Contributions to the Design of Reconfigurable Embedded Systems: from Modelling to Implementation

Jean-Christophe Prévotet

Mémoire

Habilitation à Diriger des Recherches

Contributions to the Design of Reconfigurable Embedded Systems: from Modelling to Implementation

Jean-Christophe Prévotet

Maître de Conférences à l'INSA de Rennes
Laboratoire IETR / Équipe de recherche SYSCOM

A soutenir le 07/06/2019 devant le jury composé de

Rapporteurs

Diana Göhringer	Professeure au TU, Dresden
Christophe Jégo	Professeur à ENSEIRB-MATMECA, IMS, Bordeaux
Gilles Sassatelli	Directeur de recherche CNRS, LIRMM, Montpellier

Examinateurs

Guy Gogniat	Professeur à l'université de Bretagne-Sud, Lab-STICC
Christophe Moy	Professeur à l'université de Rennes 1, IETR
Frédéric Pétrot	Professeur à l'université de Grenoble, TIMA
Fabienne Nouvel	Maître de conférences HDR, INSA de Rennes, IETR

Sommaire

Sommaire

Partie I

Synthesis of Research Works

Preliminaries

The second part of this document describes my research activities in details. Chapter 1 describes the context of the different studies and the research tracks that have been followed. The subsequent chapters describe the different topics that have been studied and provide the main contributions for each of them.

Outline

Introduction

Chap. 1

1 Summary of studies

Nowadays, algorithmic complexity keeps on increasing in many domains such as signal, image processing, communications or control. Applications that implement these algorithms are often executed on embedded systems that generally require a significant power of computation. As a response to this issue, hardware architectures are now composed of numerous heterogeneous and optimized computation units that operate in parallel. For example, devices such as SoC (System on Chip) may exhibit programmable computation units, reconfigurable units, or even dedicated data-paths to reach the highest level of performance.

Such devices are extremely complex to design and manage and require a lot of expertise to be handled. In this context, the use of an operating system (OS) is often considered since it allows users to obtain an abstraction of the underlying hardware. Services such as communications, memory management, tasks' scheduling, resources management are directly handled by the operating system, which considerably relieve the systems' management.

The first research subject that I have been focusing on has been to propose new methods that would help designers in building their custom embedded Operating System(s) to manage heterogeneous platforms. This work is presented in Section 1.1.

A second research axis deals with the specificity of the platform itself and particularly focuses on reconfigurable architectures. The issues related to reconfiguration management have been studied thoroughly in my career and the challenges that I tried to tackle are summarized in Section 1.2.

Finally, for several years, I have also investigated the energy efficiency of reconfigurable embedded systems from a design perspective. The privileged domain of study was wireless communications but it can, of course, be extended to other domains. In this axis, I have proposed new exploration methods and tools to help designers in choosing the right algorithms for a given platform. The research issues as well as the proposed solutions are described in Section 1.3.

1.1 OS Modelling

Today embedded systems have become so complex that they are able to process applications with various types of constraints. This diversity of constraints has a direct impact on the underlying management of these applications and on operating systems specifically. For example, some applications may be real-time constrained imposing a Real-Time Operating System (RTOS), with specific features, scheduling properties, inter-blocking resolution, etc. Some other applications may be more general-purpose and have other types of constraints (like in multimedia, consumer electronics, etc.). For the latter, General-Purpose Operating Systems (GPOS) are more appropriate.

In modern embedded systems, it is not rare to find several operating systems with different purposes. This is the case, for example, in mobile phones, where an OS is dedicated to the user-machine interface and another one is hidden, which is responsible for managing communicating parts implemented in hardware. In order to get an optimal use of resources, hardware virtualization can help, by enabling the concurrent execution of an application OS (Linux, Windows, etc.) and of a real-time OS on the same processor.

Virtualization has a lot of interests. It makes the support of architectural abstraction possible since a given application may migrate from several physical cores to a single virtual one. It also enables to execute legacy applications without any modification since the underlying operating system may be easily

ported as a guest within a virtual machine. Another major aspect of virtualization is security since virtual machines are fully isolated and each of them has its own encapsulated application code that cannot access the rest of the system. Only the Virtual Machine Monitor (VMM) or hypervisor, that is significantly more secure, may have specific rights to access resources.

Today, such systems have become so cumbersome that their design process is very elaborate and require huge skills and expertise. In order to help designers in their task and increase their productivity, it seems very important to first raise the design abstraction level. Second it also seems important to provide designers with dedicated tools that automate design tasks from a high-level model.

In this work, the privileged target has been Reconfigurable Systems-on-Chip (RSoC) or any device that is composed of a processing core that is associated with a reconfigurable region. This, of course, includes current System on Programmable Chip (SoPC) devices based on FPGA circuits. Here are some questions we have tried to tackle :

— How to design the software architecture of an embedded OS that takes hardware specificity (e.g the reconfigurable parts) into account ?

— How can designers simulate the OS behaviour, which are the obtained metrics, and what is the degree of confidence that we can get from the results ?

— Is it possible to generate pieces of code corresponding to the software architecture from high-level models ?

These questions have been first treated in a single-OS environment. With the advent of virtualization in embedded systems, we have tried to adapt our research to take into account multi-OS environment. This has, of course, raised specific issues :

— Is the methodology proposed for single-OS architectures compatible with more complex systems ?

— Is it possible to evaluate the feasibility and performance of virtualization in the early stages of the design flow ?

— What is the overhead that is due to the hypervisor on the application performance ?

— How must the hypervisor be designed to abstract all hardware resources ?

1.2 Reconfiguration Management

The second main axis of my research has always been devoted to reconfigurable systems and particularly the management of such systems. Reconfiguration has always been a concept guaranteeing a system to adapt to environmental changes in many technical or biological systems. In the domain of hardware architectures, reconfiguration allows an architecture to adapt to a specific task by modifying its own functionality. This is exactly the opposite of classical computing schemes in which variable tasks require to be modified to adapt to a fixed architecture.

In this research axis, I have mostly focused on specific types of reconfigurable architectures. These architectures are composed of one or several cores that are implemented in hardware as part of a SoC fabric. Besides these cores, a reconfigurable fabric is available that makes it possible to implement reconfigurable architectures with different granularity of configuration. These architectures may be seen as hardware accelerators that aim to alleviate the processors' tasks. In our studies, we have proposed several levels of reconfiguration management : at physical level, at operating system level and at application level. For each of these levels, specific issues have been met and my PhD students and myself have tried to tackle them independently.

First, at physical level, we have mainly focused on all problems that are related to the hardware reconfiguration mechanisms and on the placement aspects of an abstract object that is a hardware task. The main issues deal with the structure of a hardware task :

— How can it interact with the software environment ?

— How may tasks communicate with each other to exchange data ?

Other issues are related to the spatial dimension of these tasks :

— how can a task be configured and dynamically placed within the reconfigurable area ?

— Which type of management is required to handle the task processing in this area ?

Second, at OS level, we also identified several issues that need to be taken into consideration. For example,

— what type of abstraction must have a given hardware task to make it easily manageable by an operating system ?

— How is it possible to keep the management of these tasks as transparent as possible, from the application point of view ?

Some other issues are directly related to the structure of the OS and deal with the additional mechanisms or services that are required to handle reconfiguration without too much overhead.

Finally, at application level, other issues may occur. These deal with the way an application may adapt to the environment. Which are the mechanisms that are necessary to allow an application to take a good decision regarding reconfiguration ? Can we imagine a common framework that make this decision transparent ? What would be the overhead due to this framework in this case ? Finally, is it possible to imagine smart reconfiguration mechanisms that are based on machine learning processes to allow a platform to reconfigure itself automatically ?

All these questions have been treated in our works and some answers have been proposed and explained in Chapter 3.

1.3 Power Modelling

Another significant aspect of my research studies has started dealing with energy efficiency of reconfigurable systems. The fact is that energy consumption has a huge role in such systems since they are often operated with limited energy resources like simple batteries. This is particularly the case of wireless communications systems that constitute the main elements of the future IoT. These systems have been our main target since they may represent a significant part of future devices that will have a real impact on worldwide energy consumption [BBFH+08].

In our works, we have decided to deal with the design aspects of these devices. More specifically, we have focused on the System Level, since most of the power optimization solutions intervene in the early stages of the design flow. When designing an embedded communicating device, one of the first questions designers have to answer deals with the functions to implement to perform processing and communication tasks. This first study generally results in a trade-off between the performance or Quality of Service (QoS) to achieve and the energy cost.

In this research axis, we have first concentrated on proposing efficient energy-aware communication schemes that might be implemented in very small devices, while guaranteeing a high level of performance. These types of solutions are classically studied by communication researchers or engineers using dedicated tools such as Matlab. Today, designers may even have the possibility to implement their algorithms on prototypes to get a first overview of their performance. Although very interesting in practice, this approach remains limited since solutions are often implemented in software, which does not give sufficient indications on the hardware complexity. This raises important questions :

— How can we exhibit the specificity of hardware in this case ?

— How is it possible to evaluate the hardware resources that will be necessary as well as the energetic cost ?

— How can we efficiently and rapidly compare different solutions to find the best ?

These questions have led to us to propose new methodologies and power models that would help designers in exploring design choices at System Level. The main issues are related to the models themselves :

— What is the nature of these models ?

— What is the expected accuracy ?

— How can we generalize the approach to other types of devices ?

All these questions have been addressed and some solutions are presented in this manuscript and specifically in Chapter 4.

2 Historical Research Background

The purpose of this section is to provide an overview of my research work chronologically. It also intends to explain the context and the strategic directions that have been followed according to the work environment in which my work was led.

2.1 PhD Studies

After a "Diplôme d'Etudes Approfondies" (DEA) obtained at the Pierre et Marie Curie University (UMPC), my first contact with research was during my PhD studies. My PhD started back in 1999 and was led in the "Laboratoire des Instruments et Systèmes d'Ile de France" under the supervision of P. Garda, B. Denby and B. Granado. The subject mainly dealt with the study of hardware architectures to implement Neural Networks in real time. The application domain of my studies was physics in general, and particle physics in particular.

During my first year of PhD, I had the opportunity to work in Munich, Germany in the "Max Planck Institut Fuer Physik" to get familiar with the context study of my work. The study aimed at designing a neural pre-processing circuit for on-line event triggering in a particle physics experiment. This experiment, H1, in which many European researchers were involved, was located in Hamburg on the DESY website. A second part of my thesis was dedicated to the hardware design of a very fast architecture destined to process neural networks within very tight timing constraints i.e 500 ns. The targeted application was again a high-performance particle recognition system as part of the ATLAS experiment, located on the site of the LHC (Large Hadron Collider) at CERN.

2.2 ETIS 2002-2007

After my PhD studies, in 2002, I obtained an "Attaché Temporaire d'Enseignement et Recherche" (ATER) and then an associate professor position at the university of Cergy-Pontoise. I joined the ARCHITECTURE team of the ETIS laboratory where my research interests have evolved towards the study of reconfigurable architectures for images or signal processing applications. This research team was led by Didier Demigny at that time and was composed of 5 people deeply involved in reconfiguration. It is in this context that I had the opportunity to work with L. Kessal on the supervision of Sonia Khatchadourian's thesis.

Other works that have been initiated in the ETIS lab have demonstrated the interest of managing dynamic reconfiguration efficiently. That is in this context that first research works started on operating systems for reconfigurable devices. At this time, the OveRSoC project, in which I was involved as the main leader has started.

2.3 IETR 2007- Today

The third part of my research career has started in 2007, time at which I joined the "INSA de Rennes" and the IETR lab. I was integrated in the Communication System team that was composed of 7 researchers mainly involved in the design of communications systems at the algorithmic level, which was quite far from my first expertise domain. Only one colleague of mine (F. Nouvel) was more interested in hardware architectures and communications prototyping.

When I first joined the SysCom team, I brought new skills related to operating systems, reconfiguration and modeling. With the help of my colleagues, I continued leading the OveRSoC project and was able to supervise PhD students as well as master students in this research area. Later, I developed the virtualization research axis, which can be seen as an extension of the studies that have been led on operating systems. Finally, M. Hélard and myself proposed a new research axis dealing with the study of energy consumption in reconfigurable systems, which gave me the chance to supervise 3 PhD students on this topic.

From OS Modelling to Implementation

Chap. 2

1 Context and Related Works

Since the beginning of computing, it has always been envisaged to design efficient architectures capable of executing more and more complex applications. In parallel, new constraints have progressively appeared such as the devices' size, that led to the concept of embedded systems. Regarding the design of such systems, one of the first issues has consisted in providing more computing power in smaller and smaller devices. Other issues have also progressively appeared due to a new usage of these systems, such as the the possibility to execute many applications at the same time, in a transparent manner.

For several decades, hardware devices such as SoCs have been an efficient solution to tackle these issues. These circuits often feature processing cores as well as dedicated hardware, which makes it possible to benefit from both software and hardware assets. Flexibility is guaranteed by the presence of software, whereas performance is ensured by hardware parts. Some of these devices feature a reconfigurable area that allows an hardware architecture to dynamically adapt during the execution of applications. These circuits constitute the targeted architecture in the works that I led since my PhD studies.

The will to efficiently use embedded systems raises a lot of questions. One of the most interesting relates to the capability of managing these systems in real-time. Today, most of them feature an Operating System (OS) that aims at virtualizing all hardware parts. This OS must, at least, manage memory, schedule different tasks under specific constraints and ensure resource sharing. Furthermore, it must be capable of offering new programming models that are completely independent of the underlying hardware architecture. When an OS has to response to real-time constraints, it is denoted as a Real-Time OS (RTOS).

Building a full embedded system based on an RTOS or several RTOSs is a delicate design process. Designers basically have to make sure that tasks operate correctly during the execution of the application while meeting the imposed constraints. Usually, the design of embedded systems is performed at high level of abstraction in order to allow designers to explore various choices and finally retain the most efficient one [JW05].

In the past, numerous methodologies and associated tools have been proposed to design complex embedded systems at high level of abstraction. Languages or libraries such as SpeC [spe] or SystemC [Sysc] have been used for years to build models. Unfortunately, a limited amount of studies only take into account RTOS modeling in their intrinsic design flow. A very complete survey in the area of RTOS modelling is presented in [Yu10]. The author introduces three criteria to classify RTOS models : application scope, software simulation method and functional accuracy. He also divides the different studies into two distinct categories according to the modeling granularity.

Regarding coarse grain models, the modeling process has several objectives. First, the full system is specified. It is then analyzed to perform hardware/software partitioning. This step intervenes very soon in the design process and since the hardware platform is not known at this stage, it is very difficult to obtain accurate results. At this level, an application is seen as a set of tasks with simple parameters such as the execution time, period and deadline [HKH04]. Furthermore, resource sharing and synchronization between tasks or with hardware parts are generally not taken into account in these models. The main interest to work at this level of abstraction is the simulation time that is greatly reduced. On the other hand, the main drawback is the lack of accuracy in the timing evaluation.

Authors in [GYG08] have presented an abstract RTOS model that is described in SpecC. Unfortunately, interrupts are not modelled in this work and interactions with hardware modules are not accurately defined. In other works, Zabel [ZMG09] uses a SystemC model that is based on [GYG08]. The SW/HW

synchronization is resolved by using the wait-for-event() method that is available in SystemC. An early work by [MVG] presents a SystemC-based abstract RTOS model. This model is a good basis for system exploration but has also some limitations. Service overhead is not included in the model and the task model is very simple and cannot really mimic a real system. This limitation is also met in another work by Hessel et al. [HRR+04]. In [LPC04], Le Moigne et al. describe a SystemC model of a multitask RTOS. This model is a part of the Cofluent tool which allows timing parametrization and validation of the RTOS model by configuring context load, context save, and scheduling duration. [HKRN08] present an abstract RTOS simulation model that is included in their SystemC-based design flow. They model pieces of software on a generic run-time system rather than directly modelling existing RTOS services.

After modelling and simulating high-level RTOS representation, another problem addressed by Gauthier et al. is the automatic generation of RTOS code. In [GYJ01], authors present a method of automatic generation of operating systems for a target processor. This method determines the OS services that are required in the code of the application and generates the corresponding code deduced from dependencies between services in an OS service library.

A first observation that we have made when starting to work on the subject was that none of the current works around RTOS modeling actually managed the dynamic creation of tasks. However, this particular point seemed very important to us and that is the reason why we decided to tackle this problem in the OveRSoC ANR project that is described in section 2 of the current chapter.

▸ From RTOS models to systems of OS

Another part of our studies has consisted in extending our RTOS modelling approach on more complex systems. The systems that we were interested in are capable of running several operating systems on the same platform. One very interesting configuration occurs when a RTOS executes concurrently with other general-purpose OS (GPOS). Today, this configuration is not rare and is met in a lot of domains. For example, in automotive, an AUTOSAR compliant RTOS and a Linux Genivi OS that supports in-vehicle infotainment application, could be co-located on the same Electronic Control Unit (ECU) [Hei].

A Virtual Machine (VM) system is a concept intended to simultaneously host multiple operating systems on a single hardware platform. Each guest operating system is executed in a separate and secure virtual machine. A virtual machine can be seen as an abstraction of the hardware resources provided to the guest operating systems. It is managed by a low-complexity kernel referred to as a Virtual Machine Monitor (VMM). This virtual machine monitor must ensure that a temporal or local fault in one virtual machine (e.g., an infinite loop, out-of-bounds array access, exhaustion of assigned resources) does not affect the operation of the other virtual machines. This feature is referred to as logical and temporal isolation in the real-time community, and as space and time partitioning in the RTOS industry.

Virtualization can be implemented in different ways. The classic approach to design a virtual machine system is to place the VMM on bare metal hardware whereas the virtual machine fits on top. The VMM runs in the most highly privileged level 1, while all guest operating systems run with lower privileges, as shown in Figure 2.1. Then, in a completely transparent way, the VMM can intercept and implement all the guest OS's actions that interact with the hardware resources. In this configuration, the VMM may also be called hypervisor.

An alternative implementation builds the VMM on top of an existing host operating system, resulting in what is called a hosted VM as shown in Figure 2.1c and Figure 2.1d. In this configuration, the installation process is similar to installing a typical application program on the host OS.

▸ Virtualization Implementation

Researches on embedded virtualization have been focusing on several features or challenges including security issue, virtualization overhead, software complexity, high-bandwidth VM inter-communication, and real-time task capability.

In conventional para-virtualization approaches, a guest OS is normally equipped with a virtualization patch to interact with the VMM, which requires the OS source code to be modified and must then be available [PKR+13]. One possible para-virtualization solution is to use a micro-kernel, which is a small Trust Computing Based (TCB) set of features defined as address space, threads and inter-process communication (see [Lie95]). Additional functionality is normally implemented at user level.

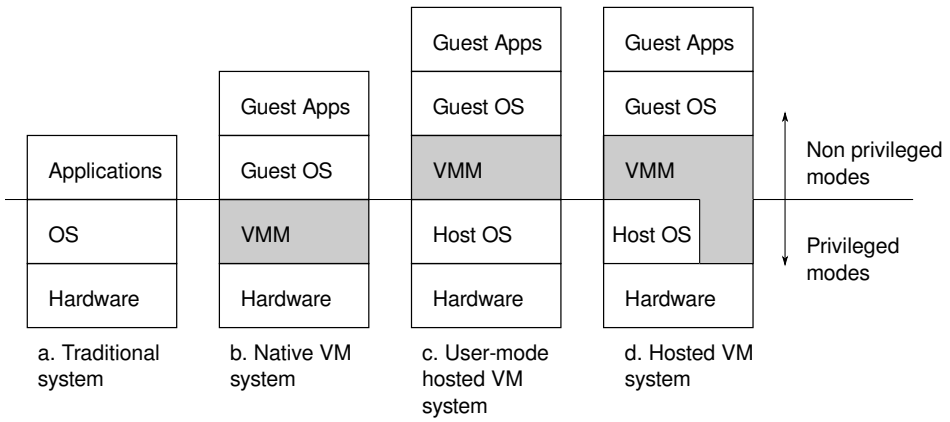


FIGURE 2.1 – Native and hosted VM Systems.

Examples of such kernels are the L4 micro-kernels usually relying on Linux to support user-level virtual machines. For example, in [GBL+09] and [XBG+10], EMUCO and Fiasco L4 were respectively used in communication and mobile phone systems by leveraging L4Linux as a client server. Some successful solutions are the OKL4 microvisor from Open Kernel Labs [HL10], or ARMvisor [DLC+12], based on KVM for the ARM architecture.

2 A new Design Methodology for Operating Systems

The works that are presented in this section have been initiated as soon as I obtained my position of associate professor at the IUT of Cergy Pontoise in 2003. They led to the elaboration of a national ANR project named OveRSoC, in which I was the main leader. This project started in 2005 and ended in 2008. The teams that were involved in the project were :

— the Architecture team of the ETIS laboratory (UMR CNRS 8051, Cergy-Pontoise)

— the R2D2 team of the IRISA laboratory (University of Rennes1, Lannion)

— the SYEL team of the LISIF laboratory (now LIP6) (University Pierre and Marie Curie, Paris)

The OveRSoC project has been proposed to not only take into account RTOS design but also define the underlying hardware platform. Its purpose was to propose an efficient design space exploration methodology and an associated simulation tool as well. Today, the OverSoC methodology is based on 4 important design concepts : exploration, separation of concerns, incremental refinement and re- usability.

We advocate the use of a high-level model of Reconfigurable SoCs (RSoC) in order to explore different critical design choices. Among these important choices, two exploration issues have been distinguished :

— the exploration of the application partitioning onto the processing resources

— the exploration of the RTOS services distribution and their algorithms.

Each design strategy belonging to these exploration levels is manually performed by designers. However, the proposed method intends to help in easily and quickly building the executable specification of the corresponding system. The underlying tools are used to evaluate performance in order to analyze and compare design strategies. The design choices corresponding to the second exploration issue (RTOS) are the architecture of the embedded RTOS (centralized or distributed, OS services organization, software, or hardware implementation, etc.), the services algorithms (scheduling policies, etc.), the interactions between OS service functions and underlying resources (reconfigurable areas, memories, interconnect type) and the software programming model.

Second, once validated the candidate design solutions are incrementally refined towards lower levels of abstraction down to the final implementation. The OveRSoC methodology permits the separation of concerns during the modeling and refinement process. It also defines modeling rules that facilitate independence and re-usability between components. For each design concern, specific and related refinement steps are proposed.

Finally, the method imposes a functional approach at each level of abstraction, which validates the application functionality and makes it possible to evaluate the performance. As depicted in Figure 2.2, the main aspect of the framework consists of an iterative process of exploration. It means that this process is repeated as many times as required, until the system constraints are met.

The exploration flow starts with the description of three different models : the application model, the architecture model and the operating system model. In a second step, all these descriptions are combined to create a global model representing the entire system. This model is generated in SystemC in order to exploit the advantages offered by this library, namely the high-level description allowing high-speed simulations. The global model is then simulated with the SystemC kernel. After this simulation step, metrics are evaluated and designers have the possibility to analyze the generated results. If the obtained results are not satisfactory, designers may restart a new iteration by modifying the corresponding attributes. Finally, the simulation normally ends up by showing no incoherence between the initial specifications and the timing behaviour of the application.

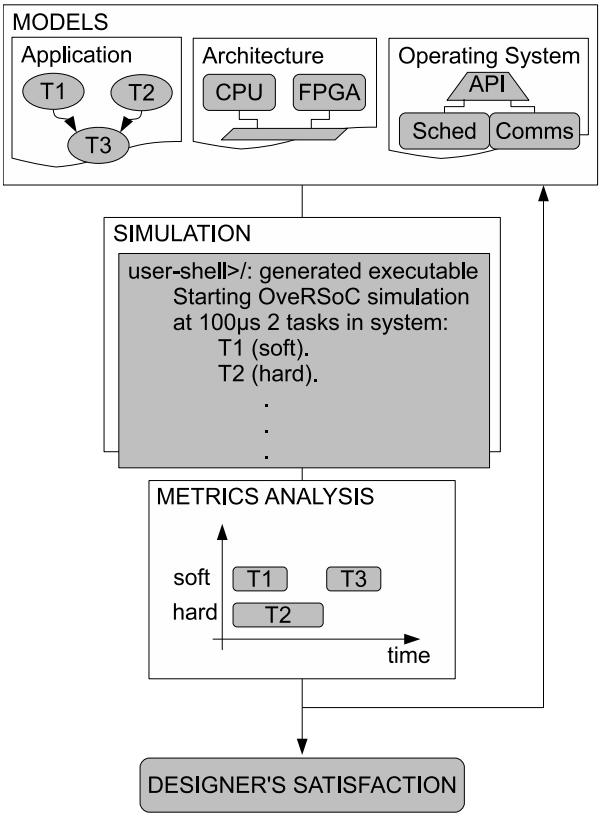


Figure 2.2 – OveRSoC design methodology.

2.1 System Specifications

This first stage of the methodology consists in setting the specifications of the system. The functional specifications define the algorithmic behaviour, i.e. the application, whereas the architecture specifications define the characteristics of the targeted hardware. Specifications also include the constraints which must be respected by the system. For instance, it may refer to timing latency for an application, or to memory size limitations for hardware elements. Moreover, it includes the operating system structure which defines the required services and their localization.

▸ Application Model

The application model is one of the components of the system framework. It is composed of a directed graph representing the set of tasks in which the application is split, and the precedence order among the tasks. The graph is introduced to the system as an XML file as depicted in Figure 2.3.

```
...
<task id= 1 function=function1>
    <successors nb=2>
        <succ id=2/>
        <succ id=3/>
    </successors>
</task>
<task id=2 function=function2>
</task>
<task id=3 function=function3>
</task>
...
```

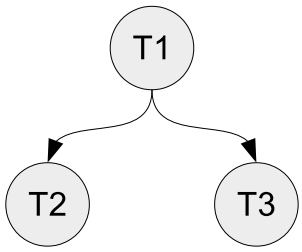


FIGURE 2.3 – XML description and the corresponding graph.

In addition, each task is related to a set of indispensable characteristics, e.g. the task's functionality. In our platform, such characteristics are called "attributes". These attributes are summarized in TABLE 2.1.

Attribute	Description
Function	Functional code written in C or C++.
Priority	Order of execution for a fixed priority scheduling policy.
Deadline	Maximum length of time to execute the task. It is related to the task's start time.
NbCycles	Execution time on the target processor.

TABLE 2.1 – Attributes for each node of the application graph.

The *Function* attribute designates the set of instructions that will be executed by the task. The task function may contain either a behavioural description written in C or C++, or just quantitative information about the time overhead. In the latter case, the platform only provides system latencies, which constitutes the primary objective. In any case, *Function* contains the Application Program Interface (API) primitives that gives access to the operating system's services.

▸ Architecture Model

The architectural specifications define the main components of the system's hardware structure (e.g. processors, buses etc.). Specifications include basic characteristics of such components, e.g. the processor's frequency and bandwidth. With the help of the platform's library, the architectural specifications can be built from predefined blocks. Currently, the components are available in two categories : Processing Elements (PE) and Communication Elements (CE). In Processing Elements, different types of microprocessors (e.g. GPP and DSP) can be found. The Communication Elements (CE) category is composed of components that ensure communication between PEs (e.g. Bus, Crossbar, Network). Memories and Timer peripherals can be also found. TABLE 2.2 lists example of elements that are currently implemented and their corresponding attributes.

Component	Attribute	Description
Processors	Clock frequency	Processing speed in cycles per second.
Memory	Size	Capacity in number of bytes.
Timer	Period	Regulates the execution of the scheduler.

TABLE 2.2 – Attributes for the architecture components.

▸ RTOS Model

The RTOS model describes the characteristics that an operating system should implement for the targeted system. For example, these characteristics ensure the right progress of the application tasks and the

handling of communication with other peripherals. In the proposed platform, designers also determine the structure of the operating system that better fits their system. Like in the other models, the RTOS model is also built from elements that are available in a library. Such elements are also characterized by predefined attributes. The RTOS model is detailed in section 3 of this chapter.

2.2 The Dogme Tool

Due to the complexity of the exploration process, it has clearly appeared that HW/SW designers should rely on tools to apply the OveRSoC methodology. The DOGME (Distributed Operating system Graphical Modeling Environment) software provides an integrated graphical environment to model, simulate, and validate the distribution of OS services on an RSoC. The goal of this tool is to ease the use of the exploration methodology and to generate a complete executable model of the RSoC platform (hardware and software), automatically. The automation is based on a flexible SystemC model of the RTOS that consists of a package of modular services. To develop each service, an Object Oriented Approach has been adopted and implemented using the SystemC 2.2 library. This tool allows an application-specific RTOS to be built by assembling generic and custom OS basic blocks using a graphical editor. The application is linked to the resulting OS thanks to a standard POSIX API. Finally, the entire platform is simulated using the SystemC kernel.

Using the DOGME tool implies to follow several design stages, which are summarized as follows :

Platform Design : the design phase consists in choosing and instantiating toolbox components into the graphical workspace editor in order to assemble the OS services and distribute them onto the RSoC processing elements. At this step, designers successively, (and according to the separation of concerns paradigm), take decisions about functions mapping into threads, perform hardware/software partitioning, instantiate the required services and distribute the services onto the PEs.

SystemC source code generation : after interconnecting all components and verifying the bindings between services, the structural source code of all objects being instantiated into the platform is automatically generated.

Compilation and simulation of the platform : to complete the design of the platform, the parametrized structural SystemC description is combined with the behavioral source code of the components provided by the user. The global SystemC description is compiled and simulated.

Analysis of the simulation results : graphical diagrams are produced to visualize the evolution of the system metrics during the simulated time. This step helps designers to evaluate the current design quality. It acts as a decision guide for the exploration of the design solution space.

The DOGME tool has been designed during the OveRSoC project by Mehdi Aichouch in the ETIS laboratory. A screen capture of the tool is illustrated in Figure 2.4.

3 OS Model Description

The works presented in this section have been led in the PhD studies of Yaset Oliva who joined the OveRSoC project in 2008. In his thesis, he proposed a modular description of OS services as well as the corresponding models.

A modular organization is one of the main assets of our RTOS model. The functions that implement the RTOS mechanisms (e.g. multitasking, interrupt management) have been gathered into independent services. A service is composed of functions sharing a common concern. For example, properties that are related to a task, such as the creation or deletion are contained within the same service. This modularity is very interesting as it favours the scalability of the model. In this context, new services can be incorporated without requiring any modification of the entire system.

The Application Program Interface (API) ensures the communication between the application and the operating system services. It contains a list of definitions representing the features that services may provide to an application. Such definitions are known as "primitives" and constitute the only means by

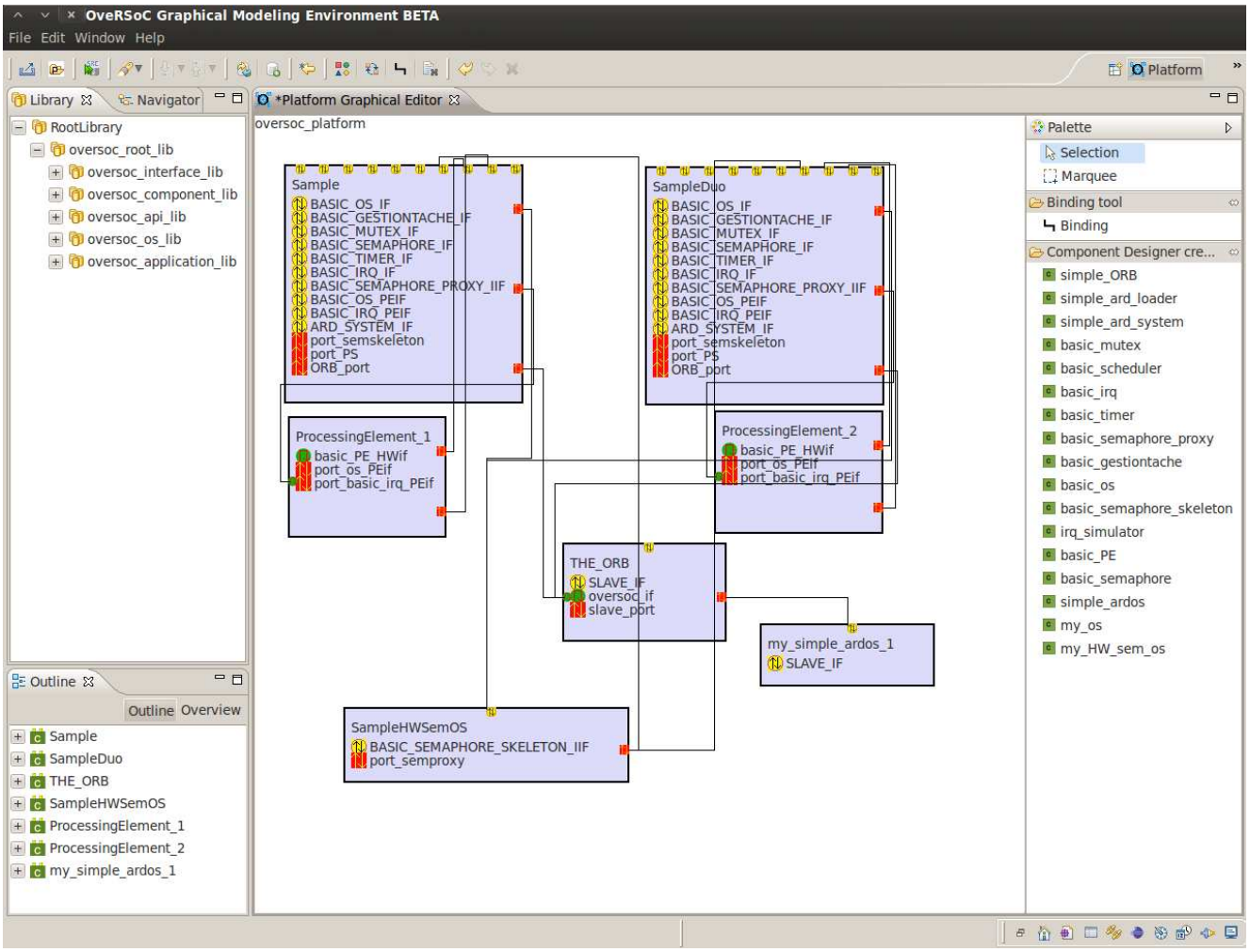


FIGURE 2.4 – The OveRSoC DOGME tool.

which a program can communicate with the operating system. In the following, the services defined in the OS model are described.

3.1 Task Manager Service

This service aims at managing the tasks' structure (i.e TCB) within the OS. TABLE 2.3 describes the primitives that are associated with this service.

Primitive	Function
OSTaskCreate	Creates an RTOS task object taking the task function as input parameter.
OSTaskDel	Deletes a task so that it is no longer considered by the scheduler.
OSTaskSuspend	Suspends a task for a given time or until it is implicitly resumed.
OSTaskResume	Resumes a task that was previously suspended.
OSTaskExec	Simulates execution times.

TABLE 2.3 – Primitives exported by the Task Manager.

3.2 Scheduling Service

The scheduling service is responsible for implementing the most important function of an operating system, that is the schedule function. The goal of the scheduler is to establish the order in which tasks are going to be implemented into the processing element.

Two types of scheduling processes are foreseen in the model. The first may be performed "off-line", which corresponds to a static table in which the tasks' order of execution has been previously stored. The second is performed "on-line".

In general, the new task selection process is straightforward. It consists in assigning a priority to each task, and then loading the context of the highest priority task that is in the ready state. How priorities are assigned to tasks is designated by the scheduling policy. In the proposed scheduling model, the selection process follows the same principle. First, the ready tasks are selected from a list of available tasks. Then, the list of ready tasks is ordered according to a predefined criterion. The criterion that is used to order the list may change from one design to another. Currently, Rate Monotonic (RM) and Earliest Deadline First (EDF) have been implemented. The type of scheduling strategy constitutes an attribute of the Scheduling service.

3.3 The IRQ Manager Service

The handling of interrupts is an indispensable mechanism for any operating system. Interrupts are a means by which the peripherals communicate with the main processor, for example I/O controllers, timers or other processing elements. If an interrupt can be serviced by the processor, a default action may be performed. The action executed after an interrupt is called an interrupt routine, and the process of reorienting the processor to the corresponding routine is called interrupt handling. Basically, when an interrupt occurs, the interrupt handler is responsible for determining the source of this interrupt. As mentioned before, several causes may interrupt a system, e.g. an event on an input port or a timer at the end of a period. In our model, the interrupt handler has been defined as a separate service named IRQManager service.

3.4 The Communication Service

Inter-tasks communication or synchronization mechanisms must also be ensured by the OS model. In order to implement this functionality, the Communication Service implements three types of mechanisms : the mailbox, the mutex and the semaphore. Mailbox objects define memory locations where to/from tasks can write/read data. Both the mutex and the semaphore provide exclusive access to a shared resource. The difference between them is that the semaphore also allows multiple concurrent accesses. Table 2.4 lists the primitives giving access to these objects.

3.5 The Intercommunication Service

It has been seen that services may need to communicate between them. For example the IRQManager calls the scheduling function from the interrupt routine. A similar situation occurs for the Task Managerservice after a new task is created. Therefore, to facilitate the addition of new services and their interconnections, a TLM based mechanism has been used. To that purpose, a special class named *oversoc_if* has been defined. This class inherits from the *sc_interface* SystemC class and thus can be used as a class template for the creation of ports. Every service inherits from this class and implements the *transport* method.

The *transport* method is responsible for translating incoming transactions into calls to the correspon-

Primitive	Function
OSMBoxCreate	Creates a mailbox object with a maximum number of N-size messages.
OSMBoxPostMsg	Writes a message into a mailbox. Blocks if the mailbox is full.
OSMBoxPendMsg	Retrieves a message from the mailbox. Blocks if the mailbox is empty.
OSMutexCreate	Creates a mutex.
OSMutexLock	Locks a mutex.
OSMutexUnlock	Unlocks a mutex.
OSSemCreate	Creates a semaphore object with an initial value.
OSSemPost	Increases the value of the semaphore.
OSSemPend	Decreases the value of a semaphore. Blocks if the value is zero.

TABLE 2.4 – Primitives exported by the Communications service.

ding functions. In such a way, it is possible to have access to service functions by just defining a port of the targeted service type and calling its *transport* method.

Additionally, since one port may be connected to only one interface, several ports would be required to connect one service to several ones. Instead, we have defined another element that interconnects all services between them. This element contains an array of *n* ports that are connected to the *n* services of the system. It also inherits and implements the *oversoc_if* so that each service port can be connected to it.

The *transport* method implementation of this element differs from the implementation inside each service. In this case, whenever a request arrives, the array of ports is examined looking for a service that is able to process the request. To do this, each request was enlarged with an identifier and a *compatible* method has been included to the *oversoc_if* class. With these two modifications, it is possible to determine to which service the request must be redirected.

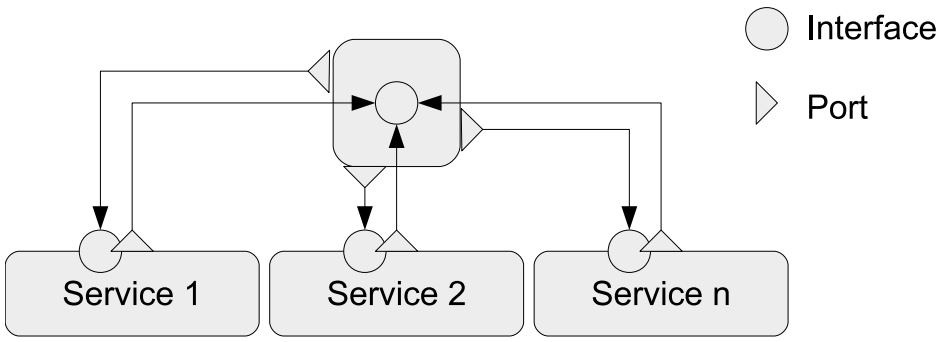


FIGURE 2.5 – Interconnection of several services.

Note that it is also possible to build a hierarchical structure which interconnects several groups of services like those depicted in FIGURE 2.5. This can be useful for systems in which the operating system is distributed between several processing elements. This is an aspect of the platform which has been studied in another work [Huc11].

3.6 The Reconfiguration Management Model

Yaset Oliva's proposed in its thesis an offloading mechanism that may be used to migrate computations at runtime in order to relieve the load of processors. It has naturally been applied to reconfigurable architectures but more generally to multiprocessors systems. In order to implement this mechanism, the concept of hardware tasks has first been introduced.

3.6.1 The HW Task Concept

A HwTask is a structure representing an off-loadable section. It contains the information that is required by the OS offloading mechanism. At present, it is assumed that a task can contain only one off-loadable section. The structure is allocated at the creation of a task, and its reference is included into the corresponding TCB (Task Control Block). Table 2.5 summarizes the most important members of the HwTask structure.

Member	Description
State	Current state
OSTask	A reference to the container TCB
NbCycles	Duration in number of cycles
BinSize	Size of the binary file in bytes
HwWidth	Width in terms of reconfigurable resources
HwHeight	Height in terms of reconfigurable resources
HwPosX	Horizontal coordinate on the RH
HwPosY	Vertical coordinate on the RH

Table 2.5 – Members of the HwTask structure.

We argue that the proposed offloading mechanisms can considerably improve the performances of embedded systems. This statement lies on the premise that the application's main computations can be offloaded onto slave processors (including RH), while the master is dedicated to the execution of the OS and other less demanding tasks (e.g. I/O operations).

In this work, an off-loadable section is defined as a part of the application code which does not contain any primitive call, and that can be executed by any slave processor. Off-loadable sections can be seen as HWTasks whose hardware and/or software implementations are stored in the system. The offloading mechanism involves several actions from the arrival of the offloading request until the end of the slave execution (in the successful case). These actions are listed below :

1. Acceptance of the offloading request.
2. Search for an available slave processor.
3. Transfer of the section's code (optional).
4. Restitution of the original task.

The offloading algorithm has been decomposed into multiple SW modules, i.e. the Dispatcher and at least the Offloader and Placer modules.

The Dispatcher module implements the OS primitive that is exported to the API. The Offloader is the most important module. It implements the decision algorithm (which is described later), and is responsible for transferring the section's code to the slave(s). The Placer module aims at finding the slave processors that are able to implement the off-loadable section. A more representative scheme of the offloading mechanism is shown in Figure 2.6.

