Данная книга посвящена модулю LabVIEW FPGA. Она будет интересна инженерам, начинающим применять LabVIEW для разработки реконфигурируемых систем, и в качестве учебного пособия студентам соответствующих специальностей.

В первой, второй и третьей вводных главах кратко рассказывается о ПЛИС, устройствах ввода-вывода и виртуальных измерительных приборах. **Четвертая и пятая** главы посвящены среде проектирования LabVIEW и принципам разработки в ней. **В шестой и седьмой** главе дается описание реконфигурируемых систем и средств LabVIEW для их разработки. **В восьмой главе** описан типичный маршрут разработки реконфигурируемых систем, а его подробности (как управлять FPGA VI и создавать Host VI; на примерах, как можно обрабатывать данные, как тестировать разработанные приложения, как использовать стандартные интерфейсы для работы с периферийными устройствами) — в главах **с девятой по одиннадцатую. Двенадцатая** глава посвящена модулям R-серии и тому, как можно измерять неэлектрические величины. **В тринадцатой** главе приведен пример разработки встраиваемой системы на платформе cRIO. А **в четырнадцатой** главе кратко рассказывается, как создать сборку проекта и инсталлятор для его развертывания на другом компьютере. **Пятнадцатая** глава логически завершает книгу; в ней рассказывается как проекты, разработанные в LabVIEW FPGA, перенести в другие САПР-ы и подготовить для компиляции под ПЛИС других производителей.

Книга будет полезна инженерам, студентам и широкому кругу читателей.

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО
www.дмк.рф

Интернет-магазин:
www.dmkpress.com

Оптовая продажа:
КТК “Галактика”
books@alians-kniga.ru

ISBN 978-5-93700-170-2

9 785937 001702

ИСТОВЫЙ ИНЖЕНЕР

ПРОЕКТИРОВАНИЕ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ В LABVIEW FPGA

Е.Д. Баран
А.Ю. Романов

КНИЖНАЯ ПОЛКА ИСТОВОГО ИНЖЕНЕРА

Е.Д. Баран
А.Ю. Романов

ПРОЕКТИРОВАНИЕ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ В LABVIEW FPGA

ВЫСШАЯ ШКОЛА ЭКОНОМИКИ
МИЭМ
ДМК ПРЕСС ИЗДАТЕЛЬСТВО

Е.Д. Баран, А.Ю. Романов

Проектирование реконфигурируемых систем в LabVIEW FPGA

УДК 621.3
ББК 32.372
Б24

Баран Е.Д. – старший преподаватель кафедры систем сбора и обработки данных Новосибирского государственного технического университета;
Романов А.Ю. – канд. техн. наук, доцент Московского института электроники и математики им. А. Н. Тихонова Национального исследовательского университета «Высшая школа экономики».

Баран Е.Д., Романов А.Ю.
Б24 Проектирование реконфигурируемых систем в LabVIEW FPGA. – М.: ДМК Пресс, 2023. – 646 с.: ил.

ISBN 978-5-93700-170-2

В этой книге изложены основы проектирования реконфигурируемых систем в среде графического программирования LabVIEW, дополненной модулем LabVIEW FPGA. Приведен обзор разновидностей ПЛИС, модулей ввода-вывода классической архитектуры и модулей с реконфигурируемыми каналами ввода-вывода производства National Instruments. Рассмотрены основные компоненты и инструменты среды проектирования, на реальных примерах показана методика разработки и отладки распределенных и встраиваемых систем измерения, управления и тестирования. Описан полный цикл проектирования, включая этапы создания исполняемых файлов и инсталляторов, а также развертывания систем на целевых устройствах.

Содержание книги отражает опыт разработок, выполненных в учебном центре «Центр технологий National Instruments» Новосибирского государственного технического университета, опыт обучения на курсах повышения квалификации и преподавания соответствующих дисциплин студентам.

Книга, которую вы держите в руках, продолжает серию «Книжная полка Истового Инженера», которая издается при поддержке компании YADRO. Это издание подготовлено к публикации Московским институтом электроники и математики им. А. Н. Тихонова НИУ ВШЭ совместно с «ДМК Пресс».

УДК 621.3
ББК 32.372

ISBN 978-5-93700-170-2

Содержание

Отзывы о книге

Современный мир невозможно представить без сложных цифровых устройств, которые претерпели существенные изменения за последнее столетие. Задачи, для решения которых пару десятков лет назад требовались огромные вычислительные машины, занимавшие целые здания, теперь способен решить процессор мобильного телефона, который может уместиться в ладони. Сейчас логические устройства встраиваются в различную технику, окружающую человека: автомобили, способные двигаться без участия человека, умные бытовые устройства, способные анализировать поведения, человека и предлагать ему разнообразные сценарии использования, и т.д. Поэтому задача разработки архитектуры новых логических устройств является важной и актуальной.

Одним из инструментов разработки логических цифровых устройств является ПЛИС (программируемая логическая интегральная схема). Благодаря своей реконфигурируемой архитектуре ПЛИС может быть встроена как логический узел в различные вычислительные устройства. Одним из инструментов работы с платой ПЛИС является расширение среды LabVIEW под названием LabVIEW FPGA. Данное расширение позволяет провести полный цикл разработки логического устройства с ПЛИС в среде LabVIEW вплоть до его переноса и развертывания в других САПР.

Стоит отметить, что на русском языке материалов по работе с данным расширением мало и они не систематизированы, поэтому рассматриваемая книга является большим подспорьем начинающим инженерам и профессионалам, которые хотят освоить принципы работы с платами ПЛИС в среде LabVIEW. LabVIEW FPGA может рассматриваться как средство проектирования для встраиваемых модулей с ПЛИС от компании National Instruments, так и как средство высокоуровневого синтеза для подготовки проекта к реализации на других аппаратных платформах.

Книга «Проектирование реконфигурируемых систем в LabVIEW FPGA» хорошо структурирована, изложена грамотным русским языком, легким для чтения. В ней подробно и полно раскрыто проектирование в LabVIEW встраиваемых систем на ПЛИС.

Эта книга будет полезна как специалистам, так и начинающим разработчикам.

Иванников А.Д.,
д.т.н., проф., Главный научный сотрудник Института
проблем проектирования в микроэлектронике
Российской академии наук

Дорогой читатель!

Ты держишь в руках книгу по проектированию реконфигурируемых систем измерения и управления, написанную профессионалами-практиками: разработчиком измерительных систем и разработчиком систем на кристалле; оба руководят вузовскими исследовательскими лабораториями и имеют большой преподавательский опыт – так что не сомневайся, они точно знают, как и о чем нужно писать в учебной литературе.

Книга стала логическим развитием предыдущего издания, увидевшего свет уже почти 15 лет назад и, конечно, требующего замены. Это учебное пособие будет крайне полезно в качестве входного портала для начинающих разработчиков, интересующихся студентов, а также и в качестве концентрированного справочника для практикующих специалистов. Оно удачно и систематически сочетает в себе 3 важнейших аспекта:

1) методология проектирования – включая необходимые сведения о ПЛИС как таковых, устройствах ввода/вывода, архитектуре конечных систем, интерфейсе для работы с периферийными устройствами;

2) введение в среду разработки (инструмент проектирования) – включая необходимые описания пользовательского интерфейса LabVIEW, правила и приемы программирования на встроенном графическом языке G, необходимые описания компонентов встроенной библиотеки FPGA;

3) множество простых и комплексных практических примеров (практику проектирования), необходимых для закрепления материала, демонстрирующих нюансы использования имеющихся возможностей.

Читатель! Эта книга даст тебе то самое «знание того, куда нужно ударить». Не теряй времени, воспользуйся короткой дорогой к профессионализму!

Самбурский Лев Михайлович,
к.т.н., доцент МИЭМ НИУ ВШЭ,
преподаватель курсов «Электроника»,
«Компоненты инфокоммуникационных устройств» и др.
Уже больше 10 лет использует LabVIEW
в преподавании и проектной работе

Об авторах

Баран Ефим Давыдович – старший преподаватель кафедры систем сбора и обработки данных (ССОД) Новосибирского государственного технического университета (НГТУ, до 1992 года – Новосибирский электротехнический институт – НЭТИ).

В 1969 г. защитил на кафедре информационно-измерительной техники (ныне ССОД) дипломный проект и работал в научно-исследовательском секторе на должностях от инженера до с.н.с., в качестве ассистента-совместителя вел на кафедре лабораторные и практические занятия по нескольким техническим дисциплинам.

С 1984 г. – старший преподаватель, основные курсы: «Микропроцессоры и микроконтроллеры», «Измерительные информационные системы», «Системы контроля и диагностики», «SCADA-системы», «Программирование в LabVIEW».

С 2004 г. руководитель созданного на базе кафедры ССОД НГТУ учебного центра «Центр технологий National Instruments»: *https://nitec.nstu.ru*. В лабораториях центра обучаются студенты нескольких факультетов НГТУ, проводятся курсы повышения квалификации для сотрудников и руководителей НИИ и КБ, промышленных предприятий, преподавателей образовательных учреждений высшего и среднего профессионального образования.

Баран Е.Д. разрабатывал и руководил разработками приборов и систем различного назначения, в том числе приборов и систем для функционального контроля и диагностики средств микропроцессорной техники, систем автоматизации научного эксперимента, учебных лабораторных стендов и лабораторных практикумов по ряду дисциплин. Работы в этих направлениях продолжаются и в настоящее время.

Автор 55 публикаций и 15 изобретений.

Романов Александр Юрьевич – доцент Московского института электроники и математики им. А.Н. Тихонова Национального исследовательского университета «Высшая школа экономики» (МИЭМ НИУ ВШЭ). В 2009 г. закончил магистратуру в Харьковском политехническом институте, работал в Киевском политехническом институте им. Сикорского. С 2014 г. работает в МИЭМ НИУ ВШЭ, где возглавляет лабораторию САПР (*https://miem.hse.ru/edu/ce/cadsystem*), специализирующуюся на проектной деятельности, а также разработке цифровых систем на ПЛИС / микроконтроллерах, робототехнических комплексов, аппаратных реализаций систем искусственного интеллекта, многопроцессорных систем, систем удаленного доступа к лабораторному оборудованию и т.д. В 2015 г. защитил диссертацию в Институте проблем проектирования в микроэлектронике РАН (г. Зеленоград), является автором более 150 научных статей, патентов и книг. Более подробно об учебном процессе в лаборатории можно узнать из интервью: *https://miem.hse.ru/news/364316102.html*.

Предисловие Е.Д. Барана

Система LabVIEW корпорации National Instruments заслуженно пользуется популярностью среди научных сотрудников и инженеров, которые разрабатывают системы автоматизации экспериментальных исследований, системы автоматизированного управления производственными процессами и испытаний объектов в различных отраслях промышленности и т.д. Основные достоинства LabVIEW – интуитивно понятный язык графического программирования, развитые прикладные библиотеки функций и простота интеграции с оборудованием ведущих производителей – позволяют в кратчайшие сроки решать сложные задачи, причем для этого не обязательно обладать знаниями и опытом профессионального программиста.

Один из модулей LabVIEW – модуль LabVIEW FPGA предоставляет возможность создавать реконфигурируемые системы, каналы ввода-вывода которых и встроенные специализированные процессоры обработки данных разрабатываются в едином процессе интеграции технических средств и программного обеспечения. Расширение функционала и улучшение технических характеристик систем достигаются благодаря конфигурированию структуры FPGA – программируемой логической интегральной схемы (ПЛИС), встроенной в готовые функциональные блоки, из которых собирается система. Чрезвычайно важно, что и разработка структуры FPGA, и разработка прикладного программного обеспечения для процессора классической архитектуры выполняются в одной среде проектирования с помощью одних и тех же инструментальных средств.

Концепция реконфигурируемых систем, основные их элементы и техника проектирования таких систем описаны в первом издании книги, которое вышло еще в 2009 году. За прошедшее с тех пор время радикально изменилась элементная база: увеличились степень интеграции и быстродействие FPGA, значительно расширился состав функциональных узлов. Теперь в один кристалл интегрированы многоядерные процессоры, аналого-цифровые и цифроаналоговые преобразователи, стандартные интерфейсные порты и т.п. Программируемые матрицы простых логических элементов эволюционировали в системы на кристалле (Systems-on-Chip).

Для программирования систем на кристалле компания Xilinx разработала новый инструментарий – в системе Vivado традиционные текстовые языки программирования адаптированы к задачам проектирования электронных схем. Благодаря этому профессиональные программисты теперь могут разрабатывать не только надстройку над предоставленной им технической базой, но и саму базу. В Vivado усовершенствован и процесс проектирования, что позволяет существенно сократить время, необходимое для конфигурирования FPGA.

Корпорация National Instruments, используя современные микроэлектронные компоненты, разрабатывает все более совершенные функциональные блоки для систем измерения, тестирования и управления. На основе новых ре-

конфигурируемых модулей ввода-вывода, контроллеров и модульных измерительных приборов стало возможным создавать системы, обладающие большей производительностью и заметно расширенным функционалом. Библиотеки модуля LabVIEW FPGA пополнились новыми программными компонентами для реализации сложных алгоритмов обработки данных, представленных не только целыми числами, но и данными в формате чисел с фиксированной и плавающей запятой. Появились библиотеки функций для решения прикладных задач цифровой радиосвязи, обработки изображений, электроэнергетики, усовершенствованы средства работы с каналами ввода-вывода, инструменты отладки кода, выполняемого в FPGA, функции для упрощения интерфейса с программным обеспечением верхнего уровня и многое другое. Хотя процесс проектирования структуры FPGA в новой системе проектирования Vivado существенно изменился, тем, кто разрабатывает реконфигурируемые системы в LabVIEW FPGA, по-прежнему не обязательно знать специфические особенности этого процесса. Достаточно щелчка кнопки, чтобы инициировать компиляцию созданной структуры кристалла, а LabVIEW FPGA выберет необходимый компилятор – ISE Xilinx или Vivado, и спустя необходимое для компиляции время двоичный файл будет загружен и начнет выполняться в FPGA.

В предлагаемой читателям книге рассмотрены далеко не все упомянутые новшества LabVIEW FPGA, это потребовало бы значительно увеличить ее объем. Первое издание [В-1] было напечатано ограниченным тиражом, и не все желающие смогли ознакомиться с принципами проектирования реконфигурируемых систем. Поэтому в этом издании сохранены главы, в которых описаны предпосылки появления этого перспективного класса систем, техника программирования в LabVIEW, технические средства и особенности разработки на их основе систем традиционной архитектуры. Содержание этих глав обновлено и переработано.

Системы цифровой радиосвязи, а также вопросы программно-технического моделирования при разработке сложных объектов, кратко описанные в главе «Примеры применения технологий реконфигурируемого ввода-вывода» в первом издании книги, в настоящее издание не вошли. Принципы построения реконфигурируемых систем и инструментарий, с помощью которого их разрабатывают, одни и те же для любых областей техники, а специфические особенности, связанные преимущественно с алгоритмами обработки данных и областью применения систем, заслуживают более полного изложения в отдельных книгах.

В дополнительной к предыдущему изданию 14-й главе описывается заключительный этап проектирования систем – этап создания и развертывания исполняемых приложений.

Немного о том, как выбирался материал, и о стиле его оформления. Вначале приводятся краткие сведения об аппаратных и/или программных компонентах систем, инструментах, необходимых для их разработки. Затем на простых примерах подробно рассматривается процесс разработки и отладки типовых структур, после чего приводится описание средств и результатов испытаний созданных приложений.

В каждом примере раскрываются только те свойства используемых компонентов и те особенности их применения, которые необходимы для решения

соответствующей задачи. В последующих примерах представление о доступных свойствах тех же компонентов расширяется. При этом описание техники программирования сводится не к абстрактному перечислению компонентов, свойств и методов, пусть даже упорядоченному, а к описанию практических приемов решения конкретных задач. Такой формат не предполагает изложения всей информации по каждой из рассматриваемых тем и всех вариантов решения той или иной задачи (это превратило бы книгу в сборник статей справочной системы LabVIEW Help и во много раз увеличило бы ее объем).

Ознакомившись с базовыми понятиями и разобравшись с тем, как реализован пример, читатель, при необходимости, сможет самостоятельно узнать об иных способах и средствах решения задачи. Опыт преподавания профильных дисциплин студентам Новосибирского технического университета, слушателям курсов повышения квалификации, проводимых для сотрудников научных и образовательных организаций, а также для специалистов промышленных предприятий, подтверждает эффективность подобной методики обучения технологиям проектирования [В-2].

Предложение подготовить второе издание книги поступило в 2015 году. Но работу над ним пришлось отложить, т.к. к тридцатилетию LabVIEW (в 2016 г.) ожидался выпуск более совершенной системы нового поколения LabVIEW NXG. Это не означало свертывания проекта LabVIEW предыдущего поколения. Корпорация National Instruments продолжила его развивать, и те, кто приобретал LabVIEW, получали LabVIEW NXG бесплатно, что должно было способствовать ускорению внедрения этой системы. Но в 2016 году в состав первой версии LabVIEW NXG модуль LabVIEW FPGA еще не был включен, а его расширение в последующие годы отставало от продолжающего развиваться модуля LabVIEW FPGA предыдущего поколения. Поэтому, чтобы не терять время, было решено бóльшую часть второго издания книги посвятить уже привычной, завоевавшей широкую популярность редакции системы LabVIEW, дополняя соответствующие разделы информацией, касающейся особенностей LabVIEW NXG.

Системы LabVIEW и LabVIEW NXG очень близки по своим функциональным возможностям и производительности, а усовершенствования в LabVIEW NXG, судя по всему, не столь принципиальны, чтобы тратить ресурсы на поддержку и развитие двух очень сложных систем. И в 2021 году корпорация National Instruments объявила о завершении работы над проектом LabVIEW NXG и прекращении расширенной поддержки всех пяти версий в 2022 году [В-3]. Мы уверены, что и корпорация, и многочисленные пользователи ее продукции от этого только выиграют – LabVIEW и все ее модули теперь будут развиваться ускоренными темпами.

От начала работы над рукописью до издания книги прошло немало времени. Многим читателям уже доступна более актуальная версия LabVIEW, чем та, на которую ссылается автор. Но это совершенно не важно – главные принципы, объекты и инструменты LabVIEW, без сомнения, остались неизменными. И это является еще одним свидетельством прочности фундамента, на котором выстроена система LabVIEW.

Представляется, что информации, предлагаемой читателю этой книги, будет достаточно для ускоренного освоения перспективной технологии проек-

тирования реконфигурируемых информационно-измерительных и управляющих систем. Книга может быть рекомендована инженерам, начинающим применять LabVIEW и особенно LabVIEW FPGA в своей практической деятельности, а в качестве учебного пособия – студентам соответствующих специальностей.

Ефим Давыдович Баран,
преподаватель кафедры систем сбора и обработки данных
Новосибирского государственного технического университета,
руководитель авторизованного регионального учебного центра
«Центр технологий National Instruments»
г. Новосибирск, Россия

Предисловие А.Ю. Романова

Дорогие друзья!

Книга, которую вы держите в руках, продолжает серию книг «Книжная полка истового инженера», которую выпускает издательство «ДМК Пресс» совместно с Московским институтом электроники и математики им. А.Н. Тихонова НИУ ВШЭ при поддержке группы компаний YADRO (yadro.com).

Если вы интересуетесь цифровой электроникой, разработкой на ПЛИС, проектированием на языках описания аппаратуры Verilog или VHDL, то вы, скорее всего, уже знакомы с книгами по цифровому синтезу, такими как: Д. Харрис и С.Л. Харрис «Цифровая схемотехника и архитектура компьютера», «Цифровая схемотехника и архитектура компьютера: RISC-V», «Цифровая схемотехника и архитектура компьютера. Дополнение по архитектуре ARM»; «Цифровой синтез: практический курс» (под ред. А.Ю. Романова и Ю.В. Панчула), Ф. Бруно «Программирование FPGA для начинающих» и др. (воспользуйтесь QR-кодами, чтобы узнать об упомянутых книгах).




Эта книга расширяет тематику и раскрывает особенности разработки на FPGA в среде проектирования от National Instruments LabVIEW, а именно в ее расширении LabVIEW FPGA.

О чем эта книга?

В **первой главе** кратко рассказывается о ПЛИС и их разновидностях. Во **второй главе** дается краткий экскурс в устройства ввода-вывода, используемые в продукции National Instruments. **Третья глава** посвящена виртуальным измерительным приборам, создаваемым в LabVIEW. В **четвертой и пятой главах** дается общее представление об организации среды проектирования LabVIEW и принципах разработки в этой среде. В **шестой главе** рассказывается, для чего нужны реконфигурируемые системы, а в **седьмой главе** – какие средства LabVIEW предназначены для разработки реконфигурируемых систем. В **восьмой главе** описан типичный маршрут разработки реконфигурируемых систем. **Девятая глава** посвящена описанию, как управлять FPGA VI и создавать Host VI. В **десятой главе** на примерах показано, как можно обрабатывать

данные в LabVIEW FPGA. В **одиннадцатой главе** – как тестировать разработанные приложения, как разрабатывать стандартные интерфейсы для связи с периферийными устройствами. **Двенадцатая глава** посвящена модулям R-серии, а также вопросам измерения неэлектрических величин. В **тринадцатой главе** приведен пример разработки встраиваемой системы на платформе cRIO. А в **четырнадцатой главе** кратко рассказывается, как создать сборку проекта и инсталлятор для развертывания приложений на пользовательских компьютерах. В завершающей **пятнадцатой главе** рассказывается, как проекты, разработанные в LabVIEW FPGA, перенести в другие САПР'ы и подготовить для компиляции под ПЛИС других производителей.

Что послужило причиной появления этой книги?

Дело в том, что на русском языке книг по LabVIEW FPGA нет. Существовала книга Е.Д. Барана **«LabVIEW FPGA. Реконфигурируемые измерительные и управляющие системы»** 2009 г., но она была выпущена небольшим тиражом и за более чем 10 лет уже порядком устарела. В то же время LabVIEW широко преподается в российских вузах, и материалов по данной тематике не хватает. У нас в МИЭМ есть отдельная лаборатория, оборудованная продукцией National Instruments, в т.ч. myRIO на основе FPGA. На них студенты изучают основы цифровой электроники, прежде чем попадают в мою лабораторию Систем автоматизированного проектирования (САПР), где уже знакомятся с проектированием на отладочных платах с ПЛИС от компаний Altera (Intel FPGA)/Xilinx и их САПР'ами. Но на самом деле я всегда считал такое несколько подчиненное положение LabVIEW несправедливым. Потому что LabVIEW – это очень мощная и разносторонняя САПР, обладающая большим функционалом и возможностями. Главный ее недостаток – это жесткая привязка к продуктам National Instruments. Но вот как раз в случае с проектами на ПЛИС в LabVIEW FPGA в последние несколько лет появилась надстройка IP Export Utility. Этот инструментарий превращает LabVIEW в мощный инструмент высокоуровневого синтеза (HLS – High Level Synthesis) наподобие MATLAB, но только со своим языком графического проектирования и возможностью аппаратной отладки проектов и создания цифровых панелей управления, после чего проекты могут быть экспортированы на нужную разработчику платформу или даже скомпилированы под ASIC с помощью открытого маршрута проектирования.

Поэтому, когда ко мне обратились из издательства «ДМК Пресс» с предложением выступить редактором нового переиздания книги Е.Д. Барана по LabVIEW FPGA, я отложил работу над другими книгами и с огромным энтузиазмом взялся за этот проект. Оказалось, что не все так просто: книга претерпела глубокую переработку, получив тысячи правок и дополнений, обзавелась новой главой (как раз про LabVIEW FPGA IP Export Utility), стала современней и более дружественной для читателя. И теперь, по прошествии 9 месяцев работы, я, уже в статусе второго автора, рад представить ее на ваш суд.

Что почитать еще?

Если вы осилите эту книгу, то я советую также ознакомиться с книгой Terry Stratoudakis «Introduction to LabVIEW FPGA for RF, Radar, and Electronic Warfare Applications» [В-4]. Она сама по себе интересна, но еще в ней есть обширный спи-

сок литературы, который собран в репозитории: *https://github.com/LVFPGABOOK* [В-5]. Также обратите внимание на список литературы, который есть у этой книги. Там собраны практически все мануалы и инструкции, а еще другие книги преимущественно в открытом доступе, которые вам могут пригодиться.

Насколько сложно сделать хорошую книгу?

Об этом я рассказываю в своем выступлении на IV конференции FPGA разработчиков FPGA Systems 2023.1: *https://www.youtube.com/watch?v=7K5l_rIL8-o*.

Надеюсь, эта книга тоже вышла хорошей и **будет интересна широкому кругу читателей** от студентов технических вузов до разработчиков научного и промышленного оборудования, которые стремятся расширить свой кругозор и познакомиться с новыми САПР. Для знакомства с ней достаточно знать основы программирования, иметь хотя бы некоторое понимание, что такое ПЛИС и для чего они нужны, а также стремление развиваться.

В добрый путь!

Александр Юрьевич Романов,
к.т.н., доцент ДКИ МИЭМ НИУ ВШЭ,
преподаватель курсов «Проектирование систем на кристалле»,
«Системное проектирование цифровых устройств»
и «Системы искусственного интеллекта»
г. Москва, Россия

Благодарности

Прежде всего благодарим читателей первого издания книги, приславших свои отзывы и замечания, мы постарались их учесть.

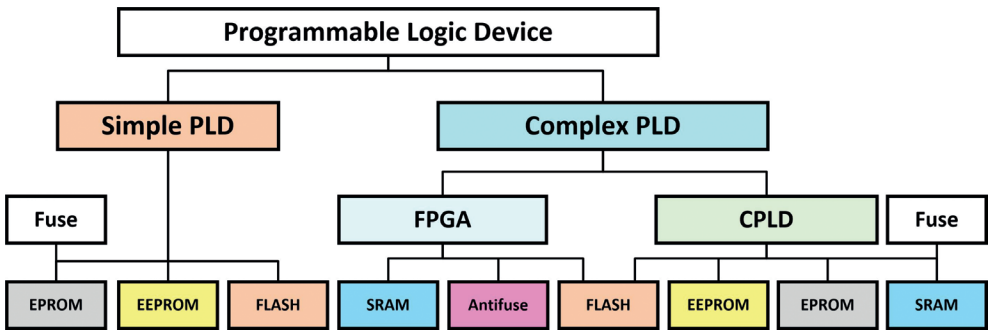
Особая благодарность всем сотрудникам Российского представительства корпорации National Instruments, оказывавшим содействие в практическом внедрении технологий реконфигурируемого ввода-вывода, инициировавшим работы по популяризации этого перспективного направления путем издания серии книг о LabVIEW, которую пополнит и настоящая книга.

Рукопись книги, которую вы сейчас читаете, подготовили и прочитали два автора, научный редактор, один корректор и еще несколько рецензентов, но, безусловно, книга не идеальна. Мы будем очень признательны тем внимательным читателям, которые обнаружат в данном издании какие-либо ошибки или опечатки и сообщат о них авторам: **baran@corp.nstu.ru**, **a.romanov@hse.ru** или **dmkpress@gmail.com** (книги постоянно перепечатываются, и в каждом новом тираже все найденные ошибки и недочеты исправляются).

