Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Воронежский государственный технический университет»

Кафедра конструирования и производства радиоаппаратуры

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для проведения практических занятий

по дисциплине «Проектирование цифровых сложнофункциональных блоков» направления 11.04.03 «Конструирование и технология электронных средств», магистерская программа «Автоматизированное проектирование и технология радиоэлектронных средств специального назначения»

по дисциплине «Методы проектирования сложнофункциональных блоков электронных средств» направления 11.04.03 «Конструирование и технология электронных средств», магистерская программа «Силовая электроника»

по дисциплине «Специализированные интегральные устройства и системы в приборостроении» направления 12.04.01 «Приборостроение», магистерская программа «Автоматизированное проектирование приборов и комплексов»

Воронеж 2022

УДК 621.3.049.7.002 (075)
ББК 38.54

Составители:
канд. техн. наук А.А. Пирогов

Методические указания для проведения практических занятий по дисциплине «Проектирование цифровых сложнофункциональных блоков» направления 11.04.03 «Конструирование и технология электронных средств», магистерская программа «Автоматизированное проектирование и технология радиоэлектронных средств специального назначения», по дисциплине «Методы проектирования сложнофункциональных блоков электронных средств» направления 11.04.03 «Конструирование и технология электронных средств», магистерская программа «Силовая электроника», по дисциплине «Специализированные интегральные устройства и системы в приборостроении» направления 12.04.01 «Приборостроение», магистерская программа «Автоматизированное проектирование приборов и комплексов» / ФГБОУ ВПО «Воронежский государственный технический университет»; сост. А.А. Пирогов. Воронеж, 2022. 21 с.

В методических указаниях изложены требования и рекомендации по подготовке и выполнению практических заданий в среде проектирования Xilinx ISE.

Методические указания подготовлены в электронном виде и содержатся в файле Мет_СФ.pdf.

Ил. 9. Библиогр.: 4 назв.

УДК 621.3.049.7.002 (075)
ББК 38.54

Рецензент - О. Ю. Макаров, д-р техн. наук, проф. кафедры конструирования и производства радиоаппаратуры ВГТУ

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ В СРЕДЕ XILINX ISE

Интегрированная система автоматизированного проектирования Integrated Software Environment (САПР ISE) является основным продуктом сквозного проектирования цифровых систем на базе программируемых логических интегральных схем (ПЛИС) фирмы Xilinx. Данный программный продукт поддерживает все выпускаемые ПЛИС и имеет несколько вариантов поставки. В методических указаниях представлены основные сведения о порядке работы и способе построения проектов в программном комплексе Xilinx ISE.

Цель занятий является практическая реализация моделей полученных в ходе лабораторного практикума по дисциплине «Специализированные БИС и устройства функциональной электроники в приборостроении», формирование прошивки на их основе и запуск с использованием отладочных плат ПЛИС Digilent Basys 2.

Задание 1. Основные этапы создания проекта Xilinx ISE. Программирование отладочной платы ПЛИС

Ознакомимся с основными принципами и методами проектирования с использованием программного комплекса Xilinx ISE для отладочных плат Digilent Basys 2. Необходимо построить проект позволяющий проверить работу базовых булевых операторов *«not, and, or, nand»*. Рассмотрим **основные этапы** создания проекта на примере данного задания.

Запуск среды проектирования Xilinx ISE. На появившемся экране нажать кнопку «New Project», в появившемся диалоге «New Project Wizard» ввести имя проекта «basic_boolean», нажать «Next» (см. рис. 1).

