Продолжение. Начало в № 9`2006

Микропроцессор своими руками-4.
Как отладить встроенный в FPGA микроконтроллер?

Иосиф КАРШЕНБОЙМ
iosifk@narod.ru

Что такое «порт JTAG»? Терминология: порт JTAG, 1149 и т. д.

Большая часть разработчиков знакома с таким термином — «порт JTAG». Но нужно признаться, что знакомство это у большинства из них — поверхностное. Да, зачастую разработчику достаточно было знать, что этот порт используется для загрузки программ или для «прошивки» микросхем. Что касается микроконтроллеров, то для них данный порт используется для подключения отладочного средства. Известно, что по умолчанию сигнал TDO должен быть подперт в «1», во время работы на сигналах TDO, TDI и TMS проскакивают «импульсики», а на TCK — тактовая частота. Кстати, на сайте фирмы «Амонтек» читатель может найти документ, в котором приведено расположение на разъеме сигналов порта JTAG для некоторых, наиболее распространенных его абонентов [9].

Все остальное «знали» и выполняли программы, которые работали с портом. Кроме «больших» программных инструментов, таких как Quartus или ISE, в которых функция работы с JTAG встроена и закрыта от пользователя, существует множество программ, которые работают с JTAG-портом. Начиная от обучающих программ, описанных в [5], и кончая открытыми проектами, например [10]. В Интернете существует довольно много описаний работы JTAG, в том числе и на русском языке.

Читатель может скачать обучающие программы, описанные в [5], и самостоятельно провести изучение работы порта в интерактивном режиме. Правда, программы, описанные в [5], ориентированы, в основном, на граничное сканирование (Boundary Scan). Но работа в режиме граничного сканирования близка к работе при отладке встроенного проекта, и, тем не менее, у этих двух режимов работы есть некоторые отличия. Поэтому читателю понадобятся более подробные сведения о работе порта JTAG в различных режимах его работы.

Основополагающий документ [8] представляет собой стандарт и описывает все то, что относится к реализации и работе этого порта. Главное, о чем нужно сказать здесь, так это то, что термины «стандарт IEEE 1149.1» и JTAG являются синонимами и именно так они будут использоваться в данной статье. Далее мы рассмотрим основные моменты, необходимые для того, чтобы понять, как работает порт и как с его помощью отлаживать свои проекты.

И, поскольку целью данной статьи является отладка проектов пользователя, то придется напомнить читателю основные моменты, которые необходимо знать при работе с портом JTAG. Здесь же будут коротко изложены только основные принципы и некоторые практические замечания. Автор приведет принципы работы только с частью команд, выполняемых портом, а именно тех из них, что необходимы для отладки проекта пользователя. Но сначала — небольшое отступление, общие представления о том, что дает применение технологии использования порта JTAG.

Тестирование и еще раз тестирование. И «внутри», и «снаружи». Что это дает пользователю и зачем это 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 дает им возможность снизить полную стоимость изделия как на этапах разработки, производства, так и на этапах испытаний и эксплуатационного обслуживания. Применение интерфейса JTAG для целей тестирования и программирования компонентов, установленных на плате пользователя, дает определенные преимущества на всех этапах жизни изделия. IEEE 1149.1 — это стандарт на последовательный интерфейс с 4 проводами, который позволяет производить испытания микросхем, плат и устройств. Чтобы описать те преимущества, которые можно получить при применении технологии по IEEE 1149.1, придется немного забежать вперед и дать краткие значения некоторых терминов, которые будут использоваться для описаний преимуществ, получаемых при работе с портом JTAG. Потом, уважаемый читатель, все то, что непосредственно относится к работе интерфейса, будет рассмотрено более подробно.

В основу работы интерфейса положен синхронный последовательный способ передачи данных и команд. Для записи команд применен метод косвенной адресации. Стандарт определяет адресацию и способ работы устройств, подключенных к порту JTAG. Он применяется для работы с такими изделиями, как отдельная микросхема, которую необходимо тестировать при изготовлении, еще до того, как она будет разварена в кристалл. Также он используется и при работе с корпусированными микросхемами, припаянными к плате, для целей внутрисхемного программирования и отладки программ. Эта технология необходима и для проверки на качество припаивания микросхем к плате. И еще этот стандарт применяется к блокам и устройствам «в целом», для того, чтобы проверить межплатный и внутристоечный монтаж. Если говорить о времени жизни изделия, то применение интерфейса JTAG начинается с момента разработки изделия и потом продолжается при серийном выпуске. И даже при обслуживании изделия на этапах эксплуатации этот интерфейс используется для тестирования и замены конфигурации изделия.

Применение устройств, работающих с этим интерфейсом, началось со стандарта IEEE 1149.1, который использовался главным образом для граничного сканирования. Но при этом выяснилось, что стандарт легко приспосабливается и для более широких задач. Поэтому несколько позже был принят стандарт IEEE 1149.4. Появление этой редакции стандарта было вызвано потребностями внутрисхемного программирования и отладки. Дальнейшее развитие технология граничного сканирования получила в связи с расширением потребности на сканирование аналоговых цепей. Расширение сфер применения технологии JTAG показано на рис. 12.

Использование интерфейса JTAG для целей программирования и отладки микроконтроллеров довольно хорошо описано в различных публикациях. Поскольку в этой статье идет речь об отладке проектов в FPGA, то здесь мы очень кратко остановимся только

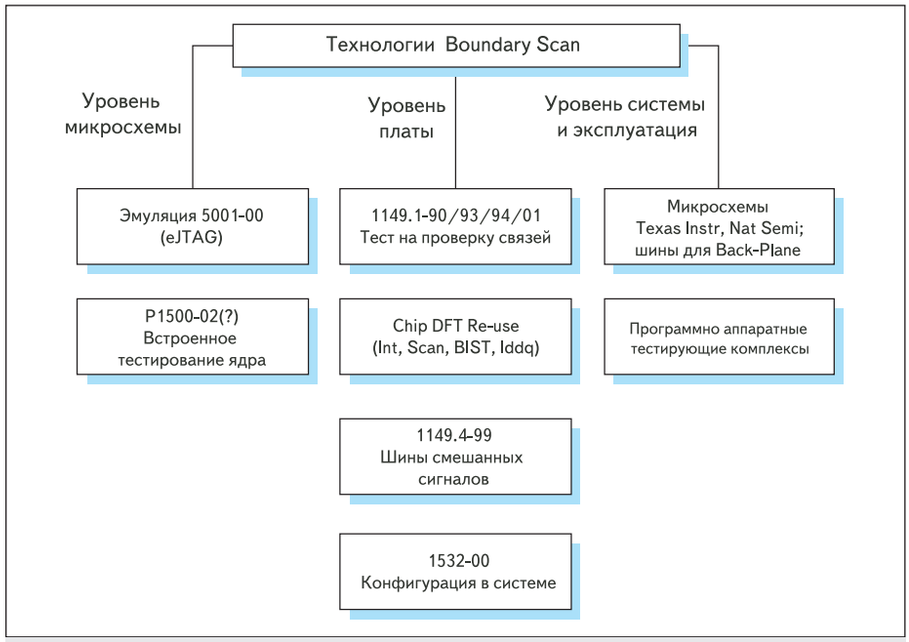

Рис. 12. Развитие технологии Boundary Scan

на применении технологии JTAG для граничного сканирования. Зачем здесь нужно останавливаться на технологии граничного сканирования, ведь «обещаны» только методики отладки проектов пользователя? Но кроме «штатного» регистра граничного сканирования («хард»), находящегося в той самой микросхеме, с которой работает пользователь, можно ввести в проект свои «софт»-регистры граничного сканирования. К тому же, эта технология может использоваться для тестирования внутри кристалла и для проверки внешних связей между микросхемами на плате. По этой же технологии можно подать в отлаживаемую микросхему внешние сигналы, приходящие от другой микросхемы. Конечно, при условии, что та, другая микросхема тоже управляется по JTAG.

