

# SPiRiT IP-XACT Extensions and Exploitation for Verification Software Methodology

Emmanuel Vaumorin – Magillem Design Services  
Jan Stuyt – NXP  
Fatih Kilic - Magillem Design Services

## Abstract

NXP and Magillem Design Services have collaborated in the frame of a European project called SPRINT. One of the objective of this work was to study the possibility to exploit in the IP integration verification area the new standard IP-XACT issued by the SPIRiT consortium for packaging and managing metadata for hardware platform. This paper presents the overall capabilities of IP-XACT for the management and automation of ESL methodologies and it presents the methodology, based on software, used by NXP for IP integration verification; extensions in the standard for supporting the automation are listed. Then after explaining how the existing tool flow could be enhanced, the paper details the specification of new generators for an automated verification flow and their application on a real platform by NXP.

## 1. Introduction

### 1.1. 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<sup>1</sup> and TGI<sup>2</sup> 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.

### 1.2. 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 the ability to run code generators/configurators and debugging them. 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.).

### 1.3. Verification software methodology in NXP's ESL design flow

NXP uses an embedded software-driven structural verification flow for IP-based system designs. This verification software (vSW) is supplied by the IP providers and therefore constitutes a reusable verification view of an IP. Most of the information needed to automatically generate the vSW for the IPs (and also for the system design itself) can be found in the related IP-XACT component and design files. However, there is still some essential information missing, and this paper addresses the required IP-XACT extensions to fully automate NXP's structural system verification flow.

## 1.4. SPRINT project

The SPRINT project is funded by the European Commission's 6th Framework Program under the IST (Information Science Technology) priority. The project started at the 1<sup>st</sup> of February 2006 and ends in November 2008. The consortium of the SPRINT project consists of industrial and research partners. Three major European companies are part of the consortium: ST microelectronics (France), NXP semiconductor (Netherlands), Infineon (Germany). The global objectives of the SPRINT project is enabling Europe to be the leader in design productivity and quality in Systems-on-Chip (SoC) design, by mastering the SoC design complexity with effective standards and design technology for reuse and integration of IP. The approach is to develop a standards-based open design platform that supports the development of interoperable and reusable IPs and their efficient integration into high quality SoCs. The goal is to stimulate a SoC design ecosystem to the benefit of SoC integrators and (emerging) IP providers and EDA companies and get early availability of advanced SoCs for system integrators to excel in their markets.

The SPRINT project provides a broad and integrated research initiative addressing the complete SoC design flow from initial specification to a verified synthesizable RT-level description of the SoC. The SPRINT SoC design flow advances over a traditional SoC design flow by providing enhanced IP reuse, consistent design across abstraction levels, and enhanced automation of IP integration, verification and debug. As a consequence it supports early verification and debug as well as early software development. This breaks the sequentiality of traditional design flows.

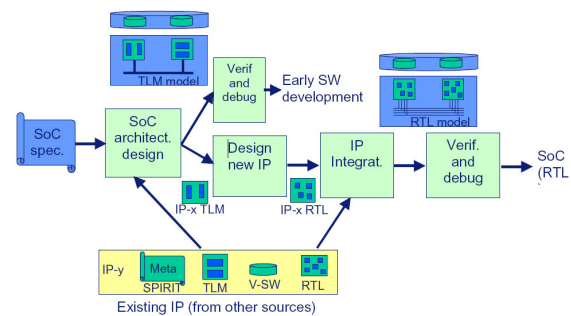


Figure 1: SPRINT SoC design flow

For more information about SPRINT, visit <http://www.ecsi-association.org>

## 2. IP-XACT for automating ESL activities

### 2.1. Overview

IP-XACT from the SPIRiT consortium is nowadays recognized by the electronics community as an apposite choice for managing properly and efficiently the new ESL design flows [1]. Nevertheless, the migration from a legacy design flow to another, taking full benefits of IP-XACT, requires some heavy and complex operations. Figure 2 presents the four steps which have to be completed. They are detailed in the following subsections.

