Лабораторная работа 1

Синтаксические конструкции

Для программирования ПЛИС (т.е. для реализации цифровых устройств на ПЛИС, ПЛИС – программируемая логическая интегральная схема, FPGA – field programmable gate array) разработаны специальные языки программирования описательного характера: HDL (Hardware Description Language). К таким языкам относятся VHDL, Verilog, SystemVerilog. Основным отличием HDL языков от языков типа C, C++, C#, Pascal, Basic является направленность первых на параллельное программирование. Также HDL языки позволяют оперировать интервалами времени и описывать системы в виде соединения подсистем.

Программирование на HDL языках можно сравнить с формализованным созданием печатной платы (а точнее выбором и размещением на ней компонентов), когда с помощью языка программирования описываются компоненты (резисторы, конденсаторы, кварцевые генераторы, процессоры, реле и т.д.) на плате, их размещение и соединение, а также алгоритмы работы отдельных (сложных) компонентов. Более подробно основные подходы при программировании на HDL языках будут рассмотрены в следующих лабораторных работах.

Одним из наиболее широко используемых описательных языков программирования является VHDL. Расшифровывается как **V**HSIC (Very High Speed Integrated Circuits) **H**ardware **D**escription **L**anguage.

Entity: структура и синтаксис

Цифровое устройство, описываемое VHDL, называется **entity** (модуль, блок, дословно – “сущность”). Entity – это одно из ключевых слов языка. Модуль VHDL (entity) состоит из двух частей: объявления интерфейса (entity declaration) и архитектуры (architecture body). Модули могут содержать в себе другие модули.

Объявление интерфейса

Объявление интерфейса модуля VHDL выглядит следующим образом:

```
entity name_of_entity is [generic (generic_declarations);]
      [port (signal_names: mode type;
             signal_names: mode type;
                  :
            signal_names: mode type);]
end [name_of_entity];
```

Полужирным текстом написаны ключевые слова, в квадратных скобках – необязательная часть. Объявление интерфейса всегда начинается с ключевого слова **entity**, после которого следует название модуля и ключевое слово **is**. Далее следует объявление параметров компиляции (**generic**) и входных и выходных портов (**port**) устройства.

Объявление портов может отсутствовать, например, в случае, когда разрабатываемый модуль представляет собой тестер.

Важно! Объявление последнего порта не заканчивается точкой с запятой. Точка с запятой ставится после закрывающей круглой скобки (выделено красным цветом).

При объявлении входных и выходных портов необходимо указать их тип данных и режим работы. Существуют следующие режимы работы:

- in – сигнал является входным портом;
- out – сигнал является выходным портом;

– inout – сигнал является двунаправленным портом;
– buffer.

В настоящей лабораторной работе необходимо использовать тип данных порта **std_logic**. Этот тип определяет одноразрядные данные, которые могут принимать следующие значения: 0, 1, X (неизвестно), U (неинициализировано), - (не важно), Z (высокий импеданс), L (слабый 0), H (слабая 1), W (слабое неизвестное). Подробная информация по возможным типам данных будет дана в следующих лабораторных работах.

Архитектура

В архитектуре описывается собственно логическая функция, которая реализуется данным цифровым устройством. Это может быть как простой сумматор, так и полноценное микропроцессорное ядро. Для описания цифрового устройства в архитектуре можно использовать структурный или поведенческий подходы, а также их комбинацию. Сущность и основные отличия этих подходов будут раскрыты в следующих лабораторных работах.

Описание архитектуры модуля VHDL выглядит следующим образом:

```
architecture architecture_name of NAME_OF_ENTITY is
[Declarations]
begin
[Statements]
end architecture_name;
```

Описание архитектуры начинается с ключевого слова **architecture,** за которым следует название архитектуры, ключевое слово **of**, название модуля, для которого предназначена данная архитектура, и ключевое слово **is**. Далее могут идти различные объявления (сигналов, функций, процедур, других модулей, констант, типов данных). Между ключевыми словами **begin** и **end** размещается реализуемый функционал.

