КОМПЬЮТЕРНАЯ ИНЖЕНЕРИЯ И ТЕХНИЧЕСКАЯ ДИАГНОСТИКА

УДК 681.325: 519.713

ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ СИСТЕМ НА СБИС ПРОГРАММИРУЕМОЙ ЛОГИКИ

(аналитический обзор)

Часть 2

РУСТИНОВ В.А., ХАХАНОВА И.В., БАРАБАШ А.А., ГЕРАСИМОВ М.А.

Данная публикация посвящена анализу современных средств проектирования на базе чипов PLD, особенностям применения пакетов САПР ведущих западных фирм.

1.Общие методы проектирования на основе PLD

Методика проектирования на основе PLD включает в себя пять основных последовательно выполняемых этапов [1]:

– ввод исходных данных проекта;

– функциональное моделирование;

– выполнение синтеза отдельных частей проекта и всего проекта в целом;

– временное моделирование проекта;

– программирование и тестирование PLD.

В общем случае разработка сложных иерархических проектов на PLD ничем не отличается от известных подходов к проектированию цифровых систем. Здесь возможно проектирование как “сверху вниз”, так и “снизу вверх”, использование библиотек стандартных блоков и функциональных узлов, а также заимствование опыта предыдущих разработок. Рассмотрим более подробно основные этапы методики проектирования на основе PLD.

1.1. Ввод проекта

Ввод проекта заключается в описании одним из способов машинного представления всех частей проекта. Различные пакеты имеют разные способы ввода, но наиболее распространенными являются следующие:

– текстовый;

– схемный;

– в виде временных диаграмм;

– задание графа переходов FSM.

Текстовый ввод предполагает описание проекта (или его части) на некотором исходном языке используемого программного средства в виде текстового файла. Наибольшее распространение получили языки описания аппаратуры VHDL и Verilog. Текстовый файл исходного описания проекта, как правило, включает заголовок, определение переменных и назначение им соответствующих выводов PLD, а также описание функционирования устройства в виде логических уравнений, алгоритма функционирования, таблицы истинности или конечного автомата. Более детально данный процесс ввода проекта будет рассмотрен ниже.

Схемный ввод осуществляется с помощью графического редактора используемого программного средства. Для удобства ввода принципиальных схем больших проектов графический редактор, как правило, содержит библиотеки стандартных элементов жесткой логики (например, серии 7400), а также библиотеку параметризированных функциональных узлов (вентилей, шифраторов, дешифраторов, мультиплексоров, демультиплексоров, триггеров, регистров, счетчиков и т.д.).

Ввод проекта в виде временной диаграммы осуществляется с помощью графического редактора. Вначале определяются переменные (сигналы) проекта, а затем описывается поведение устройства в виде временной диаграммы.

Способ ввода проекта в виде графа переходов позволяет просто и наглядно задать поведение проектируемых цифровых автоматов при разработке различных устройств управления.

Практически все современные пакеты САПР допускают ввод иерархических проектов, причем в одном проекте применяется сочетание разнообразных способов ввода. Сложный иерархический проект не обязательно сразу вводить полностью. Его можно вводить и компилировать по ветвям дерева иерархии.

1.2. Синтез проекта

Автоматизированный синтез цифровых проектов фактически заключается в компиляции исходного описания проекта во внутреннее представление используемого программного средства (абсолютный файл и/или JEDEC-файл для настройки PLD). В последующем абсолютный файл используется для моделирования. Многие пакеты также позволяют формировать файлы для связи проекта с другими пакетами функционально-логического и конструкторского проектирования. Задачи оптимизации, решаемые на этом этапе, сводятся к минимизации логических функций, управляющих выходными макроячейками PLD.

Синтез сложных проектов на CPLD дополнительно требует специальной программы, называемой упаковщиком (fitter), которая выполняет “подгонку” проекта (fitting) в заданную структуру CPLD. При синтезе сложных проектов следует планировать 20-40% использования ресурсов CPLD для возможности корректировок и эффективной работы упаковщика. Проект рекомендуется создавать путем выполнения ряда итераций: вначале реализуется ядро проекта, а затем добавляется остальная логика до полного проекта. При этом обеспечивается лучшее решение по упаковке проекта и наиболее рациональное назначение сигналов внешним выводам.

1.3. Моделирование проекта

Моделирование делится на функциональное, временное (временной анализ) и физическое [2]. Функциональное моделирование выполняется программным обеспечением на основании тестовых входных векторов и анализа полученных выходных сигналов. Тестовые векторы могут задаваться в

файле исходного описания проекта или находиться в отдельном файле. Многие пакеты допускают задание тестовых векторов в виде временных диаграмм. При функциональном моделировании возможно решение следующих задач:

– определение выходных значений по заданным входным воздействиям;

– сравнение вычисленных выходных значений с эталонными;

– моделирование неисправностей устройства.

Временное моделирование выполняется на временных моделях PLD и заключается в определении времени прохождения и формирования различных сигналов. Результаты временного моделирования могут также представляться в виде временных диаграмм.

