УДК 004.424.2

А.В. Григорьев, Д.А. Грищенко
Донецкий национальный технический университет
grigorie@r5.dgtu.donetsk.ua, darya.grischenko@gmail.com

Анализ средств автоматизации построения VHDL-программ

В статье выполнен анализ уровня автоматической генерации VHDL-программ, определены достоинства и недостатки существующих средств генерации VHDL-программ, приведена классификация библиотек программ как источников знаний о методиках проектирования аппаратуры. Предложен общий подход решения задачи автоматической генерации программ с учетом использования накопленных VHDL-решений и уровня квалификации проектировщика.

Автоматическая генерация, VHDL-программа, проектирование аппаратуры

Введение

В настоящее время существует большое количество универсальных средств описания аппаратуры, которые позволяют ускорить процесс проектирования, поддерживают разные стили описания устройств и обладают большим количеством достоинств. Среди них стоит отметить такие популярные языки как VHDL, Verilog, надстройка System C, System Verilog. Язык VHDL является одним из наиболее популярных языков такого класса. Он позволяет совместить достоинства графических моделей устройств и текстовых описаний алгоритмов их функционирования.

Длительное использование языков проектирования аппаратуры обеспечило наличие большого количества накопленных и апробированных решений в форме, удобной для хранения, передачи и редактирования. К достоинствам описания проекта в текстовом виде на HDL-языках можно отнести:

– компактность представления сложных логических проектов;

– легкость понимания и хорошая скорость создания проектов, включая относительную простоту поиска ошибок и внесения модификации в проект;

– возможность переноса проекта в различные среды разработки;

– простота настройки или перенастройки параметров разрабатываемых устройств или их фрагментов, например, разрядности или подмножества реализуемых функций;

– сближение методов проектирования аппаратуры и программных средств;

– улучшение взаимопонимания между проектировщиками различных подсистем вычислительных комплексов;

– возможность перевода программ пользователя, построенных на базе стандартных универсальных языков (например, C), в языки описания аппаратуры и обратно;

– наличие стандартных комментариев в текстах программ [1].

Однако, в настоящее время, нет средств обучения САПР, позволяющих обеспечить процесс передачи экспертных знаний о методиках проектирования либо напрямую – от эксперта, либо на основе ряда примеров. Обеспечение такой возможности позволит улучшить эффективность использования САПР на базе языка HDL за счет повышения уровня автоматизации процесса проектирования.

Актуальность задачи автоматического синтеза HDL-программ, обуславливается еще и тем, что благодаря скачку развития технологий, увеличению количества вентилей, которые могут быть размещены на плате, проблема оптимального размещения элементов отходит на второй план. В таких условиях более важными задачами становятся задачи быстрой разработки необходимой схемы и обеспечение ее устойчивой работы.

Наиболее эффективным подходом для решения задачи автоматического синтеза HDL-программ с использованием накопленных решений (библиотек программ) является применение методов искусственного интеллекта, разработка интеллектуальных средств и методик. На данный момент имеются некоторые интеллектуальные методы, обеспечивающие синтез таких программ. Однако данная задача еще далека от полного решения. Таким образом, целями данной работы являются:

– выполнить обзор существующих языков проектирования аппаратуры;

– проанализировать существующий уровень автоматизации генерации HDL- программ в существующих САПР;

– выполнить анализ интеллектуальных методов автоматизации синтеза HDL-программ;

– классифицировать возможные источники знаний о методиках проектирования VHDL–программ;

– определить цели и задачи будущих исследований.

Языки проектирования аппаратуры

Язык VHDL является проблемно-ориентированным и позволяет описывать структуры и (или) поведения широкого класса цифровых устройств с целью дальнейшего синтеза и (или) моделирования схем.

VHDL поддерживает три стиля описания аппаратных архитектур:

– структурное описание, в котором архитектура представляется в виде иерархии связанных компонентов;

– потоковое описание, в котором архитектура представляется в виде множества параллельных регистровых операций, каждая из которых управляется вентильными сигналами;

– поведенческое описание, в котором преобразование описывается последовательными программными предложениями, которые похожи на имеющиеся в любом современном языке программирования высокого уровня.