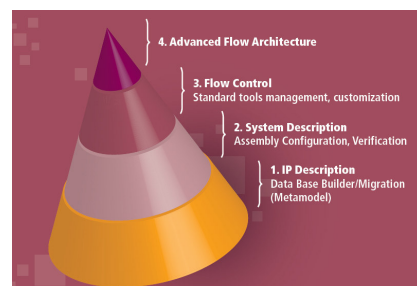


Figure 2: A 4-step methodology to build ESL flows

<sup>1</sup> LGI: Loose Generator Interface allows SPIRiT database access by file exchanges

<sup>2</sup> TGI: Tight Generator Interface allows SPIRiT database through a software API

## 2.2. IP description

The goal of this first step is to package all the components of an IP library into XML files in accordance with the IP-XACT schema, which describes the syntax and semantic rules for the description of three kinds of elements: the bus definitions, the components and the designs (in which components are instantiated). Thus the purpose of the IP packaging is to fill in for each component the XML fields that describe its attributes: physical ports, interfaces, parameters, generics, register map, physical attributes, etc. An important part of the schema is dedicated to referencing the files related to the different views of a component: a view may be for instance a simulable model in a specific language (VHDL, Verilog, SystemC, etc) or documentation files (e.g. PDF, HTML, Framemaker). This work facilitates future reuse of existing components, because all of their features are easily accessible for its integration and configuration in a bigger system, as it will be explained in the next step.

## 2.3. System description

After this step, is it possible to import, configure and integrate components into the system, assemble the design, resolve connections issues, and automate design tasks, thus lightening the verification steps. Some examples of the use of IP XACT at this level are: partial or full automation of design assembly and configuration, detection of communication protocols mismatch, toplevel netlisting, or automatic customization of compilation and simulation of designs. The work that is the topic of this paper takes place in this category: managing the generation of specific verification code from the IP-XACT description of a system and its components.

## 2.4. Design Automation and Flow control

The third step of the methodology, depicted in Figure 3, aims at linking the design activities around the centric IP-XACT database by means of a dedicated environment which provides access to the IP-XACT information. The tool suite chosen for this study (Magillem environment) provides an IP Packager, a Platform Assembly tool, as well as a Generator Studio to develop and debug additional TGI-based generators. These may be encapsulated within the IP-XACT representation of an IP and may for example simply launch the execution of a script, getting arguments values from the design description in IP-XACT, or be on the contrary a more complex engine, the role of which would be to modify the design itself (e.g. add connections, insert adapters, or configure components).

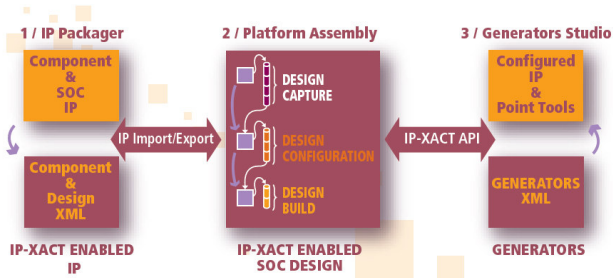


Figure 3: principle schema for an IP-XACT flow

Checkers can also be developed and used to verify design rules at some point, before going further in the design flow. Besides, IP-XACT provides mechanisms to describe the sequences of chained generators and checkers.

## 2.5. Advanced design flow architecture

This last step in the methodology has a high potential because it exploits all features described previously and allows the actual implementation of advanced ESL activities, such as architecture exploration or software application automated mapping on a hardware platform. These examples show the complexity that has to be managed by the three first steps: all components must be packaged and their configurability must be taken into account; the design assembly automation should be maximized, while any

architecture choice should be handled. At last, the generator chains, as defined previously, can be configured and controlled by supervisor engines: for instance a validation sequence will configure and execute several times the generators dedicated to testbench configuration, compilation and simulation.

## 3. vSW methodology in NXP

For IP-based (sub-)system designs it is a tedious job to find low level interconnection bugs through functional simulation on application level. For that reason NXP first applies structural system verification in order to find design errors that are not related to an application, but only to the platform standard that is being used in the design. Examples are bugs in address decoding, DMA channels, interrupts, clock programming, and so on.

This methodology requires that IPs are supplied with their own verification software (vSW) in plain ANSI-C. This software can be used by a SoC integrator to verify that the IP was successfully integrated in the system design. This is done by checking the connections of the IP to surrounding infrastructure IP.

As it turns out, the interfaces of any IP can be assigned to an interface class. Each of these (about 10) classes can be represented by a software API, which enables the IP to control or query the connected infrastructure IP. For example, if the IP's verification software is causing the IP to generate an interrupt, an interrupt controller API function must be used to query the status of the connected interrupt controller in order to see if the interrupt request was successfully received.

Most of the IP verification software can be generated automatically from the information in the IP-XACT component file of that IP, but what is still missing are tags on an IP interface that tell to which interface class it belongs.

