

Public




FP7-ICT-2009- 4 (247999)

COMPLEX

COdesign and power Management in PPlatform-based design space EXploration

Project Duration 2009-12-01 – 2012-11-30

Type IP

	WP no.	Deliverable no.	Lead participant
	WP5	D5.2.1	ECSI
Preliminary Report on Standardization and Dissemination Activities			
Prepared by	Adam Morawiec (ECSI), all		
Issued by	ECSI		
Document Number/Rev.	COMPLEX/ECSI/RD5.2.1/1.0		
Classification	COMPLEX Public		
Submission Date	2010-12-21		
Due Date	2010-11-30		
Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)			

© Copyright 2010 OFFIS e.V., STMicroelectronics srl., STMicroelectronics Beijing R&D Inc, Thales Communications SA, GMV Aerospace and Defence SA, SNPS BelgiumNV, ChipVision Design Systems AG, EDALab srl, Magillem Design Services SAS, Politecnico di Milano, Universidad de Cantabria, Politecnico di Torino, Interuniversitair Micro-Electronica Centrum vzw, European Electronic Chips & Systems design Initiative.

This document may be copied freely for use in the public domain. Sections of it may be copied provided that acknowledgement is given of this original work. No responsibility is assumed by COMPLEX or its members for any application or design, nor for any infringements of patents or rights of others which may result from the use of this document.

History of Changes

ED.	REV.	DATE	PAGES	REASON FOR CHANGES
AM	1.0	2010-12-21	24	Final additions and corrections

Contents

1	Introduction	4
1.1	Scope	4
1.2	Acronym List	4
1.3	Glossary	5
2	Dissemination Activities	6
2.1	Publications	6
2.1.1	FDL2010 - Forum on specification and Design Languages	6
2.2	Conferences	6
2.2.1	FDL2010 Publications & Poster Session.....	7
2.2.2	DATE 2011 Booth.....	9
2.2.3	Conference Papers & Publications	9
2.2.4	Presentations	11
2.3	Organisation of Open Workshops	12
2.3.1	ECSI Workshop: What Future to IP-XACT / IEEE 1685?	12
2.3.2	M-BED'2011, the 2nd Workshop on Model Based Engineering for Embedded Systems Design.....	13
2.3.3	Third Friday Workshop on Designing for Embedded Parallel Computing Platforms: Architectures, Design Tools, and Applications	14
2.3.4	DATE 2011 Exhibition Theatre Session: Early Timing and Power Information of Complex SoC Designs using Augmented Virtual Platforms	15
2.3.5	2nd Workshop on Parallel Programming and Run-Time Management Techniques for Many-Core Architectures (PARMA)	16
2.4	Specific Focused Dissemination Actions	16
2.4.1	HIFSuite Dissemination	16
2.4.2	SCNSL Dissemination.....	17
2.5	Project e-Platform.....	17
2.6	E-mailing	17
3	Standardization Roadmap	18
3.1	Standardization Bodies	18
3.1.1	OSCI	18
3.1.2	ACCLLERA	18
3.1.3	OMG UML	19
3.1.4	Other standardization bodies	19
3.2	Interfaces	19
3.3	Past and future contributions	19
3.3.1	IEEE P1666	19
3.3.2	OSCI	20
4	Summary	21
5	Conclusions	22
6	References	23
7	Annex 1: Proposal for SystemC Extension: sc_vector: A flexible container for modules, ports and channels.....	24

1 Introduction

This document summarises all dissemination and standardisation activities that were undertaken by the COMPLEX project consortium as a whole and by each individual partner of the consortium in the first year of the project execution.

1.1 Scope

This document corresponds to the output deliverable of tasks 5.2 “Standardisation” and 5.3 “Dissemination” for the work package 5 within the COMPLEX project (see DoW [1]).

This is a public document that describes the standardisation activities and the interface with the appropriate standardization bodies to manage the elaboration and review of standards proposals, and the interaction with the related Working Groups. This document is also devoted to the dissemination activities to ensure the visibility and awareness of the project results and to support the adoption of the project results to the maximum extent possible in the industry and research institutions.

This document is not intended for evolving along the COMPLEX project. As development of the technologies progresses and experience is gained from its use for industrial use cases, further refinements to the standardization and dissemination plans described in this document will be made by the project partners and reflected in scheduled reports in the Description of Work (see DoW, [1]).

1.2 Acronym List

The following table lists all the acronyms used along this document:

Table 1-1: Acronym list

Acronym	Definition
DoW	Description of Work
ECSI	Electronic Chips and System design Initiative
MDA	Model Driven Architecture
OSCI	Open SystemC Initiative
PDM	Platform Dependent Model
PIM	Platform Independent Model
PSM	Platform Specific Model
RTES	Real-Time Embedded System
TBC	To Be Completed
TBD	To Be Defined
UML	Unified Modelling Language

1.3 Glossary

The following table summarizes the most important concepts provided along this document:

Table 1-2: Definition list

Definition	Description
FDL	Forum on specification and Design Languages, and ECSI conference, www.ecsi.org/fdl
IP-XACT	Design data representation standard, maintained by Accellera organisation, previously within the SPIRIT Consortium
M-BED	Model Based Engineering for Embedded Systems Design, workshop organised by ECSI within ECSI MARTE Users' Group
MARTE	Modeling and Analysis of Real-Time and Embedded Systems
OCP-IP	Open Core Protocol International Partnership

2 Dissemination Activities

This section describes the dissemination paths of the exploitable results included in document [2] for the COMPLEX project. The groupings are as follows:

- Submission to standardization bodies (see Section 3 for a detailed description).
- Publication of project results and attendance to conferences.
- Organisation of open workshops and attendance to external workshops.
- Generation of electronic material and deployment of a web-based e-platform.
- Promotion and e-mailing of open results through the ECSI contact lists.

A summary table is provided at the end of this section. For completeness project results that will utilise multiple dissemination paths appear under multiple dissemination path categories.

2.1 Publications

2.1.1 FDL2010 - Forum on specification and Design Languages

The FDL2010 conference was organised by ECSI in Southampton, UK on September 14-16, 2010. A complete program is presented at www.ecsi.org/fdl [4].

COMPLEX Project presented several related papers at the conference:

- **Session ABD1, Tuesday, September 14, 14:15-15:45**
Mapping of Concurrent Object-Oriented Models to Extended Real-Time Task Networks Matthias Büker, Kim Grüttner and Philipp A. Hartmann (OFFIS Institute for Information Technology), Ingo Stierand (University of Oldenburg)
- **Session LBSD2, Tuesday, September 14, 16:15-17:45**
Towards an ESL Framework for Timing and Power Aware Rapid Prototyping of HW/SW Systems Kim Grüttner, Kai Hylla, and Sven Rosinger (OFFIS Institute for Information Technology), Wolfgang Nebel (Carl von Ossietzky University Oldenburg)
- **Session ABD+LBSD, Wednesday, September 15, 10:30-12:00**
Formal Support for Untimed SystemC Specifications: Application to High-level Synthesis Eugenio Villar, Fernando Herrera, and Victor Fernández (University of Cantabria)
- **Session ABD+LBSD, Wednesday, September 15, 10:30-12:00**
Modeling of Communication Infrastructure for Design-Space Exploration Franco Fummi (EDALab), Davide Quaglia (EDALab), Francesco Stefanni, and Giovanni Lovato (University of Verona)
- **Session UMES1, Thursday, September 16, 10:30-12:30**
Formal Foundations for MARTE-SystemC Interoperability Pablo Peñil, Fernando Herrera, and Eugenio Villar (University of Cantabria)

2.2 Conferences

The following COMPLEX project results disseminated as published specifications in order that other technology vendors and industrial embedded systems development organisations are able to create tools and platforms that exploit COMPLEX technologies.

2.2.1 FDL2010 Publications & Poster Session

In addition to the presentation of COMPLEX related papers (as presented in the previous session) a Poster Session has been organized at FDL to facilitate the exchange with conference participants.



