МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

ЗНАКОМСТВО С ПРОГРАММИРУЕМОЙ ЛОГИЧЕСКОЙ ИНТЕГРАЛЬНОЙ СХЕМОЙ СЕМЕЙСТВА SPARTAN-3AN ФИРМЫ XILINX

Учебно-методическое пособие

Рекомендовано методической комиссией радиофизического факультета для студентов ННГУ, обучающихся по направлениям подготовки 03.03.03 «Радиофизика», 02.03.02 «Фундаментальная информатика и информационные технологии»

Нижний Новгород
2018

УДК 681.3 (075.8)
ББК 32.973.2 я73
З-71

Рецензент: кандидат физ.-мат. наук **М.А. Мищенко**

З-71 ЗНАКОМСТВО С ПРОГРАММИРУЕМОЙ ЛОГИЧЕСКОЙ ИНТЕГРАЛЬНОЙ СХЕМОЙ СЕМЕЙСТВА SPARTAN-3AN ФИРМЫ XILINX: учебно-метод. пособие / сост. В.Ю. Семенов, В.В. Артемьев. – Нижний Новгород: Издательство ННГУ, 2018. – 42 с.

Учебно-методическое пособие содержит описание структуры программируемой логической интегральной схемы семейства Spartan-3AN и основ работы в системе автоматизированного проектирования Xilinx ISE WebPACK. Подробно рассматривается процесс создания проекта в этой программной среде на языке VHDL и в схемотехническом редакторе, а также временного моделирования во встроенном симуляторе ISim.

В общем виде даётся описание структуры программы, а также тестирующего модуля к ней. На примере элементарных программ на языке VHDL разобраны случаи формирования в программируемой логической интегральной схеме периодических импульсных сигналов, а также сигналов определенной формы. Даны необходимые пояснения. Приводятся задания для самостоятельного выполнения. Имеется перечень контрольных вопросов.

Методические указания к лабораторной работе разработаны для преподавания на радиофизическом факультете дисциплины «Аппаратные и программные средства цифровой обработки сигналов» по направлению подготовки 03.03.03 «Радиофизика» и 02.03.02 «Фундаментальная информатика и информационные технологии».

Ответственные за выпуск:
председатель методической комиссии радиофизического факультета ННГУ,
к.ф.-м.н., доцент Н.Д. Миловский,
зам. председателя методической комиссии радиофизического факультета ННГУ,
д.ф.-м.н., профессор Е.З. Грибова

УДК 681.3
ББК 32.973.2

1. Введение

Программируемые логические интегральные схемы являются одним из перспективных направлений современной цифровой микроэлектроники. В последнее время это направление имеет высокий темп развития и уже получено существенное улучшение характеристик этих устройств.

Промышленно выпускаемые программируемые микросхемы с электрическим программированием и автоматизированным процессом перевода схемы пользователя делают проектирование новых цифровых устройств сравнимым с разработкой программного обеспечения.

Целью лабораторной работы является знакомство и получение базовых навыков в составлении и отладки программ для программируемых интегральных схем (ПЛИС) фирмы Xilinx семейства Spartan-3AN [1].

2. Программируемая логическая интегральная схема

Проектирование цифровых устройств с применением ПЛИС имеет свои особенности. Для разработки конкретных схем используются специально созданные системы автоматизированного проектирования, в которых для ввода могут использоваться языки описания аппаратуры интегральных схем или универсальные схемные редакторы. Для программирования микросхем применяются программаторы, использующие стандарт IEEE 1149.4 JTAG. Этот стандарт позволяет не только производить загрузку ПЛИС, но и проверять правильность работы микросхемы.

Использование ПЛИС обеспечивает максимальную гибкость при необходимости модификации аппаратуры. Применение ПЛИС позволяет сократить процесс проектирования и отладки цифровых устройств. При этом время, требуемое для получения работающей микросхемы, составляет от нескольких часов до нескольких дней, весь процесс разработки и получения готовой микросхемы производится на одном рабочем месте.

Характерной особенностью ПЛИС является использование для вычислений только целых чисел, включая представление чисел с фиксированной точкой. Никакие вычисления в формате чисел с плавающей точкой напрямую невозможны.

Примером ПЛИС является микросхема XC3S700AN линейки Spartan-3AN фирмы Xilinx [1,2], на базе которого предполагается реализация заданий лабораторной работы.

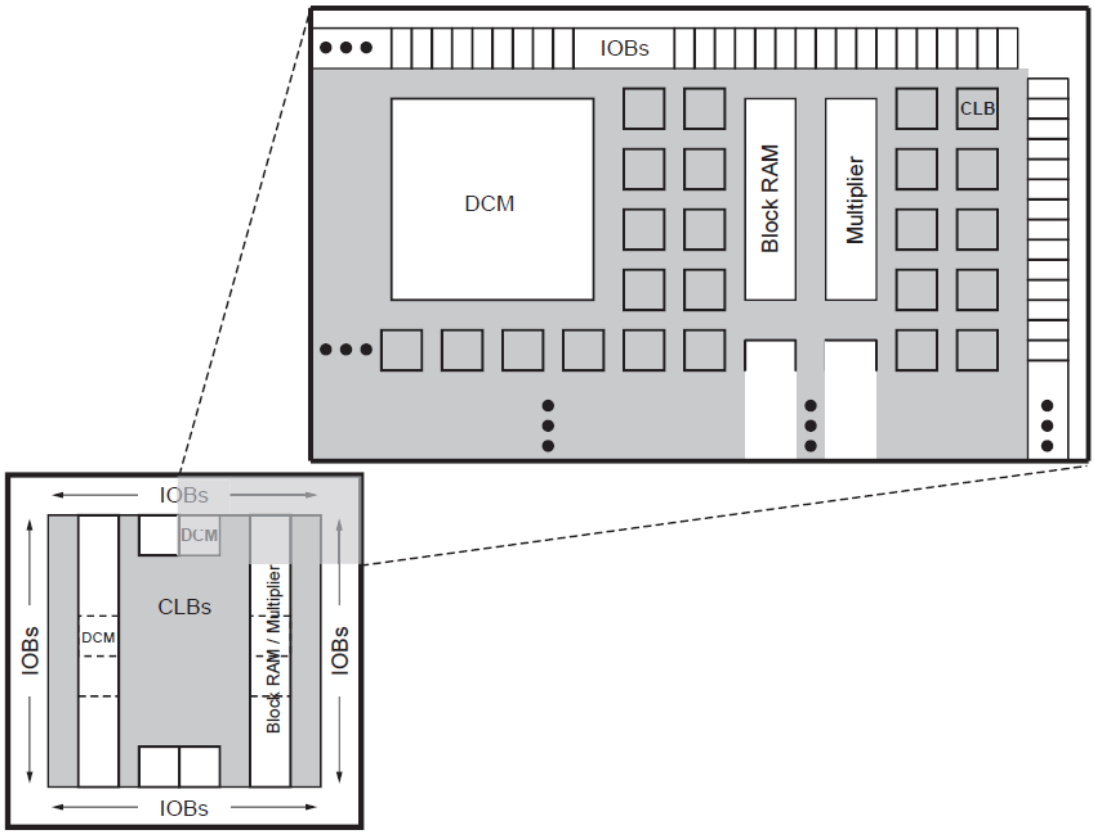

Рис. 1. Структура ПЛИС Spartan-3AN

На рисунке 1 приведена общая структура ПЛИС XC3S700AN. В его состав входят:

• Настраиваемые логические блоки (CLBs), которые содержат в себе гибкие Look-Up table (LUTs), реализующие логику триггера или защелки, используемые в качестве элементов хранения.

• Блоки ввода/вывода (IOBs) посредством которых осуществляется управление потоком данных между контактами микросхемы и внутренней логикой устройства. Блоки ввода/вывода могут обеспечивать двунаправленный поток данных и высокоимпедансное состояния вывода. Они поддерживают различные стандарты сигнала, в том числе несколько высокопроизводительных дифференциальных стандартов. В состав блоков входят регистры с двойной скоростью передачи данных (DDR).

• Блоки оперативной памяти. Каждый блок обеспечивает хранение данных в виде 18-Кбитных блоков с двойными портами.

• Блоки аппаратного умножителя. Каждый аппаратный умножитель производит умножение двух 18-разрядных чисел.

• Блоки цифрового управления тактовым сигналом (DCM). Блоки являются цифровым решением задач распространения, задержки, умножения, деления и фазового сдвига тактовых сигналов и поддерживают самокалибровку.

Подробную информацию об устройстве ПЛИС XC3S700AN можно найти в [1,2,3].

3. САПР Xilinx ISE WebPACK

Программирование ПЛИС осуществляется в САПР Xilinx ISE WebPACK 14 [4]. В состав САПР входят менеджер проектов, редактор кода с подсветкой синтаксиса, схемотехнический редактор, синтезатор HDL описания XST, компоновщик, трассировщик, симулятор ISIM, программатор iMPACT, инструментальные средства отладки. Внешний вид менеджера проектов САПР Xilinx ISE WebPACK 14 показан на рисунке 2. Синтезатор XST осуществляет создание из HDL описания принципиальной схемы на простых элементах. Компоновщик и трассировщик осуществляют размещение элементов в базисе ПЛИС и трассировку сигналов между элементами. Загрузка внутренней конфигурационной структуры ПЛИС осуществляется с помощью iMPACT.

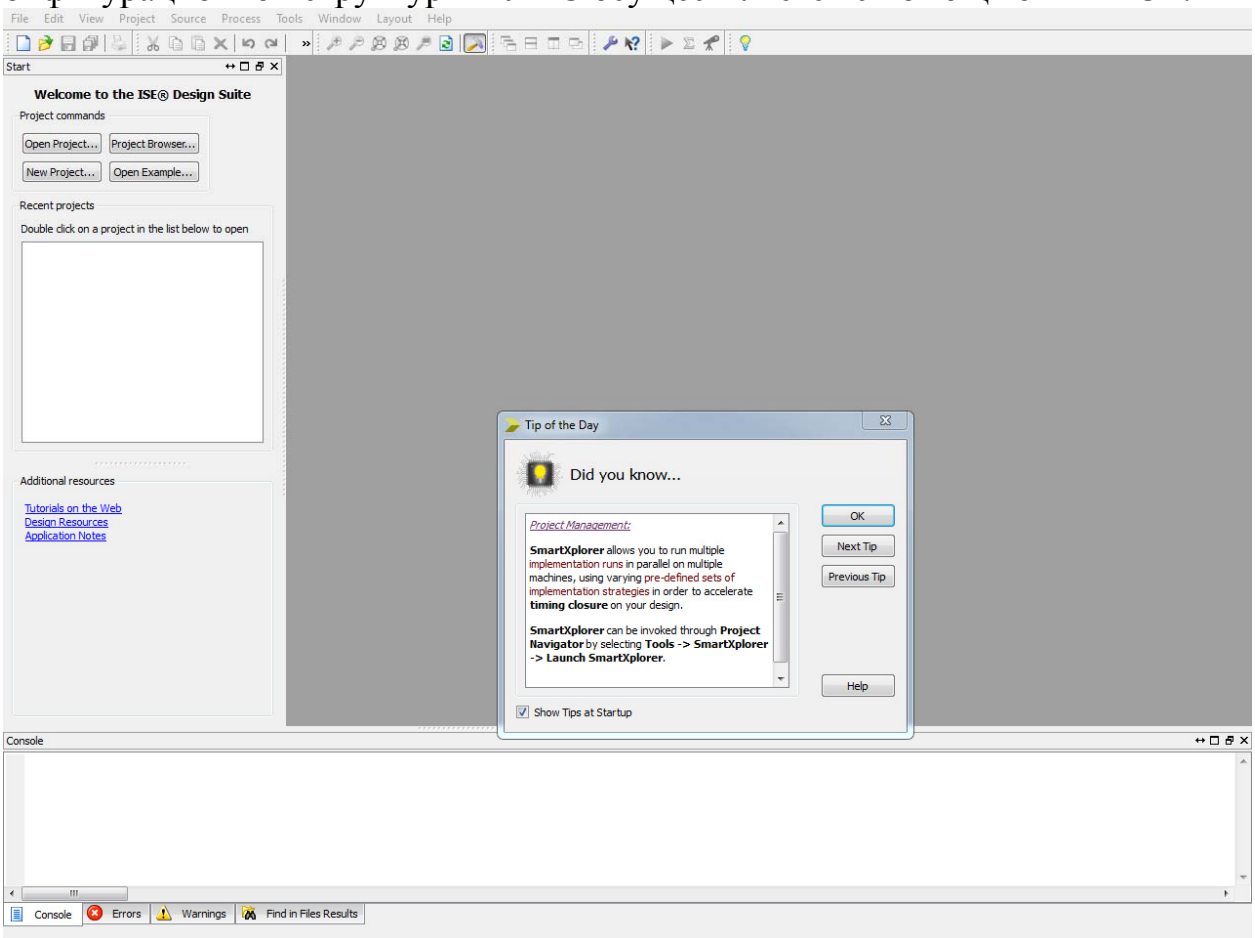

Рис. 2. САПР Xilinx ISE WebPACK 14

3.1. Начало работы

Запуск САПР Xilinx ISE WebPACK 14 может быть осуществлен или с помощью иконки расположенной на рабочем столе:

или через главное меню операционной системы: «**Пуск**» → «**Программы**» → «**Xilinx Design Tools**» → «**ISE Design Suite**» → «**ISE Design Tools**» → «**Project Navigator**».

В результате будет запущена основная программа САПР – это менеджер проектов. Менеджер проектов позволяет создавать новые проекты, осуществлять доступ к файлам проекта, разделять файлы по типу на файлы с исходным описанием проектируемого устройства, тестовыми файлами, файлами содержащими временные и топологические ограничения проекта. Менеджер предоставляет доступ ко всем процессам, необходимым при проектировании цифрового устройства на базе ПЛИС.

Разработку цифрового устройства на базе ПЛИС в общем случаем можно разделить на этапы:

1) Создание нового проекта с выбором семейства и микросхемы ПЛИС;
2) Создание описания цифрового устройства в схемотехническом или текстовом редакторе;
3) Синтез устройства;
4) Функциональное моделирование и тестирование;
5) Компоновка, размещение и трассировка проекта в кристалле;
6) Временное моделирование;
7) Генерирование и загрузка проекта в ПЛИС.

Каждому их этих этапов соответствует свой набор процессов, к которым можно получить доступ из менеджера проектов.

На рисунке 3 проиллюстрированы основные компоненты менеджера проектов.

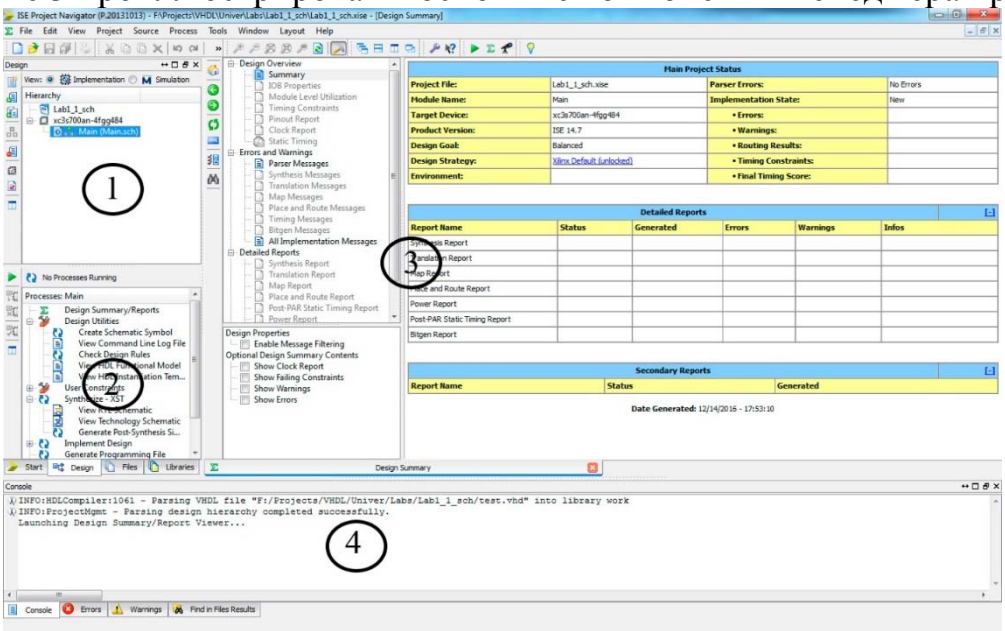

Рис. 3. Менеджер проектов САПР Xilinx ISE WebPACK 14:
1 – окно описания проектов («Source window»); 2 – окно процессов («Process window»); 3 – рабочее пространство («Workspace»); 4 – окно отчетов («Transcript window»)

3.2. Создание нового проекта

Для создания нового проекта необходимо выбрать в меню пункт «**File**» → «**New Project…**». Запустится мастер создания нового проекта, внешний вид которого показан на рисунке 4.

