УДК 519.685

ИНСТРУМЕНТЫ ОТЛАДКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Г.Е. Замосковный, С.Г. Казанцев
(ФГУП «НПП ВНИИЭМ»)

Рассматриваются наиболее эффективные устройства отладки программного обеспечения для информационных систем сбора и обработки данных. Одной из самых эффективных систем отладки может быть многоуровневая система, в которой устройством отладки является JTAG, а отладчиком – GDB.
Ключевые слова: *многоуровневая система отладки, логический анализатор, эмулятор, отладочный монитор.*

Выбор инструментов отладки

Отладка программного обеспечения сводится к получению информации о состоянии целевой машины, на которой запускается разрабатываемая программа: о содержимом регистров, памяти, произошедших прерываниях, состоянии периферийной аппаратуры и т. д. Такая информация предоставляется так называемыми устройствами отладки, представляющими собой комбинацию программных и аппаратных средств.

Устройство отладки, как правило, выдает информацию в виде, который не позволяет напрямую соотнести полученное содержимое регистра и содержимое переменной в отлаживаемой программе, особенно если программа написана на одном из языков высокого уровня.

Существуют специальные программы, называемые отладчиками (debuggers), позволяющие выполнять необходимые преобразования, одновременно контролируя способы и устройства получения исходной информации.

При разработке пользовательских программ отладчики чаще всего запускаются прямо на отлаживаемой – целевой машине (target).

При разработке операционных систем или программного обеспечения для встраиваемых систем такой возможности обычно нет, поэтому отладчики запускаются на другой машине, называемой рабочей (host) и связанной с устройством отладки каким-либо информационным каналом.

Существует большое количество отладчиков. Одним из наиболее эффективных считается GNU Debugger (GDB), разработанный по лицензии GNU GPL и представляющий собой приложение с открытым кодом.

Его основные преимущества: бесплатный; имеет возможность модификации; переносим, т. е. может быть использован для разнообразных платформ [1].

Недостатком является то, что GDB использует в качестве устройства отладки отладочный монитор (ОМ), что не позволяет применять его на многих этапах разработки операционной системы (ОС).

Устройства отладки

Различают два метода работы устройств отладки – пассивный и активный.

Пассивная отладка – это методы получения информации о состоянии целевой машины без вмешательства в ее работу.

Активная отладка – это методы получения информации о состоянии машины, основанные на приостановке ее работы, например, добавление точек останова, запуск программных и аппаратных прерываний, пошаговое исполнение команд.

Логические анализаторы. Эти устройства отладки позволяют пассивно отслеживать поток данных программы. Логические анализаторы контролируют шины адреса и данные целевой машины. Они обычно генерируют список выполненных инструкций, иногда уточняя какие данные были затронуты. Это позволяет получать полную информацию о работе старых микроконтроллеров, где выполнение каждой команды вызывает обращение в память. Однако для современных архитектур, использующих процессорный кэш, данный метод не пригоден: команды, расположенные в кэше, не будут считываться из памяти, делая невозможным снятие полной трассы программы [2].

В то же время встречается встроенная в контроллер специальная трассировочная аппаратура, которая тесно связана с ядром микроконтроллера, что позволяет ей записывать каждую исполняемую команду, без необходимости слежения за памятью. Объем получаемой с помощью трассировки информации становится весьма значительным в системах, работающих на высоких частотах, что усложняет как получение этой информации с целевой машины, так и нахождение нужных участков выполненного кода.

Эмулятор ПЗУ. Эмулятор ПЗУ подключается вместо микросхемы постоянной памяти (на кото-

рой обычно записан загрузчик) и подменяет ее на память со свободным доступом (RAM) таким образом, что доступ к ней может иметь не только целевая машина, но и отладчик на рабочей станции. Это сильно упрощает работу отладчика, позволяя ему вставлять в ключевые места вызовы программных прерываний и точки останова. Тестирование и отладка кода значительно ускоряется, так как более не требуется программировать микросхемы памяти на внешнем программаторе. Хотя эмулятор ПЗУ не предоставляет никаких возможностей для получения информации о состоянии машины, но существенно ускоряет отладку по методу «Crash and burn» [3].

