PDESIGNER – A MPSOC MODELING FRAMEWORK

Andre Souza, Millena Almeida, Williams Azevedo, Cristiano Araújo, Edna Barros
Informatics Center (CIn) - Federal University of Pernambuco

Filipe Rolim, Abel Silva
Escola Politécnica – University of Pernambuco
Recife, PE, Brazil

ABSTRACT

This paper presents the PDesigner Framework, a set of tools based on ArchC and SystemC and integrated in the Eclipse workbench, which allows the modeling, simulation and analysis of multiprocessor SoCs. The framework supports platforms modeled at different abstraction levels. PDesigner supports processor modeling, platform modeling, component distribution, simulator generation, execution an energy consumption estimation of cache components. Using the analysis capabilities of the tool the designer can explore the platform architecture space in an efficient manner. Moreover, all these features are integrated in a unified and configurable graphical environment. The PDesigner is based on components library distributed at a standard that enables the user to import and export components for other tools or designers. That library was used to explore platform architectures and to design a platform to execute the FFT and inverse FFT applications of the Mibench benchmark.

1. INTRODUCTION

The fierce competition in the IC market is characterized by the design of complex circuits. This reflects the design of multiprocessor platforms to implement such circuits. These platforms are normally implemented as SoCs (System-on-chip [1]) and may be composed by several heterogeneous processors like RISC, DSP, high performance interconnection structures and memory hierarchy. Moreover the increase in the complexity, designers have to work with tighten design times[2].

In the design of SoCs there are two main trends: the reuse of components previously designed and verified (IP-cores), and the increase of the model abstraction level, through system level description languages, such as SystemC [3]. IP reuse and high abstraction level allows faster system modeling, analysis and validation through simulation. Moreover, the use of high abstraction levels provides architecture exploration in earlier design phases and with a smaller effort, and also allows earlier software development.

IP reuse presents some particularities. It is only useful when the effort to integrate the component is smaller than creating a new one. Difficulties arise during the integration of these components due to incompatibilities in the component interface and insufficient documentation. Those problems cause, in complex cases, a great effort to adapt the interface and protocols of the IP to the interfaces and protocols of the system. This effort is, in some cases, similar to designing a new component from scratch. In order to decrease the integration effort, industry and academy have proposed standards like the SPIRIT Consortium IP-XACT [4] for the IP distribution through XMLs descriptions of its structure and use, easily adapted for use in platform modeling tools.

This paper presents the PDesigner framework, a set of SystemC based tools for multiprocessor SoC design that enables the modeling, simulation, architecture exploration and component analysis in an integrated framework compliant to SPIRIT IP-XACT 1.2.The rest of the paper is structured as follows. Related work is described in section 2. PDesigner design flow is presented in section 3 The PDesigner framework is explained in section 4. PDesigner analysis plugins are described in section 5. An integrated IP distribution tool is described in section 6. The results are discussed in section 7. Finally, some conclusions are given in section 8.

2. RELATED WORK

CoWare [5] Virtual Platform, is a commercial framework that enables the modeling of virtual platforms, from its creation until its distribution. This framework allows parallel software and hardware development and its validation through the simulation of the virtual modeled platform. The CoWare management infra-structure allows the platform packaged distribution and auto installation, that includes a simulator and tools for performance analysis, system controllability and observability.

The ArchC Reference Platform (ARP) [6] is a platform structure model based on ArchC[13] processors. It defines a directory structure for platform components, as processors, buses, memories, configured by a script that generates, compiles and simulates the platform. Using ARP the designer has to integrate the components in the platform manually.

3. PDESIGNER DESIGN FLOW

The PDesigner design flow (Figure 1) enable the entire platform modeling, since the description of a component until the generation of the simulator used to analyze it.

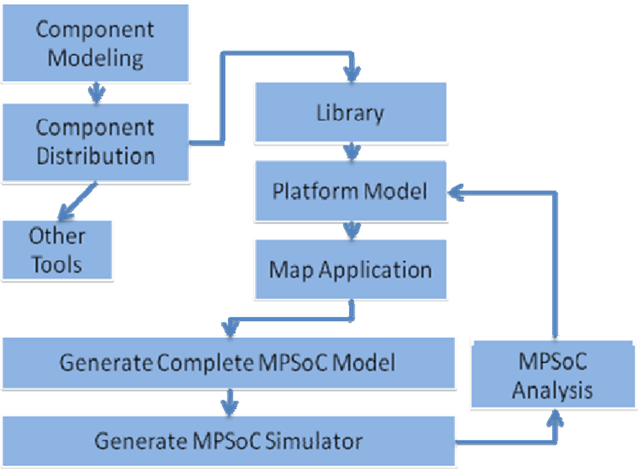


Figure 1 PDesigner Design Flow

The designer can model a platform within PDesigner framework using components designed by different ways and tools. Other SystemC components can be distributed using an IP distribution tool and added to the same library.