## 4. Extension of the IP-XACT schema and exploitation within Magillem

### 4.1. Extensions of IP-XACT for Context labels

Context labels are defined for components and for their interfaces (i.e. individual ports and/or buses). When the hierarchy of an IP-XACT design is traversed, the constituting components are inspected for context labels. These context labels are attached to the components themselves, to bus interfaces or to individual ports. They can be overruled by similar context labels in a design file where they are attached to component instances, interconnections and ad-hoc interconnections, respectively.

These context labels are formally defined in document [SCIPIV], which is a SPRINT proposal to enhance the SPIRIT IP-XACT 1.4 standard to contain behavioral properties of an IP component or interface. The following tables define these labels for components (and infrastructure components) and interfaces.

#### Component labels

Contemporary SoC designs consist of IP cores like interconnection network, processors, peripheral cores, etc. Such core-based designs contain design IPs that belong to one of the main IP groups which have been listed in the following label definitions:

Component labels	
isNetworkComponent	bridges, tunnels, adapters, buses, ...
isConnectivityComponent	off-chip interface: UART, USB, I2C, GPIO
isStreamingComponent	video data processor, ...
isInfrastructureComponent	clock and reset generator, interrupt controller
isProcessorComponent	CPU, DSP: ARM, MIPS, TriMedia, ...
isInternalComponent	peripheral without off-chip interface, e.g.mem.
isTestControlComponent	dedicated DFT component
isStubComponent	testbench component connected to off-chip interface
isVerificationComponent	verification monitor, or other testbench component
isUndefinedComponent	to capture all other components

Infrastructure component sub-labels	
isClockInfrastructure	clock generator
isResetInfrastructure	reset generator
isInterruptInfrastructure	interrupt controller
isDmaInfrastructure	DMA controller
isConfigurationInfrastructure	configuration register
isPowerInfrastructure	power switch controller
isEventInfrastructure	event controller
isTriggerInfrastructure	cross-trigger matrix
isPadMuxInfrastructure	IC pad multiplexer
isUndefinedInfrastructure	Other cases

### Interface labels

The interface labels are tags showing the intended use (design intent) of a bus interface or port. A system designer may have a different use (user intent) for an IP interface (e.g. a GPIO port normally used to drive an off-chip interface, but now used to control internal system signals); this should then be annotated to the IP instance in the system design, such as to overrule the original design intent.

Interface labels	
isNetworkInterface	MMIO slave bus interface
isConnectivityInterface	off-chip interface
isStreamingInterface	direct IP-to-IP interface
isInfrastructureInterface	interface to infrastructure IP
isTestWrapperInterface	DfT interface; not software-accessible
isUndefinedInterface	to capture all other interfaces

Connectivity interface sub-labels	
stubReference	VLNV reference to stub verification component in testbench
Infrastructure interface sub-labels	
isClock	clock input
isReset	reset input
isInterrupt	interrupt request output
isDmaHandshake	DMA flow control interface
isConfiguration	tie-off input or monitor output
isPowerControl	power switch control input
isEvent	event output
isTrigger	debug trigger output
isUndefined	all other infrastructure interfaces

Production test interface sub-labels (based on IEEE P1500)	
isSerialPort	wrapper serial input or output (WSI, WSO)
isSerialControl	wrapper serial control port (WSC)
isParallelPort	wrapper parallel input or output (WPI, WPO)
isParallelControl	wrapper parallel control port (WPC)

Because IP-XACT is intended to only capture fixed structural facts about an IP and avoids capturing functional information, the proposed context labels are put in so-called vendor extensions. Figure 4 and Figure 5 are presenting examples of IP-XACT code integrating the context labels for a component and for a design.

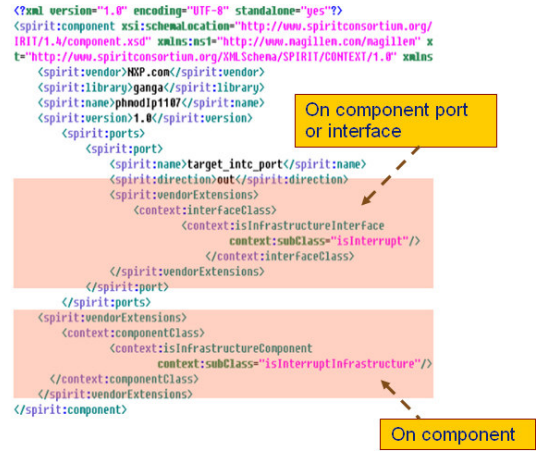


Figure 4 : Context labels in a IP-XACT component file



Figure 5: Context labels in a IP-XACT design file

## 4.2. Problems of the existing flow for vSW

The flow initially in place for the vSW within NPX is depicted in Figure 6. Starting from paper or excel like documentations, the HW platform is built, using RDL<sup>1</sup> and HDL resources are packaged in a proprietary SPIRIT like XML.

