Automatic generation of modifiable platform models in SystemC for Automatic System Architecture Exploration

Héctor Posadas, Gerardo de Miguel & Eugenio Villar

Microelectronics Engineering Group, University of Cantabria

ETSIIT Av. Los Castros 39005 Santander

{posadash, gdmiguel, villar}@teisa.unican.es

***Abstract*[1]—Early Design Space Exploration (DSE) is crucial to achieve optimal designs in large, configurable embedded systems. In platform-based design, the exploration process must cover two areas: selecting the base platform and customizing the configuration parameters. Although exploring the optimal parameters' configuration is a well-known topic, there is a lack of works exploring both areas together. There are no mechanisms to describe all the platform possibilities or simulation infrastructures capable of supporting automatic DSE of both areas. This work proposes a XML-based methodology oriented to describe and automatically create system models of fully configurable systems. The XML descriptions are adequate to be handled by common multi-objective exploration tools. The simulation infrastructure developed automatically creates the models of the different possible architectures and obtains their performance features to perform the exploration. To allow efficient architecture exploration, the high-level system model is automatically built at run-time, avoiding recompiling times.**

***Keywords*—Multiprocessor System-on-chip, system modeling, architecture exploration, design space exploration, native co-simulation**

I. INTRODUCTION

Nowadays, complex electronic systems combine several processors and specific HW components. As well as providing large computational power, this growth of complexity also implies multiple design possibilities. To cope with this complexity, platform-based design is a widely used solution.

Any given design space has a limited number of good solutions to its basic problems [1]. Platform-based design techniques apply this idea by defining platforms that captures the good solutions to the important design challenges in the different design spaces. These platforms define the number and kind of system components and the system architecture.

Platforms are composed of configurable components. Thus, the platforms are customizable through a set of configuration parameters. Each platform instance derived from the customizable platform maintains enough flexibility to support an application space that guarantees the production volumes necessary for economically viable manufacturing [1].

When developing a specific design, platform-based design requires defining first the most suitable platform among the set of platforms oriented to this design space. Then, the best configuration of their components has to be identified.

To evaluate the different possibilities and to select the best configuration, Design Space Exploration (DSE) techniques have been proposed. DSE strategies are commonly based on leveraging Design of Experiments (DoE) and Response Surface Modeling (RSM) techniques. The DoE consists in deciding the system simulations, called experiments, required to obtain the required information for RSM.

Current DSE solutions are focused on exploring the best configuration once the platform has been selected. However, the exploration required to select the best platform architecture is not covered. Exploring the best platform architecture for a design implies adding and removing components from the system and trying different communication infrastructures. Current simulation tools are prepared to allow modifying certain values of the system components, such as size, or frequency [2,3]. These parameters only require modifying a value in the system model, not the model itself.

Two issues must be considered to allow exploring which is the best system architecture among the available platforms for a certain design. First, it is required a method capable of describing the possible platforms and all its configurable options. Second, it is required a simulator capable of creating the system models from these descriptions.

When the system architecture is modified, a new system model has to be created. Current tools and methods for system description and simulation are not prepared for automatic system model creation. System description languages, such as IP-XACT [4], provide facilities to describe configurable components. However, the system architectures described with these languages are fixed.

1 This work has been supported by the Spanish MICyT and the EC through MULTICUBE FP7-216693 and the TEC2008-04107 projects.

In addition, simulation and modeling tools require fixed system models. They do not have enough intelligence to add, connect or remove components automatically. Creation of new models of these modifiable platforms requires interaction with the designer, which is not a feasible solution for automatic, efficient DSE.

This paper provides a solution for both problems. A set of XML rules to allow platform architectures exploration is presented. Furthermore, it is provided a simulation engine capable of automatically building the model from the XML description considering the selected parameters. The XML rules have been developed in a generic way and can be added to any other XML-based system description language.

To perform efficient DSE, the execution of the simulations must be as fast as possible. To achieve that goal, the presented infrastructure automatically builds the system models at run-time, avoiding recompiling times.

Once obtained the performance results of each possible configuration from the simulator, the exploration can be tackled using common multi-objective exploration solutions [12]. Thus, neither the DoE nor the RSM techniques will be considered in this paper. It is limited to platform description and automatic system model creation.

The DSE tools define the proper DoE to explore the architecture possibilities from the XML. The simulation engine builds the platform model and obtains the performance results for each experiment. Finally the DSE tool can apply RSM techniques to obtain the optimal configuration.