Каждый объект проекта состоит из описания интерфейса (entity declaration), где определяются входы и выходы объекта, и одного или более архитектурных тел (architecture body).

Программа снабжена мощным ядром моделирования. Поддерживается совместная работа с программами MatLab и Simulink[4].

Еще один из наиболее популярных языков является Verilog. Verilog — язык, сходный с языком программирования С как по синтаксису, так и по «идеологии». Если сравнивать VHDL и Verilog, то VHDL обладает большей универсальностью и может быть использован не только для описания РЭА, но и для описания других моделей. Однако на описание одной и той же конструкции в Verilog потребуется в 3–4 раза меньше символов, чем в VHDL [2].

В Verilog существуют специфические блоки проектирования, не имеющие аналогов в VHDL. Кроме того, этот язык позволяет подключать к проекту модули, написанные пользователем на языке С.

Основной структурной единицей Verilog описания – module (entity в VHDL). Модуль описывается ключевыми словами «module — endmodule». В файле может быть описано несколько модулей. Модули, связанные между собой, образуют иерархическую структуру.

Построение иерархической структуры начинается с нахождения Verilog-симулятором модуля верхнего уровня, который, как правило, содержит список портов, необходимых для подключения других модулей. Далее для определения порядка следования модулей учитываются либо имена портов, использованные при описании модулей либо расположение сигналов.

System C - представляет собой надстройку над языком программирования С++, реализованную в виде отдельных библиотек специальных классов. Основная идея создания данного языка проектирования состояла в том, что бы разрабатывать устройства на высоком уровне абстракции, добиться сокращения времени разработки систем на кристалле, в том числе, за счет отсутствия необходимости перехода с одного языка описания на другой. То есть данный язык был призван обеспечить все этапы проектирования аппаратуры вплоть до перехода к синтезируемой RTL-модели на языке VHDL.

Однако, наряду с базовой библиотекой SystemC 2.0, необходимо также использовать такие библиотеки как SCV (библиотека функциональной верификации) и TLM (библиотека транзакционного моделирования)[3].

Своеобразным синтезом С++ и Verilog стал язык System Verilog, отличительной чертой которого можно считать автоматизацию и упрощение процесса тестирования.

Ставшие классическими системы проектирования типа VHDL, Verilog постоянно совершенствуются и модифицируются. Так, например, система проектирования Active VHDL поддерживает не только язык VHDL, но и языки Verilog и SystemC. С помощью программы можно графически проектировать схемы устройств, а также конвертировать HDL-описание в графические структурные схемы и обратно. Система имеет мощный аппарат моделирования, поддерживает работу с программами MatLab и Simulink.

Анализ уровня автоматизации генерации VHDL-программ в существующих САПР

Среди имеющихся средств автоматизации создания моделей на языке VHDL в различных САПР можно назвать:

– использование шаблонов, макросов, мастеров для синтеза текстов программ;

– формирование языковых описаний по графической модели алгоритма, представленной в виде блок-схемы;

– использование наборов подключаемых библиотек программ;

– языковые помощники;

– генераторы IP-блоков (Intellectual Property).

Макросы служат средством автоматизации интерфейса самой системы проектирования и интеграции её в проект создаваемых HDL-

программ. Они включают: а) описания, заключаемые в системные ключевые слова; б) входные данные для программы; в) время симуляции в системе. Во время запуска макроса происходит передача параметров в систему проектирования. У разработчика появляются предпосылки расширить возможности языка за счёт введения собственных макроопределений, ориентированных на класс реализуемых проектов.

Мастера помогают быстро подготовить шаблон для заданной архитектуры программируемой системы и даже определить количество начальных параметров и типов. Но все остальные дальнейшие заботы по написанию программ ложатся на программиста-разработчика [4].