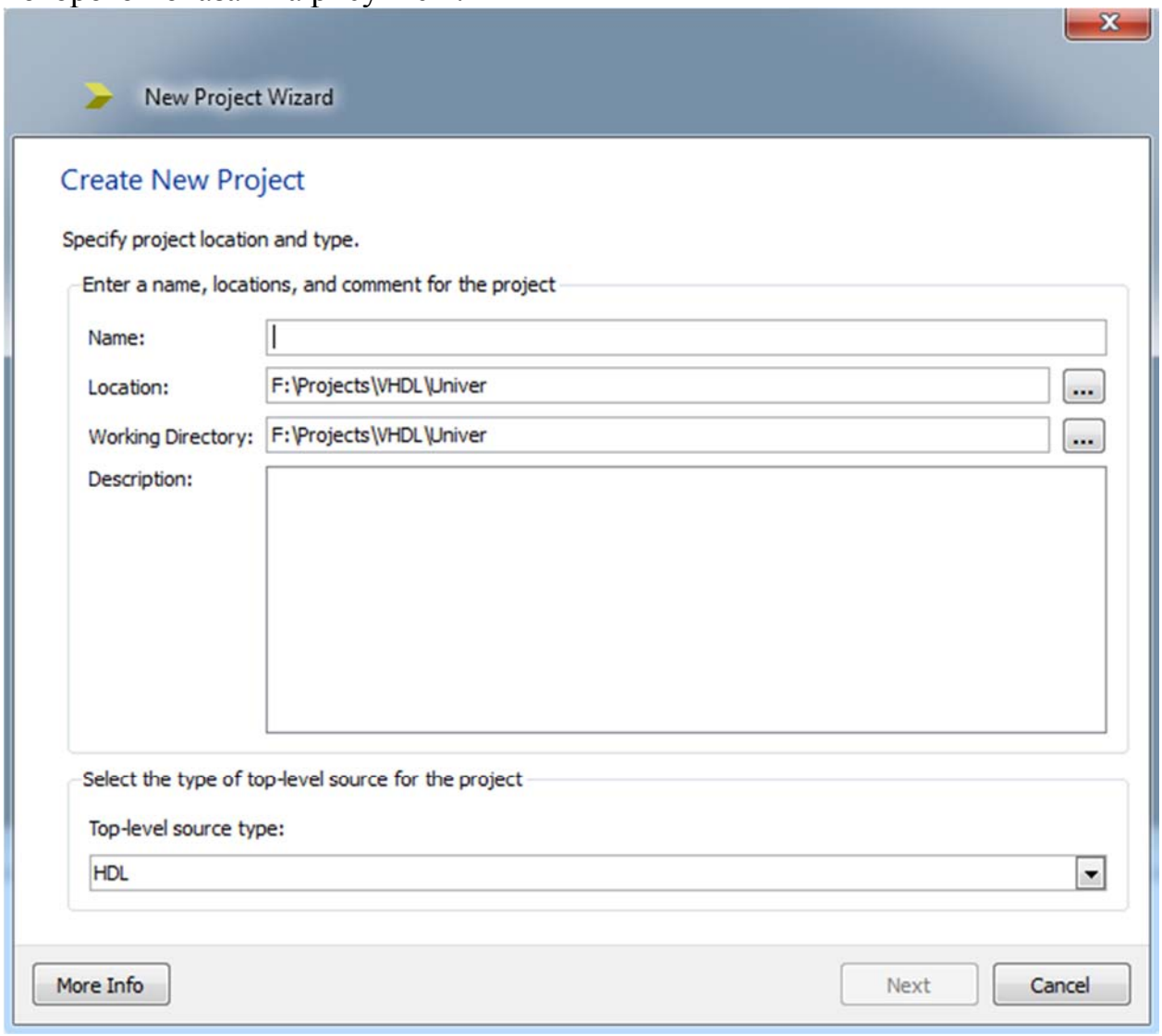

Рис. 4. Окно мастера создания новых проектов (Шаг 1)

В поле «**Location**» задается местоположение создаваемого проекта. В поле «**Name**» вводится имя нового проекта. С помощью поля «**Working Directory**» можно указать рабочую директорию проекта. По умолчанию она равна полю «**Location**». Важно заметить, что при вводе названия проекта, в поле «**Location**» и «**Working Directory**» автоматически добавляется каталог с названием проекта, который будет создан по завершению работы мастера. Тип основного файла описания устройства можно выбрать в выпадающем списке «**Top-level source type**». Для работы с текстовыми файлами описания устройства рекомендуется выбирать «**HDL**», а для схемотехнического редактора «**Schematic**». После указания всей требуемой информации необходимо нажать на кнопку «**Next**».

На следующем шаге работы мастера создания нового проекта предлагается заполнить основные свойства проекта. Внешний вид окна показан на рисунке 5.

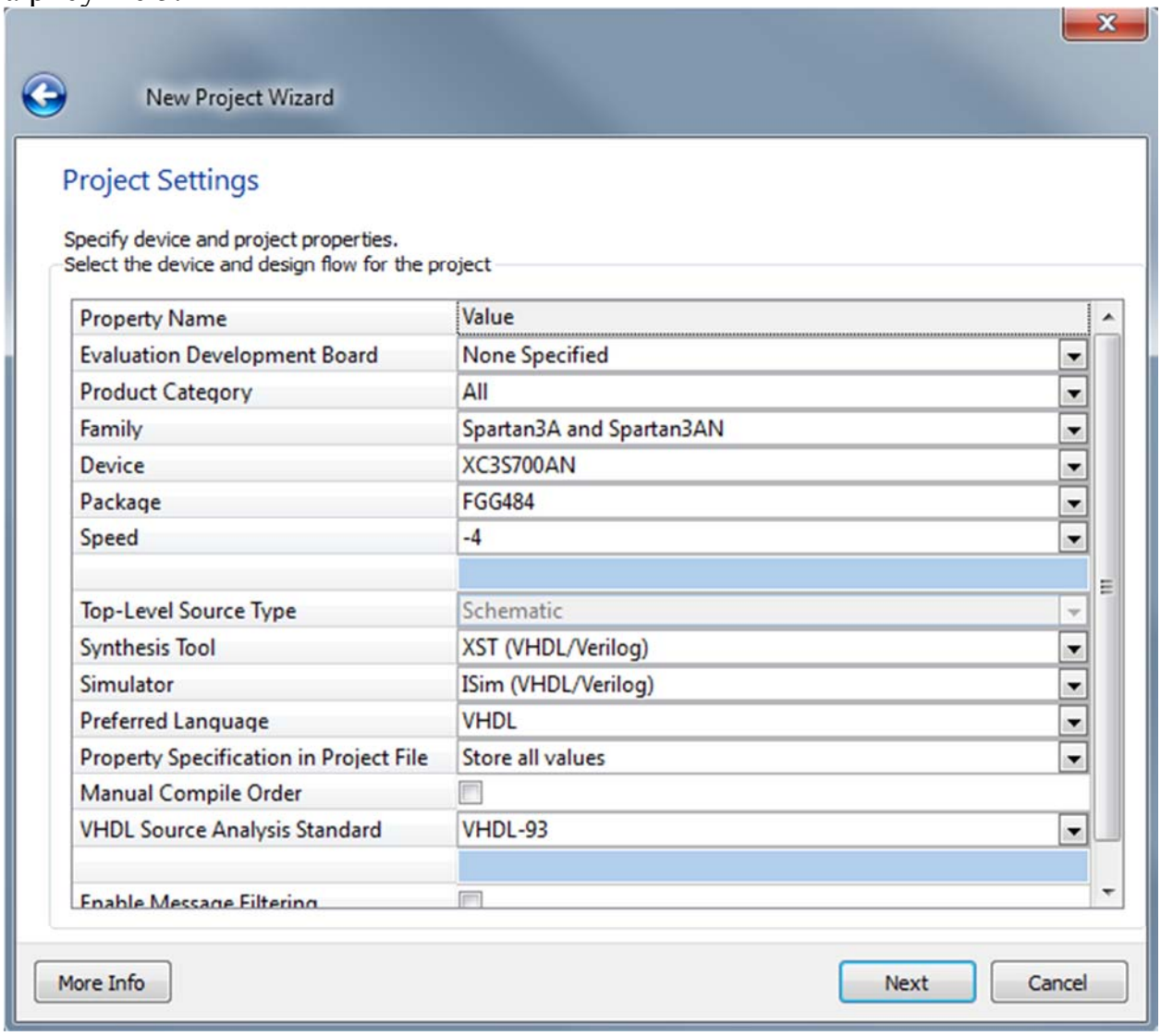

Рис. 5. Окно мастера создания новых проектов (Шаг 2)

Основными свойствами проекта являются:

- «**Evaluation Development Board**» - используемая оценочная плата (при наличии);
- «**Product Category**» - категория применения ПЛИС;
- «**Family**» - семейство ПЛИС;
- «**Device**» - ПЛИС;
- «**Package**» - тип корпуса ПЛИС;
- «**Speed**» - характеристика быстродействия ПЛИС;
- «**Top-Level Source Type**» - тип основного файла описания устройства, который выбирался на предыдущем шаге;
- «**Synthesis Tool**» - программное обеспечение используемое при синтезе схемы;
- «**Simulator**» - симулятор работы ПЛИС;

- «**Preferred Language**» - предпочитаемый язык описания аппаратуры.

Более подробно об остальных свойства проекта можно почитать, нажав на кнопку «**More Info**».

Для работы с лабораторной установкой необходимо выбрать «**Spartan-3AN Starter Kit**» в поле «**Evaluation Development Board**». При этом автоматически заполняются поля:

- «**Product Category**» - **All**;
- «**Family**» - **Spartan3A and Spartan3AN**;
- «**Device**» - **XC3S700AN**;
- «**Package**» - **FGG484**;
- «**Speed**» - **-4.**

Так же необходимо выбрать:

- «**Synthesis Tool**» - **XST (VHDL/Verilog)** ;
- «**Simulator**» - **ISim (VHDL/Verilog)** ;
- «**Preferred Language**» - **VHDL.**

После указания всей требуемой информации необходимо нажать на кнопку «**Next**».

На следующем этапе работы мастера подводится итог работы. По нажатию кнопки «**Finish**» мастер создает проект с указанными свойствами и завершает свою работу.

3.3. Создание HDL описания устройства

Для создания файла VHDL [5,6] описания устройства необходимо выполнить следующие действия:

1) Запустить мастер создания файлов проекта путем выбора «**New Source…**» в меню пункт «**Project**» или в контекстном меню, появляющимся по правому щелчку мыши в области 1 - Окна описания проектов на рисунке 3. Внешний вид мастера показан на рисунке 6;

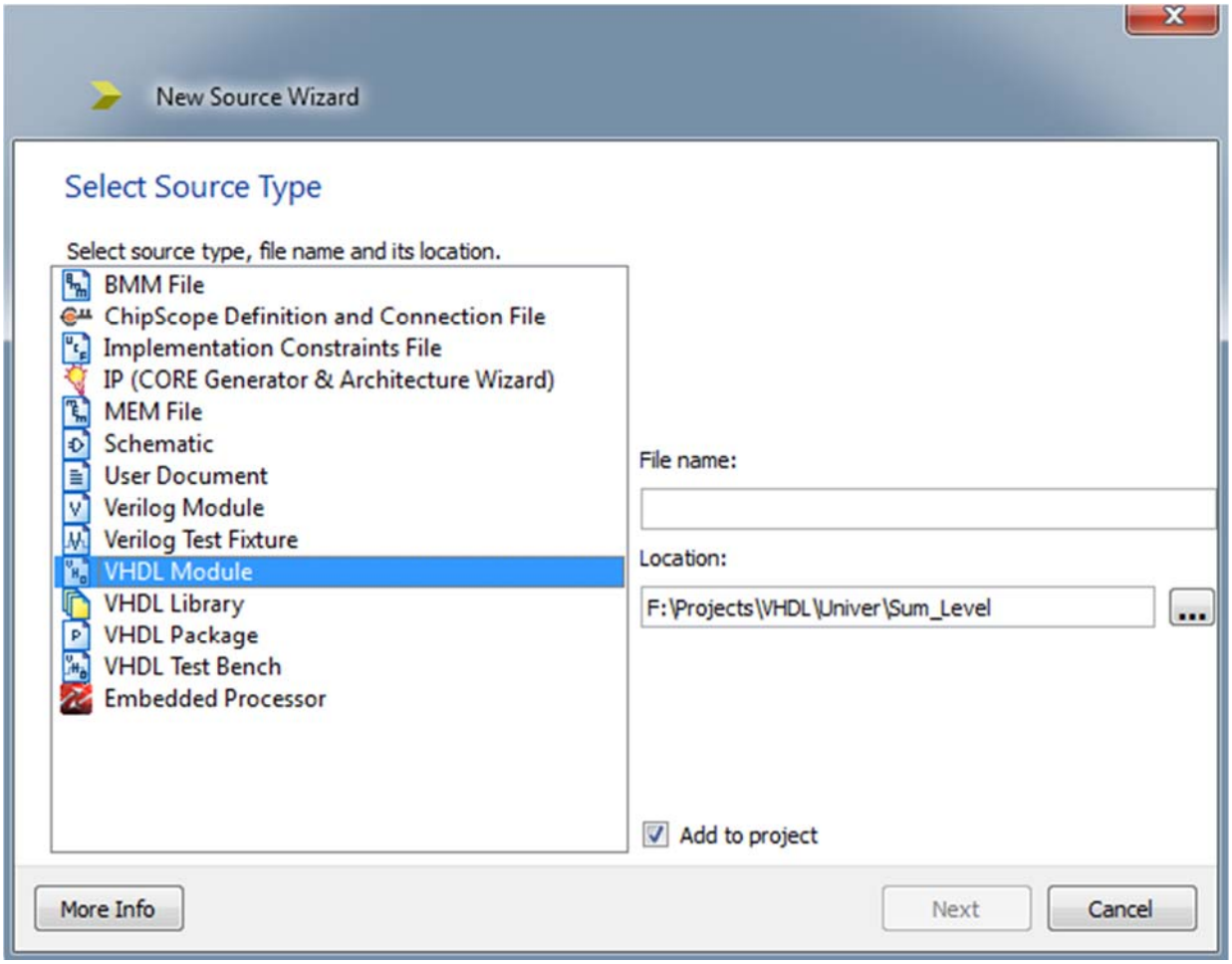

Рис. 6. Окно мастера создания новых файлов проекта (Шаг 1)

2) В качестве типа нового файла выбрать «**VHDL Module**»;
3) Ввести имя файла в поле «**File name**»;
4) Убедиться что опция «**Add to project**» включена;
5) Нажать на кнопку «**Next**»;
6) На втором шаге работы мастера создания файлов проекта необходимо установить свойства портов ввода-вывода для HDL файла описания проектируемого устройства. Причем если файл является верхним по иерархии проекта, то названия портов ввода-вывода будут привязываться и к выводам микросхемы. В поле «**Port Name**» вводится имя порта. Поле «**Direction**» задает направление порта, где «**in**» соответствует входному порту, «**out**» - выходному, а «**inout**» двунаправленному порту. Опция «**Bus**» определяет является ли порт одиночным или это группу нескольких портов объединённых в одну шину. Установка этой опции разблокирует поля «**MSB**» и «**LSB**», которые задают для шины максимальный и минимальный индекс соответственно.

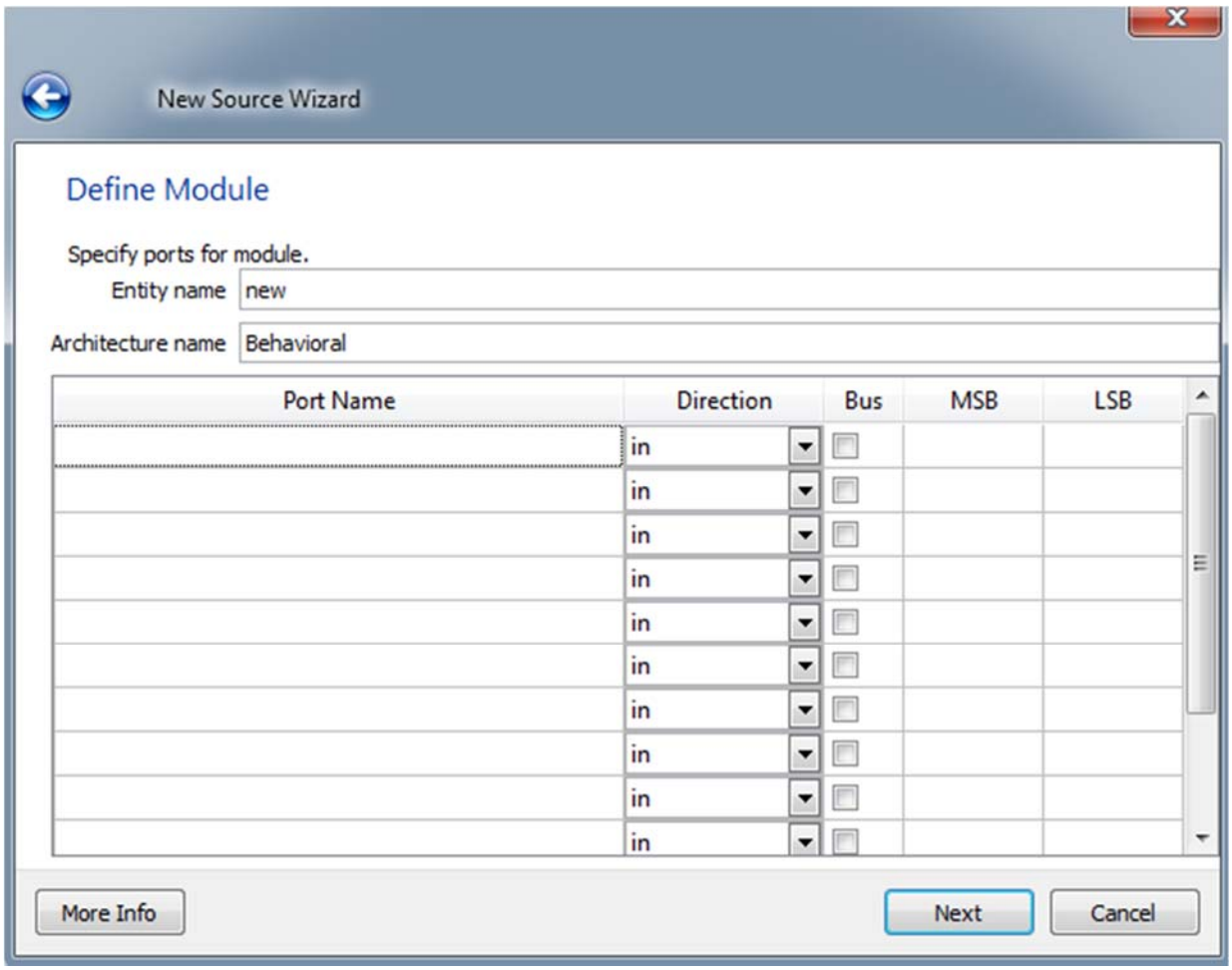

Рис. 7. Окно мастера создания новых VHDL файлов проекта (Шаг 2)

7) По завершению настройки портов необходимо нажать на кнопку «**Next**».

