Modelling of the SpaceWire Communication Protocol

Valentin Olenev, Ilya Korobkov, Nikita Martynov, Arkady Shadursky
Institute HPCNT
Saint-Petersburg University of Aerospace Instrumentation
Saint-Petersburg, Russia
Valentin.Olenev@guap.ru, Ilya.Korobkov@guap.ru,
Nikita.Martynov@guap.ru, Arkady.Shadursky@guap.ru

Abstract

Modelling of the communication protocols becomes highly needed for the large developing companies. It is not only helps to make important specification decisions, but to check the standard and also save money and a lot of time on a hardware implementation stage.

Current article gives an overview of activities in a SpaceWire communication protocol modelling. Also we describe the joint modelling of SpaceWire and transport layer protocols such as RMAP and STP. We describe the model itself, the testing process and present the results, which we get during the modelling.

INDEX TERMS: MODELLING, PROTOCOLS, SYSTEMC, SPACEWIRE, RMAP, STP.

I. INTRODUCTION

Modelling takes more important role in the development process as a solution to perform detailed check of the specification and project verification to a stage of physical realization. Modelling could be applied to the different stages of system design and a number of different languages and modelling platforms can be used. The models could have different structures and internal logic [6].

SpaceWire is a perspective and fast-developing communication standard. Nowadays there are number of large companies take part in the development process. The standard is supported and implemented to the modern spacecrafts by European Space Agency, NASA (USA) and JAXA (Japan) [5]. There are a number of transport layer protocols which are applied for use in tandem with SpaceWire.

We implemented a layered model of a SpaceWire, RMAP and STP protocols for testing of the specifications and standards. This article gives an overview of our activity and shares the results.

II. MAIN PART

A. Modelling for the specification development process

Main objectives in the course of the project specification are definition and specification of basic functions of the system and creation of an executable system model. This system model is used to verify the correctness of system operation from the functional point of view, and also to estimate necessary hardware resources and define system architecture. It is used also to check the mechanisms and approaches in the specification for correctness and compatibility.

At this stage following problems are solved:

- Creation of the system functional model, description of the system in terms of the algorithms and functions which it should implement, without binding to further implementation techniques;
- System modelling in its operational environment with real data and signals;
- Specification of the system architecture in terms of necessary resources and hardware-software realization of functional model.

With the executable system specification, behavioral models and the general architecture, some stages of the further design, verification and topological implementation of the system could run in parallel.

The general specification development flow at this stage is shown at the fig. 1.

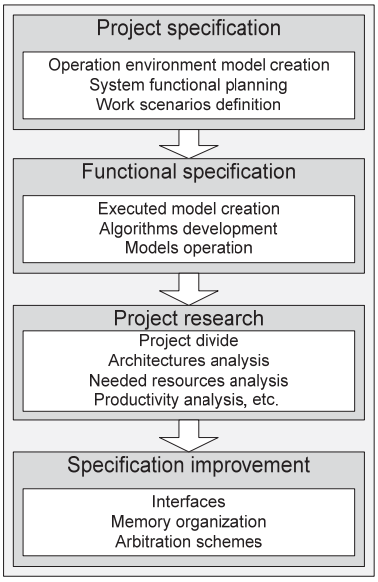

Fig. 1. Specification development flow

Then the functional specification of the system is created. Its purpose is to define and model the system functionality in terms of its operation algorithms. System behavior as a whole or its separate blocks could be set and modeled here. At this level of system functionality used to be modeled with real data and signals [6, 9, 11].

B. SpaceWire communication protocol

SpaceWire is a spacecraft communication network based on an IEEE 1355 communication standard. The purpose of SpaceWire is:

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

Within a SpaceWire network the nodes are connected through:

- low-cost;
- low-latency;
- full-duplex;
- point-to-point;
- high-rate serial links;
- packet switching wormhole routing routers.

