ГЛАВА 18

ПЕРЕХОД ОТ ЗАКАЗНЫХ МИКРОСХЕМ К ПЛИС И НАОБОРОТ

Сценарии проектирования

Существует несколько сценариев проектирования электронных систем с применением ПЛИС (**Рис. 18.1**).

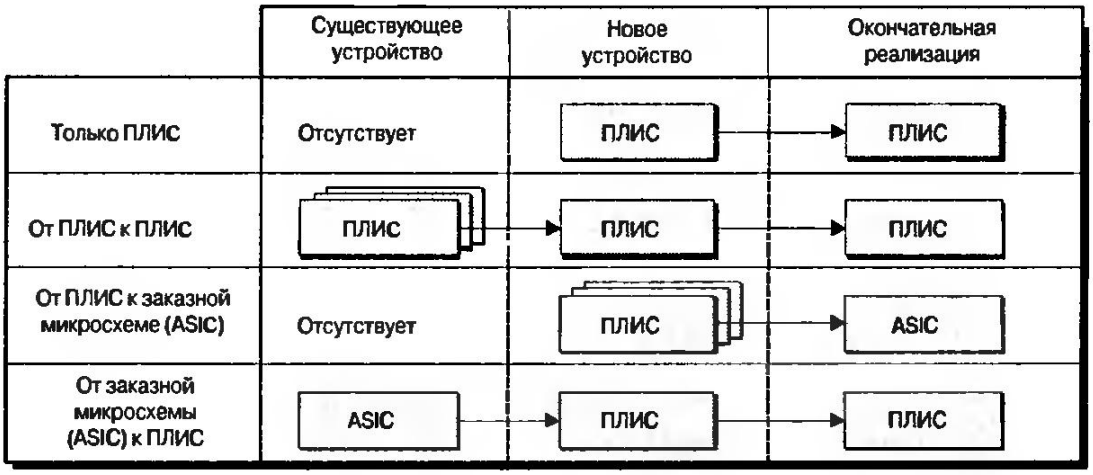

Рис. 18.1. Сценарии проектирования

Только ПЛИС

Этот сценарий относится к устройствам, предназначенным для реализации только на основе ПЛИС. В этом случае могут быть использованы любые, упоминаемые в этой книге, методы и средства проектирования.

От ПЛИС к ПЛИС

Такой подход подразумевает переход от уже существующего устройства на основе ПЛИС к ПЛИС, изготовленной по новой технологии. Новая технология очень часто выступает в форме нового семейства микросхем от того же поставщика, что и существующая микросхема, но также можно перейти и к микросхемам другого производителя.

При развитии событий по такому сценарию довольно редко имеет место переход от одного устройства к другому по принципу 1:1, т. е. просто взять содержимое одного компонента и реализовывать его в другом. Наиболее часто такой переход осуществляется в виде перехода от нескольких ПЛИС к одному новому. Также можно собрать функциональность одной или нескольких существующих ПЛИС, добавить к ней окружающую дискретную логику и реализовать всё в одном новом устройстве.

В подобных случаях обычно собирают в одно целое RTL-описания всех существующих устройств и дискретной логики, настраивают полученный код для улучшения отдельных характеристик схемы, и затем его вновь синтезируют в одно новое устройство.

От ПЛИС к заказной микросхеме

Под переходом от ПЛИС к заказной микросхеме (ASIC) обычно понимают использование одной или нескольких ПЛИС в качестве прототипа устройства на основе заказной микросхемы. При этом, если пользователь работает не с малыми или средними заказными микросхемами, часто возникает необходимость разделить устройство на несколько ПЛИС. Некоторые поставщики ПЛИС и САПР электронных систем предлагают (или предлагали) приложения, которые выполняют такое разбиение автоматически[1], но эти средства появляются и исчезают с завидной регулярностью. Их характеристики, возможности и качество работы также меняются почти каждую неделю (другими словами, вам придётся самостоятельно оценить предлагаемые решения и сделать свой выбор).

Некоторые функции, такие как очереди FIFO или блоки двухпортовой памяти, имеют довольно-таки специфичную реализацию, если они формируются из блоков ОЗУ, встроенных в ПЛИС. Способы реализации этих функций обычно отличаются от тех, которые используются при реализации заказных микросхем, что может послужить причиной некоторых проблем. Решение этого вопроса заключается в создании своей собственной RTL-библиотеки функций заказных микросхем, таких как умножители, компараторы, блоки памяти и так далее, которые будут полностью соответствовать своим ПЛИС-аналогам. К сожалению, такой подход подразумевает реализацию этих элементов в пользовательском устройстве в виде отдельного RTL-кода, вместо того чтобы взять универсальный RTL-код всего устройства и позволить все сделать алгоритму синтеза. Своего рода уравновешивающее действие, как и многие другие, используемые в проектировании.