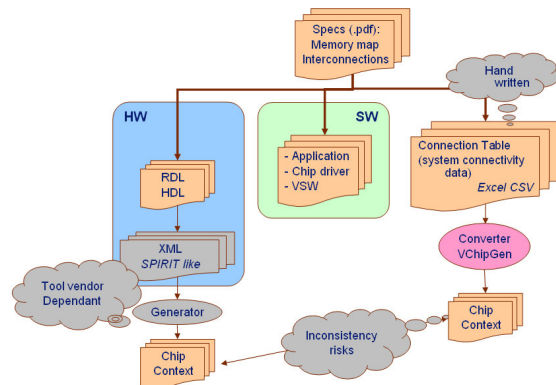


Figure 6: Existing flow for vSW

These files are exploited by a generator to create the chip context file, required to manage the verification. The same file could be generated through another flow; the architecture of the platform could be manually described in a CSV<sup>2</sup> format (excel sheet) to feed another chip context generator. This flow is divided in two parts and could generate inconsistencies and the manual creation of the CSV

<sup>1</sup> RDL : Register Description Language

<sup>2</sup> CSV : Coma Separated Value

file may be error prone. Figure 7 shows the format of the CSV file that has to be hand written and which is used to and that should be automatically generated from IP-XACT data, as it will be explained in the next paragraph.

component	number	cpu	clk	rst	int	dma	reg	tlb	stb
arm966v1	1		0,1	9,2	1,0		11,0		
intc1109v1	2	1,0x1000e000	0,1	9,0	1,0				
intc1109v1	3	1,0x1000f000	0,1	9,0	2,0				
rtc3011v1	4	1,0x10002000	0,2	9,0	3,0				
gpio4004v1	5	1,0xffff1000	0,0	9,0				nc	nc
uart3106v1	6	1,0xffff2000	0,0	9,0	3,1	10,0	11,5	nc	7,0
uart3106v1	7	nc	nc	nc	nc	nc	nc	5,0	nc
cpu412v1	8	1,0xffff8000		9,0					
xpu1077v1	9	1,0xffffc000							
dma1903v3	10	1,0xffff2c00	0,1	9,0					
creg9147v1	11	1,0x10005000	0,1	9,0					

Figure 7: CSV file manually edited as input of CC generator

#### 4.3. Improving the tool flow with IP-XACT

As described in Figure 8, we have worked for adapting and enhancing the existing flow by using the SPIRIT standard. The HW components were packaged in IP-XACT, allowing using any compliant tool environment to build the platform and exploit the meta data base by TGI generators: a new generator as been written in JAVA and is used to generate the intermediate CSV format from the description of a design in IP-XACT.

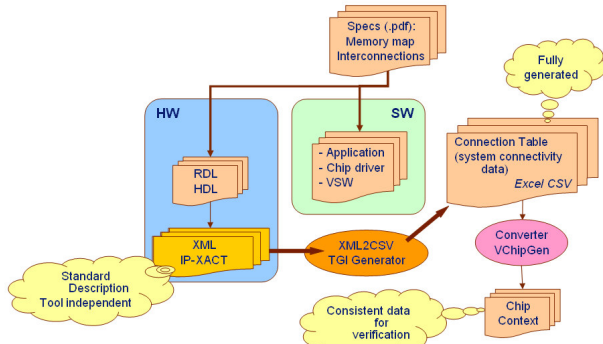


Figure 8: Improved tools flow for vSW

#### 4.4. Automate vSW generation from IP-XACT

The Figure 9 explains how the realized work is exploited in the vSW methodology. Each component of the library is available through its IP-XACT description, which is referring the HDL/RDL description files and the corresponding verification software view in C.

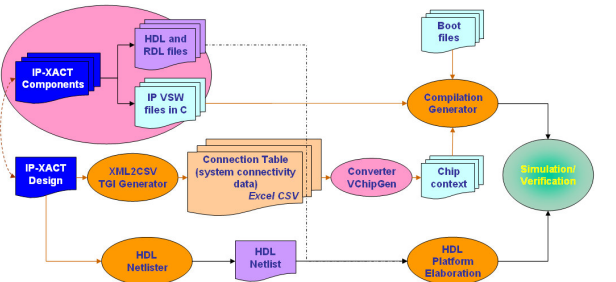


Figure 9: New vSW flow exploiting IP-XACT meta-data

The HW platform construction and elaboration for simulation is trivial and is based in IP-XACT and a netlister. In parallel, the tool chain described previously is used to drive the generation of the chip context from the same IP-XACT design description. The simulation can be launched in a press button manner, as all the elements of the

chain is packaged in SPIRIT generators linked in the generator chain.