Как же работает механизм граничного тестирования? Вся «хитрость» здесь заключена в том, что между ядром микросхемы и выводами помещается мультиплексор, который может отключать ядро микросхемы от выводов и вместо ядра подключать к выводам сдвиговый регистр, называемый регистром граничного сканирования (регистр Boundary Scan). В микросхему передаются те данные, которые посылает ей мастер интерфейса. А какие же данные получает пользователь при чтении из микросхемы? Ведь у каждой границы, как известно, есть две стороны. Сигналы на регистр внутри микросхемы подаются через мультиплексор, и он может брать сигналы, во-первых, изнутри микросхемы, и тогда пользователь может прочитать состояние выводов ядра микросхемы, а во-вторых, сигналы на сдвиговый регистр могут подаваться и извне микросхемы, то есть непосредственно с ее входов/выходов. Первый из этих режимов называется INTEST, а второй — EXTEST. Далее команды, выполняемые портом JTAG, а следовательно, и его режимы работы будут описаны более подробно, пока же приведем только основные команды, используемые при граничном сканировании. Обычно для него используют следующие режимы:

- В режиме работы EXTEST обеспечивается возможность снимать или устанавливать логические значения на рабочих контактах электронных компонентов, чем обеспечивается частичная или полная проверка внешних цепей, имеющих непосредственное отношение к тестируемому компоненту.
- В режиме работы INTEST обеспечивается возможность снимать или устанавливать логические значения внутри микросхемы, то есть на входах ядра микросхемы, чем обеспечивается частичная или полная проверка ядра микросхемы.
- Режим работы SAMPLE/PRELOAD позволяет тестировать ядро электронного элемента в статическом режиме, выставляя или снимая значения логических уровней на границе его выходных буферов.
- BYPASS — команда обхода, когда вся цепочка внутри микросхемы «вырождается» только в один триггер. При этом данные с входа передаются на выход с задержкой в один такт синхрочастоты интерфейса. Такой режим позволяет эффективно использовать возможности последовательного интерфейса при организации длинных последовательно объединенных цепочек.

Кроме этих режимов существуют и другие режимы работы микросхемы, оговоренные в стандарте. Все эти режимы, по возможности, будут рассмотрены далее при более подробном описании работы порта JTAG. А сейчас давайте рассмотрим аспекты применения этой технологии.

Применение режима граничного сканирования на уровне микросхемы

Итак, коротко напомним тезисы о применении тестирования на уровне микросхемы. Поскольку именно это и является целью данной статьи, то здесь, в этом пункте целесообразно остановиться только на следующих моментах. При помощи граничного сканирования можно, при отладке проекта в FPGA, получать на входах ядра микросхемы те сигналы, которые в реальной жизни получить трудно. Например, какой-нибудь сигнал аварии или сбоя при приеме 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 аппаратура, которую мы разрабатываем. Так как же проверить, получает ли FPGA сигнал и как его обрабатывает? По технологии JTAG, применяя граничное сканирование, надо только подать пару команд и сдвинуть соответствующую последовательность данных в регистр сканирования. Сигнал, который мы хотели увидеть, — появится. Конечно, он будет иметь другие временные характеристики, но ведь, как говорится, это почти даром.

Мало того, еще одним важным достоинством технологии граничного сканирования является то, что она позволяет вместо множества пробников использовать единственный интерфейс с 4 проводами.

В случае применения микросхем, имеющих порт JTAG (рис. 13а), охват тестированием может достигать 100%, и при этом отпадает необходимость в применении контрольных гнезд для проверки наличия сигналов. В том случае, когда часть микросхем на плате не имеет портов JTAG (рис. 13б), такие микросхемы при тестировании могут быть выделены в отдельные кластеры, на которые выдаются тестовые воздействия и с которых, через микросхемы, имеющие порты JTAG, получаются результаты тестирования.

Охват испытаниями граничного сканирования уровня платы

Для того чтобы проводить проверку на уровне платы, компоненты, имеющие интерфейс граничного сканирования, должны быть соединены в последовательную цепочку, начинающуюся от TDI и заканчивающуюся на TDO, так, чтобы они сформировали единую цепочку граничного сканирования. Платы, составленные из компонентов, которые на 100% соответствуют 1149.1, могут быть проверены векторным тестовым набором,

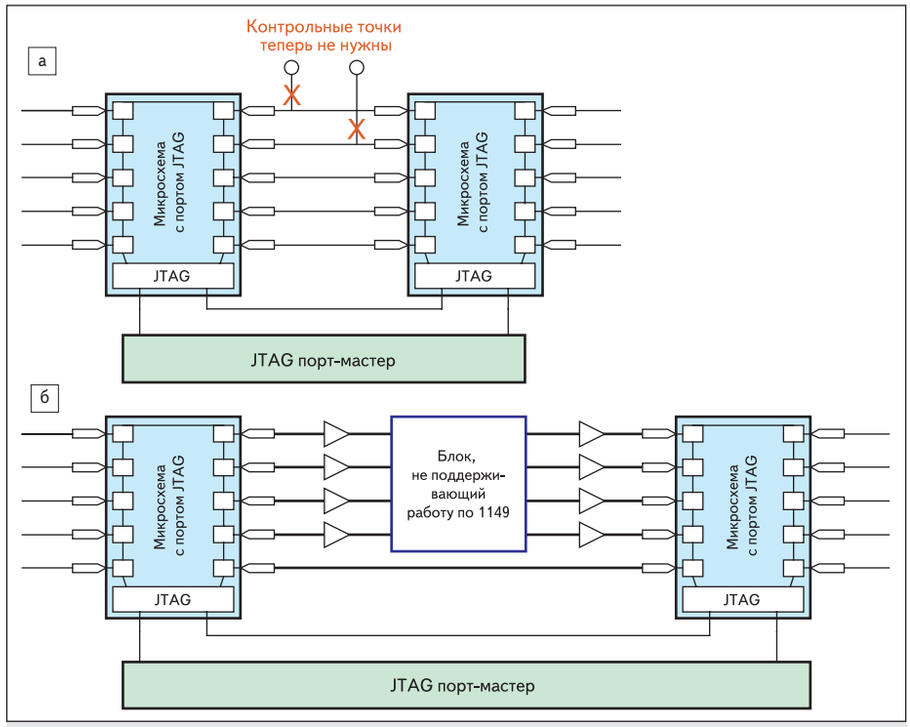

Рис. 13. а) Применение микросхем, имеющих порт JTAG;
б) применение микросхем на платах, не имеющих портов JTAG

сгенерированным программно. Такой класс программ называется ATPG (Automatic Test Program Generation) и дает до 100% охвата по поиску ошибки. Многочисленные аппаратные тестеры, проверяющие платы по интерфейсу JTAG, обладают разной производительностью и, соответственно, разной ценой. Следовательно, разработчик устройств всегда может выбрать то тестирующее оборудование, которое для его условий производства даст наибольшую экономию усилий и устранит необходимость проведения дорогих испытаний, проводимых вручную. Пример такой схемы приведен на рис. 13а.

