Construction of validation modules based on reference functional models in a standalone verification of communication subsystem

D.A. Lebedev <lebedev_d@mcst.ru>
I.A. Stotland <stotl_i@ mcst.ru >
MCST, 24 Vavilov st., Moscow, 119334, Russia

Abstract. The paper proposes some approaches to functional verification of microprocessor communication controllers based on developing layered UVM (Universal Verification Methodology) test systems. In modern microprocessor systems there are a lots of controllers operating with their own data types. Communication controllers support transferring and transformation data between microprocessor units. Such transformation must be carried out quickly and without data corruption for the correct functioning of the whole system. Communication controllers could carry additional functions such transmission values of copies of the system registers, address translation and others. Brief overview of verification tools and benefits of application standalone simulation based verification for checking the correctness of communication subsystems are marked out in the paper. We present the approaches of construction a standalone UVM-based verification environment with checking module implemented in external functional reference model. We also propose some techniques for checking the correctness of communication subsystems: checking multiple-clock controllers using parametrized clock generator, supporting of credit exchange mechanisms. Presented approaches were used to verify the communication subsystem — Host-Bridge — of Sparc V9 eight-core microprocessor developed by MCST. The difficulties discovered in the process of test system developing and its resolutions are described in the paper. The results of using presented solutions for verification of communicating subsystem controllers and further plan of the test system enhancement are considered.

Keywords: test system; communication controller; functional verification; Universal Verification Methodology (UVM); reference model.

DOI: 10.15514/ISPRAS-2018-30(3)-13

For citation: Lebedev D.A., Stotland I.A. Construction of validation modules based on reference functional models in a standalone verification of communication subsystem. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 3, 2018, pp. 183-194. DOI: 10.15514/ISPRAS-2018-30(3)-13

1. Introduction

The state of the art in microprocessors composition includes a variety of hardware controllers, which differ in complexity, speed rate, volume and types of data transmitted over them. The characteristics of data are continuously increasing. At the same time, verification costs are increasing, because the possibilities of verification methods are significantly lagging behind the development of microprocessor systems and, accordingly, the correctness checking requires more resources [1].

Each peripheral controller in the system could have its own data format. Converting data format is one of the functions of the interface communication controllers. The communication controllers can be a part of communication subsystem also known as northbridge. The northbridge typically handles communications among the CPU, I/O and in some cases RAM. Therefore, these transformations must be carried out quickly and without data loss. For this reason, the verification of communication controllers is an important step in the development of the microprocessor system.

The rest of the paper is organized as follows. Section 2 reviews the existing techniques for verifying communication controllers. Section 3 suggests an approach to the problem of developing test system. Section 4 describes a case study and the suggested approaches. Section 5 reveals results and Section 6 concludes the paper.

2. Functional verification of the communication controllers

It is necessary to simulate the operation of entire environment while providing standalone verification of a controller. This requires a test system, the development of which could be started at the earliest stages of whole microprocessor development, as soon as module specification and the RTL-model becomes available. Standalone verification allows detecting errors in the early stages of project. In addition, it helps to create complicated, critical and incorrect situations for the verified module. The achivement of such situations using system verification of the whole microprocessor model takes lot of resources. It is also important to note that the localization of the error is faster, what reduces the debugging time of the controller.

Due to its location between the CPU and the peripheral interface controller, the communication controller, in addition to its basic data format transformation function, could include copies of registers, buffers, FIFOs, parts of distributed control systems, and perform other additional functions. A number of these features should be taken into account in the standalone verification of communication controllers.

It is essential that, according to the classification proposed in [2], the properties of communication controllers include the absence of a pipeline, the absence of strict time (in the system clock frequency) restrictions on transaction processing and tagging of transmitted data. Accordingly, when the devices of this type are verified it is possible to use event-checking modules.

