Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет информационной безопасности

Кафедра защиты информации

С. Л. Прищепа

ПРОГРАММИРОВАНИЕ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 1-98 01 02 «Защита информации в телекоммуникациях»

Минск БГУИР 2023

УДК 621.3.049.77(076.5)
ББК 32.844.1
П77

Р е ц е н з е н т ы:

кафедра физики твердого тела и нанотехнологий
Белорусского государственного университета
(протокол № 7 от 17.02.2023);

и. о. заведующего Центром широкозонной нано- и микроэлектроники
Института физики НАН Беларуси кандидат физико-математических наук,
доцент Е. В. Луценко

Прищепа, С. Л.

П77 Программирование логических интегральных схем. Лабораторный практикум : учеб.-метод. пособие / С. Л. Прищепа. – Минск : БГУИР, 2023. – 60 с. : ил.

ISBN 978-985-543-733-9.

Состоит из шести лабораторных работ, каждая из которых включает в себя описание цифрового устройства, которое необходимо реализовать в САПР MAX + Plus II, лабораторное задание, содержание отчета, контрольные вопросы к каждой теме.

Предназначен для студентов специальности 1-98 01 02 «Защита информации в телекоммуникациях», изучающих дисциплину «Проектирование программируемых цифровых устройств».

УДК 621.3.049.77(076.5)
ББК 32.844.1

ISBN 978-985-543-733-9

Содержание

Лабораторная работа № 1
Знакомство с основными модулями САПР MAX+PLUS II. Настройка ПЛИС на реализацию функции одноразрядного двоичного полусумматора и сумматора

Общие положения

Отметим основные этапы выполнения задания в САПР MAX+PLUS II, которые являются общими для решения разных задач:

– создание проекта с помощью редакторов проектов (графического и текстового);

– компиляция проекта, в которую входят его анализ, детализация и синтез;

– работа в редакторе временных диаграмм, в котором задаются входные переменные;

– симуляция (моделирование) работы проекта во времени, которое необходимо для тестирования проекта на соответствие спецификации (техническому заданию);

– размещение и трассировка схемы проекта на кристалле;

– программирование (конфигурирование) проекта в ПЛИС.

В данной и последующих лабораторных работах будут применяться только первые четыре этапа работы с изучаемой системой автоматизированного проектирования (САПР). Более подробно структура САПР описана в [1; 2].

Цель работы

1. Ознакомиться:

– с САПР электронных устройств MAX+PLUS II;

– работой в графическом редакторе этой системы;

– работой компилятора;

– работой в графическом редакторе. Приобрести навыки анализа временных диаграмм;

– редактором символов. Научиться создавать и использовать при выполнении проекта созданные символы;

– работой иерархического дисплея;

– работой поуровневого планировщика.

2. Настроить программируемую логическую интегральную схему (ПЛИС) на реализацию функции полусумматора.

3. Настроить ПЛИС на реализацию функции одноразрядного двоичного полного сумматора.

4. Проанализировать полученные результаты и убедиться в правильности работы алгоритма.

Часть 1. Одноразрядный двоичный полусумматор

В качестве примера выберем схему одноразрядного двоичного полусумматора, выполненного на логических вентилях. Перечислим основные этапы работы при выполнении проекта.

Программа работы

1. Создание нового проекта или открытие существующего.

Для выполнения этого пункта необходимо выполнить следующие действия:

– запустить MAX+PLUS II с иконки на рабочем столе компьютера;

– установить имя проекта, выбрав команду меню **File | Name** (Файл | Имя). Имя проекта назначать только латинскими буквами (не кириллицей);

– для создания нового файла схемы разрабатываемого устройства выбрать кнопку **New** на панели инструментов или использовать команду меню **File | New** (Файл | Новый). (Для открытия существующего файла щелкнуть по кнопке **Open** или выполнить команду **File | Open** (Файл | Открыть));

– в диалоговом окне выбрать один из редакторов для создания нового файла изображения схемы. В данном задании выбираем графический редактор Graphic Editor. Файл изображения схемы будет иметь расширение gdf. Вид окна программы для выбора графического редактора приведен на рисунке 1;

– в окне выбранного редактора сохранить файл с нужным именем **File | Save as** (Файл | Сохранить как)). Проект сохранять в предварительно созданную папку с номером своей группы;

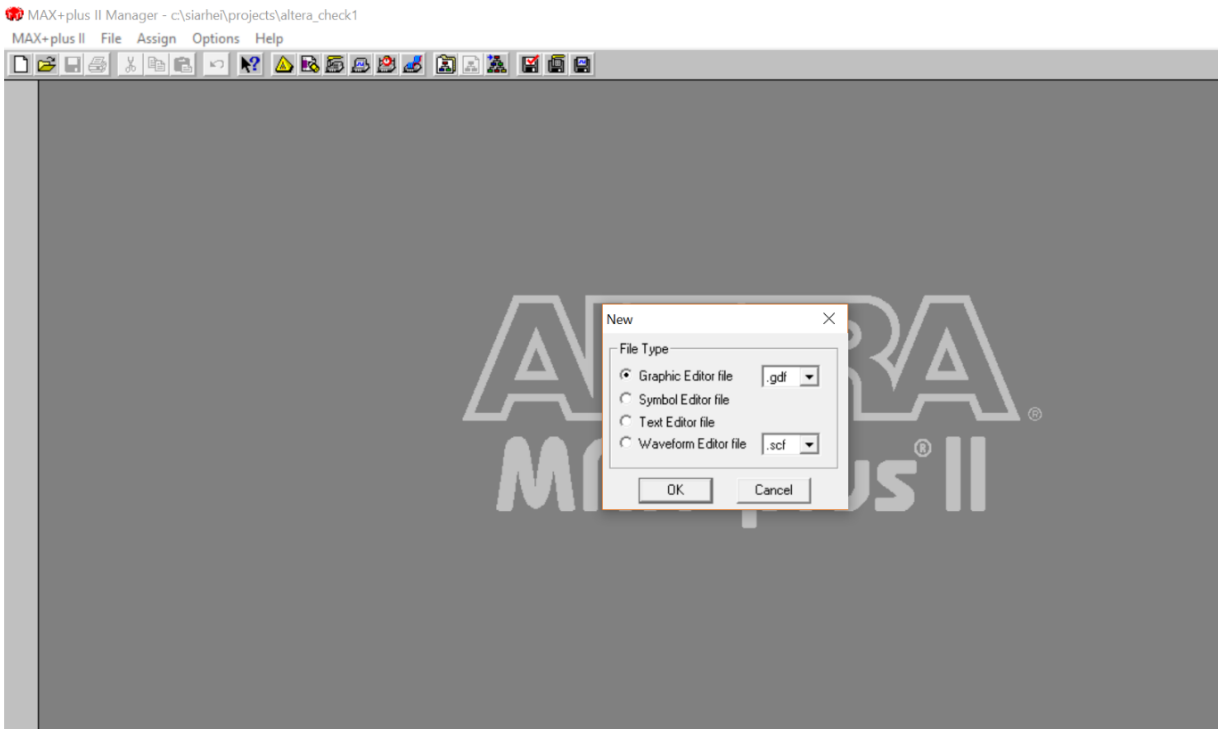

Рисунок 1 – Выбор графического редактора с расширением gdf

2. Ввод функциональной схемы полусумматора с помощью Graphic Editor.

