
**European Space Agency
Directorate of Technical and Quality
Management**

STATEMENT OF WORK

**HW-SW SYSTEMC CO-SIMULATION SOC
VALIDATION PLATFORM**

Reference: TEC-EDM/2008.27/BG

Issue: Final version Revision: 2

17/06/2009

Table of Contents

| | | |
|--|--|-----------|
| 1 | INTRODUCTION..... | 3 |
| 1.1 | SCOPE OF THE DOCUMENT | 3 |
| 1.2 | APPLICABLE AND REFERENCE DOCUMENTS | 3 |
| 1.2.1 | <i>Applicable Documents (ADs)</i> | 3 |
| 1.2.2 | <i>Reference Documents (RDs)</i> | 4 |
| 1.3 | ACRONYMS AND ABBREVIATIONS | 5 |
| 2 | BACKGROUND AND OBJECTIVES..... | 6 |
| 2.1 | BACKGROUND..... | 6 |
| 2.1.1 | <i>Transaction Level Modelling and SystemC</i> | 6 |
| 2.2 | DEFINITIONS | 8 |
| 2.3 | OBJECTIVES OF THE ACTIVITY..... | 9 |
| 3 | WORK TO BE PERFORMED..... | 10 |
| 3.1 | WORK LOGIC | 10 |
| 3.2 | PHASE 1: ANALYSIS AND DESIGN OF KEY IPS..... | 11 |
| 3.2.1 | <i>Task 1: Survey of Available Tools and Techniques</i> | 11 |
| 3.2.2 | <i>Task 2: High-Level Modelling of SystemC IPs</i> | 12 |
| 3.2.3 | <i>Task 3: Verification of TL Models</i> | 13 |
| 3.3 | PHASE 2: VIRTUAL PLATFORM DEVELOPMENT AND VALIDATION | 14 |
| 3.3.1 | <i>Task 4: Definition of the Design Flow</i> | 14 |
| 3.3.2 | <i>Task 5: Proof-of-concept VP Development</i> | 15 |
| 3.3.3 | <i>Task 6: High-Level DSE Demonstration</i> | 16 |
| 4 | REQUIREMENTS FOR MANAGEMENT, REPORTING, MEETINGS AND DELIVERABLES | 17 |
| 4.1 | MANAGEMENT | 17 |
| 4.2 | REPORTING | 17 |
| 4.3 | MEETINGS..... | 18 |
| 4.4 | DELIVERABLES | 19 |
| 5 | SCHEDULE AND MILESTONES..... | 20 |
| 5.1 | DURATION | 20 |
| ANNEX 1: TECHNICAL REQUIREMENTS | 21 | |
| A1.1 | SYSTEMC IP MODELS REQUIREMENTS | 21 |
| A1.2 | VPI REQUIREMENTS | 23 |
| A1.3 | PROOF-OF-CONCEPT VIRTUAL PLATFORM REQUIREMENTS | 25 |
| ANNEX 2: VIRTUAL PLATFORM TOOLS | 26 | |
| A2.1 | OPEN SOURCE TOOLS | 26 |
| A2.1.1 | <i>ReSP</i> | 26 |
| A2.1.2 | <i>Unisim</i> | 26 |
| A2.1.3 | <i>SoCLib</i> | 26 |
| A2.1.4 | <i>OVP — Open Virtual Platform</i> | 27 |
| A2.2 | COMMERCIAL TOOLS | 27 |
| A2.2.1 | <i>Synopsys Innovator</i> | 27 |
| A2.2.2 | <i>CoWare Platform Architect</i> | 28 |
| A2.2.3 | <i>Carbon SoC Designer</i> | 28 |

1 Introduction

1.1 Scope of the Document

This document describes the activity to be executed and the deliverables required by the European Space Agency in relation to *HW-SW SystemC Co-Simulation SoC Validation Platform*.

It will be part of the contract and shall serve as an applicable document throughout the execution of the work.

1.2 Applicable and Reference Documents

1.2.1 Applicable Documents (ADs)

The following documents, listed in order of precedence, contain requirements applicable to the activity:

- AD[1] IEEE Standard SystemC Language Reference Manual
Available at <http://www.systemc.org>
- AD[2] OSCI TLM 2.0 Standard
Available at <http://www.systemc.org>
- AD[3] AMBA 2.0 Specification
Available at http://www.arm.com/products/solutions/AMBA_Spec.html
- AD[4] The SPARC Architecture Manual Version 8
Available at <http://www.sparc.org>
- AD[5] ESA C and C++ Coding Standard BSSC(2000)1
Available at [ftp://ftp.estec.esa.nl/pub/wm/wme/bssc/bssc2000\(1\)i10.PDF](ftp://ftp.estec.esa.nl/pub/wm/wme/bssc/bssc2000(1)i10.PDF)

1.2.2 Reference Documents (RDs)

The following documents can be consulted by the Contractor as they contain relevant information:

- RD[1] GNU GDB
Available at <http://www.gnu.org>
- RD[2] Automatic Generation Including Fast Timed Simulation Models of Operating Systems
in Multiprocessor SoC Communication Design – S. Yoo, G. Nicolescu, et al.
In *Proceedings of Design Automation and Test in Europe 2002*
- RD[3] GRLIB User's Manual
Available at <http://www.gaisler.com>

