# JTAG-тестирование

**Иосиф Каршенбойм (Санкт Петербург)**

**В статье представлен обзор материалов, посвящённых тестированию плат и устройств по интерфейсу JTAG. В обзоре приводятся описания обучающих программ, программ-симуляторов, рабочих инструментов, рассматриваются принцип работы интерфейса и методы тестирования плат.**

## Введение

Ещё не так давно казалось, что DSP-микропроцессор с тактовой частотой 133 МГц имеет предел производительности, а для микроконтроллеров частота 20 МГц была достаточно большой. Сегодня производительность микропроцессоров и микроконтроллеров рванулась к невиданным доселе высотам. Невероятная прежде тактовая частота – 750 МГц – для микроконтроллера уже не кажется слишком высокой, при том что цена на такой микроконтроллер теперь уже не превышает двадцати долларов. Из эксклюзивных подобный продукт уже попадает в категорию стандартных. Этот факт, как ни странно, становится источником проблемы. И дело здесь, конечно, не в самом изделии, а в отношении потребителей данного продукта – разработчиков к самому продукту и к технологии проектирования. Разработчики, которые изначально занимались обработкой высокоскоростных сигналов, имеют необходимый опыт и знают методики отладки своих проектов. Они уже овладели «Курсом черной магии» (известное название книги) и знают, как производить трассировку линий связи и измерять сигналы. А вот на тех, кто ещё вчера использовал в своих проектах обычные 8-битовые «однокристаллки», при переходе на новые микросхемы обрушиваются горы проблем. Мало того что с повышением тактовых частот приходится применять специальные меры при конструировании печатных плат, так поменялись ещё и сами корпуса микросхем. Возникли новые требования: хотите иметь высокие частоты – используйте короткие линии связи; хотите иметь короткие линии связи – применяйте многослойные печатные платы и BGA-корпуса. Но эта проблема имеет, скажем так, физическую природу. Есть ещё одна проблема, которую можно назвать человеческим фактором. Когда-то в 1981 г. автор наблюдал следующую картину взаимоотношений разработчика и программиста. «Стандартный» разработчик приносил свой модуль, который он отлаживал на обычном для того времени кнопочном стенде, и заявлял программисту буквально следующее: «Я проверил запись и чтение из регистров, а то, что у вас это не делается программно, – уже не мое дело»... «Стандартный» программист на это заявлял: «Протокол обмена написан, я его выполняю, – значит, всё должно работать»... В конце концов, через некоторое время появился «универсальный солдат» – программирование стандартных микроконтроллеров стало обычным для «стандартных» разработчиков.

А вот картина сегодняшнего дня. «Стандартный» разработчик оценил производительность и ресурсы новой микросхемы, заложил её в проект. Да вот беда – изучать её программирование ему было некогда, да и сил на такой курс часто не хватает. И дело не в том, что в программировании новых микроконтроллеров есть что-то необычайно сложное, просто разработчику необходимо осваивать много нового материала. Поэтому программирование поручили «стандартному» программисту, и он быстро освоил стартовый набор инструментов. Примеры, драйверы Linux, визуальный отладчик, загрузчик – чего ещё можно пожелать для нормальной работы!? Наконец, плата смонтирована, программы на стартовом наборе отлажены. И тут возникает ситуация, что и 25 лет назад... «Стандартный» разработчик приносит свой модуль и просит «стандартного» программиста «просто помигать светодиодом». Разработчику это нужно для того, чтобы понять, правильно ли припаяны микросхемы. «Стандартный» программист отвечает: «Не могу, потому что у меня там Linux! Дайте мне полностью исправную плату, интерфейс с памятью, сетевой стык для отладки, загрузчик, и вот тогда я напишу программу – приложение к операционной системе, которая будет вам мигать светодиодом, и вы точно узнаете, что процессор припаян правильно»! Знакомая картина? Те из читателей, кто узнал себя в данной зарисовке, уже понимают, что проблема действительно есть, и эта статья предназначена в первую очередь для них. Те же, кто ещё не прошел через аналогичную ситуацию, скорее всего, столкнутся с ней в своем следующем проекте.

Ниже речь пойдёт об одном из способов проверки «железа», причём без программирования микроконтроллера или загрузки тестового проекта в FPGA. В качестве «вводной» представим, что мы хотим сделать: например, систему теленаблюдения или портативный измерительный прибор. А в качестве «подопытного кролика» выберем микроконтроллер BlackFin. Почему именно BlackFin, а не FPGA или ARM? Принципиальной разницы для тестирования здесь нет. Просто если выбрать для проекта планарный корпус, то, возможно, его удастся выполнить, используя старые наработки по тестированию. А вот в случае с микроконтроллером BlackFin новые методики тестирования изделия будут актуальны уже с текущего проекта. Что же касается FPGA, то здесь отличие от микроконтроллера в том, что сама микросхема FPGA по своей сути имеет регулярную структуру, а микроконтроллер имеет множество шин и выводов разного типа. Вот поэтому выбранный микроконтроллер является более наглядным примером, нежели FPGA. Но вместе с тем, учитывая большой интерес читателей к FPGA, некоторые моменты описания будут приводиться и на примере микросхем FPGA.

## Описание ядра микроконтроллера BlackFin

Внешний вид микроконтроллера BlacFin показан на рис. 1.

»Сердцем» микроконтроллера BlackFin является 16-разрядное ядро, выполненное по передовой техно-

логии фирмы Analog Devices. Это ядро, созданное совместными усилиями фирм ADI и Intel, имеет три выдающиеся особенности: высокую производительность, динамическое управление питанием, простоту применения.

### Высокая производительность

Процессоры BlackFin имеют ядро, способное выполнять одновременно две операции умножения с накоплением (MAC), а также обладают функциями микроконтроллера с эффективными RISC-командами и средствами обработки мультимедийной информации. Все эти свойства объединены в одной простой архитектуре с оптимизированным набором команд.

### Динамическое управление питанием

Микроконтроллер BlackFin обладает схемой тактирования с несколькими режимами пониженного потребления (powerdown). Гибкое программное управление даёт возможность ограничиться достаточной в данный момент производительностью процессора посредством динамического изменения напряжения питания и частоты тактовых импульсов.

### Простота применения

Для микроконтроллера BlackFin имеется оптимизированный компилятор в сочетании с архитектурой, рассчитанной на применение языков высокого уровня для разработки ПО. Этим обеспечивается плотность кода, сравнимая с плотностью кода для традиционных микроконтроллеров. Кроме того, у архитектуры есть особенностей, направленных на поддержку эффективного использования операционных систем реального времени. Ядро процессора имеет параллельную конвейерную архитектуру, которая позволяет выполнить максимум операций за один цикл, например:

- по одной команде в каждом из двух арифметико-логических устройств;
- две 32-битные пересылки (две операции чтения или одна операция чтение/запись);
- два обновления счётчика;
- операцию обновления аппаратного цикла.