3.6.2 The Dispatcher

The Dispatcher module implements the primitive which allows the application to designate an off-loadable section. Note the interest of changing the task's state to *offloading*. This is especially profitable in case where the Offloader module is configured as a separate hardware module. Since the processor continues executing and calls the scheduler, a new task is going to be selected as the current task. By modifying the old task's state, it is ensured that the same task is not going to be selected by the scheduler. Undetermined behaviour would result if this precaution is not taken into account.

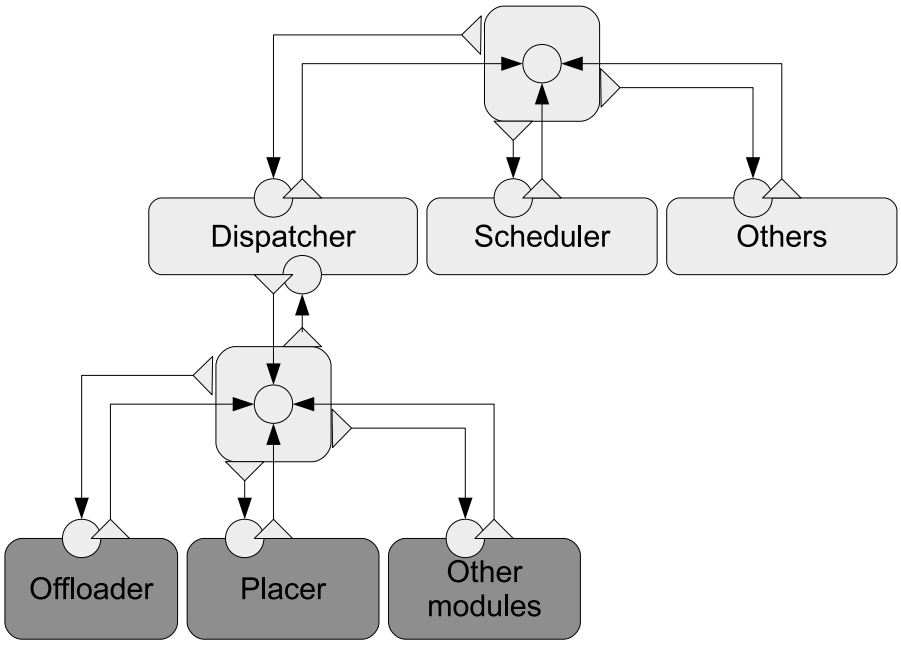


FIGURE 2.6 – Offloading mechanism blocks.

3.6.3 The placer

The placer is the module that is the closest to the hardware structure. In order to place a HwTask, the Offloader must determine available processing elements from the Placer. To that purpose, the latter must keep control of the slave processors' status. The behaviour of this module partially depends on the type of the slave processing elements contained in the architecture.

The Placer module's model implements the *PlaceHwTask()* function. In this proposed service, two types of algorithms are considered, one for reconfigurable hardware and one for a processor set. Moreover, in the case of reconfigurable hardware, the algorithm can use two different strategies : BestFit and FirstFit. Note that a mixed architecture comprising both types of slave elements is also possible. Since each algorithm executes its own placement strategy, it incurs a different timing overhead which is accumulated and simulated.

3.6.4 The Offloader

The Offloader module manages tasks' off-loadable sections. To this purpose, it implements the *OffloadingAlgorithm* function whose principle is simple. It basically consists in sending a request to the Placer, and then updates the OS task and HwTask states according to the Placer's response.

Currently, three levels of complexity have been proposed. The first level implements a direct response, the HwTask is executed by the master processor itself. At the second level, the slaves' status are analyzed. If any slave has implemented a Hwtask that is finished, it is replaced by a new HwTask. Finally, at the third level, if the previous solutions have failed, it means that all slaves are busy. In this case, a computation is performed to approximate the time delay that is required to handle the task by the processor.

In order to give an example of a reconfiguration management model, we focus here on the mechanisms that are implemented at level 3. The following algorithm computes an estimate of the time that is required before resources are available. Then it determines whether it is worth delaying the HwTask. The decision can be based on the HwTask's deadline attribute. In short, if the computed delay does not cause the HwTask to miss its deadline, it continues waiting for the required resources. Another alternative is to compare estimates of the execution end time : if the estimation of the execution end time on the slave is closer than the execution end time on the master, the task waiting for the slave is freed. In order to

compute the waiting time, estimates of the end times of each HwTask in running state are computed in a specific function.

Therefore, if the HwTask to be transferred targets a given processor, then the waiting delay simply corresponds to the estimated end time of the HwTask currently running in that processor. In the case where the target processor is reconfigurable hardware, since several HwTasks may occupy the required resources, the waiting time corresponds to the maximum end time of the HwTasks.

However, if a HwTask can be placed anywhere, the computation of the waiting time is a bit more complex. In the set of processors' case, it is required to find the earliest estimated end-time among all running HwTasks. In case of reconfigurable hardware, the placement algorithm furnishes a list of areas in which the new HwTask can be placed. Each area comprises a group of running HwTasks with their corresponding estimated end times. The estimated end-time of a group, and thus of an area, is the latest estimated end-time among all the contained HwTasks. Then the candidate area in which the new HwTask will be placed, is the area (or group) with the earliest estimated end-time among all the areas. The behaviour of the algorithm is represented in Figure 2.7.

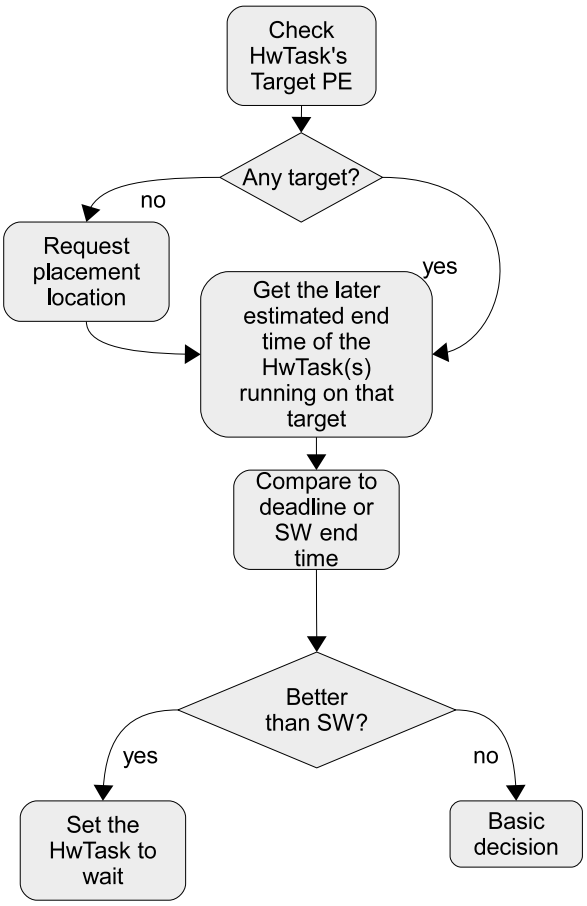


Figure 2.7 – Scheme of the ***EstimateEndTimes*** algorithm.

4 Modelling Evaluation

The role of this section is to demonstrate the use of the OveRSoC methodology on a real use case. The proposed system consists of a terminal whose mission is to decode the signals it receives through a wireless channel (see Figure 2.8). This type of system is typical in the wireless communications domain. This use-case has been studied in the context of the GDR-ISIS, young researcher project in collaboration with Laura Conde-Canencia of the Lab-STICC Laboratory.

4.1 Description

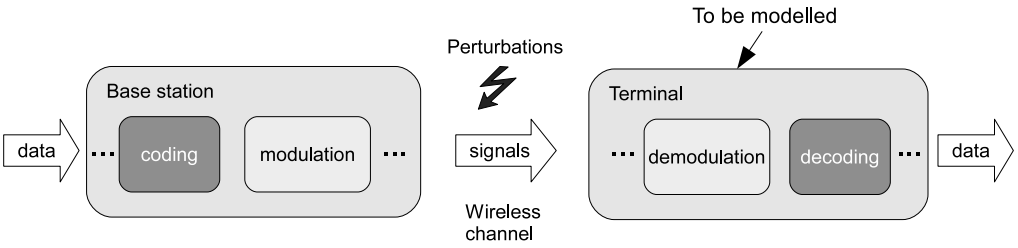


Figure 2.8 – Target system.

In this application, high-order modulation schemes and error-correcting codes are jointly used to transmit more bits per Hz bandwidth and maximize performance. However, as the radio signal often propagates in a varying environment, the transmission scheme should be able to adapt itself to the state of the channel in real-time. A promising scheme is the Adaptive Coded Modulation (ACM) [CCEPO12], where the transmitter switches between signal constellations (of varying size) and code rates at discrete time instants. In other words, at a given time, the transmitter chooses symbols from the biggest constellation meeting the Bit Error Rate (BER) requirements and thus ensures maximum spectral efficiency for a given acceptable BER.

The ACM technique consists in choosing the Modulation and Coding Scheme (MCS) pattern that maximizes the system throughput, while guaranteeing an acceptable BER. Each MCS is characterized by a modulation (of order M) and a coding rate (R). Besides, M and R are used to calculate the spectral efficiency (η) of a communication, according to the equation :

$$\eta = R * \log_2 M$$

where η is defined as the maximum throughput (in bits/s/Hz).

In this case study, we consider an application which is compatible with the WiMAX 802.16m standard [CCEPO12]. In this standard, 12 MCSs are supported. These MCSs are detailed in Table 2.6, which is to be read as : MCS_1 corresponds to a QPSK modulation (M = 4) and (R = 1/2) or MCS_7 corresponds to a 16-QAM (M = 16) and (R = 3/4).

MCS	Modulation	Code rate
1, 2, 3, 4	QPSK	1/2, 2/3, 3/4, 5/6
5, 6, 7, 8	16-QAM	1/2, 2/3, 3/4, 5/6
9, 10, 11, 12	64-QAM	1/2, 2/3, 3/4, 5/6

Table 2.6 – MCSs in the WiMAX 802.16m standard. Source [CCEPO12]

Regarding the channel (or error-correcting) coding, this work has considered COD codes [DM98]. A Non Binary Low Density Parity Check (NB-LDPC) code is defined by an ultra-sparse matrix, characterized by specific parameters such as the frame length (N) and the code rate (R). In the ACM context, the system switches from one code rate to another depending on the channel state. Also, an NB-LDPC over GF(64) coding has been considered, thus each symbol corresponds to 6 bits and a frame contains 192 symbols.

Figure 2.9 shows η as a function of SNR for two decoders and two types of channels using the ACM technique. The decoders are the Convolutional Turbo Decoder (CTC) and NB-LDPC while the channels are, the Additive White Gaussian Noise (AWGN) channel and the ITU Pedestrian B (at 3 km/h) using a bandwidth of 20 MHz. The curve marked with the arrow indicates the behaviour considered in our case : COD coding and AWGN channel.

For our case study, we defined a scenario where signals are constantly sent to the receiver. Also, at given intervals, channel conditions are analyzed. As soon as the channel changes, the system must be reconfigured without interrupting the transmission. Moreover, each signal is equivalent to a frame and a

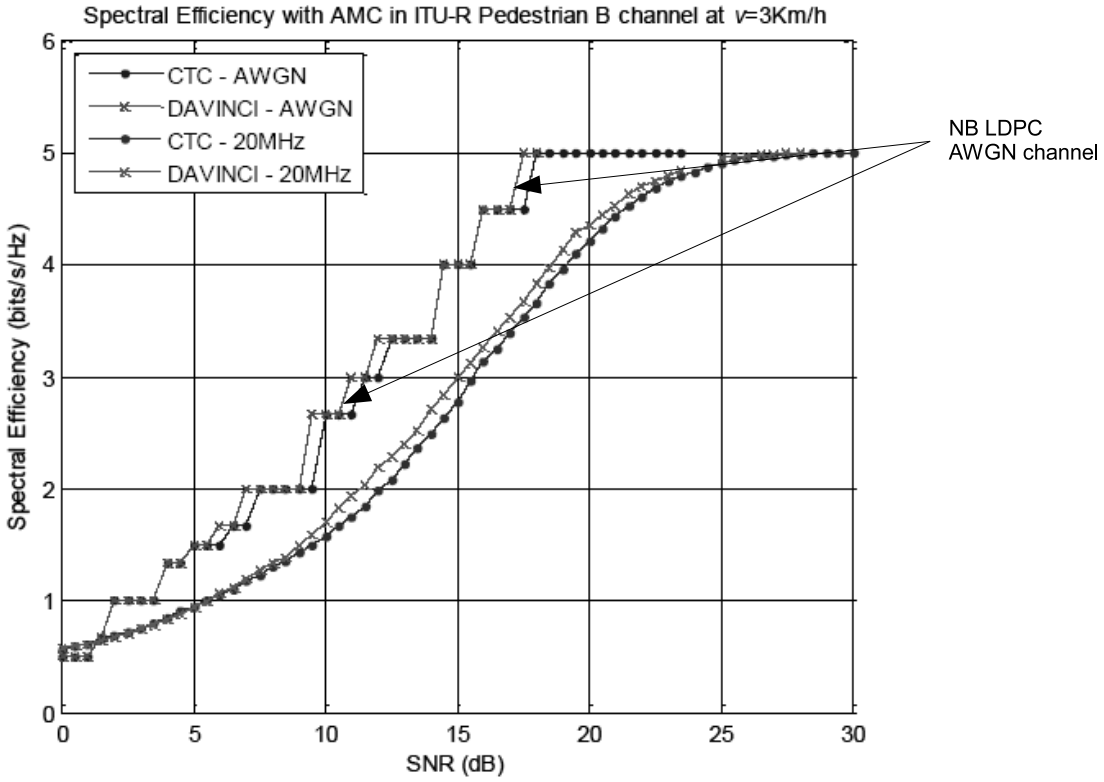


FIGURE 2.9 – Spectral efficiency as a function of the SNR with ACM. Source [CCEPO12]

pause is required between the transmission of two frames. This is due to side effects of the channel, e.g. multiple paths. In our case, this pause lasts 10 μs, which corresponds approximately to 10% of the frame's duration.

The tasks that constitute the system are the following :

The sensor task is responsible for inspecting the transmission channel and estimating the SNR. Whenever the channel conditions change, **sensor** configures the MCS that is the most suitable for the next transmission. Afterwards, **sensor** locks on a MUTEX until the next transmission is started.

The MCS_c task represents the MCS scheme that is being used for the current transmission. At the end of each frame, the task is suspended for the inter-frame time. After the pause, MCS_c unlocks the MUTEX that is required by **sensor**, to indicate that a new transmission is going to be started. Every time a channel change has been indicated by **sensor**, the $\mathbf{MCS_c}$ activates the next MCS scheme and deletes itself.

The MCS_n task represents the MCS scheme that has been configured by the **sensor** for a future transmission. Whenever a change on the channel's conditions is detected, this task becomes the current MCS.

4.2 System Model

4.2.1 Application Model

TABLE 2.7 lists the attributes of each system task. The execution time of the sensor task has been estimated according to the results of previous implementations of communication systems. Assuming a basic channel estimation, the time of this task is negligible with respect to the frame processing time. Regarding the MCS tasks, their execution times as well as their bitstream's sizes have been extracted from an implementation on a Virtex5 FPGA [CCLJ12]. Note that the Columns and Rows of the MCS tasks

correspond to the number of columns and rows of the FPGA in terms of reconfigurable resources.

Task	Type	Mean cycles	Columns (resources)	Rows (resources)	Bitstream size (Mb)
SENSOR	SW	1000	-	-	-
MCS_1 to MCS_{12}	HW	8750 to 18350	20	10	2.25

TABLE 2.7 – Task models and their attributes.

4.2.2 Architecture Model

The dynamic behaviour of the ACM-based terminal requires a flexible architecture, which is able to adapt to new channel conditions. In addition, MCS operations require considerable computing power and should ideally be implemented as dedicated hardware elements. Given these characteristics, an architecture containing a reconfigurable part has been selected. Moreover, we have considered the use of two processors, one master and one slave. The master is responsible for executing the kernel functions. The slave processor executes the application tasks. The attributes of the architecture model are summarized in Table 2.8.

Element	Attribute	Value
Timer	Frequency	50 MHz
Processor	Frequency	50 MHz
RH	Width	200 columns
	Height	200 rows
	Frequency	50MHz

TABLE 2.8 – Architecture model attributes.

4.2.3 Kernel Model

Considering the target system, the model of the kernel is composed of the services that are listed in Table 2.9. In this model, the Task Manager service has been parametrized to handle 13 tasks at most. In a first model, since the target architecture contains two processors, the scheduler type has been chosen to be Symmetric Multiprocessing (SMP) and the scheduling policy is classical rate-monotonic.

Service	Attribute	Value
Task Manager	MAX_NB_TASKS	13
Scheduler	Scheduling type	SMP
IRQ Manager	Sources	IRQ_END_TASK IRQ_TICK
Communications	Mechanisms	MUTEX
Offloading	ALLOW_IRQ	yes
	OFF_LEVEL	1
	PLACEMENT_POLICY	BEST_FIT_ALGO
	PROCESSORS_SET_SLAVE	no
	REMOTE_BLOCK	no
	RH_SLAVE	yes
	IRQ_TRANSFER	no

TABLE 2.9 – Kernel model.

Moreover, since HW tasks are considered in this architecture, an IRQManager is required. This manager must handle interrupts coming from hardware elements. Additionally, a tick interrupt is also provided to manage the tasks' delay. According to the synchronization requirements, it has been decided to only implement mutexes since they are simple and sufficient for this application. An offloading algorithm has also been included since reconfigurable hardware must be managed. Initially, the algorithm has been parametrized with the first level of complexity. In this configuration, the offloading algorithm will be executed only in software. Moreover, the configuration data transfer (IRQ_TRANSFER) is not preemptible. Finally, the placement policy has been chosen to be a best-fit algorithm.

4.3 Simulation and results

After the SystemC code generation step, the initial model has been simulated. The Gantt diagram displaying the tasks' executions is shown in Figure 2.10. As can be seen in this figure, the system is unable to run continuously since no more frames are decoded after the second configuration. Only MCS_{01} and MCS_{02} can be executed. Note that no other MCSs are configured and executed and that the two processors remain idle.

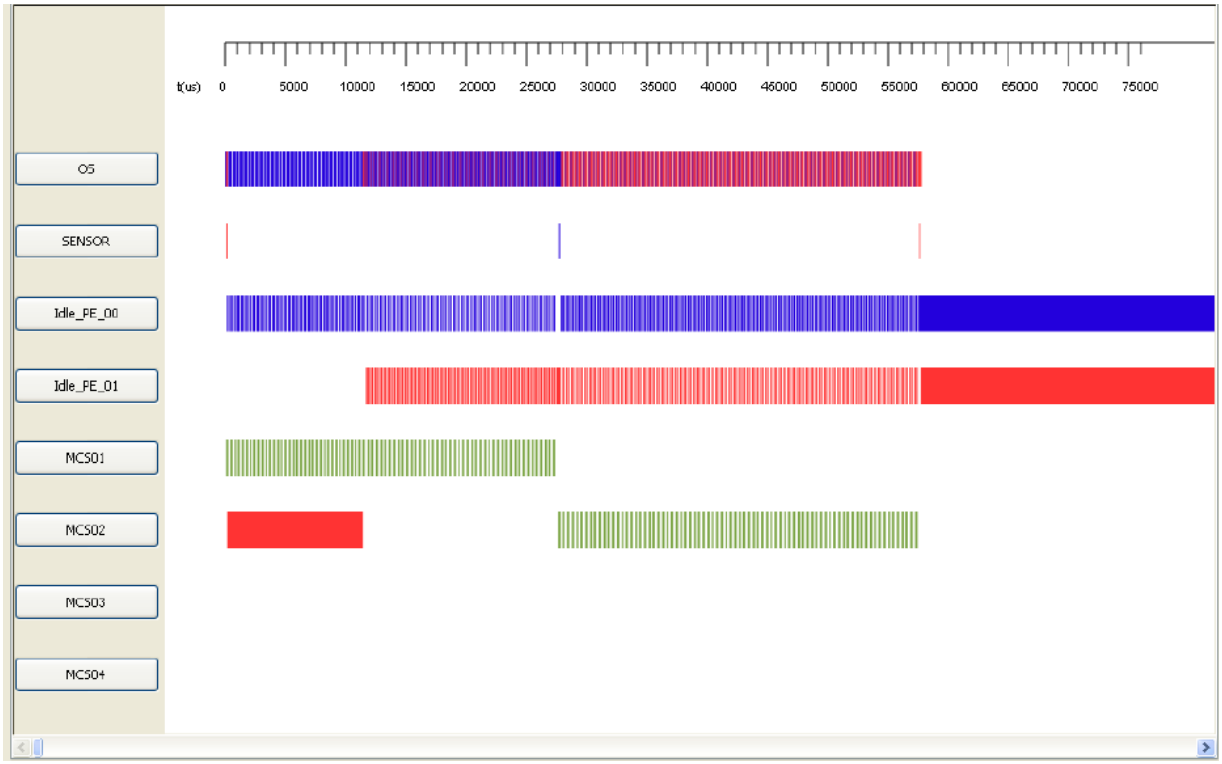


Figure 2.10 – Gantt diagram for 2 processors and the Level1-Offloader.

In this situation, it is then necessary to modify the initial model by considering different values of attributes in the kernel, application and architecture models. In this scenario, among all available attributes, it has been decided to modify the offloading algorithm complexity.

After the modification of the kernel model, the simulation has produced the results shown in Figure 2.11. Now, the system is working properly because the frames are decoded without interruption, despite the changes in MCS. All MCS are periodically executed. Since the offloading is now parametrized at level 2, the system behaves in such a way that, each time a new MCS cannot be configured, the offloader checks whether there are other idle MCs whose resources can be taken. In this case, the old MCS is replaced by the new one. For example, MCS_{03} uses the resources of the MCS_{01} which is not currently being used.

However, taking a look at the processors' usage (in Table 2.10), it can be seen that the processors are not used efficiently. Therefore, it would be interesting to consider a single processor architecture in

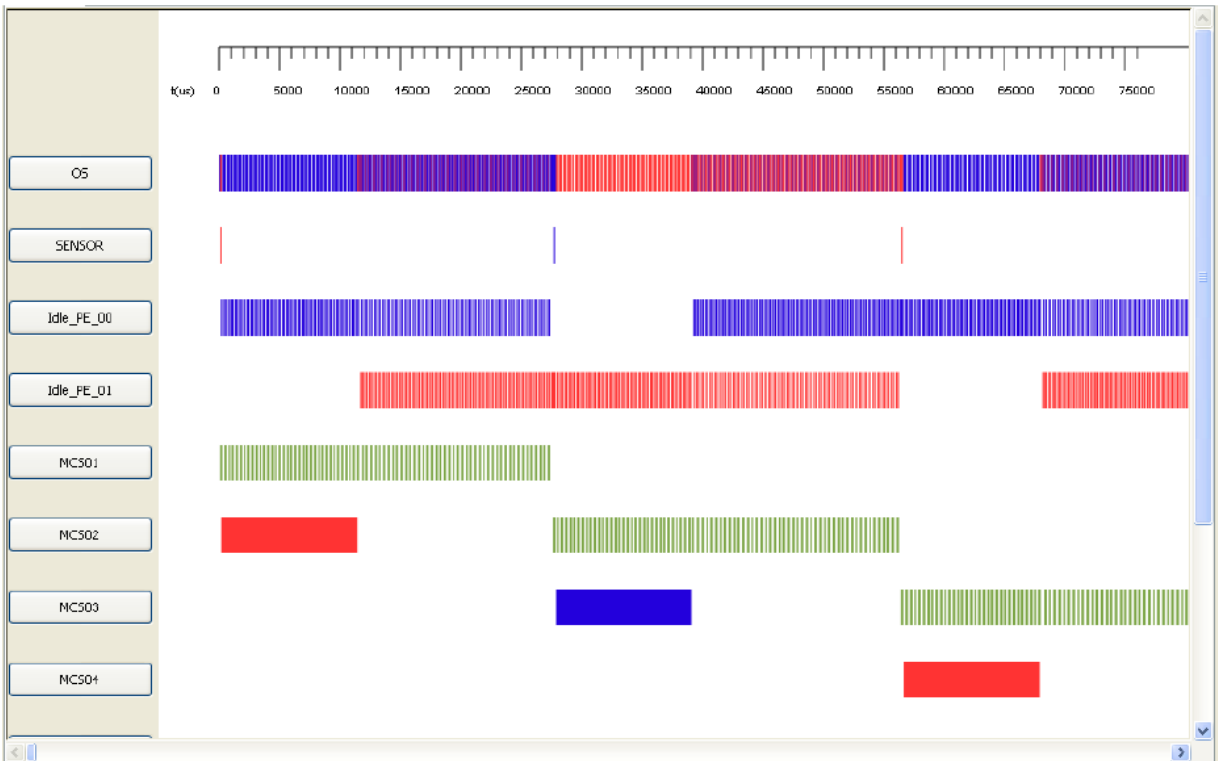


FIGURE 2.11 – Gantt diagram for 2 processors and the Level2-Offloader.

order to optimize the system. It has then been decided to modify the architecture model. In the new configuration, only one processor and reconfigurable hardware have been considered. FIGURE 2.12 shows the simulation of such model.

Processor	Idle time	Percentage of use
P_0 (master)	97.121	51.44%
P_1	56.619	71.7 %

TABLE 2.10 – Processor usage.

As shown in FIGURE 2.12, the system also works properly with one processor. However, it may be noted that the overall latency has increased. In this new configuration, processing 80 frames is performed in 37ms whereas in the previous configuration (in FIGURE 2.11), the same processing was executed in 28ms.

5 OS CODE GENERATION

One of the goal of the OveRSoC methodology was to produce binary code that could be implemented on real hardware. Regarding software aspects, simulation models have to be transformed into a source code that could be compiled and executed on the specified target. The transformation mainly concerns the conversion of the RTOS models into RTOS executable software models. Such a transformation implies the complete re-writing of the software parts in order to make them executable on the platform. Unfortunately, this operation is very difficult and requires a huge time-consuming engineering effort.

To simplify this operation, we have proposed a method based on model-driven engineering techniques. Given that software simulation models are composed of structural and behavioral components, the idea was to use a model-to-model transformation approach to automatically create an executable model representing the structural part of the simulation model. Thereafter, using the information extracted from this structural executable model, and from an existing source code that can execute on real hardware, it is possible to automatically generate executable programs. This work has been led during the PhD studies

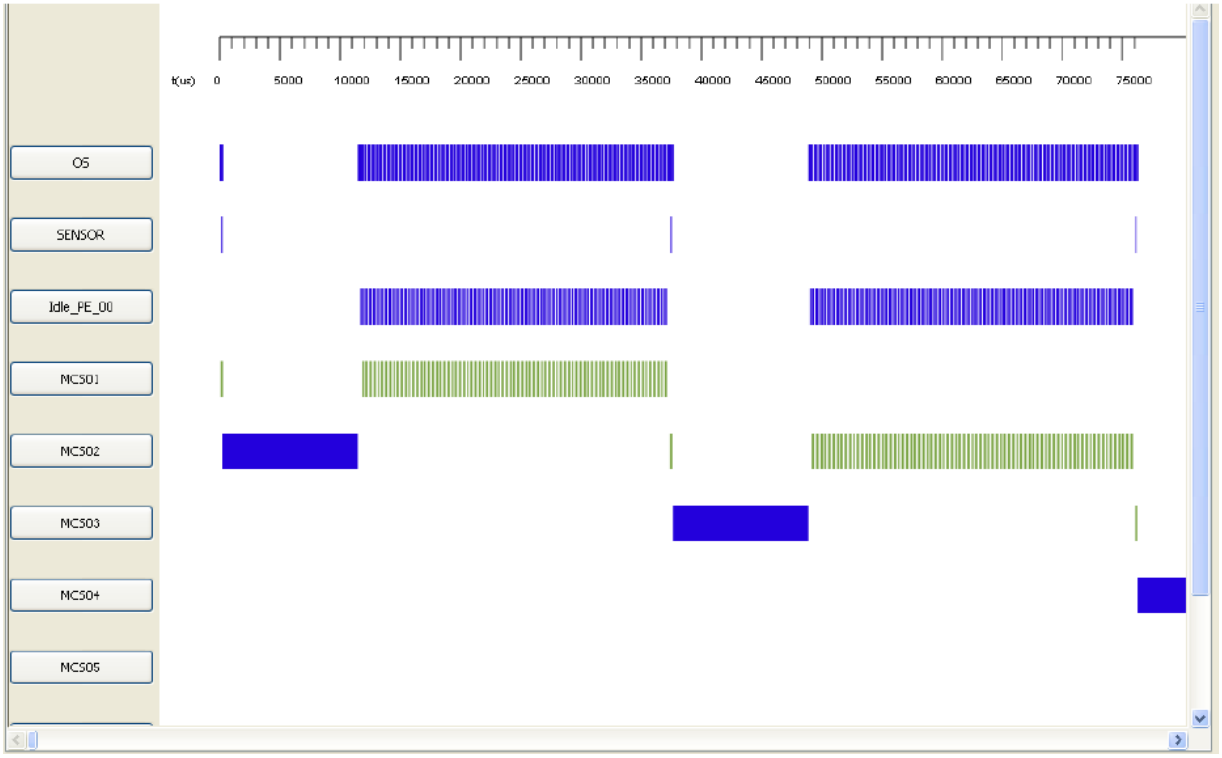


FIGURE 2.12 – Gantt diagram for 1 processor and the Offloader level2.

of Mehdi Aichouch, who defined the OS meta-model that was required in this context.

5.1 OS META-MODEL

The OS Meta-model has been implemented according to the MOF (MetaObject Facility) standard. The OS model has been defined according to three classes : the "RTOSModel", "Service" and "Operation", and two containment relationships between them. The ”RTOSModel” class has a list of ”Service” objects, and the "Service" class contains a list of "Operation" objects as depicted in Figure 2.13. These three entities and their relationships are sufficient to define the structure of an RTOS in its abstract form.

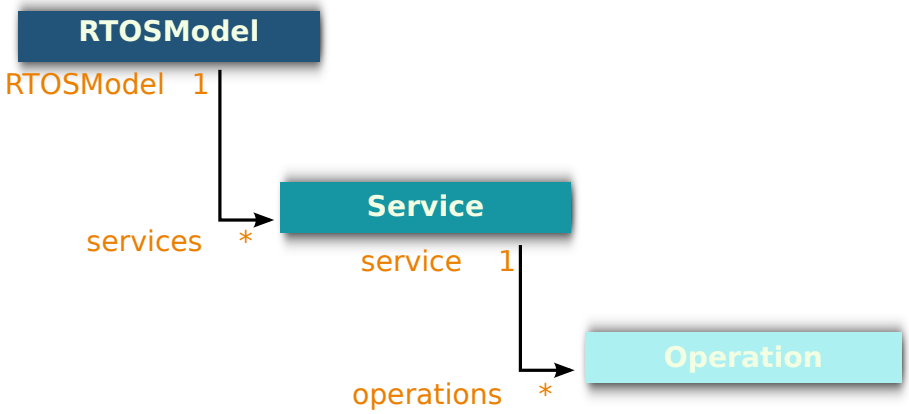


FIGURE 2.13 – Meta-model reflecting an abstract RTOS structure.

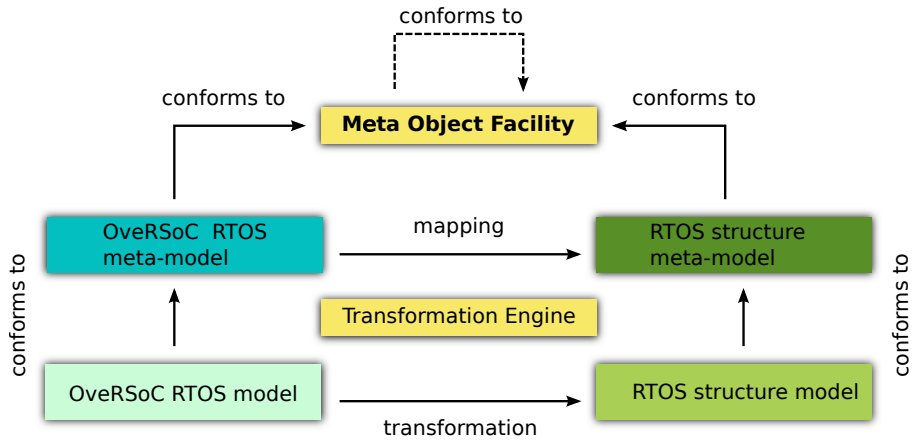


FIGURE 2.14 – Model to model transformation process.

5.2 Model to Model Transformation

After creating a model representing the structure of an RTOS and generating the final source code that is executable on a real hardware, a mechanism to transform an OveRSoC simulation model into a structural RTOS model has been implemented. This technique is called a model-to-model transformation in the model-driven engineering domain. This technique relies on the fact that any meta-model is mandatory defined using the MetaObject Facility language. So, it is possible to create a mapping between each element from a meta-model *A* and each element from a meta-model *B* as long as both meta-models are defined according to the MOF standard.

For example, we have created a mapping between the "Component" entity present in the OverSoC metamodel and the "Service" entity present in the structural RTOS meta-model. After that, this rule is used by a transformation engine to convert a model instance of the OveRSoC meta-model into a model instance of the structural RTOS meta-model, as illustrated in FIGURE 2.14.

6 From the OS to the Hypervisor

After having proposed a method of exploration dedicated to real-time OS in particular, we have extended this work to systems that may contain several heterogeneous operating systems. These systems are generally based on an hypervisor that can also be seen as a particular instance of an operating system. This hypervisor supports the execution of virtual machines that may execute other guest operating systems, with different priority levels.

One of the first studies that we have led on virtualization was the evaluation of its impact on the system's performance. In particular, we wanted to determine if virtualization was compatible with real-time constraints, and if yes, at which price.

6.1 Is virtualization compatible with real time constraints?

In his thesis, Mehdi Aichouch has studied the efficiency of virtual systems in terms of performance and adequacy to real-time constraints [Aic14a]. The purpose was to determine the feasibility of co-locating a real-time operating system and other general purpose OS on the same platform.

In other classic systems such as linux, the POSIX SCHED FIFO (fixed-priority First-in First out) algorithm is used to schedule virtual machines. Whereas this policy is efficient in the case where there is only one virtual machine that is running on a CPU, it could create a problematic situation in the case where there are multiple virtual machines sharing the same CPU. In specific conditions, this situation can even

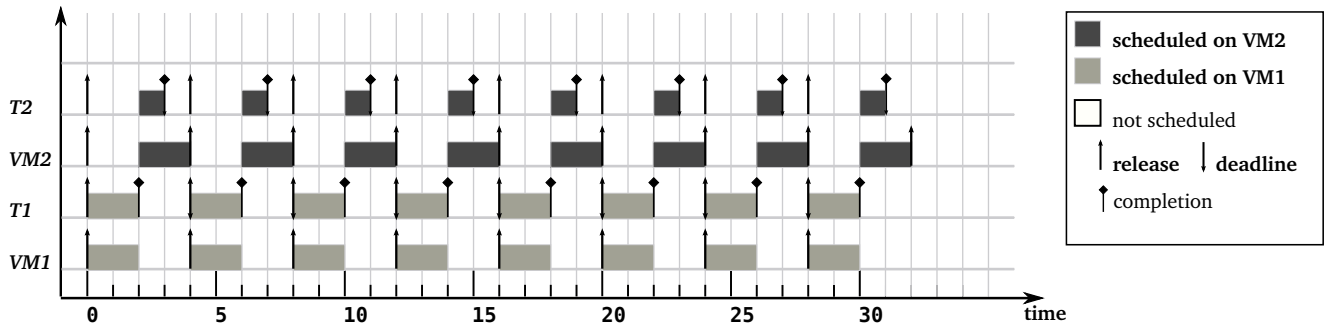


FIGURE 2.15 – Scheduling of virtual machines according to the RM algorithm.

prevent a real-time system from executing correctly.

This problem can be partially resolved by adopting a scheduling method that enforces the temporal isolation between virtual machines. Such a scheduling method defines for each virtual machine a tuple (Θ, Π), where the budget θ and the period Π together represent the CPU share that a VM requests. The VM will receive at least Θ units of time in each period of length Π. FIGURE 2.15 shows an example of the algorithm that shares the CPU between two virtual machines. For example, by assigning $(\Theta = 2, \Pi = 4)$ to each virtual machine and setting a higher priority to VM_1 than VM_2, 50% of the CPU time is allocated to VM_1 and VM_2. Given these timing parameters, the algorithm prevents VM_1 from over-utilizing the CPU resources after consuming its budget and allocates the remaining CPU time to VM_2.

In [CPG+11], an analytical method to compute the efficient budget and period of a virtual machine is presented. The intuition of the method may be described as follows. The execution length is the largest amount of time that is taken by a virtual machine to execute its assigned budget. Its depends on the virtual machines scheduling and impacts the schedulability of real-time tasks running on them. It is then the first parameter to define in schedulability analysis.

Our study has been based on this previous work. First, we considered that the virtual machines are scheduled according to the fixed-priority Rate Monotonic (RM) algorithm. Second, we assume that every virtual machine V_l can finish executing its assigned budget Θ_l within Π_l time units from its release.

As it can be seen in FIGURE 2.16, the execution length of a VM depends on the interference from the execution of the other higher-priority VMs. To determine the time at which the VM finish executing, the worst-case response time analysis [LSD03], [ARW+10], may be used in this case. According to the rate-monotonic algorithm, the VMs with shorter periods are the highest priority VMs. Using this policy, VMs are sorted by decreasing priority. That is, the highest priority VMs are scheduled before the lowest priority VMs. Thus, the second term at the right of equation 2.1 corresponds to all higher-priority VMs in the system. This equation can be verified iteratively starting from t^1 and until $t^{(c+1)} = t^c$ is satisfied for some $c \geq 1$. The value of $t^{(c+1)}$ is equal to the V_l's worst-case response time, denoted by Ω_l.

$$t^{(c+1)} = \theta_l + \sum_{j=1}^{l-1} \left\lceil \frac{t^{(c)}}{\Pi_j} \right\rceil . \Theta_j \tag{2.1}$$

From FIGURE 2.16 we define the V_l 's execution length by :

$$L_l = \Pi_l - \Theta_l + \Omega_l$$

We denote by $d_{l,min} = min_{i=1}^{|\tau_l|}(d_i)$ the smallest deadline in the task set τ_l, where $|\tau_l|$ is the number of tasks in τ_l. We also denote by $e_{l,min}$ the worst-case execution time of the task with $d_{l,min}$. The task with $d_{l,min}$ is the highest priority task in V_l and its execution is not interrupted once V_l starts running.

We assume that Θ_l is at least equal to $e_{l,min}$:

$$e_{l,min} \leq \Theta_l$$

Given the precedent two equations, we can derive a necessary condition to ensure that V_l guarantees that all $d_{l,min}$ are respected, that is, the execution length L_l must be less than or equal to $d_{l,min}$.

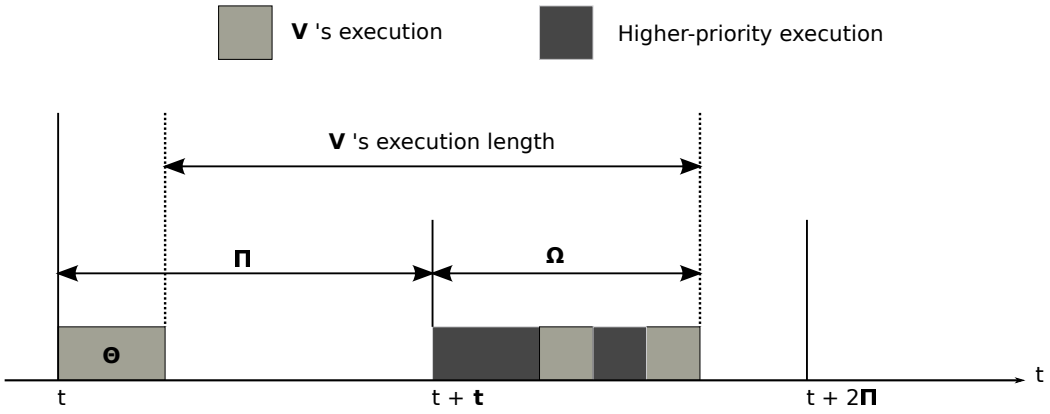


FIGURE 2.16 – Execution length.

$$\Pi_l - \Theta_l + \Omega_l \leq d_{l,min} \tag{2.2}$$

▸ Schedulability condition on a VM

After defining the condition that allows a VM to meet all deadlines of the highest-priority task, we analyze the schedulability of a real-time task set running in a VM. The worst-case execution demand of a task $T_i \in \tau_l$ within d_i time units is denoted by ω_i, and given by

$$\omega_i = e_i + \sum_{j=1}^{i-1} \left\lceil \frac{d_i}{p_j} \right\rceil . e_j \tag{2.3}$$

If we assume that all tasks in τ_l are sorted by decreasing priority, which corresponds to increasing period under RM algorithm, then, the second term at the right of Equation 2.3 determines the worst-case execution demand of all the $(i-1)$ higher-priority tasks on V_l.

And if we assume that the worst-case execution demand within d_i time units of a task $T_i \in \tau_l$ is less than or equal to d_i, that is $w_i \leq d_i$, then the necessary schedulability condition for a task T_i to meet its deadline on V_l is :

$$k_{l,i}.\Theta_l + min(\Theta_l, \alpha_l(t_i - k_{l,i}.\Pi_l)) \geq \omega_i \tag{2.4}$$

where t_i is equal to $d_i - (\Pi_l - \Theta_l)$ and $k_{l,i}$ is computed by $\left\lfloor \frac{t_i}{\Pi_l} \right\rfloor$.

The $\alpha_l(t)$ function returns the amount of time that V_l is able to run in a time interval of length t. This function takes into account that V_l is released together with all higher-priority VMs at the beginning of the interval length t. According to equation 2.4, it is then possible to evaluate the value of the tuple (Θ_l, Π_l) to make the VMs and their associated tasks schedulable on the system. The method is described in [Aic14a] and illustrated in section 6.3 .