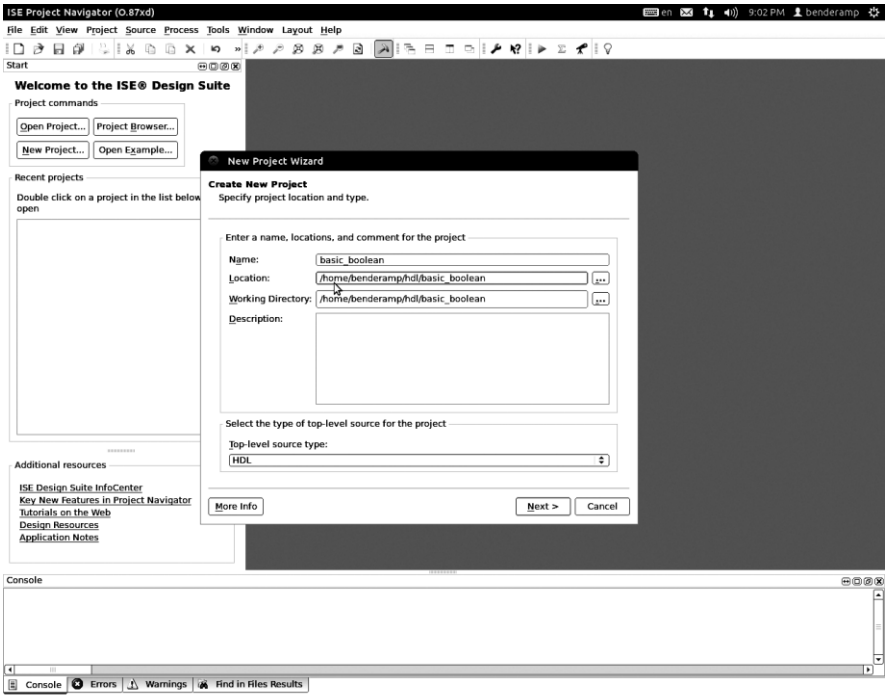

Рис. 1. Диалоговое окно редактора Xilinx ISE. Создание проекта

На следующем экране «Specify device and project properties» указать следующие настройки для отладочной платы ПЛИС Digilent Basys 2:

Product Category: All
Family: Spartain3E
Device: XC3S250E
Package: CP132
Speed: -5

Далее следует экран со сводкой по новому проекту – нажать «Finish».

Создание файла с исходным кодом на языке описания аппаратуры Verilog. Выбираем меню «Project > New Source...», в открывшемся диалоге «New Source Wizard» выбираем в списке тип файла исходного кода (Source Type) «Verilog Module», вводим имя создаваемого файла в поле «File name»: «basic_boolean», нажимаем кнопку «Next» (см. рис. 2).

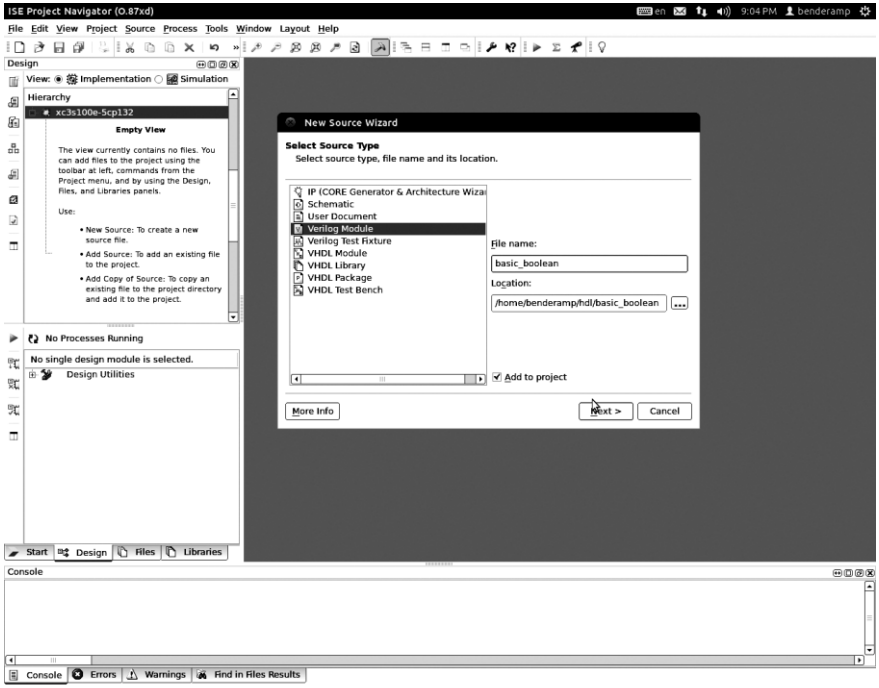

Рис. 2. Диалоговое окно редактора Xilinx ISE. Создание файла проекта с исходным кодом

Следующее диалоговое окно («Specify ports for module») осуществляет определение портов ввода и вывода для модуля, в данном случае определим их вручную, нажимаем «Next», следующая сводка информации по новому файлу – «Finish».

Получен новый файл с исходными кодами модуля на языке *Verilog*. Часть кода комментариев и директивы «timescale 1ns / 1ps» можно удалить, оставить только главное определение (пока пустого) модуля (см. рис. 3):

```
module basic_boolean(
);
endmodule.
```