Кроме того, с помощью отдельной программы, называемой временным анализатором, производится анализ в целях обнаружения путей сигналов, критичных по скорости. Оптимизация путей распространения сигналов может повысить быстродействие всего проекта.

Моделирование проекта на физическом уровне (тестирование на программаторе) осуществляется после настройки PLD. Для этого в файл, содержащий информацию о настройке PLD и управляющий работой программатора, добавляется информация для тестирования устройства. Последняя может быть получена на основании тестовых векторов и результатов функционального моделирования, причем здесь возможно моделирование в реальном масштабе времени, в том числе на предельной частоте работы PLD.

PLD, поддерживающие JTAG-стандарт, могут тестироваться непосредственно на плате методом граничного сканирования. Для этого программным обеспечением на основании тестовых векторов создаются тестовые последовательности в JTAG-стандарте. При этом допускается тестирование:

– одного PLD;

– цепочки PLD;

– всех устройств проекта (в том числе и цепочки PLD), поддерживающих JTAG-стандарт.

1.4. Программирование PLD

Программирование PLD заключается в его настройке на заданный алгоритм функционирования. Стандартные PLD программируются с помощью программаторов. Технологии программирования (последовательности подаваемых сигналов, уровни напряжений и др.) могут существенно отличаться даже для одних и тех же PLD, но производимых различными фирмами. Поэтому важное значение имеет использование только сертифицированных программаторов, рекомендуемых фирмами-изготовителями PLD. PLD, в которых настраиваемым элементом является SRAM, конфигурируются всякий раз при включении питания. Процесс конфигурирования состоит из двух фаз: загрузки данных и обнуления всех регистров. Данные о настройке CPLD могут поступать от управляющего компьютера, микропроцессора, ПЗУ, ОЗУ, других PLD. Форма передаваемых данных может быть как последовательная, так и параллельная. Имеется также ряд режимов программирования, когда PLD выступает в качестве активного устройства (само управляет процессом загрузки данных) и в качестве пассивного устройства (другое устройство управляет процессом загрузки данных).

Как правило, фирмы-изготовители PLD выпускают специализированные ППЗУ для программирования PLD. В такие ППЗУ с помощью программатора записывается информация о настройке PLD и они устанавливаются на плате вместе с PLD. В целях минимизации площади платы для передачи данных между ППЗУ и PLD используется последовательный интерфейс, а PLD выступает в качестве активного устройства.

Современные PLD, поддерживающие JTAG-стандарт, могут программироваться на плате, используя сигналы JTAG-стандарта. Для этого на границе платы устанавливается специальный разъем для передачи сигналов управления процессом программирования. Несколько PLD на одной плате могут объединяться в цепочки, но в каждый момент времени допускается программирование только одного PLD.

1.5. Общая структура алгоритмов проектирования на основе CPLD и FPGA

Анализируя сказанное выше, видим, что процесс проектирования сильно отличается для различных PLD. Наиболее простую структуру имеет алгоритм проектирования на основе стандартных PLD. Фактически это последовательное выполнение рассмотренных выше четырех этапов, за исключением того, что моделирование разделено на логическое и физическое. Логическое моделирование выполняется сразу после компиляции проекта, а физическое осуществляется с помощью программатора сразу после настройки PLD. В настоящее время стандартные PLD применяются крайне редко, поэтому уделим особое внимание этапам проектирования на двух самых мощных классах современных PLD: CPLD и FPGA. Общая структура алгоритма проектирования на основе CPLD приведена на рис.1.

Данный алгоритм проектирования применим для CPLD, программируемых с помощью программатора. Он отличается от стандартного тем, что в нем добавляется операция подгонки проекта в заданную структуру CPLD. Если проект не удается “вложить” в некоторую CPLD, следует выбрать другую микросхему с большими возможностями. В процессе подгонки САПР обычно указывает, каких именно ресурсов CPLD недостаточно для реализациии проекта. В алгоритме проектирования на основе CPLD, настраиваемого от ПЗУ, добавляется операция программирования ПЗУ, а при настройке с помощью JTAG-стандарта соответственно добавляются операции подготовки CPLD для программирования в этом режиме. Отметим, что выполнять физическое моделирование в последнем случае гораздо проще.

Этапы проектирования чипов FPGA несколько отличаются от рассматриваемых выше особенностей проектирования CPLD [3]. Это связано, прежде всего, с архитектурными особенностями микросхем данного класса. Кроме того, большинство современных пакетов проектирования на базе FPGA используют языки описания аппаратуры не только при вводе проекта, но и на стадии моделирования. При этом тестовые воздействия представляют собой простой код (например VHDL), подключаемый к проекту на различных этапах моделирования. Общая структура методологии проектирования на основе FPGA приведена на рис.2.