Внутрисхемный эмулятор (ВЭ). Внутрисхемный эмулятор (in-circuit emulator, ICE) – это устройство, которым заменяют отлаживаемый микроконтроллер, – специальным его вариантом, включающим в себя устройства отладки или чаще выводы, по которым можно определять его состояние в процессе работы программы. ВЭ подключен к рабочей станции, на которой запущена программа-отладчик. Это позволяет осуществлять как пассивную, так и активную отладку, давая возможность наблюдать за прохождением программы (на целевой машине) и позволяя контролировать ее исполнение, состояние процессора и содержимое памяти.

ВЭ может поддерживать аппаратные точки останова, контролируя шину адреса и переводя систему в состояние отладки, когда на шине обнаруживается нужный адрес. Это позволяет устанавливать точки останова в коде, содержащимся в ROM, без использования эмулятора ROM.

Существенными недостатками ВЭ являются их высокая стоимость и низкая эффективность при работе на высоких частотах, так как данные передаются по параллельному интерфейсу [4].

В современных устройствах установка ВЭ вместо отлаживаемого устройства невозможна из-за отсутствия заменяемых посадочных мест (микроконтроллер или процессор припаивают на материнскую плату в процессе производства).

Отладочный монитор. Отладочный монитор (ОМ) – это термин, определяющий программу, запущенную на целевой машине и соединенную с рабочей станцией, на которой работает отладчик. Однако для запуска ОМ необходимо уже проинициализировать процессор, память и необходимый интерфейсный канал для связи с другой машиной. Это делает невозможным использование ОМ для отладки собственно кода инициализации.

ОМ обычно использует для работы аппаратное прерывание, выдаваемое интерфейсным каналом. Например ОМ, использующий для связи последовательный порт, использует вектор прерывания, выдаваемый последовательным портом. Когда отладчик посылает задание ОМ, генерируется прерывание, передающее ему управление целевой машиной. ОМ ставит точку останова в заданное место, возвращает управление, и когда целевой код доходит до нее, управление передается обратно ОМ, который может считать и послать отладчику всю необходимую ему информацию о состоянии машины в момент прерывания.

Упомянутая инициализация и использование ресурсов целевой машины являются главным минусом ОМ, однако, для использования ему требуется очень небольшая аппаратная поддержка со стороны целевой машины, что делает использование ОМ оправданным в ситуациях, когда другие инструменты отладки недоступны или слишком дороги [3, 4].

Интегрированные отладочные схемы (on-chip debugging, ocd). Отладочные устройства, интегрированные в микросхему, дают все возможности ВЭ при гораздо большей гибкости использования. Вместо того, чтобы заменять микроконтроллер на его специальную, созданную для отладки копию, каждая выпущенная микросхема обладает возможностями самоотладки. Последовательный канал, способный работать на высокой скорости, используется для прямого соединения выводов отладочного блока микроконтроллера с отладчиком на рабочей машине, используя минимум дополнительных выводов из микропроцессора.

Недостатком подобных устройств является необходимость их поддержки производителем микросхем. Однако простота реализации и низкая стоимость внешних устройств привела к высокой распространенности микросхем, поддерживающих подобные способы отладки. Основным стандартом для таких устройств является IEEE 1149-1, называемый также JTAG [3, 5].

JTAG: использование и возможности технологии

JTAG – сокращение от Joint Test Access Group, название группы, сформулировавшей свои предложения по стандарту для тестирования печатных плат и микросхем. Ее предложение было принято Институтом инженеров по электротехнике и радиоэлектронике (Institute of Electrical and Electronics Engineers, IEEE) и известно как IEEE Std. 1149.1-2001 (IEEE Standard Test Access Port and Boundary-Scan Architecture) [5].

В настоящее время большинство фирм-производителей электронных компонентов встраивают в свои изделия средства тестирования и отладки, соответствующие стандарту JTAG. Столь высокая популярность данного стандарта объясняется его гибкостью и возможностью эффективного решения с его помощью задач тестирования, внутрисхемной инициализации и отладки:

– стандартизирования обмена информацией с микросхемой и способа исполнения передаваемых команд;

– создания расширенных версий JTAG-интерфейса (enhanced JTAG interface) по таким направлениям, как программирование устройств, отладка внутрикристальных ресурсов различного назначения, тестирование не только цифровой, но произвольной смеси цифровой и аналоговой части, тестирование свойств цепей соединений;

– обеспечения адресного приема или передачи информации;

– использования возможностей внутрисхемного программирования (In System Programmability, ISP), т. е. настройки программируемого элемента без его физического извлечения из системы;

– использования внутрисхемного конфигурирования (In System Configurability, ISC);

– изменения конфигурации устройства, без вынимания самой платы из компьютера (In Site Programmability, ISP), если микросхема программируемой логики расположена на плате расширения с интерфейсом PCI (поскольку среди внешних выводов таких плат стандартом PCI предусмотрено наличие контактов JTAG);

– расширения для целей внутрисистемного программирования формата файлов BSD (Boundary Scan Description), что позволило программировать интегральные схемы памяти, включенные в состав JTAG-цепочек.