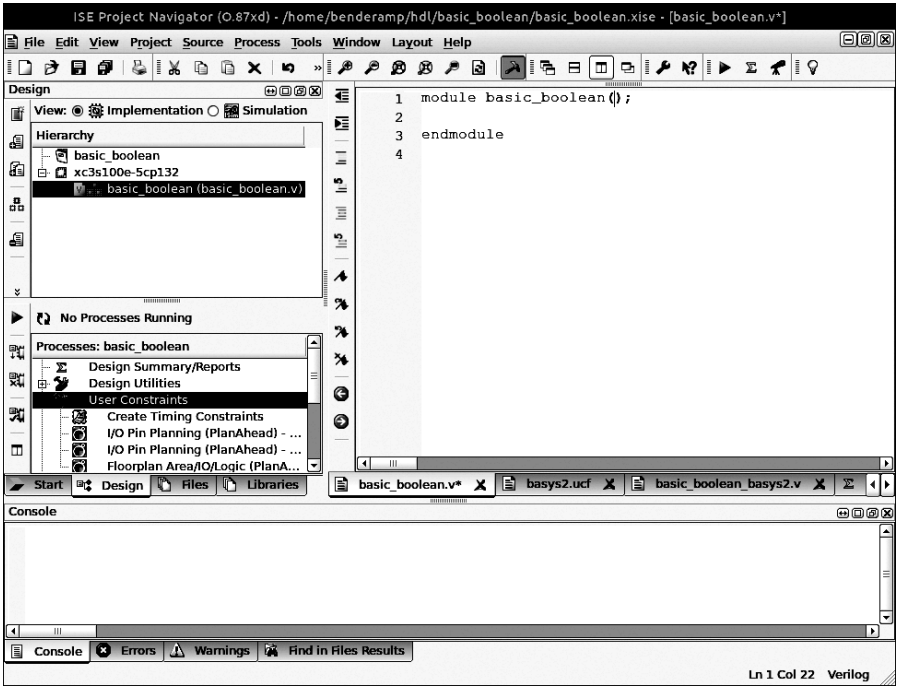

Рис. 3. Диалоговое окно редактора Xilinx ISE. Файл с исходным кодом

Модуль *Verilog* - это основная структурная единица построения цифровой системы. Определение любого модуля начинается с перечисления его входов (input) и выходов (output).

В модуле *basic_boolean* будет практически реализована работа базовых булевых операторов *NOT, AND, OR, NAND*, поэтому у него будет два входа и несколько выходов - по одному на каждый оператор:

- два входа (input): a и b
- пять выходов (output): not_a, not_b, a_and_b, a_or_b, a_nand_b.

В соотвествии с данным условием добавляем код:

```
module basic_boolean(input a, input b,
output not_a, output not_b, output a_and_b,
output a_or_b, output a_nand_b
);
endmodule
```

В теле модуля входами и выходами можно оперировать – выходам можно присваивать значения при помощи оператора assign. При этом значением может быть результат действия стандартных булевых операторов (или их комбинаций) над входами (или другими внутренними переменными или числовыми константами).

Каждый вход и выход в данном случае несет ровно 1 бит информации *(1=TRUE, 0=FALSE).*

В результате получим следующий код (см. рис. 4):

```
module basic_boolean(input a, input b,
output not_a, output not_b, output a_and_b,
output a_or_b, output a_nand_b
);
assign not_a = ~a; // NOT
assign not_b = ~b; // NOT
assign a_and_b = a & b; // AND
assign a_or_b = a | b; // OR
assign a_nand_b = ~(a & b); // NAND
endmodule
```

