УДК 004.453

АНАЛИЗ СТРУКТУРЫ ИСХОДНЫХ ФАЙЛОВ ПРОЕКТА ДЛЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ

Г. Н. Мальцев[a]***,*** *доктор техн. наук, профессор*
А. В. Панкратов[a]***,*** *канд. техн. наук, докторант*
А. А. Макунин[a]***,*** *адъюнкт*
[a]*Военно-космическая академия им. А. Ф. Можайского, Санкт-Петербург, РФ*

Постановка проблемы: *технология автоматизированного проектирования «систем на кристалле» на базе программируемых логических интегральных схем включает всестороннее тестирование проекта. Одним из методических инструментов тестирования является инженерный анализ, позволяющий осуществлять обратное проектирование микроэлектронных изделий на основе изучения их текущего состояния и находить программные ошибки проекта. Для эффективного решения задач инженерного анализа систем на базе программируемых микросхем авторы предлагают методику анализа структуры исходных файлов проекта, позволяющую распознавать блоки программы или группы элементов схемы.* ***Методы:*** *анализ структуры исходных файлов проекта программируемой логической интегральной микросхемы на основе перехода от низкоуровневого описания к описанию регистровых передач.* ***Результаты:*** *на примере проекта для программируемой логической интегральной микросхемы выявлено, что фреймовая структура с фиксированным размером слов в файлах форматов BIT и RBT позволяет восстанавливать логику работы системы на уровне регистровых передач с последующим восстановлением файлов форматов NCD и XDL. Предложено представление файлов проекта в удобном для определения структурных блоков и выявления внутренних связей проекта формате, представляющем из себя разбитые на фреймы именованные области с комментариями.* ***Практическая значимость:*** *полученные результаты позволяют сократить время отладки «систем на кристалле» за счет выявления особенностей конфигурации, проявляющихся в процессе компиляции проекта в генерируемых системой автоматизированного проектирования файлах.*

Ключевые слова *— программируемые логические интегральные микросхемы, исходные файлы проекта, конфигурационный вектор, обратное проектирование.*

Введение

В настоящее время при создании информационно-управляющих систем различного назначения широко используются программируемые логические интегральные схемы (ПЛИС) [1, 2]. На современном этапе ведущей тенденцией развития цифровой техники является применение программируемых микроэлектронных компонентов для реализации не только отдельных блоков, но и проектируемых устройств в целом, вплоть до создания так называемых «систем на кристалле». Наряду с прямыми задачами проектирования таких систем в ряде случаев, например, при анализе программных ошибок или при перепроектировании, решаются задачи обратного проектирования микроэлектронных изделий на ПЛИС.

Обратное проектирование является результатом инженерного анализа [3]. Процесс обратного проектирования (reverse engineering) принято рассматривать как инверсию процесса прямого проектирования — построение по готовому изделию функционального представления высокого уровня, призванное облегчить понимание работы устройства [4]. При инженерном анализе цифровых систем на ПЛИС необходимо оперативно распознавать их архитектуру и принципы работы, и особую ценность представляют сведения о принципах функционирования ПЛИС в данном конкретном изделии. Эту информацию можно получить из исходных файлов проекта (конфигурационных файлов, файлов регистровых передач и др.). В настоящей статье на примере анализа ПЛИС типа *FPGA* (*Field Programmable Gate Array*), основанных на архитектуре матриц логических элементов, представлено описание структуры исходных файлов проекта для обеспечения процесса обратного проектирования «систем на кристалле».

Общая характеристика процесса обратного проектирования ПЛИС

В современных условиях процесс обратного проектирования микроэлектронных изделий является существенной частью создания конкурентоспособной продукции и обычно направлен на разработку устройств более эффективных, чем имеющиеся у конкурентов. Обратное проектирование позволяет выявить программные ошибки устройств, полные спецификации на которые отсутствуют. Дополнительная область применения обратного проектирования — перепроектирование на современной элементной базе устаревших компонентов, находящихся в составе долговечного оборудования сложных технических систем. Как показывает анализ, в полупроводниковой промышленности заказчиками перепроектирования являются те, кто интересуется либо техни-

ческой информацией об устройстве, либо компонентами устройства, защищенными патентом [4].