С помощью JTAG также можно упростить процедуры отладки. Для этого используют два способа:

– окружение внутренних блоков интегральной схемы ячейками граничного сканирования;

– помещение внутри отлаживаемой схемы логического анализатора, соединенного с системой, управляющей испытаниями посредством специализированного интерфейса.

Аппаратная поддержка JTAG со стороны микросхемы. Устройство, которое удовлетворяет стандарту JTAG, содержит один регистр команд, несколько регистров данных и TAP-контроллер (Test Access Port Controller), который обеспечивает выполнение функций JTAG-схемотехники (рис. 1).

Технология периферийного сканирования использует ячейки сканирования, соединенные с входами и выходами микросхемы, создавая последовательный сдвиговый регистр. Тестовый вектор вдвигается (или сканируется) ведущим устройством шины TAP (bus master) в этот регистр через TDI и, таким образом задает необходимое значение в ячейках сканирования. Предыдущее значение ячеек выдвигается из TDO и может быть проанализировано ведущим устройством шины.

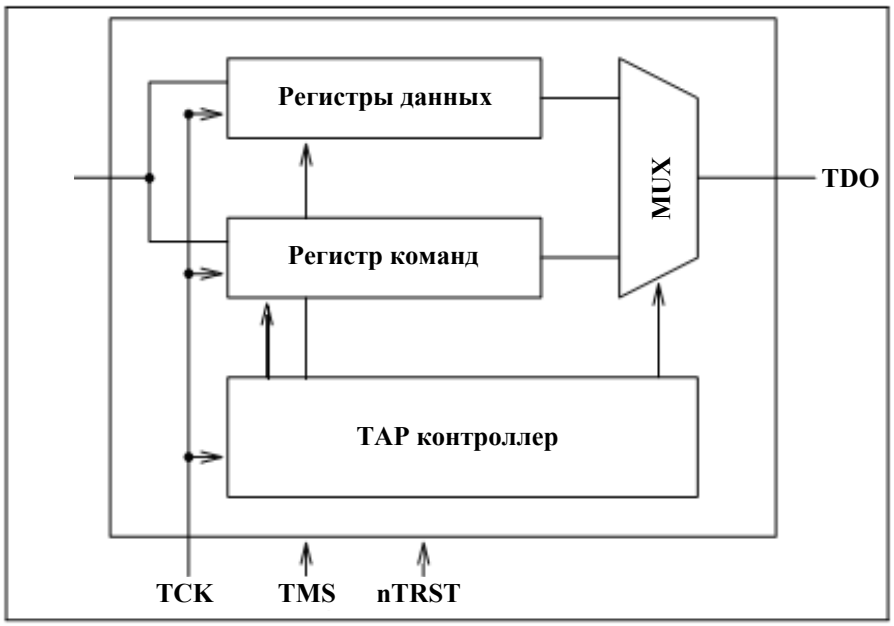

Рис. 1. JTAG-схема, встраиваемая в микросхему

Таблица 1

Интерфейсные сигналы JTAG

Название (аббревиатура)	Описание	Направление для Bus Master
Test Clock (TCK)	Сигнал синхронизации последовательного канала	Выход
Test Mode Select (TMS)	Контроль переходов конечного автомата TAP-контроллера	Выход
Test Data Input (TDI)	Последовательный канал данных, передаваемый тестовый вектор	Выход
Test Data Output (TDO)	Последовательный канал данных, результаты тестового запроса	Вход
Test Reset (nTRST)	Опциональный сигнал мгновенного ресета тестовых схем	Выход

Регистр команд применяется для использования разных функциональных возможностей устройств и должен иметь размерность не менее двух бит. Набор регистров данных создается конкретно под проектируемое устройство, но в соответствии со стандартом требуется как минимум два регистра: однобитовый регистр Bypass и регистр периферийного сканирования.