1.3 Acronyms and abbreviations

| | |
|-------|--|
| AD | Applicable Document |
| AHB | AMBA High-performance Bus |
| BSD | Berkeley Software Distribution [License] |
| BSP | Board Support Package |
| DSE | Design Space Exploration |
| ESL | Electronic System Level |
| FPGA | Field Programmable Gate Array |
| GDB | GNU DeBugger |
| GNU | GNU is Not Unix |
| GPL | GNU Public License |
| HDL | Hardware Description Language |
| HW | HardWare |
| IP | Intellectual Property |
| MPSoC | Multi-Processor System-on-Chip |
| NoC | Network-on-Chip |
| OSCI | Open SystemC Initiative |
| RD | Reference Document |
| SoC | System-on-Chip |
| SW | SoftWare |
| VHSIC | Very-High-Speed Integrated Circuits |
| VHDL | VHSIC Hardware Description Language |
| VP | Virtual Platform |
| VPI | Virtual Platform Infrastructure |
| RTL | Register Transfer Level |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

2 Background and Objectives

2.1 Background

Increasingly large portions of electronic systems are being implemented in software and its development cost starts dominating the cost for the whole system. Software is also becoming the critical part of the development schedule, mainly because deploying and testing it on the real target hardware is complicated. The concurrent design of hardware and software aims at mitigating these issues, allowing software to be developed before the final hardware is ready. Three main methodologies have been devised RD[2]:

- the use of an FPGA emulator of the HW under development,
- the use of a workstation with appropriate peripheral boards,
- or running the software in a “virtual platform” or a full system simulator

In particular, the latter technique offers an approach that has many advantages (flexibility, accuracy, observability, etc.) with respect to the other methods.

According to RD[2], a Virtual Platform (see Figure 2-1) is a system level model that emulates real system behaviour. It operates at the level of processor instructions, function calls, memory accesses and data packet transfers, as opposed to the bit-accurate, nanosecond-accurate logic transitions of a register transfer level (RTL) model.

Virtual system prototyping has three principal use cases:

1. System architecture development, analysis, optimization and validation.
2. Software development, validation and debug using co-developed hardware models.
3. Hardware development, verification and debug using co-developed software.

Nowadays we can observe that there are two distinct users of Virtual Platforms: system designers and SoC designers. System designers emphasize early software development, while SoC designers tend to use the methodology for HW/SW co-development.

2.1.1 Transaction Level Modelling and SystemC

SystemC enables the description of hardware systems at different abstraction levels and with different modelling styles. The accuracy of SystemC descriptions can be independently analyzed on two dimensions: communication among the components and their internal functionality (i.e. the computation aspect of the component).

Recently, OSCI released the Transaction Level Modelling (TLM) 2.0 standard AD[2], which supports the abstract modelling of memory-mapped buses, together with an extension mechanism to support the modelling of specific bus protocols whilst maximizing interoperability.

