ПРОБЛЕМЫ ВНЕДРЕНИЯ SystemC В КАЧЕСТВЕ КАРКАСА ДЛЯ ВИРТУАЛЬНОГО ПРОТОТИПИРОВАНИЯ БОРТОВОЙ АППАРАТУРЫ И СТЕНДОВ ДЛЯ РАЗРАБОТКИ БПО

PROBLEMS OF IMPLEMENTING SystemC INTO EXISTING PROCESS OF VIRTUAL PROTOTYPING HARDWARE AND TEST BENCHES FOR EMBEDDED SOFTWARE DEVELOPMENT

УДК 004.414.2

МАДУМАРОВ ТАЛГАТ АСХАТОВИЧ
Madrook00@gmail.com

СТЕФАНЦОВ АЛЕКСЕЙ ВЯЧЕСЛАВОВИЧ
Stef_zel@mail.ru

АО «Научно-исследовательский институт «Субмикрон»
124460, г. Москва, г. Зеленоград,
Георгиевский проспект, 5, стр. 2

MADUMAROV TALGAT A.
Madrook00@gmail.com

STEFANTSOV ALEXEY V.
Stef_zel@mail.ru

"Scientific Research Institute "Submicron" JSC
2/5, Georgievsky Ave.,
Zelenograd, Moscow, 124460

Рассмотрены технические проблемы внедрения виртуального прототипирования на SystemC в существующий маршрут разработки встраиваемых систем. Особое внимание уделено проблемам, связанным с управлением симулятором SystemC извне и оптимизацией производительности виртуального прототипа.
Ключевые слова: виртуальное прототипирование; SystemC; встраиваемые системы; ESL-проектирование; C/C++.
SystemC is one of most popular frameworks for virtual prototyping of embedded systems, but it requires a certain toolchain to be used for model usage and verification. This work is about solving problems of implementing SystemC into existing toolchain.
Keywords: virtual prototyping; SystemC; embedded systems; ESL-modelling; C/C++.

При проектировании встраиваемых систем часто возникает необходимость в модели системы высокого уровня, на которой была бы возможной отладка бортового программного обеспечения. Такая модель часто называется виртуальным прототипом (virtual prototype) или виртуальным прототипом системного уровня (system-level virtual prototype) [1] и должна обладать следующими свойствами:

- достаточной функциональностью и точностью, для того чтобы позволять выполнять на ней немодифицированное ПО, включая драйверы, операционные системы и приложения;
- достаточным быстродействием для комфортной отладки и тестирования этого ПО;
- необходимым отладочным функционалом для работы совместно с отладчиком.

Изначально в НИИ «Субмикрон» виртуальное прототипирование аппаратуры велось без использования сторонних каркасов. В качестве виртуального прототипа использовалась модель уровня такта шины, написанная на чистом С. С помощью имеющихся моделей успешно решались задачи отладки и тестирования бортового ПО, включая специальное программное обеспечение (СПО) и тестовое программное обеспечение (ТПО).

При выборе каркаса для ускорения разработки виртуальных прототипов было принято решение использовать в этом качестве библиотеку SystemC [2], обладающую рядом достоинств:

- богатый функционал библиотеки, поддерживающий глобальный отсчет времени, событийно-ориентированное моделирование с возможностью моделирования одновременных событий, раздельное моделирование отдельных блоков и интерфейсов взаимодействия между ними;
- SystemC является промышленным стандартом, поддерживаемым целым рядом САПР, среди которых Mentor Graphics, Cadence и другие;
- SystemC является библиотекой для языка C++, что позволяет использовать богатый синтаксис этого языка программирования, различные библиотеки, а также имеющиеся наработки и модели;
- в библиотеке SystemC имеется синтезируемое подмножество [3], которое делает возможным синтез аппаратуры из RTL-моделей, описанных при помощи этой библиотеки;
- SystemC поддерживает моделирование аппаратуры на различных уровнях абстракции, от чисто функционального уровня до синтезируемых моделей RTL-уровня, причем поддерживает одновременную симуляцию моделей разного уровня на одном движке.

В настоящий момент библиотека SystemC является очень популярным средством для разработки моделей встраиваемых систем на различных уровнях абстракции в целом, в частности для виртуального прототипирования.

При внедрении SystemC в уже имеющийся конвейер виртуального прототипирования возник ряд проблем, решение которых и представлено в докладе.