Блок-схема микроконтроллера приведена на рис. 2.

В настоящее время фирмой-производителем выпущено несколько микропроцессоров семейства BlackFin, которое и далее будет развиваться, так что разработчики смогут выбрать для применения как малобюджетный микроконтроллер, так и более функциональный вариант. Что касается процессорной части микроконтроллера – это тема для отдельных статей. В настоящее время имеется множество информационных ресурсов с описаниями микроконтроллера, примерами применений, образцами проектов. Эту информацию можно найти на сайтах фирм-изготовителей [1], а также на сайтах компаний-дилеров [2]. Как видно из рисунка, микроконтроллер имеет развитую периферию, в состав которой входят практически все типовые микропроцессорные узлы – таймеры, часы реального времени, последовательные и параллельные порты. Следствием развитой периферии и высоких скоростей обработки информации как раз и является корпус BGA с числом выводов 182. Этот корпус mBGA-182 соответствует одному представителю семейства BlackFin.


**Рис. 1. Внешний вид микроконтроллера BlackFin**

## О PCB и не только

Печатные платы тоже претерпели изменения. Термин «разводка печатных плат» постепенно перерос в «дизайн PCB». Кроме трассировки линий связи, конструктор PCB выполняет теперь ещё и тепловой расчёт платы, расчёты импедансов линий связи. Платы становятся многослойными, монтаж – поверхностным. Как только дело доходит до того, чтобы установить на плате компонент, запаиваемый в отверстия, у конструктора начинаются новые проблемы. Ряды отверстий «дырявят» плату сразу во всех слоях, и это приводит к уменьшению быстродействия работы такой платы. А на поверхности платы – свои проблемы. Многорядные разъёмы перегораживают пути для линий связи. Переход линии связи из слоя в слой через переходные отверстия увеличивает ёмкость и ухудшает высокочастотные свойства платы. Всё это, вместе взятое, заставляет разработчика отказываться от технологических контактов, контрольных точек и аналогичных ухищрений в пользу технологичности конструирования платы и повышения быстродействия разрабатываемого устройства. Ну а как же проверить такую плату? Как, например, проверить, если сигнал от одной микросхемы под её корпусом и тут же «ныряет» в 3-й слой, а к другой микросхеме этот сигнал прихо-

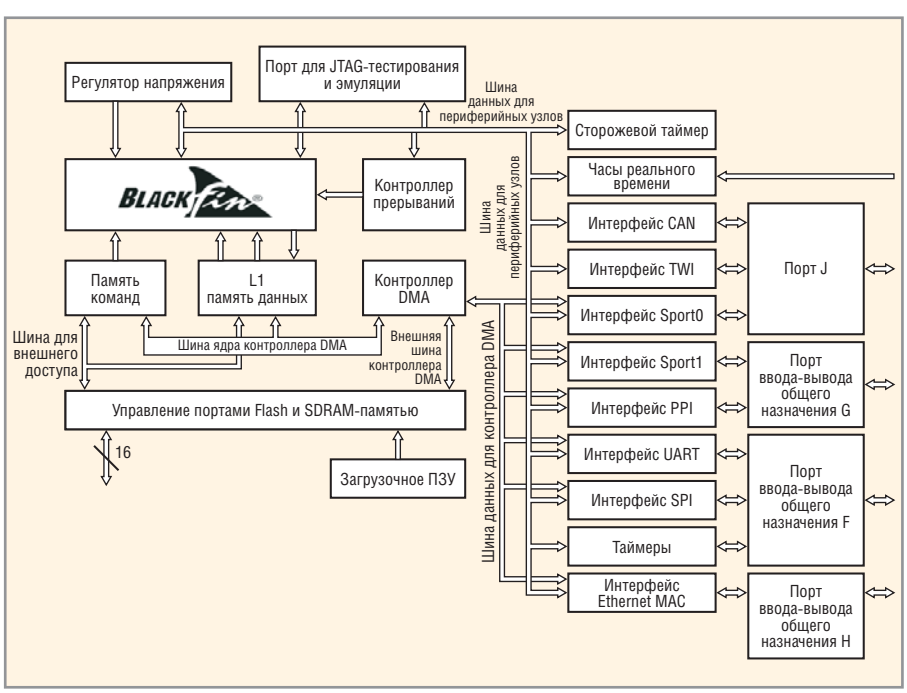


**Рис. 2. Блок-схема микроконтроллера BlackFin**

дит в 6-ом слое. Из 6-го слоя сигнал также выходит под корпусом микросхемы. Если отверстия при переходе из слоя в слой сделаны не сквозные, то сигнал, о котором шла речь, вообще не появляется на поверхность платы. Как убедиться, что связь между микросхемами есть и эта связь выполнена качественно? Поголовный рентгеновский контроль? Это дело дорогое, да и не всем доступное. Что же предлагается? Предлагается технология Boundary Scan (Граничное Сканирование).

Эта технология известна уже давно, но у нас в стране она применяется пока довольно редко, потому что использование высокотехнологичных корпусов BGA ещё не достигло массовости, ведь основное производство в России мелкосерийное или штучное.

## Проверка, проверка и ещё раз проверка

Чем сложнее проверяемое изделие, тем менее достоверны результаты испытаний. Принципы тестирования сложных устройств известны уже очень давно. Сложное изделие или процесс при проверке заменяются на более простые. Сравнение работы сложного и простого процессов или устройств определяет результаты тестирования. Сложный процесс – выполняемая процессором программа – сравнивается при проверке с образцовым процессом, который выполняется в таймере. А точнее, время, затрачиваемое процессором на обработку программы, сравнивается со временем работы таймера. И поскольку таймер гораздо надёжнее и проще, чем процессор, память команд, память данных, выполняемая программа и пр., то и результат получается более-менее достоверный.

Что же необходимо добавить в микросхему, если мы хотим её тестировать, используя описанный принцип? Ответ очевиден – надо добавить то, что будет наиболее простым, а следовательно, наиболее надёжным – простой сдвиговый регистр [13] и блок, управляющий функциями тестирования. Далее под объектом тестирования мы будем понимать микросхему, так как это тот «атом», из которого состоят все устройства. Что касается проверки плат или устройств в целом, то по отношению к JTAG-порту они ведут себя точно так же, как и отдельная микросхема.