Ввод функциональной схемы осуществляется в следующей последовательности:

– ввод элементов схемы в графическом редакторе осуществляется двойным щелчком левой кнопки мыши (ЛКМ) в выбранном поле графического редактора. В результате появляется диалоговое окно **Enter Symbol** (Ввод | Символ). Необходимо выбрать нужную библиотеку элементов (для данного примера это соответствует библиотеке примитивов **prim**) и имя элемента. Следует повторить вышеуказанную процедуру для всех элементов схемы (имена символов в библиотеке: or2 – «2ИЛИ», and2 – «2И», not – «Инвертор»). Пример диалогового окна **Enter Symbol** (Ввод | Символ) с указанием содержимого в библиотеке примитивов приведен на рисунке 2;

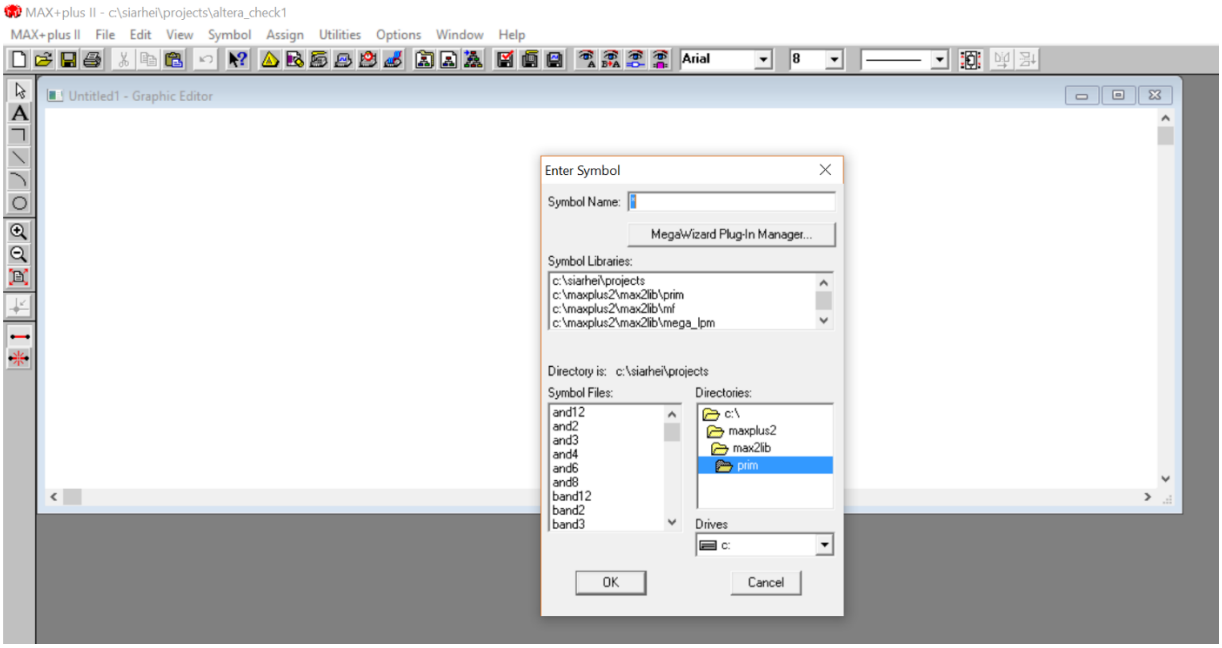

Рисунок 2 – Диалоговое окно **Enter Symbol** (Ввод | Символ)

– для удобства работы в рабочем поле проекта используется сетка для привязки элементов к координатам экрана. Как правило, она указана тонкими пунктирными линиями. Чтобы показать сетку в поле графического редактора, необходимо выполнить команду **Options | Show Guidelines** (Опции | Показать сетку). Для изменения размеров сетки используется команда меню **Options | Guidelines Spacing** (Опции | Интервал сетки);

– перемещение элемента по полю проекта выполняется стандартным образом в режиме Drag&Drop. Для этого следует выделить необходимый элемент и движением указателя мышки перетащить его в нужное место;

– чтобы получить копию любого элемента схемы, надо выделить его и использовать режим Drag&Drop, но уже с нажатой клавишей <Ctrl>. Указанная процедура является быстрым заменителем известного алгоритма копирования элемента через CTRL+C – CTRL V;

– установить режим «резиновой нити» **Options | Rubberbanding** (Опции | Резиновая нить). Этот режим позволяет легко соединять нужные элементы между собой в поле графического редактора. Причем линии могут быть и не привязаны к сетке в поле графического редактора, а проводиться под разными углами к горизонтальным линиям в зависимости от необходимости и целесообразности построения компактной схемы;

– для соединения элементов выделить соединяемый контакт и, удерживая левую кнопку мыши, протащить курсор до следующего присоединяемого контакта. Проследить, чтобы соединение между элементами было установлено. Для соединения элементов через шину, состоящую из множества проводов, нужно выбрать жирную линию в контекстном меню, отметив пункт меню **Line Style** (Стиль линии), и повторить ранее описанные действия. Для удаления соединения надо выделить его и нажать клавишу <Delete>;

– особенностью построения схемы в графическом редакторе является то, что просто собрать необходимую схему недостаточно. Для правильного функционирования устройства, помимо логических элементов и/или символов, необходимо наличие входных (input) и выходных (output) контактов. Тем самым обозначаются входы и выходы схемы. Чтобы внести входные (input) и выходные (output) контакты в схему, нужно в окне **Enter Symbol** в библиотеке примитивов в поле имени элемента выбрать соответственно элемент input или output;

– каждый вход и выход схемы должен иметь свое уникальное имя. Двойной щелчок в области имени контакта позволяет назначить новое имя. Имя необходимо набирать латинскими буквами.

В соответствии с вышеуказанным алгоритмом ввода функциональной схемы необходимо собрать в графическом редакторе схему полусумматора, которая изображена на рисунке 3. Для этого задания необходимо в поле графического редактора из библиотеки примитивов поместить следующие элементы:

– дизъюнктор OR2 (2ИЛИ) – 1 шт.;
– конъюнктор AND2 (2И) – 2 шт.;
– инвертор NOT (НЕ) – 1 шт.;
– входы INPUT – 2 шт.;
– выходы OUTPUT – 2 шт.

Далее необходимо сохранить файл схемы **File | Save** (Файл | Сохранить) и выполнить важную процедуру привязки проекта к данному файлу. Для этого необходимо выбрать меню **File | Project | Set Project to Current File** (Файл | Проект | Привязать проект к данному файлу).

В дальнейшем понадобится использовать схему полусумматора в качестве символа. Для того чтобы в САПР MAX+PLUS II создать символ, надо выбрать команду меню **File | Create Default Symbol** (Файл | Создать символ, заданный по умолчанию). В результате система автоматизированного проектирования автоматически создаст файл символа с расширением sym. После выполнения указанных процедур можно закрыть файл с помощью операции **File | Close** (Файл | Закрыть).

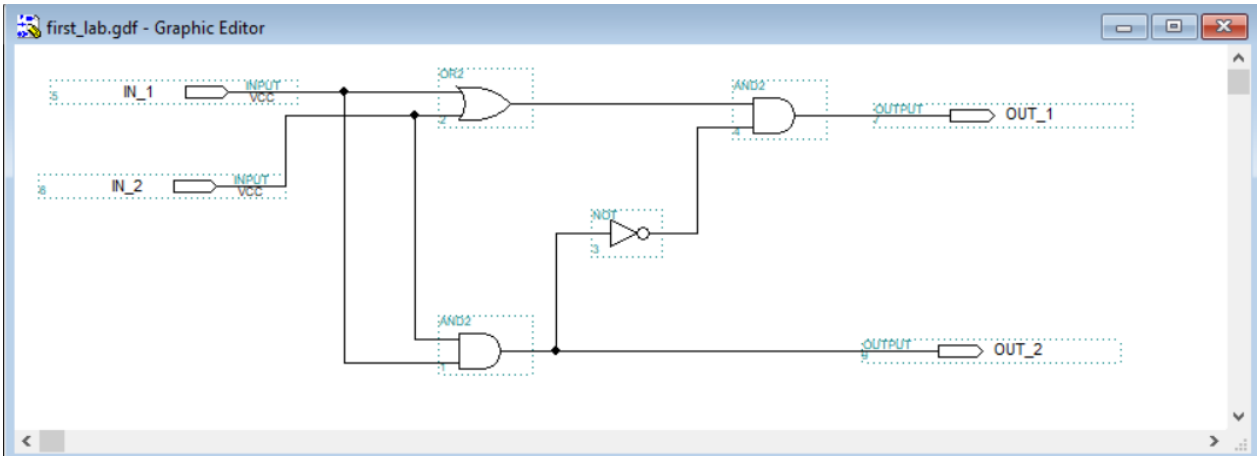

Рисунок 3 – Схема полусумматора в графическом редакторе

3. Компиляция проекта.