8) На завершающем шаге работы мастера приводится краткий отчет по работе. По нажатию кнопки «**Finish**» мастер создает файл HDL описания с указанными свойствами и завершает свою работу.

Если требуется создать файл Verilog описания устройства, то необходимо на действии 2) выбрать «**Verilog Module**».

3.4. Работа в текстовом редакторе кода

На рисунке 8 показан внешний вид текстового редактора кода.

VHDL описание разбивается на две части:

1) Entity (сущность) – определяет имя проекта/модуля, порты ввода-вывода и их настройку.

2) Architecture (архитектура) – описывает структуру и/или функционирование устройства.

В начале VHDL описания перед Entity идет подключение стандартных библиотек. Как правило это:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

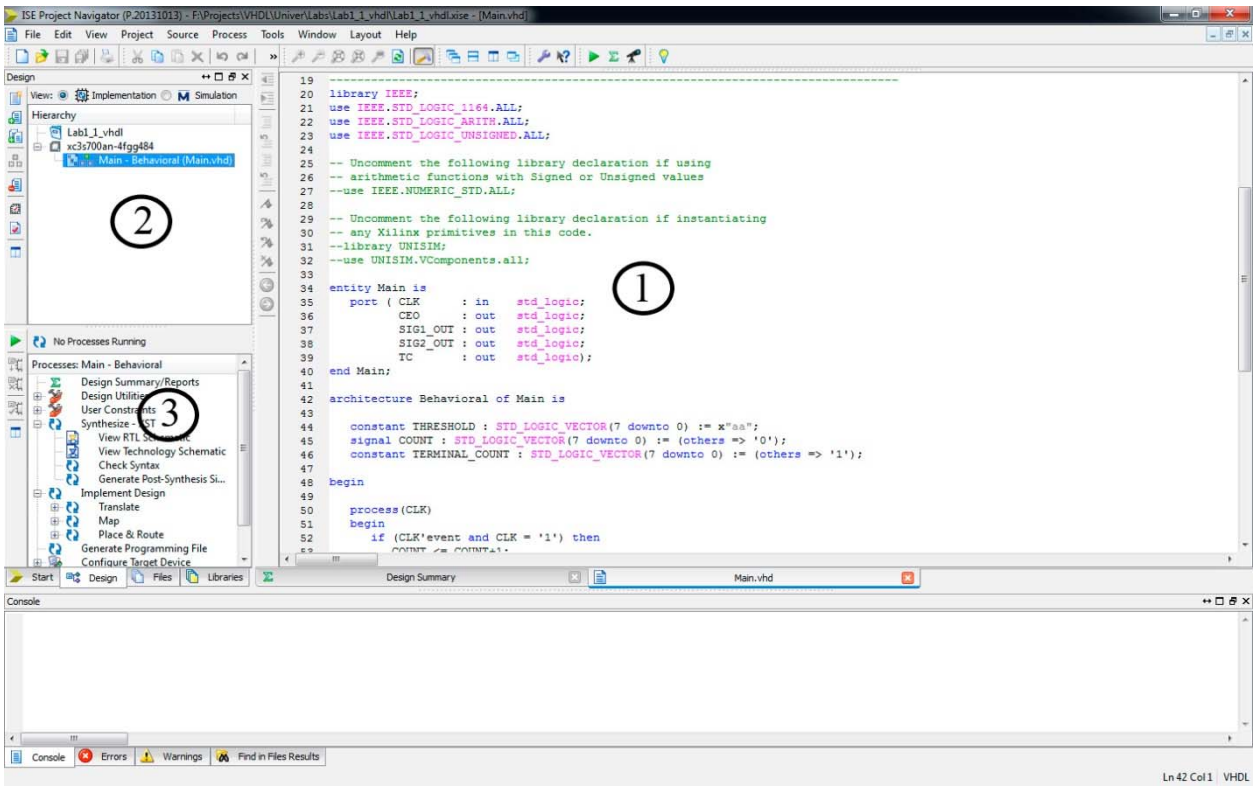

Рис. 8. Внешний вид текстового редактора кода:
1 – рабочее пространство редактора («Workspace»); 2 – окно описания проектов («Source window»); 3 – окно процессов («Process window»)

В САПР Xilinx ISE WebPACK 14 имеется библиотека шаблонов. Читателю рекомендуется самостоятельно ознакомиться с возможностью её применения.

По завершению редактирования HDL описания устройства необходимо проверить файл на наличие синтаксических ошибок. Это можно сделать с помощью «**Check Syntax**», находящимся в окне процессов в разделе «**Synthesize**», как показано на рисунке 9.

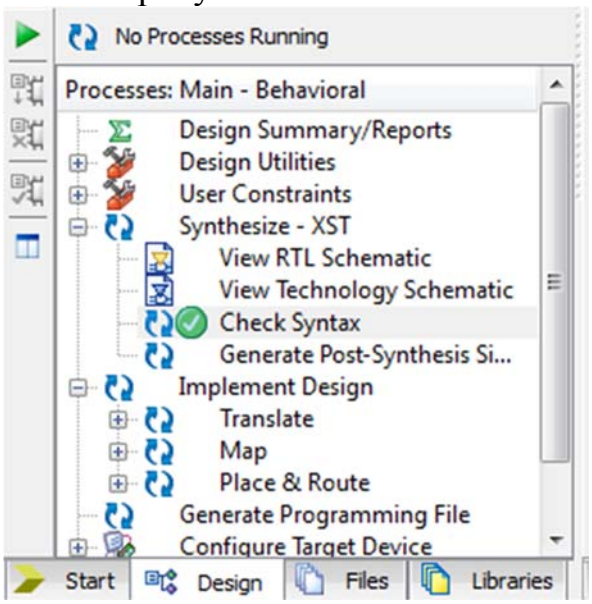

Рис. 9. Проверка синтаксиса

Сообщения обо всех найденных ошибках отображаются в консоли - область 4 на рисунке 3. При отсутствии ошибок в консоли будет выведено сообщение «Process "Check Syntax" completed successfully» об успешной проверке синтаксиса.

3.5. Создание схемотехнического описания

Для создания файла схемотехнического описания устройства необходимо выполнить следующие действия:

1) Запустить мастер создания файлов проекта путем выбора «**New Source…**» в меню пункт «**Project**» или в контекстном меню, выпадающим по правому щелчку мыши в области 1 - Окна описания проектов на рисунке 3;
2) В качестве типа нового файла выбрать «**Schematic**»;
3) Ввести имя файла в поле «**File name**»;
4) Убедиться что опция «**Add to project**» включена;
5) Нажать на кнопку «**Next**» ;
6) На следующем шаге приводится краткий отчет по работе мастера создания описаний, и он является завершающим, поскольку создание схемотехнических файлов не требует каких-либо ещё дополнительных настроек. По нажатию кнопки «**Finish**» мастер создает файл схемотехнического описания и завершает свою работу. В проекте допускается использование файлов различного типа описания устройства

3.6. Работа в схемотехническом редакторе

На рисунке 10 показан внешний вид схемотехнического редактора.

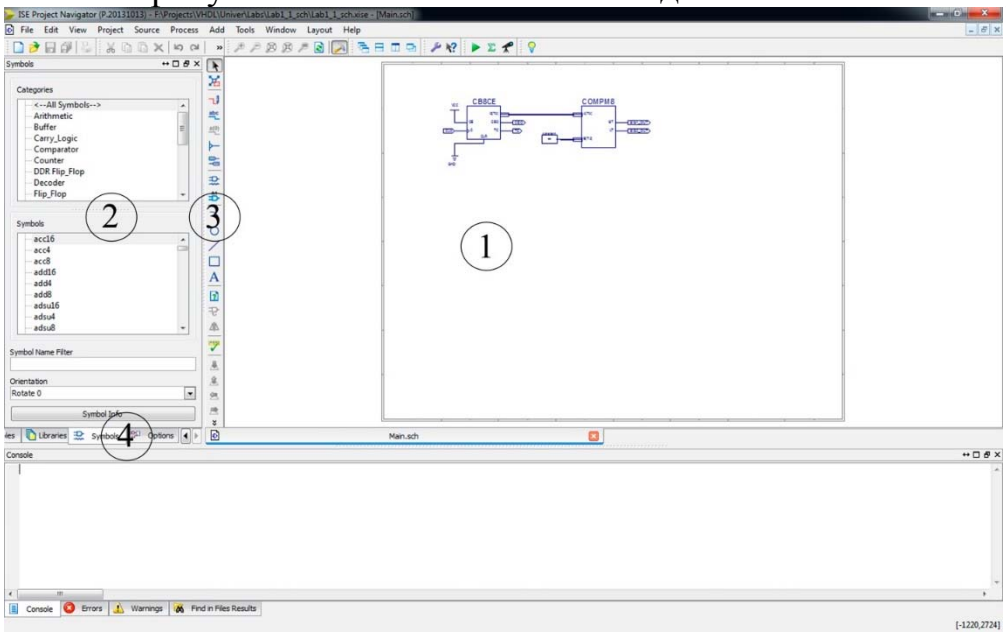

Рис. 10. Внешний вид схемотехнического редактора:
1 – рабочее пространство схемотехнического редактора («Workspace»); 2 – окно символов («Symbols window»); 3 – панель инструментов («Toolbar»); 4 – вкладки окон

Для добавления нового символа на схему необходимо выделить соответствующее название символа в списке «**Symbols**» и далее щелкнуть левой кнопкой мыши по свободному пространству схемы. Символ будет добавлен на схему. Все символы сгруппированы по категориям «**Categories**». Для добавления цепи, связующей символы, необходимо на панели инструментов область 3 рисунка 10 нажать «**Add Wire**». Далее щелкнуть левой кнопкой на начальную точку, а затем на конечную. Будет добавлена связь между точками. Для добавления порта ввода/вывода необходимо на панели инструментов область 3 рисунка 10 нажать «**Add I/O Marker**» и затем разместить на схеме. Для смены названия порта необходимо открыть свойство порта путем двойного щелчка по символу порта или выбора в контекстном меню «**Object Properties**» по правому щелчку мышки на символе. Внешний вид окна свойств показан на рисунке 11.

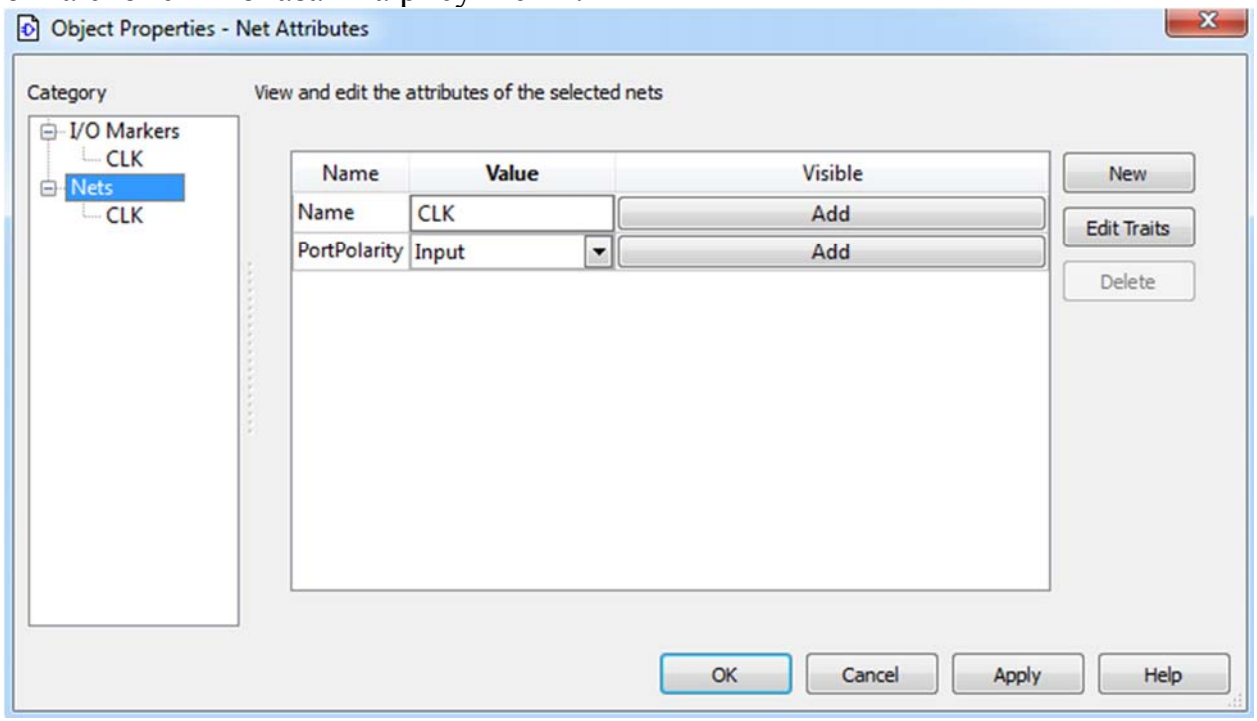

Рис. 11. Окно свойств

В окне свойств необходимо перейти в категорию Nets и затем в поле «**Name**» ввести новое имя порта.

3.7. Добавление IP Core

При разработки проекта на практике зачастую требуется решать задачи, которые уже были решены и реализованы на ПЛИС, например, быстрое преобразование Фурье. Для того чтобы снизить затраты времени на повторную разработку решения известной задачи применяются IP (Intellectual Property) ядра.

Для добавления IP ядра в проект описания устройства необходимо выполнить следующие действия:

1) Запустить мастер создания файлов проекта путем выбора «**New Source…**» в меню пункт «**Project**» или в контекстном меню, появляющимся по правому щелчку мыши в области 1 - Окна описания проектов на рисунке 3. Внешний вид мастера показан на рисунке 6;

2) В качестве типа нового файла выбрать «**IP (CORE Generator & Architecture Wizard**»;

3) Ввести имя файла в поле «**File name**»;

4) Убедиться, что опция «**Add to project**» включена;

5) Нажать на кнопку «**Next**»;

6) На втором шаге работы мастера создания файлов проекта необходимо выбрать IP ядро, которое планируется использовать. Внешний вид окна мастера показан на рисунке 12. Сортировка IP ядер по функции и по названию может осуществляться путем переключения вкладок;

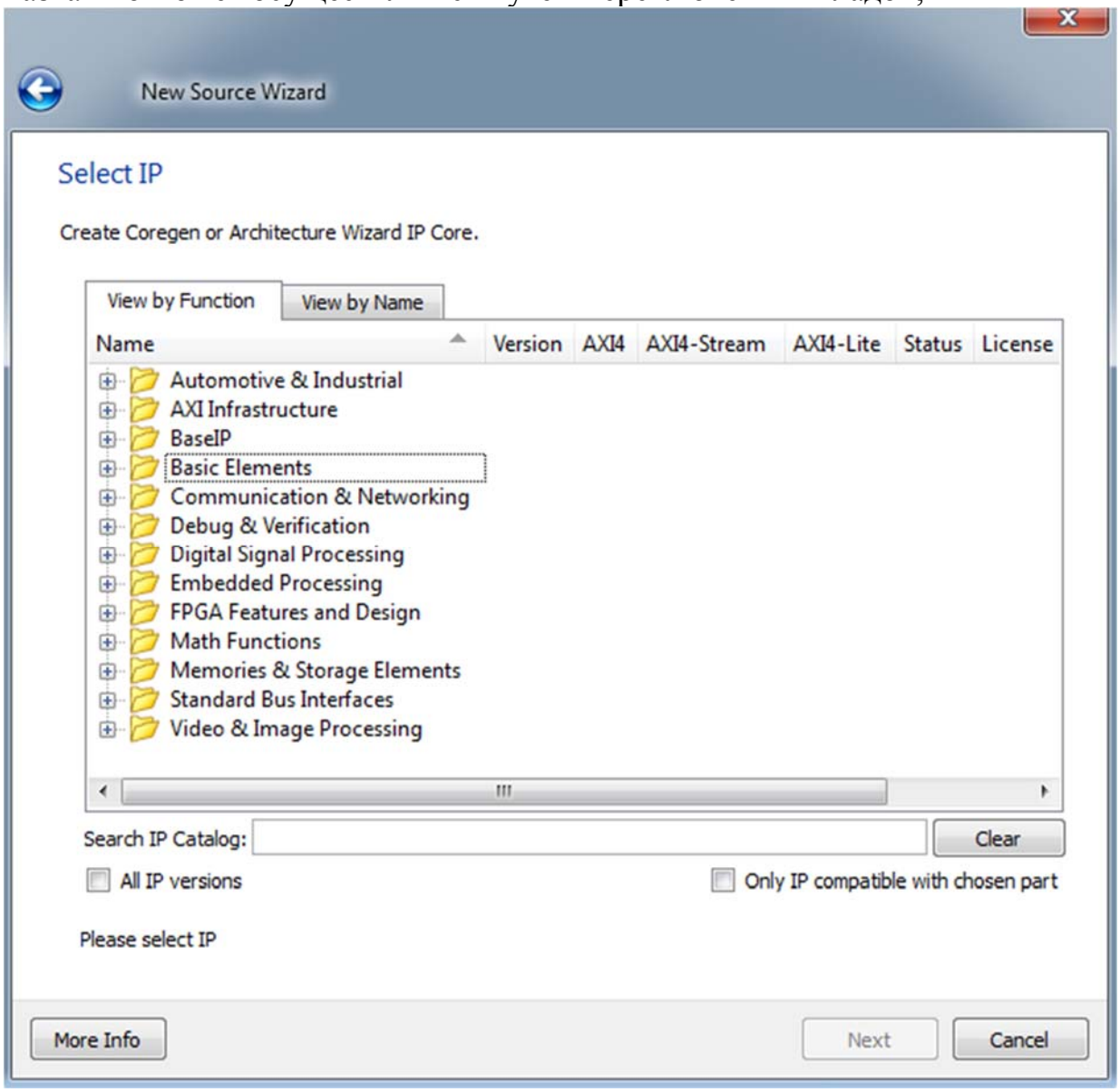

Рис. 12. Выбор IP ядра (Шаг 2)

7) После выбора IP ядра необходимо нажать на кнопку «**Next**»;

8) На завершающем шаге работы мастера приводится краткий отчет по работе. По нажатию кнопки «**Finish**» мастер запускает процедуру по добавлению выбранного IP ядра в проект и завершает свою работу.