Note that within SpaceWire, a separate definition of levels that are encompassed by the Physical Layer and Data Link Layer of the OSI model are defined. These are as follows:

- ***Physical level:*** Defines connectors, cables, cable assemblies and printed circuit board tracks;
- ***Signal level:*** Defines signal encoding, voltage levels, noise margins, and data signaling rates;
- ***Character level:*** Defines the data and control characters used to manage the flow of data across a link;
- ***Exchange level:*** Defines the protocol for link initialization, flow control, link error detection and link error recovery;
- ***Packet level:*** Defines how data for transmission over a SpaceWire link is split up into packets;
- ***Network level:*** Defines the structure of a SpaceWire network and the way in which packets are transferred from a source node to a destination node across a network. It also defines how link errors and network level errors are handled.

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.

This Standard specifies the physical interconnection media and data communication protocols to enable the reliable sending of data at high-speed (between 2 Mb/s and 400 Mb/s) from one unit to another.

SpaceWire provides a means of sending packets of information from a source node to a specified destination node. SpaceWire does not specify the contents of the information packets.

SpaceWire provides a unified high speed data handling infrastructure for mutual connection of sensors, processing elements, mass memory units, downlink telemetry subsystems and EGSE equipment [3].

C. Streaming Transport Protocol

Streaming Transport Protocol (STP) is a new connection-oriented transport layer protocol. STP is used for streaming data transfer over SpaceWire link.

STP supports coherent data sources that can synchronous transmit data flows. It is oriented for asymmetric transport connection use: there is a host (master), and the slave device on the other hand. The host device is an initiator of a transaction session. It performs set-connection control, initialization of connection parameters and flow control.

STP has three packet-commands formats:

- packet-commands with parameters;
- packet-commands without parameters;
- packet of data

STP has nine types of commands:

- open_connection
- ack_connection
- set_connection
- close_connection

- ack_close_connection
- finish_connection
- start_transfer
- stop_transfer
- credit_transfer

STP is developed for streaming data transmission from a local memory of the host device. The transmission is performed every specified time period. Host system opens new connection and run the transfer. Slave system independently determines the moment to transmit next portion of data to the host [2].

Thereby, STP allows initializing of the connection to control data traffic. Also it allows checking order of arriving packets and its correctness [1].

D. Remote Memory Access Protocol

The remote memory access protocol (RMAP) has been designed to support a wide range of SpaceWire applications. Its primary purpose however is to configure a SpaceWire network, to control SpaceWire units and to gather data and status information from those units. RMAP may operate alongside other communications protocols running over SpaceWire. RMAP may be used to configure SpaceWire routing switches, setting their operating parameters and routing table information. It may also be used to monitor the status of those routing switches. RMAP may be used to configure and read the status of nodes on the SpaceWire network. For example, the operating data rate of a node may be set to 100 Mbits/s and the interface may be set to auto-start mode. For simple SpaceWire units without an embedded processor, RMAP may be used to set application configuration registers, to read status information and to read or write data into memory in the unit. For intelligent SpaceWire units RMAP can provide the basis for a wide range of communications services. Configuration, status gathering, and data transfer to and from memory or mailboxes can be supported.

RMAP is used to write to and read from memory, registers, FIFO memory, mailboxes, etc, in a destination node on a SpaceWire network. Input/output registers, control/status registers and FIFOs are assumed to be memory mapped, so are accessed as memory. Mailboxes are indirect memory areas that are referenced using a memory address. All read and write operations defined in the RMAP protocol are posted operations i.e. the source does not wait for an acknowledgement or reply to be received. This means that many reads and writes can be outstanding at any time. It also means that there is no timeout mechanism implemented in RMAP for missing acknowledgements or replies. If an acknowledgement or reply timeout mechanism is required it must be implemented in the source user application.

Write commands can be acknowledged or not acknowledged by the destination node when they have been received correctly. If the write is to be acknowledged and there is an error with the write request, the destination will send an error code to the source that sent the command. The error can only be sent to the source if the write command header was received intact, so that the destination that detected the error knows where to send the error message. If no acknowledgement is requested then the fact that an error occurred may be stored in a status register in the destination node. Write commands can perform the write operation after verifying that the data has been transferred to the destination without error, or it can write the data without verification. To perform verification on the data requires buffering in the destination node to store the data while it is being verified, before it is written. The amount of buffering is likely to be limited so verified writes ought only be performed for relatively short sets of data, that will fit in the available buffer at the destination. Longer writes can be performed but without verification prior to writing. Verification in this case is done after the data has been written. Verified writes should always be used when writing to configuration or

control registers. The acknowledged/non-acknowledged and verified/non-verified options to the write command result in four different write operations:

- Write non-acknowledged, non-verified;
- Write non-acknowledged, verified;
- Write acknowledged, non-verified;
- Write acknowledged, verified.

