2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Virtual Prototyping in SpaceFibre System-on-Chip Design

System-level design

Elena Suvorova, Nadezhda Matveeva, Ilya Korobkov, Alexey Shamshin, Yuriy Sheynin

Saint-Petersburg State University of Aerospace Instrumentation,
Saint-Petersburg, Russian Federation
(*suvorova@aanet.ru nadezhda.matveeva@guap.ru ilya.korobkov@guap.ru alexey.shamshin@guap.ru sheynin@aanet.ru*)

***Abstract*—The paper provides a description of our experience in the field of the System-On-Chip (SoC) design. We use different languages, methodologies and tools in the system level development and virtual prototyping of a new SpaceFibre SoC – Network Memory Controller. They are: SystemC, TLM 2.0 languages, OVP technology and Cadence VSP software. They allowed to perform simulation and to analyze performance of the Controller. The received results helped to find limitations of the examined SoC architecture. It saved time and resources due to Cadence and Imperas tools and methodologies. This approach allowed to develop and to debug our tests and SoC embedded software before the device release.**

Keywords—SystemC; OVP; TLM (Transaction Level Modeling); SoC design; Virtual prototyping

I. INTRODUCTION

Simulation is widely used in the system level development of a System-On-Chip (SoC). It can be used to evaluate performance of designed system. The simulation results could be used in the process of decision making on what functions will be realized in hardware or software. There are different approaches for system level modeling. Varied languages are used for this goal: SpecC, SystemC, SystemVerilog, etc. In this article we will consider using SystemC/TLM languages and Virtual System Platform (Cadence) for SoC modeling in its HW/SW design for a new SpaceFibre SoC.

SystemC is a C++ extension language developed to model hardware and software blocks with an unified language. TLM is a set of SystemC modules (i.e. C++ classes). Each of them can provide one or more sockets through which the SystemC modules may read and write data. TLM has two parts – its functional model and its interface model. The functional model embodies the complete set of supported component specification functions. A specification function is a task that the device must perform. For example, a bus must perform arbitration and burst reads and writes. The interface model is composed of a set of functions declared in terms of operations on transactions that facilitate communication with the TLM. The behavior of each module is provided by a number of parallel threads. Threads are functions of the C++ class which communicate with the threads in other modules by passing data (i.e. reading or writing) through the sockets. This communication is known as a transaction and the data passed as a payload. The TLM supports reuse and increases simulation speed.

SystemC and TLM-2.0 can be used for developing Behavior models. These standards are widely used in the Virtual System Platform (VSP) of Cadence, [1]. The targeted use of a virtual platform is to refine the hardware/software architecture of the system in design and to enable early development of its embedded software.

The Cadence VSP is used to create, test, and debug individual TLM blocks, assemble the blocks into a virtual platform, and perform debugging and analysis to assure that system-level requirements are met. Cadence tools allow to create a virtual prototype that may consist of processors, memories, switches/routers, bus and peripherals. Developer can use his own IP model as well as Fast Models from the ARM or Imperas model catalogs. Cadence provides opportunities for developing proper IP models. After all components are determined, SoC designer should assemble the IP blocks into a subsystem or a virtual platform; then designer can debug the

* The Ministry of Education and Science of the Russian Federation under grant agreement no. RFMEFI57814X0022

virtual platform. Full SystemC debugging capabilities are available to debug construction or connectivity issues.

Modern SoC is very complex. Its performance depends on hardware implementation as well as on software. The opportunity to compile and to link embedded software into a program for running on the virtual platform are very important. Simulating and debugging the functionality of the virtual prototype and embedded software helps the SoC designer to find mistakes before realization. It is especially critical with using ASIC technology. Statistics information visualizes the behavior and performance of the current system version. Based on this analysis the system designer can modify and re-evaluate the virtual prototype until it behaves and performs as required. Capabilities of Cadence tools are very interesting for different SoC development projects.