Как уже отмечалось в гл. 7, устройство, предназначенное для реализации с помощью ПЛИС, обычно содержит меньшее количество уровней логики (последовательно соединенных логических вентилей) между регистрами, чем при реализации на основе заказной микросхемы. В некоторых случаях целесообразно создать RTL-код устройства, предназначенный для реализации на основе заказной микросхемы и пойти на уменьшение производительности ПЛИС-прототипа.

Можно также сгенерировать два варианта RTL-кода, один для использования в ПЛИС-прототипе, а другой для конечного устройства на основе ASIC. Но, как правило, это решение может стать причиной многих проблем, так как в двух представлениях довольно легко потерять синхронизацию и прийти в итоге к двум совершенно разным результатам.

В этом случае могут использоваться средства, основанные на чистой версии языка C/C++, которые были рассмотрены в гл. 11. Напомню, что идея состоит в том, чтобы не вручную добавлять информацию о реализации в исходный RTL-код (таким образом делая его зависимым от конкретной реализации), а все информацию формировать (синтезировать) с помощью управляемой пользователем C/C++ машиной (алгоритмом) синтеза (**Рис. 18.2**).

После обработки исходного кода средствами синтеза, его можно использовать для формирования микроархитектурных компромиссов и затем оценить их с точки зрения размеров и скорости. Пользовательская конфигурация для каждого возможного сценария может быть

---

[1] Хорошим примером приложения, которое предлагает подобную функциональность, является пакет Certify® от компании Synplicity (www.synplicity.com).

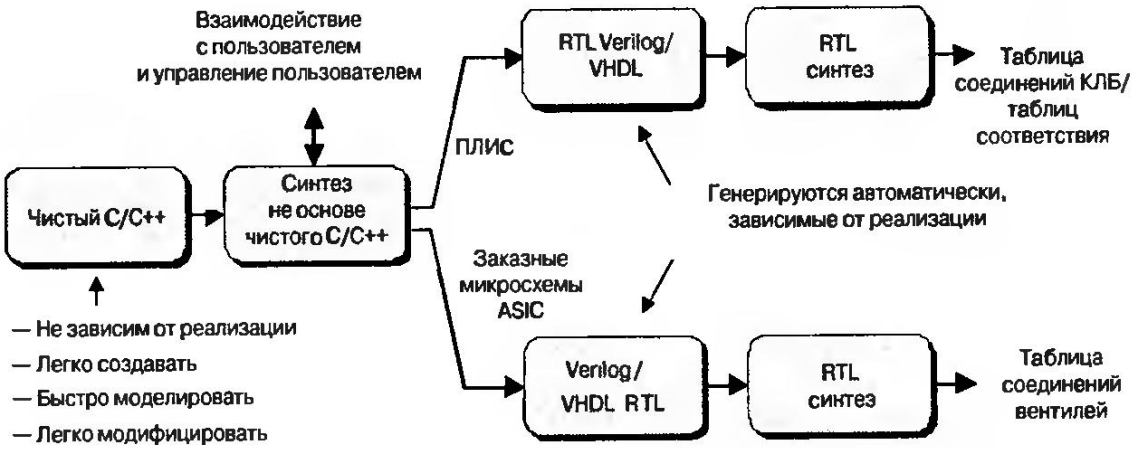

Рис. 18.2. Проектирование на основе «чистой» версии языка C/C++

именована, сохранена и, при необходимости, повторно использована. Следовательно, сначала можно создать конфигурацию для ПЛИС-прототипа и после проверки создать второй её вариант для окончательной реализации в ASIC. Суть состоит в том, что один и тот же исходный код C/C++ может использоваться в обеих цепочках проектирования.

Следующая особенность заключается в том, что современные заказные микросхемы могут содержать невообразимо большое количество зон (доменов) и подзон синхронизации (речь идёт о сотнях таких зон и подзон). В отличие от них, количество зон первичной синхронизации в ПЛИС ограничено (около 10). Это значит, что при использовании одной или нескольких микросхем ПЛИС для создания прототипа заказной микросхемы, надо хорошенько продумать организацию системы синхронизации.

И в завершении замечу, что существует интересный Европейский патент под номером EP0437491 (B1), при чтении которого — о, ужас, какой он нудный — кажется, что возможность использования нескольких программируемых устройств, таких как ПЛИС, для реализации прототипов ASIC, сильно преувеличена. В действительности, как я думаю, этот патент, вероятно, описывал применение ПЛИС для создания логического эмулятора, но был сформулирован таким образом, что стал препятствовать использованию нескольких ПЛИС в качестве прототипа заказной микросхемы.