The read command reads one or more bytes of data from a specified area of memory in a destination node. The data read is returned in a reply packet.

The read-modify-write command reads a register (or memory) returning its value and then writes a new value, specified in the command, to the register. A mask can be included, in the command, so that only certain bits of the register are written. This provides an atomic operation that can be used for semaphores and other handshaking operations [4, 8].

E. SpaceWire SystemC Model

For the modelling of SpaceWire we choose the modelling per layer method. The layered modelling of the protocols stack is shown on fig. 2. In this method the decomposition of one task into a set of simpler tasks or modules is heavily used technique. Decomposition procedure includes accurate definition of functions for each module that has to solve particular separate problem and clear specification of the interfaces between them. The consequence is a logical simplification of the problem. Another result is an ability to update these separate modules without changing other parts of the system.

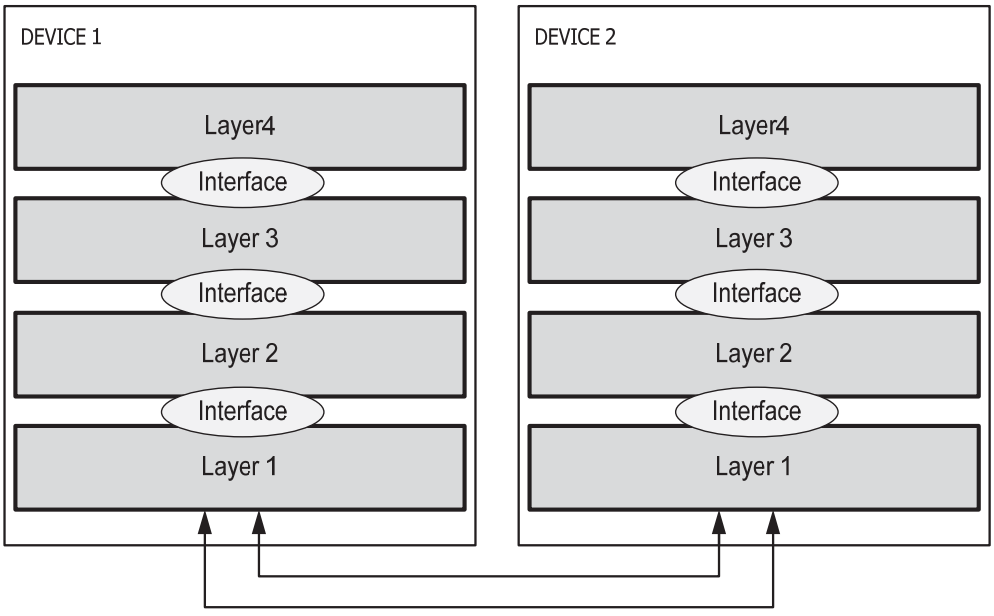

Fig. 2. Scheme of per layer modelling for the SpaceWire testing

The network interaction between two devices can be presented in the form of hierarchically organized set of modules. The process of interaction between two devices can be described by a set of rules for every corresponding pair of modules placed at the corresponding layers of interaction.

Thus for the proper modelling of the protocol stack it is recommended to adopt the following procedure. Define basic split to the functional layers for the given protocol stack. When split is done, each layer should be addressed separately. For every layer we have to define its functional mission and communication interface with other layers and then it is

possible to divide it into smaller blocks. After that it is possible to start implementing layer's model starting from simple blocks, which aggregates to more and more complex structures. Only after definition of each layer the full scheme of earlier specified interaction should be implemented.

The purpose of creation of such a model is to check the protocol characteristics. In the beginning of model creation it is necessary to check the specification and algorithms on presence of errors and occurrence of impasses. Also it is necessary to check conformity of model and the protocol.