Следует отметить, что существуют различные интеллектуальные средства автоматизации формирования VHDL, Verilog, SystemC – модулей. Так, в САПР Active-HDL имеются такие инструменты, как Language Assistant (языковой помощник) и IP Core Generator (генератор интеллектуальных блоков-ядер) [5].

Языковой помощник содержит некоторое множество шаблонов типовых конструкций языков HDL, из которых можно создавать исходный код, не вникая в тонкости используемого языка. Однако, языковой помощник позволяет генерировать только основную структуру шаблона и не дает возможность задавать входные и выходные порты. Однако, существует возможность сделать шаблон диалоговым, что является, безусловно, положительным моментом с точки зрения пользователя создаваемого шаблона.

В качестве недостатка языкового помощника САПР Active-VHDL нужно отметить необходимость знания о месте расположения шаблонов в библиотеке. Кроме того, для описания недостающих в шаблоне портов необходимы знания языка VHDL.

В пакете OrCAD так же имеется языковой помощник — VHDL Samples, который позволяет описывать один входной (PORT1_NAME) и один выходной (PORT2_NAME) сигналы. Описания недостающих портов легко добавить при помощи копирования [5].

В САПР Xilinx ISE такого рода инструмент называется Language Templates. Преимуществом этого инструментария является возможность описания различных портов (входных, выходных и двунаправленных), как для одиночных, так и для шинных сигналов [5].

В САПР QUARTUS II также существует возможность использования коллекции HDL-шаблонов, которые содержат различные типы HDL–выражений, таких как ENTITY-описание, CASE-выражение [6].

Результатом работы второго типа инструментов (генераторов интеллектуальных блоков-ядер или IP-блоков), повышающих эффективность создания HDL-блоков, является полностью готовый для применения модуль.

В последнее время разработка IP-блоков становится самостоятельным направлением. Данной проблемой уже занимаются не только проектировщики интегральных схем, но и специализированные фирмы.

Такие модули могут использоваться в любой VHDL- или Verilog-системе, они построены на синтезируемом подмножестве языка, а значит, пригодны для инструментов синтеза и ПЛИС-технологий.

Выделяют три класса IP-блоков:

– программные IP-блоки (soft blocks);
– схемотехнические блоки (firm blocks);
– аппаратные блоки (hard blocks).

Программные IP-блоки, как правило, обладают определенной спецификой языка проектирования аппаратуры на функциональном или логическом уровне. Программные IP-блоки – гибкие, могут использоваться в различных проектах. Однако сложно определить характеристики разрабатываемого устройства при использовании определенного IP-блока в заданной технологии реализации.

Схемотехнические IP-блоки обладают предсказуемыми характеристиками площади и быстродействия, используются на вентильном уровне. Схемотехнические IP-блоки могут содержать некоторые данные о взаимном расположении элементов на площади кристалла.

Аппаратные блоки (hard blocks) предназначены для использования на физическом уровне проектирования интегральных схем. То есть относятся к уровню реализации топологии на кристалле, имеют определенные характеристики быстродействия, площади и прочие. Однако перенос одного аппаратного блока в другой проект весьма проблематичен и может привести к проектированию устройства «с нуля»[7].

Так, например, работа IP Core-генератора (Active VHDL) осуществляется в три этапа:

– проверяется наличие ошибок при вводе имен и значений параметров;

– выполняется собственно этап генерации;

– генератор создает необходимые файлы и папки, куда и помещает результат своей работы.

В результате генерации получается три файла:

– файл с расширением *.vhd (содержит сгенерированный VHDL-код);

– файл с расширением *.bds — содержит графическое изображение сгенерированного модуля;

– файл, позволяющий поместить графическое изображение сгенерированного модуля в папку проекта[5].

Подобного рода интеллектуальные блоки могут быть организованы в виде библиотек. В качестве примера такой библиотеки можно привести библиотеку GRLIB компании Gaiser Reseach, которая предназначена для работы в различных системах и операционных системах [8].

К достоинствам подобных средств автоматизации можно отнести:

– сокращение времени и сложности процесса проектирования за счет возможности генерации и использования стандартных блоков программ, описывающие те или иные типы устройств;

– снижение требований к квалификации эксперта.