#### 4.5. Validation platform and demonstrator

The validation of the tools has been done on a PCD<sup>1</sup> which is a real platform from NXP, composed by an ARM9 processor and peripheral resources. The bus definitions, components and the design were available packaged in IP-XACT in Magillem environment. We have used the vendor-extension mechanism to fill in the existing XML files with the context labels.

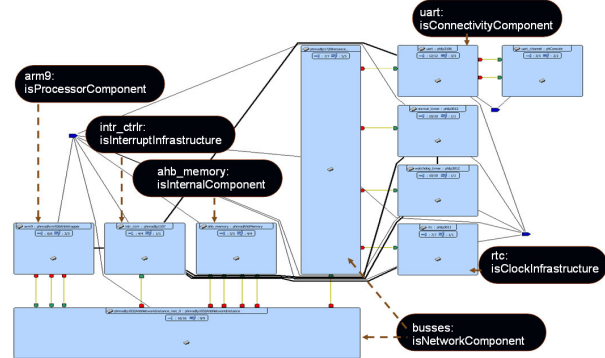


Figure 10: PCD captured in Magillem

The experimentations done on this PCD shown that the code required to launch the verification pattern is generated automatically for the IP-XACT design description

### 5. Conclusion

We have seen in this paper that the exploitation of the IP-XACT standard could bring a real benefit in existing flows such the one used by NXP for IP integration verification. When the platform design flow is based on IP-XACT (components packaging and platform description), then the exploitation by a SPIRIT compliant environment facilitates the creation of new generators for automating steps in the process.

A next step in the project objectives is to go ahead in the exploitation of the SPIRIT-based flow and study for instance how to generate and assemble the chip context code for hierarchical components. Thus it would be possible to verify the right integration of components deeply in the hierarchy of complex systems. Another research axe is to envisage the generation of the verification software code from IP-XACT description of an IP. There we should define a standard way to describe simple test using primitives of an API (e.g. to generate an interrupt) and handle it into an XML structure compatible with IP-XACT. An easy way to begin could be to reference a verification file (using test API) to registered interface and thus deriving the generation of the vSW code from it.

### Glossary

**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

<sup>1</sup> PCD : Prove of Concept Design

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**

**NXP** is an independent semiconductor company with a fifty-year history of providing engineers and designers with semiconductors and software that deliver better sensory experiences for mobile communications, consumer electronics, security applications, contactless payment and connectivity, and in-car entertainment and networking. Building on its heritage in consumer research, significant R&D investment and world-class industry partners, NXP's 'vibrant media technologies' allow consumers to enjoy better sensory experiences – brilliant images, crisp clear sound and easy sharing of information in homes, cars and mobile devices.

More information on <http://www.nxp.com>

**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 <http://www.magillem.com>

## **References**

- [1] Bailey, B., Martin, G. and Piziali, A. 2007. ESL Design Verification. Morgan Kaufmann Publishers, 2007
- [2] M. Volkmann, S. Balacco (Venture Development Corporation) - ESL Tools: Global Market Demand Analysis – [www.vdc-corp.com](http://www.vdc-corp.com) - June 2007
- [3] G. Martin, Chief Scientist (Tensilica, Inc.) - Barriers to ESL Adoption – [www.soccentral.com](http://www.soccentral.com) - May 2007
- [4] R. Schutten (Synopsys, Inc) - Complexity and Software Drive ESL Solutions – [www.soccentral.com](http://www.soccentral.com) – February 2005
- [5] Byte Paradigm - Electronic system-level development: Finding the right mix of solutions for the right mix of Engineers - [www.byteparadigm.com](http://www.byteparadigm.com) - April 2005
- [6] G. Harper (Bluespec, Inc.) - Are You Building Your ESL Design Flow on Sand? – [www.soccentral.com](http://www.soccentral.com) - February 2005
- [7] R. Goering – Designers adopt ESL, but tools are lacking – [www.design-reuse.com](http://www.design-reuse.com) – June 2005
- [8] R. Goering – Tools missing as ESL rolls – [www.design-reuse.com](http://www.design-reuse.com) – June 2005