В том случае, когда не все микросхемы, «участвующие в деле», имеют порты для подключения в цепь JTAG, тестовые воздействия составляются таким образом, чтобы учесть характеристики кластера, не охваченного цепочкой граничного сканирования (рис. 13а и б). Таким образом, и в этом случае платы также могут быть протестированы по методологии граничного сканирования. На рис. 13б показано, что те компоненты платы, которые не имеют непосредственного подключения к цепочке граничного сканирования, могут быть выделены в кластер, который тестируется, как «черный ящик».

Фактически, даже одна микросхема, находящаяся на плате и имеющая интерфейс граничного сканирования, значительно упростит проведение испытаний и может улучшить контролируемость платы. Особенно, если этот компонент, имеющий интерфейс граничного сканирования, представляет собой сложную микросхему, например, если это микропроцессор, FPGA или специализированная интегральная схема. Особенно этот выигрыш становится заметным в случае применения микросхем в корпусах BGA с большим количеством входов и выходов. Именно по этой причине применение граничного сканирования сейчас стало особенно необходимо. При выполнении печатных плат с большим количеством слоев возможны случаи, когда часть проводников пройдет от одной микросхемы к другой только по внутренним слоям и не будет иметь выхода на поверхность платы.

Как было сказано выше, для создания 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, какие выводы имеет микропроцессор. Важно только, может ли конкретный выход быть входом, выходом, входом/выходом, и можно ли его переключать из состояния «прием» в состояние «передача». Далее состояние такого входа 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, для того, чтобы описать выводы ИС, и порядок работы встроенной в нее 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.

Преимущества, получаемые при использовании граничного сканирования на уровне многоплатной системы

На уровне многоплатной системы также могут быть организованы одна или несколько цепочек граничного сканирования. В системной интеграции цепочка граничного сканирования каждой платы должна быть привязана к архитектуре самой платы. Предпочтительный метод объединения плат в систему состоит в том, чтобы подключать цепочки граничного сканирования каждой платы к основной плате так, чтобы каждая плата проверялась по отдельной цепи граничного сканирования. Тогда 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 системе самотестирования для встроенных систем. В том случае, когда проверяемая система не имела встроенного порта для проведения граничного сканирования, изготовителю оборудования необходимо было выполнять трудоемкие испытания на функционирование. Для этого необходимо было разрабатывать специальные тесты для каждого режима испытаний. Преимущество проекта, поддерживающего режим граничного сканирования, состоит в том, что возможно применить те же самые тесты, которые уже использовались для испытаний плат и узлов, составляющих данное изделие. Фактически одни и те же тесты, которые применялись на разных этапах производства, могут многократно использоваться, возможно, с небольшой модификацией, и для испытания встроенных систем.

Преимущества, получаемые на разных этапах жизни изделия при использовании граничного сканирования

Использование технологии граничного сканирования в микросхеме, на плате или в устройстве действительно увеличивает как их стоимость, так и время разработки проекта. Однако эти затраты легко покрываются при рассмотрении выгод и снижении стоимости при проведении автоматического тестирова-

ния, которое обеспечивается на каждой стадии цикла жизни изделия. То, что было первоначально разработано как производственный испытательный инструмент, дает определенные преимущества до начала производства, во время серийного производства и после производства, то есть на этапе эксплуатации.

Проектировщики используют граничное сканирование, чтобы сэкономить время на стадии макетирования и на этапах отладки проекта. Это происходит потому, что различные структурные ошибки, не связанные с самим проектом, такие как дефекты проектирования и изготовления печатной платы, могут быть обнаружены и затем устранены еще на уровне тестирования смонтированной платы и компонентов.

Но, 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 обеспечения дополнительного снижения стоимости и времени на проверку изделий при выходном контроле.

Основные преимущества в производственной фазе — экономия времени при разработке испытательных тестов, улучшенный охват при поиске ошибки и диагностировании и улучшенная производительность при проведении испытаний, при одновременном уменьшении времени испытания. Как и в фазе проектирования, граничное сканирование обеспечивает дополнительное преимущество при эксплуатации у потребителя, в случае применения тестов, которые были использованы при производстве.

Применение граничного сканирования при эксплуатации изделия также позволяет получить определенные преимущества. Отказы при эксплуатации часто происходят из-за структурных отказов, которые вызываются повышенной температурой, влажностью, вибрацией, потому что изделия эксплуатируются в промышленной среде. Используя граничное сканирование, техники имеют возможность быстро проверить изделие на структурные ошибки вплоть до уровня компонентов без трудоемкого исследования или возвращения платы изготовителю на завод. Устранение трудоемких тестов позволяет производить более эффективный диагноз и ремонт, что уменьшает стоимость и системное время простоя.

Возвращаемся к описанию интерфейса. Что такое «интерфейс»?

Для начала давайте определим, что мы будем обсуждать в данном разделе. Сейчас нам необходимо определить, что такое интерфейс с портом JTAG. Именно это, а не то, «как оно устроено внутри».

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 приведенном выше описании все представляется довольно просто. Поскольку сам JTAG не является основным интерфейсом аппаратуры, то производители и не придерживаются жестких правил при реализации данного порта, в противоположность тому, что имеет место для таких основных интерфейсов, как, например, PCI. Что касается электрических характеристик сигналов, то они обычно выбираются так, чтобы соответствовать TTL, LVTTL, CMOS, LVCMOS или другим распространенным стандартам, и тут тоже нет жесткой привязки к какому-то определенному стандарту. Что же касается конструкции разъемов, то тут, можно смело сказать, полный хаос. Каждый из производителей микросхем предлагает свой вариант реализации разъема для связи с портом. Как указывалось выше, читатель сможет найти различные варианты выполнения конструкции разъемов в [9].

Таким образом, остается описать только названия и логические соотношения сигналов при работе порта. Автор заранее просит прощения у тех читателей, которые знакомы с описанием данного интерфейса, они могут смело пропустить последующую главу. Но, поскольку есть и те, кто не знаком с работой данного интерфейса, то приведенное далее описание будет ориентировано именно на начинающих.

Интерфейс JTAG — это очень просто!

Чтобы не повторять учебники, занудные переводы и другую надоевшую техническую документацию, попытаемся разобраться с этим интерфейсом «по-чапаевски», но не на картошке, а на часах.

