

ЛАБОРАТОРНАЯ РАБОТА №1 «ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ»

Цель работы: изучить основные инструменты САПР Vivado, научиться моделировать в ней работу схем на основе простых логических элементов и познакомиться с платой BASYS 3.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

1. Создание проекта

Перед началом работы необходимо создать новый проект. Для этого в окне «Quick Start» нужно выбрать «Create Project». При этом откроется окно создания нового проекта (рис. 1). В окне необходимо нажать кнопку «Next».

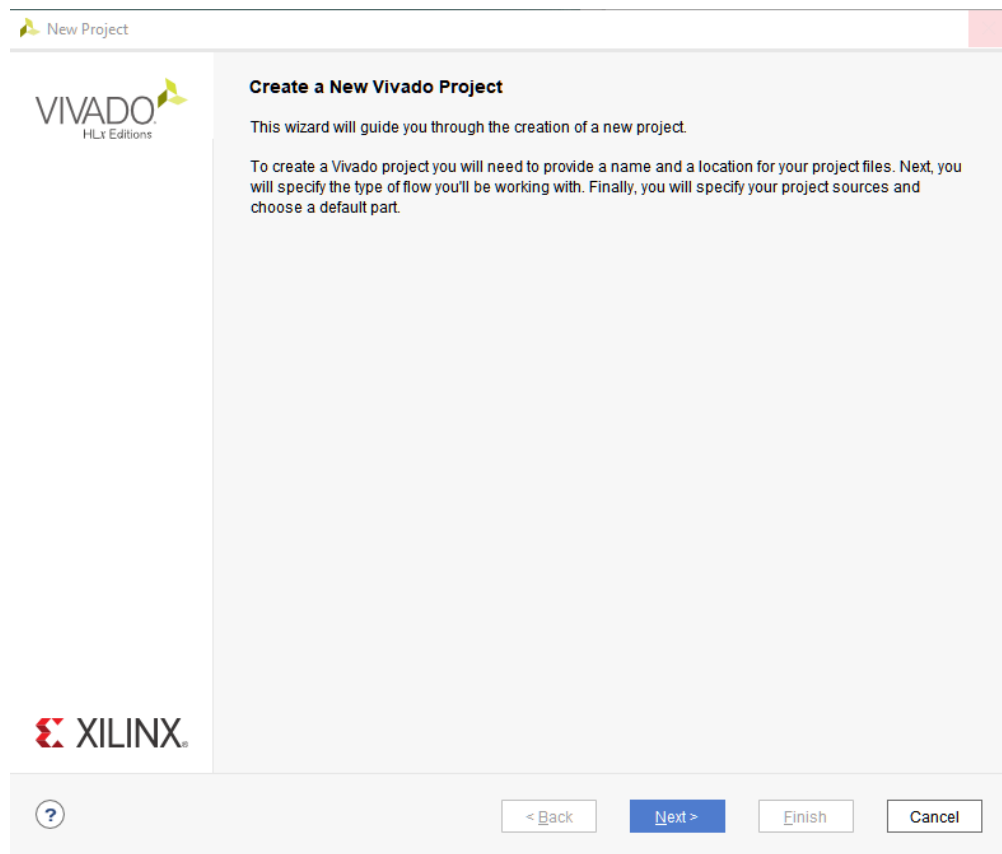


Рисунок 1 – Окно создания проекта

В следующем окне (рис. 2) необходимо в поле «Project name» ввести название проекта, в поле «Project location» путь, по которому будет создан проект. Проекты желательно создавать там же, где находится сама САПР и чтобы проекты не мешали, под них надо создать папку проектов (projects), а в ней указать папку своей группы и своё ФИО. Папка под сам проект создастся автоматически по имени проекта. Затем нажать кнопку «Next».

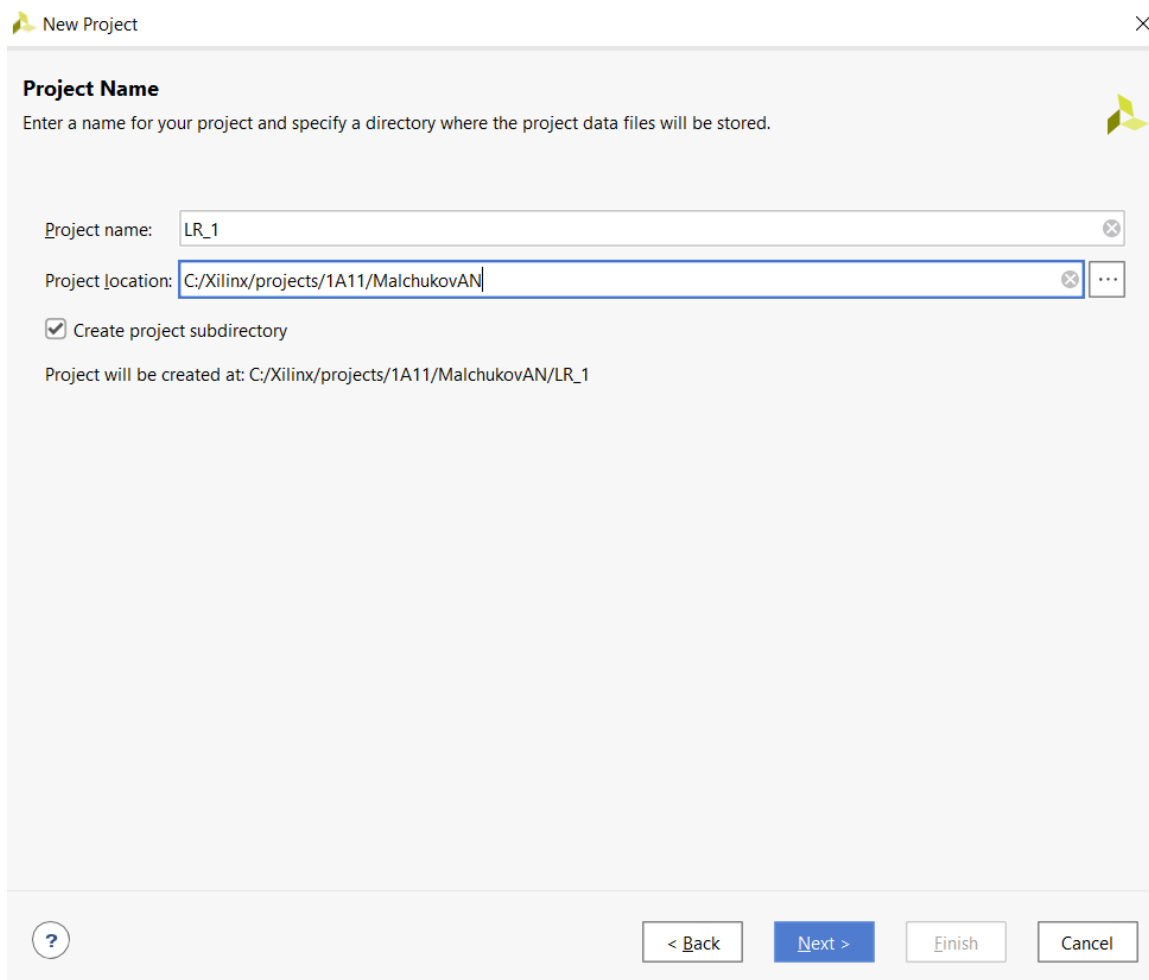


Рисунок 2 – Название проекта и путь к нему

В следующем окне (рис. 3) нужно выбрать тип проекта «RTL Project» и прожать галочку «Do not specify sources at this time». Затем нажать кнопку «Next».

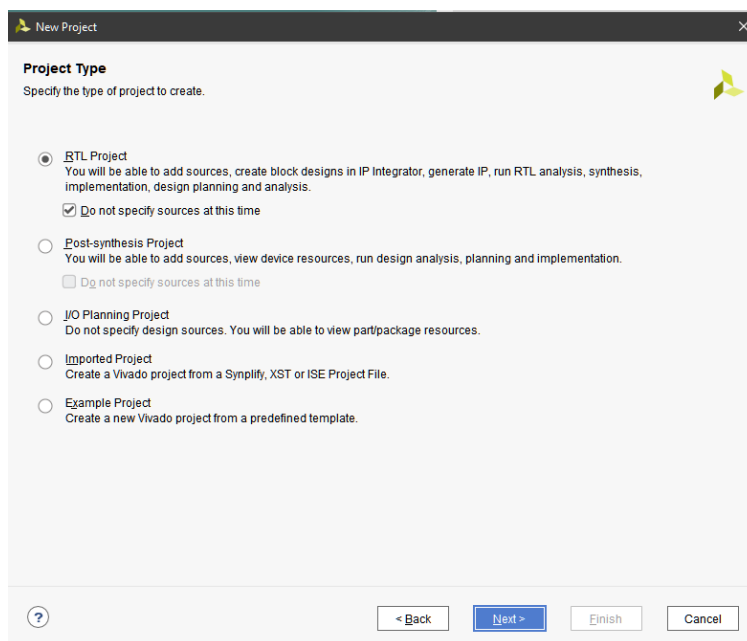


Рисунок 3 – Выбор типа проекта

В следующем окне (рис. 4) нужно выбрать ПЛИС, с которой планируется работать в проекте. В нашем случае нужно обязательно выбрать «XC7A35TCPG236-1», чтобы сократить список можно выбрать Family «Artix-7», Package «cpg236» и Speed «-1».

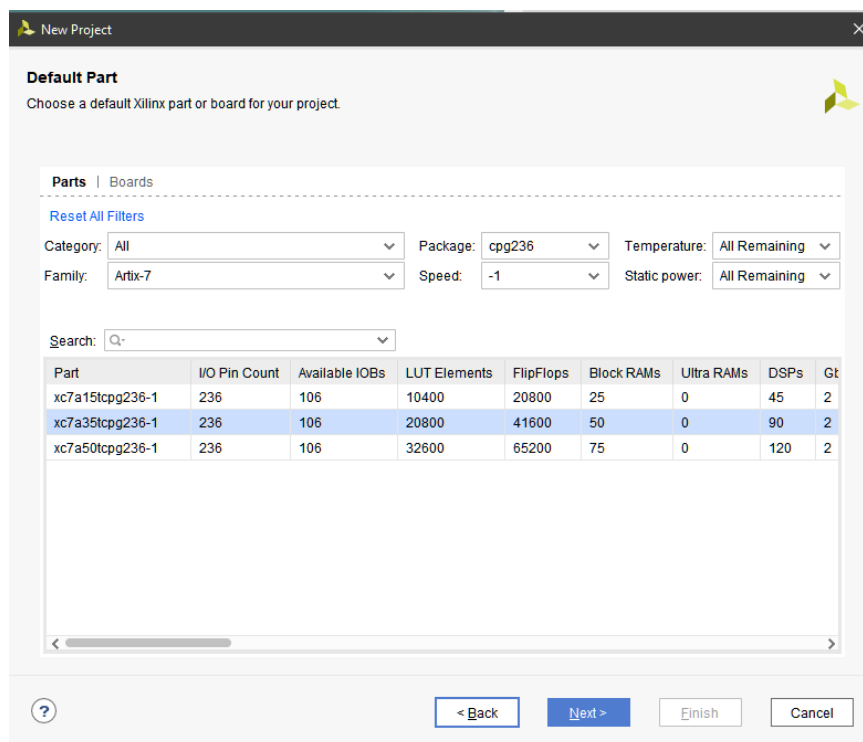


Рисунок 4 – Настройка кристалла ПЛИС

В следующем окне нажать кнопку «Finish» (рис. 5).

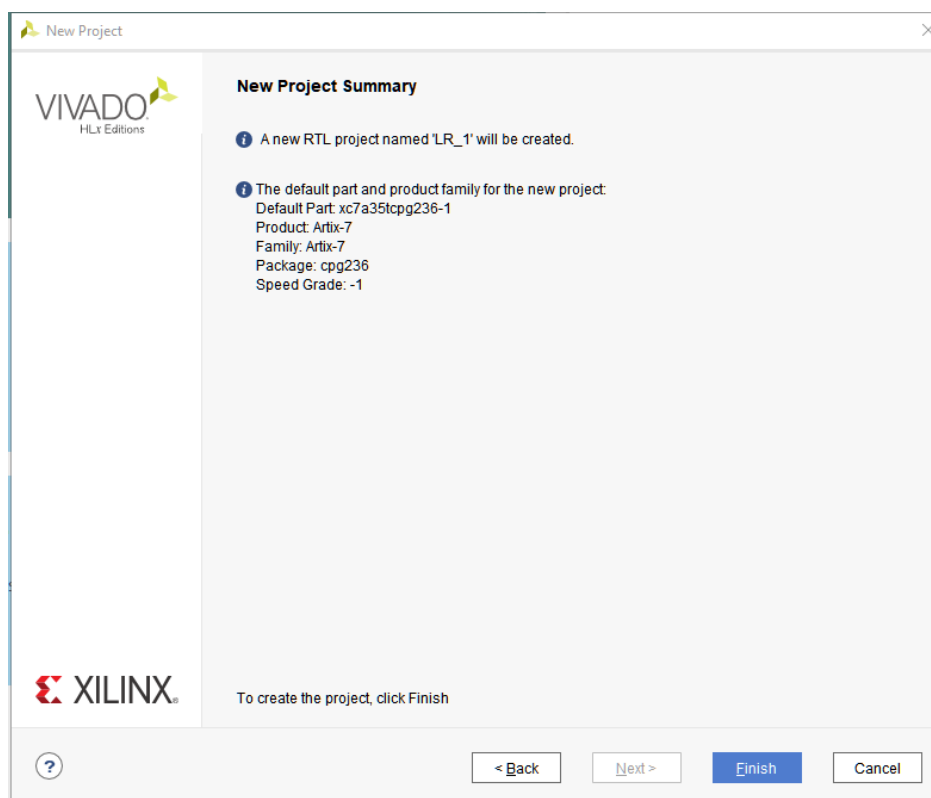


Рисунок 5 – Окно завершения создания проекта

Проект успешно создан! При создании проекта открывается менеджер проекта (рис. 6). Менеджер содержит в себе поле слева «Flow Navigator» – окно, содержащее в себе несколько разделов. В рамках этой лабораторной работы познакомимся с «Simulation», «Synthesis», «Implementation», «Program and debug».

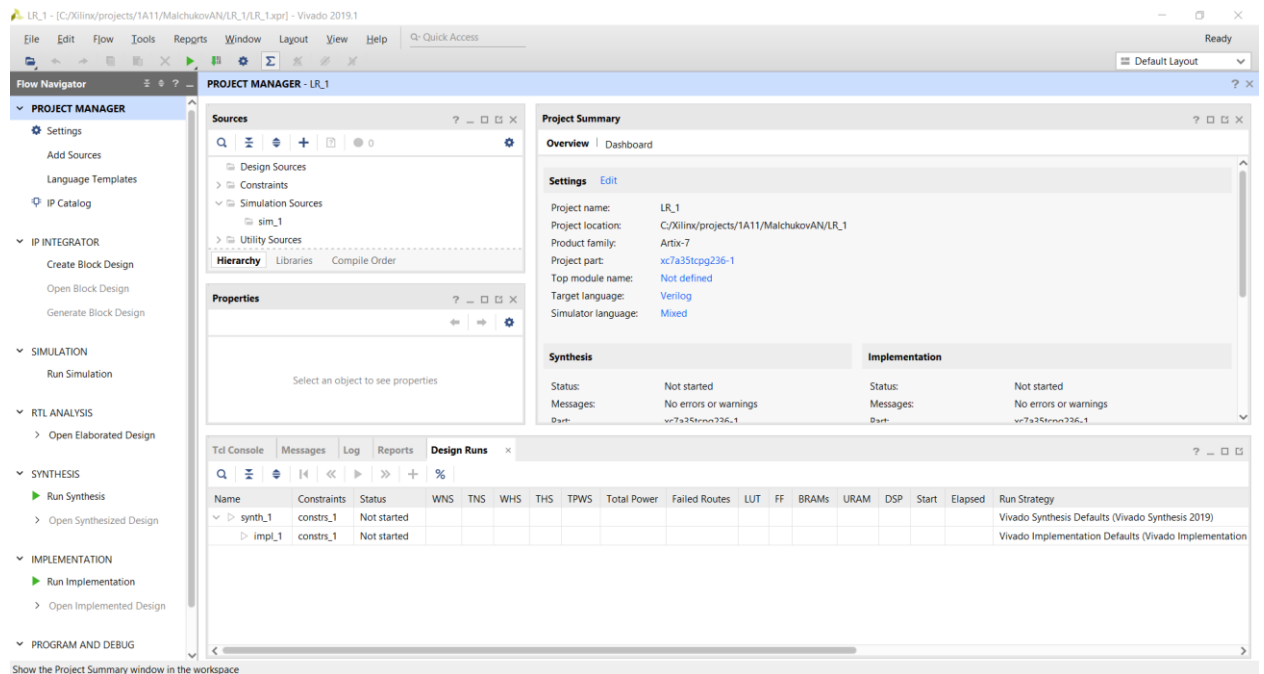


Рисунок 6 – Окно менеджера проекта Vivado

Также менеджер содержит в себе окно «Sources». Для работы необходимо нажать правой кнопкой мыши (ПКМ) по вкладке «Design Sources». В контекстном меню выбрать «Add Sources...». Либо слева на панели «Flow Navigator» в разделе «PROJECT MANAGER» нажать ЛКМ на «Add Sources». Откроется окно создания «Add Sources». В перечне выбрать «Add or create design sources». Затем нажать «Next».