На рис. 2 приведена электрическая схема соединения двух микросхем с использованием JTAG-схемотехники, соединенных в цепочку вместе с TAP bus master [5].

Интерфейсные сигналы JTAG. В табл. 1 перечислены сигналы, определенные в стандарте JTAG, и их направление, с точки зрения bus master.

Сигнал TCK определяет частоту работы всей цепочки, независимо от частот тестируемых микросхем. При остановке сигнала TCK (0 на линии) все компоненты тестовых схем JTAG гарантированно останутся в своем текущем состоянии.

Сигнал TMS определяет по какому пути пойдет управление в конечном автомате TAP-контроллера (рис. 3). Этот сигнал воспринимается по восходящему фронту сигнала TCK, и ожидается, что он будет изменен bus master по спаду TCK. Конечный автомат TAP-контроллера спроектирован так, что состояние Test-Logic-Reset достигается из любой точки за 5 шагов, если на вход подается TMS = 1. Стандарт устанавливает, чтобы в случае, когда отладочная аппаратура не подключена, TMS всегда равнялся 1. Это позволяет микросхеме нормально функционировать, без помех со стороны JTAG.

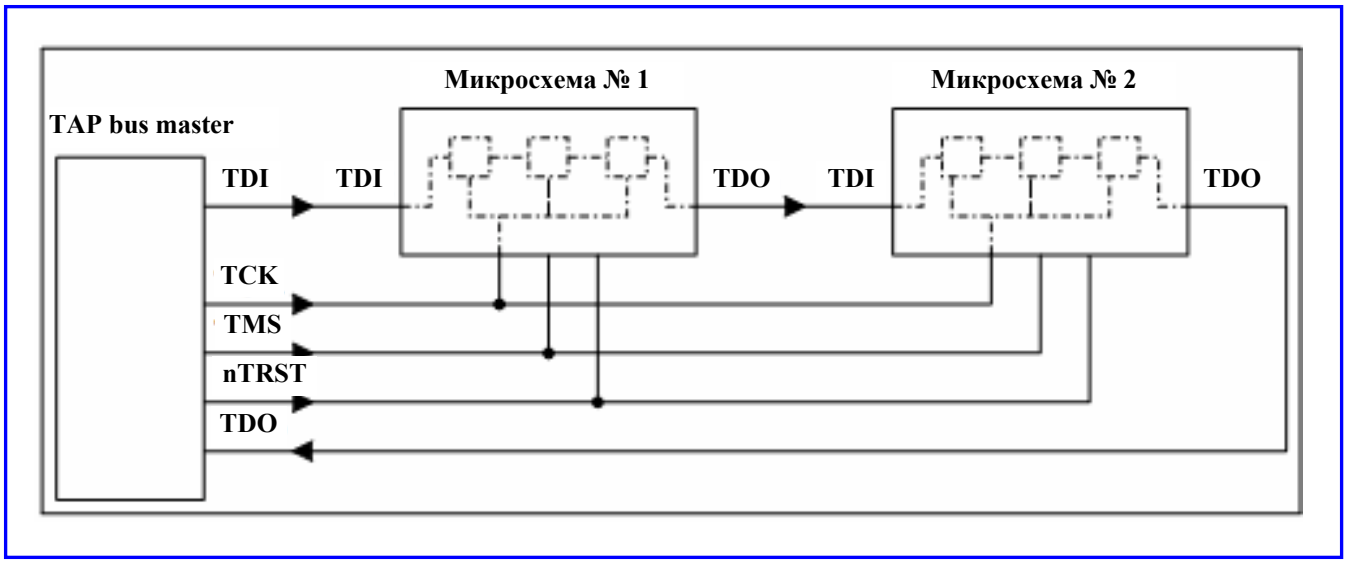