Авторы благодарны всем, кто помогал готовить эту книгу. Вот некоторые из тех, кто помогал сделать эту книгу лучше: **Максим Колпаков**, **Оксана Батонова**, **Денис Палуха**, **Максим Ельцов**.

Глава 1

Программируемые логические интегральные схемы



В эту главу входят следующие параграфы:

С развитием микроэлектронных технологий и увеличением степени интеграции элементов на кристалле все очевиднее становилось противоречие между потребностями создавать компактные специализированные устройства из небольшого количества компонентов и нерентабельностью расширения номенклатуры выпускаемых массовым тиражом типовых логических микросхем и функциональных блоков. Действительно, уже с конца 60-х годов прошлого века стало возможным на одном кристалле размещать десятки и сотни логических вентилей, но чем выше становилась плотность размещения компонентов на новом кристалле, тем больше возрастала его специализация и тем меньше становилась относительная доля объема его выпуска.

Появление универсальных логических устройств, микропроцессоров, конечная функция которых определялась загружаемой в них программой, казалось, сблизило возможности изготовителей микросхем, владеющих высокотехнологичными средствами производства, с одной стороны, и потребности и возможности разработчиков прикладных устройств и систем – с другой. Но инженеры по-прежнему вынуждены были для решения своих задач использовать компоненты высокой (по меркам того времени) степени интеграции. Процессор был основным функциональным блоком, а десятки логических элементов малой степени интеграции использовались для того, чтобы состыковать между собой микросхемы разных классов и типов, а также чтобы реализовать на аппаратном уровне специфические для каждого конкретного случая узлы.

Например, контроллер выполнялся на основе 4- или 8-разрядного микропроцессора. А несколько микросхем памяти содержали тысячи логических вентилей и до десятка схем более низкой степени интеграции (дешифраторов, регистров, логических элементов), необходимых для реализации системной шины, каналов ввода-вывода и т.п. Для изготовления такого контроллера требовалась печатная плата с площадью, достаточной для установки микросхем повышенной степени интеграции (микропроцессор, память), размещения большого количества микросхем малой степени интеграции и всех необходимых печатных проводников. Сборка относительно несложных логических компонентов контроллера проводилась с помощью паяльника, в то время как существенно более сложные микропроцессор и блоки памяти поступали в виде готовых микросхем. При этом стоимость монтажа оказывалась непропорциональной сложности устройства, а его надежность определялась надежностью элементов малой степени интеграции, количеством их выводов и соединительных проводников на печатной плате.

Таким образом, разработчики цифровой аппаратуры, получив возможность использовать достаточно мощные и универсальные процессорные компоненты, вынуждены были по-прежнему применять простые и тоже универсальные элементарные логические схемы, собирая их в специализированные блоки, без которых невозможно было спроектировать и изготовить устройство, решающее задачу измерений и обработки информации, управления или тестирования.

Назревала необходимость усовершенствования технологии проектирования и производства конечных изделий, для того чтобы приблизить ее к тех-

нологии проектирования и производства микроэлектронных компонентов. Массовое производство микросхем, где связи между элементами в кристалле создавались с помощью масок, было передовым. Но использовать эту технологию в многочисленных лабораториях разработчиков цифровых устройств невозможно, т.к. изготовление масок – весьма трудоемкий и дорогой процесс. Необходимо было придумать более доступный механизм произвольного соединения элементов кристалла между собой и с внешними выводами микросхемы, а разместить на кристалле необходимый набор типовых логических элементов (десятков или сотен) к тому времени проблемой уже не являлось.

Другими словами, разработчикам требовалась технология, позволяющая более простыми и в то же время более эффективными средствами **создавать собственную микросхему**, которая могла бы заменить горсть универсальных микросхем малой степени интеграции, объединяемых в специализированный блок. Такая возможность была реализована в программируемых логических интегральных структурах – **ПЛИС** (**PLD – Programmable Logic Device**). На кристалле ПЛИС размещались простые логические вентили различных типов и объединяющие их регулярные шины проводников. Подобная микросхема представляла собой некоторую универсальную заготовку, в которой разработчик доступными средствами мог удалить ненужные связи между вентилями, изменяя тем самым функцию ПЛИС и, по существу, создавая уникальное, необходимое только ему специализированное логическое устройство, сохраняя почти все преимущества его интегрального выполнения.

1.1. Простые программируемые логические устройства

Первые кристаллы, структуру и функции которых мог определять пользователь, – программируемые постоянные запоминающие устройства – **ППЗУ** (**PROM– Programmable Read Only Memory**). ППЗУ состоит из двух матриц логических элементов – матрицы элементов «И» и матрицы элементов «ИЛИ».

Входы элементов «И» соединяются с внешними входами микросхемы через повторитель или инвертор, а входы элементов «ИЛИ» – с выходами элементов «И» [1-1]. Таким образом, на выходах элементов «ИЛИ», являющихся внешними выходами микросхемы, формируется логическая сумма произведений входных переменных.

В показанной на рис. 1-1 схеме переменные, подаваемые на входы адреса, в дешифраторах строк образуют все возможные конъюнкции – функции «И» от всех входных переменных. Объединением выходов конъюнкторов в столбцах матрицы реализуются логические функции «ИЛИ» – отключением определенных строк от столбцов определяются выходные функции ППЗУ.

От изготовителя ППЗУ поступало «чистым» – были подключены все входы всех элементов «И» и «ИЛИ», но пользователь мог удалять ненужные соединения, адресно разрушая их в специальном режиме. В качестве разрушаемых соединений (перемычек) использовались выжигаемые током проводники или пробиваемые *p*-*n*-переходы (диоды), сохраняемые же связи и являлись запоминающими элементами и определяли логическую функцию для каждого выхода

ППЗУ. Следует отметить, что использовались и иные способы программирования ППЗУ – не удалением ненужных из предварительно созданных межсоединений, а наоборот – созданием нужных соединений. Но подобные технологические нюансы для разработчика прикладных устройств были непринципиальны.

Различают ППЗУ двух основных типов:

- микросхемы, получившие название **PROM**, в которых можно изменять только подключения входов элементов «ИЛИ». Матрица связей элементов «И» в них фиксирована (пример на рис. 1-1);
- микросхемы типа **PAL/GAL** (**Programmable/Generic Array Logic**). В них программируются связи элементов «И», а соединения элементов «ИЛИ» изменению не подлежат.

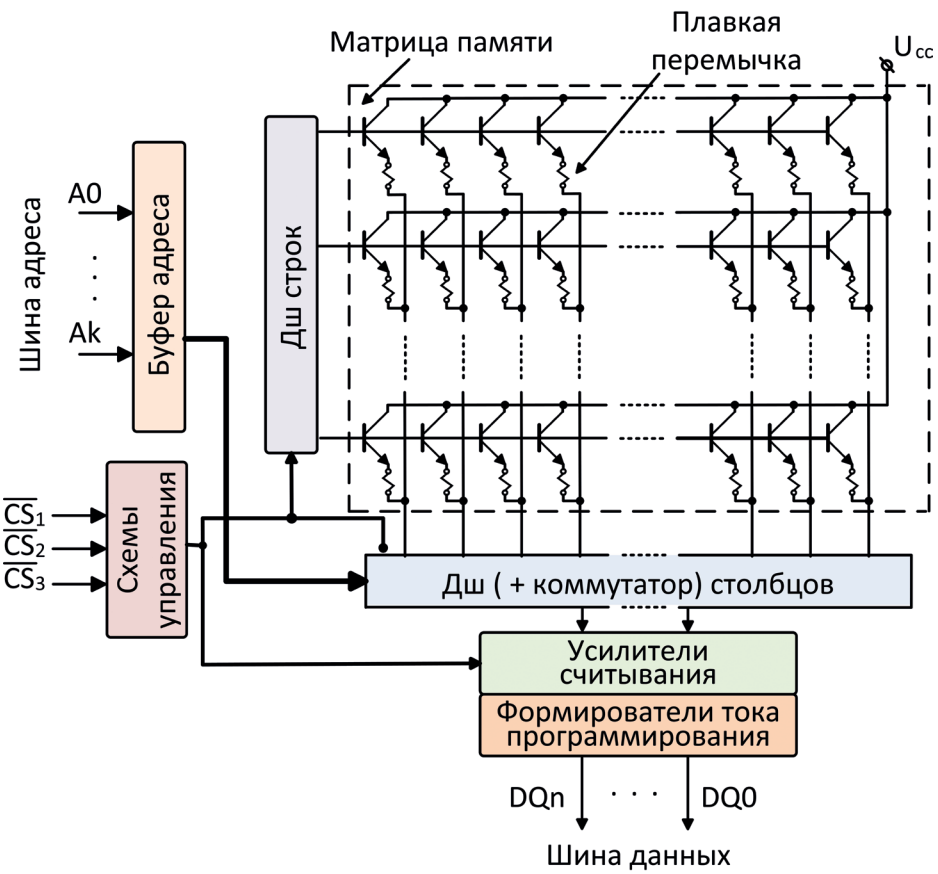


Рис. 1-1. Структура ППЗУ

Набор из N произвольных логических функций от k входных переменных может быть реализован в одной микросхеме ППЗУ, содержащей N k-входовых логических элементов. Для этого достаточно записать эти функции в совершенной дизъюнктивной нормальной форме и задать их таблицы истинности в программаторе.

В программируемых ПЗУ площадь кристалла расходуется неэкономно – для всех возможных конъюнкций и дизъюнкций входных и промежуточных переменных должны быть зарезервированы входы элементов «И» и «ИЛИ», значительная часть которых при программировании отключается.

Более рационально логические функции реализуются на базе **ПЛМ** – программируемых логических матриц (**PLA – Programmable Logic Array**) [1-2, 1-3]. В ПЛМ могут быть запрограммированы обе матрицы логических элементов – «И» и «ИЛИ», при этом количество входов конъюнкторов и дизъюнкторов может быть уменьшено без существенных потерь сложности реализуемых логических функций (рис. 1-2).

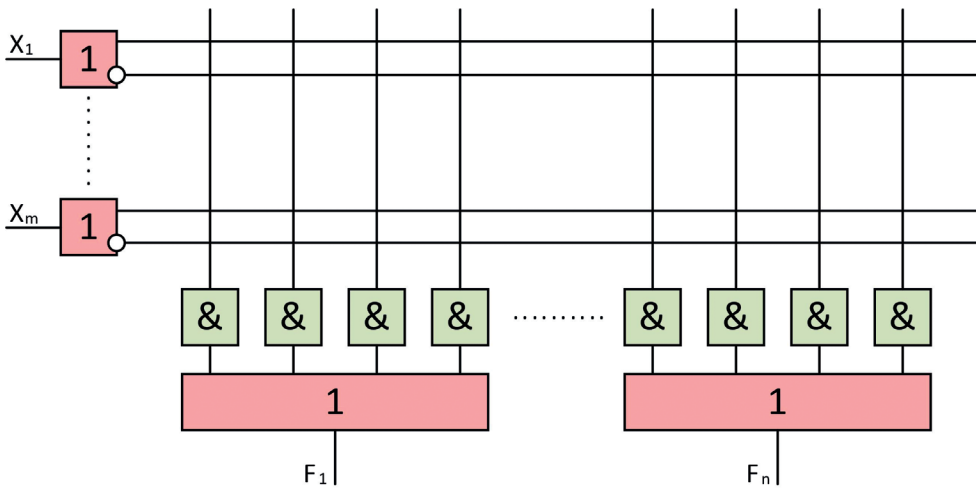


Рис. 1-2. Структура ПЛМ

ПЛИС, рассмотренные выше (PROM, PAL, GAL, PLA) дали разработчикам определенную свободу в реализации проектов, обеспечили возможность уменьшения количества компонентов, повышения надежности изделий и снижения их стоимости. Вычислительные устройства стали создавать не только на основе микропроцессоров с жесткой системой команд, но и на базе микропроцессорных секций с микропрограммным управлением, причем набор команд уже определялся самим разработчиком, который записывал микропрограммы в ПЛИС.

Таким же образом проектировались специализированные комбинационные устройства – преобразователи кодов, шифраторы и дешифраторы, а ПЛИС совместно с элементами памяти (триггерами, регистрами, счетчиками) стали применять для разработки быстродействующих устройств микропрограммного управления различными объектами.

Дальнейшее развитие технологии ПЛИС позволило включить в них элементы памяти. Это перевело разработку на качественно новый уровень, когда стало возможно создавать не только простые комбинационные устройства, но и законченные последовательностные цифровые автоматы, полностью реализующие требуемую диаграмму состояний-переходов без использования дополнительных элементов малой и средней степени интеграции. Достаточно наглядное представление о возможностях, предоставляемых ПЛИС

с памятью, дает приведенная на рис. 1-3 схема популярной микросхемы **PAL22V10**, клоны которой выпускают до сих пор многие компании [1-4, 1-5].

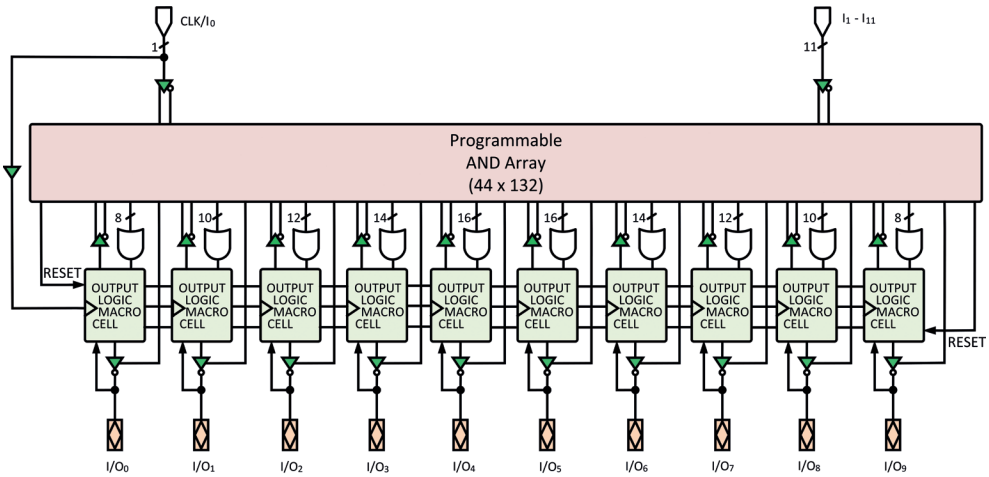
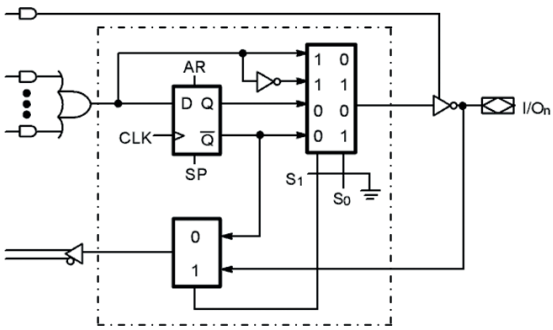


Рис. 1-3. Структура ПЛИС с элементами памяти

Эта микросхема, относящаяся к классу PAL, содержит программируемую матрицу элементов «И» (**Programmable AND Array**), на входы которых $\mathbf{I_1 \div I_{11}}$ поданы 11 переменных (или их инверсии) с внешних входов ПЛИС и 10 внутренних переменных (или их инверсии), ассоциируемых с внешними входами/выходами ПЛИС. Выходы элементов «И» объединяются в линейке из 10 непрограммируемых элементов «ИЛИ» с фиксированным количеством входов – от 8 до 16. Сигналами с выходов элементов «ИЛИ» управляются выходные макроячейки (**OUTPUT LOGIC MACROCELL**), каждая из которых содержит тактируемый D-триггер с входами сброса и установки.

Макроячейка может быть сконфигурирована для работы в одном из 4 режимов, выбираемых с помощью программируемых перемычек **S0** и **S1** (рис. 1-4). Этими перемычками основной мультиплексор ячейки настраивается таким образом, что на выходной контакт ПЛИС поступает сигнал, реализуемый комбинационной логикой (матрицами «И» и «ИЛИ»), или тот же сигнал, зафиксированный D-триггером по приходу синхроимпульса **CLK**. Выходной сигнал ($\mathbf{I/O_n}$) с помощью дополнительного мультиплексора может быть возвращен во входную комбинационную схему в качестве сигнала обратной связи для текущей макроячейки или в качестве дополнительной логической переменной – для остальных макроячеек.

Любой выход может быть переведен в высокоимпедансное состояние и использован как вход или двунаправленный вход/выход ПЛИС. Перечень полезных свойств этой микросхемы расширяют возможности сброса элементов памяти в исходное состояние по включению питания, загрузки в регистр произвольного кода через внешние контакты (что улучшает тестопригодность создаваемых на основе этой ПЛИС устройств), а также возможность установки защиты от несанкционированного копирования внутренней структуры ПЛИС – интеллектуальной собственности (**Intellectual Property**) разработчика.



S0	S1	Выходной сигнал
0	0	С выхода триггера, инверсный
0	1	С выхода триггера, прямой
1	0	Комбинационный, инверсный
1	1	Комбинационный, прямой

Рис. 1-4. Макроячейка PAL22V10

Микросхемы типа PAL с элементами памяти, вместе с ПЛИС комбинационного типа PROM, PAL, GAL, в последующем были отнесены к классу простых программируемых логических устройств – **SPLD** (**Simple Programmable Logic Device**).

Отличительные признаки SPLD:

- каждая макроячейка имеет свой внешний выход и свой собственный триггер;
- в микросхеме SPLD реализовано не менее двух макроячеек;
- обычно все макроячейки одинаковые;
- логическая функция макроячейки описывается одним логическим термом;
- логическая функция макроячейки реализуется матрицами элементов «И» и «ИЛИ».

К достоинствам SPLD обычно относят простоту проектирования специализированных устройств, постоянное и, как правило, одинаковое время прохождения сигналов с входов на выходы, возможность замены одной или несколькими микросхемами SPLD достаточно большого количества типовых микросхем малой и средней степени интеграции.

Основные недостатки SPLD – неэффективное использование ресурсов (логических вентилей) и, как следствие, проблематичность создания на их основе сложных цифровых устройств.

1.2. Технологии программирования ПЛИС

Последовательное увеличение степени интеграции микросхем и функциональной насыщенности реализуемых в кристалле ячеек, которые мог сконфигурировать разработчик, сопровождалось совершенствованием технологии программирования ПЛИС. Поэтому прежде чем продолжить обзор архитектур ПЛИС, в этом разделе будут рассмотрены технологии программирования, которые развивались параллельно с технологиями производства микросхем.

В первых семействах SPLD (PROM, PAL / GAL, PLA), выпускаемых на основе биполярных полупроводниковых элементов, однажды реализованная пользователем функция не могла быть изменена, структура соединений логических вентилей в ПЛИС записывалась однократно [1-5, 1-6]. Из-за этого если в процессе отладки вновь разработанного устройства обнаруживались ошибки, то

для их исправления необходимо было запрограммированную («прошитую)» с ошибками ПЛИС извлечь из печатной платы и заменить на исправную. То есть по-прежнему разработчик был вынужден использовать паяльник (хоть и в меньшей степени, чем раньше) или, если это допускалось с точки зрения надежности изделия, монтировать микросхему ПЛИС на печатной плате в гнезде (сокете), позволяющем при необходимости оперативно ее заменить.

Совершенно новое качество проектирования было достигнуто после перехода на кристаллы с униполярными полупроводниковыми компонентами, в которых вместо **однократно** разрушаемых (создаваемых) связей-перемычек стали применять запоминающие элементы на МОП-транзисторах с изолированным затвором. Заряд, создаваемый на затворе МОП-транзистора при программировании таких структур, сохранялся годами и не исчезал при отключении питания. Заряд этот можно снять, восстановив тем самым исходное состояние ячейки памяти, а при необходимости вновь записать – т.е. появилась возможность **неоднократного** перепрограммирования ПЛИС. Технологии производства таких изделий разработаны в 1971 г. фирмой **Intel**, а в 1974 г. – фирмой **Toshiba** и реализованы в **перепрограммируемых ПЗУ** (**ППЗУ**).

Запись информации в РППЗУ производится в специальном режиме импульсами напряжения повышенной амплитуды, а стирание осуществляется двумя способами:

- ультрафиолетовым излучением. Такие микросхемы называются **UV-EPROM** (**Ultraviolet Erasable PROM**), УФ РППЗУ;
- электрическим напряжением противоположной полярности – микросхемы **EPROM** (**Erasable Programmable ROM**). В отечественной литературе их обычно называют ЭСППЗУ (электрически стираемые программируемые ПЗУ).

Для стирания информации ультрафиолетовым излучением в корпусах микросхем РППЗУ создается закрытое кварцевым стеклом окошко, через которое облучается кристалл.

Теперь инженеру не нужно было иметь запас однократно программируемых устройств, чтобы заменить неправильно спроектированную микросхему. Вновь разработанное прикладное изделие можно было запускать в производство, комплектуя его набором универсальных типовых элементов и достаточно дешевыми микросхемами «собственного изготовления» – специфическими только для данного изделия и запрограммированными самим разработчиком устройства. При этом по-прежнему необходимо было предусматривать возможность извлечения микросхемы РППЗУ из печатной платы для стирания и последующего перепрограммирования, а самое главное – надо было оснастить рабочее место специальным программатором и устройством для стирания ультрафиолетовым излучением.

Следующий шаг в развитии программируемых пользователем компонентов был сделан, когда технология позволила реализовать стирание/перезапись информации внутри кристалла без использования внешнего специального оборудования. Подобные микросхемы – **Electronically Erasable PROM** – были разработаны фирмой **Intel** в 1979 г. и, по существу, исключили из процесса разработки, изготовления и отладки макетных образцов аппаратуры демонтаж исправных, но неправильно запрограммированных инженером компонентов.

Наилучшими качествами в семействе микросхем класса Electronically Erasable PROM обладает энергонезависимая память типа **Flash**, предложенная в 1984 г. фирмой **Toshiba** (а с 1988 г. развиваемая фирмой **Intel** и рядом других фирм), которая широко применяется сейчас в твердотельных накопителях информации большой емкости. Flash-память характеризуется большим количеством циклов перезаписи (до 10^6 и более), высоким быстродействием. Некоторые разновидности такой памяти допускают возможность стирания/модификации содержимого произвольной ячейки. Благодаря этим качествам, высокой технологичности в производстве, а следовательно и более низкой стоимости Flash-память практически вытеснила остальные разновидности программируемой энергонезависимой памяти, используемые при разработке цифровых устройств и систем.

1.3. Пограничное сканирование и JTAG-интерфейс

Упомянем еще о некоторых задачах, которые приходилось решать в процессе развития микроэлектроники и системотехники. С увеличением степени интеграции ПЛИС, как, собственно, и любых других интегральных схем, обострялась и становилась все более важной проблема тестирования компонентов и систем, создаваемых на их основе. Радикально усложнились отладка и верификация цифровых автоматов при массовом использовании ПЛИС в новых изделиях и массовом производстве систем, содержащих микросхемы высокой степени интеграции. Тестирование подобных компонентов и устройств требует значительного времени как на подготовку тестов, так и на их проведение. Необходимо также специализированное дорогостоящее оборудование и программное обеспечение, сопоставимое по сложности и трудоемкости разработки с прикладным (целевым) программным обеспечением.

Эффективным решением проблемы тестирования стала взятая на вооружение разработчиками концепция проектирования **тестопригодных** или легко тестируемых компонентов и систем (**Design For Testability – DFT**). Из нескольких опробованных на практике подходов к созданию тестопригодных устройств наиболее универсальным и экономным был признан подход, основанный на методе тестирования, получившем название **метода пограничного сканирования** (**Boundary-Scan**). Суть метода заключается в том, что в режиме тестирования обеспечиваются возможности:

- изоляции любого компонента от всех остальных компонентов системы;
- доступа к любому изолированному компоненту для его автономного тестирования;
- выполнения отдельной процедуры проверки межсоединений компонентов.

При этом в основном режиме работы и компоненты, и система в целом по-прежнему функционируют по прямому назначению.

Концепция пограничного сканирования в 1990 г. была закреплена международным стандартом **IEEE-1149.1 – IEEE Standard Test Access Port and Boundary-Scan Architecture**, определяющим архитектуру устройств с пограничным сканированием, а также структуру и функционирование специального тестового порта (**Test Access Port – TAP-порт**) [1-7]. Этот порт обычно на-

зывают **JTAG**-интерфейсом по наименованию объединенной рабочей группы по тестированию **Joint Test Action Group**.

Стандарт стал руководством к действию для подавляющего большинства производителей электронных компонентов, разработчиков и изготовителей цифровых систем. Теперь в каждую микросхему, разработанную в соответствии со стандартом IEEE-1149.1, встроены TAP-порт и совокупность **JTAG-ячеек**, по одной на каждый из выводов микросхемы. Они образуют регистр пограничного сканирования. JTAG-ячейка позволяет:

- подключить ядро микросхемы (**Internal Core Logic**) к внешним выводам для основного режима работы микросхемы в составе устройства;
- подключить к внешним выводам микросхемы регистр пограничного сканирования и отключить от них ядро микросхемы для тестирования соединений между микросхемами с помощью внешнего тестера;
- отключить внешние выводы микросхемы от ядра и подключить к ядру регистр пограничного сканирования для тестирования ядра.

Таким образом, регистр пограничного сканирования может работать в нескольких режимах. В основном режиме работы микросхемы регистр «прозрачен» для внешних сигналов, а в тестовом – ядро микросхемы изолировано от внешней среды. При этом на входы ядра тестовые сигналы поступают с выходов регистра пограничного сканирования, а выходные сигналы ядра могут быть записаны в этот же регистр. В тестовом режиме загрузка и считывание данных в регистр осуществляются последовательно через TAP-порт (рис. 1-5).

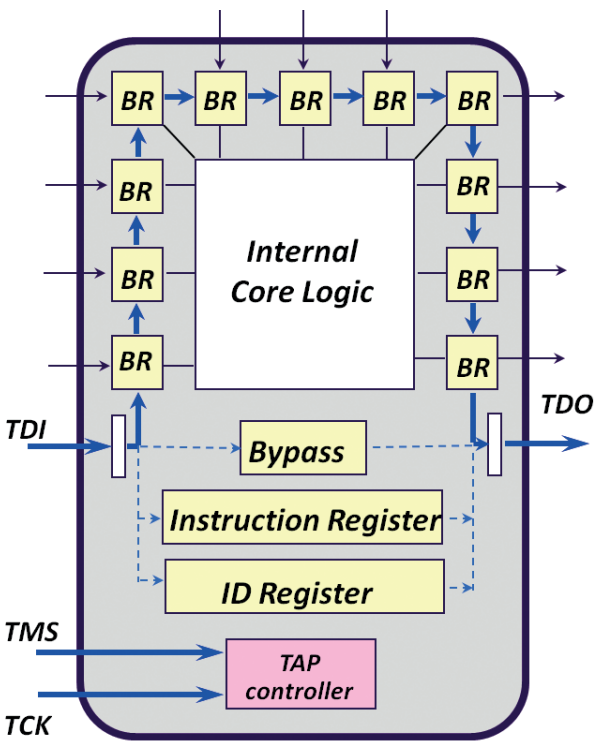


Рис. 1-5. TAP-порт

TAP-порт содержит 4 обязательные и одну дополнительную линию для передачи сигналов:

- **TDI – Test Data Input** – вход последовательного потока данных. В зависимости от состояния интерфейса в микросхему передаются либо команды для управления JTAG-ячейками, либо тестовые данные;
- **TDO – Test Data Output** – выход последовательного потока данных. Через эту линию в тестер передаются результаты выполнения команд и состояния JTAG-ячеек;
- **TCK – Test Port Clock** – последовательность тактовых импульсов для синхронизации последовательного обмена. Этот сигнал вырабатывается тестером и необязательно должен быть фиксированной частоты. Максимальная частота следования импульсов TCK для многих микросхем находится в пределах 5÷10 МГц. Стандарт не ограничивает частоту тактирования снизу;
- **TMS – Test Mode Select** – этот сигнал управляет режимами JTAG-интерфейса;
- **TRST*** – **Test Port Reset** – сброс узлов тестовой логики (сигнал необязателен). Этот сигнал не заменяет сигнал сброса ядра микросхемы, поскольку JTAG-логика управляет только JTAG-ячейками и не влияет на работу ядра.

