Микросхемы ПЛИС Speedster22i от Achronix: самые быстрые и самые большие. Часть 2

Иосиф КАРШЕНБОЙМ
iosifk@narod.ru

В этой части мы рассмотрим вопросы, связанные с конфигурацией микросхем и с программированием флэш-памяти, предназначенной для загрузки конфигурации. Также будет приведено очень краткое описание языка STAPL.

Конфигурация

В микросхемы встроена достаточно развитая логика для того, чтобы выполнять множество режимов для конфигурации и программирования. На рис. 1 показана блок-схема основного блока программирования и конфигурации, включая дополнительную логику, которая реализует средства защиты. Блок управления конфигурацией контролирует запуск и последовательность завершения работы — от режима конфигурации до пользовательского режима, а также выход из пользовательского режима. Этот блок имеет возможность для конфигурации микросхемы зашифрованным потоком битовых данных, используя 256-битовое шифрование по стандарту AES (Advanced Encryption Standard). Микросхема также содержит небольшую энергонезависимую память для хранения необходимого ключа шифрования AES.

Далее мы рассмотрим процессы конфигурации более подробно.

Архитектура конфигурации в микросхемах HD Speedster22i состоит из следующих основных частей:

1. Выделенные входы управления для выбора режима конфигурации, определяющие, от какого из внешних интерфейсов данные будут передаваться в ПЛИС.
2. Блок конфигурации ПЛИС (FCU, FPGA Configuration Unit) является встроенным IP-блоком, содержащим интерфейсы, конечные автоматы управления, регистры режимов и другую управляющую логику, что позволяет этому блоку получать данные от различных входов, выполнять необходимые действия при конфигурации и преобразовывать входящий поток данных в формат, нужный для использования в регистрах конфигурации внутри ПЛИС.
3. Регистры конфигурации, в которые и записываются данные из потока битовых данных, записываемых в FCU.

На рис. 2 приведена блок-схема, демонстрирующая составные части блока конфигурации. Данные от различных входов конфигурации приходят в FCU, расположенный непосредственно рядом с входами/выходами, выделенными для конфигурации. В зависимости от режима конфигурации эти данные проходят через один из четырех интерфейсов, затем попадают в управляющую логику и конечные автоматы в FCU. Здесь данные принимаются и приводятся к требуемому для конфигурации виду, а затем разветвляются и передаются к регистрам конфигурации в кольце входов/выходов, а также параллельно загружаются в ядро структуры ПЛИС, в те столбцы, где находятся фреймы памяти конфигурации. Как только все биты конфигурации успешно загружены, FCU переводит ПЛИС в пользовательский режим, и микросхема начинает выполнять действия, заложенные в проекте пользователя.

Последовательность включения питания и конфигурация микросхемы HD Speedster22i показана в таблице 1 и более подробно описана далее.

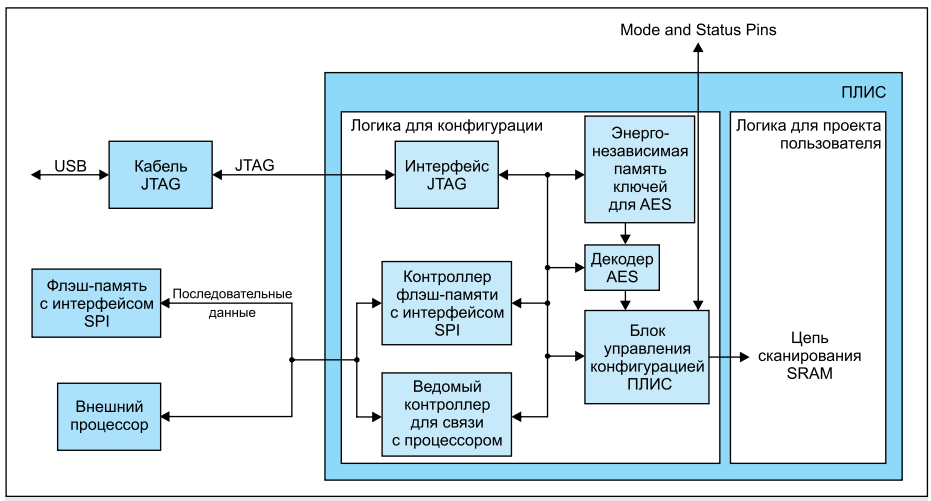

Рис. 1. Блок-схема основного блока программирования и конфигурации, включая дополнительную логику, которая реализует средства защиты

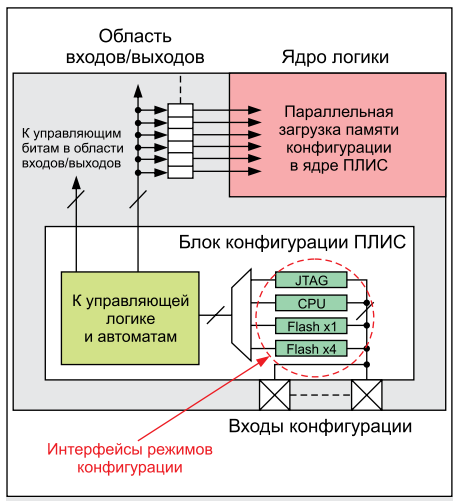

Рис. 2. Блок-схема, которая показывает составные части блока конфигурации

Таблица 1. Последовательность включения питания и конфигурация микросхемы HD Speedster22i

1	Включение питания микросхемы
2	Чтение энергонезависимой памяти
3	Очистка конфигурационной памяти
4	Синхронизация потока битовых данных и чтение ID микросхемы
5	Загрузка битов конфигурации
6	Проверка CRC
7	Последовательность команд, перевода микросхемы в пользовательский режим
8	Пользовательский режим

Включение питания микросхемы

Первый шаг по переводу в рабочее состояние микросхемы HD Speedster22i происходит при включении питающего напряжения. Последовательность подачи питающего напряжения более подробно изложена в [1], где и приведены примеры соотношений между последовательностью подачи электропитания и подачей управляющих сигналов на входы конфигурации так, чтобы гарантировать успешное включение питания ПЛИС:

- Необходимо произвести включение всех питающих напряжений за исключением напряжений, питающих SerDes, — это PA_VDD1, PA_VDD2, имеющие номинал 0,95 и 1,8 В, и напряжение VDDL, питающее ядро логики и имеющее номинал 1 В.
- После того как напряжение VCC достигает номинала, необходимо включить напряжение питания на PA_VDD1 и PA_VDD2.
- После того как PA_VDD1 и PA_VDD2 достигнут номинала, следует установить низкий уровень на сигнале сброса конфигурации CONFIG_RSTN. Это вызовет сброс ПЛИС.
- После некоторого времени (несколько миллисекунд) требуется снять низкий уровень с CONFIG_RSTN. Как только ПЛИС будет снята со сброса, начнут выполняться шаги 2 и 3 в таблице 1, то есть произойдет чтение энергонезависимой памяти и очистка памяти конфигурации. После того как память конфигурации очистится, будет установлен сигнал CONFIG_STATUS.
- Производится включение питания VDDL и выдерживается пауза, чтобы это напряжение достигло номинала. Затем ПЛИС будет готова принять поток битовых данных.

