МИНИСТЕРСТВО ОБЩЕГО И ПРОФЕССИОНАЛЬНОГО
ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ТАГАНРОГСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

КАФЕДРА КОНСТРУИРОВАНИЯ ЭЛЕКТРОННЫХ СРЕДСТВ

МЕТОДИЧЕСКОЕ РУКОВОДСТВО

Описание проектов СБИС с использованием языка VHDL

Для студентов специальностей 2002, 2008, 2201, 2203, 2205.

Таганрог 1998

УДК 621.382.82(07)

Составители: Б.Г.Коноплев, Е.А.Рындин, В.Г.Ивченко

Методическое руководство. “Описание проектов СБИС с использованием языка VHDL”. Таганрог: Изд-во ТРТУ, 1998. 28 с.

Рассматриваются концепции языков описания проектов СБИС, назначение, основные понятия и определения языка VHDL, стили описания и библиотеки проектов, автоматизированные средства обнаружения ошибок в VHDL-описаниях. На конкретных примерах показаны возможности функционально-логического моделирования цифровых устройств.

Табл. 1. Ил. 2. Библиогр.: 5 назв.

Рецензенты:

В.Ф.Гузик, д-р техн. наук, профессор, заведующий кафедрой ВТ ТРТУ;
Б.Г.Фрадкин, канд. техн. наук, заместитель директора научно-конструкторского бюро вычислительных систем, г.Таганрог.

ВВЕДЕНИЕ

Основными проблемами на всех этапах проектирования СБИС являются обеспечение бездефектности и сокращение времени проектирования. Поскольку современные СБИС содержат миллионы полупроводниковых структур на кристалле, решение данных проблем возможно лишь посредством расширенного использования различных методов автоматизации проектирования в САПР, опирающихся на мощную вычислительную базу.

Как известно, основой любой САПР является программное обеспечение, позволяющее реализовать методы и алгоритмы автоматизированного проектирования. Для различных этапов проектирования используются различные алгоритмы и программы и, естественно, различные исходные данные. Следовательно, необходимы специальные языки описания проектов, позволяющие быстро и эффективно представлять исходные данные для проектирования в форме, воспринимаемой существующими пакетами программ.

Грамматика, определяющая форматы подготовки исходных данных, называется языком описания.

В зависимости от типа программы и набора исходных данных используются самые разнообразные языки описания: язык описания логических связей, язык описания соединений транзисторов, язык описания электрических постоянных и др. В процессе развития и совершенствования этих языков было выявлено несколько проблем.

Основной из них являлось то обстоятельство, что эти языки описания были абсолютно несхожи между собой. Каждый из языков отражал входной формат независимо разработанных программ и поэтому имел индивидуальные особенности. Еще на ранних этапах развития САПР этот факт был признан неудобным, но, тем не менее, долгое время оставался без должного внимания.

В настоящее время с точки зрения конструктора формирование входных данных для каждой программы в отдельности не просто неудобно, а практически невозможно. Особенно, если речь идет о проектировании СБИС. Поскольку используемые в САПР языки описания специализированы, они не взаимозаменяемы. Поэтому при проектировании СБИС оказывалось необходимым готовить входные данные примерно по 10 различным грамматикам. Непросто осуществить и автоматическое преобразование данных между языками, так как различаются принципы построения моделей описания. Поэтому одной из основных движущих сил создания высокоэффективных САПР СБИС явилась разработка концепции общего языка описания проектов.

В соответствии с существовавшей ранее концепцией, каждая программа обработки имела собственный язык описания и языковый процессор. В противоположность этому, концепция общего языка описания предполагает наличие общей грамматики, общей базы данных и общего языкового процессора. При этом необходимый набор входных данных содержит информацию для всех программ обработки. Поскольку все эти данные готовятся в едином формате, можно избежать избыточности, характерной для первой концепции, так как

множества входных данных для различных программ обработки частично перекрываются. Этот подход позволяет сократить время подготовки исходных данных, а также снизить вероятность ошибки при вводе больших массивов данных.

Примером общего языка описания проектов СБИС является VHDL (VHSIC Hardware Description Language - язык описания аппаратуры на базе сверхскоростных интегральных схем), являющийся формальной записью, предназначенной для описания функции и логической организации цифровых систем. Функция системы определяется как преобразование значений на входах в значения на выходах, причем время в этом преобразовании задается явно. Организация системы задается перечнем связанных компонентов.

Язык VHDL в настоящее время используется в качестве международного стандарта описания вычислительных систем (ВС) любого уровня сложности (микросхема, плата, блок, устройство, ЭВМ, комплекс).

Язык описания VHDL может быть использован на всех этапах разработки электронных систем: проектирование, верификация, синтез и тестирование аппаратуры, передача данных о проекте.

1. ПЕРВИЧНАЯ АБСТРАКЦИЯ ЯЗЫКА VHDL

Объект проекта (entity) представляет собой описание компоненты проекта, имеющей заданные входы и выходы и выполняющей определенную функцию. Объект проекта может представлять всю проектируемую систему, некоторую подсистему, устройство, узел, стойку, плату, кристалл, макроячейку, логический элемент и т.п.

В описании объекта проекта можно использовать компоненты, которые, в свою очередь, могут быть описаны как самостоятельные объекты проекта более низкого уровня. Таким образом, каждый компонент объекта проекта может быть связан с объектом проекта более низкого уровня. В результате такой декомпозиции проекта пользователь строит иерархию объектов проекта, представляющих весь проект в целом. Такая совокупность объектов проекта называется иерархией проекта (design hierarchy).

Каждый объект проекта состоит, как минимум, из двух различных типов описаний: описания интерфейса и одного или более архитектурных тел.

Интерфейс описывается в объявлении объекта проекта (entity declaration) и определяет только входы и выходы.

Для описания поведения объекта или его структуры служит архитектурное тело (architecture body).

Для задания объектов, использованных при создании проекта, применяется объявление конфигурации (configuration declaration).

В языке VHDL предусмотрен механизм пакетов для часто используемых описаний, констант, типов, сигналов. Эти описания помещаются в объявлении пакета (package declaration).

Если пользователь осуществляет нестандартные операции или функции, их интерфейсы описываются в объявлении пакета, а тела содержатся в теле пакета (package body).

Таким образом, при описании цифровой системы на языке VHDL можно использовать пять различных типов описаний: объявление объекта проекта, архитектурное тело, объявление конфигурации, объявление пакета и тело пакета. Каждое из описаний является самостоятельной конструкцией языка VHDL, может быть независимо проанализировано анализатором и поэтому получило название "Модуль проекта" (design unit). Модули проекта, в свою очередь, можно разбить на две категории: первичные и вторичные. К первичным модулям относятся различного типа объявления. Ко вторичным - отдельно анализируемые тела первичных модулей. Один или несколько модулей проекта могут быть помещены в один файл MS DOS, называемый файлом проекта (design file).

