UDK 004.274
DOI: 10.15587/2312-8372.2015.51197

Захарченко Т. Л.

ІНТЕГРАЦІЯ АДАПТИВНОГО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

У статті описано процес інтеграції апаратного забезпечення, розробленого з застосуванням адаптивного підходу, в обчислювальні системи. Процес інтеграції описаний на усіх рівнях абстракції: від його моделі, описаній HDL-мовою, до драйверу операційної системи. Це дозволяє спростити застосування такого апаратного забезпечення кінцевими користувачами, та покращити застосовність адаптивного підходу для реальних задач.

Ключові слова: *адаптивне апаратне забезпечення, обчислювальні системи, інтеграція з ОС, прискорення обчислень.*

1. Introduction

Rapid computing systems development in recent years imposes challenges in the field of computing systems design. They are becoming more and more complex, but approaches to the design remain ineffective. Firstly, one cannot determine whether a design is correct or not. Here, the design is correct when created with means and tools, which suit both the designer and pragmatics of the design. Such conditions must guarantee correctness of the design in proposed sense, it would be the best possible achieved. Moreover, one cannot determine whether the produced solution is «bug-free». An algebraic approach in the development process allows applying mathematical methods to design process in order to ensure from every start of it, that it will be bug free.

An attempt to address these problems was made in the article on pragmatics dependent hardware design [1]. It describes how to design hardware computing solutions with an adaptive approach. In the kernel of the method is sequential creation and application of compositions. Here, composition is mathematical term, which means operation on functions [2]. Hardware designs, produced with such design method are called «adaptive hardware».

However, pragmatics dependent hardware design imposes new challenges to address. First, adaptive hardware, designed with the new approach must be integrated into already existing computing systems and some solutions must be proposed. When this problem will be solved, a full design process support and automation from the process of semantic solution creation [1] directly to hardware and software solution will be possible. Statements made above prove necessity and importance of the research in this area.

2. Sources analysis, problem statement

Hardware domain of this research narrowed to FPGAs, which allow reconfiguration of designs they enclose in runtime, increasing in this way the practical value of such approach. FPGAs are especially effective in interaction with hardware processor systems. Both major producers of FPGA chips, Altera and Xilinx are proposing such SoC solutions: Cyclone V, Arria 10, Stratix 10, UltraScale MPSoC [3–6]. Even Intel announced new Xeon CPU with integrated FPGA [7]. Verilog is chosen as hardware solution description language.

The paper is to propose integration method for hardware designed with adaptive approach (adaptive hardware) in computing systems. In such approach hardware considered as a function (in mathematical sense) with arguments and value. Nevertheless, low-level programming languages and HDL's do not operate with concept of «function» at all. The goal is to provide the developer with an easy-to-use framework for application of such hardware.

The article will consist of following the sections: Verilog model of compositional hardware, Putting compositional hardware on bus, Linux kernel integration, Conclusions.

3. Object, goal, and tasks

The object of the research is pragmatic dependent hardware (adaptive hardware).

The goal of the research is to propose integration method for hardware designed with adaptive approach (adaptive hardware) in computing systems.

Tasks to solve: define physical interaction interface of adaptive hardware with traditional computational hardware; provide technology for creating an operating system drivers for adaptive hardware.

4. Verilog model of adaptive hardware

A universal form of hardware representation of adaptive hardware must be defined, as well as the form of interaction of user with such hardware computing device. The interface of adaptive hardware module includes the following ports (Fig. 1):

— RST — reset signal, stops all current processing and clears output;
— ST — «start» signal, starts a new computation from any internal state, gets valid IN1...INn arguments;
— CLK — clock signal;
— RD — «ready» signal, to show that on RES is valid output;
— RES — the result of computation;
— IN1...INn — arguments for piece of adaptive hardware.

Width of RES and IN1...INn ports is the same. Now, let us consider consider the time diagrams of adaptive hardware module (Fig. 2).