This way of modelling is better for the protocol testing on presence of errors and general debugging, as here we model the detailed work mechanism. It is convenient to model communication of only two devices, which is should be enough for most of the purposes of initial study [7, 10].

The implemented SpaceWire model includes four standardized layers. The bottom layer is Character layer, than the Exchange, Packet and Network layers are implemented above. To provide an exchange between two or more nodes we use the other model with implementation of Physical and Signal layers of the standard and it is out of the scope of current article. The Traffic Generator is located over the Network layer to generate the outgoing traffic and analyze the incoming one. The current structure of the SpaceWire node model is shown on fig. 3.

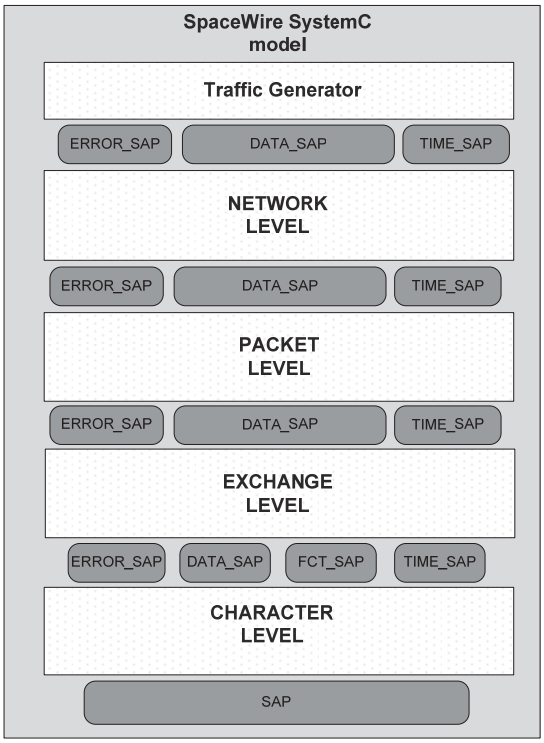

Fig. 3. SpaceWire SystemC model architectural diagram

Currently the point-to-point connection for debugging and testing of model is used. Two instances of SpaceWire protocol stack are connected to each other via the channel. This channel has SystemC implementation also.

Each layer of designed model is implemented in respect to specification and includes all required functionality. The functionality of each layer is distributed between internal blocks, which divide layer for efficient and demonstrable modelling. Internal blocks interact with each other via ports and interfaces.

The mechanism of interaction between layers is out of the SpaceWire specification scope, so we developed Serves Access Points (SAPs) and used it in a model. These SAPs are located between layers of the model. It supports transmission of service information for correct and accurate work of the model, and also supports user data transmission (from Traffic Generator). An amount of SAPs between layers is various; it depends on a number of different information types for layer exchange. For example, between Character layer and Exchange layer is four SAP instances which support bidirectional information exchange:

- for user data transmission which initiated by user application;
- for time-code transmission which initiated by user application;
- for error-code transmission in the case of errors during the data transmission;
- for flow control tokens transmission initiated by Exchange layer;

Between all other layers, three SAPs are located:

- for user data transmission;
- for time-code transmission;
- for error-code transmission;

SpaceWire standard has no specified Transport layer. But there are number of specifications of transport protocols operating over SpaceWire. We model two of them: RMAP and STP. If RMAP is one of the widely known transport layer protocols for SpaceWire, but the other one, STP, is in the process of development and research.

F. STP and RMAP modelling on top of SpaceWire

Let's consider the work of STP and RMAP blocks over SpaceWire model. The example is depicted on fig. 4.