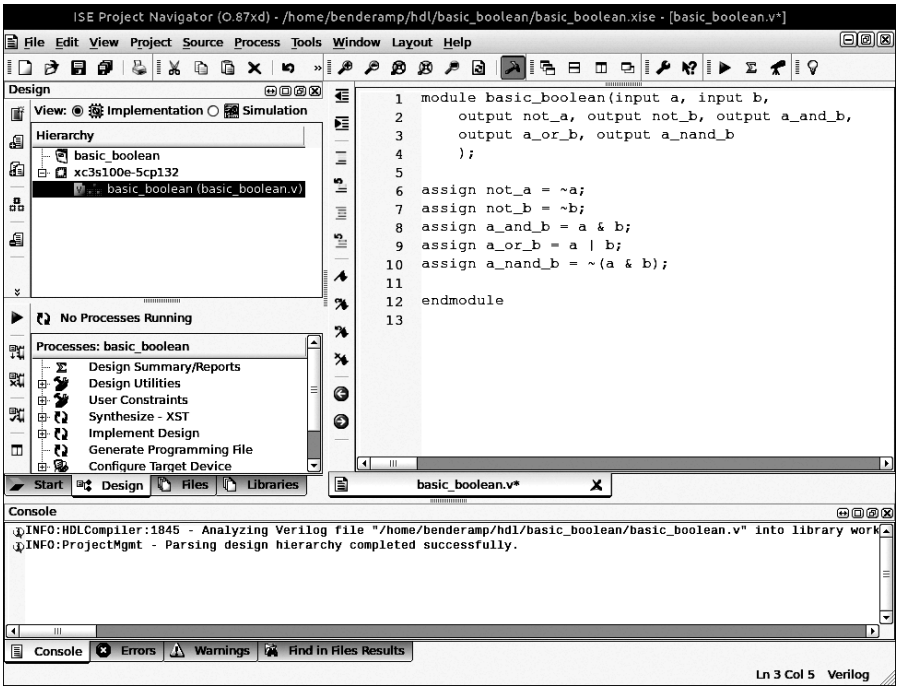

Рис. 4. Диалоговое окно редактора Xilinx ISE. Описание проекта на языке Verilog

Подключение файла конфигурации. Полученный программный код позволяет осуществить проверку указанных базовых булевых операторов. Данный проект, возможно, реализовать с использованием программных симуляторов или специализированной отладочной плате ПЛИС.

Перед синтезом прошивки для платы ПЛИС отметим, что любая плата ПЛИС представляет из себя «черный ящик», «внутри» которого находится прошивка, синтезированная из программы на языке *Verilog*, а «снаружи» физические устройства ввода/вывода (на отладочной плате Digilent Basys 2 установлены переключатели (*SW*), кнопки (*BTN*) управления входными портами, светодиоды и дисплей для вывода

информации и т.п.). Для того чтобы программа на языке *Verilog* получила возможность связаться с устройствами ввода/вывода определенной платы, нужен соответствующий механизм. Для этого в среде разработки Xilinx ISE используется инструмент специальных файлов конфигурации с расширением **.ucf* (файл *basys2.ucf*).

Все устройства ввода/вывода на плате пронумерованы уникальными идентификаторами. На Digilent Basys 2 эти идентификаторы указаны на самой плате (см. рис. 5) или же доступны в документации на сайте производителя продукта.

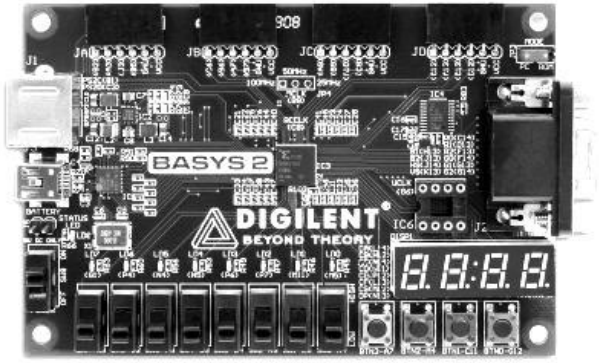

Рис. 5. Отладочная плата Digilent Basys 2

Файл *basys2.ucf* нужно подключить к проекту, для этого необходимо выбрать меню «Project > Add Copy of Source...» и найти в файловой системе.

Файл должен появиться в дереве проекта, после этого его нужно открыть в редакторе и оставить только те строки, которые соответствуют используемым в проекте устройствам ввода/вывода, а строки для остальных устройств нужно или удалить или закомментировать (символ «#» в начале строки), т.к. иначе система выдаст ошибку при синтезе прошивки (см. рис. 6).

#basys.ucf

NET "ld<4>" LOC = "n5" ;
NET "ld<3>" LOC = "p6" ;
NET "ld<2>" LOC = "p7" ;
NET "ld<1>" LOC = "m11" ;
NET "ld<0>" LOC = "m5" ;
NET "sw<1>" LOC = "l3";
NET "sw<0>" LOC = "p11";