Ячейки регистра пограничного сканирования (**Boundary Register – BR**) всех микросхем могут объединяться в один общий последовательный регистр, чтобы обеспечить доступ к любой микросхеме устройства. Для этого в каждой микросхеме предусмотрен однобитовый регистр обхода (**Bypass**). Кроме того, TAP-порт содержит **TAP Controller** и 4-битовый регистр команд **Instruction Register** для управления режимами работы, а также регистр идентификатора микросхемы **ID Register**.

Стандарт IEEE-1149.1 определяет только 4 обязательные команды, остальные 12 возможных кодов команд зарезервированы для дальнейших расширений, и их действие определяется изготовителем микросхем.

В целом реализация метода пограничного сканирования предельно проста, как с точки зрения необходимых аппаратных ресурсов, так и с точки зрения программирования. В микросхеме может потребоваться не более 4÷5 дополнительных выводов для TAP-порта и от нескольких десятков до нескольких тысяч дополнительных логических элементов и триггеров, что обычно составляет не более единиц процентов от общего количества логических вентилей на кристалле.

Исключительно важно, что тестирование методом пограничного сканирования не требует сложного и дорогого внешнего тестера [1-8] – его функции может выполнять персональный компьютер с адаптером, позволяющим использовать обычный последовательный или параллельный порт компьютера в качестве TAP-порта (рис. 1-6).

Благодаря стандарту IEEE-1149.1 качественное тестирование стало возможным не только в крупных компаниях, которые смогли заплатить миллионы долларов за современный тестер, но и в любой лаборатории, где тестером становится персональный компьютер со свободным стандартным портом.

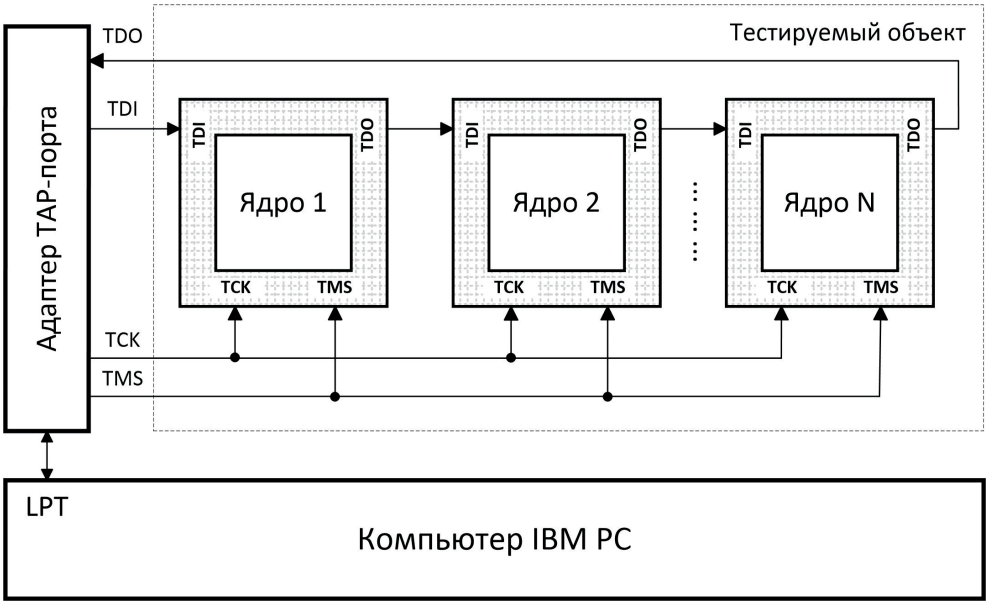


Рис. 1-6. Тестирование методом пограничного сканирования

Одновременно существенно упростилась и разработка программ тестирования вследствие регулярности структуры ПЛИС и доступности большого количества внутренних узлов синтезируемых в ПЛИС автоматов. С целью облегчения проектирования тестов и унификации тестового обеспечения были созданы специальные языки программирования высокого уровня – **Boundary Scan Description Language** (**BSDL**, язык описания устройств с пограничным сканированием), **Jam** и др.

В настоящее время JTAG-интерфейс и созданные для работы с TAP-портом языки программирования могут быть использованы **не только для тестирования ПЛИС, но и для их конфигурирования** [1-9]. Собрав схему, подобную изображенной на рис. 1-6, разработчик в считанные минуты может «загрузить» в чистую ПЛИС только что спроектированное устройство, провести его тестирование (прототипирование), а в дальнейшем, при необходимости, оперативно изменять структуру ПЛИС, находясь в одной и той же среде проектирования. Аналогично, если на печатной плате вместе с ПЛИС смонтированы другие микросхемы, поддерживающие стандарт IEEE-1459.1, например микропроцессоры или запоминающие устройства, то и в них можно загрузить программу, воспользовавшись TAP-портом, осуществить тестирование отдельных компонентов или системы в целом.

Таким образом, концепция пограничного сканирования дала в руки инженеру универсальное средство реконфигурирования аппаратных компонентов, программирования микропроцессорных и иных устройств, а также тестирования. При использовании этих средств стала доступной **технология программирования в системе** (**In-Circuit Programming**) – без паяльника и гнезд под микросхемы, благодаря которой не только упростился процесс проектирова-

ния и производства, но стало доступней и проще и необходимое специальное оборудование. Исчезла необходимость в отдельных устройствах – программаторах, устройствах стирания информации в ПЛИС, сложном и дорогостоящем тестовом оборудовании.

1.4. Сложные программируемые логические устройства

Вернемся к обзору архитектур программируемых логических схем. Вслед за простыми, SPLD, появился класс сложных программируемых логических устройств – **Complex Programmable Logic Device** (**CPLD**).

Одной из первых подобные изделия под названием **Classic Programmable Logic Device** (аббревиатура та же) выпустила на рынок компания **Altera**[1] (сейчас Intel FPGA). В качестве примера на рис. 1-7 показана структурная схема одной из микросхем семейства **CPLD – EP610** [1-10].

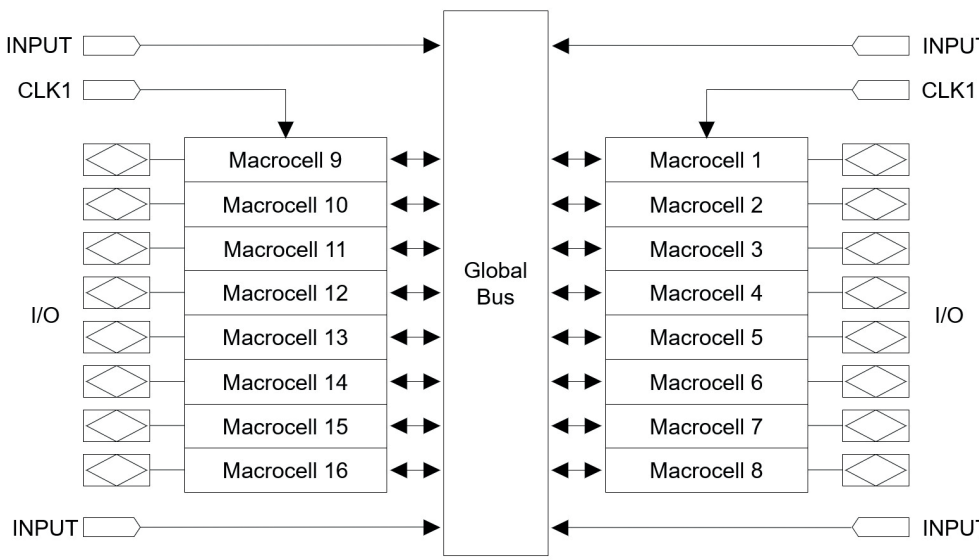


Рис. 1-7. Структурная схема Classic PLD Altera EP610

Чем сложные ПЛИС отличаются от простых?

В SPLD логические макроячейки, содержащие триггер, соединялись преимущественно с выходными контактами ПЛИС, причем количество макроячеек было невелико.

Сложные ПЛИС при бóльшем количестве макроячеек (в самой простой CPLD микросхеме типа EP610 их уже 16) обеспечивают возможность практически произвольного соединения макроячеек (**Macrocell**) друг с другом и с внешними выводами микросхемы (**I / O**) и позволяют создавать существенно более

[1] В декабре 2015 г. компания Altera приобретена корпорацией Intel за 16,7 млрд долларов. В настоящее время она является подразделением Intel и называется Intel FPGA, входит в Programmable Solutions Group (PSG).

сложные последовательностные устройства. Для организации связей внутри ПЛИС используется программируемая матрица (шина) глобальных проводников (**Global Bus**). В ПЛИС EP610, кроме 16 двунаправленных выводов (**I/O**), имеются также 4 дополнительных логических входа (**Input**) и 2 глобальных входа синхронизации (**Clk1** и **Clk2**).

Возможности реализации логических функций в CPLD, в отличие от SPLD, определяются не только связями между макроячейками через разветвленную общую шину коммутации Global Bus, но и связями, создаваемыми внутри каждой макроячейки (рис. 1-8). Элемент памяти макроячейки (**Programmable Register**) может быть сконфигурирован как D, T, SR или JK триггер с индивидуальным управлением установки в «0» или «1».

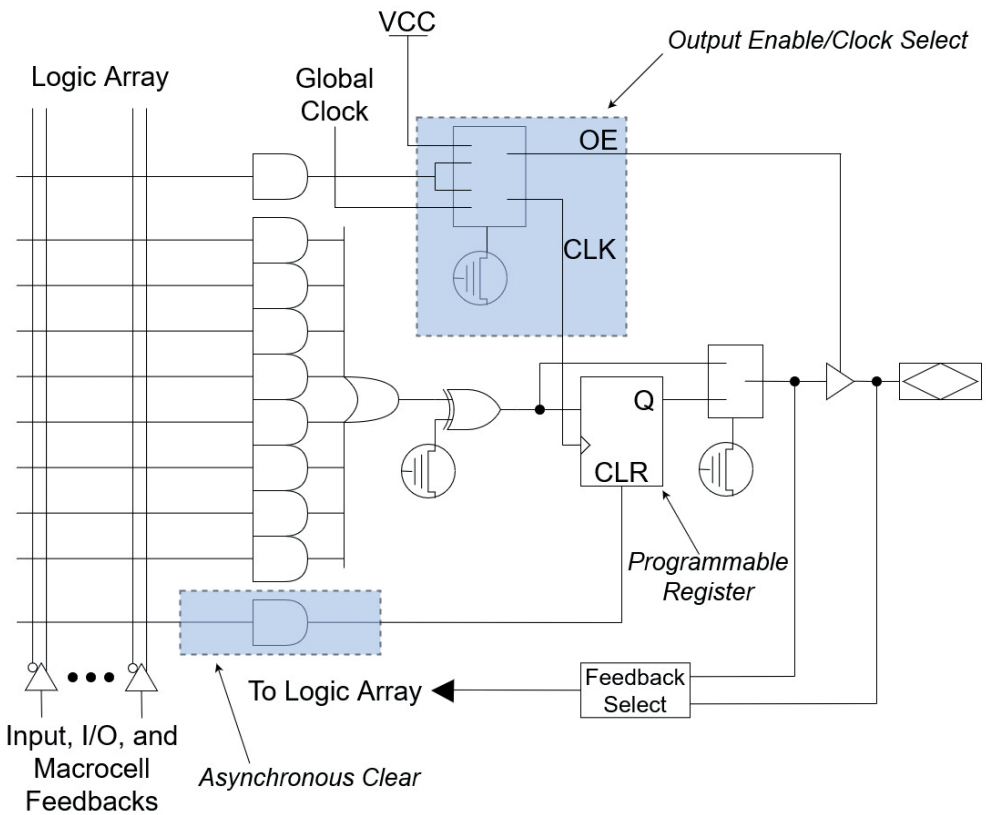


Рис. 1-8. Макроячейка EP610

Комбинаторная логическая функция макроячейки образуется программированием линейки элементов «И» на первом уровне при фиксированных соединениях с 8-входовым элементом «ИЛИ» на втором уровне. Возможно программирование цепей синхронизации, входов/выходов, использование сигналов обратной связи и т.п.

В этом первом семействе Classic PLD пока еще отсутствует TAP-порт и внутрисхемное программирование невозможно. Но уже в следующих семействах – **MAX 3000** (5000, 7000, 9000 и др.) компании **Altera**, так называемых «настоящих» сложных программируемых устройств (**Complex PLD**), концепция пограничного сканирования поддерживается и тем самым обеспечивается достижение всех связанных с этой концепцией преимуществ – простоты тестирования и программирования структуры ПЛИС.

Общее представление об особенностях CPLD этих семейств дает рис. 1-9, на котором представлена архитектура самой простой микросхемы MAX 3000A [1-11, 1-12].

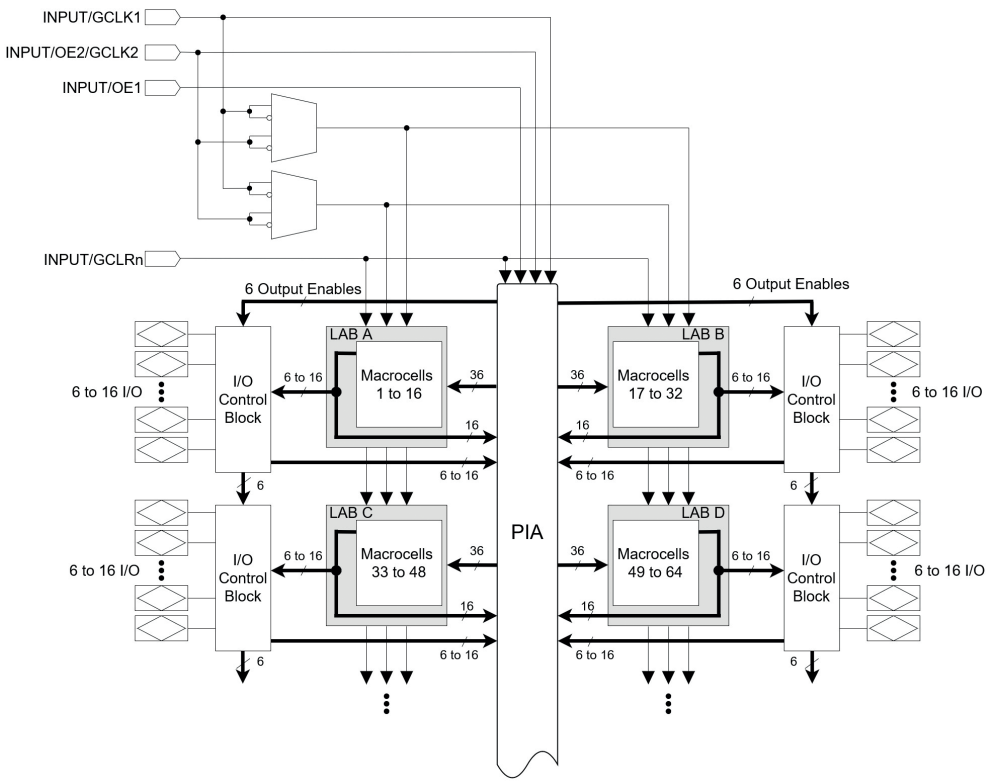


Рис. 1-9. Архитектура CPLD MAX 3000A

Основным крупным компонентом MAX 3000 является блок логических матриц – **Logical Array Blocks** (**LAB**), каждый из которых содержит по 16 макроячеек (**Macrocell**), подобных рассмотренным выше. Макроячейки и логические блоки соединяются между собой, с внешними программируемыми контактами ввода-вывода (**I/O**), а также с внешними общими линиями управления и синхронизации (**Input/GClkX**, **Input/OEX**, **Input/GClrn**) с помощью глобальной программируемой матрицы межсоединений – **Programmable Interconnect Array** (**PIA**) (рис. 1-10).

Эта матрица представляет собой набор переключателей, управляемых ячейкой памяти с электрическим стиранием (EEPROM), которые позволяют подсо-

единить любой из 36 входов логического блока (LAB) к источнику любого из сигналов (внешнего, внутреннего обратной связи и т.д.), подключенного к вертикальным линиям матрицы.

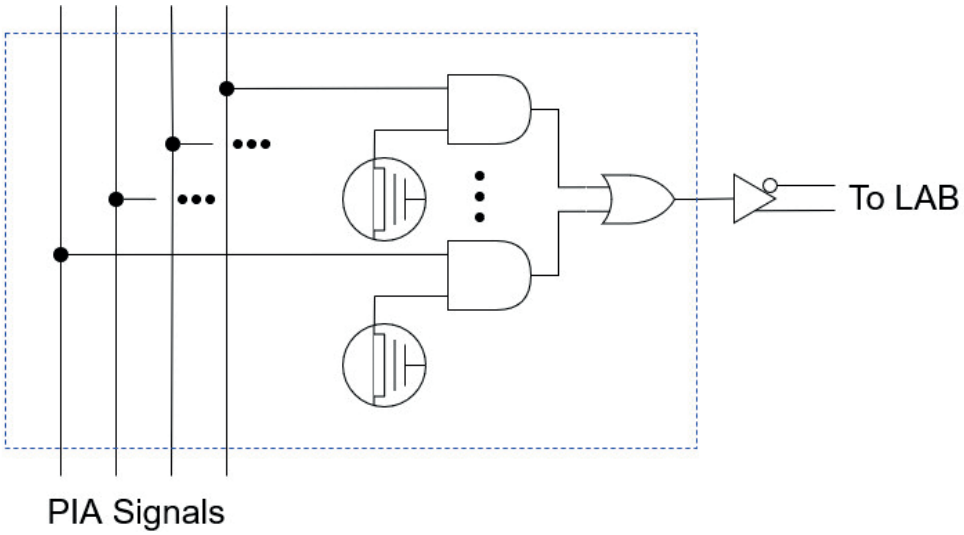


Рис. 1-10. PIA – программируемая матрица межсоединений CPLD MAX 3000

Для увеличения гибкости и сложности реализуемых в рассматриваемых CPLD схем предусмотрены возможности расширения комбинаторной логики макроячеек. В каждой макроячейке некоторый промежуточный терм логической функции через расширитель общих ресурсов (**Shareable Expander**) может быть подключен к линии матрицы межсоединений и использован в других логических блоках. Соответственно, в логическую функцию «ИЛИ» в каждой макроячейке с помощью схемы расширения может быть добавлен терм, созданный в других макроячейках. Тем самым практически снимаются ограничения на сложность логических схем проектируемых устройств при одновременной минимизации расходуемых ресурсов (логических вентилей).

Не останавливаясь на схемотехнических особенностях макроячейки, коротко рассмотрим, как выполнен блок управления внешними выводами MAX 3000A (**I/O Control Block**) (рис. 1-11).

Каждый контакт ввода-вывода микросхемы (**I/O pin**) конфигурируется как входной, выходной или двунаправленный и может устанавливаться в высокоимпедансное состояние. Индивидуальное управление логикой контакта осуществляется с помощью мультиплексора выбора режима (**OE Select Multiplexer**) шестью глобальными сигналами разрешения (**Global Output Signals**), поступающими из матрицы межсоединений (**PIA**), и константными уровнями «0» и «1». Дополнительно функциональность контакта расширяется сигналом перевода выхода в режим с открытым стоком (**Open-Drain Output**) и сигналом управления скоростью изменения выходного напряжения (**Slew-Rate Control**).

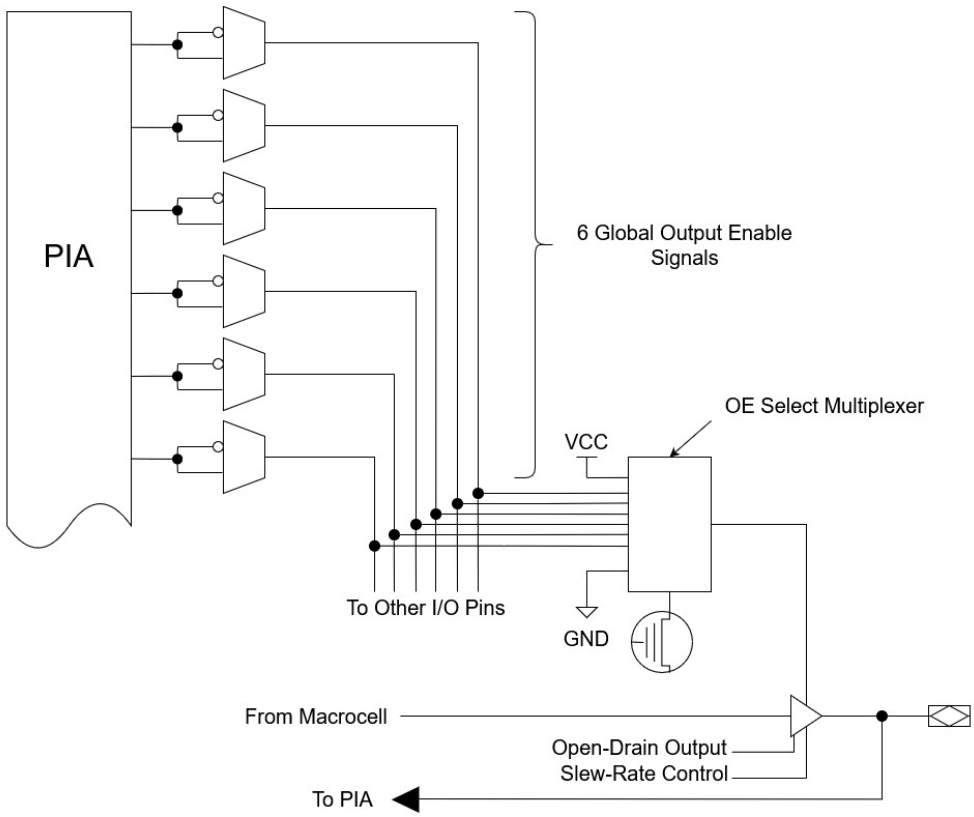


Рис. 1-11. Блок управления внешними выводами CPLD MAX 3000

Дальнейшее развитие архитектуры программируемых логических устройств класса Complex PLD происходит в направлении усложнения логики макроячеек, расширения возможностей конфигурирования внутренних связей, повышения быстродействия и снижения потребляемой мощности, реализации дополнительных пользовательских функций и т.п. Так, например, компания Altera выпускает CPLD (MAX II, MAX V) с тактовой частотой свыше 300 МГц, количеством эквивалентных макроячеек до 2210, количеством каналов ввода-вывода до 272, возможностью выбора уровней и типа сигналов TTL-логики, блоком конфигурационной памяти Flash-типа, блоком пользовательской энергонезависимой (Flash) памяти емкостью 8 Кбит. В этих CPLD реализованы функции управления потребляемой мощностью и быстродействием, полная поддержка технологии многократного внутрисхемного программирования и тестирования по принципу пограничного сканирования и многое другое.

CPLD с аналогичными характеристиками выпускают и другие компании. Подробно ознакомиться с возможностями и характеристиками современных CPLD-устройств можно по посвященным этой теме книгам и технической документации различных фирм. Здесь же приведем обобщенные характеристики ПЛИС этого класса.

Основные свойства и отличительные признаки CPLD:

- в одной микросхеме находится не менее двух логических блоков;
- в одном логическом блоке находится не менее двух макроячеек;
- обычно все макроячейки одинаковые;
- каждая макроячейка имеет свой собственный триггер;
- логическая функция реализуется в макроячейке;
- связи между логическими блоками конфигурируются с помощью глобальной матрицы межсоединений.

Достоинства CPLD – регулярность и универсальность топологии кристалла, благодаря чему обеспечиваются высокая степень использования ресурсов и высокое быстродействие. Это также позволяет легко рассчитать время задержки распространения сигналов.

Основной недостаток CPLD является продолжением их достоинств – для проектирования сложных цифровых автоматов необходима сложная глобальная матрица межсоединений, которая утилизирует много ресурсов кристалла. Другими словами, универсальность структуры ПЛИС становится ограничением для количества пользовательских компонентов.

1.5. Оперативно программируемые логические матрицы

Существует еще одна широко распространенная разновидность сложных программируемых логических устройств, получившая название **Field Programmable Gate Array** (**FPGA**). Слово Field в определении этой разновидности устройств не характеризует принципиальное отличие FPGA от ПЛИС других типов. Они практически все программируются на рабочем месте разработчика аппаратуры, а не на заводе, как, например, непрограммируемые ПЗУ или заказные интегральные схемы высокой степени интеграции, конфигурируемые на завершающих стадиях производства (**ASIC**, **Applications Specific Integrated Circuit**). В литературе многие из рассмотренных разновидностей ПЛИС имеют альтернативные названия, в том числе содержащие слово Field, которые им присваивают разработчики или изготовители часто из конъюнктурных соображений. Более существенное отличие FPGA от других программируемых устройств заключается в том, что их структура хранится в оперативном запоминающем устройстве. Это позволяет перепрограммировать FPGA непосредственно в процессе функционирования системы.

В данной книге будут использоваться те наименования и аббревиатуры, которые применяются чаще всего и впервые введены фирмами-пионерами в данной области, которые создали соответствующий тип ПЛИС. Одной из таких фирм является компания **Xilinx**, выпустившая на рынок в **1984** году новую разновидность сложных программируемых логических устройств под названием **FPGA** [1-6]. Основные особенности FPGA рассмотрим на примере микросхем семейства **Virtex II** компании **Xilinx** [1-13, 1-14], архитектура которых в общем виде представлена на рис. 1-12.