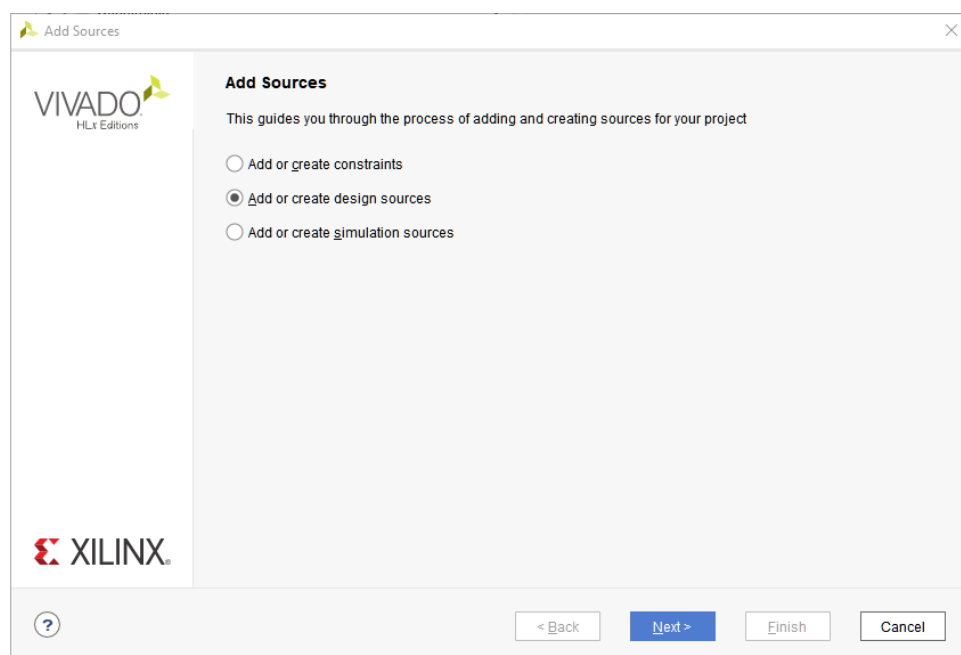


Рисунок 7 – Окно создания модулей

В следующем окне нажать «Create File» (рис. 8).

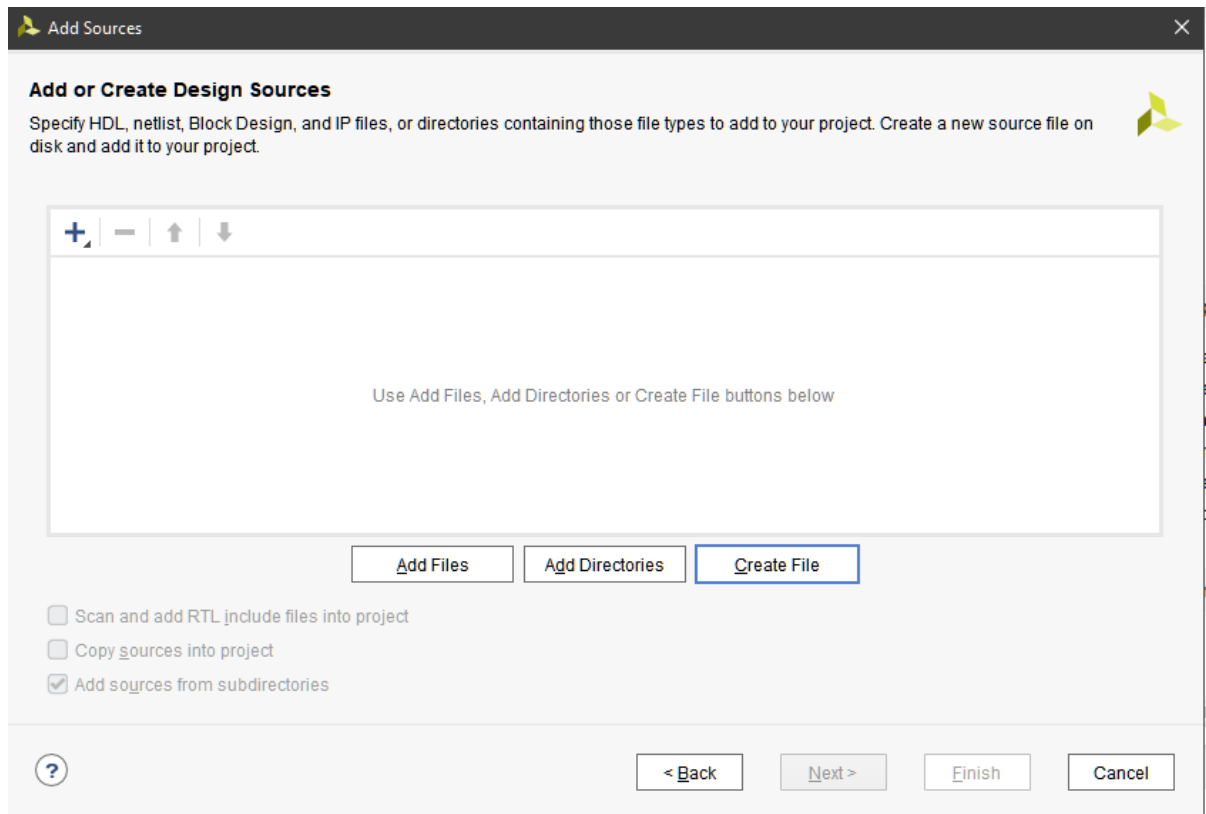


Рисунок 8 – Окно создания файлов

Пользователю предлагается выбрать тип модуля: SystemVerilog или VHDL. Нужно ввести название файла и нажать «OK» (рис. 9 и 10).

SystemVerilog – это расширение языка Verilog, которое предоставляет более обширные возможности верификации. Синтаксис SystemVerilog очень похож на синтаксис языка C и Pascal.

VHDL (VHSIC (Very high speed integrated circuits) Hardware Description Language) – язык описания аппаратуры интегральных схем. Язык VHDL является одним из базовых языков описания аппаратуры, наравне с Verilog (SystemVerilog), при проектировании современной аппаратуры. В VHDL подобно языку программирования C++, многие функции реализованы в виде подключаемых библиотек. Основная используемая библиотека – IEEE, в которой, согласно стандартам института инженеров электротехники и электроники, находятся реализации соответствующих функций.

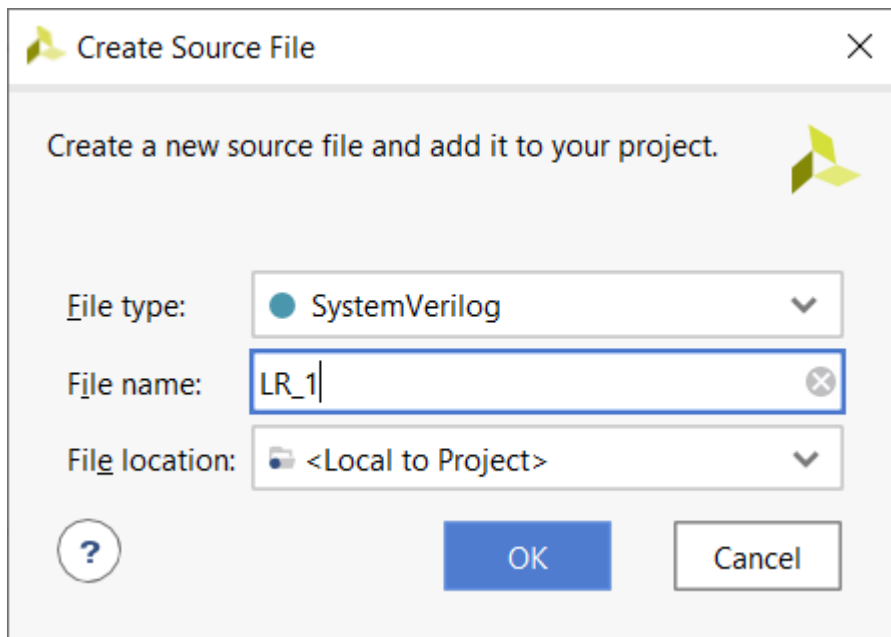


Рисунок 9 – Создание файла типа SystemVerilog

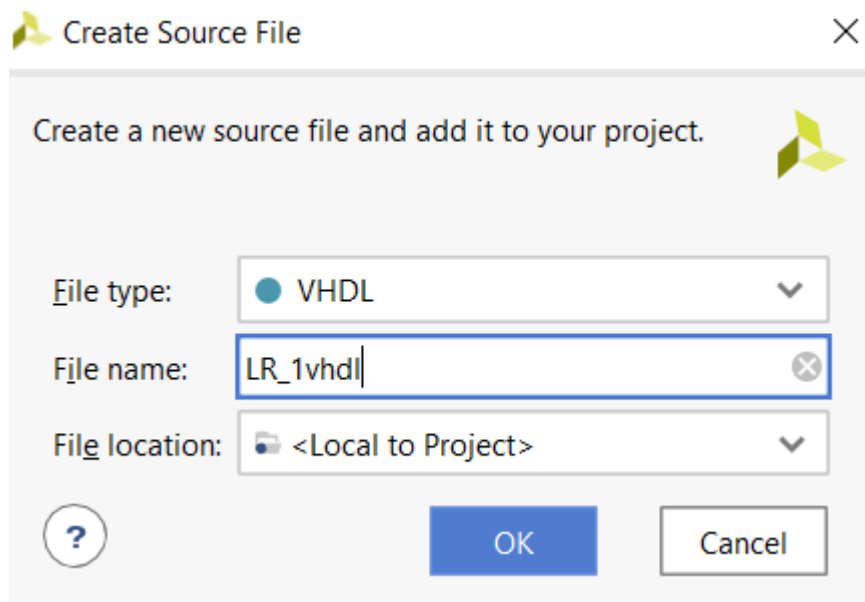


Рисунок 10 – Создание файла типа VHDL

Далее пользователю предлагается заранее создать порты, используемые в проекте (рис. 11). Для добавления портов в проект нужно нажать +, вписать название порта, определить как вход, выход или двунаправленный, при наличии нескольких портов в одной шине указать это и указать количество портов в шине (MSB – старший номер порта, LSB – младший номер порта). Этот этап можно пропустить и вписать порты непосредственно уже в файле. После нажать «ОК».

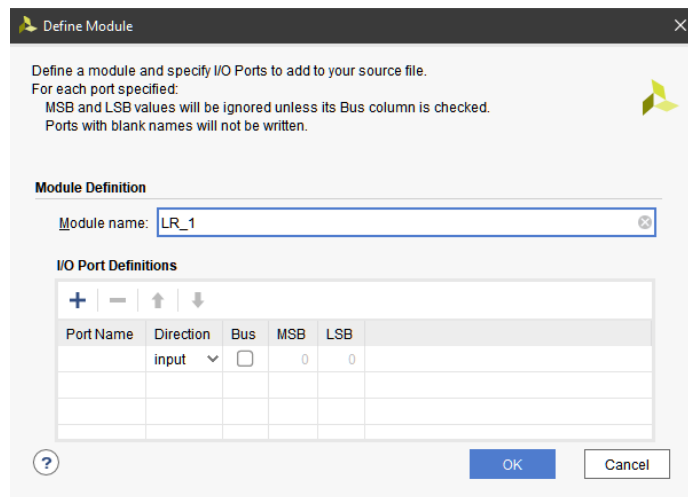


Рисунок 11 – Создание портов в Design файле

После создания модуля, он отобразится во вкладке «Design Sources» (рис. 12 и 13). При открытии файла справа окна Sources откроется окно редактора модуля.

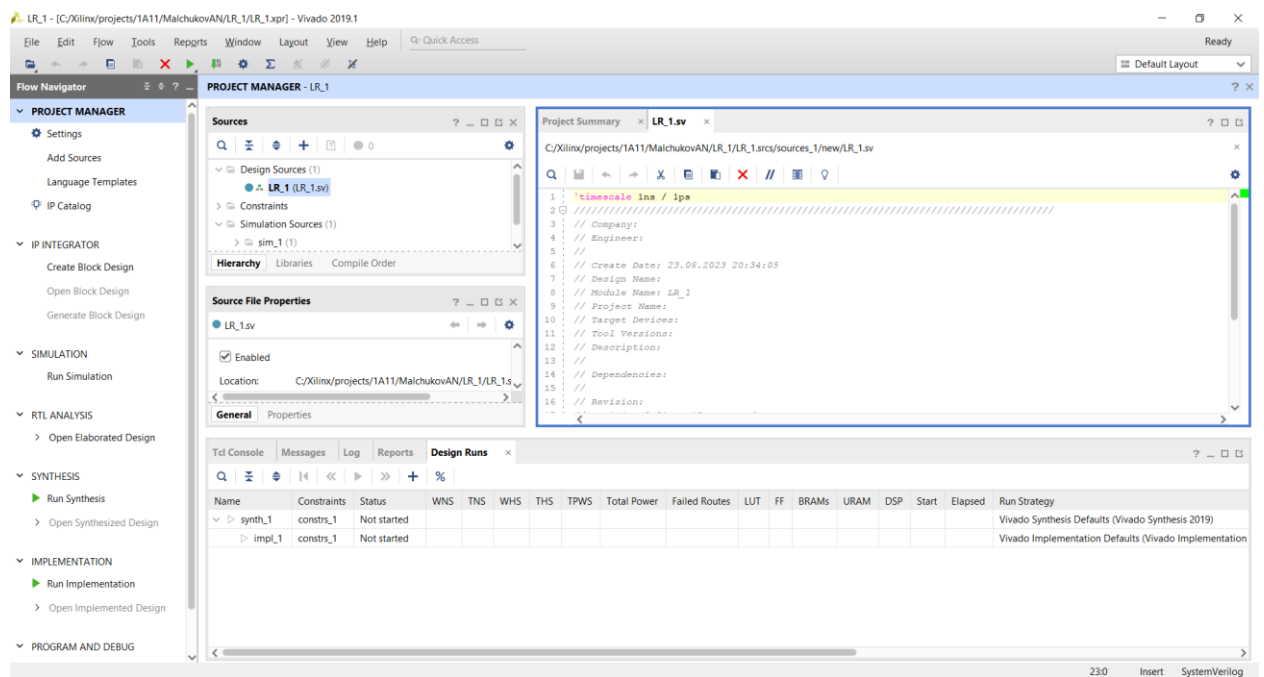


Рисунок 12 – Редактор файла LR_1.sv в проекте

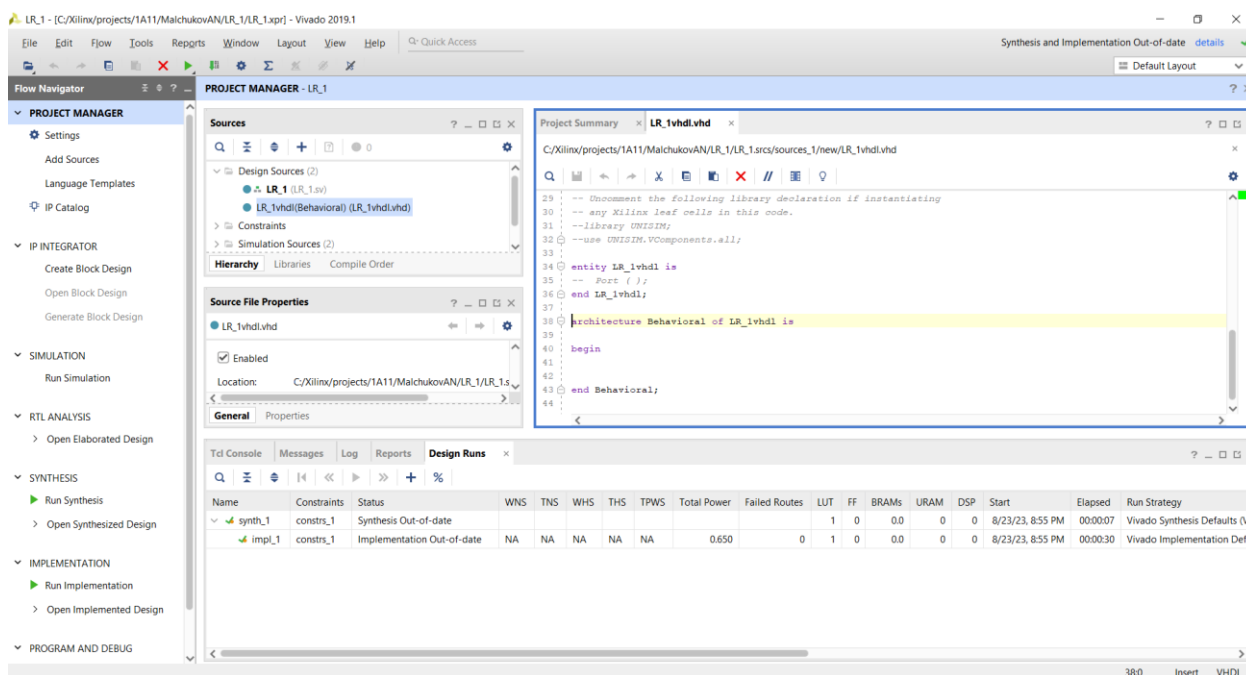


Рисунок 13 – Редактор файла LR_1.vhdl.vhd в проекте

На этом создание проекта закончено, можно приступать к работе!