Рис. 2. Шина TAP / Архитектура периферийного сканирования

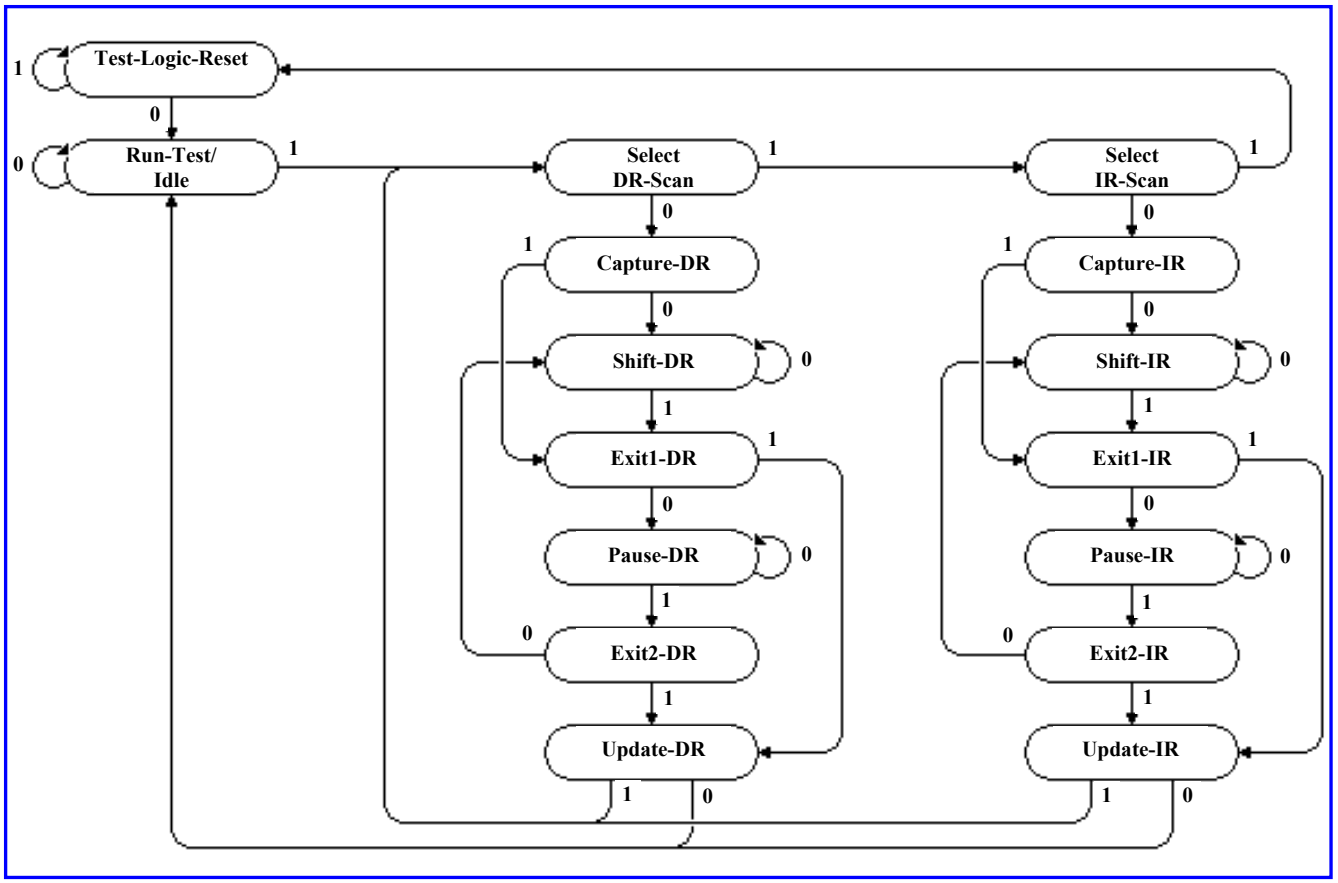

Рис. 3. Схема конечного автомата TAP-контроллера

Сигнал TDI последовательно передает данные на вход TAP-контроллера. Как и TMS, значение сигнала защелкивается по восходящему фронту TCK, и в случае, когда отладочная аппаратура не подключена, TDI всегда равен 1.

Сигнал TDO выдает последовательные данные на выход JTAG-схемы. Значение сигнала должно меняться по спаду TCK, и быть считано bus master по восходящему фронту TCK [5].

Конечный автомат TAP-контроллера. Все действия JTAG управляются через TAP-контроллер, который представляет собой конечный автомат. Он управляется сигналом TMS и совершает переходы с частотой сигнала TCK. Когда начинается процедура тестирования, bus master должен инициализировать все подсоединенные TAP-контроллеры, переведя их в состояние Test-Logic-Reset (TLR). Состояние TLR достигается либо посылкой 0 в nTRST, либо подачей последовательно пяти команд TMS = 1. После этого необходимо подать команду BYPASS либо команду IDCODE. После этого инициализация контроллера завершена.