*.XDC и *.UCF файлы

Разработанное цифровое устройство (описанное на VHDL) внешнему миру представлено как некий “черный” ящик с входами и выходами. Для того, чтобы создать прошивку и, в конечном итоге, разместить реализованный модуль в ПЛИС, необходимо привязать все порты разработанного модуля к определенным ножкам ПЛИС. Эта операция может быть выполнена как в объявлении интерфейса, так и с помощью отдельного файла с расширением XDC (Vivado) или UCF. Для привязки портов к ножкам ПЛИС и указания режимов их работы используются специальные команды. Пример *.XDC и *.UCF файлов, которые будет использоваться в данной лабораторной работе, представлены в задании к лабораторной работе 1.

Задание к лабораторной работе 1

Реализовать различные логические вентили в ПЛИС. В качестве входов логических вентилей использовать слайдеры на плате, в качестве выходов – светодиодные индикаторы.

Параметры используемых ПЛИС, необходимые для корректного создания проекта в Vivado, представлены в таблице 1.

Таблица 1. Параметры ПЛИС

	Artix 7
Family	Artix 7
Device	XC7A35T
Package	CPG236

Для этого необходимо выполнить следующие действия (см. *“Работа с Vivado.pdf”*).

1. Создать проект в среде Vivado (для ПЛИС Artix7). В процессе создания проекта указать семейство и тип используемой ПЛИС, а также язык программирования VHDL.

2. Написать enity на языке VHDL, в котором реализованы двухвходовые логические вентили (см. пример ниже). Цифровое устройство должно иметь два входа и семь выходов, каждый выход должен выводить результат соответствующей логической операции.

3. Написать файл *.xdc, в котором входные порты разработанного устройства подключены к двум слайдерам демонстрационной платы, а выходы – к семи светодиодным индикаторам (см. пример ниже).

4. Синтезировать проект. Открыть и сохранить схему синтезированного устройства.

5. Сгенерировать прошивку для ПЛИС. Открыть и сохранить схему разведенного устройства.

6. Загрузить сгенерированную прошивку в ПЛИС, убедиться в работоспособности программы.

7. Составить общую таблицу истинности для разработанного цифрового устройства (два входа, семь выходов).

8. Выполнить автоматическую проверку разработанного модуля (см. *“Автоматическая проверка результатов лабораторной работы.pdf”*).

Исходный код реализуемого модуля

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY basic_gates IS
     PORT (a, b: IN STD_LOGIC;
           y_and, y_or, y_xor, y_not,
           y_nand, y_nor, y_xnor:
                OUT STD_LOGIC);
END ENTITY;

ARCHITECTURE rtl OF basic_gates IS
BEGIN
     y_and <= a AND b;
     y_or <= a OR b;
     y_xor <= a XOR b;
     y_not <= NOT a;
     y_nand <= a NAND b;
     y_nor <= a NOR b;
     y_xnor <= a XNOR b;
END ARCHITECTURE;
```

Вариант XDC файла для Artix A7 35

```
## Switches
set_property PACKAGE_PIN V17 [get_ports {a}]
      set_property IOSTANDARD LVCMOS33 [get_ports {a}]
set_property PACKAGE_PIN V16 [get_ports {b}]
      set_property IOSTANDARD LVCMOS33 [get_ports {b}]

## LEDs
set_property PACKAGE_PIN U16 [get_ports {y_and}]
      set_property IOSTANDARD LVCMOS33 [get_ports {y_and}]
set_property PACKAGE_PIN E19 [get_ports {y_or}]
      set_property IOSTANDARD LVCMOS33 [get_ports {y_or}]
set_property PACKAGE_PIN U19 [get_ports {y_xor}]
      set_property IOSTANDARD LVCMOS33 [get_ports {y_xor}]
set_property PACKAGE_PIN V19 [get_ports {y_not}]
      set_property IOSTANDARD LVCMOS33 [get_ports {y_not}]
```

```
set_property PACKAGE_PIN W18 [get_ports {y_nand}]
      set_property IOSTANDARD LVCMOS33 [get_ports {y_nand}]
set_property PACKAGE_PIN U15 [get_ports {y_nor}]
      set_property IOSTANDARD LVCMOS33 [get_ports {y_nor}]
set_property PACKAGE_PIN U14 [get_ports {y_xnor}]
      set_property IOSTANDARD LVCMOS33 [get_ports {y_xnor}]
```