В середине 1980-х автор этой статьи на деле изучал, что такое «братство народов». Это был период, когда вся наша страна строила новый ракетный старт на Байконуре для «Бурана-Энергии». Кое-что об этом можно прочесть на сайте автора. Среди наших смежников были ленинградские фирмы, конечно, были и московские, а также специалисты из Омска, Харькова, Нальчика, Ужгорода и Мукачева и еще из многих других городов. Так, вот именно эти западно-украинские ребята и привезли тогда на полигон полчемодана импортных, как говорили тогда, электронных часов. Такие часы для нас были в диковинку, и эти-то часы и шли в обмен на колбасу и тушенку. Таким образом, мы впервые и опробовали JTAG'о-подобный тип интерфейса. На рис. 14 изображены аналогичные современные цифровые часы, которые также управляются четырьмя кнопками. Если мы хотим переставить на часах время, то нам нужна как минимум одна кнопка для того, чтобы «подать» в часы значение данных — то есть времени, и нужна еще одна кнопка для того, чтобы можно было задать два значения — «0» и «1». Теоретически, этим можно было бы и ограничиться, но только при условии, что мы имели бы ограничение на длину передаваемых данных и число пунктов меню. Тогда алгоритм работы был бы такой:

- первый пункт меню, данные первого пункта (длина пакета данных — фиксированная), переход ко второму пункту меню;
- второй пункт меню, данные первого пункта (длина пакета данных — фиксированная), переход к третьему пункту меню;
- остальные пункты меню аналогичны предыдущим, переход к последнему пункту меню;
- последний пункт меню, данные первого пункта (длина пакета данных — фиксированная), переход к первому пункту меню.

Как можно заметить из такого алгоритма работы, при двух интерфейсных сигналах мы получаем очень жесткие ограничения на работу интерфейса. А как же этот недостаток устранен в часах? Для этого там есть еще одна кнопка — Mode. Именно этот сигнал и позволяет избежать всех ограничений при выборе режима работы и тех из них, что связаны с длиной передаваемых данных. И кнопка Mode вступает в дело перед тем, как передавать данные в часы, и после того, как нужные данные уже переданы, и пользователь хочет перейти к новому режиму работы. Единственное отличие при управлении часами состоит в том, что кнопку Mode можно нажимать асинхронно, без сопровождения сигналом тактирования. Итак, интерфейс часов состоит из трех кнопок, с их помощью информация передается «туда» и с дисплея, на котором отображаются данные, «принятые оттуда». Ну, вот вам, уважаемый читатель, совсем не классическое, но очень наглядное описание принципов работы порта JTAG.

Кстати, когда автору в те далекие 1980-е годы понадобилось разработать модуль индикации для проверки работы процессорной стойки устройства управления, то в качестве управляющих органов на ручку модуля были выведены именно такие три кнопки — «Режим», «Бит» и «Сдвиг». И тремя кнопками удалось переключать 6 режимов работы модуля. Такой способ управления режимом работы индикатора и был выбран именно после детального знакомства с новыми часами. И еще, раз уж зашла речь об этом модуле индикации, именно тогда мы впервые применили микросхему ПЗУ в качестве CPLD, как сказали бы теперь. Правда в ПЗУ была «зашита» только комбинаторная логика, которая управляла режимами работы разных узлов модуля, а триггеры находились «снаружи», но сам метод «зашивки» схемотехники не «в провода», а в ПЗУ был тогда применен автором впервые.

Рис. 14. Управление часами
1 — кнопка, которая передает в часы данные. В интерфейсе JTAG такой сигнал называется TDO;
2 — дисплею часов соответствует сигнал интерфейса JTAG, называемый TDI. По этому сигналу данные от проверяемого устройства передаются к мастеру;
3 — это кнопка — Mode, а в интерфейсе JTAG этот сигнал называется TMS;
4 — эта кнопка эквивалентна сигналу интерфейса JTAG — TCK. Этот сигнал применяется для тактирования всех остальных синхронных сигналов

Представьте, что вы управляете часами по последовательному интерфейсу. Кнопки — это сигналы интерфейса, направленные от вас — «мастера интерфейса» — к проверяемому изделию, а дисплей часов — это ответный сигнал проверяемого изделия, передаваемый к вам, то есть к «мастеру» (рис. 14).

Теперь посмотрим на сигналы интерфейса JTAG. Все то же самое: сигнал данных на передачу — TDO, сигнал данных на прием — TDI, тактовая скорость — TCK и сигнал управления — TMS (табл. 1). Так же, как и в примере с часами, есть только один мастер интерфейса, который полностью контролирует работу всех абонентов, подключенных к цепочке. И так же, как и в часах, кнопка Mode только выбирает нужное нам меню, а сама команда передается в данных. Следовательно, JTAG — это интерфейс с косвенной адресацией. (Возможно, это определение звучит не совсем научно, но именно эту фразу нужно «сильно» запомнить при работе с этим интерфейсом!)

Здесь мы не будем останавливаться на всех подробностях топологии, которые могут иметь место при конструктивной реализации длинных JTAG-цепочек, состоящих из большого числа микросхем, на возможностях подключения в цепочку JTAG дополнительных контроллеров, которые позволяют добавить к основной цепи JTAG дополнительные ветви, которые, в свою очередь, также состоят из цепочек микросхем и т. д.

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

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 приема 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».

автомата управления TAP (см. далее диаграммы работы TAP контроллера) — Shift-IR или Shift-DR. Выходные данные выставляются под задний фронт сигнала TCK.

Порт JTAG + еще один порт JTAG — это тоже очень просто!

Как мы теперь уже знаем, JTAG применяется как универсальный интерфейс, который можно использовать не только для тестирования отдельных микросхем, но и для тестирования плат и даже целых устройств, состоящих из нескольких плат. На рис. 15 показано соединение четырех 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 так далее, последний абонент возвращает данные в порт.

Вот этот процесс необходимо описать более подробно. Поскольку сигналы управления включены параллельно, то у нас есть возможность управлять микросхемами, но только всеми одновременно! А вот чтобы к нужной микросхеме в нужный момент пришли данные, необходимо учитывать ее место в цепочке, поскольку данные передаются от одной микросхемы к другой по цепочке. Для этого в специализированном языке SVF, который применяется 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 мы должны поступить следующим образом. Сначала в испытуемую цепь «проталкиваются» данные, относящиеся к Header, затем к Target и наконец к Trailer (рис. 15).

Да, но как же тогда все это работает? — спросит читатель. А вот как. Действительно, все микросхемы в цепочке одновременно выполняют команду, например «принять код команды» (более подробно это будет описано далее). Но вот сами коды команды передаются по последовательной цепочке, и эти коды могут быть различными для разных абонентов. 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.