Чтение энергонезависимой памяти

После выхода из сброса FCU читает содержание энергонезависимой памяти (fuse) и запоминает полученные данные. Все данные в энергонезависимой памяти на заводе устанавливаются в нули и далее могут быть запрограммированы. Код производителя и ID могут быть запрограммированы во время тестирования ATE. Те же ячейки энергонезависимой памяти, которые содержат информацию, необходимую для безопасности проекта, могут быть запрограммированы пользователем.

Очистка памяти конфигурации

После того как произведено чтение энергонезависимой памяти, FCU переходит к состоянию очистки памяти конфигурации. Элементы памяти конфигурации являются 6-транзисторными ячейками SRAM и очищаются при записи в них «0». Если это состояние вводится после включения питания ПЛИС, то всю память конфигурации следует очистить до включения VDDL, иначе ячейки SRAM включатся в неопределенных состояниях. Поскольку в маршрутизации межсоединений имеются мультиплексоры, то в результате это приведет к неправильному поведению микросхемы.

Этот шаг можно обойти при отладке или оптимизации, устанавливая сигнал BYPASS_CLR_MEM. Данное действие выполняется только в том случае, если производят реконфигурирование ПЛИС без выключения питания.

Как только вся память очищена, микросхема устанавливает активный уровень сигнала CONFIG_STATUS, и небольшая внешняя подтяжка к питанию покажет на этом сигнале высокий уровень. Это означает, что FCU готов считать поток битовых данных.

Нужно отметить, что в данном шаге очищается только память конфигурации. Встроенные блоки памяти BRAM и LRAM НЕ очищаются и находятся после конфигурации в пользовательском режиме в неопределенных состояниях. Если необходимо предварительно загрузить содержание памяти, то используют файлы инициализации встроенной памяти.

Синхронизация потока битовых данных и ID-микросхемы

Потоки битовых данных ПЛИС HD Speedster22i всегда начинаются с синхронизирующего кода, который предварительно установлен как 0xAA55AA55. Синхронизирующий код всегда пишется в потоке битовых данных программным обеспечением ACE и прозрачен для пользователя. После кодов синхронизации микросхема должна получить определенный код идентификации. Такая проверка необходима для того, чтобы избежать программирования микросхемы «чужим» потоком битовых данных. Этот код также добавляется в поток битовых данных программным инструментом ACE.

Загрузка битов конфигурации

Поток битовых данных конфигурации является серией слов данных, которые попадают в микросхему и затем загружаются в память конфигурации в ядре ПЛИС. В потоке битовых данных, кроме самих данных, есть и команды, управляющие адресом загрузки битового потока, то есть данные из битового потока могут загружаться как в регистры конфигурации входов/выходов, так и в память конфигурации ядра логики.

Размер конфигурационного файла и время конфигурации прямо пропорциональны числу фреймов памяти конфигурации, которые должны быть запрограммированы в структуре ПЛИС. Размер конфигурационного файла тоже зависит от используемого режима программирования, этот размер может изменяться менее чем от 1 Мбайт для очень маленьких проектов и почти до 100 Мбайт для самых больших проектов, которые занимают всю ПЛИС и выполняют предварительную загрузку памяти BRAM.

Проверка CRC

В конце потока битовых данных выполняется проверка CRC, которая гарантирует, что данные, приходящие в память конфигурации, не имеют ошибок.

Последовательность запуска

После того как память конфигурации загрузится, в потоке битовых данных может быть выдана последовательность команд, чтобы перевести микросхему в пользовательский режим. Вводом и выводом из пользовательского режима управляет конечный автомат запуска/завершения работы, также реализованный в FCU. Процесс перехода в пользовательский режим действует независимо от конечного автомата конфигурации, и, таким образом, конечный автомат конфигурации способен обработать команды потока битовых данных даже после ввода пользовательского режима.

Последовательность запуска состоит из последовательной выдачи нескольких сигналов, что гарантирует правильное функционирование во время пользовательского режима. Эти события описаны в таблице 2. Последовательность завершения работы по своей сути подобна последовательности запуска, но в данном случае происходит сброс активных состояний этих же самых сигналов, хотя и в обратном порядке.

Таблица 2. События в последовательности запуска (Startup Sequence Events)

Этапы	Событие
1	Устанавливается сигнал Global Clock Enable
2	Устанавливается сигнал I/O Enable
3	Устанавливается сигнал Global Reset Enable
4	Устанавливается сигнал Global Core Enable
5	Устанавливается сигнал Config Done
6	Устанавливается сигнал User Mode Enable

Пользовательский режим

Как только микросхема попадает в пользовательский режим, это означает, что проект пользователя был полностью загружен и пользователь может начать отправлять и получать данные к/от ПЛИС и выполнять заложенные в проекте действия.

Режимы программирования и конфигурации

При проведении разработки и во время штатного функционирования ПЛИС

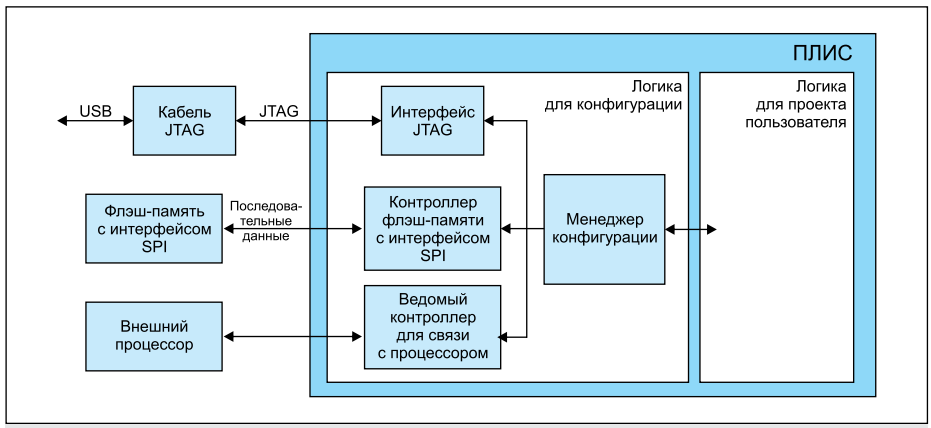

Рис. 3. Блок-схема вариантов режимов конфигурации

используется несколько режимов программирования и конфигурации. Чтобы избежать какого-либо непонимания, определим, что термин «программирование» относится к действию по записи потока битовых данных во флэш так, чтобы на следующем цикле подачи питания записанный поток битовых данных мог использоваться для конфигурации ПЛИС. Надо отметить, что рекомендуемый объем памяти, необходимый для хранения данных конфигурации для Speedster22iHD, составляет 128 Мбайт. Термин «конфигурация» относится непосредственно к процессу конфигурирования ПЛИС, чтобы реализовать в ней ту функциональность, которая требуется пользователю.