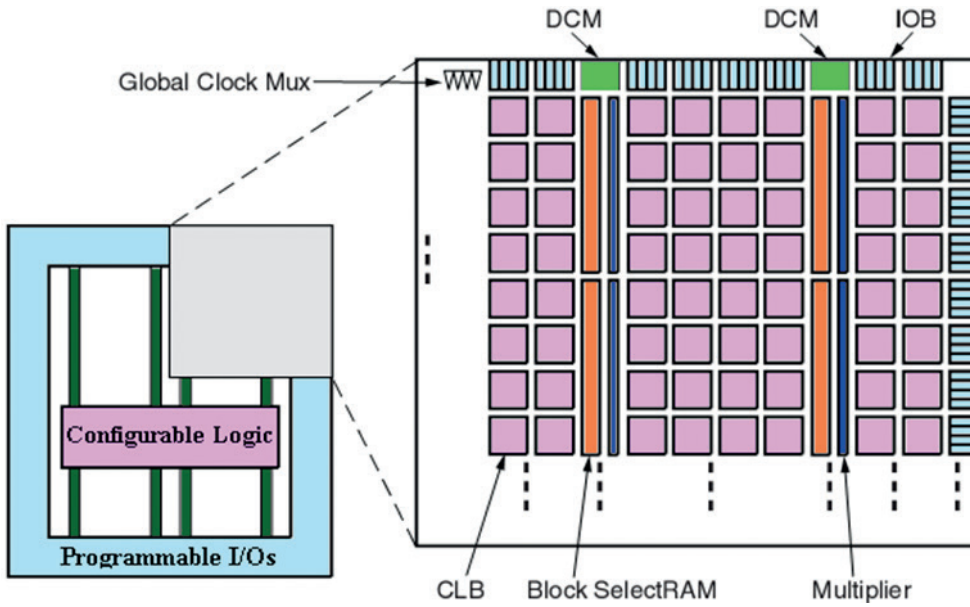


Рис. 1-12. Архитектура FPGA семейства Virtex II

В FPGA применяется многоуровневое конфигурирование различных по сложности логических компонентов, основным из которых является конфигурируемый логический блок (**Configurable Logic Block**, **CLB**). На рис. 1-12 они показаны квадратами розового цвета. Эти логические блоки подключаются к внешним контактам микросхемы через программируемые блоки ввода-вывода (**Input/Output Block, IOB** – голубого цвета).

Кроме упомянутых универсальных логических компонентов, FPGA Virtex II содержит несколько специализированных функциональных компонентов:

- отдельные блоки двухпортовой оперативной памяти общего назначения (**Block Select RAM**) емкостью до 18 Кбит (оранжевого цвета);
- отдельные блоки 18-разрядных умножителей (синего цвета);
- модули цифрового управления синхронизацией (**Digital Clock Manager**) (зеленого цвета);
- мультиплексор глобальных цепей синхронизации (**Global Clock Mux**).

Для конфигурирования Virtex II используется технология активных иерархических буферизируемых межсоединений – **Active Interconnect Technology** (рис. 1-13).

В приведенном фрагменте FPGA вертикальными (зелеными) и горизонтальными (синими) линиями показаны шины матрицы глобальных межсоединений. Конфигурируемые логические блоки, блоки ввода-вывода и другие основные компоненты FPGA могут подключаться к шинам глобальной синхронизации (по 8 в каждом квадранте), а также к вертикальным и горизонтальным глобальным шинам, проходящим вдоль каждого столбца и каждой строки, для создания логических связей второго уровня. Соединения первого

уровня устанавливаются между любыми блоками FPGA и глобальными шинами через матрицы коммутации (**Switch Matrix**).

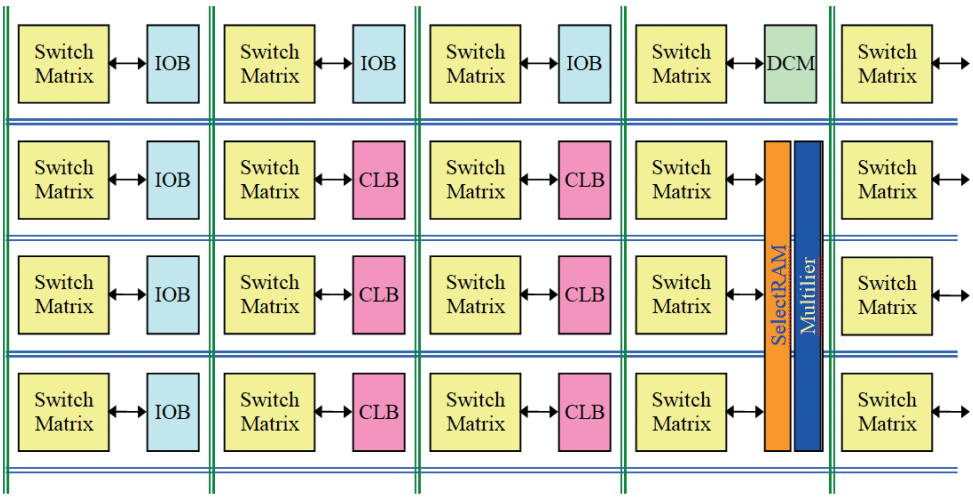


Рис. 1-13. Обобщенная схема соединений компонентов FPGA семейства Virtex II

Существует несколько типов глобальных шин. К шинам, состоящим из 24 «длинных» двунаправленных линий (**long Lines**), могут подключаться любые блоки из соответствующих столбца или строки матрицы блоков. Второй тип шин (**hex Lines**) состоит из 120 линий, позволяющих соединять в шахматном порядке блоки, расположенные в ячейках матрицы блоков с номерами, кратными четырем. Шины из 40 линий (**double Lines**) также предназначены для селективного соединения блоков – каждого второго в строке или столбце. И наконец, еще 16 линий могут быть использованы для непосредственного соединения соседних блоков по всем направлениям – по горизонтали, по вертикали или по любой диагонали.

Возможности реализации логических функций в FPGA определяются структурой конфигурируемого логического блока – CLB (рис. 1-14).

Каждый блок состоит из четырех секций (**Slice**), связанных с коммутирующей матрицей (**Switch Matrix**), в том числе с возможностью организации локальных обратных связей, а также цепями быстрых подключений с соседними блоками (**Fast Connects to neighbors**). Кроме того, логические схемы секций имеют входы **CIN** и выходы **COUT** для образования цепей ускоренного переноса.

В составе каждого CLB есть два драйвера линий с тремя состояниями (**TBUF**), позволяющие создавать двунаправленные магистральные связи между блоками.

Каждая секция (рис. 1-15) содержит два 4-входовых генератора логических функций, схему арифметической логики с цепями переноса для организации многоразрядных вычислителей, набор мультиплексоров и два элемента памяти.

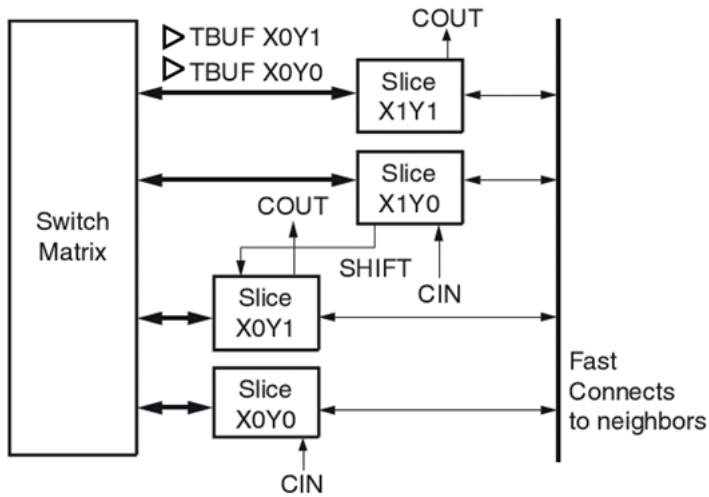


Рис. 1-14. Структура CLB

Генератор логических функций может быть сконфигурирован как 4-входовая таблица преобразования (**Look-Up-Table**, **LUT**), 16-битовая распределенная память (**RAM16**) или как 16-разрядный сдвиговый регистр (**SRL16**). Сигналы на основные выходы секций могут поступать непосредственно с выходов генераторов функций или через элементы памяти (**Register**). Таблицы преобразования (**LUTF** и **LUTG**) реализуют произвольные логические комбинационные функции от 4 переменных, а с помощью мультиплексоров **MUXFx** и **MUXF5** количество переменных генерируемых функций может быть увеличено до 8.

При конфигурировании генератора логических функций в качестве устройств памяти (**RAM16**) на базе нескольких логических блоков можно создать блок распределенной оперативной памяти с различной организацией: от 16×8 бит до 128×1 бит для однопортового доступа и от 16×4 до 64×1 – для двухпортового доступа.

За подробным описанием всех возможностей конфигурирования CLB, включая режимы сдвигового регистра, организации арифметических операций, мультиплексирования и т.п., рекомендуем обратиться к технической документации компании Xilinx.

Особого внимания заслуживает блок ввода-вывода (**Input/Output Block**, **IOB**). На рис. 1-16 показана упрощенная функциональная схема блока ввода-вывода FPGA Virtex II.

Каждый внешний контакт микросхемы соединен с тремя буферными узлами ввода (**Input**), вывода (**Output**) и двунаправленного ввода-вывода (**3-State**) блока IOB. Эти узлы предназначены для согласования внешних устройств, подключаемых к FPGA, по электрическим уровням сигналов и по импедансам источника и приемника сигналов. Каждый из упомянутых узлов содержит также по два элемента памяти (**Reg ICK** и **Reg OCK**), необходимых для буферизации данных.

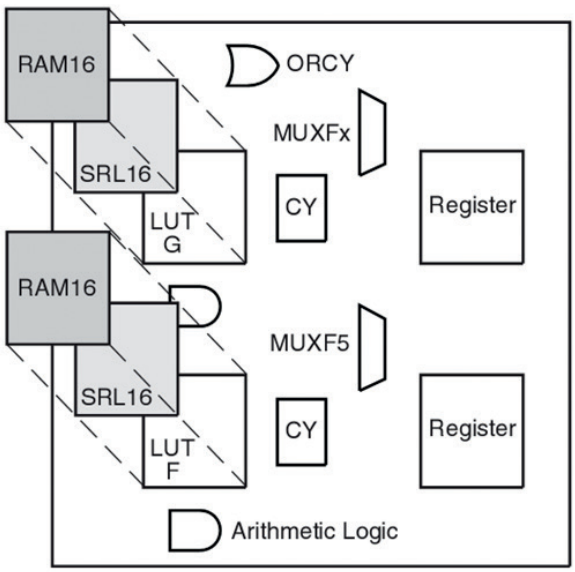


Рис. 1-15. Структура секции Slice

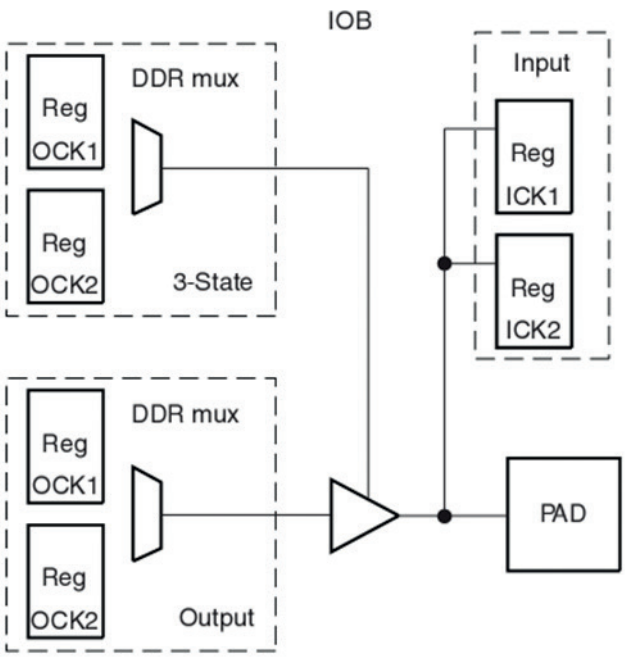


Рис. 1-16. Упрощенная схема блока IOB

Все блоки ввода-вывода компонуются в банки и располагаются по периметру кристалла. Каждый блок может быть сконфигурирован как входной, выходной или двунаправленный контакт FPGA для несимметричной линии связи. Два бло-

ка IOB используются в качестве драйвера дифференциальной линии связи. Поддерживаются свыше 30 стандартов обмена данными по несимметричным или дифференциальным линиям связи, в том числе сигналами с низковольтовыми уровнями ТТЛ- и КМОП-логики (LVTTL, LVCMOS, LVDS, BLVDS и др.), обеспечивается совместимость со стандартными шинами PCI, PCI-X, AGP, CardBus и т.д.

Предусмотрена возможность конфигурирования синхронизации буферных элементов памяти по фронту или уровню синхроимпульсов. Реализованы поддержка механизма обмена данными с удвоенной скоростью (**Double Data Rate – DDR**), настройка параметров выходных каскадов по нагрузочной способности и многое другое.

При проектировании каскадируемых цифровых устройств важно обеспечить точность и стабильность синхронной работы всех каскадов. Для этих целей в FPGA Virtex II служит модуль цифрового управления синхронизацией (**Digital Clock Manager**, **DCM**), который позволяет синтезировать широкую сетку частот синхроимпульсов, синхронизировать их с внутренним или внешним задающим генератором, формировать требуемые сдвиги фаз, автоматически компенсировать задержки прохождения сигналов и т.п.

В современные ПЛИС (как в FPGA, так и в CPLD) включают не только регулярные многофункциональные конфигурируемые устройства вроде CLB или Macrocell, но и специализированные блоки, позволяющие создавать законченные сложные системы без применения дополнительных микросхем общего назначения. Так, в состав FPGA Virtex II входят упоминавшиеся ранее отдельные блоки оперативной памяти общего назначения – **Block Select RAM** емкостью 18 Кбит, которые могут быть использованы как одно- или двухпортовая память с организацией 16K×1, 8K×2, 4K×4, 2K×9, 1K×18 или 512×36 бит.

Второй из специализированных блоков в Virtex II – блок **Multiplier** аппаратного умножения 18-разрядных чисел. Эти умножители применяются совместно с блоками памяти или независимо от них и предназначены для реализации алгоритмов цифровой обработки сигналов.

В табл. 1-1 приведены количественные оценки основных ресурсов FPGA семейства Virtex II, доступные для разработчика специализированной цифровой аппаратуры.

Таким образом, на кристалле FPGA Virtex II может размещаться от 64 до более чем 10 000 конфигурируемых логических блоков, 4 ÷ 168 аппаратных 18-разрядных умножителей, 8 Кбит ÷ 1,5 Мбит распределенной и 72 Кбит ÷ 3 Мбит блочной памяти. При этом количество несимметричных линий ввода-вывода составляет от 88 (для младшего представителя семейства) до 1108. Их попарное конфигурирование образует дифференциальные линии для ускоренной помехозащищенной передачи данных. Основная тактовая частота ПЛИС этого семейства равна 420 МГц.

Компания Xilinx впервые использовала для хранения конфигурации ПЛИС статическую оперативную память, что позволяет не только быстрее изменять структуру связей, но и снимает принципиальные ограничения на количество циклов реконфигурирования. Таким образом, становится возможным создавать системы, функциональное назначение и характеристики которых могут в полном смысле слова изменяться «на лету» в процессе эксплуатации изделия конечным потребителем.

Таблица 1-1. Доступные ресурсы FPGA семейства Virtex II

Модель FPGA Virtex II	Количество логических элементов	Количество конфигурируемых логических блоков CLB (1 CLB = 4 Slices = 128 bits)			Количество блоков умножения	Блочная память (SelectRAM)		Количество модулей цифрового управления синхронизацией (DCM)	Количество контактов ввода-вывода
		Размер матрицы CLB (строк × столбцов)	Количество секций (Slices)	Объем распределенной памяти, кбит		Количество блоков SelectRAM	Объем распределенной памяти, кбит		
XC2V40	40K	8 × 8	256	8	4	4	72	4	88
XC2V80	80K	16 × 8	512	16	8	8	144	4	120
XC2V250	250K	24 × 16	1536	48	24	24	432	8	200
XC2V500	500K	32 × 24	3072	96	32	32	576	8	264
XC2V1000	1M	40 × 32	5120	160	40	40	720	8	432
XC2V1500	1,5M	48 × 40	7680	240	48	48	864	8	528
XC2V2000	2M	56 × 48	10 752	336	56	56	1008	8	624
XC2V3000	3M	64 × 56	14 336	448	96	96	1728	12	720
XC2V4000	4M	80 × 72	23 040	720	120	120	2160	12	912
XC2V6000	6M	96 × 88	33 792	1056	144	144	2592	12	1104
XC2V8000	8M	112 × 104	46 592	1456	168	168	3024	12	1108

Конфигурирование FPGA Virtex II осуществляется через стандартный JTAG-порт или через дополнительные линии специального последовательного порта. При этом несколько микросхем, соединенных в цепочку, могут быть сконфигурированы последовательно из одного источника данных конфигурации, например из ПЗУ с последовательным интерфейсом. Кроме того, в этом семействе FPGA предусмотрен режим быстрой загрузки конфигурационного файла через 8-разрядную шину данных.

Принято считать, что для FPGA характерны следующие отличительные признаки:

- матрица конфигурируемых логических блоков может состоять из узлов разных типов, т.е. конфигурируемая матрица может быть неоднородной;
- FPGA может содержать специализированные цифровые узлы;
- существуют FPGA с «крупнозернистой» или «мелкозернистой» архитектурой, отличающиеся сложностью логических блоков и возможностями связи между ними;
- в «крупнозернистых» FPGA логические блоки содержат, по меньшей мере, один комбинационный логический элемент и один элемент памяти;

- в «мелкозернистых»FPGA логические блоки обычно содержат отдельный комбинационный логический элемент или один элемент памяти;
- каждый логический блок может быть соединен с другим логическим блоком или блоком ввода-вывода;
- логические функции реализуются с помощью глобальных коммутационных матриц и иерархических локальных связей.

К главным достоинствам FPGA относят возможность создания на их основе систем высокой сложности и эффективное использование ресурсов, а также высокое быстродействие.

Вместе с тем FPGA свойственны и недостатки:

- несмотря на то что в FPGA применяется гибкая система организации связей между блоками, реализовать произвольную логическую структуру оказывается достаточно сложно;
- для проектирования систем необходимо использовать высококачественные инструментальные средства;
- существуют проблемы с обеспечением предсказуемости временны́х задержек.

1.6. Сравнение архитектур ПЛИС

Завершая рассмотрение основных типов и архитектур ПЛИС, приведем их укрупненную классификацию в наглядном графическом виде (рис. 1-17), одновременно показав используемые технологии программирования [1-6, 1-15, 1-16].

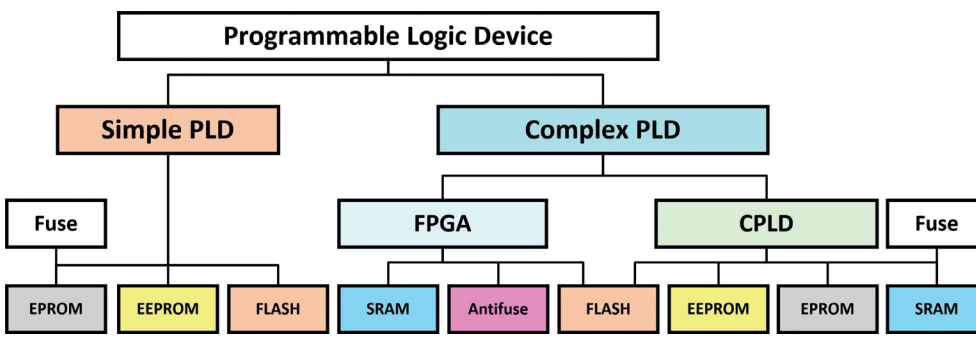


Рис. 1-17. Классификация ПЛИС

Из показанных на рис. 1-17 технологий программирования ранее не упоминался термин **Antifuse**, относящийся к технологии конфигурирования путем создания соединений (перемычек), подобной применяемой в самых первых однократно программируемых ПЗУ. Также используется и «инверсия» этой технологии – выжигание перемычек (**Fuse**). Технология **Antifuse** отличается радиационной стойкостью, повышенной надежностью сохранения конфигурации в экстремальных условиях и применяется в компонентах и системах специального назначения.

Выбор между SPLD и CPLD сделать сравнительно просто. Он определяется логической сложностью проектируемого изделия.

Выбор между CPLD или FPGA не столь очевиден. Даже главные конкуренты и законодатели мод в ПЛИС – компания Intel FPGA (бывшая Altera) и компания Xilinx – производят изделия обоих типов с сопоставимыми характеристиками. При этом в процессе развития ПЛИС и совершенствования технологий многие достижения одной архитектуры заимствуются или реализуются тем или иным способом в другой архитектуре. В табл. 1-2 приведены результаты сравнения свойств CPLD и FPGA, отмечаемые в многочисленных публикациях на эту тему.

Таблица 1-2. Сравнение свойств ПЛИС[1]

Свойство	FPGA	CPLD
Уровень интеграции	++	+
Структура межсоединений	Сегментированная	Непрерывная (однородная)
Временны́е соотношения	Неодинаковые (непредсказуемые)	Фиксированные (предсказуемые)
Производительность	+	++
Использование ресурсов	+	++
Доводка конфигурации «вручную»	Может потребоваться	Не требуется
Возможность реконфигурирования	Есть	Есть
Компиляция	+	++
Стоимость	++	+

Принципиальные свойства архитектур этих двух классов ПЛИС определяют области их предпочтительного применения. Как правило, CPLD эффективнее использовать при реализации сложных логических функций, а FPGA дают лучшие результаты при создании систем обработки данных. Но редко задачи обработки данных могут быть решены без использования сугубо логических алгоритмов, поэтому в конечном счете выбор элементной базы должен проводиться самим разработчиком прикладной системы с учетом не только особенностей решаемой задачи, назначения и области применения системы, но и ряда других факторов, вплоть до собственного опыта создания подобных систем.

[1] Большему количеству знаков «+» соответствует более высокое значение свойства.

Следует отметить, что существует еще одна проблема – как выбрать производителя ПЛИС? Например, дискуссии о достоинствах и недостатках FPGA компаний Altera и Xilinx велись много лет [1-17, 1-18]. Сравнение проводилось с использованием разнообразных методик и критериев, но далеко не всегда весьма убедительные по отдельности, но противоречащие друг другу результаты сравнения зачастую не могут послужить надежным основанием для выбора производителя и семейства FPGA. Как это часто бывает, в подобных случаях надо опираться на свой опыт разработки, самим сравнивать отдельные типы FPGA и принимать решение, исходя из конкретных требований к конечному изделию.

Компания National Instruments (NI), о продукции которой пойдет речь далее, для создания устройств измерения, обработки данных и управления, названных **устройствами реконфигурируемого ввода-вывода** (**RIO**), выбрала FPGA производства **Xilinx**. В первых модулях и контроллерах NI RIO использовались FPGA семейств **Virtex II** и **Spartan 3**, содержащие до 14,3 и 20,5 тыс. секций (Slices) соответственно (эквивалентно 2 и 3 млн логических вентилей). Каждая секция этих FPGA содержит по две 4-входовые таблицы преобразования (LUT) и два триггера. Этих ресурсов оказалось достаточно, чтобы реализовать новые функциональные возможности, радикально повысить детерминизм и скорость обработки данных систем реального времени. Непрерывному расширению номенклатуры и улучшению характеристик изделий RIO способствовало появление все более мощных программируемых интегральных схем и средств их программирования.

Ниже приводится краткий обзор некоторых новых компонентов Xilinx.

1.7. Современные программируемые логические схемы Xilinx FPGA 7-й серии

В эту серию входят FPGA семейств **Spartan**, **Artix**, **Kintex** и **Virtex**, выполненные по технологии 28 нм. По сравнению с FPGA предыдущих серий у них увеличена производительность, количество логических секций и других традиционных компонентов ПЛИС, объем памяти, а также снижена потребляемая мощность. Поскольку каждая секция состоит из четырех 6-входовых таблиц преобразования и восьми триггеров, можно реализовать более сложные логические функции, а увеличение разрядности вычислительных блоков позволяет повысить точность вычислений и расширить динамический диапазон обрабатываемых данных [1-19, 1-20].

Не меньший интерес представляют разнообразные функциональные блоки, реализованные в этих FPGA: мультигигабитные трансиверы для последовательного обмена данными, интегрированные порты интерфейса PCIe, секции DSP. Почти во всех моделях FPGA 7-й серии есть блок XADC, в состав которого входят сдвоенный 12-разрядный 17-канальный аналого-цифровой преобразователь (АЦП) с частотой преобразования 1 МГц. Этот блок позволяет контролировать состояние кристалла с помощью интегрированных датчиков (температура, питание), а также предоставляет возможность использовать FPGA для работы с аналоговыми сигналами.

В табл. 1-3 приведены характеристики семейств FPGA7-й серии.

Таблица 1-3. Характеристики FPGA7-й серии

Характеристика семейств FPGA (максимальное значение)	Spartan-7	Artix-7	Kintex-7	Virtex-7
Логических секций	16K	33,65K	74,65K	305,4K
Логических ячеек	102K	215K	478K	1955K
Емкость блочной памяти RAM[1]	4,2 Mb	13 Mb	34 Mb	68 Mb
Секций DSP	160	740	1920	3600
Производительность / быстродействие				
DSP[2] GMAC / s	176	929	2845	5335
MicroBlaze CPU[3] DMIPs	260	303	438	441
Интерфейс памяти Mb / s	800	1066	1866	1866
Трансиверы				
Количество	–	16	32	96
Скорость обмена данными Gb /s	–	6,60	12,50	28,05
Пропускная способность Gb/s	–	211	800	2784
Интерфейс PCIe	–	x4 Gen2	x8 Gen2	x8 Gen3
Количество	–	1	1	4
Блоков XADC	1[2]	1	1	1
Контактов ввода/вывода	400	500	500	1200
Напряжение каналов ввода/вывода	1,2В÷3,3В			

Дальнейшее развитие FPGA Xilinx связано с архитектурами **UltraScale** (20 нм) и **UltraScale+** (16 нм), 3D транзисторными структурами (**Fin-FET** транзисторами) и технологией **3D-on-3D**, реализуемой с использованием стековых соединений между пластинами кремния (**Stacked Silicon Interconnect**). Характеристики этих FPGA, также применяемых в устройствах RIO, можно найти в [1-22].

Системы на кристалле Xilinx

В класс изделий систем на кристалле (**System-on-Chip**, **SoC**), выпускаемых под маркой **Zynq**, входят «простые» SoC, реализуемые на основе процессо-

[1] Без учета распределенной памяти.

[2] Есть не во всех моделях.

ров ARM Cortex-A9 и FPGA Artix, Kintex или Virtex [1-22], многопроцессорные SoC (**Multiprocessor System-on-Chip**, **MPSoC**), которые ориентированы на самые разные области применения [1-23], и многопроцессорные SoC, предназначенные для применения в системах радиосвязи (**RFSoC**) [1-24]. В многопроцессорных SoC используются процессоры ARM Cortex-A53 и Cortex-R5, вокруг которых размещается программируемая логика архитектуры UltraScale+.

Состав и функциональные возможности SoC Xilinx рассмотрим на примере многопроцессорной системы Zynq UltraScale+ MPSoC семейства EV [1-25, 1-26], блок-схема которой приведена на рис. 1-18.