A special page has been devoted to COMPLEX in the FDL2010 Final Program to announce all the activities related to the project at the conference:



COMPLEX Project at FDL 2010

The COMPLEX (COdesign and power Management in PLatform-based design space EXploration) consortium develops a new design environment for platform-based design-space exploration offering developers of next-generation mobile embedded systems a highly efficient design methodology and tool chain. The integrated environment allows iterative exploration and refinement of advanced applications to meet market requirements. The design technology in particular enables fast simulation and explores the use of different implementations at Electronic System Level (ESL) with up to bus-cycle accuracy at the earliest instant in the design cycle. The main objectives are:

- Highly efficient and productive design methodology and holistic framework for design space exploration of embedded HW/SW systems.
- Combination and augmentation of well-established ESL synthesis & analysis tools into a seamless design flow enabling performance & power aware virtual prototyping of the HW/SW system.
- Interfacing next-generation model-driven SW design approach and industry standard model-based design environments.
- Multi-objective co-exploration for assessing design quality and optimizing the system platform with respect to performance, power, and reliability metrics.
- Fast simulation and assessment of the platform at ESL with up to bus-cycle accuracy at the earliest instant in the design cycle.
- Optimization benefits from run-time mode adaptation techniques, such as dynamic power management or application adaptation to varying workloads.

Session ABD1, Tuesday, September 14, 14:15-15:45

Mapping of Concurrent Object-Oriented Models to Extended Real-Time Task Networks

Matthias B ker, Kim Gr ttner and Philipp A. Hartmann (OFFIS Institute for Information Technology), Ingo Stierand (University of Oldenburg)

Session LBSD2, Tuesday, September 14, 16:15-17:45

Towards an ESL Framework for Timing and Power Aware Rapid Prototyping of HW/SW Systems

Kim Gr ttner, Kai Hylla, and Sven Rosinger (OFFIS Institute for Information Technology), Wolfgang Nebel (Carl von Ossietzky University Oldenburg)

Session ABD+LBSD, Wednesday, September 15, 10:30-12:00

Formal Support for Untimed SystemC Specifications: Application to High-level Synthesis

Eugenio Villar, Fernando Herrera, and Victor Fern ndez (University of Cantabria)

Session ABD+LBSD, Wednesday, September 15, 10:30-12:00

Modeling of Communication Infrastructure for Design-Space Exploration

Franco Fummi, Davide Quaglia, Francesco Stefanni, and Giovanni Lovato (University of Verona)

Session UMES1, Thursday, September 16, 10:30-12:30

Formal Foundations for MARTE-SystemC Interoperability

Pablo Pe il, Fernando Herrera, and Eugenio Villar (University of Cantabria)

More information about the COMPLEX project can be found at <http://complex.offis.de>

A poster session will be organized at FDL to present COMPLEX objectives and current results.

The FDL public that was addressed by the COMPLEX presentation is composed of industry and academic representatives involved in various aspects of system specification and design: high-level graphical specifications with UML, C-based, SystemC modelling, assertion-based dynamic and formal verification, analog and mixed signal modelling.

Industry sectors represented at FDL are: system companies, SoC integrators, IP providers, EDA partners.

ESCUG (European SystemC Users' Group) at FDL2010

ECSI supported the organization of the ESCUG meeting organized in conjunction with the conference. COMPLEX-related presentations has been given at this meeting by the project partners.

2.2.2 DATE 2011 Booth

The COMPLEX project initiated the organisation of the presence at the DATE Conference. There will be several ways in which COMPLEX will present its approach and results achieved:

- Booth in the European Projects exhibition area
- Presentation at the Proposal for DATE 2011 Exhibition Theatre Session (see description in the Section below):

Early Timing and Power Information of Complex SoC Designs using Augmented Virtual Platforms

- M-BED 2011 Workshop (see description in the Section below)
- Magillem company booth

2.2.3 Conference Papers & Publications

UML/MARTE specification and generation of SystemC, XML and IP/XACT

P. Peñil, F. Herrera, and E.Villar. Formal Foundations for MARTE-SystemC Interoperability. In Proceedings of FDL'2010. September, 2010.

F.Herrera, E. Villar. Generation of Abstract IP-XACT Platform Descriptions from UML/MARTE for System-Level Performance Estimation. In 2nd Workshop on Model Based Engineering for Embedded System Design. March, 18th. 2011.

Wireless Sensor Networks

P. Bellasi, A. Faisal, G. Serazzi, "Queueing Network Models for Performance Evaluation of ZigBee-based WSNs". 7th European Performance Engineering Workshop (EPEW), Bertinoro, 09/2010. pp.147-159.

C. Brandolese, L. Rucco, "A Genetic Approach for WSN Lifetime Maximization Through Dynamic Linking and Management," ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, (PE-WASUN'10), Bodrum, Turkey, October, 2010.

C. Brandolese, W. Fornaciari, L. Rucco, "A Lightweight Mechanism for Dynamic Linking in Wireless Sensor Networks," IEEE Latin America Symposium on Circuits and Systems, (LASCAS'10). Foz do Iguazu, Paranà, Brazil, February 2010.

Mult-OS performance modelling

H. Posadas, E. Villar, Dominique Ragot, M. Martínez . "Early Modeling of Linux-based RTOS Platforms in a SystemC Time-Approximate Co-Simulation Environment". IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC'10). 2010-05

Real-time multiprocessor scheduling

P. Bellasi, W. Betz, L. M. De Marchi, W. Fornaciari, "A Step Toward Exploiting Task Affinity in Multi-core Architectures to Improve Determinism of Real-time Applications". International Conference on Real-Time and Embedded Systems (RTES), Singapore, 11/2010.

P. A. Hartmann, K. Grüttner, A. Rettberg, I. Podolski; "*Distributed Resource-Aware Scheduling for Multi-Core Architectures with SystemC*"; 7th IFIP Conference on Distributed and Embedded Systems (DIPES'2010), Brisbane, Australia, 2010.

P. A. Hartmann, K. Grüttner, F. Oppenheimer, A. Rettberg; "*Exploiting Parallel Computing Platforms with OSSS*", presented at W3: Designing for Embedded Parallel Computing Platforms: Architectures, Design Tools, and Applications, co-located with DATE'2010, Dresden, Germany.

Source level power estimation of software

C. Brandolese, W. Fornaciari, D. P. Scarpazza, "Source-Level Energy Estimation and Optimization of Embedded Software," IEEE Latin America Symposium on Circuits and Systems, (LASCAS'10). Foz do Iguazu, Paranà, Brazil, February 2010.

Runtime management of resources

W. Fornaciari, P. Bellasi, "Cross-Layer Constrained Power Management: Application to a Multimedia Mobile Platform". IEEE Latin American Symposium on Circuits and Systems (LASCAS), IEEE Computer Society, 2010.

P. Bellasi, W. Fornaciari, D. Siorpaes, "A Hierarchical Distributed Control for Power and Performances Optimization of Embedded Systems". Conference on Architecture of Computing Systems (ARCS), Hannover, 02/2010, pp. 37-48.

P. Bellasi, S. Bosisio, M. Carnevali, W. Fornaciari, D. Siorpaes, "CPM: A Cross-Layer Framework to Efficiently Support Distributed Resources Management". Workshop on Parallel Programming and Run-time Management Techniques for Many-core Architectures (PARMA), Hannover, 02/2010, pp. 293-298.

P. Bellasi, W. Fornaciari, D. Siorpaes, "Constrained Power Management: Application to a Multimedia Mobile Platform". Conference on Design, Automation and Test in Europe (DATE), Dresden, 03/2010, IP, pp. 982-985.

P. Bellasi, S. Corbetta, W. Fornaciari. "*Hierarchical Power Management*", Poster session, Design and Automation Test in Europe (DATE), Dresden, Germany, March, 2010.

Andrea Calimera, Mirko Loghi, Enrico Macii, Massimo Poncino, "Dynamic Indexing: Concurrent Leakage and Aging Optimization for Caches", 2010 ACM International Symposium on Low-Power Electronics and Design, Austin, Texas, August 18-20 2010. pages 343-348.

The run-time management architecture, including its exploration framework for embedded multi-core platforms, and to be used as starting point for the IMEC tool, has been published and presented in July 2010 at the IC-SAMOS:

Ch. Ykman-Couvreur. "Exploration framework for run-time resource management of embedded multi-core platforms", IEEE Int. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation, Samos, July 2010.