Рис.1

После синтеза проекта на концептуальном уровне существует два пути для его реализации. Например, можно использовать языки описания аппаратуры для реализации всего проекта целиком. Однако такой способ достаточно трудоемок, ведь весь проект тяжело реализовать, пользуясь только текстовым редактором. Более предпочтительным является комбинированный способ ввода проекта. При этом одни части проекта можно реализовывать с помощью схемного редактора, другие – при помощи редактора графов переходов, третьи – при помощи кодирования на языке VHDL или Verilog. Если за основу проектирования принят язык VHDL, то все полученные таким образом модули можно представить VHDL-кодом и связать определенным образом. Это можно сделать при помощи как текстового редактора, так и редактора схем. Главное удобство при этом заключается в том, что на языке описания аппаратуры будут представлены не только части проекта, реализуемые в различных редакторах, но и процедуры тестирования, необходимые для проверки проекта на различных этапах синтеза. При разработке проектов на базе FPGA используется три этапа моделирования, как это показано на рис. 3.

Главная задача такого тестирования состоит в проверке того, что на завершающей стадии проектирования разрабатываемая цифровая система функционирует так же, как и на уровне регистровых передач.Тестирование представляет собой отдельный набор VHDL или Verilog-кода, который связан со входами и выходами проекта системы. Единожды созданный, код тестирования остается неизменным в течение всего процесса проектирования и используется для проверки правильности функционирования системы. Тестирование стенда преследует две цели. Во-первых, оно предоставляет информацию о входных и выходных сигналах системы, таких как синхронизация, сброс и входные данные, с которыми она столкнётся после физической реализации и установки в конечную аппаратуру. Во-вторых, становится возможным производить регрессивное тестирование выполняемых функций системы в любой момент процесса VHDL-проектирования.

Рис. 2

Помимо всего прочего, такая методика разработки позволяет делать проекты “открытыми”. Это значит, что закодированный на языке описания аппаратуры проект может служить частью другого, более сложного проекта. При этом главный проект может разрабатываться совсем на другой платформе, с использованием семейств чипов других классов и фирм. Кроме того, помимо экспорта проекта возможен и импорт из других САПР, что делает разрабатываемый проект более гибким. Естественно, что время разработки таких комплексных проектов значительно уменьшается, что соответствует современным требованиям проектирования.

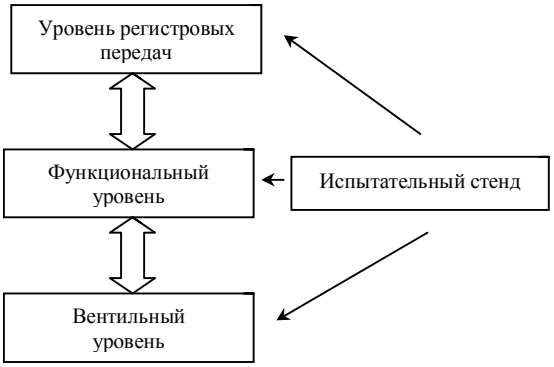

Рис. 3

2. Системы проектирования фирм Altera, Xilinx, Actel

Рассмотрим САПР крупнейших фирм-изготовителей микросхем PLD. Самой известной у нас в стране является корпорация Altera. Именно с ее системы MAX+PLUS II мы и начнем наш обзор.

2.1. САПР MAX+PLUS II фирмы Altera

Название системы MAX+PLUS II является аббревиатурой от Multiple Array MatriX Programmable Logic User System [4-6]. Система MAX+PLUS II имеет средства удобного ввода проекта, компиляции и отладки, а также непосредственного программирования устройств. Данная САПР свободно доступна на сайте компании, а также на свободно распространяемом CD-ROM Altera Digital Library. ПО системы MAX+PLUS II содержит 11 приложе-

ний и главную управляющую программу. Одни и те же команды разных приложений работают одинаково, что облегчает задачу разработки логического дизайна. В табл. 1 приведено описание приложений.

В системе MAX+PLUS II легко доступны все инструменты для создания проекта. Разработка проекта ускоряется за счёт имеющихся стандартных функций, в том числе примитивов, мегафункций, библиотеки параметризованных модулей (LPM) и макрофункций устаревшего типа микросхем серии 7400.

Все пять редакторов MAX PLUS II и три редактора создания дизайна (графический, текстовый и сигнальный) имеют общие функции. Кроме того, приложения редактора MAX PLUS II имеют следующие общие функции: создание файлов символов и файлов с прототипами функций (*Include*-файлы); поиск узлов; траверз иерархического дерева; всплывающие окна меню, зависящего от контекста; временной анализ; поиск и замена фрагментов текста; отмена последнего шага редактирования, его возвращения, вырезка, копирование, вставка и удаление выбранных фрагментов, обмен фрагментами между приложениями MAX+PLUS II или приложениями Windows; печать.

Графический редактор (**Graphic Editor**) обеспечивает проектирование в реальном формате изображения (WYSIWIG). Графические файлы проекта (*.gdf*) или схемные файлы OrCAD (*.sch*), созданные в данном графическом редакторе, могут включать любую комбинацию символов примитивов, мегафункций и макрофункций. Символы могут представлять собой любой тип файла проекта.

