

**SPIRIT IP-XACT controlled design flow applied
on a high level communication synthesis.
Scientific collaboration between
OFFIS & MAGILLEM Design Services
in the frame of ICODES**

Emmanuel Vaumorin (Magillem Design Services)
Cornelia Grabbe (OFFIS)
Dr. Frank Oppenheimer (OFFIS)
Vincent Thibaut (Magillem Design Services)

Abstract

OFFIS and Magillem Design Services have collaborated in the frame of a research project called ICODES. The overall goal of ICODES was to develop a new modeling and synthesis technology for embedded hardware/software systems, especially regarding communication aspects. The research topics consider modeling, evaluation and high-level communication synthesis of systems written in a language based on SystemC. The aim of this whitepaper is to present the result of the joined work to provide an efficient design flow based on IP-XACT.

As an introduction, this paper presents the OSSS library, its associate design methodology and synthesizer, the IP-XACT standard issued by the SPIRIT consortium, the MAGILLEM environment and the ICODES project context.

Then the OSSS methodology developed by OFFIS will be described. Especially the refinement of system communications by mapping an application layer model on different virtual target architectures will be presented in detail.

The third part explains how components and system description based on OSSS can be packaged in SPIRIT IP-XACT and how this packaging allows a semi-automated mapping of the application to the architectural layer and the integration of the OSSS design flow into the MAGILLEM environment.

1. Introduction

1.1. Communication synthesis OSSS and OFFIS synthesizer

One key to cope with the complexity of embedded HW/SW systems is the development of new design methodologies coupled with design tools to raise the level of abstraction above RTL. This will allow continuing the successful completion of ever more complex electronic systems.

The goal of ICODES (Interface- and COmmunication-based Design of Embedded Systems) is to develop an efficient design methodology for communication dominated HW/SW systems. The aim is on one hand to start with an executable specification where many hardware and software modules communicate with each other and on the other hand to analyze, optimize and automatically synthesize the design to a chosen target platform.

OFFIS introduces a design language called OSSS (Oldenburg System Synthesis Subset) that is based on SystemC to model the embedded hardware/software system. Designs written in OSSS can be automatically synthesized to a target platform.

1.2. IP-XACT from the SPIRIT consortium

IP-XACT is an XML based open standard defined by the SPIRIT consortium. This non-profit organization provides a unified set of high quality IP-XACT™ specifications for documenting IP using meta-data. This meta-data will be used for configuring, integrating, and verifying IP in advanced SoC design tool sets and interfacing tools using APIs: the LGI and TGI APIs can be used to access design meta-data descriptions of complete system designs. The specification for the schema is tailored to the requirements of the industry, and focused on enabling technologies for the efficient design of electronic systems from concept to production.

1.3. MAGILLEM environment

MAGILLEM tool suite is a database management system for the IP-XACT standards by the SPIRIT consortium and a complete design environment including debugging functions and the ability to run generators. MAGILLEM 4.0 offers an innovative tooling for IPs import and packaging, design assembly and flow control. It also provides an implementation of the IP-XACT APIs (TGI and LGI), which is required to support point tools encapsulation within a controlled design flow, and complex configurable IPs (most re-usable IPs are configurable). MAGILLEM 4.0 is not just another EDA or ESL tool, but an environment that enables interoperability and federates existing point tools, languages, methods and models into a seamless, automated flow. The MAGILLEM solution is a Java-based plug-in in Eclipse™ and runs on all architectures (Windows, Linux, etc.).

1.4. ICODES project

The ICODES project is funded by the European Commission's 6th Framework Program under the IST (Information Science Technology) priority. The project started at the 1st of August 2004 and ends at the 30th of October 2007. The consortium of the ICODES project consists of industrial and research partners. Three major European companies, Robert Bosch GmbH (Germany), Nokia Siemens Networks, S.P.A. (Italy) and Thales Communications S.A. (France) have defined the requirements on the methodologies and tools developed in ICODES. Research and development in the project are carried out by research partners: Politecnico di Milano (Italy), and OFFIS (Germany) and an SME: Magillem Design Services (France). The industrial partners evaluate the results in comparison with their existing design methodologies. For more information about ICODES, visit <http://icodes.offis.de/>

2. OSSS methodology

2.1. Overview

The language OSSS (Oldenburg System Synthesis Subset) extends the synthesizable subset of SystemC by supporting not only C++ concepts, like classes, templates, inheritance but also unique OSSS concepts, like the Shared Object and Polymorphic Objects [3][4]. OSSS (like SystemC) is provided by an open C++ class library which can be used in any valid SystemC model.