## О технологии Boundary Scan. Что такое «Порт JTAG»?

Многие разработчики знакомы с термином «порт JTAG», но нужно признать, что знакомство это часто поверхностное. В большинстве случаев разработчику достаточно было знать, что этот порт используется для загрузки программ или для «прошивки» микросхем. Что касается микроконтроллеров, то с ними порт JTAG используется как порт, к которому подключается отладочное средство. Известно, что по умолчанию сигнал TDO должен быть «лог. 1», во время работы на выводах сигналов TDO, TDI и TMS должны проскакивать «импульсики», а на выводе TCK должна присутствовать тактовая частота. Всё остальное «знали» и выполняли программы, которые работали с портом.

Теперь настало время подробно рассмотреть описание и работу порта JTAG. Основополагающий документ [13], представляет собой стандарт и описывает всё то, что относится к реализации и к работе этого порта. Термины «стандарт IEEE 1149.1» и «JTAG» являются синонимами, и именно так они будут использоваться в этой статье.

Далее будут изложены принципы работы с частью команд, выполняемых портом, а именно тех команд, которые необходимы для тестирования изделий и отладки проекта пользователя. Но сначала необходимо сделать небольшое отступление, чтобы дать читателю общие представления о том, что даёт применение технологии использования порта JTAG.

## И «внутри», и «снаружи». Что JTAG-тестирование даёт пользователю и зачем оно нужно

Для того чтобы организовать массовое производство продукции и иметь при этом минимальные затраты, были выработаны специальные критерии, называемые «тестопригодное проектирование» – DFT (Design For Test). Эти методы и средства проектирования сегодня активно используются многими компаниями, что даёт им возможность снизить стоимость изделия как на этапах разработки и производства, так и на этапах испытаний и эксплуатационного обслуживания. Стандарт IEEE 1149.1 – это стандарт на последовательный интерфейс с четырьмя проводами, который позволяет производить испытания микросхем, плат и устройств.

В основу работы интерфейса положен синхронный последовательный способ передачи данных и команд. Для записи команд применён метод косвенной адресации. Стандарт определяет адресацию и способ работы устройств, подключенных к порту JTAG. Стандарт используется как при работе с корпусированными микросхемами, припаянными к плате, так и для целей внутрисхемного программирования и отладки программ. Эта же технология применяется для проверки на качество припайки микросхем к плате. Стандарт также применяется при проверках межплатного и внутристоечного монтажа плат и блоков. Если же говорить о времени жизни изделия, то использование интерфейса JTAG начинается с момента разработки изделия и продолжается при серийном выпуске. Даже при обслуживании на этапах эксплуатации этот интерфейс используется для тестирования и для изменения конфигурации изделия.

Применение устройств, работающих с этим интерфейсом, началось со стандарта IEEE 1149.1, который применялся главным образом для граничного сканирования. Но при использовании технологии граничного сканирования выяснилось, что стандарт легко приспосабливается и для более широких задач. Поэтому несколько позже был введён стандарт IEEE 1149.4. Принятие этой редакции стандарта было вызвано потребностями внутрисхемного программирования и отладки. Дальнейшее развитие технология граничного сканирования получила в связи с потребностью сканирования аналоговых цепей.

Так как же работает механизм граничного сканирования и как осуществляется тестирование? Вся «хитрость» здесь заключена в том, что между ядром микросхемы и её выводами помещается мультиплексор, который может вместо ядра подключать к выводам сдвиговый регистр, называемый регистром граничного сканирования (Boundary Scan). Каждому конкретному выводу соответствуют так называемые «ячейки». В состав ячейки входит один из триггеров регистра граничного сканирования, мультиплексор данных и буфер,

связывающий вывод микросхемы с мультиплексором и триггером. Ячейки бывают нескольких типов в зависимости от вывода микросхемы и команд интерфейса, для которых этот вывод предназначен.

При передаче по JTAG в микросхему поступают данные от мастера интерфейса. А какие же данные получает пользователь? Сигналы на регистр микросхемы подаются через мультиплексор, который может считывать как состояние выводов ядра микросхемы (режим INTEST), так и данные, поступающие на сдвиговый регистр «извне» (режим EXTEST). Обычно для граничного сканирования используют несколько режимов, обеспечивающих следующие возможности:

- в режиме работы EXTEST обеспечивается возможность установки логических значений на рабочих контактах электронных компонентов, что позволяет частично или полностью проверить внешние цепи, имеющие непосредственное отношение к тестируемому компоненту;
- в режиме работы INTEST обеспечивается возможность установки логических значений внутри микросхемы, то есть на входах её ядра, что позволяет частично или полностью проверить ядро микросхемы;
- режим работы SAMPLE/PRELOAD позволяет тестировать ядро электронного элемента в статическом режиме, устанавливая значения логических уровней на границе его выходных буферов;
- BYPASS – команда обхода, при которой вся цепочка внутри микросхемы «вырождается» только в один триггер. При этом данные со входа передаются на выход с задержкой в один такт частоты синхронизации интерфейса. Такой режим позволяет эффективно использовать возможности последовательного интерфейса при организации длинных последовательно объединённых цепочек.

Существуют и другие режимы работы микросхемы, оговоренные в стандарте. Все эти режимы по возможности будут рассмотрены далее при более подробном описании работы порта JTAG.

