



A Flexible Modeling and Simulation Framework for Design Space Exploration

Camille Jalier and Didier Lattard
CEA-LETI, MINATEC
38054 Grenoble, FRANCE
{firstname.lastname}@cea.fr

Gilles Sassatelli
LIRMM
161 rue Ada, Montpellier, FRANCE
sassatelli@lirmm.fr

Abstract—Applications like 4G baseband modem require single-chip implementation to meet the integration and the power consumption requirements. These applications involve a high computation performance with real-time constraints, low power consumption and low cost. The concept of MPSoC is well suited to this problem. It makes it possible to adjust the architecture, by allocating the computational power where it is needed to fit the application needs. This often implies that the software has to be developed at the same time the platform is refined. Algorithm designers need accurate performance estimation to guide their decisions and system architects need to provide a design with enough calculation capacity and flexibility.

Based on the methodology used for the design of the 4G FAUST chipset, this paper presents a modeling and simulation framework for Design Space Exploration (DSE) which enables a rapid evaluation of the application-to-platform adequation. The key element of this work is a simple and flexible way of modeling application and architecture. Our SystemC-based simulation environment can support a broad range of architecture components (ASIC, DSP, NoC, bus, shared or distributed memory, ...) and application features (control flow, data exchange, interrupts, data-dependent processing, dynamic reconfiguration). Application and architecture models are separated to allow independent design space exploration. The simulation basically executes the algorithms on the architecture and monitors dynamic behavior such as communication transfers, resource conflicts, starvation, dynamic reconfiguration, etc.

I. INTRODUCTION

The increasing complexity of telecommunication algorithms in mobile terminals, requires high computational performance and low power consumption. The algorithms proposed for 4G applications are adaptive so the System On Chip (SoC) requires enough flexibility/reconfigurability to support multiple modulation schemes, Multiple Input Multiple Output (MIMO) transmissions, etc. The implementation of such algorithms on the FAUST platform [1] has proved the efficiency of MPSoC to provide an attractive solution. This development has stressed the need for exploration tools to help the designer in the platform implementation choices:

- Processing resources: ASIC, ASIP, DSP, RISC
- Communication media: bus, point-to-point, Network-on-Chip (NoC)
- Memory topology: shared, distributed, FIFO
- Application: task partitioning, communication protocol, scheduling, mapping

The implementation of a baseband application onto a reconfigurable Multiple Processor SoC (MPSoC) adds stages in the design flow:

- Mapping and scheduling of the algorithm functionality onto each processing resources to exploit parallelism.
- Optimization of data transfer and scheduling to reduce time-overhead.
- Optimization of resource allocation to minimize the chip.
- Performance evaluation of the application on the architecture to guarantee the real-time constraints.

In order to address these issues, we propose a SystemC-based simulation framework. This framework incorporates both application and architecture aspects of the MPSoC to enable a behavioral approach. The performance is extracted through a simulation of the application behavior onto the platform. Because the application and the architecture are abstracted, it is possible to significantly speed up and ease the exploration process, to perform dynamic performance analysis and to quickly change parameters without rewriting models. In this context, this framework is useful to perform an early Design Space Exploration (DSE) and/or to provide an early execution platform for the software development. This tool might allow a faster verification of the second generation FAUST platform. The rest of this paper is organized as follows. The importance of the DSE in the design flow and the related works are discussed in section II. Section III describes the application and architecture abstraction models. Section IV describes the DSE environment.

II. SYSTEM LEVEL DESIGN

A. Framework Overview

In baseband modems, the algorithm choices widely influence the MPSoC design. In the traditional top down approach, the designer starts with an informal specification and develops a reference model of the telecom standard using a high-level language such as Matlab or C++. At this functional level of abstraction, the algorithm is developed independently from the architecture. This modeling approach is used by software designers to study different algorithms under complexity and precision aspects [2].