Once the designer has all the desired IPs into the PDesigner library, he/she can model the platform and load the application into its processor(s). The complete MPSoC model is generated after platform configuration through component parameters set.

The PDesigner allows the user to simulate the MPSoC modeled and use its information to analyze it and explore the architecture using some available analysis tools integrated to PDesigner.

4. PDESIGNER FRAMEWORK

PDesigner integrates several platform tools in a single Eclipse[12] based framework. Working with platforms and components at Electronic System Level (ESL)[7], the PDesigner user is able to develop software and hardware projects concurrently using virtual platforms. They enable a fast simulation, exploration and validation of MPSoC projects.

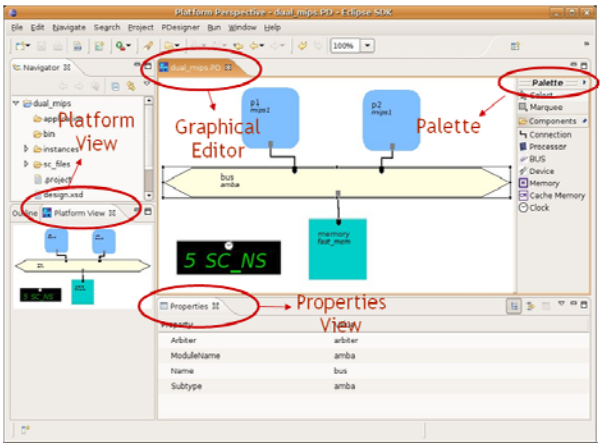
All the components used to model the platforms are stored in a dynamic library that allows the insertion and removal of IPs distributed in the IP-XACT 1.2 standard. Those components can be described in different abstraction levels using SystemC TLM[8] and be modeled and inserted in the library by the user. The user access the component library using the palette, selecting any component to add to his/her platform design.

Beyond a graphical editor, also provided by others platform modeling frameworks, the PDesigner focus on connection transparency. The connection between components is done automatically when the user selects the start and end point of the connection. Connection points are represented by master and slave interfaces and are enabled by the protocol, abstracting signals. Wrappers are used to enable the connection between different protocol ports. In order to support automatic component connection, wrappers must exist in the PDesigner component library. The framework recognizes the interface and protocol of each component to allow the connection and configure it. In case of different protocols, an appropriate wrapper is automatically inserted to the platform to allow the connection.

One advantage of PDesigner is that it supports the automatic connection and simulation between components at different abstraction levels. This feature allows the designer to balance the trade-off between simulation performance and accuracy.

4.1. Views

The PDesigner offers specific views to allow a better visualization and manipulation of the components and design information. Figure 2 shows the views that composes the PDesigner platform modeling plugin.



Property	Value
Arbiter	arbiter
ModuleName	amba
Name	bus
Subtype	amba

Figure 2 PDesigner Framework

Platforms are graphically modeled at a Graphical Editor called *PEditor*. The *Platform View*, a non editable outline view, shows the miniature of the project. The *Instance View* lists the instantiated components and can be used to navigate through component instances and, in addition, to configure them through an editable *Properties View*.

The framework also contains a *Console View*, where all compilation, simulation and analysis information are displayed.

4.2. Architecture

PDesigner is a set of tools build on Eclipse IDE, composed by modules implemented as plugins that communicate among them through XML configuration files. It is composed by five plugins as depicted in Figure 3. The three main plugins, PDLibrary (PDesigner component library), PBuilder (platform modeling and simulation plugin) and PArchC (processors modeling plugin), are responsible by component and platform modeling and library manipulation. It is also composed by another plugins that implements analysis (PCacheAnalyser PCacheEnergyAnalyser) and IP distribution tools, all of them explained in the next sections.