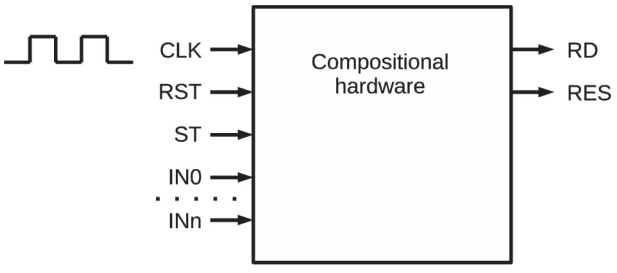

Fig. 1. Adaptive hardware module

— output data (ODATA).

Addressing on Avalon bus is byte-wise, so it is necessary be aware of the case when inputs of adaptive hardware device are more than one byte width. Under the research, a generator of Avalon peripherals from adaptive hardware device was created. It takes Verilog model of adaptive hardware device and wraps it in Avalon peripheral module, and also creates TCL file from template, which describes it as peripheral to Altera Qsys SoC designer. So, the output adaptive hardware device becomes ready to use peripheral.

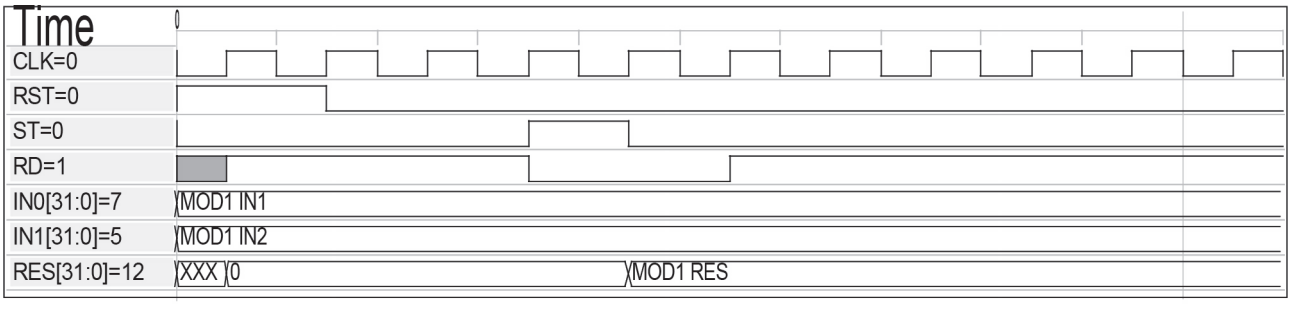

Fig. 2. Time diagram for adaptive hardware module

On the diagram, a fragment of computing process is displayed. On the power on hardware module is reset. Then valid values on IN1...INn are asserted. On the next moment «start» signal is received and the computation process started. With the start of computational process «ready» signal is deasserted. After a while, the valid result appears on the RES port and «ready» signal is asserted again.

Of course, this is not conventional interface to interact with other hardware devices. So adaptive hardware in such representation is barely applicable.

5. Putting adaptive hardware on bus

One of the efficient uses of adaptive hardware is running it on CPU based system under control of the conventional software. In order to run it on CPU based system, the hardware must be a peripheral device on CPU bus. SoCs with HPU (hardware processor unit) included is a good solution to do it. FPGA, combined with HPU, can be reconfigured with software running on HPU, which allows changing adaptive hardware peripherals in runtime, and it is very handy when resources of FPGA are limited. For testing, Altera Cyclone V SoC with quad ARM Cortex-A9 HPU was selected. Peripherals on the FPGA are interconnected with Avalon bus, which bridged to HPU native (AXI) [8].

Peripheral on the Avalon bus contains the following ports:

— reset signal (RST);
— clocking (CLK) signal;
— write anable (WE) signal;
— chip select or strobe (CS or STB);
— address (A);
— input data (IDATA);

Such devices connected to HPU via lightweight HPS-to-FPGA master interface. The lightweight HPS-to-FPGA interface is a 32-bit AXI master that allows HPS masters to issue transactions to the FPGA fabric. The interface is AXI-3 compliant. The HPS-side AXI bridge manages clock crossing, buffering, and data width conversion where necessary (Fig. 3).

