

Содержание

Отзывы о книге.....	8
Об авторах.....	10
Предисловие Е.Д. Барана.....	12
Предисловие А.Ю. Романова.....	16
Благодарности.....	19
Глава 1. Программируемые логические интегральные схемы	20
1.1. Простые программируемые логические устройства.....	22
1.2. Технологии программирования ПЛИС.....	26
1.3. Пограничное сканирование и JTAG-интерфейс.....	28
1.4. Сложные программируемые логические устройства.....	32
1.5. Оперативно программируемые логические матрицы.....	37
1.6. Сравнение архитектур ПЛИС.....	44
1.7. Современные программируемые логические схемы Xilinx FPGA 7 серии.....	46
1.8. Средства проектирования цифровых устройств на ПЛИС.....	51
1.9. Применение ПЛИС.....	61
1.10. Заключение.....	65
Глава 2. Устройства ввода-вывода.....	66
2.1. Многофункциональные модули ввода-вывода классической архитектуры.....	68
2.1.1. Блок аналогового ввода.....	70
2.1.2. Блок аналогового вывода.....	73
2.1.3. Блок цифрового ввода-вывода.....	74
2.1.4. Блок таймерного ввода-вывода.....	76
2.1.5. О взаимодействии блоков многофункционального модуля вывода-вывода.....	77
2.2. Модули ввода-вывода со встроенным кондиционированием сигналов.....	79
2.3. Модули ввода-вывода С-серии.....	84
2.4. Реконфигурируемые модули ввода-вывода серии R.....	89
2.5. Реконфигурируемые модули ввода-вывода серии FlexRIO.....	92
2.6. Заключение.....	97
Глава 3. Виртуальные измерительные приборы и программное обеспечение National Instruments.....	98
3.1. История появления LabVIEW.....	100
3.2. Основные свойства LabVIEW.....	101

3.3. История развития технологии виртуальных измерительных приборов	105
3.4. Measurement and Automation eXplorer	107
3.4.1. Создание симулируемых устройств ввода-вывода	109
3.4.2. Конфигурирование и тестирование симулируемых устройств ввода-вывода.....	113
3.4.3. Конфигурирование и тестирование реальных устройств ввода-вывода	120
3.4.4. Создание задачи.....	122
3.4.5. Конфигурирование программного обеспечения среды проектирования.....	129
3.4.6. Конфигурирование сетевого окружения.....	133
3.5. Заключение	137

Глава 4. Организация среды проектирования LabVIEW..... 138

4.1. Запуск LabVIEW. Начало работы	141
4.2. Создание проекта.....	145
4.3. Редакторы для проектирования программ LabVIEW	149
4.4. Инструменты редакторов программ	151
4.4.1. Инструментальные линейки кнопок	151
4.4.2. Палитра инструментов Tools Palette	154
4.4.3. Объекты программ LabVIEW. Пример программы.....	156
4.4.5. Палитра объектов лицевой панели Controls Palette	163
4.4.6. Палитра объектов блок-диаграммы Functions Palette.....	176
4.5. Заключение	201

Глава 5. Техника программирования в среде LabVIEW..... 202

5.1. Разработка лицевой панели и настройка объектов лицевой панели.....	203
5.1.1. Настройка свойств объекта из контекстного меню.....	207
5.1.2. Задание свойств объекта в окне Properties	210
5.1.3. Массивы и кластеры на лицевой панели.....	212
5.2. Разработка блок-диаграммы.....	215
5.2.1. Соединение узлов блок-диаграммы. Первая программа.....	218
5.2.2. Техника проектирования программ. VI генератора сигналов	221
5.2.3. Разработка пиктограммы VI	222
5.2.4. Вызов подпрограмм subVI. Цикл While. Ошибки проектирования.....	227
5.3. Техника отладки программ в LabVIEW.....	232
5.3.1. Устранение ошибок до компиляции, или почему в LabVIEW мало грубых ошибок	232
5.3.2. Отладка программ с помощью пробников и контрольных точек	233
5.3.3. Средства пошаговой отладки программ. Анимация выполнения программы.....	238
5.3.4. Кластер ошибок. Интерпретация ошибок выполнения программы	240
5.3.5. Помощь в среде проектирования LabVIEW.....	242