6.2 Virtualization Overhead

In order to obtain an idea of the virtualization overhead (compared with a native system), we considered the delay incurred by a real-time task when it is released in a virtual machine. We described also how this delay could be divided into four steps. In the following list of overheads and latencies, we have re-defined each step according to the overhead or latency that is specific to the operating system kernel. We then measured each overhead and latency separately in order to observe where the bottleneck is.

The different delays that are involved in these steps are :

— Event Latency (Δ^{event}) is the delay from the raising of the interrupt signal by the hardware device until the start of execution of the associated interrupt service routine (ISR).

— Release Overhead ($\Delta^{release}$) is the delay to execute the release ISR. The release ISR determines that a job J_i has been released and updates the process implementing a task T_i to reflect the parameters of the newly-released job.

— Scheduling overhead (Δ^{sched}) is the time taken to perform a task selection.

— Context-switch overhead (Δ^{cxs}) is the time required to perform a context switch.

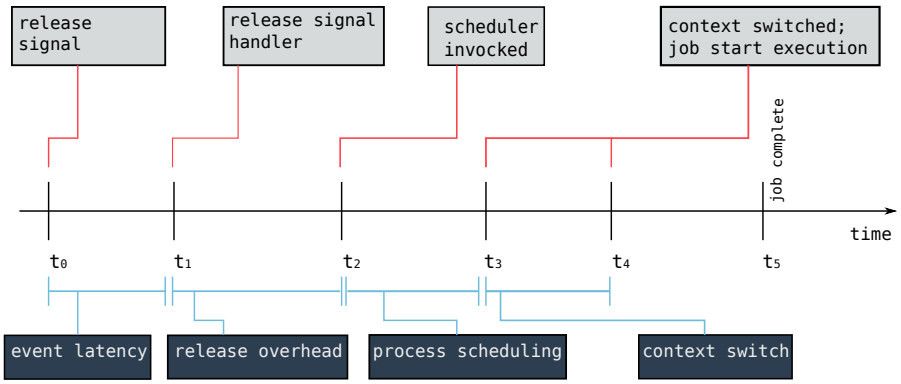


FIGURE 2.17 – Real time task latency.

FIGURE 2.17 illustrates the overhead and latencies that are ordered on a timeline. We have evaluated two different configurations : the native and virtual configurations. In the native configuration, we made use of a dual core Intel 1.86 GHz as a hardware platform. Real-time Linux, LITMUS^{RT} [BBC+07] was used as a native RTOS and implemented on the platform. In the second configuration, a real time Linux patched with PREEMPT_RT was employed as a host operating system. KVM was used to support the LITMUS^{RT} RTOS on a virtual machine.

The results show that the average-case overhead and latencies of a virtualized RTOS are similar to a native RTOS, except for the event latency where a slight increase in the virtual case has been detected. FIGURE 2.18 depicts the increase of the event latency in comparison to the native RTOS.

This evaluation also showed that the obtained worst-case overhead and latency were very far from the average-case. The analysis of the probabilities of these worst-case values led us to conjecture that these events are caused by two combined factors : 1) interference from the interrupts that occurred in the host OS and 2) virtualization overhead, such as switching between two worlds (the virtual machine and the virtual machine monitor), emulation of code, page-fault, cache miss, etc.

Given the average-case performance and the lower probability of the very high overheads and latencies, we have concluded that a soft real-time application should present the same performance when it is running on a virtual machine as it is running on a native RTOS.

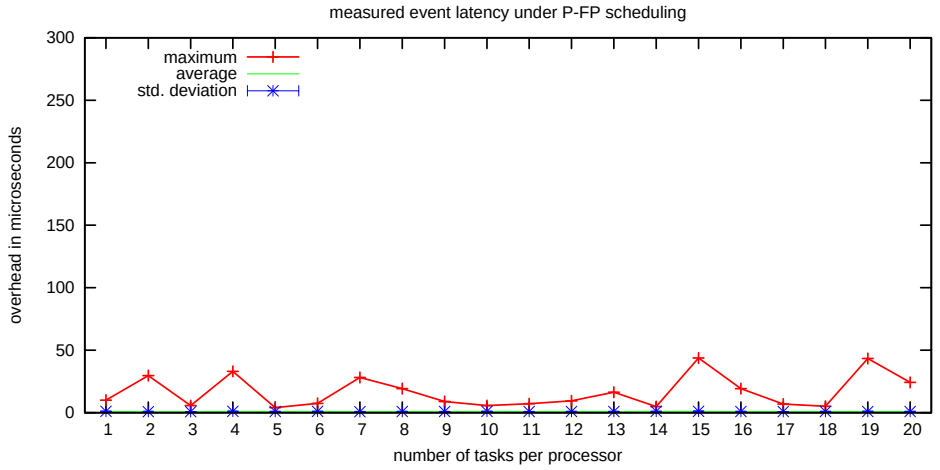
6.3 Overhead aware schedulability analysis

In order to study the schedulability of a system, the theoretical method presented in section 6.1 has some limitations. The major one is that it does not take into account the overhead observed in practice that has been described in the previous section. The idea has then consisted in integrating this overhead into a practical schedulability analysis. Since, the overhead may have different sources, a task's execution time has to be inflated as follows :

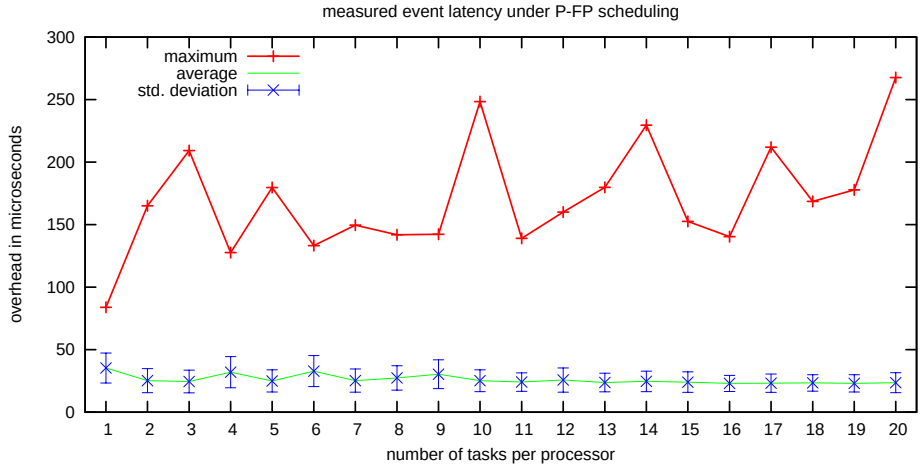
$$e'_i = e_i + \Delta^{relEv} \tag{2.5}$$

where $\Delta^{relEv} = \Delta^{event} + \Delta^{rel} + \Delta^{sched} + \Delta^{cxs}$

This method is then used to re-compute the parameters of all the tasks in a workload $\tau = \tau_1, \tau_2, \ldots \tau_n$ of each component C in the system. Using these inflated tasks' parameters and the method presented in the previous section, we can compute the periodic resource model $\gamma = (\Theta, \Pi)$ for each component C.



(a) Event latency in the native case



(b) Event latency in the virtual case

FIGURE 2.18 – Event latency in both native and virtual cases.

To validate this overhead-aware schedulability analysis, we may use the following lemma taken from [PXL+13].

Lemma

A component $C =< \tau, A >$ is schedulable by a periodic resource model Γ in presence of inflatable overhead if its inflated workload τ' is schedulable by Γ under the algorithm A when there are zero overhead.

▸ Evaluation

An empirical evaluation has been performed to validate the periodic resource model (PRM). A first experiment based on the Vsched user-level library [BD05] has been led. This library implements an EDF realtime scheduling algorithm that co-exists with the default scheduling classes of Linux. The Vsched library allows to attribute a PRM interface (Θ, Π) to each virtual machine.

In an experiment, we have tested two real-time applications. We designed the system to use two virtual machines, VM_1 executed task set 1 and VM_2 executed task set 2, the parameters of the tasks are given in TABLE 2.11. The experiment was led on an Intel core i7 2.6 GHz VT-X with 8GB of RAM. The VMM was KVM and Qemu, whereas the host OS was Linux-3.14.rc6. Linux-3.4 PREEMPT_RT was used as a guest RTOS.

The PRM parameters of each virtual machine were computed based on the necessary condition, defined by Equation 2.2 and the schedulability condition defined by Equation 2.4. The period of the virtual

Application	Task	Execution Time	Period
Task set 1	T_1	300 ms	1500 ms
	T_2	500 ms	2000 ms
Task set 2	T_3	300 ms	1200 ms
	T_4	400 ms	2400 ms

TABLE 2.11 – Simple real-time experiment.

machines VM_1 and VM_2 was set according to the necessary condition using Equation 2.2 :

$$\Pi_l \leq d_{i,min}$$

Recall that $d_{i,min}$ is the deadline of the highest priority task T_i executed on a virtual machine V_l. As in our experiment, we used the period of a task as its deadline, so, $d_{i,min}$ is 1500ms in the case of VM_1, and is 1200ms in the case of VM_2 (see TABLE 2.11). So, we assigned $\Pi_1 = 200ms$ for VM_1, and $\Pi_2 = 200ms$ for VM_2, in order to verify the necessary condition. The budget of VM_1 and VM_2 was set according to the schedulability condition which is derived from Equation 2.2 :

$$k_{l,i}.\Theta_l \geq \omega_i$$

where ω_i is the worst-case execution demand of a task T_i executed on the virtual machine that we determined using Equation 2.3, and $k_{l,i}$ is the number of times that the virtual machine V_l needs to execute before the expiration of the deadline d_i of a task T_i, and calculated using :

$$k_{l,i} = \left\lfloor \frac{d_i - (\Pi_l + \Theta_l)}{\Pi_l} \right\rfloor$$

The same budget was set to both virtual machines, VM_1 (160ms, 200ms) and VM_2 (160ms,200ms) and this budget verifies the schedulability conditions for both cases.

After executing this workload for 10 one-minute runs, we used the Deadline Miss Ration (DMR) metric to verify that no deadline was missed. The results show that we did not observe any deadline miss in both task sets. The results are presented according to two different metrics in order to observe the behavior of the system. The first metric is the job's response time of a periodic task, which is the difference between the job's finish time minus the job's dispatch time. The second metric is the job's release delay, which is the difference between the job's actual dispatch time minus its "theoretical" release time.

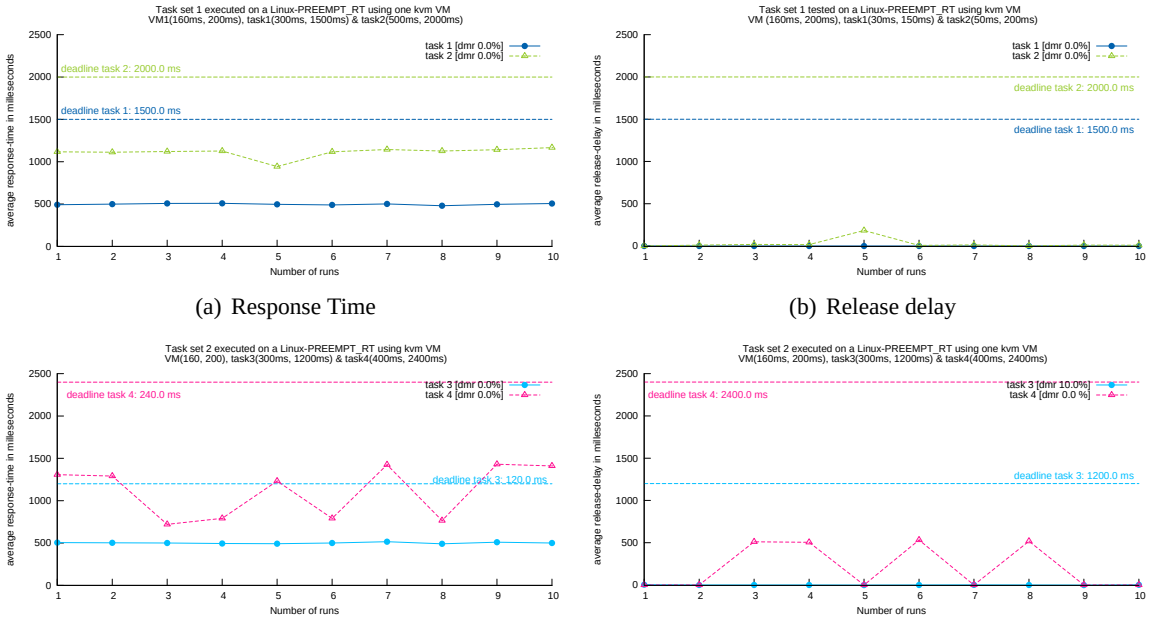
The average-case response time and release delay results are presented in FIGURE 2.19(a), FIGURE 2.19(b) for VM_1, and FIGURE 2.19(c), FIGURE 2.19(d) for VM_2. Note that observing the response time alone is not sufficient to see that a task did not miss its deadline. It is also necessary to observe the tasks' release delay to see at what time the jobs were actually dispatched. If a job is dispatched too late it could miss its deadline even if its response time corresponds to its execution time.

Comparing the trend of the response time and the release delay with the deadline of all tasks demonstrate that no deadline miss occurred. This proves the efficiency of the method to guarantee the respect of the tasks' deadlines even in the presence of virtualization overhead.

▸ Towards a custom system

In Mehdi Aichouch's thesis, we first apprehended OS virtualization and identified its various advantages in terms of performance and flexibility. We have also shown that virtualization may be compatible with real-time constraints as soon as an efficient scheduling algorithm is considered.

After studying complex kernels and hypervisors such as Linux and KVM, we have rapidly felt the need to propose our custom system of hypervision. The main reason was to build a very simple and modular system that could be designed incrementally according to the users' requirements. One of the main interest of the approach is that we are fully aware of the design steps and features of the kernel. We can then use it to test and validate new research ideas such as scheduling policies, protection mechanisms, memory management, etc. A first kernel proposal has been elaborated at the end of Mehdi Aichouch's thesis and the works have been pursued in Tian Xia's thesis. The features of our custom kernel, named Ker-ONE, are explained in the following section.



(a) Response Time (b) Release delay

(c) Response Time (d) Release delay

FIGURE 2.19 – Two tasks sets of the synthetic real-time application are executed on two separate virtual machines.

6.4 PROPOSAL : KER-ONE : A LIGHWEIGHT MICRO-HYPERVISOR

The first version of the Ker-ONE kernel has been proposed in 2013 during the PhD of Tian Xia [Xia16]. Its simplicity makes it possible to implement this light kernel in very simple systems with only few hardware resources. The typical application domain that we had in mind at that time was the IoT or sensor networks. In these domains, lots of applications could benefit from the isolation of virtual machines to provide security while guaranteeing a high level of performance. Moreover, two types of OS may be considered in such systems : an RTOS may be used to managed to most critical parts whereas a GPOS may deal with other tasks that are definitely less crucial.

6.4.1 OVERVIEW

Ker-ONE makes use of para-virtualization and follows the principle of low complexity. A focus has been made on critical virtualization functionality and non-mandatory features were eliminated. Ker-ONE is composed of a micro-kernel providing a small TCB footprint, and proposes to support RTOS that are capable of handling real-time constraints. Currently our research is based on few assumptions :

— In a first step, we only have considered single-core architectures, leaving multi-core systems to future prospects.

— We mainly deal with virtualization of simple guest OSs, instead of complex systems such as Linux, since para-virtualizing this type of OSs would be quite expensive and error prone.

— In order to implement real-time virtualization with low-complexity scheduling, we make sure that all critical real-time tasks execute in one specific guest RTOS, whereas tasks with lower priorities execute in General-Purpose OSs (GPOSs). Thus, Ker-ONE is designed to co-host a single RTOS guest and one or multiple additional GPOSs guests.

Ker-ONE consists of a host micro-kernel and a user-level environment. In its virtualization framework (see FIGURE 2.20), the micro-kernel is the only component that runs at the highest privilege level, in the supervisor mode. Since the TCB has been kept as small as possible, only the basic features that

are security-critical have been implemented in the micro-kernel : the scheduler, memory management, the inter-VM communication, and exceptions handling. The VMM is executed on top of the basic micro-kernel to support the execution of guest operating systems in their associated virtual machine.

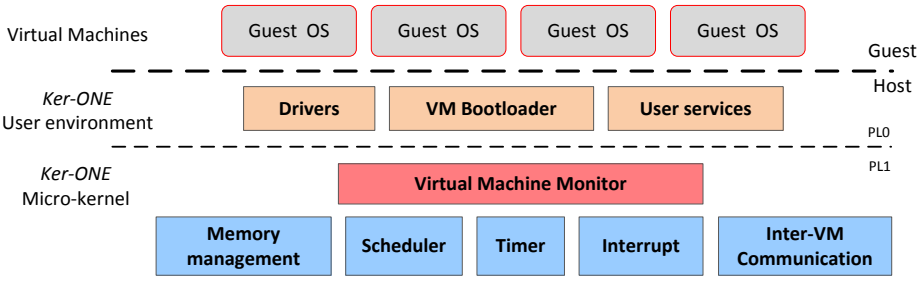


FIGURE 2.20 – Ker-ONE consists of a micro-kernel and of a Virtual Machine Monitor running in a privileged level. The User environment executes in a non-privileged level.

The kernel is based on a virtual hardware layer, emulates sensitive instructions and has the possibility to handle virtual machines' hyper-calls. The user environment runs in user mode and is composed of additional system applications, such as device drivers, file systems, VM bootloaders and special-purpose services. One of these services is the Hardware Task Manager, the role of which is to manage hardware accelerators in the partially reconfigurable FPGA area. Each VM may run a para-virtualized OS or a software image of a user application. It has its own independent address space and executes on the virtual piece of hardware that is managed by the VMM.

6.4.2 Resource Virtualization

▸ CPU Virtualization

In Ker-ONE, a virtual machine is provided with a virtual CPU (vCPU) to run on, which abstracts all processor resources. A virtual CPU mainly consists in emulating the commonly used ARM registers, which include general purpose registers (R0-R14), the state registers (PSR) and some co-processor registers (CP15). These registers are virtual in the vCPU, so that a VM executes on virtual resources instead of physical ones. We should note that, in our system, physical memory and translation page tables are managed by the micro-kernel and out of the vCPU's range.

A vCPU aims at gathering all exceptions that may occur in a VM, and is responsible for identifying and dispatching them. These exceptions include hyper-calls that are intended to be trapped or unexpected errors that may occur during the VM's execution. To speed up CPU emulation, all hyper-calls and traps are first processed in the vCPU, in which exceptions are collected, before being handled locally or re-directed to other VMM functions. Scheduling, virtual Generic Interrupt Control, inter-VM communications and memory management are examples of such functions. As an example, when a data-abort exception is trapped by a VM, the vCPU will analyze it and re-direct this exception to the memory manager for further processing.

▸ Memory Management

In virtualization, memory management is essential to provide isolated virtual memory spaces for virtual machines. Ker-ONE offers three memory privilege levels for host (for VMM), guest kernel (for guest OS kernel) and guest user (for guest OS applications).

In the Memory Management Unit (MMU) translation table, the host space is configured to be only accessible with privilege level 1 (PL1) and cannot be directly accessed by VMs. We leverage the Domain Access Control Register (DACR) in the MMU to set different access permissions in order to control the accesses between the guest kernel and the guest user. By changing the flags in the DACR, pieces of software

running in the guest user space are forbidden to access the OS kernel. This guarantees that a guest OS kernel is protected from user applications.

For each VM, an independent page table is created by the VMM. A data structure of vMMU is associated with each VM and holds the address space information. Guest OSs can change the page table mapping under the supervision of the VMM, by passing the request to the VMM via hyper-calls.

Some memory virtualization techniques, such as shadow mapping, also aim to support multiple user-level independent address spaces (i.e. user-level processes with independent page tables) to host complex general purpose OSs such as Linux. However Ker-ONE is intended to deal with very simple OSs. In this case, supporting multiple user-level protections domains is not mandatory since guest OSs only have single-domain page tables, e.g. uC/OS-ii, FreeRTOS, etc.

6.4.3 Event Management

▸ Interrupt Management

In Ker-ONE, the micro-kernel manages all hardware interrupts. When running in a virtual machine, all interrupts are forwarded to the VMM. The VMM first handles a physical interrupt by acknowledging it and by clearing its source. Then, the VMM sends a corresponding virtual interrupt to the target virtual machine if necessary. In this case, the hardware interrupts is managed in the host space, so that the VMM keeps complete control of the hardware resources. Meanwhile, the virtual machines receive and handle the emulated interrupts as if they were native machines.

The ARMv7 architecture features a Generic Interrupt Controller (GIC) to control interrupts. This controller is divided into an Interrupt Distributor module and one or more CPU Interrupt Interface modules. The role of these modules may be described as follows :

— The **Interrupt Distributor** manages interrupts' priority and provides interrupts to the target CPU Interrupt Interface blocks. The role of the distributor is to store the interrupt state as well as the context (such as enable/disable, sensitive type, etc.).

— The **CPU Interrupt Interface** performs priority masking and manages the preemption of interrupts. This interface delivers the highest priority interrupt to the processor. Interrupt handlers directly communicate with the interface to handle interrupts properly.

For virtual machines, virtual interrupts are generated by the VMM. In order to keep the guest OS IRQ handler, we have decided to maintain the guest OS original interrupts handling routine by simply providing a virtual interface to the GIC. The virtual registers are similar to the physical GIC registers. In our architecture, a custom virtual GIC (vGIC) has been designed to emulate the interrupts' management. The vGIC stores the state of virtual interrupts for each VM and emulates the GIC behavior by handling virtual interrupt states. The flow described in Figure 2.21(b) illustrates the virtual interrupts processing in guest OSs.

6.5 Performance Evaluation

In order to evaluate the performance of our proposed approach, several experiments have been performed. Some of them are described in the next sections.

6.5.1 Basic Virtualization Functions Overhead

The first experiment has consisted in measuring the overhead of fundamental virtualization functions, such as VMM scheduling, hyper-calls, interrupt management, etc. . Then the impact of virtualization on

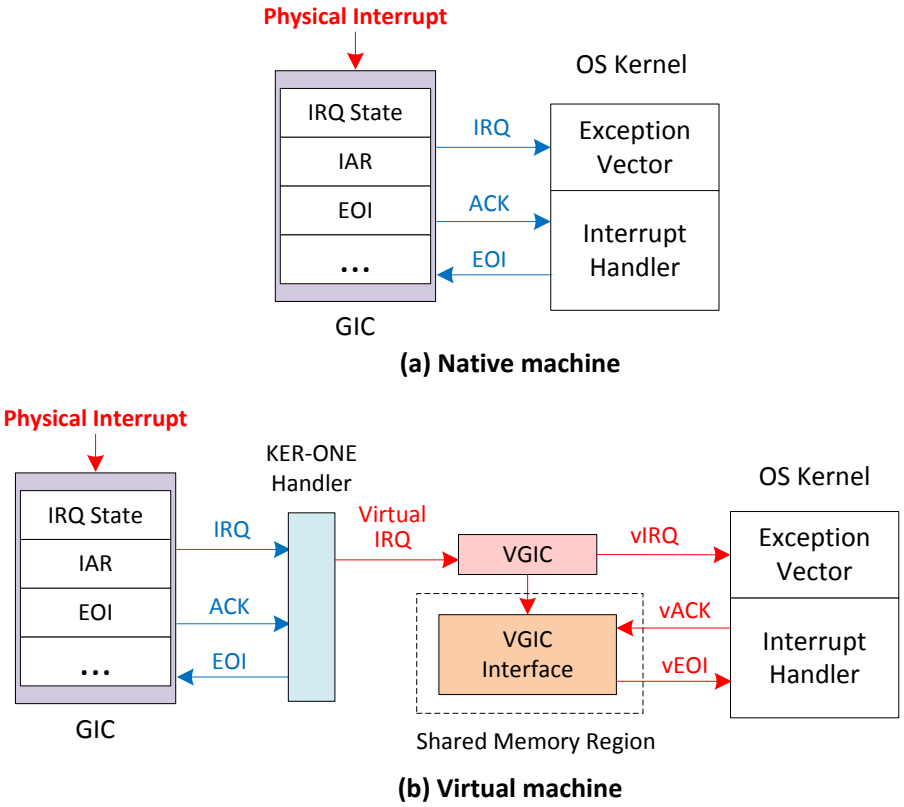


FIGURE 2.21 – Physical interrupts and virtualization through the virtual GIC.

the RTOS execution has been quantified by measuring the overhead that is due to the VM scheduling. This study has been led using a standard RTOS benchmark. Finally, our platform has been used to implement specific applications taken from standard benchmarks to demonstrate its feasibility.

In order to evaluate the performance of our platform, we have implemented multiple guest OSs (i.e. µC/OS-II) on top of Ker-ONE. These guest OS had to execute specific applications on a huge number of samples. Two main benchmarks have been considered, [thr] and MiBench [GRE+01].

In all our tests, the VMM scheduling period was set to 33 *ms*. Guest OSs used a 1 *ms* timer tick for their own schedule. These values are quite common timing configurations in this context and especially for µC/OS-II [YY14]. Guest OSs were either configured as GPOS or real-time OS according to the experimental requirements.

The different measures that have been performed in the experiments allowed us to identify the most critical VMM functions. The platform has been configured to host four similar µC/OS-II at the same priority level. These were considered as general-purpose OSs and scheduled according to a round-robin strategy. Tasks were implemented in the guest OSs and consisted in using hypercalls. The overhead of the corresponding VMM services that were required to handle these hypercalls have then been recorded by a background monitor during several hours. FIGURE 2.22 depicts the experiments results, where minimal, average and maximum overhead are presented in microseconds.

The overhead latency that is required to generate an hyper-call, to process this hypercall in the VMM and to return back to a virtual machine has been evaluated. This corresponds to the VM entry/exit latency overhead. At this point, it is important to note that hyper-calls are generally implemented within the guest OS and rarely in user tasks that are not sensitive. Since Ker-ONE is mapped to the VMs' address space, no switching between VM is required. Hyper-calls entries and exits are relatively low-cost processes since they only involve the save/restore of the CPU context.

Another very important metric is the virtual IRQ emulation latency that represents the cost of emulating a virtual interrupt for a VM. This functionality is critical for event-driven OSs and this latency has a huge impact on the events' response time. This metric is also closely related to the guest OS' scheduling overhead since a guest OS is driven by a virtual timer tick to handle virtual time.

This overhead is measured from the physical event's arrival time until the time at which the virtual

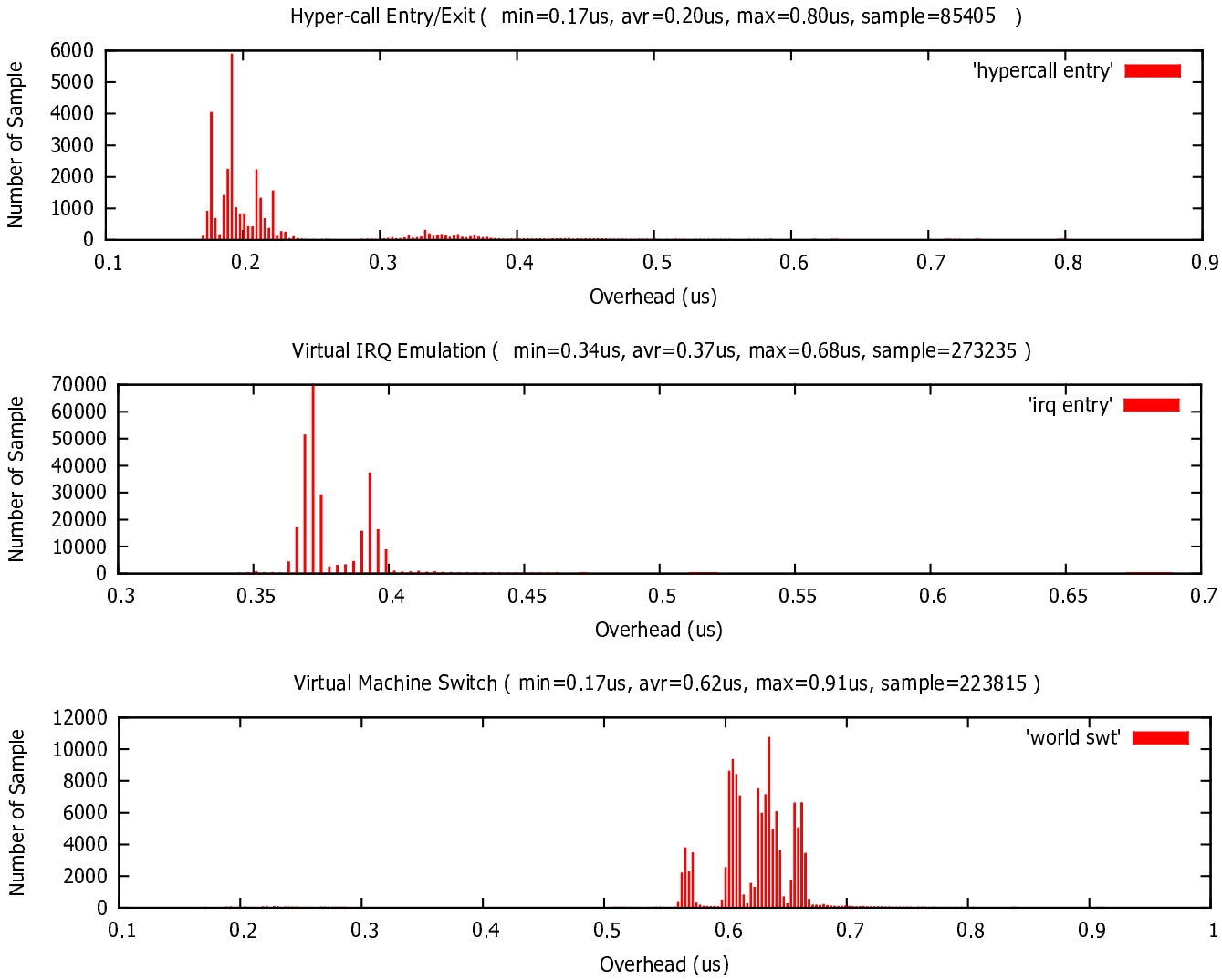


FIGURE 2.22 – Basic virtualization functions overhead in microseconds (μs) with minimum, average and maximum values.

machine is forced to its local exception vector. This process involves the handling of physical IRQ and the emulation of the virtual GIC interface registers. This latency is measured from the arrival time of a physical event until the time at which the VM is forced to switch to its local exception vector.

Finally, the virtual machines switch latency represents the cost of switching from one virtual machine to another and may be relatively heavy. The overhead of the virtual machine switch is one of the key metric in most virtualization approaches, as it is usually quite cumbersome, and has a huge impact on the VMM efficiency. In Ker-ONE, this switch is performed when a virtual machine consumes its time quantum and moves to its successor, or when it suspends itself and the VMM resumes another virtual machine. This switch includes several major procedures : (1) re-scheduling; (2) vGIC context switch; (3) timer state update; (4) address space (page table) switch; and (5) CP15 registers update. Note that changing the address space causes a higher TLB/Cache miss rate and thus increases the switch latency.

Usually, the VMM uses these functions for management and emulation purposes and they are of great importance. Virtualization efficiency is closely related to the performance of these functions. In our case, we can note that these functions exhibit a low overhead. As shown in the results, frequently-called functions, i.e. hyper-calls and vIRQ emulation can be handled in less than 1 μs. Furthermore, the virtual machine switch overhead, which constitutes the most expensive process could be limited to 1 μs.

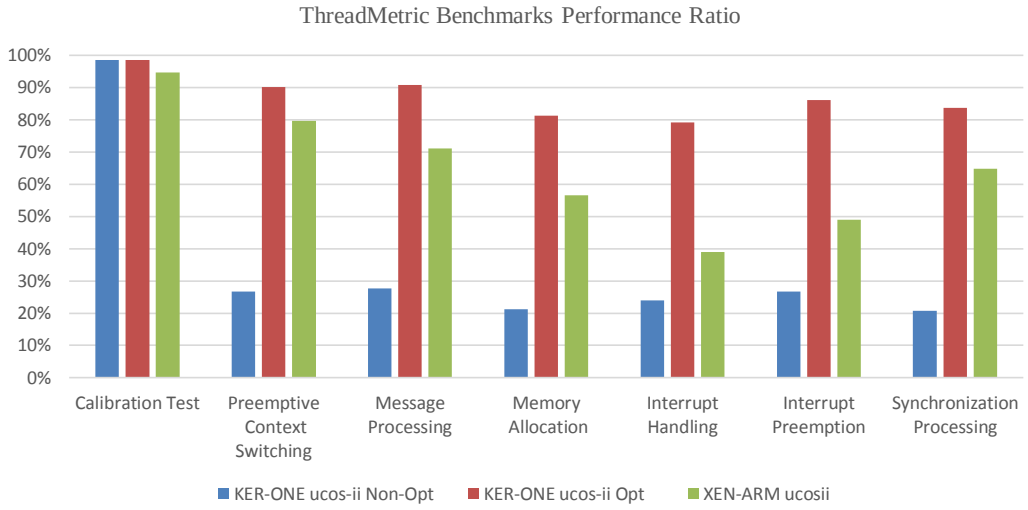


FIGURE 2.23 – Comparison of Thread-Metric Performance Ratio (R_P) for para-virtualized µC/OS-II on Ker-ONE and XEN-ARM.

6.5.2 RTOS VIRTUALIZATION EVALUATION

To estimate the impact of virtualization, two experiments have been performed. The first consists in implementing a native RTOS on an ARM Cortex-A9 processor. In the second experiment the same RTOS is implemented on top of Ker-ONE. Differences in terms of performance have been measured.

To perform the experiments, µC/OS-II has been implemented as an RTOS in a virtual machine. Three other virtual machines were used to host another instances of µC/OS-II, which play the role of a GPOS. Benchmarks have run as applications in the RTOS and for each test, a comparison between native execution and execution on a virtual machine has been performed.

This section provides the performance results of the virtual µC/OS executing the Thread-Metric benchmark suite. This suite is dedicated to RTOS performance measurement [Exp07]. Thread-Metric has been developed by Express Logic in 2007 and has been applied to several researches to measure and compare the performance of multiple RTOSs [Bes13]. In our experiment one RTOS and three GPOSs (all µC/OS-II) run on Ker-ONE. The benchmark set is executed on the RTOS. In order to acquire the performance loss due to virtualization , the benchmarks results on the native µC/OS-II has been used as a reference.

To provide an extensive evaluation, the XEN-ARM hypervisor has been employed to achieve a comparison with our micro-kernel. The XEN-ARM hypervisor Version-3.0 [HSH+08a] has been ported to our platform that is based on a Zyng 7000 device. A para-virtualized version of µC/OS-II (denoted as xeno-µC/OS), that is available on the XEN website has been used as reference. The Thread-Metric benchmark has also been executed on this kernel.

Based on the experiments, the Performance Ratio metric has been defined and denoted as R_P, which is computed as :

$$R_P = \frac{S_{vm}}{S_{native}} \times 100\%, \tag{2.6}$$

where S_{vm} is the benchmark score obtained by the guest OS, and S_{native} concerns the native OS. R_P measures the influence caused by virtualization. A better virtualization technology means less performance loss and thus a higher R_P value.

FIGURE 2.23 provides a comparison between three different systems that are implemented in the same platform. The first is the non-optimized Ker-ONE kernel. The second is an optimized version of the same kernel. The third is the Xen-ARM hypervisor. One may first note that the virtualization performance is significantly improved in the optimized version.

7 Summary

In this chapter, I have presented a modeling framework for the design of a complete reconfigurable platform including processor(s) associated to reconfigurable resources. The proposed design flow is based on a System Level modeling approach which facilitates the exploration of the RTOS services distribution both onto processors and directly inside a reconfigurable region. The main contribution of this work has consisted in proposing a unified modeling and refinement methodology for the software and the hardware parts of a dynamically reconfigurable system.

We have also proposed specific services that are required for the management of the reconfigurable resources of the architecture. Thanks to a modular and flexible modeling approach, we developed a library of generic components for the description of RSoC platforms. Among them, we developed basic hardware services such as hardware task management, hardware/software synchronization and bitstream management at high level of abstraction. The global method and the SystemC models were validated on a real application.

After dealing with RTOS modelling and implementation, we have studied more complex systems composed of several heterogeneous operating systems capable of being executed on the same processors and share the same resources. These systems rely on an hypervisor whose role is to virtualize underlying hardware resources and make them transparent to the guest OS.

We also have studied the impact of virtualization on the execution of real time applications and evaluated the overhead due to virtualization. In this context, we have proposed a new scheduling strategy based on the periodic resource model (PRM) that is compatible with real-time constraints. We have also proposed a method to evaluate the optimal scheduling parameters to allocate efficiently the CPU resources to the virtual machines.

Finally, we have proposed and designed an original hypervisor, named Ker-ONE, based on para-virtualization, with a very small footprint. Its small size makes it a very interesting architecture to be used in small embedded systems.

Summary (PhD/Dissemination/Projects)

- Thesis Supervision : 4 PhD students :Yaset Oliva, Mehdi Aichouch, Tian Xia, Ye Tian
- 20 Related Publications
- 2 ANR Projects : OveRSoC, COMPA

Reconfiguration Management

Chap. 3

This chapter presents a second research axis that I followed as soon as I obtained my position as associate professor at the Cergy-Pontoise University, back in 2003. This axis deals with Reconfigurable Computing (RC) and more particularly with the management of this reconfiguration. In my work, I have proposed 3 levels of reconfiguration management that have been studied along the supervision of several PhD thesis and students' works.

The first level i.e Hardware Layer gathers the study of all mechanisms that can be imagined in order to manage hardware reconfiguration in real-time. These mechanisms are implemented at very low level of abstraction and generally only rely on some dedicated APIs that can be implemented in simple bare-metal applications.

The second level provides reconfiguration abstraction by proposing new management mechanisms in the Operating System layer. User applications may rely on these mechanisms to reconfigure hardware parts in a transparent manner without requiring any implementation details.

Finally, a third reconfiguration level may be seen at the application layer in which user tasks may decide to reconfigure a system functionality without knowing any details about the underlying hardware or software architecture.

1 Context and Related Works

For several decades, Reconfigurable Computing (RC) has appeared as a solution to bridge the gap between hardware performances and software flexibility. Systems based on RC, are basically composed of Reconfigurable Hardware (RH) and at least one microprocessor. The RH's mission is to efficiently execute intensive computations, whereas the processor is responsible for controlling the RH configuration and executing operations that cannot be performed in hardware, e.g. data-dependent computations [CLC+02].

A main feature of RC systems is the connection between the processor and the RH. In some early architectures, the RH was designed as an independent element or a co-processor, e.g. the Splash 2 [Arn95] consists of a Sun Sparc Station processor, an interface board, and from one to sixteen Splash array elements. Other similar configurations are the PRISM II [LAL+93], DISC [Wir95] and Reconfigurable Pipelined Datapaths [ECF96].

▸ FPGAs

Since their introduction in the mid 1980's, Field Programmable Gate Arrays (FPGA) have been the typical architecture that has allows RC systems to be implemented. This is clearly related to their intrinsic nature that makes it possible to implement a rich variety of digital functions.

Initially, FPGAs may have be seen as a matrix of logic cells that are programmable by end-users at a very fine grain (bit-level). Within these circuits, logic blocks can be configured to perform complex combinatorial and sequential functions. In most FPGAs, logic blocks also include memory elements, which may be simple D flip-flops or more complete blocks of memory like Random Access Memory (RAM)s and Look Up Tables (LUT)s.

Today, FPGAs are not only a simple collection of LUTs and programmable routing. They now feature embedded multipliers, clock managers, Gigahertz-rate source-synchronous transceivers and bitstream encryption modules to protect the Intellectual Property (IP) of the design [Tri15]. More recent devices generalize the use of CPUs inside the chip to combine the flexibility of software and the performance of hardware. One example of these CPU-FPGA hybrid platforms is the Zynq device. This device has been

released by Xilinx and consists of a dual-core ARM Cortex-A9 processor which is integrated with a 7-series FPGA fabric [Xil14c]. ARM processors have also been introduced in Cyclone-V and Arria-V FPGAs Altera family [Alt15a]. Intel's solution is the Intel Atom processor E600C Series, which pairs an Intel Atom processor SoC with an Altera FPGA in a multichip package [Kon12]. In this approach, a single-core dual-threaded Intel Atom processor SoC and an Altera mid-range FPGA, Arria-II, are bonded side-by-side and linked over a PCI interface. This approach is the only CPU/FPGA hybrid processor supporting Intel x86 architecture.Figure 3.1 briefly depicts the general architecture of these platforms.

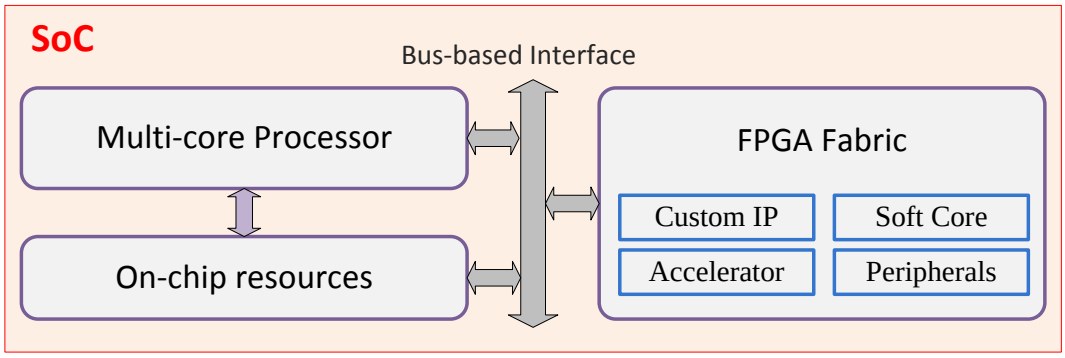


Figure 3.1 – General architecture of CPU/FPGA hybrid processors.

CPU-FPGA hybrid processors inherit the advantages of both sides. On one hand, the high-end general purpose ARM and Intel processors are capable of proving complex computer systems, and existing software stack can be directly shipped without obstacle. On the other hand, the adoption of FPGA accelerators offers a compelling improvement in performance when performing intensive computations. Moreover, with the help of CPU processing, FPGA accelerators can be managed much more efficiently with higher-complex strategies, which will inevitably enhance the acceleration. Today, FPGAs are already playing an important role in enterprise, cloud computing, datacenters, networks, and high performance computing (HPC) markets. In the future, it can be easily foreseen that a significant portion of the workload in these domains will be shifted from CPUs to FPGAs over the coming years, which may result in a performance jump.