As shown in Fig. 1, in the next step of the design flow, this description is refined into an abstract platform. The platform

is composed of modules, each of them executes a part of the application. They represent future hardware or software components (RISC, DSP, ASIC, etc). The transition between a functional view to a platform view is complex [3]. This implies deciding Hardware/Software partitioning, defining the architecture components and finally perform the algorithm mapping. Each architectural decision affects the global performance and the application behavior considering processing time, memory latency, data-transfer latency, possible parallel processing, resource sharing, etc. Flexible frameworks, that can model easily a broad range of platform and quickly estimate resulting performance, are required to perform an efficient DSE and converge to a satisfying solution. The exploration process at high level allows very efficient system optimizations. Our modeling and simulation framework simplifies and accelerates this process and is well suited for high complexity systems such as complex 4G baseband chipsets.

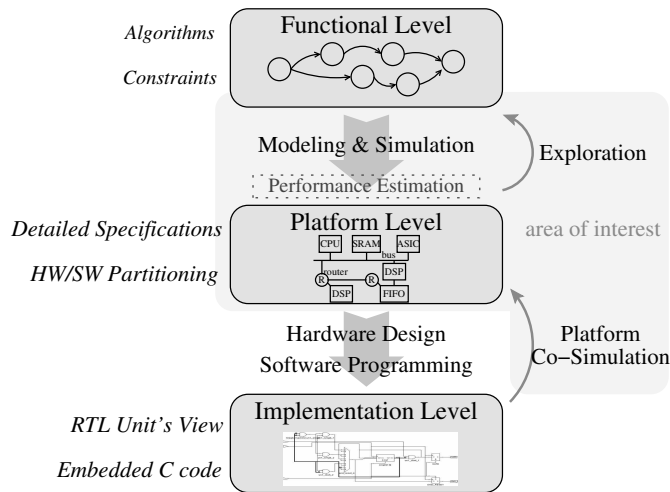


Fig. 1. Simplified Design Flow

Once both algorithm and platform specifications are decided, every component is refined into either Register Transfer Level (RTL) model or embedded C code. The used approach is incremental for verification purposes: every low-level cycle-accurate RTL hardware component is simulated within the platform model for ensuring proper behavior.

B. Related work

Several methodologies and tools have been introduced to help the designer to model and simulate an application onto an architecture. The most relevant are mentioned in this paper.

Syndex [4] is a system level CAD software tool in which applications and architectures are modeled with independent tasks. In this approach, the application model is well separated from the architecture model to ensure an easy exploration of each aspect. However the application graph is limited to a *Data-dependence Graph*, in this case it is impossible to capture data-dependent processing or synchronization by interrupts.

ArchAn [5] is an architectural simulation environment in which applications are modeled in a task modeling language

and the abstract architecture is modeled at the cycle-accurate level. In this approach, the task modeling language includes details of the underlying architecture. Algorithms and architectural details are combined, the exploration becomes more difficult.

Artemis [6] is a modeling and simulation environment to explore the design space of heterogeneous embedded-systems architectures. This tool allows to model the architecture at multiple level of abstraction using generic building blocks provided in a library. Regarding application modeling, Artemis uses the Kahn Process Network (KPN) computational model. The KPN semantic forbids the use of interrupts, some dynamic behavior, such as run-time mapping, cannot be simply modeled.

III. APPLICATIONS AND ARCHITECTURES ABSTRACTION MODELS

A. Application abstraction models

In this work, the used approach considers the telecom algorithm (OFDM demodulation, MIMO decoding, channel decoding, etc) and also all the additional mechanisms (synchronization between resources, requests for memory space, etc) as part of the application. All these mechanisms are not defined in the telecom standard but they are mandatory to complete the platform model and they significantly influence the execution behavior, they are time-consuming. One of the motivations of this work is to keep the architecture and the application well separated so that exploration is facilitated; i.e. any functionality can be mapped to either in software (on a RISC) or in hardware (ASIC) without any expensive rewriting phase.