5.4. Разработка блок-диаграммы. Основные структуры.....	245
5.4.1. Структуры итерационной обработки данных (циклы While и For)	245
5.4.2. Туннели и регистры сдвига	247
5.4.3. Разработка блок-диаграммы. Продолжение	251
5.4.4. Структура выбора.....	253
5.4.5. Управление свойствами объектов	254
5.4.6. Переменные в LabVIEW	257
5.4.7. Обмен данными с использованием Channel Wire	259
5.5. Программирование операций ввода-вывода	264

Глава 6. Архитектура реконфигурируемых систем

измерения и управления.....	272
6.1. Компоненты реконфигурируемых систем.....	274
6.2. Системы на основе модуля R-серии	277
6.3. Высокопроизводительные системы на платформе FlexRIO	279
6.4. Системы на платформе CompactRIO	282

Глава 7. Среда проектирования реконфигурируемых систем

7.1. Особенности среды LabVIEW FPGA.....	292
7.2. Генерация кода, загружаемого в FPGA.....	293
7.3. Палитра Controls LabVIEW FPGA	294
7.4. Палитра Functions LabVIEW FPGA.....	296
7.4.1. Субпалитра функций обработки данных.....	298
7.4.2. Субпалитра функций управления Control	301
7.4.3. Субпалитра Utilities	306
7.4.4. Субпалитра High Throughput Math.....	309
7.4.5. Субпалитра матричных вычислений	312
7.4.6. Экспресс-функции субпалитры FPGA Math & Analysis.....	313
7.4.7. Субпалитра ввода-вывода FPGA I/O	314
7.4.8. Субпалитра узлов для работы с памятью FPGA.....	319
7.4.9. Функции управления таймингом. Субпалитра Timing FPGA	326
7.4.10. Субпалитра функций синхронизации задач в FPGA.....	328
7.4.11. Интеграция IP в FPGA VI	330
7.5. Методы и средства отладки FPGA-приложений.....	332

Глава 8. Разработка реконфигурируемых систем в LabVIEW

8.1. Этапы разработки реконфигурируемых систем.....	339
8.1.1. Создание проекта системы на основе модуля R-серии.....	341
8.1.2. Программирование целевой платформы. Разработка FPGA VI.....	346
8.1.3. Тактирование и синхронизация в FPGA.....	358
8.2. Сборка и компиляция FPGA VI.....	362
8.3. Параллелизм выполнения операций в FPGA	373
8.3.1. Разделяемые ресурсы	378
8.4. Оптимизация FPGA VI	384
8.4.1. Оптимизация ресурсов FPGA	385
8.4.2. Оптимизация производительности FPGA VI	388

Глава 9. Управление FPGA VI. Разработка Host VI.....	394
9.1. Субпалитра FPGA Interface	398
9.1.1. Функция Invoke Method	402
9.2. Программный обмен данными через элементы лицевой панели	405
9.2.1. Обработка событий интерфейса оператора. Структура Event	411
9.3. Обмен данными по прерываниям.....	416
9.4. Обмен данными с использованием канала прямого доступа к памяти	420
Глава 10. Обработка данных в приложениях LabVIEW FPGA.....	428
10.1. Фильтрация	429
10.1.1. Применение экспресс-функций для фильтрации в FPGA VI	430
10.1.2. Разработка Host VI для Filter_FPGA.vi	435
10.1.3. Разработка нестандартных фильтров.....	438
10.1.4. Тестирование FPGA приложений. FPGA Desktop Execution Node	449
10.2. Быстрое преобразование Фурье в FPGA	455
Глава 11. Стандартные интерфейсы для работы с периферийными устройствами.	
Комплексное тестирование приложений	462
11.1. Интерфейс SPI в FPGA.....	466
11.2. Разработка SPI Master в FPGA на основе шаблона конечного автомата	471
11.3. Интерфейс I2C в FPGA	477
11.4. Логический анализатор	484
11.5. Тестирование разработанных устройств.....	509
11.6. Дополнения и выводы	516
Интерфейсы	516
Логический анализатор	517
Глава 12. Измерение неэлектрических величин.	
Расширение систем, выполненных на модулях R-серии	520
12.1. Системы на основе модуля R-серии с шасси расширения и модулями C-серии	522
12.2. Программирование модулей C-серии. FPGA VI	530
12.3. Программирование модулей C-серии. Host VI	532
Глава 13. Разработка встраиваемых и распределенных систем на платформе cRIO	544
13.1. Модели программирования систем CompactRIO	547
13.2. Компоновка контроллера cRIO	550
13.3. Создание и компоновка проекта	552
13.4. Конфигурирование компонентов проекта.....	556
13.5. Разработка FPGA VI	563