## Применение режима граничного сканирования на уровне микросхемы

При помощи граничного сканирования можно, например, при отладке

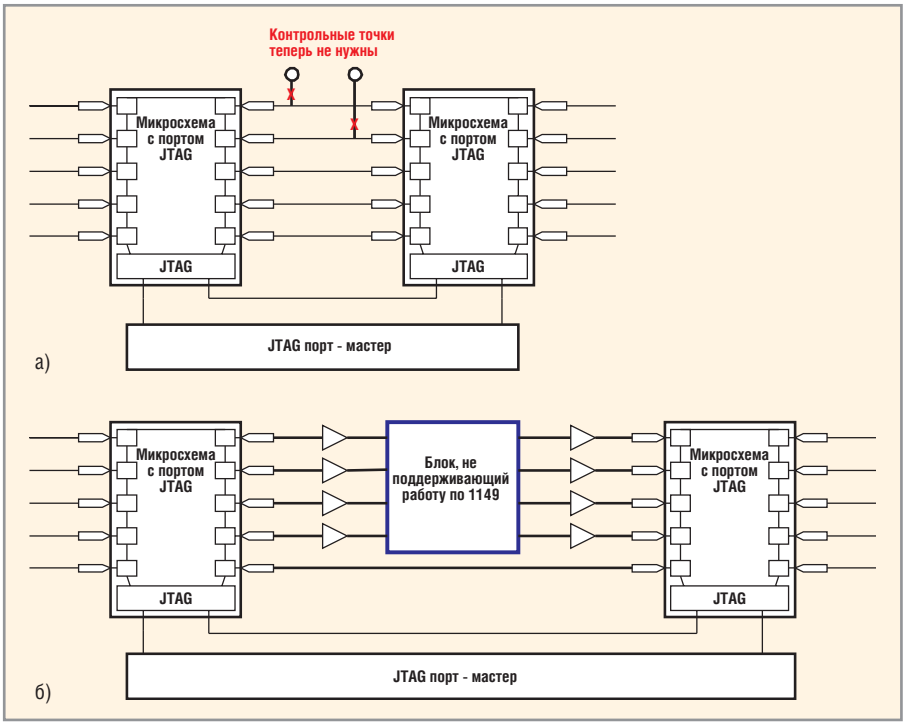


**Рис. 3. Подключение в цепочку сканирования микросхем, имеющих порт JTAG (а); подключение в цепочку сканирования кластера из микросхем, не имеющих порта JTAG (б)**

проекта в FPGA, получать на входах ядра микросхемы те сигналы, которые в реальной ситуации получить трудно. Примером может послужить какой-нибудь сигнал аварии или сбоя при приёме информации. Для того чтобы воспроизвести такой сигнал при реальной работе, необходимо в поток данных вносить специальную «неисправность». Представьте, что мы хотим увидеть то, как будет реагировать наш проект в FPGA на поступающий к нему извне сигнал рассинхронизации цифрового потока данных, называемого в телефонии E1. Аппаратура, воспроизводящая «неисправный» цифровой поток, может оказаться дороже и «дефицитней», чем та, которую мы разрабатываем. Так как же проверить, что FPGA получает сигнал и его обрабатывает? По технологии JTAG, применяя граничное сканирование, надо только подать пару команд и сдвинуть соответствующую последовательность данных в регистр сканирования, и сигнал, который мы хотели увидеть, появится. Конечно, он будет иметь другие временные характеристики, но ведь, как говорится, это «почти даром». Ещё одним важным достоинством технологии граничного сканирования является то, что она позволяет вместо множества пробников использовать только единственный интерфейс с четырьмя проводами – JTAG.

## Применение режима граничного сканирования на уровне платы

Для того чтобы проводить проверку на уровне платы, компоненты, имеющие интерфейс граничного сканирования, должны быть соединены в последовательную цепочку, начинающуюся от вывода TDI и заканчивающуюся на выводе TDO. Платы, составленные из компонентов, которые на 100% соответствуют 1149.1, могут быть проверены векторным тестовым набором, сгенерированным программно. Такой класс программ называется ATPG (Automatic Test Program Generation). В случае применения микросхем, имеющих порт JTAG, ATPG обеспечивает 100-% охват тестируемого изделия при поиске ошибок, и при этом отпадает необходимость в применении контрольных гнёзд для проверки наличия сигналов (см. рис 3а). Многочисленные аппаратные тестеры, проверяющие платы по интерфейсу JTAG, обладают разной производительностью и, соответственно, разной ценой. Разработчик устройств всегда может выбрать то тестирующее оборудование, которое при его условиях производства даст наибольшую экономию усилий и устранит необходимость проведения дорогих испытаний, проводимых вручную. Если часть микросхем

на плате не имеет портов JTAG, такие микросхемы при тестировании могут быть выделены в отдельные кластеры, на которые подаются тестовые воздействия, и с которых через микросхемы, имеющие порты JTAG, получают результаты тестирования (см. рис. 3б). В последнем случае тестовые воздействия составляются таким образом, чтобы учесть характеристики кластера, не охваченного цепочкой граничного сканирования и считающегося «чёрным ящиком».

Фактически даже один компонент, находящийся на плате и имеющий интерфейс граничного сканирования, значительно упростит проведение испытаний, особенно если он представляет собой сложную микросхему, например, микропроцессор, FPGA или специализированную интегральную схему. Положительный эффект становится наиболее заметным в случае применения микросхем в корпусах BGA с большим количеством входов и выходов.

Как было сказано выше, для разработки тестов обычно используют программные инструменты. Поскольку программы проверяют только связи между компонентами, им не надо «знать» начинку микросхем. То есть вся внутренняя логика микросхемы не участвует в создании тестовых наборов. Для таких программ совершенно нет различий в том, что за выводы имеет микропроцессор. Важно только, может ли конкретный выход быть входом, выходом, входом-выходом и можно ли его переключать из состояния «приём» в состояние «передача». Далее состояние такого вывода приписывается к его граничной ячейке. Таким образом, сигнал с каждого входа устройства может быть прочитан его граничной ячейкой, и, соответственно, сигнал на каждый выход устройства может быть выдан из его граничной ячейки.

Для проведения испытаний применяются специальные языки описания. Стандарт IEEE 1149.1 определяет синтаксис языка описания граничного сканирования – BSDL. Язык описывает выводы ИС и порядок работы встроенной в неё испытательной схемы, например, граничный регистр, дополнительные регистры, набор команд и коды операций. Файлы BSDL поставляются изготовителями устройств, соответствующих IEEE 1149.1.

Таким образом, на уровне платы производство испытаний на граничное сканирование может быть полностью автоматизировано. Для ATPG требуется только список цепей платы (NET-лист) и модели BSDL для каждого из находящихся на плате устройств, соответствующих IEEE 1149.1.

## Применение граничного сканирования на уровне многоплатной системы

На уровне многоплатной системы также могут быть организованы одна или несколько цепочек граничного сканирования. Цепочки граничного сканирования, входящие в систему плат, желательно подключать к основной плате так, чтобы каждая плата проверялась по отдельной цепи граничного сканирования. В этом случае ошибки в цепи граничного сканирования одной платы не будут влиять на проверку других плат и всей системы в целом.

То же самое можно сказать и о самотестировании для встроенных систем. Если у проверяемой системы нет встроенного порта для проведения граничного сканирования, изготовителю оборудования необходимо выполнять трудоёмкие испытания на функционирование. Для этого надо разрабатывать специальные тесты для каждого режима испытаний. Преимущество проекта, поддерживающего режим граничного сканирования, состоит в возможности применить те же самые тесты, которые уже использовались для испытаний плат и узлов, составляющих данное изделие.

