DELTA

“Development of SpaceWire and CAN SystemC Transaction Level Models for ESA IP cores”

DELTA Executive summary

FINAL

PUBLIC

Date: 25 November, 2010

Prepared by:

Qualtek Sprl.
36 Avenue Gabriel Emile Lebon, B-1160, Brussels, Belgium

For:

ESA-ESTEC
Keplerlaan 1, Postbus 299, 2200 AG Noordwijk, The Netherlands

Page intentionally blank

TITLE	**DELTA Executive summary** **DELTA** **"Development of SpaceWire and CAN SystemC Transaction Level Models for ESA IP cores"**
COPYRIGHT	Copyright © 2010 Qualtek Sprl., Brussels, Belgium

CONFIDENTIALITY		**DISTRIBUTION**
Public	**X**	ESA-ESTEC Qualtek Sprl.
Government Confidential		
Military Confidential		

REVISION HISTORY

Version	**Date**	**Author**	**Status**
1.0	28 September, 2010	**Nikos Mouratidis**	FINAL
1.1	25 November, 2010	**Nikos Mouratidis**	FINAL

List of Changes

Index	Change Description	Revision	Date
1	Modifications suggested at the Final Review	1.1	25-11-10

DOCUMENT APPROVED BY:

PROJECT MANAGER	QUALITY MANAGER	MANAGING DIRECTOR	CUSTOMER'S REPRESENTATIVE
Date:	Date:	Date:	Date:
Name: Dr. N. Mouratidis	Name:	Name:	Name:
Signature:	Signature:	Signature:	Signature:

Table Of Contents

Paragraph Page

1 SCOPE 7
1.1 IDENTIFICATION 7
1.2 DEFINITIONS 8
1.3 DOCUMENT OVERVIEW 8
2 APPLICABLE DOCUMENTS 9
2.1 ORDER OF PRECEDENCE 9
3 DEFINITIONS 10
3.1 TERMS AND DEFINITIONS 10
3.2 C++ CODE TERMS AND DEFINITIONS 10
4 PROJECT OBJECTIVES AND PHASES 11
4.1 SPACEWIRE CODEC 11
4.2 SYSTEMC 11
4.2.1 Transaction Level Modelling 12
4.2.2 TLM 2.0 coding styles 12
4.2.3 Data abstraction levels 13
4.3 DESIGN & DEVELOPMENT PLAN 13
5 PROJECT REQUIREMENTS 15
5.1 CUSTOMER NEEDS AND REQUIREMENTS 15
5.1.1 Performance Requirements 16
5.1.2 Requirements compliance matrix 17
6 VALIDATION AND VERIFICATION 18
6.1 VALIDATION 18
6.2 SIMULATION EXECUTION ENVIRONMENT 18
6.3 BENCHMARK EXECUTION RESULTS 18
6.1 SCALABILITY 20
7 PRODUCED/DELIVERED ITEMS 21
7.1 SOFTWARE 21
7.2 DOCUMENTATION 21
8 CONCLUSIONS 23
LIST OF ACRONYMS 24
REFERENCES 25

List of Tables

Table **Page**

Table 1: Customer requirements....16
Table 2: Performance requirements....17
Table 3: Requirements compliance matrix....17
Table 12: Simulation results - SpaceWire packets = 10, time codes = 10, packet size = 1000....18
Table 16: Simulation results - SpaceWire packets = 100, time codes = 100, packet size = 100...18
Table 23: Simulation results - SpaceWire packets = 1000, time codes = 1000, packet size = 10019
Table 28: Simulation results - SpaceWire packets = 1000, time codes = 1000, packet size = 10000
....19
Table 6: Scalability aspects of SpaceWire TLM....20
Table 5: SpaceWire-CAN SystemC TLM Project documentation....22

1 SCOPE

1.1 IDENTIFICATION

This document presents the executive summary of the DELTA project, which has delivered a SpaceWire TLM 2.0 model. It aims at providing the information necessary for the attainment of a complete overview of the project, its tasks and objectives.

1.2 DEFINITIONS

Shall

Identifies the mandatory requirements on the item or items. Statements which include the ***shall*** statement are to be considered as the only requirements to be tested or otherwise validated.

Should

Suggests an approach that is to be assumed as the approach to be taken and is a reflection of the current status at the time of document issue. Terms such as '***may***' or '***can***' also fall into this advisory but not mandatory category.

Will

Indicates factors that are imposed on the scope of this specification from outside and is to be regarded as a definition of factors that are mandatory by implication.

