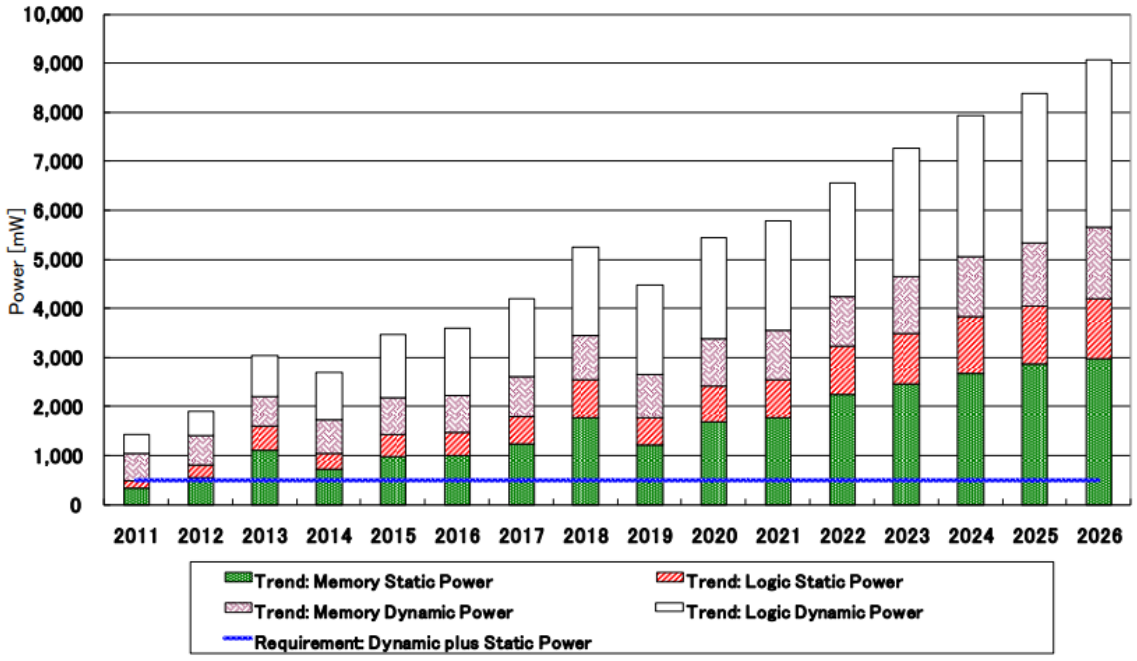


Figure 43. The expected trend in power consumption for logic and memory circuits in portable consumer devices, as reported by the 2011 edition of ITRS

As illustrated in chapter II.5, various DRAM simulators have been developed focusing on different aspects but a few of them include an accurate power model. Among them, DRAMPower appears as a good candidate in the perspective of developing a co-simulation framework with SystemC-TLM.

1. DRAMPower tool

DRAMPower is a high-level DRAM power tool that performs high-precision modeling of the power consumption of different DRAM operations, state transitions and power-saving modes at ESL level [5], [75]. The tool can be employed using two interfaces, the command-interface, which assumes that a DRAM controller is available in the system model, and the transaction-interface, in which the DRAMPower tool command scheduler is used together with additional options including interleaving, request size, bank group and power strategies. Basically, DRAMPower is composed of a front-end and back-end as illustrated in Figure 44. The back-end evaluates the power consumption (DRAMPower combines dynamic and static power) of the DRAM based on the schedule of commands generated by the front-end. The front-end receives read and write transactions from masters and calculates the time slots where appropriate commands can be scheduled. The schedule is computed according to the timing characteristics of the DRAM, the values of addresses in the transaction requests and the state of the memory banks in the DRAM. Thus, the transaction-interface is used as input of the front-end and the command-interface is used as input of the back-end. In our case we use DRAMPower as the combination of both the front-end, as functional model, and of the back-end, as power model.

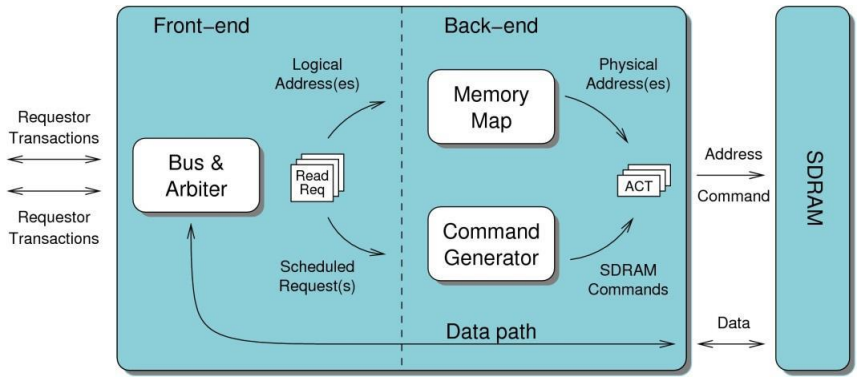


Figure 44. Overview of general SDRAM controller [38]

Defining a transaction trace format is a simple operation in DRAMPower. A line in the trace file is composed of three comma delimited fields:

Timestamp, READ/WRITE, Address

Where Timestamp is the delay in cycles from the previous request. *READ/WRITE* is the type of transaction, and *Address* is the address in the physical memory.

The tool provides as examples thirty-six memory specifications, including different configurations of DDR and LPDDR based on Micron Technology datasheets and the Joint Electron Device Engineering Council (JEDEC) memory standards timing specifications.

Although STT-MRAM technology was introduced only around ten years ago, STT-MRAM devices are already approaching DRAM in terms of capacity, frequency and device size.

Since STT-MRAM is a relatively new technology with no specific standard, the manufactures have two options: make a STT-MRAM main memory by deploying specific STT-MRAM specific memory controllers, or reuse the DDRx standard.

In [64], the authors compare a 4Gb LPDDR3 DRAM and a 4Gb LPDDR3 STT-MRAM by parameterizing the two simulators CACTI [57] and NVSIM [76]. They notice that the two types of memory provide the same JDEC interface (based LPDDR3). Therefore, this work illustrates that once the timing and power parameters of a STT-MRAM are available, a SDRAM simulator can be also used for simulating an MRAM technology.

Initially, we use the DRAMPower tool as standalone to test it and remake the comparison done in [64] from which we obtain the timing and power parameters of a DRAM and MRAM LPDDR3 devices. In [64] authors design an LPDDRx-compatible MRAM interface using three optimization techniques: EarlyPA, DynLat and ComboAS. The former is used to leverage the non-destructive reads of the MRAM (II.5.3). The other two items are used to solve the small page size issue. In their work, they use a modified CACTI and NVSIM to simulate and compare the two memory technologies. Figure 45 presents the comparison between the two memory systems of the average power consumption when running four main use cases, which are available in the DRAMPower tool: JPEG, H263, MPEG2, and EPIC. Compared to DRAM, the MRAM consumes less power, thanks to the lack of power-refresh procedures. In [64], authors found that the performance of their optimized MRAM system is similar to the DRAM system. They also compared the energy consumption between the two systems (Figure 46), in which they plot the refresh energy, the burst energy, the activation/Precharge energy and the background peripheral circuit energy separately. They proved that compared to DRAM, MRAM-based system consumes zero refresh energy and this is because of its non-volatility. As a conclusion, MRAM is an attractive candidate to build the main memory system and the zero-refresh energy is the major source of the MRAM energy saving.

However, our first operation on the tool was not good enough for us. In fact, feeding the tool in the standalone mode with execution traces makes it not possible to measure the access times to the memory on the time stamps of read/write events, and such limitation causes a suboptimal or an even erroneous performance analysis.

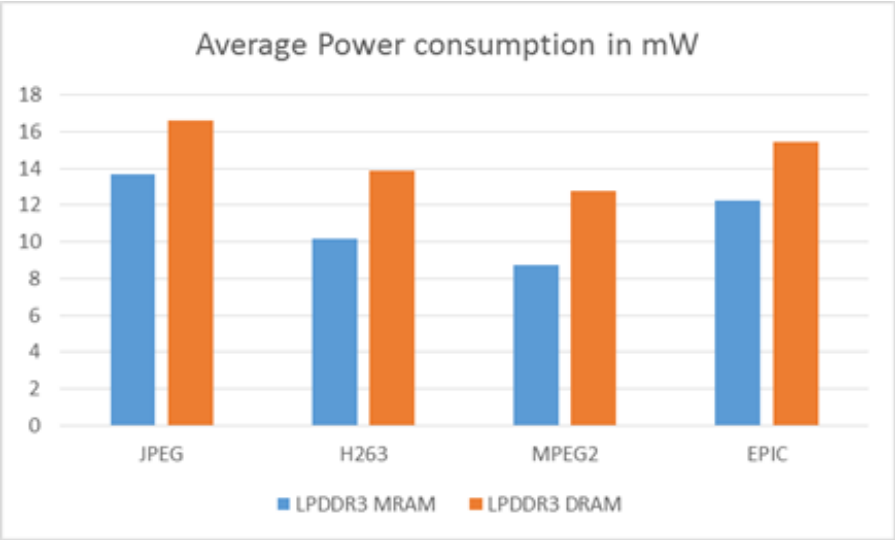


Figure 45. Average power consumption in mW for MRAM and DRAM

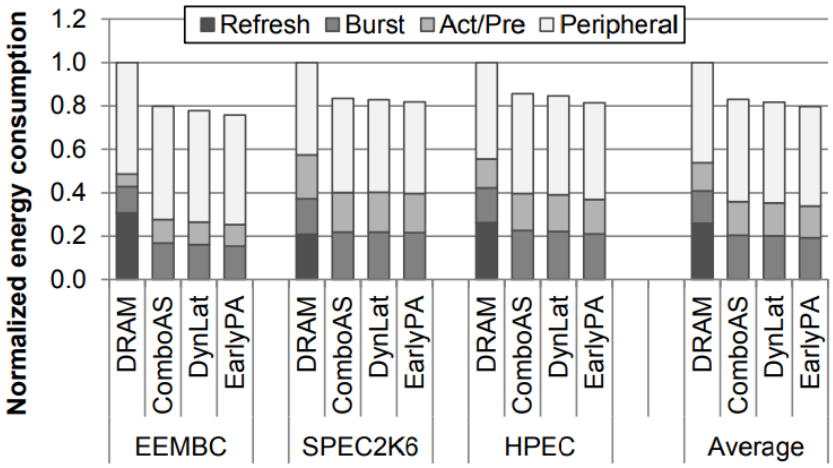


Figure 46. Normalized energy consumption of DRAM system and MRAM system adopted different techniques [64]

These preliminary results show that the use of the DRAMPower tool can help us to have an accurate estimate of the energy consumption of a given memory technology. And so, connecting SystemC-TLM and PwClkARCH with DRAMPower will allow us to have a framework that estimates the overall performance of the system and the power consumption of each component of the platform, including the memory system with a precise power model taking all memory operations into account. Power management strategies could then be explored considering the mutual effects of the processing architecture part and the memory storage part of the whole system.

2. Memory exploration flow

DRAM has a big impact on performance and power consumption in mobile devices. Evaluating the efficiency of power management strategies and performance of DRAMs requires an approach that includes an appropriate model of the DRAM, a realistic controller, a high-precision power model, and a representative workload of real use cases. All of the above requires a full system simulation framework able to represent the impact of the different mutual interactions between all the components in mobile platform and the memory subsystem.

In this work, we present a framework that allows exploration of different power management strategies of given SystemC-TLM based VP platforms, taking into account the power consumption of the different operations of DRAM including the main related operations (e.g., read, write, refresh, activate, etc.), using the DRAMPower tool cited above. We incorporate one MRAM specification into DRAMPower in order to explore different memory technologies and to study the impact of memory parameters on power and performance values. We first make use of our framework on a basic SystemC-TLM platform.

We consider a simple SystemC-TLM VP (DCT Platform) that executes a Discrete Cosine Transform (DCT). In addition to the DCT hardware block (Figure 47), the platform includes a CPU, a bus and a memory. In this example, the bus and the memory are assigned to the same power domain PD_AO and clock domain CLKD_M, the CPU is in a power domain PD_Native and in a clock domain CLKD_C, and the DCT unit is in power domain PD_DCT and in clock domain CLKD_DCT.

Figure 48 illustrates the behavior of this architecture: the CPU initializes first in memory a vector of 256 elements, then it transfers this vector to the DCT unit and activates the computation of the DCT by the DCT unit. The CPU polls until the end of the DCT computation and then reads the DCT results and stores the values in the memory.

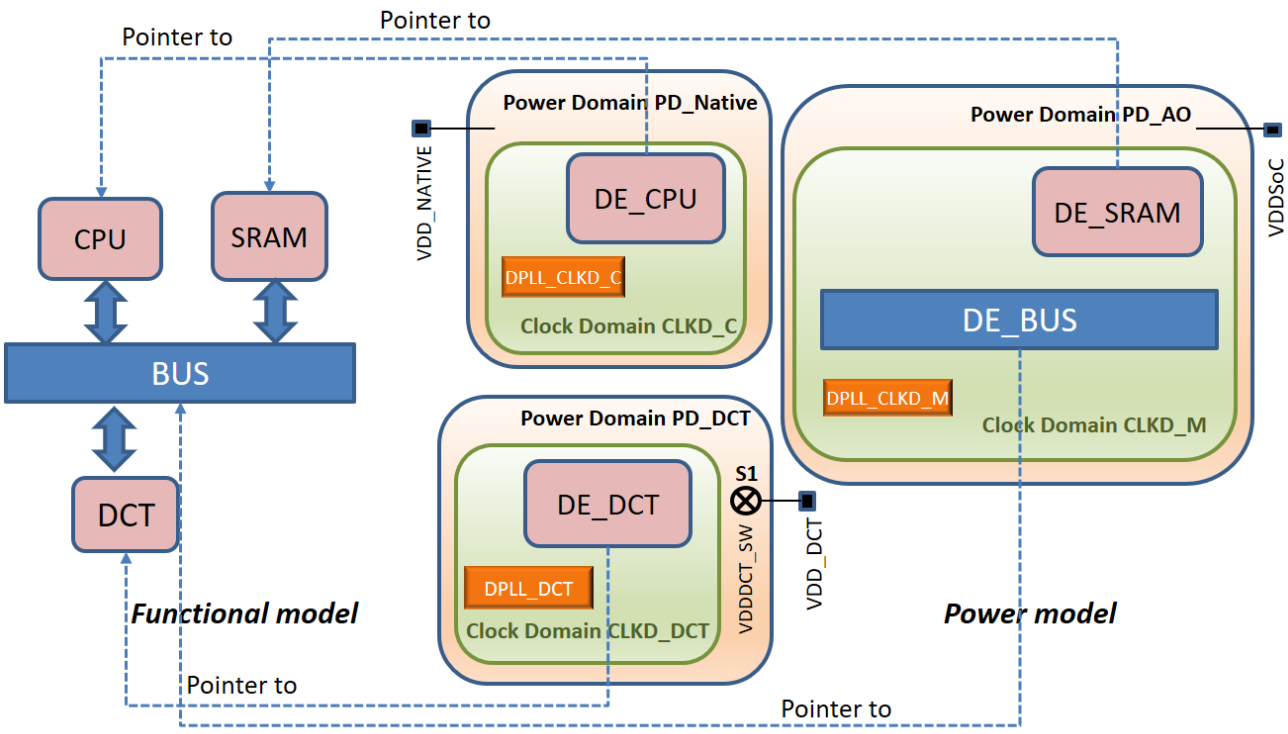


Figure 47. DCT Power model example

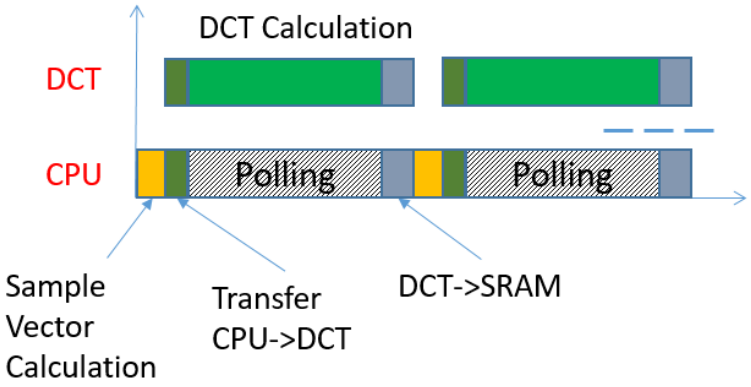


Figure 48. Functional behavior of the DCT platform

As the power model developed in DRAMPower has proven to be accurate [5], it is linked through our framework to the PwClkARCH library, so to define the memory power model and energy consumption that offer a complete power and functional model of a mobile platform.