Поддерживаемый режим программирования — при помощи флэш-памяти с последовательным интерфейсом SPI.

Поддерживаемые режимы конфигурации:

- конфигурация ПЛИС по интерфейсу JTAG (JFC);
- конфигурация ПЛИС из флэш-памяти с последовательным интерфейсом SPI (SFC — Serial flash configuration);
- конфигурация ПЛИС внешним процессором (EFC — External CPU FPGA configuration);
- конфигурация ПЛИС из нескольких внешних микросхем памяти с последовательным интерфейсом SPI (MSF, Multiple Serial Flash interfaces).

Примечание. *Во всех перечисленных режимах загрузки из флэш-памяти ПЛИС являются ведущим устройством и от ПЛИС к флэш-памяти передается синхрочастота, которая и управляет синхронизацией конфигурации.*

На рис. 3 показана упрощенная блок-схема, где представлены варианты режимов конфигурации.

Выбор режимов конфигурации определяется уровнями напряжений, поданными на входы CONFIG_MODESEL, так как это показано в таблице 3.

Первые три режима определяются состояниями входов, а четвертый режим конфигурации — JTAG, не зависит от состояния входов и может быть активирован при записи соответствующих битов в пользовательском регистре данных, который расположен в JTAG TAP-контроллере (User Data Register). Как только этот режим прописан через JTAG, он запрещает загрузку во всех других режимах конфигурации, до тех пор пока не будет отключен.

Таблица 3. Режимы конфигурации и состояние управляющих входов CONFIG_MODESEL

Режимы конфигурации (Configuration Mode)	Состояние управляющих входов CONFIG_MODESEL[2:0]
CPU	100
Serial ×1 Flash	001
Serial ×4 Flash	010
JTAG	Всегда активное состояние

Описания входов, используемых для конфигурации

Режим конфигурации определяется статическим состоянием, задаваемым на входах управления режимами, — CONFIG_MODESEL [2:0]. В таблице 4 приведено описание входов ПЛИС, применяемых для различных режимов конфигурации. Одни и те же выделенные входы, предназначенные для конфигурации, могут быть использованы в разных режимах (например, конфигурация ПЛИС из одной флэш-памяти с последовательным интерфейсом SPI, конфигурация ПЛИС из четырех внешних микросхем памяти с последовательным интерфейсом SPI, конфигурация ПЛИС внешним процессором).

Конфигурация ПЛИС внешним процессором

Режим конфигурации ПЛИС от внешнего процессора реализуется после подачи питания на микросхему и может использоваться во время создания проекта пользователем или для серийно работающего изделия. В данном режиме процессор действует как ведущее устройство и управляет операциями программирования для ПЛИС. Конфигурации ПЛИС от внешнего процессора производится по 8-разрядному параллельному интерфейсу. Синхронизация выполняется сигналом процессора CPU_CLK, также процессором выдается сигнал выбора кристалла, который стробирует достоверные данные. Это самый быстрый режим программирования, поскольку он имеет интерфейс шины данных с самой большой разрядностью и максимальную поддерживаемую тактовую частоту 25 МГц.

На рис. 4 показана блок-схема конфигурации ПЛИС внешним процессором. Преимущество данного режима в том, что он может иметь несколько вариантов загрузочных файлов, к тому же разработчик может предусмотреть дистанционное обновление загрузочного файла.

Таблица 4. Входы, используемые в различных режимах конфигурации

Название внешнего вывода	EFC	SFC	MSF	JFC
SDI	DQ[0]	Последовательный выход данных к флэш-памяти		–
SDO[3]	DQ[1]	–	Вход данных конфигурации от флэш-памяти	–
SDO[2]	DQ[2]	–	Вход данных конфигурации от флэш-памяти	–
SDO[1]	DQ[3]	–	Вход данных конфигурации от флэш-памяти	–
SDO[0]	DQ[4]	Вход данных конфигурации от флэш-памяти		–
HOLDN	DQ[5]	Удерживает состояние выхода к флэш-памяти		–
CSN[3]	DQ[6]	–	Сигнал выбора кристалла, активный низким уровнем	–
CSN[2]	DQ[7]	–	Сигнал выбора кристалла, активный низким уровнем	–
CSN[1]	Неиспользованный	–	Сигнал выбора кристалла, активный низким уровнем	–
CSN[0]	Сигнал выбора кристалла, активный низким уровнем			–
CPU_CLK	Тактовая частота внешнего процессора	–	–	–
CONFIG_RSTN	Сигнал сброса конфигурации, активный низким уровнем			
CONFIG_DONE	«Конфигурация выполнена» — выход с открытым коллектором			
CONFIG_STATUS	«Инициализация SRAM выполнена» — выход с открытым коллектором			
CONFIG_MODESEL[2:0]	Выбор режима конфигурации должен быть 100	Выбор режима конфигурации должен быть 001	Выбор режима конфигурации должен быть 010	Выбор режима конфигурации. Не используемый в JFC, но на этих входах должны быть установлены 100, 001, 010 или 000
CONFIG_SYSCLK_BYPASS	Синхрочастота, при которой происходит обход конфигурации системы. Подключить к «0» или «1»	Синхрочастота, при которой происходит обход конфигурации системы. Установить в «0»		Синхрочастота, при которой происходит обход конфигурации системы. Подключить к «0» или «1»
CONFIG_CLKSEL	Выбор синхрочастоты конфигурации, установить в «0»			Подключить к «0» или «1»

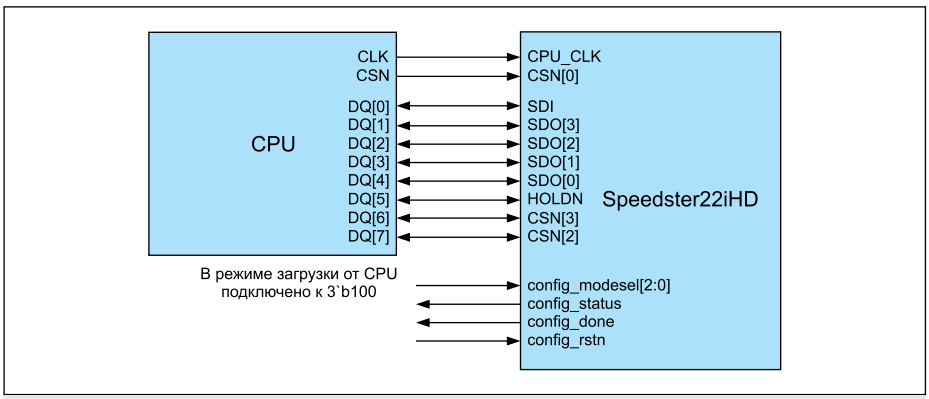

Рис. 4. Конфигурация ПЛИС внешним процессором, в этом варианте ПЛИС — ведомая

Как описано в разделе, посвященном последовательности включения питания и конфигурации, когда микросхема находится в режиме конфигурации, выполняются определенные этапы конфигурации, происходящие между установкой сигнала CONFIG_STATUS (указывает, что память конфигурации очищена и ПЛИС готова принять данные битового потока) и установкой сигнала CONFIG_DONE (указывает, что конфигурация завершена). На рис. 5 показаны диаграммы сигналов, на которых отмечена последовательность событий, тактовых сигналов и состояния управляющего сигнала, необходимых для успешной конфигурации ПЛИС от внешнего процессора:

