DOI: 10.15514/ISPRAS-2019-31(3)-7

Standalone verification of IOMMU with virtualization supporting

[1]A.A. Petrykin, ORCID: 0000-0001-5779-1980 <petrykin_a@mcst.ru>
[1]I.A. Stotland, ORCID: 0000-0002-4359-3059 <stotl_i@mcst.ru>
[1,2]A.N. Meshkov, ORCID: 0000-0002-8117-7398 <alex@mcst.ru>
[1] INEUM, 24, Vavilova st., Moscow, 119334, Russia
[2] MCST, 1, Nagatinskaya st., Moscow, 117105, Russia

Abstract. This article presents an approach to standalone verification of I/O Memory Management Unit with virtualization supporting. We presented the base architecture of the test system. The main problems encountered during the verification of IOMMU with virtualization support are considered. One of the key problems was the formation of translation table pages. The number of translation tables depends on the mode of IOMMU operation and the type of translation. As a solution of this problem the approach to the dynamic generation of translation tables is proposed. The algorithm for formation of translation table pages in the generator is presented. The problem of validating the translation of a virtual address into a physical one using two-level translation tables is solved. The features of the reference model implementation are considered. Reference model and test system which have been used for IOMMU verification of microprocessor with the 6th generation «Elbrus» architecture are described. The main components of the test system and the methods of communication between test system and IOMMU model are presented. The results of IOMMU verification are considered.

Keywords: I/O Memory Management Unit; test system; reference model; Elbrus

For citation: Petrykin A.A., Stotland I.A., Meshkov A.N. Standalone verification of IOMMU with virtualization supporting. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 3, 2019. pp. 77-84. DOI: 10.15514/ISPRAS-2019-31(3)-7

Автономная верификация IOMMU с поддержкой виртуализации

[1] А.А. Петрыкин, ORCID: 0000-0001-5779-1980 <petrykin_a@mcst.ru>
[1] И.А. Стотланд, ORCID: 0000-0002-4359-3059 <stotl_i@mcst.ru>
[1,2] А.Н. Мешков, ORCID: 0000-0002-8117-7398 <alex@mcst.ru>
[1] ПАО «ИНЭУМ им. И.С.Брука», 119334, Россия, г. Москва, ул. Вавилова, д. 24
[2] АО «МЦСТ», 117105, Россия, г. Москва, ул. Нагатинская, д. 1, стр.23

Аннотация. В данной статье представлен подход к автономной верификации блока управления памятью ввода / вывода с поддержкой виртуализации. Мы представляем базовую архитектуру тестовой системы. Рассматриваются основные проблемы, возникающие при верификации IOMMU с поддержкой виртуализации. Одной из ключевых проблем стало формирование страниц таблицы трансляции. Количество таблиц трансляции зависит от режима работы IOMMU и типа трансляции. В качестве решения этой проблемы предложен подход к динамической генерации таблиц трансляции. Представлен алгоритм формирования страниц таблиц трансляции в генераторе. Решается проблема проверки трансляции виртуального адреса в физический с использованием двухуровневых таблиц трансляций. Рассмотрены особенности реализации эталонной модели. Описаны эталонная модель и тестовая система, которые использовались для верификации микропроцессора IOMMU с архитектурой 6-го поколения «Эльбрус». Представлены методы связи между тестовой системой и моделью IOMMU. Рассматриваются результаты проверки IOMMU.

Ключевые слова: блок управления памятью ввода/вывода; тестовая система; эталонная модель; «Эльбрус»

Для цитирования: Петрыкин А.А., Стотланд И.А., А.Н. Мешков. Автономная верификация IOMMU с поддержкой виртуализации. Труды ИСП РАН, том 31, вып. 3, 2019 г., стр. 77-84 (на английском языке). DOI: 10.15514/ISPRAS-2019-31(3)-7

1. Introduction

An I/O Memory Management Unit (IOMMU) is a hardware device that translates virtual address received from the I/O subsystem requests to proper machine physical address. IOMMUs have long been used for prohibiting devices from DMA'ing into the wrong memory and for performance optimization. With the hardware support of operating system virtualization IOMMU is also used for extending the protection and isolation properties of VMs (Virtual Machines) for I/O operations, supporting isolation of interrupts from devices and external interrupt controllers and recording of DMA and interrupt errors to system software that may corrupt memory or impact VMs isolation [1][2]. Therefore, modern IOMMUs are quite complex devices that have many modes of operation and their verification is an important step in the development of the microprocessor system.

In the paper, we present a case study for functional verification of IOMMU with virtualization supporting of microprocessor with the 6th generation «Elbrus» architecture developed by MCST. The paper addresses the problem and methods of standalone verification of IOMMU with virtualization supporting.