II. SoC ARCHITECTURE

Architecture of the Network Memory Controller (NIC) with an embedded processor core and a switch could be considered as the case study of this methodology and tools, Figure 1. It consists of a RISC core, memory, external input and output ports. The ports operate in accordance with the SpaceFibre standard, [2], for onboard systems of spacecrafts, aircrafts and etc. NIC must provide data reading from remote devices and data writing to memory. Also it supports data reading from memory according to requests from remote units. Data exchange between network devices and NIC is performed by the Remote Memory Access protocol (RMAP) protocol.

The system level's model of NIC and the subsequent researches were carried out with the Cadence and Imperas software. The NIC model is implemented in Transaction Level Modeling 2.0 (TLM), [3], with its high-level approach to digital systems modeling, where details of communication between modules are separated from details of implementation of the functional units or of the communication architecture.

The TLM-2.0 standard supports several coding styles. They are loosely-timed (LT) and approximately-timed (AT). Firstly, LT style uses the blocking transport interface. For the blocking transport interface each transaction has just two timing points, marking the start and the end of the transaction. AT style uses the non-blocking transport interface. For the non-blocking transport interface each transaction has multiple timing points. A transaction is broken down into multiple phases, with an explicit timing point marking the transition between phases. Secondly, LT supports modeling of timers and interrupts, the direct memory interface (DMI). AT does not support DMI. The loosely-timed coding style is appropriate for use cases of software development using a virtual platform model of an MPSoC. The software content may include operating systems. The approximately-timed coding style is appropriate for the use cases of architectural exploration and performance analysis, [5]. The LT modeling time is less than the AT modeling time. We use LT coding style for our blocks of NIC.

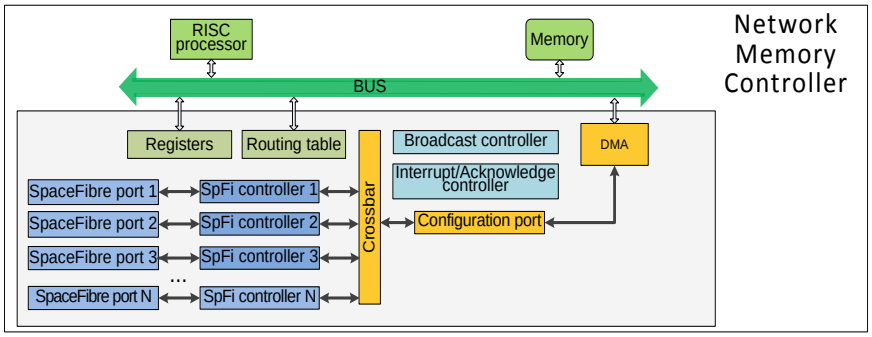

Figure 1. Structure of the Network Memory Controller

To simulate the processor core we used a processor model by the Open Virtual Platform (OVP) technology, [4]. OVP provides libraries of processor and units behavioral models (memory, buses, etc). They are designed to make it easy to assemble multi-core heterogeneous or homogeneous platforms with complex memory hierarchies and cache systems. SystemC, [5], and TLM 2.0 are used for building the SoC virtual platform. If it's necessary, SoC developers can independently make changes in a unit model. The OVP technology enables to run processor model at speeds of 100s of MIPS. OVP supports instruction accurate models, but not cycle-accurate ones. Blocks of SpaceFibre ports, configuration port, DMA, routing table, broadcast controller and interrupt/acknowledge controller are written on SystemC and TLM. The SpaceFibre port block, controllers can be used in other projects, which related with simulation of system supported data transmission according to SpaceFibre standard.