While being considered as quite promising, CPU-FPGA systems also bring up new issues. One major challenge is how to efficiently offload software tasks to the FPGA side. FPGA resources can either be accessed as flexible input/output peripherals or be considered as co-processors with local workloads. With different strategies, scheduling, sharing and security mechanisms should be carefully discussed.

▸ Dynamic Partial Reconfiguration

Dynamic Partial Reconfiguration (DPR) has been a trending topic during several decades [BHH+07b], [Koc13] and has been included in the recent mainstream FPGA vendor devices, such as Xilinx Virtex family and Altera Stratix family. This technology allows users to reconfigure particular areas of an FPGA while the rest continues executing, as shown in Figure 3.2. After pre-compilation step, certain areas of the FPGA fabric can be defined as reconfigurable. In other words, gate arrays in these areas can be re-programmed by receiving commands and configuration information from other hardware or software modules.

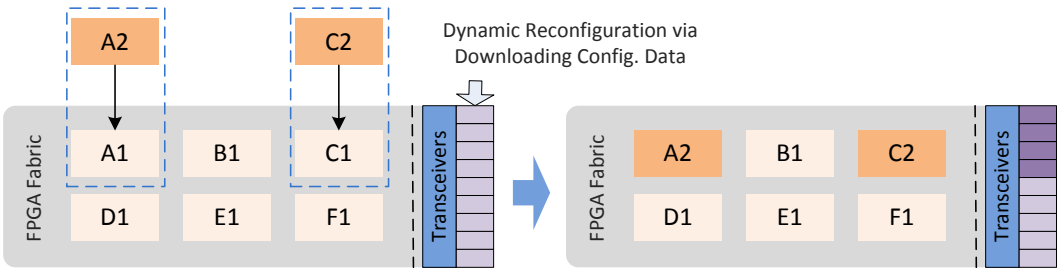


Figure 3.2 – Dynamic partial reconfiguration.

Compared to static fabric, DPR benefits from several advantages. One of the main advantages is dynamic resource allocation and re-use. Using this technique, users may implement complex architectures breaking them down into smaller mutually exclusive modules. In this case, a lot of hardware resources may be saved.

However, DPR technology still suffers from expensive reconfiguration overhead, which remains a crucial issue in practice [McD08]. In modern high-end FPGAs, which may have tens of millions of configuration points, one reconfiguration of a complex module may be very time-consuming. Especially in a computing-intensive system, where several mutually exclusive components are sharing reconfigurable resources, the time lost on reconfiguration severely degrades the overall performance [HD10]. Therefore, a dedicated efficient management associated to a high data transfer bandwidth for configuration is essential in such systems.

Exploitation of DPR has been under massive researches for the last decade. Though these researches covered various topics e.g. software frameworks design [LKLJ09], [HGNB10], [KG16], power consumption optimization, security [BGB06], a major effort has been made to provide efficient management of DPR resources in computing systems. In the following part, we introduce the related works focusing on DPR management in the context of CPU/FPGA architecture.

▸ Reconfiguration Management at OS Level

Due to the nature of heterogeneous architectures, one major challenge for CPU/FPGA systems is the cooperation and coherency between software applications and hardware accelerators. With software/hardware applications executing in parallel, the classical software issues such as task scheduling and resource sharing are extended to the FPGA side. In fact, from the software developers' viewpoint, these challenges can be abstracted and consist in mapping reconfigurable resources to the software space. In Figure 3.3, we have classified the model types of existing approaches according to their strategies, namely CPU/FPGA offload model and CPU/FPGA unified model [BPS15].

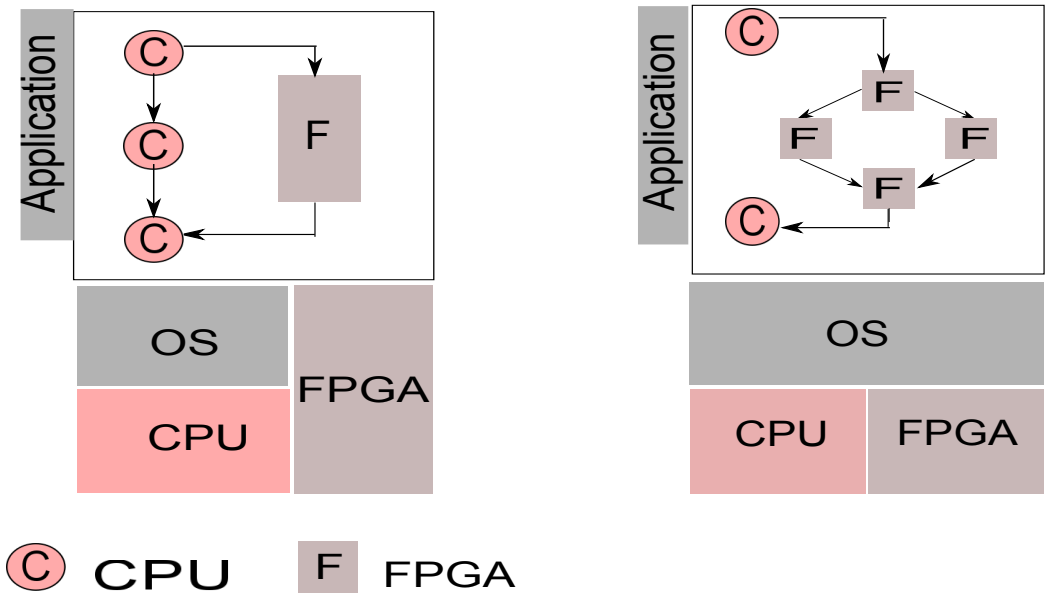


Figure 3.3 – CPU/FPGA models (offload model on the left and unified model on the right).

In the Offload Model, OS or bare-metal software applications are executing in CPU and DPR resources are used as accelerators. In this case, DPR resources can be fully exploited since they are directly accessed and programmed by applications. However, it also undermines the generality as the processing of DPR accelerators are exposed to software users. The Offload Model is often used for bare-metal applications or simple OS, whose usage scenarios are relatively simple and single-purposed. These systems require applications to fully control the behaviors of accelerators, including computation and reconfiguration. Researches for these system focus on faster reconfiguration path [HGNB10] and efficient partial reconfiguration controller, as they are critical for the overall performance. One typical approach is ZyCAP [VF14] based on ARM/FPGA system, which was proposed as an efficient partial reconfiguration

controller. This controller provides software users with an interface that permits the overlapping of software execution and hardware partial reconfiguration. Furthermore, this approach proposed a high reconfiguration throughput, by enhancing the ICAP interface with high-bandwidth Direct Memory Access (DMA). The measured reconfiguration throughput turned out to be much higher than the default Processor Configuration Access Port (PCAP).

In the Unified Model, on the other hand, CPU and FPGA are unified via well-defined interfaces. The OS is in charge of the workload allocation and migration between the two resources. In some cases, middleware is implemented inside the OS. Thus, with the help of library calls, applications can make use of both resources as OS services, without knowing the actual implementation at the physical layer. This model abstracts the CPU/FPGA platform so that user applications may exploit it efficiently. Nevertheless this approach also normally requires complicated scheduling and allocation mechanisms.

The Unified Model is based on the principle that in DPR systems, the programmable logic fabric can be interpreted as several engine containers where multiple hardware accelerators can be hosted in a time-multiplexed sharing strategy. In this case, the model of multi-kernel multi-threads [RVdlTR14] is often used to study DPR architecture [PG11] [KBT08], which defines FPGA as a group of computing agencies with multiple hardware threads, as shown in Figure 3.4. Based on this concept, CPU can treat hardware computations as schedulable hardware tasks.

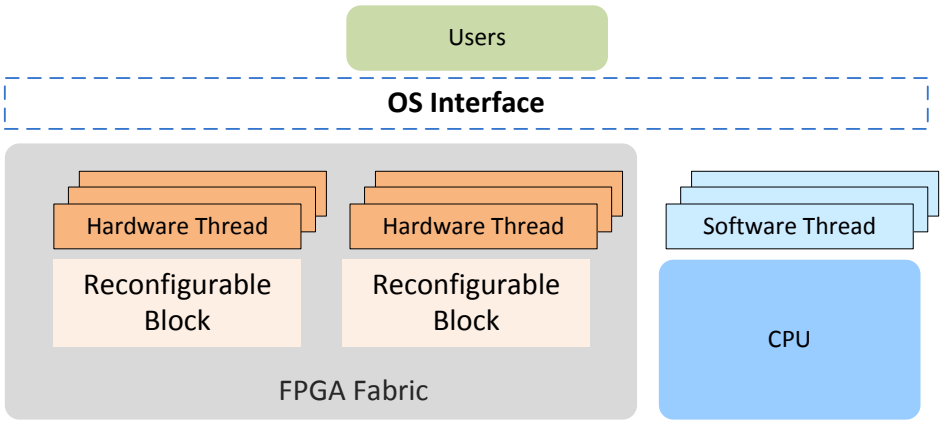


Figure 3.4 – Dynamic partial reconfiguration architecture modelling as a group of computing agencies with multiple hardware threads.

In the academic domain, embedded CPU-FPGA based systems have also been massively studied. Numerous works have consisted in providing current reconfigurable FPGA devices with OS support ([JHE+13], [GHZB10], [GHZB11], [APA+06a], [VSS+15]). One successful approach in this domain is ReconOS [AHK+14], which is based on an open-source RTOS (eCos) that supports multithreaded hardware/software tasks. ReconOS provides a classical solution for managing hardware accelerators in a hybrid system in a standard thread model. In [HH09], this concept is implemented by providing the OS4RS framework in Linux. Virtual hardware allows the same devices and the same logic resources to be simultaneously shared between different applications. However, this approach is proposed for a single OS only, without considering virtualization.

Another study is described in [WBP13]. This was one of the earliest researches in this domain. The authors try to extend the Xen hypervisor to support FPGA accelerator sharing among several virtual machines. However, this research proposes an efficient CPU/FPGA data transfer method, with a relatively simple FPGA scheduler that provides a FCFS (first-come, first served) sharing of the accelerator, without including DPR technology.

Another interesting research [JPC+14] consists in proposing a framework dedicated to hardware task virtualization on a hybrid ARM-FPGA platform. In this work, the authors modified the CODEZERO hypervisor to manage reconfigurable accelerators.

▸ Reconfiguration at Application Level

Another reconfiguration level that we were interested in is the application level.A wide range of application exploiting DPR have been studied in the literature. They mainly deal with adaptability, overhead reduction, reliability improvement and hardware computing [VF18].

In our works, we mainly focused on adaptability of hardware systems that can modify their behaviour according to the environment changes. This is extremely important for applications that require a lot of hardware resources to execute and that require huge computation efforts. One particular well-identified class of applications that we have considered is the wireless communications systems in general and Software Defined Radio (SDR) or cognitive radio in particular [MP15].

Numerous works dedicated to reconfigurable wireless communication systems have already taken advantage of the partial reconfiguration technique in FPGAs [DPML07b], [WW09]. In [Has16], a review of partial reconfiguration techniques in wireless communication systems is presented. In [Kum12] and [Kum13a], the authors propose an M-Phase Shift Modulation (M-PSK), and M-Quadrature Amplitude Modulation (M-QAM) modems implemented on FPGA with reconfigurable modulation and demodulation modules based on the partial reconfiguration technique for Software Defined Radio (SDR) and CR applications. In [Kum14], the authors propose a full reconfigurable OFDM transmitter, in which most of the blocks (modulator, encoder, and FTT) of the OFDM are partially reconfigured using the PR technique. However, they do not describe how the partial reconfiguration is managed and controlled in real-time, and under which conditions the system decides to apply the PR process.

In [SABN14], a similar work based on the DPR concept is discussed. It consists in applying DPR on the modulation and encoder blocks of the IEEE 802.11g physical layer. The OFDM physical layer is implemented with various encoding and modulation schemes to achieve different data rates using the PR technique. The design has been implemented in Xilinx Virtex-5 board. Unfortunately, in this work, the authors do not describe how they manage the partial reconfiguration process as well as the required real-time control.

The authors in [VSR+13] describe an adaptive reconfigurable transmitter for OFDM-based cognitive radio. In this transmitter, the modulation block is adapted according to the Signal to Noise Ratio (SNR) value. The authors implement a simple configuration controller but do not describe the bit-stream transfer to the FPGA. The configuration controller is look-up table based. Moreover, it is not clear whether the system can run multiple adaptive processes in parallel or not.

The authors in [DPML07a] and [DMN+08] benefit from the PR technique to propose a reconfigurable radio system. In [DPML07a], PR is applied dynamically to reconfigure a convolution encoder, a constellation mapper and a FIR filter. In [DMN+08], the PR concept is applied in a Network on Chip (NoC) architecture for SDR applications. The authors propose to benefit from the PR technique to reconfigure the communications blocks of a NoC-based SDR platform. Authors tested the proposed system with a video demonstration implementing two Virtex 4 FPGAs. The code rate and constellation mapper are dynamically reconfigured and results are evaluated regarding the quality of the video at the receiver. The partial reconfiguration management is controlled by a MicroBlaze soft processor implemented on FPGA with an ICAP controller.

At the application level, a part of our works has also consisted in studying how a wireless system can switch between several standards. This is known as vertical handover (VHA). In the context of vertical handover in heterogeneous networks, several works can be mentioned. In [KJU+15], the authors proposed an algorithm composed of two steps. In the first step, the handover is triggered according to the data rate requested by the user. Before starting the handover, the system evaluates the speed of the node. If the speed is higher than a certain predefined threshold that does not allow benefiting efficiently from switching to another system, the handover is canceled and considered as unnecessary. Otherwise, if the speed is appropriate, the available networks are compared according to five parameters : the bandwidth, jitter, delay, cost and bit error. Then, the best standard is selected and the vertical handover is performed if necessary.

Based on the SINR, the authors in [AMM12] propose a vertical handover between WIFI and WIMAX standards. According to a custom mathematical function, the equivalent SINR value for each network is computed and compared. The handover decision is based on the best equivalent SINR value.

The most promising proposed VHAs are those based on machine learning techniques [AA16]. They can be categorized as intelligence based schemes defined in [ABG14] with the other fuzzy logic based VHAs. VHAs usually detect available wireless standards and make a rough decision based on current values of specific parameters. On the other hand, VHAs based on machine learning techniques have the

ability to study and learn from the environment to take efficient decisions.

2 General Framework

As part of our work, and through the supervision of several thesis works, we have started to set up a system to virtualize computing resources. This system has been built incrementally while maintaining a modular aspect in order to relieve the development time. This system is also generic in the sense that it is not constrained by the definition of the underlying hardware platform.

In our approach, virtualization has been implemented at 3 different levels corresponding to the levels that have been identified for the reconfiguration management. These levels are represented in Figure 3.5. In this scheme, each level has the possibility to communicate with lower levels by using dedicated APIs.

The Hardware Level corresponds to the management of hardware components that are implemented in the FPGA. This level features Partial Reconfiguration Regions (PRR) that are handled by specific hardware components.These components handle bitstream manipulations to physically configure the device. This can be performed by making use of dedicated vendor interfaces such as PCAP or ICAP for Xilinx devices. These components also aim at facilitating hardware resources from upper levels.

The OS Level corresponds to the reconfiguration management from the different services that are parts of the operating system. In our model, these services are implemented in a custom micro-kernel that may be seen as an hypervisor capable of running several virtual machines. All these services allow the different virtual machines to access hardware resources in a transparent manner. Two types of virtual machines are considered in our approach : the real-time VM and non real-time VM. The first requires specific mechanisms that may be implemented within dedicated services.

Finally, at the Application Level, different tasks may need to share the same reconfigurable resources simultaneously or even take decisions regarding the configuration of the full system according to external metrics. This is typically the place to implement "smart" algorithms to adapt the platform according to environmental changes.

In the next sections, all these reconfiguration levels are described in details.

3 HW Level

In our model, reconfigurable modules are implemented in predetermined reconfiguration partitions, by downloading the corresponding bitstream files via PCAP transfer. In our designation, a reconfiguration partition is denoted as partial reconfiguration region (PRR) and a reconfigurable module is denoted as a HW task.

HW tasks provide accelerators for various functions and algorithms. Each function or algorithm is denoted as a virtual device (VD) and is completely isolated from the implementation details. Therefore, one specific device can be implemented by different HW tasks in different PRRs.

3.1 HW Task Model

In Tian Xia's thesis, we have proposed a new HW Task descriptor. This task descriptor contains the relevant information that can help in making reconfiguration decisions at different levels. The fields that are included in the HW descriptor are :

— HW task ID

— Bitstream Address

— Bitstream Size

— Reconfiguration latency

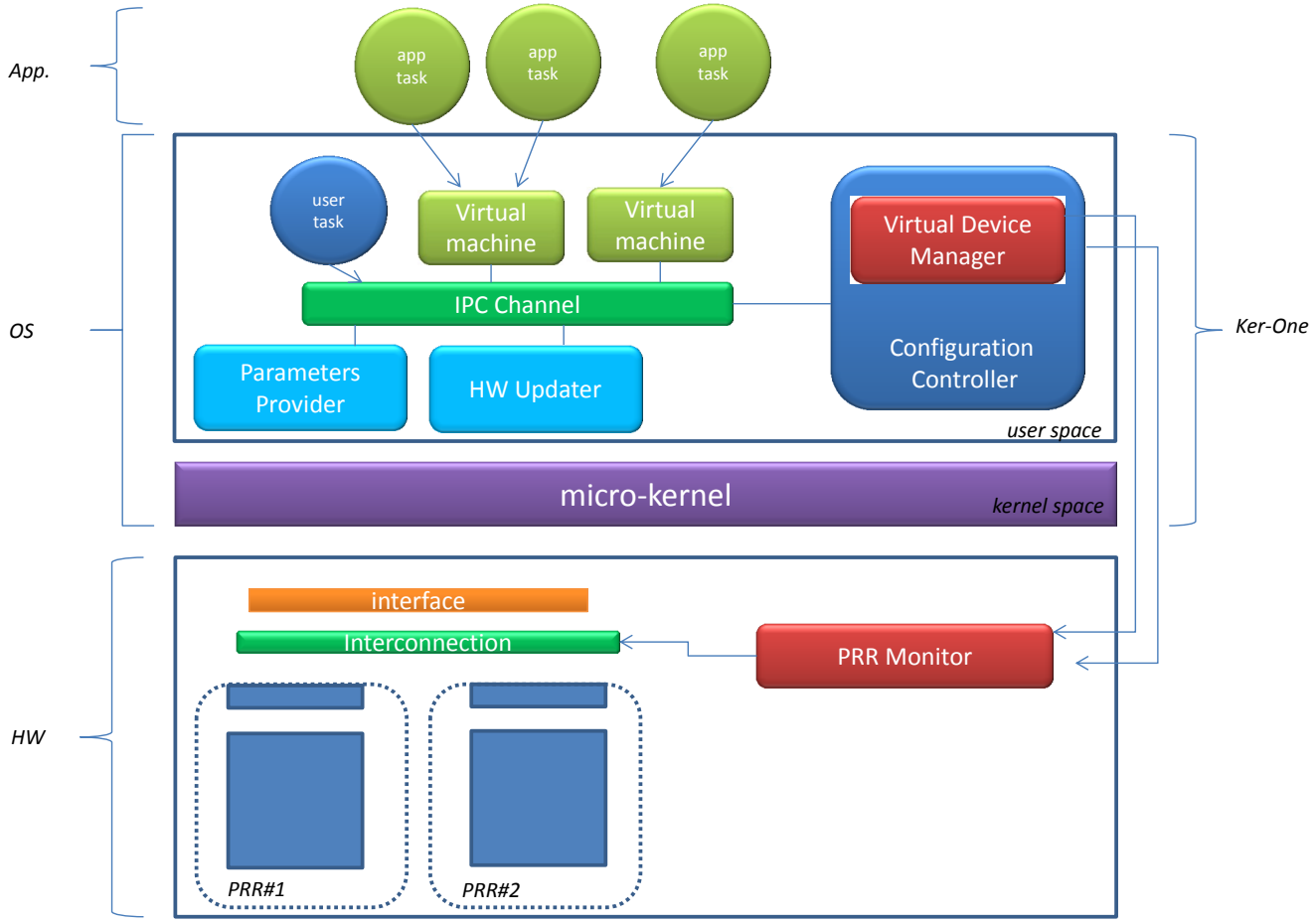


FIGURE 3.5 – Global scheme.

TABLE 3.1 – List and description of ports in PR accelerator interface.

Register	Width	Description
STAT	32-bit	HW task status register
START	8-bit	Start flag
OVER	8-bit	Over flag
CMD	32-bit	Command register
DATA_ADDR	32-bit	Data buffer address register
DATA_SIZE	32-bit	Data buffer size register
RESULT	64-bit	Computation result register
INT_CTRL	32-bit	Interrupt controller register
Custom Ports	8*32-bit	Provide 8 IP-defined ports

To facilitate the communication between tasks, software (SW) tasks running on processors, access HW tasks via interfaces. We have proposed a standard interface to facilitate the management of PR modules, denoted as PR accelerator interface. The ports of PR accelerator interface are given in TABLE 3.1 and are common to all hardware tasks.

From FIGURE 3.6, we may notice that a PR accelerator interface structure of registers is implemented in the IF. When an accelerator is disconnected with a PRR, the states of virtual device execution (e.g. results, status) are still stored in this structure in IF, so that the SW task can restart from the interrupt point of the virtual device when it gets re-allocated. In this way, the consistency of the virtual device interface is guaranteed.

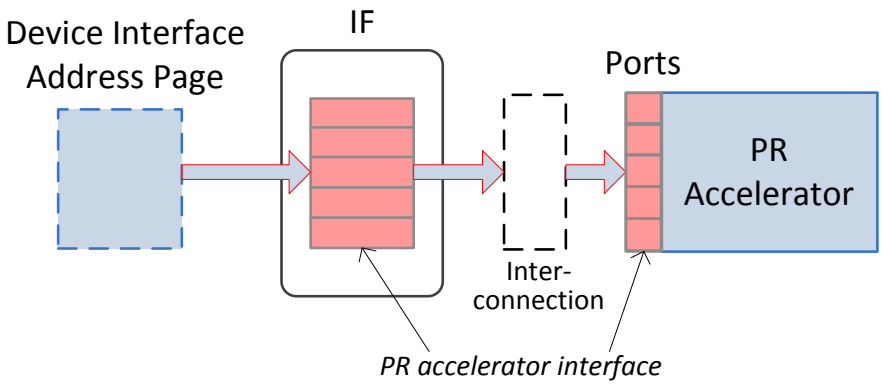


FIGURE 3.6 – PR accelerator interface.

The SW tasks that are currently using HW tasks are denoted as clients. HW tasks inherit the priorities of these SW clients. We use a preemptive policy for HW tasks, meaning that the HW task corresponding to the low-priority SW tasks can be forced to stop and get replaced by the desired HW task corresponding to the VM of higher priority.

This raises another issue that is data integrity. A running HW task cannot be stopped or preempted at any time, otherwise it may cause a loss of data consistency. For example, in some algorithms, unbreakable execution paths may exist. Therefore, designers of HW tasks have to provide the points in the code where their execution can be stopped. Moreover, these points must allow the HW task to be fully resumed from the same point of interruption. These points are denoted as consistency points.

3.2 PRR HW Management

At the HW level, it is required to store information about PRRs. This information is stored in a PRR descriptor that contains the current state of the PRR, an identifier, the device that is currently implemented, a priority as well as a reconfiguration latency.

In our model, a PRR may be seen as a five states state machine. The state flow of this machine is depicted in Figure 3.7 :

— **Idle** : The PRR is idle without any ongoing computation and is ready for allocation.

— **Busy** : The PRR is in the middle of a computation

— **Preempt** : The PRR is running, but the computation will be stopped (preempted) once it reaches a consistency point.

— **Hold** : The PRR is allocated to a VM and is preserved for a certain amount of time

— **Reconfig** : The PRR is in the middle of a PCAP reconfiguration.

As depicted in Figure 3.7, a PRR can only be directly allocated to VMs when it is in Idle state and requires no reconfiguration. In other situations, the allocation process requires extra overheads caused by PCAP transfer or preemption. In the particular case of virtualization, this causes VM requests and PRR allocations to be asynchronous.

As an example, let us imagine that we allocate PRR_1 to VM_1 via reconfiguration and that the PCAP transfer is performed in parallel with software. Then, if VM_1 gets scheduled before the PCAP transfer completes, it can only start to use the allocated device when it gets scheduled again. In this case, there is a risk for PRR_1 to get re-allocated to another VM and that VM_1 could never use the requested PRR_1.

To solve this problem, we have introduced Hold as a special intermediate state. The PRRs that are allocated to a VM will first enter this state. This indicates that the PRR is assigned to a certain VM client. PRRs in the Hold state will block any re-assignment and will wait to be used by the VM. PRRs will be

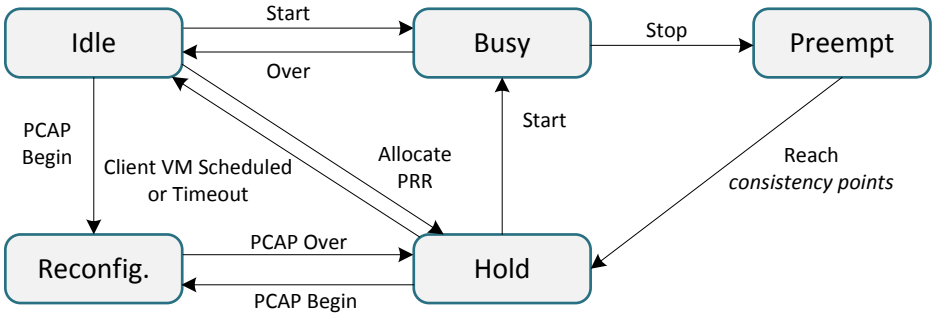


FIGURE 3.7 – PRR states.

released and return to the Idle state under two conditions : the target VM is scheduled in the CPU, or the pre-set waiting time "Expire" runs out.

3.3 The PRR Monitor

The PRR Monitor aims to take a reconfiguration decision upon requests of the Virtual Device Manager that is a software service at OS level. Each request has the following structure : Request (VM_id, dev_id, prio), which is composed of the VM identifier, the virtual device identifier and a request priority. The request priority is equal to the priority of the calling VM.

A complete solution is formatted as :

$$Solution\{\mathrm{VM}, dev, Method(prr\_id), Reconfig\}, \tag{3.1}$$

which includes the target VM, the required device, the actual allocation method and reconfiguration flag. The different methods include :

— **Assign** (prr_id) : this solution directly allocates the returned PRR (i.e. prr_id) to the request VM. If the requested device dev_id is not implemented in this PRR, a Reconfig flag will also be added.

— **Preempt** (prr_id) : this solution means that no PRR can be directly allocated, but the returned PRR (i.e. prr_id) can be preempted and re-allocated. If the requested device dev_id is not implemented in this PRR, a Reconfig flag will also be added.

— **Unavailable** : this state means that currently no PRR is available for Request (vm_id, dev_id, prio).

FIGURE 3.8 depicts the interaction between the PPR Monitor and the Virtual Device Manager. Normally, the selected solution is sent to the Virtual Device Manager for further handling. However, if there is no valid solution (i.e. Unavailable), this unsolved request is added to the search list, which is a waiting list of all unsolved requests. The PPR Monitor keeps searching solutions for requests in this list, and acknowledges the Virtual Device Manager whenever a new solution is found. The searching runs in parallel with VMs, following the priority-based FIFO principle, so that when a requests conflict occurs, the PRR Monitor always chooses the highest priority request.

4 OS Level

4.1 The Configuration Controller (Virtual Device Manager)

The Virtual Device Manager is a special service provided by Ker-ONE, running in an independent VM with the highest priority. Running in an isolated VM improves the security of its functioning. This

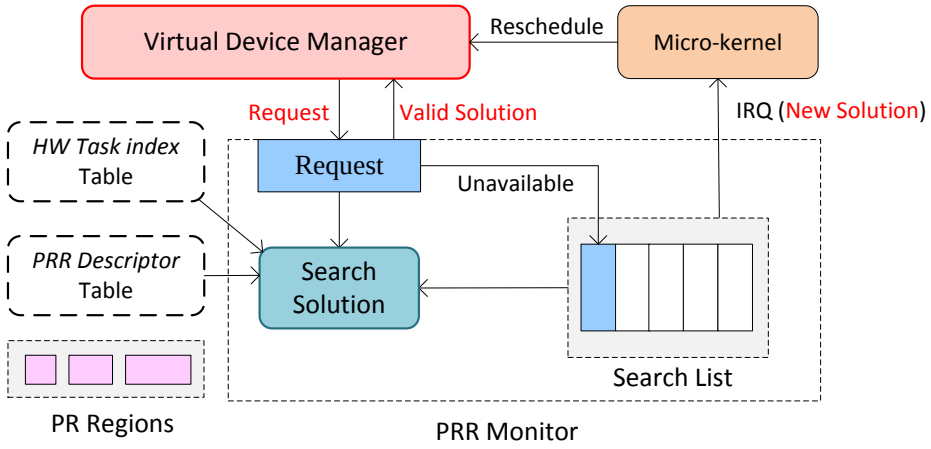


FIGURE 3.8 – PRR monitor and Virtual Device Manager interaction.

service stores all the HW task bitstreams in its memory and is the only component that can launch PCAP reconfigurations. The main task of this manager is : (1) to communicate with VMs and manage the virtual devices in their space ; (2) to correctly allocate PR resources to VMs.

In FIGURE 3.9, the process of solution handling in the Virtual Device Manager is presented. The routine is composed of one main function Run_Solution() and several interrupt handlers. Note that preemption and reconfiguration solutions are performed in two stages : (1) the manager launches the reconfiguration or preemption and then goes to sleep, (2) the manager is awakened by IRQs and completes the solution. For the Preempt solution, the manager first stops the preempted accelerator, and then handles it as Assign solutions.

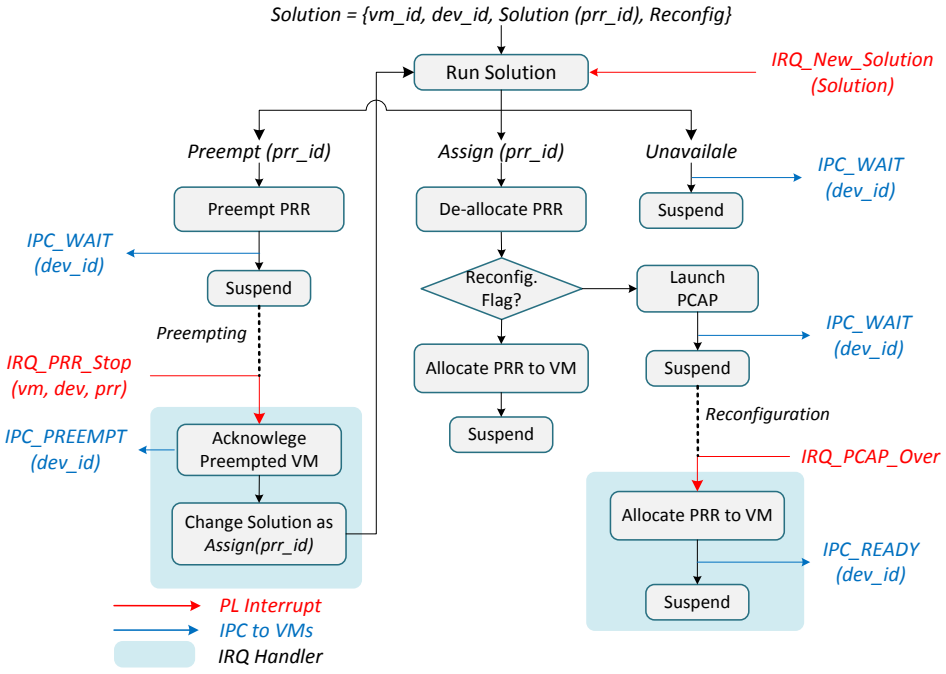


FIGURE 3.9 – Virtual Device Manager handling mechanism.

4.2 OTHER OS SERVICES TO HANDLE RECONFIGURATION

Due to the modularity of the proposed system, it is very easy to add dedicated services to be handled by the micro-kernel. Two of these services have been developed in Mohammad Rihani's PhD thesis : the Parameters Provider and the Hardware Updater.

4.2.1 The Parameters Provider

The parameters provider is another important SW service running at OS Level, in the user space. It is considered as an interface for all the parameters of the system. This module provides all the required parameters to the user processes in order to take a decision regarding the configuration of a specific module. The parameters may be collected from the HW itself (sensors, .) or from SW (task execution time, deadline, etc.)

4.2.2 The HW Updater

This module also runs as a service that can be added to the kernel. In this module, we benefit from one of the advantages of the PR that makes it possible to implement new versions of modules or even update other available modules at runtime. In a partial reconfiguration system design, PRRs may be seen as black boxes with predetermined input and output ports. The role of this service is to update a version of a given hardware task from a distant client or from the cloud.

4.3 The particular case of Virtualization : Security Mechanisms

The strong isolation among virtual machines is one of the most essential features of virtualization, which guarantees the security of each component. The sharing of DPR accelerators may undermine the system isolation since it increases the attack surface of a virtual machine. A compromised OS may try to use accelerators of another VM, or try to leverage its accelerators to attack other VM domains. In this part we discuss the potential security threats and propose our solutions to tackle these issues.

To protect the isolated environment of a virtual machine, it is mandatory to follow two principles : first, one accelerator can be shared by any VM, but should be exclusively used once it is dispatched to a specific guest OS. Second, accelerators should only access the memory region of the VM which is currently using it. Accessing a memory space outside the specific section is forbidden.

The solution for the first challenge is addressed by the allocation mechanism we described. During the allocation, the Virtual Device Manager manipulates the page tables of VMs to guarantee that a VM and a PRR are exclusively connected, so that a VM is only permitted to access the DPR resource that is allocated to it.

Meanwhile, the protection of memory space requires extra mechanisms. In classic virtualization systems, the separate execution environment relies on the MMU, which automatically controls accesses from different privilege levels and blocks illegal accesses. Isolated memory spaces are generally ensured by managing page tables. However, considering that the Zynq-7000 provides AXI-HP interfaces as bus master on the PL side, the FPGA accelerators can directly access physical CPU memory without going through MMU, which means that it is impossible to monitor and control the FPGA access via the page table mechanism. In this case, there is a risk that the accelerator accesses other VM domains or even the micro-kernel domain to attack the system.

Therefore, we created a custom unit, denoted as the hardware Memory Management Unit (hwMMU), to monitor any access to the CPU memory. The hwMMU creates a memory region table to store the physical memory regions of each VM. This table is initialized during the start-up stage of the system. Whenever a PR accelerator attempts to access the CPU memory via AXI-HP, the hwMMU checks the target address according to the memory region table, and any access outside the current client VM domain is rejected. This mechanism guarantees that DPR accelerators are strictly constrained in a determined VM domain, and are isolated from other parts of the system.

5 Application level

5.1 Context

Today, most applications require flexibility. In our model, reconfiguration at the application level aims at taking reconfiguration decisions according to specific metrics that are sensed from the environment. The main purpose of this approach consists in adapting the system according to environmental changes. At this level, decisions are taken within user tasks and may have a big impact on the system behaviour. In our approach, the main idea has always been to keep the lower layers (OS and Hardware) as transparent as possible to facilitate the decision making at the application level. Once the decision is made, it is up to the system to reconfigure itself at different levels.

The main application domain that we targeted in our works was wireless communications and to some extend software or cognitive radio. In fact, increasing the performance of wireless communication systems has always been one of the first objectives of designers. Today, due to the characteristics of wireless channels (path loss, fading, shadowing ...), the multiplicity of standards, the frequency allocations, and the mobility features of wireless devices, the operating environment is becoming more and more complex to comprehend.

In Figure 3.10, the general four steps (sense, analyze, decide, and reconfigure) applied to reconfigurable radio is presented. This concept aims to create smart radios which have the ability to reconfigure themselves to adapt to channel conditions. The first step consists in sensing the environment continuously, and then analyze and understand the environment. Upon the analysis, the system can decide whether to reconfigure itself or not. If a reconfiguration decision is performed, reconfiguration is applied and the system continues sensing the environment again. Each step of the four mentioned steps may be considered as domain of research. For example, finding the right parameters to be sensed and analyzed is a real issue to make an efficient decision. Another example deals with the decision making process itself, which can be implemented by a lot of algorithms.

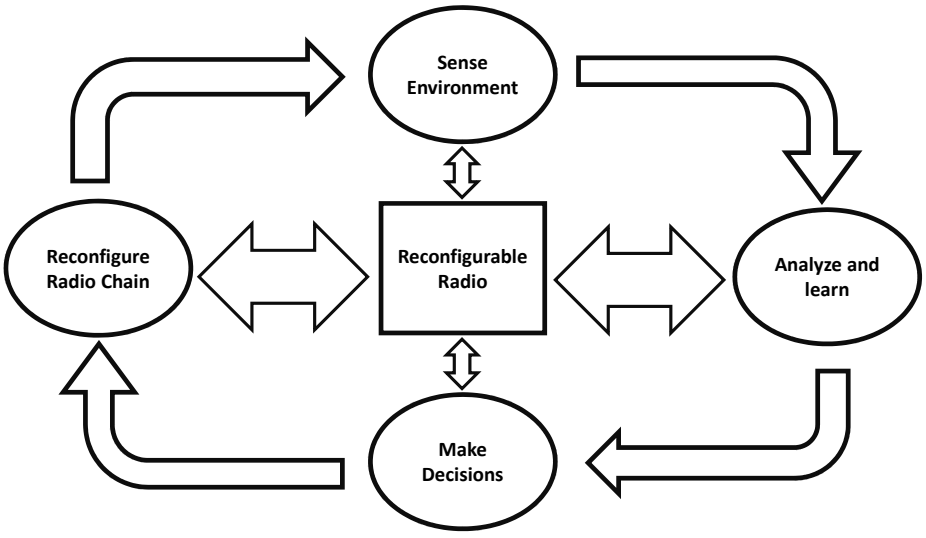


Figure 3.10 – Reconfigurable radio block diagram.

In this section, we have defined and mentioned some of the adaptive techniques used to improve the wireless communication and make an efficient decision to reconfigure the wireless chain. Some of these techniques have been studied and modeled in Chapter 2. In the Mohammad Rihani's thesis, we have focused on proposing new algorithms destined to select the best available wireless standard according to the channel conditions. This study deals with the Vertical Handover (VHA) concept, which is explained more deeply in the next section.

TABLE 3.2 – WiFi and WiMax networks comparison.

Characteristics	WiMax	WiFi
Range	up to 40 Kms	up to 100m
Scalability	user scale 1 to 100	user scale 1 to 10
Bit Rate	up to 100 Mbps in 20 MHz channel	up to 54 Mbps in 20 MHz channel
QoS	several levels	no guarantee of QoS
Mobility	support mobility	fixed

5.2 Case study : VHA for WI-FI and WiMax heterogeneous networks

A wireless network composed of multiple wireless standards is called an heterogeneous network. In this network, radios have the main purpose to sense the wireless spectrum continuously and detect the available wireless standards. Some algorithms may evaluate different parameters to learn, study, and then decide whether to perform a vertical handover or keep on communicating through the current standard. The key of taking a good decision is driven by the parameters collected from the environment. These parameters can be divided into four categories :

Wireless Channel Parameters These parameters are measured from the wireless channel and provide information about the channel status. RSS, SNR, and BER are examples of such parameters.

Network Information Parameters These parameters inform the VHA about the network status. This category includes the number of connected nodes to the access point, data rate, delay, jitter and cost.

User Parameters These parameters implement user's data requirements at the application layer such as the required data rate and high priority data.

System Parameters Such parameters provide the algorithm with the device status. Examples of these parameters are power consumption, battery level, and device speed.

In our work, and particularly in Mohammad Rihani's thesis, we have studied how VHA could be implemented in our reconfigurable platform. As a case-study, we have considered two heterogeneous networks i.e the WiFi and WiMax standards. In these standards, each Radio Access Technology features different specifications in terms of supported Uplink and DownLink data rate and coverage range. Both WiFi and WiMax standards support the IP protocol that is widely used nowadays for voice, data, and video streaming. The main features of both standards are given in TABLE 3.2.

5.3 The Adaptive Scoring System

The VHA that we have proposed for the WiMax-WiFi network is composed of three stages : an Handover Trigger, an Initial Decision and a Final Decision. In a first version of the algorithm, the mechanisms that have been implemented were quite simple.

As an example, let us assume that the system operates in WiMax by default. The Handover Trigger is initiated as soon as the handover process detects that the WiFi RSS is greater than that of WiMax. The trigger is based on the RSS because it is considered as the simplest and fastest way to detect the availability of wireless standards. The Initial Decision state is used to make fast decisions and detect fake handover. Fake handover is detected if the speed of the device is greater than a certain limit defined by WIFI_*max_speed_limit* (similar to that defined in [JH09b]). In fact, if VHA was based on RSS only, the system could have directly switched to WiFi : when a node is moving at high speed, it may leave the WiFi Access Point (AP) area and the system may switch back to WiMax almost instantly. In this case, two handovers may occur without any benefit but a waste of time and power. Our system is able to detect this situation and avoid this type of fake handover.

If the Initial Decision is successfully done, the system goes into the a Final Decision state. In this final state, an adaptive scoring system evaluates multiple parameters to decide which network is the best

TABLE 3.3 – Example of scoring system.

Parameters	WiFi	WiMax
Data Rate (3)	3	-
Delay(4)	-	4
Jitter(3)	-	3
SNR(3)	-	3
BER(3)	-	3
Power Consumption(5)	5	-
Total Score	8	13

according to user preferences. If the scoring system decides that WiFi is better than WiMax, then vertical handover occurs.

The general concept of the scoring system is based on assigning points to each selected parameter in the VHA decision. Points that are assigned to parameters are different in each standard. They are added within one standard to obtain a final score. The wireless standard with the highest score wins and the VHA decides to switch to it. An example dealing with the operation of the proposed scoring system is shown in TABLE 3.3, where each parameter is assigned different points.