Figure 1 shows the overall OSSS design methodology for hardware/software system-design. It begins with the description of an embedded hardware/software system at *Application Layer*. The *Application Layer model* is executable and abstracts from the details of the communication links between the components.

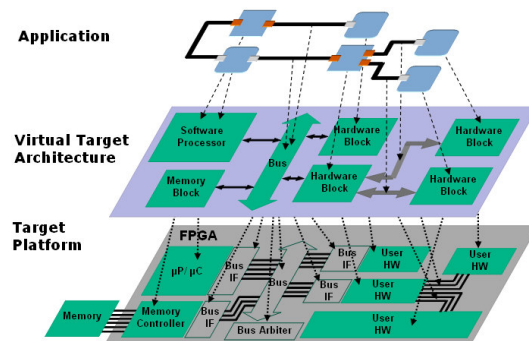


Figure 1: The OSSS design methodology of hardware/software systems

Hence, *Application Layer model* needs further refinement before the automatic synthesis process can be applied. Clear guidelines define refinement steps which separate the pure application from the physical architecture and lead to the so-called *Virtual Target Architecture (VTA) model*.

The *VTA model* remains executable and reflects now all relevant details of the implementation model. Therefore the *VTA model* can be used as direct input for the automatic synthesis process.

In a next step, the OSSS synthesis process transforms the *VTA model* to a representation which can be further processed by vendor specific compilers and synthesis tools to end up with a successful physical implementation on the chosen *Target Platform*.

In the following, the *Application Layer*, the *Virtual Architecture Layer* and the refinement concept is described. Afterwards, the overall OSSS synthesis flow from the application design to the target implementation on a Xilinx target platform is shown.

2.2. Application Layer

The *Application Layer* is the OSSS design entry to model the desired hardware/software design.

On this layer the design methodology provides the concepts to model hardware modules, software tasks and passive objects which can be shared between active parts of the design, named Shared Objects (see Figure 2).

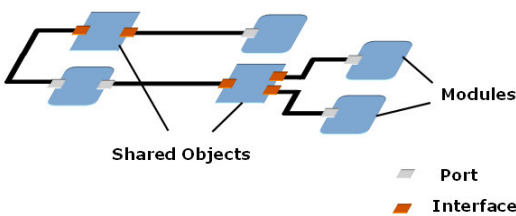


Figure 2: The *Application Layer* and its components

Active parts of the design are hardware modules and software tasks that contain processes and thus have an own thread of execution. Shared Objects are passive which means they do not initiate any execution on their own. Shared Objects [1] have two major properties: on one hand they provide a method based interface for inter-process communication and on the other hand they are a kind of shared resource which can be used by different processes.

On *Application Layer*, communication between modules or tasks and Shared Objects is performed by user-defined method calls with parameters of any valid C++ data type (except pointers & references) through communication links. These communication links are always directed¹ and method calls are always blocking².

2.3. The Virtual Target Architecture Layer

On this layer, several architecture building blocks are available that can be used to assemble the system architecture. These building blocks can be of type software processors, memories or user-defined hardware. The blocks can be connected via busses or point-to-point connections. All architecture building blocks are stored in a hierarchical library that can be extended by user-defined architecture elements.

Communication links are described by so-called OSSS-Channels [4]. The OSSS-Channel approach enables the designer to model the communication structure independently from the algorithmic behavior of the components.