1. После того как снимается сигнал CONFIG_RSTN, тактовая частота CPU_CLK должна продолжать вырабатываться из процессора. Это гарантирует, что ПЛИС пройдет через все состояния FCU и память конфигурации очистится. В данный момент времени сигнал CONFIG_STATUS, имеющий подпор к напряжению питания, устанавливается в высокий уровень.
2. Через некоторое время после установки высокого уровня сигнала CONFIG_STATUS сигнал CSN должен быть сброшен в низкий уровень, затем начинается запись данных из потока битовых данных в ПЛИС. Когда последнее слово данных запишется в ПЛИС, CSN должен быть установлен в высокий уровень.
3. После того как CSN устанавливается в высокий уровень, импульсы CPU_CLK должны выдаваться еще не менее чем 12 000 тактов. После 9000 тактов сигнал CONFIG_DONE должен установиться в высокий уровень, и это показывает то, что конфигурация завершилась, остальные 3000 тактов необходимы для того, чтобы FCU гарантированно перешел в пользовательский режим.

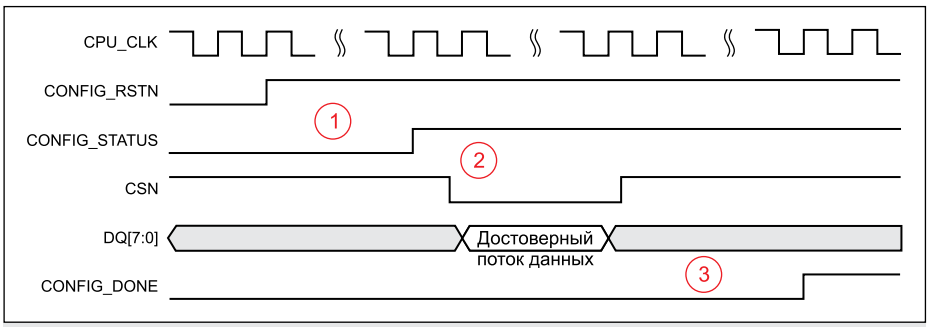

Рис. 5. Последовательность событий, тактовых сигналов и состояния управляющего сигнала, необходимых для успешной конфигурации ПЛИС от внешнего процессора

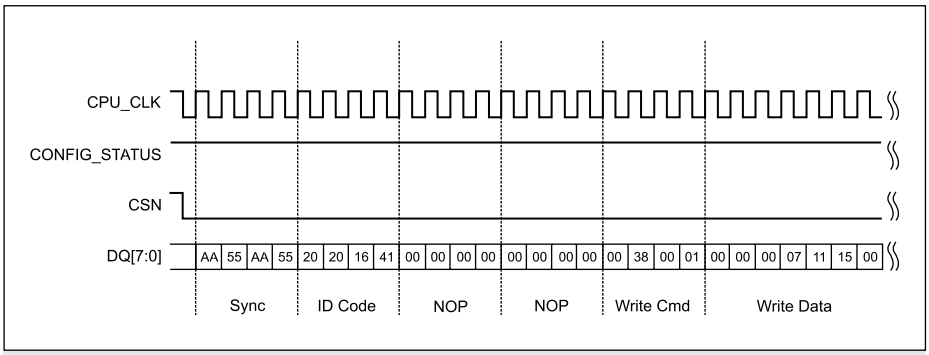

Рис. 6. Окно с частью процесса, происходящего при программировании битовым потоком данных конфигурации

На рис. 6 показано окно с частью процесса, происходящего при программировании битовым потоком данных конфигурации.

Конфигурация с помощью внешней последовательной флэш-памяти

Режим конфигурации SFC позволяет применять для конфигурации ПЛИС флэш-PROM с последовательным интерфейсом. В этом режиме ПЛИС является ведущим устройством, а потому выдает свою синхрочастоту в PROM. В таком режиме конфигурации применяется встроенный в ПЛИС блок быстрого чтения данных, который читает информацию из флэш-памяти по адресу 0. В блоке чтения данных имеется специальный регистр управления, куда записывается число, соответствующее числу слов, которые блок должен прочесть из битового потока данных. Это число слов также извлекается из битового потока данных. Кроме того, блок содержит управляющий контроллер, взаимодействующий через интерфейс с блоком JTAG и предназначенный для того, чтобы запрограммировать последовательную флэш-память. Команды по программированию флэш-памяти выдаются через JTAG.

Конфигурация в режиме работы с помощью одной внешней последовательной флэш-памяти подобна режиму конфигурации ПЛИС от внешнего процессора. Единственное отличие состоит в том, что во время записи потока битовых данных сигнал SCK используется для синхронизации, а поток битовых данных передается только по одному разряду интерфейса SDO [0]. Сигнал CSN [0] устанавливается в низкий уровень, что показывает достоверность передаваемого потока битовых данных, и устанавливается в высокий уровень, как только последний бит данных передан (простробирован тактовой частотой). Переход от конца потока битовых данных к пользовательскому режиму выполняется точно так же, как в режиме конфигурации ПЛИС от внешнего процессора, причем ПЛИС также выдает синхрочастоту по SCK.

На рис. 7 показана конфигурация ПЛИС из флэш-памяти с последовательным интерфейсом SPI.