1.3 DOCUMENT OVERVIEW

This document is identified as follows:

Document type	-	Executive Summary
Document identifier	-	0310-01-006-01
Revision	-	1.1
Issue Date	-	25 November, 2010

2 APPLICABLE DOCUMENTS

The following documents of the exact issue shown form a part of this specification to the extent specified herein.

RFQ/3-12785/09/NL/JK/al	The Request For Quotation provides details on the expected form of the delivered model in terms of use cases, as well as compatibility with or dependency on simulation tools. **Revision. -, November 10, 2009**
OSCI TLM-2.0 LANGUAGE REFERENCE MANUAL	The LRM defines the classification, nomenclature, and details of the applicable coding style with reference to the model use cases. **Revision. JA32, July, 2009**
SpaceWire CODEC IP - User Manual	The User Manual of the RTL IP core provides grounds for the utilisation of the TLM together with the developed transactor in simulations, replacing the RTL IP **Revision. 2.4, March 27, 2009**

2.1 ORDER OF PRECEDENCE

In the event of a conflict between the text of this specification and the references cited herein, **except references to higher-level program-unique specifications for this program**, the text of this specification takes precedence. Nothing in this specification however, supersedes applicable laws and regulations, unless a specific exemption has been obtained.

3 DEFINITIONS

3.1 TERMS AND DEFINITIONS

Bit rate	Number of bits transmitted/received every second
byte	Eight bits of information
DDR	Double data rate. Two bits of data transmitted for each transmit clock period.
FCT	Flow control token. Transmitter sends one FCT when room in receive buffer for eight more N-chars.
N-char	Data character, EOP or EEP.
Null	Control code transmitted by SpaceWire link to keep connection with other end.
SDR	Single data rate. One bit of data transmitted for each transmit clock period.

3.2 C++ CODE TERMS AND DEFINITIONS

The term ".h file" is used in this document to refer to the public interface file because of the long-standing Unix/C practice of using a ".h" extension to the file name, although other operating systems and compilers may require a different extension. The term ".cpp file" is used in this document to refer to an implementation file for similar historical reasons.

The interface file provides a single point of declaration of the data entities and functions provided by a particular module. The interface is #included in all modules which refer to or make use of the data entities and functions provided. Sections of the interface file are not copied into other modules.

The contents of the interface file are surrounded by #ifdef/#endif pre-processor directives in order to avoid problems of multiple inclusion.

4 PROJECT OBJECTIVES AND PHASES

4.1 SPACEWIRE CODEC

One of the principal aims of SpaceWire is the support of equipment compatibility and reuse at both the component and subsystem levels. In principle, a data-handling system developed for an optical instrument, for example, can be used for a radar instrument by unplugging the optical sensor and plugging in the radar one. Processing units, mass-memory units and down-link telemetry systems developed for one mission can be readily used on another mission, reducing the cost of development, improving reliability and most importantly increasing the amount of scientific work that can be achieved within a limited budget.

Integration and test of complex on-board systems is also supported by SpaceWire with ground support equipment plugging directly into the on-board data-handling system. Monitoring and testing can be carried out with a seamless interface into the on-board system.

This Standard addresses the handling of payload data and control information on board a spacecraft. It is a standard for a high speed data link, which is intended to meet the needs of future, high capability, remote sensing instruments and other space missions. SpaceWire provides a unified high speed data-handling infrastructure for connecting together sensors, processing elements, mass-memory units, downlink telemetry subsystems and EGSE equipment.

The purpose of this Standard is:

- to facilitate the construction of high-performance on-board data-handling systems;
- to help reduce system integration costs;
- to promote compatibility between data-handling equipment and subsystems;
- to encourage reuse of data-handling equipment across several different missions.

SpaceWire has taken into consideration two existing standards, IEEE 1355-1995 and ANSI/TIA/EIA-644. SpaceWire is specifically provided for use onboard a spacecraft.

The SpaceWire CODEC, which has been the focus of this development, is responsible for making a connection with the SpaceWire interface at the other end of a link and managing the flow of data across the link. The interface transmits and receives SpaceWire characters which can be link characters (L-Char) or normal characters (N-Char). L-Chars are characters that are used to manage the flow of data across a link (NULL & FCT). N-Chars are the characters that are used to pass information across the link (data characters, EOP, EEP and time-codes).

