A VERIFICATION SYSTEM OF RMAP PROTOCOL CONTROLLER *

V.Kh. Khanov, A.V. Shakhmatov, S.A. Chekmarev
Siberian State Aerospace University named after Academician M.F. Reshetnev
31 “Krasnoyarskiy Rabochiy” prospect, Krasnoyarsk, 660014, Russia.

E-mail: khvkh@mail.ru

Abstract. The functional verification problem of IP blocks of RMAP protocol controller is considered. The application of the verification method using fully-functional models of the processor and the internal bus of a system-on-chip is justified. Principles of construction of a verification system based on the given approach are proposed. The practical results of creating a system of verification of IP block of RMAP protocol controller is presented.

Functional verification is a mandatory procedure of the development of IP blocks which is very laborious. There are several approaches and methods of verification. In this paper, for the functional verification of IP blocks RMAP-controller uses a method that has the name of processor-driven verification (PVD) [1]. In a more general setting, this method can be classified as a method of transaction (Transaction-Based Verification, TBV) [2, 3], based on the use of full-scale models of the processor and bus. Originally TBV-method based on the use of high-level transactions entering the device under test (DUT) from the computer via the bus functional model (BFM). BFM provides conversion of transactions into test signals applied to respective inputs of DUT. This implementation of verification is less costly on time and computing resources than using full-scale models of the processor and bus used in PVD. However, the PVD method becomes necessary or high complexity of the project, for example, when the DUT is designed to enable a system on chip (SOC), based on the processor used, naturally, for verification; or in case of complexity of the verification, for example, when the DUT is a device designed to interact with the network. RMAP controller is just such a device: on the one hand it through the internal bus communicates with the SoC processor, on the other hand provides of SoC interaction with other devices SpaceWire network, which uses the RMAP. In this case PVD method provides good opportunities for verification.

RMAP-protocol [4] is a protocol for service and information packets transmission for configuration and data transmission of network devices in the network SpaceWire. SpaceWire is a network technology of integrating spacecraft on-board systems into a unified communication environment [5]. Hardware RMAP-protocol controller in the form of IP-block allows simplifying the interaction of SpaceWire network nodes, that is, spacecraft on-board systems on this protocol.

RMAP-controller (fig. 1) is designed as IP-block for inclusion into projects such as SoC through an intrasystem bus AMBA 2.0 [6]. Input and output signals are connected to LVDS driver of the physical layer of SpaceWire (SpW) network. A freely distributed open SpW Light IP-block [7] is used as a SpaceWire network codec. RMAP-controller is designed to process not only RMAP-packets, but also conventional SpW network packets with the help of

---

* Research was done supported by the Ministry of Education and Science of the Russian Federation in accordance with the agreement № 14.574.21.0041.
SpW-packets processing block. RMAP-packets processing block consists of a RMAP-target receiver and RMAP-initiator transmitter. Switching of data coming from the SpW Light codec or into SpW Light codec is made by means of a multiplexer unit. RMAP-controller operation is controlled through a block of control and status registers.

**Figure 1.** The structure of RMAP-controller.

RMAP-controller verification system includes a verification model that includes a controller, a processor and an internal bus; a debugger which realizes a verification model in FPGA hardware; and test software, which is the same for these two methods of verification.

A verification model for RMAP-controller testing which was created in ModelSim medium [1] is shown in Fig. 2. In this model RMAP-controller is DUT. The remaining blocks including test software represent the environment for testing which specifies input actions for RMAP-controller.

**Figure 2.** A verification model for RMAP-controller testing.

RMAP-controller is connected with SpWLight codec by SpW-connection implemented in the model. A soft-processor LEON3 [8] is used as a control processor. This processor was chosen because subsequently RMAP-controller is planned to be used in the system based on LEON3 and AMBA bus. The processor executes the test programs which are loaded into memory (AMBA RAM). A set of test programs transmits test packets in both the direction from SpWLight to RMAP-controller and back from RMAP-controller to SpWLight.

If a test program checks the direction from SpWLight to RMAP-controller, it generates and transmits a test RMAP-packet through SpWLight codec along SpW-connection. After
receiving the packet the RMAP-controller processes it and informs the test program about the end of processing. The test program reads the received information and performs its analysis. The analysis results are recorded in the log-file of the personal computer where modeling is done, or are output on the console of LEON3 processor.

Checking in the opposite direction is carried out similarly. The difference is that RMAP-controller forms RMAP-packet itself after taking corresponding instructions from the test program, and the test program produces the packet decoding and analysis after getting it from SpWLight.

Simulation is done as follows:
- the test program is downloaded into the RAM of LEON3 processor;
- the downloaded program is executed, forming the corresponding test signals and analyzing the results of test;
- debug information about the results is output in the log-file and the console during the test;
- the operation of the processor stops after the completion of the simulation.