Однако данные средства имеют следующие недостатки:

– закрытость;

– трудоемкость создания модулей автоматизации синтеза HDL-программ и необходимость высокого уровня квалификации разработчика-программиста;

– отсутствует возможности адаптации на другие уровни квалификации эксперта, а также на различные источники воплощения методик проектирования;

– отсутствует возможность построения вложенных шаблонов, формируются архитектуры только базовых типов блоков, допускающие только одноуровневую декомпозицию.

Под закрытостью подразумевается невозможность изменить алгоритм работы экспертной системы (ЭС) в других условиях, то есть отсутствует возможность обучения тем или иным методикам проектирования. Таким образом, ограничиваются возможности обычного пользователя САПР, обладающего низкой квалификацией в инженерии знаний, по передаче своей методики проектирования определенного вида устройств.

Программисты, формирующие модули автоматизации синтеза HDL-программ должны обладать высоким уровнем квалификации в инженерии знаний, то есть должны быть способны определить набор правил, входящих в методики проектирования, а также организовать диалог синтеза требуемого решения.

Интеллектуальным подходом создания HDL-программ можно также считать вариант использования специализированных CASE-технологий, обеспечивающих синтез программного обеспечения на требуемом языке программирования на основе диаграмм или блок-схем, представляющих алгоритм работы устройства.

Еще одним типичным подходом автоматизации формирования HDL-программ является синтез HDL-файлов на основе граф-схемы алгоритмов (ГСА), представляющих собой автоматную модель вычислительного устройства [9].

Наиболее распространенными CASE–системами проектирования РЭА являются системы, использующие UML (Rational Rose, MS Visio Enterprise, Argo UML/ Poseidon и др.). Однако, для того чтобы получить HDL-описание разрабатываемого устройства, одних UML диаграмм не достаточно, как правило, в связке с ними используется давно ставший стандартом XMI-формат представления проектируемых моделей. Кроме того, необходимо создать программу, позволяющую осуществлять переход от XMI к VHDL. Такого рода системы, чаще всего, имеют структуру, изображенную на рисунке 1.

Вместо блока «модель приложения», изображенного на рис.1, может быть использована RTL-модель проектируемого устройства, а также XMI-модель может быть заменена собственным форматом текстового описания UML-модели устройства. Примерами использования описанного подхода могут быть разработки, описанные в следующих источниках [10,11,12].

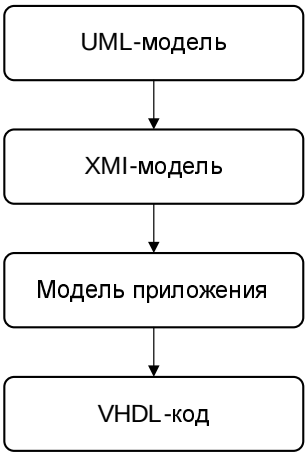

Рисунок 1 – Структура CASE-системы проектирования РЭА на основе UML

Недостатком такого рода подходов, использующих для описания проектов диаграммы и схемы, является практически ручное создание диаграмм, описывающих алгоритм работы вычислительного устройства, а также необходимость знания особенностей представления необходимой модели (типов блоков диаграмм и отношений между ними) в инструментальной среде разработки. Кроме того, возникает необходимость использования промежуточных модулей, позволяющих получать HDL-описание разрабатываемого устройства.

В качестве аналогичного подхода можно рассматривать синтез VHDL-файлов на основе программ, написанных на языках высокого уровня: C, C++, решающих дополнительно

оптимизационные задачи, например, позволяющие выполнять распараллеливания алгоритмов функционирования устройства [13].

Кроме того, существует ряд так называемых «надстроек», среди которых особое место занимает System C. Данная надстройка позволяет решать дополнительно оптимизационные задачи, например, выполнять распараллеливания алгоритмов функционирования устройства. Кроме того, такая надстройка позволяет пользователям-программистам, не зная специализированных HDL-систем, заниматься разработкой РЭА на языке высокого уровня. Однако эта надстройка используется только для создания поведенческих моделей и не приспособлена для использования на всем жизненном цикле проектирования устройств[2,3].