The OSSS-Channels (e.g. `xilinx_opb_channel`, or `oss_point_to_point_channel`) are part of the *Virtual Target Architecture communication library*. They implement the so-called RMI (Remote Method Invocation) concept [6] and encapsulate the Physical Layer of the communication.

2.4. Mapping of the Application on the Virtual Target Architecture

The mapping step from the *Application Layer* to the *Virtual Target Architecture Layer* involves the mapping of modules and Shared

Objects to appropriate architecture building blocks. Modules and Shared Objects that should be implemented in software have to be mapped onto a software processor. Shared Objects that should be implemented in hardware have to be mapped on so-called sockets provided by the *Virtual Target Architecture Library*. Figure 3 shows an example of a *Virtual Target Architecture* composed of different `oss_architecture_objects`. It includes a `xilinx_opb_channel<...>`³ that connects a Xilinx MicroBlaze processor block (as bus master) and two `oss_object_socket<...>`s components (as slave). Two user-defined hardware-blocks use a simple point-to-point channel to connect to the second Shared Object Socket.

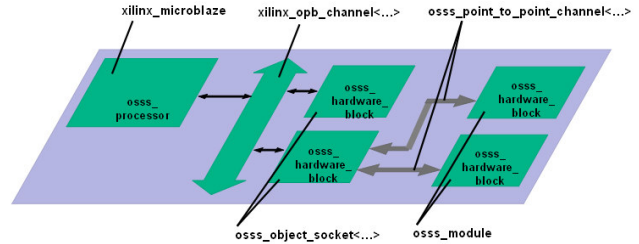


Figure 3: Example of a *Virtual Target Architecture* with a single processor and user-defined hardware

Besides the mapping of modules and Shared Objects the logical communication links in the *Application Layer Model* have to be mapped on physical communication resources in the *Virtual Target Architecture Model*. In this refinement process, multiple communication links can either be bundled in a physical shared bus or each communication link can be mapped to a dedicated point-to-point connection.

Figure 4 shows a mapping of an application to the *Virtual Target Architecture* presented in Figure 3. Obviously an `oss_software_task` can only be mapped on an `oss_processor` architecture building block. Peripherals of the MicroBlaze are attached to the On-Chip Peripheral Bus (OPB). Thus, both communication links from the `oss_software_task` to the Shared Objects are mapped on a `xilinx_opb_channel<...>`. Each Shared Object implementing a specific method interface is mapped to a dedicated `oss_object_socket<...>`. SystemC modules (`sc_module`) are used to specify user-defined hardware blocks and therefore are mapped on the specific `oss_module` of the *Virtual Target Architecture*. Both communication links from the modules to the second Shared Object are mapped on dedicated point-to-point channels.

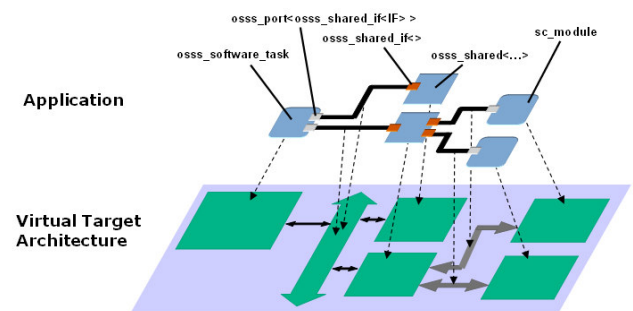


Figure 4: Mapping of the application on the *Virtual Target Architecture*

¹ The source (method caller) of a communication link is a port while the destination (method provider) is an interface, i.e. port to interface binding.

² Blocking here means that a method call on a port will not return until its execution has completed.

³ The Xilinx On-Chip Peripheral Bus (OPB) is a special implementation of the general IBM OPB; some of the main differences are described in [5].

3. Automation of ICODES design flow using a tooling framework based on IP-XACT

3.1. Presentation

One of the goals in the ICODES project was to implement and evaluate the possible integration of the OSSS methodology (developed by OFFIS) into a design automation environment (Magillem). We determined three axes to realize this integration:

1/ Packaging and design: study how the existing IP-XACT schema exploited in MAGILLEM could support the packaging of SystemC modules using the OSSS library and the description of systems at application and architectural layer; then define extensions in this schema and new features in MAGILLEM for non supported specific attributes.