## Граничное сканирование на разных этапах жизни изделия

Использование технологии граничного сканирования в микросхеме, на плате или в устройстве добавляет стоимость и увеличивает время разработки проекта. Однако эти затраты легко окупаются при проведении автоматического тестирования, которое обеспечивается на каждой стадии цикла жизни изделия. То, что было первоначально разработано как производственный испытательный инструмент, используется до начала производства, во время серийного производства и после производства, то есть на этапе эксплуатации.

Кроме непосредственно граничного тестирования, проектировщики используют технологию JTAG для того, чтобы производить самотестирование (BIST) (в тех компонентах, где оно реализовано) и/или загружать внутренние значения в регистры устройства или программировать микросхемы ПЗУ. Тесты, которые были разработаны и использованы на этапе проектирования, могут быть переданы производству, для того чтобы обеспечить дополнительное снижение стоимости и времени на проверку изделий при выходном контроле.

Основные положительные эффекты от применения технологии JTAG в производственной фазе – экономия времени при разработке испытательных тестов, улучшенный «охват» тестируемого изделия при поиске ошибки и диагностировании и улучшенная производительность испытаний при одновременном уменьшении времени испытания.

Применение граничного сканирования при эксплуатации изделия также даёт определённый положительный эффект. Отказы при эксплуатации часто происходят из-за структурных отказов, которые вызываются повышенной температурой, влажностью, вибрацией. Используя граничное сканирование, техники имеют возможность быстро проверить изделие на структурные ошибки вплоть до уровня компонентов без трудоёмкого исследования или возвращения платы изготовителю на завод. Устранение трудоёмких тестов позволяет производить более эффективную диагностику и ремонт, что уменьшает стоимость и время простоя.

## Несколько слов о программах тестирования изделий

В качестве примера программы тестирования можно привести пакет программ фирмы Corelis [14]. Эти программы позволяют автоматически сформировать файл тестовых воздействий, проанализировать, насколько полно «покрывается» тестированием проверяемое изделие, выполнить тестовую программу и сообщить пользователю о результатах тестирования. Кроме автоматического режима тестирования существует

ручной интерактивный режим. Результаты тестирования могут быть представлены как в виде таблиц с сообщениями об ошибках, так и более наглядно – в виде фрагмента PCB, с указанием возможной точки проявления неисправности. Существует режим, когда тестовые воздействия визуализируются, как на экране логического анализатора.

Тесты для отдельных плат могут быть объединены в тесты для всего проверяемого изделия. Кроме программ тестирования в пакет входят программы In-System Programming. Они позволяют программировать Flash-микросхемы непосредственно в устройстве.

## Интерфейс JTAG – это очень просто!

Что же такое интерфейс с портом JTAG?

*Интерфейс – это совокупность названий сигналов, их логических и электрических взаимосвязей и конструкторская реализация устройства для этих сигналов.*

Что касается интерфейса с портом JTAG, то для двух последних составляющих интерфейса в приведённом выше определении всё представляется довольно просто. Поскольку сам JTAG не является основным интерфейсом аппаратуры, то производители не придерживаются жёстких правил при его реализации, как это, например, имеет место для интерфейсов PCI. Электрические характеристики сигналов обычно выбираются так, чтобы соответствовать стандартам TTL, LVTTL, CMOS, LVCMOS и др. Что же касается конструкции разъёмов, то тут царит полный хаос. Каждый из производителей микросхем предлагает свой вариант разъёма для связи с портом. На сайте фирмы Амонтек можно найти документ, в котором приведено расположение сигналов порта JTAG для некоторых наиболее распространённых абонентов порта и различные варианты конструкции разъёмов

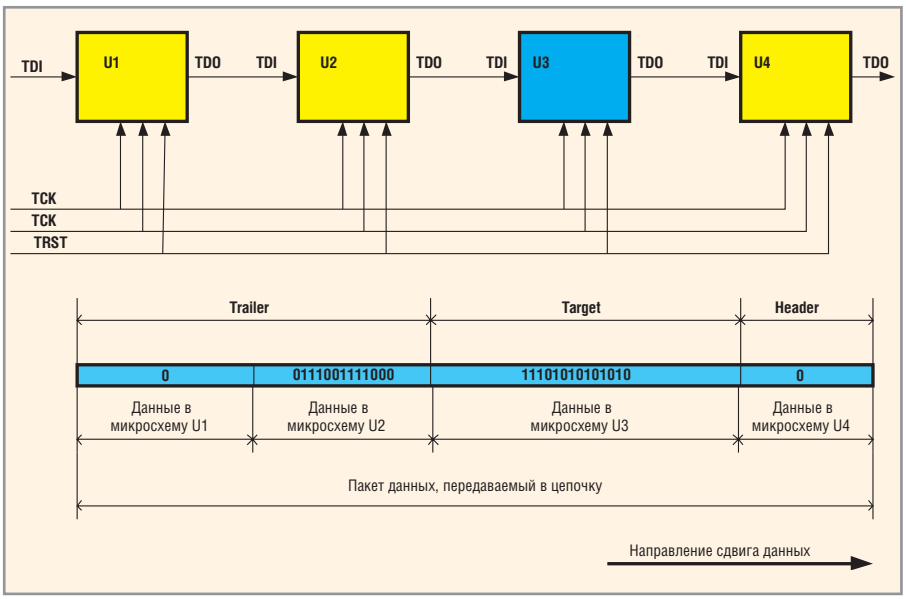


**Рис. 4. Соединение четырёх абонентов интерфейса в цепочку, подключенную к одному порту мастера**

[16]. Таким образом, остаётся описать только названия и логические соотношения сигналов при работе порта. Автор заранее просит прощения у тех читателей, которые знакомы с данным интерфейсом. Они могут смело пропустить последующую главу.

Весь набор сигналов интерфейса JTAG представлен в табл. 1. К сигналам интерфейса относятся:

- сигнал данных на передачу – TDO;
- сигнал данных на приём – TDI;
- тактовая частота – TCK;
- сигнал управления – TMS.

Есть только один мастер интерфейса, который полностью контролирует работу всех абонентов, подключенных к цепочке. И так как интерфейс JTAG – это интерфейс с косвенной адресацией, то все тестовые воздействия выдаются двумя циклами – сначала записывается адрес и только потом данные. Причём можно выдавать несколько команд сканирования данных по одному адресу.

Как указывалось выше, интерфейс JTAG – синхронный. Сигналы данных и управления принимаются по переднему фронту сигнала TCK. Данные передаются младшим значащим битом «вперед» и только в течение состояний автомата управления TAP (см. диаграммы работы TAP контроллера) – Shift-IR или Shift-DR. Выходные данные выставляются под задний фронт сигнала TCK.