There are a number of methods to build a test system and implement a standalone verification of microprocessor controllers. Among them there is a tool created in the "MCST" named Alone-env, the development of the ISP RAS named C++TESKHW and methodology UVM [3]. The Alone-env tool simplifies implementation of standalone Verilog tests by creating test sequences in C++. Its library provides a wrapper-class over Verilog description of the verified module. Despite the relative simplicity of using Alone-env tool, there are some disadvantages: the lack of collecting coverage means, high requirements for the testing reference model and the inability to reuse the test system. One of the C++TESKHW tool features is availability of test generation based on the device state graph traversal. However, sometimes it is very hard to define all of the states of device and it needs high accuracy of documentation and checking reference model. UVM is the most widespread verification methodology developed by Accellera Systems [4]. UVM is a library with well-described tools for building portable and reusable testbenches and their components. The test system based on UVM can generate pseudo-random constrained input requests to cover all the possible states of the verified device. Most of well-known simulation tools (like Incisive, VCS, etc.) support the methodology. Moreover, most of VIP (Verification IP) support UVM-based interfaces. We also have a number of test systems and libraries already written and debugged. Therefore, we choose to use UVM for verification of RTL implemented modules of microprocessor systems.

Alone-env and C++TESKHW do not support UVM and we cannot use these tools for UVM-based test system development. UVM provides the universal approach for all types of devices to develop test systems. In this way, test systems are becoming more complex and worse in debugging. UVM also has no additional approaches for construction of validation modules based on reference functional models. Therefore, the main purpose of our investigation is to develop and extend the methods of standalone verification of communication controllers using UVM and program reference models.

3. Principles for testing communication controllers

Standalone verification of communication controllers can be carried out using simulation reference models that are part of the test systems - specially implemented software environment for the verified device. Test system functions includes:

- generating of input requests;
- monitoring of reactions from the verified device and the reference model;
- checking of reactions;
- forming a conclusion about the completeness of testing.

Uvm_sequence_item and uvm_sequence extension classes are defined to generate pseudorandom constrained impacts. The first one defines a set of variables that are required for serialization of set of impacts into a serial bit format. The second performs a single or multiply generating of a set of variables to transmit a request.

The request generated by the uvm_sequence_item object is processed by a special uvm_sequencer class and passed to the uvm_driver class. Uvm_driver produces a transformation of generated random requests into sequential bit-vectors in accordance with the interface exchange protocol. Uvm_monitor class is passive. It tracks changes in the interface of the verified device, indicating the appearance of input or output data, then packages the serial bit signals into the uvm_sequence_item format and transmits for further analysis to the checking blocks. To simplify the structure of the test environment perception, the uvm_driver, uvm_monitor, and uvm_sequencer are combined in the uvm_agent class, shown on fig.1.

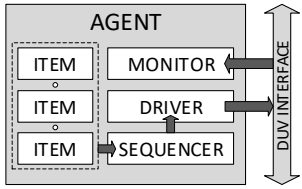

Fig 1. An example of the uvm_agent structure

Checking the reactions of the verified device can be carried out by internal means of the UVM library, however, if the verified device has a complex structure and many states, the checking module is based on the external to the controller environment reference model usually written in C++. A typical reference model-based test system is shown in Fig. 2.

In Fig. 2 DUV (Design Under Verification) is RTL-model of the verified device, ENV (Environment) - test environment. The number of agents are determined by the number of interface groups of the verified device (tracking the reactions uvm_monitor object can be taken outside from the agents). The reference model generates reference responses when impacts from the test environment are implied. Uvm_scoreboard is a checking module compares the response from the verification device and the reference model and makes a conclusion about the correctness of the operating. Using the DPI (Directed Programming Interface) of System Verilog is necessary to reconcile the types and classes of the test system written in SystemVerilog hardware description language with the C++ language. in which the reference model is developed.