Существуют также надстройки над САПРами, которые чаще всего используются не для проектирования РЭА, а для других задач. Примером такой надстройки является объектно-ориентированная надстройка над AutoCAD, позволяющая разрабатывать чертежи интегральных схем и печатных плат [14].

Наиболее обще процесс синтеза структуры цифрового управляющего устройства на основе содержательного перечня микроопераций и отличающийся более эффективными графическими моделями представления алгоритма функционирования устройства, представлен в [15]. Данная работа реализована на основе модульного принципа и может быть использована в качестве надстройки для любого САПР цифровых устройств.

В качестве основного недостатка существующих надстроек над САПР-ами можно отметить отсутствие полной функциональности, обеспечивающей все этапы разработки устройств.

Таким образом, проанализировав существующие методы автоматизации создания HDL-файлов, можно сделать вывод, что задача автоматической генерации VHDL-программ является актуальной и быстро развивается.

Эта задача распадается на две составляющие: определение источников знаний о методах проектирования VHDL-программ и определение методов представления знаний и организации вывода.

Задача представления знаний рассматривается с точки зрения извлечения методик проектирования из VHDL-текстов с учетом адаптации на возможные источники возникновения библиотек программ, а так же с учетом квалификации проектировщика в области создания баз знаний и области проектирования устройств.

Классификация источников библиотек программ

В настоящее время наблюдается тенденция создания библиотек VHDL-программ, реализующих те или иные аппаратные решения. Данные решения, как правило, апробированы на практике, то есть имеют высокий уровень достоверности. Посредством библиотек решения могут накапливаться и систематизироваться для облегчения доступа. Данные библиотеки позволяют многократно использовать удачные решения или модифицировать их с учетом новых задач. Они являются объектами интеллектуальной собственности, то есть могут продаваться и покупаться. Эти библиотеки доступны в различных САПР или на web-порталах. Можно сказать, что данные библиотеки являются аккумулятором успешного опыта проектирования тех или иных устройств по тем или иным эвристическим методикам проектирования. Но выбор требуемого решения из библиотек, как правило, производится вручную и требует определенных навыков и хорошей ориентации в структурах библиотек программ.

Таким образом, актуальной является задача автоматического выявления эвристических методик проектирования, предназначенных для разработки требуемого класса устройств, из библиотек программ и формализации этих методик в форме баз знаний.

Следует отметить, что синтаксические и семантические особенности VHDL–программ, содержащихся в библиотеках, напрямую зависят от источника, из которого данные программы попадают в библиотеку.

Рассмотрим существующие источники библиотек-программ. Это могут быть:

– пользовательские программы, написанные вручную, которые сам пользователь может характеризовать и складывать в структуру, отражающую его личную библиотеку;

– программы, полученные путем автоматической перекодировки в язык VHDL из наборов решений, подготовленных пользователем в САПР как модели структур (САПР типа OrCAD);

– библиотечные программы, предоставляемые пользователю в тех же САПР (сам САПР VHDL) для создания своих решений;

– сгенерированные пользователем существующими в САПР генераторами или мастерами в процессе создания своих решений.

То есть библиотеки программ можно классифицировать следующим образом: написанные вручную, перекодированные, подготовленные, сгенерированные.

Кроме того, множество библиотечных программ, которыми располагает пользователь и на основе которых строится база знаний ЭС,

может иметь различную мощность по числу программ: программа может быть одна или их может быть много.

Библиотечные программы, независимо от их числа, могут иметь различную иерархию структуры модулей, отличаться числом типов используемых модулей, глубиной вложенности блоков.

Следует отметить, что в различных классах VHDL-программ имеются как типичные библиотечные идентификаторы с известной семантикой, так и - многочисленные идентификаторы, семантика которых привязана к конкретной разработке и не имеет универсального характера. Это означает, что если имеются две VHDL–программы, описывающие структуру одного и того же устройства в разных контекстах применения, или - описывающие близкие структуры, не имеющих принципиальных отличий и относящиеся к одному типу, то они могут быть представлены с использованием различных идентификаторов и с различным порядком описания общих структур.