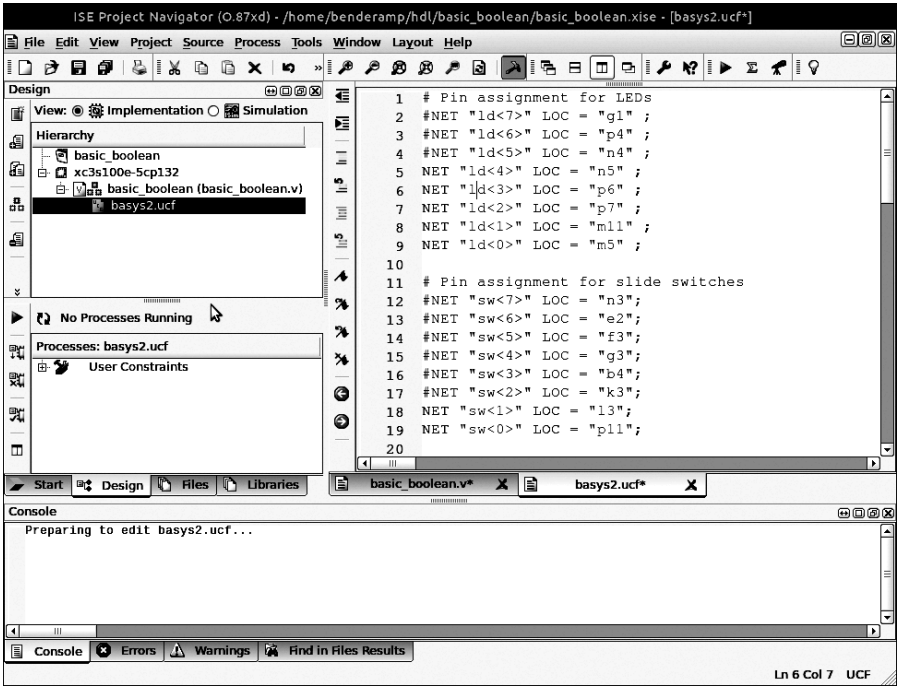

Рис. 6. Диалоговое окно редактора Xilinx ISE. Структура файла конфигурации

Создание модуля верхнего уровня для связи с устройствами ввода/вывода платы ПЛИС. Как уже было сказано выше, основная структурная единица дизайна на языке

Verilog - это модуль с набором входов и выходов. В рамках дизайна можно "размещать" модули на одном уровне друг рядом с другом и подключать выходы одного модуля к входам другого. А еще можно поместить несколько модулей внутрь другого модуля более высокого уровня - тогда входы внешнего модуля будут подключены к входам внутренних модулей, а выходы внутренних модулей будут подключены к выходам внешнего модуля.

При данном вложенном размещении модулей должен быть один главный модуль самого верхнего уровня (top module), который будет содержать все остальные модули дизайна вложенные один в другой. При синтезе проекта для ПЛИС в качестве такого модуля верхнего уровня (top module) выбирается модуль, входами и выходами которого являются реальные устройства ввода/вывода, доступные на плате, а внутри этого модуля размещены другие логические модули проекта (в нашем случае один модуль - *basic_boolean*).

Создаем новый модуль *Verilog* как было показано выше («Project > New Source... > Verilog Module») *basic_boolean_basys2*. Определяем входы и выходы с теми именами, которые указаны в файле *basys2.ucf* и направляем их в модуль *basic_boolean*. Получаем следующий код (см. рис. 7):

```
module basic_boolean_basys2(
input [0:1] sw,
output [0:4] ld);
basic_boolean impl(.a(sw[0]), .b(sw[1]),
.not_a(ld[0]), .not_b(ld[1]),
.a_and_b(ld[2]), .a_or_b(ld[3]),
.a_nand_b(ld[4]));
endmodule
```

basic_boolean - имя модуля

impl - указание внутреннего модуля внутри текущего модуля (может быть использован модуль на языке *Verilog* или

VHDL), в скобках указывается ассоциирование параметров текущего модуля с параметрами внутреннего модуля.

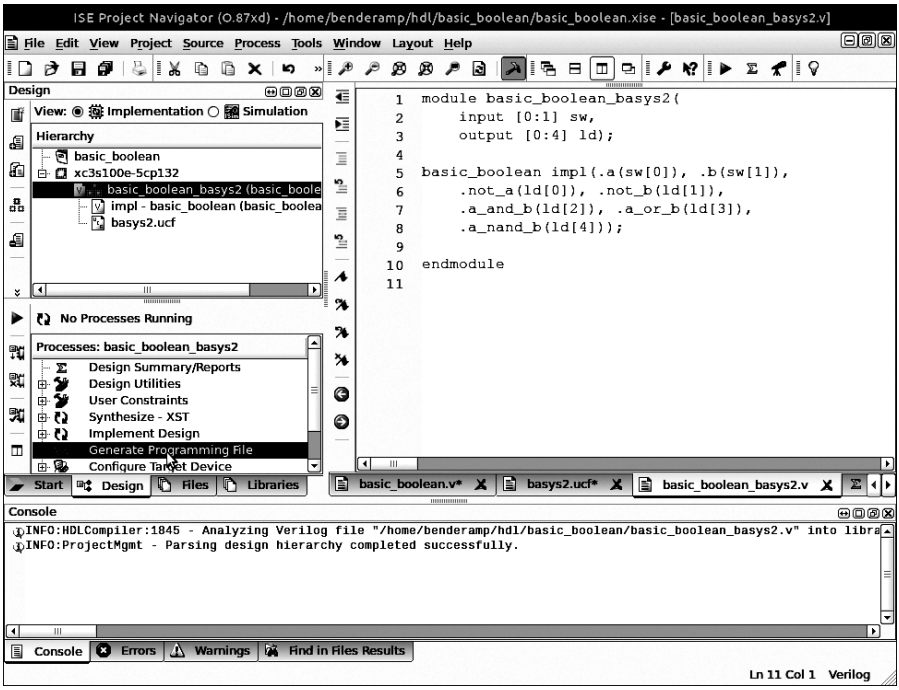

Рис. 7. Диалоговое окно редактора Xilinx ISE. Проект на языке Verilog (top module)