HDL manipulation tools

N. Bombieri, F. Fummi, G. Pravadelli, "Automatic Abstraction of RTL IPs into Equivalent TLM Descriptions", IEEE Transactions on Computers. To appear in 2011.

N. Bombieri, G. Di Guglielmo, M. Ferrari, F. Fummi, G. Pravadelli, F. Stefanni, A. Venturelli, "HIFSuite: Tools for HDL Code Conversion and Manipulation", Eurasip Journal of Embedded Systems. To appear in 2011.

Bombieri N., Di Guglielmo G., Di Guglielmo L., Ferrari M., Fummi F., Pravadelli G., Stefanni F., Venturelli A., HIFSuite: Tools for HDL Code Conversion and Manipulation , Atti di "IEEE International High Level Design Validation and Test Workshop " , Anaheim Convention Center, Anaheim, CA , June 11-12, 2010 , 2010 , pp. 40-41

N. Bombieri, F. Fummi, V. Guarnieri, G. Pravadelli, S. Vinco, Redesign and Verification of RTL IPs through RTL-to-TLM Abstraction and TLM Synthesis , Atti di "IEEE International Workshop on Microprocessor Test and Verification (MTV)" , Austin, TX, USA , 13-15 December , 2010

Bombieri Nicola; Forrini Diego; Fummi Franco; Laurenzi Matteo; Vinco Sara, RTL IP abstraction into optimized embedded software , Atti di "East-West Design and Test Symposium" , St. Petersburg , 17-20 settembre 2010 , 2010 , pp. 64-68

K. Grüttner, H. Kleen, F. Oppenheimer, A. Rettberg, W. Nebel; "*Towards a Synthesis Semantics for SystemC Channels*", International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS'2010), Scottsdale, AZ, USA, October 2010.

2.2.4 Presentations

Wednesday, Dec. 16th 2009 from 4.00 pm to 5.30 pm

Princeton University - Department of Electrical Engineering, HOST: Ruby LEE

Title: System-wide run-time resource management for energy optimization

Speaker: William Fornaciari, Politecnico di Milano

Title: "Automatic Design Space Exploration for Chip-Multi Processors"
Speaker: Cristina Silvano, Politecnico di Milano

Thursday, Dec. 17th 2009 from 10.30am to 12.00 am
NEC Laboratories America, HOST: Marcello Lajolo

Title: System-wide run-time resource management for energy optimization
Speaker: William Fornaciari, Politecnico di Milano
Title: "Automatic Design Space Exploration for Chip-Multi Processors"
Speaker: Cristina Silvano, Politecnico di Milano

Thursday, Oct. 28, 2010, 3pm
Engineering Hall 2111, University of California, Irvine, USA

Title: "Towards an ESL Framework for Timing and Power Aware Rapid Prototyping of HW/SW Systems"
Speaker: Kim Grüttner, OFFIS

2.3 Organisation of Open Workshops

The following workshops will be organized at the European Level by ECSI, inviting partners to present and disseminate specific project results to broad group of researches, system designers and managers. These workshops are focused on potential stakeholders on specific technology or standardization bodies.

2.3.1 ECSI Workshop: What Future to IP-XACT / IEEE 1685?

Web page: www.ecsi.org/ip-xact2010
Date & Place: November 29, 2010 - Grenoble, France
Organization: ECSI

The workshop was organized by ECSI on November 29, 2010 in Grenoble, France in conjunction with IP-SOC 2010. For the COMPLEX Project that objective was to assess the current evolution of the IP-XACT standard, the future requirements and industry needs. It offered also an opportunity to evaluate the current usage of IP-XACT and the real industry support.

Outline

In the last years industry has spent a significant effort in developing and making evolve the IP-XACT standard to incorporate relevant features. This standard has been used successfully by large SoC providers and system houses into production flows and its usage solves critical issues: standardizing the management of heterogeneous IP portfolios, supporting growing complexity in RTL design assembly, providing a backbone for ESL methodologies. IP-XACT is now recognized as IEEE 1685 and undisputed but cannot sit on its laurels. Thus it is crucial today to take a careful look into strategic evolutions required to take in account system aspects like TLM or HW/SW interfaces, power analysis, early prototyping, AMS, verification, debug, etc. The question still is where are the priorities and major needs? Beyond the technical questions, what are the business reasons to invest into the adoption of IP-XACT? What is the business mode for system, SoC and eventually EDA companies?

The workshop gathered the representatives from all industry sectors: System companies, SoC integrators, EDA providers, IP providers, standardization bodies, research: ARM LTD, Atmel, Docea Power, Duolog, ECSI, ELDA Technology, ISLI, Magillem, Robert Bosch, Silicon-IP, inc., Sonics, Inc., ST-Ericsson, STMicroelectronics, Texas Instruments, TIMA, Université de Bourgogne, University of Frankfurt, Institute for Computer Science, Wind River

The complete programme and set of presentations is available from: www.ecsi.org/ip-xact2010

2.3.2 M-BED'2011, the 2nd Workshop on Model Based Engineering for Embedded Systems Design

Web page: <http://www.ecsi.org/m-bed>

Date & Place: March 2011, Friday DATE Workshop – Grenoble, France

Organization: ECSI

Description: The application of model-based engineering (MBE) methods for software and systems development in industry is increasing. Moreover, the integration of component-based approaches with MBE has further accelerated its adoption along is also providing a basis for a sounder theoretical underpinning.

The focus of this workshop is on the use of MBE for embedded systems development (e.g. in the industrial transport sector for applications such as railway systems, automotive, aerospace, and related domains). In this context, special focus is given to MARTE, the UML profile for Modelling and Analysis of Real-Time and Embedded systems, which has proven successful in a number of projects. In particular, the program concentrates on the following topics:

- *The infrastructure that supports MBE*, that is, the requisite languages, tools, and standards, as well as the combination of design and V&V activities, and the diverse engineering disciplines involved in embedded system design.
- *Process and methodology related issues*, such as guidelines for deciding when and how to use domain-specific languages, appropriate integration of tools, and advanced methods to assist on architecture exploration subjected to multiple non functional constraints.
- *Experience with applying MARTE* and suggestions of improvements to this standard.

The aim of the workshop is to bring together researchers as well as system designers and tool developers from both industry and academia to discuss applications of model-based engineering in general and MARTE usage in particular. A significant portion of time will be reserved for discussion.

Complementing the accepted paper presentations will be several invited presentations by members of the MARTE standardization task force, specialists participating in relevant European projects, and representatives of MBE tool vendors. The one-day workshop is organized in multiple sessions, each focusing on a particular topic. Rather than have questions at the end of each presentation, all discussion will be conducted at the end of the session, with all presenters in the session responding as a group to questions of the session moderator as well as other attendees.

Organisers:

Pierre Boulet, Univ. Lille, FR

Daniela Cancila, Sherpa Engineering, FR

Sébastien Gérard, CEA-LIST, FR
Adam Morawiec, ECSI, FR
Chokri Mraihda, CEA-LIST, FR
Laurent Rioux, Thales RT, FR
Bran Selic, Malina Software Corp., CA

In this workshop some preliminary results from UC side will be presented regarding the IP-XACT generation from UML/MARTE. Specifically, the following paper will be presented:

F.Herrera, E. Villar. Generation of Abstract IP-XACT Platform Descriptions from UML/MARTE for System-Level Performance Estimation. In 2nd Workshop on Model Based Engineering for Embedded System Design. March, 18th. 2011.

2.3.3 Third Friday Workshop on Designing for Embedded Parallel Computing Platforms: Architectures, Design Tools, and Applications

Web site: <http://conferenze.dei.polimi.it/depcp/>

Date & Place: March 18, 2011, Grenoble, France

Co-located with DATE'11 (Friday Workshop W3),

Organization: POLIMI

The future of embedded computing is shifting to multi/many-core designs to boost performance due to the unacceptable power consumption and operating temperature increase of fast single-core CPUs. Hence embedded system designers are increasingly faced with the following new big challenges: the support for a variety of concurrent applications, and the platform heterogeneity. These challenges lead to the following significant issues:

- How can we write applications that exploit the underlying (parallel) architecture, without burdening the application designer?
- What does the application designer really need to know of the underlying architecture?
- What tools are needed to efficiently map applications and what part of the process should be automated?
- How should we design the underlying architectures?