Главные особенности обратного проектирования в сравнении с прямым проектированием состоят в следующем. Отличается, во-первых, порядок следования процессов, составляющих прямое и обратное проектирование, во-вторых, возможности и условия достижения результата. Прямое проектирование представляет собой процесс превращения спецификации в продукт, удовлетворяющий этой спецификации. Между спецификацией и продуктом находятся процессы разработки и изготовления. Соответствующая организация и корректная постановка задач создают возможность реализации всех процессов прямого проектирования и условия получения ожидаемого результата. Обратное проектирование является инверсией прямого проектирования в смысле порядка следования и направленности составляющих его процессов. Задача заключается в построении спецификации на основании анализа продукта. При этом результат обратного проектирования не гарантирован, что является следствием решения в процессе обратного проектирования обратных задач, в общем случае не имеющих решения.

Процесс обратного проектирования включает следующие основные стадии: анализ продукта, извлечение описания продукта промежуточного уровня, анализ описания продукта для построения новой спецификации. По построенной спецификации можно разработать новый продукт, обеспечивающий эффективную реализацию тех или иных функций в другом технологическом базисе [4, 5]. Разработка новой спецификации при перепроектировании ПЛИС может быть автоматизирована, если возможно использовать промежуточные описания программных продуктов, созданных при прямой разработке.

В настоящее время применение современных информационных технологий существенно расширяет возможности инженерного анализа при обратном проектировании. Это обусловлено следующими факторами. Во-первых, методики проектирования становятся более формализованными, все большая часть работы по проектированию реализуется средствами автоматизации. Возникает возможность распознавания блоков программы или группы элементов схемы на основании знания преобразований, производимых компьютерной программой системы автоматизированного проектирования (САПР), одной и той же в разных компаниях-изготовителях. Внутреннюю структуру продукта в этом случае легче исследовать и интерпретировать, чем в случае, когда продукт является результатом проектирования конкретного разработчика, использующего не в полной мере формализованные методики проектирования. Во-вторых, методики искусственного интеллекта для распознавания образов в исследовании и интерпретации достигли уровня, на котором распознавание структуры в продукте можно выполнить автоматически [6].

Существенной особенностью ПЛИС является то, что алгоритм их функционирования задается конфигурирующей битовой последовательностью, являющейся результатом работы САПР. Поэтому для выявления особенностей работы систем на ПЛИС в процессе их обратного проектирования наряду с анализом выходных электрических сигналов используется анализ конфигурирующей битовой последовательности и промежуточных исходных файлов проекта. Следует отметить, что тестирование ПЛИС по их выходным сигналам является достаточно трудоемким, а вероятность получения правильного результата при анализе выходных сигналов достаточно низкая, и не всегда с результатами такого тестирования удается сопоставить весь функционал работы ПЛИС [5]. В то же время анализ исходных файлов проекта предоставляет широкие возможности обратного проектирования ПЛИС и является наиболее приемлемым способом их инженерного анализа.

Практический интерес представляет инженерный анализ получающих широкое использование в цифровых системах различного назначения ПЛИС типа FPGA, основанных на архитектуре матриц логических элементов, разработки ведущих мировых производителей.

К числу ведущих производителей ПЛИС относится фирма Xilinx, которая предоставляет разработчикам широкий спектр кристаллов с различной технологией производства, степенью интеграции, архитектурой, быстродействием, потребляемой мощностью и напряжением питания. Фирма выпускает линейку ПЛИС в различных типах корпусов и в нескольких вариантах исполнения, включая промышленное, военное и радиационно-стойкое [2, 7]. Кристаллы, выпускаемые фирмой Xilinx, в полной мере реализуют преимущества современных ПЛИС по сравнению с микроэлектронными устройствами с жесткой логикой. К числу достоинств ПЛИС фирмы Xilinx относятся высокие быстродействие и степень интеграции, возможность перепрограммирования непосредственно в системе, наличие мощных инструментов САПР, позволяющих устранить возможные ошибки в процессе проектирования устройства, сравнительно низкая стоимость (в пересчете на один логический вентиль).

При разработке проекта исходных файлов проекта ПЛИС анализируемого типа пишется исходный код на языке *HDL* (*Hardware Description Language*) [6]. Основной интерес при обратном проектировании таких ПЛИС представляют файлы типов *NCD*, *BIT*, *XDL*, *RBT* (названия типов

файлов соответствуют их расширениям). Файл с расширением **.ncd* является файлом регистровых передач, который создается САПР по завершении написания программы, если не обнаружены программные ошибки. Этот файл после компиляции преобразуется в файл с расширением **.bit*, который впоследствии с помощью программатора записывается в ПЛИС. Одновременно создаются файлы их текстового отображения с расширениями **.xdl* и **.rbt*: для файла *NCD* — файл *XDL*, для файла *BIT* — файл *RBT*. В настоящей статье представлено описание структуры этих файлов, которое может быть использовано при обратном проектировании ПЛИС фирмы Xilinx.