Символьный редактор (**Symbol Editor**) позволяет просматривать, создавать и редактировать символ. Символьный файл имеет то же имя, что и проект, и расширение "*.sym*". Команда Creat Default Symbol меню File, которая есть в графическом, текстовом и сигнальном редакторах, создает символ для любого файла проекта. Символьный редактор обладает следующими характеристиками: можно переопределять символ, представляющий файл проекта, создавать и редактировать выводы и их имена, используя входные, выходные и двунаправленные выводы, а также задавать варианты ввода символа в файл графического редактора, задавать значения параметров и их значения по умолчанию; сетка и направляющие помогают выполнять точное выравнивание объектов, в символе можно вводить комментарии.

Текстовый редактор (**Text Editor**) является инструментом для создания текстовых файлов проекта на языках описания аппаратуры: AHDL (*.tdf*), VHDL (*.vhd*), Verilog HDL (*.v*). Данный редактор имеет встроенные возможности ввода файлов проекта, их компиляции и отладки с выдачей сообщений об ошибках и их локализацией в исходном тексте или в тексте вспомогательных файлов. Кроме того, существуют шаблоны языковых конструкций для AHDL, VHDL и Verilog HDL, выполнено окрашивание синтаксических конструкций. В данном редакторе можно вручную редактировать файлы назначений и конфигурации (*.acf*), а также делать установки конфигурации для компилятора, симулятора и временного анализатора.

Пользуясь данным текстовым редактором, можно создавать тестовые векторы (*.vec*), используемые для тестирования, отладки функций и при вводе

Таблица 1

Приложение	Выполняемая функция
Hierarchy Display	**Обзор иерархии** – отображает текущую иерархичнскую структуру файлов в виде дерева с ветвями, представляющими собой подпроекты
Graphic Editor	**Графический редактор** – позволяет разрабатывать схемный логический проект в формате реального отображения на экране
Symbol Editor	**Символьный редактор** – позволяет редактировать существующие символы и создавать новые
Text Editor	**Текстовый редактор** – позволяет создавать и редактировать текстовые файлы логического дизайна, написанные на языках AHDL, VHDL, Verilog
Waveform Editor	**Сигнальный редактор** – выполняет двойную функцию: инструмент для разработки дизайна и инструмент для ввода тестовых векторов и наблюдения результатов тестирования
Floorplan Editor	**Поуровневый планировщик** – позволяет графическими средствами делать назначения контактам устройства и ресурсов логических элементов
Compiler	**Компилятор** – обрабатывает графические проекты
Simulator	**Симулятор** - позволяет тестировать логические операции и внутреннюю синхронизацию проектируемой логической цепи
Timing Analyzer	**Временной анализатор** – анализирует работу проектируемой логической цепи после того, как она была синтезирована и оптимизирована компилятором
Programmer	**Программатор** – позволяет программировать, конфигурировать, проводить верификацию и тестировать ПЛИС фирмы ALTERA
Message Processor	**Генератор сообщений** – выдает на экран сообщения об ошибках, предупреждающие и информационные сообщения

сигнального проекта. Можно также создавать командные файлы (*.cmd* — для симулятора и *.edc* — для EDIF), а также макробиблиотеки (*.lmf*).

Сигнальный редактор (**Waveform Editor**) служит инструментом создания описания проекта, ввода тестовых векторов и просмотра результатов тестирования. Пользователь может создавать сигнальные файлы проекта (*.wdf*), которые содержат временные диаграммы, описывающие логику работы проекта, а также файлы каналов тестирования (*.scf*), которые содержат входные векторы для тестирования и функциональной отладки. Разработка описания проекта в сигнальном редакторе является альтернативой его созданию в графическом или текстовом редакторе. Здесь можно графическим способом задавать комбинации входных логических уровней и требуемых выходов. Созданный таким образом файл WDF может содержать как логические входы, так и входы цифрового автомата, а также выходы комбинаторной логики, счётчиков и цифровых автоматов. Способ разработки дизайна в сигнальном редакторе лучше подходит для цепей с чётко определёнными последовательными входами и выходами, т.е. для цифровых

автоматов, счётчиков и регистров. Сигнальный редактор имеет следующие отличительные черты: можно создать или отредактировать узел, задав его тип; при разработке WDF можно задать тип логики узла, задать значения по умолчанию в логическом узле, а также имя состояния по умолчанию в узле типа цифрового автомата; для упрощения создания тестового вектора можно легко добавить в файл тестируемых каналов SCF несколько узлов или все из информационного файла симулятора (*.snf*), существующего для полностью откомпилированного проекта; можно объединять от 2 до 256 узлов для создания новой группы (шины) или разгруппировывать объединённые ранее в группу узлы. Можно также объединять группы с другими группами. Значение группы может быть отображено в двоичной, десятичной, шестнадцатеричной или восьмеричной системе счисления с преобразованием или без в код Грэя. Можно копировать, вставлять, перемещать или удалять выбранную часть ("интервал") сигнала, а также весь узел или группу.

После выполнения всех назначений и задания проекта приступают к его компиляции. Сначала компилятор извлекает информацию об иерархических связях между файлами проекта и проверяет проект на простые ошибки ввода описания проекта. Компилятор применяет разнообразные способы увеличения эффективности проекта и минимизации использования ресурсов устройства. Если проект слишком большой, чтобы быть реализованным в одном устройстве, компилятор может автоматически разбить его на части для реализации в нескольких устройствах того же самого семейства, при этом число соединений между устройствами минимизируется. В файле отчёта (*.rpt*) затем будет отражено, как проект будет реализован в одном или нескольких устройствах.