Когда данная книга уже буквально была готова к изданию, компания Synopsys (www.synopsys.com) заявила о создании ПЛИС-оптимизированной версии *Design Compiler FPGA* (DC FPGA), в основе которой лежал хорошо известный алгоритм синтеза заказных микросхем *Design Compiler*® (DC ASIC). Кроме того, этот компилятор поддерживал весьма интересные нововведения, которые назвали Adaptive Optimization™ Technology. Главное его достоинство состояло в том, что DC ASIC и DC FPGA могут использовать один и тот же исходный RTL-код для создания одного и того же устройства с помощью и ПЛИС и заказных микросхем. Каждое программное ядро может быть настроено на использование различных микроархитектурных схем, таких как распределение ресурсов и конвейерная обработка. Кроме того, компилятор для ПЛИС может выполнять автоматическое преобразование любых, ориентированных на заказные микросхемы, встроенных функций в код RTL. Поэтому DC FPGA и DC ASIC знаменуют собой значительное событие в контексте использования ПЛИС как прототипов устройств на основе заказных микросхем.

От заказной микросхемы к ПЛИС

Этот сценарий подразумевается перевод устройства, реализованного на основе заказной микросхемы, к реализации на основе ПЛИС. Причин для этого может быть много, но очень часто преследуется цель усовершенствовать существующую функциональность устройства на основе заказной микросхемы без существенных финансовых затрат. Иногда первоначальная технология, по которой изготавливалась ASIC, может устареть, но устройства, в которых используется данная микросхема, всё ещё востребованы для поддержки текущих контрактов (особенно часто это встречается в военной сфере). Интересно, что последнее поколение ПЛИС обычно настолько далеко вырывается вперёд, что в одной такой микросхеме возможно разместить целиком все устройство, использующее заказную микросхему, разработанную несколько лет назад (при разбиении устройств на несколько ПЛИС можно воспользоваться некоторыми средствами автоматизации, облегчающими эту задачу, как описывалось в подразделе «От ПЛИС к заказной микросхеме»).

Прежде всего, необходимо тщательно просмотреть RTL-код, чтобы удалить или, по крайней мере, оценить, всю асинхронную логику, комбинационные петли, цепи задержки и другие подобные им блоки (*см.* гл. 7). Если в устройстве используются триггеры с входами как установки, так и сброса, можно изменить их и использовать только один из этих входов (*см.* гл. 7). Необходимо также найти все защёлки и переделать схемы под использование триггеров. Кроме того, следует проанализировать выражение вида *if-then-else*, в которых отсутствует конструкция *else*, так как в этом случае средства синтеза будут вставлять в описание устройства защёлку (*см.* гл. 9).

Что касается средств синхронизации, необходимо убедиться, что выбранная ПЛИС поддерживает достаточное количество зон синхронизации, чтобы удовлетворить требования заказной микросхемы, в противном случае придётся переделать имеющиеся цепи синхронизации. Кроме того, если заказная микросхема использует методы стробирования тактовых сигналов, следует исключить их из нового устройства и, по возможности, заменить средствами разрешения тактовых сигналов (*см.* гл. 7). Повторюсь, что некоторые поставщики ПЛИС и САПР электронных систем предоставляют средства синтеза, которые могут автоматически преобразовывать устройства на основе ASIC, использующие методы стробирования тактовых сигналов, в их ПЛИС-эквиваленты, использующие разрешение тактовых сигналов[1)].

При использовании сложных функциональных элементов, например блоков памяти (очередей FIFO и двухпортовых блоков ОЗУ), вероятно, вам придётся подстроить RTL-код под особенности ПЛИС. В некоторых случаях придётся заменить некоторые стандартные операторы RTL-кода (которые будут обрабатываться алгоритмом синтеза) на вызовы специфичных фрагментов схем или элементов ПЛИС.

И в завершении замечу, что конвейеры, реализованные в заказной микросхеме, вероятно, имеют большее число уровней логики между регистрами, чем при реализации на ПЛИС, если желательно сохранить её производительность. Большинство современных средств логического и физического синтеза поддерживают возможности восстановления синхронизации, которые позволяют им передвигать логику

1927 г. *Харолд Стивен Блэк (Harold Stephen Black) предложил идею использования* ***отрицательной обратной связи,*** *благодаря которой стало возможным создание усилителя Hi-Fi класса.*

[1)] Хорошим примером приложения, которое обеспечивает решение задач этого рода, может служить пакет Amplify® от компании Synplicity (www.synplicity.com).

1927 г. *Америка. Фило Фарнзуэрт (Philo Farnsworth) собрал полную электронную телевизионную систему.*

вперёд и назад по конвейеру через регистровые границы для достижения лучших временных показателей. С этой работой обычно лучше справляется алгоритм физического синтеза (*см.* гл. 19).

Наверняка не ошибусь, если скажу, что современная ПЛИС, вероятно, изготовлена с использованием более современной технологии (например, изготовлена по технологии 130 нм), чем заказная микросхема в существующем устройстве (например, изготовлена по технологии 250 нм). Это даёт ПЛИС определённые преимущества в скорости, что компенсирует более существенную задержку сигнала на проводниках. И в завершении заметим, что, возможно, все равно, придётся доводить настройку кода вручную, чтобы добавить в конвейер дополнительный элемент.