2. Написание кода и компиляция

Для примера рассмотрим работу логического элемента XOR с 3 переменными:

$$X = A \oplus B \oplus C.$$

Таблица 1

Таблица истинности функции $X = A \oplus B \oplus C$

Входы			Выход
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Функция содержит 3 входа и 1 выход, её таблица истинности представлена выше. Входы и выходы относятся к портам и вносятся в заголовок модуля (рис. 14 и 15).

Программный код на языке SystemVerilog начинается с объявления «module <название_файла>». Затем в скобках записаны порты: входы, выходы или двусторонние порты. После объявления портов скобки закрываются и затем следует исполняемый код.

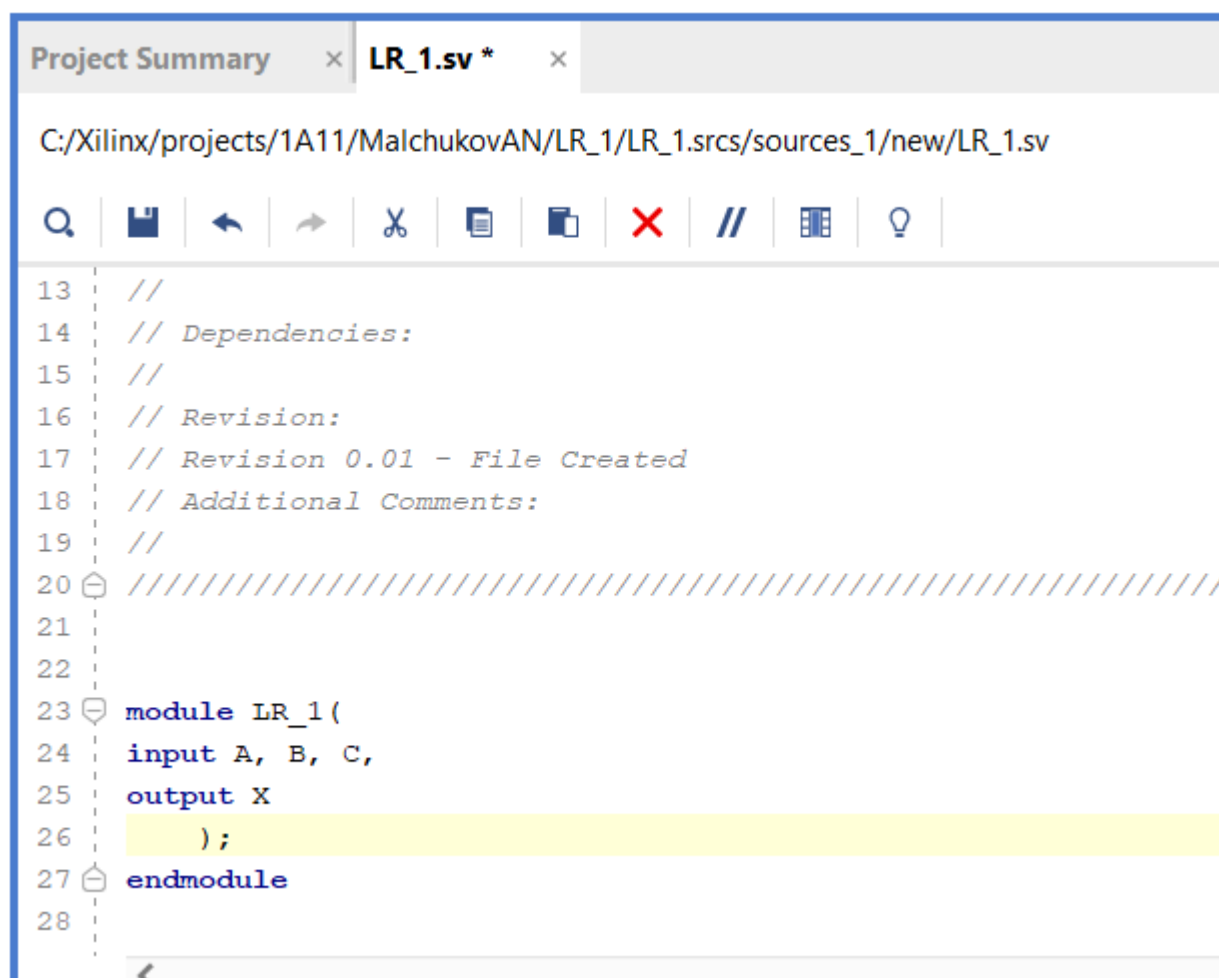


Рисунок 14 – Редактор файла LR_1.vh

На языке VHDL прежде всего необходимо объявить необходимые библиотеки из коллекции «library IEEE;». В нашем случае потребуется только одна «use IEEE.STD_LOGIC_1164.ALL;». Если необходимо работать с операцией вычитания, то потребуется библиотека «use IEEE.NUMERIC_STD.ALL;». Для преобразования целого числа в STD_LOGIC_VECTOR и обратно, кроме уже указанных, потребуется ещё библиотека «use IEEE.STD_LOGIC_UNSIGNED.ALL;».

Далее объявление устройства «entity <название_файла> is». Затем в port(); записаны порты: входы, выходы или двусторонние порты. После объявления портов скобки закрываются и затем следует исполняемый код, начало которого предваряет «architecture Behavioral of <название файла> is».

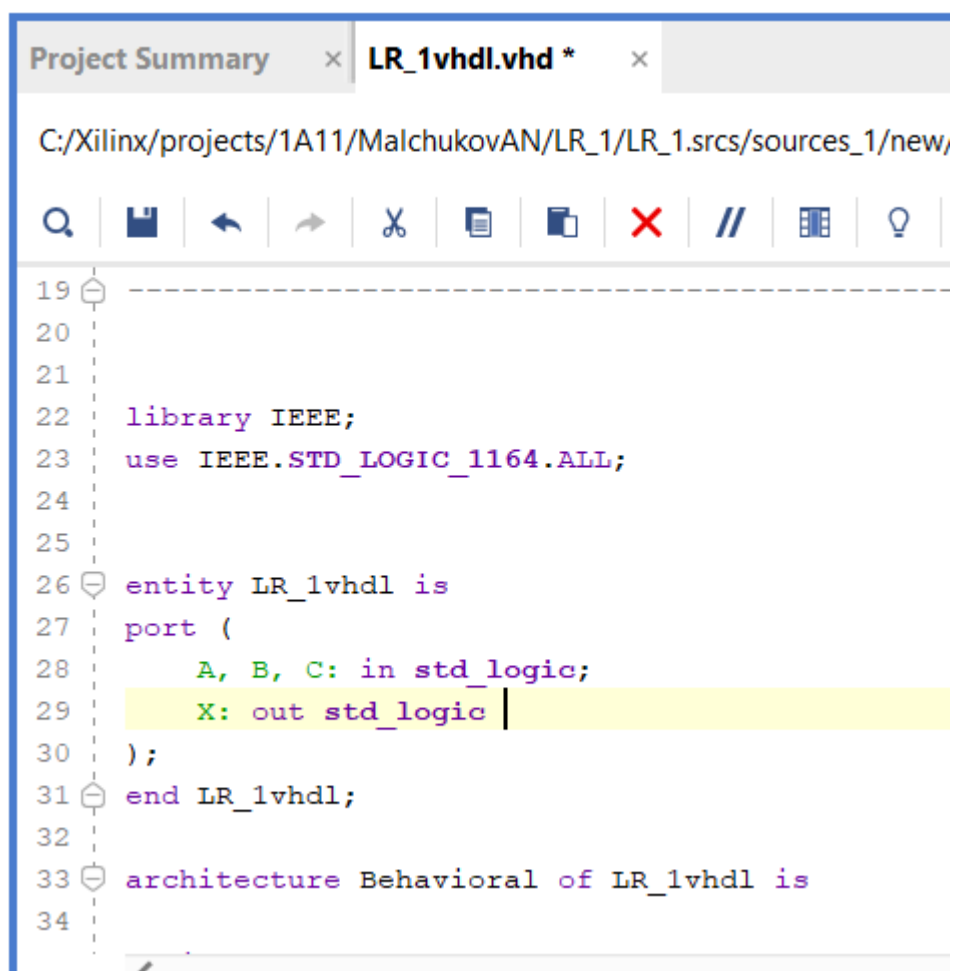


Рисунок 15 – Редактор файла LR_1vhdl.vhd

Объявление простейшей логической функции на языке SystemVerilog начинается с записи *assign*, после чего записывается сама функция (рис. 16). Программный код должен заканчиваться командой «*endmodule*».

```

23  module LR_1 (
24      input A, B, C,
25      output X
26  );
27      assign X = A ^ B ^ C;
28  endmodule

```

Рисунок 16 – Функция исключающего ИЛИ (XOR) на SystemVerilog

Простейшая логическая функция на языке VHDL записывается между «*begin*» и «*end Behavioral;*» (рис. 17).

```

22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity LR_1vhd1 is
26 port (
27     A, B, C: in std_logic;
28     X: out std_logic
29 );
30 end LR_1vhd1;
31
32 architecture Behavioral of LR_1vhd1 is
33 begin
34     X <= A xor B xor C;
35 end Behavioral;

```

Рисунок 17 – Функция исключающего ИЛИ (XOR) на VHDL

После сохранения кода (Ctrl+S или синяя кнопка дискеты панели текстового редактора) необходимо убедиться, что нет никаких замечаний на окошке «Sources» там, где список файлов.

Когда в проекте необходимо скомпилировать и промоделировать работу отдельных модулей, чтобы не удалять файлы из проекта, необходимо менять файл верхнего уровня иерархии (Top Level), нажимая на нужный файл раздела «Design Sources» ПКМ (рис. 18) или других разделов, это же касается и файлов тестирования.

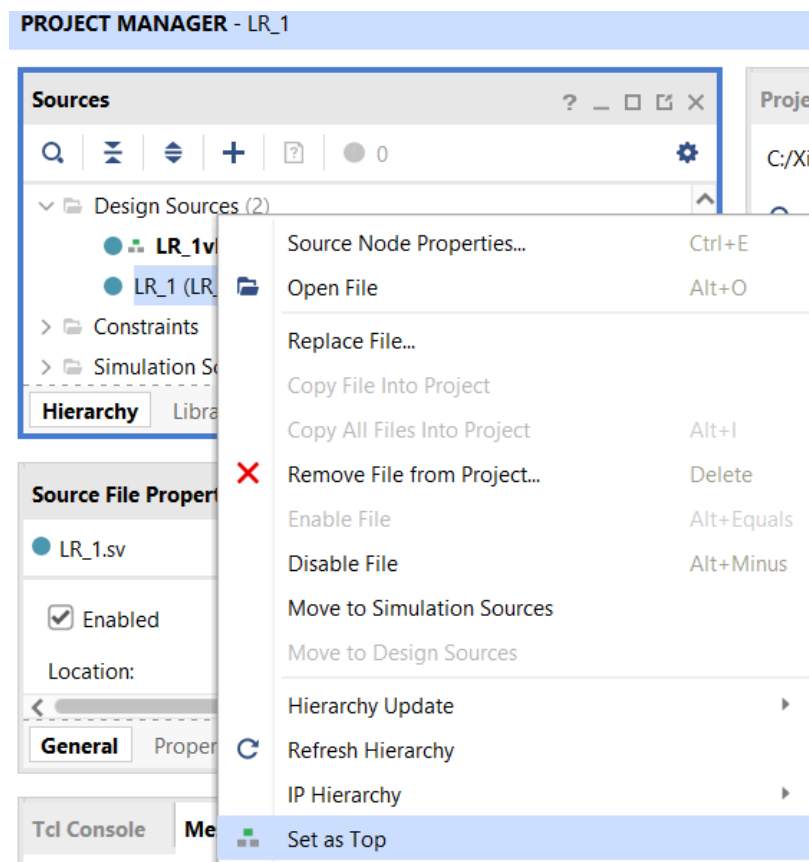


Рисунок 18 – Выбор файла верхнего уровня иерархии

Файл верхнего уровня иерархии в разделе «Design Sources» пишется жирным шрифтом и слева от его названия имеется иконка ■■.

После настройки (если таковая требуется) файла верхнего уровня на левой панели «Flow Navigator» в разделе «Synthesis» нажимаем «Run Synthesis» (рис. 19). Будет предложено сохранить изменения, нажимаем «ОК».

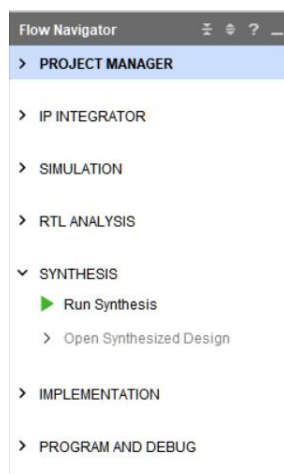


Рисунок 19 – Раздел Synthesis на левой панели

После запуска компиляции появится окно «Launch Runs» (рис. 20), чтобы оно более не появлялось нажимаем галочку «Don't show this dialog again». Нажимаем ОК.

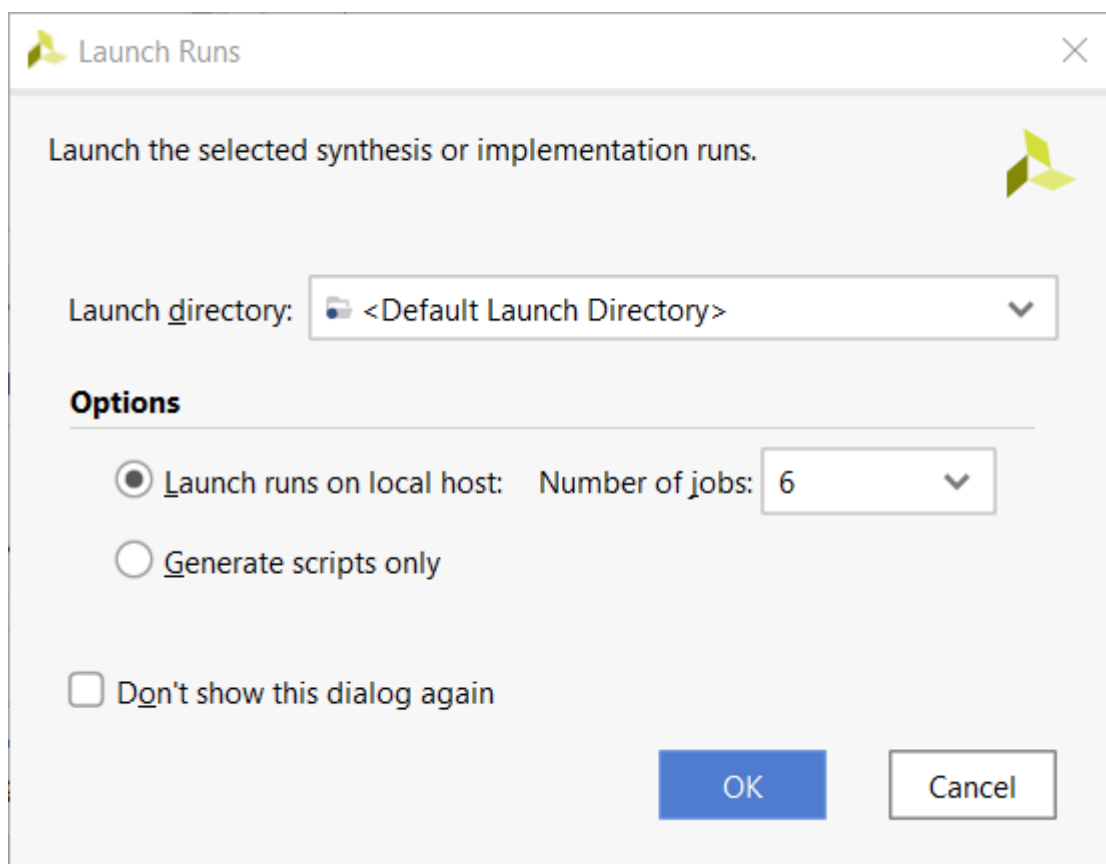


Рисунок 20 – Окно Launch Runs

Если синтез кода пройдет успешно, то далее предложат продолжить компиляцию проекта и перейти к этапу реализации (рис. 21). Нажимаем ОК.