This paper is structured as follows. First, the related work is reviewed. Then, the complete DSE execution flow proposed is presented. Next, the proposed solutions to describe modifiable systems in XML are shown. Then, the automatic model generation is explained. Finally results and conclusions are inferred.

II. State of the Art

Design Space Exploration is an important research area. Several tools capable of defining adequate DoEs and applying RSMs have been proposed [11, 12, 13]. These solutions allow defining multi-objective explorations, which can be used to obtain the optimal platform configurations. Nevertheless, these tools require simulation engines working together to perform the exploration. In fact, the simulators are the bottleneck for the proposed platform exploration.

Some works have been focused on automating the exploration of component interconnection [7, 8, 10]. However, these works do not provide complete design models.

Automatic generation of system models oriented to specific target architectures has been proposed [14, 15]. However, as specific solutions they cannot be used to explore which is the best platform for a specific application.

To provide more generic techniques, transaction level modeling techniques based on system-level design languages like SystemC have been proposed [6, 9, 16].

In [5] a TLM framework for automatic system model generation is proposed. The framework receives a fixed system description and generates the executable system model.

Some commercial tools [2, 3] can model designs at this TLM level. The schematic entry tools simply provide a graphical interface for plugging existing database models together. These models are described and connected at the transaction-level. They also provide shell interfaces which allow modifying the characteristics of the system components. However, the system architecture is fixed and cannot be modified.

An alternative solution to schematic entries for system description and model generation is using XML based descriptions. IP-XACT [4] standard describes an XML schema for meta-data documenting Intellectual Property (IP) used in the development, implementation and verification of electronic systems. This schema provides a standard method to document IP that is compatible with automated integration techniques. Several tools have been developed to support that integration [17, 18]. The resulting models can only configure certain parameters on the system components. However, modifiable platforms cannot be described through IP-XACT and modeled with these tools. Thus, the exploration of the best platform architecture cannot be performed with these tools.

This work presents a solution to describe modifiable architectures and automatically generate the corresponding system models. These models can be configured by modifying the parameters of the system components and also modifying the system architecture itself. This modeling capability will allow the DSE tools not only to find the optimal tunning of the system components, but also to optimize the system itself.

System descriptions will be performed in a simple XML format, although the proposed solutions can be easily adapted to other XML descriptions.

The modeling tool has been achieved starting from the work proposed in [16]. The XML extension allows automatic system model creation and support modifiable descriptions. To dynamically create the models, some ideas of [5] have been applied to ensure correct automatic interconnection of the system components.

III. DSE Proposed Flow

SCoPE [16] is a Simulation framework based on SystemC and oriented to system modeling. It is based on approximately timed system simulations. As a SystemC extension library, the tool was designed to receive the system descriptions as SystemC code. However, to allow automatic system model generation at run-time, this approach must be changed.

The tool has been extended to accept the system description in a friendly format and automatically generate the system model. Two input XML files and a XML output file have been defined to provide the system description and return the simulation results. Figure 1 shows how these files are used to configure the simulator and to connect it with an external DSE tool, as those presented in the state of the art.

The two input files are the following:

– a file describing the system and its configuration options, called System Description file

– a file of pairs identifier-value, fixing the selected configuration for each experiment, called System Configuration file

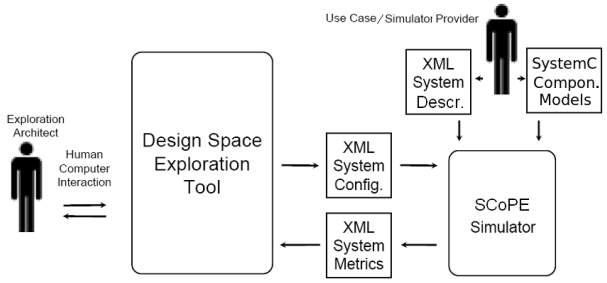

FIGURE 1: SIMULATOR AND DSE TOOL INTERCONNECTION

The external DSE explorer only indicates the simulator about the configuration to be analyzed each time by generating the corresponding System Configuration file. The simulation tool interprets the file, builds the system model and performs the simulation. This means that no user interaction or model recompiling is required once the exploration process starts.

The tool generates an output file, when the simulation finishes. This file contains the values of the metrics obtained during the simulation. The information returned is used by the DSE explorer to perform the search of the best solutions applying the RSM techniques.

