ГЛАВА 7

РАЗЛИЧНЫЕ ПОДХОДЫ К ПРОЕКТИРОВАНИЮ СИСТЕМ НА ОСНОВЕ ПЛИС И ЗАКАЗНЫХ МИКРОСХЕМ

Введение

Моя мама очень гордится тем, что её сын стал инженером в области электронных систем. И эта гордость зиждется на абсолютно непоколебимой уверенности в том, что я могу все: создать, понять и отремонтировать любое электронное устройство, какой бы эпохе оно ни принадлежало и где бы оно ни находилось на нашей планете. В реальности, конечно, дела обстоят намного скромнее, так как очень немногие из нас становятся асами во всех областях науки электроники[1].

В то время как одни тратят свои лучшие годы на разработку, по всей видимости бесконечной, серии заказных микросхем, другие изнывают в своих офисах, постигая тайны области применения ПЛИС. Проблема становится очевидной, когда инженер, владеющий в совершенстве одной из этих технологий, неожиданно вынужден все бросить и освоить другие технологические процессы. Типичная картина для наших дней, когда разработчики, которых буквально распирает от гордости и уверенности, что они знают всё о микросхемах, вдруг получают задание создать устройство, реализованное на основе ПЛИС.

Это довольно трудная глава ввиду многогранности обсуждаемых в ней проблем. Самое оптимальное, что можно сделать в подобной ситуации, — ограничиться обзором наиболее важных отличий подходов к проектированию систем на основе ПЛИС и систем на основе заказных микросхемах (ASIC).

Подходы к проектированию

Когда дело доходит до представления устройства с помощью одного из языков описания аппаратных средств (*см.* гл. 9), инженеры, работающие с заказными микросхемами, предпочитают написать портативный код на языке VHDL или Verilog и свести к минимуму конкретизацию ячеек.

В отличие от них разработчики ПЛИС-систем предпочитают работать с конкретными ячейками. Если разработчиков ПЛИС-систем, например, не устраивают результаты работы средств синтеза, создающих нечто вроде мультиплексора, они могут вручную создать собственную версию и включить её в состав кода. Искушенные разработчики систем на основе ПЛИС применяют, как правило, более или менее технологически зависимые решения при синтезе, чем их коллеги, работающие с заказными микросхемами.

---

[1] Совсем недавно мне пришлось столкнуться со старым Wortsel Grinder Mark 4, у которого был ажурный корпус и зеркальная зигзагообразная панель. Я и понятия не имел, с какой стороны к нему подступиться. Даже трудно представить, насколько беспомощным я казался себе.

1854 г. *Крым. В крымской войне используется телеграф.*

Конвейеры и уровни логики

Что такое конвейер

Слово «*конвейер*» довольно часто употребляется. Однако очень редко можно встретить его определение. Несомненно, инженеры прекрасно знают, что значит этот термин, но поскольку эта книга рассчитана на широкий круг читателей, уделим этому вопросу некоторое время, чтобы быть уверенными, что все танцуют степ в одном и том же ритме[1]. Предположим, что мы строим нечто вроде автомобиля и все части этого автомобиля находятся у нас под руками. Предположим также, что весь процесс сборки состоит из следующих пяти этапов:

1. Присоединение колес к шасси.
2. Присоединение двигателя к шасси.
3. Присоединение сидений к шасси.
4. Присоединение кузова к шасси.
5. Покраска.

Да... знаю, знаю. Инженеры сразу начнут упрекать меня в том, что в этом списке отсутствует рулевое управление, освещение и много другое. Но, господа, это же всего лишь пример!

Допустим, что для выполнения каждой из перечисленных операций потребуется один специалист. Один из способов выполнения сборки заключается в том, чтобы рассадить всех специалистов в круг для игры в карты. Первый парень (или девушка, так как любая дискриминация исключается)[2] встает, присоединяет колёса к шасси и возвращается к игре. После его (или её) возвращения второй парень встает, устанавливает двигатель и возвращается к игре. Теперь третий парень встает прогуляться, чтобы присоединить сиденья к шасси. После его возвращения четвертый парень подходит к шасси, чтобы установить на него кузов и так далее. После того как машина будет собрана, начинается всё сначала.