An application is modeled with tasks and buffers. The functionality is divided into a set of tasks. Each task may or may not contain the C++ code of the algorithm, but some specific annotations are required in order to indicate:

- communications with external tasks or buffers (data or control information)
- complexity of the data processing
- control flow (loop, conditions, ...)

These three types of annotation capture the interactions between the application and the system [7]. Data transfers between tasks are memorized in buffers. All communications can be represented as edges. The possibility for tasks to exchange some control information offers a great flexibility. The processing algorithm can be conditioned by a control event. Fig. 2 illustrates the proposed approach on a simple example that exhibits both data and control interactions among tasks. Five steps are used for describing task 3 behavior. The first one (A) synchronizes the execution on a control message *START* sent by the task 5. Steps (B), (D) and (E) respectively model the reading, processing and writing operations of the task. Step (C) is the C++ code of the interleaving algorithm used in 4G transmissions. This piece of code is not time-consuming for simulation though, it can be omitted. Finally, the complexity of the algorithm is modeled in the step (D).

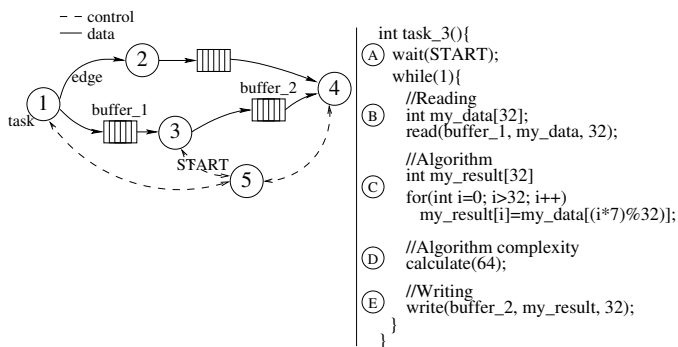


Fig. 2. Example of an application model

B. Architecture abstraction models

The platform architecture, which is the physical support for execution, is modeled with the same high-level abstraction approach as the application. To meet the well-known power, performance and cost constraints, architectures are often heterogeneous with multiple differentiated processors, complex memory hierarchy and hardware accelerators [8]. Modeling that kind of SoC involves flexible and high-level models. In the literature, two approaches are often used. Either models, like Random Access Machine [9], are employed to evaluate computing complexity but they intentionally neglect the underlying architecture details. Or low level models, like Instruction Set Simulator (ISS) or HDL descriptions, are employed but they do not allow rapid design space exploration as rewriting code is often required, they are too over-detailed.

Our approach is based on abstract models that can describe very complex and different MPSoC implementations without rewriting code. For enabling this, a set of abstract entities that can cover current platform such as FAUST and future HW/SW MPSoC are required. In the previous section III-A, the application is modeled with tasks, buffers and edges. The abstract models for the architecture are directly inspired from the application view. Three different types of abstract entities are used to describe the architecture:

- a processor model that schedules several tasks (calculate requests)
- a memory model that manages several buffers (read, write requests)
- a communication media model that manages several simultaneous communications (edges)

The abstract model for each entity is written in SystemC and is composed of a resource manager (scheduler [10]) to allow the concurrent execution of multiple application elements and a communication manager which resolves exchanges between entities. The platform architecture is specified in an XML file that describes the numbers of entities, their types and the interconnections. Each instantiated entity has its own parameters that can be tuned to predict the high level characteristics of the future physical module.

As shown in the Fig. 3, the Processor model is an abstract model used for modeling any type of processing element (DSP,

CPU, ASIC). At this stage, the partitioning between hardware and software is not defined and the architecture description is independent of the application model.