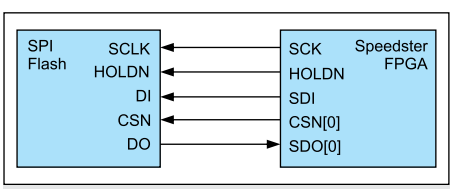

Рис. 7. Конфигурация ПЛИС из флэш-памяти с последовательным интерфейсом SPI

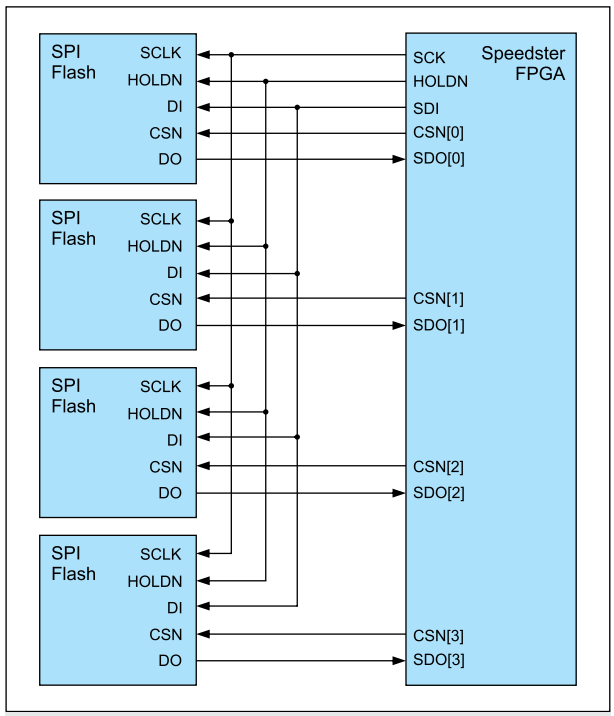

Рис. 8. Конфигурация ПЛИС из четырех внешних микросхем памяти с последовательным интерфейсом SPI

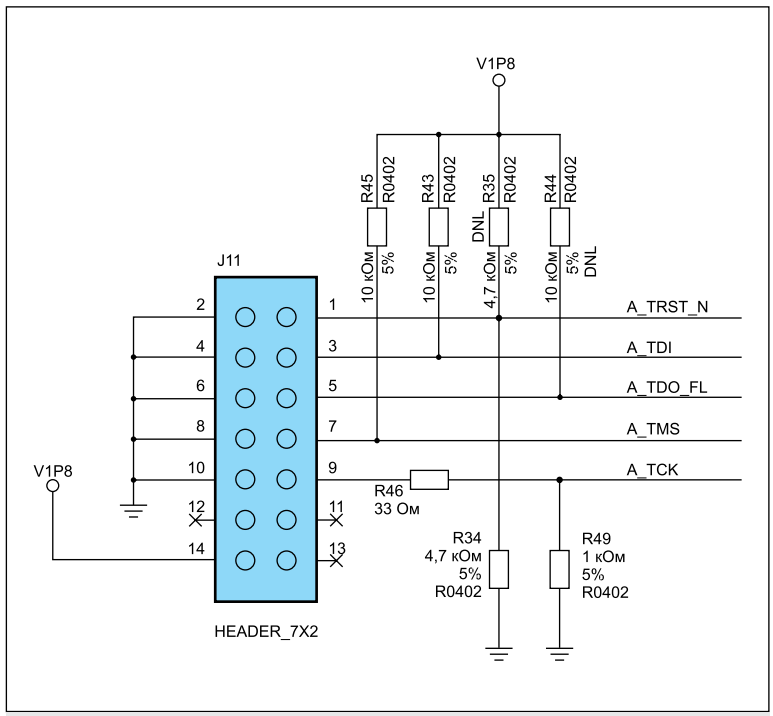

Рис. 10. Разъем JTAG, расположение контактов по разъему (напряжение, обозначенное как V1P8, соответствует 1,8 В)

Конфигурация ПЛИС из нескольких внешних микросхем памяти с последовательным интерфейсом SPI

Для того чтобы сократить время программирования и не применять сложный контроллер параллельной флэш-памяти, можно использовать четыре микросхемы флэш-памяти с интерфейсом SPI и производить из них загрузку одновременно.

Так же как и в режиме конфигурации с одной микросхемой памяти, в данном режиме ПЛИС является ведущим устройством и взаимодействует через интерфейс не с одной, а с четырьмя микросхемами флэш-памяти, что увеличивает полосу пропускания данных в четыре раза. На рис. 8 показана конфигурация ПЛИС из четырех внешних микросхем памяти с последовательным интерфейсом SPI. При работе от четырех микросхем флэш-памяти ПЛИС подключается к каждой микросхеме своим сигналом CSN.

При записи ПЛИС работает с микросхемами поочередно, выбирая нужную микросхему и устанавливая в низкий уровень требуемый сигнал CSN. При чтении все четыре сигнала CSN устанавливаются в низкий уровень, и через порты SDO одновременно производится чтение данных конфигурации, но, соответственно, по четырем линиям SDO чтение производится в 4 раза быстрее. Как только операции потока битовых данных завершаются, то есть все содержимое флэш-памяти прочитано, происходит переход от конца потока битовых данных к пользовательскому режиму точно так же, как в режимах конфигурации ПЛИС внешним процессором или в режиме конфигурации ПЛИС из одной внешней микросхемы памяти с последовательным интерфейсом.

Конфигурация ПЛИС по JTAG

Режим JFC позволяет выполнять конфигурацию ПЛИС непосредственно через кабель загрузки по JTAG. Этот режим обычно применяется разработчиками во время создания и тестирования своих проектов.

Конфигурация и работа по JTAG происходят независимо от того, как настроены входы режима CONFIG_MODESEL, хотя рекомендуется устанавливать на входах CONFIG_MODESEL только следующие значения: 100, 001, 010 или 000. Это позволит избежать неизвестных или недопустимых состояний контроллера загрузки.

Выводы TMS и TCK определяют, какая из операций будет выполняться: работа с регистром инструкций или с регистром данных. Сигналы TMS и TDI выбираются на положительном фронте TCK, в то время как TDO изменяется на отрицательном фронте.

Компания Achronix рекомендует использовать на платах пользователей JTAG-разъем, такой же, как и у «Битпортера».

«Битпортер» представляет собой устройство, предназначенное для программирования и загрузки конфигурации в микросхемы Speedster22i HD. Он программируется на языке STAPL, в нем предусмотрена возможность отладки и выполнения снимков Snapshot и отладки интерфейсов SerDes в PMA GUI.

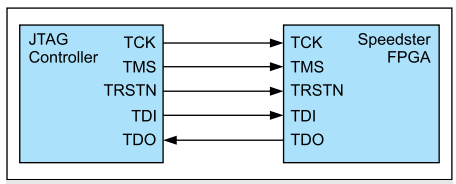

Рис. 9. Конфигурация ПЛИС по интерфейсу JTAG

Конфигурация по JTAG может быть сделана для микросхемы Speedster22iHD, которая станет единственным устройством в цепочке JTAG-сканирования или частью серии устройств, соединенных в общую цепочку. На рис. 9 показана конфигурация одной ПЛИС по интерфейсу JTAG. На рис. 10 представлен фрагмент схемы стартового набора — разъем для подключения JTAG и расположение сигналов на этом разъеме. На рис. 11 дана конфигурация одной ПЛИС по интерфейсу JTAG, но включенная в цепочку с еще двумя микросхемами.

На рис. 12 показана упрощенная блок-схема логики управления конфигурацией ПЛИС, а именно мультиплексор, предназначенный для выбора синхрочастоты конфигурации ПЛИС.