Первой проблемой является внедрение виртуального прототипа в имеющийся процесс разработки ТПО. Дело в том, что ТПО состоит из программы, загруженной в тестируемую машину,

и сценария отладчика, который занимается генерацией тестовых данных, передачей этих данных тестируемой машине при помощи аппаратуры стенда, а также анализом полученных результатов. В связи с этим возникает необходимость моделирования взаимодействия аппаратуры со стендом, причем управляемое из сценария отладчика. Это делает невозможным использование методики тестирования, описанной в литературе, рекомендованной разработчиком SystemC [4], при которой и генерация тестовых данных, и верификация результатов управляются средствами SystemC. Конечно, такой подход дает высокую точность симуляции, но, с другой стороны, поскольку виртуальный прототип стенда во время отладки ТПО должен вести себя так же, как и аппаратура, то и выдавать тестовые данные он должен не в один и тот же момент, а с некоторым допуском. Проблема синхронизации виртуального прототипа с потоком, выдающим внешние данные, была решена при помощи введения в основной конечный автомат симуляции SystemC нового состояния, во время которого симуляция приостанавливается и появляется «окно» для получения специальным интерфейсом указателя на новые данные и нотификацию события модели блока приема этих данных о том, что данные были переданы. Такой подход обладает рядом достоинств: он универсален, подходит для любого уровня моделирования и не нарушает спецификации SystemC. С другой стороны, он не гарантирует абсолютной точности синхронизации сам по себе и, для того чтобы использовать его там, где необходима по крайней мере ограниченная сверху ошибка синхронизации, данный подход неприменим без соответствующих изменений.

Вторая проблема заключается в том, что SystemC не поддерживает многопоточность внутри симуляции, что негативно сказывается на производительности симуляции, особенно для таких высокоуровневых моделей, как виртуальный прототип, где подчас синхронизация между модулями происходит довольно редко. Например, процессоры семейства Multicore [5] обладают дополнительным ядром DSP, синхронизация с которым происходит довольно редко. Однако, если вынести моделирование всех вычислительных процессов всех DSP-ядер виртуального прототипа многопроцессорной ВМ в отдельные потоки, то можно получить некоторый прирост производительности. Для этого в симуляцию был введен параметр точности: через какое количество тактов DSP-сопроцессора ему необходимо синхронизироваться с движком симуляции всей системы. Подбирая эмпирическим путем этот коэффициент для каждой задачи, можно добиться существенного ускорения симуляции.

Данные моменты собраны вместе, поскольку они на самом деле обращаются с двух разных сторон к одной проблеме. С одной стороны, вычислительные машины наращивают производительность за счет увеличения числа ядер на борту, с другой — библиотека SystemC не позволяет разработчику без модернизации производить симуляцию параллельно, максимально используя вычислительные мощности компьютера. С учетом растущей сложности встраиваемых систем не за горами такой момент, когда быстродействия стандартного ядра SystemC не будет хватать для разработки виртуальных прототипов таких сложных систем, как встраиваемые системы космического назначения. Поэтому если не сама библиотека, то по крайней мере подход к ее использованию должен быть модернизирован с учетом тех вызовов, которые бросит разработчикам завтрашний день.

ЛИТЕРАТУРА

1. Bailey B., Martin G. E*SL Models and Their Application: Electronic System Level Design and Verification in Practice*. Springer Science & Business Media, 2009. 446 p.
2. IEEE Standard System C Language Reference Manual.
3. SystemC Synthesizable Subset Version 1.4.7 [Electronic resource] // http://www.accellera.org. 2016. URL: http://www.accellera.org/images/downloads/standards/systemc/SystemC_Synthesis_Subset_1_4_7-Apache.pdf.
4. Grötker T. *et al. System Design with SystemC™*. Springer Science & Business Media, 2007. 219 p.
5. http://multicore.ru.