This workshop brings together researchers actively working on architectures, design tools, and applications for embedded parallel computing platforms to address these questions and related issues.

Topic Areas:

The workshop will have three main sessions:

- Architectures: on the most relevant problems arising during the design exploration and optimization of many/multi core architectures.
- Design tools: on the state-of-the-art of tool development, showing where we are now and the directions we need to move in.
- Applications: on the analysis, development, modification and integration of applications with respect to parallel computing platforms.

Organisation:

General Co-Chairs: Cristina Silvano and Giovanni Agosta, Politecnico di Milano, Italy

Architectures Session Chair: Maurizio Palesi, KORE University, Italy

Design Tools Session Chair: Chantal Ykman-Couvreur, IMEC, Belgium

Applications Session Chair: Diana Göhringer, Fraunhofer IOSB, Germany

Poster Session Chair: Michael Hübner, Karlsruhe Institute of Technology, Germany
Panel Session Co-Chairs: Jürgen Becker and Michael Hübner, Karlsruhe
Inst. of Technology, Germany
Web and Posters Submission Chair: Fabrizio Castro, Politecnico di Milano, Italy

This workshop organized by POLIMI with the support also of IMEC has the objective to create networking around some of the project activities and to spread-out the knowledge especially focused on design space exploration, programmability, software optimization and estimation and run-time management issues for embedded systems. In this workshop POLIMI will present research and development activities related to COMPLEX.

2.3.4 DATE 2011 Exhibition Theatre Session: Early Timing and Power Information of Complex SoC Designs using Augmented Virtual Platforms

Web-site: <http://www.date-conference.com/>
Date: Preliminary scheduled: 17.03.2011, 12:45-13:45
Organization: OFFIS

Rationale and Abstract

Consideration of an embedded system's timing behaviour and power consumption at system-level is an ambitious task. Sophisticated tools and techniques exist for power and timing estimations of individual components such as custom hard and software as well as IP components. But prediction of the composed system behaviour can hardly be made. In this session we present the concept of an ESL framework for timing and power aware rapid virtual system prototyping of embedded HW/SW systems. Our proposed flow combines system-level timing and power estimation techniques available in commercial tools with platform-based rapid prototyping. Our proposal aims at the generation of executable virtual prototypes from a functional C/C++ specification. These prototypes are enriched by static and dynamic power values as well as execution times. They allow a trade-off between different platforms, mapping alternatives, and optimization techniques, based on domain-specific workload scenarios. The proposed flow will be implemented in the COMPLEX FP7 European integrated project (<http://complex.offis.de>).

Scheduled Presentations:

- The COMPLEX ESL Framework for Timing and Power Aware Rapid Prototyping of HW/SW Systems (Kim Grüttner - OFFIS)
- Using Virtual Platforms for Energy Efficient SW-Design (Bart Vanthournout - Synopsys)
- Automatic Abstraction of RTL IPs into Equivalent TLM Descriptions for Platform Simulation (Franco Fummi - EDALab)
- Choosing IP-XACT IEEE 1685 standard as a unified description for timing and power performance estimations in virtual platforms (Emmanuel Vaumorin – Magillem Design Services)

2.3.5 2nd Workshop on Parallel Programming and Run-Time Management Techniques for Many-Core Architectures (PARMA)

Web site: <http://conferences.microlab.ntua.gr/parma2011>

Date & Place: Feb. 23, 2011 - Lake Como, Italy

Co-located with ARCS 2011 - Architecture of Computing Systems

<http://conferences.dei.polimi.it/arcs2011>

Organisation: POLIMI

Scope

The current trend towards many-core architecture requires a global rethinking of software and hardware design approaches. The PARMA workshop focuses on parallel programming models, design space exploration and run-time resource management techniques to exploit the features of many-core processor architectures.

The PARMA workshop is composed of three main sessions:

- S1: Programming models and languages, compilers and virtualization techniques
- S2: Runtime management, power management and memory management
- S3: Design space exploration and many-core architecture customization

Organizing Committee

Workshop General Co-Chairs

Dimitrios Soudris, National Technical University of Athens, Greece

Giovanni Agosta, Politecnico di Milano, Italy

Session Chairs

S1: Torsten Kempf, Institute for Integrated Signal Processing Systems (ISS), RWTH Aachen University, Germany

S2: Gianluca Palermo, Politecnico di Milano, Italy

S3: Benno Stabernack, Heinrich Hertz Institute, Fraunhofer Institute, Germany

This workshop organized by POLIMI has the objective to create networking around some of the project activities and to spread-out the knowledge especially focused on design space exploration, programmability, software optimization and estimation and run-time management issues for embedded systems. In this workshop POLIMI will present research and development activities related to COMPLEX

2.4 Specific Focused Dissemination Actions

2.4.1 HIFSuite Dissemination

EDALab is particularly promoting HIFSuite, upon which some COMPLEX tools rely, through a massive marketing campaign started at the DATE 2010 University Booth. A new ad hoc web site (<http://hifsuite.edalab.it>) has been set up at the beginning of March 2010. Then, a demo version of HIFSuite 3.4 has been launched on June 11th 2010 in occasion of the presentation of HIFSuite in the Industrial Practice Session of the IEEE High-Level Design Validation and Test Workshop 2010.

A mailing campaign has been launched to promote HIFSuite among research groups all over the world.

2.4.2 SCNSL Dissemination

An open source project has been created on SourceForge website (<http://scnsl.sourceforge.net>) to increase sharing of ideas and cooperative development.

2.5 Project e-Platform

The project web portal will provide access to the project results at three different levels:

- Public level: includes public information, presentations and open results.
- Restricted level: includes information like deliverables and presentations for project reviews, accessible only to project members and European Commission (EC) reviewers and/or jointly agreed additional companies out of the project.
- Confidential level: contains all partial and final results, working documents, presentations and articles.

The project web portal can be found at <https://complex.offis.de/>.

Additionally the project will provide access to an open source repository for the open source tools developed within the project. This plan needs to be completed and confirmed in the next stage of the project.

2.6 E-mailing

The announcements of events and availability of open results will be largely disseminated on ECSI's email lists for system design/embedded systems (currently containing more than 22000 entries) and other companies' lists.

3 Standardization Roadmap

This section describes the standardization bodies, tasks and responsibilities necessary to coordinate all standardization activities of the project in a consistent and coherent way. These activities encompass the following sub-activities:

- Elaboration of standard proposals (definitions, recommended practices, requirements, example models, etc.)
- Internal review of proposals, including the nomination of reviewers, planning of review meetings and control of received comments.
- Participation in relevant Working Groups closely related with standardization bodies.
- Interaction between the project and standardization bodies.

3.1 Standardization Bodies

The following standardization bodies are identified in the context of the COMPLEX project:

3.1.1 OSCI

COMPLEX will provide inputs to the standardisation of SystemC/TLM. Project partners will directly participate in relevant OSCI Working Groups, namely the Language Working Group (LWG), Transaction-Level Modelling Working Group (TLMWG), and the Control Configuration and Introspection Working Group (CCIWG).

More information available at: <http://www.systemc.org/>

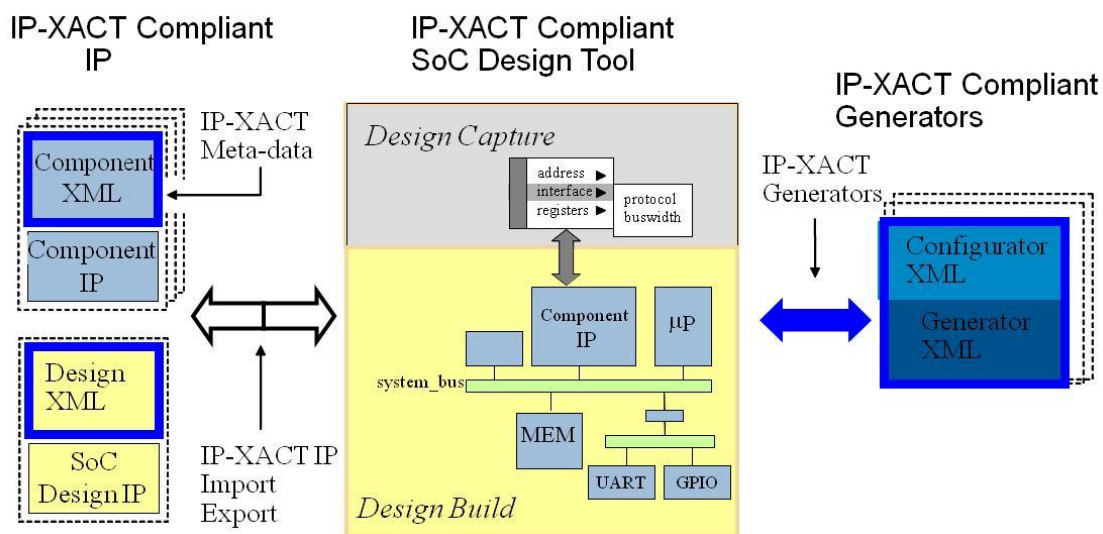
The COMPLEX work on SystemC concerns also the IEEE1666 standardization.