4.2 SYSTEMC

SystemC is a set of C++ classes and macros which provide an event-driven simulation kernel in C++ (see also discrete event simulation). These facilities enable a designer to simulate concurrent processes, each described using plain C++ syntax. SystemC processes can communicate in a simulated real-time environment, using signals of all the datatypes offered by C++, some additional ones offered by the SystemC library, as well as user defined. In certain respects, SystemC deliberately mimics the hardware description languages VHDL and Verilog, but is more aptly described as a system-level modelling language. It is applied to system-level modelling, architectural exploration, performance modelling, software development, functional verification, and high-level synthesis. SystemC is often associated with Electronic system level (ESL) design, and with Transaction-level modelling (TLM).

The objectives behind the utilisation of a pure software language like C++ for the creation of hardware models are twofold:

- Speed of operation: running a native application on the host computer shall be faster and more efficient than utilising an intermediate layer such as a simulation kernel and associated simulation application
- Early availability of model: in order for the complex systems of today to be developed and brought to market efficiently and promptly, design teams need to work in parallel. This means that a model of the hardware is necessary for the development of the associated

software to commence before the actual hardware development is finished, thus before the RTL hardware model becomes available

In order to speed things up in both of the aforementioned aspects, modelling tends to focus on the transaction rather than bit and cycle level (i.e. Transaction Level Modelling – TLM).

Having IP specified in the form of Transaction Level Models is key for the Hardware/Software co-design: functionality of the IP is represented in a simulation model performing significantly faster than a RTL (or other bit/cycle level accurate) counterpart; at the same time, such a model, which abstracts internal operations of the actual IP, may become available much faster than the IP itself, thus enabling early software development using virtual platforms. Hence, software development may be carried out, and assessed against the TLM model, in parallel to the development of the hardware.

4.2.1 Transaction Level Modelling

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

There has been a longstanding discussion in the ESL community concerning what is the most appropriate taxonomy of abstraction levels for transaction level modelling. Models have been categorized according to a range of criteria, including granularity of time, frequency of model evaluation, functional abstraction, communication abstraction, and use cases. The TLM-2.0 activity explicitly recognizes the existence of a variety of use cases for transaction-level modelling [1], but rather than defining an abstraction level around each use case, TLM-2.0 takes the approach of distinguishing between interfaces (APIs) on the one hand, and coding styles on the other. **The TLM-2.0 standard defines a set of interfaces which should be thought of as low-level programming mechanisms for implementing transaction-level models, then describes a number of coding styles that are appropriate for, but not locked to, the various use cases.**

4.2.2 TLM 2.0 coding styles

Models can be separated into two coding styles depending on the timing-to-data dependency that they must obey. Sometimes these coding styles are confused with “levels of abstraction”, but the presented coding styles can both be used in very detailed models or models that include little detail of the modelled component.

Loosely-timed: also called Programmer’s View (PV) models, these models have a loose dependency between timing and data, and are able to provide timing information and the requested data at the point when a transaction is being initiated. These models do not depend on the advancement of time to be able to produce a response. Normally, resource contention and arbitration are not modelled using this style. Due to the limited dependencies and minimal context switches, these models can be made to run the fastest and are particularly useful for doing software development on a Virtual Platform. Reaching simulation speeds of 50 M Transactions per second allows software developers to boot an OS and run test code in seconds.

Approximately-timed: these models have a much stronger dependency between timing and data. They are not able to provide timing information and/or the requested data when a transaction is being initiated. These models can depend on internal/external events firing and/or time advancing before they can provide a response. Resource contention and arbitration can be modelled easily with this style. Since these models must synchronize/order the transactions before processing them, they are forced to trigger multiple context switches in the simulation, resulting in performance penalties.

4.2.3 Data abstraction levels

In developing the model for a system, and apart from the timing detail this shall adopt, the level of detail adhered to by its interface (as well as its central processing mechanisms) needs also be defined. The greater the detail of data represented and handled by the model, the more elaborate its internal mechanisms shall be, and the lower the simulation speed that can be achieved.

The SpaceWire standard [2] analyses the various levels of data abstraction a compliant system may operate at, depending on the layer of the protocol it corresponds to, going as low as the bit level. For the task at hand, namely the implementation of a CODEC model to operate and communicate with its surroundings at the transaction level, it was deemed necessary to adhere to the exchange level provisioned by the standard; the data granule at that level is the character, with the addition of the flow control mechanisms that realise the protocol of the standard.

In order to have the option for increased simulation speed, at the expense of accuracy and detail, it was decided that an additional data abstraction layer would be explored, that referring to complete packets of data, rather than individual characters. In this mode of operation, the model would be capable of preserving the integrity of the communicated data, but flow control mechanisms would greatly be eliminated, thus giving rise to a model that simulates faster, but does not observe timing in the sense the real hardware shall.

