ГЛАВА 14

МОДУЛЬНОЕ И ПОШАГОВОЕ ПРОЕКТИРОВАНИЕ

Работа с одним целым

Для того чтобы подготовиться к разговору на эту тему, давайте рассмотрим структуру ПЛИС как набор столбцов, каждый из которых содержит большое количество программируемых логических блоков вместе с блоками ОЗУ и другими аппаратными элементами, например умножителями с накоплением (MAC), *см.* **Рис. 14.1**.

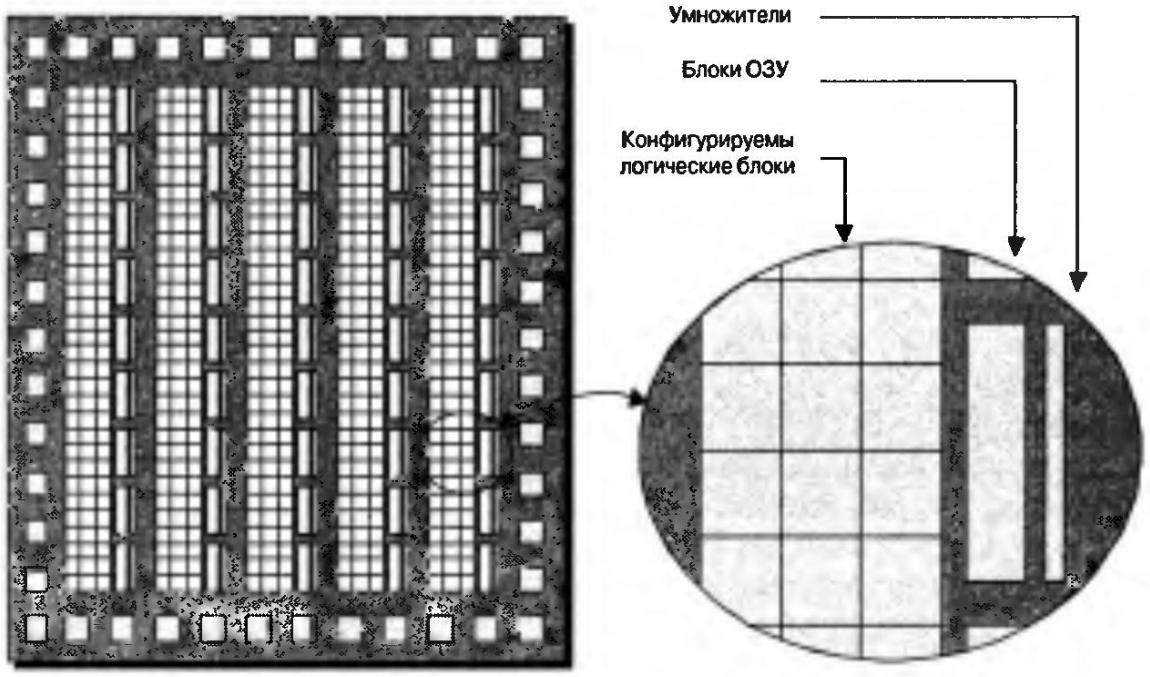

Рис. 14.1. Архитектура в виде столбцов

Конечно, этот рисунок сильно упрощен, так как современные устройства могут содержать больше столбцов, чем можно себе вообразить, и каждый столбец может состоять из огромного множества программируемой логики, и так далее.

При рассмотрении программирования ПЛИС на основе ячеек статического ОЗУ, было установлено, что все конфигурационные ячейки памяти можно рассматривать как один большой сдвиговый регистр (см. гл. 5). Рассмотрим, например, простое изображение поверхности кристалла, на котором показаны только контакты ввода/вывода и конфигурационные ячейки ОЗУ (**Рис. 14.2**).

Снова повторюсь: мы рассматриваем конфигурационные ячейки как совокупность столбцов, каждый из которых соответствует столбцу программируемой логики, как показано на **Рис. 14.1**. Такое представление является существенно упрощенным, так как ПЛИС может содержать десятки миллионов таких конфигурационных ячеек, но оно позволяет сделать более доступным для восприятия и понимания со-

1917 г. *Кларенс Бердзай (Clarence Birdseye) стал хранить продукты при помощи заморозки.*