13.6. Организация обмена данными между контроллером и компьютером	566
13.6.1. Тестирование обмена данными в распределенной системе	576
13.7. Разработка RT VI	578
13.8. Разработка PC VI	583
13.9. Тестирование	589

Глава 14. Создание и развертывание исполняемых файлов592

14.1. Создание и развертывание двоичного файла, исполняемого в FPGA	593
14.2. Создание и развертывание исполняемого файла для выполнения на контроллере cRIO	596
14.3. Создание приложения для выполнения на компьютере	599

Глава 15. Генерация HDL из проекта LabVIEW FPGA..... 614

15.1. Установка LabVIEW FPGA IP Export Utility	615
15.2. Начало работы с LabVIEW FPGA IP Export Utility	615
15.3. Элементы управления и индикаторы	618
15.4. Экспорт IP-блока	620
15.5. Описание портов IP-блока	623
15.6. Использование IP-блока в проекте Vivado	624
15.7. Сопряжение IP по AXI	627
15.8. Запуск поведенческого моделирования	628
15.9. Использование однократно запускаемого IP-блока	628
15.10. Работа с множественными доменами тактовых сигналов	629
15.11 Экспорт HDL-кода в САПР других производителей ПЛИС	633
Заключение	634
Список литературы	638

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, сейчас способен решить процессор мобильного телефона, который может уместиться в ладони. Сейчас логические устройства встраиваются в различную технику, окружающую человека: автомобили способные управляться 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 чём 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 студентами, 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 приёмы 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: увеличилась степень интеграции и быстродействие FPGA, значительно расширился состав функциональных узлов. Теперь в один кристалл интегрированы многоядерные процессоры, аналого-цифровые и цифро-аналоговые 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, представленными 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 задаче. 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 [B-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 составе первой версии LabVIEW NXG модуль LabVIEW FPGA еще не был включен, а его расширение в последующие годы отставало от продолжающегося развиваться модуля LabVIEW FPGA предыдущего поколения. Поэтому, чтобы не терять время, было решено большую 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 [B-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 много 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 построена система LabVIEW.

Представляется, что информации, предлагаемой читателю этой книги, будет достаточно для ускоренного освоения перспективной технологией про-

ектирования 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 примере 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 инструмент 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 работу. 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 современной и более дружественной для читателя. И теперь, по прошествии 9 месяцев работы, я, теперь 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 также 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.

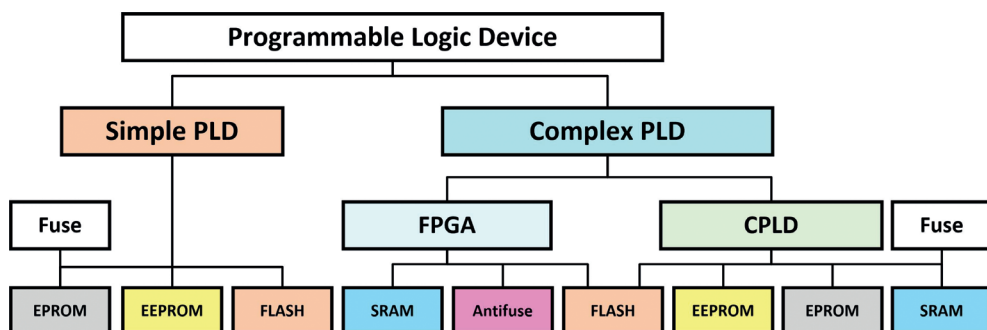
Рукопись книги, которую вы сейчас читаете, подготовили и прочитали два автора, научный редактор, один корректор и еще несколько рецензентов, но, безусловно, книга не идеальна. Мы будем очень признательны тем внимательным читателям, которые обнаружат в данном издании какие-либо ошибки или опечатки и сообщат 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:

- 1.1. Простые программируемые логические устройства
- 1.2. Технологии программирования ПЛИС
- 1.3. Пограничное сканирование и JTAG-интерфейс
- 1.4. Сложные программируемые логические устройства
- 1.5. Оперативно программируемые логические матрицы)
- 1.6. Сравнение архитектур ПЛИС
- 1.7. Современные программируемые логические схемы Xilinx FPGA 7 серии
- 1.8. Средства проектирования цифровых устройств на ПЛИС
- 1.9. Применение ПЛИС
- 1.10. Заключение

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 вентилях, 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, сблизил 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, содержащих тысячи логических вентилях, 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.