Кроме того, компилятор создает программирующие файлы, используемые программатором для программирования одного или нескольких устройств. У разработчика также есть возможность настроить обработку проекта. Например, можно задать стиль логического синтеза проекта по умолчанию и другие параметры логического синтеза в рамках всего проекта, что позволит провести логический синтез в соответствии с вашими потребностями. Кроме того, можно ввести требования по синхронизации в рамках всего проекта, точно задать разбиение большого проекта на части для реализации в нескольких устройствах и выбрать варианты параметров устройств, которые будут применены для всего проекта в целом. Загрузку готового проекта в ПЛИС или конфигурационное ПЗУ выполняют с помощью программатора (*Programmer*).

2.2. САПР фирмы Xilinx

Пожалуй, из всех производителей ПЛИС фирма Xilinx может считаться лидером по номенклатуре серий PLD и программного обеспечения (ПО). При этом новые версии ПО поддерживают старые серии PLD, позволяя разработчику производить плавную миграцию проектов на новые серии. Среди программных продуктов Xilinx имеются как относительно простые свободно распространяемые системы, так и мощные, интегрированные пакеты, позволяющие разрабатывать схемы эквивалентной ёмкости более 1000000 вентилей. Среди бесплатных САПР Xilinx следует выделить систему WebFITTER, первый в своем роде продукт, основанный на использовании Internet [7-9].

В табл. 2 приведены основные характеристики системы WebFITTER. Однако для большинства пользователей в СНГ использование данного продукта может оказаться затруднительным, поскольку, к сожалению, скоростной доступ в Internet доступен пока немногим. Тем не менее, следует обратить внимание на тенденцию применения технологий глобальных компьютерных сетей в разработке PLD.

Таблица 2

Поддерживаемая архитектура	CPLD
Способ описания проекта	VHDL, Verilog, ABEL, EDIF, TDF, XNF
Задание ограничений на проект	Определяется пользователем
Выходные данные	Отчет о временных параметрах проекта, отчет о трассировке, файл программатора в формате JEDEC, встроенная модель для моделирования в формате VHDL, Verilog, EDIF

Разумной альтернативой использованию WebFITTER является применение пакета WebPack, позволяющего работать с CPLD XC9500 и CoolPLD, ввод описания проекта возможен как с помощью схемного редактора, так и с использованием языков описания аппаратуры ABEL и VHDL. Возможно программирование устройств непосредственно в системе с использованием аппаратного загрузчика XChecker. К сожалению, в WebPack пока отсутствует опция моделирования алгоритмов, описанных с помощью VHDL, поддерживается только синтез. Данный пакет доступен на сайте фирмы Xilinx.

Для работы с FPGA фирмой Xilinx в кооперации с Aldec и Synopsys разработан мощный пакет Foundation 2.1, последняя версия которого обеспечивает ряд новых функций, позволяющих использовать PLD в качестве основной элементной базы для построения "систем на кристалле" (system-on-chip, SOC). В основе идеи SOC лежит интеграция всей электронной системы в одном кристалле (например, в случае ПК такой чип объединяет процессор, память и т. д.). Компоненты этих систем разрабатываются отдельно и хранятся в виде файлов параметризируемых модулей. Окончательная структура SOC-микросхемы выполняется на базе этих "виртуальных компонентов", называемых также "блоками интеллектуальной собственности", с помощью САПР. Благодаря стандартизации, можно объединять в одно целое "виртуальные компоненты" от разных разработчиков. Для поддержки работы над кристаллами, ёмкость которых составляет 2000000 эквивалентных вентилей, необходимо обеспечить возможность коллективной работы над проектом. Foundation 2.1 обеспечивает поддержку коллектива разработчиков как в локальной сети, так и с использованием ресурсов Internet. Данная технология разработки получила наименование Internet Team Design (ITD).Основу системы составляет оболочка Foundation Project Manager, разработанная фирмой Aldec.

Использование Project Manager позволяет обеспечить удобное задание всех параметров проекта, а также быстрое управление вводом описания проекта, его компиляцию, временное и функциональное моделирование, верификацию и программирование ПЛИС.

Пакет Foundation выпускается в различных по конфигурации модификациях. В максимальном варианте доступны приложения, описанные в табл. 3. Приведем небольшое описание основных компонентов пакета.