ТЕХНОЛОГИЯ УДАЛЕННОЙ ОТЛАДКИ АППАРАТНОЙ ЧАСТИ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМ НА ОСНОВЕ ПЛИС

REMOTE HARDWARE AND SOFTWARE DEBUGGING TECHNOLOGY OF FPGA-BASED SYSTEMS

УДК 004.41

ПАВЛОВ АНТОН НИКОЛАЕВИЧ[1]
antony@niisi.msk.ru

ЛИВЕНЦЕВ ЕВГЕНИЙ ВАСИЛЬЕВИЧ[2]
liventsev@gmail.com

СИЛАНТЬЕВ АЛЕКСАНДР МИХАЙЛОВИЧ[2]
olmer.aod@gmail.com

[1] *ФГУ ФНЦ НИИСИ РАН*
117218, г. Москва, Нахимовский просп., 36, к. 1
Тел.: +7 (495) 719-78-29
[2] *НИУ МИЭТ*
124498, г. Москва, г. Зеленоград, площадь Шокина, 1
Тел.: +7 (499) 720-87-70

PAVLOV ANTON N.[1]
antony@niisi.msk.ru

LIVENTSEV EVGENII V.[2]
liventsev@gmail.com

SILANTIEV ALEXANDER M.[2]
olmer.aod@gmail.com

[1] *SRISA RAS*
36/1 Nakhimoskiy Ave., Moscow, 117218
Tel.: +7 (495) 719-78-29
[2] *National Research University of Electronic Technology — MIET*
1 Shokin Square, Zelenograd, Moscow, 124498, Russia
Tel.: +7 (499) 720-87-70

В данной работе рассматриваются вопросы повышения производительности труда разработчиков систем на основе ПЛИС, в состав которых входит микропроцессорное ядро.
Ключевые слова: ПЛИС; микропроцессорное ядро; разработка аппаратуры; разработка ПО; встречная верификация; удаленное управление.
This paper considers hardware and software development of FPGA-based systems with microprocessor core. The phenomenon under study is increasing software and hardware development efficiency.
Keywords: FPGA; microprocessor IP-core; open software; hardware development; software development; hardware/software co-verification; remote control.

При создании системы на основе ПЛИС с микропроцессорным ядром одновременно разрабатывается как аппаратная часть, так и ПО, в первую очередь системное (программа начальной загрузки, ядро ОС).

Как при разработке аппаратной части, так и при разработке ПО, отладка и тестирование являются важным и трудоемким этапом. Для проведения данного этапа, как правило, создаются стенды либо на базе готовых отладочных комплектов с ПЛИС, либо на базе специально разработанных плат. Одновременная разработка аппаратной части и ПО позволяет организовать на стенде встречное тестирование: разработчики аппаратной части используют в качестве тестов приложения пользователя [1], разработчики системного ПО используют стенды для проверки работоспособности ПО на целевой платформе.

При использовании стендов на базе ПЛИС приходится сталкиваться со следующими трудностями:

- стенды на базе ПЛИС, как правило, имеют высокую стоимость, а их количество ограничено, и значит, затруднительно выделить каждому разработчику отдельный стенд в персональное пользование; передача стенда от разработчика к разработчику происходит не мгновенно: как правило, требуется переналадка стенда, во время которой использование стенда невозможно;
- стенды могут быть громоздки и требовать создания специальных условий для эксплуатации; т. е. не всегда возможно разместить стенд поблизости от рабочего места разработчика и наоборот;
- для тщательного тестирования системного ПО приходится регулярно проверять промежуточные (внутренние) версии ПО, что может быть затруднено, если стенд в этот момент занят другим разработчиком.

Указанные трудности порождают проблему: уменьшается полезное рабочее время стендов, которое можно использовать для тестирования разрабатываемой системы, простаивают разработчики, что приводит к снижению производительности труда. Кроме того, несложные, но трудоемкие, времязатратные ручные операции со стендами повышают значение человеческого фактора как источника ошибок в работе.

Одним из путей решения данной проблемы является переход к удаленному режиму работы со стендами, при котором как можно больше операций, необходимых для тестирования и отладки аппаратной части и программного обеспечения, можно производить без непосредственного физического доступа к стенду.

В этой связи актуальной является задача создания технологии удаленной отладки аппаратной части и программного обеспечения систем на основе ПЛИС, которая позволит использовать стенды удаленно и организовать совместную работу нескольких разработчиков с одним и более стендами.