Очевидно, что это очень нерациональный подход. Если, например, каждая операция занимает один час, весь процесс будет длиться пять часов. Кроме того, в течение каждого часа только один человек будет работать, в то время как четверо других будут слоняться без дела. Очевидно, что более эффективной будет сборка одновременно пяти автомобилей на сборочной линии. Тогда, как только первый парень присоединит колеса к первому шасси, второй парень начнет устанавливать двигатель на это шасси, а в это время первый парень начнет устанавливать колеса на второе шасси. Следовательно, сборочная линия будет полностью заполнена, и каждый работник будет занять все рабочее время, и новые автомобили будут сходить с конвейера каждый час.

Конвейер в электронных системах

Дело в том, что схему конвейера можно применить во многих электронных системах. Предположим, например, что имеется устройство, или функциональный блок устройства, которое может быть реализо-

---

[1] В молодости, перед Второй мировой войной, мой отец со своим братом исполняли чечётку в варьете-холлах Англии. Держу пари, они не ожидали найти упоминание об этом факте в книге по электронике, написанной в 21-м веке.

[2] Женщины сейчас очень эмансипированы и при описании любого процесса о них нельзя не упоминать, исключая ситуации, когда такая трактовка противоречит правилам грамматики.

вано в виде последовательно соединенных блоков комбинационной, или комбинаторной, логики (**Рис. 7.1**).

Предположим также, что каждый блок требует Y наносекунд для выполнения своей задачи. Допусти, что имеется пять таких блоков (на **Рис. 7.1** показано только три). В этом случае понадобится 5×Y наносекунд для передачи слова данных через эту функцию, начиная с достижения им входов первого блока и кончая его уходом с выходов последнего блока.

1855 г. *Англия. Джеймс Клерк Максвелл (James Clerk Maxwell) математически описал открытые Фарадеем силовые линии поля.*

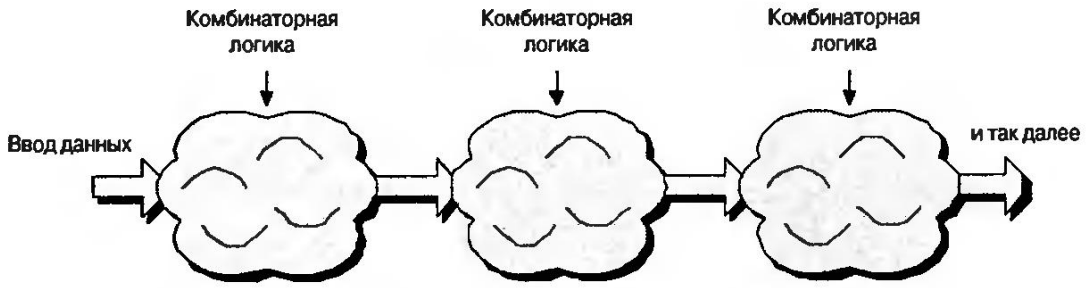


Рис. 7.1. Функция, реализованная с помощью комбинаторной логики

В такой ситуации не хотелось бы выставлять новые данные на входы блока до тех пор, пока не будет сохранен результат, относящийся к первому слову данных. Это значит, что возникает та же ситуация, которая имела место при неэффективной сборке автомобиля. Другими словами, потребуется много времени для обработки каждого слова данных, и большинство «рабочих», т. е. логических блоков, основную часть времени будут простаивать без работы. Проблема решается методом конвейерной обработки, в котором «острова» комбинационной логики чередуются с блоками регистров (**Рис. 7.2**).

Существует метод волнового *конвейера*, когда через логику одновременно проходят несколько «волн» данных. Однако их рассмотрение выходит за рамки этой книги, и, к тому же, этот метод не применяется в ПЛИС.