Файлы форматов NCD и XDL

Файлы формата *NCD*, называемые также файлами регистровых передач (ФРП), несут информацию о том, какие связи и ячейки в ПЛИС должны быть активированы для выполнения функции, предусмотренной проектом. Данные файлы получаются путем компиляции из исходного кода проекта, написанного на языке программирования *HDL*. Из файлов формата *NCD* с помощью САПР получают файлы формата *BIT* — конфигурационные файлы.

Процесс получения ФРП представлен на рис. 1, где обозначены:

.vhdl — источник кода на языке описания аппаратуры *VHDL* (*Very high speed integrated circuits Hardware Description Language*);

.v — источник кода на языке описания аппаратуры *Verilog*;

.syr (*sythesiser report*) — подробный отчет синтезатора, который будет сгенерирован по окончании процесса синтеза. Часть отчета попала в вывод консоли;

.ngd — список соединений для интегрированной программной среды *ISE* (*Integrated Software Environment*) версии до 6.1i;

.ngc — список соединений для интегрированной программной среды *ISE* версии старше 6.1i;

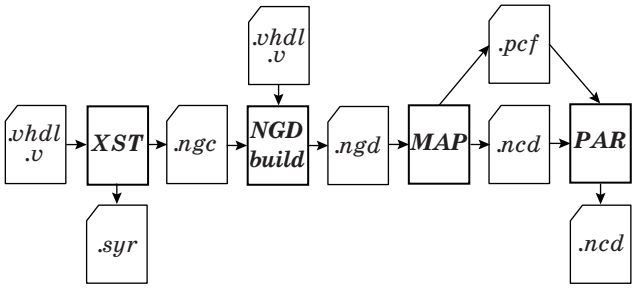
.ncd — файл регистровых передач;

.pcf — список ограничений проекта.

На рис. 1 также выделены следующие программные модули (утилиты) САПР:

XST — утилита-синтезатор (*Xilinx Synthesis Technology*). Преобразует исходное описание проекта в так называемый список связей (*NetList*). Эти связи устанавливаются между специфичными для архитектуры программного обеспечения ПЛИС примитивами (элементарными объектами);

NGD build — утилита-транслятор. Во время трансляции переводит файл списка цепей *NGC* (*Netlist file with Constraint information*) в файл списка данных *NGD* (*Native Generic Database*). Файлов *NGC* может быть несколько, при этом они



■ ***Рис. 1***. Процесс получения ФРП

все будут объединены в один файл. Например, это могут быть уже синтезированные файлы коммерческих компонентов, код которых закрыт;

MAP — утилита отображения. «Упаковывает» примитивы, полученные в результате трансляции в стандартные последовательности (слайсы), а также генерирует файл физических ограничений *PCF* (*Physical Constraints File*). В результате отображения образуется файл *NCD* (*Native Circuit Description*), в этом файле учитываются ресурсы конкретной модели ПЛИС;

PAR — средство размещения и трассировки (*Place and Route*). Управляет размещением слайсов, сформированных на предыдущем этапе в матрице логических блоков (*Place*). После этого выполняется трассировка (распределение) связей между сигналами слайсов (*Route*).

Структурно ФРП формата *NCD* состоят из заголовка и основной части файла. В заголовке содержатся маркеры структуры функциональных ячеек и связей, а также контрольная информация для проверки целостности и настройки синтезатора конфигурационного файла. Эта служебная информация является исходной для анализа основной части файла.

Заголовок ФРП включает в себя следующие элементы:

1) маркер использования стандарта построения ячеек (*XDB — Xilinx Design Build*);

2) маркер использования стандарта построения связей между ячейками (*XDM — Xilinx Design Manager*);

3) количество байт оставшейся части файла после этого блока (*Start Control*). Служит для контроля целостности файла;

4) версию программы *NGDBUILD.EXE*, построившей данный ФРП (*Version XILINX*);

5) количество байт оставшейся части файла после этого блока (*End Control*). Служит для контроля целостности файла;

6) размер основной части ФРП (*Main Size*).

Основная часть ФРП состоит из блоков, которые могут выполнять одну из следующих функций: быть адресом ячейки ПЛИС; быть адресом связи между этими ячейками; быть определением

того, что будет выполняться в ячейке; быть определением типа связи между ячейками. Как правило, все эти возможные функции ФРП следуют друг за другом для определенной ячейки.