III. THE TLM INTERFACE GENERATOR

The TLM Interface Generator automatically generates hardware interfaces to assist in developing peripheral IP blocks and SoC design. The tool, tlmgen, is used to generate a set of TLM module templates for modeling hardware peripheral devices that contain memory mapped registers. Input data for tlmgen can be specified in the form of a text file, using a simple register definition language (simpleRDL), or in IP-XACT XML format. We choose the simpleRDL because port generation that is based on specification in IP-XACT is not supported. IP developer can describe memory-mapped registers together with field, register and register bank; with ports and parameter specification. The examples of input simpleRDL source are shown in Figure 2 and Figure 3. Figure 2 is the description of registers with their fields. Figure 3 is the register bank description with their addresses. The tlmgen utility simplifies the creation of custom catalog-ready TLM 2.0 blocks. Specifications rules are simple and clear.

```
REG DMA_AREA_SIZE_R4 {
    /* Register description */
    ACCESS = rw;
    REGWIDTH = 32;
    RESET = 0x0000;

    FIELD { ACCESS = rw; } DMA_AREA_SIZE_R4 [31:0];
}
REG SpF_PORT_MODE_VC_PARAMS {
    /* Register description */
    ACCESS = rw;
    REGWIDTH = 32;
    RESET = 0x0000;

    FIELD { ACCESS = rw; } VC_LNUM [7:0];
    FIELD { ACCESS = rw; } VC_THROUGHPUT [15:8];
    FIELD { ACCESS = rw; } VC_PRIORITY [18:16];
    FIELD { ACCESS = rw; } VC_WORK_EN [19:19];
    FIELD { ACCESS = rw; } CREDIT_OVERFLOW [20:20];
    FIELD { ACCESS = rw; } DATA_OVERFLOW [21:21];
}
```

Figure 2. Description of some registers of the "DMA" block

```
REGISTER_BANK dmaRegs {
    /*Register bank definition*/
    SIZE = 0x4dc;
    //Name of the busport generated
    BUSPORTNAME = cpu_access_target;
    //buswidth of the cpu_access_target
    BUSPORTWIDTH = 32;

    REG DMA_REGIONS_12_STATUS   0x10 ;
    REG DMA_REGIONS_34_STATUS   0x14 ;
    REG DMA_START_ADDR_R1   0x28 ;
    REG DMA_AREA_SIZE_R1    0x4c ;
    REG DMA_START_ADDR_R2   0x70 ;
    REG DMA_AREA_SIZE_R2    0x94 ;
    REG DMA_START_ADDR_R3   0xb8 ;
    REG DMA_AREA_SIZE_R3    0xdc ;
    REG DMA_START_ADDR_R4   0x100 ;
    REG DMA_AREA_SIZE_R4    0x124 ;
    REG SpF_PORT_MODE_VC_PARAMS 0x1e4 ;
    REG SpF_PORT_MODE_VC_TSLOTS_L   0x214 ;
    REG SpF_PORT_MODE_VC_TSLOTS_H   0x244 ;
    REG SpF_PORT_STATUS_VC1 0x294 ;
    REG INCORRECT_VC_INFO   0x414 ;
}
```

Figure 3. Description of a register bank of the "DMA" block

The resulting hardware model template represents the derived class in SystemC code with TLM ports, transport methods and the register access methods defined. IP designer should complete the proper IP model by adding functionality in the generated model template. For example, you can add specific behavior for the top-most class, I/O signals (like interrupt), more threads or methods to the model behavior, implement pre- and post-register access methods, and so on. We add threads to model behavior, supplement additional members of the class, change constructor. The scheme of one of our case models is shown in Figure 4. This tool decreases time of an IP block developing, helps to avoid common errors as naming inconsistencies, register overlapping, and illegal register accesses by software. However, TLM generator supports only one socket per device.

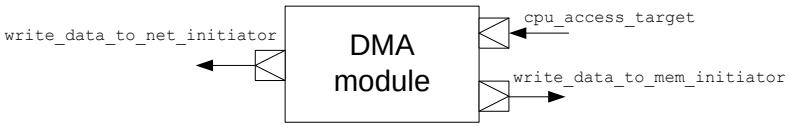

Figure 4. The scheme of the model "DMA" block

IV. ANALYZING PERFORMANCE OF A SOC

The Virtual System Platform provides analysis information at different levels. The first level is static. Static information about SystemC design summary can be produced during elaboration time: summary of each static objects sc_modules, tlm2_initiator_sockets, tlm2_target_sockets and so on. The second level is dynamic. It has simulation run-time information for finding performance bottlenecks and for tuning the design description for better simulation performance. Simulation information, such as simulator version, resource usage (CPU and Memory), is printed at the beginning. Information in the profile files shows a SystemC summary count of the SystemC execution, e.g., one line is the percentage and number of the process hits (Figure 5).