Идея, лежащая в основе предлагаемой технологии, заключается в том, что как можно большее число информационных интерфейсов стенда замыкается на специально разработанное *устройство управления*, подключенное к компьютерной сети; по возможности взаимодействие со стендом для тестирования и отладки производится при помощи устройства управления стендом удаленно (разумеется, кроме случаев, когда необходимо произвести ремонт стенда или внести физические изменения в аппаратуру стенда). Подобный подход уже описан ранее в работах [2, 3] и [4, 5], но в отличие от предлагаемой технологии, упор в них делается на автоматический прогон тестовых сборок ПО в фиксированных конфигурациях.

ТИПОВЫЕ ОПЕРАЦИИ И ИСПОЛЬЗУЕМЫЕ ИНТЕРФЕЙСЫ

Типовой цикл отладки системы на базе ПЛИС с микропроцессорным ядром включает в себя обновление конфигураций ПЛИС стенда, прогон тестового ПО для проверки по крайней мере базовой работоспособности обновленной конфигурации ПЛИС и дальнейшую совместную работу разработчиков ПО и разработчиков аппаратуры.

Для обновления конфигураций ПЛИС стенда традиционно используется интерфейс JTAG.

Прогон тестового ПО связан с получением информации от тестового ПО. Обычно для этой цели используются интерфейсы UART и/или сеть Ethernet.

При отладке ПО и аппаратной части нередко возникают ситуации зависания, для выхода из которых в разрабатываемую систему приходится подавать сигнал СБРОС; в тех случаях, когда подача сигнала СБРОС не приводит к выводу системы из состояния зависания, разработчику приходится прибегать к отключению питания стенда с последующим включением и, возможно, повторным конфигурированием ПЛИС.

В составе разрабатываемой системы нередко присутствуют накопители информации на базе микросхем ППЗУ с интерфейсами SPI и I2C; в этом случае в составе стенда отладки предусматриваются отдельные технические средства для программирования этих микросхем (программаторы).

Нередко отлаживаемая система поддерживает несколько режимов работы в зависимости от состояния входных линий интерфейса дискретных сигналов по снятию сигнала СБРОС; для отладки работы системы в разных режимах в составе стенда предусматривается возможность установки состояния входных линий интерфейса дискретных сигналов, например, при помощи внешних замыкателей.

СУЩЕСТВУЮЩИЕ РЕШЕНИЯ ДЛЯ ОРГАНИЗАЦИИ УДАЛЕННОЙ ОТЛАДКИ СИСТЕМ НА БАЗЕ ПЛИС

Существующие решения для организации удаленной отладки на базе ПЛИС представлены устройствами, подключаемыми к сети Ethernet и обеспечивающими конфигурирование ПЛИС через интерфейс JTAG. К таковым можно отнести EthernetBlaster II Communications Cable [6] и Catapult EJ-2 Ethernet-to-JTAG Cable [7]. Использованию данных устройств в качестве полноценных устройств удаленного управления стендом отладки препятствуют:

- отсутствие поддержки интерфейсов UART, SPI, I2C;
- невозможность использовать устройство для подачи общесистемного сигнала СБРОС;
- отсутствие возможности управления питанием отлаживаемой системы.

Вместе с тем существует решение, которое можно использовать в качестве составных частей устройства удаленного управления: Bus Pirate [8], многофункциональное устройство, предоставляющее пользователю возможность подключиться к отлаживаемой программно-аппаратной системе через интерфейсы UART, I2C, SPI, JTAG. У Bus Pirate имеется ряд недостатков:

- Bus Pirate не имеет сетевого интерфейса, подключение к ПК пользователя происходит через интерфейс USB;
- интерфейсы UART, I2C, SPI, JTAG мультиплексированы, а значит, нет возможности их одновременного использования без перекоммутации;

While creating FPGA-based systems with microprocessor core both hardware and software are being developed simultaneously (the system part in the first place: bootloader, kernel).

Debugging and testing are a very important and tedious stage in developing both hardware and software. Usually debugging is done with devices based on either existing FPGA debug kits or specifically designed ones. Simultaneous development of hardware and software allows organizing a cross-testing: hardware developers use user applications as tests [1], system software developers use prototype stands for testing as target platform.

While using FPGA-based testing stands the following difficulties are encountered:

- FPGA-based testing stands are usually very expensive and their number is limited, so it's impossible to have a separate stand for each developer's personal use. The transfer of the stand from one developer to another is not instant, and it takes some time to reconfigure it making its use impossible during the period.
- Stands could be bulky and may need special working conditions; that's why it's not always possible to place the stand near the developer's workplace, and vice versa.
- For scrutinous testing of the system software it's important to check intermediate (internal) software versions which could be difficult if the stand is already in use by another developer.