The information which is output in the log-file and the console in the testing process includes input data for the test (for example, the contents of the fields of test packets), the intermediate and final results of the simulation, as well as the result of testing: whether the test is passed successfully or not. In addition, timing diagrams of all external and internal signals of RMAP-controller are available in ModelSim environment at the end of the simulation.

At the end of all test operations on the verification model the whole verification can be provided only by a subsequent verification of the RMAP-controller IP-block using FPGA-prototype (Fig. 3). It contains a verification model and uses test software previously developed for the verification model. Unlike the verification model the connection between SpWLight codec and RMAP-controller is physical, and the memory is external. There aren't any other changes.

![Figure 3. The architecture of a FPGA-prototype containing a verification model.](image)
Test software consists of two large groups of programs: static and dynamic tests. Static tests include the tests of all commands of RMAP-protocol (write, read, read-modification-write) with a fixed contents of all fields of a corresponding packet. For each static test each field was determined (formed or calculated) by a testing expert manually. On condition that the detailed structure of the input impacts is known this allows exploring the timing diagrams of the signals of the controller internal blocks. Thus, both the debug information which is output in the log-file and the signals timing diagrams corresponding to internal processes are available for the analysis of the controller operation when applied static tests.

Writing static tests is very tedious work. Some RMAP standard items required verification consisting of several static tests. Static tests conditionally include tests of control and status registers, as well as tests of exceptions of RMAP-controller blocks.

In dynamic tests the contents of the fields is generated automatically in accordance with the requirements of the standard, while the test involves sending a series of packets that simulate of a particular scenario of interactions of two nodes in the network with «point-to-point» connection. In this case, it is impossible to predict the data in the fields of transmitted and received packets, so the analysis of timing diagrams of the controller signals is impossible. Dynamic test scenarios are defined by three main opportunities of RMAP protocol, namely: recording a certain number of bytes of data in a remote node; reading a certain number of bytes of data from a remote node; reading, modifying, writing data from a remote node. Thus, the dynamic tests are at a higher level with respect to the static ones. In general, such a combination of static and dynamic tests allowed of a more complete coverage of DUT with verification tests. It should also be noted that it is impossible to do dynamic tests for a verification model because of the large amount of time spent on modeling.

Verification plan defines the verification first in ModelSim environment, and then with the help of FPGA-prototype. Verification in ModelSim was done on the basis of static tests with the analysis of timing diagrams of RMAP-controller work and debug information output to the console and log-file at the end of the test. Verification in FPGA-prototype was carried out on the basis of both static and dynamic tests with the analysis of debugging output, but without consideration of the timing diagrams. The priority in program tests running was defined by the logic set by the standard requirements of RMAP protocol.

The results of a verification system creation are estimated in terms of resources required to perform static tests. Testing of a verification model was carried out in ModelSim environment on a personal computer. The time of static tests performance ranged from 5 to 20 minutes. Some tests required several hours to be performed.

A physical prototype was developed on the basis of flash FPGA ActelA3PE3000. 56% of FPGA resources for this project were spent, which means that a project of such complexity is quite feasible, even using LEON3 processor which demands rather many FPGA resources. Time of execution of the most complex static tests, such as tests aimed at checking the timeouts waiting of packet response does not exceed 5-10 seconds. The results obtained can be considered to be quite satisfactory. They do not significantly hamper the process of verifying if tests were chosen preliminarily according to their complexity, that is, there shouldn't be attempts to perform tests a priori requiring much time in the model. FPGA prototype should be used at once in such situations.

Thus as a result of the research developed a verification system for IP-block of RMAP-protocol controller of SpaceWire network. The verification system includes a verification model intended for the verification of the controller in the form of a computer model; a physical prototype, which is intended for RMAP-controller verification under the conditions...
of hardware tests and implements a verification model hardwarely in FPGA; test software which provides functional testing in accordance with the requirements of the standard for RMAP-protocol. Experimental studies have shown that the construction of the verification system is realizable, provides satisfactory verification time and complete coverage with functional tests.

References
[1] Methodologies of Functional Verification, Mentor Graphics, available at: http://www.mentor.com/products/fv/methodologies/.
[2] Andreas Meyer, Principles of Functional Verification. Newnes, 2003 – 217p.
[3] Paul Wilcox, Professional Verification. A Guide to Advanced Functional Verification. Kluwer Academic Publishers, 2004 – 208p.
[4] ECSSE-ST-50-52C SpaceWire – Remote memory access protocol. – European Cooperation for Space Standardization (ECSS), 2010 – 109p.
[5] ECSSE-ST-50-12C SpaceWire – Links, nodes, routers and networks. – European Cooperation for Space Standardization (ECSS), 2008 – 129p.
[6] AMBA Open Specification, ARM, available at: http://www.arm.com/products/system-ip/amba/amba-open-specifications.php.
[7] Joris van Rantwijk, SpaceWire Light, v20110709, available at: http://opencores.org/project,spacewire_light.
[8] SPARC V8 32-bit Processor LEON3/ LEON3-FT Companion Core Data Sheet Template Design – Gaisler Research, 2010 – 41 p.