The connection of DRAMPower with the DCT Platform augmented with the power intent is realized using the memory unit from the functional model and the design element DE_SRAM from the power intent (Figure 49). A specific DE_SRAM component has been developed which inherits from the design element class of PwClkARCH and includes the power model of the memory part, both for DRAM device and

memory controller. Particularly, DE_SRAM redefined the function Update_dpow() which is basically used to calculate the power model of the unit. On occurrence of a read or write transaction, the functional model SRAM calls the SetActivity () function of DE_SRAM telling that the SRAM is active. In turn, DRAMPower is activated inside the overloaded Update_dpow() function and the average power consumption (average_power) and the average access time (mean_access_time) of the memory part are collected back by DE_SRAM.

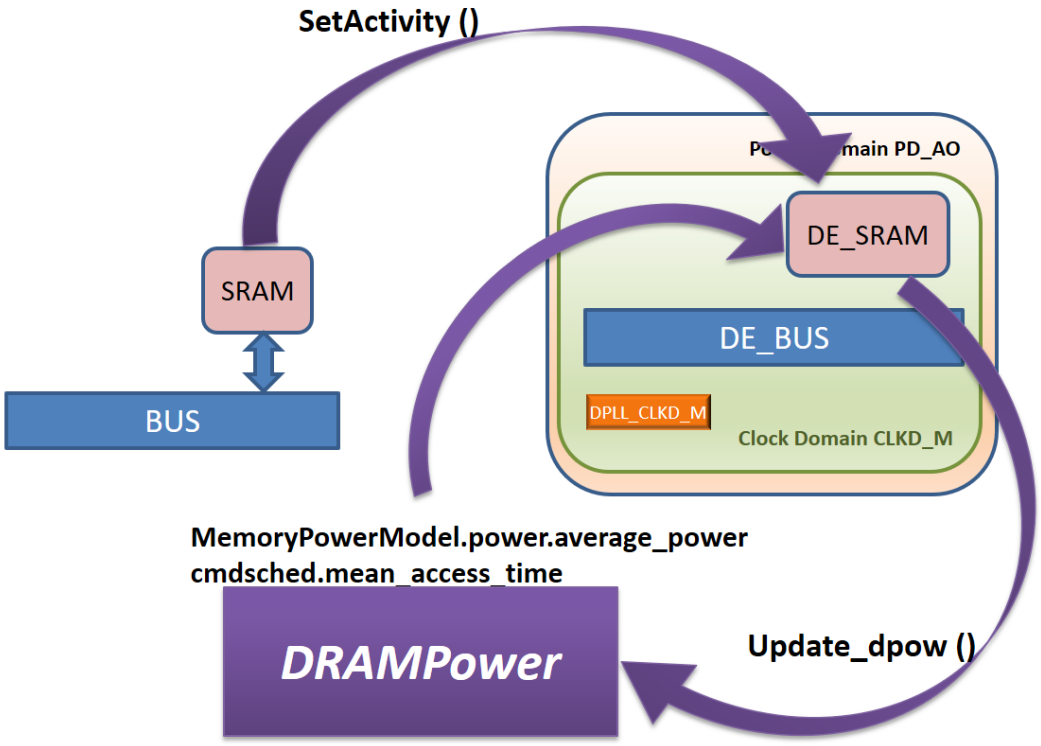


Figure 49. connection of DRAMPower with the DCT Platform

In order to evaluate with DRAMPower the power consumption of the DRAM part, we consider successive windows of transactions such that DRAMPower is activated each time a window is filled by DE_SRAM with the appropriate number of transactions.

Figure 50 illustrates how read/write transactions are grouped in a trace file by the simulation from the functional model of the DRAM. Each time a window of N transactions is stored in a file by the DRAM functional model, DRAMPower is activated. DRAMPower performs the simulation of the memory part based on the input trace file and evaluates different parameters such the average power consumed by the memory part to execute the requests in the window. In Figure 50, DRAMPower is activated after N=8 transactions and the average power consumption P_{avg_dram} associated with the first eight transactions is available after time t_8. Therefore, there is a delay (according to the SystemC-TLM global time) between the

N-1 first transactions in the window and the power value sent back by DRAMPower for these transactions, i.e. the new power value will be logged by PwClkARCH at time t_8, once all the N previous transactions are processed by the functional model. Similarly, the mean access time of the N transactions evaluated by DRAMPower is known at time t_8, this value is used as the new mean access time for the next window of N transactions. Assume a refresh operation needs to be scheduled in the first window of N transactions. Therefore, the mean access time includes the time required to execute this refresh operation and as this mean access time is passed as the memory access time for the next N transactions, the impact of the memory behavior is globally considered in the global SystemC-TLM simulation.

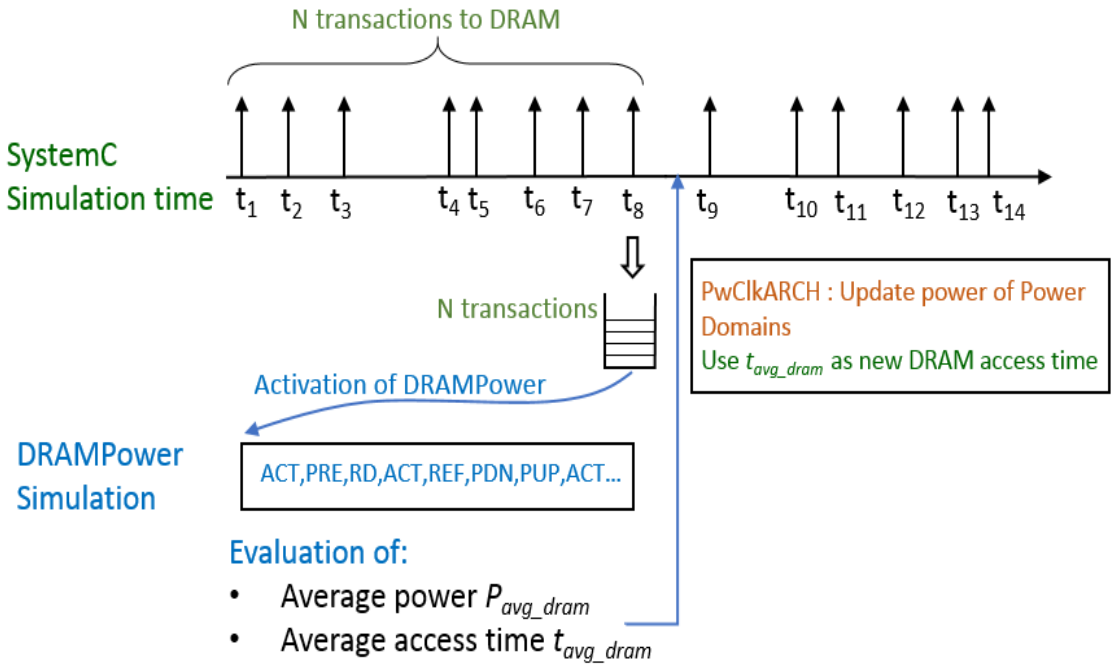


Figure 50. Connection of DRAMPower with a SystemC-TLM model augmented with PwClkARCH

Obviously, the number N of transactions should have an impact on results. If N is too large, the accuracy of the power/energy evaluation is weak, but we reduce the number of DRAMPower invocations. If N is reduced, the accuracy may be better, if N is not too small otherwise the memory controller cannot make any optimizations, but the simulation time is augmented. Among the optimizations performed by the controller, memory banks can be forced in self-refresh mode or in power down mode if these banks are not accessed. In the next section the sensitivity to N is evaluated.

3.1. Sensitivity of results to the number of transactions processed by DRAMPower

As mentioned above, read/write transactions emitted by the SystemC-TLM functional model are gathered before invoking DRAMPower to evaluate the power consumption of the memory block.

The relative accuracy of the energy estimation calculated by DRAMPower and the evolution of the simulation execution time to N are given in Figure 51. These results are generated for the memory model MICRON-16Gb-LPDDR3-1600-32bit-EDF232A2PB with interleaving disabled and DPD power mode enabled. First, we have evaluated the energy consumption of the memory with DRAMPower in standalone mode (0.136mJ) on the full trace generated by the SystemC-TLM model. The energy consumption of the memory when the SystemC-TLM model is connected with DRAMPower is, anyway, lower than 2% when N>5. The energy accuracy plotted in Figure 51 is relative to this range. Similarly, the simulation time is relative to the case N=5. In this example the value of N has a great impact on the results. We notice that a value of N=10 or N=20 is a good compromise since the accuracy is larger than 0.9 and the simulation time is lessened significantly (about 40%). All the experiments developed in Chapter VI are done with N=10.

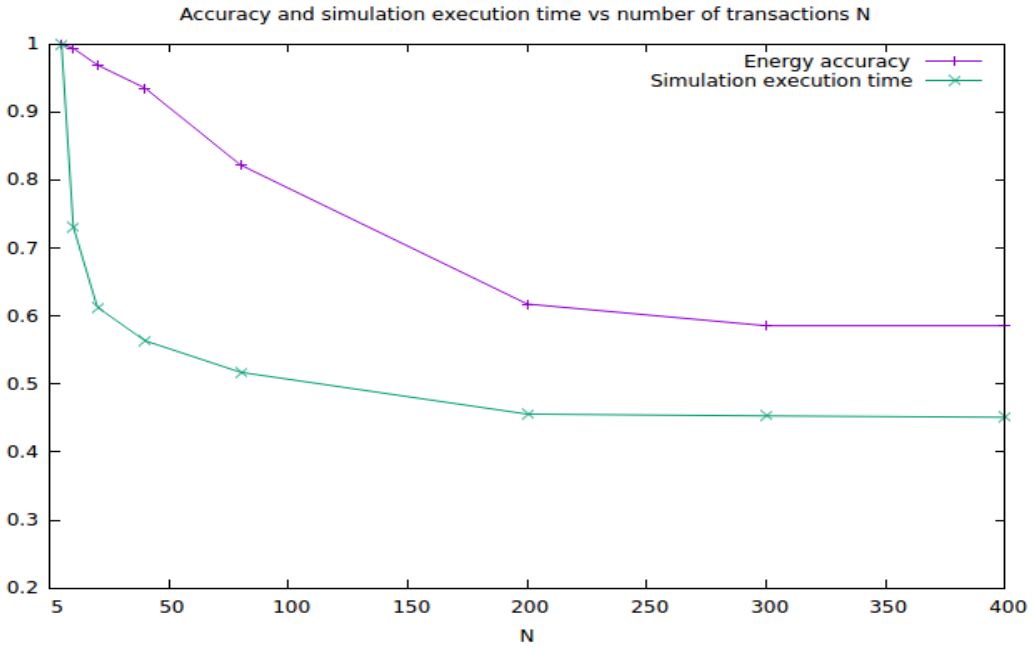


Figure 51. Accuracy of energy estimation and the simulation time evolution vs Number of transactions

3.2. DRAM device power evaluation

The power evaluation on a window of N transactions by DRAMPower poses the problem of managing the time between the two simulators and as a consequence the power evaluation of the DRAM part. Consider the example shown in Figure 52.

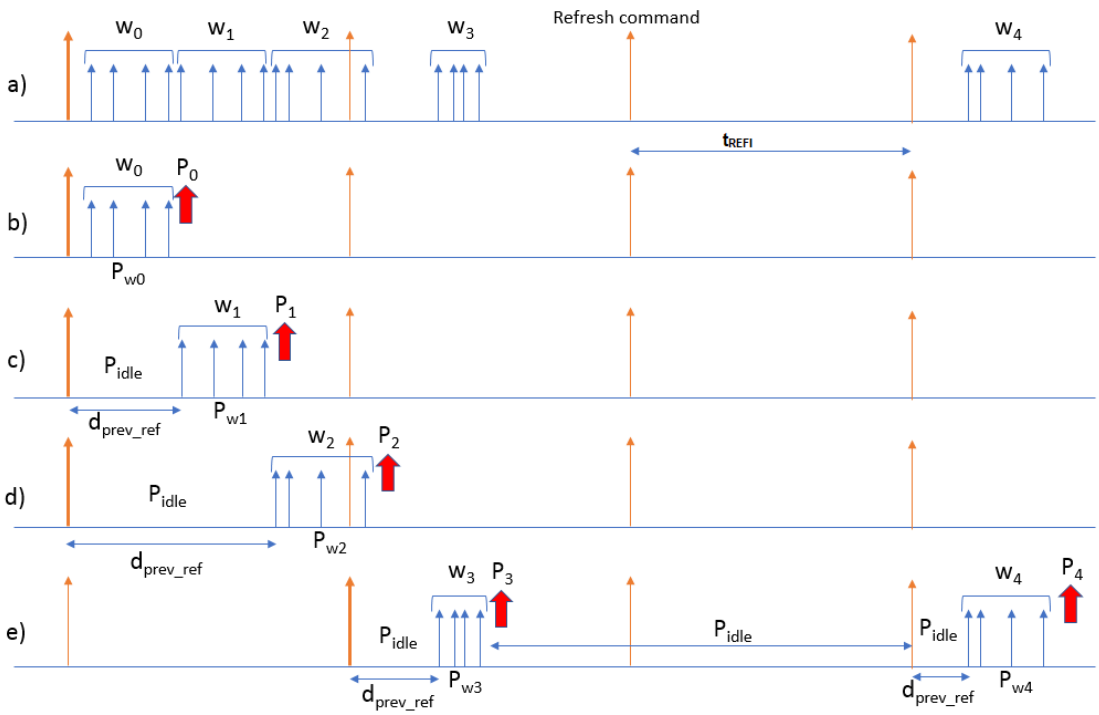


Figure 52. Example of power evaluation of groups of N transactions

The line a) describes a sequence of read/write transactions (vertical blue arrows) grouped in windows (w_0, w_1, …, w_4) of N=4 transactions. The horizontal axis denotes the simulation time. The memory controller translates these read/write transactions into internal memory commands (not represented in the figure) and periodically inserts a refresh command (orange arrows) according to the period t_{REFI} depending on the SDRAM characteristics.