Other interface standards in the FPGA fabric, such as connecting to Avalon-MM interfaces, can be supported through the use of soft logic adapters. The Qsys system integration tool automatically generates adapter logic to connect AXI to Avalon-MM interfaces. Each AXI bridge accepts a clock input from the FPGA fabric and performs clock domain crossing internally. The exposed AXI interface operates on the same clock domain as the clock supplied by the FPGA fabric.

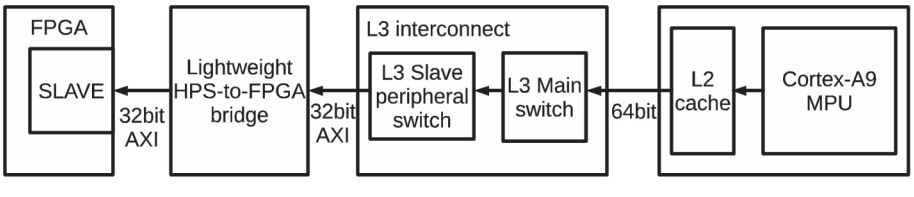

Fig. 3. Connecting adaptive hardware to HPU via AXI bus

6. Linux kernel integration

The next task to solve after the system bus integration is OS integration. Linux based operating systems were chosen as the most robust and widespread operating systems with strong community support. Of course one can not simply access physical addresses of computing system, every user application running in Linux-based operating system, runs in own address space, such address spaces are called «userspace». There is no direct access to physical memory from the userspace [9]. But there is another address space in Linux based operating systems — kernel space. Physical address space can be

easily mapped to kernel space. There are two ways to access kernel space: to patch the kernel or to make own kernel module. The first way, obviously, leads to considerable overhead on kernel build process and system reboot in order to apply freshly build kernel to the system. From other side the concept of loadable kernel modules can mitigate problems mentioned before. Loadable kernel module (or LKM) is an object file that contains code to extend the running kernel of an operating system. When the functionality provided by a LKM is no longer required, it can be unloaded in order to free memory and other resources. The other problem to solve is how the kernel code will interact with user space applications. There two major ways of the interaction: system calls and virtual file system. The first way is undesirable because adding new system calls mainly conducted by the Linux kernel developers and kernel call code once taken by software developer, may be used in the following version of the Linux kernel by kernel developers. So the device will be integrated in sysfs. Sysfs is a virtual file system provided by the Linux kernel. By using virtual files, sysfs exports information about various kernel subsystems, hardware devices and associated device drivers from the kernel's device model to user space. In addition to providing information about various devices and kernel subsystems, exported virtual files are also used for their configuring. User model of the device contains $n+1$ files, each of n files corresponds one input and one corresponds output. They are named in following manner in1, in2, in3,...inn, res (Fig. 4). Each «in» file is write only and the «res» is read only. Computation launches after last input (inn) written. All the files are located */sys/kernel/mm/devicename* (Fig. 4).