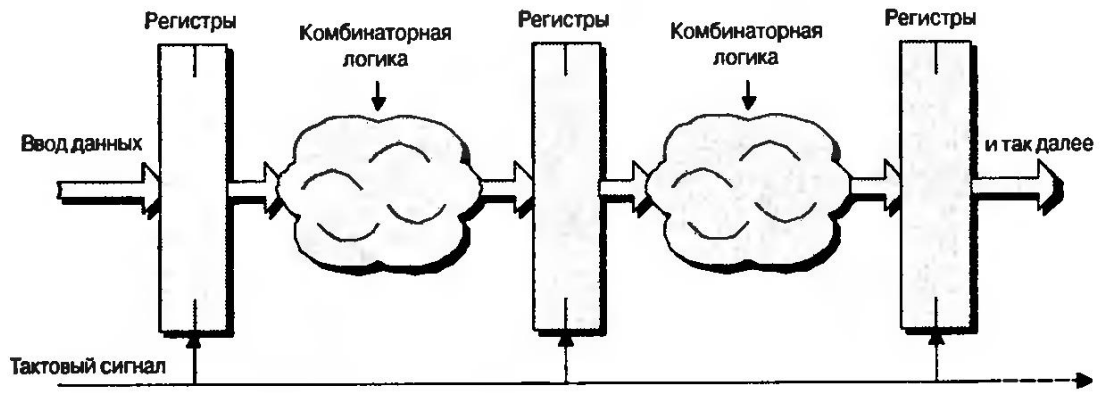


Рис. 7.2. Конвейерная конструкция

Все регистры синхронизируются с помощью общего тактового сигнала. С приходом каждого синхроимпульса результаты с предыдущего блока загружаются в регистры, подключенные к входам логических блоков. Эти значения затем передаются через этот логический блок на его выходы, где они будут готовы для загрузки в следующий регистр на следующем такте.

При такой схеме, после того как «насос будет запущен» и конвейер полностью заполнен, новые слова данных могут поступать через каждые Y наносекунд.

1858 г. *Америка. Агенты судоходной компании Cunard послали первую коммерческую телеграмму о столкновении двух пароходов.*

Логические уровни

Всё вышесказанное сводится к тому, что инженерам следует соблюдать определённый баланс. Разделение комбинационной логики на меньшие блоки и увеличение количества регистров приведет к росту производительности устройства и одновременно к увеличению потребления ресурсов микросхемы, в том числе и места на кристалле, что увеличит *задержку, или латентность*, устройства.

В электронных системах термин «*латентность*» означает время, (или количество тактовых импульсов) необходимое для прохождения блока данных через функцию, устройство или систему.
Чтобы лучше понять, что такое *латентность*, рассмотрим еще раз пример конвейерной сборки автомобиля. В этом случае пропускная способность системы может составлять один автомобиль в минуту, а *латентность* системы может составлять 8 часов, в течение которых выполняются сотни операций по сборке автомобиля (каждая такая операция соответствует одному регистру в конвейерной цепочке).

Рассмотрим концепцию *логического уровня*, т. е. количество логических элементов между входами и выходами логического блока. Так, например, на **Рис. 7.3** показано три логически уровня, так как перед тем как достичь выхода сигнал, в наихудшем случае, пройдет через три логических элемента.

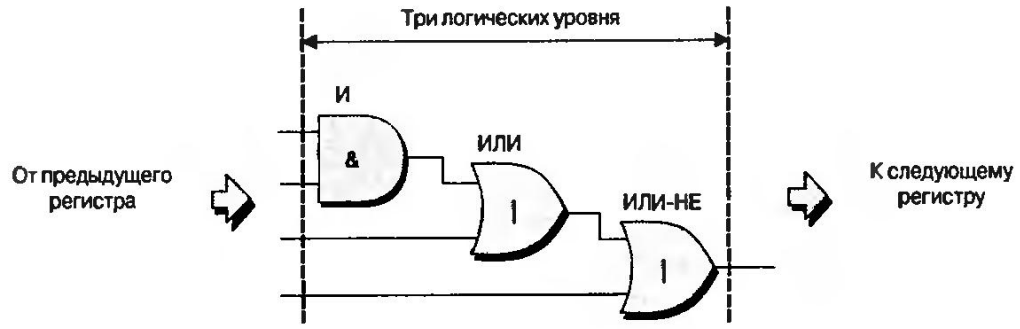


Рис. 7.3. Логические уровни

При использовании заказных микросхем логические элементы, как показано на **Рис. 7.3**, могут быть размещены в непосредственной близости друг к другу, т. е. задержка сигнала на проводниках между ними будет минимальной. Следовательно, если позволяет устройство, разработчики заказных микросхем могут иногда позволить себе некоторую фривольность с задачами подобно рода. Однако она не заставить себя долго ждать и сразу проявится, если сигналу предстоит пройти путь через, скажем, 15 и более логических уровней.

При реализации этого устройства на ПЛИС, в которых каждый логический вентиль реализуется в виде отдельной таблицы соответствия (LUT), сигнал будет двигаться с черепашьей скоростью, так как задержка на проводниках в ПЛИС существенно больше, чем у заказных микросхем. Конечно, на практике таблица соответствия может объединять несколько логических блоков (функция, показанная на **Рис. 7.3**, может быть реализована в одной 4-входовой таблице соответствия), так что ситуация не столь ужасна, как это могло показаться на первый взгляд.