Как мы теперь уже знаем, JTAG применяется как универсальный интерфейс, который можно использовать для тестирования не только отдельных микросхем, но и плат, и даже целых устройств, состоящих из нескольких плат. На рис. 4 показано соединение четырёх абонентов интерфейса в цепочку, подключенную к одному порту мастера. Абоненты интерфейса включаются следующим образом:

- сигналы управления интерфейсом TMS, TCK и опциональные сигналы TRST и STRST включаются ко всем абонентам параллельно;
- сигналы данных передаются последовательно от порта к первому абоненту, от него к следующему, и т.д. Последний абонент возвращает данные в порт.

Поскольку сигналы управления подаются параллельно, у нас есть возможность управлять микросхемами, но только всеми одновремен-

**Таблица 1. Основные сигналы интерфейса**

| Название сигнала | Описание |
|---|---|
| TDO (Test Data Output) | Выход последовательных данных. Этот сигнал используется для передачи данных. Этот выход должен находиться в третьем состоянии, когда данные не передаются на этот выход. |
| TDI (Test Data Input) | Вход последовательных данных. Этот сигнал используется для приёма данных. По умолчанию на этом порте должна присутствовать 1 |
| TCK (Test Clock Input) | Вход синхроимпульсов, формируемых внешним хостом |
| TMS (Test Mode Select Input) | Вход выбора режима. Этот сигнал управляет режимом работы TAP. По умолчанию на этом порте должна присутствовать 1 |
| Test Reset (TRST*) | Дополнительный сигнал порта, предназначенный для асинхронного сброса TAP контроллера. По умолчанию равен 1. Активный уровень – 0 |

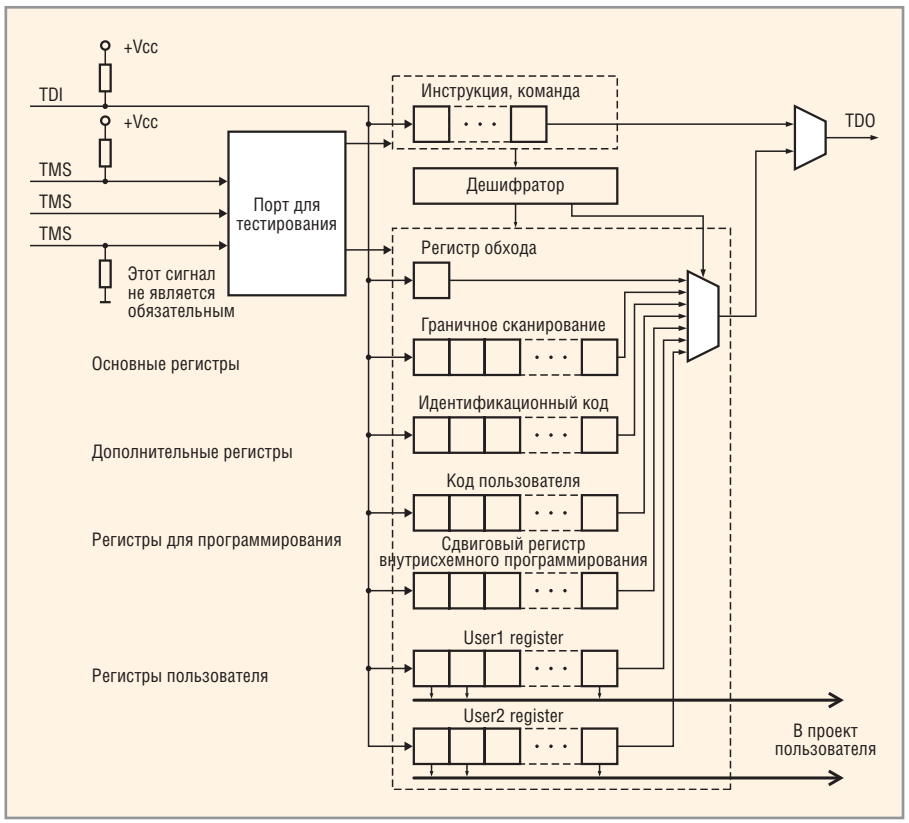


**Рис. 5. Блок-схема TAP**

но! Чтобы к нужной микросхеме в нужный момент пришли данные, необходимо учитывать её место в цепочке. Для этого в специализированном языке SVF, применяющимся для описания процедур работы с портом, имеются следующие термины – Header (заголовок), Target (цель), Trailer (хвостовик). Представим, что мы хотим изменить режим работы микросхемы Target. Для этого сначала в испытуемую цепь «проталкиваются» данные, относящиеся к Header, затем к Target и, наконец, к Trailer (см. рис. 4).

Действительно, все микросхемы в цепочке одновременно выполняют команду, например, «принять код команды». Но вот сами коды команды передаются по последовательной цепочке, и эти коды могут быть разными для разных абонентов. На рис. 4 зелёным цветом показан пакет данных, передаваемых в JTAG-цепочку. На этапе передачи команды в такой кадр выстраиваются команды, которые будут выполнять микросхемы. Соответственно, получив разные команды, абоненты выполняют разные действия. После этого на этапе передачи данных в такой же кадр собираются данные, передаваемые в цепочку.

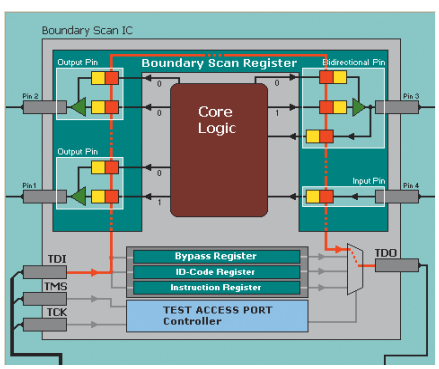


**Рис. 6 Структура узлов JTAG-сканирования**

## Пора «заглянуть внутрь» микросхемы

Вот теперь настал момент «заглянуть внутрь» микросхемы. Как работает интерфейс? Как обрабатываются данные? Чем данные отличаются от команды? Как принятая команда поступает на исполнение?

Поскольку используется интерфейс с последовательной передачей данных, то, без сомнения, в микросхеме должны присутствовать входной и выходной сдвиговые регистры. Также должен быть автомат, который управляет режимом работы TAP, и выходной мультиплексор, коммутирующий на выход данные с регистра, выбранного для текущего режима работы. Структура интерфейса представлена на рис. 5.