В результате получаем, что значения переключателей отладочной платы *sw[0]* и *sw[1] (0=FALSE или 1=TRUE)* подаются в качестве значений входных параметров модуля *basic_boolean* a и b, а светодиоды *ld[0]...ld[4]* будут показывать значения результатов работы модуля *basic_boolean not_a, not_b, a_and_b, a_or_b, a_nand_b (0=FALSE=выкл или 1=TRUE=вкл).*

Синтез прошивки. Для синтеза прошивки ПЛИС необходимо в списке действий под деревом проекта выбрать элемент Generate Programming File, если синтез прошел без

ошибок программный комплекс формирует *bit*-файл (*basic_boolean_basys2.bit*) с прошивкой (см. рис. 8).

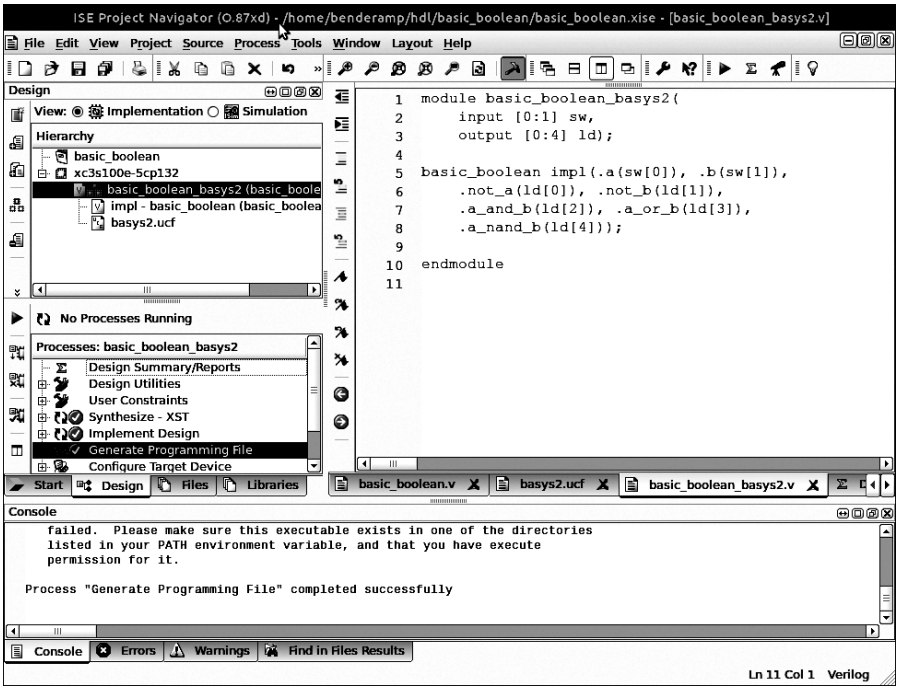

Рис. 8. Диалоговое окно редактора Xilinx ISE. Синтез прошивки

Программирование платы ПЛИС Digilent Basys 2. Осуществляется по средствам программы Adept компании Digilent, поставляется в комплекте с отладочной платой Digilent Basys 2 (см. рис. 9). Данная отладочная плата включает в себя две программируемые микросхемы FPGA XC3S250E и PROM XCF02S.

Перед программированием необходимо подключить отладочную плату к ПК, переключить плату в режим «PC», осуществить удаление предыдущей прошивки и произвести программирование. В режиме «ROM» производиться

непосредственная работа установленной прошивки (в случае программирования PROM).

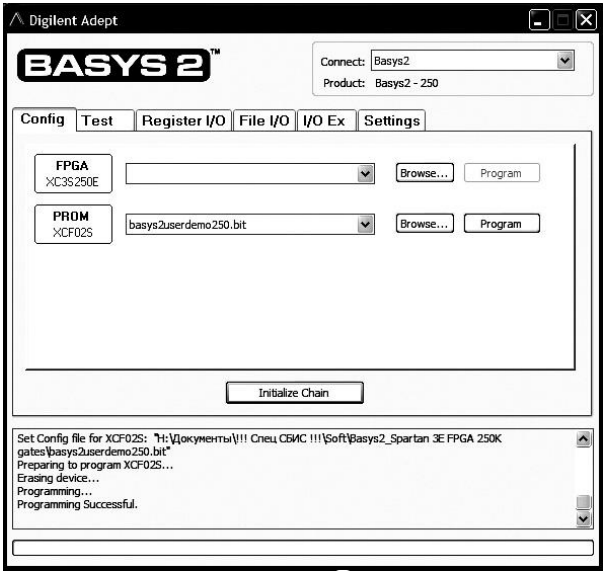

Рис. 9. Диалоговое окно программы Adept.