IV. CONFIGURABLE XML SYSTEM DESCRIPTIONS

The XML System Description file includes information about the HW components, the HW architecture, and the SW tasks. A simple XML language has been developed to easily explain the proposed way to describe modifiable platforms. The language guarantees fast model creation and efficient system simulation.

A really simple example of an XML description using this language is shown in figure 2. To keep it simple, no configurable options has been added. The example proposes a system with a processor and a memory connected to a bus. A "hello world" application has been selected to execute in the processor.

```
<HW_Platform>
  <HW_Components>
    <HW_Component category="bus" name="AMBA" frequency="200MHz" />
    <HW_Component category="processor" name="arm926t" frequency="200MHz"/>
    <HW_Component category="memory" name="Memory"
      mem_size="500MB"  frequency="200MHz"  mem_type="RAM" />
  </HW_Components>
  <HW_Architecture>
    <HW_Instance component="AMBA" name="my_bus" >
      <HW_Instance component="arm926t" name="my_proc" />
       <HW_Instance component="Memory" name="my_memory"
          start_addr="0x80000000" />
    </HW_Instance>
  </HW_Architecture>
</HW_Platform>
<Application>
    <Functionality>
      <Exec_Component name="hello" category="SW" function="hello_main" />
    </Functionality>
    <Allocation>
      <Exec_Instance name="Hello_world" component="hello"
          processor="my_proc"/>
    </Allocation>
</Application>
```

FIGURE 2: SIMPLE XML SYSTEM DESCRIPTION

To describe a system with several platform architecture options and its configuration possibilities three XML mechanisms are provided. All three methods can be used simultaneously to describe highly configurable systems.

A. Configuration of the System Components

The first configuration possibility is to tune the characteristics of the system components. The values of all parameters in the XML file can be replaced by identifiers when the parameter is a configurable one. For example, in Figure 3, the bus frequency that was indicated in Figure 2 has been replaced by the identifier "FREQ".

To select a configuration, the values of all identifiers must be assigned in the System Configuration File. Thus, to perform different simulations it is only required to modify the value-identifier pairs in the System Configuration file (figure 1).

Applying this solution, the simulation of each experiment required by the DoE is performed by substituting the identifiers of the configurable parameters by the selected values and creating the corresponding system model.

```
<HW_Platform>
  <HW_Components>
    <HW_Component category="bus" name="AMBA" frequency="FREQ" />
      ...
  </HW_Components>
    ...
</HW_Platform>
```

FIGURE 3: DEFINING A CONFIGURABLE PARAMETER

B. Replication of system components

The second possibility is to indicate the number of times a system component is replicated. To do so, a new XML "repeat" clause is provided. This clause defines the number of times the element is repeated, an index identifier and the initial index value. Figure 4 corresponds to an extension of the system description in Figure 2 considering that the number of CPUs in the system can be set by the "CPUS" parameter. This parameter must be assigned in the System Configuration file.

```
<HW_Platform>
  <HW_Components>
    ...
  </HW_Components>
  <HW_Architecture>
    <HW_Instance component="AMBA" name="my_bus" >
                <repeat numer="CPUS" index="i" init="1">
          <HW_Instance component="arm926t" name="my_proc[%i]" />
                </repeat>
        <HW_Instance component="Memory" name="my_memory"/>
    </HW_Instance>
  </HW_Architecture>
</HW_Platform>
<Application>
    <Functionality>
        <Exec_Component name="hello" category="SW" function="hello_main" />
    </Functionality>
    <Allocation>
                <repeat numer="CPUS" index="j" init="1">
          <Exec_Instance name="task[%i]" component="hello"
                processor="my_proc[%j]"/>
                </repeat>
    </Allocation>
</Application>
```

FIGURE 4: XML DESCRIPTION WITH MULTIPLE PROCESSORS

The "Repeat" clause can be used to replicate both

single components and groups of components, copying complete parts of the system architecture. If the value is set to '0', the element is not placed in the system. This option is used to add or delete different components within the system, including modifying SW components, HW components and the communication infrastructures. As a consequence, different platform architectures can be described.

C. Selecting complete configurations

The third solution is to define several complete configurations and select one on each simulation. For example, in figure 5, two different HW architectures are described ("arch1" and "arch2"). The one selected for each simulation is defined in the "Implementation" clause. In this example, the architecture selected depends on the "ARCH" identifier. Its value must be set in the System Configuration file to "arch1" or to "arch2".