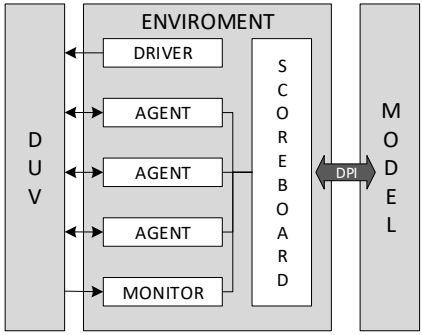

Fig 2. A typical structure of a test system for testing communications controllers.

The reference models could be divided into three types: cycle-accurate, discrete-event with time accounting and event models [5]. The choice of model type depends on the type of verified device, its architecture, and the complexity of developing a test environment. As stated earlier, the use of event models is justified for communication controllers, as they require less time to develop, maintain changes and can fully simulate the operation of such a controllers.

4. Functional verification of communication controller – Host-Bridge

Host-Bridge is a part of northbridge of microprocessor MCST-R2000 CPU with Sparc V9 architecture developing by MCST. The Host-bridge interface communication controller connects the system with external devices, accepting requests from the system and the I/O space while maintaining the transaction formats accepted in the system and I/O space. The Host-bridge receives requests from the System Commutator, communicate with two I/O channel controllers (IO-links) and provide translation of the virtual address to physical addresses. In addition, controller provides access to the system registers, registers of inter-processor links, memory controller's registers, transferring the new values of the registers to the local copies of them, transmitting interrupts, status signals and collecting snoop-responses. Each type of the registers has its own interface for communication with the destination device. All these features should be taken into account when verifying Host-bridge.

Some approaches for standalone verification using UVM and reference models were observed in [2, 6-8]. In [2] authors used buffers between testbench and reference model for checking marked transactions. In [6, 7] assertions and checking reference model were applied. In [8] there was reference model with very complicated algorithms. In our work, we used model buffers and assertions for checking correctness of transactions. In addition, we present a number of new solutions of the standalone verification process, which was applied for verification of Host-Bridge.

4.1 Several synchro signals parameters randomization

The main task of communication controllers is to coordinate the requests and data of several devices of the microprocessor system operating at different frequencies of the synchro signal. The parts of the communication controller in which several synchro signals interact should be checked carefully. Generating of random periods of synchro signals and their shifts that are relative to each other can be used for this purpose. The controller specification defines the operating ranges of each synchro signal. At the beginning of each test the frequency and start time of random synchro signal generators are pre-calculated. Thus, it is possible to detect errors in synchronization of internal units: the detection of metastability, desynchronization of releasing requests and data, sticking data in some positions of the buffers.

4.2 Support of credit exchange mechanisms

To control the flow of requests and free positions in the transaction buffers, Host-bridge supports a credit mechanism, which is a one-bit signal transmission that informs about the availability of free space in the buffers of connected devices. The management of this mechanism allows creating tests with full filling of all positions in the device buffers and needing to wait the vacated space to handle new requests, or on the other hand, tests, when the release of positions provides very fast, and the requests are executed almost instantly. As a result, it is possible to create test scenarios that are difficult to implement during system testing.

4.3 Verification of address translation controller

One of the components of the Host-Bridge is the address translation controller IOMMU. It translates a virtual address received from the I/O subsystem requests to physical addresses. The controller sends a request for information about the physical address to a special memory area for providing translation. Virtual address mapping to physical addresses is stored in a special controller buffer – IOTLB (Input-Output Translation Lookaside Buffer). If the buffer is full, the oldest element is displaced. The algorithm of the translation could be represented in the form of several consecutive steps:

1) receiving DMA request $p \leftarrow start(x)$,
2) analyzing of the input request then matching in the cache IOMMU with the following scenarios:
   a) match is found (hit IOMMU) – a request with the translated address x'' is executed;
   b) match is not found (miss IOMMU) – a request for a physical address x' is executed , then waiting for a response $p.receive(y)$ with the data, and only after that translation of the address is done.