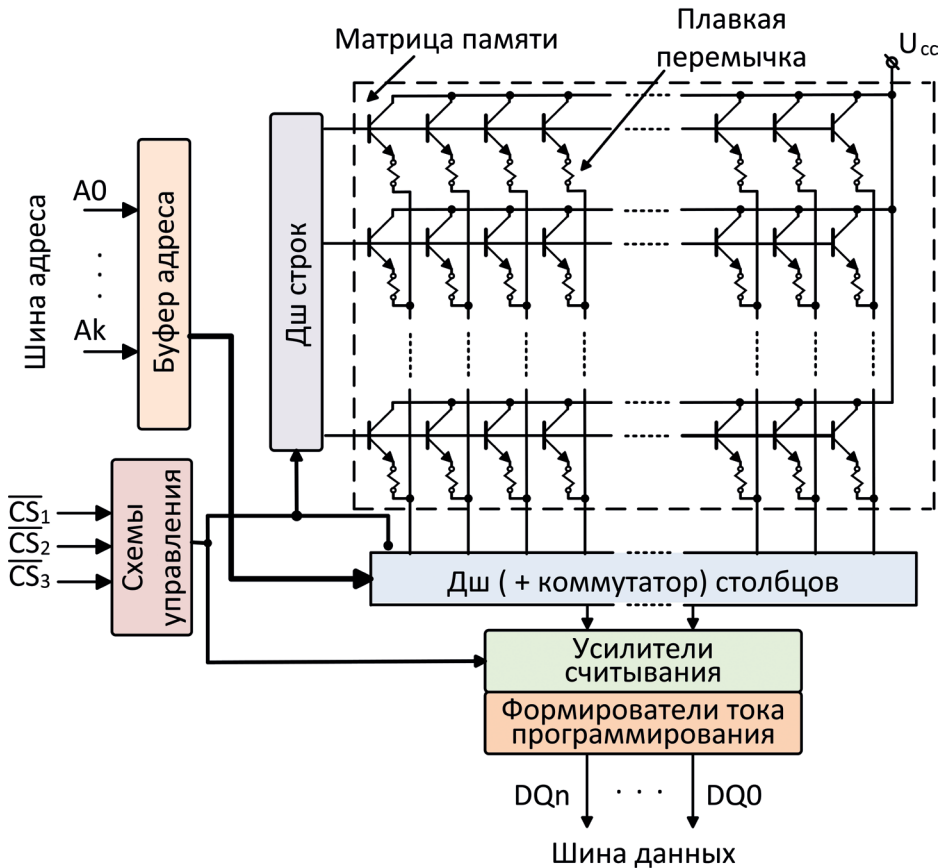


Рис. 1-1. Структура ППЗУ

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).

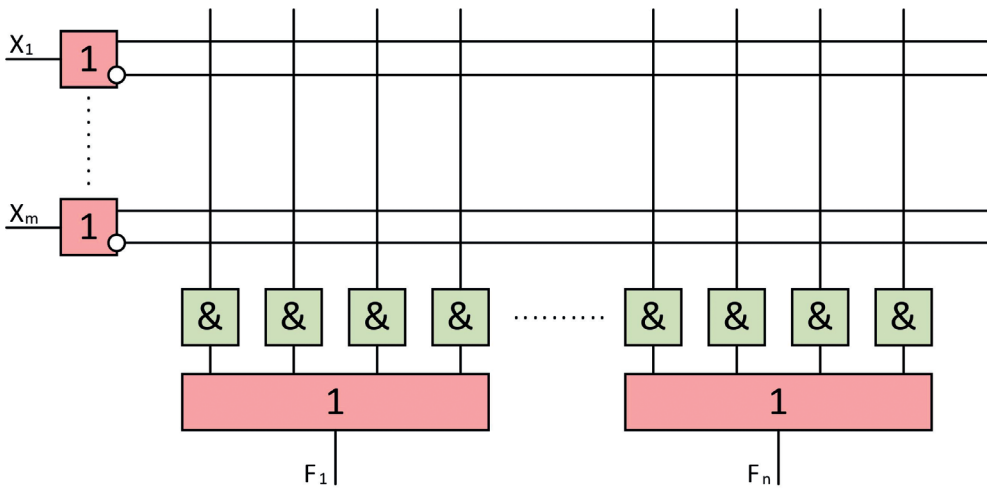


Рис. 1-2. Структура ПЛМ

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, а 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 памя-

