Оглавление

Принятые сокращения

АЛУ — арифметико-логическое устройство.
ОПО — оператор параллельной обработки.
ПО — программное обеспечение.
САПР — система автоматизированного проектирования.
СБИС — сверхбольшая интегральная схема.
ЦУ — цифровое устройство.
ПУЗС — простая установка значения сигнала.
УУЗС — условная установка значения сигнала.
СУЗС — селективная установка значения сигнала.

ГЛАВА 1

ВВЕДЕНИЕ

1.1. Современные языки проектирования цифровых устройств

VHDL является аббревиатурой от Very high speed integrated circuits Hardware Description Language, что переводится как *язык описания устройств на сверхбольших интегральных схемах* (СБИС). В середине 1980-х гг. Министерство обороны США и IEEE[1)] спонсировали разработку этого языка описания цифровой аппаратуры с целью получения простого в использовании средства проектирования и моделирования логических схем для всех этапов разработки электронных систем, начиная от модулей микросхем и кончая крупными вычислительными системами. Первая версия стандарта была издана в 1987 г. (IEEE 1076-1987). Очередные версии выходили в 1991, 1993, 1996, 1997, 1999, 2000 и 2002 гг. В настоящее время действует стандарт VHDL, изложенный в документе IEEE 1076-2002 и являющийся промышленным стандартом, который широко используется для описания работы цифровых систем. В июне 2006 г. была опубликована версия 3.0 проекта стандарта VHDL-2006, в который вошли все дочерние стандарты, разработанные в ходе создания стандартов VHDL (IEEE

[1)] IEEE (Institute of Electrical and Electronic Engineers — Институт инженеров по электротехнике и радиоэлектронике, ИИЭР) — международная организация, созданная в США в 1963 г. Является разработчиком ряда стандартов для локальных вычислительных систем, в том числе по кабельной системе, физической топологии и методам доступа к среде передачи данных. — *Примеч. ред.*

1064, 1076.2, 1076.3), а также добавлены другие усовершенствования, такие как интерфейс с языками высокого уровня C/C++ и ряд других.

В РФ язык VHDL закреплен стандартом ГОСТ РФ 50754-95 «Язык описания аппаратуры цифровых систем VHDL. Описание языка».

В данной книге рассмотрены основные положения языка VHDL, которые определены в стандарте IEEE 1076-1993, поскольку последующие изменения (за исключением пока еще разрабатываемого стандарта VHDL-2006) были несущественными. Кроме того, все последующие стандарты VHDL, включая VHDL-2006, обеспечивают совместимость с проектами, разработанными в соответствии со стандартом 1076-1993.

Еще одним представителем языков описания цифровой аппаратуры является язык Verilog, или Verilog HDL. Разработчики Verilog сделали его синтаксис очень похожим на синтаксис языка C. Verilog имеет препроцессор, очень похожий на препроцессор языка C, а основные управляющие конструкции Verilog также подобны одноименным конструкциям языка C. В настоящее время действует стандарт Verilog, изложенный в документе IEEE 1364-2005, который представляет собой несколько доработанный вариант очень популярного среди приверженцев Verilog стандарта IEEE 1364-2001. Язык Verilog применяется в промышленности так же широко, как и VHDL, поскольку оба этих языка позволяют описывать и имитировать работу сложных цифровых систем.

Третьим представителем языков VHDL является язык ABEL (Advanced Boolean Equation Language — расширенный язык двоичных уравнений), который был разработан в 1983 г. для создания проектов цифровых устройств посредством *программируемых логических устройств* (PLD — Programmable Logic Devices). Язык ABEL менее мощен, чем VHDL и Verilog, в частности не позволяет проектировать устройства на перепрограммируемых логических матрицах FPGA (Field Programmable Gate Arrays), поэтому он менее популярен в промышленности, хотя все еще находит своих сторонников.

1.2. Краткая характеристика языка VHDL

Хотя язык VHDL внешне выглядит так же, как и другие традиционные языки программирования (поскольку имеет литералы, разделители, операторы и т. д.), он обладает некоторыми важными отличительными характеристиками.