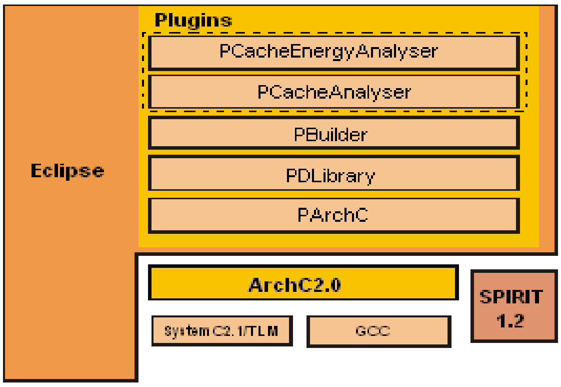


Figure 3 PDesigner Architecture

The framework was build compliant to three standards: *SystemC*, that enables *MPSoC* modeling and simulation; *ArchC[14]*, used to describe, compile and simulate processors; and *SPIRIT* used for IP distribution and integration.

The PDesigner has an architecture divided in layers as shown in Figure 3. The two main layers are formed by the *Eclipse* framework, and *Support* tools represented by the lower level.

The support layer is composed by languages, standards and tools that enable platform modeling project. SystemC and ArchC are used to describe components and SystemC is used to describe, compile and simulate platforms using gcc. All those tools and components are integrated through XML configuration files.

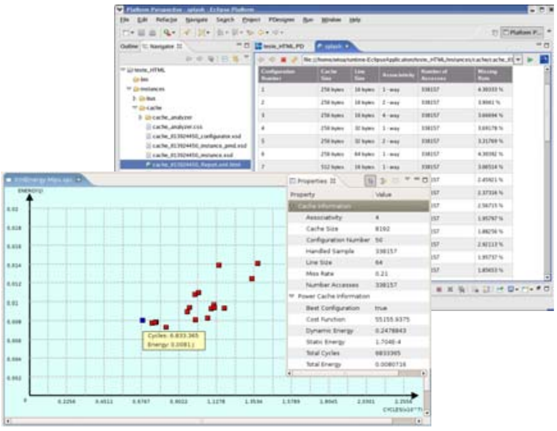
The Eclipse layer is composed by the PDesigner plugins: PArchC, PBuilder, PDLibrary, PCacheAnalyzer[9] and PCacheEnergyAnalyzer. The PArchC does not communicate directly with the PBuilder, only through the PDLibrary. These two form the base for the PBuilder, where the other plugins are added above, as PCacheAnalyzer and PCacheEnergyAnalyzer.

5. PDESIGNER ANALYSIS PLUGINS

PDesigner is also focused on platform analysis as a way to help the designer to decide the better architecture for the system. The analysis tools use the PBuilder as a source of information to generate its data and graphics.

One of the main focus on embedded system is the performance, that can be obtained through the use of specific application processors, parallelism and improving the communication speed between the devices that compose the platform. Another important way to improve the system performance is through the use of cache memories.

Based on this fact, PDesigner provides two analysis plugins for cache analysis. The PCacheAnalyser plugin provides caches miss an hit rates. The PCacheEnergyAnalyser provides cache energy consumption estimation. They all run at simulation time and generates charts (Figure 4) to show the desired information for more than one cache configuration, showing to the designer the best choice he/she can have.



Property	Value
Cache Information	
Associativity	4
Cache Size	8192
Configuration Number	50
Handled Sample	338157
Line Size	64
Miss Rate	0.21
Number Accesses	338157
Power Cache Information	
Best Configuration	true
Cost Function	55155.9375
Dynamic Energy	0.2478843
Static Energy	1.704E-4
Total Cycles	6833365
Total Energy	0.0080716

Figure 4 PCacheAnalyzer and PCacheEnergyAnalyzer Plugins charts

6. IP DISTRIBUTION TOOL

An IP distribution tool, called IPZip[10], is integrated to PDesigner environment. It generates, semi-automatically, the SPIRIT IP-XACT 1.2 distribution package Thus, enabling SystemC TLM developers to distribute their IPs in a manner that they can be reused automatically in a SoC modeling environment.

The IPZip is based on wizards that create IP Cores distribution package in zip format using a graphical framework, allowing the designer to create it in an easy and transparent way. The generated zip file is composed by the components source files and the component description in IPZip SPIRIT standard, that can be used for design tools as PDesigner.