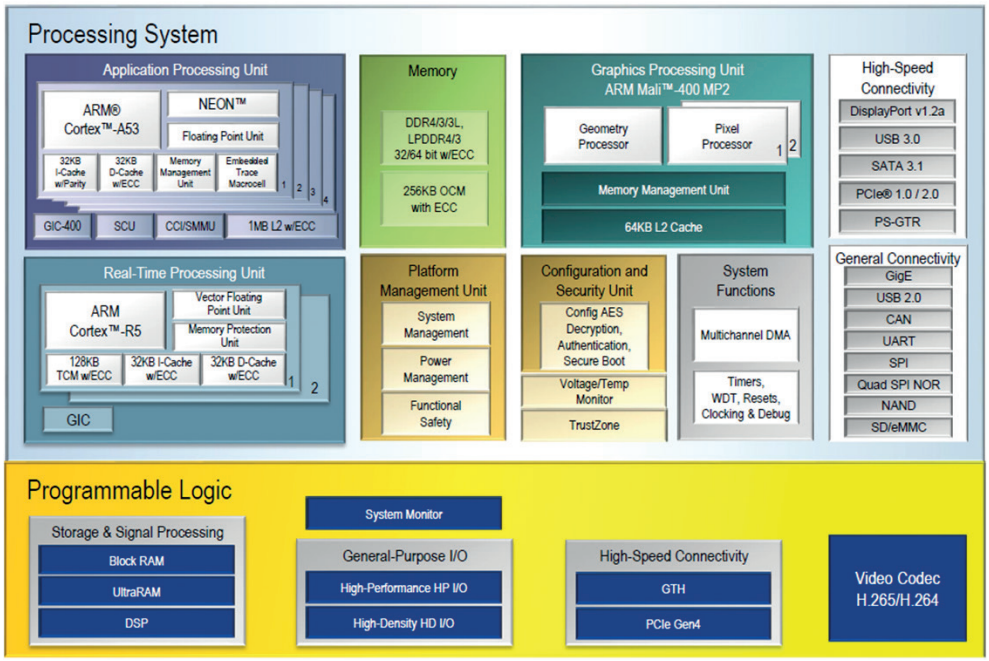


Рис. 1-18. Блок-схема системы на кристалле семейства Zynq UltraScale+ MPSoC

На кристалле SoC размещены две подсистемы: вычислительная подсистема с фиксированной системой команд (**Processing System**), функционирующая под управлением загружаемых программ, и подсистема, алгоритм работы которой реализуется аппаратно в FPGA (**Programmable Logic**).

В состав подсистемы **Processing System** входят:

- 4-ядерный 64-разрядный процессор общего назначения (**Application Processing Unit**) ARM Cortex™-A53 с сопроцессором обработки чисел с плавающей запятой двойной точности (**Floating Point Unit**) и сопроцессором обработки медиаданных (**NEON**);
- 2-ядерный 32-разрядный процессор ARM Cortex™-R5 реального времени (**Real-Time Processing Unit**) архитектуры ARMv7-R с сопроцессором обработки чисел с плавающей запятой двойной точности (**Vector Floating Point Unit**);

- встроенный блок памяти и интерфейс для подключения внешней памяти (**Memory**);
- графический процессор ARM Mali™-400 MP2 (**Graphics Processing Unit**);
- блок управления загрузкой, питанием, тестированием и обработкой системных ошибок (**Platform Management Unit**) на основе специализированного процессора с трехкратным резервированием;
- блок конфигурирования и безопасности (**Configuration and Security Unit**), который выполняет также мониторинг температуры и напряжения питания системы;
- блок **System Functions**, в котором условно объединены контроллер прямого доступа в память (**Multichannel DMA**), таймеры (**Timers, WDT**), схемы сброса (**Resets**), синхронизации (**Clocking**) и отладки (**Debug**);
- порты общего назначения **GigE**, **USB 2.0**, **CAN**, **UART** и др., объединенные в единый блок **General Connectivity**;
- блок **High-Speed Connectivity** мультигигабитных трансиверов **PS-GTR** для высокоскоростного обмена данными через интерфейсы – **USB 3.0**, **PCIe1.0 / 2.0** и др.

В состав подсистемы **Programmable Logic** входят:

- блок **Storage & Signal Processing**, представляющий собой блоки памяти **Block RAM** емкостью 36 Кбит, блоки памяти **UltraRAM** емкостью 288 Кбит (больше возможностей для каскадирования в память большого объема), а также **DSP** блоки для реализации алгоритмов цифровой обработки сигналов;
- **System Monitor** – блок контроля состояния подсистемы **Programmable Logic**;
- блок **General-Purpose I/O** портов ввода/вывода общего назначения: **High Performance HP I/O** (1,8 В) и **High Density HD I/O** (3,3 В);
- блок **High-Speed Connectivity** средств для высокоскоростного обмена данными, содержащий в себе мультигигабитные трансиверы **GTH**, **PCIe Gen4**;
- блок для сжатия видеоизображений **Video Codec H.265/H.264**.

Основные отличия **Zynq UltraScale+ RFSoC** от **Zynq UltraScale+ MPSoC**: в подсистеме **Processing System** отсутствует блок **Graphics Processing Unit**, а в подсистему **Programmable Logic** включены быстродействующие АЦП и ЦАП (блок **RF Signal Chain**). Это позволяет выполнять прямую дискретизацию радиосигналов, что существенно упрощает создание программируемого радио (измерения и генерации) при одновременном снижении потребляемой мощности.

В каждом из трех типов SoC – Zync-7000, Zync UltraScale+ MPSoC и Zync UltraScale+ RFSoC – содержится несколько семейств, отличающихся количеством ядер процессора, наличием графического процессора и видеокодека, количеством, разрядностью и быстродействием АЦП/ЦАП радиоканала и другими свойствами.

В табл. 1-4 приведены основные характеристики каждого типа SoC [1-27].

Таблица 1-4. Характеристики систем на кристалле Zynq

Processing System	Zynq-7000 SoC	Zynq UltraScale+ MPSoC	Zynq UltraScale+ RFSoC
Процессор общего назначения	Single/Dual-core ARM Cortex-A9 1GHz	Dual/Quad-core ARM Cortex-A53 1,5GHz	Quad-core ARM Cortex-A53 1,5GHz
Процессор реального времени	–	Dual-core ARM Cortex-R5 600MHz	Dual-core ARM Cortex-R5 533MHz
Графический процессор[1]	–	GPU ARM Mali™ 400 MP2 667MHz	–
Видеокодек[1]		H.264-H.265	
Интерфейс памяти	DDR3, DDR3L, DDR2, LPDDR2	DDR4, LPDDR4, DDR3, DDR3L, LPDDR3	
Высокоскоростные интерфейсы	USB 2.0, Gigabit Ethernet, SD/SDIO	PCIe® Gen2, USB3.0, SATA 3.1, DisplayPort, Gigabit Ethernet, SD/SDIO	
Безопасность	RSA, AES, SHA, ARM TrustZone®		
Контактов ввода/вывода, макс.	128	214	214
Порты общего назначения	UART; CAN; USB 2.0; I2C; SPI; 32b GPIO		
Programmable Logic	**Zynq-7000 SoC**	**Zynq UltraScale+ MPSoC**	**Zynq UltraScale+ RFSoC**
Логических ячеек	444 K	1143 K	930 K
Емкость памяти	26,5 Mb	70,6 Mb	60,5 Mb
Секций DSP	2020	3528	4272
Трансиверов 12,5G/16G/33G	16/–/–	–/44/28	–/–/16
Контактов ввода/вывода, макс.	250	668	456
Монитор системы	2 XADC (12 бит, 1 MSPS, 17 каналов)	2 (10 бит, 1 MSPS, 17 каналов)	1 (10 бит, 1 MSPS, 17 каналов)
RF блок	**Zynq-7000 SoC**	**Zynq UltraScale+ MPSoC**	**Zynq UltraScale+ RFSoC**
RF АЦП (разрядов/скорость)	–	–	16 (14 бит/6,55 GSPS)
RF ЦАП (разрядов/скорость)	–	–	16 (14 бит/6,55 GSPS)

[1] Не во всех моделях.

При этом подсистемы **Processing System** (**PS**) и **Programmable Logic** (**PL**) не просто дополняют друг друга ресурсами. Между ними создан набор высокоскоростных интерфейсных шин для обмена 32-, 64- и 128-разрядными данными, событиями, запросами прерывания, средствами для доступа к общей памяти и т.п. Это позволяет реализовать на одном кристалле сложные алгоритмы обработки информации и управления динамическими объектами, рационально распределяя выполнение задач между вычислительными ресурсами, работающими под управлением операционных систем общего назначения и операционных систем реального времени (**Soft Processors**), а также специализированными аппаратными средствами цифровой обработки сигналов, создаваемыми пользователем и обладающими экстремальными детерминизмом и производительностью (**Hard Processors**).

Причем процессоры, контроллеры, кодеры и другие разнообразные средства обработки данных, реализованные аппаратно в подсистеме Programmable Logic, функционируют параллельно, обеспечивая возможность одновременно обрабатывать множество данных по одному или множеству алгоритмов (**MIMD – Multiple Instructions Multiple Data**). Например, сотни блоков DSP могут использоваться для параллельного спектрального анализа, в то время как процессор общего назначения может использоваться для управления вводом-выводом данных, анализа результатов обработки, принятия решений и динамического реконфигурирования всей или части подсистемы **Programmable Logic**. Системы на кристалле такого класса открывают уникальные возможности создания современных систем измерений и управления.

Примечание: хотя в состав SoC входят аналого-цифровые и цифроаналоговые преобразователи, коммутаторы и усилители аналоговых сигналов, их часто относят к ПЛИС, т.к. в таких SoC может изменяться только конфигурация подсистемы Programmable Logic.

1.8. Средства проектирования цифровых устройств на ПЛИС

Разработка цифровых устройств на современной элементной базе невозможна без систем автоматизированного проектирования (САПР). Действительно, для того чтобы создать специализированный процессор, аппаратно реализующий какой-либо алгоритм обработки данных, или дискретное устройство управления, необходимы тысячи различных логических элементов и элементов памяти, которые должны быть соединены многочисленными проводниками. Объект разработки характеризуется очень большим количеством возможных внутренних состояний, переходы между которыми должны осуществляться в соответствии с реализуемым алгоритмом. При этом необходимо учитывать требования к задержкам распространения сигналов в логических схемах, ограничения на допустимые коэффициенты разветвления цепей, потребляемую мощность и ряд иных факторов. Поэтому инженеру приходится использовать разнообразные инструментальные средства, позволяющие выполнить все требования технического задания за достаточно короткое время.

На рис. 1-19 приведен упрощенный алгоритм классического процесса проектирования на ПЛИС.

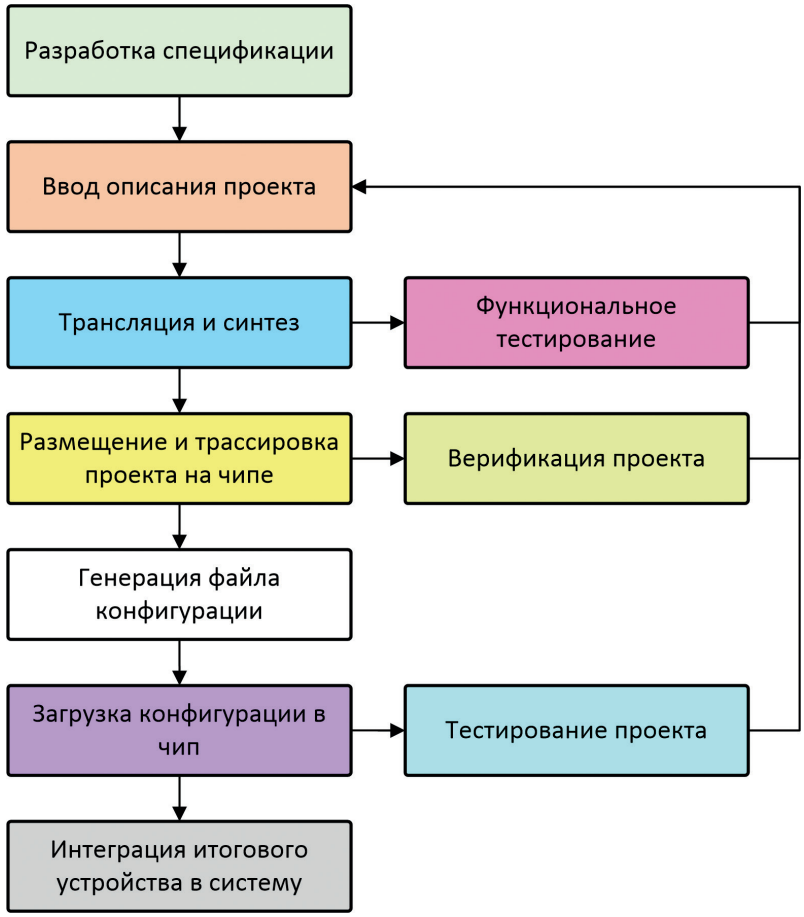


Рис. 1-19. Основные этапы проектирования ПЛИС

Разработка устройств на CPLD и FPGA, за исключением непринципиальных для рассматриваемого уровня деталей процесса проектирования, практически идентична. Наиболее трудоемким и требующим от разработчика значительных затрат времени является этап ввода описания проекта. Как правило, система проектирования позволяет вводить описание проекта в различных форматах – схемном (**графическом**), текстовом на одном из языков описания аппаратуры (**Hardware Description Language** – **HDL**) или в виде диаграммы (графа) состояний-переходов конечного автомата. Все эти форматы реализованы в САПР двух ведущих производителей ПЛИС – в интегрированной системе проектирования **ISE** (**Integrated Synthesis Environment**) компании **Xilinx** [1-28] и в системе **Quartus Prime** компании **Intel FPGA** (**Altera**) [1-29]. Проекты, созданные в одной из САПР, могут быть достаточно легко импортированы в другую САПР [1-30, 1-31]. Особенности системы проектирования **Vivado** компании **Xilinx** будут описаны позже.

Чтобы осознать сложность проектирования на ПЛИС, рассмотрим в качестве примера, как это делается в среде ISE 14.7 Xilinx.

На рис. 1-20 показано окно **ISE Project Navigator** навигатора по проекту, состоящее из нескольких блоков (в качестве примера используется скриншот окна проекта генератора случайных чисел, создание описания которого в графическом формате и некоторые основные этапы реализации проекта рассматриваются дальше).

В блоке **Design** два раздела. В верхнем разделе (1) отображается иерархия проекта **View**, где можно создавать и просматривать компоненты проекта в режимах реализации (**Implementation**) и моделирования (**Simulation**). В нижнем разделе (2) блока **Design** находятся средства запуска и контроля основных этапов процесса проектирования – от создания символов (**Create Schematic Symbol**), требований и ограничений к разрабатываемому объекту (**User Constraints**) до генерации (**Generate Programming File**) и загрузки двоичного файла конфигурации в FPGA и тестирования проекта на целевом устройстве (**Analyze Design Using ChipScope**). С помощью закладок, расположенных внизу блока **Design**, можно выбирать тип компонентов – файлы (**Files**), библиотеки текстовых (**Libraries**) или графических примитивов (**Symbols**).

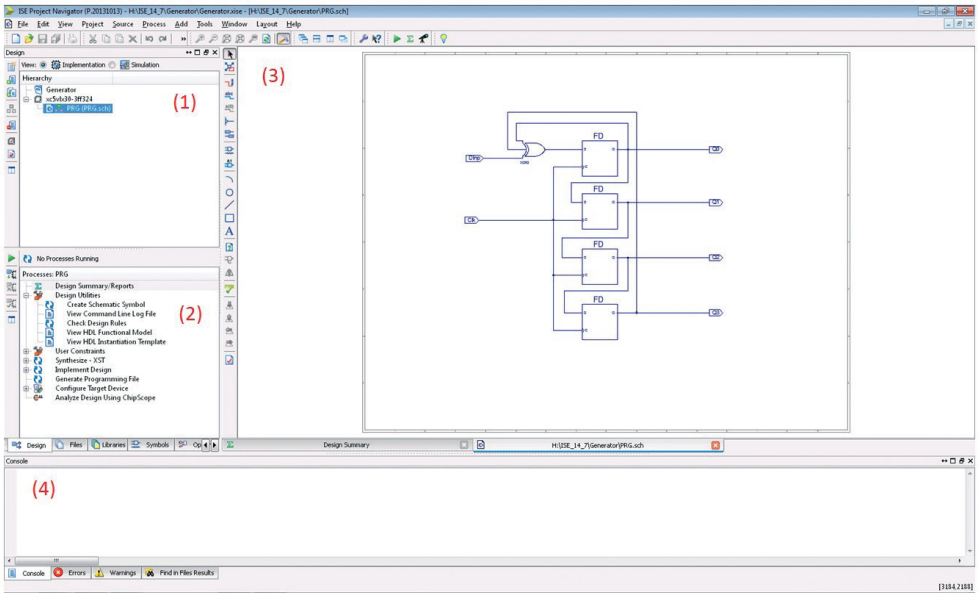


Рис. 1-20. Проект генератора случайных чисел (ввод в графическом формате)

В блоке (3) справа, в зависимости от этапа проектирования, отображаются поле для ввода графической схемы или исходного текста проекта в соответствующем редакторе, тексты или графические изображения генерируемых в процессе проектирования промежуточных файлов (например, исходные коды функциональных моделей в различных форматах **VHDL** или **RTL** (**Registers Transfer Level**, уровень регистровых передач), отчеты о результатах выполнения каждого этапа или всего процесса проектирования (**Design Summary**) и др. Закладки этого блока служат для переключения между документами, ге-

нерируемыми при запуске соответствующего этапа проектирования из контекстного меню пунктов **Design**.

В нижней части окна навигатора ISE в блоке (4) выводятся текущие результаты выполнения операций каждого этапа проектирования (**Console**), сообщения об ошибках (**Errors**), предупреждения (**Warning**) или результаты поиска в файлах отчетов (**Find in Files Results**).

Разработка структуры FPGA начинается с создания нового проекта. На этом этапе выбирается семейство и конкретный тип FPGA, некоторые параметры микросхемы, способ ввода описания проекта (графический или текстовый), предпочитаемый язык – **VHDL** или **Verilog**, среда моделирования и т.п.

Рассмотрим процесс разработки генератора случайных чисел на примере проекта **Generator.xise**. Описание генератора создадим в графическом формате, используя специализированный графический редактор схем и стандартные библиотеки графических примитивов (символов), каждый из которых представляет собой условное графическое обозначение типового логического элемента или устройства – «И», ИЛИ», «D-триггера», дешифратора, сумматора по модулю и т.п.

Приведенная на рис. 1-20 схема **PRG.sch** генератора случайных чисел состоит из четырех D-триггеров (**FD**), объединенных в регистр сдвига с обратной связью через элемент «Исключающее ИЛИ» (**XOR3**). У генератора два входных контакта (данных **DInp** и синхронизации **Clk**) и четыре выходных контакта (**Q0÷Q3**). Внешние входы и выходы подключены к логической схеме через элементы буферизации **IBUF** и **OBUF** соответственно. Схема создана из библиотечных компонентов с помощью инструментов графического редактора, расположенных в вертикальной линейке у левой границы поля редактирования. Инструменты позволяют выбирать компонент (символ) из библиотеки, устанавливать и перемещать символ, соединять выводы компонентов проводниками или шинами, редактировать текст и т.д.

В проекте может быть несколько схем (модулей), связанных между собой соответствующим образом. Один из модулей должен быть назначен главным – модулем верхнего уровня иерархии.

После завершения описания схемы и проверки ее корректности (**Check Design Rules**) запускается компиляция графического описания в код VHDL (**View HDL Functional Model**). Полученный файл может использоваться для предварительного функционального моделирования спроектированного устройства. Ниже на рис. 1-21 приведен фрагмент модели (часть откомпилированного файла **PRG.vhf**).

Функциональное моделирование выполняется в какой-либо стандартной системе моделирования. В ISE 14.7 встроена система моделирования **Xilinx ISim**, которую можно выбрать на этапе конфигурирования проекта. Для моделирования необходимо добавить в проект файл тестовых воздействий **VHDL Test Bench**, создание которого не намного проще, чем разработка самой схемы устройства. Но простота проектируемого устройства позволяет воспользоваться встроенными в ISim простыми средствами подготовки тестовых воздействий.

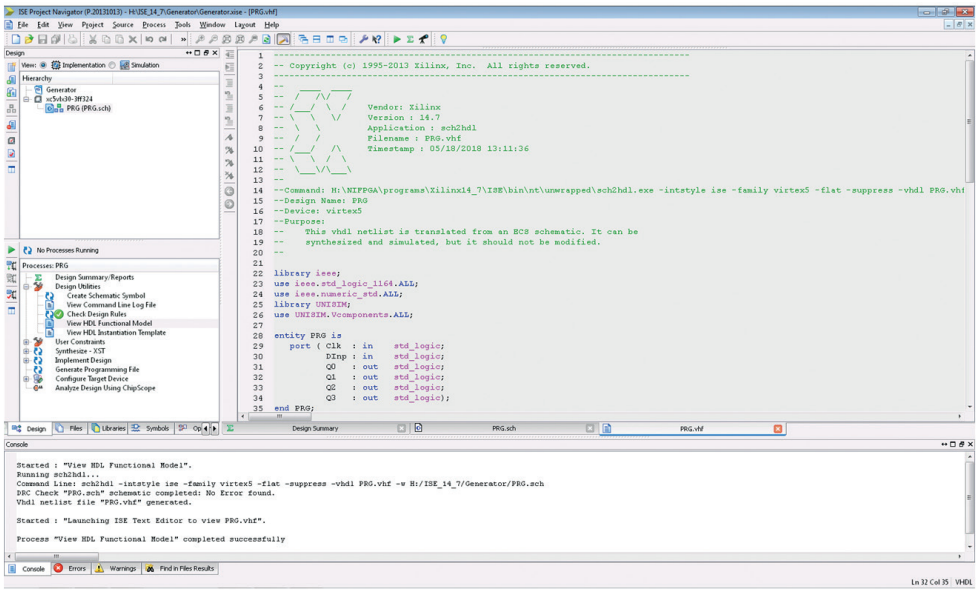


Рис. 1-21. HDL Functional Model

Чтобы начать моделирование, необходимо в навигаторе проекта переключиться из режима **Implementation** в режим **Simulation** и запустить **ISim** из меню **Processes** стандартным способом. Слева в окне **ISim** отображаются имена элементов и процессов проекта (блок **Instances and Process Name**), обозначения цепей (в том числе внешних выводов микросхемы), заданных разработчиком или назначенных автоматически (блок **Simulation Objects for …**), и значений сигналов в этих цепях в исходном состоянии.

Тестовые воздействия для соответствующих входов устройства задаются как константные значения (0 или 1) или как последовательности импульсов заданных периода и скважности, формируемых в течение заданных интервалов времени. В правой части окна ISim отображаются результаты моделирования в формате временных диаграмм (рис. 1-22).

Если поведение модели соответствует ожидаемому, в ISE запускается синтез логической структуры из абстрактных логических элементов и соединений между ними, а затем синтез производится с учетом реальной элементной базы FPGA (**Synthesis – XST**).

Следующий этап компиляции – размещение элементов схемы на ресурсах кристалла и трассировка соединений (**Implement Design: Translate»Map»Place&Route**).

Чтобы проверить соответствие реализованного устройства заданным требованиям к временны́м характеристикам, моделирование выполняется после каждого из этих этапов.

В САПР предусмотрена возможность редактирования вручную топологии схемы размещения элементов на чипе (**PlanAhead**), регионов синхронизации, назначения внешних выводов и др.

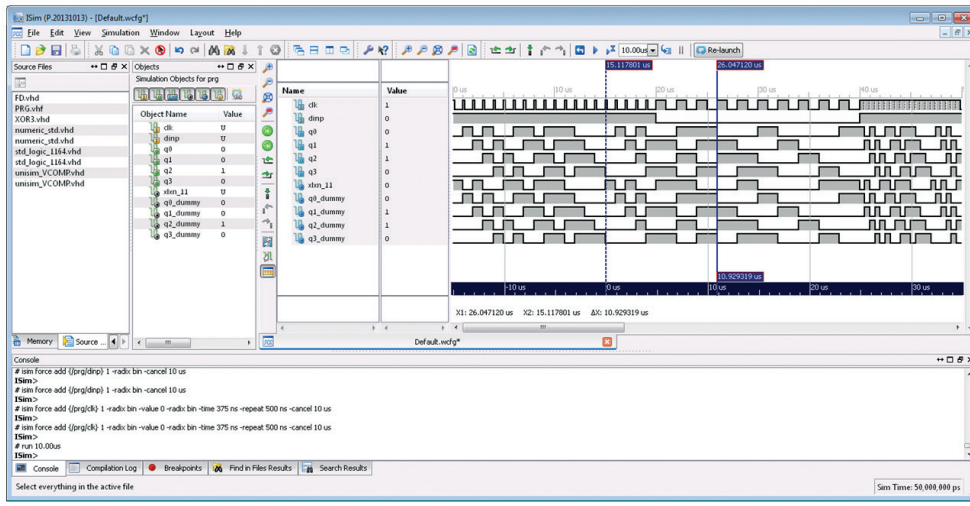


Рис. 1-22. Моделирование в ISim разработанного генератора

Завершается процесс проектирования созданием двоичного файла конфигурации FPGA (**Generate Programming File**), загрузкой этого файла в целевую микросхему отладочного модуля (**Configure Target Device**) и тестированием устройства (**Analyze Design Using ChipScope**) с использованием средств пограничного сканирования и встроенного логического анализатора (при условии что модуль анализатора включен в проект). В результате проектирования для каждого этапа и по завершении всех процессов реализации генерируются отчеты.

На рис. 1-23 справа показано дерево сообщений и отчетов. В основном окне раскрыт итоговый отчет (**Design Overview**), содержащий общие сведения о проекте (**Summary**), сведения об использованных ресурсах FPGA (**Device Utilitization Summary**), а также виден фрагмент отчета об итоговой производительности проекта (**Performance Summary**).

Из дерева отчетов также доступны подробные сведения о размещении элементов на кристалле, о трассировке, о временны́х характеристиках проекта, энергопотреблении и др.

В результате реализации рассматриваемого проекта системой ISE 14.7 создано более 100 файлов, включая файлы отчетов. Следует отметить, что, несмотря на наличие высокого уровня автоматизации, – в процессе проектирования используются готовые шаблоны, специальные вспомогательные средства (мастера), библиотеки простых компонентов и IP-ядер высокой сложности, созданные многочисленным сообществом разработчиков, конфигурирование прикладной структуры FPGA достаточно трудоемко и требует не только знания специализированных языков программирования, но и опыта проектирования цифровых устройств.

Разработка логической схемы выполнялась нами в базовой версии системы проектирования **ISE Foundation**, которая является частью пакета **ISE Design Suite Logic Edition**. В этот пакет входят также **Plan Ahead Design and Analysis**, **ChipScope Pro Logic Analyzer**, **ChipScope Pro Serial I/O Toolkit**, **IP CORE Generator** и другие инструменты, предназначенные для повышения произ-

водительности труда разработчиков, качества моделирования и тестирования создаваемой конфигурации FPGA.