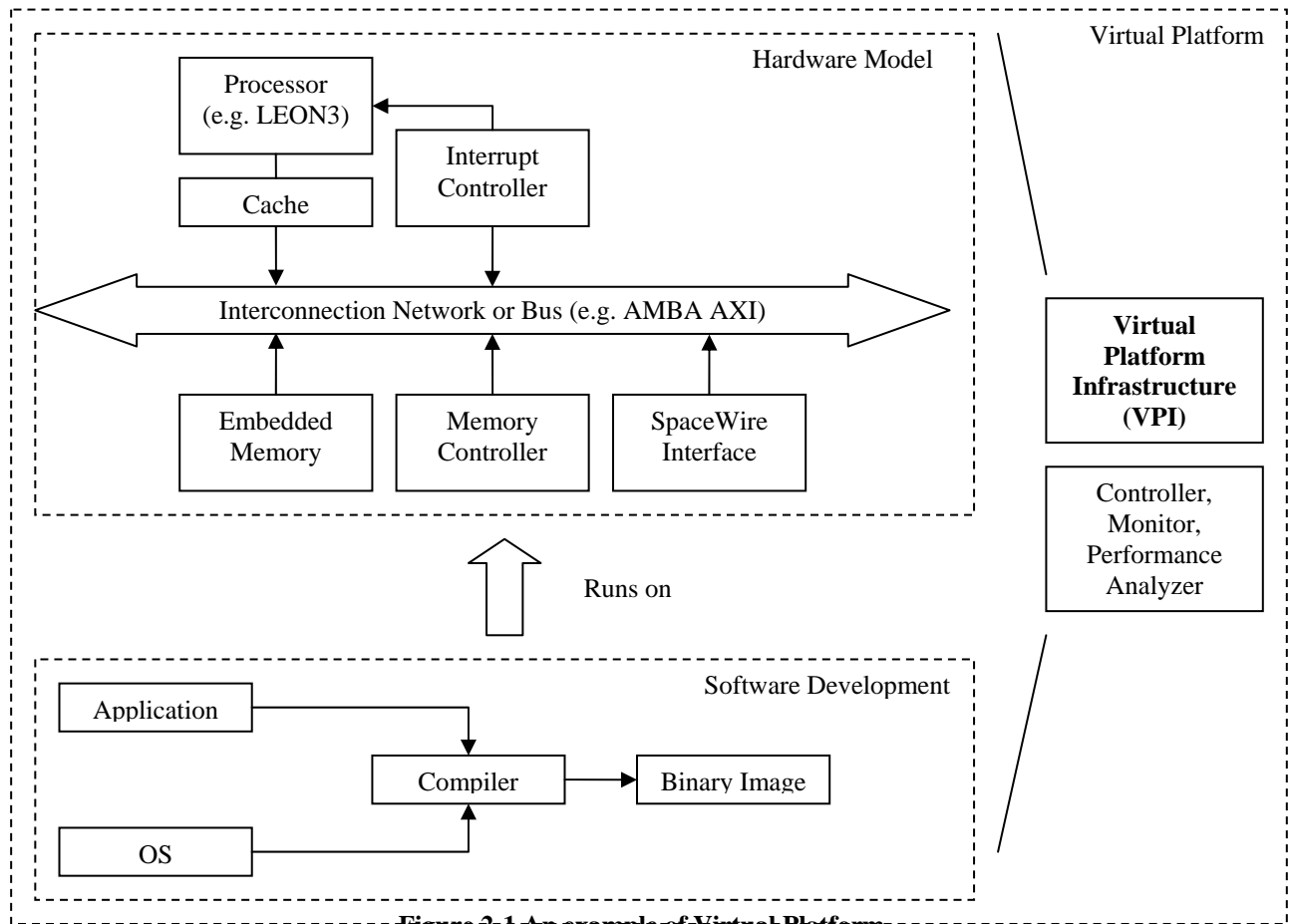


Figure 2-1 An example of Virtual Platform

2.2 Definitions

In the frame of this activity, the following terms are used with the indicated meaning:

Design Flow (DF): the explicit combination of electronic design automation (EDA) tools used in a sequential and systematic order, with defined inputs and output files, in order to accomplish the design of an integrated circuit. In particular, this document refers to Electronic System Level (ESL) design and verification, an emerging electronic design methodology that focuses on the higher abstraction level concerns. ESL is now an established approach at most of the world's leading System-on-a-chip (SoC) companies, and is being used increasingly in system design.

RTL IP Cores: low-abstraction level models, consisting of HDL files, of well-specified functionalities that can be implemented in hardware (e.g. integrated circuits) after additional EDA processing.

SystemC IP Models (IPs): high-abstraction level models, consisting of SystemC language files, of well-specified functionalities (such as instruction and data processors, memories, busses, etc.) that can be implemented in hardware (e.g. integrated circuits) after additional EDA processing which will normally involve a conversion to lower-abstraction level models (e.g. RTL IP Models). For example a SpaceWire interface model, a LEON3 processor model, etc.

Virtual Platform (VP): a system level model that emulates real system behaviour, formed by the interconnection of a set of SystemC IP Models. Software can be run on the virtual platform as it was running on the real hardware.

Virtual Platform Infrastructure (VPI): this document refers to VPI as formed by two components, the *Interconnection Infrastructure* and the *Analysis Tools*.

- The *Interconnection Infrastructure* is a set of interfaces, scripts and languages that are used to connect IP models into a single entity, namely a Virtual Platform (VP) and load one or more software applications in its memory. This infrastructure is necessary to define the components of the VP and their configuration.
- The *Analysis Tools* are a set of software tools that can be used to analyse the performance of the applications running on a VP. E.g. Throughput and Bandwidth analysis between the VP components.

Virtual Platform Software: the software running *on the Virtual Platform*, i.e. applications to be simulated and analysed, compiled in binary images and executed by one or more processor models simulated in the VP, together with the appropriate stimuli at I/O ports.

2.3 Objectives of the activity

The objectives of the activity are:

- The definition of a Design Flow (DF)
- The high-level modelling of key IPs in SystemC TLM 2.0, as listed in Requirement R.3 in Annex 1. These IPs shall follow the requirements and guidelines of Annex 1.
- The functional validation and timing accuracy analysis of said IPs
- The development of a proof-of-concept Virtual Platform using the produced IPs and an appropriate Virtual Platform Infrastructure, documenting every step of the process, and thus defining the Design Flow first draft.
- The simulation performance and accuracy analysis of the proof-of-concept Virtual Platform, also using the chosen Virtual Platform Infrastructure and documenting every step of the process, complementing the Design Flow draft.

The intended uses of the activity outputs are:

- Allow ESA to develop new VPs based on the available SystemC IP Models and distribute them to contractors without being subject to any fee or restriction, allowing software development before SoC hardware is ready
- ESA will license the SystemC IP Models without being subject to any fee or restriction and allow contractors to develop new VPs using the guidelines in the DF and perform design space exploration for future SoCs to be implemented either as ASICs or on FPGA.

3 Work to be performed

The work to be performed within the frame of this activity is specified in this chapter.