3.8. Добавление IP ядра ПЗУ памяти

Рассмотрим работу мастера добавления IP ядра на примере добавления ПЗУ памяти. В качестве мастера используется «**Block Memory Generator**». Внешний вид показан на рисунке 13.

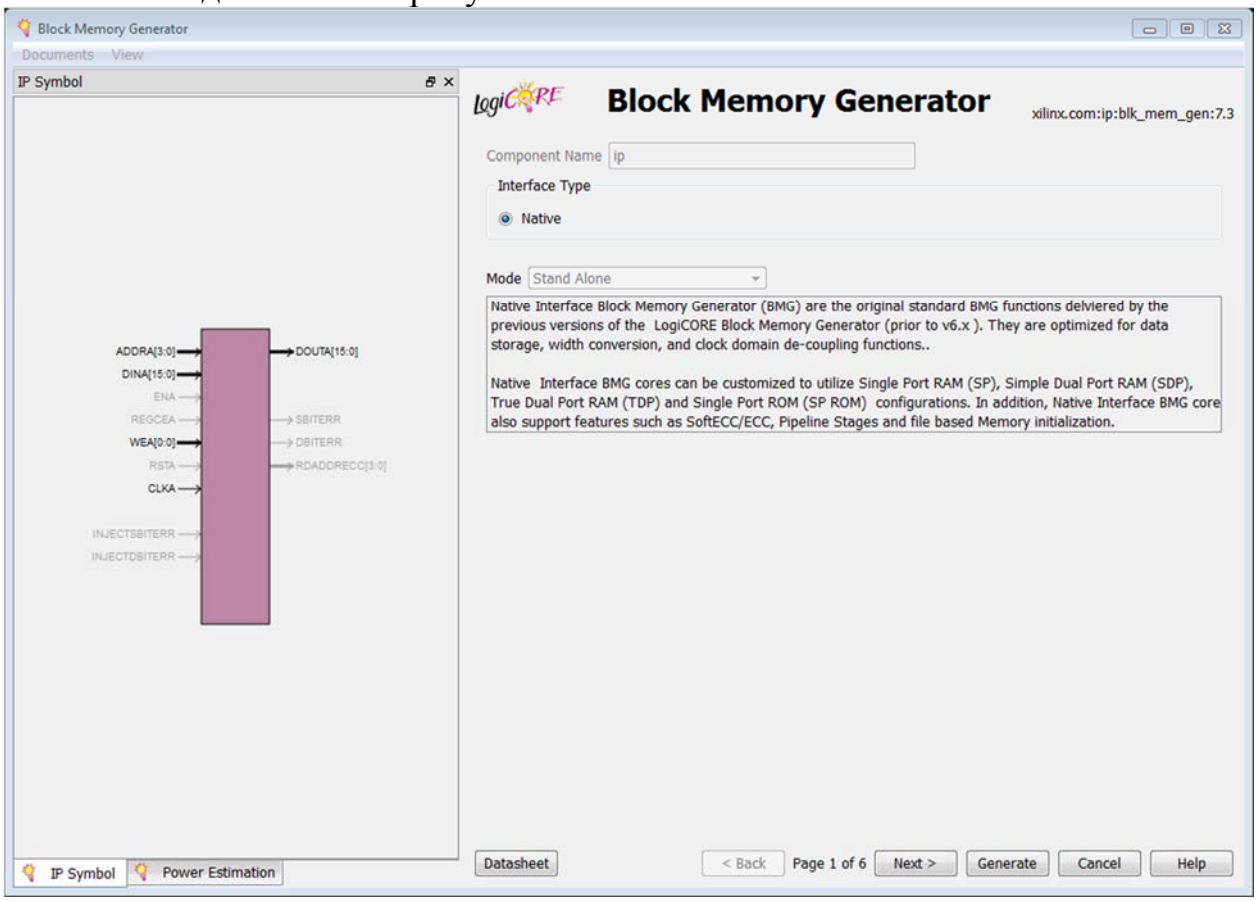

Рис. 13. «**Block Memory Generator**» (Шаг 1)

На первом шаге приводится общая информация о «**Block Memory Generator**». Для перехода к следующему шагу необходимо нажать на кнопку «**Next >**».

На втором шаге работы мастера предлагается указать тип используемой памяти из раскрывающегося списка «**Memory Type**». Внешний вид окна мастера показан на рисунке 14. Для ПЗУ необходимо указать «**Single Port ROM**». В зависимости от указанного типа памяти часть опций становится доступной, а другая часть нет. На втором шаге также можно выбрать алгоритм применения памяти. После установки всех данных необходимо перейти к следующему шагу путем нажатия на кнопку «**Next >**».

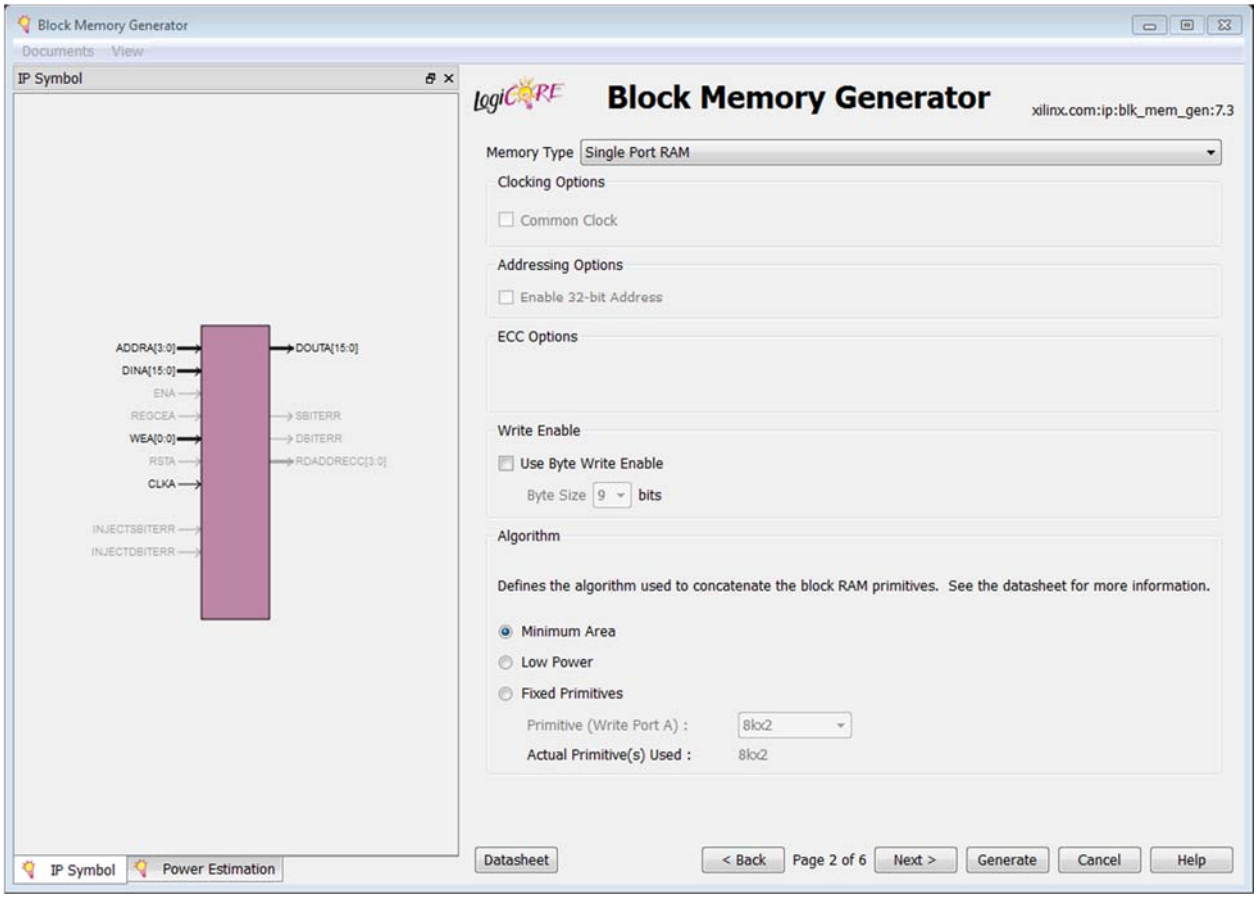

Рис. 14. «**Block Memory Generator**» (Шаг 2)

Внешний вид окна мастера для шага 3 показан на рисунке 15. На третьем шаге предлагается указать размер памяти (Depth) и ширину шины данных (Width). Так же можно установить опцию добавляющую сигнал управления для памяти. После указания всех данных необходимо перейти к следующему шагу путем нажатия на кнопку «**Next >**».

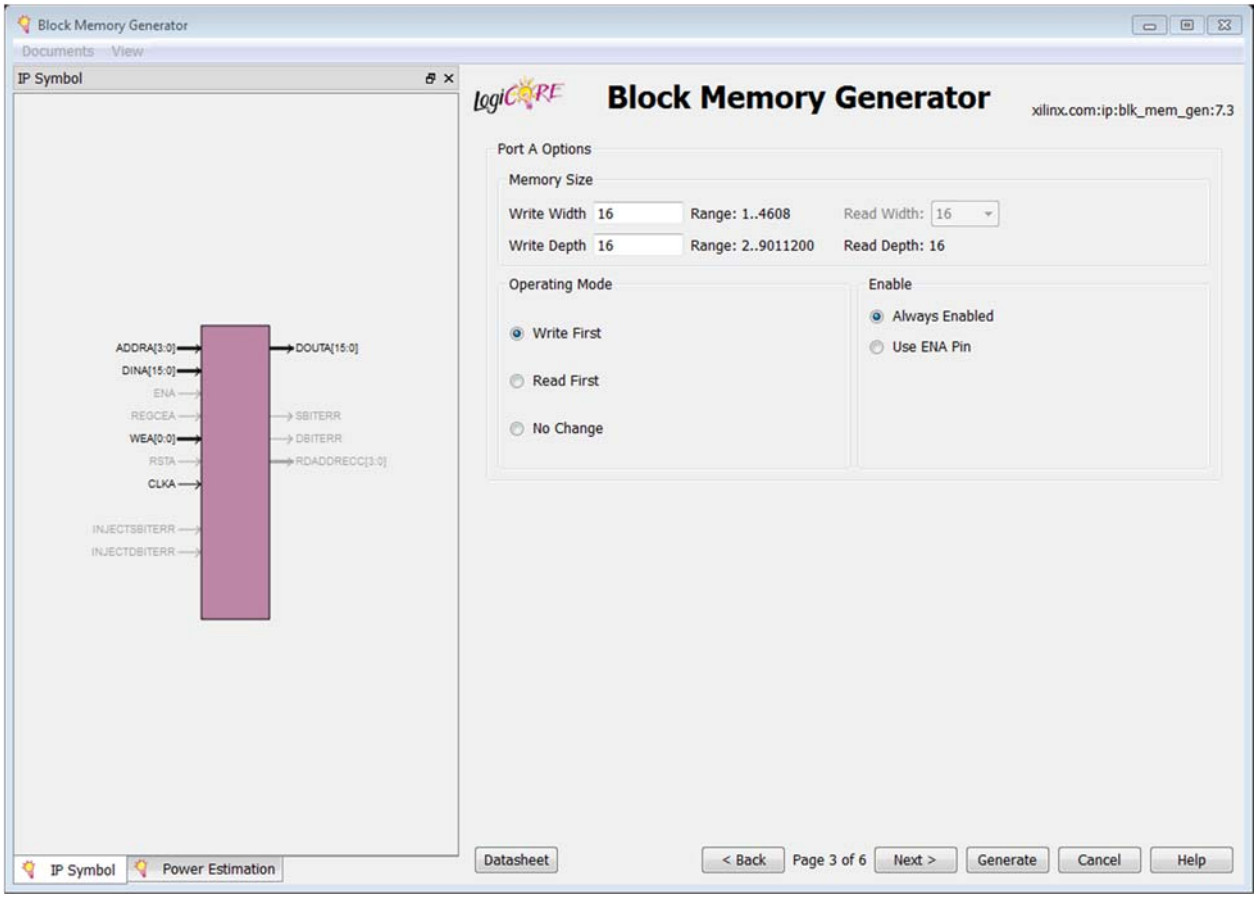

Рис. 15. «**Block Memory Generator**» (Шаг 3)

Внешний вид окна мастера шага 4 показан на рисунке 16. На четвертом шаге работы мастера определяются параметры работы выходного порта памяти, а также задаются значения, хранимые в памяти ПЗУ, с помощью включения опции «**Load Init File**» в разделе «**Memory Initialization**». После включения этой опции необходимо выбрать файл, в котором хранятся значения памяти, через кнопку «**Browse**». После указания всей необходимой информации нужно перейти к следующему шагу путем нажатия на кнопку «**Next >**».

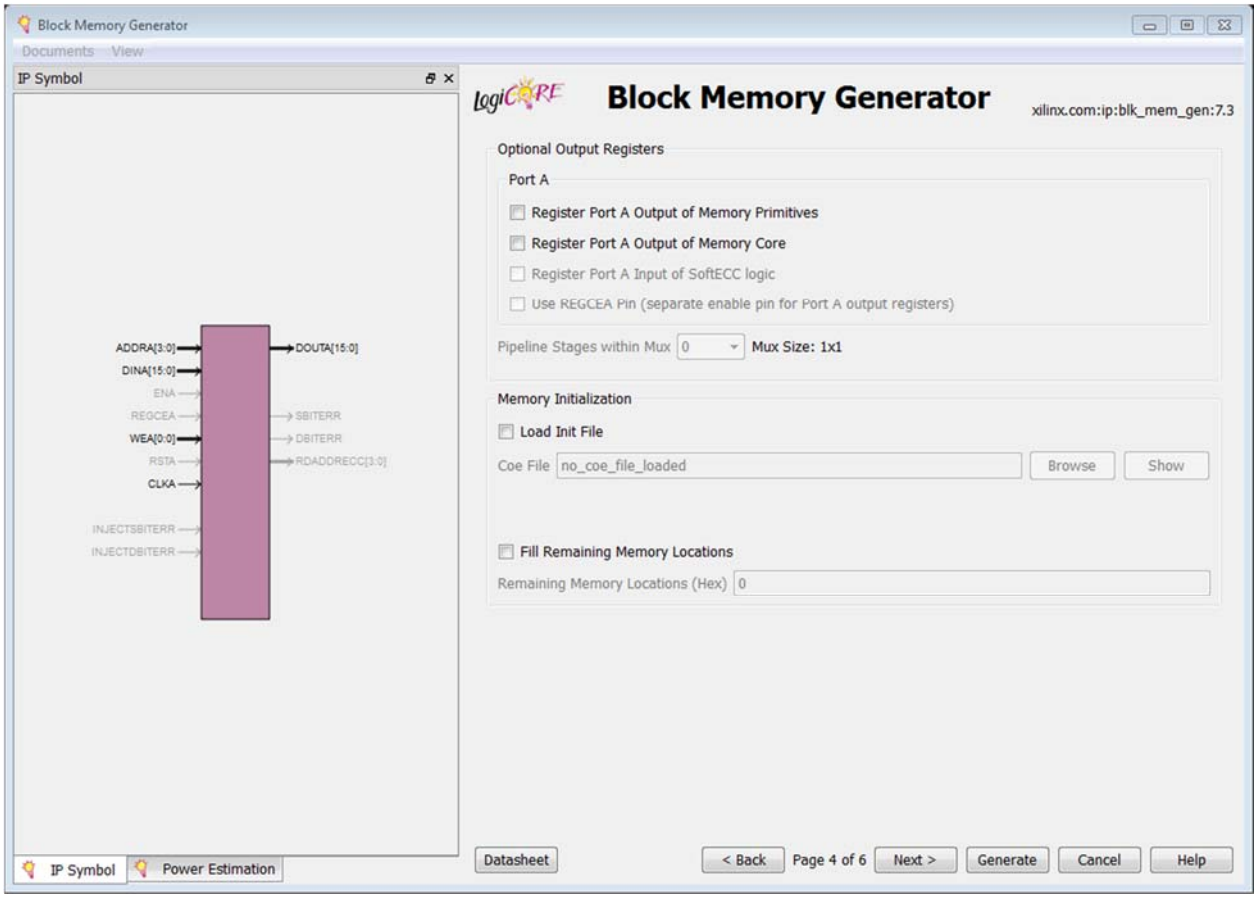

Рис. 16. «**Block Memory Generator**» (Шаг 4)

На пятом шаге работы мастера можно настроить параметры сброса выходного порта. После настройки необходимо перейти к следующему шагу путем нажатия на кнопку «**Next >**».