The system description mechanism allows dividing the system description in parts and exploring different combinations. Multiple HW component lists, HW architectures or SW allocations can be described to be explored by the DSE tool.

```
<HW_Platform>
  <HW_Components>
    ...
  </HW_Components>
  <HW_Architecture name="arch1">
    <HW_Instance component="AMBA" name="my_bus" />
      ...
  </HW_Architecture>
  <HW_Architecture name="arch2">
    <HW_Instance component="NoC" name="my_noc" />
      ...
  </HW_Architecture>
</HW_Platform>
<Application>
      ...
</Application>
 <Simulation>
      < Implementation HW_Architecture="ARCH" / >
</Simulation>
```

FIGURE 5: XML DESCRIPTION DIFFERENT HW ARCHITECTURES

V. BUILDING MODIFIABLE SYSTEM MODELS

To simulate each one of the configurations selected by the DSE tool, different system models must be created. The model descriptions can be obtained applying the values of the XML System Configuration file to the XML System Description.

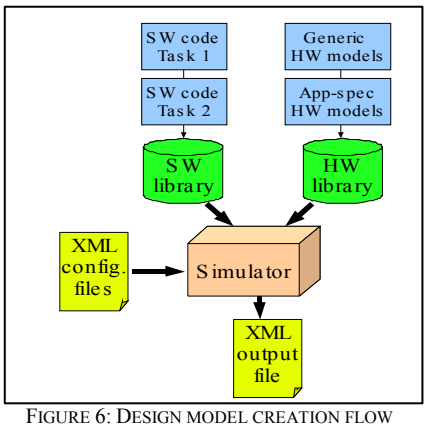

FIGURE 6: DESIGN MODEL CREATION FLOW

The simulation infrastructure integrates a library of generic HW components, which can be easily extended with SystemC models of the application-specific components. The SW components must be compiled for native execution and integrated in the SW components library. The SW infrastructure, as the OS model is also provided by the simulation engine. The tool provides models for POSIX and uCOS OS. Other OS models can also be added by providing the required OS models. The OS model is in charge of starting and managing the task execution. In case of SMP systems, it is also in charge of deciding the punctual allocation of each task.

To generate the system model, the simulator dynamically creates instances of the required models and builds a platform model by interconnecting them as specified in the XML files.

To ensure an efficient DSE process, not only the dynamic system creation is required. As it can be required to optimize a large number of parameters for the system, the associated design space can be really large. As a consequence, a large DoE is obtained, thus requiring multiple simulations. The simulations themselves must be as fast as possible. To achieve that, the loosely-timed abstraction level has been selected for the system models. This abstraction level is the most adequate one to ensure rapid exploration of the design space.

Nevertheless, the proposed solution can also be easily applicable to other abstraction levels, as cycle accurate level.

A. HW components instantiation

The simulation engine contains models of some of the most usual system components. Models of processors, memories, bus, DMAs, etc. can be selected to create the system model. The first step is thus to create instances of all the components indicated in the XML files.

The component models are configurable ones. These models contain a long range of configuration details to describe their functionality. Response times, delays, area, mean power consumption, power for access, frequency, memory size, IRQ or associated memory map addresses are some of the configuration possibilities.

To instantiate a component in the system model, all this parameters must be set. Parameter values are obtained from the values indicated in the corresponding "HW_Instance" clause of the XML System Description file (Figure 2). These parameters can be either defined as explicit values ("200MHz", "500MB") or identified as configurable values.

To set the parameters which are not specified in the "HW_Instance" clause, the simulator checks the "HW_Component" clause corresponding to the type of component instantiated (Figure 2). Similar to the previous ones, these parameters can be fixed or configurable ones.

Finally, if any of the parameters has not been fixed, the default values for this component model are applied.

The components are also prepared to store all the performance information required to generate the output reports that the DSE tool analyze in order to

select the optimal configurations.

B. HW components interconnection

The instantiated components must be interconnected to create an executable system model. To simplify the interconnection work, TLM techniques have been used. TLM accurately describes the system communication architecture down to the level of individual read and write transactions. The use of transfers instead of signals, reduce the complexity when automatically interconnecting the system components.

To allow easy automatic interconnection of the system components, all component models have been created using an generic template provided by the simulation engine. This template is oriented to ensure interface compatibility without limiting the component communication requirements. Ensuring that both ends of each interconnect have compatible interfaces, the automatic connection is possible.