Каждый проанализированный модуль проекта помещается в библиотеку проекта (design library) и становится библиотечным модулем (library unit). Данная реализация позволяет создать любое число библиотек проекта. Каждая библиотека проекта в языке VHDL имеет логическое имя (идентификатор). Фактическое имя файла, содержащего эту библиотеку, может совпадать или не совпадать с логическим именем библиотеки проекта. Для ассоциирования логического имени библиотеки с соответствующим ей фактическим именем предусмотрен специальный механизм установки внешних ссылок.

По отношению к сеансу работы VHDL существует два класса библиотек проекта: рабочие библиотеки и библиотеки ресурсов.

Рабочая библиотека - это библиотека, с которой в данном сеансе работает пользователь и в которую помещается библиотечный модуль, полученный в результате анализа модуля проекта.

Библиотека ресурсов - это библиотека, содержащая библиотечные модули, ссылка на которые имеется в анализируемом модуле проекта.

В каждый конкретный момент пользователь работает с одной рабочей библиотекой и произвольным числом библиотек ресурсов.

2. СТИЛИ ОПИСАНИЯ В ЯЗЫКЕ VHDL

VHDL поддерживает три различных стиля для описания аппаратных архитектур:

1) структурное описание (structural description), в котором архитектура представляется в виде иерархии связанных компонентов;
2) потоковое описание (data-flow description), в котором архитектура представляется в виде множества регистровых операций, каждая из которых управляется вентильными сигналами (потоковое описание соответствует стилю описания, используемому в языках регистровых передач);
3) поведенческое описание (behavioral description), в котором преобразование описывается последовательными программными предложениями, которые похожи на имеющиеся в любом современном языке программирования вы-

сокого уровня. Все три стиля могут совместно использоваться в одной архитектуре.

2.1. Структурное описание

При структурном описании (structural description) объекта проекта архитектура представляется в виде иерархии связанных компонентов.

Каждый экземпляр компонента представляет часть проекта, которая, с другой стороны, может быть описана объектом проекта низшего уровня, также состоящим из связанных компонентов. Таким способом может быть построена иерархия объектов проекта, которая представляет весь проект.

Компонентом может быть один вентиль, микросхема, плата или целая подсистема. Иерархия может представлять структурное разбиение проекта или функциональную декомпозицию.

2.2. Потоковое описание

При потоковом описании (data-flow description) объекта проекта его архитектура представляется в виде множества регистровых операций, каждая из которых управляется вентильными сигналами. Потоковое описание соответствует стилю описания, используемому в языках регистровых передач.

Структурный тип описания отражает декомпозицию на компоненты и делает акцент на соединениях, которые должны быть проведены между компонентами. Компоненты могут быть некоторыми абстрактными функциональными устройствами или соответствовать физическим элементам, микросхемам, печатным модулям.

В потоковом описании, наоборот, акцент делается на потоке информации между памятью и элементами с вентильным управлением. Этот поток информационного обмена регулируется и направляется управляющими элементами, которые логически отделены от путей данных. Так как пути данных показаны явно, то потоковое описание не уделяет внимания структуре возможных реализаций.

2.3. Поведенческое описание

Слова "поведенческое описание" ассоциируются с последовательно выполнимым кодом процедурного типа, подобным коду на языках программирования Фортран или Ада. Некоторые языки описания аппаратуры обеспечивают механизм для вызова кода, написанного на Фортране, Паскале или Си, для того чтобы описать сложное поведение. Другие, такие как VHDL, включают поведенческие предложения в язык.

Имеется много причин для использования поведенческого стиля в описании аппаратуры, но основным является то, что поведенческое описание определяет с любой желаемой степенью точности функционирование устройства без определения его структуры.

Например, разработчик может подробно описать поведение подсистемы, а проработку деталей реализации передать другим. Реализация может выполняться либо специализированным предприятием, либо может быть передана в другую группу в той же самой организации. При использовании поведенческого стиля описания разработчик избегает чрезмерной спецификации структуры или уклона разработки в сторону какой-либо одной технологии реализации. Это позволяет обеспечить технологическую инаприантность (независимость от технологии) проекта и упростить модернизацию изделия в будущем.

Используя поведенческое описание изготовителя, потребитель может построить соответствующее структурное описание для подсистемы. Со структурной моделью должны быть связаны временные спецификации и спецификации данных для каждого компонента подсистемы в определенных управляемых и наблюдаемых выходах. Основываясь на результатах, потребитель может определить пригодность подсистемы для включения в большую систему.

Предложения поведенческого описания в VHDL представляют современный язык структурного программирования, который ближе всего к языкам Паскаль или Модула-2 по мощности и легкости использования.

3. СОСТАВ УЧЕБНОГО ПАКЕТА ПРОГРАММ VHDL

В состав изучаемого учебного пакета программ VHDL входят четыре основных раздела:

1) резидентный справочник (каталог VHDLHELP);
2) система синтаксического и семантического контроля VHDL-описаний “VHDL-анализатор” (каталог VA);
3) демонстрационный пример (каталог DEMO);
4) библиотечные модули и пакеты (каталог UMB).

3.1. Резидентный справочник

Резидентный справочник предназначен для получения справки по языку VHDL в процессе работы различных компонентов САПР вычислительных систем. При этом программная связь этих компонентов с резидентным справочником не требуется. В качестве компонентов могут выступать средства моделирования, анализа, обучения, подготовки описаний, в том числе универсальные текстовые процессоры.

Справочник содержит полное формализованное описание языка VHDL, построенное на основе текста стандарта IEEE 1076 VHDL, а также примеры и дополнительную методическую информацию, помогающую в составлении описаний вычислительных систем на языке VHDL.

Вся справочная информация тематически упорядочена и сгруппирована по разделам. Структура справочной информации имеет иерархическую концепцию построения.

Наличие большого количества примеров, информации о семантике языка, различных стилях описания позволяет использовать справочник также автономно для обучения языку VHDL.

Для работы с резидентным справочником необходимо перед началом сеанса работы инициализировать справочник. Далее справочником можно пользоваться в процессе работы других программ, вызывая его с помощью функциональных клавиш.

Инициализация справочника осуществляется запуском файла VHDLHELP.EXE, находящегося в каталоге VHDLHELP.

После этого при наличии достаточного объема свободной оперативной памяти и места на диске для размещения временных файлов обмена в текущем каталоге будет выдано сообщение об успешной инициализации и справочник будет резидентно размещен в памяти до тех пор, пока не будет набрана командная строка, включающая параметр выгрузки из памяти /U:

VHDLHELP /U

Взаимодействие со справочником в течение одного сеанса начинается с вызова справочника и заканчивается закрытием окна справочника.