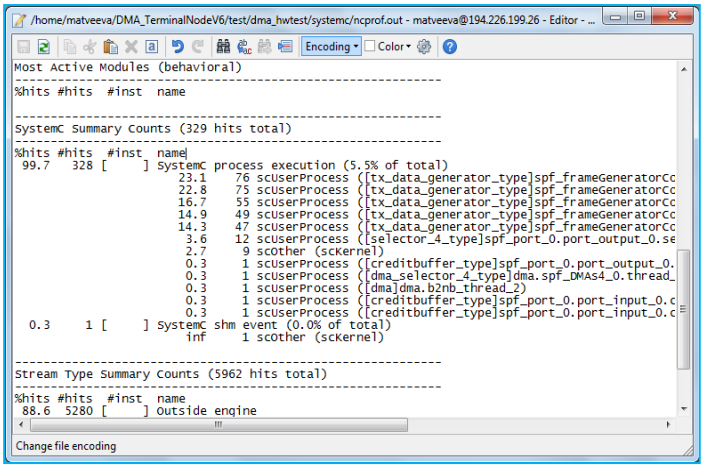

Figure 5. Run-time distribution between processes

User of the Virtual System Platform gets simulation run-time information about SystemC design. Developer can use different types of charts that helps analyze how much data was transmitted between sockets and data transmission distribution between different sockets in the system (Figure 6).

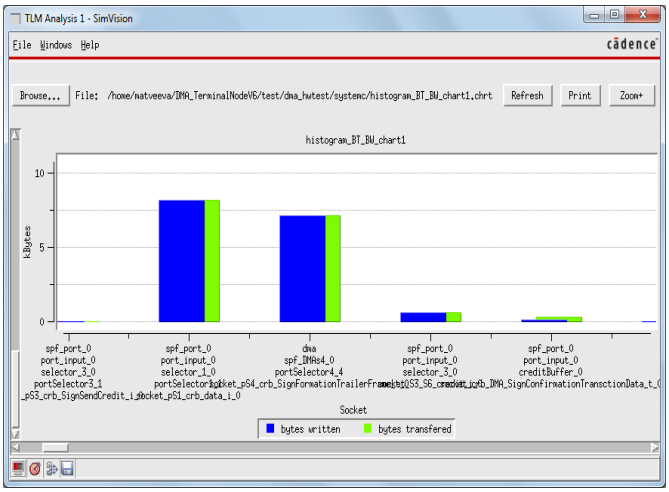

Figure 6. Plots with required parameters

V. PRELIMINARY SIMULATION RESULTS

We used VSP Cadence for performance testing of the implemented SpaceFibre ports which are the part of the NIC to test performance of SpaceFibre ports for streaming data, Figure 7.

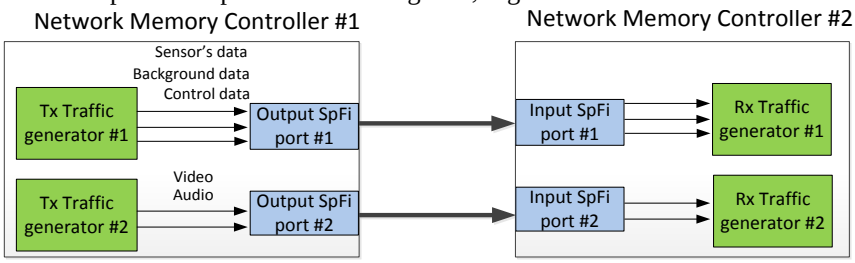

Figure 7. The general structure of the SpaceFibre ports performance testing

Traffic generators were used to simulate the issue of different streaming traffics: streaming video, audio, sensor data and etc., Table 1.