- Проекты цифровых устройств (ЦУ), созданные с помощью языка VHDL, имеют, как правило, иерархическую структуру.
- Каждый автономный проектируемый модуль (субблок проектируемого ЦУ) имеет:
  — строго определенный интерфейс взаимодействия с другими модулями;
  — точную спецификацию внутреннего устройства проектируемого модуля, описывающую концепцию и функционирование модуля.
- Спецификации модулей VHDL-проектов могут использовать или математические алгоритмы, описывающие их работу, или описание аппаратной структуры проектируемого модуля. В соответствии с уровнями абстракций проектов описание модуля может иметь *поведенческую* или *структурную* форму.
- Моделирование алгоритма работы проекта основывается на событийном принципе управления.
- VHDL-проект позволяет выполнять моделирование протекания параллельных процессов в электрических схемах, временной анализ сигналов и их параметров.
- VHDL поддерживается инструментальными средствами синтеза и системами автоматизированного проектирования (САПР) многих производителей программного обеспечения (ПО), которые могут создавать прямо из описания VHDL-проекта его аппаратную реализацию (связанные между собой структуры логических элементов, содержащихся в СБИС).
- Используя VHDL, можно проектировать, моделировать и синтезировать практически любое ЦУ, начиная от простой комбинационной схемы до законченной микропроцессорной системы на СБИС.

Перечисленные выше характеристики языка VHDL как специализированного языка описания ЦУ реализуются с помощью следующих языковых средств:

- Библиотеки и пакеты.
- Проекты: интерфейс и архитектура проекта.
- Подпрограммы: функции и процедуры.
- Скалярные типы данных: перечислимые, числовые, физические.
- Программные элементы данных: константы, переменные, сигналы, порты, идентификаторы.

- Математические операции: логические, отношений, арифметические.
- Программные операции: установка значений сигналов, присвоение значений переменным, реализация связи портов и сигналов.
- Математические выражения: логические, алгебраические, логико-алгебраические.
- Операторы объявления программных элементов данных.
- Операторы комбинаторной логики: простой установки значения сигнала (ПУЗС), условной установки значения сигнала (УУЗС), селективной установки значения сигнала (СУЗС), оператор **`process`**, оператор реализации компонента **`port map`**.
- Операторы регистровой логики: оператор **`process`**, ПУЗС, оператор условной передачи управления, оператор цикла, оператор выбора.

В отличие от процедурных языков программирования, языковые средства которых обеспечивают выполнение вычислений над абстрактными данными и управление ими, VHDL-проект описывает ЦУ, учитывая его многогранность, поведение, структуру, функциональные и физические свойства, а также взаимодействие со специальной аппаратурой, физически реализующей проект ЦУ в СБИС.

Остановимся особо на некоторых важных отличиях языка VHDL. VHDL по существу является языком *параллельного программирования*, т. е. в его конструкции существуют операторы, соответствующие логическим вентилям. Эти операторы обрабатываются (т. е. вычисляются) по *параллельному принципу.* Суть данного принципа состоит в том, что, как только сигнал, содержащийся в описании проектируемого ЦУ, изменяет свое значение (говорят, что «происходит событие на сигнале»), все операторы, принимающие участие в его обслуживании, мгновенно запускаются на выполнение и одновременно выдают конечный результат. Поэтому такие операторы называются *операторами параллельной обработки* (ОПО) (concurence operator). Программа, написанная на VHDL (как и на любом другом HDL-языке, например Verilog), моделирует физическое поведение системы (как правило, цифровой), сигналы в которой распространяются мгновенно. Такая программа позволяет формировать временную спецификацию (время задержки распространения сигнала на логическом элементе), а также описывать систему как соединение различного рода компонентов, или функциональных блоков.

1.3. Этапы проектирования с использованием VHDL

В подавляющем большинстве случаев аппаратная реализация проекта ЦУ с использованием VHDL протекает в соответствии со следующими этапами:

Разработка иерархической блок-схемы проекта. Выяснение базового конструктивно-технологического метода и стандартных блоков на уровне структурной схемы. Поскольку большие логические проекты являются, как правило, иерархическими, использование VHDL позволяет легко разбить проект на модули (субпроекты) и определить их интерфейсы.