3.1 Work Logic

As shown in Figure 3-1, the activity is split into two technical phases. At the end of each phase, a review meeting (see 4.3) shall be held in order to check the relevant achievements. The successful completion of phase 1 is considered as necessary for commencing phase 2.

Each phase is sub-divided into tasks. If not differently specified, the order in which tasks are numbered also represents the precedence in the workflow.

Phase 1 includes the high-level modelling of key IPs listed in Requirement R.3 in Annex 1, needed for the VP implementation. Task 1 shall consolidate the requirements, identify the VPI that will be used for the rest of the activity and define the strategy for Task 2, which consists of the high-level modelling and implementation in SystemC of each IP. Task 3 validates the design of each SystemC IP and measures its performance in terms of accuracy and simulation speed.

The objective of **Phase 2** is to validate the overall approach by using the SystemC IPs, the VPI and the DF to produce a Virtual Platform for a representative space application. Task 4 is devoted to the establishment of the VPI with interconnection and performance measurement mechanisms between the various SystemC IP models; the Contractor shall develop the platform supporting the application in Task 5 and demonstrate and document the chosen DF performing high-level design space exploration and analysing design trade-offs in Task 6.

The following reviews shall be held:

MDR (Model Design Review); at the end of Task 3

DFR (Design Flow Review); at the end of Task 5

AR (Acceptance Review); at the end of Task 6

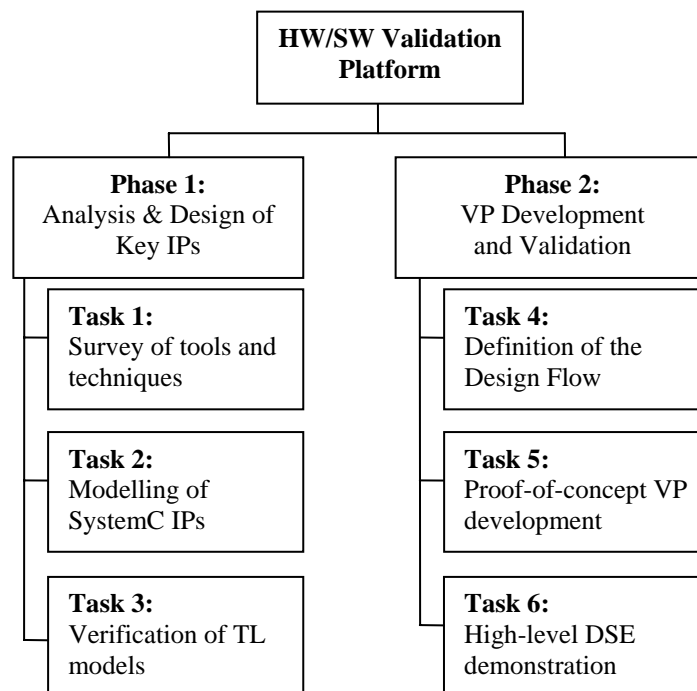


Figure 3-1 Work logic

3.2 Phase 1: Analysis and Design of Key IPs

The objective of the first technical phase is to develop the necessary high-level SystemC models of existing RTL IP cores to produce a VP representative of a typical space SoC. These models shall be made in such a way that they can be reused and adapted to different platform configurations. The required IPs are listed in Annex 1.

3.2.1 Task 1: Survey of Available Tools and Techniques

The Contractor shall evaluate the required SystemC IPs and determine the best tools, techniques and methodologies to develop their high-level models according to the SystemC TLM 2.0 standard and their validation.

The Contractor shall also consolidate the requirements for such SystemC IPs, considering:

- Timing Accuracy and Bit Accuracy
- Simulation Performance

in the context of the following use cases (see AD[2]):

- Software Development: the IPs will be used for developing software
- Software Performance: the IPs will be used to evaluate the performance of the software running on the VP
- System Architectural Analysis: the configuration and number of instantiations of the IPs will be modified to tune the system to specific workloads

These have to be summarized in two documents: the *Development Plan* and the *IP Requirements Specification*

Tasks:

- 1.1 Review the existing specifications of the required IPs.
- 1.2 Define a methodology for the development and validation of the required TLM SystemC IPs based on the state-of-the art ESL design

Outputs:

- IP Requirements Specification
- Development Plan

3.2.2 Task 2: High-Level Modelling of SystemC IPs

The Contractor shall develop high-level models of the IPs specified in Annex 1 using the SystemC language and adhering to the TLM 2.0 standard.

Each IP shall be available in both Loosely-Timed and Approximately-Timed TLM-2 coding styles and shall be accompanied by a User Manual (UM) and a Development Document (DD). The UM describes the IP interface and functions and its use from the perspective of the system architect and the programmer, including examples. The final structure of the UM is to be agreed with ESA during Task 1. The DD describes the rationale behind the adopted modelling style and functional block architecture, how it differs from an RTL implementation and the expected accuracy and simulation performance expressed as required in Annex 1.

The models must come with the appropriate scripts (and all other necessary files) and test benches for compilation and verification of functionality.