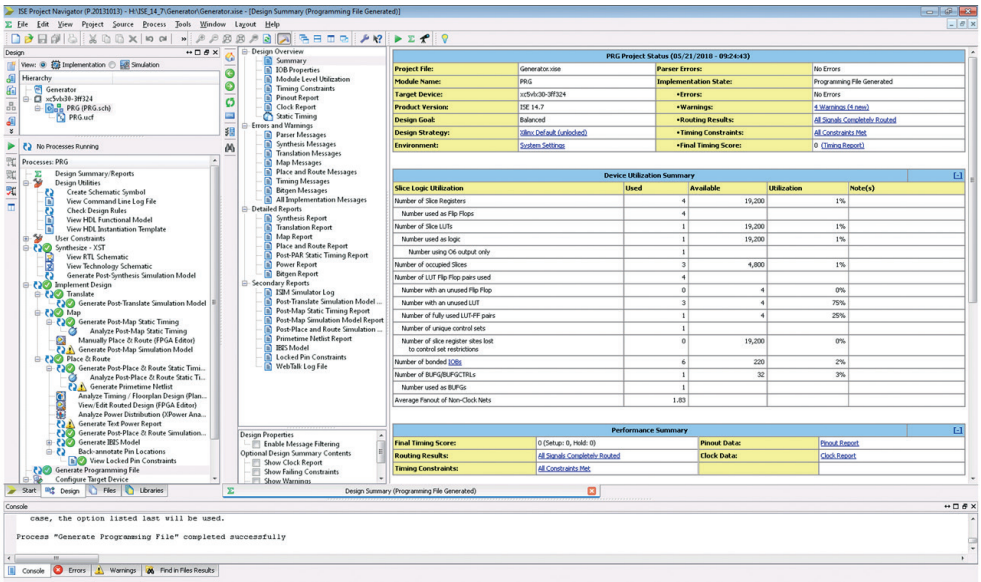


Рис. 1-23. Сводный отчет о результатах проектирования

Для проектирования встраиваемых систем, в составе которых используются ядра микроконтроллеров/микропроцессоров **Microblaze** или **Power PC** (софт-процессоры), Xilinx предлагает пакет **ISE Design Suite Embedded Edition**. В этот пакет, дополнительно к инструментам Logic Edition, входят комплект **Xilinx Platform Studio (XPS)** интеграции IP-ядер софт-процессоров с их периферией, комплект для разработки прикладного программного обеспечения **Software Development Kit**, средства генерации моделей, отладки проекта и т.п.

Пакет **ISE Design Suite DSP Edition** представляет собой пакет Logic Edition, расширенный средствами реализации алгоритмов цифровой обработки сигналов **System Generator for DSP**. Возможности всех пакетов проектирования логики, встраиваемых систем и систем обработки сигналов объединены в полнофункциональном пакете **ISE Design Suite System Edition**. Кроме того, для начинающих пользователей доступна версия **ISE WebPACK Edition**, позволяющая создавать системы на основе FPGA малой и средней сложности.

На овладение перечисленным выше инструментарием требуется много времени. Хотя продолжительность курсов обучения проектированию FPGA, предлагаемых Xilinx и авторизованными партнерами компании, составляет всего от одного до трех дней, но одного курса, как правило, недостаточно даже для инженера, знакомого с HDL. Реализация сколько-нибудь сложных проектов при нетривиальных требованиях к временны́м характеристикам невозможна без практического опыта. Для его приобретения необходимо время, существенно большее, чем для ознакомления с задачей и инструментальными средствами для ее решения.

Трудоемкость проектирования возрастает вместе с увеличением сложности FPGA и распространением систем на кристалле, которые становятся обычным компонентом многих создаваемых систем. В каждом таком компоненте миллионы логических ячеек, сотни (и даже тысячи) блоков цифровой обработки сигналов, несколько софт-процессоров стандартной архитектуры, многоканальные аналого-цифровые преобразователи с блоками преобразования аналоговых сигналов, десятки разнообразных интерфейсных портов для связи с периферией и т.д. Все это размещается на площади порядка сотен квадратных миллиметров вместе с тысячами проводников, длина которых на частотах в десятки гигагерц ограничена допустимыми задержками распространения сигналов.

Как правило, для достижения требуемых характеристик разработка программируемой системы выполняется в несколько итераций с использованием ручного размещения крупных блоков, назначения элементов для их реализации, отличных от предлагаемых САПР по умолчанию, после чего производится тестирование и итерация повторяется до достижения требуемого результата.

Таким образом, классический подход к проектированию FPGA и SoC с использованием языков описания логических схем, таких как VHDL, Verilog или средств разработки, основанных на описании регистровых передач RTL, тормозит дальнейшее развитие и распространение программируемых аппаратных и программно-аппаратных однокристальных устройств. В 2011 г. для новых FPGA седьмой серии и SoC Zynq компания Xilinx предложила новую методологию высокоуровневого синтеза – **High-Level Synthesis** (**HLS**). Реализованная сначала как дополнительный тулкит к ставшей привычной интегрированной среде проектирования ISE, в настоящее время методология **HLS** является основой нового пакета инструментальных средств **Vivado**.

Суть методологии **HLS** заключается в следующем [1-32, 1-33]:

- процесс проектирования распараллеливается – отдельно разрабатываются специализированные логические блоки и оболочка, состоящая из унифицированных блоков – периферийных устройств, интерфейсов и т.п.;
- разработка специализированных (нестандартных) блоков осуществляется с использованием популярных классических языков программирования C, C++ или SystemC. Исходный текст, описывающий алгоритм функционирования таких блоков, затем преобразуется в эквивалентный код на одном из языков описания аппаратуры;
- средствами C-подобных языков выполняется моделирование и тестирование нестандартных блоков, после чего создаются IP-ядра;
- в созданную ранее оболочку интегрируются специализированные IP-ядра, стандартные IP-ядра, разработанные Xilinx и сторонними производителями FPGA, а также IP-ядра, созданные альянсом специалистов по применению FPGA.

После размещения и трассировки блоков синтезированного таким образом проекта генерируется двоичный файл, а затем запрограммированная микросхема FPGA (SoC) тестируется в составе оценочной платы или целевой системы. При этом в проект возможна интеграция IP-ядер, созданных другими сред-

ствами, например с помощью MATLAB Simulink, а также подготовка скриптов, с помощью которых автоматизируется процесс выполнения итераций, целью которых является оптимизация проекта.

Наибольший эффект от применения HLS достижим в проектах, требующих интенсивных математических вычислений, легко реализуемых конструкциями С-подобных языков [1-34]. Прежде всего это разработка систем связи, систем машинного зрения, роботов и т.п. Разработка по традиционной методологии (HDL, RTL) требует наибольших усилий при реализации деталей проекта, в то время как высокоуровневый синтез (HLS) позволяет сосредоточить усилия на решении принципиальных вопросов, на более полном тестировании и оптимизации. Распараллеливание проектирования, повторное использование оттестированных стандартных IP-ядер, созданных различными средствами, упрощение моделирования нестандартных блоков и автоматизация выполнения итераций радикально повышают производительность разработчика.

Различия традиционной и высокоуровневой методологий проектирования наглядно иллюстрируют два следующих рисунка [1-35].

На рис. 1-24 представлен маршрут проектирования, реализуемый с помощью языков описания аппаратуры. Составление описания разрабатываемого объекта, подготовка и выполнение предварительного (упрощенного) тестирования выполняются «вручную». Это сложный итерационный процесс, требующий значительных затрат времени.

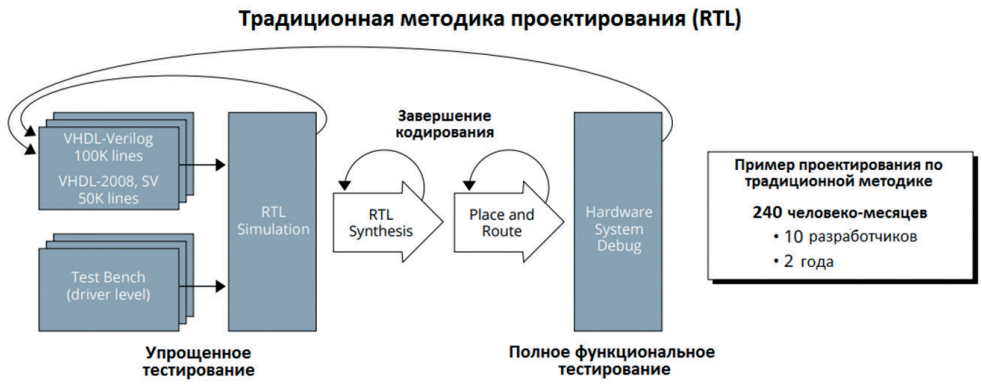


Рис. 1-24. Проектирование по классической методике

Методика HLS (рис. 1-25) не требует знаний специальных языков описания аппаратуры. Алгоритм функционирования специализированных блоков разрабатываемой системы может быть описан на широко распространенном высокоуровневом языке программирования (например, C/C++). Для отладки и тестирования алгоритма также могут использоваться стандартные средства языка С. При этом благодаря существенно меньшему объему исходного кода (тестируются только специализированные блоки) обеспечивается более высокое качество верификации и сокращается время, необходимое для выполнения этих этапов проектирования. После успешного завершения тестирования

вновь разработанные блоки автоматически конвертируются в формат RTL и внедряются средствами HLS и IP-интегратора в скомпонованную из стандартных IP-ядер оболочку.

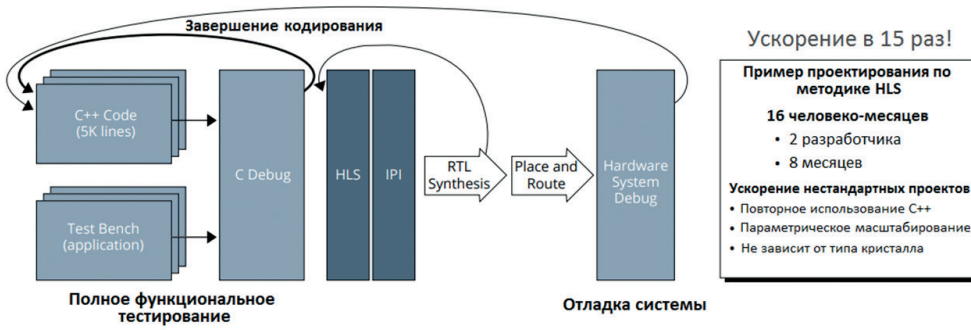


Рис. 1-25. Проектирование по методике HLS

Необходимость в тестировании RTL-модели отпадает, что позволяет еще больше уменьшить затраты времени. Окончательная отладка и верификация проводятся в составе целевой системы.

Приведенные на рисунках оценки трудоемкости реализации некоторого достаточно сложного проекта по рассматриваемым методикам убедительно свидетельствуют в пользу высокоуровневого синтеза – вместо 240 человеко-месяцев по традиционной методике методика HLS требует всего 16 человеко-месяцев. При таких условиях потенциально достигается 15-кратный выигрыш.

В работе [1-32] приводится обобщенная оценка, согласно которой HLS позволяет на порядок сократить время разработки нестандартных блоков и практически в 4 раза ускорить процесс проектирования в целом. При этом коэффициент качества результатов проектирования по методологии HLS составляет от 0,7 до 1,2 по сравнению с традиционной методологией.

Еще одно важное достоинство HLS – использование C-подобных языков создает предпосылки для вовлечения большого количества программистов, владеющих этими языками, в сообщество разработчиков программируемых устройств. Это способствует еще более широкому внедрению FPGA и SoC. При этом следует отметить, что программирование схем на C-подобных языках программирования далеко не эквивалентно программированию на языках HDL. Все-таки необходимо учитывать особенности аппаратной и программной реализаций операторов, параллелизм выполнения алгоритмов, требования к синхронизации и временным задержкам распространения сигналов. Есть также и другие особенности, из-за которых программисту, знакомому только с традиционными языками программирования, совсем не просто перейти к парадигме проектирования схем [1-32].

Результаты проектирования на C отличаются от результатов, получаемых средствами HDL, имеют меньшее быстродействие и, как правило, занимают бóльший объем используемых ресурсов. Но это в большинстве случаев с лихвой окупается выигрышем во времени разработки и выхода на рынок.

Завершая краткий обзор средств проектирования ПЛИС, необходимо отметить, что на несколько лет раньше Vivado инженеры начали использовать систему LabVIEW FPGA компании National Instruments. Являющийся дополнением к среде графического программирования LabVIEW, модуль LabVIEW FPGA дал возможность разрабатывать аппаратно реализуемые в FPGA блоки систем измерений, вычислений, управления. И для этого не нужны не только языки описания аппаратуры типа VHDL, но не обязательно и знание языка C. Инструментарий LabVIEW FPGA обладает еще более высоким уровнем абстракций (по сравнению с VHDL и C), интуитивно понятен и легок в освоении. И что очень важно, язык G LabVIEW является языком параллельного программирования, что естественно вписывается в парадигму параллелизма FPGA. Об этом более подробно будет рассказано в следующих главах.

1.9. Применение ПЛИС

Что же дало появление новой элементной базы и новой технологии проектирования электронных устройств и систем?

Во-первых, инженер получил в свое распоряжение дешевые и доступные средства для разработки и изготовления собственных микросхем. ПЛИС, содержащие миллионы логических вентилей, позволяют заменить десятки типовых цифровых микросхем малой и средней степеней интеграции, существенно уменьшая размеры проектируемых печатных плат и увеличивая их надежность. Одновременно сократились сроки передачи новых изделий в производство за счет использования высокопроизводительных систем автоматизированного проектирования и значительного упрощения (а зачастую и исключения) трудоемкого макетирования разрабатываемых изделий. Дополнительная экономия времени стала возможной при переходе от однократно программируемых ПЛИС к перепрограммируемым, особенно к ПЛИС, перепрограммируемым внутри системы.

Отметим, что если вначале инженер использовал ПЛИС как элемент, являющийся дополнением к универсальным, выпускаемым массовыми тиражами микропроцессорам, то со временем эти два элемента микропроцессорного устройства были объединены в одном кристалле. Так на рынке появились микроконтроллеры, объединенные с ПЛИС, в которой можно реализовать специализированные интерфейсные узлы, каналы ввода-вывода, таймеры и т.д. Кроме процессорного ядра с типовой **RISC-архитектурой** и сравнительно небольшого массива программируемых пользователем логических блоков, эти изделия могут содержать аналоговые узлы, аналого-цифровые и цифро-аналоговые преобразователи, типовые интерфейсные блоки и узлы цифрового ввода-вывода. Все это делает их настоящими однокристальными системами (System-On-Chip, SoC), конфигурация и функциональные возможности которых в значительной степени определяются на стадии проектирования системы. В таких микросхемах структура и функции основных блоков фиксированы на этапе изготовления кристалла, а пользователем (инженером-разработчиком прикладного устройства) конфигурируются лишь интерфейсные или дополнительные блоки.

Слияние традиционных процессорных устройств, функции которых определяются заложенной в них при изготовлении системой команд и исполняемой ими программой (Soft Processors), и конфигурируемых аппаратно устройств (Hard Processors) может осуществляться и принципиально иным способом. С использованием одной и той же технологии программирования структуры ПЛИС можно реализовать и классические процессоры, и специализированные логические устройства. Для многих семейств ПЛИС разработаны и могут быть использованы готовые модули в виде мегафункций (Mega Core или IP Core), воспроизводящие все архитектурные особенности и технические характеристики процессорных ядер, таких как ARM, Power PC, X86 и т.п. На одном кристалле может быть размещено несколько подобных вычислителей, что позволяет радикально увеличить производительность всей системы.

В дополнение к типовым процессорным устройствам разработчик может сконфигурировать в том же кристалле разнообразные специализированные процессорные и коммуникационные блоки, используя мегафункции цифровой обработки сигналов (DSP). С помощью DSP реализуют, например, цифровые фильтры, модуляторы и демодуляторы, а также обычные и высокоскоростные последовательные интерфейсы (гигабитный Ethernet, приемопередатчики RocketIO GTX и многое другое). Таким образом, программируемые логические блоки могут быть размещены внутри кристалла с микроконтроллером, и наоборот, типовые процессорные вычислители в виде IP-ядер могут быть реализованы на ПЛИС.

Другое направление внедрения ПЛИС в инженерную практику обусловлено непрерывным увеличением степени интеграции и объемов выпуска при одновременном снижении стоимости микросхем. Первоначально ПЛИС применяли для макетирования вновь создаваемой аппаратуры. Если эта аппаратура после отладки макета запускалась в серийное производство, то экономически целесообразным было перевести проект, реализованный на ПЛИС, в полузаказные микросхемы **ASIC** (**Application Specific Integrated Circuit**). Многие ведущие производители ПЛИС разработали технологии, позволяющие ускорить и удешевить достаточно трудоемкий и дорогой процесс изготовления фотошаблонов для организации выпуска заказных микросхем на основе кристаллов, совместимых по архитектуре и ряду технологических операций с ПЛИС, полностью программируемыми разработчиком аппаратуры.

Переход на ASIC оправдан только при сравнительно массовом производстве. Поэтому все чаще ПЛИС применяют и в промышленных образцах, выпускаемых средними и даже малыми сериями. Технология изготовления конечных продуктов проста – на печатную плату монтируется «чистая» ПЛИС, в которую на одной из заключительных стадий производства методом внутрисистемного программирования (**In System Programming**) конфигурируется требуемая структура.

Применение ПЛИС также оправдано и на стадии разработки новой продукции. Перед запуском в производство радиоэлектронных, механических, электромеханических, гидравлических и других устройств и систем необходимо изготовить, отработать и испытать экспериментальный, а затем и опытный образец изделия. Этот процесс требует значительных времен-

ных и финансовых затрат. Для ускорения выхода на рынок все чаще используют комплексное программно-аппаратное моделирование (Hardware in the Loop, HIL) различных узлов и систем в целом, воспроизводя с высокой точностью и, что особенно важно, в реальном времени свойства и характеристики проектируемых изделий, а также основные условия их эксплуатации. Математическое (программное) моделирование механических, электрических и иных свойств систем является довольно сложным и не всегда реализуемо в реальном времени. Кроме того, это не позволяет провести поэтапную интеграцию проектируемого программного обеспечения с реальными техническими средствами.

Программно-аппаратное моделирование выполняется с использованием технических средств ввода-вывода электрических сигналов. При этом программно, на математическом уровне, воспроизводятся сложные функциональные зависимости, реализуемые узлами систем или объектов, и изменения условий окружающей среды. А на аппаратном уровне моделируются физические или электрические свойства датчиков и исполнительных механизмов. В таких задачах ПЛИС и системы на кристалле не имеют конкурентов, поскольку дают возможность оперативно изменить конфигурацию аппаратных средств модели, алгоритм функционирования отдельных узлов модели или алгоритм обработки данных в реальном времени. Это в конечном счете способствует ускорению отработки и проверки принципов, закладываемых в проектируемое изделие, улучшению его характеристик и повышению его надежности за счет более полного и тщательного тестирования на всех стадиях разработки.

Новые подходы к решению самых разнообразных задач, обусловленные возможностями реализации сложных алгоритмов обработки данных и высокопроизводительных архитектур, а также совершенствование технологий производства полупроводниковых компонентов стимулируют развитие рынка ПЛИС. Объем рынка FPGA в 2021 году достиг 7 млрд долларов и до 2027 года достигнет 12 млрд долларов, по оценкам экспертов, среднегодовые темпы роста могут составлять от 8 % до 15 % [1-36, 1-37].

В каких областях используются FPGA и SoC, а также устройства и системы, созданные на их основе? Просто перечислим основные из известных приложений:

Телекоммуникации (проводная и беспроводная связь): масштабирование Ethernet от 10 Мб/с до 112 Гб/с; реализация 5G с MIMO и mMIMO; реконфигурирование под различные протоколы и топологии сетей.

Аэрокосмическая промышленность и оборона: авионика; системы управления военными, гражданскими и беспилотными летательными аппаратами; радары и разведка в различных диапазонах спектра; надежная и безопасная военная связь; орбитальные и межпланетные космические аппараты; автономные мобильные планетоходы.

Здравоохранение: ультразвуковые сканеры и томографы высокого разрешения; комплексные мониторы физиологических параметров пациента; оперативная обработка диагностической и клинической информации; роботизированные малоинвазивные хирургические системы с высоким качеством изображения и точным многокоординатным управлением инструментами.

Центры обработки данных и облачные сервисы: высокопроизводительные вычисления; серверы; базы данных с минимальным временем поиска; анализ больших объемов данных (Big Data).

Промышленность: робототехника с точным многокоординатным управлением на основе сигналов от интеллектуальных датчиков; Industrial Ethernet, в том числе с поддержкой чувствительных ко времени сетей (TSN); промышленный интернет вещей (IIoT); 3D-принтеры; масштабируемые промышленные контроллеры; «умная» сеть электроснабжения; интеллектуальные приводы.

Автомобили: системы помощи водителю (ADAS); 3D-видеокамеры; радары; лидары; системы мониторинга салона, состояния водителя и пассажиров; системы безопасности; беспилотные автомобили.

Обработка видео- и аудиоинформации, телевидение: маршрутизация между любыми источниками и приемниками сигналов; обработка и анализ медиаданных в реальном времени; распознавание образов; увеличение качества телевещания (4K; 8K).

Бытовая техника: смартфоны; телевизоры; игры; виртуальная и дополненная реальность; средства безопасности; системы «умный дом».

Измерения и испытания: реконфигурируемые измерительные устройства и системы; тестеры микроэлектронных компонентов и устройств; тестеры систем проводной и беспроводной связи; системы автоматизированного контроля и диагностики промышленной продукции различного назначения; системы автоматизации научных экспериментов, где требуется большое количество каналов измерений, а также быстрое и точное управление экспериментальными установками, проведение экспериментов с помощью автономных исследовательских аппаратов, например в космическом пространстве или на поверхности планет Солнечной системы.

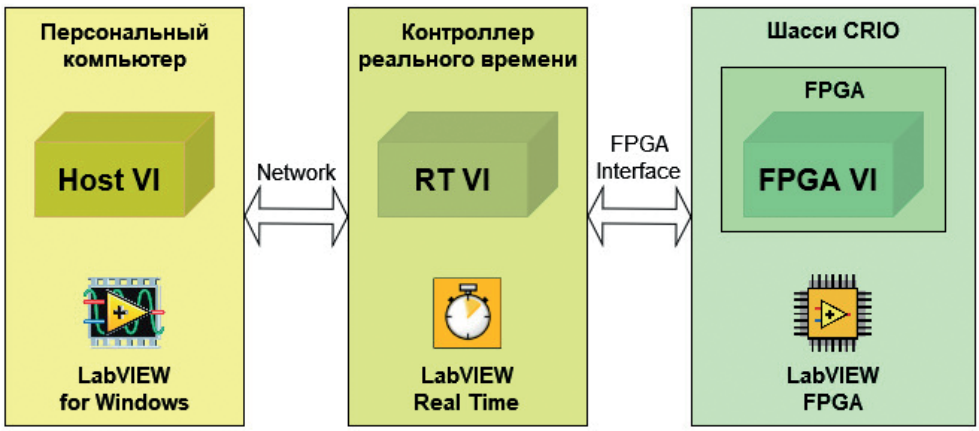
Важно отметить, что во многих из перечисленных выше приложений используются методы искусственного интеллекта. FPGA и SoC хорошо подходят для инференса нейронных сетей. Благодаря своей способности к реконфигурированию они позволяют достаточно просто переходить от одной модели нейронной сети к другой, изменять алгоритмы машинного обучения и формировать логический вывод. Параллелизм аппаратной архитектуры обеспечивает высокую производительность при существенном уменьшении задержки получения результатов. Разработчики FPGA и SoC поставляют не только компоненты, но также создают инструментальные средства для конфигурирования нейронных сетей, подготовки исходных данных, реализации алгоритмов машинного обучения и т.д. (например, OpenVINO) [1-38, 1-39].

1.10. Заключение

Данная книга в первую очередь касается вопросов применения FPGA и SoC для решения задач измерений, испытаний и управления, без решения которых не могут быть разработаны и изготовлены любые изделия – от микросхем различного назначения до интеллектуальных датчиков, от смартфонов и беспилотных автомобилей до космических аппаратов, средств медицинской техники, промышленных роботов и т.д. В том числе подобные задачи стоят и перед разработчиками систем автоматизации научных экспериментов. Поэтому в следующих главах будет рассмотрено устройство, принцип действия и характеристики модулей ввода-вывода, на основе которых создаются средства автоматизации измерений, испытаний и управления. Будет дано краткое описание модулей классической архитектуры, а также модулей, в которые встроены FPGA. Будет показано, как FPGA влияют на характеристики модулей и как при этом изменяется подход к проектированию систем.

Глава 6

Архитектура реконфигурируемых систем измерения и управления



В эту главу входят следующие параграфы:

Что же общего между рассмотренными в предыдущих главах ПЛИС, модулями ввода-вывода и LabVIEW?

Как и в других современных электронных устройствах, ПЛИС используются в модулях ввода-вывода для реализации интерфейса с системной шиной компьютера, создания специализированных блоков управления, запуска и синхронизации процессов аналого-цифрового и цифроаналогового преобразования, ввода и вывода цифровых сигналов, генерации и измерения частотно-временных параметров импульсов и т.д. Применение ПЛИС позволяет заменить большое количество цифровых микросхем малой и средней степени интеграции, сэкономить место на печатной плате и сделать ее дешевле, а также установить дополнительные компоненты для улучшения функциональности модулей ввода-вывода.

Такие ПЛИС, как FPGA, стали вытеснять цифровые сигнальные процессоры, применяемые в модулях ввода-вывода для встроенной обработки данных в реальном времени. Появилась возможность реализации гибких алгоритмов и режимов работы модулей, сокращения сроков их разработки и запуска в производство и, самое главное, увеличения производительности и детерминизма при одновременном уменьшении нагрузки на центральный процессор системы [6-1]. В настоящее время многие компании, лидеры в области измерений и испытаний (такие как Keysight, Teledyne SP Devices, Marvin Test Solutions, VTI Instruments и др.), встраивают FPGA в свои приборы и системы. При этом, как правило, FPGA используются в качестве простой замены программируемых на этапе производства заказных ПЛИС (ASIC). Если предусмотрена возможность загрузки собственных алгоритмов в FPGA, их реализация может быть выполнена или традиционным способом с помощью специальных языков программирования аппаратуры, или с использованием достаточно сложных для освоения средств проектирования более высокого уровня. При этом FPGA разрабатывается как отдельный компонент, слабо связанный с системой, в которую он должен быть интегрирован.

LabVIEW позволяет на основе модулей ввода-вывода, в том числе содержащих FPGA, создавать прикладные системы для решения задач измерения, управления и испытаний, программное обеспечение которых разрабатывается на простом и интуитивно понятном языке графического программирования G с привлечением обширных библиотек функций высокого уровня. Система LabVIEW доступна инженерам и научным сотрудникам, а также специалистам в различных областях промышленности и науки, не обладающим квалификацией программиста. Но функциональность создаваемых систем ограничивается жесткой архитектурой аппаратных средств – классические модули ввода-вывода вместе с их содержимым определены на этапе производства. А ведь в этих модулях находятся компоненты, внутренняя структура которых потенциально может быть изменена даже в условиях эксплуатации промышленного изделия.

Таким образом, сложилась ситуация, когда свобода и легкость проектирования программного обеспечения сложных систем стали тормозиться ограничениями используемых в системах аппаратных средств, внутри которых находятся реконфигурируемые компоненты. К тому же их конфигурирование под

конкретную задачу было доступно только профессионалам, умеющим работать в специфической среде программирования. Ситуация радикально изменилась после того, как корпорация National Instrument создала дополнительный модуль к LabVIEW – модуль **LabVIEW FPGA**. С его помощью стало возможным проектировать структуру микросхемы FPGA, расположенной в модуле ввода-вывода, непосредственно в процессе разработки прикладной системы [6-2, 6-3].