3.1.2 ACCLERA

COMPLEX will provide inputs to the IP-XACT standardisation working group of Accellera. Feedback and improvements proposals for existing versions of the standard (e.g. IP-XACT 1.4 and 1.5) as well as requirements for upcoming versions of IP-XACT will be provided. Project partners will directly participate in relevant IP-XACT Working Groups: Extensions, Registers.

More information available at: <http://www.accellera.org/>

Below a typical IP-XACT based design flow is presented.



Among the current and future topics of evolution Accellera enumerates specific extensions (new area) development for:

- Power analysis (IP Power model, Power domains)
- Early-prototyping (Area / Timing / Power expressions)
- Verification and Debug (Testbench, systemVerilog)
- AMS

where power domain has a privileged priority.

3.1.3 OMG UML

COMPLEX will provide inputs to the OMG for UML profile MARTE standardisation and evolution. Project partner involved in the definition of the COMPLEX design entry based on the UML/MARTE modelling language analyzed current MARTE specification. During the use case implementation they will provide any possible feedback to the OMG organization in relevant aspects related to the design of Real-Time Embedded Systems (RTES) that are not currently covered by the standard but are important for COMPLEX objectives.

More information available at: <http://www.omgmarte.org/>

3.1.4 Other standardization bodies

Relations and provision of relevant input to other standardisation organisations: e.g. OCP-IP, IEEE DASC, IEEE SA, or other.

OCP-IP related work to IP-XACT is of a special interest for the COMPLEX Project. A Meta-Data WG of OCP-IP is the target working group that can accommodate inputs from COMPLEX. An initial discussion has been carried out between ECSI and OCP-IP (Ian Mackintosh, the OCP President) on the ways of cooperation.

3.2 Interfaces

The following table lists the interface responsible and the different contributors for the transfer of input and interaction between the project and the standardization bodies. The list of standardization bodies is tentative and might be completed with additional organization in future reports.

Table 3-1 Standardization interfaces

Standardization body	Standard	Interface	Contributors
OSCI	SystemC	Synopsys, ST-I, OFFIS	Synopsys, ST-I, OFFIS
ACCELLERA	IP-XACT	MDS, ECSI	Synopsys
OMG	UML/MARTE profile	Thales	GMV, UC
OCP-IP	OCP	ECSI	MDS, Synopsys

3.3 Past and future contributions

This section includes all the contributions provided to the different standardization bodies.

3.3.1 IEEE P1666

The IEEE working group P1666 is currently working on the next revision of the IEEE Standard 1666 for the SystemC language. As voting members of IEEE-SA, SNPS and ST can have substantial on the approval of new and improved features of SystemC/TLM.

In addition to the proposal of minor enhancements and clarifications, diligent review of proposals, and an active participation in the discussion in the different forums of the P1666 working group, an extended proposal for a new container class `sc_vector` for modules, ports, and other objects has been proposed by OFFIS. The detailed proposal can be found in the Annex of this document. The `sc_vector` has been approved by the Working Group and will be included in the next revision, if the balloting process permits.

Detailed information, including meeting minutes, proposal details and an archive of the e-mail based discussions during the preparation of the next version of the IEEE 1666 standard can be found on the IEEE P1666 website at <http://www.eda.org/systemc>.

3.3.2 OSCI

In addition to the contributions to the IEEE WG P1666, several COMPLEX partners are actively contributing to the on-going evolution of the SystemC eco-system within the Open SystemC Initiative. This includes activities in the relevant OSCI working groups LWG, TLMWG, and CCIWG.

Alongside with the proposed addition of the `sc_vector` class to the SystemC standard, a proof-of-concept implementation has been contributed by OFFIS to the OSCI reference implementation of SystemC. Several other proposals from P1666 have been implemented and contributed as well, most notably

- `sc_process_handle` extensions for compatibility with standard C++ containers (OFFIS)
- `sc_event_and/or_list`, and `sc_event_and/or_expr` for extended event sensitivity (OFFIS)
- `sc_signal` → `sc_writer_policy` safely enabling multiple writers (OFFIS)
- virtual `bind()` methods for ports (OFFIS)
- addition of `float` support for fixed-point types (SNPS)

Within the OSCI LWG, the inclusion of features that have been rejected or not yet considered by the IEEE P1666 WG for the next revision of the SystemC standard (IEEE 1666-2011) are discussed, and COMPLEX partners are actively contributing to resolve the open issues and stabilize these features for industry adoption. The most important challenge in that regard is the extension of the SystemC core language for exploiting multi-core parallelism of the simulator platform. This can lead to increased simulation performance, which is a core requirement of today's large-scale virtual platform models, and therefore directly in the scope of the COMPLEX project.

The current work of the CCIWG to define a standardised configuration API is closely followed by the COMPLEX consortium and considered as a future basis for virtual platform configuration. The identified requirements of the COMPLEX framework are reviewed and potential conflicts are communicated back to the CCIWG. In the further course of the COMPLEX project, additional contributions of (parts of) the COMPLEX framework to appropriate OSCI working groups are planned.

Another contribution to OSCI is the standardization of the SystemC Network Simulation library (SCNSL) to introduce simulation of packet-based networks in SystemC simulation (by EDALabs).

4 Summary

The following lists the exploitable results expected from the project and current plans for how these results will be disseminated to benefit the embedded systems development community.

Table 4-1: Dissemination Planning Summary

Dissemination Planning Summary	
Exploitable Project Result	Dissemination Path Category
Inconsistencies and improvements in the specification of the UML/MARTE profile. IP-XACT improvements SystemC/TLM improvements	Standardization Open Workshops on Standards / Standard Evolutions
COMPLEX MDA methodology and toolset architecture	Publications and Conferences Exhibition Booths Demos
Training material Project results presentations Demonstrations	Open Workshops Dedicated training sessions Tutorials Demonstration Sessions Conference Booth Presentations
COMPLEX methodology and toolset architecture Transformation tool engines	Project E-Platform
Open documentation	E-Mailing Project web pages ECSI web page

5 Conclusions

The COMPLEX project results will substantially address many of the challenges facing embedded systems developers today in exploiting model-driven development methods and state-of-the-art power and timing estimation techniques in the development of embedded systems, especially in finding the optimal balance between performance and power consumption.

Overall, the pillars that support the dissemination planning for the project are the following:

Publish the profiles and interfaces used for the COMPLEX tools and technologies to encourage further improvements on current tools or new ones to be developed.

Submit to standards bodies the extensions developed within the COMPLEX project to existing standards such as MARTE, IP-XACT, SystemC/TLM.

Make the development tools and training material available using the well-established dissemination channels of the project partners (i.e. project web-page, (potentially) source code repository).

These coupled with the publication of technical papers and presentations at various conferences will allow ensuring the visibility and awareness of COMPLEX results to a wide range of industrial organisations in Europe and abroad to utilise and exploit for their own interests and benefit the results from the COMPLEX project.