Программирование. Запись VHDL-кода для модулей и их интерфейсов.

Компиляция. Анализ программного кода VHDL-проекта для выявления синтаксических ошибок, а также проверка его совместимости с другими модулями. В ходе компиляции также собирается внутренняя информация о структуре проекта, которая необходима для моделирования работы проектируемого ЦУ.

Моделирование. Определение и применение входных воздействий к откомпилированному коду проекта с наблюдением выходных реакций. Моделирование может выполняться как в форме *функционального контроля*, т. е. проверки логики работы проекта без учета временных соотношений и задержек распространения сигналов на логических элементах, так и в качестве одного из этапов *верификации* завершенного проекта.

Синтез. Преобразование VHDL-описания в набор примитивов или логических элементов, которые могут быть реализованы с учетом конкретной технологии.

Компоновка, монтаж и разводка. Отображение проекта на карте синтезирующих элементов, содержащихся в СБИС.

Временной анализ. Получение фактических задержек реализованной в СБИС цифровой схемы проекта с учетом длины соединений, электрических нагрузок и других известных факторов.

ГЛАВА 2

УРОВНИ ПРЕДСТАВЛЕНИЯ И ФОРМЫ АБСТРАКЦИИ ЦИФРОВЫХ СИСТЕМ

Цифровая система на языке VHDL может быть представлена на различных уровнях и различными формами абстракции. Представление цифровой системы на нескольких уровнях абстракции показано на **Рис. 2.1**.

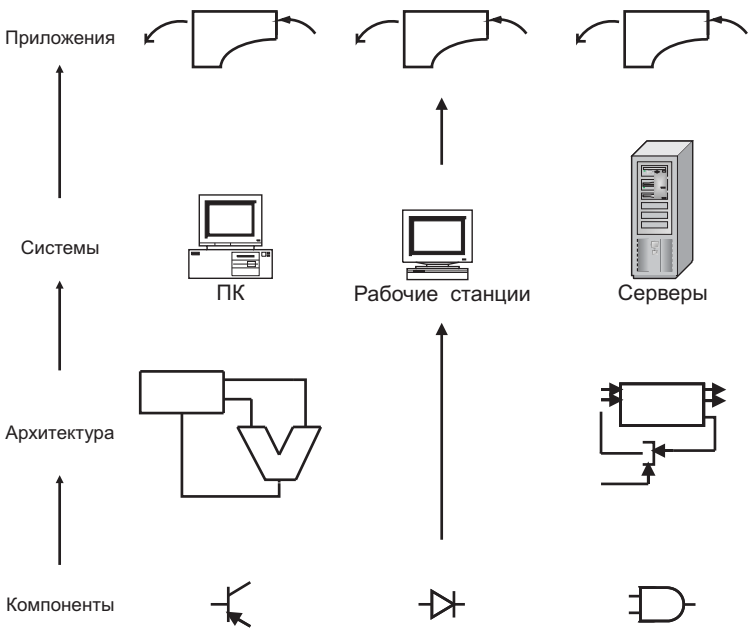

Рис. 2.1. Представление цифровой системы на нескольких уровнях абстракции

Как видно на **Рис. 2.1**, цифровая система может быть описана на уровне компонентов (транзисторов, диодов, логических элементов), на уровне архитектуры (структурной схемы, содержащей АЛУ, регистры, компараторы и т. д.), на уровне автономной системы (ПК, рабочей станции, сервера и т. д.), а также на уровне приложений (программных модулей, входящих в состав систем более высокого уровня).

Различные формы абстракции цифровой системы дают возможность сохранять описание и проект как комплексную управляемую систему. На **Рис. 2.2** показаны различные формы абстракции.