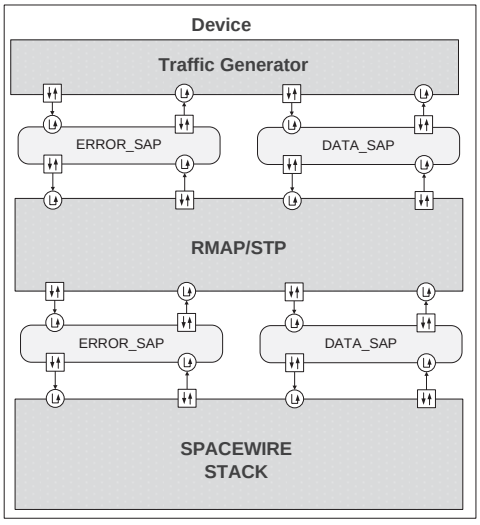

Fig. 4. Architectural diagram of STP (or RMAP) + SpaceWire model

STP and RMAP modules interact with SpaceWire model by means of two SAPs: ERROR SAP and DATA SAP. The ERROR SAP is used for service information transfer. Service information contains some management information for Upper layer (or Traffic Generator) and for SpaceWire model. For example, information about errors, which occurred during the data transfer, information about new data transfer rate, which SpaceWire model should support after data transmission. The second SAP is DATA SAP. It is used for packets transmission from SpaceWire model to STP or RMAP model.

We model and test STP and RMAP protocols based on sending of data packets and commands to STP and RMAP blocks. The following modelling example is implemented. We send erroneous packets and commands to STP and RMAP blocks and analyze how protocols

react on it. Thereby, we can find errors and dead-lock situations in specification of these protocols itself.

Preliminary modelling of the protocol protects a developer from the specification defects and consequently from the errors which could occur during the ready devices execution. Therefore, without modelling of protocol it is hard to evaluate and analyze all critical cases that can be occur during the work with device.

G. Testing methods

There are a lot of methods to test device operation. Implementation of one instance of device model and testing couldn't give us enough information to recognize the correct operation of model. It is because of the huge test bench needed for traffic generation and its analysis. So the most convenient method is a point-to-point connection of two models. Doing so each model can generate the traffic, but the analysis of the outgoing data mostly would be done by the receiving sides of the both models according to the mechanisms defined in the specification. Current method imply the link start, link connection establishment, data flow initialization and interactions of both devices. Such kind of testing method is shown on fig. 5:

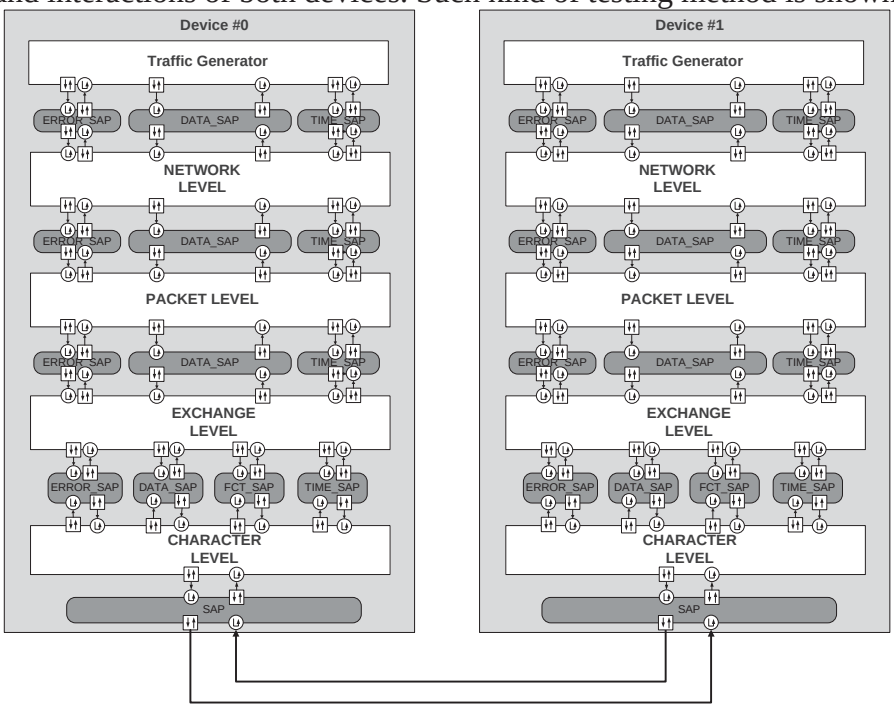

Fig. 5. An example of SpaceWire point-to point modelling case

So, such modelling method gives a number of benefits:

- The model could test all the internal mechanisms;
- There is a possibility to trace the whole data exchange process in both directions;
- The traffic analysis is done by the model itself according to the specification;
- There is a possibility to test not only the protocol itself, but to see how the other protocols could operate with it, work on top or in the bottom of the protocol.