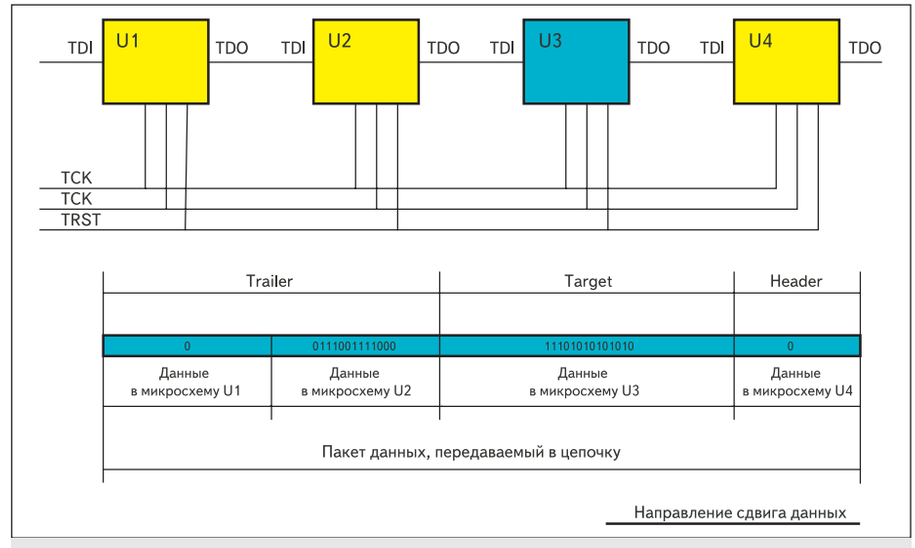

Рис. 15. Соединение четырех абонентов интерфейса в цепочку, подключенную к одному порту мастера (зеленым цветом показан пакет данных, передаваемых в JTAG-цепочку)

Теперь пора «заглянуть внутрь» микросхемы

Как работает интерфейс? Как обрабатываются данные? Как данные отличаются от команды? Как принятая команда поступает на исполнение?

Давайте рассмотрим все эти вопросы. Поскольку используется интерфейс с последовательной передачей данных, то, без сомнения, присутствуют входной и выходной сдвиговые регистры. Далее должен быть автомат, который управляет режимом работы TAP, и выходной мультиплексор, коммутирующий на выход данные с регистра, выбранного для текущего режима работы. Блок-схема, приведенная на рис. 16, схематично показывает структуру интерфейса.

Рассмотрение режимов работы начнем с описания работы управляющего автомата. (Это, конечно, классика, но без этого обойтись невозможно.) Этот автомат имеет 16 состояний. Переходы автомата выполняются под управлением сигнала TMS на переднем фронте сигнала TCK. На рис. 17 представлена диаграмма переходов автомата, который управляет режимами работы TAP. Рядом со стрелками указано значение сигнала TMS, при котором происходит соответствующий переход.

Контроллер TAP при включении находится в состоянии Test_Logic Reset («ждущее» состояние). До тех пор пока сигнал TMS остается в состоянии «1» (значение по умолчанию), состояние автомата остается неизменным. Обычно в таком состоянии по умолчанию выбран регистр Идентификации 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 этот сигнал не менее пяти тактов частоты TCK. Именно поэтому на входе TMS устанавливают резистор, создающий подпор данного сигнала в «1».

Если сигнал TMS будет переключен хостом в низкий уровень, то автомат перейдет к состоянию Run_Test/Idle (активное состояние, в котором ничего не делается). Обычно из этого состояния можно двигаться 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 перейдет в состояние Run_Test/Idle. Если же кратковременная помеха (длительностью не более 1 периода синхрочастоты) прекратится, то автомат через 3 такта снова вернется в исходное состояние — Test-Logic-Reset (рис. 18). Исходя из этого тезиса, 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 надо сдвинуть 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. 18 и выделена красным цветом.

Главное, что необходимо здесь дополнительно подчеркнуть, так это то, что команда передается в состоянии автомата Shift_IR. И при этом на входе TMS присутствует сигнал низкого уровня. А для того чтобы выйти из этого состояния и перейти к следующему, необходимо на входе TMS установить