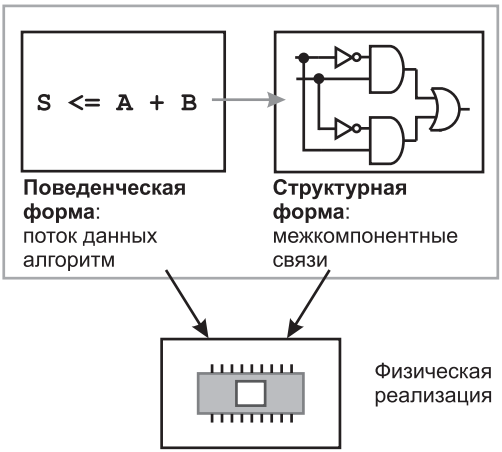

Рис. 2.2. Формы абстракции: поведенческая, структурная и физическая

Первичная и наивысшая форма абстракции — это *поведенческая* (behavioral) форма, которая позволяет описывать систему в терминах того, что она делает (или как она ведет себя), а не в терминах ее элементов либо компонентов и соединений между ними. Поведенческая форма представления определяет зависимость между входными и выходными сигналами. Поведенческая форма может быть булевым (Boolean) выражением либо более абстрактным описанием, например описанием *межрегистровых пересылок* (register transfer) или описанием в *алгоритмической* форме.

В качестве примера представления поведенческой формы рассмотрим простое устройство `SIREN`, которое выдает сигнал предупреждения (`Warning`) всякий раз, когда автомобильный ключ вставлен в замок зажигания (`Ignition_on`), если дверь открыта (`Door_open`) или отстегнут ремень безопасности (`Seatbelt_off`). На поведенческом уровне это словесное описание может быть выражено как:

```
Warning = Ignition_on and (Door_open or Seatbelt_off)
```

Структурная (structural) форма, с другой стороны, представляет систему как набор логических элементов и компонентов, которые связаны между собой таким образом, чтобы выполнить нужную функцию. Структурную форму представления можно сравнить со схемным решением связанных логических вентилей. Структурный уровень является обычно окончательным представлением физической реализации системы. Структурное представление рассмотренного выше примера показано на **Рис. 2.3**.

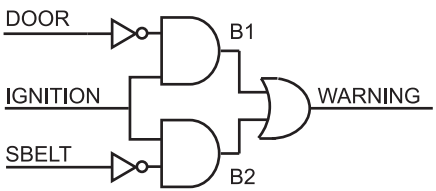

Рис. 2.3. Структурное представление устройства `SIREN`

Как говорилось выше, язык VHDL позволяет описать цифровую систему и в *структурной*, и в *поведенческой* форме. При этом поведенческая форма может быть реализована в одном из двух стилей: *в виде потока данных* (data flow style) и *в алгоритмическом виде* (algoritmic style).

Представление системы в виде потока данных позволяет описать систему с учетом направления потока перемещаемых через нее данных. Этот стиль характерен для описания поведения системы на уровне межрегистровых пересылок.

При представлении системы в алгоритмическом виде поведение системы описывается с помощью *операторов* (operator). При этом для описания поведения системы могут использоваться как операторы *параллельной обработки* (concurrent), которые выполняются параллельно,

как только данные поступают на входы, так и *последовательные* (sequential) операторы, которые выполняются последовательно в порядке их записи. Язык VHDL допускает как параллельную, так и последовательную установку значений сигналов, которые определяют способ и порядок их выполнения. Примеры обоих представлений будут даны в последующих главах книги.

ГЛАВА 3

БАЗОВАЯ СТРУКТУРА VHDL-ФАЙЛА

3.1. Общие сведения

Проект любого ЦУ на языке VHDL — это прежде всего программа, которая содержит *ключевые* и *зарезервированные* слова. Эти слова не могут использоваться как имена сигналов или как идентификаторы. В языке VHDL ключевые слова, зарезервированные слова и определяемые пользователем идентификаторы нечувствительны к регистру.

Строки с комментариями в VHDL-программе начинаются с двух смежных дефисов (`--`) и при компиляции игнорируются компилятором так же, как символы конца строки и пробелы.