На рис. 6 приведён фрагмент программы Boundary Scan Coach, а именно типовая структура узлов JTAG-сканирования, встроенных в микросхему. На этом и на других рисунках, представляющих окна программы Boundary Scan Coach, различные функциональные узлы выделены:

- функциональная часть микросхемы Core Logic – коричневым цветом;
- JTAG-регистры – зеленым цветом;
- выводы микросхемы – серым цветом;
- контроллер JTAG-порта – голубым цветом;
- путь прохождения данных по регистру граничного сканирования – красным цветом.

## Переходы, выполняемые при работе с командами

Последовательный интерфейс JTAG предназначен для записи и чтения данных и команд управления от хост-процессора. Команды управления интерфейсом записываются в регистр косвенного адреса. Данные записываются или считываются из того регистра, на который указывает регистр косвенного адреса. Итак, существуют раздельные циклы записи адреса и чтения-записи данных.

Рассмотрим режимы работы управляющего автомата. Автомат имеет 16 состояний. Управление интерфейсом осуществляется путём воздействия на автомат контроллера JTAG-порта. Сигналы управления от хоста приходят на контроллер JTAG-порта и поступают на вход TMS. Переходы автомата выполняются под управлением сигнала TMS по переднему фронту сигнала TCK. Чтение и запись данных производится одновременно. Данные в микросхему выдаются на вывод TDI, а принимаются из микросхемы с вывода TDO.

Наилучшим образом управляющий автомат JTAG-сканирования представлен в описании работы микросхем фирмы Altera [10]. Диаграмма переходов автомата приведена на рис. 7, где рядом со стрелками указано значение сигнала TMS, при котором происходит соответствующий переход.

Ниже описание работы автомата будет сделано только для тех переходов, которые необходимы для понимания работы интерфейса. Кроме этих переходов, автомат может выполнять и другие переходы, но при-

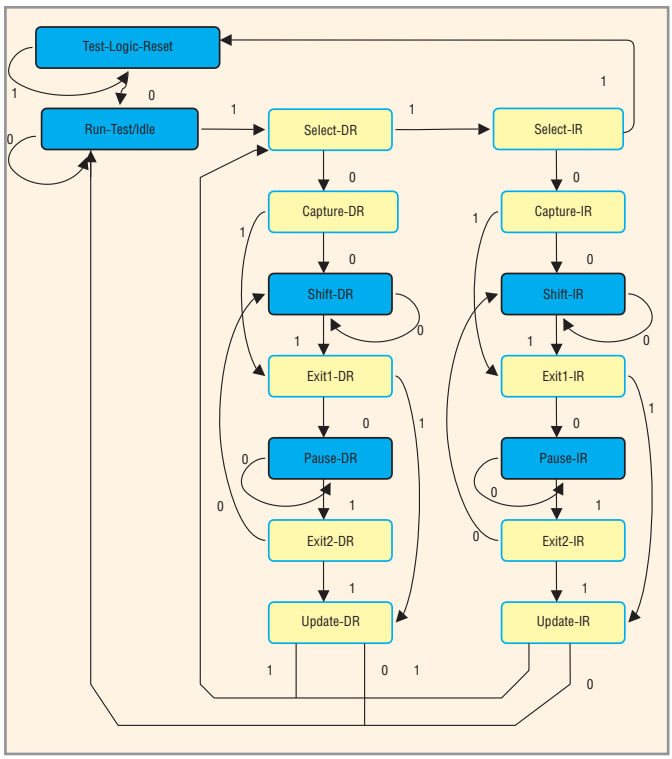


**Рис. 7. Диаграмма переходов статического автомата, управляющего режимами работы TAP**

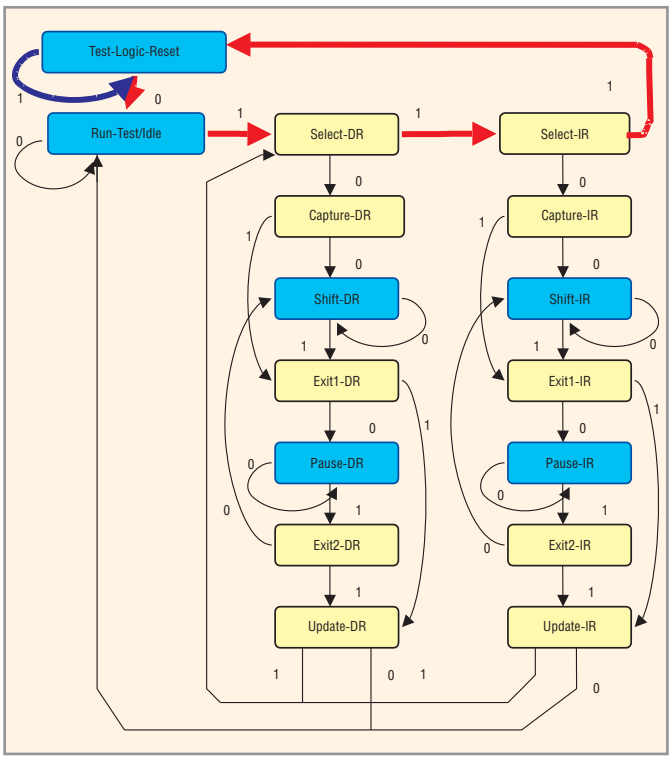


**Рис. 8. Диаграмма переходов при подаче на вход TMS сигнала «лог 1»**

ведённых здесь сведений будет достаточно для того, чтобы читатель смог сам проанализировать действия управляющего автомата.

Состояния диаграммы переходов:

- Test-Logic-Reset – исходное состояние;
- Run-Test/Idle – переходное состояние контроллера при выполнении тестов или ожидании следующей команды;
- Select-IR, Select-DR – состояние, после которого будет производиться тестирование команд, данных;
- Capture-IR, Capture-DR – состояние приёма команд, данных;
- Shift-IR, Shift-DR – состояние сдвига команд, данных;
- Exit1-IR, Exit2-IR – выход из режима работы с командами;
- Exit1-DR, Exit2-DR – выход из режима работы с данными;
- Pause-IR, Pause-DR – состояние паузы;
- Update-IR, Update-DR – состояние перезаписи данных в выходные регистры.