2/ Communications refinement automation: study how the mapping rules from application layer to virtual target architecture layer in the OSSS methodology could be partially automated, taking advantage of the IP-XACT packaging for data management and code generation by TGI generators.

3/ Design flow automation: taking in account the several design activities applied by the industrial partners in ICODES and bring the capability to control the corresponding flows by assembling SPIRIT generators.

3.2. OSSS modules library import and IP-XACT packaging

The goal of this phase is enabling the integration and manipulation of OSSS based SystemC modules in the MAGILLEM environment by describing the required attributes in the extended IP-XACT standard. This action must be done once for all components to facilitate the reusability of the components in design projects and to maintain the exchangeability between design services or companies. The second role of the IP-XACT packaging is to provide an accurate management of the existing components library (directory structure, parameters and configurability management, component types, interfaces description, etc.). It allows describing the correspondence between the application and architecture layers by enabling the multi views of each system element. The packaging information will be used by a specific engine which will automate the mapping by generating automatically parts of the virtual target architecture layers following integrated rules and partitioning choices of the designer, as it will be shown later in this paper.

The packaging of the SystemC modules is done using the SPIRIT IP-XACT schema: an XML file is created for each component. As shown in Figure 5, the creation of this file can be done using a parser which will extract structural information from header SystemC files (.h) and generate the corresponding attributes¹. Other information, relative to the OSSS methodology has then to be added in the XML file. This can be done using the graphical editor of MAGILLEM, or through import of CSV² files. All the packaging steps may be assembled and fully automated for a complete import and maintenance of an existing library.

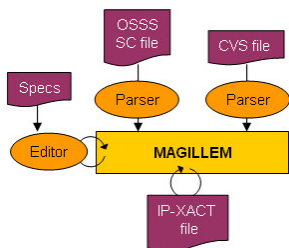


Figure 5: Packaging automation

Note: XML editors could be used for this packaging phase as well, but as they are generic and not specific for the electronic design

¹ Port list, interfaces with their type, constructor and template parameters, file sets.

² CSV: comma separated value files are text based and contain structured information using a separator like comma

(especially for IP-XACT), their usage is not recommended; it would make the packaging more time consuming and error prone.

3.3. IP-XACT Packaging guideline for OSSS

In order to be able to import OSSS IPs into MAGILLEM, we must first package them into IP-XACT. In the scope of ICODES three levels of packaging are possible: a basic packaging, a flow packaging and a full packaging. Those three levels are incremental and compatible for mixing them in a design. The packaging level is directly related to the feature needed for each IP.

Basic packaging is enabling the integration of a component into MAGILLEM: the minimum structural data required to facilitate the setting, the instantiation and the connection of the component into a larger design. Graphical representation of these data will be possible, the management of the hierarchical structure and some checks of the connections. With this level of packaging it will be also possible to apply a TGI generator, such as the application to architecture layer generator presented in part 3.5.

The required information for the design flow level packaging is contained in the interconnection and IP description attributes of the IP-XACT schema. Interconnection description can be found in bus definition files. At the application layer we mainly use the IP-XACT standard way to define a bus: describing a group of interconnect wires which are typically used to group all data and address port of a protocol for example. At the application layer we use several naming convention: bus definitions represents software interfaces available on a component, ports naming correspond to available methods inside the interface and port protocols are used to describe the method templates (OSSS_GUARDED).

IP description represents the basic packaging required for integrating an OSSS component into IP-XACT:

- Ports are defined in the attribute model>Ports: at basic level only name and direction are required.
- Interfaces are defined in the IP-XACT attribute busInterface, which allows regrouping bundle of ports corresponding to an existing IP-XACT busdefinition.
- The layer identification (application or architecture) is supported using the attribute IP-XACT model>view.
- The hierarchy in the layers is supported using the IP-XACT attribute model>view>hierReference, referencing a pointer to a design which describes the sub component.