Our proposed scoring system operates as follows : if a standard provides a better performance regarding a given parameter (ex. SNR, BER), then the points related to this concerned parameter are added to the score of the corresponding network. For example, if the Data Rate parameter is assigned 3 points, and WiFi provides better throughput than WiMax, then 3 points are added to the WiFi score. On the other hand, if the delay value in WiMax is less than that in WiFi, 4 points are added to WiMax. Finally, the network with the highest score is selected.

Adding adaptivity to the scoring system improves the efficiency of the system. For example, we consider the user preferences parameter to be the main parameter to take into account. In this case, it is possible to adapt the weight of each parameter in the scoring system according to the type of application running in the user space. As shown in TABLE 3.4, the points attributed to a given parameter may change according to the type of the running application.

TABLE 3.4 – Scoring system example according to different applications.

Parameter	VOIP	Video	Browsing	Ideal
Data Rate	3	4	2	1
Delay	4	4	2	1
Jitter	4	4	1	1

In our example, applications can be divided into four categories. The first category includes the video applications that demand high data rate and low delay and jitter. The second category is the VoIP applications. This type of applications requires less data rate than video applications but similar delay and jitter specifications. The third category concerns browsing applications which do not require specific real-time services. The ideal case, is when the node is only connected to the wireless network but without running any application [CF04].

Although very flexible, this scoring system remains based on static weights, which is an important limitation. In order to improve our system, we have proposed a more efficient approach based on machine learning techniques. This work has also been led in Mohammad Rihani's thesis. In section 5.4, a VHA based on neural networks combined with reinforcement learning is proposed and analyzed.

5.4 Towards a Smart Reconfiguration Management

5.4.1 Overview

In this part, we describe the works that have been led to propose a system that combines both supervised and unsupervised machine learning algorithms to switch between different standards. This system is implemented in the End-Device. It is based on an improvement of the previously proposed adaptive scoring system. The main idea is to benefit from this previous work in a first step to take a first decision. During this step, records and rewards are collected to obtain information about the environment. The first decisions that are taken are then classified based on the obtained rewards.

In a second step, only decisions leading to good rewards are considered and constitute a data set that is used to train a neural network. This neural network is used to take a more efficient decision regarding VHA. At a certain time and after collecting a large data set, the neural network is considered as sufficiently trained to take a right decision. At this point, it may be adopted by the global system to take further decisions. The system can repeat the procedure of training the neural system after updating the data set in order to optimize its operation and improve its behavior. The proposed system consists of the modules shown in Figure 3.11. These modules are described in the subsequent sections.

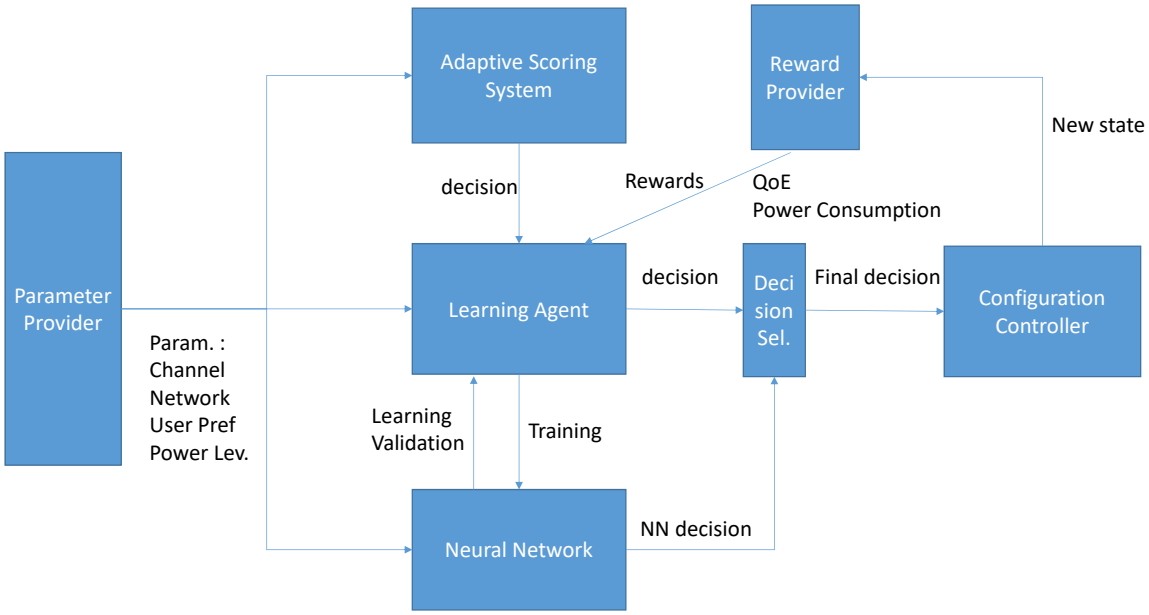


Figure 3.11 – Proposed VHA design.

5.4.2 Modules Description

▸ Parameters Provider

This module collects all parameters from different layers (physical, MAC, Network) to get information about the status of the channel, networks, user applications. It also provides information related to the node status i.e its location, power and speed. The parameters provider is considered as an interface for all the parameters of the system. The input of this module is composed of all parameters coming from different layers. Note that, typically, this module is implemented as a service of our micro-kernel as described in Figure 3.5.

▸ The Neural Network

5. Application level

In this study, we consider basic Multi-Layer Perceptrons (MLP) feed forward networks. These types of networks allow the modelling of complex behaviours and allow performing approximation of multi-dimensional functions. Such networks are composed of one or more hidden layers that consist of neurons with non linear transfer function. They also provide an output layer that implements output neurons with a linear or non linear activation function.

In a first learning phase, the network learns the relationship between inputs and targeted outputs that are available in a training set. This is generally performed by a back-propagation algorithm that optimizes the weight values of the connections between the neurons of different layers.

In a second phase (the forward phase), the network estimates the correct output for any given input pattern. In our models, three layers have been used. Each layer receives its inputs from the precedent layer and forwards its outputs to the subsequent one. In the forward phase, the weight matrix of the hidden layer is multiplied by the input vector $X = (x_1, x_2, x_3, \ldots, x_n)^T$, to compute the layer output, as expressed in equation 3.2.

$$y_{h,j} = f\left(\sum_{i=1}^{N_i} w_{h,ji} x_i - \theta_j\right) \tag{3.2}$$

where $w_{h,ji}$ is the weight connecting input i to unit j in layer h. θ_j is an offset term or bias that is also connected to each neuron j. f is the non-linear activation function. In order to train the network, the classic back-propagation algorithm has been used.

▸ Adaptive Scoring System

As previously explained , we have proposed a VHA based on adaptive scoring system for WiFi-WiMax networks. The proposed VHA is composed of an initial decision and a final one. The Initial Decision is a direct function based on the speed and the power level of the node. The Final Decision is made by the adaptive scoring system.

The trigger is initiated based on the RSS value of the available wireless networks. The Initial Decision first provides a fast decision and avoid fake handover triggers. If Initial Decision is passed, the Final Decision is based on the score of each network. The score of each network is the sum of the weights of the parameters. The weights of the parameters are adapted according to the power level and user preferences.

The role of the adaptive scoring system consists in taking decisions at a time when the neural network is not completely trained and not fully efficient. These decisions are saved as data records to progressively train the neural network. Rewards are evaluated after each decision, and data with good rewards are tagged whereas data with bad rewards are rejected. Only tagged data are used to train the neural network to identify the best solutions to perform.

▸ Learning Agent

This module has an important role in the system. Its goal is to train the neural network with the best records collected from the adaptive scoring system. The records are formed by parameters and decisions at time t and rewards at time $t+1$. The best records are tagged based on the received rewards. The learning agent also changes the decision made by the adaptive scoring system for similar parameters to get a different reward at time $t+1$. In this case, the learning agent will tag the record with the best reward to use it later in the training of the network. The role of the learning agent can be summarized by the following points :

— Collect and save records : parameters, control signals, decisions, and rewards.

— Invert the decision for a given record with the same parameters and control signals to get a new record.

— Select the best records to provide a new input for the NN training.

— Train the NN during runtime when new records are available.

— Validate and test the results from the NN to use them only when the NN provides efficient decisions.

▸ Reward Provider

The reward provider gets and provides the learning agent with the rewards after taking a decision to switch from a state to another. For example, if we consider a system including the QoS and power consumption as input parameters, the system can study the effect of any decision based on these parameters. The reward can be a numerical value for both power consumption and QoS. The QoS values reflect the quality of communication (throughput, delay, jitter) that must be achieved for the running user application. Power consumption is measured after each decision to switch from one standard to another. If power consumption diminishes, the reward is considered as a good reward. In the other case, the reward is negative.

▸ Decision Selector

This module receives the decision from the neural network and from the learning agent. When the network is validated by the learning agent, the decision agent forwards the decisions made by the network to the configuration controller and other modules are turned off. At this level, the system is considered to have learned making good decisions based on the experience. The NN module is validated when it is sufficiently trained by the minimum required number of tagged records. This is selected by experimental results.

▸ The Configuration Controller

The goal of the configuration controller consists in transferring partial bit-streams to the FPGA. For additional security reasons, the partial bit-streams memory locations are only accessible from the configuration controller. This controller eliminates any conflict that may occur when several user processes decide to access the FPGA fabric at the same time. In this case, it transfers the bit-streams consecutively. Note that, in our model, this module is implemented as a kernel service.

5.5 Results

In order to demonstrate the feasibility of our approach and obtain performances, the proposed system has been tested and all modules have been implemented. A simulation has been performed on a WiFi-WiMax scenario using a Java simulator developed during Mohammad Rihani's thesis. Based on the adaptive scoring system and a simple reward system, the simulator has been setup to generate the required data (Input Parameters – Decisions – Rewards – and Learning Decisions). The Java simulator randomly generates the input parameters : RSS, Network Load, User Preferences, Power Level and Status. Then, decisions and related rewards for power and QoS can be selected according to a simple decision and a reward engine. The engine operates based on the input parameters and previous decisions. The records with good and bad rewards are tagged with different values to be discriminated during the training process. Finally, generated data are mapped to numerical results and extracted to a file. This file may contain typical values :

- RSS : High 2, Med 1, Low 0
- Network Load : High 2, Med 1, Low 0
- User Preferences : Video 3, Voip 2, Browsing 1, Idle 0
- Power Level : High 2, Med 1, Low 0
- Status : Wimax 0, WiFi 1
- Decision : No Handover 0, Handover 1
- Reward Power : Decreased 2, Increased 1, Stable 0
- Reward QoS : Increased 2, Decreased 1, Stable 0
- Learn NN : Yes 1, No 0

As an example, the *0-1-2-2-2-0-1-1-2-2-1* sequence of bits represents : WiFi_RSS : low -Wimax_RSS : med -WiFi_Network Load : high -Wimax_Network Load : high -User Preferences : Voip -Power Level : low -Status : Wifi –Decision : handover -Reward_Power Consumption : decreased -Reward_QoS : increased -Learn_NN : Yes.

These generated data were used to train the neural network with 7 inputs only : RSS_WiFi, RSS_WiMax, Nl_WiFi, Nl_WiMax, user preferences, power_level, status). Only one output was used : (Handover/ No Handover) as a real value between 0 and 1.

In Table 3.5, we can notice that for records with similar parameters, the decision is inverted during the next cycle to collect new rewards and to identify which decision is the best. Then, the good records are tagged to be used later to train the neural network. This provides better performance compared to the adaptive scoring system. In this example, only the first and last records are included in the training data set.

As discussed before, the learning agent has the role to re-train the neural network every time enough tagged records are available. It is then important to determine the best number of tagged records that are necessary to provide good results. Table 3.6 provides the Mean Standard Error (MSE) for different number of records used to train the NN system. As indicated, the more the number of used samples is considered, the better the obtained MSE becomes. Hence, selecting large number of samples will give better results but more time is required to collect these samples. A simple threshold-based decision can then be applied according to the level of performance to be reached.

As an example, the proposed system was tested in terms of power consumption and QoS. Figure 3.12 presents the rate of successful decisions with respect to the number of handovers using tagged records only.

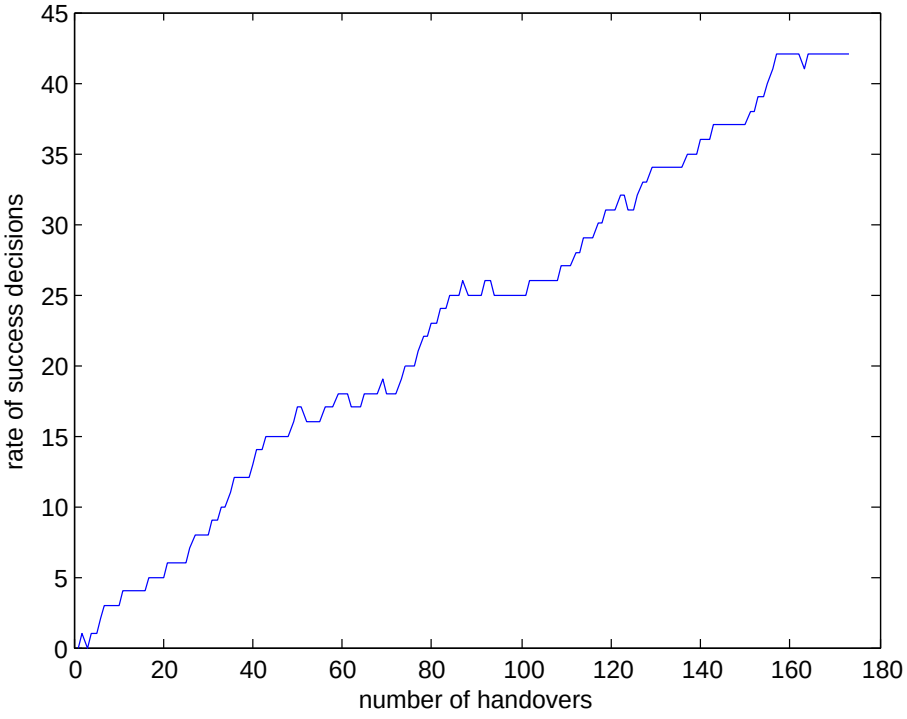


Figure 3.12 – Rate of successful decisions using tagged records (power consumption).

6 Performances Evaluation

In this part, the work related to the modelling of our global system (see Figure 3.5) is presented. This study has been performed in Tian Xia's thesis in order to qualitatively evaluate the overhead of the different elements that are involved in the reconfiguration process, from the application layer down to the hardware layer. The model that has been envisaged consists of the Ker-ONE micro-kernel that hosts several guest OS (at least 2), each of them, running several tasks. All guest OS have a different priority and may access a reconfigurable part of an FPGA device.

TABLE 3.5 – Example showing how the learning agent selects the best records.

WiFi-RSS	Wimax-RSS	Wifi-NL	Wimax-NL	User pref	PWL	State/ Decision	Rew-PWC	Rew-QoS	Learn NN
High	Med	High	low	Voip	med	WIFI/ Hand.	Increased	Increased	Yes
High	Med	High	low	Voip	med	Wifi/ No-Hand.	———–	————	No
High	low	High	High	Browsing	low	Wimax/ No-Hand.	————	————	No
High	low	High	High	Browsing	low	Wimax /Hand.	Increased	————	Yes

NL : Network Load ; UP : User Preference ; PWL : Power Level ; PWC : Power Consumption ; Rew : Reward.

TABLE 3.6 – MSE on training and validation sets for different numbers of samples.

Number of Samples	MSE Training	MSE Validation
175	3.9 e-2	3.8 e-2
688	2.7 e-2	2.9 e-2
5500	2.04 e-2	2.03 e-2

6.1 Overhead Analysis

Our evaluation first focused on the allocation latency, i.e. the delay that occurs before the accelerator is properly allocated and ready to start. This latency may be seen as the response time of an HW accelerator, which has a significant impact on OS timing. The additional latency that is due to allocation have two main sources : the allocation mechanism itself and the Ker-ONE micro-kernel. Additional overhead is to be deplored if the allocated accelerator requires reconfiguration. Besides, the virtualization mechanism takes up extra time. For example, the page-table faults handling, IPCs and VM scheduling noticeably contribute to the total allocation latency.

Based on the allocation mechanism that we have proposed, the execution path is determined by the different solutions defined in section 3.3, which leads to 4 execution paths :

— Path 1 (i.e. Solution {assign}) : Allocates an Idle accelerator to a virtual machine without reconfiguration, which is also called immediate allocation, since the virtual machine can start using the virtual device immediately.

— Path 2 (i.e. Solution {assign with reconfig.}) : Reconfigures an Idle partial reconfiguration region (PRR) with the desired accelerator and allocates it to a virtual machine.

— Path 3 (i.e. Solution {preempt}) : Preempts a running accelerator and allocates it to a virtual machine without reconfiguration.

— Path 4 (i.e. Solution {preempt with reconfig.}) : Preempts a running accelerator, which will then be reconfigured with the desired accelerator and gets allocated to a virtual machine.

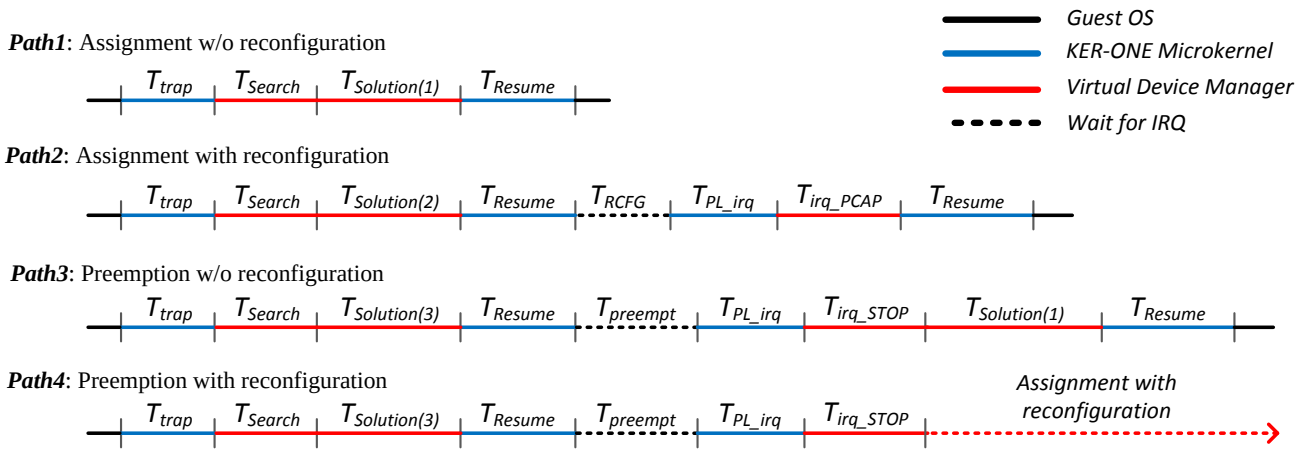


FIGURE 3.13 – Execution paths of DPR resource allocation.

The models of execution paths for different solutions have been determined and displayed in Figure 3.13. In these models, the allocations consist of four different solution paths that can be decomposed into smaller atomic execution overheads :

— T_{trap} : Time required by Ker-ONE to detect a page-table exception in VM domain and to invoke the Virtual Device Manager.

— T_{resume} : Time required by Ker-ONE to schedule back to a VM.

— T_{PL_irq} : Time required by Ker-ONE to receive IRQs from the PRR Monitor and to redirect them to the Virtual Device Manager.

— T_{Search} : Time required by the Virtual Device Manager to receive the VM requests and to search for solutions.

— $T_{Solution(1)(2)(3)}$: Execution time to handle different solutions : (1) direct assignment, (2) assignment with reconfiguration, (3) preemption.

— T_{irq_pcap}, T_{irq_stop} : Time required by the Virtual Device Manager to handle the following IRQs (i.e. IRQ_PCAP_Over, IRQ_PRR_Stop).

— $T_{preempt}$: Overhead of waiting for a preemption.

Therefore, the total allocation latency for the different allocation execution models can be calculated as :

$$\begin{aligned} T_{Path1} &= T_{trap} + T_{Search} + T_{Solution(1)} + T_{resume}, \\ T_{Path2} &= T_{trap} + T_{Search} + T_{Solution(2)} + 2 * T_{resume} \\ &\quad + T_{PL\_irq} + T_{irq\_pcap} + T_{RFCG}, \\ T_{Path3} &= T_{trap} + T_{Search} + T_{Solution(3)} + 2 * T_{resume} \\ &\quad + T_{PL\_irq} + T_{irq\_stop} + T_{Solution(1)} + T_{preempt}, \\ T_{Path4} &= T_{Path3} + T_{Path2} - T_{trap} - T_{Search} - T_{resume} - T_{Solution(1)} \end{aligned} \tag{3.3}$$

After having qualitatively determined the different steps involved in the reconfiguration of a system, it has also seemed necessary to evaluate the impact of each of these steps. This is the object of the next section, in which a real case study has been considered.

6.2 Experiments and Results

6.2.1 Description

Our experiments were performed on a Xilinx ZedBoard, which provides a dual-core ARM Cortex-A9 processor and a partially reconfigurable FPGA fabric. The CPU operating frequency has been set to 667 MHz and the FPGA logic was driven by a 100 MHz clock. The architecture of the experiment is shown in Figure 3.14.

The FPGA PL part has been initially implemented with three PRRs of different sizes. Four hardware accelerators, i.e. QAM16, QAM64, FFT512, FFT1024, have been implemented and stored into bitstream files. During the initialization stage of Ker-ONE, these files have been loaded into the RAM and are only accessible from the Virtual Device Manager.

This experiment is based on the example of an OFDM receiver that is intended to be very flexible by considering several configurations of modulators and mappers according to the channel conditions. QAM blocks aim to take a complete frame of incoming bits into account and generate 16-bits width I and Q symbols. FFT blocks work on the QAM I and Q symbols to perform demodulation. The data frame is set to be 18,800 bits, according to actual OFDM requirements.

Regarding the guest OSs running in virtual machines, we have modified the µC/OS-II RTOS to execute on top of Ker-ONE. In our experiment, two µC/OS-II guests are hosted with different priority levels. For each guest OS, four available virtual devices have been implemented. Two and three tasks run respectively in both guest OSs to periodically command virtual devices to process data frames containing 18,800 bits, which causes requests for allocations during the experiment. Accelerators are then allocated at run-time.

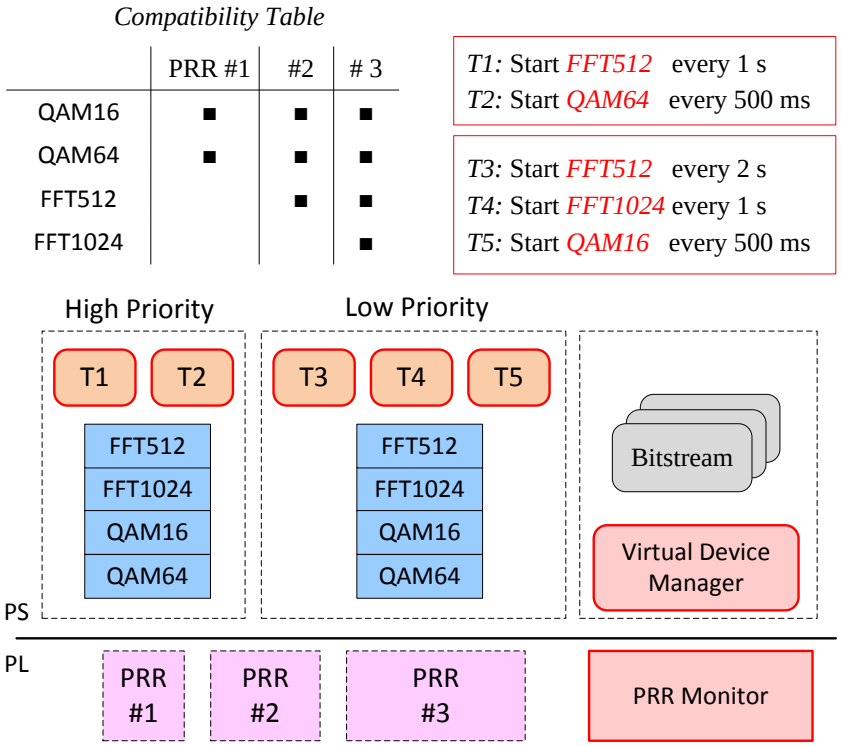


FIGURE 3.14 – Experimental architecture for performance evaluation.

TABLE 3.7 – Overhead measurement during DPR allocation .

Micro-kernel		Virtual Device Manager	
Operation	Overhead (μs)	Operation	Overhead (μs)
T_{trap}	0,76	T_{Search}	0.50
T_{resume}	0,64	$T_{Solution(1)}$	1.13
T_{PL_irq}	0.81	$T_{Solution(2)}$	2.77
		$T_{Solution(3)}$	0.34
		T_{irq_pcap}	0.64
		T_{irq_stop}	0.28

6.2.2 Results

According to the model that has been described in section 6, the obtained measurement results are given in Table 3.7. One may notice that Ker-ONE guarantees high performance in virtualization. Virtual machine scheduling as well as virtual interrupt emulation are performed with a low overhead that is less than 1 μs.

According to the performed measurements, the different allocation overhead values may be modeled as :

$$\begin{aligned} T_{Path1} &= 3.03\mu s, \\ T_{Path2} &= 6.76\mu s + T_{RFCG}, \\ T_{Path3} &= 5.10\mu s + T_{preempt}, \\ T_{Path4} &= 9.96\mu s + T_{preempt} + T_{RFCG}. \end{aligned} \tag{3.4}$$

Latencies that are provided in Equation 3.4 are related to the OS with the highest priority. We may notice that a 3 μs latency is obtained for a direct allocation. Other solutions have additional latencies due to preemption or reconfiguration time.

TABLE 3.8 – Reconfiguration and preemption delays .

Virtual Device	$T_{preempt}(\mu s)$ (WCET)	$T_{RCFG}(\mu s)$		
		PRR#1	PRR#2	PRR#3
QAM16	47.0	231	810	1,206
QAM64	31.0	231	810	1,206
FFT512	24.1	-	810	1,206
FFT1024	33.6	-	-	1,206

TABLE 3.9 – Comparisons between SW and HW implementation.

Algorithm	$T_{HW}(\mu s)$ (per frame)	$T_{SW}(\mu s)$ (per frame)	FPGA Resource Usage
QAM-16	47.0	1,513	2%
QAM-64	31.0	1,174	2%
FFT-512	71.1	6,582	8%
FFT-1024	90.6	12,784	13%

The costs of preemption ($T_{preempt}$) and reconfiguration (T_{RCFG}) are mostly depending on the implementation and application of accelerators. In TABLE 3.8, the value of these costs are provided. T_{RCFG} is determined by the size of the bitstream, and therefore corresponds to three PRR areas. The worst-case preemption time $T_{preempt}$ is determined by the computation granularity. In our case, in order to respect the integrity of the OFDM process, QAM and FFT modules are set to be preemptive only when their determined data frame is completely processed. Note that, since the implementation of the PRRs and accelerators are fixed beforehand, these costs can then be predicted and considered for guest OS tasks schedulability. In TABLE 3.8, for the accelerators used in our experiment, $T_{preempt}$ is significantly lower than T_{RCFG}. For a system in which preemptions occur frequently, it is possible that a low priority virtual machine may never get access to hardware resources.

Choosing between preemption and consistency is not a simple task. In our work, we have made the assumption that allocating Idle PRRs is always better than preempting them. The reason is that we wanted to make sure that low priority task can always be executed. The PRR Monitor has been designed accordingly. In a system that manages critical tasks, a new policy may be followed that gives more importance to preemption. In this case, high priority tasks could always be executed first.

In TABLE 3.9, HW acceleration approach is compared with software. The results show that the accelerator performance of heavy computation (i.e. FFT512/1024) significantly surpasses software implementation. Even though these accelerators suffer from allocation latency that may increase the execution time, their benefit is still considerable. On the other hand, for relatively light computation(e.g QAM), although hardware accelerators are still faster, this advantage gets undermined when taking T_{RCFG} into account. These results also indicate that DPR technology is more suitable for large complex computation algorithms. Furthermore, in this example, the FPGA only implements 3 PRR areas, taking around 23% of the available resources. Compared to static circuits with accelerators for both VMs, which may take up to 50% resources, the usage of FPGA is greatly reduced.

7 SUMMARY

In this chapter, the various works that we have performed on reconfiguration management have been presented. Three levels of management have been proposed, i.e Hardware Level, OS Level and Application Level.

At the Hardware Level, our contribution has consisted in proposing a structure of HW task as well as dedicated hardware and software mechanisms. The mechanisms aims at handling hardware resources and at managing the execution of hardware accelerators in a reconfigurable context. Hardware task preemp-

tion solutions have also been proposed to be compatible with real-time constraints that may be imposed by applications. Finally, memory isolation strategies have also been described to ensure security in the accelerators' access.

At the OS level, we have described our custom micro-kernel that intends to be kept as simple as possible by taking advantage of modularity. Specific services have been proposed to handle reconfigurability and abstract hardware reconfiguration management for the user tasks.

At the application level, we have proposed several management mechanisms to adapt a full system to environmental changes. These mechanisms are based on a complex decision process that combines both machine learning and re-inforcement learning.

The complete system has then be modelled and implemented in a real platform in order to evaluate its feasibility and performance.

Summary (PhD/Dissemination/Projects)

— 3 PhD students : Yaset Oliva, Tian Xia, Mohammad el Fadl Rihani

— Related publications : 19

From Power Modeling to highly Energy-Efficient Devices

Chap. 4

1 Context and Related Works

This chapter describes another research axis that I have followed since 2007, when I first joined the IETR lab. This study deals with the energy efficiency of embedded systems and particularly focuses on their design. The application field which is described here concerns the wireless communicating devices since it constitutes the main domain of expertise of the SYSCOM team, in which all these works have been led. However, most of the works presented in this chapter may be easily adapted to other domains of applications without significant efforts.

Nowadays, energy efficiency has become a major concern all over the world and particularly for new 5G communicating devices [BCLK+16]. These devices aims at providing ubiquitous connectivity as well as innovative services and will be massively used in a near future. It is forecast that by 2020, there will be more than 50 billion connected devices, meaning that each person will own at least 6 connected devices [Dav18].

This energy efficiency concern is also shared by mobile networks operators, who have recently stated that future 5th Generation (5G) of mobile networks "should support a 1,000 times traffic increase in the next 10 years timeframe, with an energy consumption by the whole network of only half that typically consumed by today's networks. This leads to the requirement of an energy efficiency increase of x2000 in the next 10 years timeframe" [All15b]. In this context, it becomes mandatory to design these communicating devices by taking the new energy dimension into account and not only focus on classic metrics such as performance and throughput.

Most of these wireless systems are composed of two parts. The first deals with RF circuitry as well as power amplifiers and are used to transmit the information over the air. The second part concerns the base-band circuits in which all processing modules are implemented. These modules are generally based on digital circuits. From a study performed in [ear], it appears clearly that the energy consumed by the processing part (base-band) of a device has an increasing contribution as the device gets smaller. For example, in femto-cells, which have a very small size, the base-band power reaches 47% of the total consumed energy.

With the multiplicity of these communicating devices, it seems reasonable to consider that the energy required by the processing part of a device will be as much important or even more than the energy required by the amplifier. This assumption is especially true if we consider that a lot of devices will probably have small communication ranges with small power requirements and that the complexity of processing will continue to increase. This situation opens the way to new research studies dealing with power optimization of digital processing in such devices.

Classically, most of the circuits that are used in communicating devices are low-cost ASICs. Since several years, FPGAs have become an interesting alternative since they exhibit a high level of performance while being low-cost and low-power. Furthermore, due to their intrinsic technology, these circuits are much easier and faster to design than their ASICs counterparts. Today, as designers want to implement a full system into an FPGA device, they usually follow a generic top-down approach which is divided into several well-identified levels (see Figure 4.1).

At System Level, the system is modelled using dedicated tools such as Matlab, which is widely used in the community, or programming languages such as C/C++. The system behaviour is usually validated using simulation tools such as simulink. Then, FPGA design implementation is performed in three

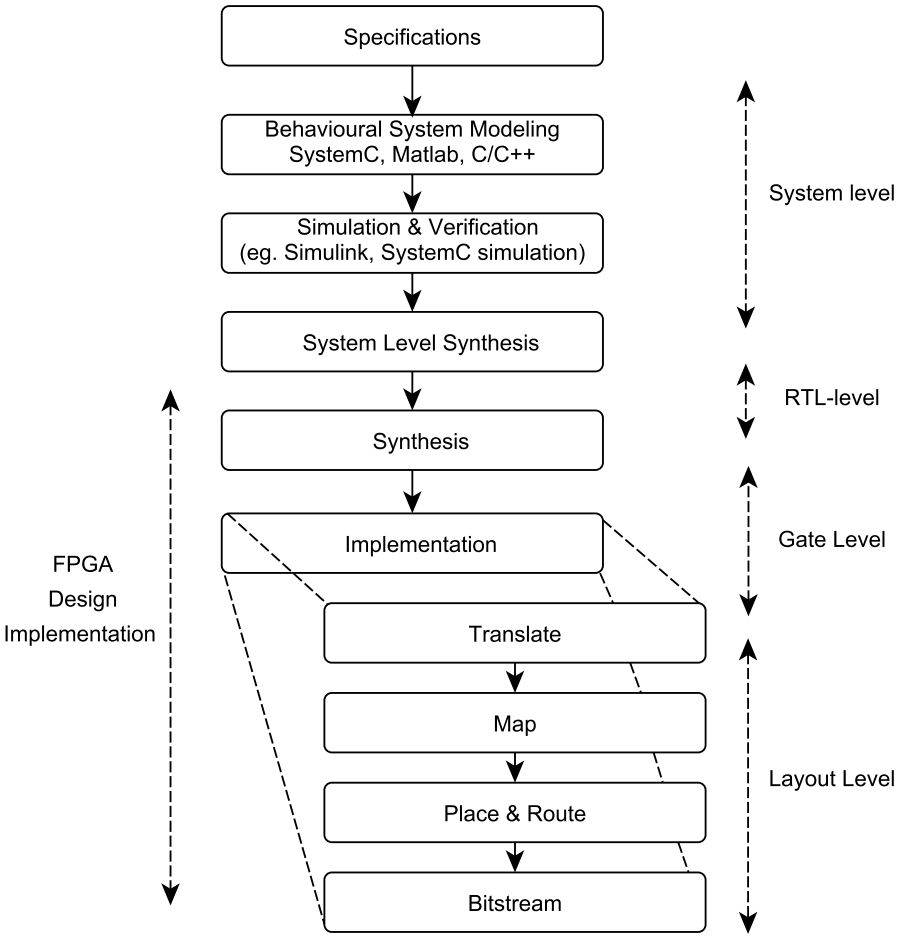


FIGURE 4.1 – Typical FPGA design flow and abstraction levels.

TABLE 4.1 – Main examples of system-level tools used in the wireless communication domain which do not support power estimation basically.

Name	Academic or Commercial (A/C)	Additional Description
Matlab - Simulink [Mat15]	C	Graphical system modelling and simulation tool
System Development Suite [Cad11]	C	Based on a set of 4 platforms, it enables system-level hardware/software development, virtual prototyping using transaction-level models, functional verification, system analysis and optimization before any implementation
SystemVue ESL software [KT15]	C	EDA environment that enables automatic simulation and test for wireless communication systems (Base-band, RF)
LabVIEW FPGA [NIC15]	C	Based on LabVIEW graphical development platform, it enables FPGA design simulation, verification and deployment
Ptolemy [Dep15]	A	Open-source software framework for modelling and simulating of actor-oriented based design
System Studio [Syn15]	C	Model-based signal processing algorithm design and analysis tool based on C data-flow modelling paradigm
Vista [Sha15]	C	Tool for architectural design exploration, verification, and virtual prototyping based on TLM 2.0

steps : first, System Level synthesis enables the translation of System Level architectures into a Hardware Description Language (HDL) model. Then, hardware synthesis takes this model as an input to generate a supported netlist compatible with FPGA vendors' implementation tools. Finally, a bitstream file is generated after these implementation steps. Note that these levels are common to all hardware design flows and are therefore adapted to wireless communication design which is the main application domain that is studied in this work.

In our studies, we mainly considered the System Level since it corresponds to the level of abstraction to which most of communication engineers are habituated. Furthermore, our motivation relied on the fact that design decisions that are taken at this level have the most impact on the final system performances.

▸ The first approach : the classic implementation flow

In an ideal case, designers generally start working at System Level and rely on software tools to automate most of the subsequent design steps. The main idea is to rapidly get an evaluation of their algorithms on a real hardware prototype without requiring costly design stages. Using this approach, designers may explore the design space very rapidly by simply evaluating specific metrics that may be collected on a real hardware target. This make it possible to compare several solutions and retain the most performing ones for a given platform.

Today, at System Level, a lot of software tools exist that easily connect to hardware boards. Matlab from MathWorks [Mat15] is one of the famous tools used by engineers. Other examples of tools and methodologies are listed in TABLE4.1. Basically, such tools do not support power estimation and designers have to integrate additional information in order to take power estimation into account. As an example, Matlab allows users to develop and to test their algorithms in a user friendly environment and with a common language. However, it requires additional tools to complete the design. For FPGA devices, System Generator [Xil12a] and DSP builder [Alt15b], respectively from Xilinx and Altera, are tools that have been integrated in the Matlab environment in order to enable HDL code generation, directly from a Simulink graphical description. The generated HDL code may be translated into a netlist or even a bitstream, according to the FPGA classic design flow. At this point, power consumption can be estimated by low-level simulations or by real on-board measurements. However, this approach can be really time consuming for large designs and design space exploration becomes limited due to the prohibitive number of required implementations. The flow has to be re-run from scratch, for each system configuration.

▸ Towards High-Level Modelling and Faster Simulation

To circumvent the issues that are due to hardware implementation, another solution consists in elaborating generic hardware models and perform fast simulations at System-Level without considering lower

design levels. This approach generally allows designers to rapidly get an idea about the system performance to the detriment of the accuracy.

Regarding energy estimation, at System Level, design functionality is modelled and theoretical performance can be easily evaluated. In general, as designers desire to get a first estimation of the power consumption of their systems, they often make use of general power models or even spreadsheets. In [DDG+12, ARFB10, DDVJM12], analytic power models have been proposed for every sub-component of a wireless system, which can include base-band processing, Radio Frequency (RF) stages, power amplifiers, microprocessors, etc. In all these works, power models are derived from data-sheet values, real measurements or are possibly based on other works presented in literature. The accuracy of power estimation is not the primary objective but rather used to draw general guidelines.

Power information can also be integrated in power models. In [HLF+11], an approach that is based on the development of power models in a SoC is presented. For IP cores, power models are built from a transistor-level simulation and linear regressions. Moreover, key signals are identified to determine the different states of the cores. An Instruction level technique is used for processor modelling. After the modelling step, a tool called PowerDepot, which is a set of C++ classes implemented in SystemC, realizes the transformation of the power models into a power monitor block used to estimate power during the SystemC simulation. Finally, the authors show that their methodology can achieve less than 2% error for power estimation. This is mainly due to the fairly accurate power models made at layout level. Power models can also be developed for IP components and processors using different modelling techniques such as FLPA or ILPA [EJH06, CA07, RBAN+11, SRH+11]. In [JC12], FPGA-based power models for DSP-oriented designs have been presented. In all these works, the application behaviour is not considered, which can lead to important error when estimating power consumption. Such a problem is discussed in [DS07] in which macro-modelling is combined with a SystemC-TLM modelling approach. Despite a simulation speed-up, it is mainly used when small designs are considered.

▸ Towards 5G New Communication Schemes

As previously mentioned, the targeted area of our studies is the new communications schemes that may be used in 5G or in close domains such as IoT. In these domains, a lot of studies has been led that identify efficient algorithms that may be used in the physical layer [GBB+17]. Among all these waveforms, we have first focused on Time Reversal (TR) that has been studied in the TRIMARAN ANR project. After the success of the Time Reversal techniques that were applied to acoustic communications, the Time Reversal concept has progressively been envisaged in the telecommunication domain [BPZ02]. It has been first studied in a (Single Input Single Output (SISO) context [DCH10], then in MISO [HYW+12] and finally on massive MIMO [PML15].

Regarding the benefits that are offered by this communication scheme when it is coupled with other types of modulations, it has recently being proposed as a good candidate for future 5G networks [CWH+16] and for the Internet of Things [CHY+14a] as well. The interest of TR relies in the fact that the propagation channel acts as an adapted filter on the transmitted signal. The received energy is then focused in time and space, which drastically reduces Inter Symbols Interference (ISI). This is particularly interesting since it permits to reduce the symbol duration and then makes it possible to increase the throughput. It also permits to decrease the transmitting power since the signal directly focuses on the receiver position. Finally, it allows designers to conceive very simple receivers, with few resources, that do not consume a lot of energy.

2 The Classic Implementation Approach

In this axis, a significant part of my research has consisted in proposing solutions to implement energy efficient algorithms while maintaining a high-level of performance. In this section, a focus is made on Time-Reversal based algorithms. The results that are presented here come from the TRIMARAN ANR project, in which I was responsible for the Work-Package dedicated to the implementation of the proof of concept (PoC).

The first purpose of the TRIMARAN project was to identify the mechanisms that make it possible to implement Time Reversal in the wireless communications domain, and more particularly in the IoT

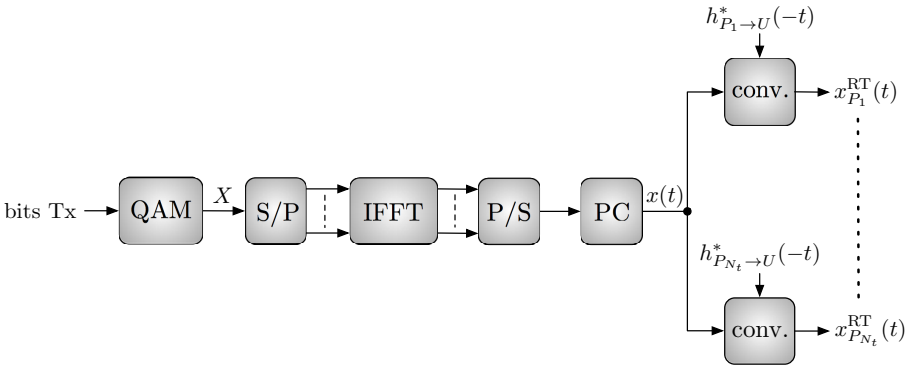


FIGURE 4.2 – Architecture of a MISO RT-OFDM transmitter.

domain. The second objective of the project was to evaluate the performance of Time-Reversal algorithms and compare them with classic wireless techniques. In order to ease the design of the prototype, we made extensive use of existing High-Level tools to explore the design space very easily and compare several algorithmic solutions.