Применение различных идентификаторов и структур описания VHDL-модулей различными пользователями и системами обуславливает сложность извлечения методики проектирования из VHDL-текстов. В этом случае при формировании базы знаний можно использовать опыт эксперта предметной области, учитывая его квалификацию в области создания баз знаний, что позволит эффективно адаптировать инструментальный комплекс на специфику условий извлечения методики проектирования.

Таким образом, учитывая множество накопленных решений в области проектирования аппаратуры и недостатки существующих интеллектуальных методов проектирования VHDL-программ, возникает задача создания инструментального комплекса способного:

– обеспечить извлечение методик проектирования из существующих библиотек VHDL-программ;

– обеспечить достаточно малую трудоемкость процесса извлечения методик проектирования, то есть комплекс должен быть адаптирован на «глупого» эксперта [16];

– учитывать уровень возможности проектировщика помочь системе в данном процессе, что позволит эффективно адаптировать инструментальный комплекс на специфику условий извлечения методики проектирования.

Анализ интеллектуальных методов построения систем автоматизации синтеза VHDL-программ

Система, решающая описанные выше задачи, может быть реализована в виде интеллектуальной надстройки над САПР VHDL. Очевидно, что предполагаемая интеллектуальная надстройка будет иметь ряд подсистем, одна из которых - подсистема формирования базы знаний накопленных решений на языке описания аппаратуры VHDL.

Рассмотрим существующие средства и методы искусственного интеллекта, способные обеспечить решение задачи формирования необходимой базы знаний. Фактически, необходимо решать два класса задач:

– адаптация к уровню квалификации пользователя в пределах возможностей «глупого» эксперта, что является задачей создания модельной процедуры синтеза различного уровня полноты. Речь идет о полноте множества технических заданий (пространство обликов системы, ПОС), множества решений-прототипов (целевого пространства системы, ЦПС) и механизме реализуемости, то есть взаимосвязи между ПОС и ЦПС;

– адаптация на источник получения программ - задача извлечения знаний из текстов, имеющих различные доли формальных (однозначных) и неформальных (субъективных, неоднозначных) компонент.

Для решения задачи извлечения знаний из текстов на языке VHDL можно использовать два класса средств и методов:

– формальные методы теории синтаксического анализа и грамматик для формальных языков;

– методы извлечения знаний из естественно-языковых текстов [17].

Достоинствами формальных методов являются многочисленные механизмы лексического, синтаксического анализа текстов, а недостатками - отсутствие средств семантического анализа.

По поводу методов извлечения знаний из языковых текстов можно сделать вывод, что формируемые модели представления знаний, извлеченные из текста, не вполне соответствуют специфике VHDL. Кроме того, в этом случае решается задача извлечения знаний, но не решается задача обобщения знаний, то есть - задача построения базы знаний. Достоинством естественно-языковых методов является наличие многочисленных способов формирования моделей объекта как результата извлечения знаний.

Учитывая специфику поставленной выше задачи, можно сделать вывод, что ни один из двух перечисленных выше классов методов извлечения знаний из текстов не обеспечивает решение поставленной задачи в целом. Таким образом, необходим третий подход, обеспечивающий решение задачи с наличием формальных и неформальных компонент. То есть необходима разработка соответствующего инструментария.

Следует отметить наличие такого подхода, представленного в работах [16,18], обеспечивающих в целом решение данной задачи.

Сделаем краткую характеристику предлагаемого подхода:

– использование теоретико-множественных операций над порождающими грамматиками как механизма обобщения, классификации и выявления свойств первого рода, задающих отличия сравниваемых структур (механизм создания ПОС);

– использование предикативных грамматик, как механизма обеспечения ввода множеств продукционных зависимостей над грамматиками, представленными в виде «и–или» дерева. Это соответствует механизму вывода в модуле знаний, отражающего проектную процедуру синтеза.