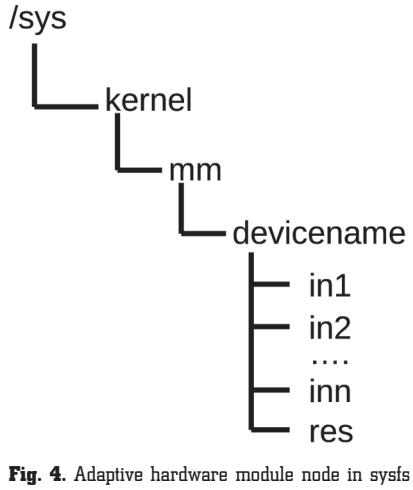

Fig. 4. Adaptive hardware module node in sysfs

Where «devicename» is node with name of adaptive hardware device. Consider kernel API used to create this device driver. Firstly *accel_kobj = kobject_create_and_add (accelerator, mm_kobj)* called, where *mm_kobj* is node of */sys/kernel/mm*, which is defined in kernel headers, and «accelerator» is «devicename», discussed previously. As the result, new kobject node returned. With call *sysfs_create_group (accel_kobj, &attr_group)* all files to node *accel_obj* were added. *attr_group* is entity of *struct attribute_group*. The attribute group interface is a simplified interface for easy addition and removal of the attributes set with a single call. An attribute group is simply an array of attributes to be added to an object, as represented by the *attrs* field. «Attribute» means leaf a (file) in sysfs filesystem. Field *attrs* of *struct attribute_group* contains NULL terminated array of struct kobj_attribute, which contains name of the attribute, read/write permissions and pointers to read/write functions for each attribute. Permissions are set to 0666, that everyone on the system could exploit the hardware. All attributes are applied with *kobject_put (accel_kobj)*.

The other thing to discuss is device memory mapping. *ioremap()* function is used to map physical memory to kernel memory space. In order to unmap memory *iounmap()* used. Final version of the driver source code is in [10].

7. Conclusions

Recently developed adaptive approach to pragmatics dependent hardware design would be incomplete without instructions and methods of its usage with other information technologies and computing devices. In this article, a new approach to integration of conventional computing systems and adaptive hardware is proposed. This allows to automate development starting with the semantical solution design up to its integration with user software. It makes adaptive hardware able to be used with common software in the way which software engineers got used to. The new framework for adaptive hardware application can be a basis of hybrid computing systems, which will involve adaptive hardware as well as conventional software. Although, there are many things to do in future. Integration with other operating systems, interrupt processing, argument transfer rate optimization, improvement of programming model are major directions of future research.

References

1. Zakharchenko, T. Pragmatics dependent hardware design [Text] / T. Zakharchenko // 2014 IEEE 34th International Scientific Conference on Electronics and Nanotechnology (ELNANO). — Kyiv: IEEE, 2014. — P. 462–465. doi:10.1109/elnano.2014.6873429
2. Maltsev, A. I. Algorithms and recursive functions [Text] / A. I. Maltsev. — Groningen: Wolters-Noordhoff, 1970. — 372 p.
3. Cyclone V Device Overview [Text]. — Altera, 2015. — 41 p.
4. Arria 10 Device Overview [Text]. — Altera, 2015. — 44 p.
5. Stratix X Overview [Electronic resource] // Altera. — Available at: \www/ URL: http://www.altera.com/devices/fpga/stratix-fpgas/stratix10/stx10-index.jsp. — September 19, 2015.
6. Xilinx UltraScale MPSoC Architecture [Text]. — Xilinx, 2014. — 50 p.
7. Marko, K. For Intel, Adding Altera, FPGA Hardware Is Easy: Next Comes Supporting Software [Electronic resource] / K. Marko // Forbes. — June 9, 2015. — Available at: \www/ URL: http://www.forbes.com/sites/kurtmarko/2015/06/09/intel-fpga-software/. — September 19, 2015.
8. Cyclone V Hard Processor System Technical Reference Manual [Text]. — Altera, 2015. — 3329 p.
9. Love, R. Linux Kernel Development [Text] / R. Love. — Ed. 3. — Addison Wesley, 2010. — 440 p.
10. Zakharchenko, T. L. Adaptive hardware module driver source code [Electronic resource] / T. L. Zakharchenko // Github. — Available at: \www/ URL: https://github.com/player999/accelerator_km. — September 19, 2015.

ИНТЕГРАЦИЯ АДАПТИВНОГО АППАРАТНОГО ОБЕСПЕЧЕНИЯ

В статье описано процесс интеграции аппаратного обеспечения, разработанного с применением адаптивного подхода,

в вычислительные системы. Процесс интеграции описан на всех уровнях абстракции: от его модели, описанной на HDL-языке, до драйвера операционной системы. Это позволяет упростить использование такого аппаратного обеспечения конечным пользователем и улучшить применяемость адаптивного подхода к реальным задачам.

Ключевые слова: адаптивное аппаратное обеспечение, вычислительные системы, интеграция с ОС, ускорение вычислений.