3.4. Design entry / assembly

Starting from an existing and packaged components library, it is then possible to make the description of the system (called “design” in the IP-XACT terminology). It is required first to select the interface types and components which will be used for the design. The design assembly is then realized by instantiating components, and configuring them by setting their parameters and eventually applying component generators for complex configurations management (e.g. a bus matrix generator taking in account the number of masters/slaves and arbitration scheme). The management of the connections is a key point that needs to be assisted by the design environment: simply dragging lines between components and interfaces should be enough to hide the complexity of the communication models in OSSS/SystemC. A schematic representation of the design and its interconnected components is required to facilitate the visual analysis and, further in the flow, back annotate the system description with performance results issued by third party tools or simulations (e.g. timing performance of components and channels).

In order to handle all these design needs, we had to extend the capabilities of the existing MAGILLEM tool, because the OSSS methodology (mapping application to architectural view) is requiring specific functionalities: links between blocks may represent a simple port-interface connection, or the mapping of a software task on a processor, or a software reference of a channel with shared objects in a component. We had also to extend the meaning of some parameters in order to enable the description of different layers (application or architecture) views of the same functional element. The reporting for performance annotation which could be generated from third party

simulation analysis tools is possible using the mechanism for vendor extensions in IP-XACT.

3.5. Semi-Automated mapping of the Application on the Virtual Target Architecture

This phase of the design flow is the heart of the automation provided around OSSS methodology: using the communication elements from the available library, the designer must refine the interfaces of the system components and map the application layer onto the architectural layer in order to get a global model that can be synthesized by the FOSSY synthesizer. When analyzing the OSSS mapping rules (partially presented in part 2.4) we can notice that some of them are rather trivial or dependent only on a known parameters, and thus some design task may be automated to accelerate the refinement. For instance, in the case of a single type processor system, each application task tagged as software will be mapped on this type of processor. This means to create all the reference and communications links as define previously. The application modules specified as hardware will be replaced by their architectural view. Thanks to the IP-XACT packaging which supports multi-views for single component the modules will be equipped with their corresponding interfaces. In some cases, if this view does not exist in the library, a shell generator can create the corresponding SystemC module with the adequate interfaces taken from the architectural layer library.

Also some parameters or constraints of the architecture can be used to generate the target platform. For instance, the designer can decide the number of processors, the type of communication channel, or more globally select a target architecture from existing ones described in IP-XACT.

This automation can be seen as assistance for designers and will avoid some errors in the connections or the parameters setting of the OSSS models.

Managing this automation can be realized using the IP-XACT description of the application components and of the design and applying what is called a “IP-XACT generator”: a small engine (in Perl script or Java) which exploits the TGI¹ to automatically generate a part of the structural layer design and eventually some structural components that are derived from the application view (Figure 6).

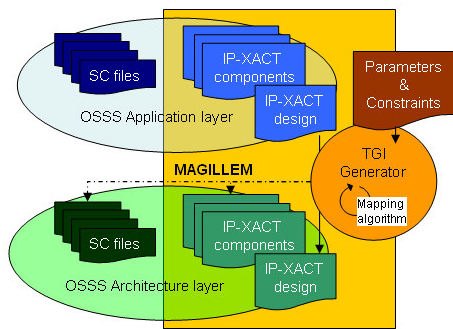


Figure 6: Mapping with TGI generator

Nevertheless, as explained in the part 3.3, the information contained in IP-XACT is not enough to support all the attributes of OSSS modules. Therefore, we have extended the schema by adding specific semantic to some model parameters. Each Application Layer component possesses a set of parameters which define its Architectural Layer counterparts. Some of them only define a hardware equivalent of the application IP, some are both: a hardware component and a set of software task, and some posses multiple hardware possible components. Starting from an application layer description the refinement generator applies a semi-automated translation from each application IP to its architectural counterparts. Three different cases are possible:

- Direct and single match from application IP to architectural IP, e.g.: Shared Object to Shared Socket.
- Direct and multi match from application IP to architectural IP, e.g. software task to processor and mapped software task.
- Multiple choices, e.g. connection to point-to-point or bus connection.