Для формализованного представления структуры файлов формата *NCD* опишем ее с помощью множеств и их элементов. Введем следующие обозначения: множество **X** — заголовок файла, множество **Y** — основная часть файла. Эти множества содержат непересекающиеся подмножества в качестве информационных единиц — элементов файла.

С учетом введенных обозначений выразим множества **X** и **Y** через их элементы:

$$\mathbf{X} = \{x_0, x_1, ..., x_n\},\ n \in N;$$

$$\mathbf{Y} = \{y_0, y_1, ..., y_m\},\ m \in N,$$

где число символов заголовка n и число символов основной части файла m в общем случае произвольное и для современных ПЛИС достигает значений $N = 10^5 - 10^6$.

Элементы множеств **X** представляются в шестнадцатеричном коде:

$\{x_0, ..., x_{28}\} - XDB$, $\{x_0, ..., x_{28}\} = \text{const}$ (*XILINX-XDB* 0.1 *STUB* 0.1 *ASCII*) — показывает версии стандарта построения ячеек *XDB* и преобразователя в кодировку *ASCII*);

$\{x_{29}\} = \{0A\}$ — константа, зарезервировано;

$\{x_{30}, ..., x_{44}\} - XDM$, $\{x_{30}, ..., x_{44}\} = \text{const}$ (*XILINX-XDM V*1.6) — показывает версию стандарта построения связей между ячейками *XDM*;

$\{x_{45}\} = \{0A\}$ — константа, зарезервировано;

$\{x_{46}, x_{47}, x_{48}\} = \{0x23, 0x23, 0x23\}$ — назначение не выяснено;

$\{x_{49}, ..., x_{52}\}$ — начальная контрольная сумма;

$\{x_{53}\} = \{3A\}$ — константа, зарезервировано;

$\{x_{54}, ..., x_{61}\} = \{0x58, 0x6C, 0x78, 0x56, 0x33, 0x32, 0x44, 0x4D\}$ — назначение не выяснено;

$\{x_{62}, ..., x_{65}\} = \{0x20, 0x20, 0x20, 0x20\}$;

$\{x_{69}, ..., x_{72}\} = \{0x20, 0x20, 0x20, 0x20\}$.

Элементы множеств **Y** представляются в двоичном коде:

$\{y_0, ..., y_{15}\}$ — размер основной части файла;

$\{y_{16}, ..., y_{31}\} = \{0b0110010101001110\}$ — назначение не выяснено;

$\{y_i, ..., y_{i+7}\} = \{0b00101111\}$ — разделитель между блоками.

В процессе обратного проектирования файлам данного типа принадлежит очень важное место. Зная структуру этих файлов, можно считать, что задача распознавания принципа работы данной ПЛИС решена. Распознавание файлов формата *NCD* позволяет выявить внутренние связи между блоками ПЛИС и тип связей, что дает представление о том, как работает данная ПЛИС и какие функции она выполняет. Зная ФРП анализируемого типа ПЛИС, можно последовательно шаг за шагом восстановить все исходные файлы проекта, созданного для этой ПЛИС.

Полное внутреннее состояние конфигурации памяти программируемых логических устройств *FPGA* содержится в программном файле, называемом *bitstream*. Это состояние при функционировании *FPGA* обычно хранится в памяти типа *SRAM* и загружается из энергонезависимого устройства памяти при каждом включении питания *FPGA*. В мире программного обеспечения изготовителями процессоров общепринято публиковать формат потока двоичных сигналов их продукта в руководстве архитектуры для использования при разработке компиляторов. В мире программирования *FPGA* ситуация противоположная. В настоящее время ни один изготовитель *FPGA* не опубликовал формат файлов *bitstream* своих устройств. Более того, чтобы защитить авторские права разработчиков при передаче устройства заказчику, файл программирования *FPGA* шифруется. Такой кодированный программный файл *bitstream* и загружается из энергонезависимого устройства памяти при включении питания *FPGA* [6].

Задача декомпиляции файла *bitstream*, извлеченного из *SRAM* в готовом устройстве, практически безнадежна. Но если перепроектирование выполняется организацией-разработчиком оригинального устройства на *FPGA*, например, с целью использовать более дешевую в массовом производстве технологию изготовления, то задача декомпиляции может быть решена. Задача декомпиляции файла *bitstream* возникает и при оригинальном проектировании, например, САПР *Xilinx ISE* поддерживает методику проектирования с использованием *hardwaremacro* (предварительно размещенные блоки в матрице элементов *FPGA* вместе с внутренними межсоединениями). Эта же САПР обеспечивает проектировщику возможность преобразования блока, скомпилированного до уровня *bitstream*, в сеть элементов *FPGA*. Таким образом, декомпиляция файла *bitstream FPGA* становится возможной, но только с использованием инструментов, составляющих собственность изготовителя *FPGA*.