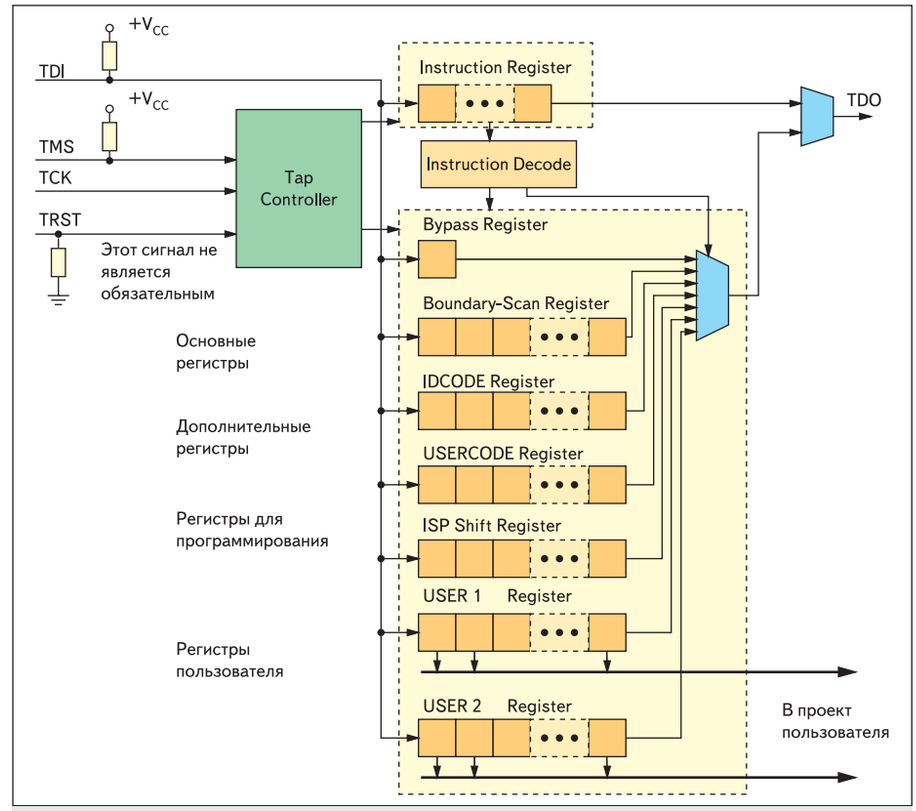

Рис. 16. Блок-схема TAP

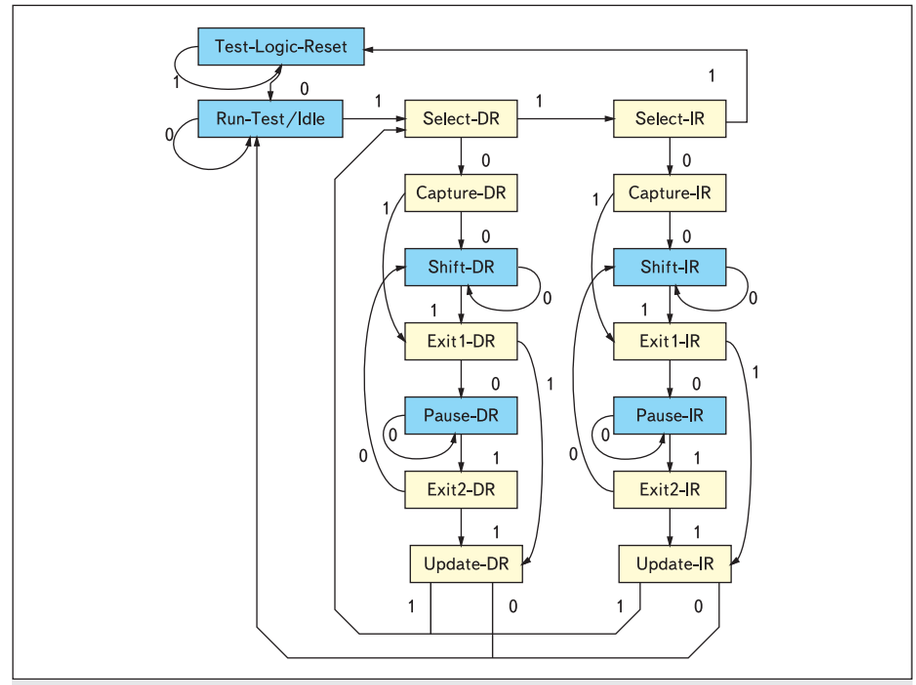

Рис. 17. Диаграмма переходов статического автомата по IEEE STD 1149

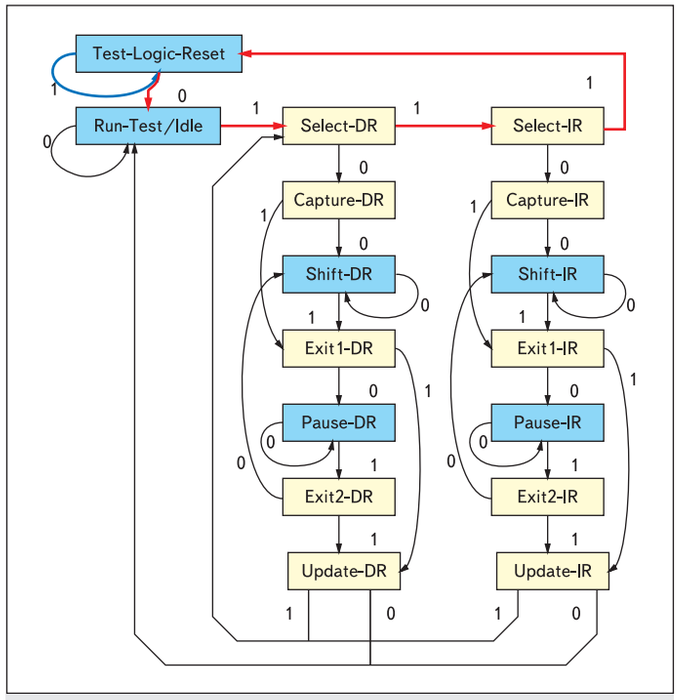

Рис. 18. Диаграмма переходов при подаче на вход TMS сигнала 1, соответствующего устойчивому состоянию сброса — Test-Logic-Reset. (Переход выделен синим цветом. Красным цветом выделены переходы при кратковременной помехе на входе.)

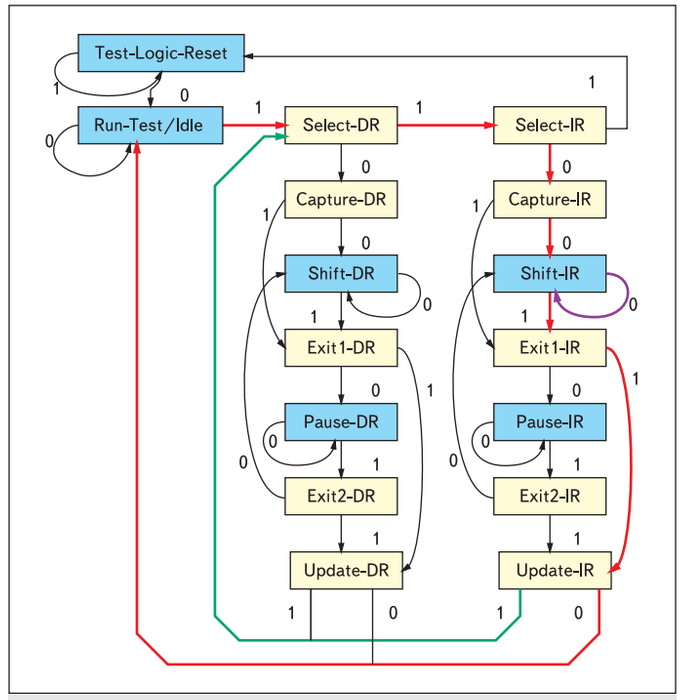

Рис. 20. Диаграмма переходов, которые необходимо выполнить для того, чтобы загрузить в контроллер новую команду (эти переходы выделены красным цветом)

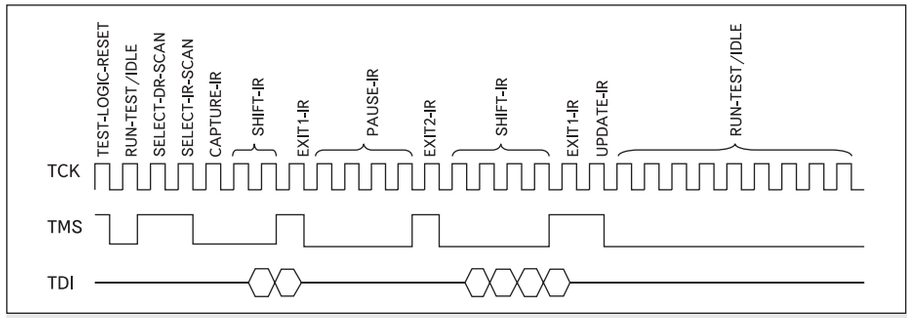

Рис. 19. Диаграмма сигналов при выполнении переходов для загрузки команды

сигнал высокого уровня. И при этом сигнал высокого уровня должен быть выдан с последним битом команды, передаваемым от TDI к TDO. То же самое условие необходимо будет соблюдать при всех задачах, выполняемых с портом JTAG при работе автомата в состояниях Shift_IR и Shift_DR.

На рис. 19 показана диаграмма сигналов при выполнении переходов для загрузки команды.

Если пользователь после загрузки регистра команд хочет работать с данными, то он также может из состояния автомата Update_IR сразу перейти в состояние Select_DR. Этот переход на рис. 20 выделен зеленым цветом.

Теперь необходимо сказать несколько слов о том, какие команды выполняет TAP. Их условно можно разбить на несколько групп, в зависимости, от того, что выполняет данная команда и от того, кто встроил в TAP тот функциональный узел, с которым данная команда работает. Группа команд, показанная в таблице 2, — это обязательные команды, выполняемые интерфейсом. Адресация этих команд, порядок их выполнения и функциональность узлов, связанных с ними, жестко оговорена стандартом.

Из обязательных команд EXTEST, INTEST, SAMPLE/PRELOAD и BYPASS уже были кратко упомянуты выше. А сейчас более подробно расскажем об одной команде — BYPASS. После ее получения TAP «превращается» в один триггер, включенный между входом TDI и выходом TDO. При этом данные на выходе TDO, относительно входа TDI, появляются с задержкой в один такт частоты TCK. Для того чтобы точно знать, какие именно команды выполняет та или иная микросхема, необходимо посмотреть BSDL-файл, который ее описывает. Но! Осторожно, дорогой читатель! Учтите, что тот файл, который вы взяли сегодня с сайта фирмы-производителя, возможно, завтра может быть заменен на другой. И неважно, что файл помечен как проверенный. Важно только то, что фирма не сообщает о замене файлов. Именно так поступает, например, фирма Xilinx. Эта компания не сообщает о замене BSDL-файлов. Поэтому всегда необходимо проверять, не произошли ли какие-либо изменения в документации. Еще одно хорошее место для поисков BSDL-файлов — это тот пакет программ, с которым вы работаете, так как обычно среди прочих папок находится и скромная папка с названием BSDL.