The algorithm that drives the behavior of the generator is the following:

- 1/Explore the design to identify all IP instances
- 2/Save the context of each module (parameters, interconnections, hierConnections, adhocConnections)
- 3/Identify for each application module the corresponding architectural elements
- 4/If needed, ask the user to choose from a list of choices
- 5/Replace application modules by architectural ones
- 6/Add additional communication elements and connect them
- 7/Dispatch saved connections between added elements

Once the generator has been applied, the application level system is replaced by one of the possible architectural targets; multiple execution of the wizard with different choices will result in multiple architectures which allow a quick exploration of communication and mapping solutions. This capability is possible thanks to a generators chain list as explained in the following part.

3.6. ICODES flow control

Flow control is a must when you need to automate part or the whole design process. We have seen that the ICODES flow is made of several tools and engines which are applied at different steps of the design flow. Reporting results from analysis or simulation phases on the design description may be important for decisions during the architecture exploration. Industrial users would also require to perform some checks at each step of the design process, ensuring safety and security of systems and significantly reducing design cycle by avoiding obsolete time consuming tasks (e.g. avoid launching a compilation when some models have not been generated yet). Flow control capabilities are also required when you want to assemble automated tasks into a chain of controlled tasks, which is one of the goals of ESL methodologies, in order to automate the architecture exploration.

Automating the ICODES flow control is possible using the IP-XACT generators. By using a set of generator, we can control all the key phases of the process: design project creation, design assembly, semi-automated mapping, simulation/reporting, and system synthesis.

To do so, we have created specific IP-XACT generators or wrap existing engines into it. For the latter, the generator is providing all the parameters of the engine and can be applied on a design or on a component from the IP-XACT view in Magillem. These generators can be chained into a generator sequence which allows the user to execute the full process in a single action. By using a checker at each step, the process is stopped if an error is found.

The following paragraphs summarize the main steps in the OSSS/MAGILLEM design flow.

Design flow context

For the following, we assume that the design flow has been integrated with all the required tools and that a library of components based on OSSS is available and already packaged in IP-XACT.

Project creation and library import

This phase is using a generator that creates a project in Magillem and imports the IP-XACT files of a specified library of bus definitions and of all the required components.

Automated design assembly

TGI generators are available to automate the creation of the initial application model. This will avoid manual manipulations by recreating it (component instantiations, configuration and connections) with the TGI functions applied on an IP-XACT design.

Semi-automated mapping

Here the generator, which has been presented in part 3.5, is applied on an IP-XACT design described at the application layer. Following the

¹ TGI: Tight Generator Interface is the API defined in the IP-XACT standard to get and set the centric database of a design environment like MAGILLEM

mapping rules, a design is generated at the corresponding virtual target architecture with application mapped on it. The designer may modify or tune it before the next design step.

Simulation

The purpose is to simulate the mapped design and verify its functional and communication behaviour, and its timing performances in order to evaluate the architecture candidate.

Firstly a SystemC netlist is created by a generator applied on the IP-XACT design. This generator embeds all the coding style rules (e.g. choices on how to do instantiations (pointer or object) or interfaces connections) which facilitates the maintenance.

Then the building/compilation phase is launched by another generator which uses the file sets information available in IP-XACT to create the makefile for gcc or a config.ini file for NCSIM or ModelSim. Build/compilation IP-XACT parameters at the component or design level are taken into account.

The results of the simulation will be used to back-annotate the IP-XACT design (in vendor extensions or parameters) and MAGILLEM will display this information through the schematic interface.

Synthesis

This final generator provides the capability to set the synthesis constraints parameters in the IP-XACT environment and launch the FOSSY synthesizer using the file set information of each component.