Для вызова справочника используются следующие функциональные клавиши, предлагаемые по умолчанию:

1) <Left Shift><F1> - вызвать справочник по слову, на которое указывает курсор;
2) <Left Shift><F2> - показать предыдущее окно справочника (окно справочника, которое было последним во время предыдущего сеанса работы со справочником; после инициализации справочника информация о предыдущих сеансах работы теряется);
3) <Left Shift><F3> - показать указатель тем справочника (главный каталог).

После вызова справочника на экране появится окно, содержащее текст раздела справочника или главного каталога.

В тексте могут содержаться фрагменты, набранные другим цветом (оттенком). Это - или понятия, на которые следует обратить внимание, или ключевые поля (перекрестные ссылки), соответствующие другим разделам справочника.

Резидентный справочник позволяет:

- просмотреть текст одного раздела справочника;
- перейти от одного раздела справочника к другому;
- изменить положение и размеры окна справочника;
- если Вы работаете в текстовом редакторе, то выделить фрагмент раздела справочника и вставить этот фрагмент в редактируемый текст.

Ниже приведены клавиши, их комбинации, которые можно использовать для просмотра текста справочника, и их назначение.

<Up> (стрелка вверх) или <Ctrl E> - перемещает курсор на одну строку вверх, прокручивает текст окна справочника, когда курсор достигает верхней строки окна.

<Down> (стрелка вниз) или <Ctrl X> - перемещает курсор на одну строку вниз, прокручивает текст окна справочника, когда курсор достигает нижней строки окна.

<Left> (стрелка влево) или <Ctrl S> - перемещает курсор на один столбец влево, прокручивает текст окна справочника, если курсор достигает крайнего левого столбца и столбец 1 текста в настоящий момент невидим.

<Right> (стрелка вправо) или <Ctrl D> - перемещает курсор на один столбец вправо, прокручивает текст окна справочника, если курсор достигает крайнего правого столбца и текст в настоящий момент не прокручен до предела, установленного для горизонтальной прокрутки.

<PgUp> или <Ctrl R> - прокручивает текст вниз на высоту окна.

<PgDn> или <Ctrl C> - прокручивает текст вверх на высоту окна.

<Home> - перемещает курсор на первый столбец текущей строки.

<Ctrl PgUp> - перемещает курсор на первый столбец первой строки текста справочника, при необходимости выполняя прокрутку.

<End> - перемещает курсор на конец текущей строки.

<Ctrl PgDn> - перемещает курсор на конец последней строки текста справочника, при необходимости выполняя прокрутку.

Есть несколько вариантов перехода к другим разделам справочника:

1) переход по перекрестной ссылке (ключевым полям, выделенным другим цветом в тексте раздела);
2) переход к главному каталогу и дальнейший выбор раздела справочника в соответствии с его иерархией;
3) возврат к ранее просматриваемому разделу справочника.

Способ перехода по перекрестной ссылке из главного каталога и разделов справочника несколько различается.

В случае главного каталога окно находится в режиме выбора и одна активная перекрестная ссылка всегда выбрана (выделена другим цветом). Изме-

нить ее можно с помощью клавиш, управляющих просмотром текста. После выбора необходимо нажать клавишу <Enter>.

Для перехода по перекрестной ссылке из раздела справочника необходимо подвести курсор к ключевому полю, соответствующему другому разделу, и нажать клавишу <Enter>.

Ниже приведены клавиши, их комбинации, которые можно использовать для перехода к другому разделу справочника, и их назначение.

<Tab> - перемещает курсор на следующую перекрестную ссылку, при необходимости прокручивая текст окна. Если следующей перекрестной ссылки нет, ничего не делает.

<ShiftTab> - перемещает курсор на предыдущую перекрестную ссылку, при необходимости прокручивая текст окна. Если предыдущей перекрестной ссылки нет, ничего не делает.

<Enter> - если курсор расположен над выделенной перекрестной ссылкой, загружается и выводится на экран указанная тема. Если же окно находилось в режиме выбора (на экране показан главный каталог справочника), загружается тема, соответствующая текущему элементу выбора, и на экран выводится ее первая страница.

<F1> - переключает окно в режим выбора и выводит на экран каталог.

<Alt F1> - загружает, если возможно, раздел справочника, который был загружен перед текущим, и прокручивает текст до позиции, которая была при последнем просмотре.

Для изменения размера и местоположения окна справочника могут быть использованы следующие клавиши:

<F5> - открытие окна справочника до размера экрана или возврат к прежнему размеру;

<F6> - перевод справочника в режим изменения размера и перемещения окна; после этого с помощью клавиш со стрелками можно переместить окно справочника, а при одновременном нажатии клавиш со стрелками на функциональной клавиатуре и клавиши Shift - изменить размер окна справочника.

Нажатие клавиши <Esc> приводит к закрытию окна справочника и окончанию сеанса работы со справочником.

Главный каталог справочника (верхний иерархический уровень) включает следующие разделы:

1. Введение.
2. Руководство пользователя.
3. Синтаксические конструкции.
4. Стили описания.
5. Проблемные подмножества.
6. Модели данных.
7. Моделирование.
8. Семантика.
9. Предопределенные элементы.
10. Лексические элементы.

11. Ключевые слова.
12. Глоссарий.

3.2. Система синтаксического и семантического контроля VHDL-описаний

Система "VHDL-анализатор" (в дальнейшем кратко VA) предназначена для синтаксического и семантического анализа описаний, составленных на языке VHDL (стандарт IEEE Std 1076/1987 г.).

Модуль проекта - минимальная конструкция языка VHDL, которая может быть проанализирована. Как уже отмечалось, к ним относят: объявление объекта (entity declaration), архитектурное тело (architecture body), конфигурацию (configuration declaration), объявление пакета (package declaration), тело пакета (package body).

В данной версии объем анализируемого модуля проекта не должен превышать 64 кбайта.

К дополнительным возможностям системы относятся:

- встроенный текстовый редактор, используемый для создания и модификации исходных описаний;
- администратор библиотек, используемый для создания, поддержки и манипулирования с библиотеками проекта;
- реверсивный анализатор, используемый для восстановления исходного описания из библиотеки проекта.

Система "VHDL-анализатор" может быть использована как в режиме меню, так и в командном режиме.

Пользователь, имеющий описание цифровой системы на языке VHDL и желающий его проанализировать, прежде всего должен ввести исходный текст описания на магнитный диск с помощью любого, имеющегося в его распоряжении, редактора текстов или использовать собственный редактор VA, правила работы с которым описаны ниже.