То есть теперь, создавая программное обеспечение системы, можно не только настраивать каналы ввода-ввода, выполнение операций измерения и генерации физических величин, сложной математической обработки, визуализацию и регистрацию данных, но и создавать свои собственные каналы ввода-вывода, специализированные блоки управления, синхронизации и запуска этих каналов, собственные специализированные вычислительные и коммуникационные устройства. И все это на аппаратном уровне. Другими словами, инженер, проектирующий систему, получил возможность определять ее функциональность как путем программирования, основываясь на имеющейся структуре технических средств, так и путем изменения структуры этих средств и выполняемых ими функций. Очень важно, что все это можно реализовать, находясь в одной и той же среде, дополненной специализированными библиотеками функций, и пользуясь одними и теми же инструментами системы графического программирования LabVIEW.

Прежде чем познакомиться с расширенной средой проектирования, в этой главе будут рассмотрены технические средства, которые можно использовать в реконфигурируемых системах, а также особенности архитектуры прикладных программ, разрабатываемых на основе этих средств.

6.1. Компоненты реконфигурируемых систем

Обобщенная структурная схема системы реконфигурируемого ввода-вывода (RIO) представлена на рис. 6-1 [6-4]. Каналы ввода **Analog I/O** системы позволяют измерять аналоговые сигналы электрического напряжения или тока, а для измерения неэлектрических величин и/или согласования с источниками сигналов по импедансу и диапазону во входные каналы встраиваются схемы кондиционирования. Аналогично реализуется согласование с нагрузкой каналов вывода **Analog I/O**, предназначенных для генерации электрического напряжения или тока.

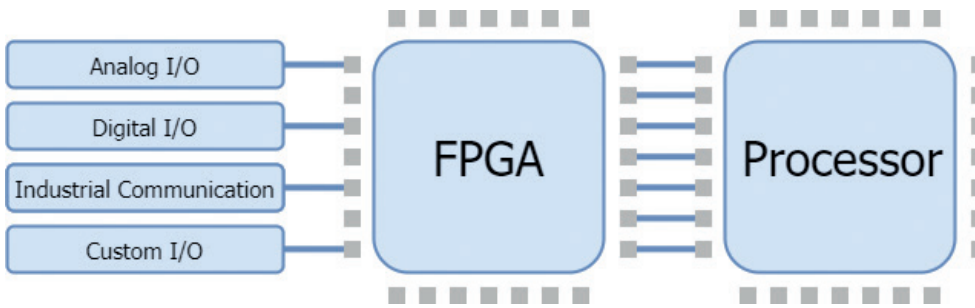


Рис. 6-1. Обобщенная структурная схема системы RIO

Каналы цифрового ввода-вывода **Digital I/O** используются для сбора и/или вывода цифровых данных. Эти каналы поддерживают ввод цифровых сигна-

лов различных стандартов, а также формирование выходных сигналов повышенной мощности.

Для обмена данными с внешними устройствами предназначены каналы **Industrial Communication**, реализуемые обычно в виде портов различных промышленных коммуникационных интерфейсов. Специализированные или пользовательские каналы ввода-вывода **Custom I/O** обеспечивают связь с различными специальными датчиками или периферийными устройствами, такими как, например, видеокамеры, двигатели и т.д.

Все каналы ввода-вывода напрямую подключены к контактам **FPGA**, в которой реализуются функции управления операциями ввода-вывода, формируются сигналы синхронизации и запуска, а также преобразуются данные в соответствии со стандартными и пользовательскими протоколами каналов связи. Как правило, в FPGA выполняется измерение частотно-временных параметров импульсных сигналов или генерация импульсных сигналов с заданными параметрами. Эти сигналы поступают через каналы цифрового ввода или вывода. Предварительная обработка данных в реальном времени и критические ко времени выполнения алгоритмы (в том числе скоростное управление внешними объектами) также реализуются в FPGA. Все эти задачи выполняются параллельно, что обеспечивает высокое быстродействие и максимально достижимый детерминизм, уменьшая при этом нагрузку на главный процессор. **Processor** осуществляет общее управление системой, обрабатывает поступающие из FPGA данные по алгоритмам, не критическим ко времени и/или требующим больших объемов памяти, организует интерфейс оператора, регистрацию результатов измерений и обработки в файлах или в базах данных, обменивается данными с другими системами по сети и т.д.

Использование в системе двух типов вычислительных устройств – аппаратно реализуемых в FPGA процессоров жесткого аппаратного реального времени (**Hard Processor**), работающих параллельно без какой-либо операционной системы, и процессора (возможно, многоядерного), выполняющего программы под управлением многозадачной операционной системы (**Soft Processor**), позволяет оптимизировать сложные процессы обработки данных и управления и достичь высоких характеристик проектируемой системы.

Преимущества технологии реконфигурируемого ввода-вывода по сравнению с классической технологией демонстрируют следующие два рисунка. На рис. 6-2 показан маршрут прохождения сигналов, которыми обмениваются система классической архитектуры и объект тестирования (или управления) [6-5].

Отклик от объекта в такой системе поступает на обычный модуль ввода-вывода, поддерживаемый драйвером DAQmx. Операционная система вызывает драйвер модуля и передает полученные данные в прикладную программу, где по результатам обработки формируется тестовое (управляющее) воздействие, которое, проходя в обратном направлении через драйвер и модуль ввода-вывода, поступает на объект. В этой системе обработка данных производится обычным процессором, функционирующим под управлением многозадачной операционной системы, а тракт передачи данных в контуре сформирован несколькими программным компонентами, которыми в основном определяется длительность цикла «тестовое воздействие – отклик», равная обычно ~25 мкс.

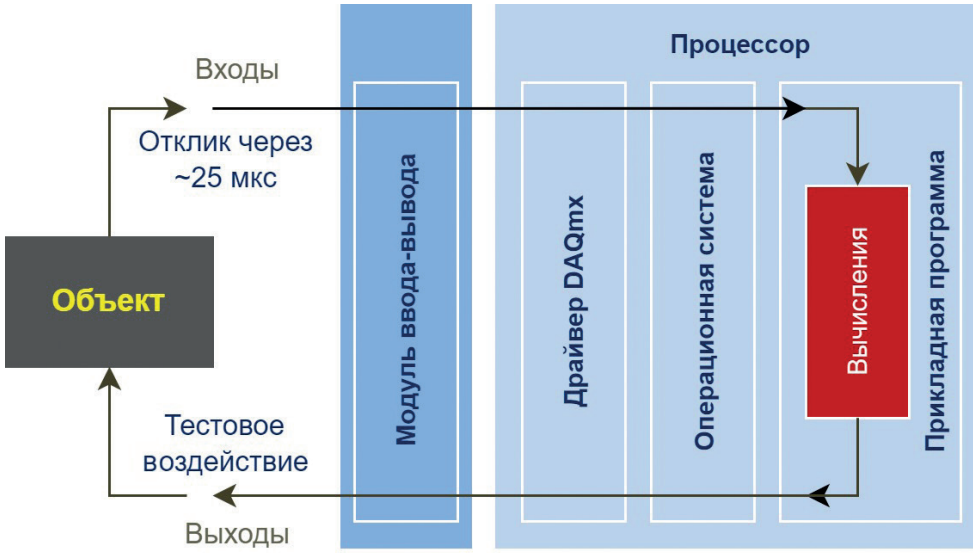


Рис. 6-2. Замкнутый контур в классической системе

Тракт прохождения сигналов в системе, реализованной по технологии реконфигурируемого ввода-вывода, намного короче (рис. 6-3). И самое главное, в него не входят компоненты, связанные с операционной системой.

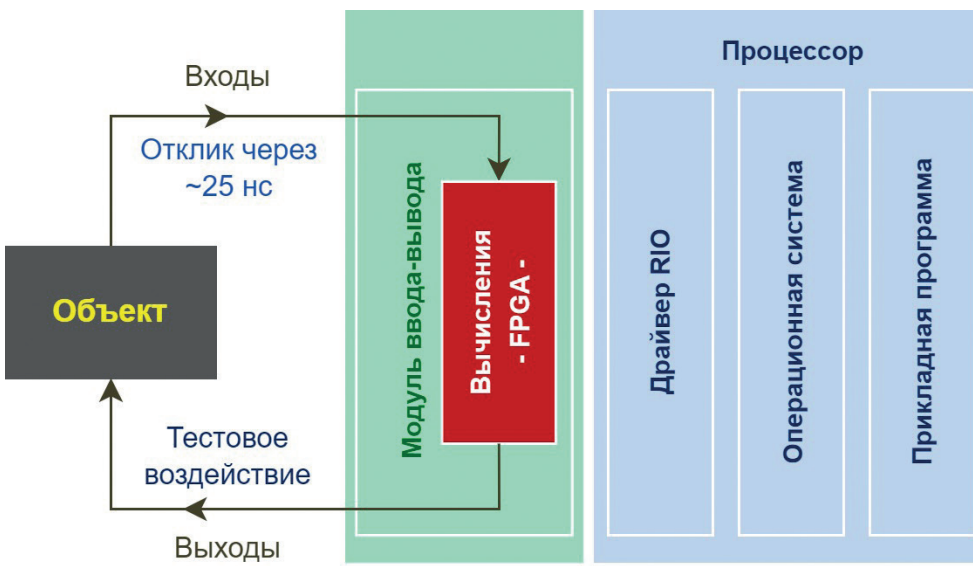


Рис. 6-3. Замкнутый контур в системе реконфигурируемого ввода-вывода

Получаемые модулем ввода-вывода данные о состоянии объекта обрабатываются по алгоритму, реализованному аппаратно в FPGA. Там же формируются передаваемые на объект тестовые (управляющие) воздействия. Длительность

цикла «тестовое воздействие – отклик» в подобных системах может достигать 25 нс, при этом одновременно повышается надежность и обеспечивается максимальный детерминизм.

Сравнение систем реконфигурируемого и классического ввода-вывода при решении задачи тестирования или управления в контуре с обратной связью убедительно свидетельствует о преимуществе первых. За почти 20 лет развития технологии RIO появились разнообразные устройства, которые стали основой новых архитектур систем измерения, испытаний и управления. Среди этих устройств такие, как отдельные функциональные блоки систем RIO, автономные встраиваемые системы, узлы распределенных систем сбора данных и/или управления и др. Устройства и системы выпускаются для разных платформ, отличаются областью применения и условиями эксплуатации, а также техническими и массогабаритными характеристиками и т.п.

Далее приводится краткое описание основных архитектур систем RIO.

6.2. Системы на основе модуля R-серии

Первыми устройствами, изготовленными по технологии RIO, были многофункциональные модули R-серии с реконфигурируемыми каналами ввода-вывода, подобные рассмотренным в главе 2, которые устанавливались в PCI-слот персонального компьютера или в шасси промышленной системы в стандарте PXI (рис. 6-4). Благодаря возможности создания в FPGA пользовательских процессоров обработки данных модули этой серии называют также интеллектуальными модулями ввода-вывода.

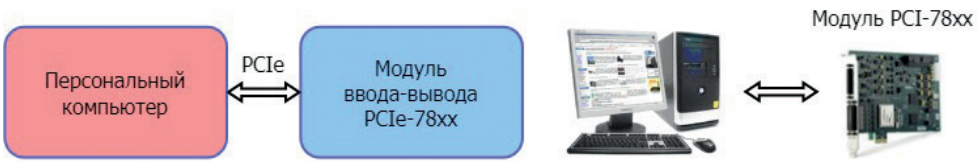


Рис. 6-4. Система на основе многофункционального модуля R-серии

Разные модули R-серии содержат до 16 каналов аналогового ввода и до 24 каналов аналогового вывода – это фиксированные аппаратные ресурсы, режимы работы которых определяются создаваемыми в FPGA блоками управления, запуска и синхронизации. Кроме того, разработчик системы может сконфигурировать в модуле до 96 (до 160 в модулях без аналоговых каналов) пользовательских каналов цифрового ввода-вывода, создавать счетчики, таймеры, генераторы импульсов и измерители частотно-временных параметров импульсных последовательностей, а также коммуникационные устройства. Все это обеспечивается реконфигурируемыми ресурсами FPGA.

В тех случаях, когда каналов аналогового ввода или вывода недостаточно, или требуется подключение датчиков различных физических величин и исполнительных устройств, схема, изображенная на рис. 6-4, модифицируется – к модулю подключается шасси расширения [6-6]. В это шасси устанавливаются модули кондиционирования сигналов C-серии, в которых осуществляются необходимые преобразования сигналов, поступающих от датчиков или выда-

ваемых на исполнительные устройства, а также аналого-цифровое или цифроаналоговое преобразование сигналов. Структурная схема и внешний вид системы на основе многофункционального модуля R-серии с увеличенным количеством каналов и функциями кондиционирования показаны на рис. 6-5.

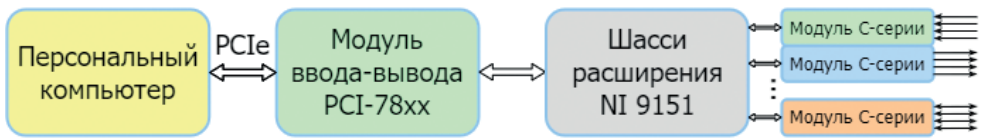


Рис. 6-5а. Структурная схема системы на основе многофункционального модуля R-серии с шасси расширения и модулями C-серии

Рис. 6-5б. Внешний вид компонентов системы на основе многофункционального модуля R-серии с шасси расширения и модулями C-серии

Компоненты прикладного программного обеспечения таких систем и среды, в которой они разрабатываются, изображены на рис. 6.6.

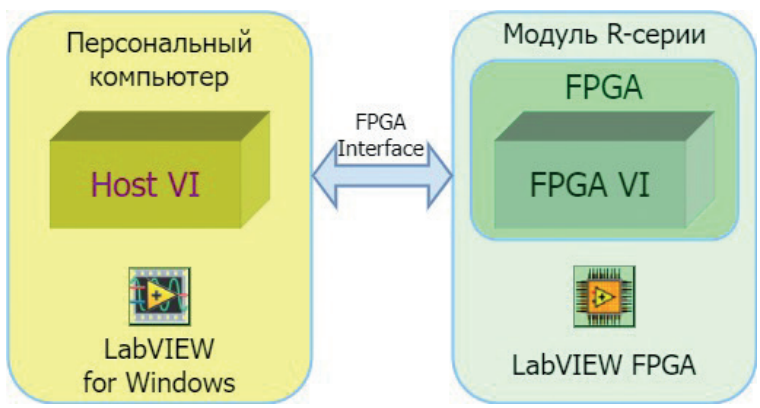


Рис. 6-6. Компоненты среды проектирования и прикладного программного обеспечения систем на основе модуля R-серии

Программное обеспечение систем состоит из двух частей. Первая (**FPGA VI**) представляет собой исполняемую в FPGA программу обслуживания каналов ввода-вывода, выполняющую также и необходимую обработку данных в режиме жесткого аппаратного реального времени. Но это не программа в традиционном смысле слова, а структура FPGA, аппаратно реализующая перечисленные выше функции. При этом она проектируется в среде LabVIEW как обычная программа, с использованием палитры функций модуля LabVIEW FPGA. В некоторых случаях может оказаться достаточно только этой программы, исполняемой в R-модуле.

Для того чтобы выполнять более сложную обработку, визуализировать данные и сохранять их на диске, а также обмениваться данными с другими системами по сети, разрабатывается вторая часть программного обеспечения (**Host VI**), которая работает на процессоре персонального компьютера, обмениваясь данными с первой частью, исполняемой в FPGA. Задачи, решаемые Host VI, не критичны к скорости выполнения и детерминизму. Программа Host VI проектируется в среде LabVIEW с использованием функций субпалитры **FPGA Interface**.

Количество модулей R-серии в системе ограничено количеством свободных слотов в компьютере. В настоящее время модули R-серии выпускаются на основе FPGA семейств Virtex 5 и Kintex 7 в стандартах PCIe, PXIe и USB.

6.3. Высокопроизводительные системы на платформе FlexRIO

Модули FlexRIO также предоставляют возможность создания в FPGA пользовательских каналов цифрового и таймерного ввода-вывода, специализированных блоков синхронизации и запуска и устройств обработки данных. В отличие от модулей R-серии, в системах FlexRIO тип и характеристики измеряемых и генерируемых сигналов определяются адаптерами ввода-вывода [6-7]. К модулю **FPGA for FlexRIO** можно подключить адаптер, изменив тем самым назначение и характеристики разрабатываемой системы (рис. 6-7).

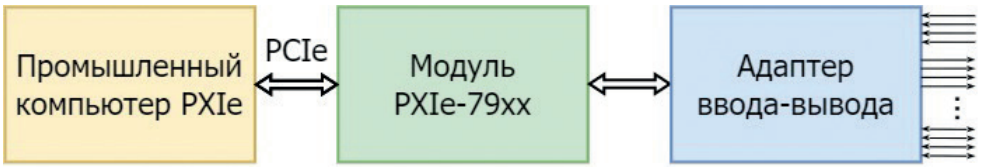


Рис. 6-7а. Структурная схема системы на основе модуля FPGA for FlexRIO с адаптером ввода-вывода

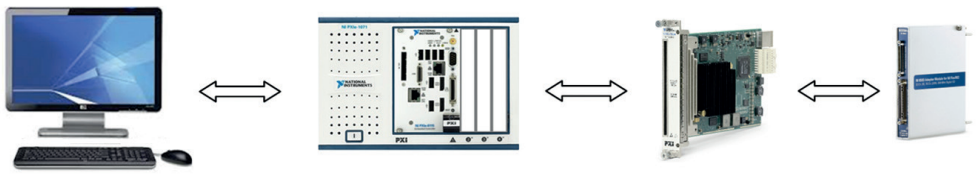


Рис. 6-7б. Внешний вид компонентов системы на основе модуля FPGA for FlexRIO с адаптером ввода-вывода

Можно даже разработать собственный адаптер ввода-вывода, используя комплект FlexRIO Adapter Module Development Kit [2-26, 2-27]. Но это требует привлечения специалистов в области проектирования электронных устройств и программирования на языках HDL.

Загружаемый в FPGA модуля FlexRIO код должен обеспечивать взаимодействие пользовательского FPGA VI с адаптером ввода-вывода определенного

типа и с установленными на печатной плате модуля микросхемами оперативной памяти DRAM. Интерфейс между FPGA VI и адаптером, а также между FPGA VI и памятью DRAM реализуется путем интеграции кода FPGA VI с HDL-кодом **Socketed CLIP** (Component Level Intellectual Property – подключаемых IP компонентного уровня) (рис. 6-8).

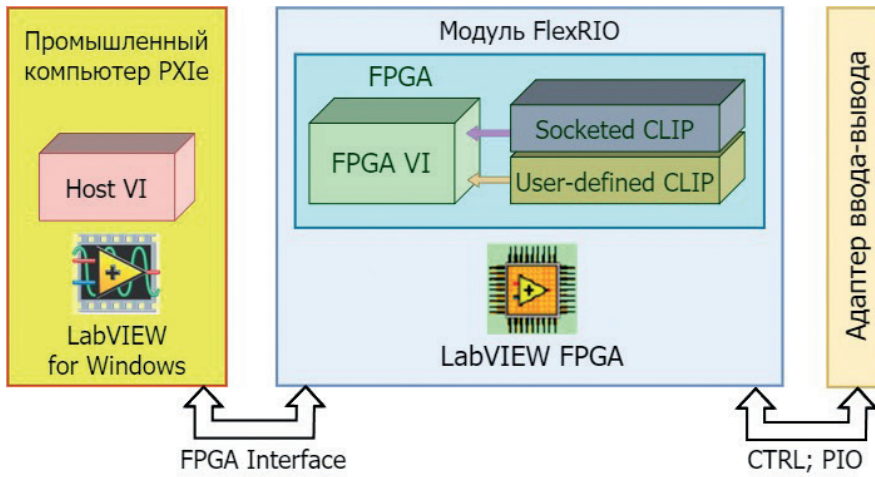


Рис. 6-8. Компоненты среды проектирования и прикладного программного обеспечения систем на основе модуля FPGA for FlexRIO с адаптером ввода-вывода

Кроме стандартных подключаемых CLIP (Socketed CLIP), поставляемых с адаптером ввода-вывода и модулем FlexRIO, в FPGA VI можно внедрять пользовательские CLIP (**User-defined CLIP**), также разрабатываемые на языках HDL. Фактически код Socketed CLIP адаптера – это аппаратный интерфейс с компонентами адаптера, в том числе с контактами его внешнего разъема. Интеграция Socketed CLIP с FPGA VI позволяет использовать компоненты адаптера узлами LabVIEW FPGA. Практически так же разрабатывается программное обеспечение для модулей **FlexRIO с интегрированными адаптерами ввода-вывода** [6-7]. Драйверы этих модулей предоставляют высокоуровневые API, обеспечивающие потоковый обмен данными и поддерживающие технологию точной синхронизации NI-TClk.

Для разработки систем с небольшим количеством каналов ввода-вывода одного типа эффективным может оказаться использование **контроллера FlexRIO**, к которому подключается **адаптер ввода-вывода FlexRIO** (рис. 6-9). Система на основе контроллера FlexRIO представляет собой автономное компактное устройство, ядром которого является система на кристалле **Xilinx Zynq 7020** с 2-ядерным процессором ARM Cortex A9 и FPGA семейства Kintex-7 [6-8]. Контроллер содержит блок энергонезависимой памяти емкостью 512 Мб и оперативную память DRAM емкостью 2 Гб. Интерфейс DDR3 шириной 512 бит при тактовой частоте 166 МГц обеспечивает максимальную пропускную способность канала обмена данными между FPGA и DRAM, равную 10,6 Гб/с. Имеется также слот для карты памяти емкостью 32 Гб, разъемы для сигналов синхронизации и запуска. Периферийные

устройства могут подключаться через порт Ethernet или через два порта USB. Контроллеры FlexRIO можно объединить в сеть, используя два мультигигабитных трансивера Xilinx MGT с пропускной способностью 10 Гб/с. Обмен данными между контроллерами, сервером или другими узлами сети по этим каналам производится в потоковом режиме.

а б

Рис. 6-9. а) Контроллер FlexRIO; б) контроллер FlexRIO с адаптером ввода-вывода

В состав программного обеспечения системы на контроллере FlexRIO, кроме FPGA VI и Host VI, входит программный модуль RT VI, который выполняется на процессоре контроллера под управлением операционной системы реального времени **NI Linux Real-Time**. Как распределяются задачи между этими тремя компонентами программного обеспечения, мы обсудим позже.

Вычислительную мощность системы дополнительно можно увеличить, подключив модуль **сопроцессора FlexRIO**, предназначенный для обработки данных в жестком аппаратном реальном времени [2-28]. В сопроцессорах (рис. 6-10) установлены FPGA семейства **Kintex UltraScale KU0xx** и оперативная память емкостью 4 Гб. Обмен данными с другими модулями системы может осуществляться в потоковом режиме напрямую через четыре мультигигабитных последовательных порта с пропускной способностью 16,375 Гб/с или через интерфейс PCIe Gen3 x8 объединительной панели шасси.

Рис. 6-10. Сопроцессор FlexRIO

Совместное использование сопроцессоров FlexRIO с модулями трансиверов или модулей ввода изображений позволяет решать сложные задачи обработки данных в реальном времени, для которых вычислительной мощности одной FPGA недостаточно.

Завершая краткий обзор архитектуры систем FlexRIO, отметим, что достигнутые в них высокие частоты дискретизации измеряемых и генерируемых сигналов, а также обработка сигналов в реальном времени на аппаратном уровне стали возможными благодаря появлению новых технологий и компонентов, прежде всего АЦП и ЦАП, высокопроизводительных FPGA и систем на кристалле.

6.4. Системы на платформе CompactRIO

В состав аппаратной части систем **CompactRIO** входят четыре основных компонента: контроллер реального времени (**cRIO Real Time Controller**), шасси реконфигурируемого ввода-вывода (**cRIO Chassis**), модули ввода-вывода **C**-серии и персональный компьютер (опционально) (рис. 6-11). Персональный компьютер связан с контроллером реального времени интерфейсом Ethernet со стандартным протоколом TCP/IP или UDP, а контроллер подключается к реконфигурируемому шасси через внутреннюю системную шину Internal PCI Bus. В шасси находится главный компонент системы – микросхема FPGA, в которой разработчик создает все необходимые узлы: блоки управления, синхронизации и запуска модулей cRIO, устройства обработки данных, коммуникационные каналы и т.д. Кроме того, FPGA используется для реализации интерфейса со всеми модулями по схеме типа «звезда», обеспечивая тем самым прямой доступ к каждому модулю.

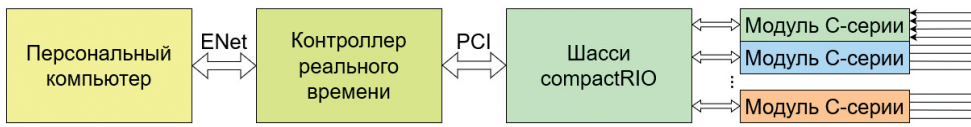


Рис. 6-11а. Система на основе контроллера реального времени и шасси реконфигурируемого ввода-вывода

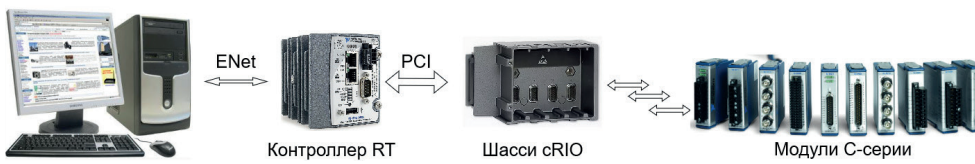


Рис. 6-11б. Внешний вид компонентов системы на основе контроллера реального времени с реконфигурируемым шасси и модулями C-серии

Шасси расширения в системах на основе модуля R-серии – это не просто объединительная панель, с помощью которой происходит объединение системы в одно целое. В то же время реконфигурируемое шасси не только соединяет модули с контроллером, но и выполняет функции управления и обработки данных в жестком реальном времени.

На рис. 6-11б устройства показаны без соблюдения масштаба, представление об их истинных размерах дает рис. 6-12, на котором изображена встраиваемая система управления объектом (**Embedded RT cRIO**).

Программное обеспечение систем на основе контроллера реального времени с реконфигурируемым шасси состоит из трех частей (рис. 6-13).