We start at time 0 with window w0. The trace file associated with w0 is transmitted to DRAMPower (line b) which evaluates the average power $P_0 = P_{w0}$. DRAMPower returns the average power $P_{avg\_dram}=P_0$ to the design element DE_SRAM which refer to the memory block in the functional model. Consider now the line d). If we fix a timestamp equal to 0 to the first transaction of w_2, the normal refresh command will not be inserted since the delay between two consecutive refresh commands is t_{REFI}. To take into account the refresh operations that can be triggered inside a group of N transactions, we fix to the first transaction a timestamp equal to d_{prev_ref} which is the time interval since the last refresh operation. Therefore, in the processing of the successive windows of transactions by DRAMPower, the first transaction in a window has a timestamp relative to the last occurrence of the refresh command. In the case d), DRAMPower returns the average power P_2 which combines the power P_{idle} consumed during d_{prev_ref} and the power P_{w2} resulting from the transactions inside w_2, including the refresh operation. However, only P_{w2} should be returned to DE_SRAM. Therefore, to evaluate the power of a window P_{wi}, we operate in two steps. In a first step, DRAMPower is

executed on a trace composed of a single transaction corresponding to the first transaction of the window w_i with a timestamp equal to d_{prev_ref}. We get back an average power equal to $P_{single} = P_{idle} + P_T$ where P_T is approximately the power consumed by a single transaction. In the second step, DRAMPower is executed on the trace w_i and the average power returned by DRAMPower is $P_N = P_{idle} + N*P_T$. The value of P_{wi} is then approximated by:

$$P_{wi} = (P_N - P_{single})*N/N-1$$

This approach for evaluating the power P_{wi} does not take into account the state of the DRAM after processing the transactions from window w_{i-1}. However, if N is not too small (e.g. N= 20), the error is relatively negligible.

Consider now the case of line e). For periods larger then t_{REFI}, no transactions are emitted to the memory. As the trace for w_4 has a timestamp d_{prev_ref} relative to the last refresh operation, the power consumed by the memory in the interval where no transactions are emitted is not evaluated. This idle period can be identified in the functional model of the DRAM: an event is scheduled with a delay of t_s after the last transaction of w_{i-1} occurring at t_{last_wi}. If in the interval $[t_{last\_wi}, (t_{last\_wi} + t_s)]$ no transactions are received, the power consumption of DE_SRAM is fixed to P_{idle} which is in this case a constant value depending of the DRAM characteristics and of its power mode. As soon as a new transaction is received by the DRAM memory, the power model in DE_SRAM commutes from the constant value P_{idle} to the power model using DRAMPower.

3.3. DRAM controller power estimation

We consider a very simple power model of the memory controller. We assume that P_{crtl_max} is the dynamic power consumption of the memory controller when it is exercised at its maximum data bandwidth ($Gbps_{max}$). $Gbps_{max}$ is estimated by: $Gbps_{max} = 2*t_{ck}/BL$ where BL is the maximum burst length provided by the memory and t_{ck} is the clock period of the memory controller (the maximum bandwidth is achieved for burst accesses issued each BL/2 cycles with a double data rate memory). The dynamic power consumption of the memory controller over a window of N transactions can be evaluated by:

$$P_{crtl} = P_{crtl\_max} * Gbps / Gbps_{max}$$

Where Gbps is the bandwidth evaluated for each window of N transactions. For example in Figure 53, the bandwidth Gbps associated with the second window is $N/(t_{16} – t_8)$ if a single read or write access is done at each transaction.

The power P_{crtl} is then cumulated by the design element DE_SRAM with the power returned by DRAMPower to provide the total power consumption of the memory sub-system for each window of N transactions:

$$P_{memory} = P_{wi} + P_{crtl}$$

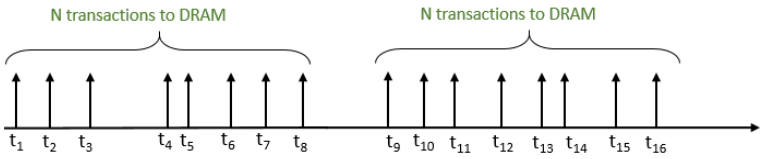


Figure 53. Example of two successive windows of N transactions

3.4. Mean memory access time estimation

As mentioned above, the memory mean access time is evaluated inside DRAMPower. Figure 54 illustrates an example of read/write commands translation to DRAM operations sequence such as ACT, PRE, REF (small blue arrows in the figure).

Let t_1 be the instant where the first transaction inside the window is emitted. If the corresponding read/write command is scheduled at t_{c1}, the access time for this first transaction is obviously $t_{a1}=t_{c1}-t_1$, without forgetting of course to add the read latency or write latency delay.

Consider now the case of the transaction at time t_5 whose corresponding read/write command is scheduled at t_{c5}. In the time interval t_{c4} - t_5 the DRAM is operating the commands associated with the transaction emitted at time t_4. Thus the access time for the transaction occurring at t_5 is calculated using $t_{a5} = t_{c5}$ - Max (t_5, t_{c4}).

By repeating the evaluation of each access time over the transactions inside the window, the mean access time for the N transactions is then the average of t_{a1}, …, t_{aN}.

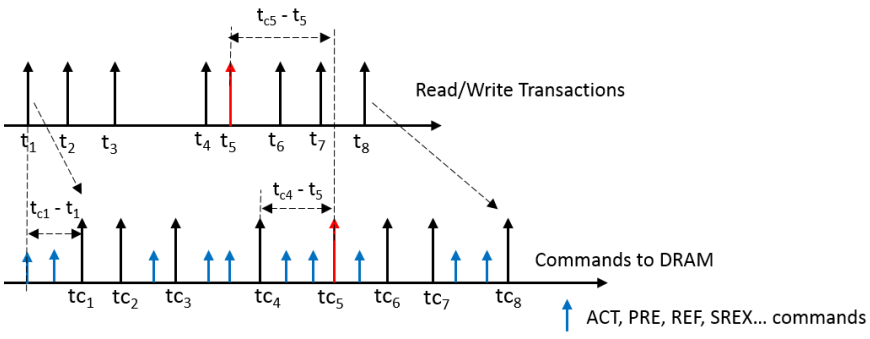


Figure 54. Example of DRAM operations generation

3. Conclusion

In order to apply this connection between PwClkARCH and DRAMPower on the Intel environment, we have developed the necessary. First, an analyzer port was added to the memory controller that allows us to extract transaction traces (Figure 55). We generated the same format requested by DRAMPower using a script. Second, as the memory used in the Intel platform is an LPDDR4 memory, we drawn up the memory specification file needed for DRAMPower tool. Note that DRAMPower doesn't support basically a LPDDR4 memory model and doesn't include any associated specification file. Notice that apart the values of their internal parameters, the main difference between LPDDR3 and LPDDR4 is that LPDDR4 provides a double channel, each one having a 16 bits width, while LPDDR3 includes a single channel only. But in the Intel platform, the two channels of the LPDDR4 are grouped to form a single channel (performance model) allowing DRAMPower to be used in this case to simulate a LPDDR4 technology. However, the values of the parameters of a LPDDR4 depend on the memory device vender. As the LPDDR4 memory device in the Intel platform is provided by a third-party vendor, we didn't have the access of the code. So the only manipulation that we did was to generate a command trace file from our simulation of the Intel platform and to run the DRAMPower tool as standalone using the LPDDR4 memory specification file that we prepared.

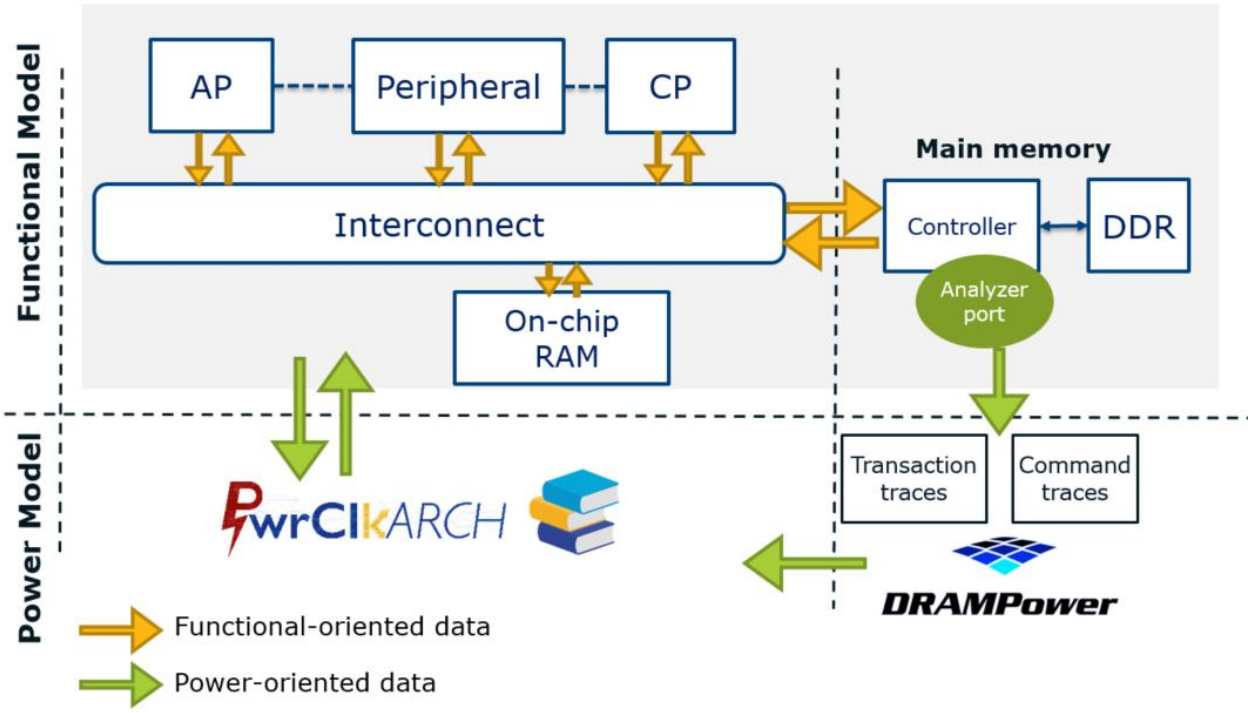


Figure 55. Framework representation with Intel platform

VI. Simulation results

In this Chapter, we present the results of our simulations. We start by giving the results that we obtained after applying clock gating power management techniques on the Intel Sophia Antipolis platform. The second part is reserved for our work done on the memory system exploration at Transactional Level using SystemC-TLM-based platform of the LEAT, and finally we conclude the chapter by giving the entire framework of our methodology that we propose in this project.

1. Clock Gating application

In the beginning, we are interested in proving that PwClkARCH library can help in defining the power model of the intel pre-silicon modem SystemC-TLM-based platform, and in controlling the energy dissipation. We define the power model of a part of the platform (L2 Coprocessor) in Chapter III.3, by connecting the power library and the functional implementation via the PMU block. Triggering a power mode change is a power management request represented by a TLM transaction from an initiator in the platform. In our case, the scheduler is the initiator. The request could be a reference to the OPP table line, or a particular state of a particular hardware block (a Tile for example). The library insures both types of control.

Thus, we add an initiator socket “Out_PM” in the scheduler interface that we connected to the target initiator of the PM, “In_PM”. TLM transactions in corresponding functions of the scheduler are added to send data (Tile name, Tile state = 0 means Inactive) each time a task is finishing to running on a Tile, or (Tile name, Tile state = 1 means Active) each time a task starts running on a Tile. We apply the clock gating technique by using the “Set_Clk_State_DE ()” function implemented in the PM of the PwClkARCH library. When triggered, it retrieves the DE name requested, and the division factor to be applied by the CM to generate the clock frequency of that DE. For clock gating, the factor should be equal to zero. The application/use-case used in these experiments is LTE User Datagram Protocol (UDP) downlink and uplink traffic.

Using PwClkARCH, we plot the frequency variation per Tile, as illustrated in Figure 56, after applying clock gating on each Tile according to its active/inactive state. It can be observed that at the beginning of each time slot, all the Tiles are busy, so their clock frequency is set up to the maximum value. As soon as a Tile switches to an idle state, its clock is gated. The frequency becomes no more constant and depends on the state of the Tile: active or inactive. This has a direct impact on the dynamic power consumption of each Tile and in the same way of the L2 Coprocessor block as presented in Figure 57. We can see that in time slots where all the Tiles are inactive, the dynamic power consumption is equal to zero. In Figure 56, the maximum clock frequency of Tiles is equal to 288 MHz, with the chosen clock frequency start time, both periodicity and deadline values used in the task graph can be well met (to be explained later on Tile 1). When reducing the maximum clock frequency value in order to define the optimal value regarding power dissipation and

performance, the dynamic power is reduced (green curve : 250 MHz; blue curve 200 MHz Figure 58), however there is a violation of the deadline of the tasks assigned to some Tiles. In Figure 59 we plot only the clock frequency variation of the Tile 1 with the three maximum values 288, 250 and 200 MHz to show the violation of the deadline constraint (red lines). Obviously reducing further the clock frequency to 200 MHz (blue curve in Figure 58) improves the power consumption but tasks on three Tiles go beyond their deadlines.

