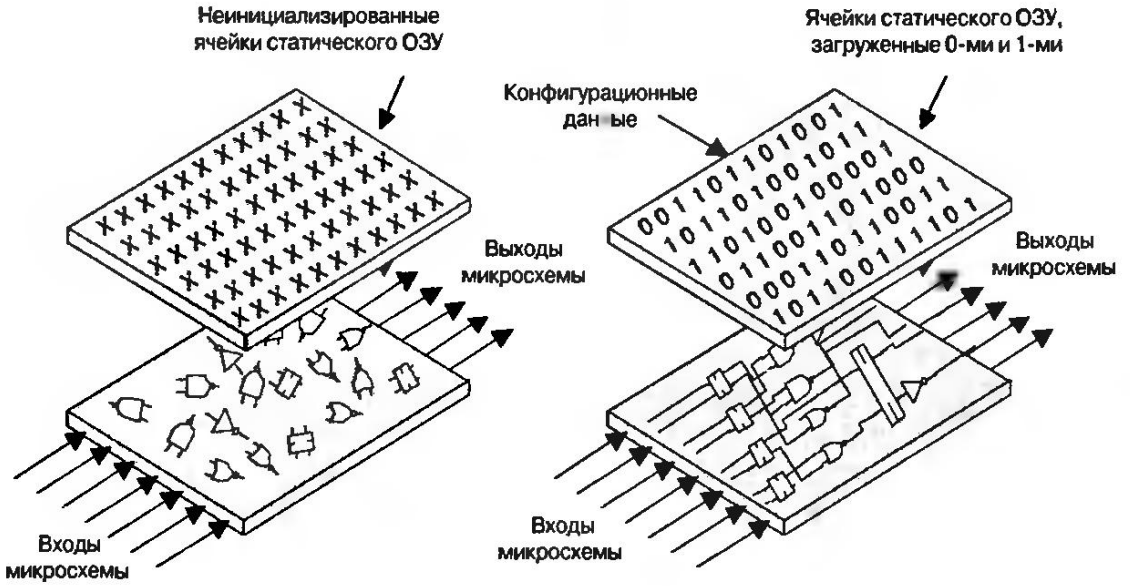
ГЛАВА 22

СИСТЕМЫ С ПЕРЕСТРАИВАЕМОЙ АРХИТЕКТУРОЙ

Динамически реконфигурируемая логика

С развитием ПЛИС на основе ячеек статического ОЗУ в электронике появились новые возможности, связанные с *динамически реконфигурируемой логикой*. Имеются в виду устройства, которые могут реконфигурироваться на лету в процессе работы системы.

Напомню, что ПЛИС содержит большое количество программируемых логических элементов и регистров, которые могут быть соединены разными способами для реализации различных функций. Примечательно, что компоненты на базе ячеек статического ОЗУ позволяют системе загружать в устройство новые конфигурационные данные. Хотя все логические вентили, регистры и ячейки статической памяти, из которых состоит ПЛИС, созданы на поверхности одного кремниевого кристалла, иногда полезно рассматривать это устройство как состоящее из двух отдельных частей — логических вентилей (и регистров) и программируемых конфигурационных ячеек статического ОЗУ (**Рис. 22.1**).



а) Несконфигурированная ПЛИС ***б)*** Сконфигурированная ПЛИС

Рис. 22.1. Динамически реконфигурируемая логика: ПЛИС на основе ячеек статического ОЗУ

Гибкость реконфигурируемых устройств открывает поистине широкие возможности. Например, при включении питания ПЛИС может быть сконфигурирована для тестирования системы или самотестирования. Как только система завершает тестирование, ПЛИС вновь может быть переконфигурирована для выполнения своих основных функций.

1957 г. *Америка. Гордон Голд (Gordon Gould) сформулирован принципы работы лазера.*

1957 г. *Америка. Создан первый компьютер серии IBM 610.*

1957 г. *В России запущен первый искусственный спутник Земли.*

Динамически реконфигурируемые внутренние соединения

Безусловно, возможность переконфигурировать функции отдельных устройств на печатной плате предоставляет дополнительные удобства для инженеров, но иногда им необходимо создавать целые системы, которые могут быть переконфигурированы на уровне печатной платы для выполнения различных, радикально отличающихся, функций.