4.3 Design & Development Plan

Figure 1 shows the flowchart of the design and development plan followed for the SpaceWire TLM development project. The project was organised in four workpackages (WPs) deploying the necessary expertise for assuring successful execution and accurate deliverable orientation.

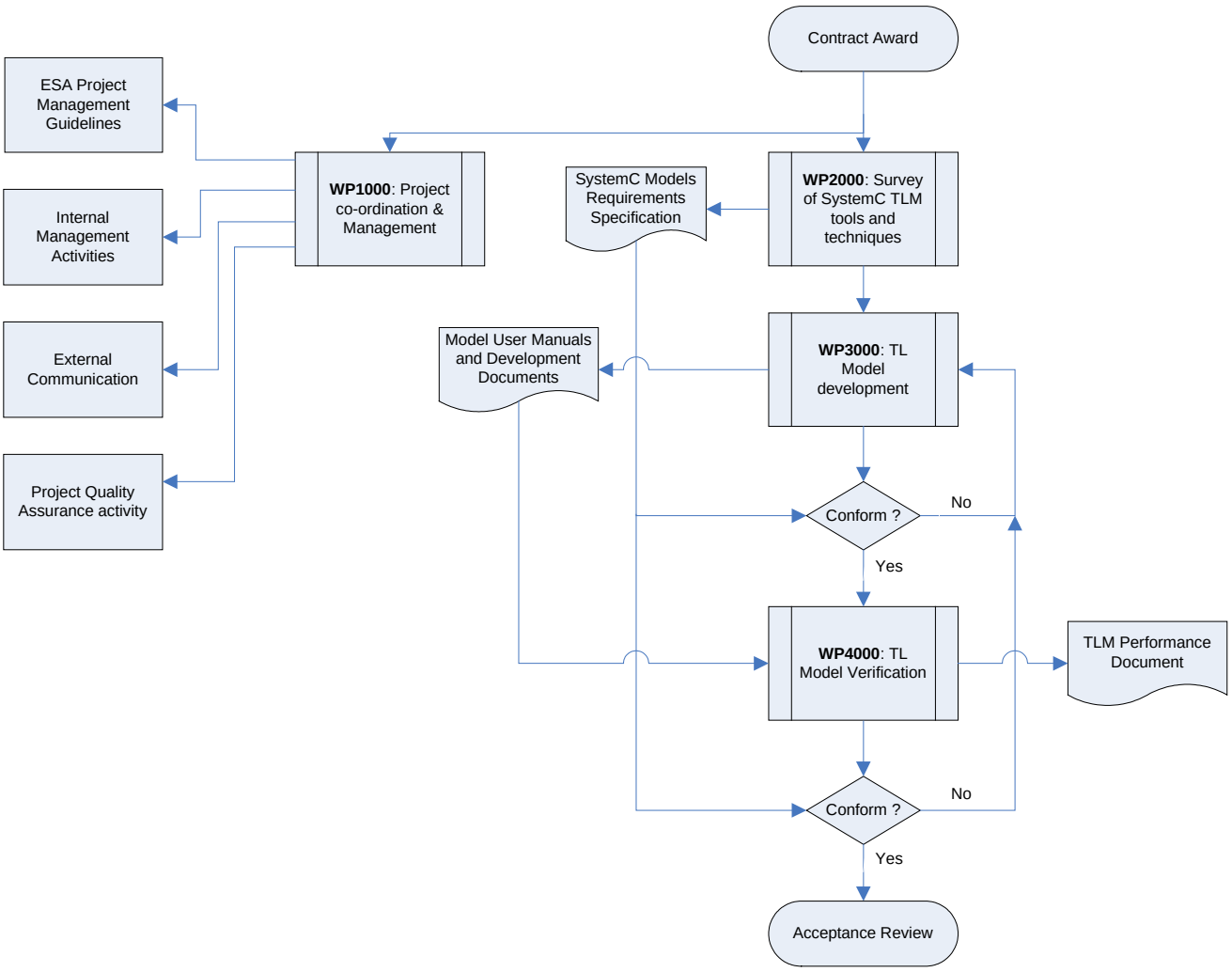

Figure 1: Design and Development Plan flowchart

The Project Co-ordination Management (WP1000) has been monitoring the overall performance of the project, and assessed conformity of the results with the project scope and goals. It ensured the alignment of technical and exploitation / dissemination activities, implemented administration of project resources and assured the quality of its deliverables. Additionally, this WP has implemented all necessary provisions for resolving any technical contingencies. To efficiently accommodate its activities, WP1000 has applied the project management and co-ordination guidelines of ESA.