Система автоматизированного проектирования *Xilinx ISE* обеспечивает декомпиляцию *FPGA* с помощью файла формата *XDL*. Формат *XDL* (*Xilinx Design Language*) — это текстовое представление структуры запрограммированной *FPGA*: компиляция и декомпиляция осуществляются программой *xdl.exe*, имеющейся в последних версиях САПР *Xilinx ISE*. Эта программа имеет опции, позволяющие построить файлы отчета (с расширением *.xdlrc*), которые содержат описательную информацию о запрограммированной *FPGA* фирмы Xilinx. Файлы отчета имеют формат, отличный от формата *XDL* (поскольку они описывают *FPGA*, а не проектируемое устрой-

ство), и содержат огромный объем информации (несколько гигабайт текста), описывающей всю сетку элементов запрограммированной *FPGA*.

Формат *XDL* фирмой Xilinx официально не опубликован. Краткое описание формата *XDL* приводится в руководстве пользователя [8]. Данное текстовое описание вполне доступно для понимания. Оно включает комментарии, которые помогают пользователю в интерпретации файла. Информация, содержащаяся в файле формата *XDL*, полностью описывает настройку конкретной *FPGA* для выполнения функций проектируемого устройства [7]. Для понимания функционирования устройства по описанию в формате *XDL* необходимы функциональные описания элементов и коммутационных блоков конкретного типа *FPGA*. Структурные описания этих элементов и непрограммируемые соединения можно найти в файле отчета.

Файл формата *XDL* может содержать два типа конструкций — блоки (*instances*) и цепи (*nets*). Блок может быть любым логическим элементом в *FPGA*, например, логическим элементом сетки *FPGA* (*slice*), блоком оперативного запоминающего устройства или процессором цифровой обработки сигналов (*DSP block*). Описание цепи содержит название цепи и компонентов блоков, которые связаны этой цепью.

Каждый элемент сетки *FPGA* имеет описание в формате *XDL*. Настройки компонентов внутри описаний элементов везде имеют один и тот же вид: *name::#parameter*. Названия (*name*) компонентов настройки приведены в описании *FPGA*. Значение параметра (*parameter*), отличное от *#OFF*, указывает, что соответствующий компонент используется. Для извлечения структуры устройства, пригодной для перепроектирования *FPGA* в другом технологическом базисе, требуется разработка анализаторов файлов формата *XDL* и файла отчета. Результат работы анализатора файлов формата *XDL* должен быть представлен в виде структурного описания, которое может служить исходными данными для нового проектирования [8].

Знание структуры файлов формата *XDL* позволит получить те же результаты, что и при распознавании файлов формата *NCD*, так как эти файлы являются взаимополучаемыми с помощью стандартных инструментов САПР. Отличие между ними заключается в том, что файлы формата *XDL* непосредственно воспринимаются человеком, а файлы формата *NCD* являются формализованными и непосредственно человеком не воспринимаются.

Файлы формата BIT и RBT

Файлы формата *BIT*, также называемые конфигурационными файлами (КФ), несут информацию, необходимую для непосредственного конфигурирования ПЛИС [9]. КФ получают с помощью САПР в результате генерации из ФРП. После генерации файлы формата *BIT* с помощью программатора записываются в ПЛИС, в результате чего ПЛИС может выполнять свои функции согласно своему предназначению.

Схема преобразований при трансляции ФРП в КФ представлена на рис. 2. Очередность выполнения преобразований указана номерами над стрелками. На рисунке введены следующие обозначения:

bitgen — модуль САПР, синтезирующий конфигурационный файл для ПЛИС;

BS_BITFILE — объект программного модуля, соответствующий конфигурационному файлу;

BS_FDCHIP — объект программного модуля, соответствующий ограничениям проекта;

BS_DB — объект программного модуля, соответствующий модели ПЛИС;

TMAPS — таблица итераций;

NC_DESIGN — объект программного модуля, соответствующий файлу регистровых передач.

Структурно КФ формата *BIT* состоят из заголовка и основной части файла. Заголовок файла необходим для правильной работы загрузчика ПЛИС, в нем содержится информация о микросхеме, для которой сгенерирован файл, информация для контроля целостности данных и сведения об источнике. Эта служебная информация является исходной для анализа основной части файла.