Для решения этой задачи необходимо динамически конфигурировать соединения между устройствами на уровне печатной платы. Справиться с поставленной задачей могут так называемые *программируемые пользователем устройства внутренних соединений* (или *FPID — field-programmable interconnect devices*), которые также известны как *программируемые пользователем кристаллы внутренних соединений* (или *FPIC — field-programmable interconnect chips*)[1]. Эти микросхемы, применяемые для соединения логических устройств, могут быть динамически переконфигурированы таким же образом, как и стандартные ПЛИС на основе ячеек статического ОЗУ. Так как каждая микросхема FPID может содержать более 1000 контактов, для соединения компонентов на печатной плате может понадобиться всего лишь несколько таких микросхем (**Рис. 22.2**).

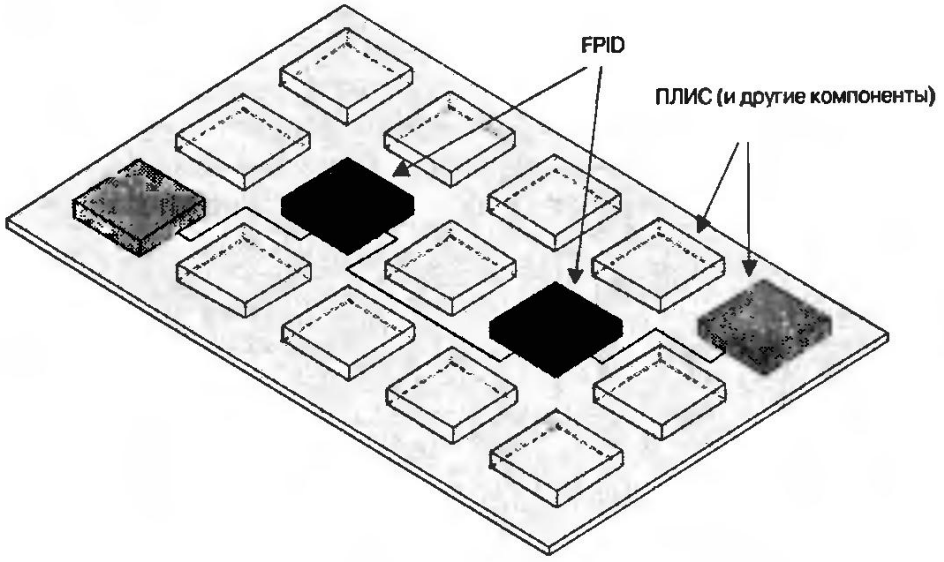


Рис. 22.2. Динамически реконфигурируемые соединения: FPID на основе ячеек статического ОЗУ

Интересно, что описанный подход не ограничивается только применением на уровне печатных плат. Любой из вышеописанных подходов потенциально может быть использован и в гибридных микросхемах, и в многокристальных модулях, и в однокристальных устройствах.

Системы с перестраиваемой архитектурой

Как часто бывает в электронике, термин *перестраиваемая архитектура (Reconfigurable Computing)* может иметь разные значения для разных людей. Одни это понятие относят к микропроцессорам, набор команд которых можно изменять или расширять на лету. Однако в нашем случае под перестраиваемой архитектурой (системой с перестраиваемой архитектурой) лучше понимать часть аппаратного обеспечения общего назначения — такого как ПЛИС (какая неожи-

---

[1] FPIC — торговая марка компании Aptix Corporation (www.aptix.com).

данность!) — которое может быть сконфигурировано для выполнения каких-либо задач, но при необходимости его конфигурацию можно изменить.

Одним из ограничивающих факторов для большинства ПЛИС на основе ячеек статического ОЗУ является необходимость затрачивать определенное время на переконфигурацию. Эта особенность вызвана тем, что ПЛИС обычно программируется с помощью последовательной загрузки данных (или параллельного загрузки данных шириной всего лишь 8 бит). Когда речь идет о высокотехнологичных устройствах с десятками миллионов конфигураций ячеек статической памяти, на перепрограммирование может уйти несколько секунд. Существует некоторые типы ПЛИС, которые решают эту проблему путем использования большого количества контактов ввода/вывода общего назначения, которые в процессе программирования образуют широкую шину конфигурации (скажем, 256 бит), а после конфигурирования возвращаются к выполнению своих основных функций (*см.* гл. 26). Кроме того, некоторые типы *программируемых пользователем массивов узлов (FPNA — field programmable node array)* также используют широкие шины конфигурации (*см.* гл. 23).

Другой недостаток ПЛИС традиционной архитектуры состоит в том, что при необходимости внести самые незначительные изменения в конфигурацию приходится перепрограммировать целиком всё устройство. Некоторые последние ПЛИС позволяют переконфигурировать их по столбцам (*см.* гл. 14), но и в этом случае обеспечивается весьма грубый уровень структурирования. Кроме того, на время реконфигурирования обычно приходится приостанавливать работу всей печатной платы. Также при программировании безвозвратно теряется содержимое всех регистров ПЛИС.