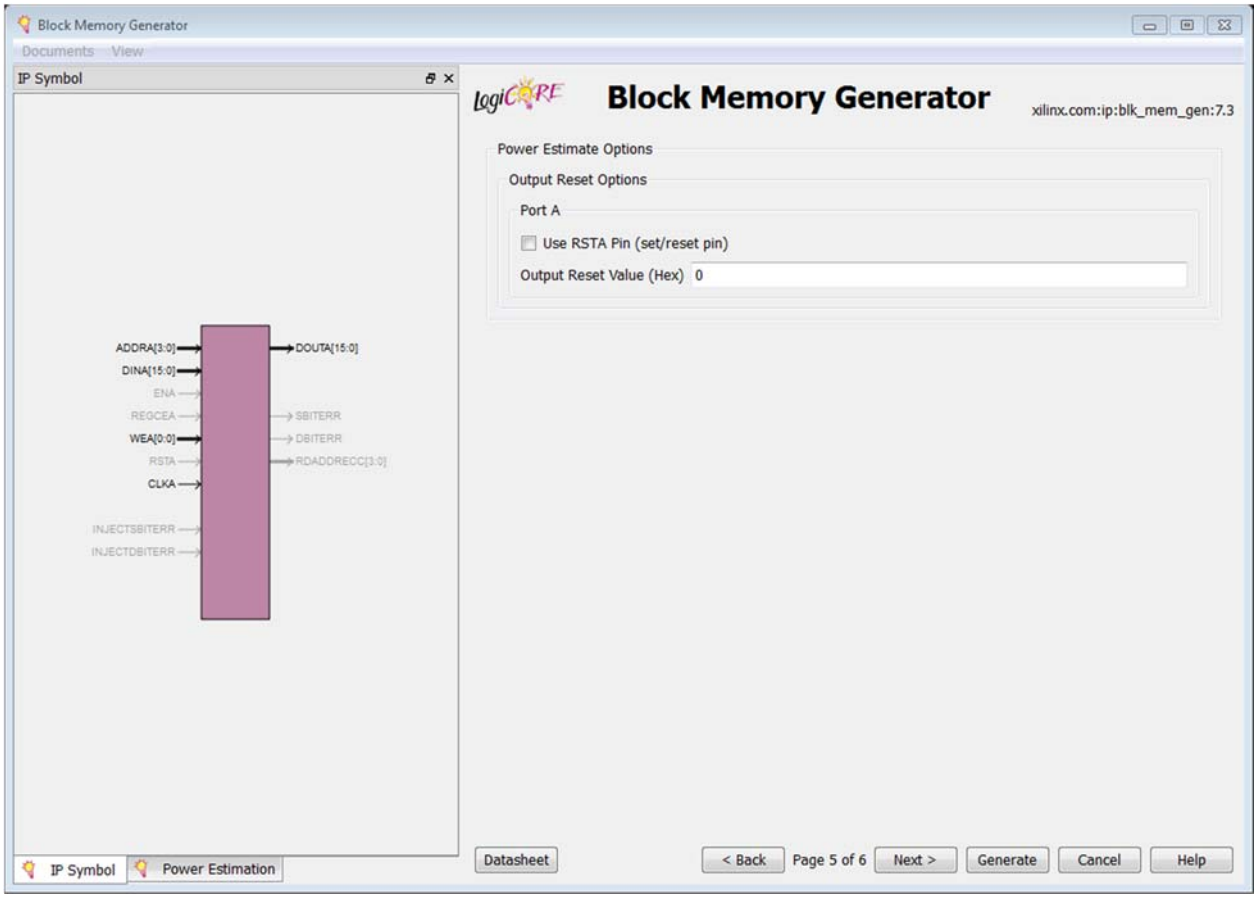

Рис. 17. «**Block Memory Generator**» (Шаг 5)

Шестой шаг является завершающим, и на нем приводится краткая информация по создаваемой памяти. Внешний вид окна мастера показан на рисунке 18. После проверки всей информации необходимо нажать на кнопку «**Generate**».

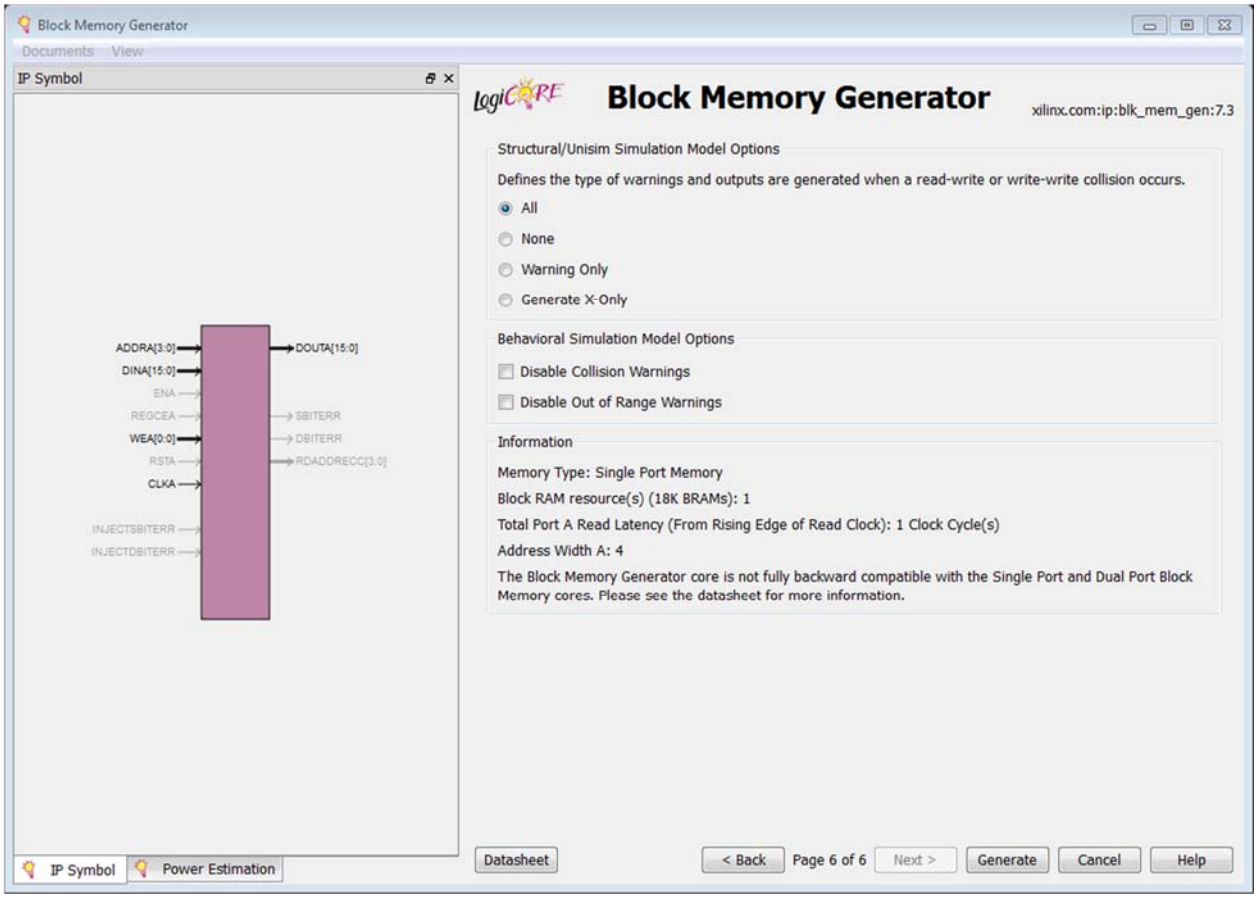

Рис. 18. «**Block Memory Generator**» (Шаг 6)

По завершению работы мастера IP ядро будет добавлено в проект.

3.9. Создание тестового модуля

Для проверки работоспособности описаний разрабатываемых устройств в САПР Xilinx ISE WebPACK 14 заложена возможность проведения моделирования работы с использованием встроенного ISIM или внешнего симулятора. Моделирование работы может производиться на различных этапах проектирования, начиная от функциональной проверки описания и заканчивая проверкой с учетом всех задержек внутри ПЛИС и внешних воздействий.

Для добавления файла VHDL тестовых воздействий необходимо выполнить следующие действия:

1) Запустить мастер создания файлов проекта путем выбора «**New Source…**» в меню пункт «**Project**» или в контекстном меню, выпадающим по правому щелчку мыши в области 1 - Окна описания проектов на рисунке 3. Внешний вид мастера показан на рисунке 6.
2) В качестве типа нового файла выбрать «**VHDL Test Bench**».
3) Ввести имя файла в поле «**File name**».
4) Убедиться, что опция «**Add to project**» включена.
5) Нажать на кнопку «**Next**».

6) На втором шаге работы мастера создания файлов проекта необходимо выбрать файл проекта, с которым он будет ассоциирован. Внешний вид окна мастера показан на рисунке 19.

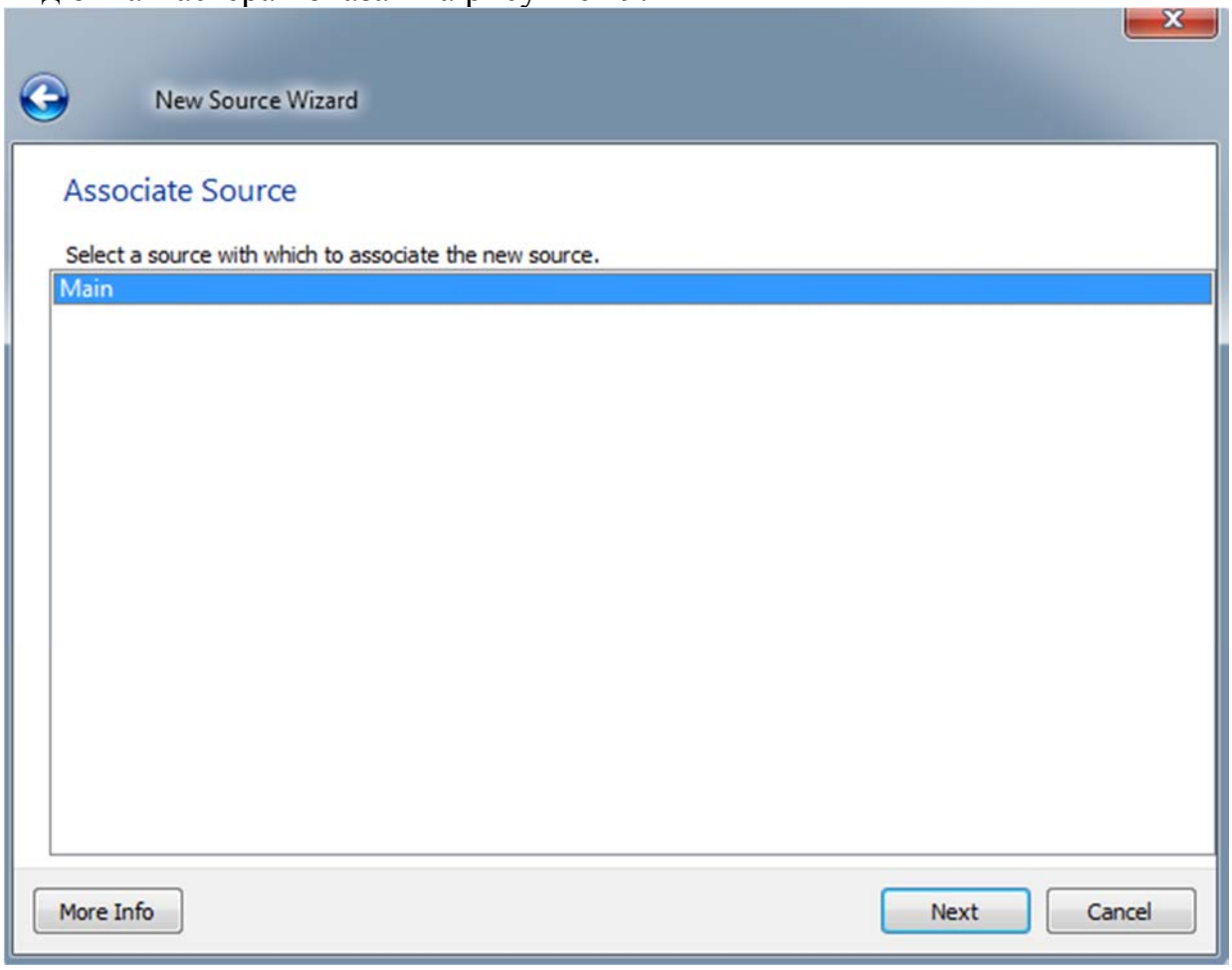

Рис. 19. Окно мастера создания новых тестовых файлов проекта (Шаг 2)

7) После выбора файла необходимо нажать на кнопку «**Next**».

8) На завершающем шаге работы мастера приводится краткий отчет по работе. По нажатию кнопки «**Finish**» мастер создает файл VHDL тестового воздействия с указанными свойствами и завершает свою работу.

Редактирование VHDL файла тестовых воздействий осуществляется в текстовом редакторе и аналогично редактированию VHDL кода.

3.10. Запуск моделирования работы

После создания и подготовки файла тестовых воздействий необходимо запустить процесс моделирования работы. Для этого необходимо:

1) Перейти в режим «**Simulation**» в окне описания проектов («Source window») область 1 на рисунке 3.

2) В выпадающем списке, показанном на рисунке 20, выбрать режим моделирования работы.

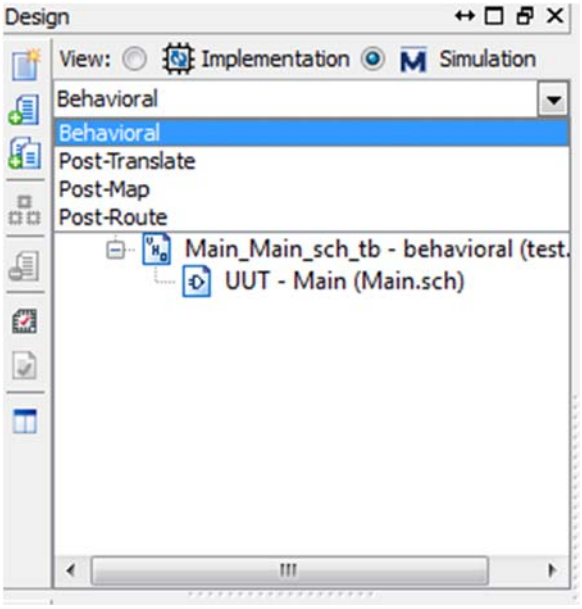

Рис. 20. Режимы моделирования

Режимы моделирования работы ПЛИС:

- «**Behavioral**» – функциональное моделирование;
- «**Post-Translate**» - моделирование после привязки к ПЛИС;
- «**Post-Map**» - моделирование после размещения в ПЛИС;
- «**Post-Route**» - моделирование после трассировки сигналов в ПЛИС с учетом всех задержек;

3) Выделить файл тестового воздействия в окне описания проектов («Source window») область 1 на рисунке 3;

4) Для первого запуска моделирования из САПР Xilinx ISE WebPACK 14, необходимо сделать настройку параметров моделирования работы. Если все уже настроено - необходимо перейти к шагу 7) ;

5) Выбрать «**Process Properties**» в контекстном меню, появляющимся при правом щелчке мыши по «**Simulate … Model**», как показано на рисунке 21;

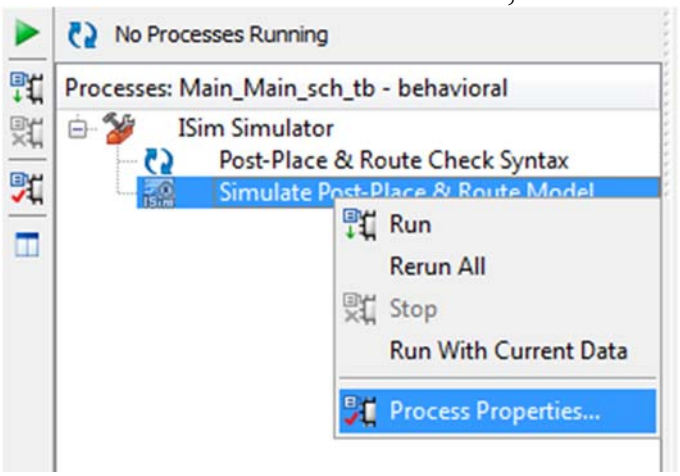

Рис. 21. Открытие окна настройки симулятора

6) Внешний вид окна настройки симулятора показан на рисунке 22

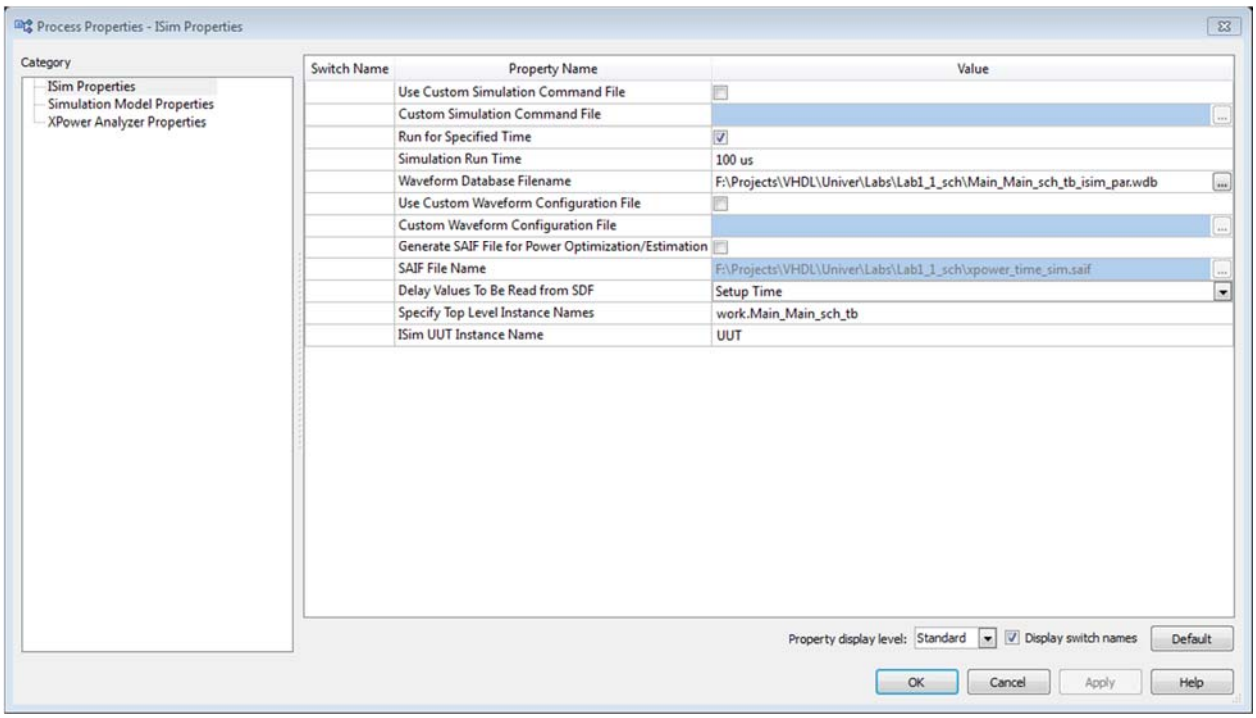

Рис. 22. Окно настройки симулятора

Основными параметрами настройки симулятора являются:

- «**Run for Specified Time**» – опция разрешения указания времени моделирования работы;
- «**Simulation Run Time**» – время моделирования работы;
- «**ISim UUT Instance Name**» – название тестируемого модуля в файле тестового воздействия.