The distribution files are composed by information about the vendor, component structure and configuration. Vendor information includes the vendor and author names. The component is described by its ports, parameters, source files and abstraction levels and also includes additional information about version and description. Configuration information is composed by how to initialize and use the component, including the constructor, connection and initialize functions structure.

7. CASE STUDY

The PDesigner framework was validated using diverse configurations for a dual MIPS platform distributed for ArchC 2.0. The initial configuration consisted of two MIPS processors (P1 and P2) connected to the generic SystemC SimpleBus, making use of only one memory, for data and instructions. This platform was simulated with a FFT (Fast Fourier Transform) application, from Mibench [11], where one of the processors generates the Fourier transformed of a sequence of signals and the other gets the inverse transformed one, established in the results of the first one. This initial platform was modified for the configurations listed in Table 1, using it to explore some possible architectures to discover the best one to run the application using the results of the simulation as an input to define the best architecture.

Platform Architecture	Simulation Speed (K inst/s)		Simulation Time (s)
	P1	P2	
2 powerpc, simple bus, memory	67,17	97,16	45,88
2 mips, amba (clock 5ns), memory	67,51	109,48	44,88
2 mips, green bus, memory	111,01	189,83	25,62
2 mips, amba (clock 5ns), memory, caches	125,53	164,48	29,87
2 mips, green bus, memory, caches	125,64	214,84	23,91
2 sparcv8 amba (clock 5ns), memory, caches	213,45	294,02	12,65
2 sparcv8, green bus, memory, caches	289,52	495,74	12,44
2 powerpc amba (clock 5ns), memory, caches	200,97	261,05	16,44
2 powerpc, green bus, memory, caches	226,39	387,21	15,44

Table 1 Platform Simulation Results

Using the results, the designer can define the architecture, components and configuration to be used based on the non functional requirements, in this case performance. After deciding that the platform will have cache memories, the designer can also analyze the best cache configuration to be used, based on miss/hit rates and energy consumption.

8. CONCLUSIONS

In this paper it has been presented the PDesigner framework, a platform based design tool, which allows MPSoC modeling, simulation and analysis.

The obtained results showed that the PDesigner reached its main purpose, model MPSoCs platforms, simulate it and give to the designer the capability to analyze different platforms architectures rapidly concerning cache energy consumption.

The paper has also presented the IPZip tool for the distribution of SystemC TLM IPs. Using IPZip the designer was able to include new IPs in the library based on the new standard IP-XACT 1.2 from the SPIRIT consortium. All the components of the library were distributed using IPZip.

9. REFERENCES

[1] U. Kamath, R. Kaundim, “System-on-chip designs – Strategy for success”, Wripo Technologies. June, 2001.

[2] A. Jerraya, H. Tenbunem, W. Wolf. “Multiprocessor systems-on-chips”. Vol 1. Elsevier. 2005.

[3] http://www.systemc.org. SystemC Community.

[4] C. Lennard et al. Industrially proving the SPIRIT consortium specifications for design chain integration. In the Proceedings of the conference on Design, automation and test in Europe DATE '06,pp 142-147, Munich, Germany, 2006

[5]http://www.coware.com/products/virtualplatform_creation.php. CoWare: Virtual Platform Creation.

[6] ARP: ArchC Reference Platform. Available at :
<http://www.archc.org> “Platform” option.

[7] B. Bayley, G. Martin, A. Piziali. ESL Design and Verification.

[8] TML available at http://www.systemc.org

[9] P. Viana, A. Gordon-Ross, E. Keogh, E. Barros, F. Vahid. Configurable Cache Subsetting for Fast Cache Tunning. In Proceedings of the 43rd Design Automation Conference DAC 2006.

[10] C. Araujo, E. Barros, M. Gomes, G. Araujo. Ipzip - An IP Distribution Framework. In: IP/SoC, 2006, Grenoble. Proc. of the IP/SoC 2006, 2006.

[11] http://www.eecs.umich.edu/mibench/index.html. Mibench

[12] http://www.eclipse.org. Eclipse Open Source Community.

[13] http://www.archc.org. The ArchC Resource Center.

[14] R. Azevedo, S. Rigo, M. Bartholomeu, G. Araujo, C. Araujo, E. Barros, The ArchC architecture description language and tools. International Journal of Parallel Programming. Vol. 33, No. 5, out, 2005