Теперь несколько слов о структуре BSDL-файла. Это VHDL-подобный файл, и он описывает то, как сформирован регистр сканирования. Там же приведены коды команд и их длина. На рис. 21 приведена часть BSDL-файла, описывающего ресурсы микросхемы BlackFin BF533. На рисунке можно увидеть разделы, описывающие коды команд, и их символические имена.

Есть только еще одно «тонкое» место. Поскольку регистр Boundary Scan — сдвиговый, то очень важно, в какую сторону производится сдвиг. Для JTAG-технологии все то, что находится в регистрах, сдвигается ВПРАВО. Поэтому в документации есть легкая путаница с младшими и старшими битами. И если в одном месте данные для сдвига изображены вот так:

```
attribute INSTRUCTION_OPCODE of XC2V250_FG456 : entity is
                «EXTEST      (000000),» &
                «SAMPLE      (000001),» & …
```

Это значит, что, например, команда SAMPLE выдается на TDI как 1, потом 0 и затем 0,0,0,0.

```
attribute INSTRUCTION_LENGTH of ADSP_BF533: entity is 5;

-- Unspecified opcodes assigned to Bypass.
attribute INSTRUCTION_OPCODE of ADSP_BF533: entity is
        «BYPASS (11111),» &
        «EXTEST (00000),» &
        «SAMPLE (10000),» &
        «IDCODE (00010),» &
        «MEMBIST (01010),» &
        «EMULATION (00100,10100,01000,11110,01100),» &
        «CUSTOMER_KEY (10110),» &
        «TESTKEY      (00110)»;


attribute INSTRUCTION_CAPTURE of ADSP_BF533: entity is
        «00001»;

attribute INSTRUCTION_PRIVATE of ADSP_BF533: entity is
        «EMULATION,» &
        «MEMBIST,» &
        «CUSTOMER_KEY,» &
        «TESTKEY» ;

attribute IDCODE_REGISTER of ADSP_BF533: entity is

        «0010» &                        -- Version
        «0010011110100101» &
        «00001100101» &
        «1»;
```

Рис. 21. Часть BSDL-файла, описывающая ресурсы микросхемы BlackFin BF533

Таблица 2. Обязательные команды интерфейса

Команда	Функция
EXTEST	Тестирование производится для сигналов, находящихся снаружи микросхемы
SAMPLE	Прием данных
PRELOAD	Предустановка данных
BYPASS	Передача данных с входа на выход через регистр обхода. Задержка при передаче данных — 1 такт синхрочастоты

Ну а кое-где в документации иногда можно встретить другие кодировки команд, путаницу в обозначениях старших и младших битов в поле команды и т. п.

Во вторую группу команд, показанную в таблице 3, входят дополнительные команды, выполняемые интерфейсом. Адресация этих команд, порядок их выполнения и функциональность узлов, связанных с ними, также в большинстве случаев жестко оговорена стандартом. Исключения в этой группе составляют команды Usercode и Runbist. Из этой группы сейчас мы более подробно рассмотрим только одну команду — Idcode. После ее получения TAP немедленно готов «в данных» выдавать на выходе TDO значение идентификационного кода данной микросхемы.

Данные, получаемые при выполнении команды «чтение идентификационного кода», могут быть, в соответствии с IEEE STD 1149.1, представлены в виде следующих полей:

Таблица 3. Дополнительные команды интерфейса

Команда	Функция
Intest	Тестирование производится внутри микросхемы
Idcode	Команда чтения идентификационного кода. Действует всегда, с момента включения микросхемы
Usercode	Команда чтения кода, загруженного пользователем
Runbist	Выполнить встроенный тест самопроверки
Clamp	Данные на выходах зафиксированы
HighZ	Выходы находятся в третьем состоянии

- Manufacturers ID — код производителя;
- Part Size Code — код, описывающий микросхему в данной серии;
- Family Code — код серии микросхем;
- Revision Code — код ревизии изделия.

В таблице 4 приведены примеры кодировки полей идентификационного регистра для нескольких микросхем.

Полную информацию о кодировке производителей микросхем можно получить из документа [10].

Несколько более подробно необходимо описать необходимость приема «1» в последнем поле. Как было сказано выше, данные сдвигаются вправо, младшим битом вперед. Поэтому программа, обслуживающая работу с портом, должна отслеживать появление бита «1» как первого бита данных в строке ID. После чего необходимо принять оставшиеся 31 бит. А как определить, когда появится нужная нам единица? Сканирование данных будет выполнено после сканирования команды. А код, отдаваемый, например микросхемой BlackFin BF533, — «00001» (рис. 21). Таким образом, последние биты, принятые после сканирования команды, — нули, а первый бит сканирования данных — единица.

По этому поводу необходимо сделать дополнительное замечание. Когда программа получает из микросхемы данные, соответствующие идентификационному коду, то необходимо учитывать, что эти данные были «зашиты» в кристалле. А кристалл «не знает», в каком корпусе он разварен. Таким образом, кристаллы с одинаковыми объемами FPGA, но находящиеся в разных корпусах будут выдавать одинаковые идентификационные коды. И, следовательно, программа может определить тип микросхемы, серию, но пользователь не имеет возможности точно определить тип корпуса и выбрать BSDL-файл, соответствующий данному корпусу.

Здесь, к сожалению, нет возможности описать команды, которые предназначены для режима загрузки микросхем FPGA. Чтобы понять, что и как делает «штатный» загрузчик, достаточно задать ему режим записи команд прошивки, например в SVF-файл, а затем посмотреть коды команд, их последовательность и данные, предназначенные для загрузки.

И в последнюю группу команд выделим те, что вводятся производителем для повышения функциональности изготавливаемых им микросхем. Адресация этих команд, порядок их выполнения и функциональность узлов, связанных с ними, не оговорена стандартом. К этой группе относятся команды, позволяющие работать с внутрисхемными эмуляторами, встроенными в микросхемы, с технологическими проверочными узлами, которые производители используют для контроля параметров микросхем. Сюда же мы отнесем и те встроенные узлы, которые производители FPGA предоставляют в распоряжение пользователей.

В таблице 5 показаны дополнительные команды, выполняемые интерфейсом. Эти команды используются для тестирования проекта пользователя в FPGA фирмы Xilinx.