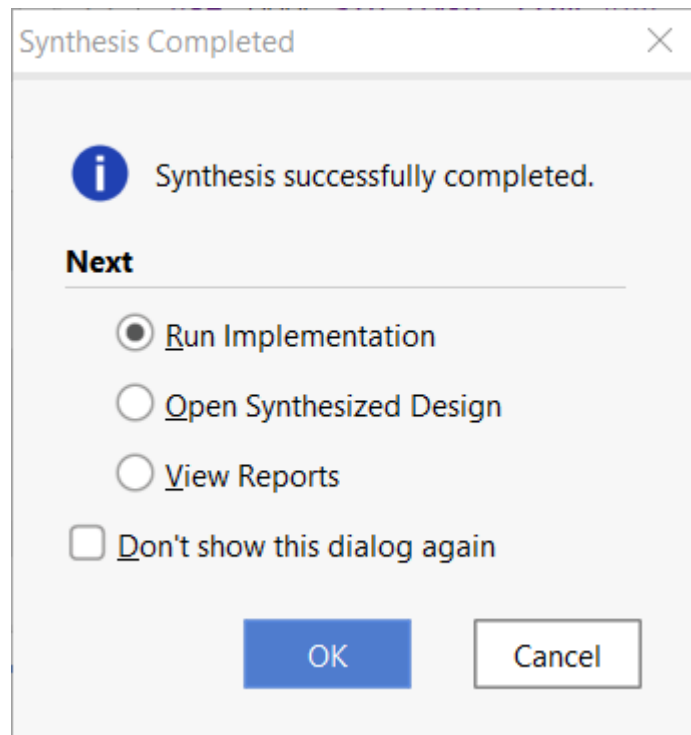



Рисунок 21 – Окно Synthesis Completed

После успешного выполнения этапа реализации в окне «Design Runs» состояние полей synth_1 и impl_1 изменятся: зеленая галочка говорит об успешном синтезе и реализации (рис. 22).

Tcl Console					Messages		Log		Reports		Design Runs		Power	DR
<input type="text"/> <input type="button" value="Q"/> <input type="button" value="≡"/> <input type="button" value="≡"/> <input type="button" value="⏮"/> <input type="button" value="⏪"/> <input type="button" value="⏩"/> <input type="button" value="⏭"/> <input type="button" value="+"/> <input type="button" value="%"/>														
Name		Constraints	Status		WNS		TNS							
✓ synth_1		constrs_1	synth_design Complete!											
✓ impl_1		constrs_1	route_design Complete!		NA		NA							

Рисунок 22 – Окно Design Runs

Появится окно с предложением посмотреть реализацию на кристалле, если согласиться, то возле окна «Sources» появится новая вкладка «Netlist», в которой нажимая на кнопку  «Schematic» (или кнопку F4) можно получить изображение схемы (рис. 23а). Однако, нормальную схему можно получить в разделе «RTL ANALYSIS», выбрав там пункт «Schematic» (рис. 23б).

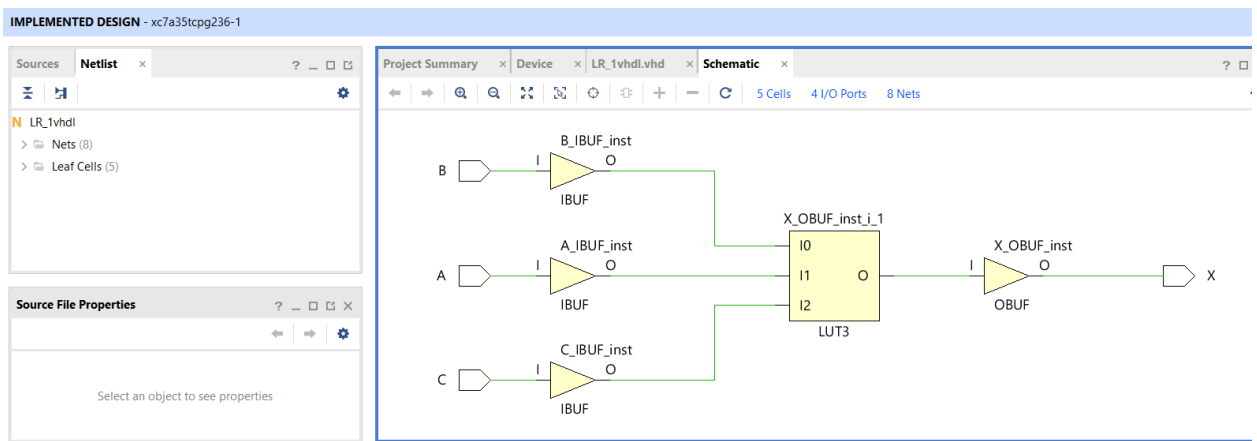


Рисунок 23а – Синтезированная схема функции $X = A \oplus B \oplus C$

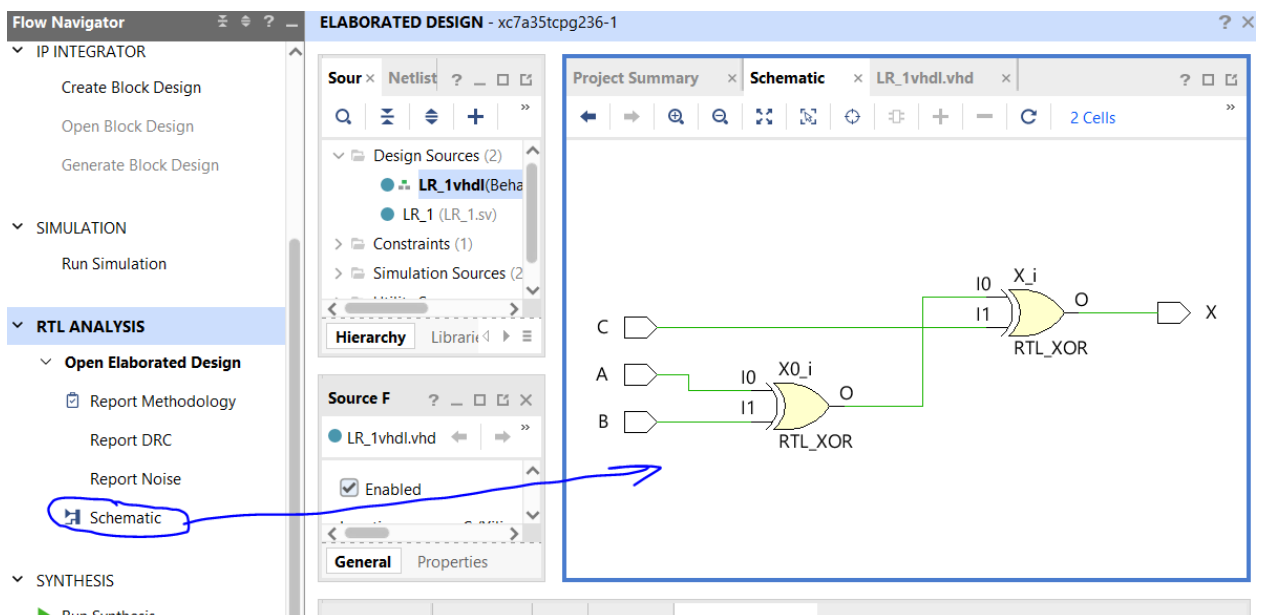


Рисунок 23б – Синтезированная схема функции $X = A \oplus B \oplus C$ из раздела «RTL ANALISYS»

3. Моделирование кода

Для проверки работы кода используется моделирование, т.е. строится временная диаграмма подачи входных сигналов и на ней получаем значения выходных сигналов. В САПР Vivado для моделирования необходимо написать сценарий моделирования на языке SystemVerilog или VHDL. В САПР QuartusII v9.1 можно создавать временные диаграммы моделирования вручную, что в некоторых случаях упрощает этот процесс. Создание проекта и пример моделирования в САПР QuartusII v9.1 показано в [этом видео](#). Саму САПР на свой ПК (в компьютерном классе уже есть эта САПР) можно скачать [по ссылке](#). Из архива просто распаковать (устанавливать не надо) в корень диск C (или D или любого другого диска). Потребуется [архиватор 7zip](#).

Для моделирования в САПР Vivado необходимо создать тестирующий модуль. В окне «Sources» во вкладке «Simulation Sources» нужно создать модуль симуляции аналогично синтезируемому модулю, только вместо «Design Sources» выбрать «Simulation Sources» (рис. 7). В файле тестирования

нет никаких входов и выходов, поэтому окно как на рис. 11 оставляем пустым, ничего там не заполняем и жмём ОК.

В результате будет создан тестирующий модуль. На рис. 24 изображён редактор с уже написанным кодом, отвечающим за инкрементирование регистра ABC_reg через каждые 50 нс; начальное значение регистра установлено в нулевое на стр. 6.

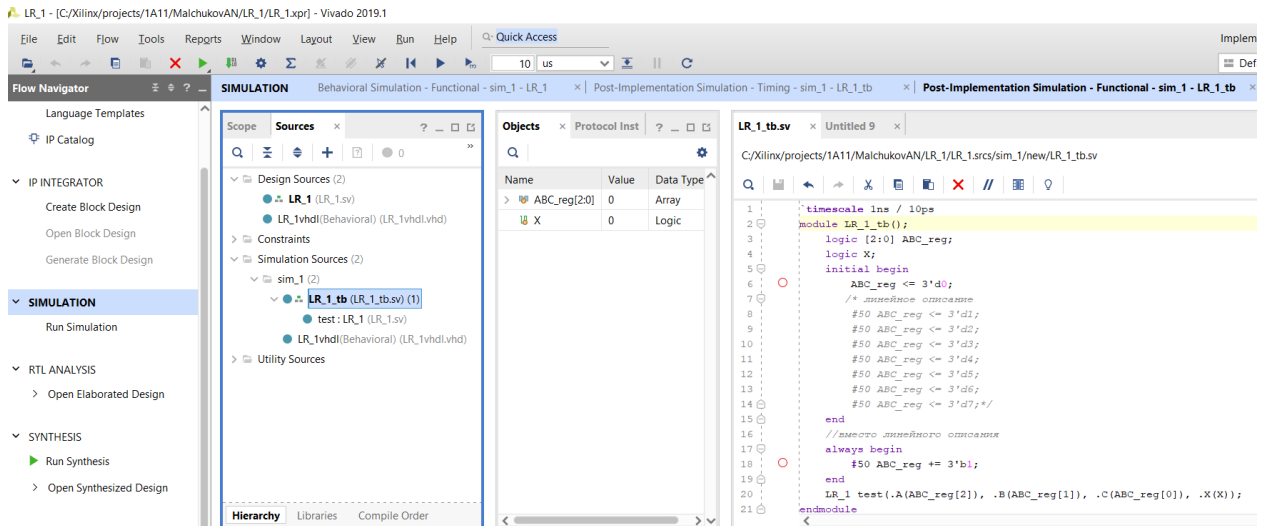


Рисунок 24 – Пример файла-теста на SytemVerilog

На стр. 3 объявляется трёхразрядный регистр ABC_reg, на следующей строке провод X – для вывода значения функции тестируемого модуля (выхода X). Далее в блоке Initial можно задать все начальные значения для входных переменных, а также используя линейный стиль описания можно последовательно задать значения каждого входного сигнала указав #задержку. Размерность единицы #задержки указывается в первой строке «`timescale 1 » через / указывается точность воспроизведения сигнала « / 10 ps».

Поведенческое описание, приведённое на строках 17-29 позволяет упростить задание входных сигналов, когда это возможно. На этих строках указан блок always, который говорит о том, что необходимо выполнять инструкции внутри блока в течении всего времени моделирования, на стр. 18 указано каждые 50 единиц задержки (в нашем случае единица задержки 1 нс, т.е. каждые 50 нс) увеличивать значение регистра на единицу. Константа 3'b1, где 3 – это количество бит (разрядов) регистра, b – двоичное основание константы и 1 – значение константы в указанной системе счисления. На стр. 6 используется другая константа – 3'd0, где 3 – также количество бит регистра, d – десятичная система счисления и 0 – значение; применяется десятичная система счисления, чтобы все разряды заполнить нулями, указывая только один ноль, т.к. в двоичном виде надо было бы написать 3'b000. Для указания констант в шестнадцатеричной системе счисления используется буква h (например 3'h0).

Тестируемый модуль подключен на стр. 20 «LR_1 test(.A(ABC_reg[2]), .B(ABC_reg[1]), .C(ABC_reg[0]), .X(X));». Сначала идёт название модуля «LR_1», затем указывается имя создаваемого экземпляра модуля «test», если один модуль подключается несколько раз, то названия его экземпляров

должны отличаться друг от друга. В круглых скобках указаны порты модуля (входы и выходы), их названия пишется через точку, например вход «.A». Входы и выходы пишутся как функции с круглыми скобками, подразумевая, что в них надо подать сигналы или для выходов подключить провода для вывода сигналов.

По умолчанию файлы VHDL создаются в нотации 1992 года, которая не поддерживает некоторые полезные изменения, добавленные в нотации 2008 года. Для того чтобы сменить нотации VHDL файла, необходимо в разделе «Sources» нажать ПКМ на файле и в меню выбрать «Source Node Properties...» (рис. 25). В появившемся окне «Source Node Properties» необходимо изменить «Type» «VHDL» на «VHDL 2008». Пример полезных новшеств в нотации 2008-го года – это константы, в формате почти как в SystemVerilog. Например, в нотации 1992 года константа для STD_LOGIC_VECTOR всегда пишется двоичным числом “1100”, а в 2008-ой нотации можно писать эту константу в разных системах счисления: 4D“12”, 4X“C”. Также в 2008-ой нотации можно использовать многострочное комментирование /* много строк */. И кроме этого, в нотации 2008-го года автоматически конвертируется значение STD_LOGIC в boolean. Например, в нотации 1992 года «if some_signal = '1' then», а в 2008-ой можно написать «if some_signal then».