6 References

- [1] "Description of Work". COMPLEX – COdesign and power Management in Platform-based design-space EXploration, FP7-ICT-2009- 4 (247999), 2009.
- [2] D5.4.1 "Preliminary Exploitation Plan". COMPLEX – COdesign and power Management in Platform-based design-space Exploration, 2010.
- [3] D5.3.1 "Plan for Dissemination to the Public". COMPLEX, 2010
- [4] FDL2010 Final Program document, www.ecsi.org/fdl

7 Annex 1: Proposal for SystemC Extension: sc_vector: A flexible container for modules, ports and channels

sc_vector: A flexible container for modules, ports and channels

Contents

1	Motivation	1
2	sc_vector	2
2.1	Description	2
2.2	Class definition	2
2.3	Template parameters	4
2.4	Constraints on usage	4
2.5	Constructors and initialisation, destructor	4
2.6	kind, size, get_elements	6
2.7	Element access	6
2.8	Iterators, begin, end	7
2.9	Binding of vectors	7
3	Looking at vector element's members — sc_vector_view	8
3.1	Class definition	8
4	Proof-of-concept implementation	9
5	TLM 2.0 — a simple router example	10
6	Further comments	12
7	Acknowledgements	12
8	Contact information	12

1 Motivation

In many designs, a parametrisable number of modules, channels, ports, or other SystemC™ objects are used. Since such SystemC objects are usually named entities in the hierarchy, it is either required (in case of modules), or at least desired to pass a name parameter to the constructor of such objects.

If one wants to create a collection of such named entities in C++/SystemC, it is usually needed to use an array/vector of pointers and to allocate the elements explicitly in a loop, since plain array values have to be created via the default constructor in C++/SystemC. Access to the elements then requires an additional dereferencing operation. This is quite inconvenient and inconsistent with regular class members.

Example 1.1 Current situation

```
SC_MODULE( sub_module ) { /* ... */ };

SC_MODULE( module )
{
    // vector of (pointers to) sub-modules
    std::vector< sub_module* > m_sub_vec;

    // dynamically allocated C-style array of ports
    sc_core::sc_in<bool>*      in_vec;

    module( sc_core::sc_module_name, unsigned n_sub )
    {
        // create sub-modules in a loop
        for( unsigned i=0; i<n_sub; ++i )
        {
            std::stringstream name;
            name << "sub_modules_" << i;
            m_sub_vec.push_back( new sub_module( name.str().c_str() ) );
        }

        // create (default named) array of ports
        in_vec = new sc_core::sc_in<bool>[ n_sub ];

        // bind ports of sub-modules -- requires dereference
        for( unsigned i=0; i<n_sub; ++i )
        {
            m_sub_vec[i]->in( in_vec[i] );
            // or (*m_sub_vec[i]).in( in_vec[i] );
        }
    }
    ~module()
    {
        // manual cleanup
        for( unsigned i=0; i<m_sub_vec.size(); ++i )
            delete m_sub_vec[i];
        delete [] in_vec;
    }
};
```

Frequent questions in the public SystemC discussion forums as well as experience from SystemC teaching courses have shown this to be difficult, especially for inexperienced C++/SystemC users.

This proposal aims to provide a convenience container called `sc_core::sc_vector<T>` for such objects directly within the SystemC standard to lift this burden.

With the proposed convenience class, Example 1.1 could be written as shown in the following Example 1.2. Note the avoidance of manually crafting names and the automatically handled memory management.

Example 1.2 Possible future situation

```
SC_MODULE( sub_module ) { /* ... */ };

SC_MODULE( module )
{
    // vector of sub-modules
    sc_core::sc_vector< sub_module > m_sub_vec;

    // vector of ports
    sc_core::sc_vector< sc_core::sc_in<bool> > in_vec;

    module( sc_core::sc_module_name, unsigned n_sub )
        : m_sub_vec( "sub_modules", n_sub ) // set name prefix, and create sub-modules
          // , in_vec() // use default constructor
          // , in_vec( "in_vec" ) // set name prefix
    {
        // delayed initialisation of port vector
        // here with default prefix sc_core::sc_gen_unique_name("vector")
        in_vec.init( n_sub );

        // bind ports of sub-modules -- no dereference
        for( unsigned i=0; i<n_sub; ++i )
            m_sub_vec[i].in( in_vec[i] );

        // or, with the provided vector-bind interfaces
        sc_view( m_sub_vec, &sub_module::in ).bind( in_vec );
    }
};
```

2 sc_vector



Important

The wording in this section may not yet be formal enough for verbatim inclusion in the IEEE 1666 standard.

2.1 Description

Utility class `sc_vector` allows to create vectors of SystemC objects, that usually require a name parameter in their constructor. It provides array-like access to the members of the vector and manages the allocated resources automatically. Once the size is determined and the elements are initialised, further resizing operations are not supported. Custom constructor signatures are supported by a template `init` function, supporting user-defined element allocations.

2.2 Class definition

```
namespace sc_core {

class sc_vector_base : public sc_object
{
public:
    const char * kind() const;
    size_type size() const;
    const std::vector<sc_object*>& get_elements() const;
};
```

```

};

template< typename T >
class sc_vector_iter : public std::iterator< std::random_access_iterator_tag, T >
{
    // implementation-defined, but conforming to Random Access Iterator category,
    // see ISO/IEC 14882:2003(E), 24.1 [lib.iterator.requirements]
};

// member-wise view of a vector (see below)
template< typename T, typename MemberType > class sc_vector_view;

template< typename T >
class sc_vector : public sc_vector_base
{
public:
    typedef T element_type;
    typedef /* implementation-defined */ size_type;
    typedef sc_vector_iter< element_type > iterator;
    typedef sc_vector_iter< const element_type > const_iterator;

    sc_vector();
    explicit sc_vector( const char* prefix );
    sc_vector( const char* prefix, size_type n );

    template< typename Creator >
    sc_vector( const char* prefix, size_type n, Creator c );

    virtual ~sc_vector();

    void init( size_type n );

    template< typename Creator >
    void init( size_type n, Creator c );

    static element_type * create_element( const char* prefix, size_type index );

    // ----- element access -----

    element_type& operator[]( size_type i );
    element_type& at( size_type i );

    const element_type& operator[]( size_type i ) const;
    const element_type& at( size_type i ) const;

    // ----- iterator access -----

    iterator begin();
    iterator end();

    const_iterator begin() const;
    const_iterator end() const;

    const_iterator cbegin() const;
    const_iterator cend() const;

    // ----- binding interface -----

    template< typename ContainerType, typename ArgumentType >
    iterator bind( sc_vector_view<ContainerType,ArgumentType> c );

    template< typename BindableContainer >

```

```
iterator bind( BindableContainer & c );

template< typename BindableIterator >
iterator bind( BindableIterator first, BindableIterator last );

template< typename BindableIterator >
iterator bind( BindableIterator first, BindableIterator last, iterator from );

template< typename ContainerType, typename ArgumentType >
iterator operator()( sc_vector_view<ContainerType,ArgumentType> c );

template< typename ArgumentContainer >
iterator operator()( ArgumentContainer & c );

template< typename ArgumentIterator >
iterator operator()( ArgumentIterator first, ArgumentIterator last );

template< typename ArgumentIterator >
iterator operator()( ArgumentIterator first, ArgumentIterator last, iterator from );

private:
    // disabled
    sc_vector( const sc_vector& );
    sc_vector& operator=( const sc_vector& );
};
} // namespace sc_core
```

2.3 Template parameters

The template parameter passed as an argument `T` to `sc_vector` shall be derived from the class `sc_object`. If not used with a custom `Creator` functor, `T` shall provide a constructor with an argument of a type convertible from `const char*`.

NOTE— In case of plain C++ types, the use of standard C++ containers like `std::vector` is recommended.

NOTE— For the constraints on the arguments to the template parameter `Creator`, passed to the templated constructor and `init` function, see Section 2.5.

2.4 Constraints on usage

An implementation shall derive class `sc_vector_base` from class `sc_object`. This implementation-defined base class provides type-agnostic access to the `sc_object` elements contained in the vector.

Objects of class `sc_vector` can only be constructed during elaboration, if the element type can only be instantiated during elaboration (like modules, ports, channels). Otherwise, a vector can be instantiated during simulation as well.

A vector shall not introduce an additional level in the object hierarchy. If the vector contains instances of classes derived from `sc_object`, such objects shall be siblings of the vector instance itself.