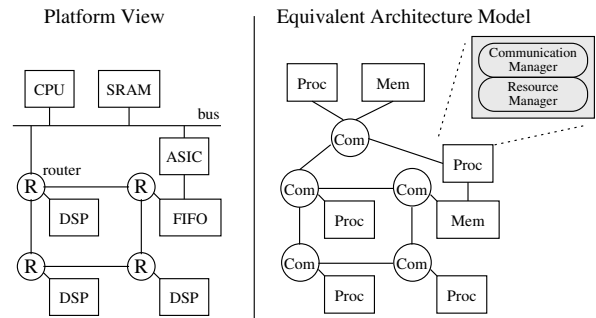


Fig. 3. A model based on Processor (Proc), Memory (Mem) and Communication (Com) entities.

C. Mapping

We have separately modeled the functionality and the architecture with independent or orthogonal models. To complete the platform description, we consider the spatial and temporal mapping. Each part of the application such as tasks (resp. buffers and edges) is placed on a Processor (resp. Memory and Communication) entity. When several tasks (resp. buffers and edges) are mapped onto a Processor (resp. Memory and Communication) module, the arbitration is done by the resource manager considering a scheduling policy. The mapping is described in an XML file to be easily changed. During this phase, the designer can efficiently distribute the application to exploit the parallelism or to share resources.

IV. DSE FRAMEWORK

Our framework follows the well-know y-chart principle [11], where a set of application models is merged with a set of architecture models in a dedicating mapping step. The platform is modeled in an orthogonal way. This mechanism is the key to enhance the flexibility during design space exploration. In our approach, we can independently explore:

- application: task partitioning, communication protocol (message passing, shared memory, etc.), synchronization model (data, interrupt, etc.), etc.
- architecture: performance for computing resources, communication topology, memory topology, number of resources, etc.
- mapping: resource allocation (to optimize performance, chip area, etc.), scheduling policy, etc.

The exploration process starts by modeling the application with tasks, buffers and edges. Then a first platform architecture is modeled with Processor models, Memory models and Communication Media models. Based on the application graph and the architecture view, the mapping consists of allocating resources for each part of the application. Using our models, the framework automatically generates a SystemC platform including the application, the architecture and the mapping.

Based on the SystemC simulation engine (Fig. 4), the application is executed on the architecture and the execution behavior is captured. All the relevant characteristics for performance estimation can then be easily extracted: application throughput, maximum latency, utilization rate of resources, transient effects, etc.

According to the estimated performance, the designer can change the architecture and/or the algorithm architect can modify the application. This exploration process, to converge to a suitable solution, is incremental but do not need to rewrite code, only to modify XML files and/or change parameters. The designer is responsible for setting these parameters with relevant values according to his background and the technology. The predicted performance results are tightly linked to the parameters of the architectures entities.

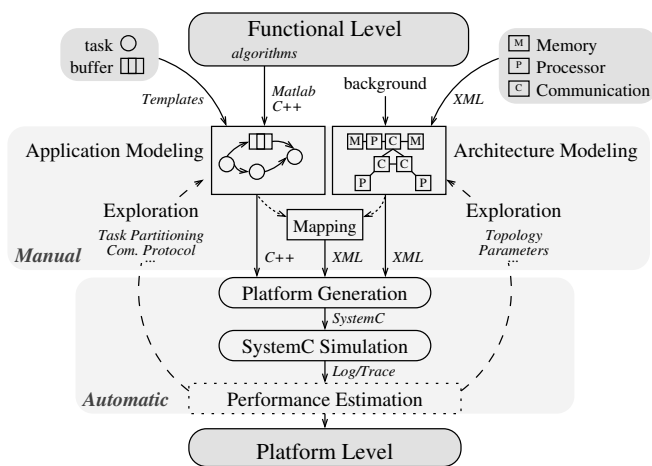


Fig. 4. A Detailed View of our Exploration Process

As all models are unified in the same framework, the algorithm architect and the designer can interact easily and work independently. In our approach, the modeling phase and the exploration is done manually by the designer relying on his skills and experience. This framework nevertheless offers an accelerated exploration process based on:

- Three parameterizable models (XML & SystemC) for architecture modeling (no code rewriting)
- XML-based description files to quickly parameterize and interconnect modules
- Automatic generation of a platform SystemC View
- Extraction of execution parameters on log files

At the end of the exploration, the designer has refined: - the application into an embedded platform software regarding communication protocols, synchronizations, etc. - the architecture together with the topology/communication architecture and the associated parameters. - the mapping.