Алгоритм выбора построен таким образом, что вход конфигурации от JTAG имеет наивысший приоритет. Поэтому если аппаратные средства программирования соединяются с входами JTAG, управляющими конфигурацией (FCU), то в качестве синхрочастоты для конфигурации CFG_CLK автоматически выбирается TCK, причем это будет выполнено независимо от настроек управления конфигурацией. В режимах

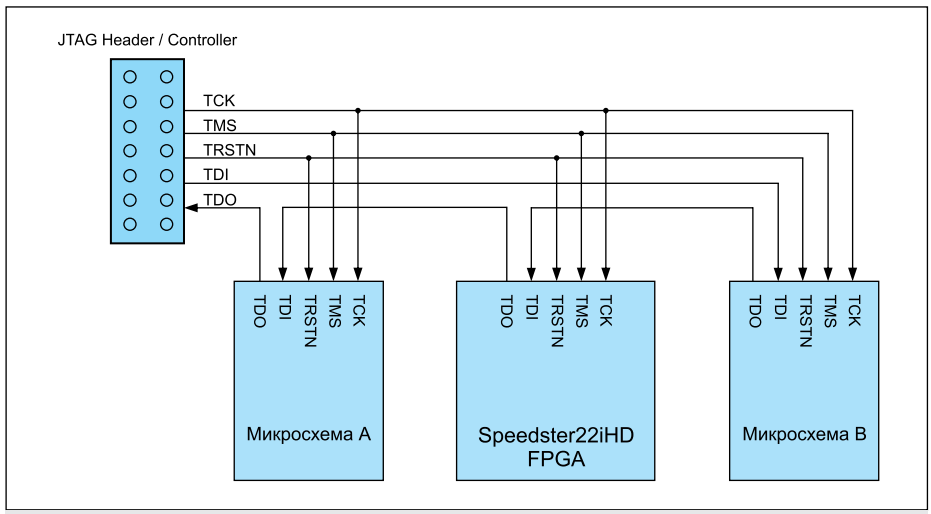

Рис. 11. Конфигурация ПЛИС по интерфейсу JTAG, но включенная в цепочку с еще двумя микросхемами

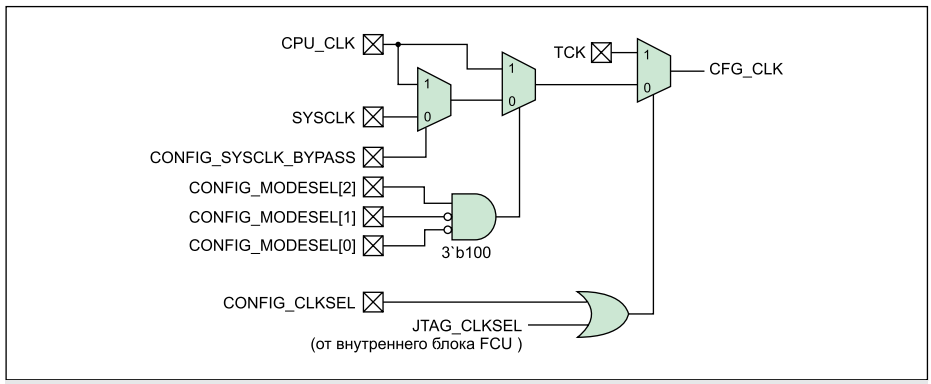

Рис. 12. Блок-схема мультиплексора, предназначенного для выбора синхрочастоты конфигурации ПЛИС

конфигурации ПЛИС из одной микросхемы или из нескольких микросхем флэш-памяти с последовательным интерфейсом SPI, если этот режим не переопределен программатором, подключенным к входам JTAG, для управления конфигурацией, то с помощью входа CONFIG_SYSCLK_BYPASS пользователи могут выбрать либо внутренне сгенерированную синхрочастоту SYSCLK, либо внешнюю синхрочастоту, например CPU_CLK. В таблице 5 приведены значения управляющих сигналов для выбора синхрочастоты конфигурации ПЛИС.

Необходимо отметить, что если программирование будет выполняться только в режиме конфигурирования по JTAG, важно понять, как управлять входами выбора синхрочастоты CONFIG_MODESEL. Чтобы очистить память конфигурации ПЛИС после включения питания или сброса микросхемы, должна быть выбрана синхрочастота, активная (не JTAG) в момент конфигурации, — тогда она сможет тактировать FCU. Например, если CONFIG_MODESEL устанавливается в 100 (таким образом, в качестве тактовой частоты до переопределения по JTAG выбирается CPU_CLK), то эта синхрочастота CPU_CLK должна быть переключена и должна существовать, что гарантирует корректную операцию загрузки.

Таблица 5. Установки сигналов выбора синхрочастоты конфигурации ПЛИС

CONFIG_SYS_CLK_BYPASS	CONFIG_CLKSEL	CONFIG MODESEL [2:0]	FCU CLK
0	0	001, 010	On-chip oscillator
1	0	001, 010	CPU_CLK
0/1	0	100	CPU_CLK
0	1	000, 001, 010, 100	TCK

Устройство для загрузки «Битпортер»

Для загрузки битовой последовательности из хост-компьютера в микросхему компания Achronix предлагает аппаратное устройство — программатор «Битпортер» (Bitporter). На рис. 13 он показан со стороны разъемов подключения к хосту. «Битпортер» соединяется с хо-

Рис. 13. «Битпортер» со стороны разъемов подключения к хосту

Рис. 14. Подключение «Битпортера» к стартовому набору (слева плоский кабель идет к порту JTAG на целевой системе, а справа USB-кабель направлен к хосту)