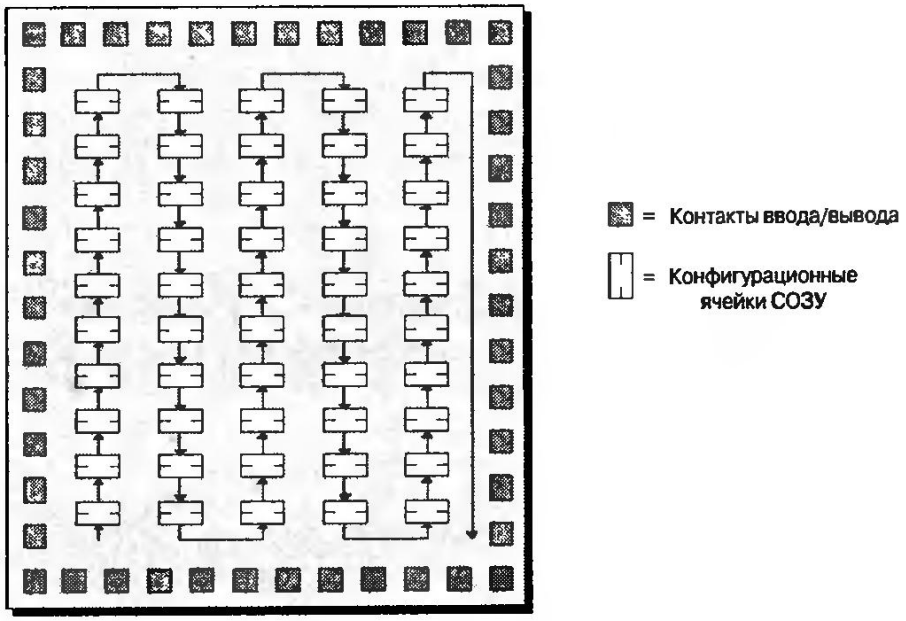

Рис. 14.2. Конфигурационные ячейки статического ОЗУ, представленные как один большой сдвиговый регистр

держание этой главы. Способы доступа к двум концам регистровой цепочки зависят от выбранного режима программирования.

На заре проектирования ПЛИС, примерно во второй половине 80-х, устройства были относительно небольшими, если говорить об этом терминами их логической ёмкости. Вследствие этого одному инженеру, как правило, было под силу создать RTL-описание всего устройства. Затем RTL-описания проходили процедуру синтеза, после чего синтезированная таблица соединений подвергалась операции размещения и разводки, которая и завершала процедуру проектирования устройства. В результате получался монолитный конфигурационный файл, который определял функциональность всего устройства и мог быть загружен в микросхему одним большим куском. Такая схема очень хорошо работала с конфигурационными ячейками, представленными в виде большого сдвигового регистра, и все были счастливы.

Разбиение на меньшие блоки

ПЛИС не стояли на месте: они непрерывно расширялись и совершенствовались, в результате чего стали стремительно расти их размеры и сложность. В связи с этим все устройство стали разбивать на отдельные функциональные блоки, и за каждым блоком закреплялся один или несколько инженеров. В этом случае каждый блок мог быть синтезирован самостоятельно, но все таблицы соединений, связанные с каждым блоком, перед операцией размещения и разводки собирались вместе. Снова повторюсь, что размещение и разводка обычно выполнялись для всего устройства, и выполнение этих операций могло занять всю ночь, если речь шла об устройстве с миллионами логических элементов.

Начиная где-то с 2002, года некоторые поставщики ПЛИС начали предлагать большие устройства, в которых конфигурационные ячейки на основе статического ОЗУ представлялись как множество относительно коротких сдвиговых регистров (**Рис. 14.3**).

Идея представления устройства с множеством цепочек возникла в связи с концепцией модульного и пошагового проектирования. Не исключено, что причина появления представления в виде множества цепо-