Чтобы поднять, или удержать, производительность устройств на основе ПЛИС, они должны быть более конвейеризованы, чем их аналоги на заказных микросхемах. Задача облегчается тем, что каждая логическая ячейка в ПЛИС содержит таблицу соответствия и регистр.

Метод асинхронного проектирования

1858 г. *Атлантика. Проложен первый трансатлантический телеграфный кабель. Впоследствии он вышел из строя.*

Асинхронные элементы

При решении некоторых задач разработчики заказных микросхем, иногда добавляют в свои схемы асинхронные элементы, полагаясь на относительную задержку распространения сигналов. Этот подход не работает по отношению к ПЛИС, так как разводка и связанные с ней задержки сигналов могут сильно изменяться после каждого нового прогона трассировщика.

Комбинационные петли

Этот подраздел в какой-то мере связан с предыдущим. Комбинационные петли возникают в схемах, в которых сигнал формируется с помощью цепей обратной связи, проходящих через один или более логический вентиль[1]. Они являются главным источником так называемых *критических состояний гонки*, при которых логические значения зависят от задержек, определяемых разводкой элементов внутри микросхемы. Хотя подобный подход осуждается в некоторых кругах, но в случае заказных микросхемах разработчики могут себе позволить подобную вольность, поскольку они в состоянии точно определить маршрут дорожки между элементами и, соответственно, связанную с ним задержку. В случае ПЛИС это невозможно, так как такие цепи обратной связи должны содержать регистры.

Элементы задержки

Для создания *элементов задержки* в заказных микросхемах инженеры используют последовательность буферов или инвертирующих вентилей. Эти элементы могут использоваться в различных целях, например для определения состояния гонки в асинхронных блоках устройства. При использовании таких элементов в ПЛИС довольно трудно прогнозировать величину задержки, к тому же, эти элементы увеличивают чувствительность устройства к условиям эксплуатации, уменьшают её надежность и могут стать источником проблем при переходе на другую архитектуру или технологию реализации.

Анализ систем синхронизации

Зоны синхронизации

Заказные микросхемы могут использовать огромное количество синхросигналов. Кто-то слышал, что существуют даже устройства с более чем 300 различными зонами (доменами) синхронизации. Однако на этот счет у ПЛИС существует ограничения: разработчикам настоятельно не рекомендуется выходить за рамки ресурсов, выделенных для каждого конкретного устройства на обеспечение синхронизации. Другими словами, не надо использовать выводы общего назначения для подачи на микросхему пользовательских тактовых сигналов.

---

[1] Конечно, каждая защёлка содержит цепь внутренней обратной связи, и каждый триггер формируется из двух защёлок. Но эти обратные связи очень жестко контролируются производителем микросхем.

1858 г. *Королева Виктория отправила трансатлантическую телеграмму президенту США Джеймсу Бьюкенену.*

Некоторые ПЛИС позволяют разделять свои деревья синхронизации на сегменты. Если такая возможность поддерживается используемой технологией, следует определить сегменты синхронизации и выделить их соответственно под внешние и внутренние тактовые сигналы.

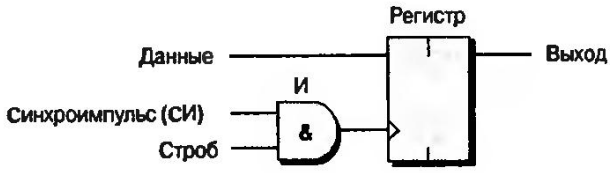
Выравнивание тактовых сигналов

В устройствах на заказных микросхемах необходимо использовать специальные методы для выравнивания задержек тактовых сигналов, распространяющихся по всему устройству. В отличие от заказных микросхем, ПЛИС обладают небольшим фазовым сдвигам по всей поверхности кристалла. Другими словами, нет необходимости выравнивать тактовые сигналы, поскольку производители ПЛИС уже позаботились об этом.

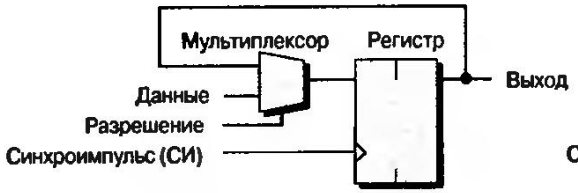
Стробирование и разрешение тактовых сигналов

В устройствах на заказных микросхемах часто используется концепция *стробирования тактовых сигналов*, что позволяет уменьшить рассеиваемую мощность (**Рис. 7.4, а**). Однако при подобных подходах устройство приобретает асинхронные свойства и становится чувствительным к выбросам импульсов, которые возникают при переключении входов, расположенных рядом с логикой стробирования.