Таблица 5. Дополнительные команды, выполняемые интерфейсом. Эти команды используются для тестирования проекта пользователя в FPGA фирмы Xilinx

Команда	Функция	Дополнительное описание
USER1	Определяется пользователем	Только в определенных микросхемах фирмы Xilinx
USER2	Определяется пользователем	Только в определенных микросхемах фирмы Xilinx

Во время передачи команды в микросхему мы можем прочесть из нее данные. Они описаны в BSDL-файле, в разделе INSTRUCTION_CAPTURE. Посмотрим на рис. 21, где описываются ресурсы микросхемы BlackFin BF533. Из этой микросхемы мы прочитаем строку «00001». Единственное, о чем мы можем судить по этим данным, так это то, что операция чтения произошла правильно. Другое дело у FPGA фирмы Xilinx, например у Virtex2. На рис. 22 приведен фрагмент BSDL-файла для микросхемы Virtex2-XC2V1000. При передаче команды в микросхему из нее читаются следующие данные:

- бит 5=1, когда сигнал DONE «отпущен» и не удерживается в низком уровне;
- бит 4=1, когда произошла очистка внутренней памяти;
- бит 3=1, когда конфигурация разрешена;
- бит 2=1, когда конфигурация выполнена.

```
attribute INSTRUCTION_CAPTURE of XC2V1000_BG575 : entity is
-- Bit 5 is 1 when DONE is released (part of startup sequence)
-- Bit 4 is 1 if house-cleaning is complete
-- Bit 3 is ISC_Enabled
-- Bit 2 is ISC_Done
 «XXXX01»;
```

Рис. 22 . Фрагмент BSDL-файла для микросхемы Virtex2-XC2V1000

Таблица 4. Примеры кодировки полей идентификационного регистра микросхем

Поле	Названия полей идентификационного регистра				
	Revision Code	Family Code	Part Size Code	Manufacturers ID	LSB
	31...28	27...21	20...12	11...1	0
Virtex XC2V250	«XXXX»	«0001000	«000011000»	«00001001001»	«1»
Virtex XC2V1000	«XXXX»	«0001000»	«000101000»	«00001001001»	«1»
ADSP BF533	«0010»	«0010011»	«110100101»	«00001100101»	«1»
Altera EPF10K20R240	«0000»	«0001000»	«000100000»	«00001101110»	«1»

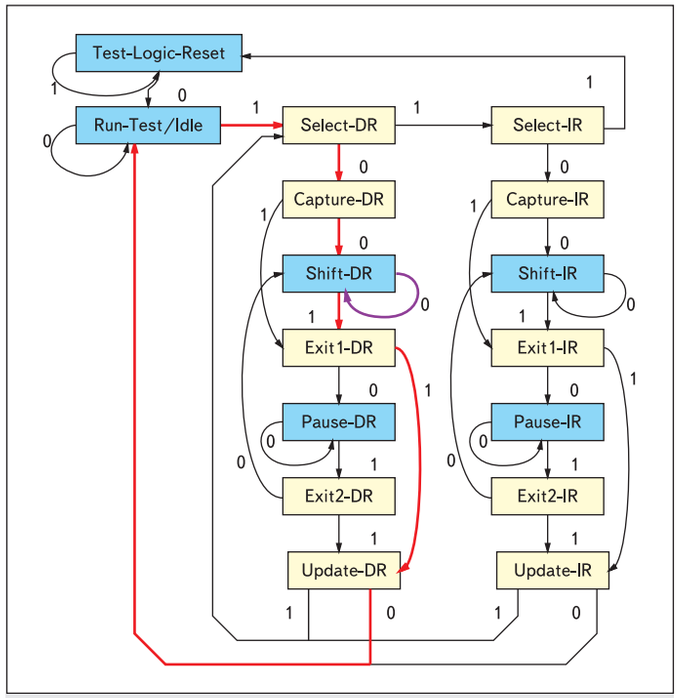

Рис. 23. Диаграмма переходов, которые необходимо выполнить для того, чтобы работать с данными

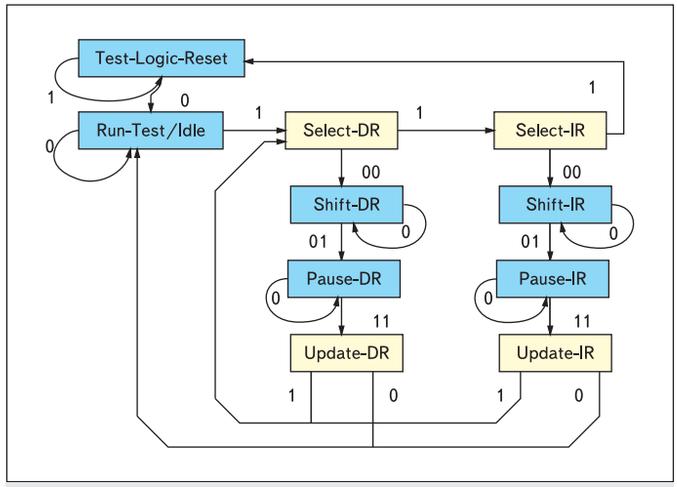

Рис. 24. Упрощенная диаграмма переходов

Именно по битам этой строки можно точно узнать, завершилась ли операция загрузки FPGA.

Диаграмма переходов, которые необходимо выполнить, чтобы работать с данными (рис. 23), также обычно начинаются от состояния Run_Test/Idle, затем Select_DR, Capture_DIR, Shift_DR. В состоянии Shift_DR происходит прием и передача данных. После работы с данными необходимо перевести автомат через состояния Exit1_DR, Update_DR и снова в Run_Test/Idle. При окончании работы с данными необходимо выполнить правило, указанное выше, а именно необходимо передать бит TMS, указывающий на завершение данного режима вместе с последним битом данных.

Кроме рассмотренных переходов, есть возможность осуществить и другие переходы управляющего автомата, но рассмотрение этих вопросов выходит за рамки данной статьи. Главное, что необходимо сейчас отметить, это то, что мы не пользуемся всеми состояниями автомата для загрузки команды и обмена данными. Поэтому для того, чтобы упростить разработку программы, работающей с JTAG-портом, нужно сократить число состояний автомата, которыми мы будем пользоваться. И тогда диаграмма переходов будет выглядеть так, как показано на рис. 24.

Дальнейшее «взаимодействие» с этой упрощенной диаграммой переходов будет происходить при описании программы, которая позволит работать с JTAG-портом пользователя.

Окончание следует

Литература

1. Семенов Н., Каршенбойм И. Микропрограммные автоматы на базе специализированных ИС // Chip News. 2000. № 7.
2. Каршенбойм И. Микропроцессор своими руками // Компоненты и технологии. 2002. №№ 6, 7.
3. Каршенбойм И. Микропроцессор своими руками-2. Битовый процессор // Компоненты и технологии. 2003. №№ 7, 8.
4. Каршенбойм И. Микропроцессор своими руками-3. Ассемблер и софт-симулятор // Компоненты и технологии. 2006. №№ 3, 5.
5. Каршенбойм И. Виртуальные кнопки и светодиоды, или Неизвестное обо всем известном JTAG-сканировании // Компоненты и технологии. 2005. № 7.
6. Каршенбойм И., Паленов К. «Встроенный» логический анализатор — инструмент разработчика «встроенных» систем // Схемотехника. 2001. № 12.
7. w ww.digilentinc.co m/Resources/DesignContest.cfm?Nav1=Design&Nav2=DesignContest
8. IEEE STD 1149.1. IEEE Standard Test Access Port and Boundary-Scan Architecture.
9. w ww.amontec.c om. JTAG Interface: Common Pinouts amt_ann003 (v1.1) Application Note
10. EIA/JEP106, JEDEC Publication 106, Standard Manufacturer's Identification Code.