2.5 Constructors and initialisation, destructor

```
sc_vector();
explicit sc_vector( const char* prefix );
sc_vector( const char* prefix, size_type n );
```

The constructors for class `sc_vector` shall pass the character string argument (if such argument exists) through to the constructor belonging to the base class `sc_object` to set the string name of the vector instance in the module hierarchy.

The default constructor shall call function `sc_gen_unique_name("vector")` to generate a unique string name that it shall then pass through to the constructor for the base class `sc_object`.

An optional argument `n` of the unsigned integral type `size_type` shall be used to determine whether a default initialisation with `n` objects of type `element_type` shall be performed. If the value of `n` is zero, no such initialisation shall occur, otherwise the function `init` shall be called argument `n` within the constructor.

```
void init( size_type n );  
static void create_element( const char* name, size_type index );
```

Calling the `init` function shall fill the vector with `n` instances of `element_type`, that are allocated by the static function `create_element`. It shall be an error to call the `init` function more than once on the same vector, or to call it after the elaboration has finished.

The `create_element` function is the default allocation function for vector elements. Within this function, an instance of `element_type` shall be allocated via operator `new` and the argument `name` shall be passed to the constructor.

The `name` argument shall be constructed within the `init` function by appending the vector's `basename` with the current `index`, separated by an underscore.

NOTE—The `init` function can be used to decouple the element creation from the construction of the vector instance itself, e.g. during `before_end_of_elaboration`.

```
template< typename Creator >  
sc_vector( const char* prefix, size_type n, Creator c );  
template< typename Creator >  
void init( size_type n, Creator c );
```

The templated versions of the constructor and the `init` function shall use a value of type `Creator` that has been passed as argument for the allocation of the element instances. Instead of calling `create_element` with arguments `name` and `index` to allocate each element instance of the vector, the expression

```
element_type * next = c( name, index );
```

shall be well-formed for an argument `c` of template parameter type `Creator`.

The expressions `v.init(n, sc_vector<T>::create_element)` and `v.init(n)` shall be equivalent for all vectors `sc_vector<T> v`.

NOTE—A frequent use case for custom `Creator` arguments are additional, required constructor parameters of the contained `element_type`. `Creator` can then either be a function pointer or a function object (providing an appropriate `operator()(const char*, size_t)`).

Example

```
SC_MODULE( sub_module )  
{  
    // constructor with additional parameters  
    sub_module( sc_core::sc_module_name, int param );  
    // ...  
};  
  
SC_MODULE( module )  
{  
    sc_core::sc_vector< sub_module > sub_vec1, sub_vec2; // vector of sub-modules  
  
    struct mod_creator // Creator struct  
    {  
        mod_creator( int p ) : param(p) {} // store parameter to forward  
    };  
};
```

```

sub_module* operator()( const char* name, size_t ) // actual creator function
{
    return new sub_module( name, param );          // forward param to sub-module
}
int param;
};

// creation via member function
sub_module* another_creator( const char* name, size_t id )
{ return new sub_module( name, id ); }

module( sc_core::sc_module_name, unsigned n_sub )
: sub_vec1( "sub_modules" )                       // set name prefix
{
    sub_vec1.init( n_sub, mod_creator(42) );       // init with custom creator

    // or init via sc_bind and local member function
    sub_vec2.init( n_sub, sc_bind( &module::another_creator
                                   , this, sc_unamed::_1, 42 ) );
}
};

```

```
virtual ~sc_vector();
```

During destruction of the vector, all elements held by the vector shall be destroyed by calling `operator delete` on their addresses.

2.6 kind, size, get_elements

```

const char * kind() const;
size_type size() const;
const std::vector<sc_object*>& get_elements() const;

```

Member function `kind` shall return the string `"sc_vector"`.

Member function `size` shall return the number of initialised objects of the vector.

Member function `get_elements` shall return a `const`-reference to a `std::vector` containing pointers to the contained elements. If the `sc_vector` object has not yet been initialised, the returned `std::vector` shall be empty. The returned reference shall be valid for the lifetime of the `sc_vector` object.

NOTE—The relation `v.get_elements().size() == v.size()` holds at all times.

2.7 Element access

```

element_type& operator[]( size_type i );
element_type& at( size_type i );

const element_type& operator[]( size_type i ) const;
const element_type& at( size_type i ) const;

```

`operator[]` and member function `at` shall return a (const qualified) reference to the object stored at index `i`.

If the given index argument exceeds the `size` of the vector, the behaviour is undefined in case of `operator[]`. In case of member function `at`, the implementation shall detect invalid indices as an error.

It is undefined, whether the relation `&v[i]+j == &v[i+j]` holds for any vector `v` and indices `i, j`.

References returned by functions defined in this clause shall be valid until the lifetime of the surrounding vector has ended.

2.8 Iterators, begin, end

For compatibility with the C++ Standard Library, `sc_vector` shall provide an iterator interface, that fulfils the *Random Access Iterator* requirements as defined in ISO/IEC 14882:2003(E), Clause 24.1 [lib.iterator.requirements].

```
iterator begin();
iterator end();

const_iterator begin() const;
const_iterator end() const;

const_iterator cbegin(); const
const_iterator cend(); const
```

`begin` returns an iterator referring to the first element in the vector. `end` returns an iterator which is the past-the-end value for the vector. If the vector is empty, then `begin() == end()`.

Once the initialisation of the vector has been performed (Section 2.5) or the simulation has started, iterators returned by `begin` or `end` shall be valid until the lifetime of the referenced vector (element) has ended. The variants `cbegin` and `cend` shall return the same `const_iterator` values as `begin` and `end`, respectively.

2.9 Binding of vectors

Since `sc_vector` is mainly intended for structural components of a model, direct support for the various `bind` combinations is useful. The vector class provides the following combinations for element-wise binding between different containers, either based on full container ranges, or sub-ranges given as explicit iterators. All different overloads return an iterator to the first unbound element within the `sc_vector` object or `end()` if all elements have been bound.

```
template< typename ContainerType, typename ArgumentType >
iterator bind( sc_vector_view<ContainerType,ArgumentType> c )
    { return bind( c.begin(), c.end() ); }

template< typename BindableContainer >
iterator bind( BindableContainer & c )
    { return bind( c.begin(), c.end() ); }

template< typename BindableIterator >
iterator bind( BindableIterator first, BindableIterator last )
    { return bind( first, last, this->begin() ); }

template< typename BindableIterator >
iterator bind( BindableIterator first, BindableIterator last, iterator from );

template< typename ContainerType, typename ArgumentType >
iterator operator()( sc_vector_view<ContainerType,ArgumentType> c )
    { return operator()( c.begin(), c.end() ); }

template< typename ArgumentContainer >
iterator operator()( ArgumentContainer & c )
    { return operator()( c.begin(), c.end() ); }

template< typename ArgumentIterator >
iterator operator()( ArgumentIterator first, ArgumentIterator last )
    { return operator()( first, last, this->begin() ); }

template< typename ArgumentIterator >
iterator operator()( ArgumentIterator first, ArgumentIterator last, iterator from );
```

The three-argument variants take a range of argument iterators, that shall be bound element-wise—either via a `bind` or an `operator()` call—to the `sc_vector` range starting from position `from`. For that matter, the expressions

```
(*from).bind( *first ); // for vec.bind(...)  
(*from)( *first );     // for vec( ... )
```

shall be well-formed for the actual template parameters. If `from` does not point to an element in the current `sc_vector` object, the behaviour shall be undefined.

NOTE— Since `sc_vector` is a class template, only those member functions are instantiated, that are actually used by an application. The addition of `bind` functions to the overall template does not restrict the general usability for types without their own `bind` functions.

3 Looking at vector element's members— `sc_vector_view`

Especially during the hierarchical binding of port or module containers, element-wise access to a vector element's *member* is needed. To avoid another cause for manually looping over container elements, a member-wise view with the same public interface as an `sc_vector` itself is provided.

Example

```
SC_MODULE( sub_module ) { sc_in<bool> in; /* ... */ };  
  
sc_vector< sc_in<bool> > in;  
sc_vector< sub_module > sub_mods;  
// ...  
  
// for( int i=0; i<in.size(); ++i )  
//   sub_mods[i].in( in[i] );  
  
// better: bind member ports over a vector view:  
sc_view( sub_mods, &sub_module::in ).bind( in );
```

The actual `sc_vector_view<T,MT>` class is implementation-defined and shall not be created explicitly by an application. An application shall use the light-weight factory function `sc_view` instead.