Заголовок КФ включает в себя следующие элементы:

1) маркер начала файла (*Start ID*);

2) название ФРП, из которого был сгенерирован КФ (*Name *.ncd*) и идентификационный номер пользователя (*User ID*), сгенерировавшего КФ;

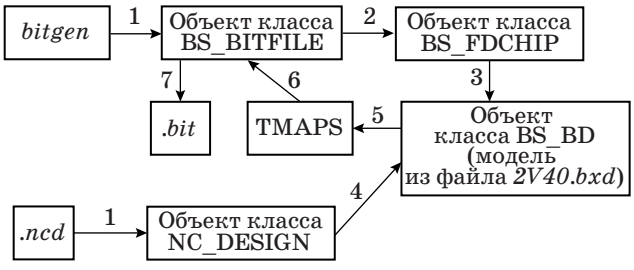
3) структуру ПЛИС, для которой был создан КФ (*Device*);

4) дату создания (*Create date*);

5) время создания (*Create time*);

6) размер основной части КФ (*Main Size*).

Для формализованного представления структуры файлов формата *BIT* опишем ее с помощью множеств и их элементов. Введем следующие



■ ***Рис. 2.*** Процесс получения КФ из ФРП

обозначения: множество **S** — заголовок файла, множество **P** — основная часть файла. Эти множества содержат определенное число элементов, каждый из которых выражает определенную информацию.

С учетом введенных обозначений выразим множества **S** и **P** через их элементы:

$$\mathbf{S} = \{s_0, s_1, ..., s_n\},\ n \in N;$$

$$\mathbf{P} = \{p_0, p_1, ..., p_m\},\ m \in N,$$

где число символов заголовка n и число символов основной части файла m в общем случае произвольное и для современных ПЛИС достигает значений $N = 10^3 - 10^4$.

Элементы множеств **S** и **P** представляются в шестнадцатеричном виде:

$\{s_0 - s_{12}\}$ — *StartID*, $\{s_0, s_1, ..., s_{12}\}$ = {0x00, 0x09, 0x0*F*, 0x*F*0, 0x0*F*, 0x*F*0, 0x0*F*, 0x*F*0, 0x0*F*, 0x*F*0, 0x00, 0x00, 0x01};

$\{s_{13}\}$ = {0x61}, $\{s_{13}\}$ — начало *Name* *.*ncd*;

$\{s_{14}\}$ = {0x00} — константа, зарезервировано;

$\{s_{15}\}$ — количество байт, отображающих имя ФРП (+*UserID*);

$\{s_{16}, ..., s_k\}$ — имя ФРП, из которого сгенерирован КФ, $k = 20, ..., 271$, $k \in N$;

$\{s_{k+1}\}$ = {3*B*} — константа, зарезервировано;

$\{s_{k+2}, ..., s_{k+18}\}$ — *UserID* (может отсутствовать);

$\{s_{k+2}, ..., s_{k+10}\}$ = {55, 73, 65, 72, 49, 44, 3*D*, 30, 78};

$\{s_{k+11}, ..., s_{k+18}\}$ = {46, 46, 46, 46, 46, 46, 46, 46} (по умолчанию);

$\{s_t\}$ = {00}, $t = (k+2)$, если *UserID* отсутствует; $t=(k+19)$, если *UserID* присутствует, $t \in N$;

$\{s_{t+1}\}$ — начало блока «Семейство ПЛИС, для которой был создан КФ (*Device*)», $\{s_{t+1}\}$ = {62};

$\{s_{t+2}\}$= {00} — константа, зарезервировано;

$\{s_{t+3}\}$ — размер блока;

$\{s_{t+4}, ..., s_b\}$ — семейство ПЛИС, для которой был создан КФ (*Device*), $b = (t + 11), ..., (t + 35)$, $b \in N$;

$\{s_{b-2}, s_{b-1}, s_b\}$ — количество ножек (*pin*) у данной ПЛИС;

$\{s_{b+1}\}$ = {00} — константа, зарезервировано;

$\{s_{b+2}\}$ — начало блока «Дата создания (*Create date*)», $\{s_{b+2}\}$ = {63};

$\{s_{b+3}\}$= {00} — константа, зарезервировано;

$\{s_{b+4}\}$ — размер блока;

$\{s_{b+5}, ..., s_{b+14}\}$ — дата создания КФ (гггг/мм/дд);

$\{s_{b+15}\}$ = {00} — константа, зарезервировано;

$\{s_{b+16}\}$ — начало блока «Время создания (*Create time*)», $\{s_{b+16}\}$ = {64};