VHDL — это *строго типизированный* язык. Это означает, что все программные элементы данных (константа, переменная, сигнал) должны явно объявляться с обязательным указанием типа элемента. Тип, указанный в объявлении программного элемента данных, определяет информационные характеристики этого элемента, диапазон допустимых числовых значений, которые может принимать этот программный элемент, а также операции, которые могут над ним выполняться. Кроме того, в языке VHDL, в отличие от других строго типизированных языков, *не допускаются* операции над разнотипными элементами без предварительного преобразования типов.

В языке VHDL (как и в классических языках программирования) из ключевых, зарезервированных слов и других лексических элементов строятся образования, которые принято называть *операторами*. Оператором в языке VHDL считается любая запись, начинающаяся с ключевого слова и заканчивающаяся символом точки с запятой (`;`). Диаграмма, иллюстрирующая одну из возможных классификаций операторов, показана на **Рис. 3.1**.

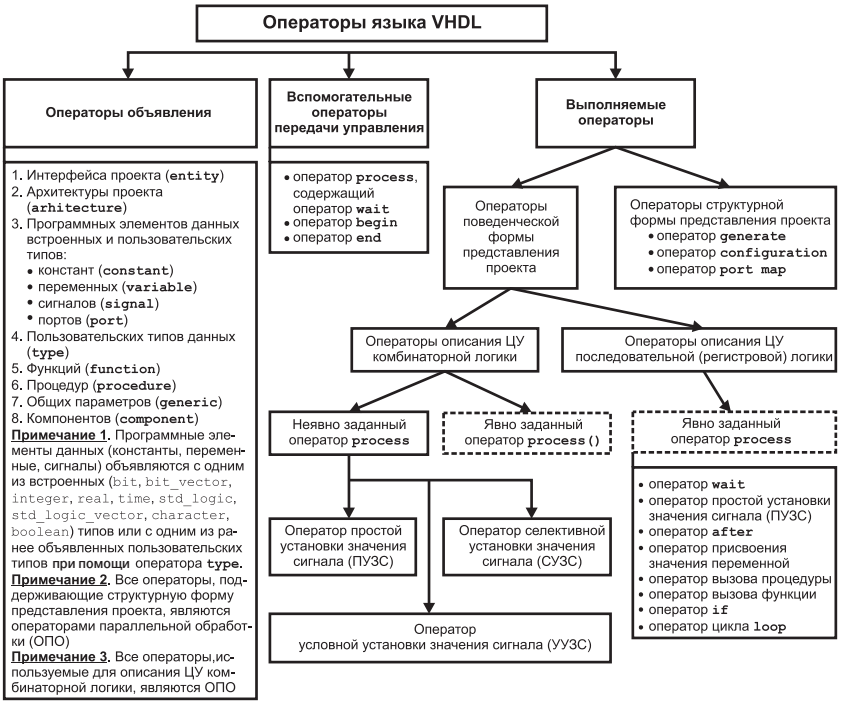

Рис. 3.1. Классификация операторов языка VHDL

В языке VHDL операторы с точки зрения их функционального программного назначения (как при поведенческой, так и при структурной методологии описания проектов) принято делить на две основные группы:

- *Операторы объявления* программных элементов данных и их типов, компонентов проектируемого ЦУ, их интерфейсов и архитектур, пользовательских типов, а также функций и процедур.
- *Выполняемые операторы*.

Кроме основных операторов в языке VHDL принято выделять еще одну группу операторов, которую называют *вспомогательными операторами передачи управления*.

С точки зрения конструктивного устройства операторы обеих групп принято подразделять на два класса:

- **Простые (однострочные) операторы**. Простой оператор представляет собой текстовую запись, располагающуюся, как правило, на одной строке, которая *не содержит* других операторов.
- **Составные (многострочные) операторы.** Составной оператор представляет собой текстовую запись, занимающую, как правило, много строк, которая *содержит* другие операторы.

В структуре языка VHDL имеется два фундаментальных оператора (составные по своему конструктивному устройству), которые поддерживают *принцип системного проектирования*. Принцип системного проектирования предполагает, что любое проектируемое ЦУ рассматривается как *автономная подсистема*, которая должна иметь:

- идентификатор (т. е. должна быть объявлена);
- способность взаимодействовать с другими проектируемыми подсистемами посредством своего интерфейса (входных/выходных портов);
- описание внутренней структуры или алгоритма функционирования.