NOTE— `sc_vector_view<T,MT>` can be assigned and copied, though.

3.1 Class definition

```
template< typename T, typename MT >  
class sc_vector_view;  
  
// factory function  
template< typename T, typename MT >  
sc_vector_view<T,MT> sc_view( sc_vector<T> & v, MT (T::*member_ptr) );  
  
template< typename T, typename MT >  
class sc_vector_view  
{  
public:  
    typedef sc_vector<T> base_type;  
    typedef /* implementation-defined */ iterator;  
    typedef /* implementation-defined */ const_iterator;  
  
    typedef typename base_type::size_type      size_type;  
    typedef typename base_type::difference_type difference_type;  
    typedef typename iterator::reference       reference;  
    typedef typename iterator::pointer        pointer;  
  
    iterator begin();
```

```

iterator end();
const_iterator begin() const;
const_iterator end() const;
const_iterator cbegin() const;
const_iterator cend() const;

size_type size() const;

std::vector< sc_object* > get_elements() const;

reference operator[]( size_type idx );
reference at( size_type idx );
const reference operator[]( size_type idx ) const;
const reference at( size_type idx ) const;

template< typename ContainerType, typename ArgumentType >
iterator bind( sc_vector_view<ContainerType,ArgumentType> c );

template< typename BindableContainer >
iterator bind( BindableContainer & c );

template< typename BindableIterator >
iterator bind( BindableIterator first, BindableIterator last );

template< typename BindableIterator >
iterator bind( BindableIterator first, BindableIterator last , iterator from );

template< typename ContainerType, typename ArgumentType >
iterator operator()( sc_vector_view<ContainerType,ArgumentType> c );

template< typename ArgumentContainer >
iterator operator()( ArgumentContainer & c );

template< typename ArgumentIterator >
iterator operator()( ArgumentIterator first, ArgumentIterator last );

template< typename ArgumentIterator >
iterator operator()( ArgumentIterator first, ArgumentIterator last, iterator from );
};

```

All member functions (except `get_elements`) shall forward their arguments to the underlying `sc_vector` instance. Iterator dereference, element access and `bind` shall operate on the element's member, given as a pointer-to-member argument to `sc_view`.

The member function `get_elements` of the view proxy class shall return a `std::vector< sc_object* >` by-value. This vector shall be formed by casting the member-pointees statically to `sc_object`.

NOTE— Since the `get_elements` function requires a `static_cast` from the member-pointer to an `sc_object` pointer, calling `get_elements` is only supported for types that are publicly derived from `sc_object`.

NOTE— Since `sc_view` enables traversal over members of an `sc_vector`, the result of `get_elements` has to be dynamically created.

4 Proof-of-concept implementation

A proof of concept implementation based on a thin, type-safe wrapper around `std::vector<void*>` has been sent to the OSCI SystemC Language Working Group and can be made available to the IEEE P1666 Working Group under the terms of the OSCI Open Source License upon request.

The proposed API documented in this article has evolved during the discussion in the OSCI Language Working Group and the IEEE P1666 Working Group. The current version is based on the received feedback and internal polishing compared to the earlier proposals to the LWG and P1666. The most important changes are:

SECOND REVISION (2010-11-04)

- Restrict template arguments to objects derived from `sc_object`
- Add `get_elements` to `sc_vector_base`, and `sc_vector_view`
- Standardise `sc_vector_base` for easier traversal.
- Allow creation during simulation, iff element type does.

FIRST REVISION

- Class names now `sc_vector`, instead of `sc_array`.
- Internally combined iterator implementation for `sc_vector_iter` and iteration of a member view.
- Default constructor added to `sc_vector<T>`.
- Added `bind`, `sc_view`, and `sc_vector_view`.
- Other reductions of explicitly exposed symbols.

5 TLM 2.0— a simple router example

Sockets from TLM 2.0 are already usable with the proposed `sc_vector` API. The following is based on a Doulos example of a simple router:

```
// *****
// Generic payload blocking transport router
// *****

template<unsigned int N_TARGETS> // could be a member instead
struct Router: sc_module
{
    // TLM-2 socket, defaults to 32-bits wide, base protocol
    tlm_utils::simple_target_socket<Router>          target_socket;

    // *****
    // Use tagged sockets to be able to distinguish incoming backward path calls
    // *****

    sc_vector< tlm_utils::simple_initiator_socket_tagged<Router> > initiator_socket;

    SC_CTOR(Router)
    : target_socket("target_socket")
    , initiator_socket( "socket" ) // set name prefix
    {
        // Register callbacks for incoming interface method calls
        target_socket.register_b_transport(      this, &Router::b_transport);
        target_socket.register_get_direct_mem_ptr(this, &Router::get_direct_mem_ptr);
        target_socket.register_transport_dbg(    this, &Router::transport_dbg);

        // create tagged sockets from a member function
        initiator_socket.init( N_TARGETS
                               , sc_bind(&Router::create_socket
                                          , this, sc_unamed::_1, sc_unamed::_2 ) );
    }
};
```

```

// or use an explicit loop
initiator_socket.init( N_TARGETS );
for( int i=0; i<N_TARGETS; ++i )
    initiator_socket[i].register_invalidate_direct_mem_ptr( ..., i );
}

// socket creation function
tlm_utils::simple_initiator_socket_tagged<Router> *
create_socket( const char * name, size_t id )
{
    tlm_utils::simple_initiator_socket_tagged<Router> * socket
        = new tlm_utils::simple_initiator_socket_tagged<Router>( name );

    // *****
    // Register callbacks for incoming interface method calls, including tags
    // *****
    socket->register_invalidate_direct_mem_ptr
        ( this, &Router::invalidate_direct_mem_ptr, id );

    return socket;
}

// *****
// FORWARD PATH
// *****

// TLM-2 blocking transport method
virtual void b_transport( tlm::tlm_generic_payload& trans, sc_time& delay )
{
    // ...

    // Forward transaction to appropriate target
    initiator_socket[target_nr]->b_transport( trans, delay );
    // instead of ugly:
    // ( *initiator_socket[target_nr] )->b_transport( trans, delay );
}

// TLM-2 forward DMI method
virtual bool get_direct_mem_ptr( tlm::tlm_generic_payload& trans,
                                tlm::tlm_dmi& dmi_data);
{
    //...

    bool status = initiator_socket[target_nr]->get_direct_mem_ptr( trans, dmi_data );

    // Calculate DMI address of target in system address space
    // ...

    return status;
}

// TLM-2 debug transaction method
virtual unsigned int transport_dbg( tlm::tlm_generic_payload& trans)
{
    // ...

    // Forward debug transaction to appropriate target
    return initiator_socket[target_nr]->transport_dbg( trans );
}

// *****
// BACKWARD PATH, & ROUTER INTERNALS unchanged

```

```
// *****  
};
```

6 Further comments

sc_vector< T > can easily support polymorphism

In the following example, a different type derived from a common base class (which is then used as `element_type` in the vector) is allocated within a user-defined factory function.

```
#include <systemc>  
  
struct derived : sc_core::sc_object  
{  
    derived( const char * n ) : sc_object( n ) {}  
    virtual const char* kind() const { return "derived"; }  
};  
  
sc_core::sc_object* create_derived( const char* name, size_t n )  
{ return new derived(name); }  
  
int sc_main( int, char*[] )  
{  
    sc_core::sc_vector< sc_core::sc_object > v( "v", 1, create_derived );  
    std::cout << v[0].name() << " " << v[0].kind();  
}  
// Output: v_0 derived
```

7 Acknowledgements

The author would like to thank John Aynsley, David C. Black, Dennis P. O'Connor, and Jerome Cornet for their valuable feedback. Thanks to Ralph G3rger and Andreas Herrholz for their proof-reading and detailed comments.

8 Contact information

Philipp A. Hartmann <philipp.hartmann@offis.de>
Hardware/Software Design Methodology Group
R&D Division Transportation
OFFIS Insitute for Information Technology
Oldenburg, Germany