Один или несколько модулей проекта любого вида могут быть помещены в один файл проекта (рис. 1). Имя файла проекта и место его расположения пользователь выбирает по своему усмотрению. Однако, если имя файла проекта имеет расширение .VHD и все файлы проекта расположены в одном каталоге, работа с VA значительно упрощается.

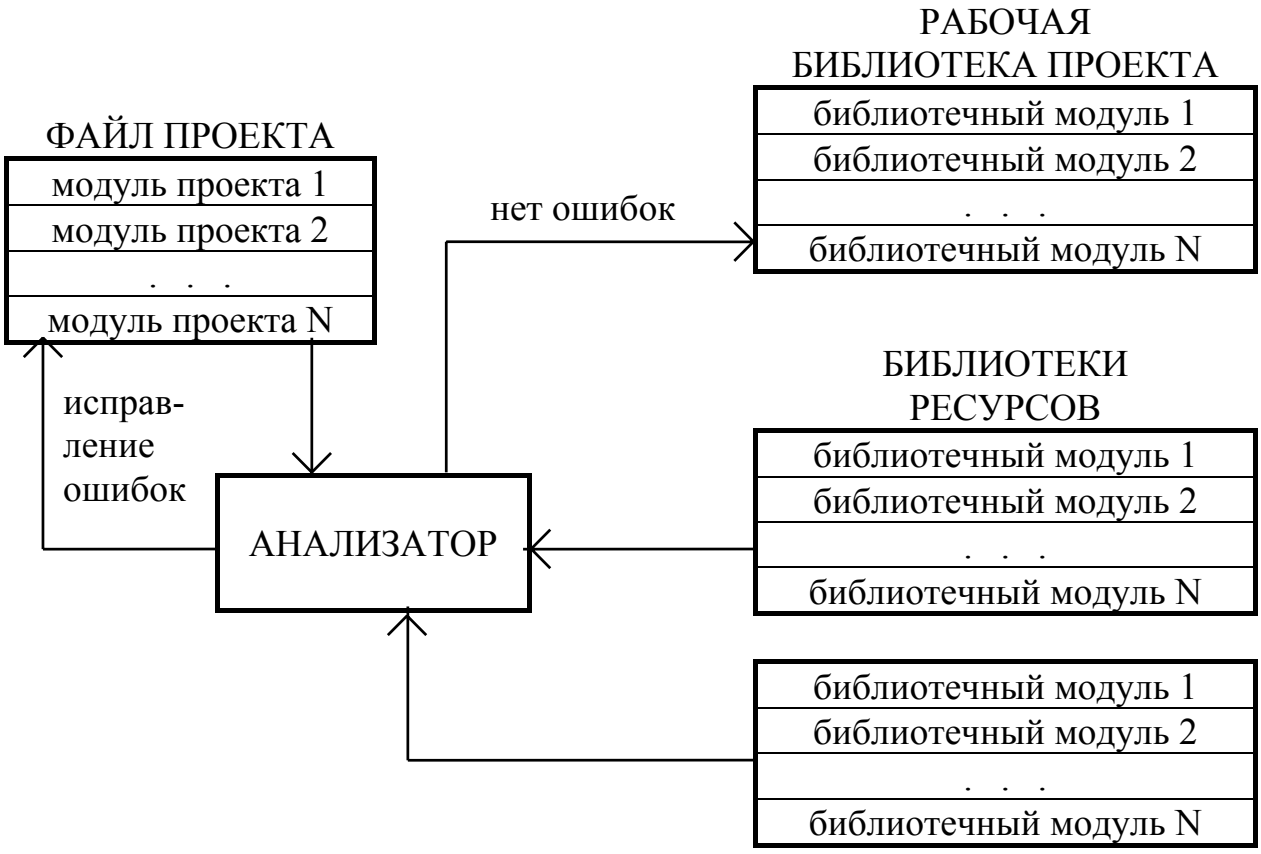

Рис. 1. Схема функционирования VHDL-анализатора

Все модули проекта, расположенные в одном файле, анализируются системой в текстуальном порядке и в виде библиотечных модулей помещаются в выбранную рабочую библиотеку. В процессе анализа конкретного файла может использоваться только одна рабочая библиотека и множество библиотек ресурсов (см. рис. 1).

При работе с VA следует соблюдать порядок, в котором должны анализироваться модули проекта:

1) если первичный модуль имеет ссылку на любой другой модуль, то последний должен быть проанализирован раньше модуля, внутри которого содержится эта ссылка;
2) вторичный модуль, соответствующий конкретному первичному модулю, может быть помещен только в ту библиотеку проекта, в которой расположен этот первичный модуль.

Следовательно:

– если первичный и соответствующий ему вторичный модули расположены в одном файле проекта, текст первичного модуля проекта должен быть помещен перед текстом вторичного модуля;
– если эти модули расположены в разных файлах проекта, то, во-первых, файл с первичным модулем следует анализировать раньше, чем файл с вторичным модулем, во-вторых, анализируя файл со вторичным модулем, следует использовать ту рабочую библиотеку, в которой находится проанализированный первичный модуль.

Для удобства работы с объектом проекта целесообразно все проанализированные модули проекта, составляющие всю иерархию этого объекта проекта, располагать в одной библиотеке. Возможность создания и использования многих библиотек ресурсов позволяет пользователю классифицировать библиотечные модули по различным признакам. Например, в одной библиотеке хранить описания микросхем одной серии, в другой - описания микросхем другой серии. Или в одной библиотеке хранить описания микросхем с одним типом задержки, в другой - описания микросхем с другим типом задержки. Если в процессе анализа модуля проекта системой VA обнаруживается какая-либо ошибка, то попытка анализа отвергается и в рабочую библиотеку модуль проекта не вносится. Если библиотечный модуль изменяется, например, в результате повторного анализа, то все библиотечные модули, которые потенциально зависят от этого изменения, становятся устаревшими (сообщение об этом можно видеть на экране дисплея при просмотре библиотеки). Это означает, что их необходимо обновить, проанализировав соответствующие модули проекта (из файлов проекта).

Анализатор VA может быть использован в двух режимах:

1) в режиме меню;
2) в командном режиме.

Первый режим предоставляет пользователю более удобный способ общения с системой. Второй режим менее удобен, т.к. все функции системы необходимо вызывать вводом соответствующих команд, и поэтому в данном руководстве не рассматривается. Преимуществом командного режима является то, что он требует меньшего количества оперативной памяти и позволяет интегрировать VA в другие системы.

Для запуска режима меню VA используется команда

VA /r

Параметр /r (если задан) вызывает загрузку драйвера русских букв для клавиатуры и дисплея.