Синтез проекта с использованием языков описания аппаратуры высокого уровня (VHDL, Verilog). Для этих целей в состав Foundation входит система синтеза FPGA Express Synthesis, разработанная компанией Synopsys. Данный компилятор поддерживает синтез устройств с заданными временными параметрами. В качестве традиционного средства ввода используется Schematic Editor, имеющий развитые библиотеки. В версии 2.1 применяется редактор схем Vista, входящий составной частью в FPGA Express. Обеспечивается поддержка ввода описания алгоритма и синтез с использованием специализированного языка описания аппаратуры ABEL, предназначенного для описания проектов, выполняемых на PLD Xilinx и некоторых других производителей. Обеспечивается ввод описания цифрового автомата с помощью его графа переходов (State Editor). Данный способ описания проекта позволяет просто и наглядно задать поведение автомата, он весьма удобен при разработке различных устройств управления. При описании проекта с использованием языков описания аппаратуры удобно использовать специализированный редактор HDL Editor, имеющий удобные средства контроля синтаксиса, шаблоны типовых конструкций и удобную связь с компиляторами.В качестве средства работы с проектом на базе HDL используется Language Assistant, состоящий из трёх основных модулей: Language Templates, Synthesis Templates и User Templates

Для описания модулей в интерактивном графическом режиме используется средство LogiBLOX. Оно позволяет создавать такие узлы, как счетчики, сдвиговые регистры, элементы памяти и мультиплексоры. LogiBLOX запускается непосредственно из редактора HDL Editor с использованием команды Synthesis / LogiBLOX.C применением этого средства достаточно просто создать описание узла на языке описания аппаратуры, не владея им в совершенстве. Для задания параметров компиляции проекта удобно использовать Express Constraints Editor. С его помощью нетрудно задавать временные ограничения для проекта. После ввода описания проекта удобно провести его функциональное (логическое, поведенческое) моделирование с использованием симулятора Logic Simulator. В нём в интерактивном графическом режиме задаются сигналы, используемые для проведения моделирования. Результаты моделирования можно наблюдать как в привычном виде временных диаграмм, в том числе в режиме Probe, так и с использованием семисегментных индикаторов. Для компиляции проекта из Project Manager запускают модуль Design Implementation, позволяющий выбрать устройство, на котором реализуется проект, подгрузить файл ограничений и параметров синтеза, созданный пользователем, а затем запустить компиляцию проекта. В случае успешной компиляции следует провести временное моделирование с использованием модуля Timing Simulation. После проведения моделирования на компьютере можно проводить аппаратную верификацию проекта с применением загрузчика Xchecker и отладочной платы.

Таблица 3

Приложение	Выполняемая функция
Project Manager	Менеджер проектов – средство управления файлами проекта
Synthesis Constraints Editor	Редактор ограничений – накладывает временные ограничения на проект перед этапом синтеза
Synopsys FPGA Express Synthesis	Программа синтеза проектов, написанных на VHDL/Verilog
HDL Design Tools	Набор утилит для упрощения ввода проекта на языках VHDL/Verilog, включающий в себя текстовый редактор, мастер шаблонов, проверку синтаксиса, редактор диаграмм состояний
ABEL Synthesis	Модуль синтеза проектов, написанных на Abel
Schematic Editor	Графический редактор позволяющий вводить проекты в схемном виде
Simulator (Functional and Timing)	Графический редактор позволяющий произвести моделирование до трассировки и после
Design Manager	Интерфейсный модуль, осуществляющий управление всеми средствами автоматической трассировки и дающий пользователю доступ к ним
Flow Engine	Отображает и выполняет все этапы по размещению проекта в кристалл, состоящие из трансляции входного файла универсального формата во внутренний формат; разбиения логики по КЛБ; создания конфигурационного файла для загрузки в кристалл; создания необходимых отчетов
LogiBLOX	Графическое средство создания параметризованных и оптимизированных под конкретную архитектуру логических элементов
CORE Gen	Встроенный модуль генерации параметризованных и оптимизированных под конкретную архитектуру модулей, выполняющих сложные логические функции
Floorplanner	Графическое средство позволяющее контролировать процесс автоматического размещения логики в кристалл FPGA или полностью "вручную" произвести размещение
FPGA Editor	Графическое средство, позволяющее просмотреть и отредактировать результаты размещения логики и связей, а также "вручную" спроектировать кристалл FPGA на уровне КЛБ и линий связи
Hardware Debugger	Программа загрузки и верификации проекта с компьютера
PROM Files Formatter	Программа создания конфигурационного файла для хранения в последовательных или параллельных ПЗУ. Доступно три формата MCS, EXO, TEX. Для микропроцессорной загрузки

2.3. Пакет ActelDeskTop фирмы Actel

САПР ActelDeskTop представляет собой интегрированный пакет, разработанный совместными усилиями трех фирм: Actel, VeriBest и Synplicity, причем большая часть реализована именно с помощью двух последних фирм – ведущих производителей программного обеспечения. Поэтому данный продукт, свободно распространяемый на CD-ROM *Actel Digital Library*, является самым удачным решением при выборе бесплатной САПР на основе PLD[10,11]. Во многом этому способствует абсолютная поддержка языка VHDL. В чем бы не работал проектировщик, в редакторе диаграмм или в редакторе схем, он всегда может получить VHDL-интерпретацию разрабатываемого им цифрового узла. Соответственно на этом языке строятся и все процедуры моделирования и отладки проекта. Рассмотрим этот пакет более подробно. Создать новый проект можно тремя основными способами:

– с помощью редактора схем, включающего в свой состав обширнейшую библиотеку цифровых компонентов, в том числе и популярной серии 7400;

– с помощью текстового редактора, в котором можно создавать как VHDL, так и Verilog-файлы;

– с помощью символьного редактора. При этом создаются произвольные блоки, в основе которых могут лежать как разработанные ранее проекты, так и узлы, проектируемые в редакторе схем или текстовом редакторе.