Для решения этих проблем, примерно в 1994 году компания Atmel Corporation (www.atmel.com) предложила ряд интересных ПЛИС. Кроме поддержки динамической реконфигурации выбранных элементов внутренней логики, эти устройства характеризуются тем, что:

- не нарушается работа входов и выходов устройства;
- не нарушается работа средств синхронизации системы;
- любые части устройства, не подверженные реконфигурации, могут продолжать свою работу;
- не теряется информация внутренних регистров, даже тех, которые находятся в области реконфигурирования.

Интерес представляет последний пункт этого списка, так как позволяет одной реализации функции передавать данные для другой. Например, группа регистров изначально может быть запрограммирована для работы в качестве двоичного счетчика. Затем в какой-то момент времени, определяемый главной системой, те же регистры могут быть переконфигурированы для работы в качестве *линейного сдвигового регистра с обратной связью*[1], начальное значение которого определяется последним содержимым счётчика до реконфигурирования.

Хотя эти устройства стали очередным эволюционным шагом в технологическом плане, с точки зрения их потенциальных возможностей шаг оказался поистине революционным. Для оценки этих возможностей были введены такие новые термины, как *виртуальное аппаратное обеспечение* и *кэш-логика*[2].

1958 г. *Америка. Осуществлена передача компьютерных данных через телефонные каналы связи.*

1958 г. *Америка. Джеку Килби (Jack Kilby) в то время, когда он работал в компании Texas Instruments, удалось изготовить несколько компонентов на одном полупроводниковом кристалле (первая интегральная микросхема).*

---

[1] Более подробно эти регистры описаны в Приложении В.

[2] Термин *кэш-логика* (Cache Logic) является торговой маркой компании Atmel Corporation, Сан-Хосе, Канада, США.

1959 г. *Америка. Разработан компьютерный язык COBOL.*

1959 г. *Америка. Роберт Нойс (Robert Noyce) разработал технологию создания микроскопических алюминиевых проводников на кремниевом кристалле, которые продолжают использоваться для изготовления современных интегральных микросхем.*

Понятие *виртуальное аппаратное обеспечение* возникло по аналогии с его программным эквивалентом — *виртуальной памяти*; оба этих термина используются для представления предметов, которые на самом деле не существуют. В случае виртуальной памяти компьютерная операционная система делает вид, что ей доступно больше оперативной памяти, чем есть на самом деле. Например, программе, выполняющейся на компьютере, может потребоваться 500 мегабайт памяти для хранения данных, а в компьютере установлено только 128 мегабайт. Чтобы решить эту проблему, каждый раз, когда программа пытается получить доступ к ячейкам памяти, которых физически не существует, операционная система хитрит и меняет часть данных, находящихся в памяти, с данными, расположенными на жестком диске. Хотя эта процедура, известная как *свопинг*, замедляет процесс обработки данных, но в то же время позволяет программе выполнять свои задачи не дожидаясь, пока кто-нибудь сбегает в магазин за дополнительными модулями памяти.

Аналогично термин *кэш-логика* произошел от родственного ему понятия *кэш-память*, которая представляет собой высокоскоростную дорогостоящую статическую память для хранения текущих активных данных, в то время как основная часть данных помещается в более медленную недорогую память, например, динамическую (DRAM). В данном контексте, к «текущим активным данным» относятся данные или инструкции, которые в настоящее время используются программой или будут, по требованию операционной системы, использованы в ближайшем будущем.

На самом деле концепцию виртуального аппаратного обеспечения довольно легко понять. Каждая большая макрофункция в устройстве обычно формируется из комбинации нескольких меньших микрофункций, таких как счетчики, сдвиговые регистры, мультиплексоры. При разделении нескольких макрофункций на ряд соответствующих микрофункций становится очевидным, что:

во-первых, перекрывается функциональность, и такой элемент, как счетчик может использоваться несколько раз в разных местах, и

во-вторых, появляется большое значение *функциональной латентности*, т. е. в любой момент времени активна только часть микрофункций. Также в течение каждого такта системы используется сравнительно мало микрофункций. Следовательно, с помощью динамической реконфигурации отдельных частей виртуального аппаратного обеспечения различные макрофункции можно реализовать относительно небольшим количеством логики.