После ввода команды VA на экране терминала будет построено главное меню системы, содержащее 5 функций:

1) “Файл проекта” - операции с файлами проекта;
2) “Редактор” - работа с текстовым редактором;
3) “Анализатор” - анализ VHDL-описаний;
4) “Библиотека” - администратор библиотек проекта;
5) “Выход” - выход из системы.

Выбор функций в этом режиме осуществляется клавишами управления курсором, а запуск функции - нажатием клавиши Enter. Для этой цели также может использоваться манипулятор "мышь". При запуске любой из функций на экране строится соответствующее подменю. Вход в подменю осуществляется тем же способом. Выход из подменю осуществляется нажатием клавиши ESC. Последняя строка экрана используется для вывода подсказок.

Подменю “Файл проекта” содержит 5 функций:

1) “Загрузить” - загрузка файла проекта;

2) “Сохранить” - сохранение файла проекта;
3) “Удалить” - уничтожение файла проекта;
4) “Текущий каталог” - изменение текущего каталога;
5) “Перейти в DOS” - временный выход в DOS.

Функция “Редактор” предназначена для редактирования существующего файла или создания нового. Максимальный размер файла не может превышать 64 Кб.

Если перед вызовом этой функции файл проекта не был загружен, то, по умолчанию, именем редактируемого файла становится WORK.VHD, а сам файл является пустым.

Подменю “Анализатор” содержит три функции:

1) “Анализ” - синтаксический и семантический анализ исходного описания и запись его в промежуточном формате в библиотеку проекта;
2) “Синтаксис” - синтаксический анализ;
3) “Реверсивный анализ” - построение эквивалентного VHDL-описания из промежуточного формата.

Выполнение функции “Анализ” имеет три фазы:

1) синтаксический анализ;
2) семантический анализ;
3) запись описания в промежуточном формате в библиотеку проекта.

При успешном контроле все три фазы выполняются автоматически без вмешательства оператора.

После вызова функции на экране появится окно запроса имени файла для библиотеки проекта. Процедура выбора имени файла полностью совпадает с аналогичной процедурой при загрузке исходного файла проекта, с той лишь разницей, что неявным расширением для имени файла является *.UDL. После того как имя будет выбрано, запускается первая фаза анализа. При обнаружении синтаксической или семантической ошибки анализатор заканчивает свою работу, и система автоматически переключается на редактирование исходного файла. При этом при входе в редактор курсор будет установлен в то место, где была обнаружена ошибка, а нижняя строка окна редактора будет содержать сообщение об ошибке. После исправления ошибки средствами редактора можно продолжать анализ нажатием клавиши F10 или нажатием клавиши ESC прервать процедуру анализа и вернуться в главное меню системы.

Анализ продолжается до тех пор (при условии, что он не был прерван клавишей ESC), пока не будут устранены таким образом все синтаксические и семантические ошибки.

Если в результате анализа ошибки не обнаружены, то система автоматически выполняет сохранение исходного описания в промежуточном формате в заданной библиотеке проекта.

ПРИМЕЧАНИЕ. Прежде чем будет запущен полный анализ VHDL-описания, по крайней мере, одна библиотека проекта должна быть уже создана.

Функция “Синтаксис” используется для проведения только синтаксического анализа VHDL-описания. Выполнение функции соответствует выполне-

нию первой фазы функции “Анализ”. Автоматическое сохранение в библиотеке проекта не выполняется.

Процедура “Реверсивный анализ” позволяет построить из промежуточного формата эквивалентное VHDL-описание.

После запуска на экране появится окно запроса имени файла для библиотеки проекта. После того, как библиотека будет выбрана, на экране появляется меню запроса вида модулей проекта, которые необходимо включить в список для реверсивного анализа. Меню имеет следующие альтернативы:

1) “Все описания” - включить все модули из заданной библиотеки;
2) “Описания объектов” - включить только объявления объектов;
3) “Описания архитектур” - включить только описания архитектур;
4) “Описания конфигураций” - включить только объявления конфигураций;
5) “Описания пакетов” - включить только описания пакетов.

После того, как вид модулей выбран, на экране появляется окно реверсивного анализатора. Окно разбито на четыре столбца, в которые выводится следующая информация:

1) “Имя модуля” - имя модуля проекта;
2) “Вид модуля” - вид модуля;
3) “Сопоставлен с” - имя сопоставленного первичного модуля (если сам модуль является вторичным);
4) “Дата анализа” - дата и время анализа.

Как только очередной модуль проекта будет найден в библиотеке проекта, в указанные столбцы будет занесена соответствующая информация, а в нижней строке окна появится запрос:

Восстановить описание ? Д/Н/В

Если вы хотите выполнить реверсивный анализ найденного модуля, то нажмите клавишу Д, в противном случае нажмите клавишу Н или В.

В первом случае после нажатия клавиши поле “Дата анализа” найденного модуля заменяется на слово "восстановлен", означающее, что этот модуль успешно восстановлен из промежуточного формата в эквивалентное VHDL-описание. Если нажата клавиша Н, реверсивный анализатор займется поиском очередного модуля проекта. Если нажата клавиша В, то работа прекращается.

По окончании на экран выдается статистика, содержащая количество найденных модулей и количество восстановленных модулей.

Все тексты VHDL-описаний, полученные при реверсивном анализе, записываются в текущий каталог в файл VM$REVER.VHD.

Администратор библиотек проекта позволяет выполнить следующие действия:

1) создать новую библиотеку проекта;
2) уничтожить библиотеку проекта;
3) установить внешние ссылки в данной библиотеке на другие библиотеки проекта;

4) разместить новые модули проекта или заменить старые, полученные в результате анализа исходного VHDL-описания (эта функция выполняется неявно только во время анализа);
5) получить список внешних ссылок в данной библиотеке проекта;
6) получить список модулей проекта, содержащихся в данной библиотеке проекта;
7) удалить модули проекта из данной библиотеки проекта;
8) удалить внешние ссылки из данной библиотеки проекта.

Меню администратора строится аналогично главному меню системы и содержит следующие функции:

1) “Библиотека” - операции с библиотеками проекта;
2) “Связи” - операции с внешними ссылками;
3) “Модули” - построение списка модулей проекта;
4) “Удалить” - удаление модулей проекта;
5) “Конвертировать” - преобразовать файл библиотеки проекта из внутреннего формата в формат ASCII;
6) “Выход” - закончить работу “Администратора библиотек”.

Подменю “Библиотека” содержит следующие функции:

1) “Выбрать” - выбрать текущую библиотеку проекта;
2) “Построить” - инициализировать новую библиотеку проекта;
3) “Стереть” - уничтожить библиотеку проекта;
4) “Текущий каталог” - изменить текущий каталог на диске;
5) “Команды DOS” - временно выйти на уровень DOS.