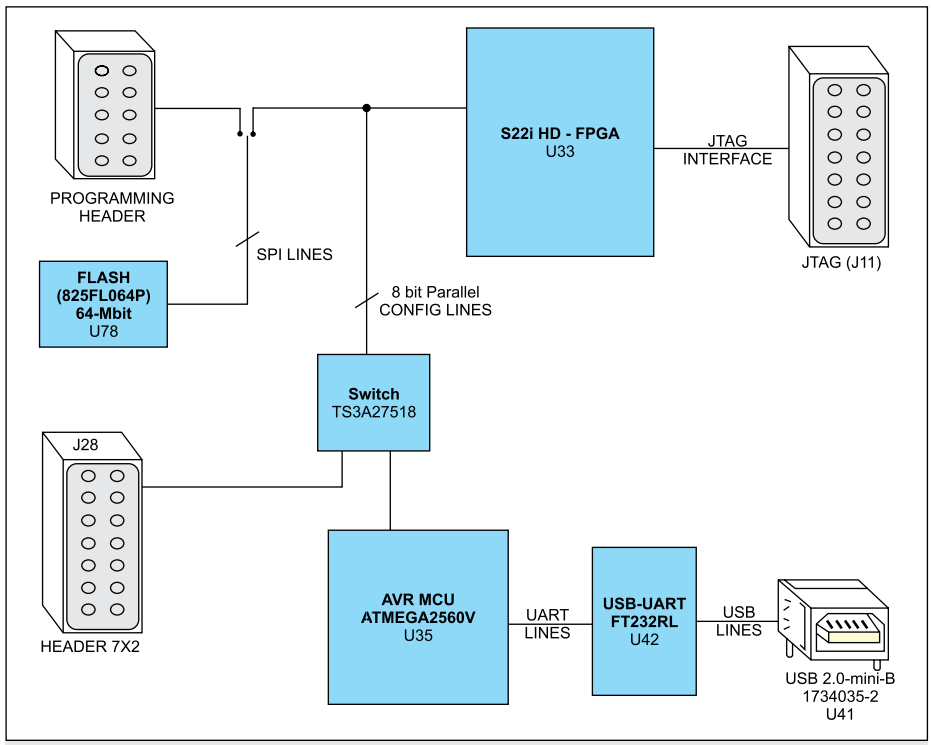

Рис. 15. Блок-схема части стартового набора, на которой показаны различные источники получения данных для конфигурации ПЛИС

стом или через 10/100 Ethernet, или через USB 2.0. Подключение «Битпортера» через 10/100 Ethernet позволяет вести удаленную загрузку и отладку.

На другой стороне «Битпортера» находится разъем для подключения к порту JTAG на целевой системе. Подключение осуществляется гибким ленточным кабелем так, как это видно на рис. 14. На рисунке показано подключение «Битпортера» к стартовому набору. Слева — плоский кабель идет к порту JTAG на целевой системе, а справа — USB-кабель направлен к хосту. Когда «Битпортер» подсоединяется к порту JTAG, то с его помощью можно выполнять конфигурацию микросхемы, отладку проектов и программирование флэш-памяти.

«Битпортер» представляет собой STAPL Player (acx_stapl_player), то есть он получает поток битовых данных от инструментального программного обеспечения Achronix ACE в формате языка STAPL, а затем выполняет эту программу на STAPL. Под управлением программы «Битпортер» преобразует коды STAPL в поток битовых данных, в котором присутствуют команды и данные, необходимые для загрузки в порт JTAG в ПЛИС.

Для того чтобы пользователь смог проверить и изучить различные режимы конфигурации ПЛИС Speedster22i HD, компания Achronix сделала стартовый набор, в котором предусмотрена возможность производить конфигурацию ПЛИС всеми требуемыми способами, а именно по интерфейсу JTAG, из флэш-памяти с последовательным интерфейсом SPI и внешним процессором. В качестве внешнего процессора применен процессор ATMEGA2560V. Причем он может быть подключен к хосту через имеющийся на плате стартового набора переходник COM-USB. Также процессор имеет свой интерфейс JTAG, что позволяет производить его отладку и программирование. В результате пользователь может реализовать нужный ему способ конфигурации ПЛИС.

На рис. 15 приведена блок-схема части стартового набора, на которой показаны различные источники получения данных для конфигурации ПЛИС.

Краткое описание STAPL

Введение в STAPL

Название языка STAPL (Standard Test And Programming Language) расшифровывается как «стандартный язык тестирования и программирования». Он был разработан для того, чтобы поддерживать программирование, конфигурирование и тестирование электронных систем, используя интерфейс по стандарту IEEE 1149.1: Standard Test Access Port and Boundary Scan Architecture (обычно называемый JTAG). Когда выполняется файл STAPL, то в интерфейсе JTAG производится запись и считывание сигналов так, как описано в файле STAPL. STAPL работает на одной цепочке IEEE 1149.1. STAPL может работать с любым IEEE 1149.1-совместимым программируемым устройством. Следует сказать, что по отношению к командам, выполняемым в интерфейсе JTAG, язык STAPL является языком высокого уровня.

На STAPL можно проводить программирование и тестирование систем с функциями пользовательского интерфейса. Один файл STAPL способен выполнить несколько различных функций, таких как программирование, проверка и стирание программируемого устройства. В файле STAPL можно делать эти высокоуровневые функции доступными через операторы языка, соответствующие управлению работой в интерактивной системе. STAPL также применяется в системах, не имеющих функций пользовательского интерфейса, например во встроенных системах.

STAPL может быть реализован или как интерпретируемый, или как компилируемый язык. Интерпретируемая реализация означает, что файлы STAPL не компилируются в двоичный исполняемый код, а выполняются непосредственно программой интерпретатора. Скомпилированная реализация означает, что файл STAPL сначала предварительно обрабатывается и затем выполняется.

Файл STAPL состоит из последовательности операторов программы. Оператор STAPL состоит из метки, которая является дополнительной, инструкций и параметров и завершается точкой с запятой (;). Каждый оператор обычно занимает одну строку файла STAPL, но синтаксис языка не воспринимает символов перевода строки, за исключением завершения комментариев. Символ апострофа (') показывает, что в этой строке находится комментарий, который игнорируется компилятором и интерпретатором.

В качестве примера программы на языке STAPL можно привести операцию чтения IDCODE от одного устройства в JTAG-цепочке (листинг).

На рис. 16 показано, как примерно будет выглядеть диаграмма сигналов чтения ID-микросхемы по интерфейсу JTAG, соответствующая коду, приведенному в листинге.

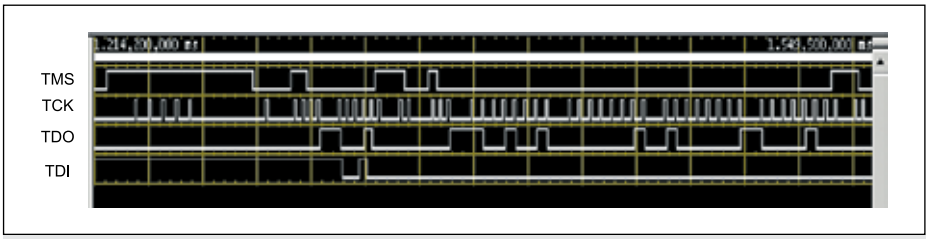

Рис. 16. Примерно так выглядит диаграмма сигналов чтения ID-микросхемы, соответствующая коду, приведенному в листинге 1