Это важный этап выполнения проекта. На этой стадии САПР осуществляет синтез структуры, трассировку связей, проверку корректности проекта и локализует обнаруженные ошибки. Кроме того, на стадии компиляции происходит формирование файлов программирования и/или конфигурирования программируемых логических интегральных схем. В состав компилятора входят несколько приложений. Кратко перечислим основные из них.

Приложение **«Compiler Netlist Extractor»** отвечает за создание списка соединений из исходного файла представления проекта, который был создан на стадии ввода проекта.

Приложение **«Database Builder»** предназначено для построения базы данных проекта.

Приложение **«Logic Synthesizer»** ответственно за проверку корректности проекта по формальным признакам и отвечает за синтез оптимальной структуры проекта. Оно наиболее часто позволяет оперативно исправить формальные ошибки в проекте, такие как дубликат названий, обрыв соединительных линий и пр.

Приложение **«Partitioner»** обеспечивает разбиение проекта на части в тех случаях, когда нет возможности реализовать проект на одной интегральной микросхеме.

Приложение **«Fitter»** представляет собой трассировщик внутренних связей, обеспечивающих оптимальную реализацию синтезированной структуры и оптимальное использование площади программируемой логической интегральной схемы.

Приложение **«Timing SNF Extractor»** отвечает за извлечение параметров проекта, которые необходимы для последующего функционального моделирования и временного анализа.

Наконец, приложение **«Assembler»** – это приложение, создающее из скомпилированного проекта файлы, необходимые для программирования или конфигурирования ПЛИС через программатор.

Для осуществления операции компиляции проекта необходимо выполнить следующие действия:

– выбрать семейство **ПЛИС** (**PLD – Programmable Logic Devices**) и конкретное устройство в этом семействе для данного проекта, используя команду меню **Assign | Device** (Назначить | Устройство). При работе с данным конкретным проектом семейство интегральной микросхемы (ИМС) в принципе не важно в силу простоты решаемой задачи. Ввиду этого лучше указать режим **AUTO** для выбора конкретного устройства, заданного по умолчанию;

– открыть окно компилятора – **MAX+Plus II | Compiler**, рисунок 4;

– нажать кнопку **Start** для запуска компилятора. В процессе работы компилятор открывает окно процессора сообщений **Message Processor**, где можно посмотреть обнаруженные ошибки.

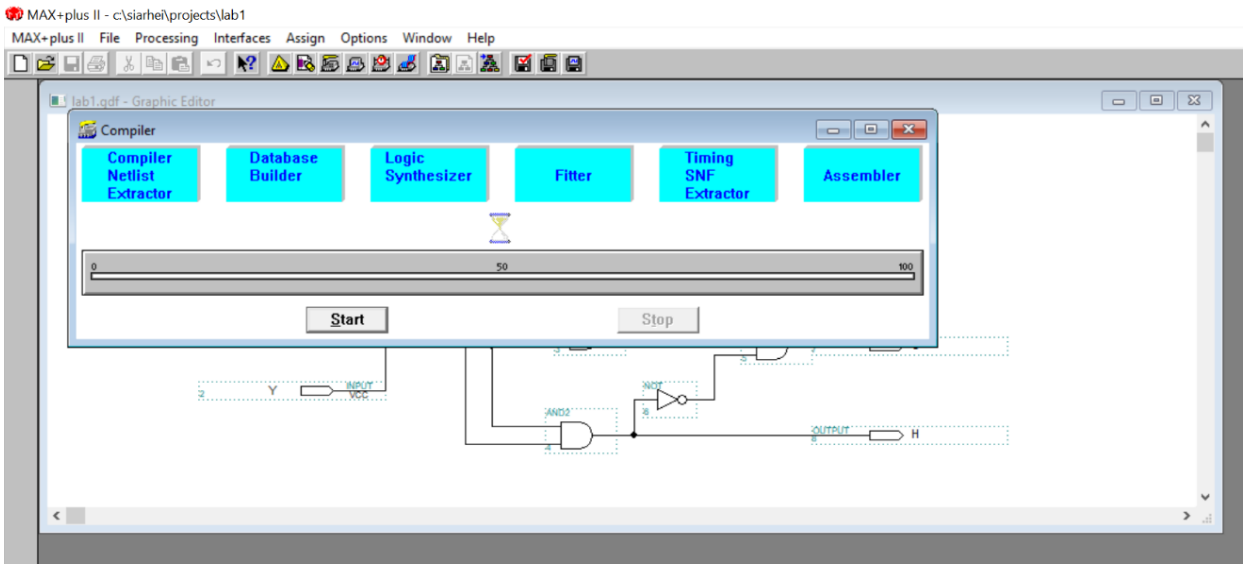

Рисунок 4 – Окно компилятора

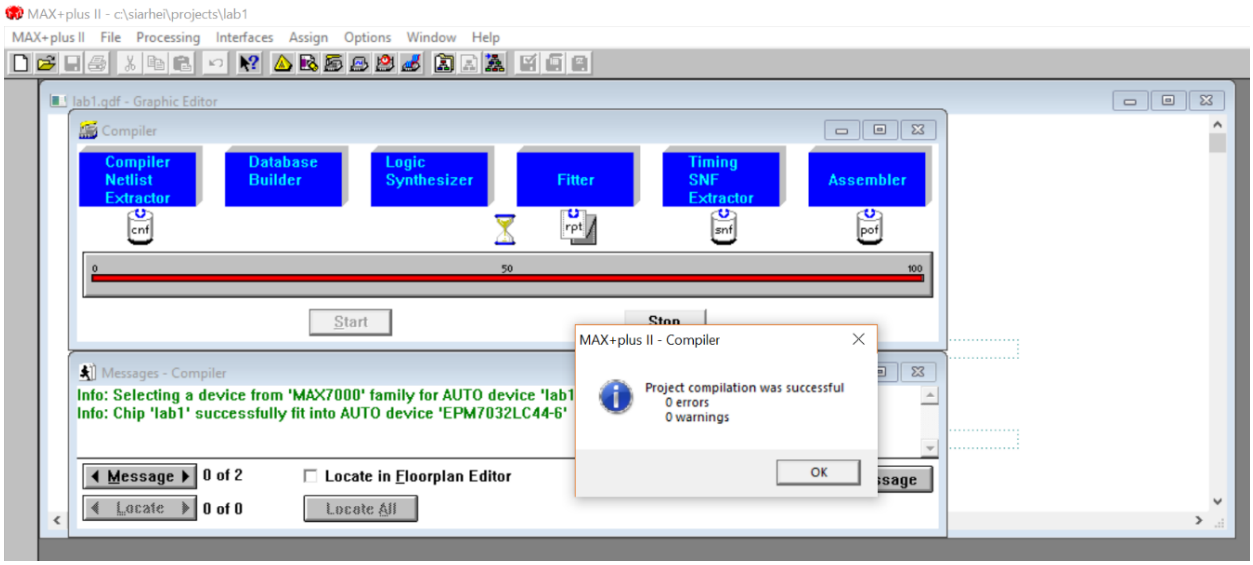

Рисунок 5 – Результат компиляции проекта

В случае обнаружения компилятором ошибок необходимо вернуться в Graphic Editor для внесения изменений в схему. После чего следует сохранить изменения и провести компиляцию проекта заново. В окне компилятора сообщения зеленого цвета носят информационный характер, сообщения синего цвета – предупреждающий. Предупреждения могут возникать в случае наличия входной переменной, которая не используется, наличия вырожденных конструкций, например триггера, выход которого не может изменить свое значение ни при каких условиях, и т. д.

Несколько наиболее часто допускаемых ошибок при создании схемы:

а) *Duplicate pin name*: несколько входов или выходов имеют одинаковое имя;

б) *Output pinstubs … and … are tied together*: выходы нескольких компонентов соединены между собой;

в) *Node missing source*: один из входов компонента не подключен;

г) *Node has more than one name*: цепь имеет более чем одно имя.