3.3. Демонстрационный пример

Файлы демонстрационного примера находятся в каталоге DEMO. VHDL-описание четырехразрядного сумматора содержится в файле CALC.VHD. Запуск демонстрации осуществляется командным файлом DEMO.BAT. При этом на экране появляется диалоговое окно “Input/Output Window”, содержащее сообщение: “Enter two operands separated by space” (Введите два операнда, разделенные пробелом). В нижней части экрана, в строке состояния появится запрос: “Standart input:”.

После ввода двух десятичных чисел (операндов), разделенных пробелом, и нажатия клавиши <Enter> на экране дополнительно появятся еще два окна: “Watch Window”, отображающее в десятичной форме состояния входных и выходной шин моделируемого сумматора, и “Trace Window”, упрощенно отображающее временные зависимости напряжений на всех четырех разрядах выходной шины. В верхней части экрана появится основное меню.

Для моделирования состояния шин по тактам необходимо навести указатель манипулятора “мышь” на функцию “Step” основного меню. При этом каждое нажатие левой кнопки манипулятора выводит на экран изменения, соответствующие одному такту, а в строке состояния (в нижней части экрана) выводится сообщение “Ready”. После четвертого такта текущий цикл суммирования

завершается, указатель манипулятора “мышь” исчезает, в диалоговом окне “Input/Output Window” вновь появляется запрос: “Enter two operands separated by space”, а в строке состояния: “Standart input:”. После этого необходимо ввести новые числа для суммирования и процесс моделирования повторяется в той же последовательности.

Для выхода из оболочки моделирования служит функция “Quit” основного меню.

Более подробные сведения о демонстрационной версии содержатся в файле README, находящемся в каталоге DEMO.

3.4. Библиотечные модули и пакеты

В каталоге UMB содержатся библиотечные модули и пакеты, которые могут быть использованы как составляющие разрабатываемого VHDL-описания или в качестве наглядных примеров.

4. ПРИМЕР ОПИСАНИЯ ПРОЕКТА НА ЯЗЫКЕ VHDL

4.1. Исходное задание

В качестве примера предлагается описание проекта простого RS-триггера, построенного на двух элементах 2И-НЕ (рис. 2). Подобная схема может быть представлена в трех различных стилях VHDL - поведенческом, потоковом и структурном.

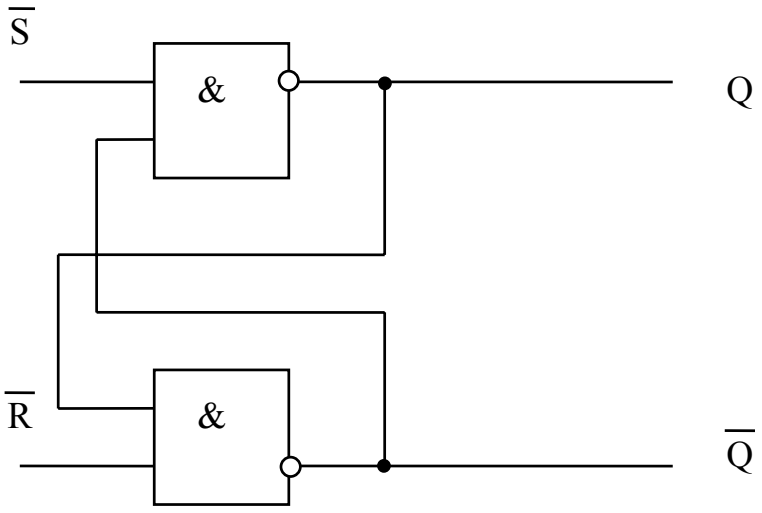

Рис. 2. Схема RS-триггера на элементах 2И-НЕ

Таблица истинности для этой схемы приведена в таблице.

Таблица истинности

S	R	Q(t)	$\overline{Q}$(t)
0	0	запр. комбинация	запр. комбинация
0	1	1	0
1	0	0	1
1	1	Q(t-1)	NQ(t-1)

4.2. Поведенческое описание проекта

Поведенческое описание ассоциируется с последовательно выполнимым кодом процедурного типа. Проект RS-триггера в поведенческом представлении VHDL имеет следующий вид:

```
entity rstr is      --  объявление объекта проекта rstr
port(s,r:inout bit; q,nq:inout bit);  --  список
                    --  портов объекта rstr вида inout,
                    --  типа bit
end rstr;           --  конец объявления объекта rstr

architecture Behavior of rstr is  --  объявление
                    --  архитектуры Behavior объекта rstr
begin               --  начало области предложений
                    --  архитектуры Behavior
      s<='1';       --  моделирование: инициализация
      r<='1';       --  начальных значений входов S и R
      gen1:process  --  моделирование: описание 1-го
            begin   --  формирователя входного сигнала
            s<='0' after 50 ns,'1' after 80 ns;
          wait on s;
          end process;
      gen2:process  --  моделирование: описание 2-го
            begin   --  формирователя входного сигнала
            r<='0' after 20 ns,'1' after 30 ns;
          wait for 120ns;
          end process;
process             --  поведенческое описание логики
   begin            --  работы RS-триггера
    if s='0' and r='1' then q<='1';nq<='0';
    elsif s='1' and r='0' then q<='0';nq<='1';
```

```
    end if;
assert not (s='0' and r='0') --   проверка
report "input error: s,r=0"  --   запрещенного состояния
 severity warning;           --   входов триггера
  wait on s,r;
end process;
end Behavior;       --   конец области объявления
                    --   архитектуры Behavior объекта rstr
```

Функция устройства описана процедурными выражениями с использованием условных операторов.

В архитектуру включен оператор утверждения `assert` для проверки запрещенного состояния входов триггера `s='0'` и `r='0'`. Если в процессе моделирования такое состояние будет обнаружено, появится предупреждающее сообщение.

В архитектуру включено поведенческое описание генераторов импульсов для инициализации входных воздействий при проведении моделирования.

Проект сохраняется в файле `rst.vhd`. Он компилируется, линкуется и направляется на моделирование следующими командами пакета программ VHDL:

```
vhdl.exe source_name
link.exe bin_name obj_name entity_name
sve.exe  bin_name configuration_name
```

где `source_name` - имя файла проекта (`rst.vhd`). Программа `vhdl` создает объектный файл с тем же именем, но с расширением `.oá`. Данный пакет программ воспринимает подобные объектные файлы как библиотечные, считая имя объектного файла (исключая расширение) логическим именем библиотеки.

Программа `link.exe` читает объектный файл `obj_name` (в нашем случае `rst.o`) и создает выполнимый файл `bin_name` без расширения (в нашем случае файл `rst`) для моделирования объекта `entity_name`, объявленного в проекте (в приведенном примере - `rstr`).