4. Conclusion

We've seen in this paper the result of the work realized in the frame of a collaboration, supported by the European Commission, between a research laboratory and a company dedicated in tooling development. The goal was to study and develop a controlled design flow around the OSSS methodology, which is providing the technology for application mapping on target architectures. To realize this, we have proven that the standardized meta model schema, called IP-XACT and delivered by the SPIRIT consortium, could be used and extended to support the packaging of HW/SW application models and target platform mappings at the architectural layer. Moreover, the central IP-XACT database provided by MAGILLEM through the standardized TGI API has been exploited and the mapping rules have been integrated into a TGI IP-XACT generator to facilitate and secure the design. More globally, it has been proven that the combination of a complete controlled design flow associated to a synthesis methodology is the key to provide the advanced ESL features, which are expected by industry to speed up and secure the design process of complex hardware/software systems.

Bibliography

- [1] "Object-Oriented Hardware Design and Synthesis Based on SystemC 2.0". Grimpe, Eike, Wolfgang Nebel, Frank Oppenheimer und Thorsten Schubert.: In: SystemC: Methodologies and Applications. - Boston u.a.:Kluwer, Seiten 217–246, 2003.
- [2] "Extending the SystemC Synthesis Subset by Object Oriented Features", Eike Grimpe and Frank Oppenheimer, CODES + ISSS 2003, October 2003
- [3] "SystemC Object-Oriented Extensions and Synthesis Features", Eike Grimpe, Bernd Timmermann, Tiemo Fandrey, Ramon Biniash and Frank Oppenheimer, Forum on Design Languages FDL'02, September 2002
- [4] "OSSS-Channels: Modelling and Synthesis of Communication with SystemC". K. Grüttner, C. Grabbe, T. Schubert, C. Brunzema and F.Oppenheimer, FDL'06 , September 2006.
- [5] "On-Chip Peripheral Bus V2.0 with OPB Arbiter (v1.10c)", Xilinx, DS401, June 2005
- [6] "Object Oriented Design and Synthesis of Communication in Hardware-/Software Systems with OSSS" Kim Grüttner, Cornelia Grabbe, Frank Oppenheimer, Wolfgang Nebel,

accepted for SASHIMI'07, October, 2007.

- [7] M. Volkman, S.Balacco (Venture Development Corporation) - ESL Tools: Global Market Demand Analysis – www.vdc-corp.com - June 2007
- [8] G. Martin, Chief Scientist (Tensilica, Inc.) - Barriers to ESL Adoption – www.soccentral.com - May 2007
- [9] R. Schutten (Synopsys, Inc) - Complexity and Software Drive ESL Solutions – www.soccentral.com – February 2005
- [10] Byte Paradigm - Electronic system-level development: Finding the right mix of solutions for the right mix of Engineers - www.byteparadigm.com - April 2005
- [11] G. Harper (Bluespec, Inc.) - Are You Building Your ESL Design Flow on Sand? – www.soccentral.com - February 2005
- [12] R. Goering – Designers adopt ESL, but tools are lacking – www.design-reuse.com – June 2005
- [13] R. Goering – Tools missing as ESL rolls – www.design-reuse.com – June 2005
- [14] SystemC methodologies and applications - Müller, Rosenstiel et Ruf, Kluwer Academic Publishers – 2003

Glossary

OSSS: Oldenburg System Synthesis Subset is the extended synthesis subset supported by the FOSSY. By including the OSSS-lib one can create OSSS models in any standard SystemC development environment.

OSSS model: An OSSS model is a SystemC model which follows the OSSS modelling guidelines and restrictions. There are two distinct levels of OSSS models. The Application Layer model describes communication between components via direct method calls. The Virtual Target Architecture Layer model describes communication via refined osss_channels which represent the links between components at signal level.

Implementation Model: An implementation model (typically containing VHDL, C++ and configuration files) is a representation of a SoC that can be directly implemented using a standard tool flow for a particular platform.

FOSSY: The "Functional Oldenburg System Synthesizer" is a tool developed by OFFIS e.V. to translate OSSS models into implementation models.

EDA: Electronic Design Automation is the category of tools for designing and producing electronic systems ranging from printed circuit boards (PCBs) to integrated circuits. This is sometimes referred to as ECAD (electronic computer-aided design) or just CAD. (Printed circuit boards and wire wrap both contain specialized discussions of the EDA used for those.) The term EDA is also used as an umbrella term for computer-aided engineering, computer-aided design and computer-aided manufacturing of electronics in the discipline of electrical engineering.