После достижения успешной компиляции проекта следует закрыть окно компилятора.

4. Назначение входных переменных.

Входные переменные назначаются следующим образом:

– выбрать команду **File | New** и **Wavefrom Editor** для создания файла входных временных диаграмм с расширением scf – Simulator Channel Files;

– задать время завершения моделирования в меню **File | End Time** (Файл | Время завершения), рисунок 6. Если речь идет о микросекундах, то буква μ (микро) с клавиатуры не набирается, а так как она похожа на латинскую букву u, то в этом случае принято использовать именно ее;

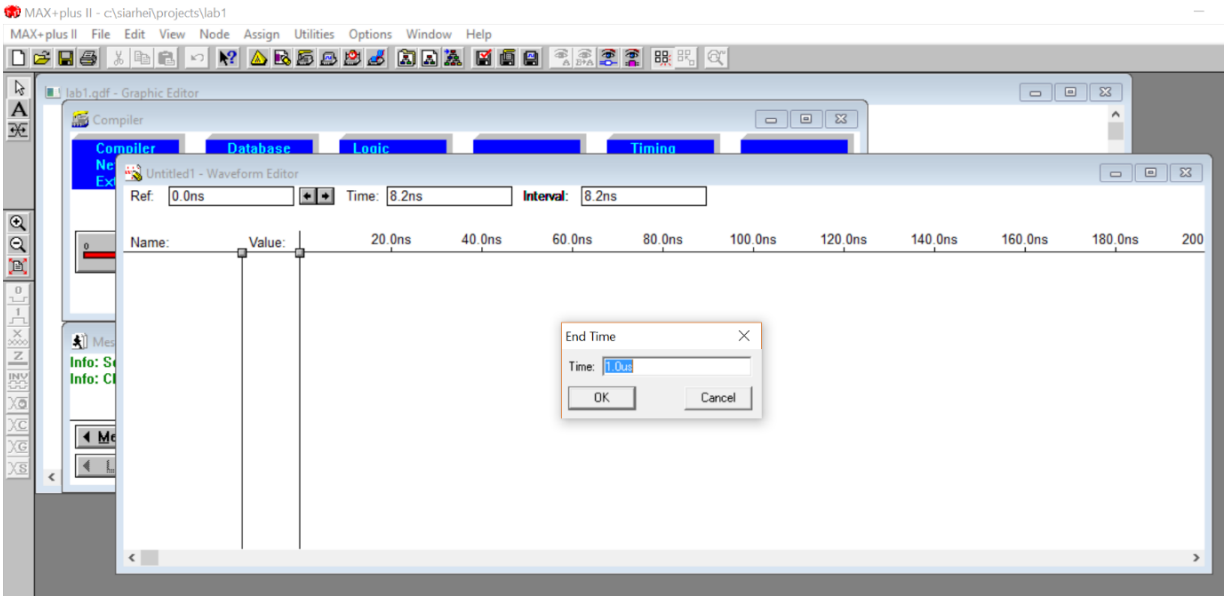

Рисунок 6 – Задание времени окончания симуляции

– выбрать имена входов, на которые должны быть поданы сигналы. Для этого удобнее всего использовать команду **Simulator Netlist File** (**SNF**). С этой целью в окне, открытом командой **Node | Enter Nodes from SNF** (Узел | Ввод узлов), нажать кнопку **List** и выбрать из списка входов (**I**) и выходов (**O**) опцию «available» (имеющиеся в распоряжении), как это показано на рисунке 7. Перенести их в выбранные (опция «selected») путем нажатия стрелки вправо в окне ввода файлов из списка. Закрыть окно выбора файлов из списка. При этом на входах на временных диаграммах по умолчанию будет установлен логический «0», а на выходах «X» – неопределенный уровень сигнала. После закрытия окна **Enter Nodes from SNF** в области **Name** (Имя) редактора Waveform появятся имена выбранных входов и выходов, в области **Value** (Величина) – установленные по умолчанию уровни сигналов;

– задать размер временной сетки для удобства просмотра диаграмм в меню **Options | Grid Size** (Опции | Размер сетки). В данном примере выставляем 10 нс, как это показано рисунке 8;

– задать форму сигнала для выбранного узла. Это можно сделать различными способами:

а) постоянный уровень сигнала задается нажатием соответствующей кнопки (**0** или **1**) слева на панели инструментов приложения при выбранной входной переменной;

б) периодический сигнал задается нажатием клавиши с часами на панели инструментов при предварительно выбранной входной переменной (иконка с красным циферблатом). В открывшемся диалоговом окне устанавливается начальный уровень сигнала и длительность импульса (коэффициент «**Multiplied by**» означает умножение на установленный размер сетки);

в) сигнал произвольной формы задается вручную. Для этого мышью выделяется временный интервал и для него устанавливается необходимый уровень сигнала (**0** или **1**). После выделения определенного интервала иконки в левой части окна становятся активными;

г) сигнал может быть инвертирован с помощью кнопки **INV** на левой панели инструментов;

– задать сигналы на входах в соответствии с рисунком 9. Комбинации сигналов можно выставить либо вручную, выделяя каждый интервал для опреления значения переменной, либо используя опцию периодического сигнала и соответствующий коэффициент «Multiplied by». Сравнить полученные сигналы на входах с рисунком 9;

– сохранить файл с временными диаграммами с именем текущего проекта. Для этого использовать меню **File | Save** или (**CTRL+S**). САПР назначит такое же имя, как и файл графического редактора, но с расширением scf. Одинаковое имя делается для того, чтобы САПР могла знать, к какому проекту относится эта временная диаграмма при симулировании.