Tasks:

- 2.1 Development and documentation of the required SystemC TLM IPs
- 2.2 Development of test benches
- 2.3 Functional model validation

Outputs:

- A SystemC TLM 2.0 model library of the required IPs
- Model User Manuals and Development Documents
- Compilation scripts, needed libraries, and test benches for each IP model

3.2.3 Task 3: Verification of TL Models

The Contractor must evaluate the timing accuracy and simulation speed of each SystemC IP model, comparing them with their RTL implementation. In case the timing performance of the models shall not meet the requirements specified in Annex A1.2 the models shall be modified and updated to rectify the issue. The results shall be reported in the *IP Performance Document*.

Tasks:

- 3.1 Compare the developed SystemC IP models with their RTL counterparts to determine their timing accuracy
- 3.2 Determine the simulation performance of each IP (see Annex A1.2)
- 3.3 Update any eventual IP not meeting the requirements

Outputs:

- IP Performance Document
- Updated SystemC Model Library

3.3 Phase 2: Virtual Platform Development and Validation

3.3.1 Task 4: Definition of the Design Flow

The contractor shall define the Design Flow and procure the VPI necessary to perform the communication, performance, and SW debugging analyses specified in A1.2. To do so, the tools shall be able to collect the necessary data from the SystemC IP models. An efficient mechanism to do so shall be defined, and the SystemC IP models shall be modified to support this mechanism if necessary. A non-exhaustive list of possible software packages that can implement the VPI is listed in Annex 2 for convenience only.

The Contractor shall define the appropriate IP interconnection infrastructure that allows the SystemC IP models to be connected together in a Virtual Platform. These mechanisms shall provide the ability to connect models specified at different levels of abstraction, via appropriate transactors (see AD[2]).

Transactors shall be written in SystemC and the interconnection methodology shall be outlined in the *Interconnection Methodology Summary*.

This first iteration of the Design Flow shall demonstrate the capability to perform co-design (hardware/software) using high abstraction level language based on System C (IEEE standard).

The VPI shall:

- Allow the interconnection of TL and RTL models of the specified IPs
- Provide a tool for the system- and high-level design phases of an embedded system, allowing HW-SW co-design and early system validation.
 - Allow Integration, verification and debugging of hardware IPs
 - Early software development on the target architecture: debugging (such as access to registers, linking to source code, breakpointing, etc.), monitoring, and profiling of application software
 - Allow performance analysis: cycle count, transaction count, execution time, communication bandwidth and throughput, etc.

The detailed requirements for the VPI are listed in Annex A1.2.

Tasks:

- 4.1 Procure the VPI
- 4.2 Determine the information that the VPI analysis tools need from the SystemC IP models
- 4.3 Define an efficient mechanism to obtain such information during simulation
- 4.4 Define the appropriate IP interconnection mechanisms
- 4.5 Implement the appropriate transactors
- 4.6 Produce a first iteration of the DF document

Outputs:

- Updated SystemC IP Model Library
- Transactor library
- Design Flow document: Interconnection infrastructure and Analysis Capability

3.3.2 Task 5: Proof-of-concept VP Development

The Contractor shall define and implement a proof-of-concept VP using **all** the SystemC IP models, the VPI and DF from the previous tasks. This VP shall run software (including an Operating System) stimulating all the connected IP models. The design flow (DF) for the implementation of such VP shall be as established in Task 4, and it will be updated with all the lessons and experience gained during the actual VP development. The VP shall be profiled and benchmarked, reporting its simulation performance in transactions per second, as specified in Annex A1.3.

ESA will provide a LEON2/3 integer unit SystemC IP model that shall be integrated with the VPI and proof-of-concept VP.

ESA will also provide, if available, a SpaceWire and a CAN interface SystemC IP model. The contractor shall consider the integration of these two additional IPs (SpaceWire and CAN) as an option.

The proof-of-concept VP shall:

- Demonstrate the application of the DF on a typical SoC, showing how it can be used for performance analysis and design trade-off
- Be a single distributable application that acts as a virtual development board, on which software can be run and performance can be estimated

Tasks:

- 5.1 Interconnect all the IP models (both developed in Task 2 and ESA-provided) using the VPI from Task 4
- 5.2 Develop and/or adapt the software to be run on the VP
- 5.3 Benchmark the produced VP
- 5.4 Update of the VPI and overall DF based on lessons learned

Outputs:

- The VP package (SystemC source, build scripts, data files, testbenches, documentation, etc.)
- VP Definition, Interconnect and Performance Report
- Updated DF

3.3.3 Task 6: High-Level DSE Demonstration

The VP produced in Task 5 shall be used in conjunction with the VPI to demonstrate its use for high-level design space exploration (DSE). The VP shall run a set of application benchmarks, and the architectural parameters of the system should be changed to optimize its performance, reporting the best configurations in a Pareto curve in the *High-Level DSE Report*.

Tasks:

- 6.1 Port the application benchmarks to the VP produced in Task 5
- 6.2 Run a set of simulations, varying the architectural parameters of the system
- 6.3 Report the variations on system performance and the best and worst configurations

Outputs:

- High-Level DSE Report
- Final Report and Executive summary

4 Requirements for Management, Reporting, Meetings and Deliverables

The standard requirements for Management, Reporting, Meetings and Deliverables (Appendix 2 to the Contract) shall apply, taking account of the following specific requirements for the present activity, which shall prevail in case of conflict.