ESL: Electronic System Level is an emerging electronic design methodology that focuses on the higher abstraction level concerns first and foremost. It is defined in the book "ESL Design and Verification: A Prescription for Electronic System Level Methodology" by Brian Bailey, Grant Martin and Andrew Piziali and published by Morgan Kaufmann/Elsevier 2007 as: "the utilization of appropriate abstractions in order to increase comprehension about a system, and to enhance the probability of a successful implementation of functionality in a cost-effective manner.

IP: Intellectual Properties are hardware or software modules used to build a system on chip.

IP-XACT: the new name of the SPIRIT schema, which will be standardized as IEEE 1685. Its purpose is to provide a well-defined XML Schema for meta-data that documents the characteristics of Intellectual Property (IP) required for the automation of the configuration and integration of IP blocks; and to define an

Application Programming Interface (API) to make this meta-data directly accessible to automation tools.

SoC: A System-on-Chip is an electronic device integrated on a single die.

SPIRIT: initially this acronym (Structure for Packaging, Integrating and Re-using IP within Tool-flows) was used to identify the meta-model schema in XML used to describe RTL and TLM IPs and systems' attributes. This schema is now about to be IEEE standardized and is called IP-XACT. The SPIRIT consortium is the owner of this schema.

TLM: Transactional level Modeling is a high-level approach to modeling digital systems where details of communication among modules are separated from the details of the implementation of functional units or of the communication architecture. Communication mechanisms such as busses or FIFOs are modeled as channels, and are presented to modules using SystemC interface classes. Transaction requests take place by calling interface functions of these channel models, which encapsulate low-level details of the information exchange. At the transaction level, the emphasis is more on the functionality of the data transfers and less on their actual implementation that is, on the actual protocol used for data transfer.

XML: this name stands for eXtensible Markup Language. XML is a markup language much like HTML; it is a file format which was designed to describe data, where tags are not predefined and where you must define your own, through a schema that will be used to set the syntactic and semantic rules for the files.

Partners presentation

OFFIS, the "Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und Systeme", is a non-profit partly state funded research institute associated with the University of Oldenburg in Germany.

OFFIS has currently some 200 employees including more than 100 full time scientists.

In the research division participating to the ICODES project, research focuses on system-level design automation and power optimization of embedded systems. The emphasis of current research concentrates on modelling and synthesis of adaptive systems and communication driven system-level design. The core technology in these activities is SystemC and OSSS for which OFFIS develops design flows and tools to improve the efficiency of system-level design processes. The near-term goal is to commercialize the OSSS methodology and in particular the FOSSY synthesis tool.

OFFIS has a successful track record of participating in and managing many EU-funded projects, several of which in the system level design domain, namely ODETTE, ICODES, and ANDRES.

Magillem Design Services has been established by a team of seasoned engineers and a group of business angels in the fall of 2006. The company has inherited "Magillem", a robust and innovative technology worth 120 man years. The headquarters are in Paris, France with a subsidiary in the USA and sales office in Asia. Our team has been a major contributor to the IP-XACT specification since 2003 and we are the only vendor providing a tool implementing all the versions including the latest one of the specification. We audit the existing industrial flows and propose a work plan to adapt them to IPXACT. We validate and verify the full compatibility of tools interfaces into a flow testbench. We test the IP deliverables against a benchmark for compliance using our SPIRIT PACK and check IP integration properties onto a test system. Using our own versatile Magillem toolbox, we are well equipped to offer a wide range of services (Analysing customer's database specifics and customizing SPIRIT Packager accordingly, Implementing new features to meet customer's requirements, Designing complex IP specific configuration nutshells, Designing and implementing system configuration dashboard and custom checkers) to streamlining flow process, developing and integrating specific point tools, developing and integrating tools for the global architecture of a start-to-end ESL flow.

More information on www.magillem.com