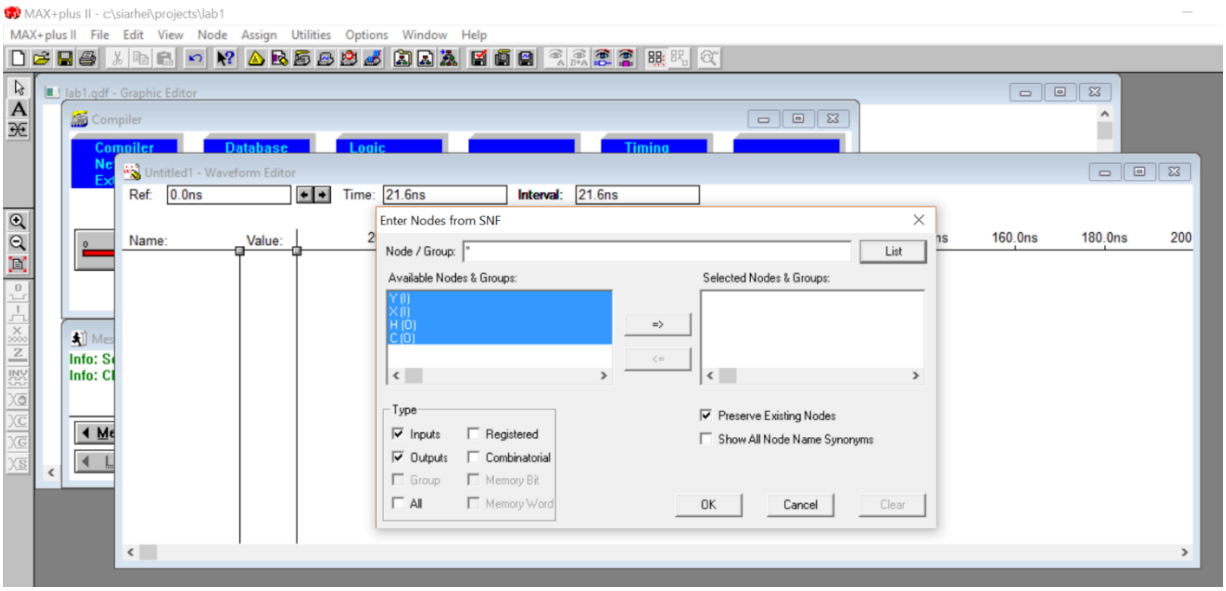

Рисунок 7 – Окно ввода файлов из списка

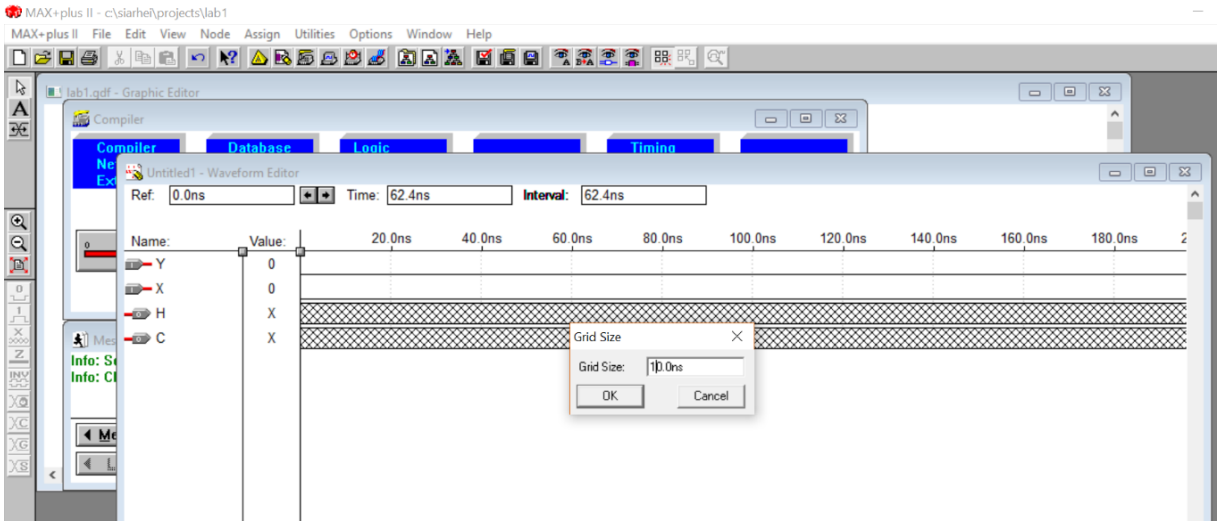

Рисунок 8 – Задание размера временной сетки

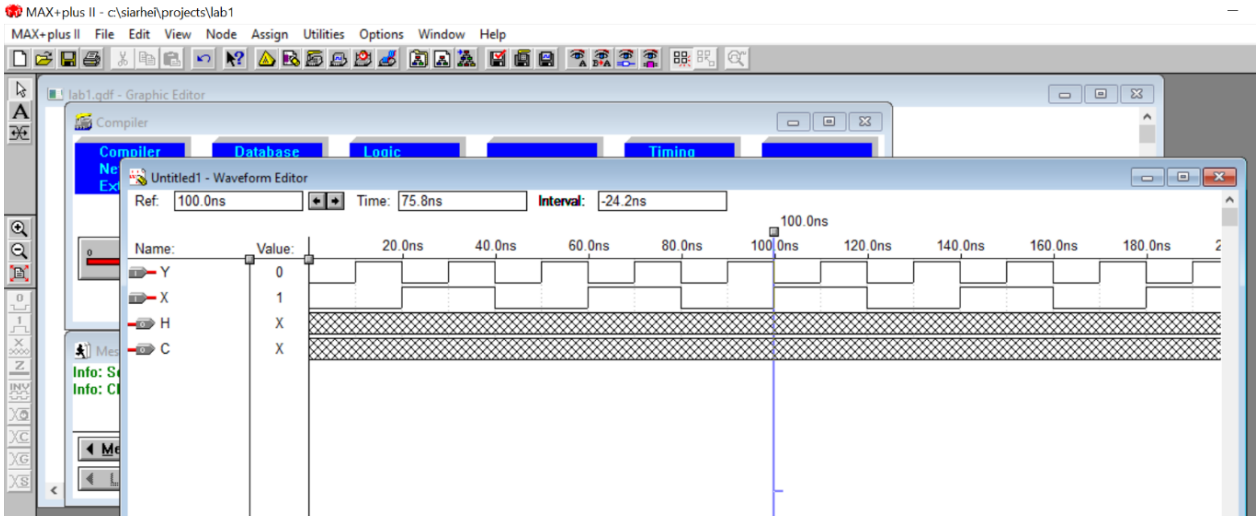

Рисунок 9 – Файл временных диаграмм с назначенными входами

5. Моделирование работы ПЛИС.

Моделирование осуществляется путем выполнения следующих последовательных действий:

– открыть окно моделирования командой **MAX+Plus II | Simulator**. При открытии автоматически создается файл с расширением SNF – Simulator Netlist File;

– нажать кнопку **Start** для начала моделирования. Результатом моделирования будут временные диаграммы, записанные в ранее созданный редактором Wavefrom Editor файл с расширением scf;

– в случае неудачного завершения моделирования список обнаруженных ошибок и сообщений можно посмотреть в окне процессора сообщений **Message Processor**;

– для просмотра полученных временных диаграмм нажать кнопку **Open SCF** (Открыть файл временных диаграмм) в окне моделирования, рисунок 10.

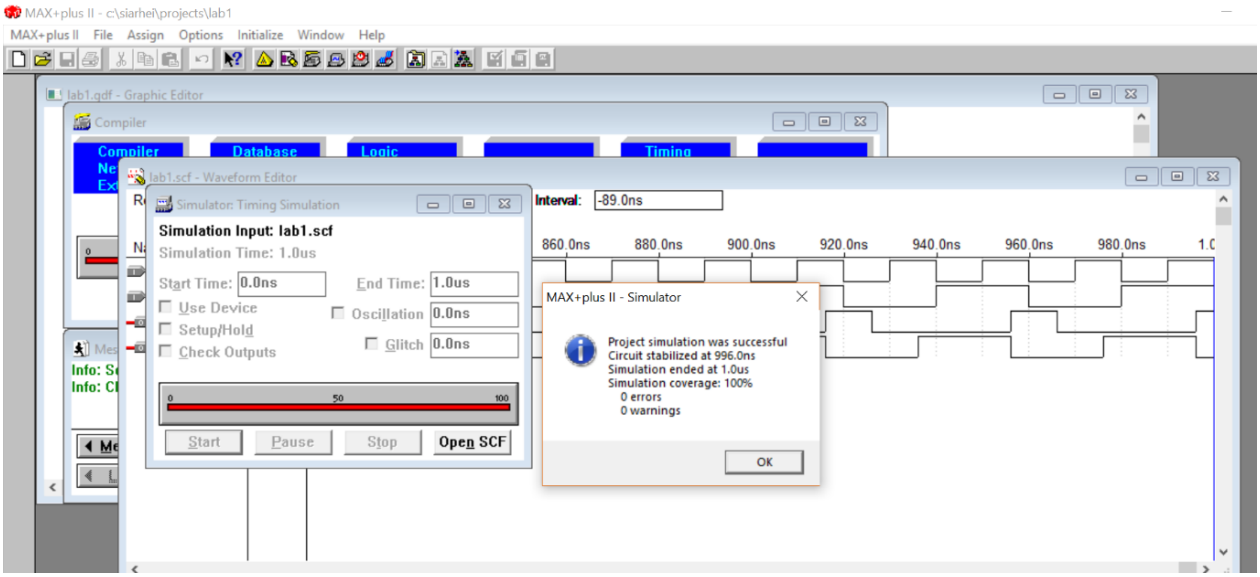

Рисунок 10 – Результат моделирования

Часть 2. Настройка ПЛИС на реализацию функции одноразрядного полного сумматора

После того как ПЛИС настроена на реализацию функции полусумматора приступаем к этапу синтеза одноразрядного полного сумматора из двух полусумматоров. Иными словами, в этой части работы на основе созданных ранее символов полусумматора настроим ПЛИС на реализацию функции одноразрядного полного сумматора.

Следует отметить, что для корректного выполнения этой части работы необходимо закрыть все предыдущие графические файлы и файлы временных диаграмм и осуществить привязку нового проекта к создаваемому файлу. Алгоритм «привязки» проекта описан в первой части работы.