To complete the HW platform generation, it is required the creation of the memory maps and ensure correct interrupt delivering.

Each time a component is connected to a bus, its associated memory area is integrated in the memory map, ensuring that it has not been used before. The solution is similar for networking communication. Network models require the node identifier in order to configure the internal routing protocols properly.

Finally, an interrupt controller is in charge of managing the interrupt delivery. This is especially important in multiprocessor systems, where not all interruptions must be managed in the same way.

C. SW components instantiation

OS models and SW tasks are finally added to the simulation as described in the XML files.

An OS is mapped to a processor or group of processors (for SMP systems). SW tasks are associated to an operating system, and thus mapped to the system processors where the OS runs.

To integrate the SW tasks in the simulation, the SW code has to be provided. The code is annotated and compiled building a library which is added to the simulation. The annotated code provides the performance information required by an external DSE tool to perform the exploration.

SW tasks are defined in the XML System Description file indicating the name of the main function of the task. To load the main function, the dynamic library management is used, by calling the dlopen and dlsym function. Additionally, other parameters like the OS where it will run, the priority, the policy and the main function arguments can be defined. All these elements can be parameterized, so the DSE flow can explore the best configuration for the SW tasks.

VI. EXAMPLE & RESULTS

To verify the validity and efficiency of the proposed system description method and its integration in an external DSE tool a large example has been developed.

A mpeg-4 encoder example has been evaluated in a modifiable multiprocessor platform. The platform contains a bus, a memory and a variable number of processors, from 2 to 8. As SW code, six different parallelization of the mpeg system have been obtained from the Atomium tool suite from IMEC [19].

In the example, the processors frequency, cache size, number of processors and code parallelization have been explored. Just to simplify the example, the number of processors and the parallelization have been explored together, forcing the number of processors to be equal to the number of threads of the selected parallelization. This is not a limitation of the tool but an engineering decision.

Processors frequency has been set in the range of 40-200 MHz, and instruction cache sizes from 4 to 32 kB.

To perform the exploration, the DSE tool modeFRONTIER has been used. The DSE tool created the required DoE and the presented simulation engine has been used to estimate the performance of each selected configuration. The optimal platform configuration has been decided in terms of application SW latency and power consumption. Furthermore, other metrics as executed instructions has been analyzed.

```
<?xml version="1.0" encoding="UTF-8"?>
<Description xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" name="mpeg">
  <HW_Platform>
    <HW_Components>
      <HW_Component category="bus" name="local_bus" frequency="200" />
      <HW_Component category="icache" name="icache"
      mem_size="__icache_size" static_power="3" read_energy="40"/>
        <HW_Component category="processor" name="arm926t"
        frequency="200" proc_type="arm926t" static_power="2" />
        <HW_Component category="memory" name="Mem" mem_size="512M"
        frequency="200" latency="10" mem_type="RAM" />
    </HW_Components>
    <HW_Architecture>
      <HW_Instance component="local_bus" name="bus" >
        <Repeat number="__num_cpus" index="i">
          <HW_Instance component="arm926t" name="Processor_%i"
          frequency="__freq">
            <HW_Instance component="icache" name="icache_%i" />
          </HW_Instance>
        </Repeat>
        <HW_Instance component="Memory" name="main_memory"
          start_addr="0x80000000" />
      </HW_Instance>
    </HW_Architecture>
    <Computing_groups>
      <Computing_group name="node1" >
        <Repeat number="__num_cpus" index="i">
          <Computing_Resource name="Processor_%i" />
        </Repeat>
      </Computing_group>
    </Computing_groups>
  </HW_Platform>
  <SW_Platform>
    <SW_Components>
      <SW_Component name="SO" type="OS" />
    </SW_Components>
    <SW_Architecture>
      <SW_Instance name="OS1" component="SO" HW_Resource="node1" />
    </SW_Architecture>
  </SW_Platform>
  <Application>
    <Functionality>
      <Exec_Component name="2" category="SW" function="par_1_main" />
      <Exec_Component name="4" category="SW" function="par_2_main" />
      <Exec_Component name="5" category="SW" function="par_3_main" />
      <Exec_Component name="6" category="SW" function="mpa_par_4_main" />
      <Exec_Component name="7" category="SW" function="par_5_main" />
      <Exec_Component name="8" category="SW" function="par_6_main" />
    </Functionality>
    <Allocation>
      <Exec_Instance name="main_func" component="__num_cpus"
      resource="node1" arguments="run.x sequence.ctl stimuli "/>
    </Allocation>
  </Application>
  <Simulation time="2000s" end_as_sw="1" backtrace="3" />
</Description>
```

FIGURE 7: XML SYSTEM DESCRIPTION FOR MPEG EXAMPLE

To demonstrate that the proposed modeling technique is adequate for DSE, the DSE tool has been programmed to simulate all the possible parameter combinations. Considering that there are 6 possible parallelization, 5 frequencies and 4 cache sizes, the overall possible configurations are 120.