Более подробно об остальных свойства проекта можно почитать, нажав на кнопку «**Help**».

7) Двойным щелчком мыши по «**Simulate … Model**», находящимся в окне процессов в разделе «**ISim Simulator**», как показано на рисунке 23, где под «…» понимается выбранный режим моделирования работы, запускается симулятор ISim.

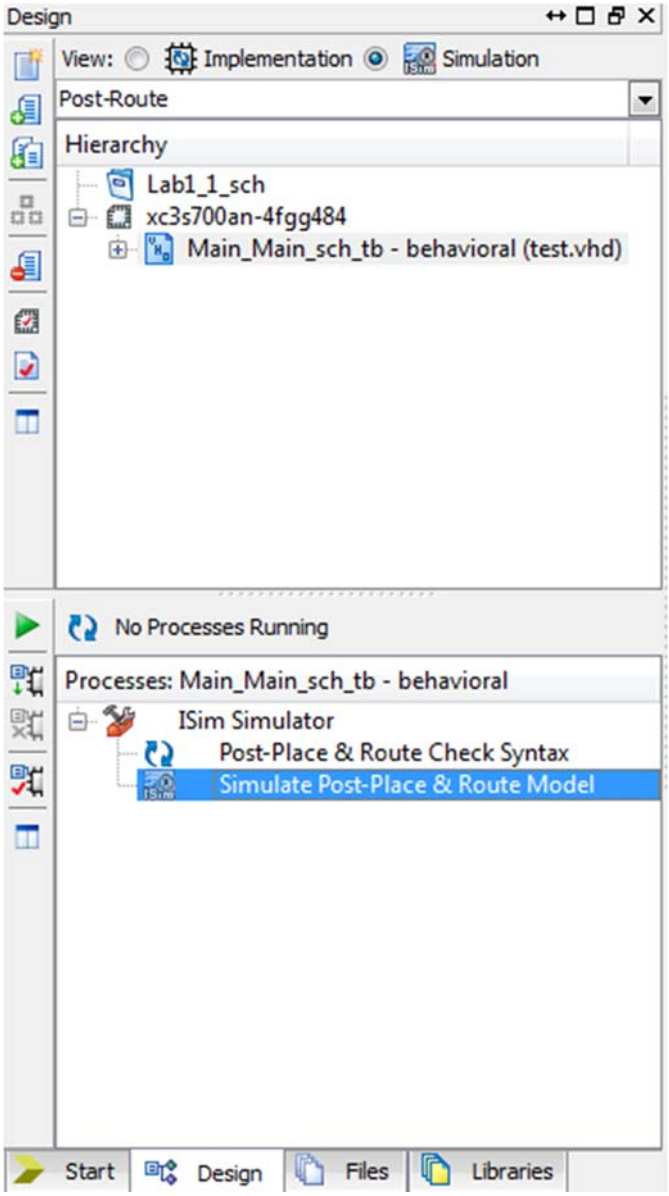

Рис. 23. Запуск моделирования работы

4. Моделирование работы в ISim

САПР Xilinx ISE WebPACK, начиная с версии 13, имеет встроенный симулятор ISim работы ПЛИС. Внешний вид программы ISim показан на рисунке 24.

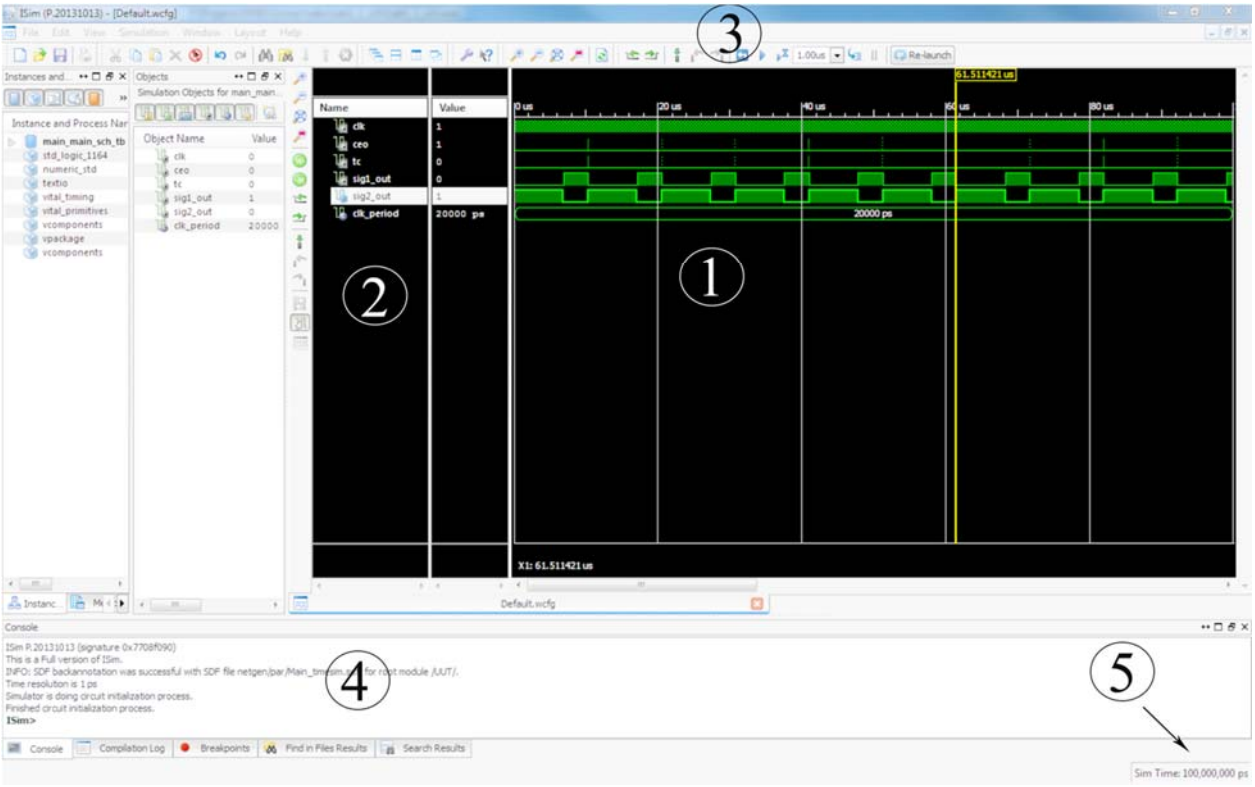

Рис. 24. Симулятор ISim:
1 – диаграммы моделирования; 2 – список сигналов; 3 – панель инструментов («Toolbar»); 4 – окно сообщений; 5 – время моделирования

Визуализация работы симулятора осуществляется с помощью области отображения диаграмм 1 рисунка 24. Список сигналов проиллюстрирован в области 2 рисунка 24. Текущее время моделирования работы показано в области 5 рисунка 24.

Масштабирование диаграмм осуществляется с помощью кнопок, расположенных на панели инструментов в области 3 рисунка 24. Внешний вид кнопок показан на рисунке 25.

Рис. 25. Кнопки масштабирования

Левый щелчок мыши по полю диаграммы устанавливает курсор на эту позицию. Для измерения длительности временного интервала необходимо нажать левую кнопку мыши на начальной позиции и, удерживая её, перевести указатель на интересующую конечную позицию, после чего отпустить. Будет добавлен курсор, и около нижней шкалы времени отобразится время между курсорами.

5. Формирование периодического импульсного сигнала на ПЛИС

Рассмотрим простейшее описание формирователя периодического импульсного сигнала на ПЛИС. Формирователь строится на базе счетчика (для примера используется 8-ми разрядный счетчик) и компаратора, который

осуществляет сравнение значения счетчика и при превышении заданного порога формирует сигнал.

Схемотехническое описание приведено на рисунке 26.

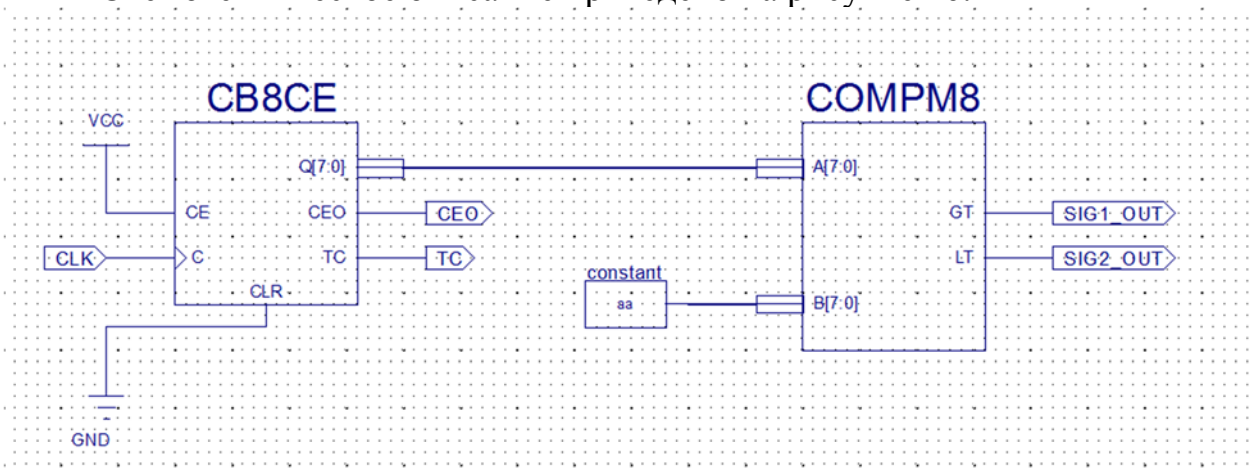

Рис. 26. Схемотехническое описание формирователя импульсов

Исходный VHDL код приведен в листинге 1.

Листинг 1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Main is
  port ( CLK      : in    std_logic;
         CEO      : out   std_logic;
         SIG1_OUT : out   std_logic;
         SIG2_OUT : out   std_logic;
         TC       : out   std_logic);
end Main;
architecture Behavioral of Main is
         constant THRESHOLD : STD_LOGIC_VECTOR(7 downto 0) := x"aa";
         signal COUNT : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
         constant  TERMINAL_COUNT  :  STD_LOGIC_VECTOR(7  downto  0)  :=
(others => '1');
  begin
         process(CLK)
         begin
                  if (CLK'event and CLK = '1') then
                           COUNT <= COUNT+1;
                           if (COUNT > THRESHOLD) then
                                    SIG1_OUT <= '1';
                           else
                                    SIG1_OUT <= '0';
                           end if;
```

```
                    if (COUNT < THRESHOLD) then
                        SIG2_OUT <= '1';
                    else
                        SIG2_OUT <= '0';
                    end if;
                    if (COUNT = TERMINAL_COUNT) then
                        TC  <= '1';
                        CEO <= '1';
                    else
                        TC  <= '0';
                        CEO <= '0';
                    end if;
                end if;
        end process;
end Behavioral;
```

Исходный код VHDL файла тестового воздействия показан в листинге 2.

Листинг 2

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY Main_Main_sch_tb IS
END Main_Main_sch_tb;
ARCHITECTURE behavioral OF Main_Main_sch_tb IS

  COMPONENT Main
  PORT( CLK      :       IN     STD_LOGIC;
        CEO  :       OUT  STD_LOGIC;
        TC     :       OUT  STD_LOGIC;
                     SIG1_OUT :       OUT  STD_LOGIC;
        SIG2_OUT  :       OUT  STD_LOGIC);
  END COMPONENT;

  SIGNAL CLK    :       STD_LOGIC;
  SIGNAL CEO    :       STD_LOGIC;
  SIGNAL TC      :       STD_LOGIC;
  SIGNAL SIG1_OUT   :       STD_LOGIC;
        SIGNAL SIG2_OUT      :       STD_LOGIC;

        constant clk_period:time:=20 ns;
```

```
BEGIN

  UUT: Main PORT MAP(
            CLK => CLK,
            CEO => CEO,
            TC => TC,
            SIG1_OUT => SIG1_OUT,
            SIG2_OUT => SIG2_OUT
  );

-- *** Test Bench - User Defined Section ***
      pclk:process
      begin
            clk<='0';
            wait for clk_period;
            clk<='1';
            wait for clk_period;
      end process;
  tb : PROCESS
  BEGIN
    WAIT; -- will wait forever
  END PROCESS;
-- *** End Test Bench - User Defined Section ***
END;
```

Задания

1) Создайте новый проект описания устройства ПЛИС. В качестве типа основного файла (поле «**Top-level source type**») задайте «**Schematic**». При создании проекта в полях настройки укажите следующее:
   - «**Evaluation Development Board**» - «**Spartan-3AN Starter Kit**»;
   - «**Synthesis Tool**» - «**XST (VHDL/Verilog)»**;
   - «**Simulator**» - «**ISim (VHDL/Verilog)»**;
   - «**Preferred Language**» - «**VHDL»**.
2) Добавьте файл схемотехнического описания (назовите его *Main*);
3) Создайте схему согласно рисунку 26. При разработке используйте 8-ми разрядный счетчик «**CB8CE»** из категории «**counter**», компаратор «**COMPM8**» из категории «**comparator**», символы «**constant**», «**VCC**», «**GND**» из «**general**». Добавьте порты ввода-вывода. Назовите их согласно рисунку 26;
4) Создайте VHDL файл тестового воздействия (назовите его *test*);
5) Доработайте файл тестового воздействия согласно листингу 2.
6) Настройте симулятор, указав следующее:
   - «**Run for Specified Time**» – опция включена;

- «**Simulation Run Time**» – время моделирования работы «**100 us**» (100 мкс);
- «**ISim UUT Instance Name**» – «**UUT**».