There are traffic-generators on top of device models for data generation. It is flexible module, used for generating various types of data and sending it to device for processing.

Generated data step by step passes all the layers of stack. Then it is translated to remote node via the channel. The data generated by Traffic Generator is compared with data received on a remote node. If a part of the message is not received, then message is corrupted and an

error is detected. If the received message is correct, it means that the test is passed and we can start the next one with another portion of data and other settings.

One of the most important advantages of this method is that point-to-point connection mode can simulate work of the model very close to the real devices communication. Traffic-generators create various situations of data exchange. Necessary corrections are applied for every unique case of modelling. Though such modelling case we could find errors in the model or in the specification itself and the developer could fix it.

III. Conclusion

A huge amount of bugs and inconsistencies of the specification could be found during the modelling. Also such kind of work could help to find the ambiguous places in the specification for the further clarifying of the standard. Modelling makes easier the protocols development on the specification stage. It gives important information to make major specification decisions.

As an example in current article we presented our work on SpaceWire model and two Transport protocols (STP and RMAP) working on top of it. Such a modelling already gave important results. We found a number of specification errors and wrongly described places in the specification. We prepared a set of additions and clarifications to the specification that were presented to the SpaceWire workgroup. Also our SpaceWire model assists development of STP protocol, which would be presented on the next International SpaceWire Conference.

References

[1] SUAI, specification STP, “Streaming Transport Protocol”, *to be presented at International SpaceWire Conference 2010*, 2010.

[2] F.Shutenko, E.Suvorova, E.Yablokov, M. Gillet “Low Power Protocols Development and Implementation”, *Proceedings of 6th Seminar of Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program (p. 113)*, Saint-Petersburg University of Aerospace Instrumentation (SUAI), Saint-Petersburg, 2009.

[3] ESA (European Space Agency), standard ECSS-E-50-12A, “Space engineering. SpaceWire – Links, nodes, routers and networks. European cooperation for space standardization”, ESA Publications Division ESTEC, Noordwijk, The Netherlands, 2003.

[4] ESA, specification RMAP, “SpaceWire Remote Memory Access Protocol”, University of Dundee, Applied Computing, Dundee, DD1 4HN, Scotland, UK, 2006.

[5] Y. Sheynin, T. Solokhina, Y. Petrichkovitch « SpaceWire technology for the parallel systems and onboard distributed systems”, ELVEES, 2006, *http://multicore.ru/fileadmin/user_upload/mc/publish/SpW-part1.pdf.*

[6] V. Olenev, Y. Sheynin, E. Suvorova, S. Balandin, M. Gillet, “SystemC Modelling of the Embedded Networks”, *Proceedings of 6th Seminar of Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program (pp. 85-95).* Saint-Petersburg University of Aerospace Instrumentation (SUAI), Saint-Petersburg, 2009

[7] V. Olenev, “Different approaches for the stacks of protocols SystemC modelling analysis”, *Proceedings of the Saint-Petersburg University of Aerospace Instrumentation scientific conference (pp. 112-113)*, Saint-Petersburg University of Aerospace Instrumentation (SUAI), Saint-Petersburg, 2009.

[8] M. Hines, J. Wang, K. Gopalan “Distributed Anemone: Transparent Low-Latency Access to Remote Memory”, 2006, *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.2259&rep=rep1&type=pdf.*

[9] V. Nemydrov, G. Martin, "Systems-on-chip. Design and evaluation problems," *Technosphera*, 2004.

[10] A. Jantsch, “Modeling Embedded Systems and SoCs”, *Morgan Kaufmann Publishers, Stockholm*, 2004

[11] A. Bykhteev “Methods and facilities for systems-on-chip design,” ChipInfo, 2008, *http://www.chipinfo.ru/literature/chipnews/200304/1.html.*