WP2000 evolved in parallel to WP1000 and had three main and one secondary objective. More specifically, WP2100 dealt with the analysis of the currently available RLT IP core for which a TL model was to be designed. This activity allowed a deeper understanding of the existing implementation and served to instantiate a number of functionality and performance related specifications for the targeted models. The accuracy and performance of the RTL core was assessed, and the corresponding metrics constituted targets used for the validation of the TL models (these have been subject to scaling as dictated by the Agency's requirements). WP2100 was also concerned with the evaluation of tools and methodologies available for work with SystemC, and lead to the definition of the methodology adopted in the project.

WP2200 analysed the different use cases aimed at for the TLM, and obtained requirements emergent from them. The requirements obtained during WP2100 and WP2200 formed the input for WP2300 that deal with the detailed specification of the SpaceWire model. Finally, the task dealt with the deployment of the versioning system that was used during the project (i.e. Subversion).

WP3000 comprised of a pure development effort for the implementation of the model, the associated test benches, and any accompanying scripts, configurations, and documentations. Within WP3000 the verification of the model with respect to its functional and basic timing specifications also took place.

Finally, in WP4000 the validation of the model against its RTL counterparts was realised. The model was assessed in terms of detailed timing, simulation performance, and overall compliance to the RTL IP. For this to be more effective, the elaborate testbench that was created as part of the RTL offering was re-used. In order for this to be attainable, a transactor was developed; this is a VHDL entity that interfaces to both the SystemC TLM and the VHDL testbench, converting abstract transactions to signal transitions, and vice-versa. This choice allowed the already validated simulation environment utilised during the RTL IP verification, to be used for the TLM, validating in this manner its behaviour, data and timing accuracy all at the same time.

5 PROJECT REQUIREMENTS

5.1 CUSTOMER NEEDS AND REQUIREMENTS

The following requirements applied to the product:

Performance Requirement	Comment
General Modelling Style	The targeted TLMs shall be developed in accordance to the Language Reference Manual and Specification ([3], [4]). No hacks or bypasses shall be realised.
TLM Interfaces	The generic payload, defined in the SystemC standard, shall be used to implement the model interfaces. It is the class type offered by the TLM-2.0 standard for transaction objects passed through the core interfaces. It is closely related to the base protocol, which, itself, defines further rules to ensure interoperability when using the generic payload. Use of the generic payload is intended to improve the **interoperability** of memory-mapped bus models, which it does at two levels: a) provides an off-the-shelf general-purpose payload that guarantees immediate interoperability when creating abstract models of memory-mapped buses where the precise details of the bus protocol are unimportant, whilst at the same time providing an extension mechanism for ignorable attributes; b) it can be used as the basis for creating detailed models of specific bus protocols, with the advantage of reducing the implementation cost and increasing simulation speed when there is a need to bridge or adapt between different protocols, sometimes to the point where the bridge becomes trivial to write. The payload shall be extended as necessary due to the development of the core(s), without, however, losing adherence to the standard.
Models to be implemented	A SpaceWire-b CODEC shall be implemented in accordance to [5]. The Agency reserves the right to accept or reject the implementation of the CAN controller model in accordance to [6], in due course.
Self containment	The development carried out in this project shall be focused on the self-containment of the final deliverables. The models produced, as well as any compilation or simulation scripts, test benches or accompanying software entities shall not make use of any libraries or other resources that are not commonly and freely available to users.
CAD tool independence	The complete deliverable shall be independent of specific CAD tools. Effort shall be placed onto attaining a set of software entities that produce identical runtime results when compiled by a variety of tool suites and executed in a variety of platforms, with the only provision that a SystemC kernel is available
SystemC TL Model User's Manual	A detailed User Manual shall be delivered for each of the delivered TL models, allowing complete adoption and deployment of the model in any appropriate system-level model. A Table of Contents for the User Manual shall be created by the Contractor and agreed with the Agency in the course of the first part of the project (before the MTR).

Performance Requirement	Comment
Build system	The CMake cross-platform, open-source build system shall be used as the basis for the build infrastructure delivered with the developed TLMs. In case changes are made to the system to cater for intricacies of the final deliverable, these shall constitute part of the deliverable and shall be provided as a whole to the end user.
Version Management	Subversion is the most prominent candidate for deployment as the version control and software configuration management platform for this project. It is currently in operation at the premises of The company, and can support a number of configuration alternatives to allow streamlined access to the code for both developers and reviewers (or users).
Code Quality	All the source code shall be written according to the ESA BSSC(2000)1 standard as per [7].