Получив на вход TMS = 0, автомат переходит в состояние Run-Test/Idle state (RTI). В зависимости от действующей команды в этот момент либо производится тестирование, либо ничего не происходит. Из RTI автомат переходит в Select-DR-Scan (SDS). SDS, Select-IR-Scan (SIS), Exit1-DR/IR (E1D, E1I) и Exit2-DR/IR (E2D, E2I) – это временные состояния, в которых не происходит никаких операций тестирования. Они используются для определения дальнейших путей прохода по конечному автомату. В состоянии Capture-DR (CD) производится подключение выбранного регистра данных между TDI и TDO. В состоянии Capture-IR (CI) производится подключение регистра команд между TDI и TDO. На чтение он всегда содержит b01 в младших битах регистра. После того как автомат переходит в состояние Shift-DR или Shift-IR, TAP bus master устанавливает TMS = 0 и начинает последовательным сдвигом вытеснять содержимое выбранного регистра в TDO, замещая его информацией, поступившей с TDI. Состояние Pause-DR/IR используется для ожидания между операциями сканирования шины. В состоянии Update-DR, по спаду TCK, содержимое регистра сдвига переключается в выбранный этой командой регистр данных. Точно также содержимое регистра сдвига перемещается в регистр ко-

манд, по спаду TCK, в состоянии Update-IR. После перемещения в командный регистр его содержимое становится новой действующей командой [5].

Таблица 2

Обязательные инструкции JTAG

Команда	Код команды	Описание
BYPASS	111111	Помещает однобитный регистр между выводами TDI и TDO, что позволяет данным синхронно пройти через схему JTAG на смежную микросхему.
SAMPLE/ PRELOAD	Определяется производителем	Позволяет сделать моментальный снимок сигналов на выходах микросхемы и исследовать его. Подготовка данных для следующих команд.
EXTEST	Определяется производителем	Позволяет протестировать внешние цепи и связи на уровне платы путем установки тестового набора на выходные выводы микросхемы и получения результата на ее входных выводах.
INTEST	Определяется производителем	Опциональная команда. Позволяет протестировать внутренние сигналы микросхемы.
IDCODE	Определяется производителем	Опциональная команда. Выбирает регистр идентификации в качестве регистра данных. Более никакие регистры не могут быть выбраны.

Инструкции JTAG. Стандарт JTAG требует, чтобы при проектировании устройства, соответствующего этому стандарту, для схемы тестирующей логики были реализованы несколько обязательных инструкций (табл. 2). В реальности производители микросхем не останавливаются только на них, а реализуют еще и множество дополнительных, стандарт это позволяет. Именно дополнительные инструкции позволяют осуществлять отладку программного обеспечения [3].

Заключение

Представленный обзор устройств отладки программного обеспечения для информационных систем сбора и обработки данных позволяет сделать вывод, что одной из самых эффективных систем отладки может быть многоуровневая система, в которой устройством отладки является JTAG, а отладчиком – GDB.

В этой системе сохраняется возможность получения информации в сыром виде для отладки других программ, где нельзя использовать GDB.

Литература

1. Debugging with gdb: the gnu Source-Level Debugger. Free Software Foundation, Inc, 2010.
2. Васильев Н.П., Гороховой В.Р. Микропроцессоры. Аппаратурно-программные средства отладки: учеб. пособие для втузов / Н.П. Васильев, В.Р. Горовой; ред. Л. Н. Преснухина. – М.: Высш. шк., 1984. – 96 с.: ил.
3. Craig A. Haller. The ZEN of BDM [Электронный ресурс] / Macraigor Systems Inc. – Brookline Village, 1996-1997. – Режим доступа: www.macraigor.com/zenofbdm.pdf, свободный.
4. Уильямс Г.Б. Отладка микропроцессорных систем / Г.Б. Уильямс. – М.: Энергоатомиздат, 1988. – 253 с.
5. IEEE Standard 1149.1-2001 «IEEE Standart Test Access Port and Boundary-Scan Architecture» / The Institute of Electrical and Electronics Engineers, Inc. – New York, 2001. – 200 p.

Поступила в редакцию 04.10.2010

***Григорий Евгеньевич Замосковный**, соискатель, т. 8-926-569-88-76.*
***Сергей Геннадьевич Казанцев**, д-р техн. наук, зам. генерального директора – генерального конструктора, т. 366-12-01.*
E-mail: vniiem@vniiem.ru.