Такими фундаментальными операторами в языке VHDL являются операторы:

```
entity ... end entity ...;
architecture ... end architecture ...;
```

Проект ЦУ, или его компонент, описанный с помощью языка VHDL, хранится в файле, который обычно снабжается расширением VHD.

Представленный в виде VHD-файла проект ЦУ может содержать другие объекты (подсистемы), которые в таком случае являются подчиненными компонентами системы верхнего уровня.

Любой компонент, независимо от того, является ли он автономным либо подчиненным компонентом, или же системой верхнего уровня, в свою очередь является совокупностью *интерфейса* (**`entity`**) и *архитектуры* (**`architecture`**).

В *объявлении интерфейса* (**`entity`** declaration) содержится *объявление портов* (ports declaration) проектируемого компонента с внешним миром. Объявление портов проектируемого компонента определяет

внешние входные и выходные интерфейсные сигналы, в то время как архитектура представляет собой набор таких взаимосвязанных программных элементов, как подчиненные компоненты, операторы **process**, операторы параллельных вычислений, последовательные операторы, подпрограммы (**Рис. 3.2**). В типичном проекте содержится несколько таких взаимосвязанных объектов, предназначенных для выполнения преобразований, в ходе которых и обеспечивается собственно требуемая функциональность проекта.

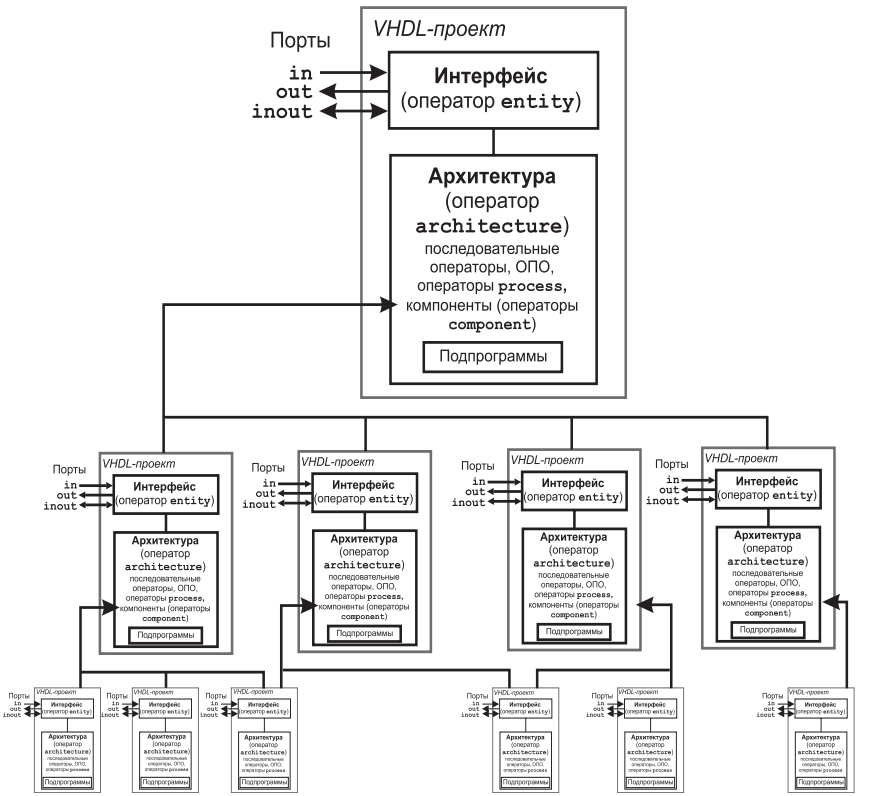

Рис. 3.2. VHDL-проект, представляя собой совокупность интерфейса и архитектуры, может использовать субпроекты в качестве компонентов, а также входить в качестве компонента в VHDL-проекты более высокого уровня иерархии