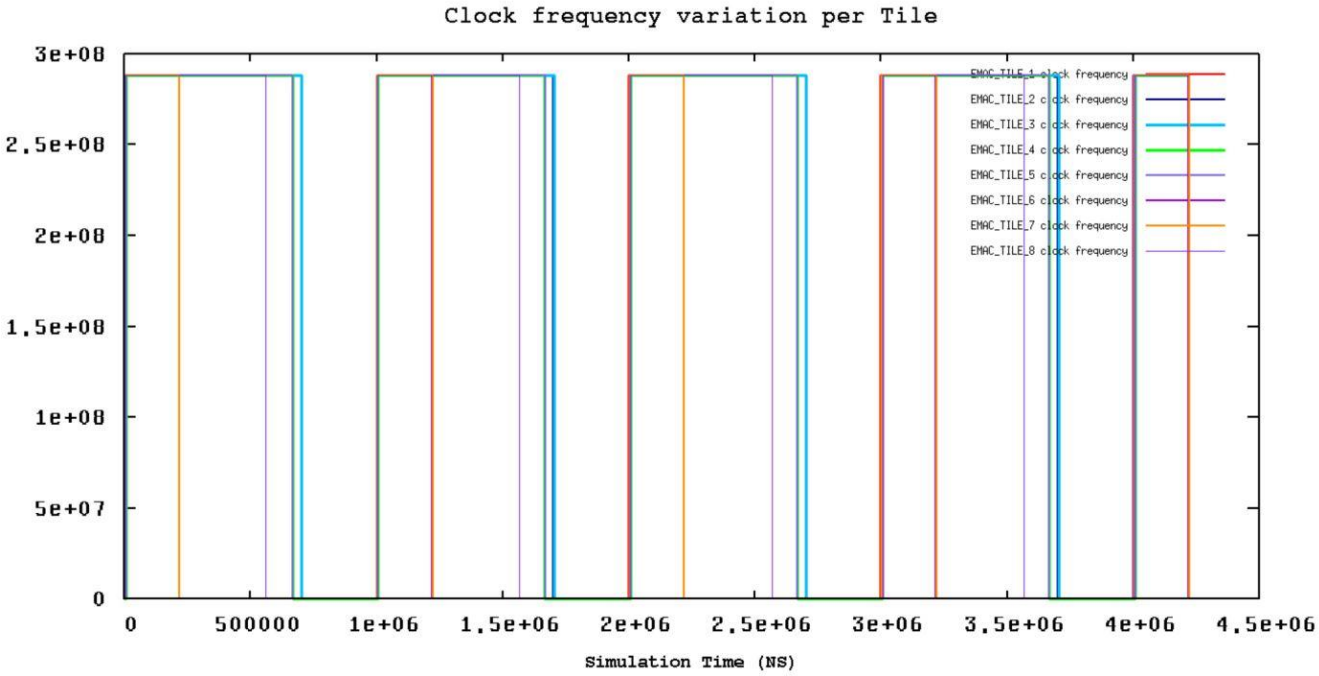


Figure 56. Clock frequency variation per Tile (x=time [ns], y=Frequency [Hz])

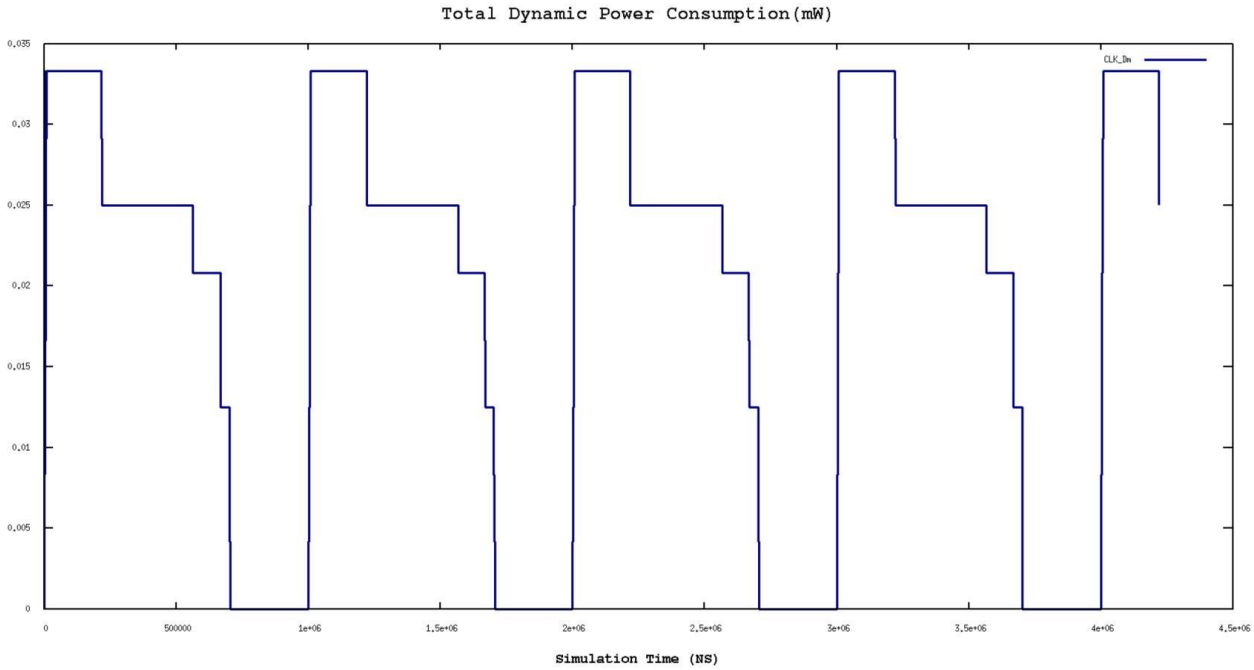


Figure 57. Total dynamic power consumption of all Tiles with (x=time [ns], y=Pdyn [mW])

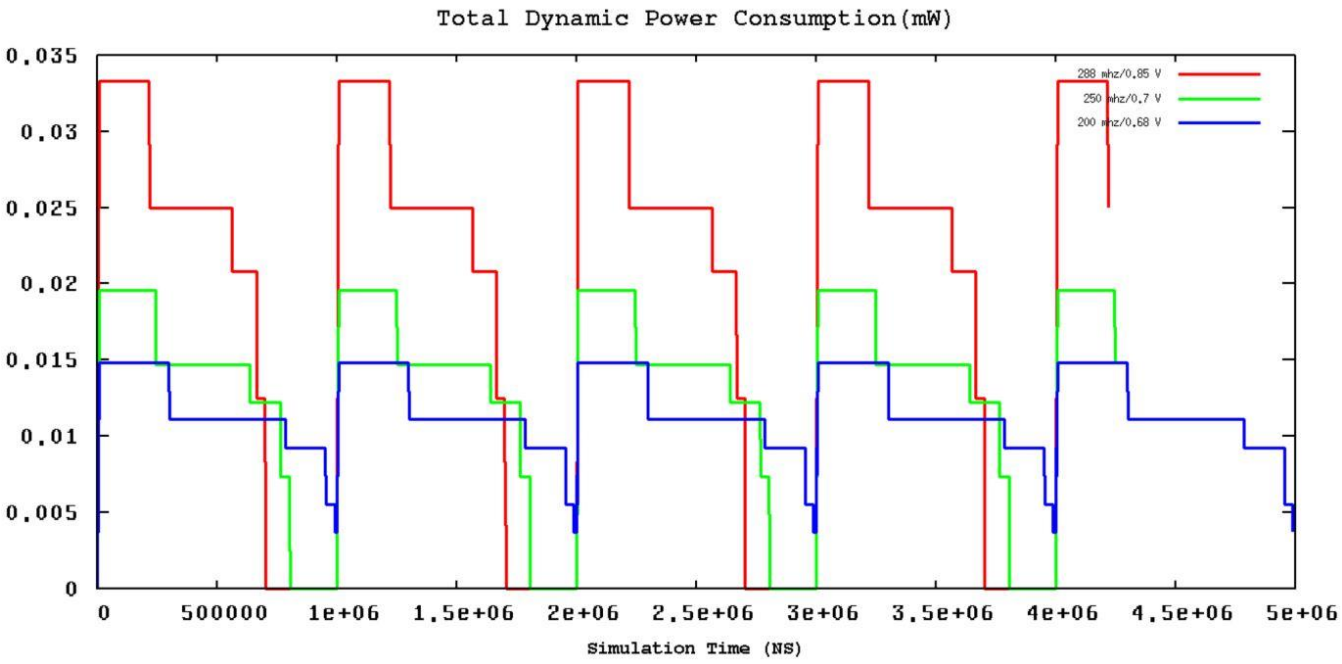


Figure 58. Total dynamic power consumption of all Tiles with different clock frequency values (x=time [ns], y=P_{dyn} [mW])

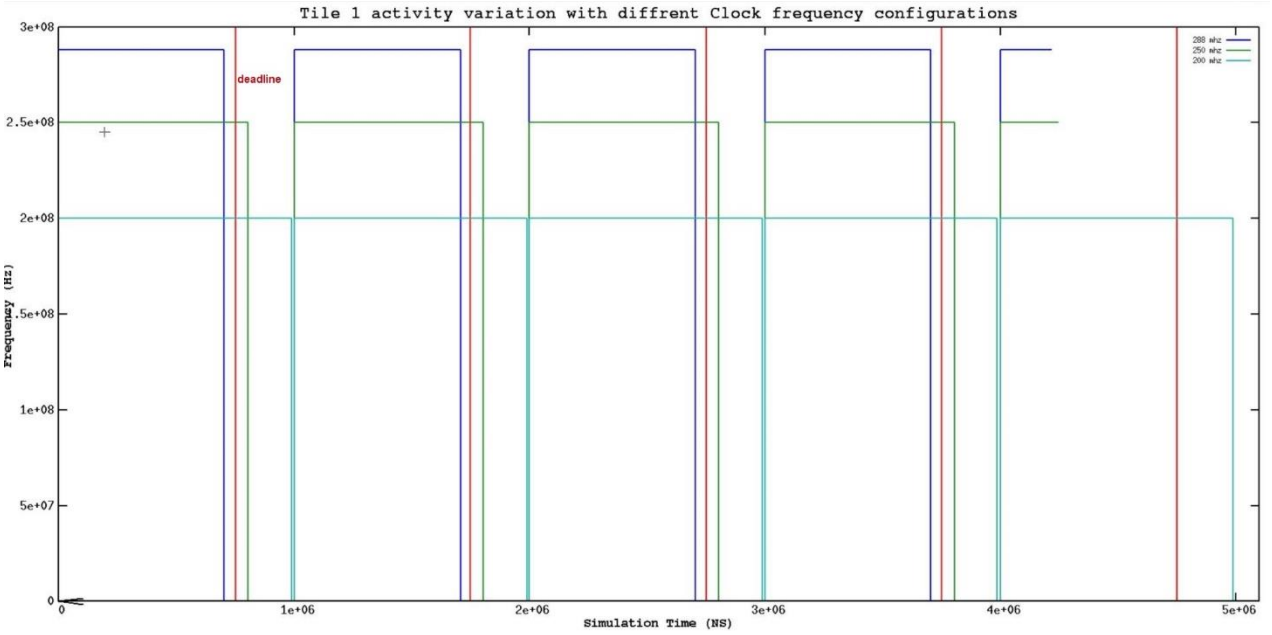


Figure 59. Clock frequency variation of Tile1 with three configurations

In the Figure 60 the information extracted from the energy versus time data are plotted. The results show that the overall energy consumed by the Tiles is strongly correlated with the clock frequency.

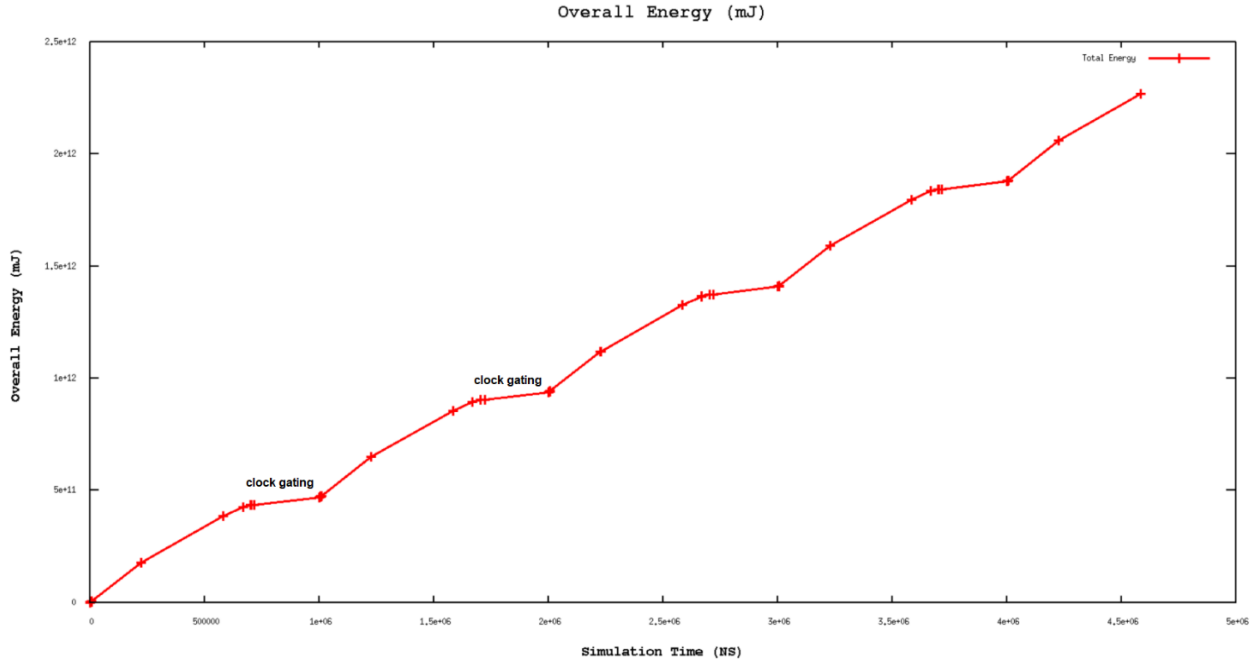


Figure 60. Overall energy (x=time [ns], y=energy [mJ])

We did the same work on the Munich pre-silicon environment in order to connect the PwClkARCH library with the performance model. Figure 61 illustrates the clock frequency variation per Tile after applying the

clock gating technique. In this experimentation, we used the task graph presented in paragraph III.2. in order to show that the power model is in total coherence with the execution of the tasks on the Tiles. Take the example of Tile 3, which is the simplest to check, it performs only the task DL_MOVE_0_3. This task is non-periodic. In the Figure 61, we can see that the frequency associated with Tile 3 (blue curve in bold) is set up to the maximum value, and then it is gated until the end of the simulation. The other Tiles perform more than one task, with probably a delay between the tasks due to the dependence, the curves show that the behavior obtained is the one expected.