В целом работа во всех трех редакторах мало чем принципиально отличается от соответствующих методов проектирования в САПР фирм Altera и Xilinx. Подобно средству LogiBLOX пакета Foundation, в среде Actel Desk Top существует так называемый ACT Macro Builder, позволяющий генерировать типовые цифровые автоматы, причем синтезированный таким образом узел может быть помещен как блок в редактор схем. Кроме того, поведенческое описание этого блока можно с успехом использовать в проектах, создаваемых непосредственным VHDL-кодированием.

На каждой стадии проектирования можно промоделировать работу спроектированного устройства. Для этого имеется универсальный симулятор, разработанный фирмой VeriBest. Это простая и очень удобная в обращении программа, позволяющая проверять поведение разрабатываемого проекта на стадии разработки, изучать временные параметры реализованной схемы после этапов синтеза проекта и размещения вентилей на кристалле. При этом используются одни и те же входные воздействия, также закодированные в VHDL-коде. При желании в эти воздействия можно добавлять дополнительные сигналы или удалять ненужные. Любой проект может содержать огромное количество таких воздействий для более полной проверки того или иного режима.

За синтез проекта, размещение элементов на кристалле, а также за задание необходимых параметров при программировании отвечает программный пакет Synplify фирмы Synplicity. Это удобное программное средство, позволяющее в интерактивной форме откомпилировать проект, произвести при необходимости его отладку (особенно эффективным на данном шаге является применение симулятора Veri Best), выбрать чип, на котором будет реализован проект, а также задать необходимые технические параметры. Произвольным образом можно назначать используемые ножки кристалла, уровни используемых в будущей системе напряжений, коэффициенты умножения частоты и многое другое. Работа с компилятором не вызывает особых проблем у проектировщика, как и весь пакет в целом.

Интересной отличительной особенностью пакета Actel Desk Top является наличие в нем программного средства Actel Silicon Explorer. Этот продукт благодаря уникальной архитектуре чипов фирмы Actel позволяет производить отладку проекта на физическом уровне. При этом к системе подключается аппаратный модуль Probe Pilot, представляющий собой 18-разрядный логический анализатор. При использовании этого аппаратно-программного комплекса имеется возможность прямо в среде проектирования изучать временные диаграммы любой из 16 точек, являющихся физическими ножками запрограммированного чипа. Более того, проектировщик может наблюдать за любыми двумя внутренними точками чипа, которые благодаря особой технологии изготовления могут подаваться на два зарезервированных с этой целью вывода. Эта уникальная возможность открывает новые горизонты в области отладки готовых проектов и диагностирования проектируемых устройств.

3. Выбор средств проектирования

В данном разделе представлен анализ пакетов САПР, необходимых разработчику цифровых устройств на базе PLD. Отметим, что проведенный анализ касается только ПО базового уровня, т.е. свободно распространяемого, или поддерживающего проекты только определенной сложности.

Основным критерием для выбора необходимого средства проектирования служит следующий. Безусловно, при проектировании на основе PLD лучше всего применять аппаратные и программные средства одной и той же фирмы-производителя. Конечно, при использовании таких САПР гарантируется поддерка семейств PLD, интересующих разработчика. Но с другой стороны, фирмы-производители микросхем обычно не являются лидерами в разработке методов проектирования. Поэтому программы синтеза и анализа фирм, не изготавливающих PLD, могут обеспечивать более качественное решение задач проектированых, а также предоставлять большее разнообразие способов ввода проекта и сервисного обслуживания. В качестве примера может служить САПР Foundation фирмы Xilinx. При ее анализе видно, что современные системы проектирования должны обладать полной поддержкой как стандартных языков описания аппаратуры (в первую очередь это VHDL и Verilog), так и языков описания аппаратуры, разработанных компаниями-производителями PLD специально для использования только в своих САПР и учитывающих архитектурные особенности конкретных семейств ПЛИС. Кроме того, современная система проектирования должна поддерживать интерфейсы ведущих фирм-производителей программного обеспечения в данной отрасли. Это позволяет проводить разработку алгоритмов, пригодных к реализации на PLD не только разных семейств, но и различных производителей, что облегчает переносимость алгоритма и ускоряет процесс разработ-

ки. Примером таких систем являются продукты серии FPGA Express фирмы Synopsys, OrCAD Express фирмы OrCAD, продукты фирм Aldec, Cadence Design Systems и других. ПО Xilinx поддерживает интерфейс со всеми из названных продуктов. САПР также должна поддерживать необходимые средства программирования PLD. Этот факт служит еще одним доводом применения САПР фирм-производителей микросхем. Ведь для успешного программирования чипов очень важно применять те методы программирования, которые сертифицированы и рекомендованы фирмой-изготовителем PLD.