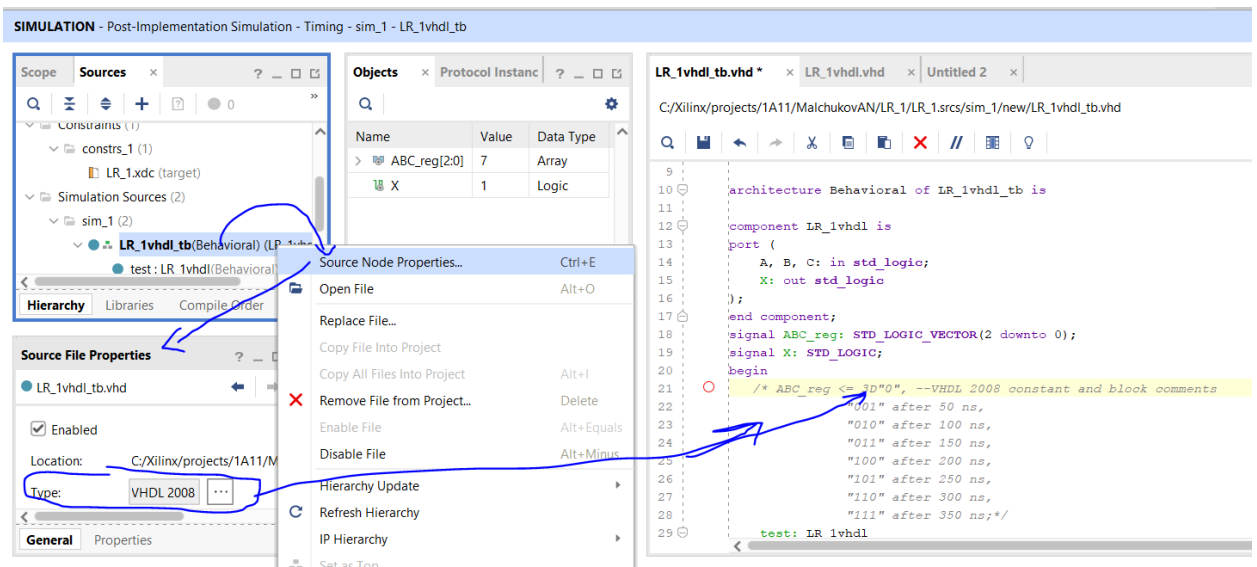


Рисунок 25 – Выбор нотации VHDL 2008-го года

Пример тестового файла на языке VHDL в нотации 2008-го года приведён на рис. 26. На стр. 2-4 все библиотеки, которые могут понадобиться в лабораторных работах. У тестовых модулей нет входов и выходов (стр. 6-8). Отличием VHDL от SystemVerilog является то, что надо объявлять все модули, которые будут подключаться (стр. 11-14). На стр. 15 объявляем трёхразрядный регистр и задаём ему нулевое начальное значение по умолчанию, используя формат константы из нотации 2008-го года. Линейный подход задания входных сигналов показан на стр. 18-25. Задержка указывается в единицах измерения времени: fs – фемтосекунда; ps – пикосекунда; ns – наносекунда; us – микросекунда; ms – миллисекунда; sec – секунда; min – минута; hr – час. Поведенческий подход показан на стр. 26-29. Если будет использован

линейный подход, то не обязательно задавать начальное значение регистра ABC_reg и необходимо будет закомментировать или убрать стр. 26-29. Подключается тестируемый модуль на стр. 30-31, где «test» – имя экземпляра модуля, если один модуль подключается несколько раз, то названия его экземпляров должны отличаться друг от друга. В круглых скобках «port map» указывается «название порта» => «подаваемый на порт сигнал или подключаемый к нему провод».

```

1      library IEEE;
2      use IEEE.STD_LOGIC_1164.ALL;
3      use IEEE.NUMERIC_STD.ALL; --для операций вычитания
4      use IEEE.STD_LOGIC_UNSIGNED.ALL; --для преобразования между STD_LOGIC_VECTOR и INTEGER
5
6      entity LR_1vhdl_tb is
7          -- Port ( ); в тестах нет входов и выходов!
8      end LR_1vhdl_tb;
9
10     architecture Behavioral of LR_1vhdl_tb is
11         component LR_1vhdl is --объявляем модуль, который подключаем
12             port (A, B, C: in std_logic;
13                  X: out std_logic);
14         end component;
15         signal ABC_reg: STD_LOGIC_VECTOR(2 downto 0) := 3D"0";
16         signal X: STD_LOGIC;
17     begin
18         /* ABC_reg <= "000", --VHDL 2008 constant and block comments
19             "001" after 50 ns, --линейное описание
20             "010" after 100 ns,
21             "011" after 150 ns,
22             "100" after 200 ns,
23             "101" after 250 ns,
24             "110" after 300 ns,
25             "111" after 350 ns;*/
26         process begin --вместо линейного описания
27             wait for 50 ns;
28             ABC_reg <= ABC_reg + '1';
29         end process;
30         test: LR_1vhdl
31             port map (A=>ABC_reg(2), B=>ABC_reg(1), C=>ABC_reg(0), X=> X);
32     end Behavioral;

```

Рисунок 26 – Пример файла-теста на VHDL 2008

Сохранив файл тестирования, нужно убедиться, что именно этот файл в «Simulation Sources» является файлом верхнего уровня, затем на левой панели «Flow Navigator» в разделе «Simulation» нажать «Run Simulation». В диалоговом окне выбрать тип моделирования (рис. 27).

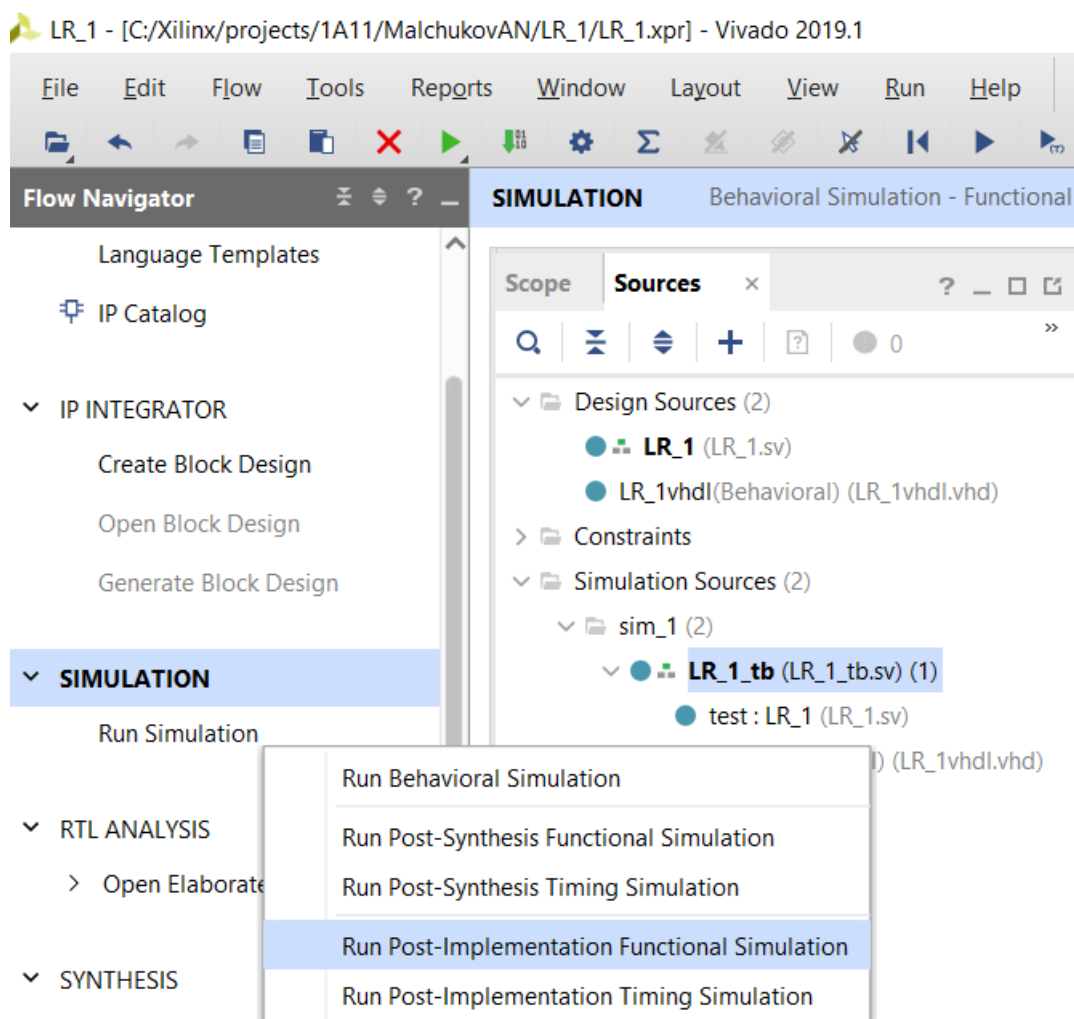



Рисунок 27 – Выбор типа моделирования

До этого тестируемый модуль должен быть успешно откомпилирован как в синтезе, так и в реализации на кристалле.

Опция моделирования «Run Post-Implementation Functional Simulation» позволяет смоделировать поведение модуля без учёта реальных задержек в кристалле (рис. 28). Необходимо использовать инструмент  для уменьшения масштаба.

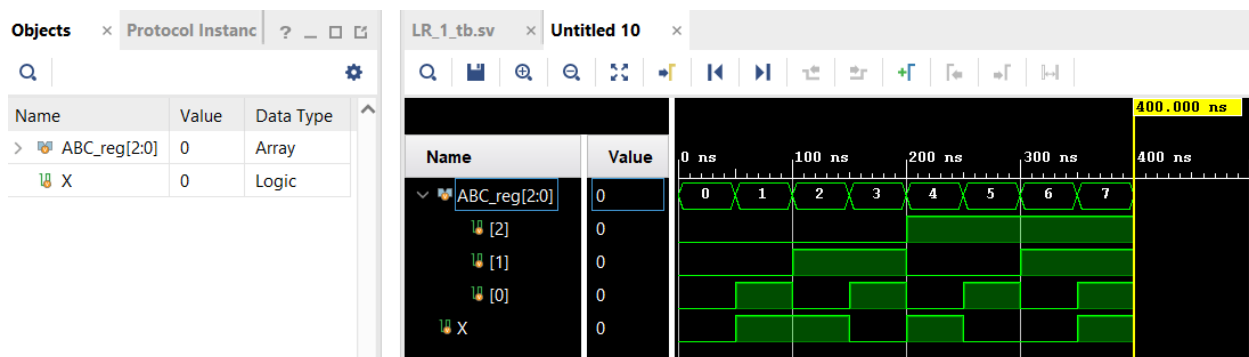


Рисунок 28 – Функциональное моделирование

Тип моделирования «Run Post-Implementation Timing Simulation» позволяет смоделировать поведение модуля с учётом реальных задержек в кристалле (рис. 29).

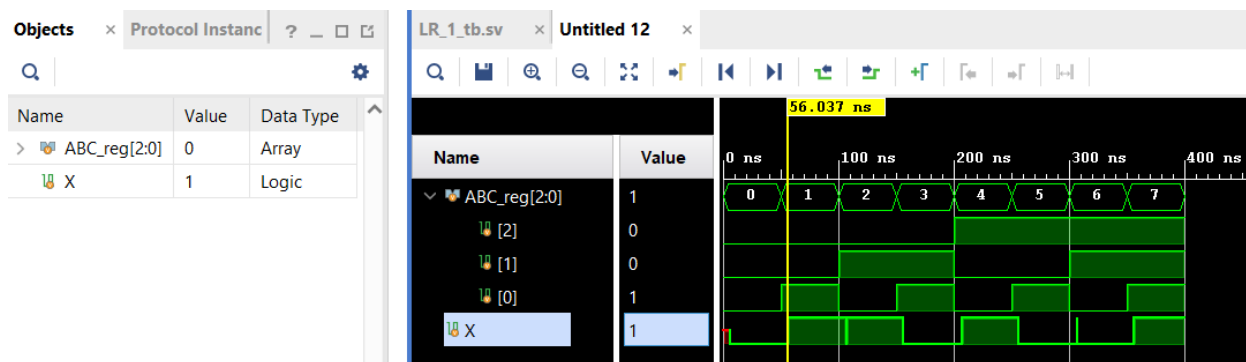


Рисунок 29 – Моделирование с учётом задержек

На рис. 29 жёлтый маркер показывает, что сигнал на выходе установился с задержкой 6,037 нс после того, как на отметке 50 нс была подана единица на вход. Задержка характеризуется временем распространения сигнала в кристалле, а также временем переключения логических ячеек. Обратите внимание на некоторые различия временной линии X на рис. 28 и 29.

По умолчанию время моделирования длится 1000 нс, чтобы его уменьшить или увеличить необходимо зайти в настройки моделирования «Tools» -> «Settings...» (см. рис. 30).

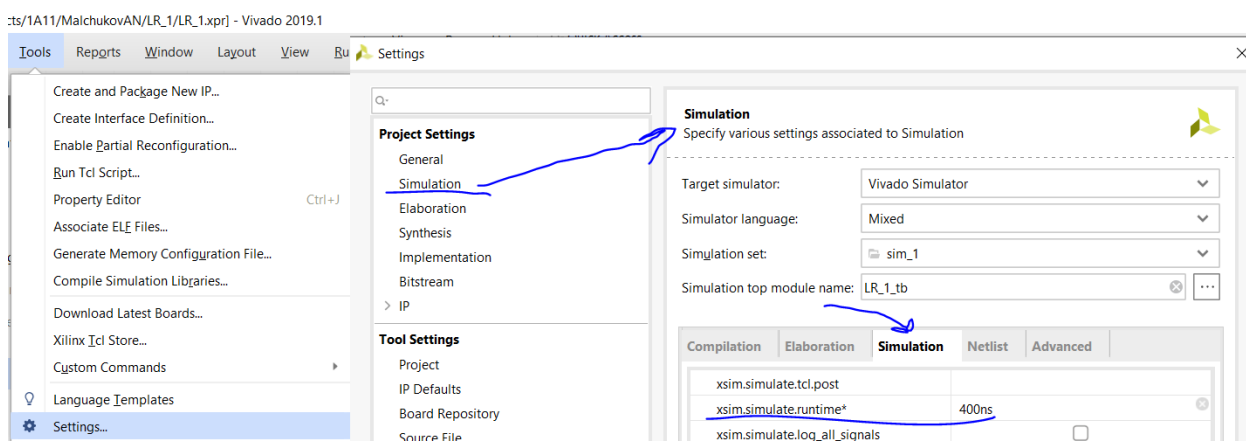


Рисунок 30 – Настройка длительности моделирования

На этом симуляция модуля закончена. Затем следует проверить работу на отладочной плате Basys 3.

4. Загрузка проекта на плату

Перед тем как создать прошивку для ПЛИС необходимо привести соответствие между входами и выходами проекта и ножками ПЛИС, чтобы связать входы и выходы с реальными устройствами на плате. Для этого необходимо открыть «I/O Ports» (см. рис. 31).

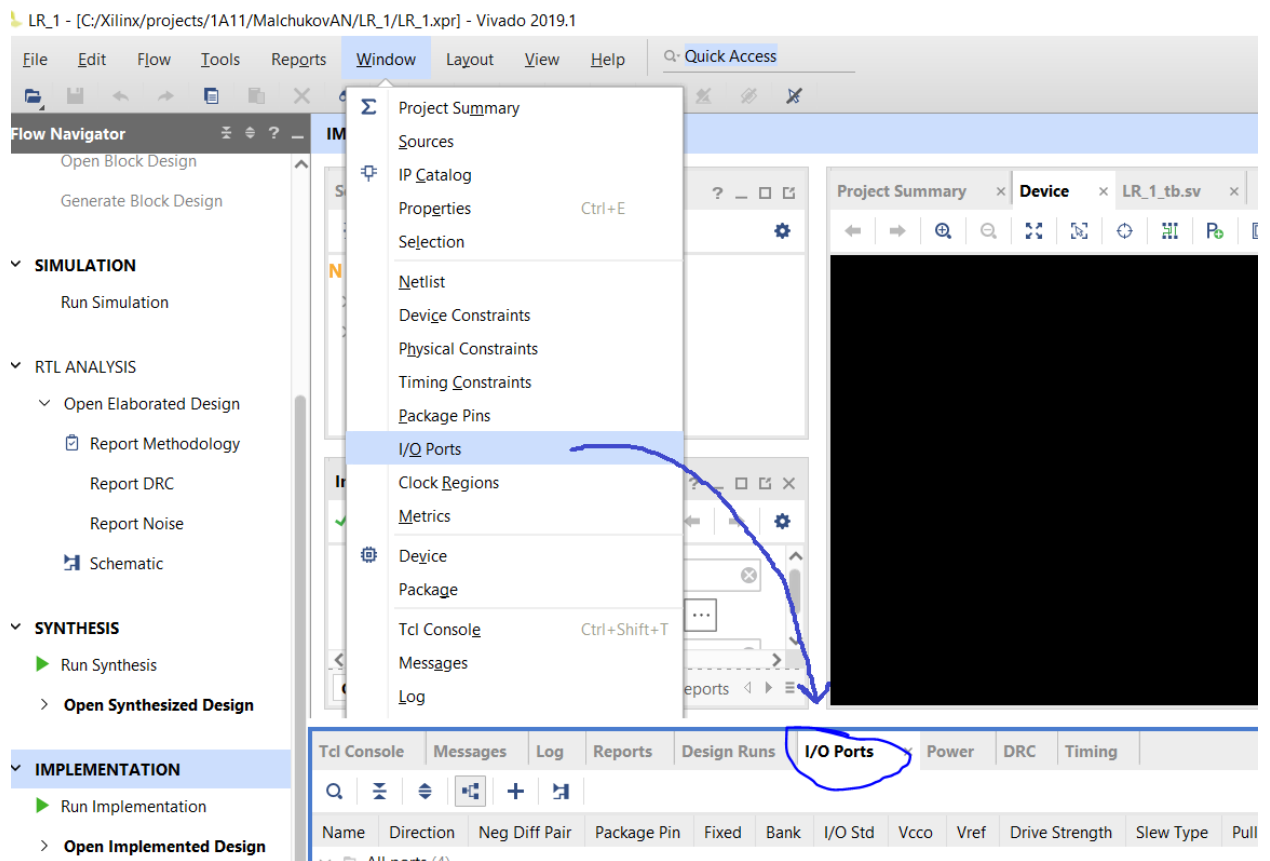


Рисунок 31 – Открытие окна для назначения соответствия портов с ножками ПЛИС

Откроется окно «I/O Ports», где определяется местоположение портов (входов и выходов) на ПЛИС и их фактическая связь с реальными устройствами на плате Basys 3 (рис. 32).