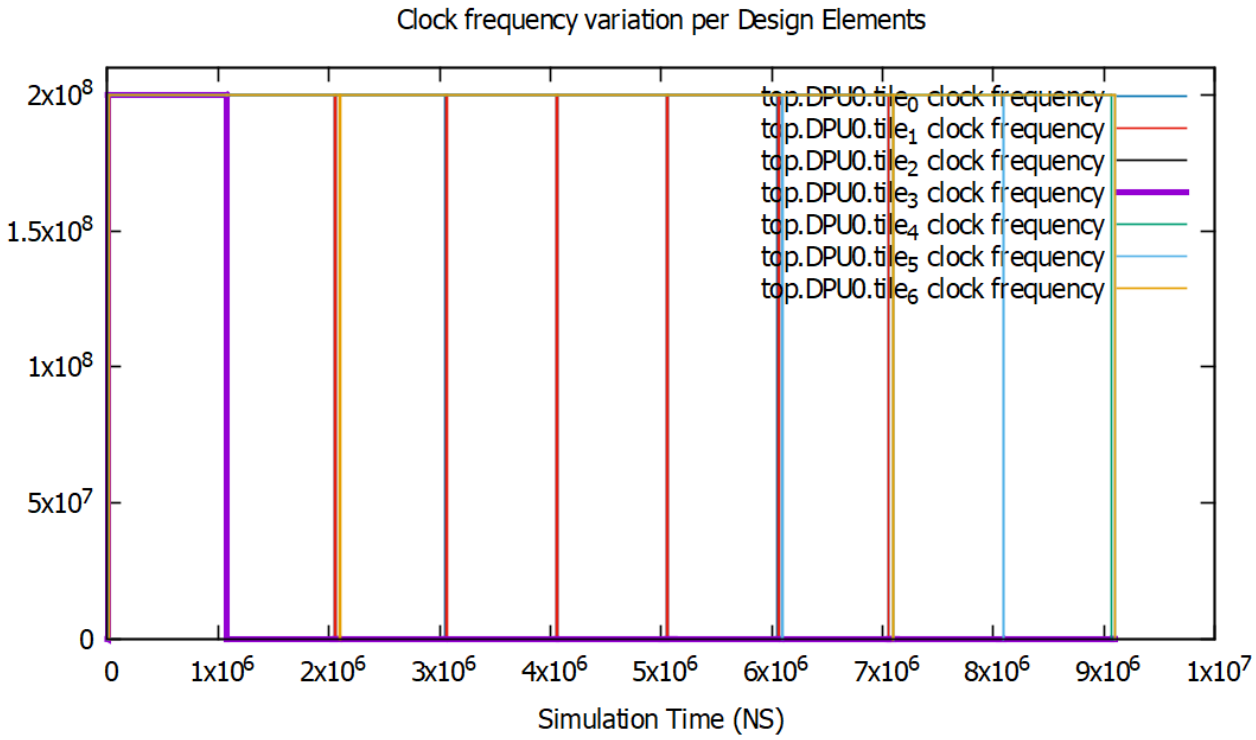


Figure 61. Clock frequency variation per Tile (x=time [ns], y=Frequency [Hz])

This experiments illustrate that such approach helps also on defining the best strategy of power management, here limited to clock gating applied to Tiles, associated with the functional activity of the Tiles (due to the scheduling of tasks on Tiles). As soon as the performance constraints are met, we can deduce the expected power profile generated by the system. If more than one scheduling algorithm is available and they all show similar performance, then power figure becomes the key parameters for the final decision on which one is to be picked.

We need to tune our power model and to include a precise memory system power model. Results show that this is feasible and there is an interest in doing so.

The next parts of this chapter focus on the introduction of a memory model within our power model, as explained in chapter V, which allows a global analysis of the system, in term of performance and power, knowing that the power strategy influences the frequency of the memory requests.

2. Datasheet based memory specification

In this paragraph, we define datasheet memory specification delivered by memory vendors. These datasheets include many parameters and specifications, from the size and the technology, to timing and output performance. Researchers have relied on current specifications that vendors provide for each DRAM part, which are known as IDD values, to estimate the power consumption of the DRAM. Usually, these datasheets are confidential, and especially for new technologies (LPDD4, MRAM, etc.), which makes our work difficult. For the exploration of memory technologies including the STT-MRAM LPDDR3 model, we used the values published in [64]. DRAMPower memory specifications are based on Micron Technology datasheets, but the format is simple and we can integrate the specification values of any vendor.

Table 1: Comparison of DRAM and MRAM LPDDR3 devices

Parameters	LPDDR3 DRAM		LPDDR3 MRAM	
Clock	533 MHz		533 MHz	
Page size	4 KByte		256 Byte	
Bank bit	BA2-BA0		BA2-BA0	
Row bit	R13-R0		R17-R0	
Column bit	C9-C0		C5-C0	
tREF	3900 ns		N/A	
tRCD	10 cycle		13 cycle	
tRL	8 cycle		6 cycle	
tWL	4 cycle		4 cycle	
tRP	10 cycle		7 cycle	
tRC	32 cycle		18 cycle	
tRTP	4 cycle		2 cycle	
tRRD	6 cycle		6 cycle	
tCCD	4 cycle		4 cycle	
tWTR	4 cycle		4 cycle	
tWR	8 cycle		14 cycle	
tFAW	27 cycle		27 cycle	
tRFC	70 cycle		N/A	
Voltage	VDD1 (1.8 V)	VDD2 (1.2 V)	VDD1 (1.8 V)	VDD2 (1.2 V)
IDD0	7.8 mA	28.3 mA	4.8 mA	41.2 mA
IDD2N	1.8 mA	3.7 mA	1.8 mA	3.7 mA
IDD2P	1.8 mA	1.7 mA	1.8 mA	1.7 mA
IDD3N	3.5 mA	6.9 mA	3.5 mA	6.9 mA
IDD3P	3.5 mA	6.9 mA	3.5 mA	6.9 mA
IDD4R	3.5 mA	140.4 mA	0 mA	170.9 mA
IDD4W	3.5 mA	145.4 mA	0 mA	267.2 mA
IDD5	23.8 mA	78.2 mA	1.8 mA	1.7 mA
IDD6	1.0 mA	3.8 mA	0.0 mA	0.0 mA

Figure 62. LPDDR3 DRAM and MRAM LPDDR3 timing and power parameters [64]

3. STT-MRAM LPDDR3 interface model

For these experiments we consider the DCT-platform introduced in Section V.2. In this work, to showcase the methodology, only DVFS is applied to CPU and DCT, but our approach is general enough that also other strategies could be equally applied.

The trace file of the N read/write transactions to the memory is constructed by the memory functional model. Once the N transactions are stored in the trace file, the power model developed in DE_SRAM is activated.

Three LPDDR power modes are considered in the experiments, all of which are represented in the DRAMPower tool as options to be set:

- No low power mode (#0)

- Deep Power Down mode (DPD) (#1)
- Self-refresh (#2)

Nine memory configurations, numbered from #0 to #8, are tested:

- ISLPED2014-4Gb_LPDDR3-1066_32bit (#0)
- MICRON-2Gb_LPDDR-266_16bit-A (#1)
- MICRON-4Gb_LPDDR3-1333_32bit-A (#2)
- MICRON-16Gb-LPDDR3-1600_32bit-EDF232A2PB (#3)
- MICRON-2Gb_LPDDR2-800-S4-16bit-A (#4)
- MICRON-4Gb_LPDDR3-1600_32bit-A (#5)
- MICRON-2Gb-LPDDR2-1066-S4-16bit-A (#6)
- MICRON-2Gb-LPDDR-333-16bit-A (#7)
- ISLPED2014-4Gb-MRAM-1066-32bit (#8)

The memory models #0 and #8 are taken from [64], all the others are selected from the database of the DRAMPower tool.

Some minor modifications are introduced in DRAMPower. These modifications are all included in the files CmdScheduler.cc and CmdScheduler.h in the DRAMPower/src directory. These modifications consist in:

- computing the mean access time over the transaction in the trace file,
- storing in a variable the date of the first transaction (which is used to estimate the average power of a window of N transactions by DE_SRAM).

3.1. No Interleaving of memory accesses

First, we run simulations disabling bank-interleaving of memory accesses. Normally the impact of having a higher interleaving factor is a higher bandwidth, but as well a higher power consumption due to the parallel memory bank accesses. Figure 63 (a) represents the execution time in millisecond (ms) and Figure 63 (b) the energy consumption in millijoule (mJ) of the overall platform. The colored bars represent the nine different memory models and for each one of them the three power models taken into consideration are used.

Results show that the lowest energy consumption is achieved using the eighth (bar #8) memory model, which represents the MRAM technology, with 1.78 mJ. However, the lowest execution time is achieved using the third (bar #3) memory model when it is used in self-refresh power mode (#2) with 6.31 ms.

Disregarding of the LPDDR model taken into consideration, the low power modes DPD and self-refresh have a very limited impact on the execution time of the DCT application. However, they provide significant power savings in most cases. The difference in energy consumption between the DPD mode and the self-refresh mode are sometimes very small, e.g., in the case of bar #7.1 and bar #7.2.

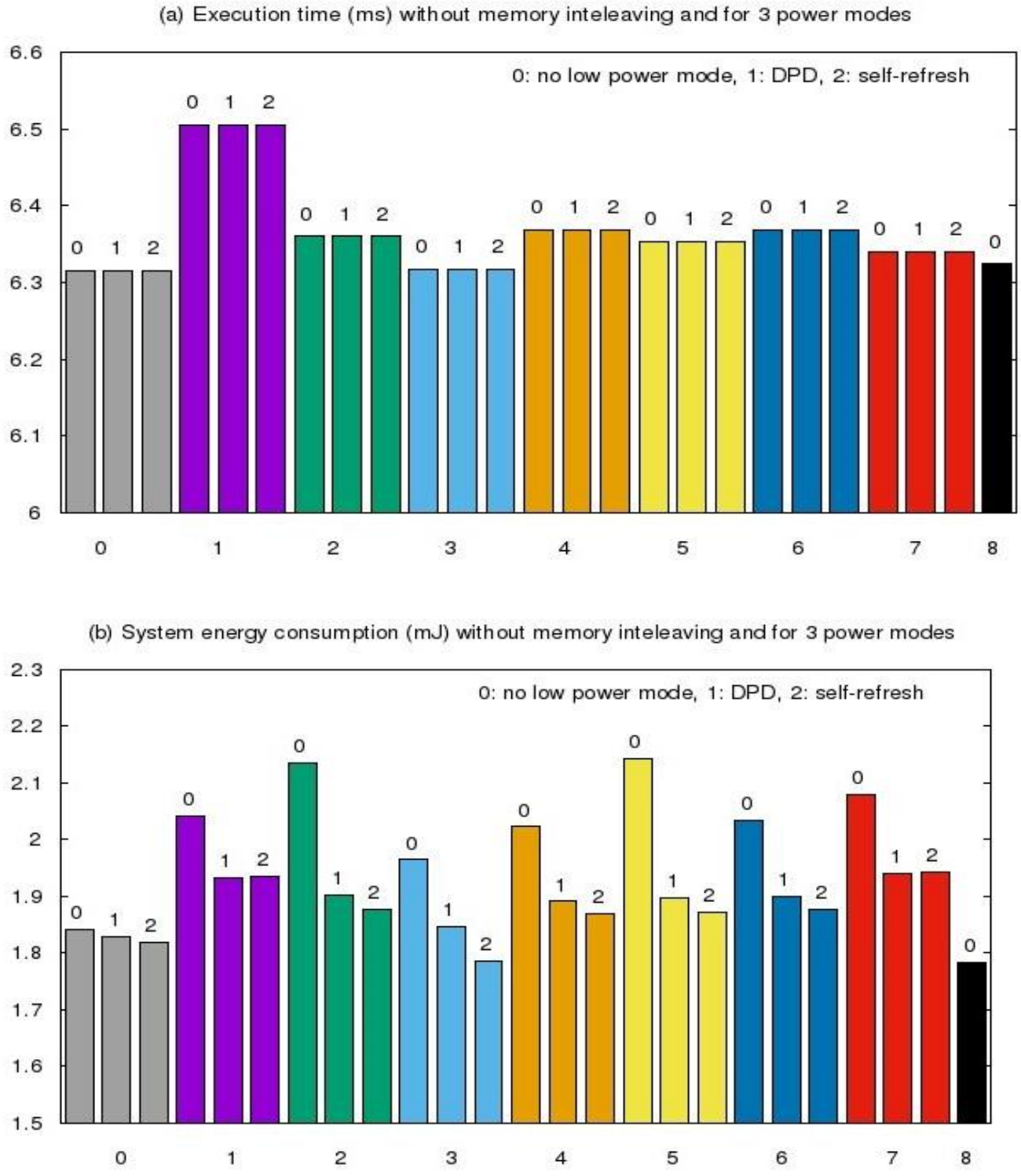


Figure 63. Execution time (a) and energy consumption (b) of the DCT platform with bank-interleaving disabled

3.2. Interleaving of memory accesses

Same simulations are run enabling bank-interleaving of memory accesses, as shown in Figure 64, thus allowing parallel activation of two banks. The memory model #3 provides the best results in execution time and energy consumption, when used in self-refresh mode (bar #3.2). The MRAM used in this study provides good results but it is less efficient than the memory model #3 (bar #8.0). Data in the DCT example are accessed in memory with an incremental addressing scheme with an equal number of write and read accesses. Due to the lower write energy consumed by the LPDDR3 and the low power consumption in self-refresh mode, the LPDDR3 behaves better than the MRAM memory. This can be explained also by the fact that when enabling bank interleaving, the number of refresh commands decreases since the application execution time is reduced. Thus, MRAM becomes less important comparing to the LPDDR memory. On other access schemes with non-incremental addressing, results could be different.

The experiments show the benefit of performing a full system simulation when more accurate performance results are looked for. For example, let us compare the power model #6 in Figure 64 and Figure 63, i.e. with interleaving enabled and disabled, respectively. Without interleaving and with no power mode applied (Figure 63 (a)), the execution time results in 6.36 ms when the complete system model is executed in simulation. We use the complete trace file of read/write transactions generated by this example as input to DRAMPower and using the same memory configuration than that in the full system simulation. In the same setup, the DRAMPower provides an estimate of the trace length to 6.27 ms, giving an error lower than 2% compared with the full system simulation. If we use the same trace file in DRAMPower with no power mode enabled, but with bank-interleaving enabled (see Figure 64 (a)), the trace length results the same, but the execution time given by the full system simulation is reduced to 5.96 ms (i.e. about 6%). This example shows that the memory sub-system impacts the timing of the rest of the system, which in turn impacts the read/write transactions to the memory generated by the system.