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

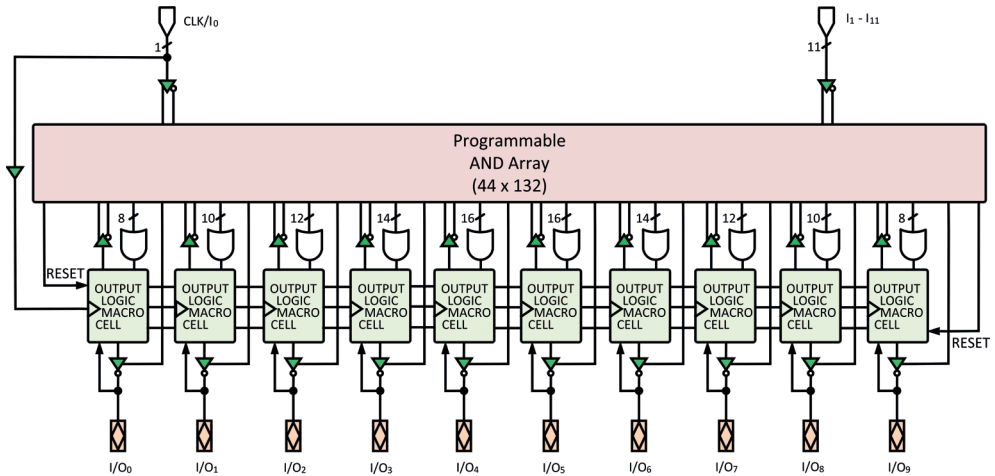
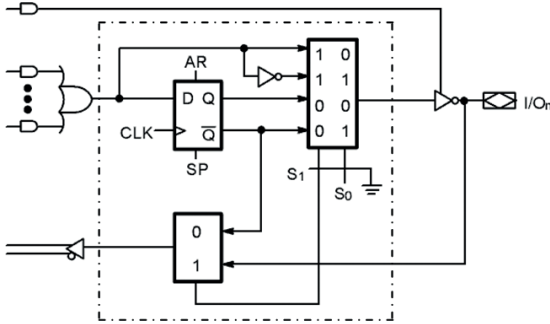


Рис. 1-3. Структура ПЛИС с элементами памяти

Эта микросхема, относящаяся к классу PAL, содержит программируемую матрицу элементов «И» (**Programmable AND Array**), на входы которых $I_1 \div I_{11}$ 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» управляют 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 переключателей **S0** и **S1** (рис. 1-4). Этими переключками 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 (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 2 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 вентилях) 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 вентилях 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 (сокетe), 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 1971г. фирмой **Intel**, а в 1974г. – фирмой **Toshiba** и реализованы в **перепрограммируемых ПЗУ (ПППЗУ)**.

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 – **Electrically 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, используемых 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 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).

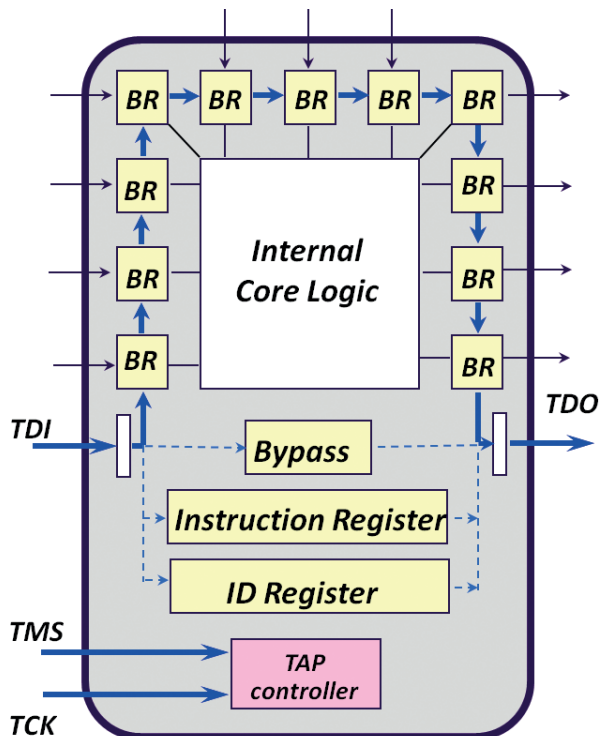


Рис. 1-5. TAP-порт



ТАР-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, ТАР-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 ТАР-порта и от нескольких десятков до нескольких тысяч дополнительных логических элементов и триггеров, что обычно составляет не более единиц процентов от общего количества логических вентилях 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 ТАР-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.