4.1 Management

The Contractor shall describe its organisation to execute the work. The Project Manager, nominated by the Contractor, shall be responsible for the management and execution of the work to be performed. The Agency Technical Officer (TO) nominated in the Contract will be responsible for the technical management of the activity on the ESA side, which includes providing guidance to the Contractor, as well as monitoring the progress of the activity and assessing the results.

4.2 Reporting

All the documents shall be written in English (UK) and spell checked before being issued. Each document shall be given an appropriate document number, version number and date of issue. All changes in updated documents shall be summarised in a change record.

When approved, documentation shall be provided to the Agency as Adobe (.pdf) and editable OpenDocument Text Format (.odt) files.

As soon as they become available, the Contractor shall submit for the Agency's approval all technical notes, WP reports and documents, which are produced during the execution of the contract. Any technical documentation to be discussed at a meeting with the Agency shall be submitted at least ten working days prior to the meeting.

The Contractor shall set up a version and project management website accessible from ESA/ESTEC during Task 1, where the roadmap, milestones and all code will be inserted during the course of the activity. The version and project management website shall be constantly updated as the activity progresses.

4.3 Meetings

As a complement to the conditions to section 3 of the Appendix 2 to the contract, the following shall apply:

| Meetings | | | | |
|-----------------|------------------|-------------|-----------------------|--|
| Meeting | Event(s) | Time | Place | Notes |
| KOM | Kick-Off Meeting | T0 | ESTEC | T0=Kick-off |
| MDR | PM-MDR | T1 | Contractor's premises | T1=end of Task 3 (Verification of TL Models) |
| DFR | PM-DFR | T2 | Contractor's premises | T2=end of Task 5 (Definition of the Design Flow) |
| AR | FP | T3 | ESTEC | T3=end of Task 7 (High-Level DSE Demonstration) |

In addition to these meetings, monthly progress meetings will be held via Telecon/Videocon. For all meetings, handouts shall be available in sufficient number for all participants.

4.4 Deliverables

Note: the numbering on applicable paragraphs refers to the numbering of the Appendix 2 to the contract.

| Deliverables List | | | |
|--|--|---------------|---------------|
| Item | Format | Copies | Due |
| Monthly Progress Reports | Email | NA | Monthly |
| Development Plan and Requirements Specifications | Electronic | NA | End of Task 1 |
| MDR (T1) | | | |
| IP models' User Manuals | Electronic | NA | T1 |
| IP models' Development Documentation | Electronic | NA | T1 |
| IP Performance Document | Electronic | NA | T1 |
| Draft SystemC IP Model Library | Model code, build scripts and data files | NA | T1 |
| DFR (T2) | | | |
| DF: Analysis Capability Report | Electronic | NA | T2 |
| DF: Interconnection Methodology Summary | Electronic | NA | T2 |
| Transactor Library | Model code, build scripts, examples | NA | T2 |
| AR (T3) | | | |
| High-Level DSE Report | Electronic | NA | T3 |
| Final Report and Executive summary | Electronic | NA | T3 |
| Design Flow Report | Electronic | NA | T3 |
| Virtual Platform Package | Model code, build scripts, examples | NA | T3 |
| Final SystemC IP Model Library | Model code, build scripts and data files | NA | T3 |
| VP Definition, Interconnect and Performance Report | Electronic | NA | T3 |

5 Schedule and Milestones

The Contractor shall maintain a Gantt chart that represents a detailed time allocation and distribution of the activities. A schedule chart copy shall be included in all progress reports. The time schedule shall be compliant with the main milestones represented by the meetings foreseen in Section 4.

5.1 Duration

The duration of the work shall not exceed 12 months from kick-off to end of the activity (delivery of final report and software).

The milestones/meetings defined in section 4.3 shall apply.

Annex 1: Technical Requirements

This document provides the top-level requirements for the design of a “HW-SW SYSTEMC CO-SIMULATION SOC VALIDATION PLATFORM” in the frame of a TRP activity with the same name.

In the first part of the project, the Contractor shall prepare a *Requirement Specification* document where the top-level requirements shall be examined in-depth and formulated more in detail.

The top-level requirement list is organised as follows:

- **Requirements** are marked with the paragraph number that allows unambiguous identification of each requirement.
- **Explaining text** (printed in *italic*) that provides information for a better understanding of the requirements or verifications (if any). In case of conflict between requirement (and/or verification) and text, the reader should report the mismatch to the author but the requirement remains the reference for the design and takes precedence over the text.
- **Shall** requirements are mandatory requirements. **Should** requirements describe design goals.

A1.1 SystemC IP Models Requirements

The SystemC IP Models shall comply with the following constraints:

R.1 *General Modelling Style*

The contractor shall develop SystemC TL models, as described by AD[1] and AD[2]. Each model shall be available in both Loosely-timed (LT) and Approximately-timed (AT) flavours, unless specified.