One major issue that we have faced in our projects is that, in some cases, there is a huge difference between theoretical studies and practical implementation. In these cases, following all design steps of the flow until a real implementation on the hardware platform is mandatory. This makes it possible to take into account the hardware specificity.

The TRIMARAN ANT project gave rise to the supervision of the PhD Thesis of Yvan Kokar, who participated in the work-package. The main objective of the thesis was to evaluate the hardware feasibility of the Time-Reversal algorithms for wireless communications. It mainly focused on studying the hardware specific features that are generally unknown in the first design stages.

2.1 STUDYING NEW WAVEFORMS

The first algorithms that are described in this part deals with Time-Reversal coupled with classic OFDM techniques in a multi antenna scheme. The implementation scheme of a MISO RT-OFDM system with N_t antennas on the access point is depicted in FIGURE 4.2.

One may notice that this scheme has a strong resemblance with an OFDM-based design. The first part of the RT-OFDM modulator corresponding exactly to an OFDM modulator, the difference with the latter comes from a TR filter that is placed before each antenna of the access point. The role of these N_t TR filters is to perform, for each of the N_t antennas of the access point, a linear convolution between the OFDM signal $x(t)$ and the impulse response of the conjugated channel, returned in time $h^*_{P_i \to U}(-t)$, where $h^*_{P_i \to U}(-t)$ corresponds to the channel impulse response between the i^{th} antenna of the access point and the user antenna.

The transmitted signal from the antenna i of the access point is then given by

$$x^{TR}_{P_i}(t) = x(t) \otimes h^*_{P_i \to U}(-t) \tag{4.1}$$

The signal received by the user during the simultaneous transmission by the antennas, in the presence of a Gaussian Additive White Noise $b(t)$ is then given by :

$$
\begin{aligned}
y(t) &= \sum_{i=1}^{N_t} x_{P_i}^{TR}(t) \otimes h_{P_i \to U}(t) + b(t) \\
&= x(t) \otimes \sum_{i=1}^{N_t} h_{P_i \to U}^{*}(-t) \otimes h_{P_i \to U}(t) + b(t) \\
&= x(t) \otimes \sum_{i=1}^{N_t} \Gamma_{h_i}(t) + b(t) \\
&= x(t) \otimes h_{eq}(t) + b(t)
\end{aligned}
\tag{4.2}
$$

where $\Gamma_{h_i}(t)$ is the auto-correlation function of the impulse response $h_{P_i \to U}(t)$. Equation 4.2 shows that the signal $x(t)$ is transmitted through an equivalent channel $h_{eq}(t)$ that corresponds to the sum of the N_t auto-correlation functions $\Gamma_{h_i}(t)$.

For OFDM systems, it has been shown that TR can be applied in either time and frequency domain and exhibit the same level of performance. In the frequency domain and in a MISO OFDM scheme, the received symbol can be expressed as expressed in [PKHC14] :

$$
R_{m,n} = \frac{1}{\sqrt{N_t}} \sum_{k=1}^{N_t} \left| H_{m,k} \right|^2 D_{m,n} + N_{m,n} \tag{4.3}
$$

where N_t is the number of transmit antennas, $H_{m,k}$ is the complex channel coefficient on the m_{th} subcarrier of the ODFM symbol for the k_{th} transmit antenna. $D_{(m,n)}$ is the data symbol and $N_{(m,n)}$ is the noise term associated to the m_{th} subcarrier of the n_{th} OFDM symbol.

2.2 Proposed Offline Hardware Platform

2.2.1 System Description

In order to demonstrate the feasibility of Time Reversal techniques, we first proposed on offline hardware platform. The prototype consists of one transmitter (T_x) and one receiver (R_x) implemented on separate WARP FPGA-based motherboards [Con]. Figure 4.3 depicts the overall architecture of the prototype including the interfaces and cables between the boards. The transmitter is composed of 3 RF modules connected to 3 transmitting antennas. On the receiver side, 1 target antenna and 1 spy antenna are respectively connected to RF boards.An additional clock board is also connected to each mother board. This board provides the various clock signals used throughout the system.

Both transmitter (T_x) and receiver boards (R_x) are connected to a dedicated cable in order to ensure a perfect frame synchronization between them and to clearly identify in which time slot each board has to send/receive data. A RF synchronization is also required to precisely have the same clock frequency for all RF boards. In the proposed system, one clock source is generated on the T_x board and is directly transmitted to the R_x board via a dedicated cable.

Finally, a global sampling clock is provided by an external clock generator and connected to both T_x and R_x boards in order to drive all DAC (Digital to Analog Converters) with an identical input signal (in frequency and phase). This clock source has a 40 MHz frequency. Note that all cables that connect the external generator to the DACs must have exactly the same length in order to make sure that all DACs share the same clock phase.

The proposed architecture has been designed to be as flexible as possible. Table 4.2 indicates the parameters that cannot be modified in the hardware platform whereas Table 4.3 summarizes the parameters that may be modified as well as their default values.

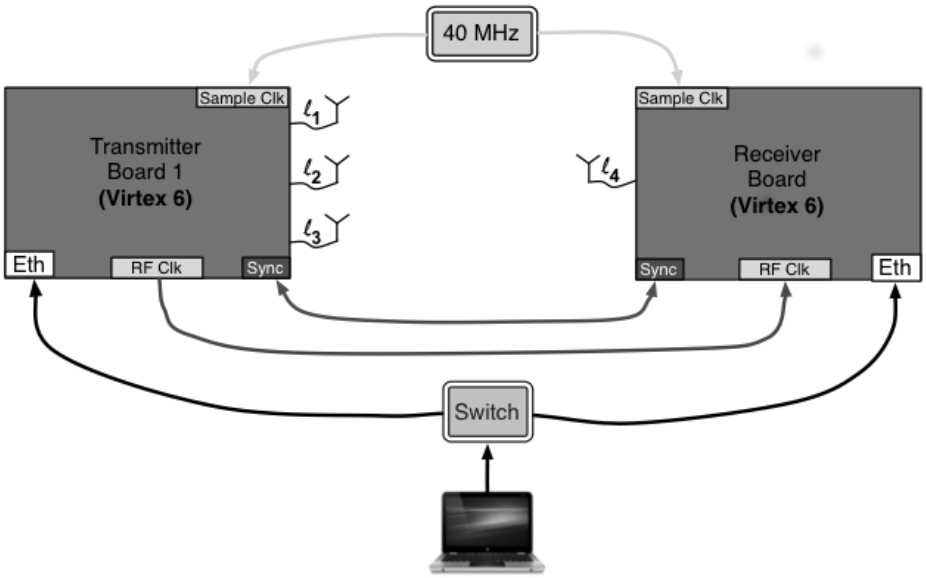


FIGURE 4.3 – Prototype description.

TABLE 4.2 – Fixed architectural parameters.

Carrier Frequency	2.4 GHz
Sampling Frequency	40 MHz

2.2.2 Studied configurations

▸ Classic OFDM

The first studied configuration is a simple OFDM chain without data pre-coding at the transmitter side. Several parameters may be modified such as the FFT size, number of OFDM symbols, number of pilot symbols, Cyclic Prefix size, etc.

▸ Time Division Duplex Time Reversal (TDD TR)

This second configuration stands for the simple Time Reversal strategy in which data are pre-coded with the following factor :

$$V_{m,k} = \frac{H^*_{m,k}}{\sqrt{N_t}} \tag{4.4}$$

where $H^*_{m,k}$ is the conjugate of the channel coefficient on the m-th subcarrier for the k-th transmit antenna and N_t is the number of transmitting antennas. Considering perfect synchronization and a well-dimensioned guard interval, the received symbol is then :

TABLE 4.3 – Variable architectural parameters.

Number of transmitted bits	19012
Modulation	QPSK and BPSK for pilots
CP size	1 to 256 (64 by default)
Number of pilot symbols in an uplink frame	1,2,4,8 (8 by default)
Number of pilot symbols in a downlink frame	1,2,4 or 8 (8 by default)
Number of OFDM symbols in a downlink frame	1 to 65535 (1000 by default)
FFT/IFFT size	256
Number of active sub-carriers	1 to 256 (190 by default)

$$R_{m,n} = \sum_{k=1}^{N_t} H_{m,k} \frac{H^*_{m,k}}{\sqrt{N_t}} D_{m,n} + N_{m,n} \tag{4.5}$$

$$= \frac{1}{\sqrt{N_t}} \sum_{k=1}^{N_t} \left|H_{m,k}\right|^2 D_{m,n} + N_{m,n} \tag{4.6}$$

▸ Equal Gain Transmission (EGT)

EGT aims at maximizing the SNR at the receiver side. This technique consists in modifying the phase of the complex symbols before their transmission on every subcarrier of the system. In the MISO-OFDM context, the pre-coding vector is :

$$V_{m,k} = \frac{e^{-j\phi H_{m,k}}}{\sqrt{N_t}} \tag{4.7}$$

where $V_{m,k}$ represents the pre-coding function associated to sub-carrier m, and $H_{m,k}$ the channel coefficient on the m_{th} sub-carrier for the k_{th} transmit antenna, ϕ is the argument of the channel coefficient and N_t is the number of transmitting antennas. In this configuration, the symbol received on each sub-carrier is then expressed as :

$$R_{m,n} = \sum_{k=1}^{N_t} H_{m,k} \frac{e^{-j\phi H_{m,k}}}{\sqrt{N_t}} D_{m,n} + N_{m,n} \tag{4.8}$$

$$= \frac{1}{\sqrt{N_t}} \sum_{k=1}^{N_t} \left|H_{m,k}\right| D_{m,n} + N_{m,n} \tag{4.9}$$

2.2.3 Results

In a simple OFDM MISO 3x1 configuration (see Figure 4.4), it may be noted a phase difference among all subcarriers that makes it difficult to retrieve initial data without effective equalization.

In the Time Reversal configuration, it may be seen that the results are better than the OFDM case. It can be observed that the resulting phase is close to zero which will make it possible to retrieve the initial signal without an equalization step (cf. Figure 4.5). We can also notice a magnitude degradation since the received signal varies according to the power of 2 of the channel coefficients' module.

In the EGT configuration, only the phase is modified which has no incidence on the magnitude of the signal. In this configuration, the phase is also close to zero which makes it possible to get rid of the equalization step (cf. Figure 4.6). Regarding the signal magnitude, we can see that the weakening is less noticeable in compliance with theory (magnitude variation in H instead of H^2)

The measures that have been performed on the developed prototype confirm the results that have been previously described in theory in [DHCG13]. The expression of the received signal for EGT differs from the received signal in the TR case by the power of the modulus of the channel coefficients. This may be observed on the downlink channel magnitude in Figure 4.5 and Figure 4.6 respectively. In both cases, the resulting phase is close to zero. Concerning the computation complexity which is of interest in terms of power consumption, it may be noticed that compared to TR, EGT requires an additional phase computation which can be easily computed. Both strategies exhibit the same level of complexity in a hardware point of view.

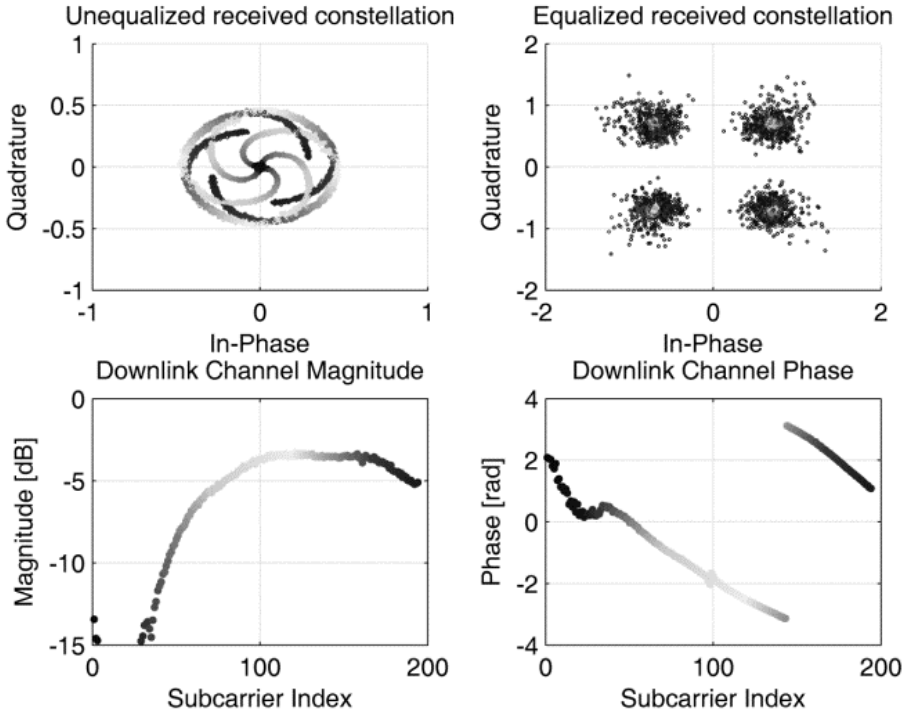


FIGURE 4.4 – MISO OFDM 3x1.

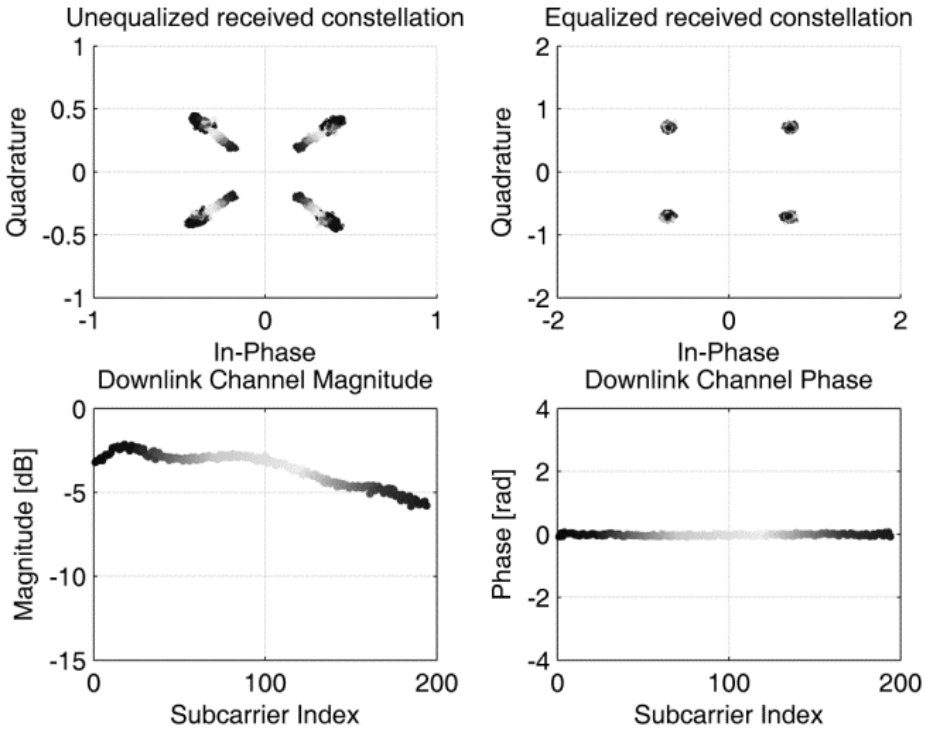


FIGURE 4.5 – TDD TR.

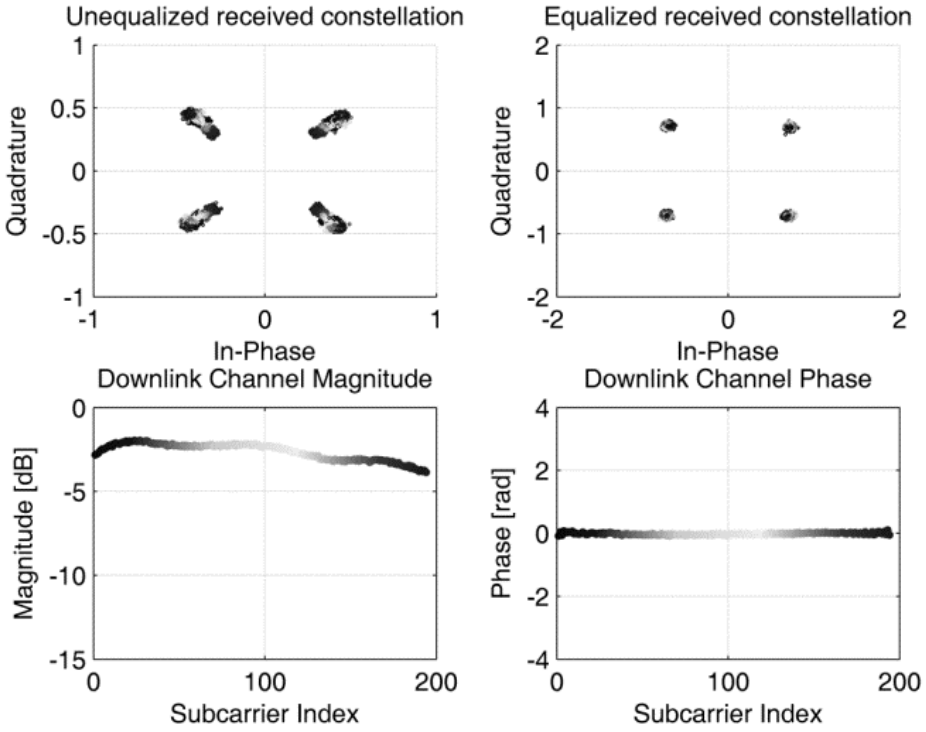


FIGURE 4.6 – TDD EGT.

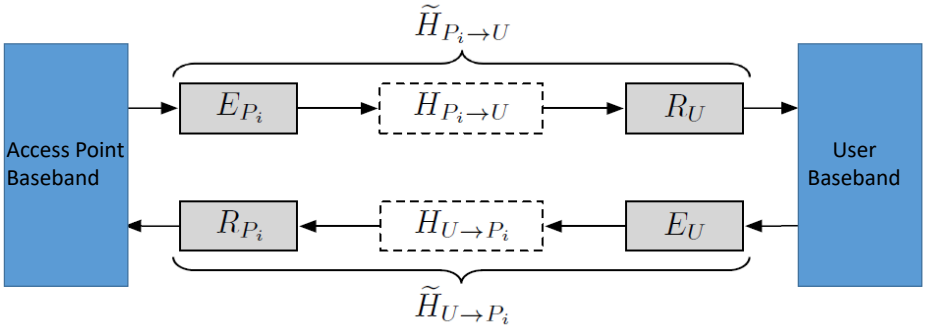


FIGURE 4.7 – BB channel representation.

2.3 Studying the SW limitations

The property of channel reciprocity may be applied when the transmission channel exclusively consists of the radio channel. This property may not be directly applicable in a real platform where the communication channel also includes digital circuits. In FIGURE 4.7, $H_{p_i \to U}$ represents the RF channel frequency response of the i_{th} antenna from the access point to the user and $H_{U \to p_i}$ the RF channel frequency response from the user to the access point. E_{P_i} is the RF transmitter stage frequency response and R_{P_i} is the RF receiver frequency response. Similarly, E_U and R_U corresponds respectively to the transmitter and receiver RF stages from the user side. The RF transmitter stages include DACs, baseband filters, RF modulation, power amplifier, RF filter and the transmitting antenna. Receiver stages are composed of the antenna, RF filter, Low noise amplifier, RF demodulation, baseband filters and ADC converters. The Base-Band (BB) channel frequency response corresponds to the contribution of both the channel itself and the hardware modules that are implemented in the digital circuits.

The BB channel frequency response, of the uplink $\widetilde{H}_{U \to P_i}$ may be expressed as :

$$\widetilde{H}_{U \to P_i} = E_U H_{U \to P_i} R_{P_i} \tag{4.10}$$

The BB channel frequency response of the downlink $\widetilde{H}_{P_i \to U}$ may be expressed as

$$\widetilde{H}_{P_i \to U} = E_{P_i} H_{P_i \to U} R_U \tag{4.11}$$

Then, during a Time Reversal transmission, and if we consider a perfect channel estimation, the estimated channel $\widehat{H}_i$ by the antenna i of the access point is given by :

$$\widehat{H}_i = \widetilde{H}_{U \to P_i} \tag{4.12}$$

During a downlink transmission, the transmitted signal by the antenna i of the access point may then be written in the frequency domain :

$$X_i^{TR} = X \widehat{H}_i^* \tag{4.13}$$

The received signal by the user after a transmission over the BB channel $\widetilde{H}_{P_i \to U}$ is then given by

$$\begin{aligned} Y &= X \sum_i \widehat{H}_i^* \widetilde{H}_{P_i \to U} + B \\ &= X \sum_i (E_U H_{U \to P_i} R_{P_i})^* E_{P_i} H_{P_i \to U} R_U + B \end{aligned} \tag{4.14}$$

where B is the Fourier transform of an AWGN.
The property of the uplink and downlink channel reciprocity allows to write

$$H_{U \to P_i} = H_{P_i \to U} \tag{4.15}$$

then

$$\begin{aligned} Y &= X \sum_{i=1}^{N_t} \left\| H_{P_i \to U} \right\|^2 (E_U R_{P_i})^* E_{P_i} R_U + B \\ &= X \sum_{i=1}^{N_t} \left\| H_{P_i \to U} \right\|^2 \left\| E_{P_i} R_U \right\|^2 \left(\frac{E_U R_{P_i}}{E_{P_i} R_U} \right)^* + B \\ &= X \underbrace{\sum_{i=1}^{N_t} \left\| \widetilde{H}_{P_i \to U} \right\|^2 \Gamma_i}_{H_{eq}} + B \end{aligned} \tag{4.16}$$

with $\Gamma_i \in \mathbb{C}$ and equals $\left(\frac{E_U R_{P_i}}{E_{P_i} R_U} \right)^*$

From Equation 4.16, we can verify that the equivalent channel is equal to the sum of module squares, weighted by Γ_i terms. The constructive sum of BB channel cannot be guaranteed anymore, which leads to a degradation in performance in terms of spatial or timing focus, depending on the Γ_i values. In order to evaluate the impact of Γ_i on the global performances, measurements of RF transfer functions have been performed. E_{P_i}, R_{P_i}, E_U and R_U have been determined. The obtained results are depicted in Figure 4.8. From this figure, it may be clearly seen that a reduction of performances occurs when RF stages are not reciprocal. The exploited diversity is 1 (10 dB) by decade instead of 2 for a perfect reciprocity.

In his thesis, Yvan Kokar has identified two main causes for the non-reciprocity of channels. The first is the Carrier Phase Offest (CPO) and the second is the Sampling Phase Offset (SPO).

▸ Carrier Phase Offset

CPO comes from the fact that the 4 antenna ports have their own frequency synthesizer with a Phase-Locked Loop (PLL) fed by a unique oscillator of 20 MHz. This PLLs make sure that the carrier frequency is exactly the same for all 4 RF modulators and demodulators. Nevertheless, these PLLs may lock at

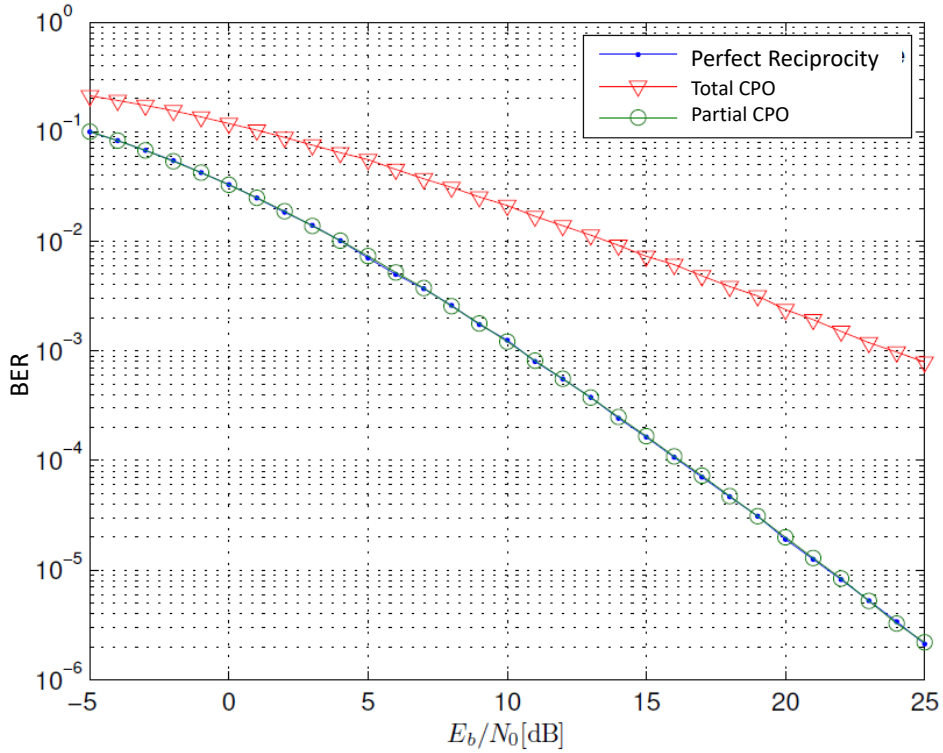
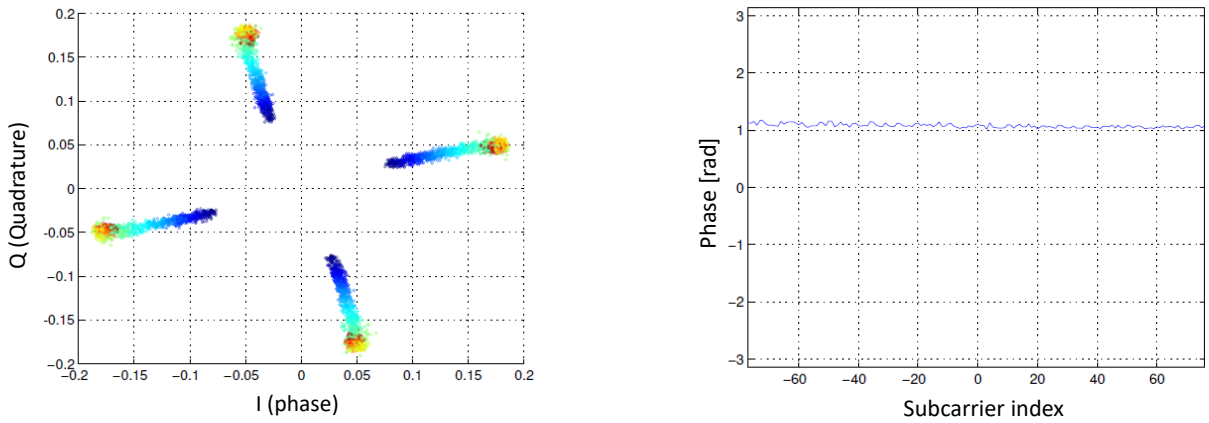


FIGURE 4.8 – BER in function of E_b/N_0.

different times when powered on, which lead to a phase difference between the 4 carriers. This constant phase difference is the source of CPO. FIGURE 4.9a depicts the QPSK symbols that are received after a SISO RT-OFDM transmission with CPO as well as the phase of the equivalent channel (FIGURE 4.9b). The rotation of the constellation which is observed is a direct consequence of CPO which introduces a phase shift for a all sub-carriers.

▸ Sampling Phase Offset

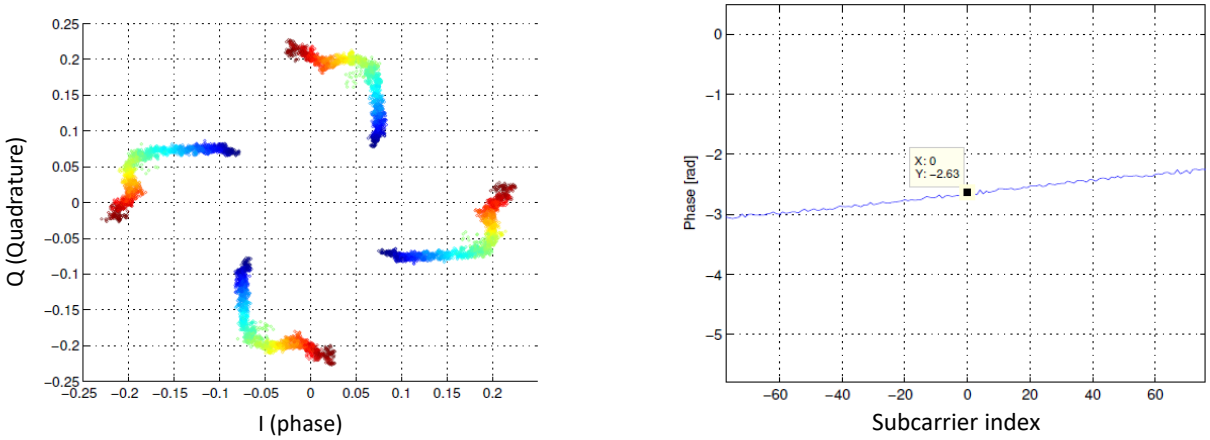
The SPO, which is the second source of channel non-reciprocity comes from the size of wires that connect the oscillator and the ADC/DAC ports. Due to hardware constraints, these sizes are different and imply a phase shift in the sampling clock. FIGURE 4.10a depicts the received QPSK symbols during a SISO TR-OFDM transmission with SPO. The SPO introduces a linear phase shift on all sub-carriers. Note that this shift is directly proportional to τ_s which is the timing difference between two edges of the clocks connected to DAC and ADC respectively.



(a) Received QPSK symbols before equalization.

(b) Equivalent channel phase.

FIGURE 4.9 – SISO RT-OFDM with CPO.



(a) Received QPSK symbols before equalization.

(b) Equivalent Channel Phase

FIGURE 4.10 – SISO RT-OFDM with both CPO and SPO.

▸ Proposed Solutions

In his thesis, Y. Kokar has demonstrated that performing a SISO TR-OFDM transmission between two antennas of the access point makes it possible to estimate the CPO and SPO from the equivalent channel phase. By considering one antenna $i = 1$ as reference, CPOs and SPOs may be evaluated for each antenna ($i = 2,\ldots,N_t$) of the access point. To do so, a calibration phase has been proposed and consists in :

1. sending a pilot symbol between the reference antenna and antenna i
2. estimating the BB channel $\widetilde{H}_{P_1\to P_i}$
3. sending a precoded pilot symbol $X_p^T R = X_P \widetilde{H}^*_{P_i\to P_1}$
4. estimating the equivalent channel $\widetilde{H}_{eq_i} = \widetilde{H}^*_{P_1\to P_1}\widetilde{H}_{P_i\to P_1}$
5. estimating the equivalent phase channel : $\Phi_{eq_i} = arg(\widetilde{H}_{eq_i})$

In our study, calibration coefficients can be directly obtained from the phase of the equivalent channel.

$$C_{P_i}(k) = \Phi_{eq_i}(k) = e^{-j2\overbrace{(\phi_c^{P_i}-\phi_c^{P_1})}^{2xCPO_{P_i\to P_1}}} e^{j\frac{k}{N_{FFT}}\overbrace{2(\phi_s^{P_i}-\phi_s^{P_1})}^{2xSPO_{P_i\to P_1}}} \tag{4.17}$$

By applying the calibration coefficients C_{P_i} on the uplink RF channels $\widetilde{H}_{U\to P_i}$, the transmitted signal by the antenna i of the access point is :

$$X_i^{TR}(k) = X(k)(C_{P_i}(k)\widetilde{H}_{U\to P_i}(k))^* \tag{4.18}$$

After transmitting X_i^{TR} on the BB $\widetilde{H}_{P_i\to U}$ the signal that is received by the user is :

$$Y(k) = X(k)\Psi(k)\sum_i \left\|H_{U\to P_i}(k)\right\|^2 + B(k) \tag{4.19}$$

with

$$\Psi(k) = e^{j2(\phi_c^{P_1}-\phi_c^{U})} e^{j\frac{k}{N_{FFT}}2(\phi_s^{P_1}-\phi_s^{U})}$$

From equation 4.19, it can be seen that the sum of RF channels is constructive. We can also note that $\Psi(k)$ is only a multiplicative term which is only composed of a phase term. Its module is unitary and does not affect spatial focusing.

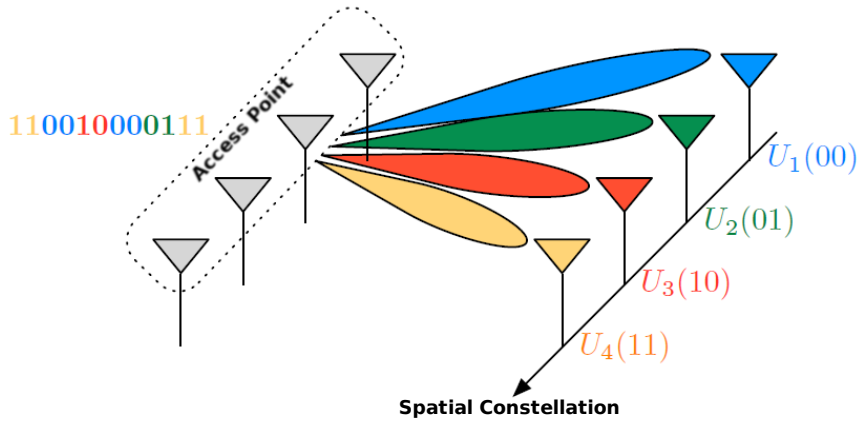


FIGURE 4.11 – RASK modulation scheme.

2.4 The Receive Spatial Modulation scheme

Among the different techniques that have been used to improve the Energy Efficiency of fifth generation (5G) cellular networks, one technique consists in using Spatial Modulation (SM) [CY01]. This technique has already provided significant performances and consists in improving spectral efficiency by using an additional spatial dimension.

In this section, a part of the work that has been carried out in the Spatial Modulation ANR project is presented. In this project, I was responsible for the the Work-Package related to the implementation of a proof of concept in order to demonstrate the feasibility of the approach, in the IoT context. This work has also mainly been performed during the PhD study of Yvan Kokar.

One of the main purpose of this work was to implement the close loop modulation scheme. In this scheme, which is also denoted as Receive Antenna Shift Keying modulation (RASK), the information is carried by the index of the antenna that receives a signal [PHH12a], [MH16]. Note that in this type modulation, the sent signal does not carry any useful information. Accordingly, a very simple demodulation, based on maximizing the real part of the received signal [PHH12a], is performed at the receiver side which makes this technique very promising for future wireless devices. Moreover, RASK modulation can be used simultaneously with another classical modulation (single carrier or multi-carrier) in order to increase the spectral efficiency [SYM+13].

An example of RASK modulation for $N_r = 4$ is depicted in FIGURE 4.11. This type of modulation implies that an access point is capable of focusing a signal towards one antenna at the user side. It could be naturally associated with Time Reversal techniques that we have previously studied. At the user side, the RASK demodulation consists in detecting the index of the targeted antenna to retrieve the sent information.

2.4.1 Prototype Description

In this work, we have used the already developed MISO TR-OFDM prototype used in the previous TRIMARAN ANR project. Here, OFDM has only been used as a support for the RASK modulation although RASK could be used with single carrier modulations. In our platform, when an antenna is targeted, a full OFDM symbol will focus on it. In the following, only RASK is considered.

▸ Uplink Processing : Channel Estimation stage

The purpose of uplink channel estimation is to evaluate the channel between each antenna of the user and every antenna on the access point. The parameters to be determined are $H_{U_j \to P_i}$ with $1 \leq i \leq N_t$ and $1 \leq j \leq N_r$. To do so, the same procedure used in Section 2.3 has been implemented. The uplink baseband

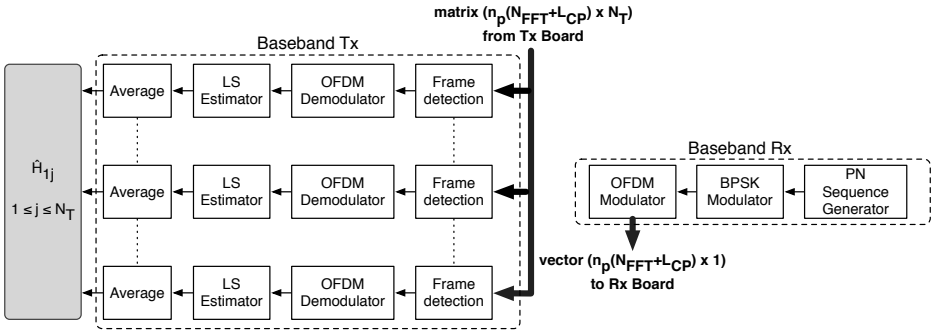


FIGURE 4.12 – Uplink baseband processing.

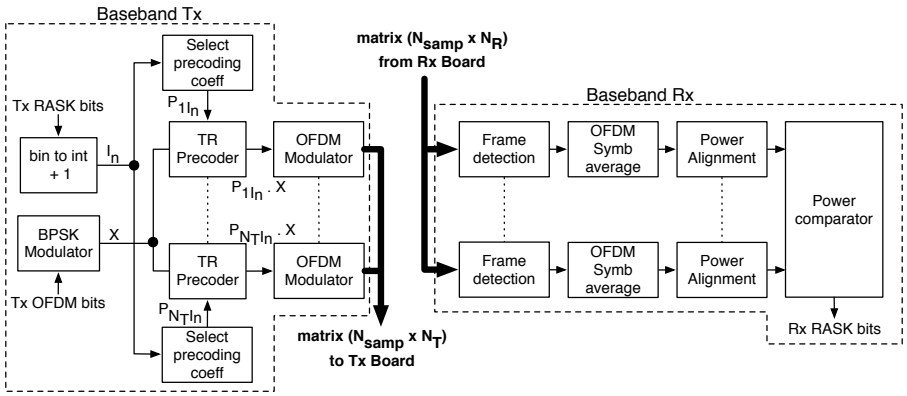


FIGURE 4.13 – Downlink baseband processing.

processing is depicted in FIGURE 4.12.

▸ Downlink Processing : Data Transmission Phase

The downlink architecture is depicted in FIGURE 4.13. For each OFDM symbol, the bit stream to transmit is divided into two sets containing $log_2(Nr)$ bits and NFFT bits respectively. The first set of $log_2(Nr)$ bits constitutes the RASK bits and determines the Rx antenna on which the current OFDM symbol will be focused. The index n of the target antenna is obtained by converting the RASK bits to an integer and add one. To focus on the target antenna, the appropriate precoding vector P_{jn} with $j = 1,\ldots,N_t$ is selected. The second set of NFFT bits, which corresponds to the OFDM bits to be transmitted, passes through a BPSK modulator, the TR filter and finally the OFDM modulator. The resulting signal of the BB Tx is a complex matrix of values that are sent to the Tx board.

At the receiver side, since our main purpose was to evaluate the performance of the RASK modulation, we have only limited our study to the RASK demodulation without considering OFDM demodulation. In our work, RASK demodulation is based on the level of the received power at each antenna, by selecting the n index of the antenna with the maximum received power. n is then used for RASK demapping.

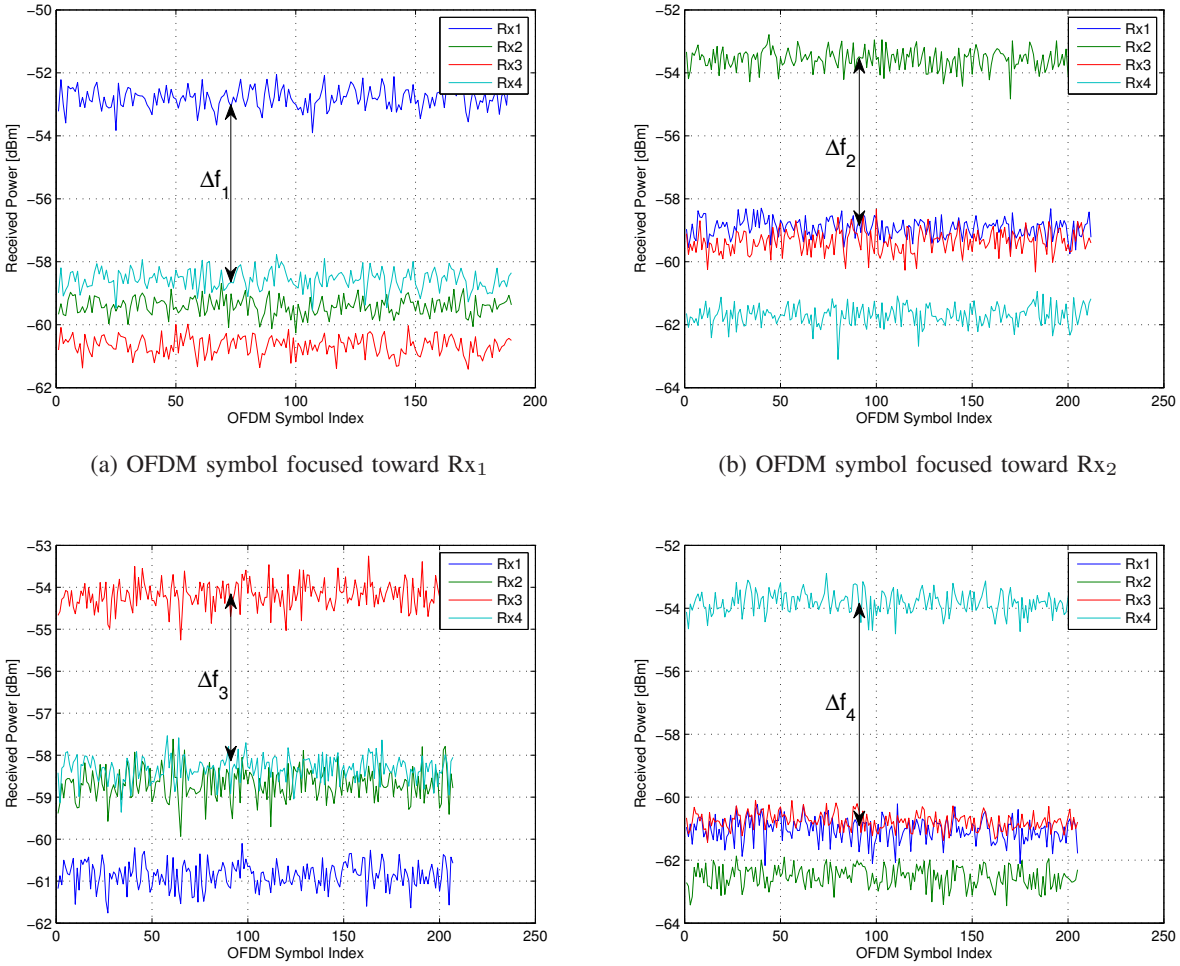
2.4.2 RESULTS

A lot of experiments have been performed using the developed prototype and most of them are described in [Kok18]. As an example, we provide here a Line of Sight (LOS) experiment, in which 2 antennas are used at the transmitter side and 4 receive antennas are implemented at the receiver side. A space of 20 cm between R_x antennas has been respected and the distance between R_x and T_x antennas are 1.7 m.