The rest of the paper is organized as follows. Section 2 considers the problems arising from the verification of IOMMU. Section 3 suggests an approach to the problem of developing page table lines generator. Section 4 presents a common approach to the design a test system and describes its components. Section 5 reveals results. Section 6 concludes the paper.

2. IOMMU verification challenges

For hardware support of operating system virtualization, the translation of a virtual address into a physical one occurs according to a scheme that includes a two-level page structure. At the first level, the virtual guest OS address (GVA) is translated into the physical guest OS address (GPA) using its translation tables. At the second level, the resulting address is translated to the physical address (PA) of the hypervisor using the hypervisor translation tables. Information about the broadcast address for each device is stored in the device table. The table consists of Device Table Entry (DTE) elements. Each DTE contains information about the Domain ID (DID), host page table root pointer (HPTP) and guest page table root pointer(GPTP). In the IOMMU of microprocessor with the 6th generation «Elbrus» architecture, DID field has an extension and is called EDID. In addition to the domain number, it contains information about whether the device belongs to the guest or the hypervisor.

The pages number of translation tables depends on the mode of device operation and the size of the page themselves. Processors with the 6th generation «Elbrus» architecture support three page sizes: 4KB, 2MB and 1GB. For guest virtual address translation through 4KB pages, the number of memory hits can reach 25. Each memory hit takes a long time to process. Therefore, as part of IOMMU can be used a lot of different caches [1].

It follows from the above that the main goal of IOMMU verification is to check the correctness of all translation modes and check the following:

- translation on pages with various size;
- error handling;
- caches correctness;
- translations for the greatest possible number of addresses;
- absence of suspensions;

- absence of unknown logic value (X-state) on output signals.

There are several main problems encountered during the IOMMU verification of processors with the 6th generation «Elbrus» architecture:

- the large number of translation tables and different page size;
- large size of the entire address space;
- different virtual addresses can be translated into one physical.

To solve these problems, it was necessary to create the translation page tables. But their formation for the entire address space would require a large amount of computing resources. The paper [3] presents the approach based on constraint-random generation page table entries (PTEs), which we used for IOMMU verification as a part of northbridge of our previous microprocessor. However, the availability of hypervisor and guest translation caches as well as a large number of translation pages required for guest virtual address translation does not allow to use the approach described in [3]. The traditional approach is to generate a static table for a limited set of addresses which is used for verification Translation Lookaside Buffer (TLB) of MMU[4]-[6]. But using this approach, it is difficult to verify error handling and virtual address translation over various size pages. Therefore, for functional verification of the IOMMU of microprocessor with the 6th generation «Elbrus» architecture, it was decided to develop a dynamic generator of translation table pages, which generates rows of translation tables for any virtual address.

3. Generator of translation table pages

The formation of pages in the generator depends on the translation mode, which is specified directly in the tests and translation request. The algorithm of the generator work could be represented in the form of several consecutive steps:

1) receiving request for translation;
2) request translation and mode analysis;
3) DTE formation;
4) translation pages entry.

Translation modes are divided into single-level or native translations and two-level translations. In turn, native translations can take place in the same domain or split into different domains. Native translations in the same domain doesn't require DTE. For the rest of translation modes DTE formation is necessary. Since a unique translation identifier in Elbrus-12c processors is an EDID, each DTE element is stored in an associative array indexed by it. The formation of a DTE begins with the selection of a random EDID for a given device, after which the presence of a DTE for a given EDID in the array is checked and if it is missing, the remaining fields are formed.

Translation tables have a 4-level structure and consist of 512 elements. Each line of the table contains information about the access rights to it and whether it is the last in the translation structure. In addition, there is a field named PPN (Physical Page Number), that indicates the line from the next translation level, in case the line is on the last translation level, it contains the physical address. Hereinafter the pages of hypervisor and guest translations will be called HP (Host Page) and GP (Guest Page), respectively.

For native translation, the address of the first line is formed from the host page table index (hptp) contained in the DTE, and a part of the translated virtual address. Physical page number for each page level is calculated as follows:

$$ppn(n) = hptp + 512 * (5 - n) + VA(n), \quad (1)$$

where n – host page level, $VA(n)$ – the part of translated virtual address on level n , $hptp$ – table root pointer.

The full list of pages for native translation is presented in Table 1.