Table I. Parameters of streaming traffic

Traffic	Description	Packet size, bytes	Data rate, packet/ms	Accepted latency, ms

Video	Critical video information is a streaming interactive video; SVGA 800*600; 60Hz; color depth 24 bit [7]	1,42M	0,06	< 16,7 ms
Audio	Audio communications; Kodec G.711; 64Kbits/sec [8]	160	0,05	< 100 ms
Sensor data	Mission specific data from sensors (e.g. science equipment) [9]	1K	50	< 5 ms
Control data	Data from control modules to devices and equipment of spacecraft, very critical data [9]	260	0,02	< 0,1
Background	Background traffic	1K or 4K	25	Not defined

We used the following NIC parameters: number or ports – 2; SpaceFibre channel rate – 1.25 Gbit/sec; simulation time – 500 ms. The streaming traffics were distributed between ports #1 and #2. We used the Priorities Quality of Service of the SpaceFibre standard, because streaming traffics have different requirements for the latencies. The highest priority is 0, the lowest – 15. The details are presented below.

Table II. SpaceFibre ports configuration

Port	**Virtual Channel**	**Traffic**	**SpaceFibre parameters**	
			Priority	***Expected Bandwidth Percentage***
1	VC 1	Sensor data	1	0,032768
1	VC 2	Control data	0	0,0003328
1	VC 3	Background	3	0,16384
2	VC 4	Audio	2	0.0000512
2	VC 5	Video	1	0,55296

When all the Controller ports are enabled for data transmission, the traffics successfully transferred via two ports. All data was delivered with accepted latencies. The latency of video frames is about 11,4 ms. If the port #2 is disabled, all data went through the port #1. Sensor data were delivered with the unacceptable latencies (Figure 9). It happened all the time with a period of ~16,7 ms, because the channel was occupied by the video traffic. All other data was delivered with accepted latencies.

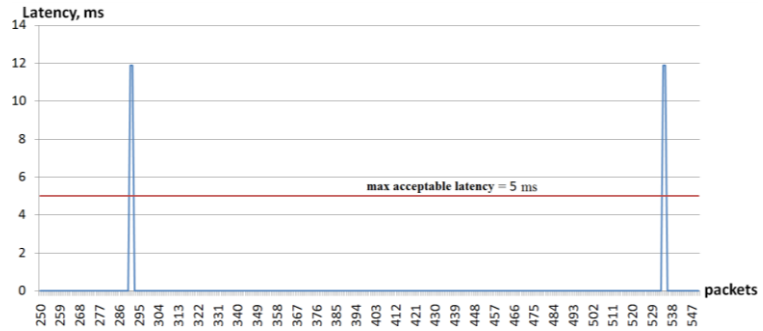

Figure 8. The latencies of the sensor data are unacceptable

It may be not acceptable for spacecraft systems and equipment. Hence, the network congestions were discovered that occurred because of two reasons:

1. High priority traffics blocked the link for others. This problem is called exclusive lock or link monopolization. It is defined by the following conditions:

- there are one or more virtual channels with the highest priorities;
- there is always data for sending in the output buffer of the high priority virtual channel;
- there is always the available free space for incoming data on the receipt side;
- the time of data reading from reception buffer and delivery of credit (FCT control word) is less than delivery time of the following data (SpaceFibre data frame);
- the Bandwidth Credit Limit value is not reached by the high priority virtual channel.

2. According to the rules of SpaceFibre Medium Access Control (MAC), [2], virtual channels shall

compete for sending data over the link. As a result sensor data will be interlaced by data from other virtual channels (e.g. control data, video frames).

To solve this problem in the SoC design we used the SpaceFibre Scheduled QoS in addition to the Priority QoS. The Scheduled QoS provides means of ensuring fully deterministic allocation of SpaceFibre network resources. Time is separated into series of time-slots, during which a virtual channel can be scheduled to send data. When a time-slot arrives in which a virtual channel is scheduled, it can send data based on its precedence. A virtual channel shall compete with other virtual channels for sending frames over the link based on the current precedence of the virtual channel and its schedule. During all the other time slots when the virtual channel is not scheduled to send data, it is not permitted to send any data even when no other virtual channel has data to send. If a virtual channel scheduled in the current time-slot does not have data to send during its time-slot or does not have space in its Virtual Channel Buffers at another end of the link, other virtual channels that are allowed to transmit in the current time-slot are permitted to use the otherwise wasted bandwidth, [2].