Under dynamic test conditions, there may be situations when a lane in the IOMMU cache is not yet displaced in the RTL-model and the address can be translated without additional request, but it is not present in the reference model. In this case,

the reference model will give unnecessary requests to the test system. A global transaction counter was introduces in the reference model to solve this problem. The task of this counter was to identify the source of the requests. In addition, the responses generated by the test system are analyzed too. In the case when the request is successfully translated on the RTL-model side, and the reference model has already given an extra request for a physical address, the test environment generates a response that is marked with a special identifier and sent it to the reference model. When processing a response, the model concludes that the translation was not performed *p.model_check(y)*, calculates the desired transaction identifier *y.id* and sends it to a special buffer of canceled requests Q_{id} for the physical address. When checking the interchange buffers with reference model after the test completes, the identifier values in the buffer of canceled requests Q_{req} are compared with the identifiers of the remaining unprocessed requests for physical addresses from the reference model. Such remaining requests are not treated as erroneous and are deleted *delete(req.id)*. The pseudo-code of the algorithm is presented below.

DMA handling:

```
while true do
   wait p ← start(x)
   if x′ then
      p.ncheck(x′)
   else if x″ then
      begin
         wait p.receive(y)
         p.model_check(y)
         Q_id← y.id
         p.finish()
      end
end
```

After test checking:

```
for i ∈ Q_req do
   if c.check(req.id, Q_id) then
      delete(req.id)
   else report(req.id)
end
```

4.4 The correct organization of the exchange in terms of the uncertainty of the issuance of queries

In high-load dynamic tests with many input requests and responses, labeling requests and responses with tags that correspond to positions in the controller's buffers may differ from the values of tags in the reference model. This happens because of the inability to predict time of release of the buffer's position in the

event-driven models. Therefore, it is important to match the input and output requests of the model and the verified device. Each request, whether it is an I/O request or a PIO request, has several stages of execution. To ensure the correct functioning of the test scenarios we need to use an associative memory (mappers) for matching. The function of this memory is to store the matching of RTL request tags with reference model tags, when the remaining request data field, such as an address, destination device ID, processor number, and others are compared. Later, when you receive responses to the request, you have to pass to the model the same tag, which was allocated by the model at the stage of forming the request.

Communication controllers in multi-core systems can participate in the coherence protocol and accept snoop responses. Depending on the mode of operation and the content of the fields of the first received for the request response could come as several responses or only one. To complete the request check in coherent mode, it needed to pass all the responses with the correct tags to the model.

4.5 After test checking

The correct behavior of the communication controller is determined in providing a certain number of responses to requests and receiving the exact number of responses to them. Incorrect operation can be identified by counting the number of received requests, converted to another format requests, and accepted responses. For this purpose, the test system includes transaction counters. They capture all kinds of transactions while the test is running. At the end of the test, special algorithm checks values of these counters and make a conclusion about correctness.

After the test scenario is complete, it also needs to verify absence of unanswered requests in the buffers that link the reference model to the test environment. The presence of such requests signals about error of either the verified device or the reference model.

5. Results

The approaches described above were applied to standalone verification of the Host-Bridge of microprocessor MCST «R2000». Parametrization of synchro signals allowed finding metastability in the controller interfaces. Using different types of credit exchange rate helped to locate deadlocks and livelocks in the controller. Based on one test system the built-in IOMMU controller was also verified. Different configuration of answers with physical address were verified, which helps finding errors in displacement algorithms. After test, checking of model buffers and requests counters provide finding of not released responses to the system.

For the Host-Bridge controller, due to its functional and structural features that belong to the class of communication controllers, a test environment is developed with a checker based on reference event-model. Due to standalone verification of the device 67 errors that have not been found by other means of verification were found and corrected. Code and functional coverage was carried out and 94%

coverage was extracted. Gaps in coverage will be eliminated with the further expanding of the test environment with external parts of interrupt system. Total result indicates about effectiveness of standalone verification of communication controller.

6. Conclusion

Communication controllers are among the important parts of multi-core microprocessor systems have to be thoroughly tested. The principles described in the article do not depend mainly on the implementation of these controllers and allow their full standalone verification. The article suggests the ways to organize the interaction of the test system and the event reference model, as well as ways to resolve the difficulties encountered in the development of the test system.