FIGURE 4.14 depicts the average power of the received OFDM symbols that have been obtained on each antennas according to the target antenna. On the 4 cases, we can notice that the maximum received power always corresponds to the target antenna and that it is quite simple to retrieve the transmitted data. A very

simple detector (threshold comparator) could be considered to recover the incoming sequence of bits. The BER has been measured in this context and is equal to zero. Note that the same results have been obtained in a Non Line of Sight (NLOS) configuration for different distances between the receive antennas.

Finally, in order to evaluate the efficiency and robustness of the proposed approach, we have introduced a focus performance gain Δf. This gain corresponds to the average power margin between the target antenna and the antenna with maximum received power among the non target antennas. According to this metric, it is then obvious that the bigger Δf is, the easier the demodulation will be performed and the more robust the system will be.



(a) OFDM symbol focused toward Rx_1

(b) OFDM symbol focused toward Rx_2

FIGURE 4.14 – Measured power on the 4 receive antennas in a 4x4 LOS configuration.

The results that have been obtained in these studies are very promising and suggest that the algorithms used to implement Spatial Modulation are very relevant in the field of the Iot or future 5G. Furthermore, since the receivers may have a very simple architecture, it can envisaged to drastically reduce the amount of resources, which considerably diminishes the energy required by the receiver.

3 Evaluation of FPGA-Based Wireless Communications Systems

This part of the chapter describes all the studies that we have led in order to propose a new methodology allowing designers to estimate the energy consumed in their communicating systems. This methodology makes use of high-level models destined to be used very early in the design process. Using these models, designers may rapidly explore design choices for a target FPGA device without entering the classic development flow that is often time consuming and error prone. With this approach, designers can easily compare various algorithms and validate hardware choices prior to their final implementation. The privileged hardware targets are FPGA devices but the proposed methodology may also be applied to

other types of devices such as ASICs, for example. The studies presented in this part have been initiated during Jordane Lorandel's PhD thesis and have been followed in Yehya Nasser's PhD works.

In FPGAs, the total power consumption has two main contributors i.e static power and dynamic power. Static power is directly related to the transistors leakage current and thus completely technology-dependent. Dynamic power is the power consumed in the logic design due to the charging and discharging capacitors when transistors are switching, in addition to the short circuit power. Dynamic power is proportional to the switching activity per clock cycle and is highly data and design dependent. Switching activity has a significant impact on this dynamic power [HDL+16]. The expression of the total power consumption is given in Equation 4.20

$$P_{Total} = P_{Dyn} + P_{Stat} = \alpha C V_{dd}^2 f + V_{dd} I_{leakage} \tag{4.20}$$

where P_{Dyn} is the dynamic power that depends on the switching activity factor α (average number of gate transitions per clock cycle), the node capacitance C, the supply voltage V_{dd}, and the clock frequency f. The static power P_{Stat} is estimated as V_{dd} $I_{leakage}$, where $I_{leakage}$ represents the leakage currents. C and $I_{leakage}$ are technology dependent.

3.1 Proposed approach

The proposed methodology is based on the assumption that any hardware system can be represented by a set of hardware IP blocks that are dedicated to a specific function (e.g. Fast Fourier Transform (FFT), encoders, mappers, etc.). The main idea consists in estimating the consumption of a global wireless system, based on an accurate power estimation of its sub-elements. Each sub-element has been fully characterized earlier and is available in a dedicated library. This methodology aims at preventing long development time and at reducing costs by encouraging models re-use. In this work, an important decision was not to focus on static power. This can be easily explained by the fact that this type of power is only related to the type of device and not on the design itself. Therefore, only dynamic power has been considered in the following.

The proposed approach consists of 3 steps :

— Definition of a scenario by the user,

— IP characterization,

— High-Level simulations.

Each step is detailed in the following subsections.

3.1.1 Scenario Definition

To perform an efficient comparison of FPGA-based wireless communication systems, the concept of scenario has been introduced in [Lor15]. This term had already been defined in [GPH+09,ZPR+13,VSP10] but referred to a different concept. In our case, a scenario refers to a set of parameters that are common to several applications within the same domain. It is composed of system and hardware-oriented parameters having an impact over power and/or performance (throughput, latency, etc.). As illustrated in Figure 4.15, the definition of the scenario is the critical entry point of our approach. This concept, that can be seen as a meta-model, has been thought to facilitate the comparison of applications in the wireless communication domain, in terms of performance and power consumption. The methodology requires the development of specific library components that will be detailed in Section 3.1.2.

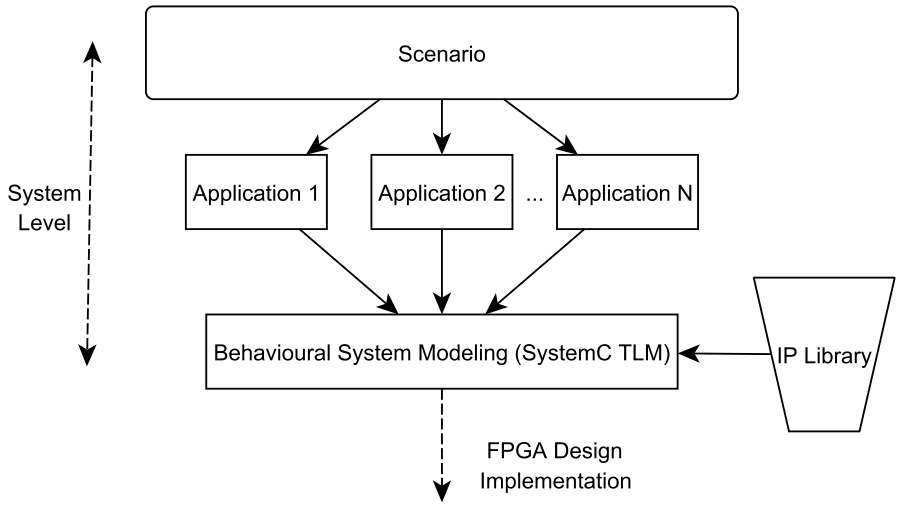


FIGURE 4.15 – The scenario concept.

3.1.2 IP CHARACTERIZATION

A library consists of high-level models of hardware IPs whose behaviour is completely described in SystemC. The library may also contain the corresponding RTL code of the hardware IPs. In our work, all models have been built on components that constitute wireless communication parts such as encoders, modulators, Fast Fourier Transforms (FFT), channel estimators, equalizers, etc.

Each associated model in the library may refer to several hardware IP configurations : (C_i) (i from 1 to n) corresponding to a given combination of identified parameters. These parameters characterize a given IP : input data width, clock frequency, number of outputs, etc. .For each C_i configuration of a high-level model, a library contains the corresponding RTL description whereas another model mimics its behaviour. The RTL description may be expressed using HDLs such a VHDL or Verilog or directly taken from a vendor library.

As depicted in FIGURE 4.16, each hardware IP corresponding to a specific configuration of its high-level model, is then fully characterized by following the different steps of the design process. Design implementation is performed throughout the synthesis, mapping, place and route steps. Note that these steps are performed for a specific FPGA device that exhibits a given number of resources and specific timing properties.

After the IP design implementation, a post Place-And-Route (PAR) VHDL simulation model is generated. This file provides accurate information about delays and timing, based on a final netlist. Furthermore, glitches can also be recorded during this type of simulation. These glitches are transient faults that occur on a signal before it settles to its intended value. They have a significant impact of the signal switching activity. In our work, the ModelSim simulator has been used to capture all internal signals' activity. Testbenches were elaborated according to the users' constraints and generated appropriate input signals. This was very important to record the IP internal activity in two configurations :

— when the IP is active during all the simulation time,

— when the IP is idle, and there is no signal activity.

During the characterization process, power analysis tools such as XPower Analyzer (XPA) [Xil12c] or PowerPlay [Alt14b] from Xilinx and Altera respectively, can be used to deliver average power consumption estimations based on the simulation results and implementation files. The first simulation allows to evaluate the average power when the IP is active whereas the second simulation evaluates the power that is consumed when the IP is idle. The second configuration is usually obtained when control signals such as clock enable, valid signals, etc. are disabled. Note that XPA delivers a complete report on the average power used by clocks, logic elements, signals, memories, DSP blocks, etc. The XPA tool also delivers an average power consumption estimation that is composed of several terms described in Equation 4.21.

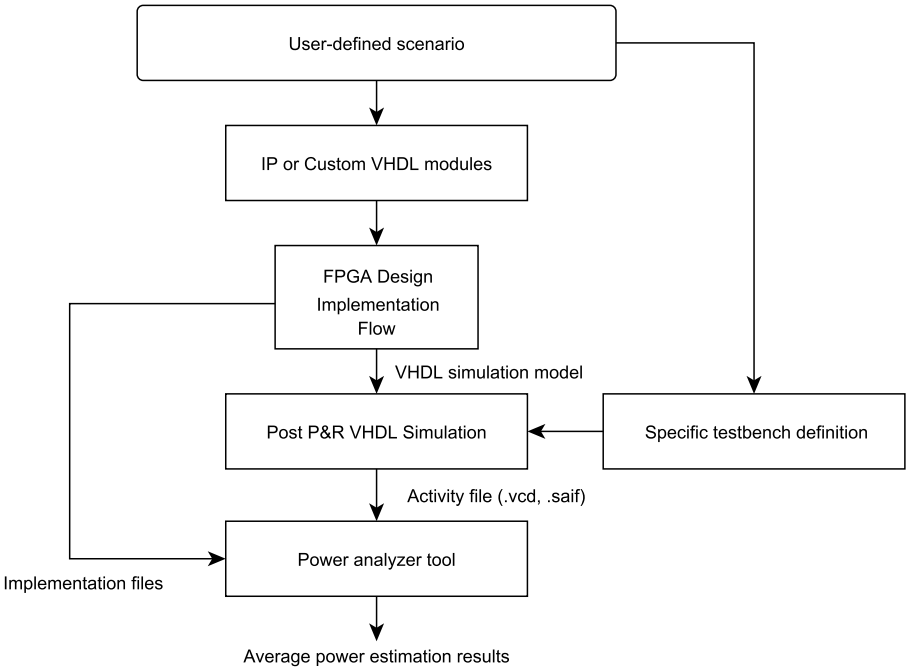


FIGURE 4.16 – IP characterization process.

$$P_{Dynamic_{IP}} = P^{IP}_{Clock} + P^{IP}_{Logic} + P^{IP}_{Signal} + P^{IP}_{I/Os} + P^{IP}_{BRAM} + P^{IP}_{DSP} \quad (4.21)$$

with

— P^{IP}_{Clock}, the average power consumed by the clock network including buffer and routing resources,

— P^{IP}_{Logic}, the average power consumed by all Configurable Logic Blocks (CLBs) including look-up-tables and flip-flops,

— P^{IP}_{Signal}, the average power consumed by the interconnect,

— $P^{IP}_{I/Os}$, the average power consumed by input/output pins,

— P^{IP}_{BRAM}, the average power consumed by specific memories,

— P^{IP}_{DSP}, the average power consumed by Digital Signal Processing (DSP) blocks.

3.1.3 Modeling and High Level Simulation

After obtaining the power metrics for each IP configuration, this information is added to SystemC models that have been developed according to a particular implementation model. As described in Figure 4.17, all SystemC models share the same implementation model that consists of a data path and a control path.

Regarding the data path, the IP functionality is basically described using SystemC. When designers have no implementation details, this description only relies on an high-level behavioural representation. However, some IP vendors provide bit-accurate C models of their hardware IPs. These can be easily integrated into the high-level model in order to provide bit-accurate results of the IP functionality. Furthermore, SystemC supports both floating-point and fixed point data representation that allows designers to evaluate the impact of data quantization in simulations.

Control paths are modelled as Finite State Machines (FSM). FSM states evolve according to both input control signals and IP configuration parameters. The latter are defined by the application. Output control

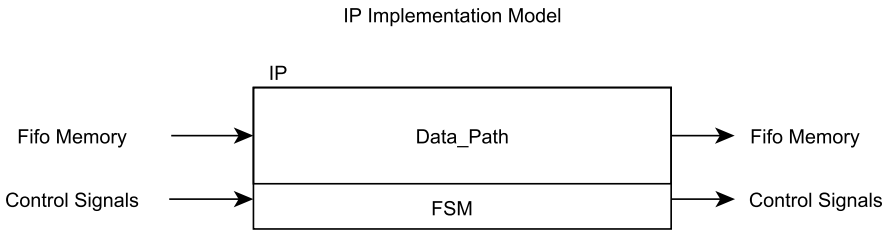


FIGURE 4.17 – Implementation model.

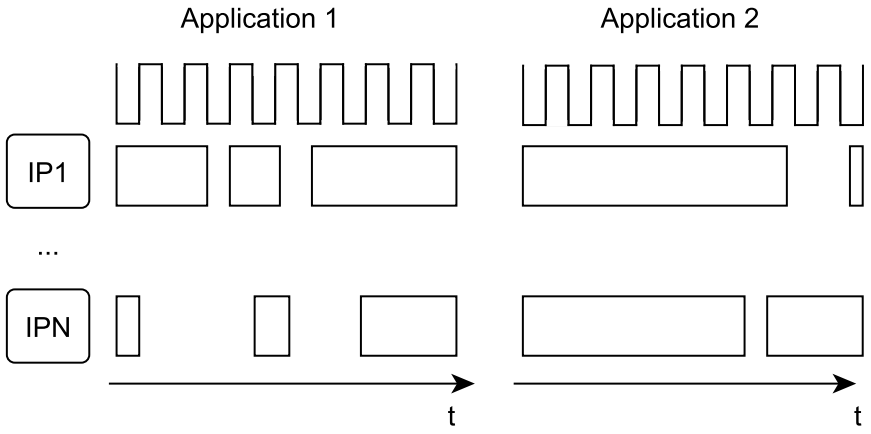


FIGURE 4.18 – IPs time-activity in 2 applications.

signals are propagated to the inputs of the subsequent FSMs, throughout the system. Note that, few implementation details are required because we only focus on the IP behaviour which is generally governed by few key signals (clock enable, input/output data valid signals, etc.). Basically, such details are available directly in IP data-sheets. Using key signals, a cycle accurate behaviour can be modelled, depending on the user-knowledge.

An innovative point of our approach consists in monitoring controls signals in order to determine the time-activity of a block during system-level simulation. To this purpose, an additional SystemC model called "Power Monitor" has been devised, which aims at evaluating the time-activity coefficients of every IP in the system. These coefficients represent the percentage of time during which the IPs are processing data or are in the idle state. They are computed based on the evolution of the control signals during the behavioural simulation.

Moreover, it is of particular importance to consider dynamic behaviours that have a direct impact on power consumption. This is illustrated in FIGURE 4.18. Indeed, two applications that share the same IPs can lead to different power consumption estimations.

The final step of the proposed approach consists in developing the global system model by connecting the different sub-models that have been stored in the library. For example, a complete system may consist of an IFFT block, an LTE encoder and a QAM modulator.

During system-level simulations, many applications can be easily evaluated according to the defined scenario. Designers only have to modify the system parameters and (re)run simulations in order to evaluate the performance of a new application. The stopping criterion of the simulation can also be defined by the user. For example, this criterion can be a simulation duration, a number of data to transmit, a maximum number of errors that are detected at the receiver, etc.

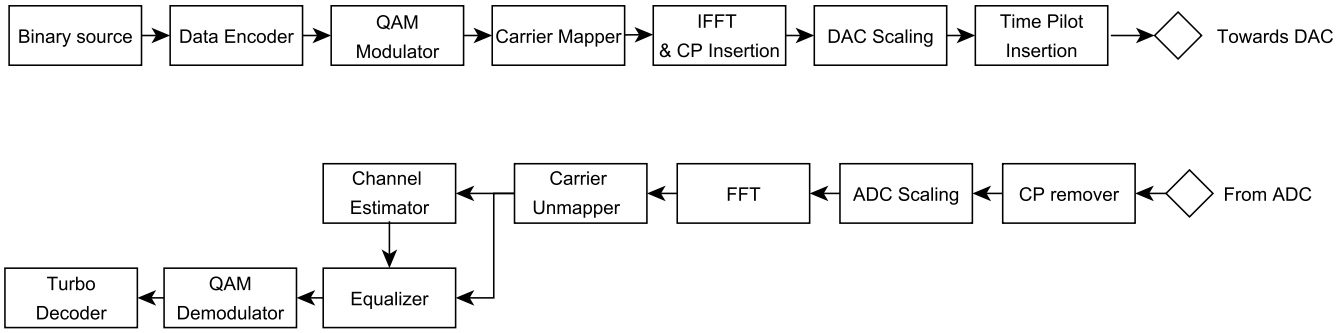


FIGURE 4.19 – SISO-OFDM communication chain diagram.

3.2 Use Case

In order to demonstrate the efficiency of the proposed methodology, a SISO-OFDM wireless communication systems has been described in VHDL in [Lor15]. The architecture of both transmitter and receiver is depicted in Figure 4.19.

3.2.1 System Description

The transmitter consists of a source which provides the binary data to send. These data can be sent to the channel encoder, if there is any, or immediately processed by the QAM modulator that delivers complex I/Q QAM symbols according to the input binary symbols. The modulator supports QPSK, 16 QAM and 64 QAM modulations. The Carrier Mapper module aims to allocate modulated symbols to the corresponding sub-carriers. Moreover, as in real systems, all sub-carriers may not be used. Some of them can be cancelled in order to avoid degradation at the border of the band.

An OFDM modulation is then performed using an IFFT block and a cyclic prefix (CP) is added. OFDM symbols coming from the IFFT can be scaled by the DAC-Scaling module according to the Digital-to-Analogue converter resolution. The last block of the transmitter realizes the time pilot insertion, which enables channel estimation at the receiver side. The receiver performs the dual operations, starting with removing the CP and ADC data scaling. Then, the FFT is computed and carrier un-mapping only keeps the useful sub-carriers. A channel estimation is performed on every dedicated OFDM pilot symbol. Finally, received signals are equalized according to the channel coefficients and a QAM demodulation is performed to retrieve the binary data. In this example, turbo decoding is used for channel decoding.

The global communication chain has been completely described in VHDL and some IP cores from Xilinx have been used. The architecture is highly configurable according to user-defined scenarios. Some parameters that can be chosen are :

— data quantization (number of bits to represent the QAM complex symbols) of every module,

— Fast Fourier transforms size [256 :2048],

— Length of the cyclic prefix,

— Modulation orders [QPSK, 16QAM, 64QAM],

— Number of used sub-carriers,

— Frame structure i.e. the number of OFDM data symbols between each OFDM pilot symbol (for channel estimation),

— Code block size for channel encoding,

— ...

TABLE 4.4 – Considered scenarios and applications in the SISO-OFDM model.

Scenario	Appli. 1	Appli. 2	Appli. 3	Appli. 4
Channel Coding	None	R=1/3		
		Block size = 1024 bits		
Channel Decoding		Turbo Decoder (1 iteration)		
Modulation	QPSK			
(I)FFT size	256	2048	256	2048
Cyclic Prefix Length (in QAM Symbols)	32	256	32	256
Number of used subcarriers	256/256	2048/2048	256/256	2048/2048
Quantification	10 bits		14 bits	
Frame	1 OFDM pilot symbol every 10 OFDM data symbols			
FPGA	Xilinx Virtex-6 LX240T			
Clock Frequency	50 MHz			

TABLE 4.5 – Power estimation results according to 4 SISO-OFDM applications.

		XPower (ref) (mW)	Proposed Methodology (mW)	Relative Error (%)[1]	Spread-sheet (mW)	Relative Error (%)[2]
Appli. 1	TX	55.36	58.19	5.11	92.33	66.8
Appli. 2	TX	84.58	80.7	-4.81	120.93	43
	RX	244.26	245.7	0.59	358.55	46.8
Appli. 3	TX	65.48	68.76	5	107.6	57.1
	RX	263.52	266.84	1.26	406.11	54.1
Appli. 4	TX	98.73	100.04	1.32	145.21	47
	RX	308.38	308.11	-0.01	477.36	54.8

[1] : Relative Error between XPower and the proposed methodology
[2] : Relative Error between XPower and the spreadsheet approach

3.2.2 Power Estimation

First, one scenario has been defined and 4 applications were tested as summarized in Table 4.4. After applying the proposed methodology, average dynamic power consumption results have been obtained for these 4 applications. Estimations from our methodology have been compared with XPower estimations for the entire design i.e. all systems have been developed in VHDL to validate our methodology. Results are given in Table 4.5 and correspond to the average power estimations without considering I/Os dynamic power. The simulation duration has been set to 5 *ms*.

Regarding Table 4.5, power estimations in the proposed methodology are close to the overall power consumption measured by the XPA tool. With a maximal power estimation error of 5% only, this demonstrates the effectiveness of our methodology according to the 4 user-defined applications. The error that is introduced is mainly due to software optimization during Mapping and Place & Route steps. It can be noticed that the error introduced by the Place & Route step can be reduced using a black-box approach and incremental synthesis. Such techniques can force the tool to map a specific core into a dedicated region of the FPGA. For a given IP, the hardware resources and the interconnections between them will be identical as in the final system. Our approach can then deliver an upper bound of power consumption. Moreover, time-activity coefficients, that are evaluated using a high-level model of the system, are approximated values. This also constitutes another source of error.

According to the results, it can be noted that, between Applications 3 and 4, the average power consumption only increases of 33 mW whereas the IFFT and the CP sizes are multiplied by a factor of 8. Channel decoding that is performed using a turbo-decoder, obviously consumes a significant part of the energy due to its high algorithm complexity. However, it enables the significant improvement of the level of performance i.e. the reduction of the BER (Bit-Error Rate) for a given SNR (Signal-to-Noise ratio).

A comparison between our approach and the power estimation using spreadsheets has been performed. In the spreadsheet approach, the sum of the average dynamic powers is computed when IPs are active, without considering time-activity coefficients. As indicated in Table 4.5, power consumption values that are estimated using the spreadsheet approach give an important relative error, ranging from 43% to 57.3% and from 46.8% to 122% for transmitters and receivers respectively. Note that the relative error

TABLE 4.6 – Simulation speed-up.

		Proposed Methodology (s)	XPower Analyzer (h)
Appli. 1	TX	1.27s	12h09
Appli. 2	TX	2s	19h51
	RX		35h07
Appli. 3	TX	1.9s	15h46
	RX		30h15
Appli. 4	TX	2s	22h
	RX		50h56

has been computed as follows :

$$RelativeError = \frac{(Value - Reference)}{Reference} * 100 \quad (4.22)$$

where $Reference$ is the power consumption given by XPower and $Value$ is the power consumption obtained using our methodology or by using the spreadsheet approach.

3.2.3 Power Estimation Speed-Up

Another key advantage of the proposed approach is to provide very fast power estimations. As indicated in Table 4.6, gate-level simulations using XPower Analyzer can take several hours or even days to simulate few milliseconds of a SISO-OFDM communication chain. As an example, for 4 applications, the low-level simulation duration ranges from 12h09 to 50h56, which is quite significant to compare performance. Moreover, such duration corresponds to the time needed for only one configuration. Therefore, it prevents an efficient exploration and the test of many configurations.

Using our approach, it only takes few seconds (ranging from 1.27s to 2s in average) to simulate a complete system (using floating point data representation). The gain is even more important during design space exploration since it allows designers to test a huge number of configurations very rapidly. At high-level, only C++ files need to be modified and compiled between two explorations whereas the entire design flow has to be rerun from scratch using a typical design flow.

4 Towards Fine grain Modeling

Although providing very promising results in terms of accuracy, the methodology presented in the previous section has still a major issue, which is its lack of genericity. This section describes an attempt to improve the energy consumption estimation while drastically reducing the number of components to be developed in the library.

4.1 Analytical Modeling

An approach based on analytic modelling have been proposed by Jordane Lorandel to improve the proposed methodology by reducing the number of configurations that need to be stored in the library. Such techniques have been widely studied in the past and aim at extrapolating power as a function of parameters of interest [EJH06, CA07, RBAN+11, SRH+11, JC12]. In our case, analytic power models have first been developed using curve fitting and linear regressions. These models have been elaborated for different hardware IPs of the wireless base-band processing. Examples of results are detailed in Table 4.7 for a Virtex-6 LX240T FPGA. As indicated in this table, the dynamic power of hardware IPs can be evaluated regarding specific parameters. For example, designers only have to give each parameter a value in

order to estimate the power of a configuration. Using the proposed analytic models, a maximal error lower than 16% is observed as compared with low-level estimations. However, accuracy can be improved using high-level simulation and the evaluation of the IP's time-activity coefficients. In this case, a maximal error lower than 7% is obtained as compared to low-level estimations.

TABLE 4.7 – Examples of analytic power models.

Hardware IP	Analytic Power model
Channel Encoder	$P_{dynamic}\ (mW) = 0.052 + 0.1785 * f + 1.835e-5 *$ CBS
QAM Modulator	$P_{dynamic}\ (mW) = q1(mod1(0.0424 * f - 0.0053) + mod2(0.04425 * f - 0.004) + mod3(0.0514 * f - 0.00133)) + q2(mod1(0.04515 * f - 0.002) + mod2(0.0479 * f + 0.00067) + mod3(0.0591 * f - 0.0013))$
Carrier Mapper	$P_{dynamic}\ (mW) = q1(0.0853 * f - 0.0027) + q2(0.106 * f - 0.0027)$
IFFT	$P_{dynamic}\ (mW) = q1(8.053 - 2.626log(NFFT) - 0.3537 * f + 0.21 * log(NFFT)^2 + 0.2432 * log(NFFT) * f) + q2(-4.613 + 1.531 * log(NFFT) - 0.3551 * f - 0.1225 * log(NFFT)^2 + 0.2863 * log(NFFT) * f)$

$[q1 = 1, q2 = 0]$ for 10 bits of quantization
$[q1 = 0, q2 = 1]$ for 14 bits of quantization
Clock frequency $f = [10 - 100]$MHz
IFFT Size NFFT $= [128 - 2048]pts$
Modulation : QPSK $= [mod1 = 1, mod2 = 0, mod3 = 0]$
16QAM=$[mod1 = 0, mod2 = 1, mod3 = 0]$
64-QAM=$[mod1 = 0, mod2 = 1, mod3 = 0]$

4.2 Extension to other FPGA Devices

One major drawback of the previous approach is that it is specific to an FPGA device and models have to be developed several times when other FPGA families are considered. To circumvent this limitation, scaling factors for each IP have been proposed to extrapolate power consumption values depending on the FPGA family.

In FIGURE 4.20, dynamic power values have been obtained on a simple FIR filter for different types of FPGAs. The size of the filter varies from N = 2 taps to N = 112 taps. As indicated in the figure, the dynamic power does not vary much within a family. It may also be noticed that the best situation occurs for the most recent technology (i.e Virtex-6 in our example). According to these results, it can be clearly seen that scaling factors may be found to obtain a power estimation for all FPGAs families. Examples of such scaling factors are summarized in TABLE 4.8.

4.3 Neural Networks based Modeling

After studying the modeling of hardware IP blocks in FPGA circuits in Jordane Lorandel's thesis, we have been interested in generalizing the approach in order to deal with a reduced set of models. The underlying idea was to overcome the fact that all the components had to be characterized as soon as a simple configuration modification had to be made.

We therefore decided to reduce the granularity of the models by making the assumption that an IP can be seen as a set of basic operators exchanging data. Unfortunately, reducing the granularity of models poses a problem. In this case, it becomes mandatory to build a full system based on these small operators, which is much more complex than the first presented approach. However, we have considered that current high-level synthesis tools are efficient enough to rapidly generate a circuit based on such operators. We could then benefit from these tools to use our models.

We have then proposed a new hardware exploration methodology that aimed at combining both accuracy and fast power estimation. This is achieved by estimating the power of specific operators after the

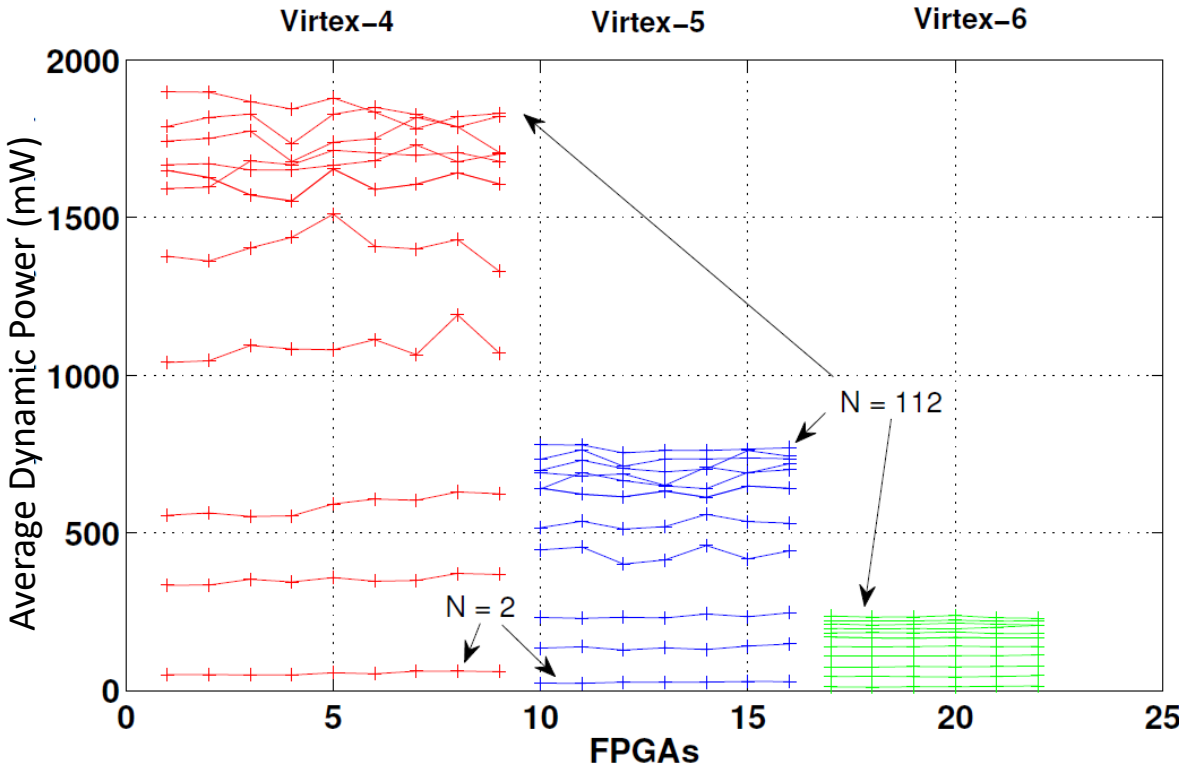


FIGURE 4.20 – Average dynamic power (mW) of a FIR filter according to different FPGAs families.

TABLE 4.8 – Dynamic power scaling factor for different FPGA families and hardware IPs.

IP name	FPGA	Scaling factors			Power estimations (mW)		
		Virtex-4 (ref)	Virtex-5	Virtex-6	Virtex-4	Virtex-5	Virtex-6
FIR Filter	2 Taps	1	0.479	0.225	54.67	26.19	12.285
	32 Taps	1	0.399	0.1015	1087	433.79	110.41
	62 Taps	1	0.391	0.104	1611	630.72	167.58
	112 Taps	1	0.414	0.1261	1850	767.14	233.42
QAM Modulator	QPSK	1	0.415	0.263	8.33	3.47	2.19
	16QAM	1	0.306	0.304	8.02	2.46	2.44
	64QAM	1	0.317	0.23	12.44	3.94	2.88
QAM Demodulator	QPSK	1	0.368	0.266	7.15	2.63	1.9
	16QAM	1	0.259	0.212	13.56	3.51	2.88
	64QAM	1	0.395	0.21	17.09	6.75	3.58
IFFT	128pts	1	0.465	0.332	697	324	231
	256pts	1	0.438	0.303	846.7	370.6	256.52
	512pts	1	0.393	0.277	1159	456	321
	1024pts	1	0.363	0.258	1407	511	364
	2048pts [1]	1	0.478	0.293	1143	546.5	335.6

[1] Lower power due to the automatic implementation of BRAMS by the design tool

hardware implementation on a real platform and by exploiting this information in new models. These pre-characterized models can be easily integrated and simulated to estimate the power consumption of an overall design. This work has been led in Yehya Nasser's thesis that aimed at proposing new efficient models based on artificial intelligence.

Some previous works have already introduced artificial intelligence based on neural networks for po-

wer estimation. Some of them are described in [CYG05], [Roy07]. In [CYG05], a neural network is used to estimate the output statistics and the power consumption of complementary metal-oxide-semiconductor (CMOS) circuits. The average absolute relative error of the proposed method is about 5.0%, at circuit level. Unfortunately, using this method, a lot of circuits have to be simulated to cover most applications, leading to a lack of generality. In [Roy07], macro-modeling based on neural networks has been applied to different components such as multipliers. A comparison of different models shows that neural networks are the most accurate method over linear equations and LUT-based power estimation models.

4.3.1 Model Definition

In our approach, the purpose of the characterization phase is to develop a power model of a component by extracting the relevant information that has a direct impact on power. An example of such operator can be a simple component such as an adder, multiplexer, multiplier, decoder, etc. Each operator has its own size and a defined number of inputs and outputs. In order to evaluate the power consumption of a given circuit, we have proposed to provide each operator with a power model that depends on the activity rate of its inputs. Each operator model consists actually of two sub-models M_1 and M_2 that are described in Figure 4.21.

M_1 constitutes a first model that predicts the power consumption for given Signal Rates (SR) (or α) and Percentage Logic High (PLH) (or p) of all inputs. The signal activity of the inputs is expressed in terms of millions of transitions per second (Mtr/sec). For every input, p refers to the percentage of time during which the signal is at HIGH level, within a period of 1 s. The global model then provides an average of the energy consumed during this period.

The second sub-model (M_2) makes it possible to estimate the signal activity of the outputs as well as the Percentage Logic High according to the operator's inputs. This second model is particularly useful when designers want to propagate the signals' rates as well as the PLH among all connected operators in their design.

In order to obtain a global power estimation for the entire design, it is then possible to sum-up the power contribution of each operator, at high level. With an accurate estimation of signals' activity, these models allow designers to obtain accurate results without spending a lot of time in simulation processes.

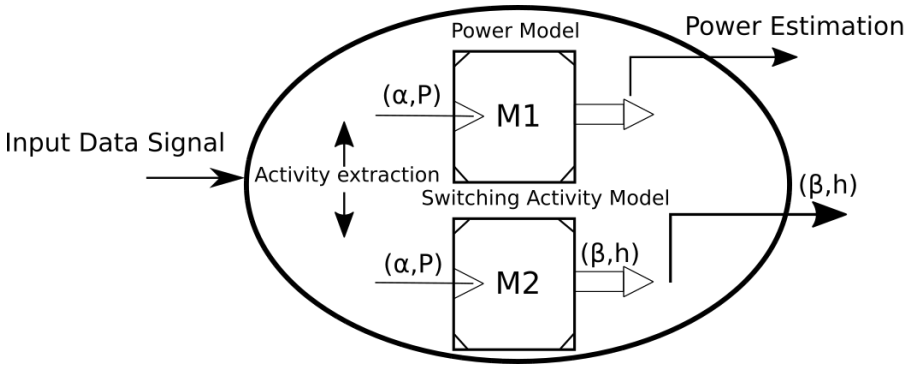


Figure 4.21 – Operator model.

Both M_1 and M_2 sub-models have been implemented using neural networks that are classical tools that have shown their efficiency in lots of domains. They especially perform very well in classification and also in non-linear regression problems. In this study, we have considered basic Multi-Layer Perceptrons (MLP) feed forward networks, that permit to model complex behaviors and may perform multi-dimensional functions approximation. Recall from Chapter 3 that such networks have one or more hidden layers composed of neurons with non linear activation functions and provide an output layer that implements output neurons with a linear activation function. Figure 4.22 shows a typical architecture of an MLP neural network.

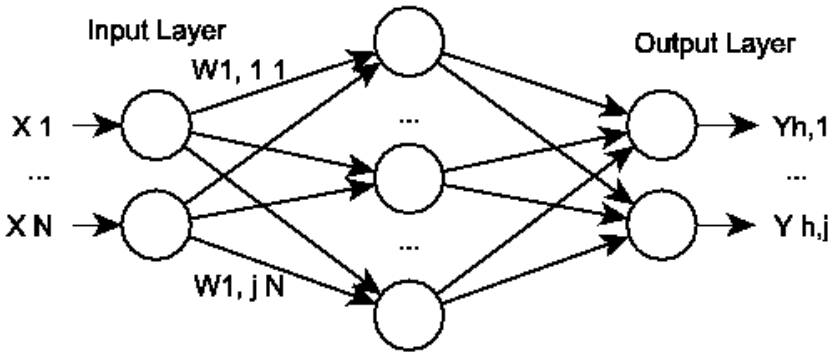


FIGURE 4.22 – MLP neural network architecture.

4.3.2 RESULTS

At operator-level, a verification phase is performed to calculate the accuracy of the proposed models. As shown in FIGURE 4.23, XPA has been used to estimate the power consumption of a 16-bits adder and a 16-bits multiplier. For the same operators, the MATLAB tool has then been used to estimate the dynamic power consumption using the neural approach.

By providing the 16-bits adder / 16-bits multiplier with 10000 samples of Signal Rates and PLH, samples of dynamic power have been extracted from XPA. The exact same inputs given to XPA were also provided to the neural models to evaluate both power consumption and the Signal Rate and PLH. Both M_1 and M_2 neural networks have then 16 (bits) x 2 (inputs) x 2 (Signal Rates + PLH) = 64 inputs.

Both models respectively contain 100 and 150 nodes in their hidden layer. The training set of each neural network consists of 64x10000 data-samples (80% for training, 10% for validation and 10% for testing) with different combinations of Signal Rates and PLH. At the output of the M_1 Neural Network, only one power value is provided. The M_2 Neural Network provides 16 (bits)x 2 (signal rates + PLH) = 32 outputs to propagate the signal activity to other subsequent models. Both M_1 and M_2 models are grouped in a single block that is described in FIGURE 4.23. TABLE 4.9 shows the relative error at operator-level in terms of power estimation. This error has been computed on 10000 data samples.

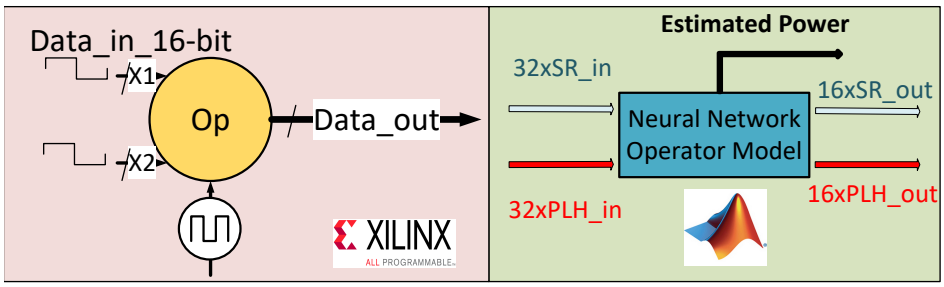


FIGURE 4.23 – Neural network accuracy measurements.

Operators	Real Power (mW)	Modeled Power (mW)	Relative Error (%)
Adder	1.7879	1.7881	0.0112
Multiplier	27.4147	27.4136	0.0040

TABLE 4.9 – Power accuracy for the considered models.

At System-Level, a verification of several scenarios have been performed. FIGURE 4.24 describes a function composed of single operators. As in the per-operator approach, XPA has been used to evaluate the

power of the global system as well as the SR and PLH parameters. In parallel, another global model has been built under Matlab and has consisted in interconnecting both M_1 and M_2 models for each operator and perform a high-Level simulation. This simulation aimed to propagate the parameters throughout all operators and to provide an overall power estimation.

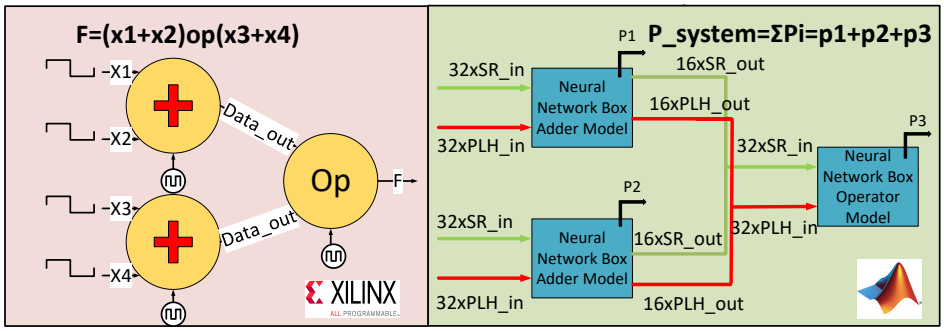


FIGURE 4.24 – System Level power estimation : XPA vs NN

For both simulations, the same inputs, signal rates and PLH combinations of 5000 data-samples have been provided to both XPA and neural models. 5000 data-samples of dynamic power values have been extracted from XPA and from the global neural power models. Note that two functions have been implemented : $F_i = (a \text{ op } b) \text{ op } (c \text{ op } d)$, where *op* can either be an adder or a multiplier.