Для настройки ПЛИС на реализацию функции одноразрядного полного сумматора необходимо выполнить следующие действия:

– используем ранее созданный символ *.sym, являющийся полусумматором. На основе двух полусумматоров создадим схему полного сумматора. Функциональная схема приведена на рисунок 11. Для этого на поле графического редактора необходимо поместить два созданных в первой части работы символа и логический вентиль 2ИЛИ (OR2), а также три входа и два выхода;

– на основе собранной схемы полного сумматора (рисунок 11) создать символ полного сумматора. Алгоритм создания символа подробно описан в первой части работы;

– проверить правильность работы устройства по описанному ранее алгоритму, используя временные диаграммы. В данном случае используем стандартный алгоритм функционирования САПР для анализа работы ПЛИС, настроенной на реализацию того или иного функционала. Для этого необходимо открыть редактор временных диаграмм, внести все входные и выходные переменные, сохранить проект, провести симуляцию;

– на основе полученных временных диаграмм проверить правильность работы полного сумматора.

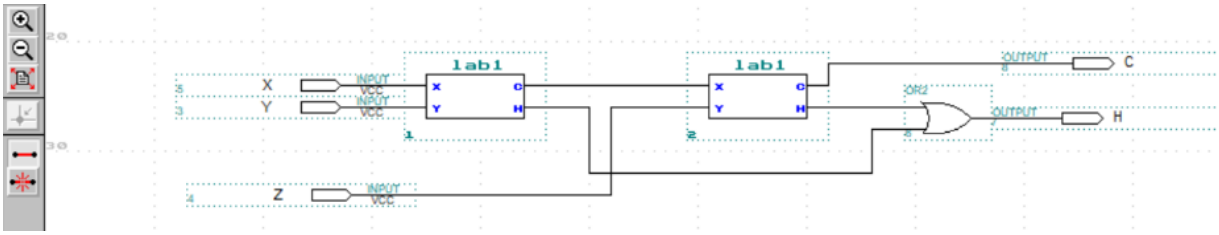

Рисунок 11 – Схема полного сумматора, собранная на основе ранее созданных символов полусумматора

Практическое задание

1. Настроить ПЛИС на реализацию функции полусумматора.

2. Получить временные диаграммы и проанализировать работу полусумматора. Изучить временные параметры поученного полусумматора, оценить его время задержки.

3. Создать символ полусумматора.

4. При построении временных диаграмм в режиме WaveForm Editor использовать период сетки и время окончания симуляции в соответствии с вариантом из таблицы 1.

5. Проанализировать временные параметры ПЛИС.

6. Проанализировать окно с иерархией созданных в процессе выполнения проекта файлов.

7. Проанализировать результат работы уровневого планировщика. Оценить степень загруженности интегральной микросхемы. Установить ножки интегральной микросхемы, задействованные в реализации проекта.

8. Настроить ПЛИС на реализацию функции полного одноразрядного сумматора.

9. Создать символ полного одноразрядного сумматора.

10. Получить временные диаграммы и проанализировать работу полного одноразрядного сумматора.

11. При построении временных диаграмм использовать период сетки и время окончания симуляции в соответствии с вариантом из таблицы 1.

12. Проанализировать временные параметры ПЛИС. Установить время задержки работы ПЛИС.

13. Проанализировать окно с иерархией созданных в процессе выполнения проекта файлов.

14. Проанализировать результат работы уровневого планировщика. Оценить степень загруженности интегральной микросхемы. Установить ножки интегральной микросхемы, задействованные в реализации проекта.

Таблица 1 – Задание к лабораторной работе № 1

Вариант	**Время окончания симуляции, мкс**		**Период сетки, нс**	
	Часть 1	**Часть 2**	**Часть 1**	**Часть 2**
1	2	11	10	50
2	3	10	20	40
3	5	9	25	30
4	8	5	30	20
5	9	3	35	15
6	12	2	50	10

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Скриншоты основных этапов выполнения проекта по каждой части работы.
4. Выводы.

Контрольные вопросы и задания

1. Перечислить основные модули САПР **MAX + Plus II**.
2. Указать путь к библиотеке примитивов, в которой находятся основные простые логические вентили.
3. Что означает зеленый цвет в информационном сообщении компилятора?
4. Что означает синий цвет в информационном сообщении компилятора?
5. Что означает красный цвет в информационном сообщении компилятора?
6. Как создается символ в графическом редакторе?
7. Как назначается интервал сетки в редакторе временных диаграмм?
8. Как назначается время окончания симуляции?

Лабораторная работа № 2
Четырехразрядный двоичный сумматор с последовательным переносом на основе одноразрядного двоичного полусумматора и сумматора

Цель работы:

– приобрести навыки использования ранее созданных в другом проекте символов;

– научиться создавать группы переменных при работе с редактором временных диаграмм;

– приобрести навыки преставления чисел в группе в различных форматах при работе с редактором временных диаграмм;

– настроить ПЛИС на реализацию функции четырехразрядного двоичного сумматора с последовательным переносом;

– проанализировать полученные результаты и убедиться в правильности работы алгоритма.

В лабораторной работе № 1 были созданы символы полусумматора и одноразрядного полного сумматора. Эти символы представлены на рисунке 12. В символ одноразрядного полного двоичного сумматора включен символ полусумматора. Для того чтобы убедиться в том, что символы выбраны правильно, их содержимое можно «раскрыть», дважды кликнув левой кнопкой мыши по символу. Раскроется схема в графическом редакторе, на основании которой был создан данный символ.

а *б*

Рисунок 12 – Созданные символы полусумматора (*а*) и полного сумматора (*б*)

На основании полученных в лабораторной работе № 1 результатов создадим схему четырехразрядного сумматора с последовательным переносом. Функциональная схема приведена на рисунке 13. Рекомендуется использовать следующие буквенные и цифровые обозначения для описания входов и выходов сумматора:

A1 – младший (0-й) разряд первого двоичного числа;
A2 – 1-й разряд первого двоичного числа;
A3 – 2-й разряд первого двоичного числа;
A4 – старший (3-й) разряд первого двоичного числа;
B1 – младший (0-й) разряд второго двоичного числа;
B2 – 1-й разряд второго двоичного числа;
B3 – 2-й разряд второго двоичного числа;

B4 – старший (3-й) разряд второго двоичного числа;
S1 – младший (0-й) разряд полученного двоичного числа;
S2 – 1-й разряд полученного двоичного числа;
S3 – 2-й разряд полученного двоичного числа;
S4 – старший (3-й) разряд полученного двоичного числа;
5 – перенос в старший разряд.