Table 1: Customer requirements

5.1.1 Performance Requirements

The following performance requirements applied to the proposed product:

Performance Requirement	Comment
Coding Style	Both Approximately-timed (AT) and Loosely-timed (LT) flavours of the models shall be created. These shall aid the deployment of the models in different use cases, whether software development with lower timing accuracy expectations, or architectural exploration with increased accuracy needs and higher simulation time headroom. In order to streamline deployment of the models, and allow users to focus on the actual use rather than on the selection, setup and instantiation, the type of model shall be selectable through a generic configuration interface that shall be delivered together with the models. Moreover, effort shall be placed towards fusing the different flavours into delivering a single set of files for each of the implemented models, which shall cater for the internal selection of the desired timing accuracy. Selection might even take place at runtime, without the need for a re-compilation of the model. The final solution shall depend on the achievable performance characteristics of the models.
Functional Behaviour	The SystemC models shall be built using not only the relevant specification documents, but also the existing RTL implementations as a reference. The objective of the development shall not simply be the delivery of a model fully compliant to the specification, but also identical, in terms of functionality, to its RTL counterpart.
Timing Behaviour	The TL models shall report timing differing from the timing of the corresponding RTL counterparts by no more than 20%. In order to ascertain the validity of the obtained values, the exact same test benches shall be executed by both implementations. Appropriate transactors may be implemented for the SystemC

Performance Requirement	Comment
	models in order for the VHDL test benches to be brought forward to the SystemC simulation sessions if deemed necessary.
TLM simulation performance	The performance of the developed models shall be measured and reported, based on test benches that provide a minimum of 90% test coverage on the functionality offered by the models.
Scalability	The models shall be designed and implemented with the notion of scalability in mind. Effort shall be placed towards obtaining models that allow multiple instantiation on the same workstation with the minimum possible memory footprint and CPU load overhead for each additional instantiation. Performance measurements shall also be made on the basis of the multiple instantiation and conclusions shall be drawn as to the efficiency of the result.

Table 2: Performance requirements

5.1.2 Requirements compliance matrix

The following table summarises the requirements presented in the RFQ by the Agency, and specifies the level of compliance to them by the implementation:

RFQ SoW requirement reference	Description	Compliance of proposal
R.1	General Modelling Style	Compliant
R.2	TL Models' Interfaces	Compliant
R.3	SystemC TL Models to Be Implemented	Compliant
R.4	Functional Behaviour	Compliant
R.5	Timing Behaviour	Compliant
R.6	TL Model Simulation Performance	Compliant
R.7	Self-containment	Compliant
R.8	CAD tool independence	Compliant
R.9	SystemC TL Model User's Manual	Compliant
R.10	Build System	Compliant
R.11	Version Management	Compliant
R.12	Code Quality	Compliant

Table 3: Requirements compliance matrix

6 VALIDATION AND VERIFICATION

6.1 VALIDATION

In order for the project results to be validated, the developed product had to be assessed against a "known good configuration". As mentioned previously, the already validated simulation testbench of the RTL offering was re-used, thus providing direct access to the assessment of the simulation results for the TLM IP. The testbench realises a reference implementation of the SpaceWire CODEC, making use of stimuli generation and validation scripts, that take into account the timing and data accuracy, as well as the compliance to the protocol set out in the standard. The TLM was successfully simulated against the reference implementation.

6.2 SIMULATION EXECUTION ENVIRONMENT

Apart from the validation of the results, the performance benefit attained by the utilization of the TLM in place of the RTL IP in simulations was assessed. Further simulations were run, identical for the two models, comparisons were made and conclusions were drawn.

The RTL and TLM models simulations were executed on a computer running the 64-bit version of Ubuntu Linux 10.0.4. The computer has a 2.10 GHz Intel dual-core CPU and 4GB of RAM. During simulation benchmarking the only user application running on the computer was the relating simulator. For RTL-only simulations the Modelsim simulator version 6.5d was used, whereas for SystemC-only simulations the open-source OSCI simulator was utilized.

6.3 BENCHMARK EXECUTION RESULTS

The following tables present some sample simulation results extracted from the "SpaceWire TL Model Performance Document" [8].