Просто, как было написано выше, все проигрыватели являются аппаратно-зависимыми. И все «существенно-значимые» фронты сигналов, конечно, будут иметь такие же соотношения, как и на приведенном рисунке, но «не значимые» фронты сигналов могут быть несколько сдвинуты. Таким образом, как видно из приведенного примера на языке STAPL, можно достаточно легко закодировать довольно длительную последовательность сигналов в интерфейсе JTAG.

```
NOTE "CREATOR" "AAAA Tool Version 1.0"
NOTE "DEVICE" "ABCD1234";
NOTE "DATE" "1997/12/31";
NOTE "STAPL_VERSION" "JEDS00-A";
NOTE "ALG_VERSION" "3";
NOTE "STACK_DEPTH" "2";
NOTE "MAX_FREQ" "10000000"; '10MHz
NOTE "TARGET" "1";
NOTE "IDCODE" "00000001";

ACTION READ_IDCODE = DO_READ_IDCODE;

PROCEDURE DO_READ_IDCODE;

'Declare variables for data arrays
BOOLEAN read_data[32];
BOOLEAN i_idcode[10] = #1001101000;
BOOLEAN ones_data[32] = $FFFFFFFF;

INTEGER i;

'Initialize device
STATE RESET;

'Load idcode instruction
IRSCAN 10, i_idcode[9..0];
'Capture idcode
DRSCAN 32, ones_data[31..0], CAPTURE read_data[31..0];

EXPORT "IDCODE", read_data[31..0];

ENDPROC;
CRC 3759;
```

Листинг. Пример программы на языке STAPL — чтение IDCODE от одного устройства в JTAG-цепочке

Создание и интерпретация файлов STAPL

Для языка STAPL существуют две программы: «композитор» (composer) и «проигрыватель» (player). Программа «композитор» предназначена для записи информации в файл, а «проигрыватель» производит интерпретацию файлов STAPL. На рис. 17 показана взаимосвязь программ и их функции.

«Композитор» STAPL генерирует файлы STAPL в соответствии со спецификацией языка. Он может быть реализован как автономная утилита или как часть интегрированного программного средства.

«Композитор» STAPL создает файлы STAPL со многими различными типами информации. Файлы STAPL могут содержать любые типы данных:

- Данные проектирования (единственное устройство или цепочка устройств по JTAG).
- Информацию об алгоритме ISP-программирования.
- Конфигурацию цепочки IEEE 1149.1.
- Тестовые векторы.
- Любые инструкции по стандарту IEEE 1149.1.
- Параллельные векторы, используя инструкции расширения VECTOR/VMAP.

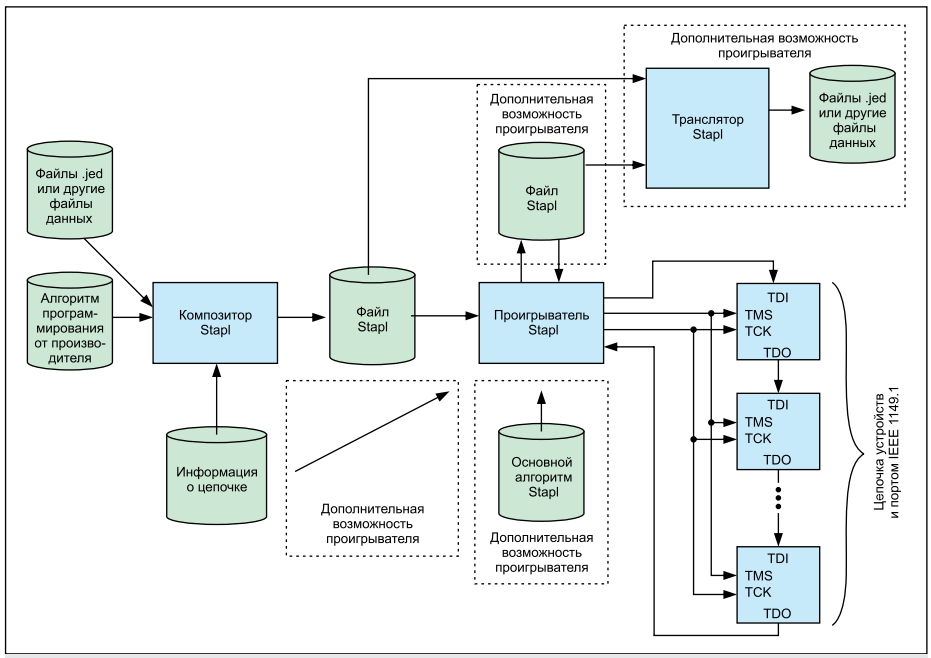

Рис. 17. Блок-схема, показывающая взаимосвязь программ «композитор» и «проигрыватель» и их функции

Например, в том случае, когда требуется запрограммировать некоторое устройство, «композитор» может создать файлы STAPL, чье содержание будет соответствовать именно той аппаратной платформе, для которой предназначается файл STAPL. Файл STAPL, создаваемый для программирования устройства через встроенный процессор, может содержать данные проекта для этого устройства в цепочке, алгоритмы программирования для каждого устройства и информацию о конфигурации цепочки. Файл, создаваемый для того, чтобы запрограммировать тестер граничного сканирования на основе стандарта IEEE 1149.1, может содержать данные проекта, алгоритмы программирования и алгоритмы тестирования, причем для этих тестов «проигрывателю» STAPL должна быть определена информация о цепочке.

В интерпретируемой реализации операторы файла STAPL выполняются непосредственно без первой компиляции этих операторов в двоичный исполняемый код. Программу интерпретатора, которая читает и выполняет файл STAPL, называют «проигрывателем» STAPL.

«Проигрыватель» STAPL является программой, зависимой от аппаратной платформы. Например, в одном случае чтение файла STAPL может производиться с помощью файловой системы, а в другом — этот файл может быть просто считан из буфера памяти как символы. У «проигрывателя» STAPL есть доступ к сигналам IEEE 1149.1 — TMS, TCK, TDO, TDI, которые применяются для всех инструкций, основанных на интерфейсе IEEE 1149.1. Если при выполнении файла STAPL требуется использование инструкций VECTOR И VMAP, то возможно, что «проигрывателю» STAPL понадобится доступ к нескольким другим сигнальным контактам. Аппаратный доступ к сигналам по IEEE 1149.1 и к другим сигнальным контактам зависит от аппаратной платформы.

Если в системе, где работает «проигрыватель» STAPL, есть консоль или хост-устройство, оно может использоваться для вывода на экран сообщений, сгенерированных файлом STAPL.

Заключение

Итак, мы ознакомились со второй статьей, посвященной новым микросхемам семейства Speedster22i HD, работающим на максимальной частоте 750 МГц и имеющим 1,7 млн LUT. Они выполнены по 22-нм техпроцессу на фабрике Intel и предназначены для Hi-End-устройств.

В этой части более подробно рассмотрены вопросы конфигурации микросхем из нескольких источников, а именно по интерфейсу JTAG, из флэш-памяти с последовательным интерфейсом SPI и внешним процессором. Было представлено краткое описание аппаратного устройства для загрузки — «Битпортера» от компании Achronix, а также очень кратко дано описание программ, работающих с языком STAPL.

В следующих статьях цикла мы рассмотрим основную «изюминку» ПЛИС Speedster — аппаратные IP-ядра, цепи ввода/вывода и средства разработки и стартовые наборы.

Литература

1. Руководство пользователя Speedster22i Pin Connections and Power Supply Sequencing.