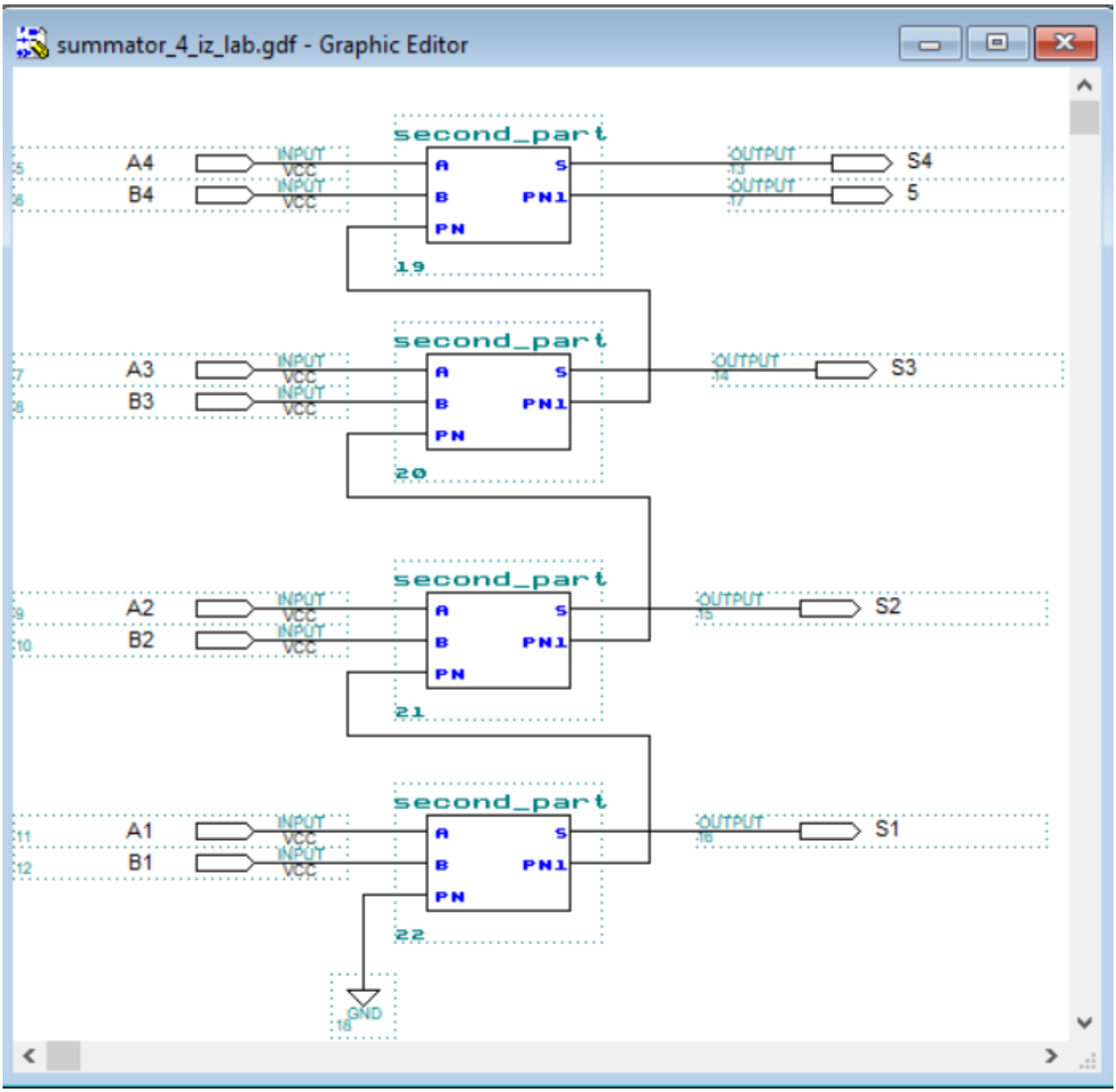

Рисунок 13 – Схема четырехразрядного сумматора на основе одноразрядных сумматоров

На стадии построения временных диаграмм удобно установить число B4B3B2B1 равным 1111, а числа A4A3A2A1 изменять последовательно от 0000 до 1111. При этом временную сетку можно установить равной, например, 50 нс, а время симуляции – 1 мкс. Этот результат показан на рисунке 14.

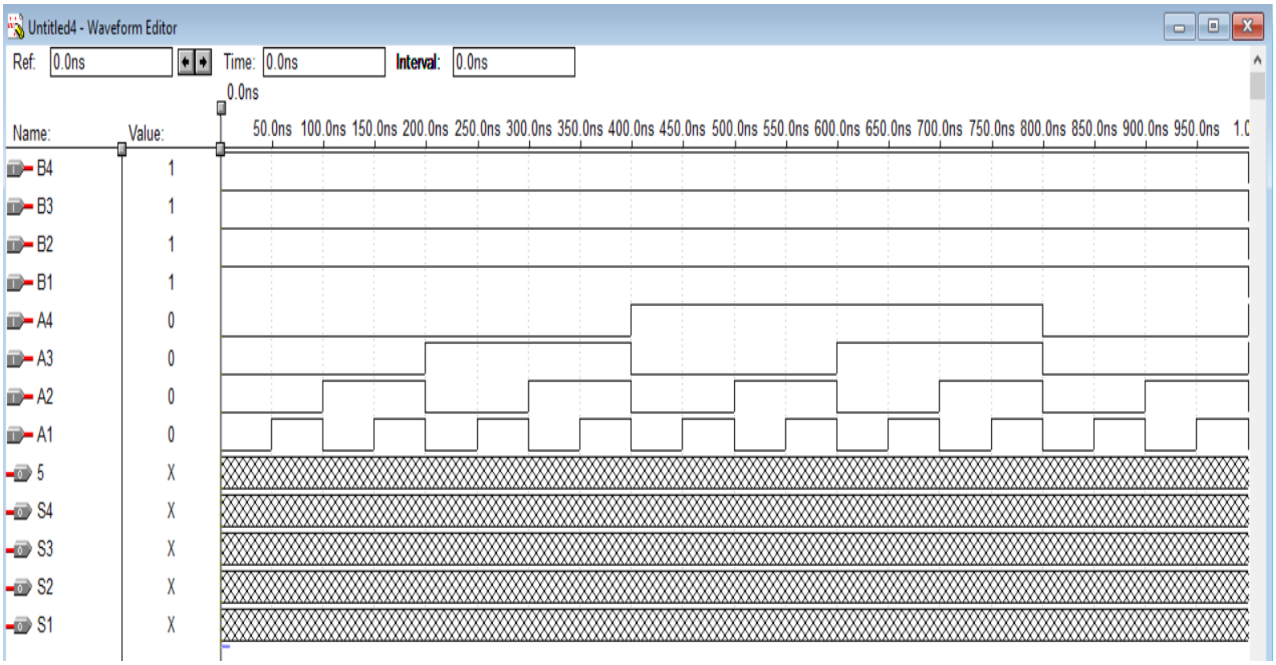

Рисунок 14 – Пример выставления нужных комбинаций чисел B4B3B2B1 и A4A3A2A1 для суммирования

После проведения симуляции получаем временные диаграммы, приведенные на рисунке 15.

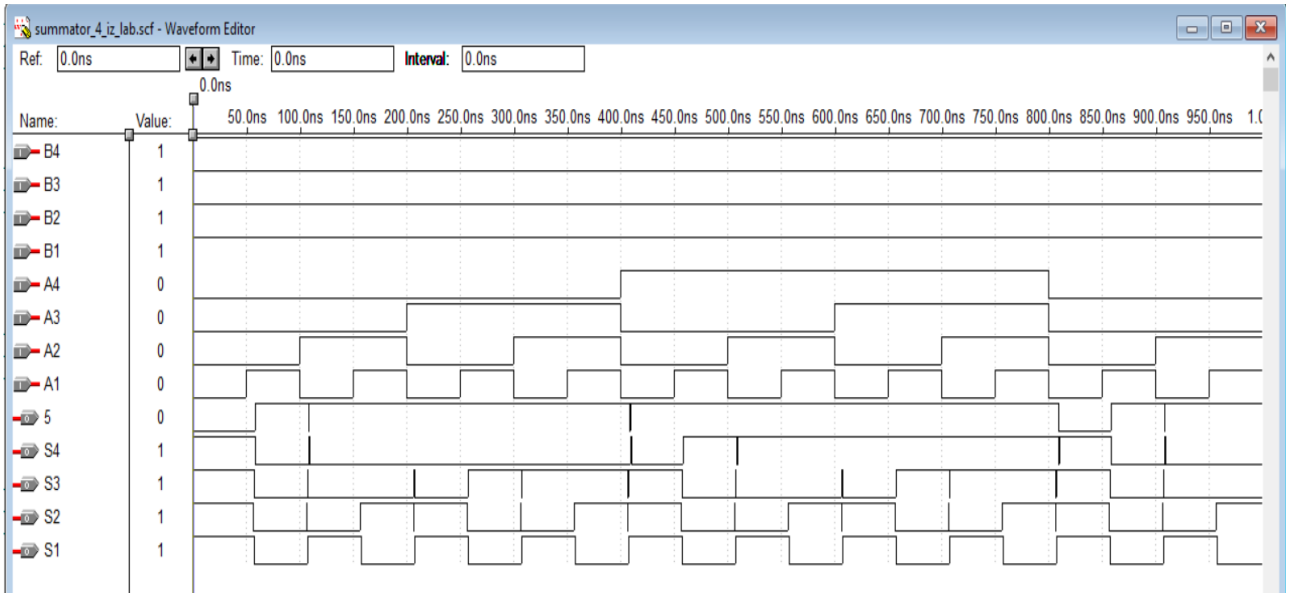

Рисунок 15 – Пример полученных временных диаграмм полного сумматора с последовательным переносом после симуляции

В данном случае для расстановки переменных группы А можно воспользоваться инструментом **MAX+PLUS II**, позволяющим быстро задать последовательно изменяющиеся наборы. Набор данных при этом представляется в виде группы (шины).