The proposed approaches have been applied in the verification of the controller Host-bridge as a part of eight-core microprocessor, developed by "MCST". The developed test system and tests made it possible to detect and correct a number of logical errors that were not detected by other test methods.

In the future, it is planned to expand the test environment by adding a part of the interrupt system transmitting signals directly to the core. Fig. 3 shows a generalized diagram of a Host¬bridge, and the dotted lines illustrate such extension.

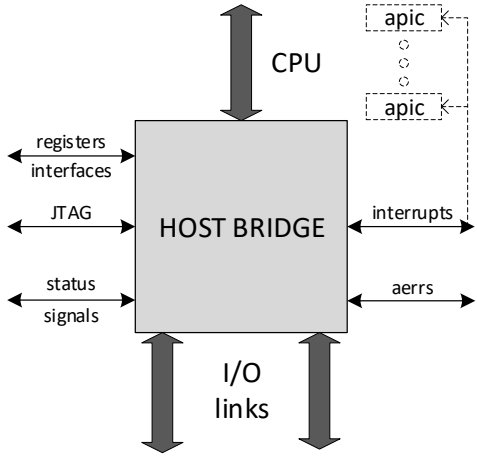

Fig 3. A typical structure of a Host Bridge controller connected with interrupt controllers

References

[1]. Kamkin A.M., Kotsynyak S.A, Smolov A.A.. Sortov A.D., Tatarnikov, Chupilko M.M. Tools for functional verification of microprocessors. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp. 149-200 (in Russian).

[2]. Shmelev V.A., Stotland I.A. Standalone verification of microprocessors using reference models with various levels of abstraction. Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem [Problems of development of promising micro- and nanoelectronic systems], no 1, 2012, pp. 435-440 (in Russian)..
[3]. A.N. Meshkov, M.P. Ryzhov, V.A. Shmelev. The developement of the verification tools of the Elbrus-2S microprocessor. Voprosy radioelektroniki [Issues of radio electronics], ser. EVT, 2014, no. 3, pp. 5-17 (in Russian).
[4]. Standard Universal Verification Methodology http://accellera.org/downloads/standards/uvm (12.06.2018).
[5]. Kelton W., Law A. Simulation modeling. Classics of CS. 3rd ed. SPb.: Piter, 2004.
[6]. Li-Bo Cheng, Francis Anghinolfi, Ke Wang, Hong-Bo Zhu, Wei-Guo Lu, Zhen-An Liu. A UVM Based Testbench Research for ABCStar. In Proc. of the IEEE-NPSS Real Time Conference (RT), 2016. https://indico.cern.ch/event/390748/contributions/1825090/attachments/1280814/1906413/CR_PosterSession2_268.pdf (12.06.2018).
[7]. Abhineet Bhojak, Tejbal Prasad. A UVM Based Methodology for Processor Verification. Proc. of the Design and Verification Conference and Exhibition (DVCON), 2015. https://dvcon-india.org/sites/dvcon-india.org/files/archive/2015/proceedings/6_UVM_Based_Processor_Verification_paper.pdf (12.06.2018).
[8]. Kamkin A., Petrochenkov M. A Model-Based Approach to Design Test Oracles for Memory Subsystems of Multicore Microprocessors. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 3, 2015, pp. 149-160. DOI: 10.15514/ISPRAS-2015-27(3)-11.