Considering that an execution of the application SW requires about 20 seconds, the 120 executions, and that the overall simulation time is 40 min, it can be claimed that the proposed modeling technique for modeling and simulating modifiable systems is feasible for exploring the system architecture and configuration, considering that ISS based solution can takes days in simulating a few set of configurations.

The entire XML system description for the mpeg example is presented in figure 7.

VII. Conclusions

DSE is commonly accepted as a powerful solution to optimize system design by selecting the best configuration options for the system components. This work demonstrates that the same flows can be also used to optimize the elements and components of the platform.

Common XML solutions for modeling system platforms can be easily extended to support really modifiable platforms. A very few set of new XML clauses are only required for this purpose.

Using TLM based component models together with state of the art solutions for automatic system modeling, automatic parameterization and modeling of modifiable platform descriptions is possible.

Furthermore, the resulting modeling technique is efficient enough to allow exploration of highly configurable platforms following multi-objective optimization requirements.

References

[1] A. Sangiovanni-Vincentelli & G. Martin: “Platform-Based Design and Software Design Methodology for Embedded Systems”, IEEE Design and Test of Computers, November-December, 2001

[2] CoWare Platform Architect. Available at http://www.coware.com/products/platformarchitect.php

[3] ARM MaxSim Tools. Available at http://www.arm.com/products/DevTools/MaxSim.html.

[4] The P1685 IP-XACT IP Metadata Standard. Design & Test of Computers, IEEE, April 2006, Volume: 23, Issue: 4, On page(s): 316- 317

[5] K. Popovici, X Guerin, F. Rousseau, P. S. Paolucci, and A.A. Jerraya: Platform-based Software Design Flow for Heterogeneous MPSoC. ACM Transactions on Embedded Computing Systems, July, 2008

[6] L. Cai, and D. D. Gajski. Transaction Level Modeling: An Overview. In Proc. of CODES+ISSS’03.

[7] T. Y Yen, and W. Wolf. Communication synthesis for
distributed embedded systems. In Proc. of ICCAD’95.

[8] R. B Ortega, and G. Borriello. Communication synthesis for distributed embedded systems. In Proc. of ICCAD’98.

[9] S. Pasricha, N. Dutt, and M. Ben-Romdhane. Extending the transaction level modeling approach for fast communication architecture exploration. In Proc. DAC’04.

[10] K. Lahiri, A. Raghunathan, and S. Dey. Efficient exploration of the SoC communication architecture design space. In Proc. ICCAD’00.

[11] modeFRONTIER. http://www.esteco.com

[12] Multicube Explorer:
http://home.dei.polimi.it/zaccaria/multicube_explorer/Home.html

[13] A. Khare, N. Savoiu, A. Halambi, P. Grun, N. Dutt, A. Nicolau, V-SAT: A visual specification and analysis tool for system-on-chip exploration. Proc of Euromicro, 99

[14] D. Lyonnard, S. Yoo, A. Baghdadi, and A. A. Jerraya. “Automatic generation of application-specific architectures for heterogeneous multiprocessor system-on-chip”. In Proc. DAC’01.

[15] A. Wieferink, R. Leupers, G. Ascheid, H. Meyer, T. Michiels, A. Nohl and T. Kogel. Retargetable generation of TLM bus interfaces for MPSoC platforms. In Proc. of CODES+ISSS’05.

[16] H. Posadas, D. Quijano, J. Castillo, V. Fernández, E. Villar, M. Martínez: "SystemC Platform Modeling for Behavioral Simulation and Performance Estimation of Embedded Systems" in the book L. Gomes and J. M. Fernandes: “Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation”, IGI Global. 2009-07

[17] Magillem 4.0, http://www.magillem.org

[18] NAUET Design Assembler, www.mataitech.com

[19] Atomium tool suite, www.imec.be/atomium