Захарченко Тарас Леонідович, *аспірант, кафедра конструювання електронно-обчислювальної апаратури, Національний технічний університет «Київський політехнічний інститут», Україна,* ***e-mail: taras.zakharchenko@gmail.com****.*

Захарченко Тарас Леонидович, *аспирант, кафедра конструирования электронно-вычислительной аппаратуры, Национальный технический университет «Киевский политехнический институт», Украина.*

Zakharchenko Taras, *National Technical University of Ukraine «Kyiv Polytechnic Institute», Ukraine,*
e-mail: taras.zakharchenko@gmail.com

УДК 004.056.055
DOI: 10.15587/2312-8372.2015.51221

Миронець І. В.

ПІДВИЩЕННЯ ДОСТОВІРНОСТІ ПРОЦЕСУ МАТРИЧНОГО КРИПТОГРАФІЧНОГО ПЕРЕТВОРЕННЯ

Швидкісні криптографічні перетворення даних є найефективнішим засобом забезпечення конфіденційності та цілісності інформаційних ресурсів, тому перспективним напрямком досліджень є розробка методів підвищення продуктивності криптографічних систем.

В статті було сформовано два підходи щодо підвищення достовірності процесу криптографічного перетворення: перший підхід дозволяє проконтролювати корекцію синтезу матриці оберненого перетворення, другий — проконтролювати весь процес проходження оберненого перетворення інформації.

Ключові слова: *криптографічне перетворення, захист інформації, конфіденційність, цілісність інформації, кодування, шифрування інформації.*

1. Вступ

Сучасні методи накопичення, обробки та передачі інформації сприяли появі загроз, пов'язаних з можливістю втрати, розкриття, модифікації даних, що належать кінцевим користувачам.

Використання подібних заходів приводить до необхідності використання комп'ютерів для вирішення специфічних завдань, для вирішення яких необхідна наявність специфічних алгоритмів.

Зважаючи на важливість інформації, яка передається відкритими лініями зв'язку, розрізняють приватну, комерційну, службову та державну таємницю. Враховуючи це, зрозуміло, що таку інформацію потрібно передавати не у відкритому вигляді, а в модифікованому таким чином, щоб перетворити її у відкритий вигляд могла лише особа, що знає необхідне обернене перетворення. Існують два основні способи перетворення даних — кодування та шифрування [1, 2].

Таким чином можна зробити висновок, що захист конфіденційної інформації, що представлена у цифровому вигляді, на сьогоднішній день є достатньо актуальним завданням. Одним із ефективних підходів щодо захисту інформації в комп'ютерних системах є використання криптографічних методів захисту, зокрема блокового шифрування даних.

2. Аналіз літературних даних та постановка проблеми

Для забезпечення конфіденційності та цілісності інформації [3–5] в багатьох випадках криптографічне перетворення є чи не єдиним шляхом (з певною стійкістю до спроб розкриття її змісту — криптографічною стійкістю). На даний час широко відомими є кілька методів криптографічного захисту інформації, серед яких досить важливими є аналітичні матричні перетворення, що розглянуті, наприклад в [3, 4], і до яких відносяться запропоновані авторами в роботах [5, 6] алгоритми криптографічного перетворення [7].

Важливий вклад у розвиток криптології та захисту інформації зробили такі вітчизняні та зарубіжні науковці, як І. Д. Горбенко, А. А. Молдовян, В. А. Мухачев, В. А. Лужецький, В. А. Хорошко, Ю. В. Кузнецов, О. Г. Корченко, Г. Ф. Конахович, Б. Шнайер, М. Хелльман, Ч. Г. Беннет, Ж. Брассар та інші.

Проте залишається цілий ряд невирішених задач, що мають важливе значення. Беручи до уваги те, що швидкісні криптографічні перетворення даних є найефективнішим засобом забезпечення таких характеристик безпеки інформаційних ресурсів, як конфіденційність і цілісність, то, безумовно, перспективним напрямком досліджень є розробка методів підвищення продуктивності криптографічних систем. А саме дане дослідження