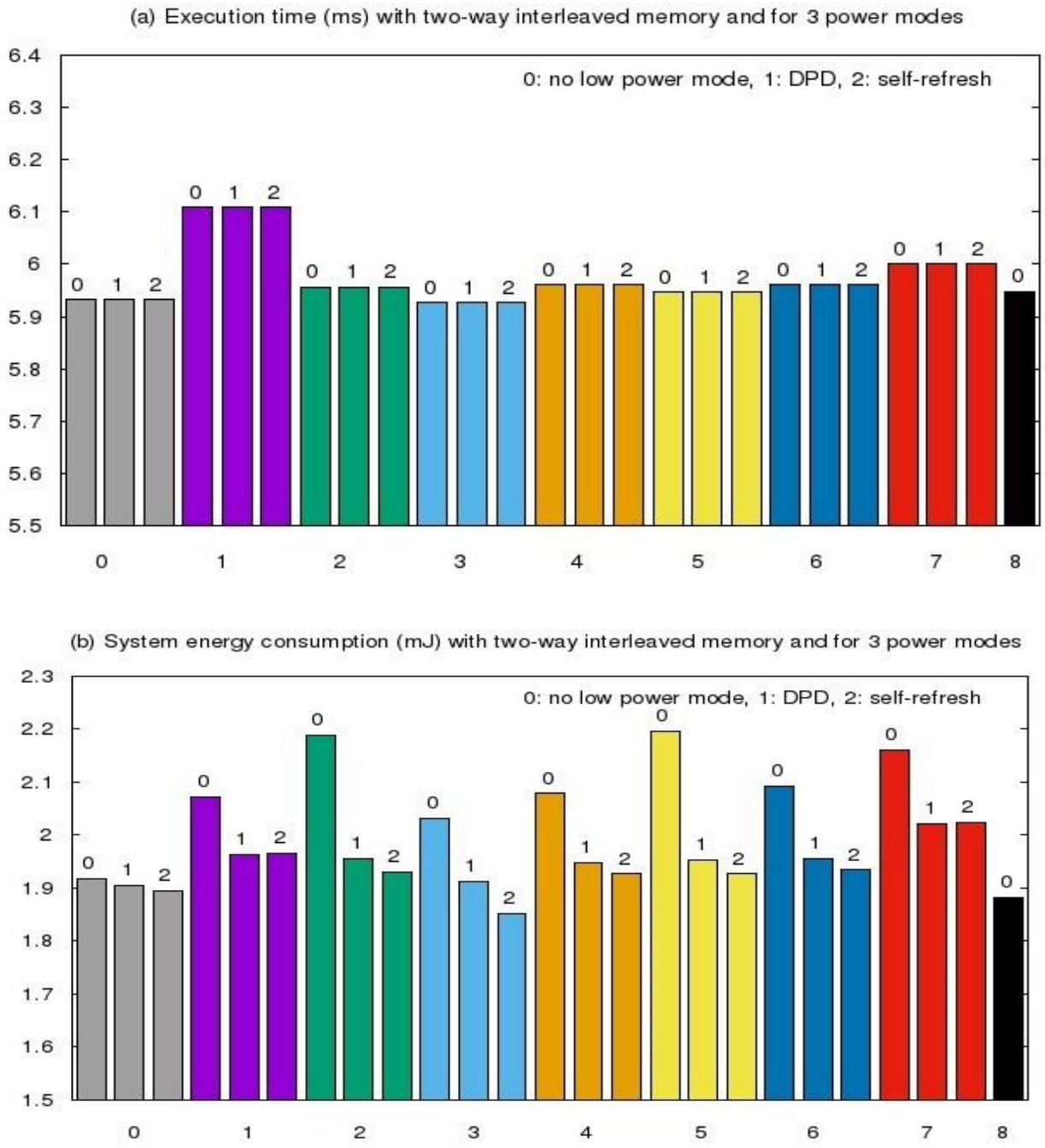


Figure 64. Execution time (a) and energy consumption (b) of the DCT platform with bank-interleaving enabled

The example above shows that it is necessary to model this mutual impact if we want to have a correct analysis of power and performance.

3.3. Applying DVFS

When the CPU transfers data from the memory to the DCT unit, a lower frequency and lower voltage operating performance point (OPP) is applied, compared with the previous case: the frequency of the CPU is divided by 4, the voltage is reduced from 1.4V to 1.15V, and the frequency of the DCT is set to ¼ of its maximum frequency. The time penalty to change dynamically the frequency is 0.2ms. When returning to the initial OPP, the time required to rise up the voltage back to its initial value is 0.07 µs per mV.

To evaluate the impact of DVFS on power and performance, we select from the list given in section IV.A, the following LPDDRx memory models: #0, #3, #4, #7 and #8.

Figure 65 represents the results of applying DVFS on the execution time (a) and the energy consumption (b) with bank-interleaving disabled. Obviously, execution time when DVFS is applied are greater than those when the maximum frequency is unchanged while the system is running (Figure 63 (a)). In the case of no low power mode of the memory (LPDDRx) enabled (bars #x.0), the total energy consumption is greater than that obtained without applying DVFS. However, when power mode 1 or 2 are enabled, some energy savings can be obtained with DVFS. These results illustrate that applying DVFS gives mixed results w.r.t. energy saving. For a too short time of processing, it can be counterproductive. Thus it is important to choose carefully memory technology and power management strategies to get energy savings. In this example, the MRAM provides the best results because of its very low static power consumption when in idle mode.

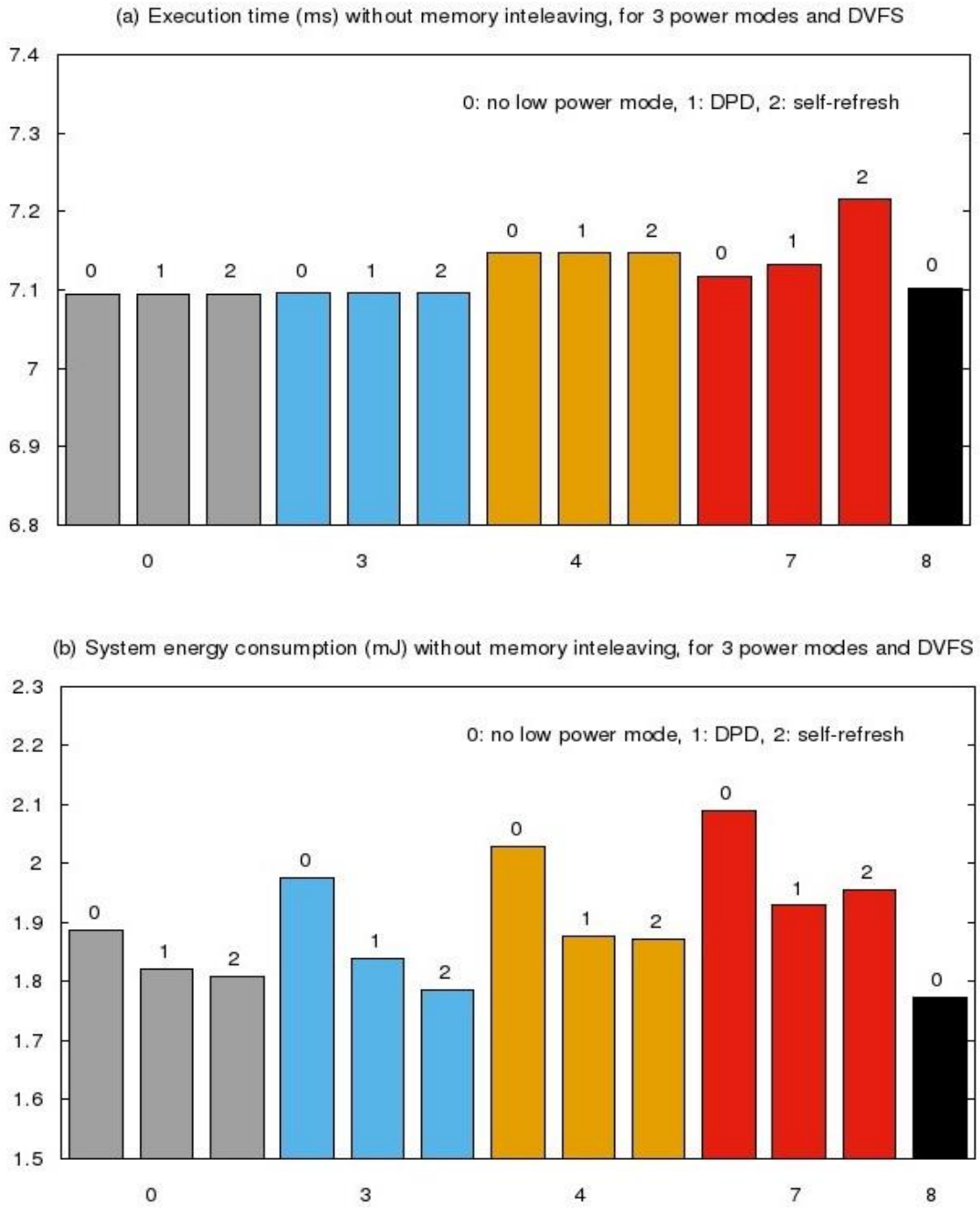


Figure 65. Execution time (a) and energy consumption (b) of the DCT platform when applying DVFS, with bank-interleaving disabled

When enabling bank-interleaving (Figure 66), the results show that whatever the memory model, the energy consumption of the system with DVFS is always greater than that without DVFS. The reason is the reduced execution time when the bank-interleaving factor is enabled, which gives more relative importance on the

time penalties induced by DVFS (0.2 ms in the example) whose durations remain constant. While the voltage and frequency are changing, all the supplied units consume some energy, including the memory, and this wasted energy is not compensated by the gain provided by the low power OPP. This behavior is well known and formulated through the break-even time [77] which represents the minimum length of an idle period to save power. Thus, the time required to change power supply voltage or clock speed have to be taken into account. If state changes take long enough, the power lost during the transition may be greater than the savings given by the state change.

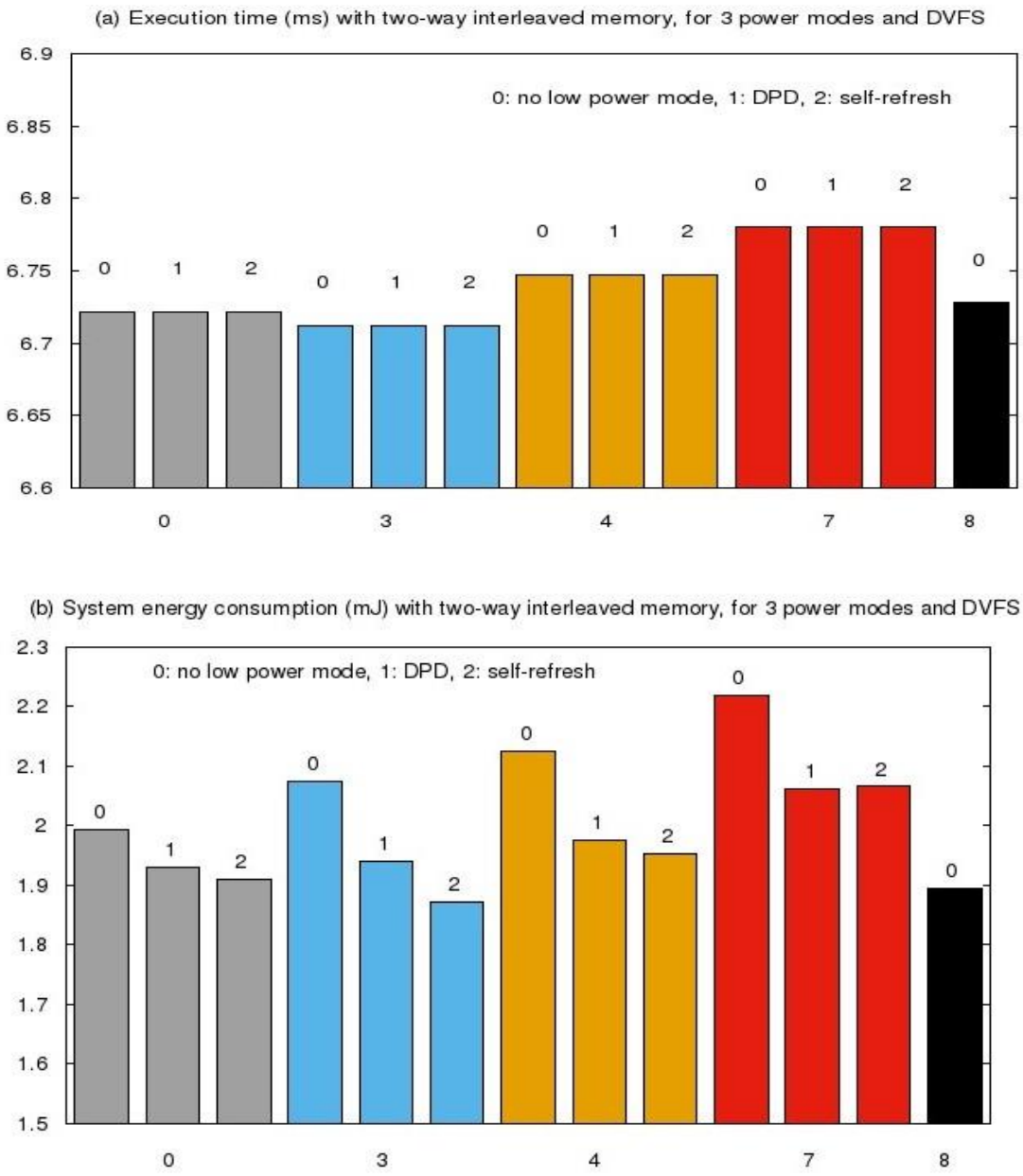


Figure 66. Execution time (a) and energy consumption (b) of the DCT platform when applying DVFS, with bank-interleaving enabled

4. Conclusion

This chapter presents firstly the results of our work on providing a joint performance-energy consumption simulation environment. Then, we prove that the coupling of DRAMPower tool with a SystemC-TLM functional model on which the power-oriented overlay PwClkARCH is added, provides an efficient framework able to evaluate the impact of various DRAM memory types on the whole system performance

and power consumption. This framework helps making early decisions on the choice of the most suitable architecture (memories size, bandwidth, frequency, etc.) and of the scheduling strategies in mobile SoCs.

VII. General conclusions and perspectives

1. Conclusion

Thanks to the publication of the OSCI TLM2 standard [13], SystemC took a large place to ease system exploration and design. Simulation is the predominant technique in hardware design. It helps on identifying performance problems and power "bugs". Our main goal was to provide a framework for transactional modeling of the performance and energy of a memory system for systems-on-a-chip in order to achieve the greatest energy saving possible. Our framework helps on improving the overall power efficiency of the design, whether it is in the clock tree, the data path, or in the memory sub-systems, without degrading the overall performance.

With the intention of moving to higher level of abstraction, we used Model Driven Engineering methodology to do it. Simon Davidmann, CEO of Imperas, said "At the same time, you could say that if we had a better high-level language, we could implement everything." Then he added "we could just describe things at a high level language like Matlab, push a button, and boom, out would come with the silicon." [78]. We propose a MDE-based flow to automatically generate joint power and performance simulation SystemC-TLM code.

We come up with an UPF-like-based approach that illustrates the possibility of defining at the ESL level a scripting language inspired by UPF standard, in order to reinforce the separation between functional model and power model. This type of scripting language should be standardized to facilitate the development of power/performance simulation tools addressing this ESL level.