R.2 *IP Models Interface*

The SystemC IP Models shall feature a TLM interface using the TLM 2.0 standard payload as described in AD[2], eventually extended with the prescribed modalities

R.3 *SystemC IPs Models to Be Implemented*

The contractor shall implement at least the following IP models

- 1) AMBA AHB (AD[3])
- 2) Aeroflex Gaisler GRLIB MCTRL Memory Controller or equivalent (RD[3])
- 3) A memory model working with IP 2)
- 4) A Harvard L1 cache

- 5) A SPARCv8 MMU or equivalent (AD[4])
- 6) Aeroflex Gaisler GPTIMER General Purpose Timer Unit or equivalent (RD[3])
- 7) Aeroflex Gaisler IRQMP Interrupt Controller or equivalent (RD[3])

R.4 Functional Behaviour

The RTL version and SystemC version of the IPs shall have identical functional behaviour, i.e. given the same inputs, they shall produce the same outputs. This shall include the effects of configuration registers and any corner case if present. Functional verification shall be performed using test benches that have to be available in full source code, including auxiliary scripts, pattern files and reference models used for test pattern generation or analysis.

Reuse of RTL IP core test benches with an appropriate wrapper is acceptable.

R.5 Timing Behaviour

The difference in timing between the RTL version of each IP and the SystemC model should not exceed 20% on any benchmark.

The SystemC TLM Model and corresponding RTL IP core shall run the same test bench and the final timing reported by the two versions shall not differ of more than 20%

R.6 IP Simulation Performance

The IP simulation performance shall be measured on a fixed workstation, and shall be reported in transactions per second, using a testbench that stimulates at least 90% of the functionalities of the IP.

R.7 Transactors

Transactors shall allow interconnection between IPs specified at all levels of abstraction, allowing communication with RTL IPs using the TLM 2.0 standard payload

R.8 Self containment

The SystemC IP Model Library (source code, test benches, scripts) must be self-contained. External dependencies with VP or VPI elements such as packages, tools, etc. which may require commercial third party licences for future users of these SystemC IP Models, VP or VPI shall be avoided, unless these can be expected to be commonly and freely available to users, like e.g. the OSCI SystemC distribution. All external references shall be identified in the documentation, detailing the version used and the way to obtain them.

R.9 CAD tool independence

The SystemC IP Model Library shall not depend on a specific CAD tool, and it is required that the compilation can be performed with the only dependence of the OSCI SystemC Kernel.

R.10 SystemC IP Model User's Manual

The User's Manual has a similar function for the SoC designer as the component datasheet has for the board designer, it should allow him to design and integrate the IP/component into the system. It should clearly indicate the SystemC IP model configuration options have to be documented in a dedicated section, ensuring clear separation between hard (variables decided at architecture design time) and soft (programmable registers or inputs) configuration. The structure and contents of the IP User's Manual shall be pre-agreed with ESA IP Cores focal point, during the course of the WPI.

R.11 Build System

The SystemC IP Models shall come with compilation scripts and shall use a scalable, configurable, multi-platform build system.

E.g. CMake, waf. The use of autotools is discouraged due to their difficult use in multi-platform environments.

R.12 Version Management

All the code shall be available during the development on an internet-accessible concurrent version management system (such as SVN).

R.13 Code Quality

All the code shall be written according to ESA BSSC(2000)1 standard (AD[5])

A1.2 VPI Requirements

This section lists the applicable requirements for the VPI

R.14 Supported Systems

The VPI and related VPs shall be able to be compiled and run on any POSIX compliant Operating System, or at least Linux, MacOS and Microsoft Windows (possibly using Cygwin).

R.15 OS Emulation

The VPI shall allow the VP to run software compiled against the standard GNU libc using the GCC compiler.

The use of a BSP or linker script is allowed but optional. Any software running on the VPI shall have the option of either incorporating an embedded OS or running with OS emulation, and the VPI shall emulate the services that are needed by the software.

Services like file access, time access, etc. shall be emulated (see RD[2])

R.16 Software Debugging

The VPI shall allow multi-processor debugging of software running on the VP. The debugger shall allow breakpointing, profiling (i.e. the generation of a call graph and the ranking of source code statements according to execution time), linking to source code and access to registers. The user shall be able to determine which function it executed on each processor at any given time.

This behaviour is the same as GNU GDB.

R.17 Performance Analysis

The VPI shall report performance figures for applications running on the VP. These shall include at least:

- Processor performance counters: instructions, branches, instruction frequency, interrupts
- Cache performance counters : hits, miss (cold, capacity, conflict), flushes, block loads
- Memory performance counters: reads, writes

R.18 Communication Analysis

The VPI shall allow communication analysis of any VP, reporting at least the following information:

- Total data transferred between each platform component
- Data throughput between each platform component
- Average, Minimum and Maximum latency between each platform component
- Bus and Network contention and utilization
- Number of transactions between each platform components

A1.3 Proof-of-concept Virtual Platform Requirements

R.19 Platform Architecture

The proof-of-concept VP shall contain all IPs of R.3, at least one instance of the LEON2/3 processor model provided by ESA. The integration of the ESA-provided SpaceWire and CAN SystemC IP models shall be considered as an option.