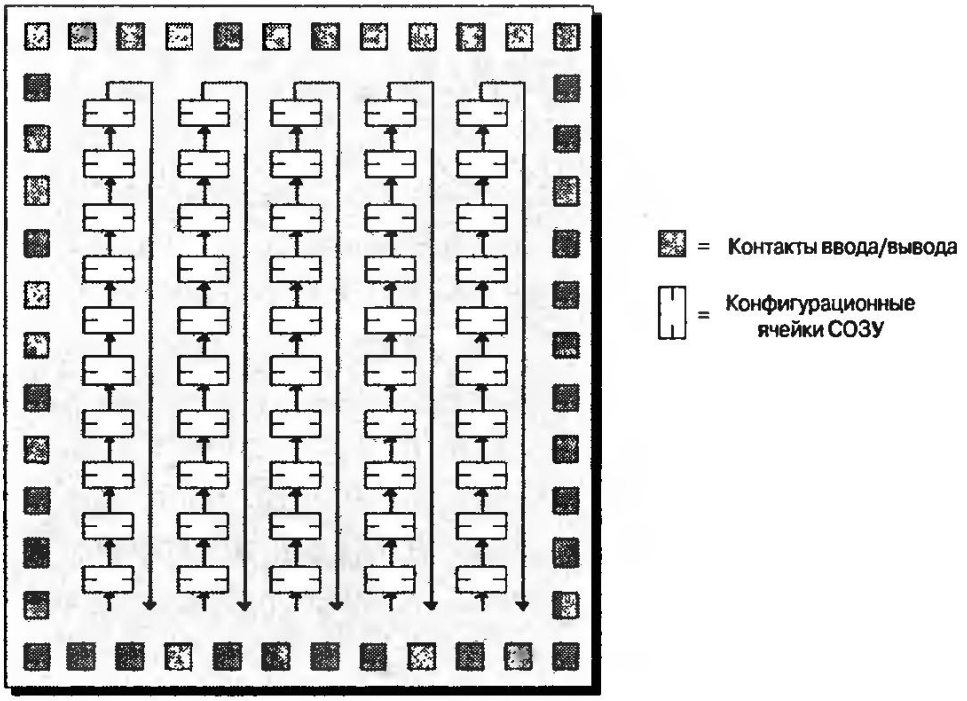

Рис. 14.3. Конфигурационные ячейки статического ОЗУ, представленные в виде множества относительно небольших сдвиговых регистров

чек была более приземлённой и имела аппаратный характер, и вообще все могло произойти примерно так — какой-то умник сказал: «Постойте-постойте, теперь, когда у нас есть множество цепочек, почему бы нам не поддержать концепцию модульного и пошагового проектирования?»

Если б я был любителем заключать пари, вероятнее всего, поставил бы на последний вариант. Но, будем терпимыми и предположим, что кто-то когда-то все-таки понимал, что он или она делает (могло же так быть). Тем не менее, это уже случившийся факт, и в результате эта архитектура вместе с соответствующим программным обеспечением может поддерживать концепцию модульного и пошагового проектирования.

Модульное проектирование

В этом разделе мы рассмотрим так называемую *командную разработку*. В этом случае большое устройство разбивается на функциональные блоки и за каждым блоком закрепляется отдельный разработчик или группа разработчиков, которые проектируют его с учётом всех временны*х* ограничений. Для каждого блока индивидуально создаётся и синтезируется RTL-код, и после завершения блочной разработки все полученные физические таблицы соединений передаются системному интегратору.

В конечном счете, каждому блоку (или небольшой группе блоков) будет назначено определённое место в устройстве. Системный интегратор несет ответственность за «стыковку» всех этих блоков вместе. По-другому это представление можно сравнить с разделённой на многие устройства ПЛИС с той лишь разницей, что в данном случае все устройства реализованы в одной микросхеме.

Главное преимущество этого метода заключается в том, что таблица соединений для каждой проектируемой области может быть индивидуально подвержена обработке приложением размещения и разводки. Это значит, что каждый член команды может заканчивать свою работу над устройством с полной уверенностью, что его блок будет соответствовать временны*м* ограничениям не только после синтеза, но и после реализации устройства.

1917 г. *Франк Конрад (Frank Conrad) построил радиостанцию, которая затем стала называться KDKA (этот позывной используется и в наши дни).*

Термин *«блочное проектирование»* иногда может относиться к модульному проектированию.

1918 г. *Осуществлена радиосвязь между Великобританией и Австралией.*

Пошаговое проектирование

Пошаговое проектирование предусматривает наличие интерфейса между блоками или столбцами и в этом случае позволяет модифицировать RTL какого-либо отдельного блока, затем пересинтезировать этот блок и локально запустить приложение размещения и разводки для этого блока. Это намного быстрее, чем перезапуск приложения размещения и разводки для всего устройства.

На самом деле понятие «локально», возможно, не совсем точно отражает то, что имеет место в действительности. Очевидно, следовало бы сказать, что пошаговое проектирование «замораживает» все неизменённые блоки на своих местах, а переделывает только модифицированный блок или блоки в контексте всего устройства. Этот подход обладает преимуществом над модульными методами проектирования, в которых другие блоки не участвуют в этой процедуре. При этом данные методы можно использовать в сочетании одного с другим.

Отрицательные стороны

Одна из проблем, присущих описанным в этой главе методам, заключается в том, что их использование может привести к большой растрате ресурсов, так как во время написания этой книги, наилучшее разрешение, свойственное этим методам, составляло величину одного целого столбца. Другими словами, если какой-то функциональный блок занимает, скажем, 75% логики этого столбца, остальные 25% будут неиспользованными. Поставщики ПЛИС, поддерживающие подобные архитектуры, говорят о некотором механизме, который в будущем позволит поддерживать лучшее разрешение.

Другая проблема заключается в том, что описанные здесь методы почти обрекают каждый логический блок на реализацию в виде «высокий-и-тонкий», так как они могут состоять только из одного или нескольких вертикальных столбцов. Это особенно огорчительно, когда речь идет о функциональных блоках, которые больше подходят для реализации в виде «низкий-и-широкий», т. е. при необходимости использования небольших частей из нескольких, находящихся рядом столбцов.