7) Запустите процесс моделирования в ISim;
8) Сравните диаграммы моделирования работы ISim, приведённые на рисунках 27-29, с полученными вами диаграммами;

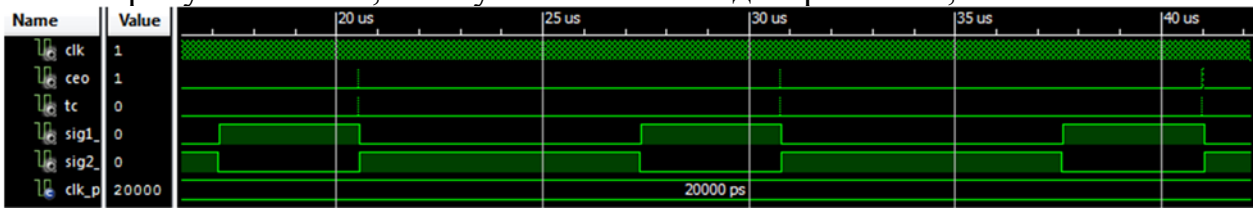

Рис. 27. Моделирование работы ПЛИС. Диаграмма 1

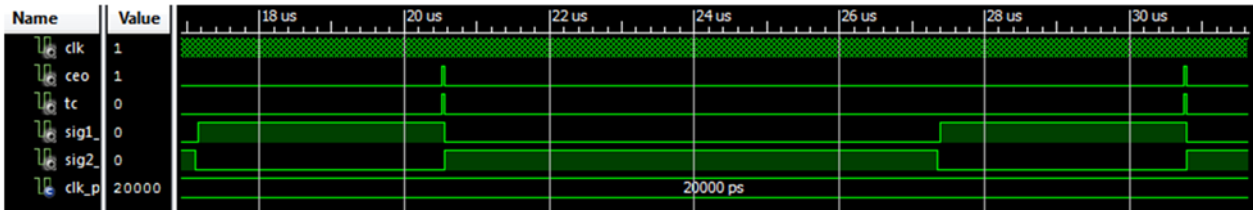

Рис. 28. Моделирование работы ПЛИС. Диаграмма 1

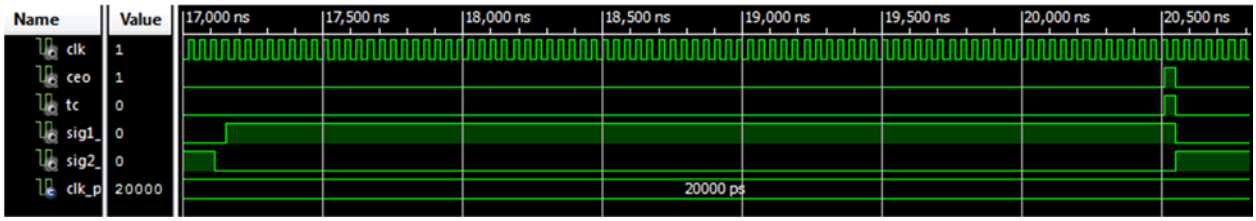

Рис. 29. Моделирование работы ПЛИС. Диаграмма 1

9) Сделайте вывод о соответствии (совпадают/отличаются);
10) Создайте новый проект описания устройства ПЛИС. В качестве типа основного файла (поле «**Top-level source type**») задайте «**HDL**». При создании проекта в полях настройки укажите следующее:
    - «**Evaluation Development Board**» - «**Spartan-3AN Starter Kit**»;
    - «**Synthesis Tool**» - **XST (VHDL/Verilog)**;
    - «**Simulator**» - «**ISim (VHDL/Verilog)**»;
    - «**Preferred Language**» - **VHDL**.
11) Добавьте файл VHDL описания устройства (назовите его *Main*);
12) В файл описания добавьте VHDL код согласно листингу 1;
13) Создайте VHDL файл тестового воздействия (назовите его *test*);
14) Доработайте файл тестового воздействия согласно листингу 2;
15) Настройте симулятор, указав следующее:
    - «**Run for Specified Time**» – опция включена;
    - «**Simulation Run Time**» – время моделирования работы «**100 us**» (100 мкс);
    - «**ISim UUT Instance Name**» – «**UUT**».
16) Запустите процесс моделирования ISim;

17) Сравните диаграммы моделирования работы ISim, приведённые на рисунках 27-31 , с полученными вами диаграммами;

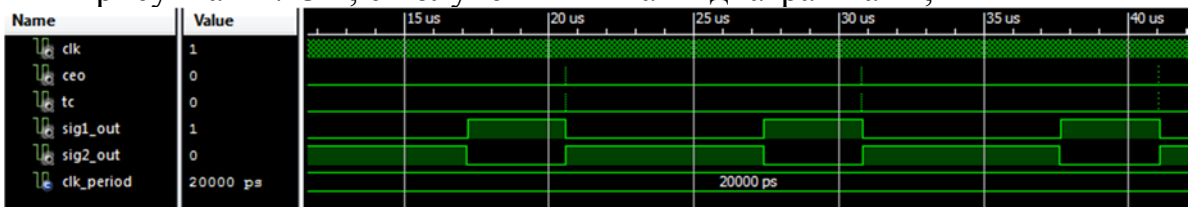

Рис. 30. Моделирование работы ПЛИС. Диаграмма 1

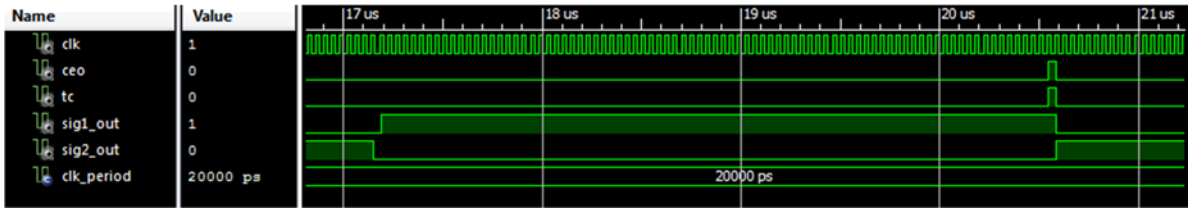

Рис. 31. Моделирование работы ПЛИС. Диаграмма 1

18) Сделайте вывод о соответствии (совпадают/отличаются).

6. Генерирование сигнала определенной формы

Рассмотрим более сложный случай формирования сигнала определённой формы на ПЛИС. Конструкции языка описания, использующие вещественную арифметику, в ПЛИС являются не синтезируемыми, т.к. в ПЛИС напрямую работать с вещественными числами нельзя, поскольку она состоит из элементарных логических элементов. Для генерирования сигнала определённой формы на ПЛИС наиболее простое решение - это предварительно записать сигнал в ПЗУ и при работе производить его чтение из ПЗУ.

Формирователь сигнала определённой формы также как и формирователь периодического импульсного сигнала строится на базе счетчика (для примера используется 8-ми разрядный счетчик), но вместо компаратора, используется ПЗУ, в котором хранятся отсчеты сигнала. Счетчик используется в качестве указания адреса ПЗУ.

Схемотехническое описание приведено на рисунке 32

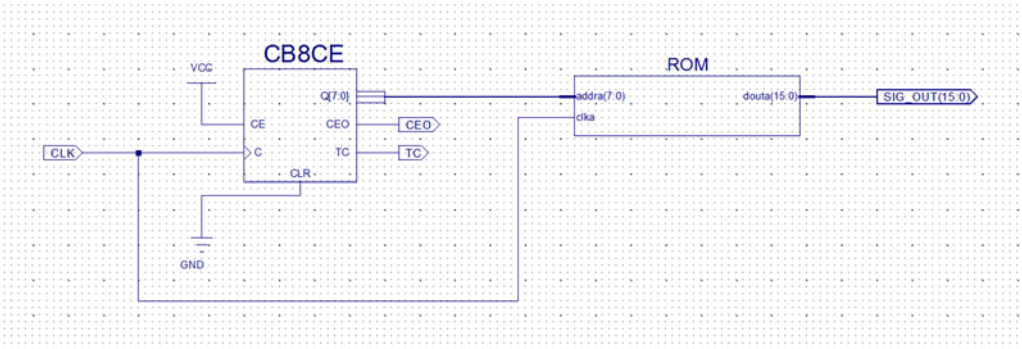

Рис. 32. Схемотехническое описание формирователя сигналов определённой формы

Исходный VHDL код приведен в листинге 3.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Main is
  port ( CLK     : in    std_logic;
         CEO     : out   std_logic;
         SIG_OUT : out   std_logic_vector (15 downto 0);
         TC      : out   std_logic);
end Main;

architecture Behavioral of Main is
         attribute BOX_TYPE   : string ;
  component rom
    port ( addra : in    std_logic_vector (7 downto 0);
           douta : out   std_logic_vector (15 downto 0);
           clka  : in    std_logic);
  end component;
         attribute BOX_TYPE of rom : component is "BLACK_BOX";

         constant THRESHOLD : STD_LOGIC_VECTOR(7 downto 0) := x"aa";
         signal COUNT : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
         constant  TERMINAL_COUNT  :  STD_LOGIC_VECTOR(7  downto  0)  :=
(others => '1');

begin

         process(CLK)
         begin
                  if (CLK'event and CLK = '1') then
                           COUNT <= COUNT+1;
                           if (COUNT = TERMINAL_COUNT) then
                                    TC  <= '1';
                                    CEO <= '1';
                           else
                                    TC  <= '0';
                                    CEO <= '0';
                           end if;
                  end if;
         end process;
  MEMORY_ROM : rom
    port map (addra(7 downto 0)=>COUNT(7 downto 0),
              clka=>CLK,
```