Построение модулей проверки на основе эталонных функциональных моделей при автономной верификации подсистемы связи

Дмитрий Лебедев <lebedev_d@mcst.ru>
Ирина Стотланд <stotl_i@ mcst.ru >
АО «МЦСТ», 119334, Москва, Россия, ул. Вавилова, д. 24

Аннотация. В статье предложены подходы к функциональной верификации контроллеров сопряжения интерфейсов в составе микропроцессоров на основе разработки многоуровневых тестовых-систем по методологии UVM. В современных микропроцессорных системах существует множество контроллеров, работающих с собственными типами данных. Контроллеры сопряжения интерфейсов учувствуют в передаче и преобразовании данных между блоками микропроцессора. Такое преобразование должно осуществляться быстро и без повреждения данных для корректного функционирования всей системы. Контроллеры сопряжения интерфейсов могут выполнять дополнительные функции, такие как передача значений копий системных регистров, преобразование адресов и другие. В статье дан краткий обзор средств верификации и преимуществ применения автономной имитационной верификации для проверки корректности контроллеров сопряжения интерфейсов в составе подсистем связи. Представлены подходы к построению автономной

верификационной тестовой системы на основе методологии UVM с модулем проверки, реализованным во внешней функциональной эталонной модели. Так же предложены методы проверки корректности подсистем связи: проверка контроллеров, работающих с несколькими синхросигналами при помощи параметризованного генератора синхросигналов, поддержка механизмов обмена кредитами. Представленные подходы использованы для верификации подсистемы связи - Host-Bridge - восьмиядерного микропроцессора с архитектурой Sparc V9, разработанного компанией АО «МЦСТ». В статье описаны проблемы, обнаруженные в процессе разработки тестовой системы и способы их разрешения. Представлены результаты использования рассмотренных решений для верификации контроллеров подсистем связи и дальнейший план совершенствования тестовой системы.

Ключевые слова: тестовая система; контроллер сопряжения интерфейсов; функциональная верификация; Universal Verification Methodology (UVM); эталонная модель.

DOI: 10.15514/ISPRAS-2018-30(3)-13

Для цитирования: Лебедев Д.А., Стотланд И.А. Построение модулей проверки на основе эталонных функциональных моделей при автономной верификации подсистемы связи. Труды ИСП РАН, том 30, вып. 3, 2018 г., стр. 183-194 (на английском языке). DOI: 10.15514/ISPRAS-2018-30(3)-13

Список литературы

[1]. А.С. Камкин, А.М. Коцыняк, С.А. Смолов, А.А. Сортов, А.Д. Татарников, М.М. Чупилко. Средства функциональной верификации микропроцессоров. Труды ИСП РАН, том 26, вып. 1, 2014, стр. 149-199.

[2]. Шмелев В.А., Стотланд И.А. Автономная верификация микропроцессоров на основе эталонных моделей разного уровня абстракции. Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), No. 1, 2012, стр. 435-440.

[3]. А.Н. Мешков, М.П. Рыжов, В.А. Шмелев. Развитие средств верификации микропроцессора «Эльбрус-2S». Вопросы радиоэлектроники, сер. ЭВТ, вып.3, 2014, стр. С. 5-17.

[4]. Standard Universal Verification Methodology http://accellera.org/downloads/standards/uvm (12.06.2018).

[5]. Кельтон В., Лоу А. Имитационное моделирование. Классика CS. 3-е изд. СПб.: Питер, 2004.

[6]. Li-Bo Cheng, Francis Anghinolfi, Ke Wang, Hong-Bo Zhu, Wei-Guo Lu, Zhen-An Liu. A UVM Based Testbench Research for ABCStar. In Proc. of the IEEE-NPSS Real Time Conference (RT), 2016. https://indico.cern.ch/event/390748/contributions/1825090/attachments/1280814/1906413/CR_PosterSession2_268.pdf (12.06.2018).

[7]. Abhineet Bhojak, Tejbal Prasad. A UVM Based Methodology for Processor Verification. Proc. of the Design and Verification Conference and Exhibition (DVCON), 2015. https://dvcon-india.org/sites/dvcon-

india.org/files/archive/2015/proceedings/6_UVM_Based_Processor_Verification_paper.pdf (12.06.2018).

[8]. Kamkin A., Petrochenkov M. A Model-Based Approach to Design Test Oracles for Memory Subsystems of Multicore Microprocessors. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 3, 2015, pp. 149-160. DOI: 10.15514/ISPRAS-2015-27(3)-11.