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
All ports (4)													
Scalar ports (4)													
A	IN		W16	<input checked="" type="checkbox"/>	14	LVC MOS33*	3.300				NONE	NONE	
B	IN		V16	<input checked="" type="checkbox"/>	14	LVC MOS33*	3.300				NONE	NONE	
C	IN		V17	<input checked="" type="checkbox"/>	14	LVC MOS33*	3.300				NONE	NONE	
X	OUT		U16	<input checked="" type="checkbox"/>	14	LVC MOS33*	3.300		12	SLOW	NONE	FP_VTT_50	

Рисунок 32 – Окно назначения ножек ПЛИС

При раскрытии вкладки «Scalar Ports(4)» отображаются порты, заданные в модуле. Необходимо в столбце «I/O Std» выбрать для всех сигналов «LVC MOS 3.3», что означает логику +3,3В. Далее нужно выбрать номера выводов («Package Pin») в соответствии с тем, с каких устройств на плате должны поступать сигналы на входы модуля и на какие устройства должны подаваться сигналы через выходы модуля на плату. Например, на рис. 32 показано, что на вход С модуля будет идти сигнал с самого правого движкового переключателя SW0 и ему соответствует ножка ПЛИС V17. Выход модуля, т.е. значение функции X будет подаваться на самый правый светодиод LD0, которому соответствует ножка ПЛИС U16 (см. рис. 33).

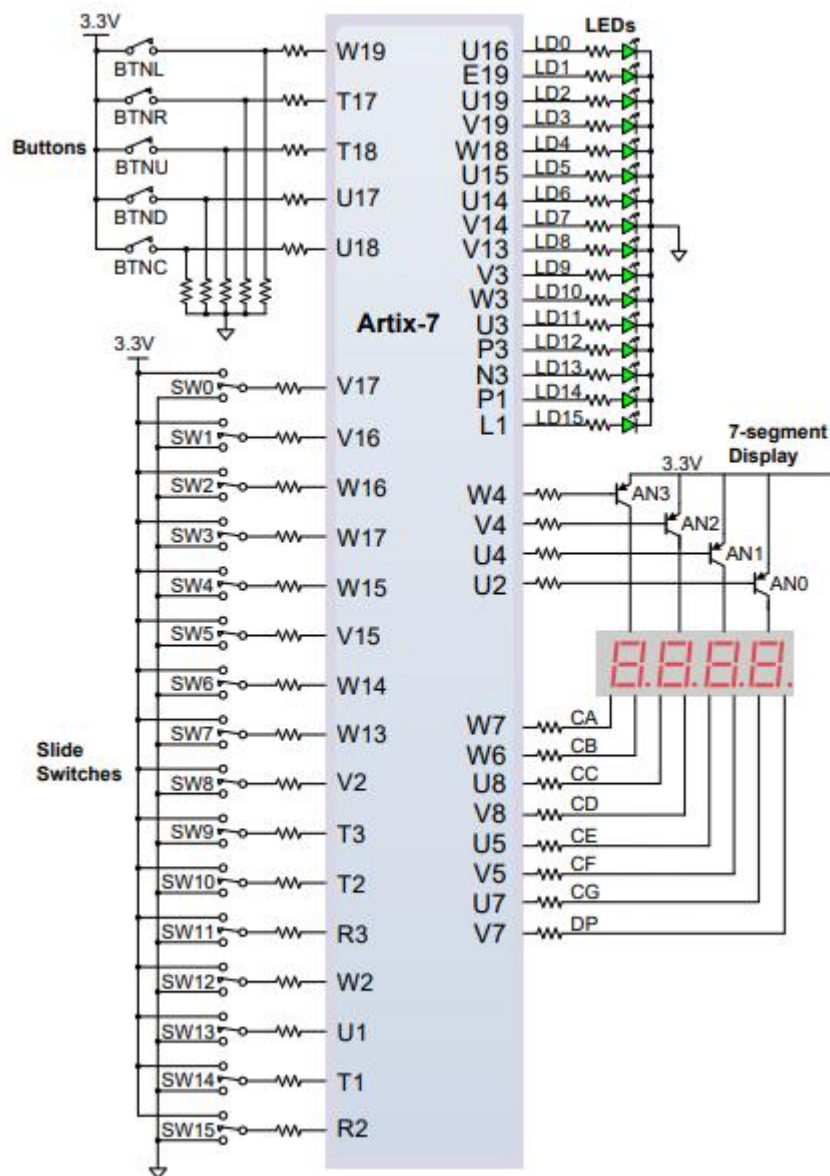



Рисунок 33 – Ножки ПЛИС для кнопок, движковых переключателей, светодиодов и семисегментных индикаторов и их подключение

После назначения ножек и «I/O Std» нажимаем сохранить (Ctrl+S или иконка дискеты  в самом левом углу) файл XDC, указав его название и нажав ОК (рис. 34).

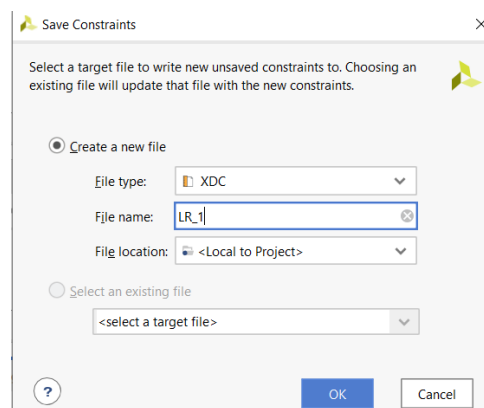


Рисунок 34 – Сохранение XDC файла

После сохранения XDC файла необходимо убедиться в его корректности. В «Sources» открываем наш файл LR_1.xdc (рис. 35).

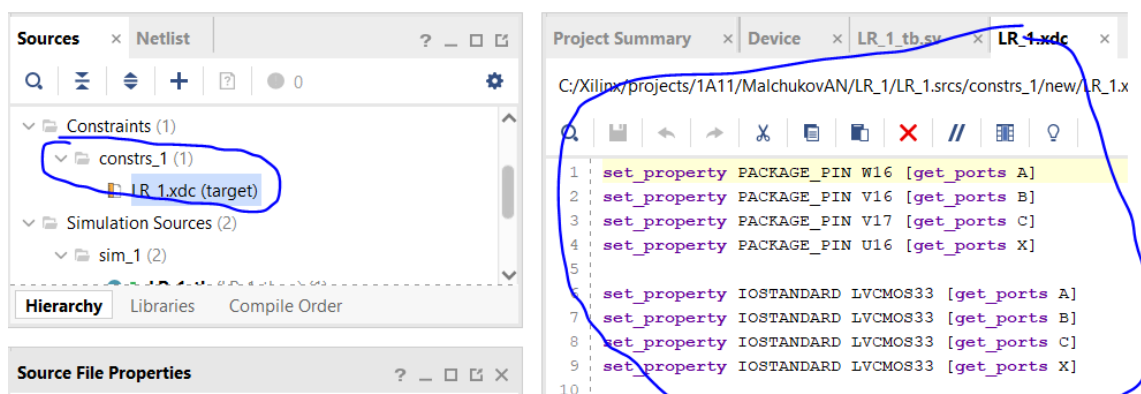


Рисунок 35 – LR_1.xdc файл

Далее нужно еще раз выполнить «Run Synthesis» вместе с «Implementation». Либо после синтеза на левой панели «Flow Navigator» в разделе «Implementation» выполнить «Run Implementation». После выполнения «Implementation» выбрать «Generate Bitstream» и нажать OK (рис. 36). Можно вручную запустить «Generate Bitstream» найдя эту опцию на левой панели «Flow Navigator» в разделе «PROGRAM AND DEBUG».

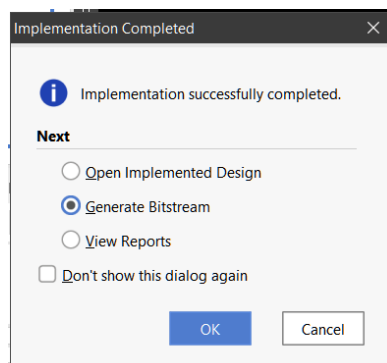


Рисунок 36 – Выбор генерации прошивки ПЛИС

После генерации прошивки выбрать «Open Hardware Manager» и нажать OK (рис. 38). Можно вручную запустить «Open Hardware Manager» найдя эту опцию на левой панели «Flow Navigator» в разделе «PROGRAM AND DEBUG».

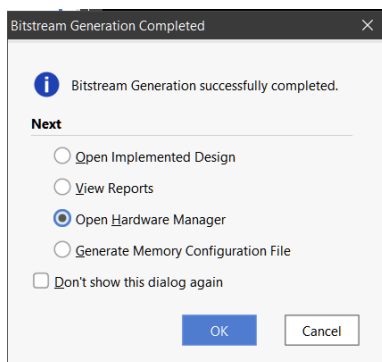


Рисунок 37 – Завершение генерации прошивки

Теперь необходимо подключить отладочную плату Basys 3 к USB порту ПК и включить плату переключателем возле USB порта платы. Затем, в

верхней части экрана в зеленом поле выбрать «Open Target» (рис. 38). В диалоговом окне выбрать «Auto Connect».

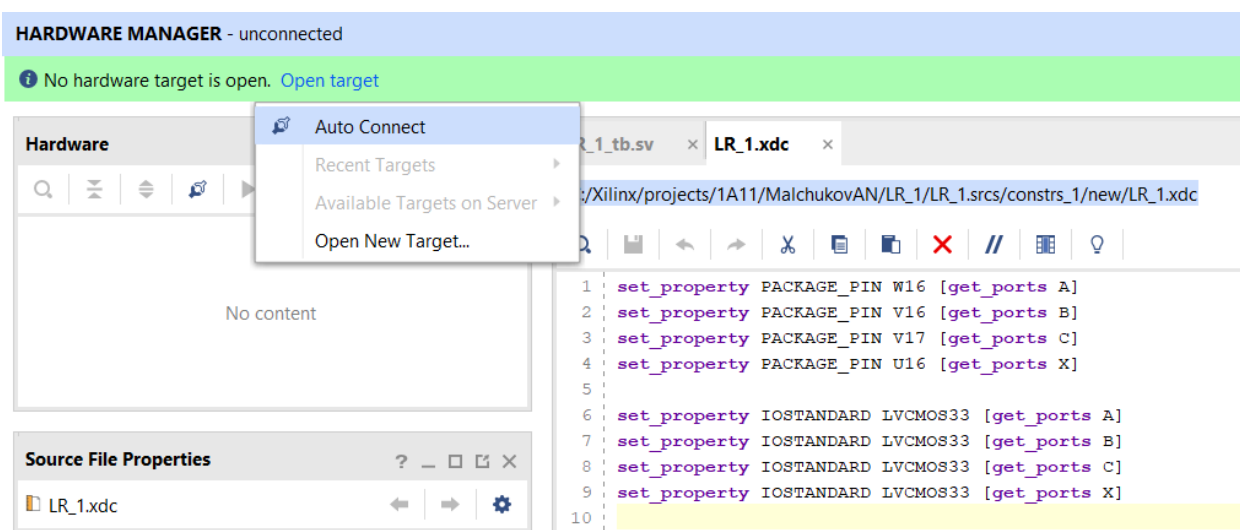


Рисунок 38 – Сообщение Hardware Manager – Open target

Затем там же выбрать «Program device» и в окне «Program Device» нажать кнопку Program (рис. 39)

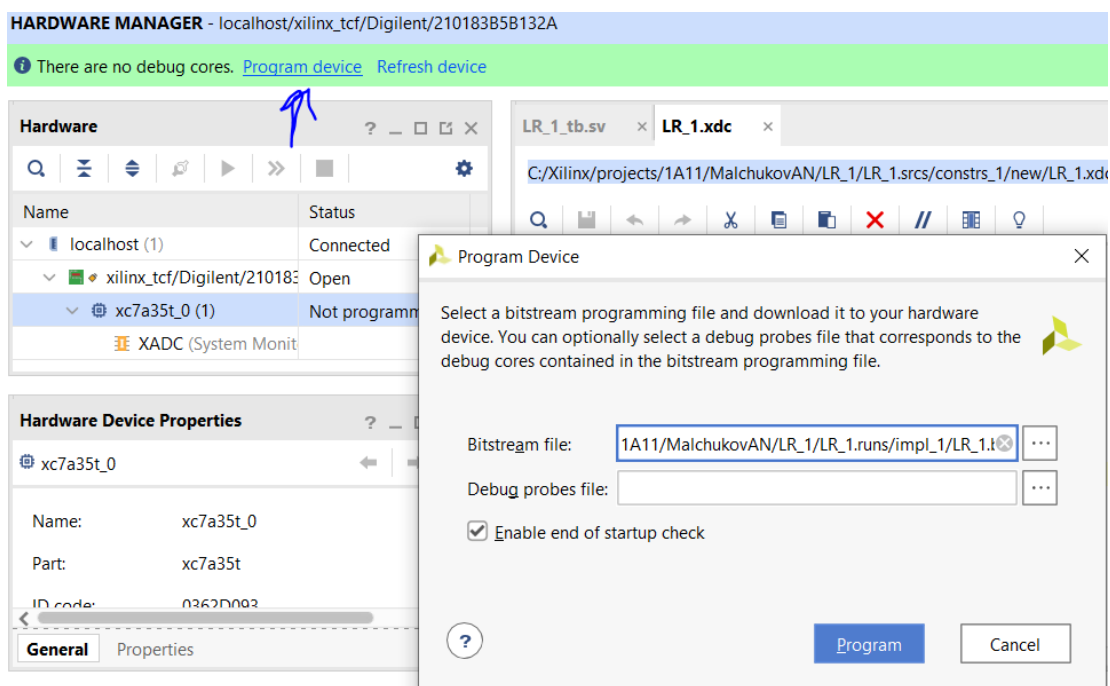
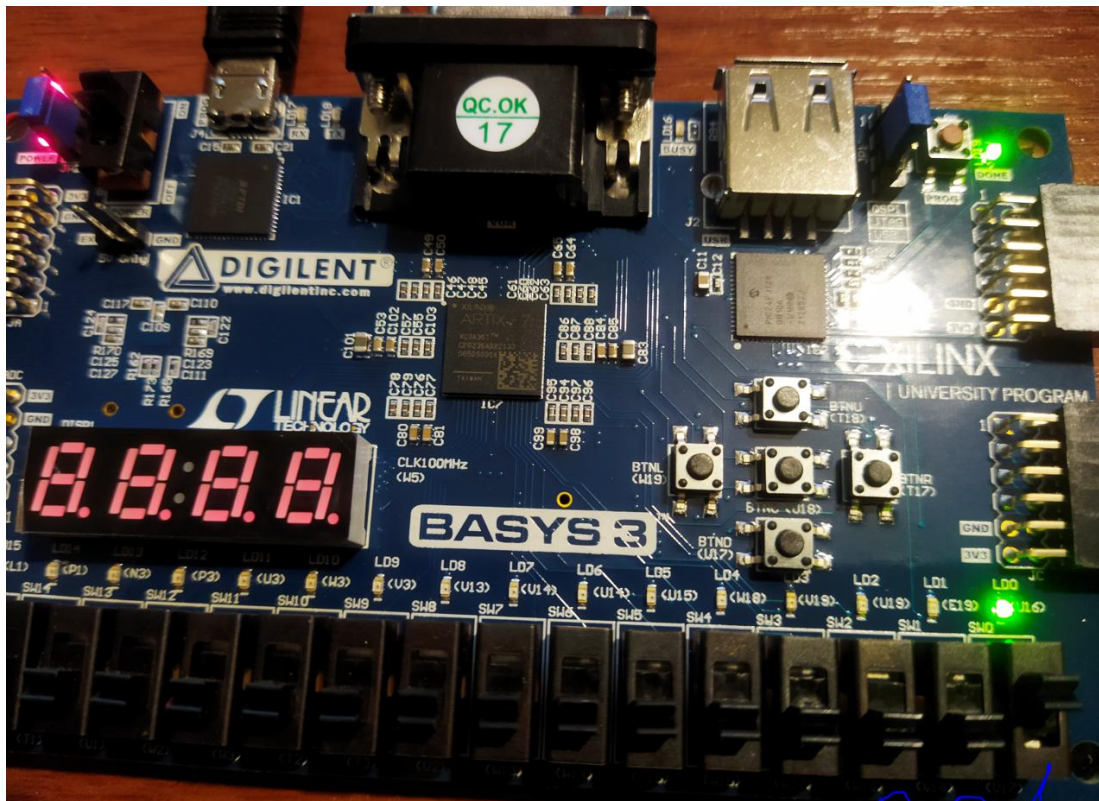


Рисунок 39 – Сообщение Hardware Manager – Program device

После успешной прошивки ПЛИС на отладочной плате можно проверить работу функции $X=A \oplus B \oplus C$. Переключая движковые переключатели SW2-SW0, будет меняться состояние светодиода LD0 в соответствии с таблицей истинности функции табл. 1 (рис. 40).