FPGA VI определяет структуру FPGA, в которой аппаратно реализуются алгоритмы обслуживания каналов ввода-вывода (в т.ч. созданных разработчиком), управления внешними объектами, обработки данных в режиме **жесткого аппаратного реального времени** и т.д. Важно отметить, что все процессы в FPGA (измерение или генерация аналоговых сигналов, цифровая фильтрация, обмен данными с внешними устройствами с использованием какого-либо

Рис. 6-12. Система Embedded RT cRIO

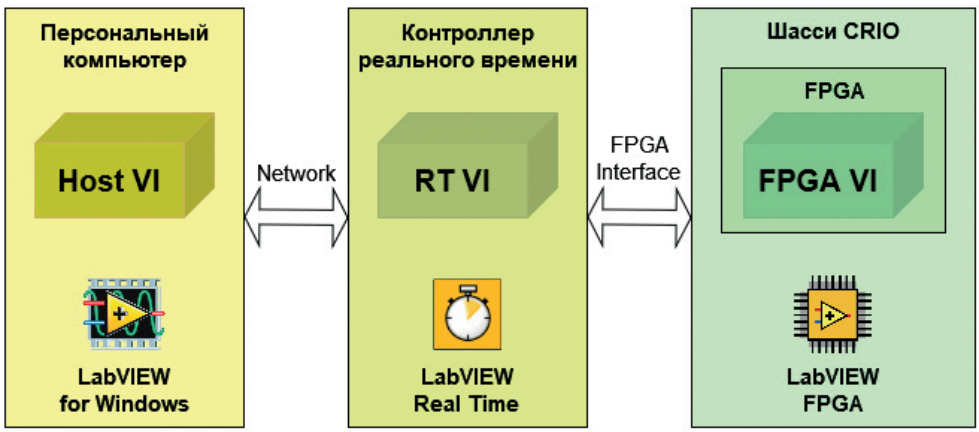


Рис. 6-13. Компоненты среды проектирования и прикладного программного обеспечения систем на основе контроллера реального времени и шасси cRIO

протокола и др.) могут выполняться независимо друг от друга и одновременно, с наивысшей степенью детерминизма, свойственной истинному параллелизму. Разработка такого VI осуществляется с помощью модуля **LabVIEW FPGA**.

Real Time VI – это программа, выполняемая в контроллере под управлением операционной системы реального времени, подобной Linux Real-Time, Windows Embedded, VxWorxs или Phar Lap ETS. Этот VI проектируется с использованием модуля **LabVIEW Real Time**. В общем случае **Real Time VI** решает задачи двух типов:

- критические по длительности и детерминированности времени выполнения. К ним относятся задачи обслуживания FPGA и каналов ввода-вывода, обработки данных на лету в процессе их поступления за строго определенное время и т.д. Для таких задач создаются VI, которым присваивается наивысший уровень приоритета (**Time Critical Interface VI**). Это программы жесткого программного реального времени.

- нормального уровня приоритета (**Normal Priority VI**). Они отвечают, например, за обмен данными с накопителями информации (**Data Storage**) или за сетевые коммуникации (**Network Communication**). Это **задачи мягкого программного реального времени**. Взаимодействие задач Real Time VI с различными уровнями приоритета организуется операционной системой путем обмена данными между программными потоками (**Inter-Thread Communication**). Взаимодействие детерминированной части алгоритма, реализуемой программно в контроллере реального времени, с компонентом, реализуемым аппаратно в FPGA, осуществляется с помощью функций **FPGA Interface**.

Host VI выполняется на процессоре персонального компьютера под управлением многозадачной операционной системы общего назначения и реализует наименее ответственную часть алгоритма, задержки в выполнении которой не могут привести к сбоям или некорректной работе всего приложения в целом. Host VI реализует некритичные к скорости и детерминизму алгоритмы обработки данных, получаемых из Real Time системы, запись данных в файлы и чтение из файлов, обеспечивает интерфейс с оператором, связь с другими системами и т.д. Программа Host VI разрабатывается в среде **LabVIEW for Windows**.

Программное обеспечение систем на основе контроллеров FlexRIO, дополненное CLIP связи с адаптером ввода-вывода FlexRIO, также организуется по схеме, показанной на рис. 6-13.

Встраиваемые системы CompactRIO применяют для автоматического управления различным промышленным и научным оборудованием, для автоматизации производства, сбора данных при экспериментальных исследованиях сложных процессов и объектов. Такие системы функционируют **автономно**, в их составе обычно нет персонального компьютера, но при необходимости к контроллеру может подключаться сенсорная панель, которая служит интерфейсом оператора.

Объединение нескольких систем **CompactRIO** и одного или нескольких компьютеров оператора в сеть образует **распределенную систему сбора данных и диспетчерского управления** (**SCADA**), программное обеспечение которой разрабатывается с использованием модуля **LabVIEW Data Logging and Supervisor Control** (**LabVIEW DSC**). В состав такой системы могут быть включены десятки контроллеров, обеспечивающих мониторинг состояния и управления большими объектами, рассредоточенных на больших площадях (например, энергосистемами районов, промышленными предприятиями или экспериментальными установками).

Характеристики систем cRIO определяются свойствами их основных компонентов – шасси и контроллеров cRIO. Вместе с модулями C-серии они образуют систему, которая может быть встроена в объект или использована в качестве узла распределенной системы. FPGA в качестве ядра системы связывает каналы ввода-вывода модулей с контроллером (рис. 6-14).

Благодаря индивидуальным каналам интерфейса SPI обеспечивается параллелизм выполнения операций измерения и генерации сигналов, поступающих от датчиков и формируемых для исполнительных устройств. А FPGA аппаратно реализует алгоритмы параллельной обработки данных. При этом результаты измерений могут быть пересланы в контроллер для более сложных вычислительных операций, так же как и подготовленные в контроллере для генерации данные могут быть отправлены через FPGA в модули вывода.

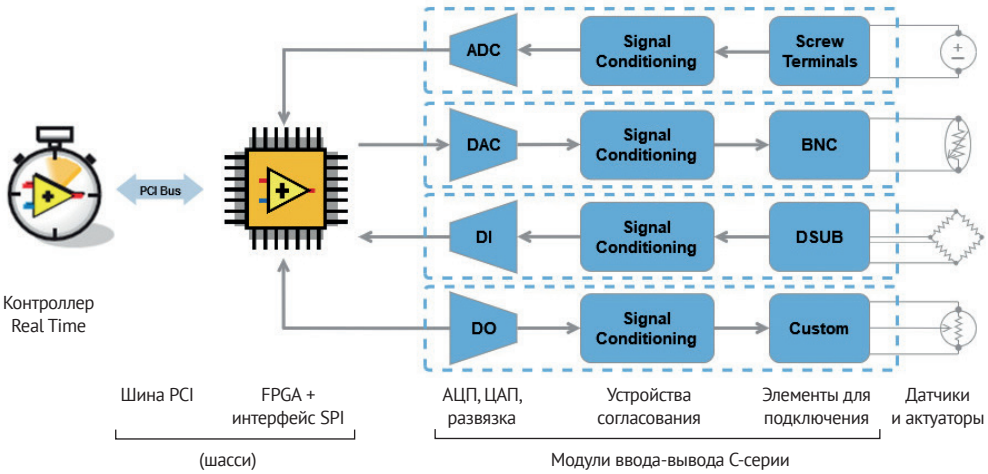


Рис. 6-14. Функциональная схема системы CompactRIO

Шасси CompactRIO первого поколения [6-9] рассчитаны на установку 4 или 8 модулей C-серии. В этих шасси находится FPGA Xilinx семейства LX (рис. 6-15а). Контроллер cRIO подключается к разъему на боковой стенке шасси. Шасси этого типа уже устарели и не рекомендуются для использования в новых разработках.

а б в

Рис. 6-15. Шасси систем CompactRIO

Новые модели шасси – это 4-, 8- или 14-слотовые шасси, предназначенные для расширения систем cRIO, PXI, систем на основе персонального компьютера и т.д. [6-10]. Такие шасси позволяют создавать системы, в составе которых используется только один главный контроллер или процессор традиционной архитектуры с операционной системой реального времени или общего назначения. На нем реализованы функции общего управления, регистрации данных, интерфейс оператора и иные функции, нетребовательные к производительности или детерминизму их выполнения. FPGA в каждом шасси расширения могут быть сконфигурированы для выполнения задач жесткого аппаратного реального времени.

Это позволяет в системе с большим количеством каналов распределить задачи обработки данных между спецпроцессорами, созданными в нескольких FPGA. В шасси расширения используются FPGA Xilinx семейств Spartan, Virtex и Zynq-7000. Шасси (рис. 6-15б) могут быть объединены в сеть Ethernet, в детерминированную сеть EtherCAT (рис. 6-15в), подключаться через интерфейс MXI и даже через беспроводной канал WSN (Wireless Sensor

Network). Тип интерфейса определяет возможное количество узлов в сети и, соответственно, количество каналов ввода-вывода, а также топологию сети и допустимое расстояние между узлами (до 100 м для Ethernet и EtherCAT и до 300 м для WSN).

Если система cRIO создается на базе простого шасси, подобного изображенному на рис. 16-5а, к шасси необходимо подключить контроллер, выполненный в виде сменного блока (рис. 6-16а). Но такая компоновка систем cRIO из поставляемых отдельно шасси и блока контроллера является морально устаревшей. В настоящее время корпорация National Instruments предлагает десятки моделей контроллеров cRIO [6-11] с уже подключенными шасси (рис. 6-15б), а также контроллеров, интегрированных в шасси (рис. 6-15в).

а б в

Рис. 6-16. Контроллеры систем CompactRIO

Контроллеры могут быть собраны на базе процессоров различных типов: от сравнительно простых Power PC с тактовой частотой 533 МГц до 4-ядерных 1,91 МГц Intel Atom, дополненных FPGA Kintex 7, или же на базе систем на кристалле семейства Zynq 70XX; оперативная память емкостью до 2 Гб, твердотельные диски объемом до 16 Гб. У всех контроллеров есть один или два порта Ethernet, некоторые модели предоставляют беспроводной Ethernet. Для подключения периферийных устройств предназначены порт RS-232 и до трех портов USB. В некоторых контроллерах есть также порты RS-485/RS-422, мини-дисплей порт и даже слот для сменной карты памяти. В современных контроллерах cRIO установлена операционная система Linux RT. Завершается компоновка системы cRIO установкой в шасси модулей ввода-вывода C-серии, краткая характеристика которых приведена в главе 2.

Для производства тиражируемой продукции с экономической точки зрения более привлекательным может оказаться использование бескорпусных одноплатных контроллеров **CompactRIO Single-Board Controller** (sbRIO) [6-12]. С точки зрения разработчика структура технических и программных средств системы на основе sbRIO полностью соответствует рис. 6-11а и 6-13. Отличие от систем cRIO, состоящих из шасси, контроллера и модулей ввода-вывода, состоит только в конструктивном исполнении.

Плата sbRIO имеет небольшие размеры (приблизительно 16×10 см или 10×10 см, рис. 6-17). На них установлены 2- или 4-ядерные процессоры Intel Atom, а также FPGA семейства Artix-7 или системы на кристалле Xilinx Zynq-70XX с 2-ядерным процессором ARM Cortex A9.

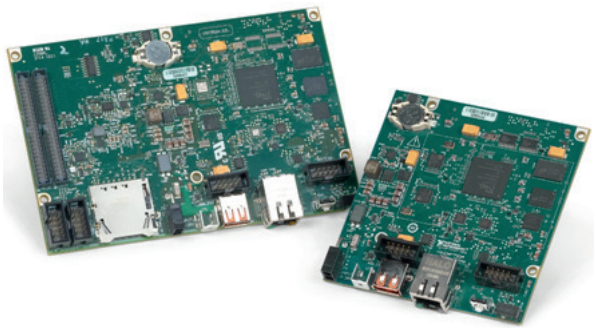


Рис. 6-17. sbRIO – одноплатные контроллеры CompactRIO

В распоряжении разработчика системы cRIO:

- оперативная память емкостью до 2 Гб;
- энергонезависимая память емкостью до 4 Гб;
- до шестнадцати 16-разрядных каналов аналогового ввода;
- до восьми 16-разрядных каналов аналогового вывода;
- до 100 линий цифрового ввода-вывода для сигналов разного типа логики;
- порты Ethernet, USB, RS-232, RS-485, CAN;
- слот для карты памяти.

У разных моделей контроллеров емкость памяти, количество портов и каналов ввода-вывода различны. В некоторых контроллерах предусмотрена возможность подключения до трех модулей C-серии, а в некоторых возможно подключение мезонинных плат **RIO Mezzanine Card** (**RMC**), в том числе собственной разработки.

К семейству одноплатных контроллеров относится также система в модуле **SOM** (**CompactRIO System on Module**) [6-13, 6-14]. Центральным элементом SOM (рис. 6-18) sbRIO-9651 является система на кристалле **Xilinx Zynq 7020**, в состав которой входит 2-ядерный процессор **ARM Cortex-A9** и **FPGA Artix 7**, а также оперативная память объемом 512 Мб.

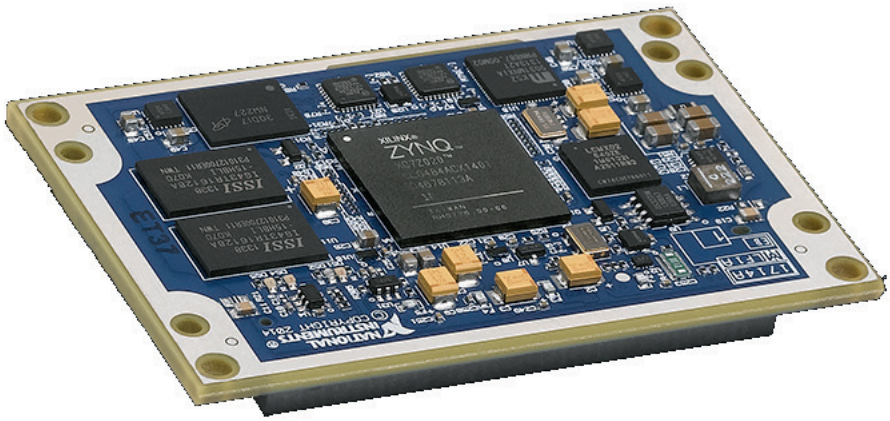


Рис. 6-18. Система в модуле sbRIO-9651

На твердотельный диск емкостью 512 Мб установлена операционная система NI Linux Real Time с пакетом драйверов каналов цифрового ввода-вывода, периферийных портов Ethernet, USB, RS-232, RS-485, CAN, карты памяти и поддержки модулей C-серии.

Из-за малых размеров (~76×50 мм) для подключения всех каналов ввода-вывода и портов, а также питания на плате SOM предусмотрен только один разъем с высокой плотностью контактов, напрямую соединенных с выводами FPGA. Поэтому для практического использования SOM необходимо установить на материнскую плату, на которой должны быть смонтированы соответствующие схемы согласования, интерфейсные драйверы портов, а также стандартные разъемы для подключения периферийных устройств, коммуникационных сетей и источников питания.

Разработчикам пользовательских плат для SOM предлагается комплект **NI sbRIO-9651 Development Kit**. В состав комплекта, кроме SOM, входит готовая материнская плата **Reference Carrier Board** со всеми необходимыми интерфейсными схемами и разъемами, блок питания, кабели и несколько миниатюрных сменных плат с датчиками, аналого-цифровым и цифроаналоговым преобразователями. Разработчиком предоставляется также подробное руководство с рекомендуемыми принципиальными и монтажными схемами [6-14].

На рис. 6-19 показаны функциональные блоки системы в модуле sbRIO-9651 и материнской платы. Их совместное использование упрощает разработку и отладку прототипов прикладных систем.

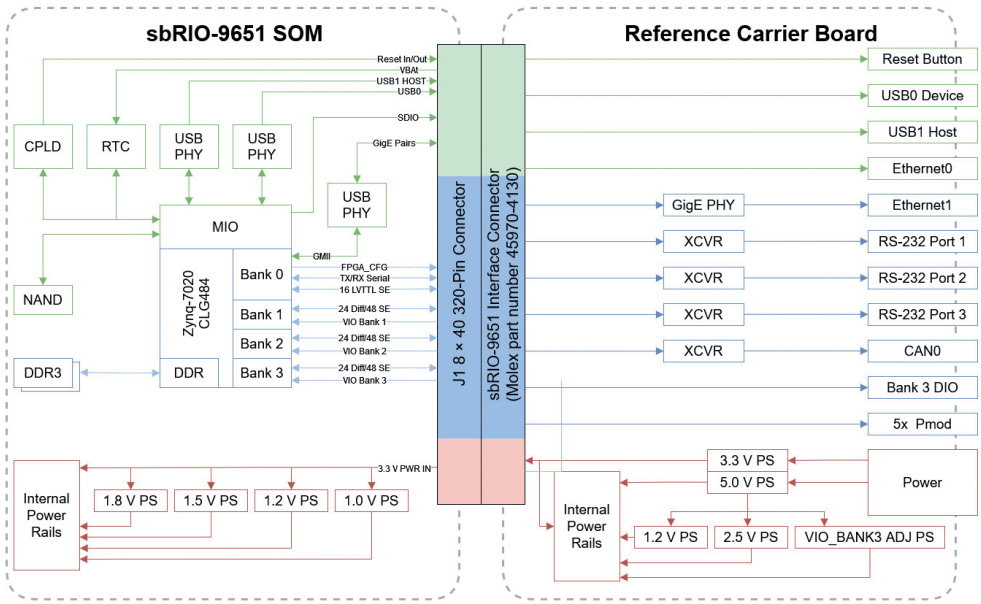


Рис. 6-19. Схема соединений sbRIO-9651 с материнской платой

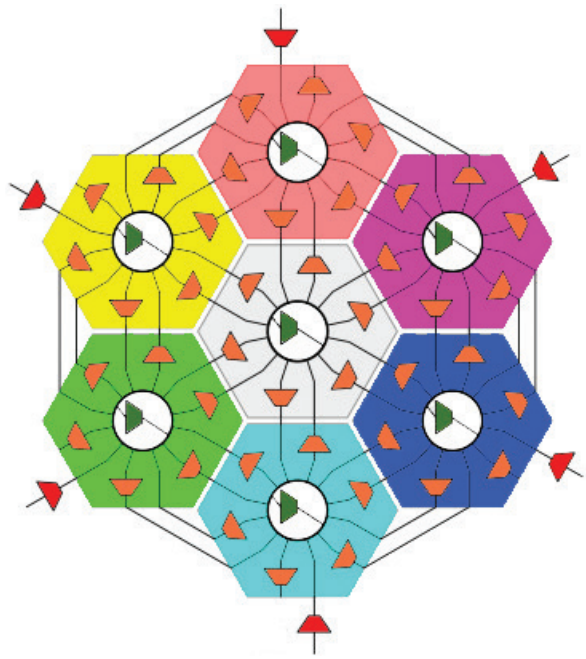
Конструктивное исполнение обеспечивает работу систем cRIO с естественным охлаждением в диапазоне температур от –20 °C до +55 °C или в расширен-

ном диапазоне от 40 °C до +70 °C (предельное значение: +85 °C), при вибрациях амплитудой до 5g и ударах до 50g.

Функциональные возможности, уникальные технические и массогабаритные характеристики, высокая надежность комплектующих и конструкции позволяют применять модульные и одноплатные системы CompactRIO в тяжелых условиях эксплуатации, в том числе на транспорте: в бортовом железнодорожном, морском и авиационном оборудовании, а также во многих других отраслях промышленности и специальных экспериментальных установках.

В обзор включены не все известные приложения технологии реконфигурируемого ввода-вывода. Эта технология эффективно применяется в осциллографах (семейство **Reconfigurable PXI Oscilloscope** – NI PXIe-517x), в промышленных контроллерах (семейство **Industrial Controller** – NI IC-31xx), в контроллерах систем машинного зрения (семейство **Compact Vision System** – NI CVS-145x), в системах программируемого радио (**USRP Software Defined Radio**) и в устройствах учебного назначения (**ELVIS**, **myRIO**). Но все типы технических средств, применяемые в упомянутых приложениях, различающиеся функциональными возможностями и техническими характеристиками, назначением и областью применения, интегрируются в системы прикладным программным обеспечением, разрабатываемым с использованием одних и тех же инструментов проектирования, о которых пойдет речь в следующей главе.

Заключение



Основная цель этой книги – дать представление о возможностях, которые открывают перед разработчиком современных систем программируемые логические интегральные схемы, а также о том, как можно облегчить и ускорить проектирование, используя простой в освоении, но эффективный инструментарий среды графического программирования LabVIEW, дополненной модулем LabVIEW FPGA.

Выбрав для создания систем аппаратные компоненты National Instruments, ядром которых является FPGA, и вооружившись таким инструментарием, можно не только повысить производительность труда, но и расширить функционал, улучшить технические характеристики разрабатываемых систем.

Чтобы убедить читателя в справедливости этих утверждений, в книге даны краткие сведения о технических компонентах (FPGA, SoC, модулях ввода-вывода, контроллерах) и программных средствах проектирования. Но самое главное – приведены примеры простых устройств и систем измерения, тестирования и управления с подробным описанием процесса разработки и получаемых результатов.

Если эти примеры оказались достаточно наглядны, их реализация описана понятно, а полученные характеристики соответствуют ожидаемым, значит, цель достигнута.

Следует отметить, что мы рассмотрели далеко не все технические средства реконфигурируемого ввода-вывода, а также не все возможности LabVIEW FPGA. В этой книге изложены только основы проектирования реконфигурируемых систем. Чтобы узнать, как в LabVIEW FPGA разрабатывать более сложные, обладающие повышенной надежностью высокопроизводительные системы, нужно глубже познакомиться с особенностями их архитектуры и техники программирования, детально описанными в специальных руководствах [E-1, E-2, E-3]. Все о LabVIEW FPGA собрано в центре документации **NI Product Documentation Center** [E-4], в подразделе **FPGA Module Related Documentation** [E-5] которого приведены ссылки на полезные ресурсы с информацией о компонентах и инструментальных средствах Xilinx, целевых FPGA-устройствах NI, а также ссылки на все необходимые руководства.

Технологии и устройства RIO эффективно применяются для решения задач автоматизации экспериментальных исследований в самых разных областях науки, при автоматизированных испытаниях промышленной продукции, на этапе программно-технического моделирования при разработке сложных объектов и т.д. Специфика ряда актуальных задач находит отражение в развитии технических и программных средств. LabVIEW открывает доступ к оборудованию и проблемно ориентированным библиотекам функций для создания и тестирования систем связи, систем машинного зрения и систем энергетики, в которых задачи управления измерениями и генерации сигналов, а также задачи обработки данных решаются на аппаратном уровне в FPGA. Благодаря этому удается реализовать нетривиальные алгоритмы и достичь высоких показателей производительности и детерминизма. Представляется, что проектированию реконфигурируемых систем для этих областей должны быть посвящены отдельные книги.

Еще одна важная сфера применения технологий, о которых идет речь в этой книге, – образование. Для разработки учебных лабораторных стендов и практикумов выпускаются универсальные лабораторные станции **NI ELVIS III**, **NI ELVIS RIO** [E-6, E-7], устройства программируемого радио семейства **USRP** [E-8] и студенческие контроллеры **myRIO** [E-9], **roboRIO** [E-10]. Все это оборудование с прикладным программным обеспечением реализовано на базе технологии реконфигурируемого ввода-вывода и широко применяется в ориентированном на проектное обучение учебном процессе при преподавании различных дисциплин. Контроллеры RIO даже применяются для проведения международных соревнований по робототехнике FRC (**The FIRST Robotics Competition**).

Хотя упомянутые устройства и поставляемые с ними библиотеки программного обеспечения в первую очередь предназначены для оснащения учебных лабораторий – на их основе разработано множество лабораторных практикумов [E-11], они могут быть эффективно использованы и инженерами для освоения принципов проектирования и прототипирования реконфигурируемых систем. Многие из рассмотренных в данной книге примеров апробированы и отлажены на контроллере myRIO. Переход с myRIO на более сложные модули и контроллеры сводился, как правило, к замене в проекте целевого устройства с соответствующим редактированием наименований ресурсов и перекомпиляции FPGA VI.

За прошедшие десятилетия FPGA стали для инженеров привычными компонентами, которые применяются в различных устройствах и системах, в том числе в составе систем на кристалле. Несомненный эффект перехода к обработке данных на аппаратном уровне и возможность сокращения времени проектирования стали основными причинами корректировки стратегических планов производителей классических микроэлектронных компонентов. В результате такой корректировки ведущие разработчики и производители микропроцессоров объединились с ведущими разработчиками и производителями программируемых логических интегральных схем – компания **Intel** с компанией **Altera**, а компания **AMD** приобрела компанию **Xilinx**. Непрерывно развиваются и создаются новые универсальные средства проектирования, с помощью которых можно программировать классические микропроцессоры (**Soft Processors**) и разрабатывать аппаратные средства обработки данных (**Logic Processors**), реализованные в том же кристалле, что и микропроцессор.

В этом отношении LabVIEW вне конкуренции – и традиционное программное обеспечение, и структура FPGA, и каналы связи с окружающим миром проектируются в одной среде по одной методике с использованием универсальных инструментов. Благодаря присущей объектам и инструментам наглядности и простоте инженерной интерпретации освоение LabVIEW не требует сколь-нибудь заметных усилий, а применение этой системы в практической деятельности существенно сокращает время, необходимое для создания новых приложений.

Необходимо также отметить, что процесс разработки новых систем, как правило, базируется на компоновке специализированных и/или универсальных технических средств, представляющих собой сравнительно крупные функциональные узлы, с последующим конфигурированием и программированием их взаимодействия. Широкая номенклатура модулей, контроллеров, адаптеров, модульных измерительных приборов, производимых компанией National Instruments по технологии реконфигурируемого ввода-вывода, позволяет выбрать функциональные узлы, в максимальной степени удовлетворяющие требованиям решаемой задачи. Функциональные возможности и технические характеристики таких средств, определяемые во многом свойствами их компонентов – микросхем FPGA, систем на кристалле, аналого-цифровых и цифроаналоговых преобразователей и т.п., непрерывно улучшаются благодаря развитию микроэлектронной технологии. Так, сформулированное в предыдущем издании книги предположение о включении систем на кристалле в состав реконфигурируемых устройств давно реализовано в промышленном производстве.

Не подтвердилось предположение о практическом внедрении аналоговых программируемых интегральных схем – **Field Programmable Analog Array** (**FPAA**) – эти технологии развиваются пока еще медленно [E-12, E-13]. Тем не менее исследования и разработки в этой области продолжаются, есть определенные достижения, и можно ожидать, что симбиоз цифровых и аналоговых программируемых интегральных схем вскоре откроет новые возможности в системотехнике.

И еще об одной перспективе, открывающейся в связи с бурным развитием аппаратной реализации методов искусственного интеллекта на базе FPGA

и FPAA. Известны примеры практического применения этих методов для решения задач на цифровом уровне в FPGA, созданы специальные системы проектирования [E-13, E-14]. По-видимому, в скором времени аналогичные возможности появятся и в LabVIEW. Хочется надеяться, что мы, вместе с читателями этой книги, по крайней мере будем успевать за развитием уже существующих и новых перспективных технологий.

Желаю всем интересных задач и успехов в их решении!

Ефим Баран

Книги издательства «ДМК Пресс» можно заказать
в торгово-издательском холдинге «КТК Галактика» наложенным платежом,
выслав открытку или письмо по почтовому адресу:
115487, г. Москва, пр. Андропова д. 38 оф. 10.
При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.
Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: **www.galaktika-dmk.com**.
Оптовые закупки: тел. **(499) 782-38-89**.
Электронный адрес: **books@alians-kniga.ru**.

Баран Ефим Давыдович
Романов Александр Юрьевич

Проектирование реконфигурируемых систем в LabVIEW FPGA

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Зам. главного редактора *Сенченкова Е. А.*
Корректор *Синяева Г. И.*
Верстка *Луценко С. В.*
Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16.
Гарнитура «PT Serif». Печать цифровая.
Усл. печ. л. 52,49. Тираж 100 экз.

Веб-сайт издательства: *www.dmkpress.com*