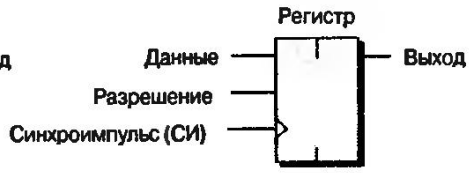
В отличие от заказных микросхем (ASIC) разработчики ПЛИС используют концепцию *разрешения тактовых сигналов*. Раньше разрешение реализовывалось с помощью отдельного мультиплексора (**Рис. 7.4, б**), теперь в регистрах ПЛИС почти любой архитектуры предусмотрен отдельный специальный вход разрешения тактовых сигналов (**Рис. 7.4, в**).



а) Стробирование синхроимпульсов



б) Разрешение тактовых сигналов (старый вариант)



в) Разрешение тактовых сигналов (новый вариант)

Рис. 7.4. Стробирование и разрешение тактовых сигналов

ФАПЧ и схема согласования тактовых сигналов

ПЛИС обычно содержат схемы фазовой автоподстройки частоты (ФАПЧ) или схемы автоматической подстройки по задержке: одну для каждого встроенного общего тактового сигнала *(см.* гл. 4). Если эти схемы используются для генерации тактовых сигналов внутри микросхемы, её конструкция должна предусматривать некий механизм отключения или шунтирования функций автоподстройки в режимах тестирования или отладки.

Достоверность передачи данных через разные зоны синхронизации

Вопрос достоверности передачи данных имеет отношение как к заказным микросхемам, так и к ПЛИС. Дело в том, что обмен данными между двумя независимыми зонами синхронизации должен выполняться предельно аккуратно, чтобы исключить потерю или искажение информации. Плохая синхронизация может привести к нестабильным результатам и проблемам при временном анализе. Для достижения надежной передачи данных через зоны с различной синхронизацией рекомендуется использовать квитирование, буферизацию или асинхронные очереди.

1859 г. *Германия. Гитторф и Паккер изобрели трубку на катодных лучах.*

Регистры и защелки

Защелки

В устройствах на основе заказных микросхем инженеры часто используют защёлки. Основываясь на опыте проб и ошибок, могу сказать, что если речь идет о проектировании системы на основе ПЛИС с использованием защелок — не делайте этого!

Триггеры с входами установки и сброса

Многие библиотеки заказных микросхем (ASIC) содержат различные триггеры, в том числе с входами *установки (set)* и *сброса (reset)*. Обычно доступны как синхронные, так и асинхронные варианты.

В отличие от ASIC в ПЛИС триггеры могут быть сконфигурированы с одним входом: либо для установки, либо для сброса. В этом случае реализация одновременно двух входов потребует использования таблицы соответствия. Часто разработчики, пытаясь решить эту проблему, предлагают свои альтернативные решения.

Общий сброс и исходное состояние

Каждый регистр ПЛИС программируется определенным начальным состоянием: либо логическим 0, либо логической 1. Кроме того, ПЛИС обычно имеют сигнал *общего сброса*, который применяется для возвращения всех регистров, но не блоков встроенного ОЗУ, в исходное состояние. Разработчики, использующие заказные микросхемы, обычно не используют подобную возможность.

Разделение ресурсов или разделение по времени

Разделение ресурсов — метод оптимизации, позволяющий использовать один функциональный блок, например сумматор или компаратор, для реализации нескольких операций. Например, умножитель сначала может использоваться для обработки двух чисел, скажем ***А*** и ***Б***. Затем тот же умножитель может использоваться для обработки двух других чисел, скажем ***В*** и ***Г***. Хороший пример разделение ресурсов рассматривается в гл. 12.

Разделение ресурсов также называют *разделением*, или *распределением, по времени* либо *мультиплексированием по времени*. В ПЛИС количество ресурсов более ограничено, чем в ASIC, поэтому разработчи-

ки ПЛИС в отличие от своих коллег, работающих с заказными микросхемами, вынуждены затрачивать определённые усилия на реализацию распределения ресурсов.