Подача входных сигналов на данной плате осуществляется при помощи набора переключателей, обозначенных *SW0 – SW7*, четырех не фиксируемых кнопок *BTN0 – BTN3*, а также есть возможность задания входных импульсов с клавиатуры, на плате для этого предусмотрен разъем PS/2. Выходные сигналы поступают на светодиоды *LD0 – LD7*, светодиодную панель или на монитор через разъем *VGA*.

Для проверки работоспособности перед началом программирования необходимо выполнить установку тестовой прошивки для проверки всех органов управления платой (файл *basys2userdemo250.bit*) или использовать вложенную утилиту «Test» программы Adept.

Задание 2. Работа с генератором тактового сигнала отладочной платы ПЛИС

В плату Digilent Basys 2 встроен генератор сигнала 25МГц. Использовать данный сигнал в модуле на *Verilog* возможно, подключив его в *ucf*-файле: *NET "clk" LOC = "b8"*. В данном задании необходимо разделить тактовый сигнал 25МГц так, чтобы он генерировал участки верхний/нижний уровень по одной секунде (воспринимаемой глазом человека). При помощи встроенного тактового генератора установить циклическую сигнализацию одного из диодов панели отладочной платы ПЛИС.

Для реализации поставленной задачи будем использовать счетчик, который будет отсчитывать каждый такт оригинального 25 мегагерцового сигнала и в нужный момент сообщать, что отсчитана секунда. Для этого счетчика используется переменная область памяти, в начальный момент времени в которую записан «0» и далее значение увеличивается на «1» на каждый такт генератора. Для создания такой переменной необходим механизм реализации памяти, которая будет сохранять каждое новое значение между тактами сигнала. В *Verilog* для этой цели можно использовать регистры.

Однобитный регистр:

reg counter;

Многобитный регистр (*bit_num* - количество бит в регистре):

reg [bit_num, 0] counter;

Для счетчика создадим многобитный регистр, который позволит досчитать 25 бит. Таким образом, когда значение переменной счетчика примет указанное выше значение, это будет обозначать, что прошла ровно одна секунда. В данной задаче фиксировать состояние будем, когда 25 битный счетчик будет переполнен, т.е. 26 старший бит переменной примет

значение «1». Соответственно, получим для счетчика 26 битный регистр:

reg [25, 0] counter;

Регистр объявлен, необходимо получить счетчик, который будет увеличиваться на 1 на каждый такт сигнала *clk*. Для таких задач в Verilog существует конструкция *always @*, состоящая из двух частей:

always @(posedge clk)
counter <= counter + 1;

1. Внутри скобок находится "список чувствительности" (*sensitivity list*), в котором перечислены условия, при осуществлении которых будет отработан блок действия.

2. Блок действия, - это все, что находится внутри конструкции *always*, следующая строка за ней, если нужно поместить несколько строк, можно отметить блок маркерами begin/end.

В качестве условий из первого пункта можно перечислять сигналы (внутренние переменные и входы модуля), а триггером к их выполнению является изменение значения одного из сигналов из списка чувствительности. В данном случае блок *always* будет отрабатывать каждый раз, когда значение сигнала *clk* будет меняться с 0 на 1, т.е. на положительном ребре сигнала, дополнительное условие накладывается ключевым словом *posedge (positive edge).*

Блок действия – *«counter <= counter + 1»* - увеличивает значение *counter* на 1 по сравнению с предыдущим значением, оператор «<=» используется для присвоения значений внутри блока *always*. Левый операнд оператора (*counter*) должен быть обязательно объявлен как регистр reg. Значение регистра *counter* будет увеличиваться на 1 на каждый такт периодического сигнала. Подключаем к 26 старшему биту регистра *counter* выходной сигнал *one_second* модуля, т.е. выход *one_second*

будет иметь значение 1, когда 26 бит регистра *counter* будет равен «1» и будет иметь значение «0», когда 26 бит регистра *counter* будет равен «0», получим:

assign one_second = counter[25];

Таким образом, 26 старший бит регистра (а вслед за ним и выход модуля *one_second*) будет держать значение «1» одну секунду, и будет держать значениеа «0» одну секунду. В этом можно убедиться, подключив к нему в модуле верхнего уровня один из диодов панели отладочной платы ПЛИС. Пример кода представлен ниже. Модуль верхнего уровня:

```
module clock_driver2(
input clk,
output [0:0] ld);
clock_driver count(.clk(clk), .one_second(ld[0]));

endmodule
```