Отслеживая местоположение и обращение к каждой микрофункции, а также объединяя функциональность и исключая избыточность, устройства с виртуальной аппаратной частью могут выполнять гораздо более сложные задачи, чем устройства, построенные по классической схеме. Например, в сложных функциях, требующих 100000 эквивалентных вентилей, в отдельный момент времени из них могут быть активны только 10000. Следовательно, с помощью сохранения или кэширования можно реализовать функций ещё на 90000 логических элементов. Тем самым небольшое и недорогое устройство со 10000 логическими элементами может заменить большее и дорогое с 100000 элементами (**Рис. 22.3**).

Теоретически также возможно в реальном масштабе времени компилировать новые варианты устройств, которые могут быть использованы как динамически создаваемые аппаратные подпрограммы.

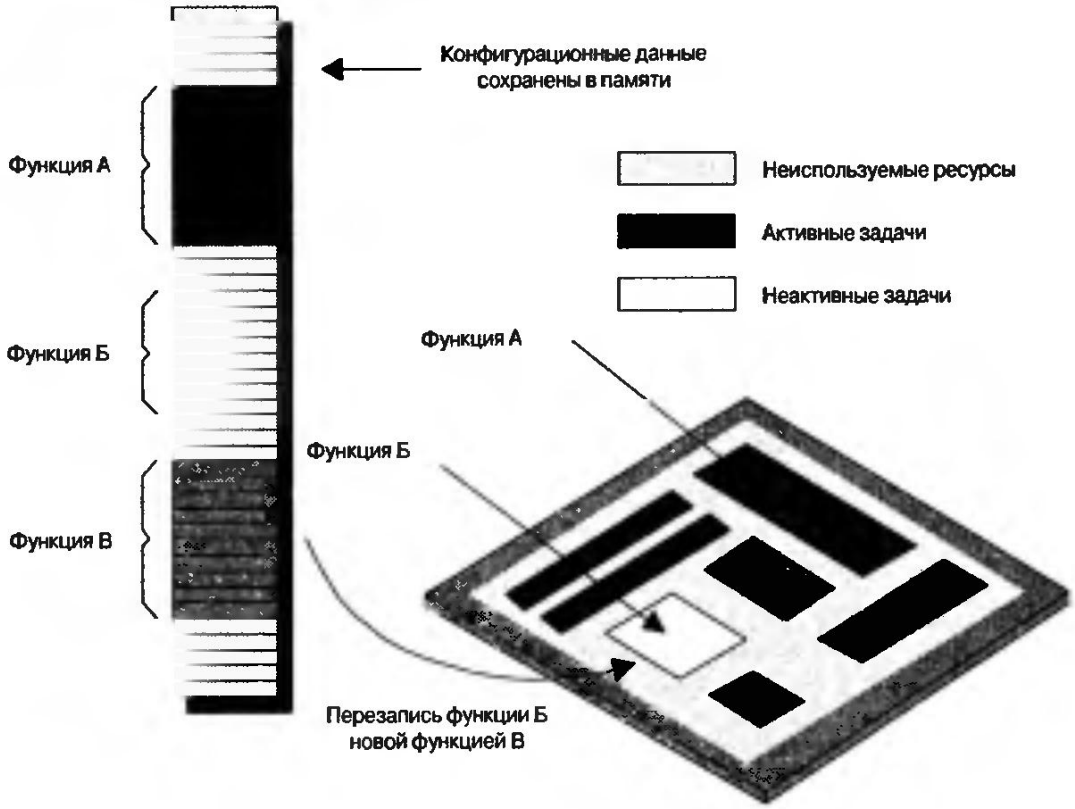


Рис. 22.3. Виртуальное аппаратное обеспечение

Во второй половине 90-х идея перестраиваемой архитектуры наделала много шума, и до сих пор встречаются ярые приверженцы-фанаты этих систем, которые размахивают символикой и носят майки с соответствующими надписями. Грустно признавать, но ничего эпохального из этого не вышло, за исключением сугубо специализированных приложений. Основная проблема кроется в том, что структура традиционной ПЛИС слишком «мелкомодульная», поэтому их реконфигурирование занимает чрезвычайно много времени (по компьютерным меркам). Чтобы поддерживать перестраиваемую архитектуру, необходимы устройства, которые можно переконфигурировать сотни тысяч раз в секунду. Собака может быть зарыта в крупномодульной архитектуре, используемой устройствами FPNA (*см.* гл. 23).

1959 г. *Швейцарский физик Джин Герни (Jean Hoerni) разработал планарную технологию, в которой для создания транзисторов использовались оптические литографические методы.*

1960 г. *Америка. Теодор Мэйман (Theodore Maiman) создал первый лазер.*

1960 г. *Америка. Управление перспективного планирования оборонных научно-исследовательских работ (или DARPA — Defense Advanced Research Projects Agency) заложило основы, благодаря которым появилась сеть Интернет.*