Table. 1. List of pages for native translation

Page level	Page address	Physical Page Number
HP_L4	hptp, VA(4)	hptp + 512 + VA(4)
HP_L3	ppn_L4, VA(3)	hptp + 512 * 2 + VA(3)
HP_L2	ppn_L3, VA(2)	hptp + 512 * 3 + VA(2)
HP_L1	ppn_L2, VA(1)	hptp + 512 * 4 + VA(1)

When generating rows of a two-level table for guest virtual address translation, the generator first calculates the guest physical address (GPA) for first guest page (GP_L4) according to the formula:

$$GPA(4) = GPTP + VA(4) \quad (2)$$

After that, the pages necessary for the translation of this GPA to HPA are written in the same way as the recording of the pages of translation VA to PA in the native mode. The list of pages for that translation may be seen at Table 2.

Table. 2. List of pages for guest physical addres translation into host physycal address

Page level	Page address	Physical Page Number
HP_L4	hptp, GPA(4)	hptp + 512 + GPA(4)
HP_L3	ppn_L4, GPA(3)	hptp + 512 * 2 + GPA(3)
HP_L2	ppn_L3, GPA(2)	hptp + 512 * 3 + GPA(2)
HP_L1	ppn_L2, GPA(1)	hptp + 512 * 4 + GPA(1)

Then the GP_L4 page itself is written and after that the guest physical address of the next page level (GPA_3) is calculated as the sum of HP_L1 page ppn and part of virtual address:

$$GPA_3 = ppn + VA(3) \quad (3)$$

And so on to get the GPA of the original GVA. For the obtained GPA, the pages of the last hypervisor translation are formed, which give the desired HPA. In order to avoid writing identical lines at different translation levels, guest pages addresses and ppns are configured as followed:

$$addr(x) = \{ppn, GPA_{x(0)}\}, \quad (4)$$

$$ppn(x) = hptp + 512 * (9 - x) + GPA_{x(1)}, \quad (5)$$

where x – guest page level, $GPA\_x(0)$ – the part of the guest physical address that is not translated by hypervisor structures.

The list of guest pages for two-level translation is presented in the Table 3.

Table. 3. List of guest pages in two-level translation

Page level	Page address	Physical Page Number
GP_L4	ppn, GPA_4(0)	hptp + 512 * 5 + GPA_4(1)
GP_L3	ppn_L4, GPA_3(0)	hptp + 512 * 6 + GPA_3(1)
GP_L2	ppn_L3, GPA_2(0)	hptp + 512 * 7 + GPA_2(2)
GP_L1	ppn_L2, GPA_1(0)	hptp + 512 * 8 + GPA_1(1)

To verify the error handling on each level of table can be prescribed a row of the page table that causing an error. That table level can be set via test parameters or randomly. In the same way, translation page sizes can be set through the PS field in a line of translation table. To verify the error handling on each level of table can be prescribed a row of the page table that causing an error. That table level can be set via test parameters or randomly. In the same way, can be setted the translation page sizes through the PS field in a line of translation table.

4. Test System Structure

Test system was implemented with using of SystemVerilog language [7] and Universal Verification Methodology [8]. Use of this language allows for an easy interface with Verilog and SystemVerilog devices, and UVM describes a general test system structure and provides a library of basic verification components.

The test system includes a set of basic components, which are presented below.

4.1 Apb (Advanced Peripheral Bus) agent

Apb agent is used to entrance the set of configuration registers in IOMMU whose access interface is implemented according to the APB protocol.

4.2 Register model

A register model is an entity that encompasses and describes the hierarchical structure of class object for each register and its individual fields. Every register in the model corresponds to and actual hardware register in the design.

4.3 Translation agent

This is the agent, in which the translations are generated and then sent on the DUT (Design Under Test) query interface and generator of table pages. Translation generation is based on constrained randomization. To specify some test scenario, one must define specific constraints for transactions that will be issued. SystemVerilog offers a native support for constrained randomization constructs. Translation agent is also receives the results of the translation from the response interface.

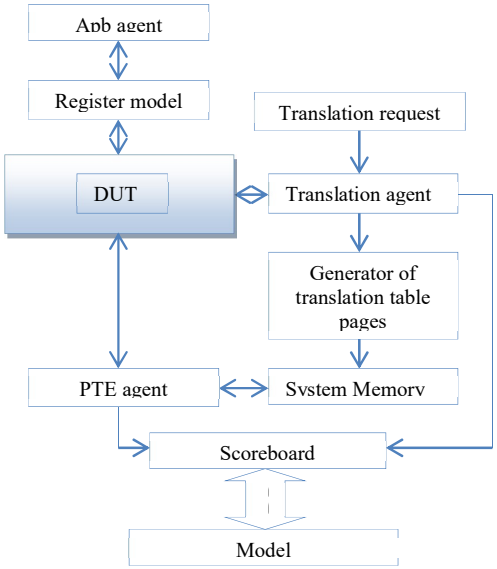

Fig. 1. Structure of a test system

4.4 Generator of translation table pages and system memory.

Each request received from the translation agent is processed by the generator as described in Section 3. All lines of translation pages are stored in system memory.