$\{s_{b+17}\}$= {00} — константа, зарезервировано;

$\{s_{b+18}\}$ — размер блока;

$\{s_{b+19}, ..., s_{b+26}\}$ — время создания КФ (чч/мм/сс);

$\{s_{b+27}\}$ = {00} — константа, зарезервировано;

$\{s_{b+28}\}$ — начало блока «Размер основной части КФ (*Main Size*)», $\{s_{b+28}\}$ = {65};

$\{s_{b+29}\}$ — размер блока;

$\{s_{b+30}, s_{b+31}\}$ — размер основной части КФ;

$\{p_0, ..., p_f\}$ — блок начала основной части, $f = 0$ (mod8);

$\{p_f, ..., p_m\}$ — конфигурация соединений между программируемыми логическими блоками.

Последовательность КФ представляет собой поток 32-разрядных слов, которые являются или командами, или конфигурируемыми данными. Основная часть КФ состоит из так называемых фреймов [8]. Фрейм данных конфигурации — это последовательность бит данных конфигурации определенной длины. Для каждой микросхемы фирмы Xilinx длина фрейма различна. Если длина бит конфигурации фрейма не кратна 32-разрядному слову, то фрейм дополняется нулевыми битами до полного 32-разрядного слова [10]. Такие фреймы, собственно, и содержат информацию о конфигурации программируемых элементов архитектуры ПЛИС.

Текстовое представление файла формата *BIT* содержится в файле формата *RBT* аналогично тому, как текстовое представление файла формата *NCD* содержится в файле формата *XDL* [11]. Файлы форматов *BIT* и *RBT* используются на первом шаге обратного проектирования как исходный материал для получения файлов *NCD* и *XDL*. Следует отметить, что данное преобразование не является тривиальным и представляется самостоятельным объектом исследований.

Заключение

Проведенный анализ структуры исходных файлов проекта ПЛИС ориентирован на обеспечение процесса обратного проектирования «систем на кристалле» по файлам конфигурации. Систематизированы и формализованы структуры исходных файлов проекта, которые используются для конфигурирования ПЛИС фирмы Xilinx. Рассмотрены схемы получения этих файлов и их место в общем процессе конфигурирования ПЛИС.

По результатам проведенного анализа можно сделать вывод, что главную роль в процессе конфигурирования ПЛИС играют файлы форматов *NCD* и *BIT*, получаемые в ходе многопроходного компилирования высокоуровневых структур описания на матрицу кристалла. Описание связей между этими файлами и разработка математической модели получения ФРП из КФ является основой инженерного анализа микроэлектронных устройств, использующих программируемую логику.

Литература

1. **Угрюмов Е. П.** Цифровая схемотехника. — СПб.: БХВ-Петербург, 2000. — 528 с.
2. **Зотов В. Ю.** Проектирование встраиваемых микропроцессорных систем на основе ПЛИС фирмы Xilinx в САПР WebPACK ISE. — М.: Горячая линия-Телеком, 2006. — 520 с.
3. **Максфилд К.** Проектирование на ПЛИС. Курс молодого бойца. — М.: Додека-XXI, 2007. — 408 с.
4. **Beckhoff C., Koch D., Torresen J.** Go Ahead: A Partial Reconfiguration Framework // 20th Annual IEEE Int. Symp. on Field-Programmable Custom Comp. Machines, Toronto, Canada, April 29 — May 1, 2012. P. 27–44.
5. **Бибило П. Н.** Системы моделирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum. — М.: СОЛОН Пресс, 2005. — 384 с.
6. **Сергиенко А. М.** VHDL для проектирования вычислительных устройств. — Киев: ЧП «Корнейчук»; ООО «ТИД ДС», 2003. — 208 с.
7. **Тарасов И. Е.** Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL. — М.: Горячая линия-Телеком, 2005. — 252 с.
8. **Суворова Е. А., Шейнин Ю. Е.** Проектирование цифровых схем на VHDL. — СПб.: БХВ-Петербург, 2003. — 576 с.
9. **Бритов Г. С., Мироновский Л. А.** Автоматизированное проектирование устройств функционального диагностирования // Информационно-управляющие системы. 2010. № 2(45). С. 55–60.
10. **Окунев К. Е., Ключарев А. А.** Программная модель системы на кристалле // Информационно-управляющие системы. 2010. № 3(46). С. 44–50.
11. **Кузелин М. О., Кнышев Д. А., Зотов В. Ю.** Современные семейства ПЛИС фирмы Xilinx: справ. пособие. — М.: Горячая линия-Телеком, 2004. — 440 с.