В области связи понятие «мультиплексирование по времени» означает объединение нескольких медленных потоков данных в один скоростной поток путём выделения каждому медленному потоку непродолжительного временного интервала для передачи своих данных.

В области разделения ресурсов временное мультиплексирование относится к общему использованию ресурсов, например, умножителя, входы которого переключаются и позволяют в различные моменты времени обрабатывать разные данные.

Использовать или потерять!

На самом деле в случае ПЛИС всё обстоит немного сложнее, чем описывалось выше, так как существует фундаментальная дилемма «использовать или потерять», относящаяся к аппаратным ресурсам микросхемы. Это значит, что ПЛИС поставляются только с фиксированным набором элементов. Другими словами, если пользователю не подходит микросхема меньшего размера, он может использовать любую доступную ему большего размера, при этом реализовав только часть её ресурсов.

Допустим, что речь идет о разработке устройства, для которого требуется два встроенных аппаратных процессорных ядер. В этом случае возможно решение, согласно которому для функционирования системы с помощью разделения ресурсов с учетом имеющихся двух процессоров можно обойтись 10-ю умножителями и 2-я мегабайтами оперативной памяти. Но, если ПЛИС с двумя процессорами содержит 50 умножителей и 10 мегабайт памяти, придется оплатить их полную стоимость. При этом избыточные средства в виде дополнительных умножителей и блоков памяти, в которых на данный момент нет необходимости, могут быть использованы для повышения функциональности системы.

Подождите, ещё не вечер!

В ПЛИС обмен данными между таблицами соответствия (или логическими блоками) и специальными компонентами, такими как умножители, обычно является более дорогостоящим (из-за связности), чем соединение с другой таблицей соответствия (или логическим блоком). Процедура разделения ресурсов увеличивает значения связности, поэтому этот процесс следует держать под контролем.

Распределение ресурсов оправдано не только в случае таких больших компонентов, как умножители или функции умножения с накоплением, но и в случае, например, сумматоров. Интересно, что при технологии с использованием схем переноса, которые реализуются в устройствах компаний Altera и Xilinx, стоимость построения сумматора, в первом приближении, почти эквивалента стоимости построения шины данных с разделяемой логикой. Например, реализация двух сумматоров с полностью независимыми входами и выходами будет стоить двух сумматоров, при этом не потребуется реализация ресурсоразделяющих мультиплексоров. При разделении ресурсов получим один сумматор и два мультиплексора, по одному для каждого набора входов.

В случае ПЛИС это будет более дорогой вариант; в случае ASIC стоимость мультиплексора намного меньше стоимости сумматора.

На практике соотношение между дилеммой «использовать или потерять» и стоимостью соединений зависит от технологии и конкретной ситуации, при этом также следует иметь в виду, что устройства компании Altera отличаются от устройств компании Xilinx.

Кодирование конечных автоматов

Схема кодирования конечных автоматов является примером области, которая очень хорошо реализуется с помощью заказных микросхем, а решение на основе ПЛИС может оказаться не вполне удачным.

Однако конечные автоматы с успехом могут быть реализованы и на ПЛИС. Как известно, каждая таблица соответствия в ПЛИС снабжается триггером, т. е. в устройстве имеется некоторое количество триггеров, находящихся в ожидании и готовых к каким-либо действиям. Следовательно, во многих случаях для реализации конечных автоматов на основе ПЛИС наилучшим образом подходит *схема прямого кодирования*, особенно если активность в различных состояниях автомата действительно независима.

Прямое кодирование — представление каждого состояния конечного автомата только одним его активным триггером, причем в каждый момент времени может быть активен только один триггер.

Методики тестирования

При создании устройств на заказных микросхемах у разработчиков масса времени уходит на работу со средствами автоматизированного проектирования, с помощью которых выполняют формирование цепочек сканирования и *автоматическую генерацию тестовых комбинаций*. Для реализации функций *встроенного самоконтроля* они могут включать в свои устройства дополнительную логику. Немало усилий требует тестирование устройства на предмет производственных дефектов. В случае ПЛИС разработчики, как правило, не беспокоятся по поводу такой формы проверки, поскольку все устройства предварительно тестируются поставщиком.

Разработчики, использующие заказные микросхемы, обычно тратят много усилий на формирование JTAG-ячеек, последовательное сканирование и проверку своих конструкций. В отличие от заказных микросхем, средства последовательного сканирования, предусмотренные в ПЛИС, встраиваются в микросхему в процессе производства.