SpaceWire CODEC Model	Simulation Time (milliseconds)	Speedup (x factor)	Simulation Duration (nanoseconds)	Accuracy (%)
RTL	6856	-	10558150	-
AT-exchange level	70	97.9	10558800	99.9938
LT-exchange level	20	342.8	10558800	99.9938
AT-packet level	50	137.1	11013900	95.6834
LT-packet level	20	342.8	10513300	99.5752

Table 4: Simulation results - SpaceWire packets = 10, time codes = 10, packet size = 1000

SpaceWire CODEC Model	Simulation Time (milliseconds)	Speedup (x factor)	Simulation Duration (nanoseconds)	Accuracy (%)
RTL	7190	-	10819350	-
AT-exchange level	85	84.6	10713600	99.0226
LT-exchange level	35	205.4	10713600	99.0226
AT-packet level	60	119.8	10279700	95.0122
LT-packet level	20	359.5	10229100	94.5445

Table 5: Simulation results - SpaceWire packets = 100, time codes = 100, packet size = 100

SpaceWire CODEC Model	Simulation Time (milliseconds)	Speedup (x factor)	Simulation Duration (nanoseconds)	Accuracy (%)
RTL	71065	-	107929350	-
AT-exchange level	850	83.6	106878600	99.0264
LT-exchange level	300	236.9	106878600	99.0264
AT-packet level	600	118.4	101937700	94.4485
LT-packet level	175	406.1	101330500	93.8860

Table 6: Simulation results - SpaceWire packets = 1000, time codes = 1000, packet size = 100

SpaceWire CODEC Model	Simulation Time (milliseconds)	Speedup (x factor)	Simulation Duration (nanoseconds)	Accuracy (%)
RTL	6382944	-	10502929350	-
AT-exchange level	62853	101.6	10501885800	99.9901
LT-exchange level	17690	360.8	10501885800	99.9901
AT-packet level	40925	156.0	10006280500	95.2713
LT-packet level	9192	694.4	10006280500	95.2713

Table 7: Simulation results - SpaceWire packets = 1000, time codes = 1000, packet size = 10000

The simulation results clearly highlight the expected outcome: longer simulation intervals for comparable data (e.g. same packet size) have no noticeable impact either on the simulation execution time improvement or on the timing accuracy of the TLMs. With respect to model simulation execution times, the measured performance improvement ratios are also shown in the tables, and range from a speedup factor of 80 (for the AT, exchange level mode), to more than 700% (for the LT, packet level mode).

The benchmark results indicate that the model implementation sits well beyond the 20% timing accuracy requirement of the DELTA project even in the case of the packet-level models. The exchange level models achieve an accuracy of 98.00% to 99.99% when compared to their RTL counterpart. The packet level models are slightly less accurate, with their worst-case precision – LT, packet level model – being in the range of 93% to 99% of the RTL model. There is an exception to the accuracy attained by the packet level model, which is associated with the exchange of less than 3 packets within a simulation. The reason for this twofold: a) there is a time offset introduced by the initialisation phase of the testbench (polling and acquisition of register settings) which, for small simulation durations is relatively significant, but diminishes as simulation durations become longer; b) the design of the packet mode section of the model makes use of a FIFO polling interval that is calculated as a function of the packet size used, an introduces a time offset to the simulation equal to this interval; for large packet sizes (e.g. 10K characters) and low number of packets (e.g. 1 or 2), this offset constitutes a significant part of the overall simulation time (in the order of 30%), thus reducing attained accuracy (simulation using 1 packet of 10K characters gives rise to an accuracy of 48%; for 2 packets accuracy rises to 71.62%, while for 3 packets it becomes 95.35%).

6.1 SCALABILITY

The DELTA SpaceWire TLM has been rigorously tested to assert that multiple instantiation within a simulation platform is possible, and causes no problems or presents no peculiarities. Regarding the impact on the host, different test setups were constructed, utilising different numbers of instances of the CODEC respectively; the associated memory footprints were logged and are reported in the following table:

SpW CODEC instances number	Virtual Memory Size (KBytes)	Resident Set Size (KBytes)
2	14824	**2472**
4	16772	**2948**
8	20536	**3880**
16	28048	**5716**
24	35588	**7496**
32	43096	**9256**

Table 8: Scalability aspects of SpaceWire TLM

7 PRODUCED/DELIVERED ITEMS

7.1 SOFTWARE

The following **new** software items have been **developed and delivered** in the course of the project as parts of the TL models development.

Item	Description	Reference WP	Milestone	Responsible Partner
SystemC TL Model Library	The SystemC TL Models developed for the SpaceWire and (if selected by the Agency in due course) the CAN RTL IP cores.	WP3000, WP4000	T1, T2	The company
TLM test benches	Test benches for the delivered models	WP3000, WP4000	T1, T2	The company
Build system scripts	Scripts and related files (e.g. configuration) needed by the model user to build and deploy the delivered models within the framework of the simulation kernel	WP3000, WP4000	T1, T2	The company

The following **existing** software items are **supplied and integrated** within the IP-core architecture.