0 0 1
A B C

Рисунок 40 – Пример работы прошивки на плате при ABC=001

На этом знакомство со средой разработки Vivado окончено.

5. Примеры логических элементов

Примеры логических функций на языке SystemVerilog приведены на рис. 41, а на VHDL на рис. 42. Тестовые файлы на SystemVerilog – рис. 43, VHDL – рис. 44. Результаты моделирования с задержками – рис. 45. Схема для кода SystemVerilog и VHDL приведена на рис. 46.

C:/Xilinx/projects/1A11/MalchukovAN/test1_logic_gates/test1_logic_gates.srscs/sources_1/new/logic_gates_sv.sv

```

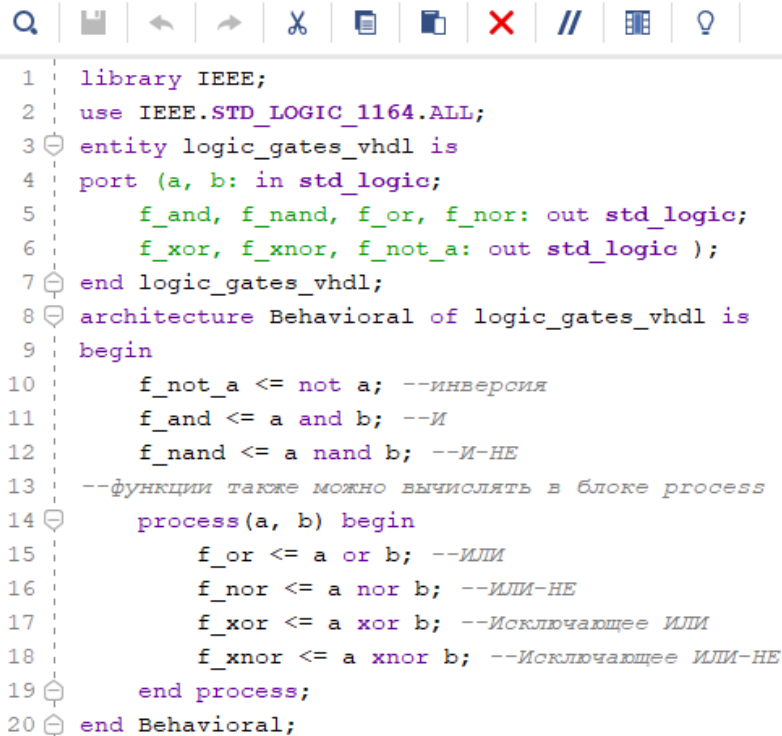
1 module logic_gates_sv(
2   input a, b,
3   output logic f_and, f_nand, f_or, f_nor,
4     f_xor, f_xnor, f_not_a);
5   assign f_not_a = ~a; //инверсия
6   assign f_and = a & b; //И
7   assign f_nand = ~(a & b); //И-НЕ
8   //функции также можно вычислять в комбинационном блоке
9   always_comb begin
10    f_or = a | b; //ИЛИ
11    f_nor = ~(a | b); //ИЛИ-НЕ
12    f_xor = a ^ b; //Исключающее ИЛИ
13    f_xnor = ~(a ^ b); //Исключающее ИЛИ-НЕ
14  end
15 endmodule

```

Рисунок 41 – Пример логических функций на языке SystemVerilog

На рис. 41 название модуля указано на стр. 1; на стр. 2-3 описаны входы и выходы; на стр. 5-7 через «assign» показано вычисление лог. функций НЕ, И, И-НЕ; на строках 9-14 с помощью комбинационного блока вычисляются лог. функции ИЛИ, ИЛИ-НЕ, искл.ИЛИ и искл.ИЛИ-НЕ.

C:/Xilinx/projects/1A11/MalchukovAN/test1_logic_gates/test1_logic_gates.srscs/sources_1/new/logic_gates_vhdl.vhd



```

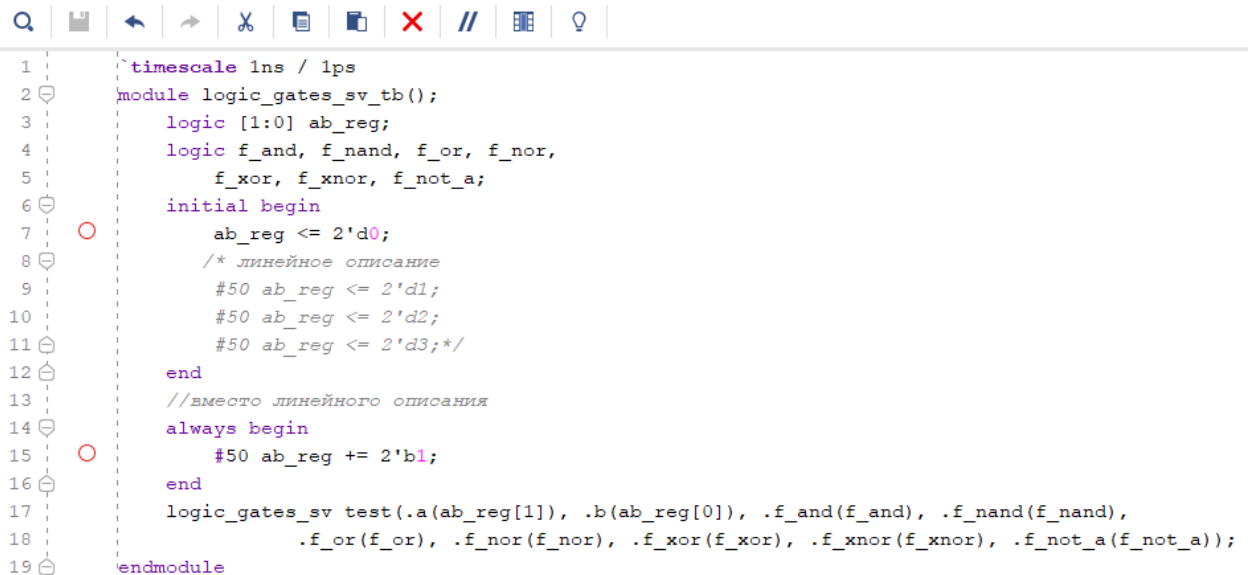
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  entity logic_gates_vhdl is
4  port (a, b: in std_logic;
5        f_and, f_nand, f_or, f_nor: out std_logic;
6        f_xor, f_xnor, f_not_a: out std_logic );
7  end logic_gates_vhdl;
8  architecture Behavioral of logic_gates_vhdl is
9  begin
10     f_not_a <= not a; --инверсия
11     f_and <= a and b; --И
12     f_nand <= a nand b; --И-НЕ
13     --функции также можно вычислять в блоке process
14     process(a, b) begin
15         f_or <= a or b; --ИЛИ
16         f_nor <= a nor b; --ИЛИ-НЕ
17         f_xor <= a xor b; --Исключающее ИЛИ
18         f_xnor <= a xnor b; --Исключающее ИЛИ-НЕ
19     end process;
20 end Behavioral;

```

Рисунок 42 – Пример логических функций на языке VHDL

На рис. 42 подключение библиотек приводится на стр. 1 и 2; название модуля указано на стр. 3; на стр. 4-6 описаны входы и выходы; на стр. 10-12 показано вычисление лог. функций НЕ, И, И-НЕ; на строках 14-19 с помощью процессного блока вычисляются лог. функции ИЛИ, ИЛИ-НЕ, искл.ИЛИ и искл.ИЛИ-НЕ, в качестве списка чувствительности процессного блока указаны входы, использующиеся для вычисления лог. функций.

На рис. 43 с помощью «timescale» указана размерность задержки и через «/» указана точность задержки на стр. 1; название файла тестирования указано на стр. 2; у файла тестирования не может быть входов и выходов; регистр на два разряда описан на стр. 3; на стр. 4 и 5 описаны провода для вывода значений функций тестируемого модуля; на стр. 6 начало блока инициализации с помощью которого задаются начальные значения входных сигналов (стр. 7), а также можно описать значение входных сигналов в течение всего времени моделирования линейно (стр. 9-11); на стр. 14-16 приведено задание значений входных сигналов для всего времени моделирования поведенческим способом, т.е. каждые 50 нс увеличивать регистр на единицу, таким образом задаётся двоичный перебор входных сигналов; на стр. 17 и 18 подключение тестируемого модуля.

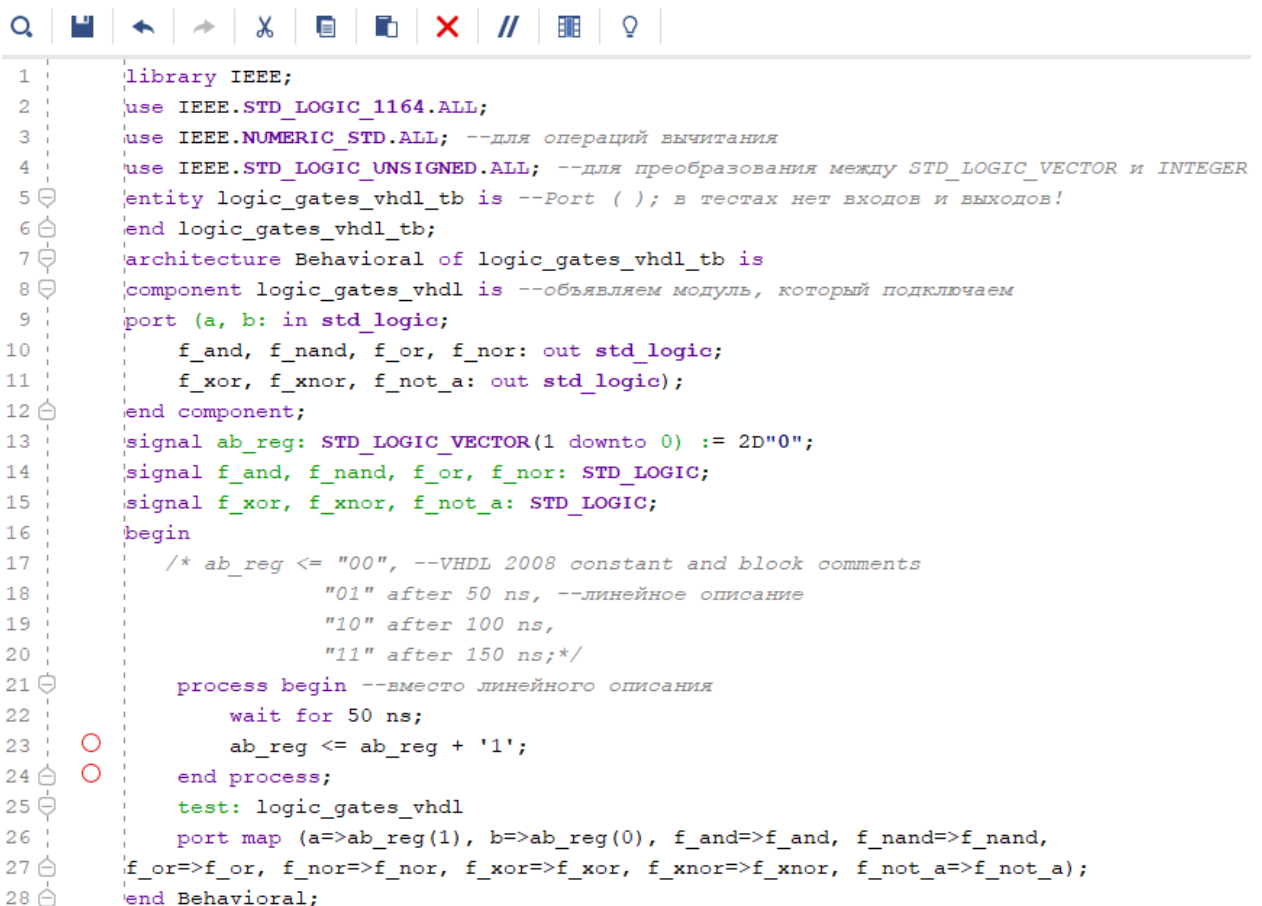


```

1  timescale 1ns / 1ps
2  module logic_gates_sv_tb();
3      logic [1:0] ab_reg;
4      logic f_and, f_nand, f_or, f_nor,
5          f_xor, f_xnor, f_not_a;
6      initial begin
7          ab_reg <= 2'd0;
8          /* линейное описание
9             #50 ab_reg <= 2'd1;
10             #50 ab_reg <= 2'd2;
11             #50 ab_reg <= 2'd3;*/
12      end
13      //вместо линейного описания
14      always begin
15          #50 ab_reg += 2'b1;
16      end
17      logic_gates_sv test(.a(ab_reg[1]), .b(ab_reg[0]), .f_and(f_and), .f_nand(f_nand),
18          .f_or(f_or), .f_nor(f_nor), .f_xor(f_xor), .f_xnor(f_xnor), .f_not_a(f_not_a));
19  endmodule

```

Рисунок 43 – Файл тестирования на языке SystemVerilog



```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL; --для операций вычитания
4  use IEEE.STD_LOGIC_UNSIGNED.ALL; --для преобразования между STD_LOGIC_VECTOR и INTEGER
5  entity logic_gates_vhdl_tb is --Port ( ); в тестах нет входов и выходов!
6  end logic_gates_vhdl_tb;
7  architecture Behavioral of logic_gates_vhdl_tb is
8  component logic_gates_vhdl is --объявляем модуль, который подключаем
9  port (a, b: in std_logic;
10      f_and, f_nand, f_or, f_nor: out std_logic;
11      f_xor, f_xnor, f_not_a: out std_logic);
12  end component;
13  signal ab_reg: STD_LOGIC_VECTOR(1 downto 0) := 2D"0";
14  signal f_and, f_nand, f_or, f_nor: STD_LOGIC;
15  signal f_xor, f_xnor, f_not_a: STD_LOGIC;
16  begin
17      /* ab_reg <= "00", --VHDL 2008 constant and block comments
18         "01" after 50 ns, --линейное описание
19         "10" after 100 ns,
20         "11" after 150 ns;*/
21      process begin --вместо линейного описания
22          wait for 50 ns;
23          ab_reg <= ab_reg + '1';
24      end process;
25      test: logic_gates_vhdl
26      port map (a=>ab_reg(1), b=>ab_reg(0), f_and=>f_and, f_nand=>f_nand,
27          f_or=>f_or, f_nor=>f_nor, f_xor=>f_xor, f_xnor=>f_xnor, f_not_a=>f_not_a);
28  end Behavioral;

```

Рисунок 44 – Файл тестирования на языке VHDL

На рис. 44 на стр. 1-4 указаны все необходимые библиотеки; название файла тестирования указано на стр. 5; у файла тестирования не может быть входов и выходов; любой подключаемый модуль в VHDL сначала необходимо объявить (стр. 8-12); регистр на два разряда описан на стр. 13, в том числе сразу задаём ему начальное значение 0 с помощью константы «2D"0"»,

которая соответствует нотации 2008 года; на стр. 14 и 15 описаны провода для вывода значений функций тестируемого модуля; на стр. 17-20 с помощью линейного описано заданы значения входных сигналов на всё время моделирования; на стр. 21-24 приведено задание значений входных сигналов для всего времени моделирования поведенческим способом, т.е. каждые 50 нс увеличивать регистр на единицу, таким образом задаётся двоичный перебор входных сигналов; на стр. 25-27 подключение тестируемого модуля.