The hardware/software partitioning is performed according to the calculation capacity and flexibility of each unit. The designer can then easily provide detailed specifications for deriving a hardware or a software implementation. The programmer can work on an embedded and optimized code

according to the target CPU and the designer can describe each hardware unit at Register Transfer Level.

As we have chosen SystemC, we can use Co-Simulation tools to plug each hardware unit, described in HDL languages, in place of our high-level model to perform a platform simulation.

V. CONCLUSION

We propose a system level simulation framework for early investigation of MPSoC platform architectures considering a specific application. The major contributions of this paper are firstly a set of flexible high level models based on XML/SystemC descriptions and secondly an orthogonal way of modeling the application, the architecture and the mapping based on a separation between the functionality and the execution support. In this context, the algorithm is modeled in an embedded manner with explicit communication protocol, processing complexity, interrupts, etc. Here the use of high-level parameterizable models enables a rapid exploration process without rewriting code. With our modeling framework, dynamic or data-dependent behavior can be easily captured.

The major advantage of our modeling approach is the combination of a quick exploration flow and an extended modeling capacity.

Based on our experience on the FAUST chip, our future work will focus on the verification of the second generation platform. This design implements more flexible components to guarantee an efficient reconfiguration. On the longer term, this framework will serve as a basis to explore homogeneous architectures in the perspective of a completely reconfigurable telecom system.

REFERENCES

- [1] Y. Durand. FAUST: on-chip distributed SoC architecture for 4G base-band modem chipset. In *Proc. of the IP/SOC'05 conference*, 2005
- [2] T. Kempf, M. Doerper. A modular Simulation Framework for Spatial and Temporal Task Mapping onto Multi-Processor SoC Platforms. In *Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)*, 2005
- [3] P. Magarshack, P. Paulin. System-on-chip Beyond the Nanometer Wall. In *Proc. of the Design Automation Conference (DAC)*, 2003
- [4] T. Grandpierre, Y. Sorel. From algorithm and architecture specification to automatic generation of distributed real-time executives: a seamless flow of graphs transformations. In *Proc. of the First ACM and IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'03)*, June 2003
- [5] A. Chatelain, Y. Mathys. Verification Strategy for Integration 3G Base-band SoC. In *Proc. of the Design Automation Conference (DAC)*, 2003
- [6] A.D. Pimentel, L.O. Hertzbetger. Exploring embedded-systems architectures with Artemis. In *Computer*, vol. 34, pages 57 - 63, 2001
- [7] M. Waseem, L. Apvrille. Abstract Application Modeling for System Design Space Exploration. In *Proc. of the 9th EUROMICRO Conference on Digital System Design (DSD'06)*, 2006
- [8] D. Lattard, E. Beigne. A Reconfigurable Baseband Platform Based on an Asynchronous Network-on-Chip. In *Journal of Solid-State Circuits (JSSC)*, vol. 43, pages 223 - 235, Jan. 2008
- [9] A. Tiskin. The bulk synchronous parallel random access machine. In *Proc. of EURO-PAR'96*, vol. 2, pages 327-338, August 1996
- [10] J.M. Paul, A. Bobrek. Schedulers as Model-Based Design Elements in Programmable Heterogeneous Multiprocessors. In *Proc. of the Design Automation Conference (DAC)*, 2003
- [11] P. Lieverse, P. van der Wolf. A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems. In *Proc. of the IEEE Int. Workshop on Signal Processing Systems (SIPS)*, 1997