As in Equation 4, a Mean Absolute Percentage Error (MAPE) has been calculated at System Level to evaluate the accuracy of the approach. P_{XPA} is the power consumption computed by XPA and P_{NN} is the power consumption estimated by our approach.

$$\%\text{MAPE}_{System-level} = \frac{100}{n} \sum_{i=1}^{n} \frac{|P_{XPA} - P_{NN}|}{P_{XPA}} \tag{4}$$

Arithmetic Function	Neural models (MAPE %)
F1(op1=+, op2=+, op3=+)	7.3699
F2(op1=*, op2=*, op3=*)	3.5158

TABLE 4.10 – Models accuracy for different functions.

According to the results given in TABLE 4.10, it can be seen that the proposed method provides a good accuracy at System Level with an error that is less than 8%. Two sources of errors can be identified : first, the SR modeling errors propagate and accumulate through the full design. Second, in XPA, the fact that operators are connected to several neighbors has a direct impact on the impedance and thus on the power consumption. Note that power consumption of the intra-connections (within an operator) is taken into consideration in each neural model

Arithmetic Function	XPA-based (mW)	Initial Estimates (mW)	Relative Error (%)
F1	4.6282	5.3637	15.8965
F2	70.2986	82.2441	16.9925

TABLE 4.11 – The effect of signal activity propagation versus initial estimates.

TABLE 4.11 shows the initial estimates that are only based on the sum of the total average power of each operator (without propagation of signal's activities). These estimates exhibit low accurate results, with a relative error that is greater than 15%, compared with TABLE 4.10. This is due to the fact that propagation is not taken into consideration in this first approach, which demonstrates the significance of signal activity propagation.

5 Summary

This chapter has described different approaches to design energy-efficient communicating systems in an FPGA target. In a first part of the chapter, I have presented the studies that have been led to propose new original waveforms. I have also highlighted the interest of prototyping in order to take the specificity of hardware components into account. The proposed waveforms that have been studied are based on the Time-Reversal techniques and allow designers to build more simple communicating systems with less hardware resources. These waveforms are particularly well suited for the IoT or 5G domains.

In a second part of the chapter, I have presented a new methodology to explore the design space of FPGA devices by focusing on the energetic aspects. The main objective of this methodology is to allow designers to rapidly simulate their algorithms at high-level. This provides a first estimate of the power consumed by their architecture in the first steps of the design flow.

The methodology has been first applied on IP components. Then our approach has moved towards smaller entities to generalize the approach. Analytic models have been proposed to significantly simplify the characterization phase and extend to several FPGA families. Finally, new models based on neural networks have been elaborated to be used at high level, which considerably reduces the simulation time. The obtained results are promising in terms of accuracy and exploration time.

Summary (PhD/Dissemination/Projects)

- 3 PhD students : Jordane Lorandel, Yvan Kokar, Yehya Nasser
- Related publications : 21
- National Projects : ANR Trimaran Project, ANR Spatial Modulation Project
- Regional Projects : Labex RELIASIC

Research Perspectives

Chap. 5

In the previous chapters, I have tried to give an overview of research works that have been led thanks to some of the PhD students that I had the chance to supervise. These chapters have identified three main topics that share a common target : reconfigurable devices. In this chapter, I expose the different research directions that I intend to follow in the next years and explain my motivation.

In Section 1, I describe the scientific issues and challenges that are related to the virtualization of resources in a reconfigurable system and the directions I intend to follow to tackle some of them. Section 2 presents the main problems that are related to reconfiguration management and provides some prospects in this domain. Finally, Section 3 deals with energy-efficient devices and discusses the new research opportunities that are going to emerge in a near future.

1 Embedded Systems Virtualization

Although virtualization has been studied a lot in domains such as servers, data centers, personal computers, this topic is much more recent in embedded systems. In our work, we have proposed and developed our custom solution targeting very simple systems. Several thesis that I supervised have contributed to build and improve the proposed system which is now ready to be fully exploited. In future prospects, I really would like to take benefit from all the work that has been done on this system to exploit new research ideas.

A lot of issues have been identified during the implementation of the Ker-ONE custom micro-kernel. Some of them are directly related to the design itself. Although methodologies like OveRSoC have considerably helped in defining and building the system, there are still a lot of considerations to take into account in order to obtain a final product. Designing an operating system to be executed on recent devices remains very challenging and tricky. These devices have become so complex that it is necessary to take into account all their architectures and features, which is almost impossible in practise.

One possible solution might be modular design. Instead of considering monolithic kernels, the idea is to separate most of the functions into isolated entities communicating with each other. The design effort of such systems may then significantly be reduced. Moreover, high-level software approaches are much more convenient in this case, focusing on the framework and the organization of these entities. Code generation is also much more efficient in this case. Nevertheless, modular approach has severe drawbacks in terms of performance since a huge overhead has to be deplored compared to monolithic solutions. Most of the overhead comes from the communication between entities. Generally, Inter-Process Communication (IPC) mechanisms have to be considered, which drastically increases the whole latency of the system. However, we decided to follow the modular approach in our kernel design and we hope having shown that the communication overhead can be very low and compatible with most applications constraints.

Other specific challenges have been identified during our studies on hardware virtualization. They are described in the next sections and will be part of our main concerns in a near future.

1.1 Hypervisor structure

The first research challenge that I would like to tackle in the future consists in improving the hypervisor structure and performance. Today, most of recent devices are equipped with virtualization extension mechanisms which allow guest OS to access hardware resources such as memory and peripherals very easily. This is, of course, a very promising approach that definitely increases virtualization performance.

For this reason, I would like to continue investigating new virtualization mechanisms that can be easily implemented, while guaranteeing a small overhead on the execution time of the applications. The virtualization approach that we followed in Ker-ONE was based on para-virtualization, and I still believe that this type of virtualization is interesting and can be very efficient in small systems. Nevertheless, I would also like to implement Ker-ONE on devices that benefit from hardware virtualization extension in order to compare both approaches in terms of performances. This could be done in a very near future.

Another interesting point dealing with the hypervisor structure is to study the possibility of migrating most of the kernel's services into the user-space in order to reduce the kernel's critical part to a minimum. This consists in seeing the kernel as a set of user services with no privilege and reduced rights. The reason behind that is to protect the system from being attacked by reducing the surface of code that has access to critical hardware parts.

Finally, in the case of reconfigurable devices, I would like to take benefit from hardware resources to propose new virtualization acceleration mechanisms. These mechanisms include memory and peripherals management, reconfigurable resources and may provide dedicated hardware services to isolate application tasks from critical hardware parts. This work has already been initiated in Tian Xia's thesis and continued in Ye Tian's thesis.

1.2 Reconfigurable Hardware Resources Sharing

Another interesting challenge in embedded virtualization is the particular context of reconfigurable devices such as FPGAs. This document has described parts of our works in this domain and has illustrated different services to handle hardware accelerators. In our work, we have assumed that software tasks running on different guest OS could access reconfigurable resources in a transparent manner. We have also described a basic preemption mechanism, leaving the management of this preemption to the designer itself. As a future prospect, I would like to investigate hardware tasks' preemption more deeply so that this process could be completely transparent to users. Preemption of hardware tasks generally requires to store the state of all flip-flops in the accelerator, and perform a context switch. This is usually performed by reading the partial bitstream corresponding to the reconfigurable area, which is very time-consuming. I would like to propose an original dedicated hardware service that is capable of storing/restoring the internal state of registers without any bitstream manipulation. This work has just started in Ye Tian's thesis.

1.3 VM Scheduling and Off-Loading Service

Another research axis dealing with embedded virtualization concerns the schedulability of virtual machines and software tasks in general. In our previous works, we have studied and proposed some scheduling algorithms that may be compatible with real-time constraints. The proposed policies are based on a hierarchical scheduler that provides promising results. In future works, I would like to investigate new scheduling policies and also focus on the software/hardware tasks interactions. This problem is quite difficult and has already been partially treated in Yaset Oliva's thesis with the off-loading concept. I would like to carry on these previous works and finally implement them on a real hardware platform.

2 End to End Reconfiguration Management

This second research prospect consists in studying the reconfiguration of a system globally. In our current work, we already have proposed a 3-layers model (Application, OS, Hardware) to manage reconfiguration, that has been presented in this document. We have proposed a set of hardware and software services that allow to abstract the underlying hardware architecture. In this model, a layer N abstracts the services of layer N – 1 in a classic approach and APIs make the communication between different layers possible.

2. End to End Reconfiguration Management

In the future, I want to continue refining and extending this model and use it as a framework for the new researches that are going to be led in the laboratory. The main purpose of this approach is to make the management of reconfiguration as transparent as possible for end-users, while guaranteeing the respect of the application constraints.

If we continue considering the wireless communications domain, an ideal system, for me, would be composed of different applications running on several Operating Systems that run on top of our hypervisor. Every application could be seen as a set of tasks with different levels of criticity and run either on an RTOS or a GPOS. An application, for example, could be in charge of executing all protocol layers of a given communication standard. Another one could be dedicated to the implementation of another communication standard. Finally, other applications could execute non-critical applications such as Graphical User Interfaces (GUI). In this ideal scenario, all applications could share a set of hardware resources that are available on a reconfigurable device like an FPGA. The final purpose would be to have a system capable of switching transparently and seamlessly from a standard to another without being perceivable by end-users.

2.1 Multi-standards Reconfiguration

In the presented works, we already have presented the concept of Vertical Handover (VHA), that may decide to switch from one communication standard to another according to the environment. This concept is very interesting and very promising and we have only started to glimpse its complexity. In our ideal scenario, it would correspond to an application running on a transmitter that would like to transmit data e.g a video to a receiver, independently of the standard and without disrupting the communication. Until now, our studies have focused on the physical layer configuration since it corresponds to our level of expertise. Nevertheless, switching from one standard to another is much more complex than simply considering this layer. According to the OSI model, all communication layers have to be taken into account, which makes it very difficult to handle.

In more recent works, we have started to address this issue in Wael Ayoub's thesis, in collaboration with the Lebanese University. The domain that is considered in this thesis is IoT and we investigate the use of IPv6 over 3 Low Power Area Networks (LPWANNs) i.e. LoRaWAN, DASH7 and NB-IoT. The idea here is to study the mechanisms that could be handled or adapted in IPv6 to run on top of other IoT protocols. The main objective remains to switch seamlessly from one standard to another and especially in the context of mobility. This work is only as its beginnings but should lead to very interesting results.

2.2 Machine Learning

Another prospect that I really would like to study in a near future is the use of Machine Learning techniques in the context of reconfiguration management. I have shown in this document, that we have started to use this type of technique in VHA. According to the obtained results and due to the fact that I made intensive use of neural networks in the past, I am really convinced about their relevance. Using machine learning as a decision process in reconfiguration management seems very promising although quite tricky. The main problem with such systems has always been to obtain enough pertinent data to be able to train the network correctly. In our current systems, this issue could be circumvented by continuously acquiring data to sense the environment and to improve our knowledge of the problem. Coupled with re-inforcement techniques, I am convinced that neural-based solutions could be used very efficiently.

Finally, using machine learning could lead to "smart" systems that are able to learn by themselves and adapt very easily to different situations. It is today conceivable to design reconfigurable systems that can automatically adapt their hardware resources according to the applications requirements.

3 Towards Energy-Efficient Communicating Devices

In this section, the last research axis to which I would like to contribute is described. This prospect deals with low-power communicating embedded systems, which will keep on constituting a hot topic in the near future. It is clear that there is today a huge need in terms of energy consumption management for this type of devices and solutions must be found to preserve energetic resources.

In our works, we have tried to tackle this challenge by considering the design process of embedded systems. A first research axis has consisted in proposing new waveforms at the physical layer to drastically reduce transmitting power. Another axis has focused on the elaboration of high-level models to facilitate the exploration of the design space and thus find the best suitable architecture for communication. Section 3.1, describes the first axis and the associated prospects, whereas section 3.2 details the second axis.

3.1 New Waveforms

In this part of our research, we have sought to propose new waveforms that can efficiently reduce the power that is required to transmit data. In this context, we have studied Time Reversal techniques and spatial modulation schemes that have been described in this document. These studies have been led in the context of 2 ANR projects in which I was responsible for the proof of concept implementation.

Today, our research team has the opportunity to benefit from the developed hardware platforms to test new algorithms and to get a clear vision on consumption and performance. Having acquired some expertise in the field, I would like to continue contributing in the platform improvement to propose and evaluate new algorithms in new case studies.

One of the envisaged case-study deals with 60 GHz communicating systems. These new systems are probably going to be used extensively in IoT and will require a lot of computing power to be used in a massive MIMO context. We would like to propose new efficient communications schemes to reduce power, while guaranteeing the best performance.

3.2 Hardware Power Models

Concerning the elaboration of new power models, several works have already been led, in particular in Jordane Lorandel and Yehya Nasser's thesis which have provided very promising results. Power models have been developed and used in our methodology to perform high-level simulations in SystemC. Some models have been described at the IP level, and others at the operator level to obtain more flexibility and genericity. Although very interesting, this approach has an important limitation that comes from the fact that power estimations are performed by vendors' software tools and may be very far from the actual consumed power.

One research track that I would like to follow would consist in elaborating power models coming from real measurements. Our research team has just invested in a power measurement platform that could help in measuring the power consumed in function of input stimuli. This type of platform could be used to obtain a lot of data that could constitute a huge database. The next step would be to exploit our neural models on this database to get even more accuracy. In a first approach, new data pre-processing and post-processing could be proposed to reduce the complexity of the problem. Then, other techniques coming from deep learning could be envisaged due to the massive number of input data that we could get to train the network.

3.3 From Device to Protocol

A last axis of research consists in extending the existing models that have been developed at IP level or at operators' level to the different elements that compose a communication device. At the physical layer, agile RF front-end, reconfigurable antennas, cognitive capabilities, flexible digital processing are possible solutions to improve system autonomy and QoS. In addition, new communication strategies at other layers (MAC, IP, etc.) could be imagined and modeled to optimize the overall efficiency of transmissions in terms of energy and performance. Today, many communication protocols are available such as LPWAN (Sigfox, Lora), BLE, Z-wave, Zigbee, IEEE 802.15.4, that may be jointly implemented on a same device. It could be interesting to study these different protocols in terms of power consumption and performance, at different layers. Finally, another solution to reduce power in IoT devices could consist in optimizing the processing load. HW/SW partitioning, computation overload, distribution of the processing load among several heterogeneous communicating objects are examples of methods that can be implemented to reduce the overall power.

While these methods can be easily considered in the design process, it is challenging to evaluate their impact in a global context. In the IoT context for example, as far as we know, there is no study that takes into account all possible sources of power reduction at a high level of abstraction, within a single simulation environment. This could be a very interesting prospect to handle.

Bibliography

[AA16] Nasser M. Alotaibi and Sami S. Alwakeel. A neural network based handover management strategy for heterogeneous networks. In *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015*, 2016.

[ABG14] A. Ahmed, L.M. Boulahia, and D. Gaiti. Enabling vertical handover decisions in heterogeneous wireless networks : A state-of-the-art and a classification. *IEEE Communications Surveys & Tutorials*, 16(2) :776–811, 2014.

[AHK+14] Andreas Agne, Markus Happe, Andreas Keller, Enno Lubbers, Bernhard Plattner, Marco Platzner, and Christian Plessl. Reconos : An operating system approach for reconfigurable computing. *Micro, IEEE*, 34(1) :60–71, 2014.

[Aic14a] Mohamed El Mehdi Aichouch. *Evaluation of a multiple criticality real-time virtual machine system and configuration of an RTOS's resources allocation techniques*. Theses, INSA de Rennes, May 2014.

[All15b] NGMN Alliance. 5g white paper. 2015.

[Alt14b] Altera Incorporation. Powerplay early power estimator user guide, July 2014. User Guide UG-01070.

[Alt15a] Altera. *Cyclone V Device Overview*. Altera Corporation, 2015.

[Alt15b] Altera Inc. Dsp builder introduction, May. 4 2015. Handbook.

[AMM12] B. Ammar A., D. B. Mohd, and I. Muhammad. Ieee 802.21 based vertical handover in wifi and wimax networks. In *2012 IEEE Symposium on Computers Informatics (ISCI)*, pages 140–144, March 2012.

[APA+06a] Jason Agron, Wesley Peck, Erik Anderson, David Andrews, Ed Komp, Ron Sass, Fabrice Baijot, and Jim Stevens. Run-time services for hybrid cpu/fpga systems on chip. In *Real-Time Systems Symposium, 2006. RTSS'06. 27th IEEE International*, pages 3–12. IEEE, 2006.

[ARFB10] O. Arnold, F. Richter, G. Fettweis, and O. Blume. Power consumption modeling of different base station types in heterogeneous cellular networks. In *Future Network and Mobile Summit, 2010*, pages 1–8, June 2010.

[Arn95] Jeffrey M. Arnold. The splash 2 software environment. *J. Supercomput.*, 9 :277–290, May 1995.

[ARW+10] N. Audsley, M. Richardson, A.J. Wellings, K. Tindell, and A. Burns. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 2010.

[BBC+07] Björn B. Brandenburg, Aaron D. Block, John M. Calandrino, UmaMaheswari Devi, Hennadiy Leontyev, and James H. Anderson. Litmus rt : A status report, 2007.

[BBFH+08] H. Breuil, D. Burette, B. Flüry-Hérard, J. Cueugniet, D. Vignolles, and H. Boisson. Tic et développement durable, Déc. 2008. Rapport du Conseil Général de l'Environnement et du Développement Durable et le Conseil Général des Technologies de l'Information.

[BCLK+16] Stefano Buzzi, I. Chih-Lin, Thierry E. Klein, H. Vincent Poor, Chenyang Yang, and Alessio Zappone. A survey of energy-efficient techniques for 5G networks and challenges ahead. *IEEE Journal on Selected Areas in Communications*, 2016.

[BD05] Bin Lin and P.A. Dinda. VSched : Mixing Batch And Interactive Virtual Machines Using Periodic Real-time Scheduling. 2005.

[Bes13] Joel Best. *Real-Time Operating System Hardware Extension Core for System-on-Chip Designs*. PhD thesis, School of Engineering,University of Guelph, 2013.

[BGB06] Lilian Bossuet, Guy Gogniat, and Wayne Burleson. Dynamically configurable security for SRAM FPGA bitstreams. *International Journal of Embedded Systems*, 2006.

[BHH+07b] Jürgen Becker, Michael Huebner, Gerhard Hettich, Rainer Constapel, Joachim Eisenmann, and Jürgen Luka. Dynamic and partial fpga exploitation. *Proceedings of the IEEE*, 95(2) :438–452, 2007.

[BPS15] Meena Belwal, Madhura Purnaprajna, and TSB Sudarshan. Enabling seamless execution on hybrid cpu/fpga systems : Challenges & directions. In *Field Programmable Logic and Applications (FPL), 2015 25th International Conference on*, pages 1–8. IEEE, 2015.

[BPZ02] Peter Blomgren, George Papanicolaou, and Hongkai Zhao. Super-resolution in time-reversal acoustics. *The Journal of the Acoustical Society of America*, 111(1) :230–248, 2002.

[CA07] S. Chandrasekaran and A. Amira. A new behavioural power modelling approach for fpga based custom cores. In *Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on*, pages 350–357, Aug 2007.

[Cad11] Cadence. Cadence system development suite, 2011.

[CCEPO12] Laura Conde-Canencia, Yvan Eustache, J-C Prévotet, and Yaset Oliva. Modeling adaptive coded modulation in real time partially reconfigurable mobile terminals. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1049–1053. IEEE, 2012.

[CCLJ12] Oliva Y. Conde-Canencia L., Eustache Y. and Pr?votet J.C. Implementing adaptive coded modulation in real time partially reconfigurable mobile terminals : A modelling approach. *European Signal Processing Conference (EUSIPCO-2012)*, 2012.

[CF04] Yan Chen and Toni Farley. QoS requirements of network applications on the Internet. *Information Knowledge Systems Management*, 2004.

[CHY+14a] Y. Chen, F. Han, Y. H. Yang, H. Ma, Y. Han, C. Jiang, H. Q. Lai, D. Claffey, Z. Safar, and K. J. R. Liu. Time-reversal wireless paradigm for green internet of things : An overview. *IEEE Internet of Things Journal*, 1(1) :81–98, Feb 2014.

[CLC+02] Katherine Compton, Zhiyuan Li, James Cooley, Stephen Knol, and Scott Hauck. Configuration relocation and defragmentation for run-time reconfigurable computing. *IEEE Trans. Very Large Scale Integr. Syst.*, 10 :209–220, June 2002.

[Con] Warp project. "http ://warpproject.org".

[CPG+11] Samarjit Chakraborty, Thomas Pfeuffer, Martin Geier, Alejandro Masrur, and Sebastian Drössler. Designing VM schedulers for embedded real-time applications. 2011.

[CWH+16] Y. Chen, B. Wang, Y. Han, H. Q. Lai, Z. Safar, and K. J. R. Liu. Why time reversal for future 5g wireless ? [perspectives]. *IEEE Signal Processing Magazine*, 33(2) :17–26, March 2016.

[CY01] Yawgeng A. Chau and Shi Hong Yu. Space modulation on wireless fading channels. *IEEE Vehicular Technology Conference*, 2001.

[CYG05] W. Q. Y. Cao, Yuanyuan Yan, and Xun Gao. Neural network macromodel for high-level power estimation of cmos circuits. In *2005 International Conference on Neural Networks and Brain*, volume 2, pages 1009–1014, Oct 2005.

[Dav18] G. Davis. 2020 : Life with 50 billion connected devices. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–1, Jan 2018.

[DCH10] T. Dubois, M. Crussière, and M. Hélard. On the use of time reversal for digital communications with non-impulsive waveforms. In *2010 4th International Conference on Signal Processing and Communication Systems*, pages 1–6, Dec 2010.

[DDG+12] C. Desset, B. Debaillie, V. Giannini, A. Fehske, G. Auer, H. Holtkamp, W. Wajda, D. Sabella, F. Richter, M.J. Gonzalez, H. Klessig, I. Godor, M. Olsson, M.A. Imran, A. Ambrosy, and O. Blume. Flexible power modeling of lte base stations. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 2858–2862, April 2012.

[DDVJM12] M. Deruyck, D. De Vulder, W. Joseph, and L. Martens. Modelling the power consumption in femtocell networks. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 30–35, April 2012.

[Dep15] UC Berkeley EECS Department. The ptolemy project, 2015. Website.

[DHCG13] Thierry Dubois, Maryline Hélard, Matthieu Crussière, and Cécile Germond. Performance of time reversal precoding technique for miso-ofdm systems. *EURASIP Journal on Wireless Communications and Networking*, 2013(1) :260, Nov 2013.

[DLC+12] Jiun-Hung Ding, Chang-Jung Lin, Ping-Hao Chang, Chieh-Hao Tsang, Wei-Chung Hsu, and Yeh-Ching Chung. Armvisor : System virtualization for arm. In *Proceedings of the Ottawa Linux Symposium (OLS)*, pages 93–107, 2012.

[DM98] M. C. Davey and D. J. C. MacKay. Low density parity check codes over gf(q). In *1998 Information Theory Workshop (Cat. No.98EX131)*, pages 70–71, Jun 1998.

[DMN+08] J. Delorme, J. Martin, A. Nafkha, C. Moy, F. Clermidy, P. Leray, and J. Palicot. A fpga partial reconfiguration design approach for cognitive radio based on noc architecture. In *2008 Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference*, pages 355–358, June 2008.

[DPML07a] J. P. Delahaye, J. Palicot, C. Moy, and P. Leray. Partial reconfiguration of fpgas for dynamical reconfiguration of a software radio platform. In *2007 16th IST Mobile and Wireless Communications Summit*, pages 1–5, July 2007.

[DPML07b] Jean Philippe Delahaye, Jacques Palicot, Christophe Moy, and Pierre Leray. Partial reconfiguration of FPGAs for dynamical reconfiguration of a software radio platform. In *2007 16th IST Mobile and Wireless Communications Summit*, 2007.

[DS07] R. Damasevicius and V Stuikys. Estimation of power consumption at behavioral modeling level using systemc. In *EURASIP Journal on Embedded Systems, 2007*. Hindawi Publishing Corporation, May 2007.

[ear] Earth project infso-ict-247733 earth deliverable d2.3.

[ECF96] Carl Ebeling, Darren C. Cronquist, and Paul Franklin. Rapid - reconfigurable pipelined datapath. In *Proceedings of the 6th International Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Compilers*, pages 126–135, London, UK, 1996. Springer-Verlag.

[EJH06] D. Elleouet, N. Julien, and D. Houzet. A high level soc power estimation based on ip modeling. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 4 pp.–, April 2006.

[Exp07] ExpressLogic. *Measuring Real-Time Performance Of An RTOS*, 2007.

[GBB+17] Robin Gerzaguet, Nikolaos Bartzoudis, Leonardo Gomes Baltar, Vincent Berg, Jean Baptiste Doré, Dimitri Kténas, Oriol Font-Bach, Xavier Mestre, Miquel Payaró, Michael Färber, and Kilian Roth. The 5G candidate waveform race : a comparison of complexity and performance. *Eurasip Journal on Wireless Communications and Networking*, 2017.

[GBL+09] Maria E Gonzalez, Attila Bilgic, Adam Lackorzynski, Dacian Tudor, Emil Matus, and Irv Badr. Ict-emuco. an innovative solution for future smart phones. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 1821–1824. IEEE, 2009.

[GHZB10] Diana Göhringer, Michael Hübner, Etienne Nguepi Zeutebouo, and Jürgen Becker. Cap-os : Operating system for runtime scheduling, task mapping and resource management on reconfigurable multiprocessor architectures. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE, 2010.

[GHZB11] Diana Göhringer, Michael Hübner, Etienne Nguepi Zeutebouo, and Jürgen Becker. Operating system for runtime reconfigurable multiprocessor systems. *Int. J. Reconfig. Comput.*, 2011 :3 :1–3 :16, January 2011.

[GPH+09] Stefan Valentin Gheorghita, Martin Palkovic, Juan Hamers, Arnout Vandecappelle, Stelios Mamagkakis, Twan Basten, Lieven Eeckhout, Henk Corporaal, Francky Catthoor, Frederik Vandeputte, et al. System-scenario-based design of dynamic embedded systems. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 14(1) :3, 2009.

[GRE+01] Matthew R Guthaus, Jeffrey S Ringenberg, Dan Ernst, Todd M Austin, Trevor Mudge, and Richard B Brown. Mibench : A free, commercially representative embedded benchmark suite. In *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pages 3–14. IEEE, 2001.

[GYG08] Andreas Gerstlauer, Haobo Yu, and Daniel D. Gajski. RTOS modeling for system level design. In *Design, Automation, and Test in Europe : The Most Influential Papers of 10 Years Date*. 2008.

[GYJ01] Lovic Gauthier, Sungjoo Yoo, and Ahmed Amine Jerraya. Automatic Generation and Targeting of Application-Specific Operating Systems and Embedded Systems Software. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2001.

[Has16] Performance evaluation of dynamic partial reconfiguration techniques for software defined radio implementation on FPGA. In *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems*, 2016.

[HD10] Scott Hauck and Andre DeHon. *Reconfigurable computing : the theory and practice of FPGA-based computation*. Morgan Kaufmann, 2010.

[HDL+16] E. Hung, J. J. Davis, J. M. Levine, E. A. Stott, P. Y. K. Cheung, and G. A. Constantinides. Kapow : A system identification approach to online per-module power estimation in fpga designs. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 56–63, May 2016.

[Hei] Gernot Heiser. Virtualizing embedded systems : why bother? In *Proceedings of the 48th Design Automation Conference*, pages 901–905. ACM.

[HGNB10] Michael Hübner, Diana Göhringer, Juanjo Noguera, and Jürgen Becker. Fast dynamic and partial reconfiguration data path with low hardware overhead on xilinx fpgas. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE, 2010.

[HH09] Chun-Hsian Huang and Pao-Ann Hsiung. Hardware resource virtualization for dynamically partially reconfigurable systems. *Embedded Systems Letters, IEEE*, 1(1) :19–23, 2009.

[HKH04] Prih Hastono, Stephan Klaus, and S.A. Huss. Real-time operating system services for realistic SystemC simulation models of embedded systems. *Proceedings of the International Forum on Specification & Design Languages (FDL'04)*, 2004.

[HKRN08] Philipp A. Hartmann, Henning Kleen, Philipp Reinkemeier, and Wolfgang Nebel. Efficient modelling and simulation of embedded software multi-tasking using SystemC and OSSS. In *Proceedings - 2008 Forum on Specification, Verification and Design Languages, FDL'08*, 2008.

[HL10] Gernot Heiser and Ben Leslie. The okl4 microvisor : convergence point of microkernels and hypervisors. In *Proceedings of the first ACM asia-pacific workshop on Workshop on systems*, pages 19–24. ACM, 2010.

[HLF+11] Chen-Wei Hsu, Jia-Lu Liao, Shan-Chien Fang, Chia-Chien Weng, Shi-Yu Huang, Wen-Tsan Hsieh, and Jen-Chieh Yeh. Powerdepot : Integrating ip-based power modeling with esl power analysis for multi-core soc designs. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 47–52, June 2011.

[HRR+04] Fabiano Hessel, Vitor M Rosa, Igor M Reis, Av Ipiranga, Porto Alegre, César A M Marcon, Altamiro A Susin, and Av B Gonçalves. Abstract RTOS Modeling for Embedded Systems Faculdade de Informática Universidade Federal do Rio Grande do Sul. 2004.

[HSH+08a] J. Y. Hwang, S. B. Suh, S. K. Heo, C. J. Park, J. M. Ryu, S. Y. Park, and C. R. Kim. Xen on arm : System virtualization using xen hypervisor for arm-based secure mobile phones. In *2008 5th IEEE Consumer Communications and Networking Conference*, pages 257–261, Jan 2008.

[Huc11] Emmanuel Huck. *High-level simulation of distributed real-time operating systems for design space exploration in heterogeneous MPSoC architectures*. Theses, Université de Cergy Pontoise, November 2011.

[HYW+12] F. Han, Y. H. Yang, B. Wang, Y. Wu, and K. J. R. Liu. Time-reversal division multiple access over multi-path channels. *IEEE Transactions on Communications*, 60(7) :1953–1965, July 2012.

[JC12] Ruzica Jevtic and Carlos Carreras. A complete dynamic power estimation model for datapaths in fpga dsp designs. *Integration, the VLSI Journal*, 45(2) :172 – 185, 2012.

[JH09b] Inwhee Joe and Seokjoon Hong. An adaptive network scanning algorithm in hybrid wireless networks. In *Proceedings of the 4th International Conference on Ubiquitous Information Technologies and Applications, ICUT 2009*, 2009.

[JHE+13] Krzysztof Jozwik, Shinya Honda, Masato Edahiro, Hiroyuki Tomiyama, and Hiroaki Takada. Rainbow : An operating system for software-hardware multitasking on dynamically partially reconfigurable fpgas. *International Journal of Reconfigurable Computing*, 2013 :5, 2013.

[JPC+14] Abhishek Kumar Jain, Khoa Dang Pham, Jin Cui, Suhaib A Fahmy, and Douglas L Maskell. Virtualized execution and management of hardware tasks on a hybrid arm-fpga platform. *Journal of Signal Processing Systems*, 77(1-2) :61–76, 2014.

[JW05] Ahmed Amine Jerraya and Wayne Wolf. The What, Why, and How of MPSoCs. In *Multiprocessor Systems-on-Chips*. 2005.

[KBT08] Dirk Koch, Christian Beckhoff, and Jürgen Teich. Recobus-builder-a novel tool and technique to build statically and dynamically reconfigurable systems for fpgas. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 119–124. IEEE, 2008.

[KG16] Tobias Kalb and Diana Gohringer. Enabling dynamic and partial reconfiguration in Xilinx SDSoC. In *2016 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2016*, 2016.

[KJU+15] M. Khan, C. Jung, P. C. Uzoh, C. Zhenbo, J. Kim, Y. Yoon, A. Nadeem, and K. Han. Enabling vertical handover management based on decision making in heterogeneous wireless networks. In *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 952–957, Aug 2015.

[Koc13] Dirk Koch. Partial reconfiguration on FPGAs. *Lecture Notes in Electrical Engineering*, 153 LNEE :1–295, 2013.

[Kok18] Yvan Kokar. *Etude de la mise en œuvre matérielle d'une transmission sans fil combinant retournement temporel et OFDM*. PhD thesis, 2018. Thèse de doctorat dirigée par Hélard, Maryline Electronique et Télécommunications Rennes, INSA 2018.

[Kon12] Kontron. *No end to the possibilities : x86 meets FPGA whitepaper*. Kontron, 2012.

[KT15] Keysight-Technologies. Systemvue electronic system-level (esl) design software, 2015. Website.

[Kum12] K. A. Arun Kumar. Fpga implementation of psk modems using partial re-configuration for sdr and cr applications. In *2012 Annual IEEE India Conference (INDICON)*, pages 205–209, Dec 2012.

[Kum13a] K. A. Arun Kumar. Fpga implementation of qam modems using pr for reconfigurable wireless radios. In *2013 Annual International Conference on Emerging Research Areas and 2013 International Conference on Microelectronics, Communications and Renewable Energy*, pages 1–6, June 2013.

[Kum14] K. A. A. Kumar. An ofdm transmitter implementation using cordic based partially reconfigurable ifft module. In *2014 3rd International Conference on Eco-friendly Computing and Communication Systems*, pages 266–270, Dec 2014.

[LAL+93] Wazlowski Agarwal Lee, L. Agarwal, T. Lee, A. Smith, E. Lam, P. Athanas, and S. Ghosh. Prism-ii compiler and architecture, 1993.

[Lie95] Jochen Liedtke. *On micro-kernel construction*, volume 29. ACM, 1995.

[LKLJ09] Ming Liu, Wolfgang Kuehn, Zhonghai Lu, and Axel Jantsch. Run-time partial reconfiguration speed investigation and architectural design space exploration. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 498–502. IEEE, 2009.

[Lor15] Jordane Lorandel. *Etude de la consommation énergétique de systèmes de communications numériques sans fil implantés sur cible FPGA*. PhD thesis, 2015. Thèse de doctorat dirigée par Hélard, Maryline Electronique et Télécommunications Rennes, INSA 2015.

[LPC04] R. Le Moigne, O. Pasquier, and J. P. Calvez. A generic RTOS model for real-time systems simulation with systemC. In *Proceedings -Design, Automation and Test in Europe, DATE*, 2004.

[LSD03] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm : exact characterization and average case behavior. 2003.

[Mat15] MathWorks. www.mathworks.com, 2015.

[McD08] Eric J McDonald. Runtime fpga partial reconfiguration. In *Aerospace Conference, 2008 IEEE*, pages 1–7. IEEE, 2008.

[MH16] Christos Masouros and Lajos Hanzo. Dual-Layered MIMO Transmission for Increased Bandwidth Efficiency. *IEEE Transactions on Vehicular Technology*, 2016.

[MP15] C. Moy and J. Palicot. Software radio : a catalyst for wireless innovation. *IEEE Communications Magazine*, 53(9) :24–30, Sep. 2015.

[MVG] Jan Madsen, Kashif Virk, and Mercury Gonzales. Abstract RTOS Modelling for Multiprocessor System-on-Chip*. Technical report.

[NIC15] National-Instruments-Corporation. Labview system design software. www.ni.com/labview/, 2015.

[PG11] François Philipp and Manfred Glesner. Mechanisms and architecture for the dynamic reconfiguration of an advanced wireless sensor node. In *2011 21st International Conference on Field Programmable Logic and Applications*, pages 396–398. IEEE, 2011.

[PHH12a] Dinh Thuy Phan-Huy and Maryline Helard. Receive antenna shift keying for time reversal wireless communications. In *IEEE International Conference on Communications*, 2012.

[PKHC14] JC Prévotet, Y Kokar, M Hélard, and M Crussière. Implementation of a time-reversal miso ofdm test-bed. In *European Workshop on Testbed based wireless research*, 2014.

[PKR+13] Niels Penneman, Danielius Kudinskas, Alasdair Rawsthorne, Bjorn De Sutter, and Koen De Bosschere. Formal virtualization requirements for the arm architecture. *Journal of Systems Architecture*, 59(3) :144–154, 2013.

[PML15] A. Pitarokoilis, S. K. Mohammed, and E. G. Larsson. Uplink performance of time-reversal mrc in massive mimo systems subject to phase noise. *IEEE Transactions on Wireless Communications*, 14(2) :711–723, Feb 2015.

[PXL+13] Linh T.X. Phan, Meng Xu, Jaewoo Lee, Insup Lee, and Oleg Sokolsky. Overhead-aware compositional analysis of real-time systems. In *Real-Time Technology and Applications - Proceedings*, 2013.

[RBAN+11] S.K. Rethinagiri, R. Ben Atitallah, S. Niar, E. Senn, and J. Dekeyser. Hybrid system level power consumption estimation for fpga-based mpsoc. In *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pages 239–246, Oct 2011.

[Roy07] K. Roy. Neural network based macromodels for high level power estimation. In *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, volume 2, pages 159–163, Dec 2007.

[RVdlTR14] Alex Rodriguez, Juan Valverde, Eduardo de la Torre, and Teresa Riesgo. Dynamic management of multikernel multithread accelerators using dynamic partial reconfiguration. In *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, pages 1–7. IEEE, 2014.

[SABN14] S. Sivanantham, R. Adarsh, S. Bhargav, and K. J. Naidu. Partial reconfigurable implementation of ieee802.11g ofdm. *Indian Journal of Science and Technology*, 2014.

[Sha15] Shashi Bhutada. A scalable approach for tlm across systemc and systemverilog, 2015. Mentor Graphics, White Paper.

[spe] Specc system. http://www.cecs.uci.edu/~specc/. Accessed : 2019-02-26.

[SRH+11] M. Streubuhr, R. Rosales, R. Hasholzner, C. Haubelt, and J. Teich. Esl power and performance estimation for heterogeneous mpsocs using systemc. In *Specification and Design Languages (FDL), 2011 Forum on*, pages 1–8, Sept 2011.

[SYM+13] Nikola Serafimovski, Abdelhamid Younis, Raed Mesleh, P. Chambers, Marco Di Renzo, Cheng Xiang Wang, Peter M. Grant, Mark A. Beach, and Harald Haas. Practical implementation of spatial modulation. *IEEE Transactions on Vehicular Technology*, 2013.

[Syn15] Synopsys. System studio, 2015. Website.

[Sysc] Systemc. https://accellera.org/downloads/standards/systemc. Accessed : 2019-02-26.

[thr] https://rtos.com/support/extra-tools/.

[Tri15] Stephen M. Trimberger. Three ages of FPGAs : A retrospective on the first thirty years of FPGA technology. *Proceedings of the IEEE*, 2015.

[VF14] Kizheppatt Vipin and Suhaib A Fahmy. Zycap : Efficient partial reconfiguration management on the xilinx zynq. *Embedded Systems Letters, IEEE*, 6(3) :41–44, 2014.

[VF18] Kizheppatt Vipin and Suhaib A. Fahmy. FPGA Dynamic and Partial Reconfiguration : A Survey of Architectures, Methods, and Applications. *ACM Computing Surveys*, 2018.

[VSP10] Peter Van Stralen and Andy Pimentel. Scenario-based design space exploration of mpsocs. In *Computer Design (ICCD), 2010 IEEE International Conference on*, pages 305–312. IEEE, 2010.

[VSR+13] C. Vennila, K. Suresh, R. Rather, G. Lakshminarayanan, and Seok-Bum Ko. Dynamic partial reconfigurable adaptive transceiver for ofdm based cognitive radio. In *2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4, May 2013.

[VSS+15] D. Viet Vu, O. Sander, T. Sandmann, J. Heidelberger, S. Baehr, and J. Becker. On-demand reconfiguration for coprocessors in mixed criticality multicore systems. In *High Performance Computing Simulation (HPCS), 2015 International Conference on*, pages 569–576, July 2015.

[WBP13] Wei Wang, Miodrag Bolic, and Jonathan Parri. pvfpga : accessing an fpga-based hardware accelerator in a paravirtualized environment. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2013 International Conference on*, pages 1–9. IEEE, 2013.

[Wir95] M. J. Wirthlin. A dynamic instruction set computer. In *Proceedings of the IEEE Symposium on FPGA's for Custom Computing Machines*, FCCM '95, pages 99–, Washington, DC, USA, 1995. IEEE Computer Society.

[WW09] Lie Wang and Feng Yan Wu. Dynamic partial reconfiguration on cognitive radio platform. In *Proceedings - 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009*, 2009.

[XBG+10] Yang Xu, Felix Bruns, Elizabeth Gonzalez, Shadi Traboulsi, Klaus Mott, and Attila Bilgic. Performance evaluation of para-virtualization on modern mobile phone platform. In *Proceedings of the International Conference on Computer, Electrical, and Systems Science, and Engineering*, 2010.

[Xia16] Tian Xia. *Research on virtualisation technlogy for real-time reconfigurable systems*. Theses, INSA de Rennes, July 2016.

[Xil12a] Xilinx Inc. System generator for dsp, Oct. 16 2012. User Guide UG640 (v 14.3).

[Xil12c] Xilinx Incorporation. Xpower estimator user guide, Jan 2012. User Guide UG440 (v13.4).

[Xil14c] Xilinx. *Zynq-7000 All Programmable SoC Technical Reference Manual (UG585)*, 2014.

[Yu10] Ke Yu. *Real-Time Operating System Modelling and Simulation Using SystemC*. PhD thesis, University of York, 2010.

[YY14] Seehwan Yoo and Chuck Yoo. Real-time scheduling for xen-arm virtual machines. *IEEE Transactions on Mobile Computing*, 13(8) :1857–1867, 2014.

[ZMG09] Henning Zabel, Wolfgang Müller, and Andreas Gerstlauer. Accurate RTOS modeling and analysis with SystemC. In *Hardware-dependent Software : Principles and Practice*. 2009.

[ZPR+13] Nikolaos Zompakis, Antonis Papanikolaou, Praveen Raghavan, Dimitrios Soudris, and Francky Catthoor. Enabling efficient system configurations for dynamic wireless applications using system scenarios. *International journal of wireless information networks*, 20(2) :140–156, 2013.