Данный подход обеспечивает:

– решение поставленной выше задачи при использовании VHDL-программ, синтезированных в среде САПР типа ORCAD без какой-либо субъективной неформальной компоненты;

– автоматическое построение ПОС, состоящее из свойств первого рода, фактически являющимися или-узлами получившегося при обобщении «и-или» дерева, что соответствует подградации совсем «глупого» эксперта [16].

Недостатки данного подхода:

– не учитываются более мелкие градации «глупого» эксперта, что не позволяет достичь высокой эффективности процесса адаптации на условия создания интеллектуальной надстройки;

– не обеспечивается возможность учета различных источников появления VHDL-программ и наличия в них субъективных компонент, связанных с описанием семантически эквивалентных или близких решений, но с наличием разницы в идентификации и порядке описания[19].

Таким образом, можно сделать вывод, что построение методов синтеза VHDL-файлов на базе предложенного подхода, лишенных названных недостатков, есть важное перспективное направление.

Заключение

В работе отмечается актуальность повышения уровня автоматизации процесса проектирования аппаратуры с целью снижения времени разработки устройств и требований к уровню квалификации проектировщиков.

Выполнен краткий обзор наиболее популярных САПР проектирования РЭА, выполнен анализ уровня автоматизации генерации VHDL-программ в существующих САПР, проведена классификация существующих возможных источников знаний о методиках проектирования, выполнен анализ интеллектуальных методов автоматизации синтеза HDL-программ.

Таким образом, возникает необходимость создания подсистемы формирования БЗ интеллектуальной надстройки над САПР VHDL путем извлечения методик проектирования из набора прецедентов в разных условиях квалификации эксперта и имеющихся источников знаний (библиотек программ). Построение такого рода оболочки позволит существенно повысить эффективность процесса проектирования VHDL-программ обычными пользователями.

Перспективным направлением работы является модификация описанного подхода с учетом специфики задачи и построение на его основе интеллектуальных средств синтеза VHDL-программ, использующих алгоритм теоретико-множественных операций над грамматиками.

Список литературы

1. Григорьев А.В. Интеллектуализация процесса проектирования аппаратуры средствами языка VHDL / А.В. Григорьев, Д.А. Кошелева // Наук. праці ДонНТУ. Серия: ІКОТ. – 2006. – Вып. 93. – С. 99-105.

2. Знакомство с System C. – Режим доступа: http://systemc.dax.ru/book/. – Заглавие с экрана.

3. Шагурин И. Применение языка System C и средств разработки на его основе для проектирования Систем на кристалле / И. Шагурин, В. Канышев // Инженерная практика. – 2006. – №9. – С.51-56.

4. Шалагинов А. Изучаем Active-HDL. Урок 5. Создание проекта в текстовом формате / А. Шалагинов // Компоненты и технологи. – 2009. – №3. – С. 118-122.

5. Шалагинов А. Изучаем Active-HDL 7.1. Урок 6. Инструменты, повышающие эффективность создания HDL-моделей / А. Шалагинов // Компоненты и технологи. – 2009. – №8. – С. 118-123

6. Quartus II Introduction Using VHDL Design. Altera Corporation. – 2008. – Режим доступа: http://people.ee.duke.edu/~dwyer/courses/ece52/tut_quartus_intro_vhdl.pdf

7. Уровни представления и проектирования СБИС. – Режим доступа: http://relec.ru/digital/256-urovni-predstavleniya-i-proektirovaniya-sbis.html. – Заглавие с экрана.

8. Шагурин И. Применение IP-библиотек для проектирования СнК / И. Шагурин, В. Канышев, А. Родионов // Электронные компоненты. – 2009. – №1. – С. 22-25.

9. Зеленёва И.Я. Система автоматизированного проектирования композиционных микропрограммных устройств управления / И.Я. Зеленёва, Л.И. Дорожко, А.Н. Мирошкин // Наукові праці Донецького національного технічного університету. Серія: Проблеми моделювання та автоматизації проектування. – 2007. – С. 54-61.