4.5 Page table entries (PTE) agent

PTE agent collects information about requested and received by the device PTEs from system memory. If for any reason the requested PTE is missing, the system memory will randomly generate it.

4.6 Model

The IOMMU reactions were tested using its reference event model. For reconciling the types and classes of the test system written in SystemVerilog with the C++ language in which the reference model is developed the DPI (Directed Programming Interface) was used. The reference model accepts input stimuli and generates output responses, which are then sent to scoreboard.

4.7 Scoreboard

Scoreboard receives transactions from translation and page table entries interfaces. After that, they are compared with the corresponding transactions received from the model. If a mismatch is detected, the module reports an error in the test system. Test system structure is presented in Fig.1.

5. Results

The approaches described above were applied to standalone verification of the IOMMU of microprocessor with the 6th generation «Elbrus» architecture. Due to standalone verification of the device, 58 errors in the RTL description that have not been found by other means of verification were found and corrected. Total result indicates about effectiveness of standalone verification of I/O memory management unit.

6. Conclusion

I/O memory management units are among the important parts of modern microprocessor systems have to be thoroughly tested. In this article, we presented a method of translation table pages forming. The main advantages of the described approach are:

- no need to create tables for the entire address space;
- the ability to dynamically set the page size;
- convenience of exception checking due to dynamic generation of page table row fields;
- ease of obtaining maximum coverage, due to the possibility of calls to any address.

The principles described in the paper do not depend mainly on the IOMMUs implementation and allow their full standalone verification.

The proposed approaches have been applied in the verification of IOMMU as a part of microprocessor with the 6th generation «Elbrus» architecture, 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.

References

[1] Intel Virtualization Technology for Directed I/O Architecture Specification. Intel, 2018.
[2] AMD I/O Virtualization Technology (IOMMU) Specification. AMD, 2016.
[3] 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-2016-28(3)-10.
[4] Alkassar E., Cohen E., Kovalev M., Paul W.J. (2012) Verification of TLB Virtualization Implemented in C. Lecture Notes in Computer Science, vol 7152, pp 209-224.
[5] Alkassar, E., Cohen, E., Hillebrand, M., Kovalev, M., Paul, W. Verifying shadow page table algorithms. In Formal Methods in Computer Aided Design (FMCAD), 2010. pp. 267-270.

[6] Kamkin A., Kotsynyak A. Specification-Based Test Program Generation for MIPS64 Memory Management Units. Trudy ISP RAN/Proc, ISP RAS vol. 28, issue 4, 2016. pp. 99-114. DOI: 10.15514/ISPRAS-2016-28(4)-6.
[7] IEEE Standard for SystemVerilog — Unified Hardware Design, Specification, and Verification Language. IEEE Std 1800-2012.
[8] 1800.2-2017 - IEEE Standard for Universal Verification Methodology Language Reference Manual.

Информация об авторах / Information about authors

Антон Алексеевич ПЕТРЫКИН получил степень магистра в 2019 г. в Российском университете дружбы народов. В настоящее время работает инженером-программистом в АО «МЦСТ». Научные интересы включают верификацию компьютерных систем и подсистем микропроцессоров.

Anton Alekseevich PETRYKIN received a M.Sc. degree in 2019 in the Peoples' Friendship University of Russia. He is currently working as a software engineer at AO «MCST». Research interests include verification of computer systems and microprocessor subsystems.

Ирина Аркадьевна СТОТЛАНД в настоящее время работает начальником сектора автономной верификации микропроцессоров в АО «МЦСТ». Она получила степень кандидата технических наук в 2012 году. Область ее научных интересов включает автономную верификацию подсистем микропроцессоров и систем на кристалле.

Irina Arkadievna STOTLAND is currently working as microprocessor verification team lead at MCST. She received a PhD degree in 2012. Her research interests include standalone verification of microprocessor and SoC subsystems.

Алексей Николаевич МЕШКОВ получил степень кандидата технических наук в ИНЭУМ им. И.С. Брука в 2013 года. В настоящее время является начальником отдела в АО «МЦСТ». Область научных интересов включает программное моделирование и верификацию компьютерных систем.

Alexey Nikolaevitch MESHKOV recieved a PhD degree at INEUM in 2013. He is currently working as a chief of department at MCST. His research interests include software modeling and verification of computer systems.