All these difficulties cause the problem: the useful working time of each stand is reduced which leads to a decrease in working efficiency. Besides, manual operations with stands which could be simple but intensive and time-consuming raise the significance of human factor as the source of errors.

One of the ways to solve this problem is transition to remote work with stands when most operations needed for hardware testing and debugging are being done without physical access to the stand.

In this regard the task is to elaborate a technology of remote hardware and software debug of FPGA-based systems, which makes it possible to work with testing stands remotely and to organize joint access of several developers to one or more stands.

The main idea of the technology is to wire all interfaces to a specially designed control device with network support. All interaction with the testing stand is done remotely via this control device (except cases when repair or hardware reconfiguration is needed) [3].

Similar approach was described earlier in works [2, 3] and [4, 5] but unlike the technology proposed here the main focus was on automated execution of software testing in fixed configurations.

TYPICAL OPERATIONS AND INTERFACES USED

The typical debugging cycle of FPGA-based system with microprocessor core includes updating the FPGA configurations of the stand, execution of the testing software for checking at least basic functioning of updated

- имеются возможность управлять питанием подключенного оборудования, однако обеспечиваемое управляемым источником питания напряжение 3,3 В годится для узкого круга отлаживаемых систем, а максимальный выдаваемый источником ток составляет всего лишь 150 мА.
- хотя имеется возможность использовать Bus Blaster в качестве JTAG-адаптера, однако это требует замены встроенного программного обеспечения Bus Pirate, а пропускная способность такого решения ограничена 115200 бит/с.
- в зависимости от потребностей отдельной отлаживаемой системы устройство удаленного управления соединяется с отлаживаемой системой требуемым набором информационных интерфейсов: UART, SPI, I2C.
- для управления подачей сигнала СБРОС в составе устройства удаленного управления предусмотрен блок реле, работающих на замыкание/размыкание; этот же блок позволяет осуществлять управление подачей питания в отлаживаемую систему.

РЕАЛИЗАЦИЯ

Структурно стенд удаленной отладки в рамках предлагаемой технологии строится следующим образом (см. рис. 1):

- основой стенда является устройство удаленного управления, которое контролирует одну или более отлаживаемых систем на основе ПЛИС (англ. device under test);
- устройство удаленного управления оснащено двумя интерфейсами Ethernet, один из которых используется для подключения к локальной вычислительной сети (ЛВС), к которой подключены рабочие места разработчиков (пользователей стенда); второй интерфейс Ethernet используется для подключения к изолированной тестовой локальной сети, используемой, при необходимости, для обменов с отлаживаемыми системами;

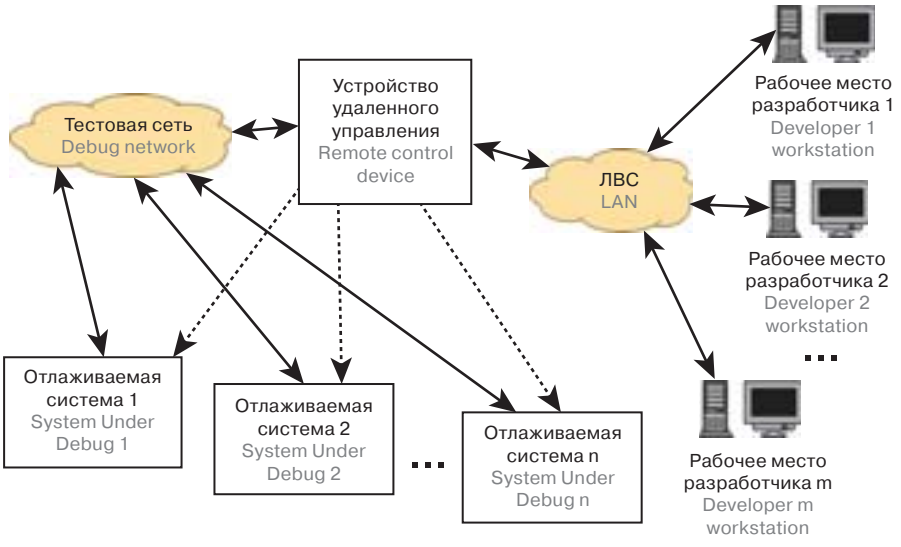

Рис. 1. Схема стенда удаленной отладки
Fig. 1. Remote debug stand

FPGA configuration and further joint work of software and hardware developers.