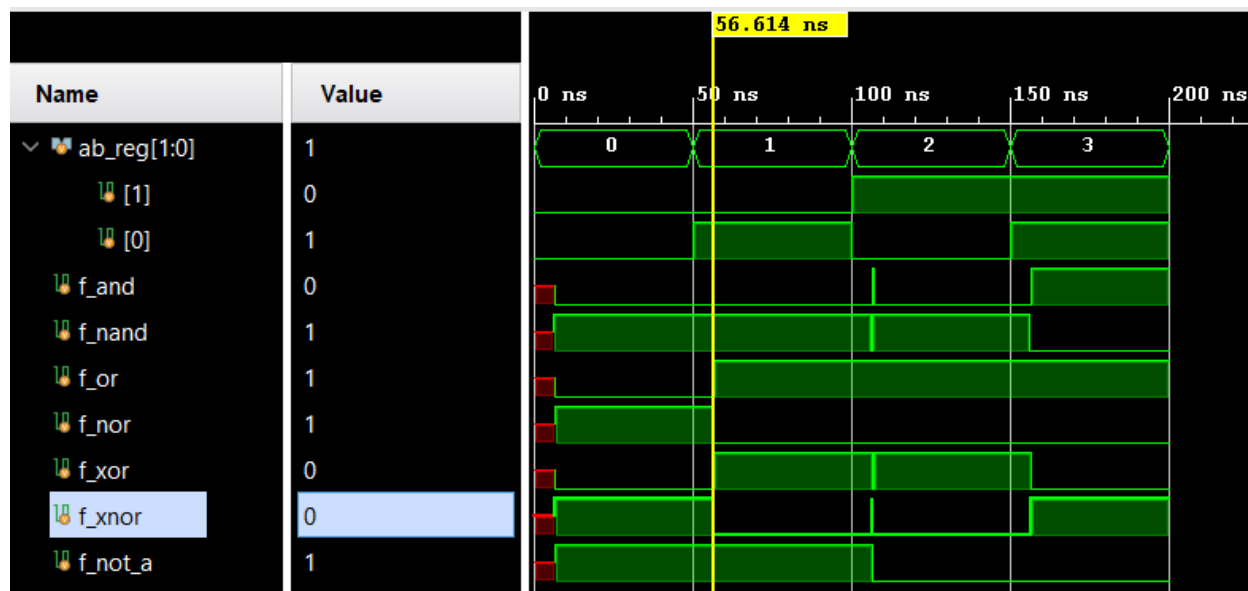


Рисунок 45 – Результат моделирования логических функций с задержками

На рис. 45 на входы «а» и «b» с помощью регистра ab_reg задаются сигналы. На промежутке времени 0-50 нс значение входных сигналов a=0 и b=0, на выходах функций И=0, ИЛИ-НЕ=1, ИЛИ=0, ИЛИ-НЕ=1, искл.ИЛИ=0, искл.ИЛИ-НЕ=1, инверсия входа a = 1; по умолчанию на выходах неизвестное состояние, задержка изменения выходных сигналов в 6,614 нс обусловлена временем распространения сигнала внутри кристалла (упрощённо: от входов (ножек) кристалла до логических ячеек и от выходов логических ячеек до выходов (ножек) кристалла) и временем переключения логической ячейки. На промежутке времени 50-100 нс значение входных сигналов a=0 и b=1, на выходах функций И=0, ИЛИ-НЕ=1, ИЛИ=1, ИЛИ-НЕ=0, искл.ИЛИ=1, искл.ИЛИ-НЕ=0, инверсия входа a = 1; задержка изменения выходных такая же. На промежутке времени 100-150 нс значение входных сигналов a=1 и b=0, на выходах функций И=0, ИЛИ-НЕ=1, ИЛИ=1, ИЛИ-НЕ=0, искл.ИЛИ=1, искл.ИЛИ-НЕ=0, инверсия входа a = 0; задержка изменения выходных такая же; в районе 106,6 нс на выходах лог. функций И, искл.ИЛИ-НЕ наблюдаются всплески, а для лог. функций И-НЕ, искл. ИЛИ – просадки; это обусловлено тем, что сигналы «а» и «b» внутри кристалла имеют разное время пути от входа (ножек) кристалла до входа логических ячеек (гонки сигналов); в данном случае сигнал «а» имеет меньшее время «пути», чем сигнал «b», таким образом возникает момент времени, когда сигнал «а» уже равен «1», а сигнал «b» ещё равен «1» и не переключился в «0»; длительность всплеска или провала – это разность между временем «путей» сигналов «а» и «b» или по-

другому это можно назвать разной длиной проводов «а» и «b», где провод «а» короче.

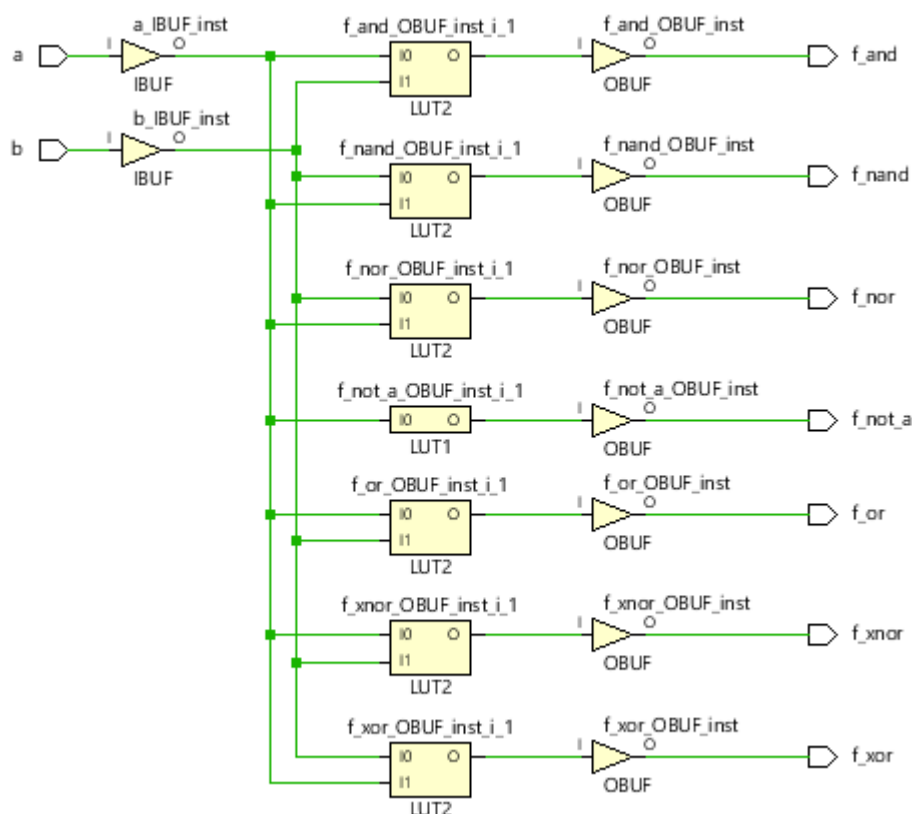


Рисунок 46 – Схема для файлов logic_gates_sv(vhdl)

На рис. 46 приведена синтезированная САПром схема для кода, на ней представлены входы «а» и «b», а также выходы лог. функций И, И-НЕ, ИЛИ-НЕ, инверсии входа «а», ИЛИ, искл. ИЛИ-НЕ и искл. ИЛИ; прежде чем попасть на соответствующие функции с ножек ПЛИС входные сигналы проходят через входные буферы; после вычисления соответствующих лог. функций выходные сигналы сначала проходят через выходные буферы, а затем попадают на выходы ПЛИС.

На рис. 47 показана осмысленная схема из раздела «RTL ANALISYS». Приведённые на ней элементы соответствуют логическим операциям.

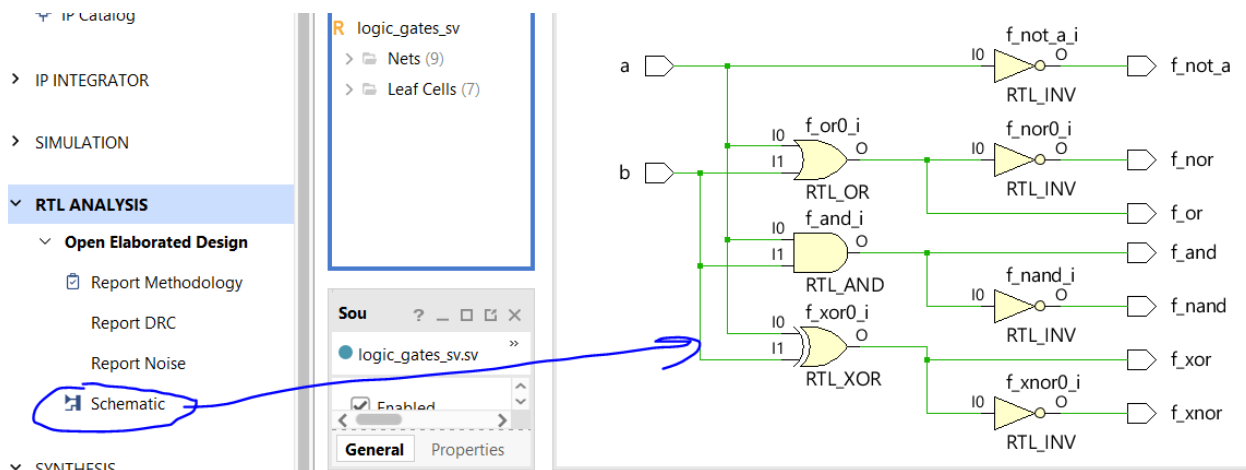


Рисунок 47 – Схема из раздела «RTL ANALISYS» для файлов logic_gates_sv(vhdl)

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. В формуле из табл. 2 для своего варианта расставьте скобки по своему усмотрению, не считаясь с приоритетом выполнения операций. Номер варианта см. в гугл-таблице, [ссылка на которую лежит в текстовом файле в папке «гугл-таблицы»](#).
2. Для полученной формулы со скобками составьте таблицу истинности.
3. Напишите код вычисления формулы на HDL согласно варианту.
4. Промоделируйте работу написанного кода, тестовый файл должен быть написан на том же HDL. Моделирование выполнить в двух режимах: «Run Post-Implementation Functional Simulation» и «Run Post-Implementation Timing Simulation».
5. Сравните результаты моделирования с составленной таблицей истинности. Если они совпадают, приступайте к следующему пункту.
6. Согласно табл. 2 назначьте соответствующие ножки ПЛИС для входов и выходов.
7. Покажите преподавателю результаты моделирования и назначенные ножки ПЛИС, чтобы получить плату Basys 3.
8. Проверьте работу написанного кода на плате Basys 3 в соответствии с составленной таблицей истинности.
9. Покажите преподавателю работу платы Basys 3.
10. Сделайте несколько фотографий работы кода на плате Basys 3.
11. Составьте отчёт согласно требованиям раздела «содержание отчёта».

Таблица 2

Варианты задания

№	Формула	Движковые переключатели	Светодиод	HDL
1	$X = A \cap B \cup C \cup D$	SW3-0	LD0	SystemVerilog
2	$X = A \cup B \cap C \cup D$	SW4-1	LD1	VHDL
3	$X = A \cup B \cup C \cap D$	SW5-2	LD2	SystemVerilog
4	$X = A \cap \overline{B} \cup \overline{C} \cup D$	SW6-3	LD3	VHDL
5	$X = A \cup B \cap \overline{C} \cup \overline{D}$	SW7-4	LD4	SystemVerilog
6	$X = \overline{A} \cup \overline{B} \cup C \cap D$	SW8-5	LD5	VHDL
7	$X = \overline{A} \cap \overline{B} \cup C \cup D$	SW9-6	LD6	SystemVerilog
8	$X = A \cup \overline{B} \cap \overline{C} \cup D$	SW10-7	LD7	VHDL
9	$X = A \cup B \cup \overline{C} \cap \overline{D}$	SW11-8	LD8	SystemVerilog
10	$X = A \cap B \cup C \oplus D$	SW12-9	LD9	VHDL
11	$X = A \oplus B \cap C \cup D$	SW13-10	LD10	SystemVerilog
12	$X = A \cup B \oplus C \cap D$	SW14-11	LD11	VHDL
13	$X = A \cap \overline{B} \cup \overline{C} \oplus D$	SW15-12	LD12	SystemVerilog
14	$X = A \oplus B \cap \overline{C} \cup \overline{D}$	SW15-12	LD13	VHDL

№	Формула	Движковые переключатели	Светодиод	HDL
15	$X = \overline{A \cup B} \oplus C \cap D$	SW14-11	LD14	SystemVerilog
16	$X = \overline{A \cap B} \cup C \oplus D$	SW13-10	LD15	VHDL
17	$X = A \oplus \overline{B \cap C} \cup D$	SW12-9	LD0	SystemVerilog
18	$X = A \cup B \oplus \overline{C \cap D}$	SW11-8	LD1	VHDL
19	$X = A \cap B \cup \overline{C \oplus D}$	SW10-7	LD2	SystemVerilog
20	$X = \overline{A \oplus B} \cap C \cup D$	SW9-6	LD3	VHDL
21	$X = A \cup \overline{B} \oplus \overline{C \cap D}$	SW8-5	LD4	SystemVerilog
22	$X = A \cap \overline{B \cup C} \oplus \overline{D}$	SW7-4	LD5	VHDL
23	$X = \overline{A \oplus B} \cap C \cup D$	SW6-3	LD6	SystemVerilog
24	$X = A \cup \overline{B} \oplus \overline{C \cap D}$	SW5-2	LD7	VHDL

СОДЕРЖАНИЕ ОТЧЕТА

На титульный лист отчёта вставить свою подпись. Подпись преподавателя из шаблона не убирать. Под каждой подписью место под дату. Даты под каждой подписью проставить, они должны совпадать с датой отправки отчёта. Без подписи отчёты будут возвращаться на доработку. Предпочтительный формат отчёта pdf, в крайнем случае docx или doc. Другие форматы не принимаются. Отчет по лабораторной работе должен быть оформлен согласно шаблону ([см. папку «шаблоны отчётов»](#)) и содержать следующие разделы.

1) цель работы и постановка задачи.

В разделе описывается цель работы.

2) постановка задачи.

В разделе описываются задачи на лабораторную работу (как общие для всех, так и задачи своего варианта), которые необходимо выполнить.

3) таблица истинности.

В разделе приводится формула с расставленными скобками и для неё таблица истинности.

4) код на HDL.

В разделе приводятся код на HDL и его описание.

5) функциональная схема.

В разделе приводятся функциональная схема из раздела «RTL ANALISYS» и её описание.

6) моделирование.

В разделе приводятся код файла тестирования и результат моделирования кода в обоих режимах, его описание и сравнение с таблицей истинности. Объяснение разницы результатов моделирования в двух режимах. На экранных снимках обязательно должны быть временные шкалы и названия

входов и выходов. Последовательность комбинаций входных сигналов должны повторять таблицу истинности – первый временной интервал соответствовать первой строке таблицы, второй временной интервал – второй строке таблицы и т.д.

7) назначение ножек ПЛИС.

В разделе приводятся содержание файла xdc и его описание.

8) фотографии макета.

В разделе приводятся фотографии с работающей прошивкой ПЛИС и их описания.

9) выводы.

Приводятся выводы о проделанной работе: в краткой форме описывается что было сделано и какие результаты были получены.

Заголовки первого уровня такие как «ЦЕЛЬ РАБОТЫ», «ПОСТАНОВКА ЗАДАЧИ» и «ВЫВОДЫ» – не нумеруются. Остальные разделы должны быть пронумерованы.

ПРИМЕРНЫЙ СПИСОК ВОПРОСОВ ПРИ ЗАЩИТЕ РАБОТЫ

1. Чем отличается ПЛИС от микроконтроллера?
2. Каким образом в САПР Vivado можно моделировать работу кода?
Есть ли другие способы?
3. Что такое HDL?
4. Что необходимо сделать, чтобы запустить на плате Basys 3 написанный на HDL код?
5. Перечислить логические элементы.
6. Написать формулы логических функций до двух переменных.
7. Нарисовать условно-графические обозначения (УГО) логических элементов отечественные и зарубежные.
8. Написать обозначения логических операций на SystemVerilog.
9. Написать обозначения логических операций на VHDL.
10. Надо ли расставлять скобки при вычислении логических функций на HDL (пояснить ответ)?