We also showed that there is a growing need to understand and analyze DRAM power consumption considering an overall system level approach. One trend may be to propose new memory architectures based on advanced technologies that consume less power such as MRAM.

Consequently, mastering the design complexity and meeting the time-to-market requirements become more challenging for the hardware system designers. The traditional methodologies, where the design starts from the RTL level, for such complex systems is error prone, and needs lots of effort. Additionally, RTL level design evaluation is very time consuming. Therefore, the recently in use design methodologies start from system level.

Mobile devices are requesting low power memory solutions. In this work, we propose a SystemC-TLM based simulation framework at ESL level that helps designers to find the optimized memory configuration in terms of energy and performance which is consistent with the rest of the system including the power management strategy. The coupling of a tool such as DRAMPower with a SystemC-TLM functional model

on which a power-oriented overlay (PwClkARCH) is added provides an efficient framework able to evaluate the impact of various DRAM memory types on the whole system performance and power consumption. For example, we evaluate one NVM memory model of an MRAM using our framework and we show that such technology is a promising candidate to be used as main memory instead of traditional SDRAMs.

2. Perspectives

Our work in this thesis opens several perspectives for future work. The first perspective concerns the PwClkARCH library. We prove that this library helps a lot on modeling and controlling power consumption of a SystemC-TLM-based platform. A future work is possible which consists of adapting the library to be able to describe and simulate a very complex hardware architecture. We think to make a distributed PMU model, and so we obtain more than a single Power Manager in a design, allowing a reduced complexity of the internal tables (PST, CST, OPP Table) of each PMU.

Two innovations in semiconductor technology have recently emerged as possible alternatives to traditional planar CMOS technology: FDSOI for “Fully Depleted Silicon On Insulator” and FinFet Technology for “Fine Field Effect Transistor”. Both technologies have promising properties to further reduce dimensions, thanks to better electrostatic control of the gate on the transistor channel for FinFET technology and a reduction in substrate losses for FDSOI technology. In the case of FDSOI technology, several adaptations are to predict in the library PwClkARCH since we can adapt the body bias voltage to play on the frequency and the leakage [80]. So, it will be interesting to implement this type of control in PwClkARCH to support new technologies.

During this thesis, we showed the importance of having a coupling between the two types of simulators, "digital part" (SystemC-TLM & PwClkARCH) and "memory part" (DRAMPower), since the performances of one act on the performance of the other. However, the validation of the entire simulator would require the study of other use cases and a more detailed comparison with measurements from a real platform. It would also be interesting to add the power model of other levels of the memory system such as caches.

The connection made between SystemC-TLM and PwClkARCH, and then DRAMPower, actually makes it possible to evaluate the performance and power of the complete system. But the influence on the simulated performance of the number of transactions before activation of DRAMPower can be significant in some cases (e.g. when there are multiple competing independent transaction flows).

Also it would be relevant to study a stronger coupling of SystemC-TLM DDR model with DRAMPower, for example by directly integrating DRAMPower into the SystemC-TLM DDR IP model.

It is necessary also to evolve DRAMPower in order to integrate new memory technologies such as LPDDR4 and LPDDR5.

This work addresses mainly the performance and power consumption estimation at ESL of a memory sub-system connected to the logic part of a system. We demonstrated that a specific DRAM simulator is required to describe the complex behavior of the memory unit since the traditional equation-based power model established for logic part (included in PwClkARCH) is not suited for modelling the power consumption of DRAM. A comparable study could be applied for the interconnection inside the system on chip since the behavior of the interconnection is dependent on the traffic generated by the units connected on it and on the internal control mechanisms to route the data in the interconnection. A first work on that issue shows the relevance of this problem [79]. A generic power model for an interconnect should be then developed and could be considered in our Intel platform augmented with PwClkARCH.

3. Publications of the author related to this theme

- Cross layer optimization in terminals.
  Valerio Frascolla, Ralph Hasholzner, Jonathan Ah Sue, Amal Ben Ameur, Muhammad Mudussir Ayub, Krzysztof Miesniak, Jürgen Englisch.
  26th European Signal Processing Conference (EUSIPCO), Rome, September 2018.
- Mobile Terminals System-level Memory Exploration for Power and Performance Optimization.
  Amal Ben Ameur, Michel Auguin, François Verdier, Valerio Frascolla.
  28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Costa Brava, Spain, July 2018.
- A framework for System Level Low Power Design Space Exploration.
  Amani Ben Mrad, Michel Auguin, François Verdier, Amal Ben Ameur.
  24th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Batumi, Georgia, September 2017.
- Power and Performance aware Electronic System Level Design.
  Amal Ben Ameur, Didier Martinot, Patricia Guitton-Ouhamou, Valerio Frascolla, François Verdier, and Michel Auguin.
  12th IEEE International Symposium on Industrial Embedded Systems (SIES), Toulouse, France, June 2017.
- An ESL framework for low power architecture design space exploration.
  Hend Affes, Amal Ben Ameur, Michel Auguin, François Verdier, Calypso Barnes.

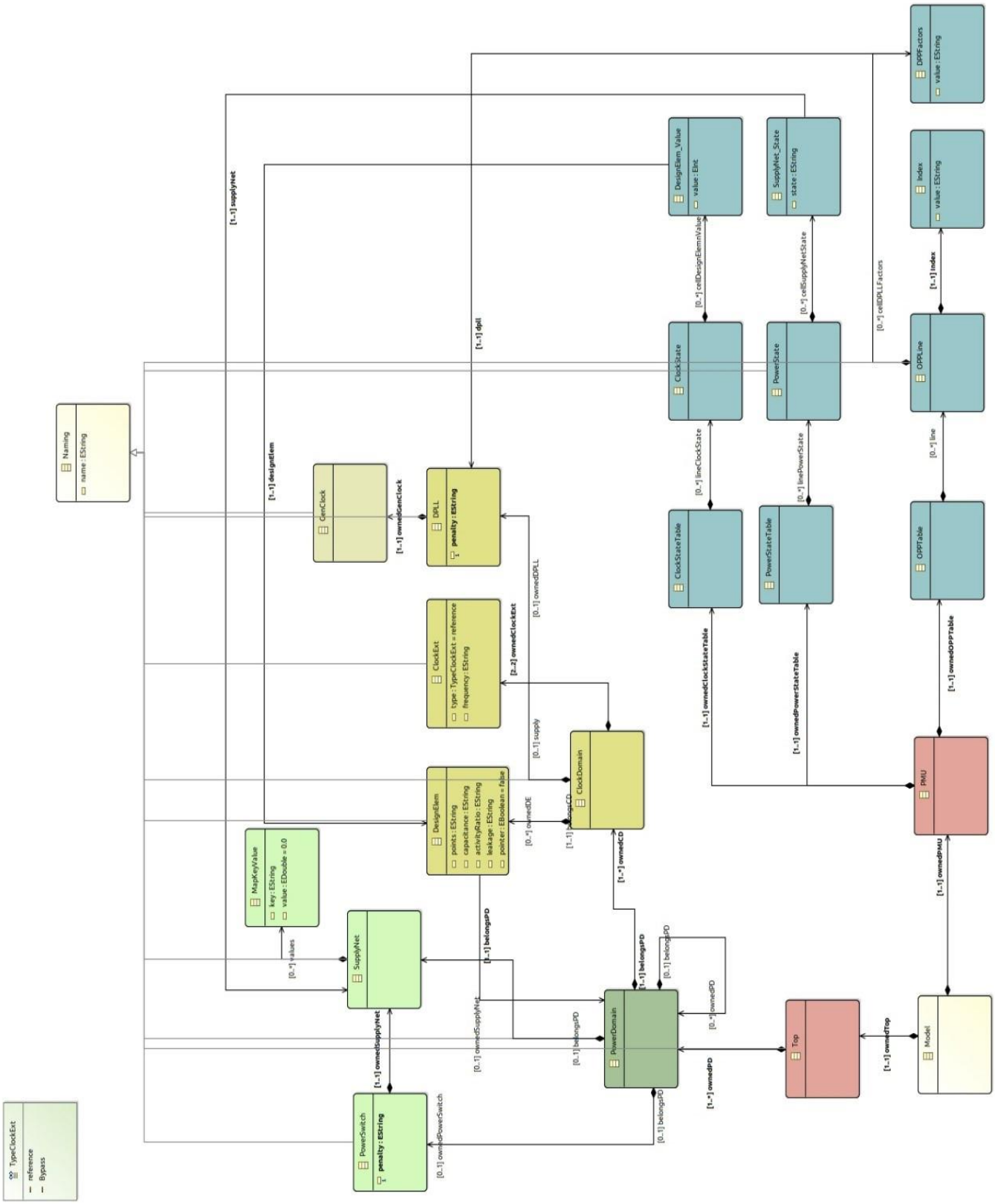
27th International Conference on Application-specific Systems, Architectures and Processors (ASAP), London, UK, July 2016.

- Clock Management and Analysis for Transaction-Level Virtual Prototypes.
  Amal Ben Ameur, Hend Affes, Michel Auguin, François Verdier, Xavier Buisson.
  Conference: Forum on specification and Design Languages (FDL'2015), Barcelona, Spain, September 2015.

Acronyms

ACT	Activate Row
CAD	Computer-aided design
CD	Clock Domain
ClkST	Clock State Table
CM	Clock Manager
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DE	Design Element
DMIF	DDR memory interface
DRAM	Dynamic RAM
ESL	Electronic System Level
KPI	Key Performance Indicator
LPDDR4	Low Power Double Data Rate 4
MPSoC	Multi-Processor System-on-Chip
MRAM	Magnetic RAM
NVM	Non Volatile Memory
OPP	Operating Performance Point
OPPT	OPP Table
PD	Power Domain
PHY	Physical layer
PM	Power Manager
PMU	Power Management Unit
PRE	Precharge
PS	Protocol Stack
PST	Power State Table
RAM	Random Access Memory
REF	Refresh
SN	Supply Net
STT-MRAM	Spin Transfer Torque-MRAM
TTI	Traffic Transfer Interval
VP	Virtual Prototyping

Appendix


TypeClockExt
reference
Bypass
Naming
name : EString
DPPFactors
value : EString
DesignElem_Value
value : Elnz
SupplyNet_State
state : EString
Index
value : EString
[1..1] supplyNet
[1..1] designItem
[1..1] dpll
[0..*] cellDesignElemValue
[0..*] cellSupplyNetState
[0..*] cellDPLLFactors
[1..1] index
ClockState
PowerState
OPPLine
[0..*] lineClockState
[0..*] linePowerState
[0..*] line
GenClock
DPLL
penalty : EString
[1..1] ownedGenClock
[0..1] ownedDPLL
ClockStateTable
PowerStateTable
OPPTable
[1..1] ownedClockStateTable
[1..1] ownedPowerStateTable
[1..1] ownedOPPTable
ClockExt
type : TypeClockExt = reference
frequency : EString
[2..2] ownedClockExt
[0..1] supply
PMU
DesignElem
points : EString
capacitance : EString
activityRatio : EString
leakage : EString
pointer : EBoolean = false
[0..*] ownedDE
[1..1] powerCD
ClockDomain
[1..*] ownedCD
[1..1] ownedPMU
MapKeyValue
key : EString
value : EDouble = 0.0
[0..*] values
SupplyNet
[1..1] belongsPD
[1..1] belongsPD
[0..1] belongsPD
[0..*] ownedPD
[1..1] ownedSupplyNet
[0..1] ownedSupplyNet
[0..1] belongsPD
PowerDomain
Top
Model
[1..*] ownedPD
[1..1] ownedTop
PowerSwitch
penalty : EString
[0..1] ownedPowerSwitch
[0..1] belongsPD

```
/************************* Create Power Domains*****************************
*****************************************************************************/
//Power_Domain(Power_Domain* container, string name, string scope);

Power_Domain* PD1 = new Power_Domain(NULL,"PD1");
Power_Domain* PD2 = new Power_Domain(NULL,"PD2");
/************************* Create Clock Domains*****************************
*****************************************************************************/
//Clock_Domain(Clock_Domain* container, string name);


Clock_Domain* CD2 = new Clock_Domain(NULL,"CD2");

Clock_Domain* CD1 = new Clock_Domain(NULL,"CD1");
/************************* Create Design Elements *************************
*****************************************************************************/
//Design_elem(Power_Domain* pDomain_name, Clock_Domain* pDomain_name, sc_core::sc_object& p_obj);

Design_elem * CortexM3 = new Design_elem (PD1,CD1,CortexM3);


Design_elem * Bus = new Design_elem (PD1,CD1,Bus);


Design_elem * UART = new Design_elem (PD1,CD1,UART);


Design_elem * G726_ENC = new Design_elem (PD2,CD2,G726_ENC);


Design_elem * G726_DEC = new Design_elem (PD2,CD2,G726_DEC);

/****************************** Create DPLLs ******************************
*****************************************************************************/
//DPLL(Clock_Domain* pClock_Domain, string name_Dpll, float penalty_delay);


DPLL * DPLL1 = new DPLL (CD1,DPLL1,10.0);


DPLL * DPLL2 = new DPLL (CD2,DPLL2,5.0);

/************************* Create Power Switches*****************************
*****************************************************************************/
//Power_Switch(Power_Domain* pPower_Domain, string name);

Power_Switch* PS1 = new Power_Switch(PD1,"PS1");

Power_Switch* PS2 = new Power_Switch(PD2,"PS2");
```

Example of sc_main generated code

Bibliography

[1] J. Yang, X. Ge and Y. Zhong, "How Much of Wireless Rates Can Smartphones Support in 5G Networks?," in *IEEE Network*, Nov. 2018.

[2] S. Traboulsi, V. Frascolla, N. Pohl, J. Hausner and A. Bilgic, "A versatile low-power ciphering and integrity protection unit for LTE-advanced mobile devices," in *IEEE 10th International New Circuits and Systems Conference (NEWCAS)*, Montreal, Jun. 2012.

[3] S. Traboulsi, N. Pohl, J. Hausner, A. Bilgic and V. Frascolla, "Power analysis and optimization of the ZUC stream cipher for LTE-Advanced mobile terminals," in *IEEE 3th Latin American Symposium on Circuits and Systems (LASCAS)*, Feb. 2012.

[4] D. Pandiyan and C.-J. Wu, "Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms," in *IEEE International Symposium on Workload Characterization*, October 26-28. 2014.

[5] K. Chandrasekar, B. Akesson and K. Goosse, "Improved Power Modeling of DDR SDRAMs," in *14th Euromicro Conference on Digital System Design*, Oulu, Finland, 31 Aug.-2 Sept. 2011.

[6] H. Foster, "Navigating the Perfect Storm: New Trends in Functional Verification," in *10th International Haifa Verification Conference (HVC)*, November 18-20, 2014.