Item	Description	Responsible Partner
SystemC library (v2.2.0)	SystemC libraries and kernel	The company
TLM library (v2.0.1)	TLM-2.0 library (proof of concept implementation for the OSCI TLM-2.0 standard)	The company
Greensocket-4.1.0 library	TLM library defining custom protocol and payload	The company

All existing software items may be procured free-of-charge and therefore are offered without extra costs.

7.2 DOCUMENTATION

The following table summarises the documentation that has been made available to ESA and the public (where appropriate, i.e. the Web site), in the course of the project. As is required by the ITT text, all documents are compatible at least with MS Word or Adobe Acrobat forms and will be available electronically.

ID[1]	Deliverable	Submission dates	Description	Format
PBC	Project detailed bar chart	Within proposal	The project detailed bar chart is provided in the form of a Gantt Chart and outlines graphically the various WPs their evolution and association between each other and with the foreseen milestones of the project. If deemed necessary it is subject to update.	Electronic

[1] All documents prescribed in the SOW of the tender bear a valid abbreviation in the ID column of the table. Those documents not indicating such abbreviation constitute interim reports produced in the context of the individual WPs and have been incorporated within the rest official documents. Their inclusion in the list has the scope of outlining in more detail the evolution of the project workplan with respect to the requirements set by the RFQ.

ID[1]	Deliverable	Submission dates	Description	Format
MOM	Minutes of Meetings	10 days after every meeting attended by the Agency	Documents the discussions taking place in project meeting alongside action points, responsibilities and decisions.	Email
AIL	Action Items List	10 days after every meeting attended by the Agency	Shall accompany meeting minutes and shall maintain a list of active action items.	Electronic
DL	Documents List	10 days after every meeting attended by the Agency	Shall accompany meeting minutes and shall maintain an updated list of project documents that are to be delivered within the course of the project	Electronic
MPR	Monthly Project Report	End of every month until AR	Reports the evolution of the active project WPs.	email
-	SystemC Models requirement specification (WP2400)	MTR	This document will discuss requirements and specification of the targeted TL models	Electronic
-	TL Models' User Manuals (WP3000)	MTR	This document describes the model's interface and functions and its use from the perspective of the system architect and the programmer, including examples.	Electronic
-	TL Models' Development Documentation (WP3000)	MTR	The document 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 of the RFQ.	Electronic
-	IP Performance Document (WP4000)	AR	This document shall assess the timing accuracy and simulation speed of each SystemC model, comparing them with their RTL implementation counterparts.	Electronic
-	Final Report and Executive Summary (WP1000)	AR	This document shall provide concluding remarks and summarize the findings of the contract concisely and, informatively.	Electronic

Table 9: SpaceWire-CAN SystemC TLM Project documentation

8 CONCLUSIONS

The preceding text has presented a summary of the DELTA project. The aim of the document has been a presentation of the background, objectives, requirements and constraints of the project, and the outline of its quantitative and qualitative results. It essentially glides over documents delivered during the course of the project, depicting their highlights.

LIST OF ACRONYMS

API	Application Programming Interface
AT	Approximately-Timed
ASIC	Application-Specific Integrated Circuit
CA	Cycle-Accurate
CODEC	CODer DECoder
EDA	Electronic Design Automation
FCT	Flow Control Token
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IDE	Integrated Development Environment
IP	Intellectual Property
LRM	Language Reference Manual
LT	Loosely-Timed
OSCI	Open SystemC Initiative
RTL	Register Transfer Level
SCV	SystemC Verification Library
SoC	System on Chip
TLM	Transaction Level Modelling

REFERENCES

[1] Requirements Specification for TLM 2.0, Version 1.1, September 16, 2007

[2] Space engineering, SpaceWire – Links, nodes, routers and networks,
ECSS-E-ST-50-12C_31July2008

[3] IEEE Standard SystemC Language Reference Manual, Available at http://www.systemc.org

[4] aSCI TLM 2.0 Standard, Available at http://www.systemc.org

[5] SpaceWire - Links, Nodes Routers and Networks, ECSS-ST-50-12C, Available at http://www.ecss.nl

[6] Controller Area Network, ISO 11898, Available at http://www.ecss.nl and http://www.iso.org

[7] ESA C and C++ Coding Standard BSSC(2000)1, Available at
ftp://ftp.estec.esa.nllpublwmlwme/bssc/bssc2000(I)HO.PDF

[8] SpaceWire TL Model Performance Document, 0310-01-007-01, Rev. 1.1, DELTA Project, Qualtek Sprl, 25 November, 2010