UDC 004.453

Analysis of Project Source Files for Programmable Logic Integrated Circuits

G. N. Maltsev[a], Dr. Sc., Tech., Professor, georgy_maltsev@mail.ru
A. V. Pankratov[a], PhD, Tech., pankratov-av@rambler.ru
A. A. Makunin[a], Post-Graduate Student, lexmak1986@mail.ru
[a]A. F. Mozhayskii Military Space Academy, 13, Zhdanovskaia St., 197082, Saint-Petersburg, Russian Federation

Purpose: "System on a chip" CAD technology on the base of programmable logic integrated circuits includes extensive testing of the project. One of the methodological tools of testing is reverse engineering of microelectronic products when you study a completed product and find its software defects. To make the reverse engineering more efficient, the authors propose a technique for analyzing the structure of the project source files, helping to recognize software blocks or groups of circuit elements. **Method:** The structure of the project source files is analyzed, going from low-level description up to register transfer description. **Results:** By the example of an FPGA project, it was shown that a frame structure with a fixed size of words in files of BIT and RBT formats allows you to recover the logic of the system at the register transfer level, with the subsequent restoration of NCD and XDL files. It was suggested to present the project files in a format of named areas with the comments split into frames. This format is convinient for defining structural blocks and identifying internal connections. **Practical relevance:** The obtained results allow you to reduce the debugging time for "systems on chip" by identifying their configuration features which manifest themselves during the project compilation to CAD-generated files.

Keywords — Programmable Logic Integrated Circuits, Configuration Vector, Reverse Engineering.

References

1. Ugriumov E. P. *Tsifrovaia skhemotekhnika* [Digital Circuitry]. Saint-Petersburg, BHV-Peterburg Publ., 2000. 528 p. (In Russian).
2. Zotov V. Y. *Proektirovanie vstraivaemykh mikroprotsessornykh sistem na osnove PLIS firmy Xilinx v SAPR WebPACK ISE* [Design of Embedded Microprocessor Systems Based on FPGA from Xilinx CAD WebPACK ISE]. Moscow, Goriachaia liniia-Telekom Publ., 2006. 520 p. (In Russian).
3. Maxfield K. The Design Warrior's Guide to FPGA's. Moscow, Dodeka-XXI Publ., 2007. 408 p. (In Russian).
4. Beckhoff C., Koch D., Torresen J. Go Ahead: A Partial Reconfiguration Framework. *20th Annual IEEE Int. Symp. on Field-Programmable Custom Comp. Machines*, Toronto, Canada, April 29 — May 1, 2012, pp. 27–44.
5. Bibilo P. N. *Cistemy proektirovaniia integral'nykh skhem na osnove iazyka VHDL. StateCAD, ModelSim, LeonardoSpectrum* [System Simulation of Integrated Circuits Based on the Language VHDL. StateCAD, ModelSim, LeonardoSpectrum]. Moscow, SOLON Press Publ., 2005. 384 p. (In Russian).
6. Sergienko A. M. *VHDL dlia proektirovaniia vychislitel'nykh ustroistv* [VHDL for the Design of Computing Devices]. Kiev, Korneichuk Publ., TID DS Publ., 2003. 208 p. (In Russian).
7. Tarasov I. E. *Razrabotka tsifrovykh ustroistv na osnove PLIS Xilinx s primeneniem iazyka VHDL* [Development of Digital Devices Based on Xilinx FPGAs Using Language VHDL]. Moscow, Goriachaia liniia-Telekom Publ., 2005. 252 p. (In Russian).
8. Suvorova E. A., Sheinin Iu. E. *Proektirovanie tsifrovykh skhem na VHDL* [Design of Digital Circuits on VHDL]. Saint-Petersburg, BHV-Peterburg Publ., 2003. 576 p. (In Russian).
9. Britov G. S., Mironovsky L. A. Automated Design of Devices for Functional Diagnosis. *Informatsionno-upravliaiushchie sistemy*, 2010, no. 2, pp. 55–60 (In Russian).
10. Okunev K. E., Kliucharev A. A. The Programming Model of the System-on-a-Chip. *Informatsionno-upravliaiushchie sistemy*, 2010, no. 3, pp. 44–50 (In Russian).
11. Kuzelin M. O. *Sovremennye semeistva PLIS firmy Xilinx* [Modern FPGA Families from Xilinx]. Moscow, Goriachaia liniia-Telekom Publ., 2004. 440 p. (In Russian).