All simulated streaming traffics were assigned to particular slots. We used all available for scheduling time-slots - 256. Figure 9 shows an example of the slot distribution for video frames.

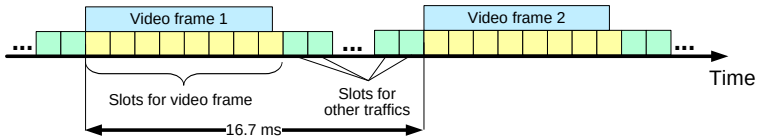

Figure 9. Example of the Scheduled QoS for streaming video

It is necessary to point out that the control traffic was assigned to all time-slots, because it is critical mission data (the highest priority) and it should be sent in any moment. The schedule was designed to deliver all traffics with acceptable latencies. The result is the Scheduled QoS provided determinism: streaming sensor data were delivered with acceptable latencies < 5 ms (figure 10).

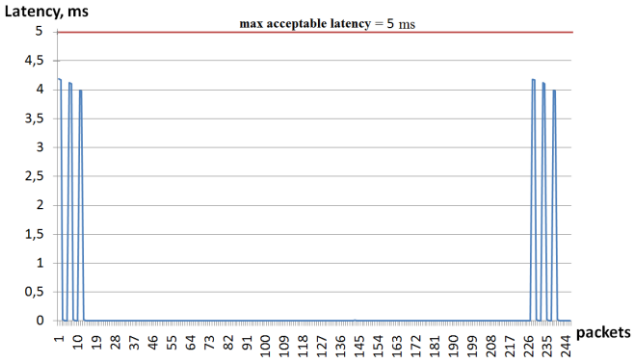

Figure 10. The Scheduled QoS provided the accepted frame latencies for sensor data

However, the Scheduled QoS has disadvantage – there is a problem of an idle virtual channel (Figure 11).

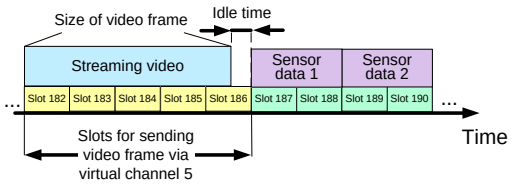

Figure 11. Example of idle virtual channel

The idle virtual channel occurred when the total duration of scheduled slots is less than the time of data sending (e.g. video frame). To solve this problem two traffics may be assigned to one time-slot (figure. 12) – the schedule compaction approach. In this case the traffics must have different priorities. It could be seen from the Figure 12 that the video stream (VC5) has the 14th priority which is higher than the 15th priority of sensor traffic (VC1). It is guaranteed that video frames will be transmitted entirely without undesirable delays.

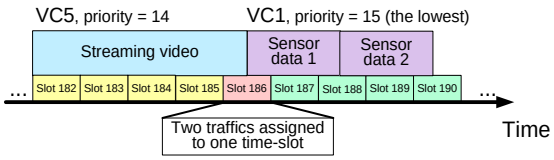

Figure 12. Two traffics are assigned to one time-slot

Otherwise, it is not guaranteed, because of video and sensor virtual channels will compete with each other for sending data over the link during the 186th time-slot (Figure. 13). As a result the sensor virtual channel may be enabled to transmit data and video frames are interlaced by sensor data. It caused unacceptable latencies of video frames.