```
        douta(15 downto 0)=>SIG_OUT(15 downto 0));
end Behavioral;
```

Исходный код VHDL файла тестового воздействия показан в листинге 4.

Листинг 4

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY Main_Main_sch_tb IS
END Main_Main_sch_tb;
ARCHITECTURE behavioral OF Main_Main_sch_tb IS

  COMPONENT Main
  PORT( CLK      :       IN     STD_LOGIC;
        CEO  :       OUT  STD_LOGIC;
        TC     :       OUT  STD_LOGIC;
        SIG_OUT    :       OUT  STD_LOGIC_VECTOR(15 downto 0));
  END COMPONENT;

  SIGNAL CLK    :       STD_LOGIC;
  SIGNAL CEO    :       STD_LOGIC;
  SIGNAL TC      :       STD_LOGIC;
  SIGNAL SIG_OUT     :       STD_LOGIC_VECTOR(15 downto 0);

        constant clk_period:time:=20 ns;

BEGIN

  UUT: Main PORT MAP(
                CLK => CLK,
                CEO => CEO,
                TC => TC,
                SIG_OUT => SIG_OUT
  );

-- *** Test Bench - User Defined Section ***
        pclk:process
        begin
                clk<='0';
                wait for clk_period;
                clk<='1';
```

```
            wait for clk_period;
        end process;
    tb : PROCESS
    BEGIN
      WAIT; -- will wait forever
    END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

Задания

1) Создайте новый проект описания устройства ПЛИС. В качестве типа основного файла (поле «**Top-level source type**») задайте «**Schematic**». При создании проекта в полях настройки укажите следующее:
   - «**Evaluation Development Board**» - «**Spartan-3AN Starter Kit**»;
   - «**Synthesis Tool**» - «**XST (VHDL/Verilog)»**;
   - «**Simulator**» - «**ISim (VHDL/Verilog)»**;
   - «**Preferred Language**» - «**VHDL»**.
2) Добавьте файл схемотехнического описания (назовите его *Main*);
3) Добавьте в проект IP ядро ПЗУ памяти (назовите его *ROM*). При добавлении ядра укажите размер памяти (Depth) 256, а ширину шины данных (Width) - 16 разрядов. Включите опцию «**Load Init File**» в разделе «**Memory Initialization**» на шаге 4 работы мастера добавления IP ядра ПЗУ. В зависимости от вашего варианта задания, выберите через кнопку «**Browse**» файл с подготовленным сигналом:
   - Sin.coe (Синус);
   - Cos.coe (Косинус);
   - Saw.coe (Пила);
4) Создайте схему согласно рисунку 32. При разработке используйте 8-ми разрядный счетчик «**CB8CE»** из категории «**counter**», IP ядро и символы «**VCC**», «**GND**» из «**general**». Добавьте порты ввода-вывода. Назовите их согласно рисунку 32.
5) Создайте VHDL файл тестового воздействия (назовите его *test*);
6) Доработайте файл тестового воздействия согласно листингу 4;
7) Настройте симулятор, указав следующее:
   - «**Run for Specified Time**» – опция включена;
   - «**Simulation Run Time**» – время моделирования работы «**100 us**» (100 мкс);
   - «**ISim UUT Instance Name**» – «**UUT**».
8) Запустите процесс моделирования в ISim;
9) В соответствии со своим вариантом, сравните диаграммы моделирования работы, приведённые на рисунках 33-39, с полученными вами диаграммами;

Диаграммы для синуса

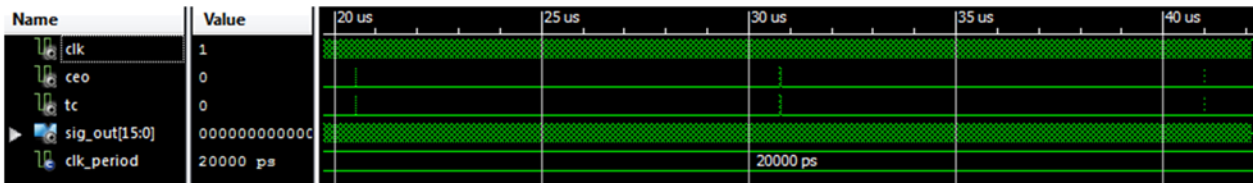

Рис. 33. Моделирование работы ПЛИС. Диаграмма 1

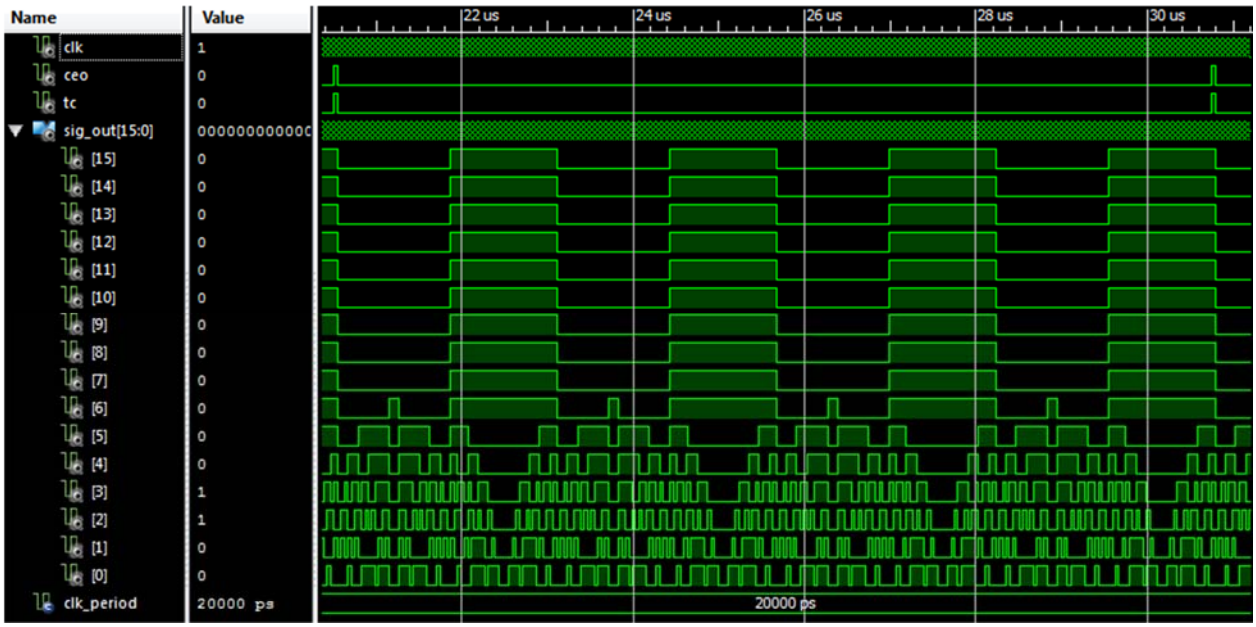

Рис. 34. Моделирование работы ПЛИС. Диаграмма 1

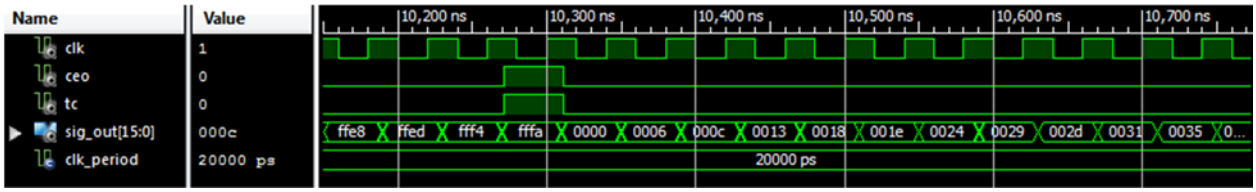

Рис. 35. Моделирование работы ПЛИС. Диаграмма 1

Диаграммы для косинуса

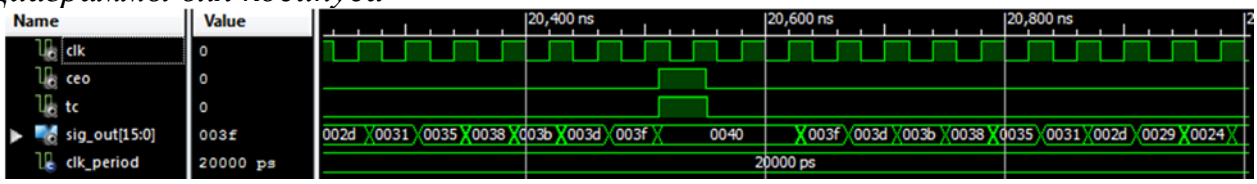

Рис. 36. Моделирование работы ПЛИС. Диаграмма 1

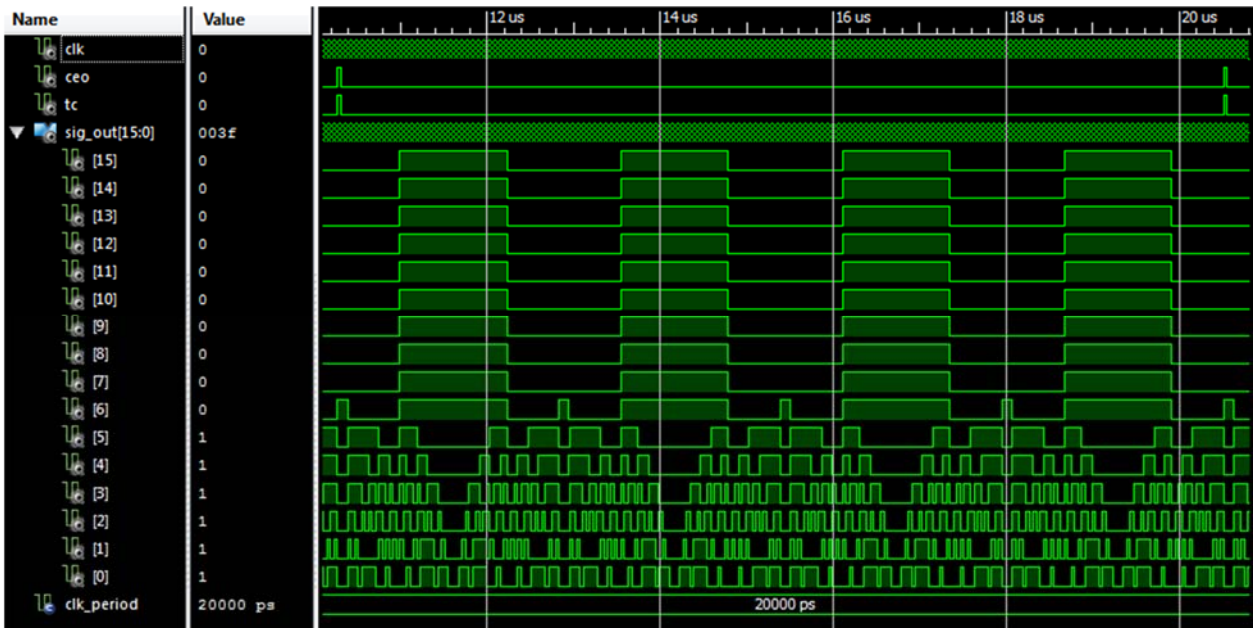

Рис. 37. Моделирование работы ПЛИС. Диаграмма 1

Диаграммы для пилы

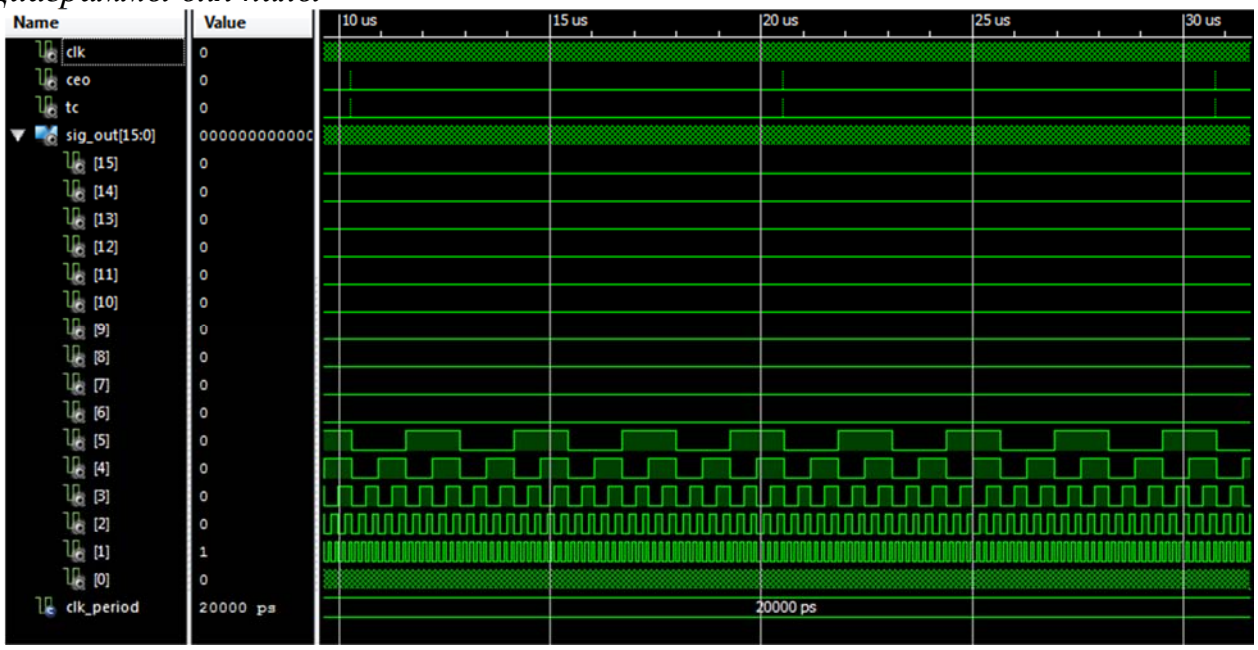

Рис. 38. Моделирование работы ПЛИС. Диаграмма 1

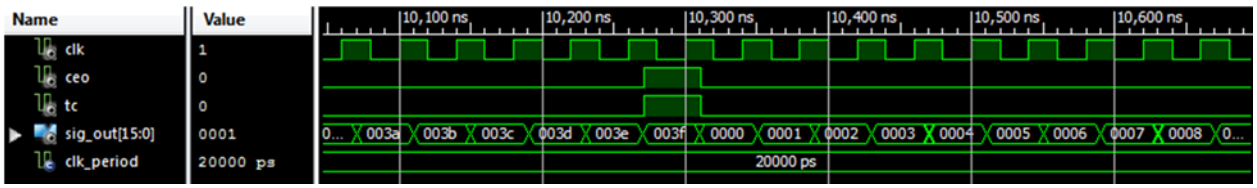

Рис. 39. Моделирование работы ПЛИС. Диаграмма 1

10) Сделайте вывод о соответствии (совпадают/отличаются);
11) Создайте новый проект описания устройства ПЛИС. В качестве типа основного файла (поле «**Top-level source type**») задайте «**Schematic**». При создании проекта в полях настройки укажите следующее:

- «**Evaluation Development Board**» - «**Spartan-3AN Starter Kit**»;
- «**Synthesis Tool**» - «**XST (VHDL/Verilog)**»;
- «**Simulator**» - «**ISim (VHDL/Verilog)**»;
- «**Preferred Language**» - «**VHDL**».

12) Добавьте файл VHDL описания устройства (назовите его *Main*);
13) В файл описания добавьте VHDL код согласно листингу 3;
14) Добавьте в проект IP ядро ПЗУ памяти (назовите его *ROM*);
15) Создайте VHDL файл тестового воздействия (назовите его *test*);
16) Доработайте файл тестового воздействия согласно листингу 4;
17) Настройте симулятор, указав следующее:
    - «**Run for Specified Time**» – опция включена;
    - «**Simulation Run Time**» – время моделирования работы «**100 us**» (100 мкс);
    - «**ISim UUT Instance Name**» – «**UUT**».
18) Запустите процесс моделирования в ISim.
19) В соответствии со своим вариантом, сравните диаграммы моделирования работы, приведённые на рисунках 40-45, с полученными вами диаграммами;

Диаграммы для синуса

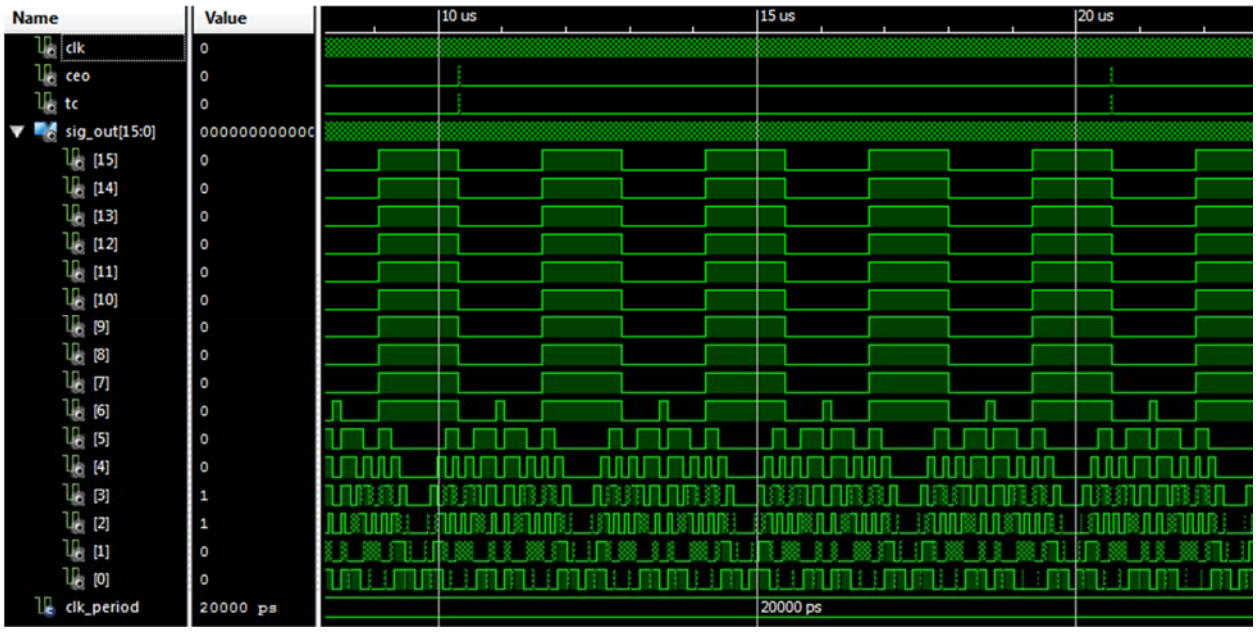

Рис. 40. Моделирование работы ПЛИС. Диаграмма 1

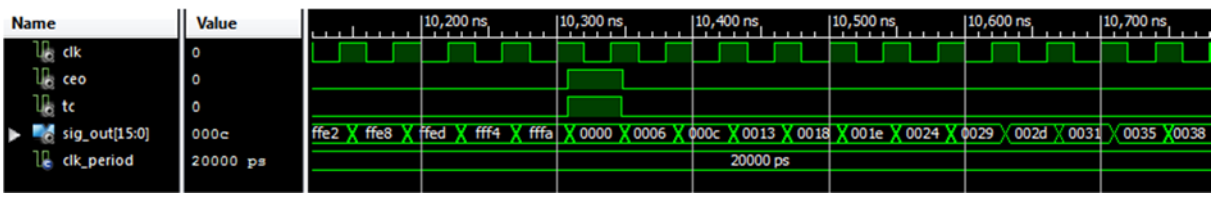

Рис. 41. Моделирование работы ПЛИС. Диаграмма 1

Диаграммы для косинуса

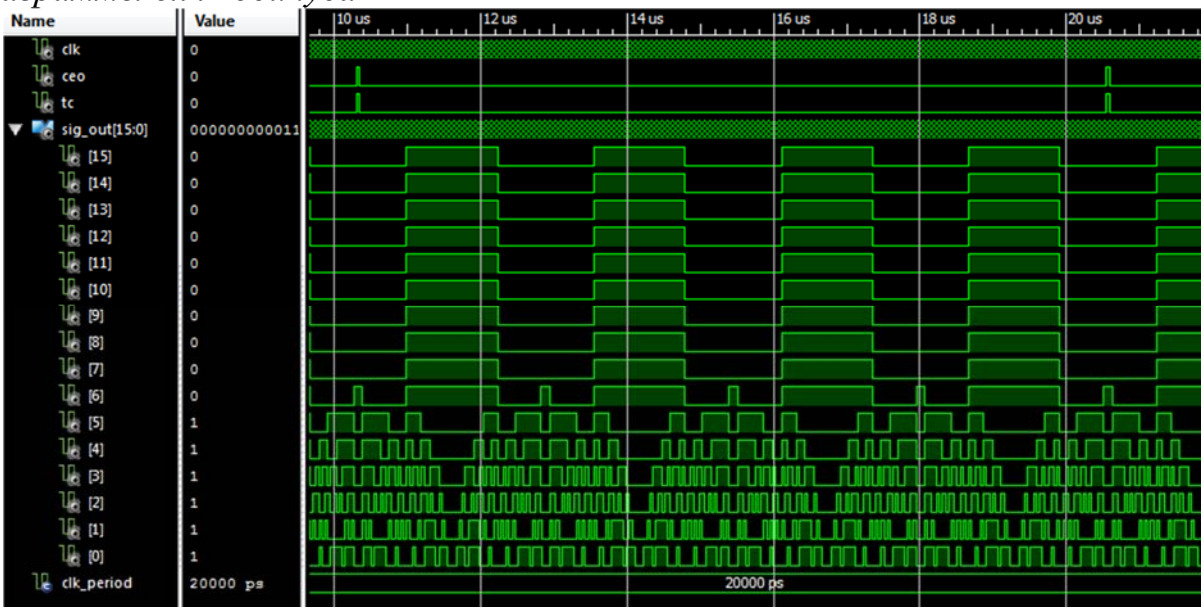

Рис. 42. Моделирование работы ПЛИС. Диаграмма 1

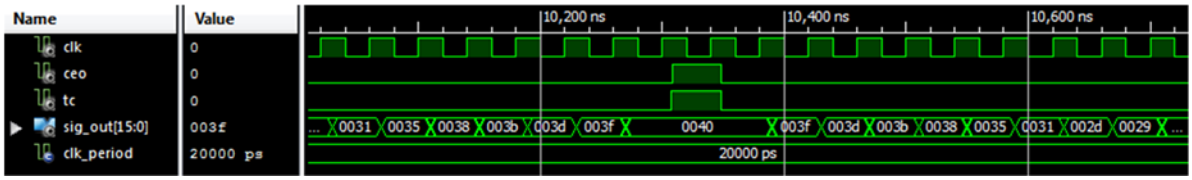

Рис. 43. Моделирование работы ПЛИС. Диаграмма 1

Диаграммы для пилы

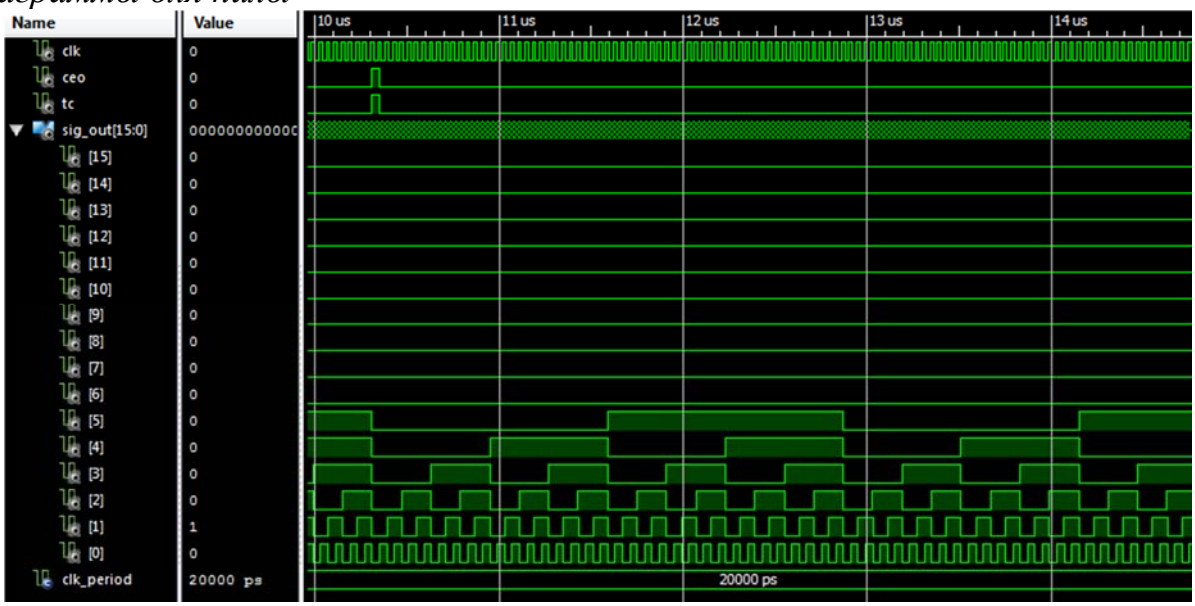

Рис. 44. Моделирование работы ПЛИС. Диаграмма 1

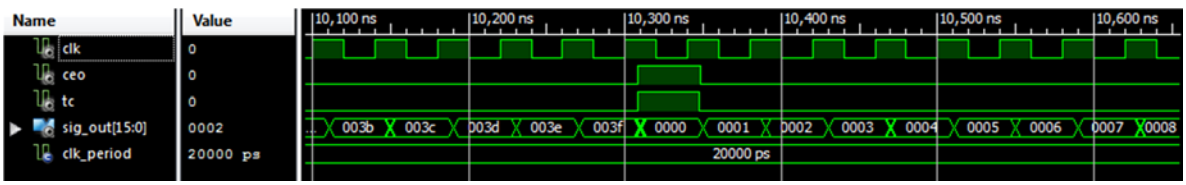

Рис. 45. Моделирование работы ПЛИС. Диаграмма 1

20) Сделайте вывод о соответствии (совпадают/отличаются).

7. Контрольные вопросы

1) Какова длительность импульса SIG1_OUT?
2) Какой период сигнала SIG1_OUT?
3) Какая скважность сигнала SIG1_OUT?
4) В чем проявляется зависимость от числа (константы) компаратора?
5) Как изменится сигнал (его длительность, период) при увеличении тактовой частоты ПЛИС в два раза?
6) Сколько ресурсов занимает проект (задание 1) в ПЛИС?
7) Какова частота сгенерированного синуса?
8) Сколько периодов синусоидального сигнала укладывается в период счетчика?
9) Какова амплитуда синусоидального сигнала?
10) Сколько ресурсов занимает проект (задание 2) в ПЛИС?

Список литературы

1. Спецификация Spartan-3AN [Электронный ресурс]. – Режим доступа: http://www.xilinx.com/support/documentation/data_sheets/ds557.pdf
2. Руководство пользователя Spartan-3AN [Электронный ресурс]. – Режим доступа: http://www.xilinx.com/support/documentation/user_guides/ug331.pdf
3. Руководство пользователя оценочной платы Spartan-3AN [Электронный ресурс]. – Режим доступа: http://www.xilinx.com/support/documentation/boards_and_kits/ug334.pdf
4. Зотов В.Ю. Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WEBPACK ISE. - М.: Горячая линия-Телеком, 2003. - 624 с.
5. Бибило П.Н. Основы языка VHDL. - М.: СОЛОН-Р, 2002. - 224 с.
6. Сергиенко А.М. VHDL для проектирования вычислительных устройств. - Киев: ТИД ДС, 2003. - 208 с.

Содержание

ЗНАКОМСТВО С ПРОГРАММИРУЕМОЙ ЛОГИЧЕСКОЙ ИНТЕГРАЛЬНОЙ СХЕМОЙ СЕМЕЙСТВА SPARTAN-3AN ФИРМЫ XILINX

Составители:
Виталий Юрьевич **Семенов**
Владимир Владимирович **Артемьев**

Учебно-методическое пособие

Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский Нижегородский государственный университет
им. Н.И. Лобачевского».
603950, Нижний Новгород, пр. Гагарина, 23.