Программа `sve.exe` служит для выполнения файла `bin_name` и графического отображения результатов моделирования. В качестве первого аргумента она принимает имя этого бинарного файла, в качестве второго - имя файла конфигурации `configuration_name` (в приводимом примере использовано имя `rst.env`).

Ниже показан пример управляющего командного файла `rst.bat`, выполняющего описанные выше команды:

```
vhdl.exe rst.vhd
link.exe rst rst rstr
```

```
sve.exe  rst rst.env
```

Для настройки параметров моделирующей программы `sve.exe` составляется файл конфигурации `rst.env`:

```
200ns view
10ns  step
s watch
r watch
q watch
nq watch
```

В первой строке командой `view` задается интервал времени, отображаемый на дисплее при моделировании. Во второй строке команда `step` определяет шаг моделирования. В 3-7 строках командами `watch` задаются сигналы (или порты), состояния на которых будут отображаться во время моделирования.

4.3. Структурное описание проекта

Для реализации и последующего моделирования структурного представления проекта RS-триггера необходимо иметь предварительно откомпилированный модуль потокового (или поведенческого) описания компонента, выполняющего функцию 2И-НЕ. Объявление и архитектура данного компонента могут иметь следующий вид:

```
entity noand is  --  объявление объекта проекта noand
 port(a,b:in bit; c:out bit);
end noand;       --   конец объявления объекта noand

architecture DFlow of noand is  --   объявление
 begin           --   архитектуры DFlow  объекта noand
  c<=not(a and b);
end DFlow;
```

В приводимом примере данный компонент сохраняется в отдельном файле `notand.vhd` и компилируется следующей командой:

```
vhdl.exe notand.vhd
```

В результате создается объектный файл `notand.o`, который будет являться библиотечным для приведенного ниже примера.

Проект RS-триггера в структурном представлении имеет вид

```
entity rstr is  --   объявление объекта проекта rstr
  port(s,r:inout bit; q,nq:inout bit); --
```

```
                    --   список портов объекта rstr
                    --   вида inout , типа bit
end rstr;           --   конец объявления объекта rstr

library notand; --   описание библиотеки  notand
use notand.all; --   описание использования
                    --   (фактически - подключение всех
        --   объявленных объектов, имеющихся в notand)
architecture Structure of rstr is    --   объявление
        --   архитектуры Behavior объекта rstr
  component noand  --   объявление компонента типа noand
    port(a,b:in bit; c:out bit);
  end component;
begin               --   начало области предложений
  dd1a:noand port map(s,nq,q); --   создается 1 компонент
                               --   noand
  dd1b:noand port map(r,q,nq); --   создается 2 компонент
                               --   noand
      s<='1';       --   моделирование: инициализация
      r<='1';       --   начальных значений входов S и R
      gen1:process  --   моделирование: описание 1-го
            begin   --   формирователя входного сигнала
            s<='0' after 50 ns,'1' after 80 ns;
          wait on s;
          end process;
      gen2:process  --   моделирование: описание 2-го
            begin   --   формирователя входного сигнала
            r<='0' after 20 ns,'1' after 30 ns;
          wait for 120ns;
          end process;
end Structure;
```

В разделе предложений создается два экземпляра компонента noand. В каждом экземпляре компонента за предложением конкретизации следует предложение карты портов. Каждый элемент в скобках является либо именем одного из портов триггера rstr, либо локально объявленным сигналом. В нашем случае компоненты присоединяются только к портам триггера rstr. Каждая последующая позиция в списке карты портов соответствует локальному порту в той же позиции объявления компонента noand.

Так же, как и в предыдущем примере, проект сохраняется в файле rst.vhd, компилируется, линкуется и моделируется под управлением командного файла rst.bat с параметрами, заданными в файле конфигурации rst.env (см. разд. 4.2).

4.4. Потоковое описание проекта

Потоковое представление проекта можно представить следующим образом:

```
entity rstr is      --  объявление объекта проекта rstr
port(s,r:inout bit; q,nq:inout bit);  --  список
                    --  портов объекта rstr вида inout,
                    --  типа bit
end rstr;           --  конец объявления объекта rstr

architecture DataFlow of rstr is  --  объявление
                    --  архитектуры DataFlow объекта rstr
begin               --  начало области предложений
                    --  архитектуры DataFlow
      s<='1';       --  моделирование: инициализация
      r<='1';       --  начальных значений входов S и R
      gen1:process  --  моделирование: описание 1-го
            begin   --  формирователя входного сигнала
            s<='0' after 50 ns,'1' after 80 ns;
          wait on s;
          end process;
      gen2:process  --  моделирование: описание 2-го
            begin   --  формирователя входного сигнала
            r<='0' after 20 ns,'1' after 30 ns;
          wait for 120ns;
          end process;
assert not (s='0' and r='0') --  проверка
report "input error: s,r=0"  --  запрещенного состояния
severity warning;            --  входов триггера
   q<=not(s and nq);       --  потоковое описание логики
   nq<=not(r and q);       --  работы RS-триггера
end DataFlow;
```

Функция устройства описана одними из наиболее простых предложений параллельного назначения сигналов.

Так же, как и в предыдущих примерах, проект сохраняется в файле `rst.vhd`, компилируется, линкуется и моделируется под управлением командного файла `rst.bat` с параметрами, заданными в файле конфигурации `rst.env` (см. разд. 4.2).

4.5. Проверка описания проекта VHDL-анализатором

Программы VA и VHDL поддерживают различные механизмы работы с библиотеками. Поэтому при проведении синтаксического и семантического контроля рассмотренного структурного описания RS-триггера необходимо предварительно проанализировать модуль с описанием логического элемента 2И-НЕ с занесением его в рабочую библиотеку в соответствии с правилами, приведенными в руководстве к программе VA. Только после этого можно производить контроль структурного описания проекта RS-триггера. При этом строки файла

```
library notand;
use notand.all;
```

становятся ненужными и их необходимо закомментировать во время анализа (комментарием в VHDL считается любая информация после сдвоенного символа `--`). Рабочей библиотекой должна быть та же самая, с которой анализировался модуль с элементом `noand`.

Возможен и другой подход. Для его реализации необходимо также проанализировать файл с описанием элемента 2И-НЕ. Затем необходимо связать библиотеку, в которую помещены результаты анализа `noand`, с логическим именем `notand` (в соответствии с правилами, приведенными в руководстве к программе VA). После этого можно анализировать описание проекта триггера с незакомментированными строками вызова библиотеки, при этом рабочей библиотекой может быть любая существующая библиотека.

Анализ приведенных поведенческого и потокового описаний проекта осуществляется в соответствии с предлагаемыми в руководстве к программе VA рекомендациями и не вызывает затруднений.