R.20 VP Application

The proof-of-concept VP shall run a set of benchmarks running with the RTEMS operating system. The benchmarks shall be at least 5 and should belong to a popular suite like MeBench or other (any ideas/references to give?).

R.21 High-Level DSE

The contractor shall demonstrate that by modifying the platform parameters it is possible to determine the best configuration for each benchmark.

Annex 2: Virtual Platform Tools

This annex lists and describes some of the available tools to design and analyse performance of virtual platforms. More information on each tool can be found on the developer's website.

A2.1 Open source tools

A2.1.1 *RESP*

<http://www.resp-sim.org>

ReSP is an MPSoC simulation platform working at a high abstraction level; components used by ReSP are based on SystemC and TLM hardware and communication description libraries. The product is licensed under GPL.

Features:

- OS Emulation: it is possible to write programs and compile them with any elf cross-compiler, using a standard POSIX interface (with the included linker and init scripts).
- Multi-processor and multi-threading emulation: a full set of POSIX threads primitives and OpenMP support through GCC 4.2 are available for testing multi-threaded programs.
- Support for cross-compilation: compilers do not need to be patched, and single sources can be compiled simultaneously across several architectures (sparc, arm, powerpc, etc.)
- Automatic Design Space Exploration

A2.1.2 *UNISIM*

<http://www.unisim.org/>

Unisim is a virtual platform developed by a large industrial consortium with focus on software simulation methodology. Product is licensed under BSD.

Features include:

- TLM and cycle-accurate models of various processors
- Checkpointing and OS Emulation support

A2.1.3 *SOCLIB*

<http://www.soclib.fr/>

SoCLib is an open platform for virtual prototyping of multi-processors system on chip (MP-SoC). The core of the platform is a library of SystemC simulation models for virtual components (IP cores), with a guaranteed path to silicon. All simulation models are written in SystemC

The project is funded by the French **National Centre for Scientific Research (CNRS)** . Code is licensed under GPL.

Features:

- A large set of components available in TLM and RTL
- A unified socket interface based on OCP (<http://www.ocpip.com>)

A2.1.4 OVP — OPEN VIRTUAL PLATFORM

<http://www.ovpworld.org>

OVP comprises three components:

1. OVPmodels. These are true Open Source Software and thus there are no licensing costs for using, copying, modifying, enhancing these models.
2. OVP APIs. The APIs enable users to model processors, peripherals behavioural models and platforms to create extremely fast software virtual platforms. This component consists of C/C++ header files and associated documentation for each API. There are no licensing costs for using the APIs or documentation.
3. OVPsim. This is released as a binary only implementation as a dynamically linked library and is provided via a click-through license. This license restricts things like reverse engineering, but allows completely free usage, including commercial usage on as many computers as you like, and can be used with any other software/simulation environment. It is provided with wrappers for C/C++ and SystemC environments. With the OVPmodels, OVPsim produces Instruction Accurate (IA) simulation of platforms running unmodified software binaries at typical speeds up to 500 MIPS.

Code is licensed under the Imperas License:

http://www.ovpworld.org/licenses/OVP_Imperas_Software_License_v.1.1.pdf

A2.2 Commercial tools

A2.2.1 SYNOPSYS INNOVATOR

<http://www.synopsys.com/Tools/SLD/VirtualPlatforms/Pages/Innovator.aspx>

“Innovator is a powerful, fully integrated tool environment for developing, running and debugging virtual platforms. It comes with full SystemC™ (IEEE 1666) support. Its main components are:

- Schematic system editor, to instantiate, configure and connect IP model components to rapidly build Virtual Platforms
- Language Editor, to interactively describe behaviour of hardware component using a combination of graphical system constructs and ANSI C and SystemC

- Design Browsers, to quickly reconfigure IP components, to edit run-time scripts and to explore the system connectivity
- Library Manager, to manage and explore re-usable model libraries
- Code Generation, to generate optimized C++ code for the graphical language front-end and to seamlessly invoke host compilers to compile the Virtual Platform
- Hardware debugger, to take run-time control over the target and support system tracing features like hardware breakpoint and single-stepping

Innovator is available as a full developer's license, for customers who want to create their own transaction-level models (TLMs) and/or assemble/modify virtual platforms. Innovator RT is a runtime license that allows software developers to execute virtual platforms created with Innovator.”

A2.2.2 COWARE PLATFORM ARCHITECT

<http://www.coware.com/products/platformarchitect.php>

“For Platform-driven ESL Design, CoWare Platform Architect is the industry's most productive SystemC-based graphical environment for capturing the entire product platform and the dash board for initiating the platform analysis functions. Platform Architect's companion product, CoWare Model Designer, offers a stand-alone subset of these features to provide the industry's most productive SystemC-aware modelling, simulation, and debug environment for capturing complex IP blocks”

A2.2.3 CARBON SOC DESIGNER

http://carbondesignsystems.com/products_socd.shtml

“Carbon SoC Designer is a complete solution for the development, execution, and analysis of virtual system platforms. Carbon SoC Designer delivers value throughout the entire design lifecycle, and provides a virtual platform for:

- System model development and IP integration
- Accurate architectural analysis
- Pre-silicon hardware/software integration”