Внутренний модуль:

```
module clock_driver(input clk,
output one_second);
reg [25:0] counter;
// count up on each positive clock
always @(posedge clk)
counter <= counter + 1;
// one second signal would be emitted when
// 26th bit of the counter would become "1"
assign one_second = counter[25];

endmodule
```

Параметры файла конфигурации *basys.ucf*:

```
NET "ld<0>" LOC = "m5";
NET "clk" LOC = "b8";
```

Задание 3. Разработка драйвера для сегментного светодиодного дисплея отладочной платы ПЛИС

Необходимо написать драйвер для сегментного диодного дисплея, состоящего из семи элементов. Дисплей должен показывать две разные цифры в зависимости от текущего положения рычага *sw* – при *sw = high* дисплей пусть показывает цифру «1», при *sw = low* – цифру «0». Для выполнения задания необходимо использовать оператор условного выбора:

assign out1 = condition ? val_cond_true : val_cond_false;

condition - это булево значение *1(TRUE)* или *0(FALSE)* (значение на одном из входов). Если *condition = 1(TRUE),* переменной *out1* будет присвоено значение *val_cond_true*, если *condition = 0(FALSE)*, переменной *out2* будет присвоено значение *val_cond_false.*

Сегменты дисплея имеют нумерацию согласно рисунку ниже и имеют следующие обозначения: *A(seg<0>), B(seg<1>), C(seg<2>), D(seg<3>), E(seg<4>), F(seg<5>), G(seg<6>).*

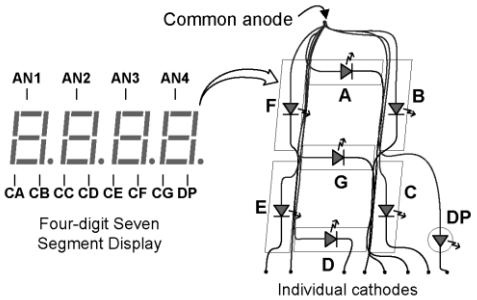

Рис. 10. Обозначение элементов диодного дисплея.

Проект драйвера дисплея содержит один рабочий модуль *seven_driver*, один модуль верхнего уровня *seven_driver_basys2*, Пример кода представлен ниже.

Модуль верхнего уровня:

```
module seven_driver_basys2(input [0:0] sw,
output [0:6] seg);

seven_driver impl(.value(sw), .segments(seg));

endmodule
```

Внутренний модуль:

```
module seven_driver(
input value,
output [0:6] segments);

    // value=true - 1
    // value=false - 0

assign segments[0] = value ? 1 : 0;
assign segments[1] = value ? 0 : 0;
assign segments[2] = value ? 0 : 0;
assign segments[3] = value ? 1 : 0;
assign segments[4] = value ? 1 : 0;
assign segments[5] = value ? 1 : 0;
assign segments[6] = value ? 1 : 1;

endmodule
```

Параметры файла конфигурации *basys.ucf*:

```
NET "seg<0>" LOC = "L14";
NET "seg<1>" LOC = "H12";
NET "seg<2>" LOC = "N14";
NET "seg<3>" LOC = "N11";
NET "seg<4>" LOC = "P12";
NET "seg<5>" LOC = "L13";
```

NET "seg<6>" LOC = "M12";
NET "sw<0>" LOC = "P11";

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Угрюмов, Е.П. Цифровая схемотехника [Текст] : учеб. пособие / Е.П. Угрюмов. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2004. – 800 с.

2. Солонина, А.И. Основы цифровой обработки сигналов [Текст] : учеб. пособие / А.И. Солонина, Д.А. Ухладович, С.М. Арбузов, Е.Б. Соловьева. – СПб. : БХВ-Петербург, 2005. – 768 с.

3. Солонина, А. И. Алгоритмы и процессоры цифровой обработки сигналов [Текст] : учеб. пособие / А.И. Солонина, Д.А. Улахович, Л.А. Яковлев. – СПб. : БХВ-Петербург, 2002. – 464 с.

4. www.xilinx.com

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по дисциплине «Проектирование цифровых сложнофункциональных блоков» направления 11.04.03 «Конструирование и технология электронных средств», магистерская программа «Автоматизированное проектирование и технология радиоэлектронных средств специального назначения»

по дисциплине «Методы проектирования сложнофункциональных блоков электронных средств» направления 11.04.03 «Конструирование и технология электронных средств», магистерская программа «Силовая электроника»

по дисциплине «Специализированные интегральные устройства и системы в приборостроении» направления 12.04.01 «Приборостроение», магистерская программа «Автоматизированное проектирование приборов и комплексов»

Составитель
Пирогов Александр Александрович

В авторской редакции

Подписано к изданию _____
Уч.-изд. л. 1,6.

ФГБОУ ВПО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14