Но, пожалуй, наиболее существенным недостатком первых версий средств и методов, использующих рассматриваемую архитектуру для реализации концепции модульного и пошагового проектирования, является то, что кто-то, скажем системный интегратор, вынужден вручную создавать компоновочный план устройства. Этому бедолаге предстоит также определить и разместить специальные интерфейсные блоки, называемые *шинными макроячейками*, для связи шин и передачи индивидуальных сигналов из одного блока в другой (**Рис. 14.4**).

Первые версии средств разработки компоновочного плана, определения и размещения шинных макроячеек были, мягко говоря, неудобными. Ходят слухи, что в недалёком будущем произойдёт замена программного обеспечения, вследствие чего выполнять эти операции станет существенно проще. Хуже, чем есть, не будет.

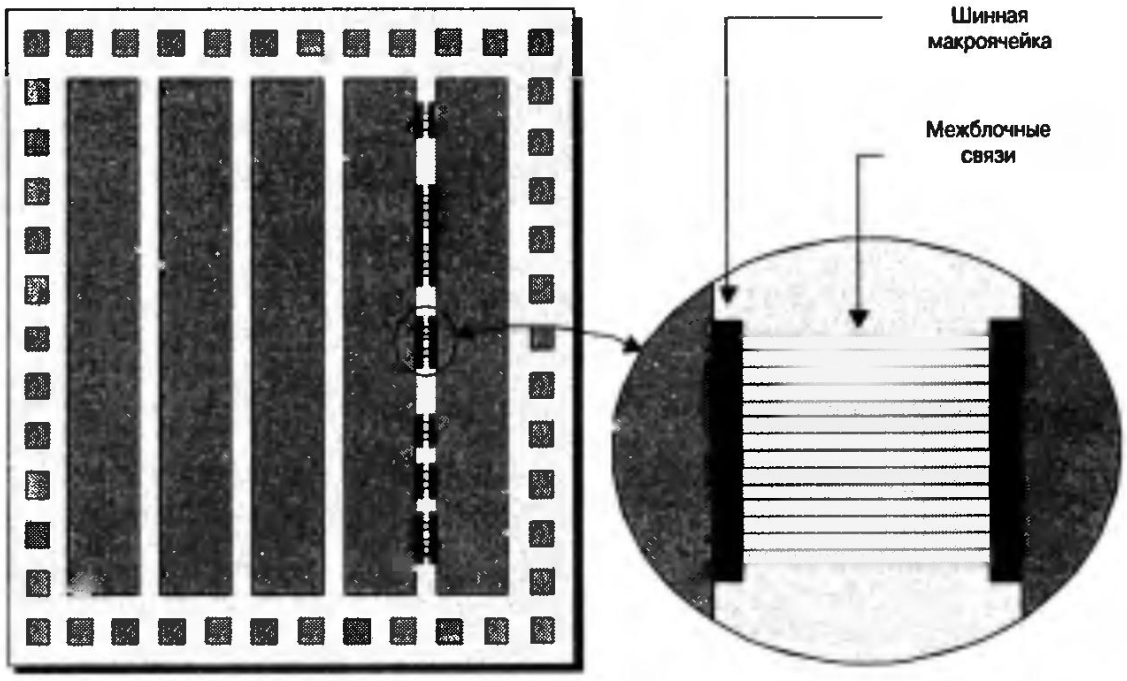

Рис. 14.4. Расположение шинных макроячеек

Всегда есть выход

В главе 10 была рассмотрена концепция *виртуальных макетов (прототипов) кристалла*. При этом было отмечено, что некоторые поставщики САПР электронных систем приступили к поставкам средства, которые поддерживали концепцию виртуальных прототипов для ПЛИС посредством планирования компоновки и предварительного, перед процедурой размещения и разводки, временного анализа. Эти средства в сочетании с возможностью выполнять операцию размещения и разводки для индивидуальных блоков устройства, существенно сокращают время создания устройства[1)].

Дело в том, что если вернуться к 10-й главе и перечитать ее, можно обнаружить, что описанные там средства виртуального макетирования ПЛИС полностью поддерживают концепцию модульного и пошагового проектирования, причём без недостатков, присущих методам, описанных в этой главе. Проблема заключается только в том, что, будучи более сложными, средства САПР электронных систем окажутся существенно дороже, чем решения, предлагаемые поставщиками ПЛИС. Как всегда, действует принцип «кто платит, тот и выбирает».

1919 г. *У абонентов телефонных станций появилась возможность самостоятельно набирать телефонные номера.*

1919 г. *Открыт коротковолновый радиодиапазон.*

[1)] Во время написания этой книги ведущим сторонником виртуального макетирования ПЛИС в форме, описываемой в этой книге, являлась компания Hier Design (www.hierdesign.com).