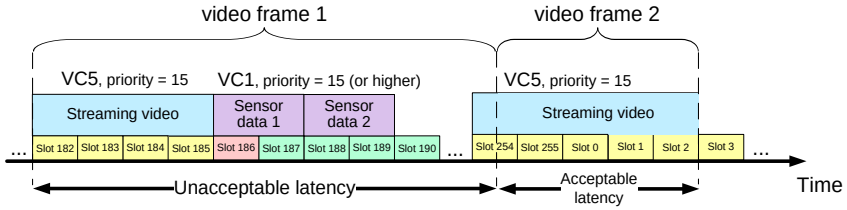

Figure 13. The Incorrect use of the Scheduled and Priorities QoS for video frames delivery

SpaceFibre limits the amount of priorities - 16. Consequently, there may be a situation when it is impossible to assign two traffics to one time-slot. For example, video stream traffic has the lowest priority –15. Sensor traffic has the same or higher priority. To guarantee the acceptable latencies of video frames the video stream and sensor traffics should be assigned to different time-slots as it is shown in figure 11. Thus the Scheduled and the Priority QoS cannot completely solve the problem of idle virtual channel.

One of solutions for removing these limitations is implementation of the adaptive QoS control (Adaptive Data Streaming Service, ADSS), [6], over SpaceFibre in the Network Memory Controller. For example, ADSS can dynamically change the virtual channel priorities (dynamic priorities approach). It will avoid the priorities limitations for the schedule compressing. Another possible approach is to try dynamic change of the time value of the Scheduled QoS (dynamic time-slots). This value defines the current number of time-slot. If ADSS changes time value and returns back it later, the limitation of available time-slots for scheduling will be removed. As a result the available amount of the scheduled time-slots will be increased. Hence, the problem of idle virtual channel could be solved. We used VSP Cadence to simulate the SpaceFibre Scheduled and Priority QoS (SPQ) and ADSS (dynamic priorities (DP) and dynamic time-slots (DT)) for sensor data transfer. The parameters of the streaming traffics were the same (table I). The dynamic time-slots allowed to increase the amount of time-slots from 256 up to 512. We measured the amount of transmitted data via sensor virtual channel. Then we compared these results with planned to transfer sensor data (figure 14).

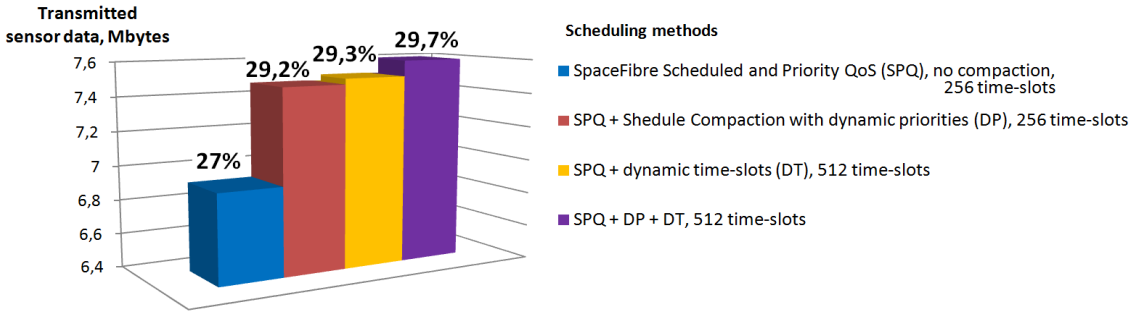

Figure 14. The comparison of transmitted and planned to data transfer (old parameters of sensor traffic) on different scheduling methods

It could be clearly seen from the figure 14 that increment of transmitted data between SPQ and ADSS approaches is about 2.7%. Much sensor data were not delivered, because the schedule was not designed for high speed sensor traffic. Sensor data were discarded when the buffer of sensor virtual channel was full. We decreased the packet rate up to 1.33 packet/ms and the packet size – 260 bytes. The results are demonstrated in figure 15.