[7] R. Duan, M. Bi and C. Gniady, "Exploring memory energy optimizations in smartphones," in *International Green Computing Conference and Workshops*, Orlando, FL, USA, 25-28 July 2011.

[8] "42 Years of Microprocessor Trend Data," [Online]. Available: https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/.

[9] "The gem5 Simulator," [Online]. Available: http://gem5.org/Main_Page.

[10] C. Menard, J. Castrillon, M. Jung and N. Wehn, "System simulation with gem5 and SystemC: The keystone for full interoperability," in *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, Pythagorion, Greece, 17-20 July 2017.

[11] *IEEE Standard for Standard SystemC® Language Reference Manual*, IEEE Std 1666–2011, 2012.

[12] "Accellera," [Online]. Available: https://www.accellera.org/.

[13] *OSCI TLM-2.0 LANGUAGE REFERENCE MANUAL*, Open SystemC Initiative, June 2008.

[14] M. Abdurohman, Kuspriyanto, S. Sutikno and A. Sasongko, "The New Embedded System Design Methodology For Improving Design Process Performance," *International Journal of Computer Science and Information Security*, vol. 8, pp. 35-43, 2010.

[15] P. Coussy, D. Gajski, M. Meredith and A. Takach, "An Introduction to High-Level Synthesis," in *IEEE Design and Test of Computers*, July-Aug. 2009.

[16] M. Sharma, R. Gautam and M. A. Khan, Design and Modeling of Low Power VLSI Systems, June, 2016.

[17] S. McCloud, "Low-Power RTL Report," Calypto Design Systems, 2012.

[18] M. Wolf, High-Performance Embedded Computing, 2014.

[19] E. Esteve, "New Power Management IP Solution Can Dramatically Increase SoC Energy Efficiency," 2018. [Online]. Available: https://www.design-reuse.com/articles/44834/new-power-management-ip-solution-can-dramatically-increase-soc-energy-efficiency.html.

[20] *IEEE Standard for Design and Verification of Low-power, Energy-aware Electronic Systems,* IEEE Std 1801–2015, 2015.

[21] P. Sheridan, "Time For A DDR Background Check. How background power consumption impacts the energy efficiency of an SoC," *semiconductor engineering*, July 28th. 2016.

[22] D. Macko, K. Jelemenska and P. Cicak, "Simplifying low-power SoC top-down design using the system-level abstraction and the increased automation," *Integration the VLSI Journal,* vol. 63, September. 2018.

[23] "Platform Architect MCO," [Online]. Available: https://www.synopsys.com/verification/virtual-prototyping/platform-architect.html.

[24] "Intel® CoFluent™ Studio," [Online]. Available: https://www.intel.fr/content/www/fr/fr/cofluent/cofluent-studio.html.

[25] "Intel® Docea™ Power Simulator," [Online]. Available: https://www.intel.com/content/www/us/en/system-modeling-and-simulation/docea/power-simulator.html.

[26] "Intel® Docea™ Power Analytics," [Online]. Available: https://www.intel.com/content/www/us/en/system-modeling-and-simulation/docea/power-analytics.html.

[27] "Mentor Graphics Underscores Low-Power Strategy with Vista Architecture-Level Power Solution," [Online]. Available: https://www.mentor.com/company/news/esl-vista-low-power.

[28] Y. MY, . K.-H. C , V. Pascal and G. DJ , "TLM Power 3.0 (CBG) User Manual," 2012.

[29] C. Gomez , J. Deantoni and F. Mallet , "Multi-View Power Modeling based on UML, MARTE and SysML," in *38th Euromicro Conf. on Software Engineering and Advanced Applications*, Cesme, Turkey, 2012.

[30] C. Trabelsi, R. Ben Atitallah, S. Meftali, J.-L. Dekeyser and A. Jemai, "A Model-Driven Approach for Hybrid Power Estimation," *Embedded Systems Design,* March. 2011.

[31] E.-Y. Kang, G. Perrouin and P.-Y. Schobbens, "Towards Formal Energy and Time Aware Behaviors in EAST-ADL: An MDE Approach," in *12th International Conference on Quality Software*, Xi'an, Shaanxi, China, 27-29 Aug. 2012.

[32] F. Ben Abdallah and L. Apvrille, "Fast Evaluation of Power Consumption of Embedded Systems Using DIPLODOCUS," in *39th Euromicro Conference on Software Engineering and Advanced Applications*, Santander, Spain, 4-6 Sept. 2013.

[33] T. Arpinen, E. Salminen, T. D. Hämäläinen and M. Hännikäinen, "MARTE profile extension for modeling dynamic power management of embedded systems," *Journal of Systems Architecture,* vol. 58, pp. 209-219, April. 2012 .

[34] F. Ben Abdallah, C. Trabelsi, R. Ben Atitallah and M. Abed, "Model-Driven Approach for Early Power-Aware Design Space Exploration of Embedded Systems," *Journal of Signal Processing Systems,* vol. 87, pp. 271-286, June. 2017.

[35] "WHAT IS SYSML?," [Online]. Available: http://www.omgsysml.org/what-is-sysml.htm.

[36] H. Affes, M. Auguin, F. Verdier and A. Pegatoquet, "A methodology for inserting clock-management strategies in transaction-level models of systemon- chips," in *Forum on Specification and Design Languages (FDL)*, Barcelona, Spain, 14-16 Sept. 2015.

[37] O. Mbarek, A. Pegatoquet and M. Auguin, "A Methodology for Power-Aware Transaction-Level Models of Systems-on-Chip Using UPF Standard Concepts," in *PATMOS'11 Proceedings of the 21st international conference on Integrated circuit and system design: power and timing modeling, optimization, and simulation*, Madrid, Spain, September. 2011.

[38] B. Akesson and K. Goossens, Memory Controllers for Real-Time Embedded Systems. Predictable and Composable Real-Time Systems, Springer, 2012.

[39] J. Stuecheli, D. Kaseridis, D. Daly, H. C. Hunter and L. K. John, "The virtual write queue: coordinating DRAM and last-level cache policies," in *37th Annual International Symposium on Computer Architecture*, Saint-Malo, France, June 19 - 23. 2010.

[40] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova and M. Prieto, "Survey of scheduling techniques for addressing shared resources in multicore processors," *ACM Computing Surveys,* vol. 45, November. 2012.

[41] K. J. Nesbit, N. Aggarwal, J. Laudon and J. E. Smith, "Fair Queuing Memory Systems," in *39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, Orlando, FL, USA, 9-13 Dec. 2006.

[42] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson and J. D. Owens, "Memory Access Scheduling," in *27th annual international symposium on Computer architecture*, 2000.

[43] T. Moscibroda and O. Mutlu, "Memory performance attacks: denial of memory service in multi-core systems," in *16th USENIX Security Symposium on USENIX Security Symposium*, Berkeley, CA, 2007.

[44] O. Mutlu and T. Moscibroda, "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems," in *International Symposium on Computer Architecture*, Beijing, China, 21-25 June 2008.

[45] M. K. Jeong, M. Erez, C. Sudanthi and N. Paver, "A QoS-aware memory controller for dynamically balancing GPU and CPU bandwidth use in an MPSoC," in *49th Annual Design Automation Conference*, 2012.

[46] E. Ebrahimi, C. J. Lee, O. Mutlu and Y. N. Patt, "Fairness via source throttling: a configurable and high-performance fairness substrate for multi-core memory systems," in *ASPLOS'10*, Pittsburgh, Pennsylvania, March. 2010.

[47] L. Subramanian, V. Seshadri, Y. Kim, B. Jaiyen and O. Mutlu, "MISE: Providing performance predictability and improving fairness in shared main memory systems," in *IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Shenzhen, China, 23-27 Feb. 2013.

[48] L. Subramanian, D. Lee, V. Seshadri, H. Rastogi and O. Mutlu, "The Blacklisting Memory Scheduler: Achieving high performance and fairness at low cost," in *IEEE 32nd International Conference on Computer Design (ICCD)*, Seoul, South Korea, 19-22 Oct. 2014.

[49] R. Das, R. Ausavarungnirun, O. Mutlu, A. Kumar and M. Azimi, "Application-to-core mapping policies to reduce memory system interference in multi-core systems," in *IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Shenzhen, China, 23-27 Feb. 2013.

[50] F. Balasa, C. V. Gingu, I. I. Luican and H. Zhu, "Design space exploration for low-power memory systems in embedded signal processing applications," in *IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications*, Taipei, Taiwan, 19-21 Aug. 2013.

[51] M. Xie, D. Tong, K. Huang and X. Cheng, "Improving system throughput and fairness simultaneously in shared memory CMP systems via Dynamic Bank Partitioning," in *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Orlando, FL, USA, 15-19 Feb. 2014.

[52] J. Hu, Q. Zhuge, C. J. Xue, W.-C. Tseng and E. H. Sha, "Optimizing Data Allocation and Memory Configuration for Non-Volatile Memory Based Hybrid SPM on Embedded CMPs," in *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, Shanghai, China, 21-25 May 2012.

[53] M. Qiu, Z. Chen, Z. Ming, X. Qin and J. Niu, "Energy-Aware Data Allocation With Hybrid Memory for Mobile Cloud Systems," *IEEE Systems Journal*, vol. 11, pp. 813-822, September. 2014.

[54] M. Monchiero, G. Palermo, C. Silvano and O. Villa, "Exploration of Distributed Shared Memory Architectures for NoC-based Multiprocessors," in *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, Samos, Greece, 17-20 July. 2006.

[55] T. Bouhadiba, M. Moy, F. Maraninchi, J. Cornet, L. Maillet-Contoz and I. Materic, "Co-simulation of Functional SystemC TLM Models with Power/Thermal Solvers," in *IEEE International Symposium*

on Parallel & Distributed Processing, Workshops and Phd Forum, Cambridge, MA, USA, 20-24 May. 2013.

[56] S. Abdi, Y. Hwang, L. Yu, H. Cho, I. Viskic and D. D. Gajski, "Embedded system environment: A framework for TLM-based design and prototyping," in *21st IEEE International Symposium on Rapid System Protyping*, Fairfax, VA, USA, 8-11 June. 2010.

[57] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman and N. P. Jouppi, "A Comprehensive Memory Modeling Tool and Its Application to the Design and Analysis of Future Memory Hierarchies," in *International Symposium on Computer Architecture (ISSCC)*, Beijing, China, 21-25 June. 2008.

[58] P. Rosenfeld, E. Cooper-Balis and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," *IEEE Computer Architecture Letters*, vol. 10, 2011.

[59] Y. Kim, W. Yang and O. Mutlu, "Ramulator: A Fast and Extensible DRAM Simulator," vol. 15, March. 2015.

[60] U. Kang, H.-s. Yu, C. Park, H. Zheng, J. B. Halbert, K. S. Bains, S.-J. Jang and J. S. Choi, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.

[61] J. Wang, X. Dong and Y. Xie, "OAP: An obstruction-aware cache management policy for STT-RAM last-level caches," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, March. 2013.

[62] S. Senni, L. Torres, G. Sassatelli, A. Gamatié and B. Mussard, "Exploring MRAM Technologies for Energy Efficient Systems-On-Chip," *IEEE Journal of Emerging and Selected Topics in Circuits and Systems*, vol. 6, 2016.

[63] N. D. Rizzo, D. Houssameddine, J. Janesky, R. Whig, F. B. Mancoff, M. L. Schneider, M. DeHerrera, J. J. Sun, K. Nagel, S. Deshpande, H.-J. Chia, S. M. Alam, T. Andre, S. Aggarwal and J. M. Slaughter, "A Fully Functional 64 Mb DDR3 ST-MRAM Builton 90 nm CMOS Technology," *IEEE Transactions on Magnetics*, vol. 49, 2013.

[64] J. Wang, X. Dong and Y. Xie, "Enabling high-performance LPDDRx-compatible MRAM," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, La Jolla, CA, USA, August. 2014.

[65] K. Asifuzzaman, R. S. Verdejo and P. Radojković, "Enabling a reliable STT-MRAM main memory simulation," in *MEMSYS '17 Proceedings of the International Symposium on Memory Systems*, 2017.

[66] "Fujitsu Develops High-Reliability Read-Method for Spin-Torque-Transfer MRAM," [Online]. Available: https://www.cdrinfo.com/d7/content/fujitsu-develops-high-reliability-read-method-spin-torque-transfer-mram.

[67] X. Dong, C. Xu, Y. Xie and N. P. Jouppi, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, June. 2012.

[68] H. Affes, A. Ben Ameur, M. Auguin, F. Verdier and C. Barnes, "An ESL framework for low power architecture design space exploration," in *IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, London, UK, July. 2016.

[69] "Eclipse Modeling Framework (EMF)," Eclipse, 2016. [Online]. Available: https://www.eclipse.org/modeling/emf/.

[70] "EcoreTools," 2013. [Online]. Available: https://www.eclipse.org/ecoretools/.

[71] S. Sendall and W. Kozaczynski, "Model transformation: the heart and soul of model-driven software development," *IEEE Software*, vol. 20, pp. 42-45, 2003.

[72] "Acceleo: GENERATE ANYTHING FROM ANY EMF MODEL," [Online]. Available: https://www.eclipse.org/acceleo/.

[73] B. Raaf, M. Faerber, B. Badic and V. Frascolla, "Key technology advancements driving mobile communications from generation to generation," *Intel Technology Journal*, vol. 18, 2014.

[74] D. Szczesny, S. Hessel, A. Showk, A. M. Bilgic, U. Hildebrand and V. Frascolla, "Joint Uplink and Downlink Profiling of LTE Protocol Processing on a Mobile Platform," *Special Issue of the International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, vol. 1, pp. 21-39, 2010.

[75] K. Chandrasekar, C. Weis, Y. Li, S. Goossens, M. Jung, O. Naji, B. Akesson, N. Wehn and K. Goossens, "DRAMPower: Open-source DRAM Power & Energy Estimation Tool," [Online]. Available: http://www.drampower.info.

[76] X. Dong, C. Xu, Y. Xie and N. P. Jouppi, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, pp. 994-1007, June. 2012.

[77] Y.-H. Lu and G. De Micheli, "Comparing system level power management policies," *IEEE Design & Test of Computers*, vol. 18, pp. 10-19, 2001.

[78] "Design Reuse Vs. Abstraction," July. 2018. [Online]. Available: https://semiengineering.com/design-reuse-vs-abstraction/.

[79] A. Genov, "Création d'un modèle SystemC/TLM2.0 du SoC i.MX8QM et modélisation de sa consommation," NXP, LEAT, 2018.

[80] Y. Akgul, D. Puschini, S. Lesecq, E. Beigne, P. Benoit and L. Torres, "Methodology for Power Mode selection in FD-SOI circuits with DVFS and Dynamic Body Biasing," in *PATMOS: Power and Timing Modeling, Optimization and Simulation*, Karlsruhe, Germany, Sep 2013.