4.6. Моделирование проекта

Использование предложенной формы управляющего batch файла позволяет последовательно запустить на компиляцию файл проекта, передать полученный объектный файл линкующей программе, а созданный исполнимый файл загрузить в моделирующую программу `sve.exe`.

Программа `sve.exe` обеспечивает графический интерфейс для визуального наблюдения за контролируемыми сигналами. Управление осуществляется с помощью координатного устройства "мышь". При моделировании с помощью программы `sve.exe` необходимо пользоваться командой Step для пошагового просмотра процесса моделирования. Просмотр осуществляется с шагом, заданным числом наносекунд во второй строке файла `rst.env` (см. разд. 4.2 - `10ns step`). Можно воспользоваться командой Run для вывода наблюдаемых сигналов в желаемом интервале времени. Для этого после выбора команды Run необходимо ввести с клавиатуры длительность необходимого интервала и подтвердить набор нажатием клавиши <Enter>.

5. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ №1

1. Ознакомиться с основными положениями, изложенными в разделах 1 и 2 настоящего руководства.
2. Изучить основы описания проектов средствами языка VHDL, используя электронный резидентный справочник VHDL (см. подраздел 3.1).
3. Изучить работу системы синтаксического и семантического контроля VHDL-описаний (см. подраздел 3.2).
4. Изучить содержание файлов демонстрационного примера и вывести на экран монитора результаты моделирования (см. подраздел 3.3).

6. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ №2

1. Получить у преподавателя схему заданного устройства.
2. Составить таблицу истинности для описываемого устройства (см. подраздел 4.1).
3. Разработать файл поведенческого описания заданной схемы на языке VHDL, файл конфигурации и управляющий файл (см. подраздел 4.2).
4. В соответствии с выполняемой функцией устройства (пользуясь схемой и таблицей истинности) подготовить описание набора входных воздействий, позволяющего наиболее полно отразить функциональные возможности моделируемого устройства.
5. Провести синтаксический и семантический анализ полученного описания с помощью пакета программ VA. Исправить обнаруженные ошибки (см. подраздел 4.5).
6. Провести моделирование заданного устройства на основе разработанного описания в пакете DEMO (см. подраздел 4.6).
7. Отразить результаты моделирования в отчете.

7. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ №3

1. Разработать файл структурного описания схемы, полученной при выполнении лабораторной работы №2, на языке VHDL, файл конфигурации и управляющий файл (см. подраздел 4.3).
2. Провести синтаксический и семантический анализ полученного описания с помощью пакета программ VA. Исправить обнаруженные ошибки (см. подраздел 4.5).
3. Провести моделирование заданного устройства на основе разработанного описания в пакете DEMO (см. подраздел 4.6).
4. Отразить результаты моделирования в отчете.

8. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ №4

1. Разработать файл потокового описания схемы, полученной при выполнении лабораторной работы №2, на языке VHDL, файл конфигурации и управляющий файл (см. подраздел 4.4).
2. Провести синтаксический и семантический анализ полученного описания с помощью пакета программ VA. Исправить обнаруженные ошибки (см. подраздел 4.5).
3. Провести моделирование заданного устройства на основе разработанного описания в пакете DEMO (см. подраздел 4.6).
4. Отразить результаты моделирования в отчете.

9. СОДЕРЖАНИЕ ОТЧЕТОВ

Отчет по лабораторной работе №1 не составляется.

Отчет по лабораторной работе №2 должен содержать:

1. Цель работы.
2. Схему заданного устройства.
3. Текст поведенческого описания устройства с комментариями, оформленными в соответствии с правилами языка VHDL.
4. Графическое отражение результатов моделирования.
5. Выводы по лабораторной работе.

Отчет по лабораторной работе №3 должен содержать:

1. Цель работы.
2. Текст структурного описания устройства с комментариями, оформленными в соответствии с правилами языка VHDL.
3. Графическое отражение результатов моделирования.
4. Выводы по лабораторной работе.

Отчет по лабораторной работе №4 должен содержать:

1. Цель работы.
2. Текст потокового описания устройства с комментариями, оформленными в соответствии с правилами языка VHDL.
3. Графическое отражение результатов моделирования.
4. Выводы по лабораторной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие задачи решаются с использованием языка VHDL?
2. Какие существуют модули проекта?
3. Какие из модулей проекта относятся к первичным, вторичным?
4. Чем отличается рабочая библиотека от библиотеки ресурсов?

5. Для чего используются пакеты (PACKAGE) в VHDL?
6. Какие пакеты являются стандартными для VHDL?
7. Что может содержаться в описании интерфейса объекта проекта?
8. Что указывается при описании портов в интерфейсе объекта проекта?
9. Как оформляется комментарий в языке VHDL ?
10. Какие модели данных поддерживает VHDL?
11. Перечислить скалярные типы.
12. Пользовательские типы и подтипы.
13. Массивы.
14. Записи.
15. Ссылочные типы и динамические объекты.
16. Перечислить и охарактеризовать модели задержки в VHDL.
17. Привести примеры генераторов импульсов.
18. Что такое драйвер сигнала?
19. Как правильно описать работу линии задержки с бесконечно большой полосой пропускания? Привести пример соответствующего назначения сигнала.

ЛИТЕРАТУРА

1. Проектирование СБИС: Пер. с япон./Ватанабэ М., Асада К., Кани К., Оцуки Т. М.: Мир, 1988. 304 с.
2. Ульман Дж. Вычислительные аспекты СБИС: Пер. с англ. /Под ред. П.П.Пархоменко. М.: Радио и связь, 1990. 480 с.
3. Киносита К., Асада К., Карацу О. Логическое проектирование СБИС: Пер. с япон. М.: Мир, 1988. 309 с.
4. Резидентный справочник по языку VHDL. Руководство пользователя. М.: РосНИИИС, 1993. 21 с.
5. Система синтаксического и семантического контроля VHDL-описаний “VHDL-анализатор”: Руководство пользователя. М.: РосНИИИС, 1991. 112 с.

Коноплев Борис Георгиевич
Рындин Евгений Адальбертович
Ивченко Владимир Геннадьевич

Методическое руководство

Описание проектов СБИС с использованием языка VHDL

Ответственный за выпуск Рындин Е.А.
Редактор Васютина О.Н.
Корректор Проценко И.А.

ЛР 0205665
Бумага офсетная
Уч. -изд. л. - 1,7
Заказ № 182

Подписано к печати
Печать офсетная
Печ. л. - 1,8
Тираж 150 экз.

“С”

Издательство Таганрогского государственного
радиотехнического университета
347928, Таганрог, ГСП-17А, пер. Некрасовский, 44
Типография Таганрогского государственного
радиотехнического университета
347928, Таганрог, ГСП-17А, ул. Энгельса, 1