Говоря о выборе необходимой САПР, следует упомянуть о такой возможности, как применение в проектах специализированных функций, оптимизированных для использования при разработке кристаллов PLD. Такие функции получили название мегафункций применительно к фирме Altera, и COREs, если речь идет о фирме Xilinx. Так как идея синтеза таких функций приблизительно одинакова для обеих фирм, рассмотрим применение мегафункции фирмы Altera. Такие функции представляют собой законченное описание самых различных компонентов при проектировании сложных цифровых устройств. Это могут быть самые различные вычислители, цифровые фильтры, компараторы, преобразователи, модули сигнальных процессоров, всевозможные PCI-компоненты и многое другое. При разработке проектов с использованием мегафункций проектировщику не надо реализовывать стандартные узлы, имеющиеся в библиотеке мегафункций (у фирмы Altera это IP MegaStore, у Xilinx – IP Center). Достаточно включить в свой проект уже готовое описание. Следовательно, время разработки сложных устройств заметно сокращается. Библиотека мегафункций является открытой благодаря программе AMPP (Altera Megafunction Partners Program) – своеобразный союз между фирмой Altera и разработчиками, созданный для поддержки и расширения использования мегафункций в проектах на PLD. Дополнительные сведения об этой программе и имеющихся мегафункциях можно получить на сайте компании.

Итак, несомненным лидером среди пакетов САПР является пакет Foundation фирмы Xilinx. Он отвечает всем современным требованиям, предъявляемым к САПР на базе PLD. Победу ему обеспечили поддержка интерфейсов ведущих фирм-производителей программного обеспечения, профессиональные средства проектирования, включенные в пакет, поддержка механизма специализированных функций, удобство интерфейса и поддерживаемые пакетом семейства PLD. Данный продукт является победителем почти по всем критериям, за исключением того, что он не распространяется бесплатно. Но и САПР такого уровня других фирм не обладают таким свойством. Конфигурация пакета MAX+PLUSII, версия 9.6 которого доступна в сети Internet каждому желающему, лишена поддержки современных семейств PLD, в ней отсутствует ввод с языков VHDL и Verilog, автоматическое разбиение проекта на несколько кристаллов и ряд других функций. Кроме того, как и в случае с пакетом Actel Desk Top, лицензия на свободное пользование этим ПО дается только на определенный период. Скорее всего эти САПР больше подходят для знакомства с технологией проектирования на основе PLD.

4. Заключение

Анализируя все сказанное выше, можно сделать следующий вывод: выбор конкретной САПР целиком зависит от фирмы-производителя того семейства чипов, на базе которых разработчик собирается проектировать необходимые ему цифровые устройства. После этого остается выяснить, проекты какого масштаба ему придется разрабатывать, и в зависимости от этого выбирать определенную конфигурацию ПО. При небольших размерах проекта (порядка 20.000 эквивалентных вентилей) можно смело использовать свободно распространяемое программное обеспечение фирм Altera и Actel. Если же проект более сложен, как впрочем и все последующие, то сначала надо определиться в выборе элементной базы, а затем уже в выборе САПР, предназначенной для ее программирования. Именно анализу новейших семейств микросхем PLD, а также их особенностям будут посвящены следующие статьи данного обзора.

Литература: 1. *Соловьев В.В., Васильев А.Г.* Программируемые логические интегральные схемы и их применение. Мн.: Беларуская навука, 1998. 266 с. 2. *Стешенко В.* Школа разработки аппаратуры цифровой обработки сигналов на ПЛИС. // Chip News. 1999. N9. 3. *Криста Дейв, Джонсон Тони.* Методология высокоуровневого проектирования устройств на базе FPGA // Electronic Design, июнь 1999 г. 4. Data Book. Altera, 1999. 5. http://www.altera.com/ 6. http://www.altera.ru/ 7. The Programmable Logic Data Book. Xilinx, 1999. 8. http://www.xilinx.com/ 9.http://www.xilinx.ru/ 10. Programmable Logic Data Book and Design Guide. Actel, 1999. 11. http://www.actel.com/

Поступила в редколлегию 18.03.2000

Рецензент: д-р техн. наук Кривуля Г.Ф.

Рустинов Владимир Алексеевич, канд. техн. наук, зам. директора ХАЭР. Научные интересы: автоматизация проектирования и диагностика вычислительных систем. Увлечения: теннис, горные лыжи, плавание. Адрес: Украина, 61057, Харьков, пер. Театральный, 11/13, тел. 43-11-83.

Хаханова Ирина Витальевна, канд. техн. наук, доцент кафедры автоматизации проектирования вычислительной техники ХТУРЭ. Научные интересы: техническая диагностика вычислительных устройств. Увлечения: книги, английский язык. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Барабаш Александр Алексеевич, студент ХТУРЭ. Научные интересы: разработка цифровых устройств на базе новых методов проектирования. Увлечения: шахматы. Адрес: Украина, 61057, Харьков, пер. Театральный, 11/13, тел. 43-11-83.

Герасимов Максим Александрович, студент ХТУРЭ. Научные интересы: разработка цифровых устройств на базе микросхем программируемой логики. Увлечения: программирование. Адрес: Украина, 61057, Харьков, пер. Театральный, 11/13, тел. 43-11-83.