## Диаграмма переходов по IEEE STD 1149.1

Контроллер TAP сразу после включения находится в состоянии Test_Logic Reset (исходное состояние). До тех пор, пока сигнал TMS имеет значение «лог. 1» (по умолчанию), состояние автомата остается неизменным. Обычно в этом случае по умолчанию выбран регистр «Идентификация устройства» или «Регистр обхода». Сигнал сброса TRST не является обязательным, поэтому для сброса автомата в исходное состояние применяют следующую процедуру. Необходимо подать на вход TMS сигнал высокого уровня и удерживать его не менее 5 тактов частоты TCK. Именно поэтому на входе TMS устанавливают опорный резистор. Если сигнал TMS будет установлен хостом в низкий уровень, то автомат перейдёт к состоянию Run_Test/Idle (активное состояни, в котором ничего не происходит). Обычно из этого состояния можно перейти в состояние Select IR_Scan, для того чтобы загрузить в контроллер новую инструкцию. Но если на вход сигнала TMS подействует не сигнал, подаваемый от хоста, а помеха низкого уровня, то, как и в предыдущем случае, автомат перейдёт в состояние Run_Test/Idle. Если же кратковременная помеха (длительностью не более одного периода синхрочастоты) прекратится, то автомат через три такта снова вернёт-

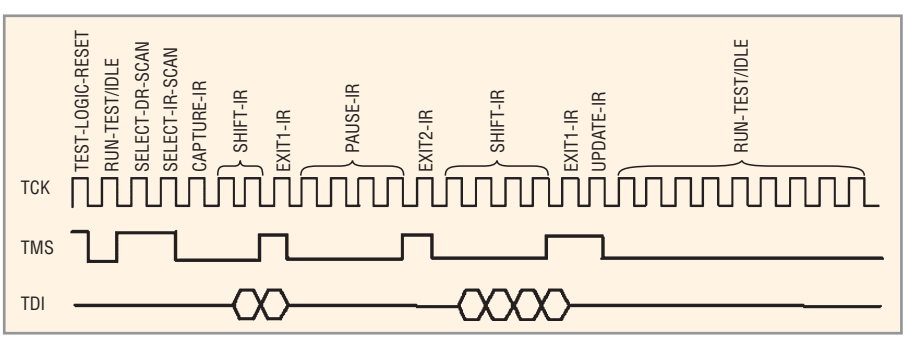


**Рис. 9. Диаграмма сигналов при выполнении переходов для загрузки команды**

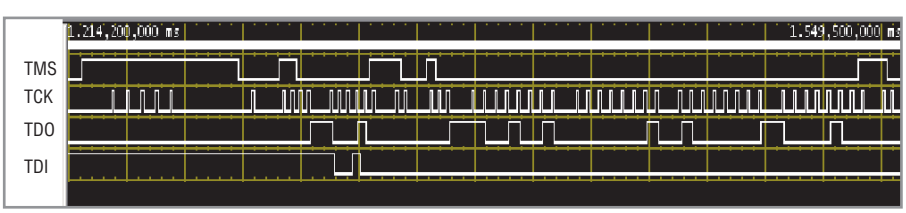


**Рис.10. Диаграмма «JTAG-ребус»**

ся в исходное состояние – Test-Logic-Reset. Диаграмма переходов при подаче на вход TMS сигнала «лог. 1», соответствующих устойчивому состоянию сброса – Test-Logic-Reset, представлена на рис. 8. Переход выделен синим цветом. Красным цветом выделены переходы при кратковременной помехе на входе. Исходя из этого, разработчик может определить максимальную рабочую частоту работы порта. Что же касается предельной тактовой частоты, то она также может быть указана в BSDL-файле.

Чтобы загрузить в контроллер новую команду, надо из состояния Run_Test/Idle перевести автомат в состояние Select_IR, Capture_IR, Shift_IR. Затем необходимо «продвинуть» в цепочку данных новую команду, а потом перевести автомат через состояния Exit1_IR, Update_IR и снова в Run_Test/Idle. Диаграмма переходов для этого показана на рис. 8 и выделена красным цветом.

Необходимо обратить внимание, что команда передаётся в состоянии автомата Shift_IR, и при этом на входе TMS присутствует сигнал низкого уровня. Для того чтобы выйти из этого состояния и перейти к следующему, необходимо на входе TMS установить сигнал высокого уровня. Сигнал высокого уровня должен быть выдан с последним битом команды, передаваемым от TDI к TDO. То же самое условие необходимо будет соблюдать во всех операциях с портом JTAG при работе автомата в состояниях Shift_IR и Shift_DR. На рис. 9 показана диаграмма сигналов при выполнении переходов для загрузки команды. На экране логического анализатора это выглядит так, как представлено на рис. 10. Без программных средств расшифровывать такие диаграммы довольно затруднительно, но при определённых знаниях возможно. Назовём нашу диаграмму «JTAG-ребус». Забегая вперед, дадим подсказку. Для того чтобы успешно «расшифровать» эту диаграмму, нужно посмотреть на те моменты времени, где присутствует сигнал TMS = 1. Это означает изменение режимов. Ответ на «JTAG-ребус» будет предоставлен читателям ниже.

*Продолжение следует*

## Литература


1. http://www.analog.com.
2. www.eltech.spb.ru.
3. Boundary Scan Coach. GOEPEL Electronic. http://www.goepel.com.
4. http://www.pld.ttu.ee/applets.
5. http://www.universalscan.com.
6. *Платунов А.Е., Постников Н.П., Чистяков А.Г.* Механизмы граничного сканирования в неоднородных микропроцессорных системах. Chip News. http://lmt.cs.ifmo.ru/article_chip_news.html.
7. *Рустинов В., Городецкий А.* Разделяй и властвуй – принцип граничного сканирования. Chip News. http://chipnews.gaw.ru/html.cgi/arhiv/01_06/stat-3.htm.
8. Каршенбойм И. Виртуальные кнопки и светодиоды, или Неизвестное обо всём известном JTAG_сканировании. Компоненты и технологии. 2005. № 6.
9. http://www.national.com/appinfo/scan/index.html.
10. IEEE 1149.1 (JTAG) Boundary-Scan Testing for Stratix II Devices. Altera. Chapter 9. www.altera.com.
11. Kuznetsov D. JTAG Boundary-Scan Test – introduction. http://www.orc.ru/~dkuzn/j_intro.htm.
12. http://jtagtools.sourceforge.net/download.html.
13. IEEE Standard Test Access Port and Boundary-Scan Architecture. IEEE Std 1149.1-2001.
14. Boundary-Scan Test and In-System Programming Software. Corelis. http://www.corelis.com/products/Test_Software.htm.
15. Serial Vector Format Specification. ASSET InterTech. Texas Instruments. www.asset-intertech.com/support/svf.pdf.
16. www.amontec.com. JTAG Interface: Common Pinouts amt_ann003 (v1.1). Application Note
17. EIA/JEP106, JEDEC Publication 106, Standard Manufacturer's Identification Code.
18. *Каршенбойм И.* Микропроцессор своими руками/4. Как отладить встроенный в FPGA микроконтроллер? Компоненты и технологии. 2006. № 11.