10. Le Beux S. A model driven engineering design flow to generate VHDL / Le Beux S., Marquet P., Honoré A., and Dekeyser J.-L. // Proceedings of the International Workshop on Model Driven Design for Automotive Safety Embedded Systems (ModEasy'07). – Barcelona, Spain, September 2007. – P. 15–22.

11.Li L. UML to SystemVerilog Synthesis for Embedded System Models with Support for Assertion Generation / Li L., Coyle F., Thornton M.A. // Proceedings of the ECSI Forum on Design Languages. – September 18-20, 2007. – Paper 10 on CD-ROM.

12.Compiling UML State Diagrams into VHDL: An Experiment in Using Model Driven Development / D.H. Akehurst, O.Uzenkov, W.G.Howells, K.D.Mcdonald-Maier2, B.Bordbar // Forum on specification and Design Languages, FDL 2007, September 18-20. – Barcelona, Spain, 2007. – P. 219-224.

13.Дубров Д.В. Экспериментальный конвертер с языка С в hdl на основе диалогового высокоуровневого оптимизирующего распараллеливателя / Д.В. Дубров, Р.Б. Штейнберг // Доклады пятой международной конференции «Параллельные вычисления и задачи управления», Москва 2010, 26-28 октября. – М.: PACO, 2011. – С. 865-870.

14.Романычева Э.Т. Инженерная и компьютерная графика / Э.Т. Романычева, Т.Ю. Соколова, Г.Ф. Шандурина. – М.: ДМК Пресс, 2001. – 592 с.

15.Защелкин К.В. Информационная технология автоматизированного проектирования цифровых управляющих устройств / К.В. Защелкин, Е.Н. Иванова // Електромашинобудування та електрообладнання: міжвідомчий науково-технічний збірник. – 2009. – Вип. 72. – С. 68-72.

16.Григорьев А.В. Пути создания интеллектуальных САПР при различных уровнях квалификации экспертов /А.В. Григорьев // Искусственный интеллект: научно-теоретический журнал. – 2005. – №3. – С. 758–763.

17.Алексеев С.С. Методы машинного обучения в задачах извлечения информации из текстов по эталону / С.С. Алексеев, В.В. Морозов, К.В. Симаков // Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Труды XI Всероссийской научной конференции RCDL'2009. – Петрозаводск: КарНЦ РАН, 2009. – С. 237-246.

18.Григорьев А.В. Теоретико-множественные операции над грамматиками как механизм работы со знаниями в интеллектуальных САПР / А.В. Григорьев // Вісник Східноукраїнського національного університету імені Володимира Даля. – 2002. – № 2(48). – С. 186-194.

19.Григорьев А.В. Перспективные направления решения задачи синтеза HDL-программ в САПР РЭА / А.В. Григорьев, Д.А. Грищенко // Сборник трудов XI международной научной конференции им. Т.А. Таран. – К., 2011. – С. 75-81.

Надійшла до редколегії 10.10.2011

О.В. ГРИГОР'ЄВ, Д.О. ГРИЩЕНКО
Донецький національний технічний університет

Аналіз засобів автоматизації побудови VHDL-програм.

У статті виконано аналіз існуючого рівня автоматизації генерації VHDL-програм, наведена класифікація бібліотек програм як джерел знань про методики проектування апаратури, визначені рівні кваліфікації експерта; запропоновані оцінки ефективності функціонування бази знань, що відповідають потрібній методиці проектування деякого типу виробів

Автоматична генерація, VHDL-програма, проектування апаратури

A.V. GRIGOREV, D.A. GRISCHENKO
Donetsk National Technical University

Analyses of VHDL-programs build automation.

In this paper the VHDL-programs automatic generation level is analyzed. Advantages and disadvantages of existing VHDL-programs generation tools is defined. Programs libraries used as knowledge source about hardware design methods are classified. Common approach to automatic programs generation problem is proposed. It uses the accumulated VHDL-solutions and adopts to the designer skill level

VHDL-program, automatic generation, hardware design