Traditionally, for updating FPGA configurations JTAG is used.

The execution of software testing is connected with receiving information from it. Usually, UART interface is used for the purpose and/or Ethernet.

During debugging of software or hardware hang-ups may occur which could be resolved by RESET signal sent to the system under debug. In cases when sending RESET signal doesn't work the developer has to turn off the system and then to turn on with possible consequent FPGA reconfiguration.

The system being developed may comprise data storage media based on PROM with SPI and I2C interfaces; in this case the debug stand may contain separate technical tools for programming such chips.

Fairly often the system being debugged supports several working modes depending on states of input lines of discrete signals interface for RESET; in order to debug the system in specific mode the working stand is capable of setting these lines into desired state with the help of external jumpers.

EXISTING SOLUTIONS FOR REMOTE DEBUGGING OF THE FPGA-BASED SYSTEMS

The existing solutions for remote debugging of the FPGA-based systems are represented by devices with Ethernet and JTAG for FPGA reconfiguration. Those include EthernetBlaster II Communications Cable [6] and Catapult EJ-2 Ethernet-to-JTAG cable [7]. Features preventing from using them as full-fledged devices for stand remote control are as follows:

- lack of UART, SPI, I2C interfaces support;
- inability to send RESET signal to the system;
- lack of power control of debugged system.

However, there is a solution that could be used as a composite part of the remote control device: Bus Pirate [8], a multi-functional device which allows connecting to debugged hardware/software system via such interfaces as UART, I2C, SPI, JTAG. But Bus Pirate has a number of flaws:

- lack of network interface; connection to PC is done via USB;
- UART, I2C, SPI, JTAG interfaces are multiplexed which makes it impossible to use them simultaneously without re-wiring.
- there is a possibility to control the power of connected devices, however the voltage of 3,3V is applicable to a narrow range of systems being debugged, and maximum current strength is mere 150mA.
- though Bus Pirate can be used as a JTAG adapter, this will require changing the embedded software, and pass-through capacity is limited to 115200bits/sec.

IMPLEMENTATION

The structure of the remote debugging stand can be represented by the following scheme (Fig. 1):

- the basis of the stand is the remote control device which is connected to one or more debugged FPGA-based systems (devices under test);
- remote control device is equipped with two Ethernet interfaces; the first is used for local area network (LAN) for connecting to developer and user workplaces; the second is used for isolated local test network for interchanging data with systems under debug;

Для реализации устройства удаленного управления использованы готовые доступные (off-the-shelf) модули. Например, в качестве управляющей ЭВМ используется одноплатная ЭВМ Raspberry Pi 3.

ПО устройство построено на базе свободно распространяемых программных компонентов. В качестве ОС используется ОС на базе Debian Linux.

РЕЗУЛЬТАТЫ

Разработана технология удаленной отладки, включающая в себя комплекс аппаратных, программных и методических средств:

- устройство управления стендом, которое реализует:
  - управление питанием стенда;
  - работу с интерфейсами RS232, UART, SPI, I2C, JTAG, Ethernet;
  - управление режимом работы отлаживаемой системы при помощи интерфейса дискретных сигналов;
  - подачу сигнала СБРОС;
  - наблюдение за состоянием стенда;
  - обновление конфигураций ПЛИС стенда.
- ПО для организации авторизованного доступа к стенду, организации совместной работы, а также технологическое ПО для управления аппаратными средствами стенда;
- методические рекомендации и типовые приемы работы со стендами.

ЗАКЛЮЧЕНИЕ

Дальнейшее совершенствование технологии удаленной отладки заключается прежде всего в расширении сферы ее применения, реализации в устройстве управления стендами функциональности измерительного оборудования и автоматизации запуска тестов.