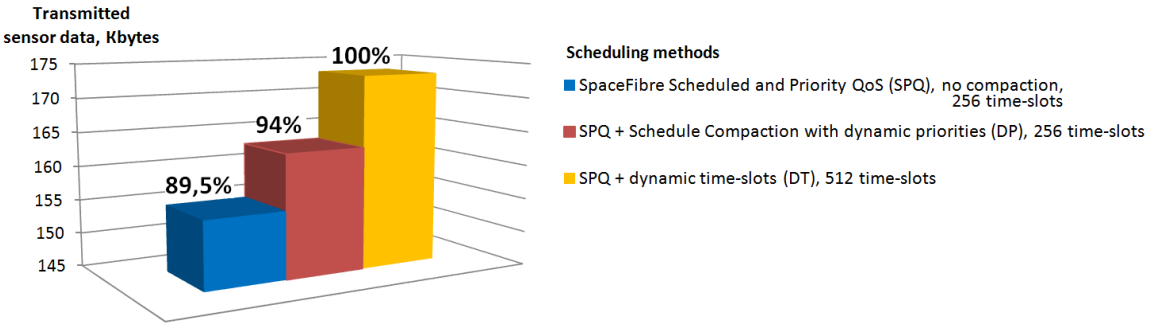

Figure 15. The comparison of transmitted and planned to data transfer (new parameters of sensor traffic) on different scheduling methods

All sensor data were delivered – 100%. There was no need to use DP and DT simultaneously. The ADDS allowed to increase the transmitted data up to 10.5%. So, the adaptive QoS control could be successfully used to solve the problem of idle virtual channel.

VI. Conclusion

The current paper presented of our experience in SoC design with SystemC, TLM 2.0 languages, OVP technology and Cadence VSP software. SystemC and TLM 2.0 are used for building the virtual platform of the SoC in design. OVP provides libraries of processor and units' behavioral models (memory, buses, etc) to make it easy to assemble multi-core heterogeneous or homogeneous platforms with complex memory hierarchies and cache systems. Cadence software provides huge capabilities for SoC design with a lot of tools for solving difficult tasks in IP development that allowed to perform simulation and to analyze performance of the Network Memory Controller project. Virtual System Platform allowed to develop and to debug our own IP blocks and SoC before the device release, to write and debug software for the embedded RISC-core.

With these methodologies and tools some important results of the SpaceFibre ports testing turned out:

- The conditions of the Exclusive lock of the SpaceFibre link were discovered and defined.
- There are limitations of the SpaceFibre network scalability. SpaceFibre standard includes limitations of the Scheduled QoS: the maximum number of slots is 256 slots and all slots are equal duration slots. SpaceFibre also limits amount of virtual channel priorities. As a result it limits the SpaceFibre network scalability.
- The problem of an idle virtual channel is discovered. Possible solutions were proposed. The schedule compaction with adaptive QoS control gave the increment of transmitted data from 2.2% to 10.5% depending on traffic parameters and the schedule.

All these results could be shared with SpaceFibre workgroup to improve the current specification.

Acknowledgment

The research leading to these results has received financial support from the Ministry of Education and Science of the Russian Federation under grant agreement no. RFMEFI57814X0022.

References

[1] Virtual System Platform User Guide, Cadence Design Systems, 2014.

[2] S. Parkes, A. Ferrer, A. Gonzalez, C. McClements, SpaceFibre Specification, Draft F3, September 2013.

[3] Open SystemC Initiative (OSCI), TLM-2.0 Language Reference Manual, 2009.

[4] Open Virtual Platforms (OVP). URL: http://ovpworld.org

[5] Open SystemC Initiative (OSCI), IEEE 1666™-2011 Standard for SystemC Language Reference Manual, 2011.

[6] I. Korobkov, "Adaptive Data Streaming Service for Onboard Spacecraft Networks", in Proceedings of 17th Conference of Open Innovations Association Finnish-Russian University Cooperation in Telecommunications (FRUCT17), P.G. Demidov Yaroslavl State University, Yaroslavl, Russia, 2015, pp. 291-298.

[7] ARINC Inc., ARINC Specification 818-2 (ARINC 818 Supplement 2), 2013. URL: http://www.arinc818.com/arinc-818-specification.html

[8] V. Olifer, N. Olifer, Computer networks: principles, technologies and protocols for network design. Saint-Petersburg: Piter, 2012.

[9] V. Grishin, P. Rastetter , P.Eremeev, A.Lobanov D 1.1 Consolidated set of Requirements for SpaceWire-RT. Version 2.0. 2012, 46 p.