ЛИТЕРАТУРА

1. Чибисов П.А. Встречное тестирование высокопроизводительных микропроцессоров: диссертация ... кандидата технических наук: 05.13.11 / Чибисов Петр Александрович; [Место защиты: Ин-т систем. программирования].— Москва, 2013. — 174 с.: ил.
2. LinuxBIOS Automated Distributed Test System Test Integration Manual V1.0 2006/11/22 [Электронный ресурс] // URL: https://www.coreboot.org/PDFs/LinuxBIOS-testing/TestIntegrationManual.pdf (дата обращения: 09.08.2017).
3. Raptor Engineering Automated Coreboot Test Stand (REACTS) [Электронный ресурс] // URL: https://static.rpteng.com/REACTS/pdf/reacts_users_guide.pdf (дата обращения: 18.12.2017).
4. Hardware Infrastructure of Free Electrons' Lab [Электронный ресурс] // URL: https://free-electrons.com/blog/hardware-infrastructure-free-electrons-lab/ (дата обращения: 18.12.2017).
5. Software Architecture of Free Electrons' Lab [Электронный ресурс] // URL: https://free-electrons.com/blog/software-architecture-free-electrons-lab/ (дата обращения: 18.12.2017).
6. EthernetBlaster II Communications Cable [Электронный ресурс] // URL: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ethernetblasterii.pdf (дата обращения: 18.12.2017).
7. Catapult EJ-2 Ethernet-to-JTAG Cable [Электронный ресурс] // URL: http://www.byte-tools.com/product/catapultej2 (дата обращения: 18.12.2017).
8. Bus Pirate [Электронный ресурс] // URL: http://dangerousprototypes.com/docs/Bus_Pirate (дата обращения: 18.12.2017)

- depending on particular needs of each debugged system the remote control device is connected to it with a needed set of interfaces: UART, SPI, I2C.
- for controlling the RESET signal the remote control device contains the relay box for locking/unlocking the circuit; the same box is used for controlling the power supply of the system under debug.

Existing off-the-shelf modules are used for implementing the remote control device. For example, single-board Raspberry Pi 3 is used as a controlling PC.

The software is based on open-source components, and the operating system is based on Debian Linux.

RESULTS

The remote debug technology has been developed including the set of hardware, software and methodology:

- stand control device that is responsible for:
  - stand power-supply control
  - RS232, UART, SPI, I2C, JTAG, interfaces, Ethernet
  - working mode control of the system under debug by discrete signals interface
  - RESET signal transmission
  - stand state surveillance
  - FPGA configuration update
- software for controlling the authorized access to the stand and organizing the joint work, as well as technological software for controlling the stand hardware;
- methodological recommendations and typical work approaches to testing stands.

CONCLUSION

Further improvement of remote debug technology lies in expanding its range of application, implementing measuring equipment in stands' control device and test running automating.

REFERENCES

1. Chibisov P.A. *High-performance Microprocessors Counter-Testing Methods*. PhD thesis, ISP RAS, Moscow, 2013 (Russian: Chibisov P.A. Vstrechnoe testirovanie vysokoproizvoditel'nykh mikroprotsessorov: dissertatsiya ... kandidata tekhnicheskikh nauk: 05.13.11 / Chibisov Petr Aleksandrovich; [Mesto zashchity: In-t sistem. programmirovaniya].— Moskva, 2013. 174 p. : il.). (In Russian).
2. *LinuxBIOS Automated Distributed Test System Test Integration Manual V1.0 2006/11/22* // URL: https://www.coreboot.org/PDFs/LinuxBIOS-testing/TestIntegrationManual.pdf (Retrieved 09 Aug 2017).
3. *Raptor Engineering Automated Coreboot Test Stand (REACTS)* // URL: https://static.rpteng.com/REACTS/pdf/reacts_users_guide.pdf (Retrieved 18 Dec 2017).
4. *Hardware Infrastructure of Free Electrons' Lab* // URL: https://free-electrons.com/blog/hardware-infrastructure-free-electrons-lab/ (Retrieved 18 Dec 2017).
5. *Software architecture of Free Electrons' Lab* // URL: https://free-electrons.com/blog/software-architecture-free-electrons-lab/ (Retrieved 18 Dec 2017).
6. *EthernetBlaster II Communications Cable* // URL: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ethernetblasterii.pdf (Retrieved 18 Dec 2017).
7. *Catapult EJ-2 Ethernet-to-JTAG Cable* // URL: http://www.byte-tools.com/product/catapultej2 (Retrieved 18 Dec 2017).
8. *Bus Pirate* // URL: http://dangerousprototypes.com/docs/Bus_Pirate (Retrieved 18 Dec 2017).