Сначала сгруппируем входные цепи A4...A1, для чего их все необходимо выделить (кликнуть мышью на название A4, затем, удерживая кнопку **Shift**,

кликнуть на название цепи A0) и в меню **Node** выбрать **Enter Group**. Этот же пункт меню доступен и из контекстного меню при нажатии правой кнопки мыши. Номера наборов удобно воспринимать в десятичной форме, поэтому выберем **DEC** (как показано на рисунке16) и нажмем кнопку **ОК**. После такого действия на временной диаграмме пропадут четыре входных сигнала A4...A1, а вместо них появится группа (шина) с названием A[3..0]. Похожий пример показан в окне редактора временных диаграмм на рисунке 17.

Рисунок 16 – Выбор десятичной формы представления чисел в группе

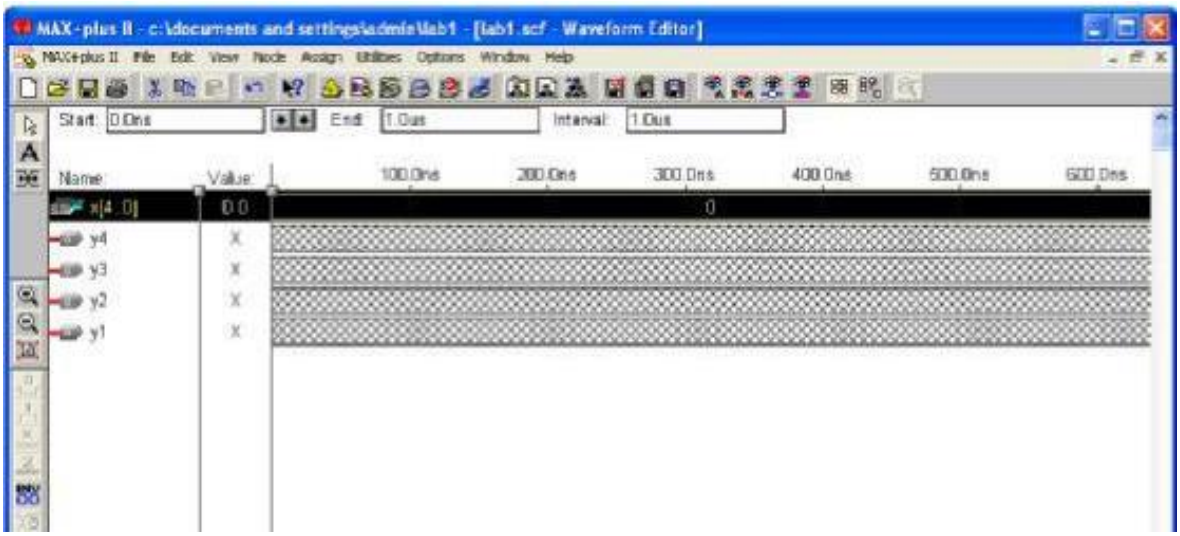

Рисунок 17 – Группа чисел вместо отдельных диаграмм

Воспользуемся инструментом, позволяющим задавать последовательные значения для группы сигналов. Для этого кликнем на названии группы A[3..0], для того чтобы ее выделить, и нажмем на кнопку XC или меню **Edit → Overwrite → Count value**. В результате появится диалоговое окно, показанное на рисунке 18, позволяющее изменять входные сигналы по закону счётчика. В данном диалоговом окне указано, что счет начинается с 0 (**Starting Value**), счет ведется в двоичном коде (**Binary**), увеличивается на 1 (**Increment By**) каждый такт (**Multiplied By**).

В случае необходимости можно разгруппировать группу входных переменных, т. е. заново представить их в виде независимых друг от друга данных.

Для этого необходимо кликнуть на ее название A[3..0] мышью, затем в меню **Node** нужно выбрать опцию **Ungroup**. Порядок расположения цепей на временной диаграмме нетрудно изменить, для этого нужно нажать левой кнопкой мыши на значок цепи и, удерживая кнопку мыши, перетащить цепь на новое место.

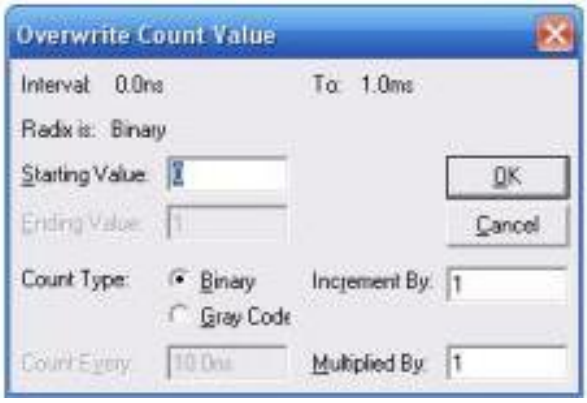

Рисунок 18 – Окно настройки сигналов в группе

После нажатия кнопки OK на временной диаграмме появятся последовательно сменяющиеся номера наборов от 0 до 15. Этот случай продемонстрирован на рисунке 19. Для изменения масштаба отображения, так же как и в графическом редакторе, рекомендуется использовать комбинацию кнопок **Ctrl+Пробел** и **Ctrl+Shift+Пробел** или кнопки с изображением лупы в левой части панели инструментов.

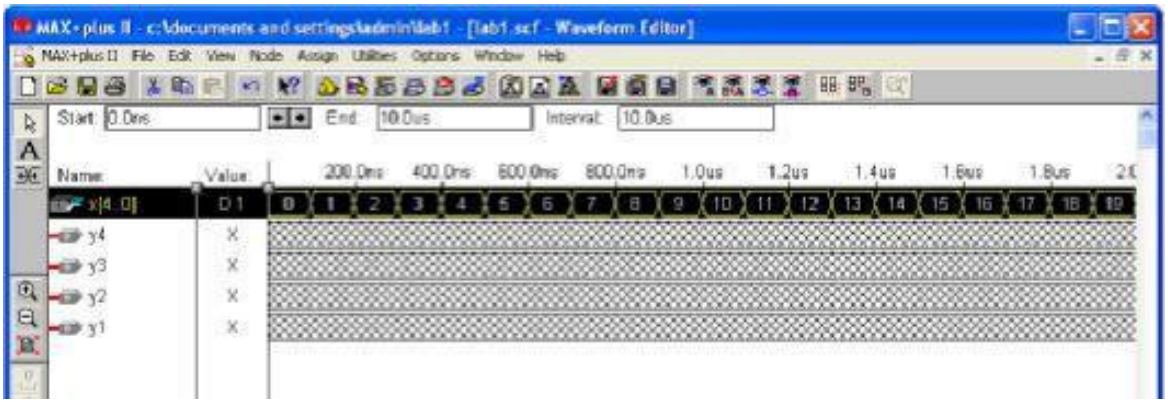

Рисунок 19 – Последовательно сменяющиеся номера наборов в группе

После успешного проведения симуляции (результат симуляции здесь не показан) необходимо проанализировать временные параметры работы ПЛИС при реализации функции полного сумматора. Для этого используем встроенную в САПР функцию временного анализа. Через функцию **Timing Analyzer** (**Временной анализ**) меню **MAX + Plus II** строим матрицу задержек, с помощью которой определяем время задержки появления сигнала на выходе относительно изменения каждого входного сигнала. Пример построенной в САПР матрицы задержек показан на рисунке 20.

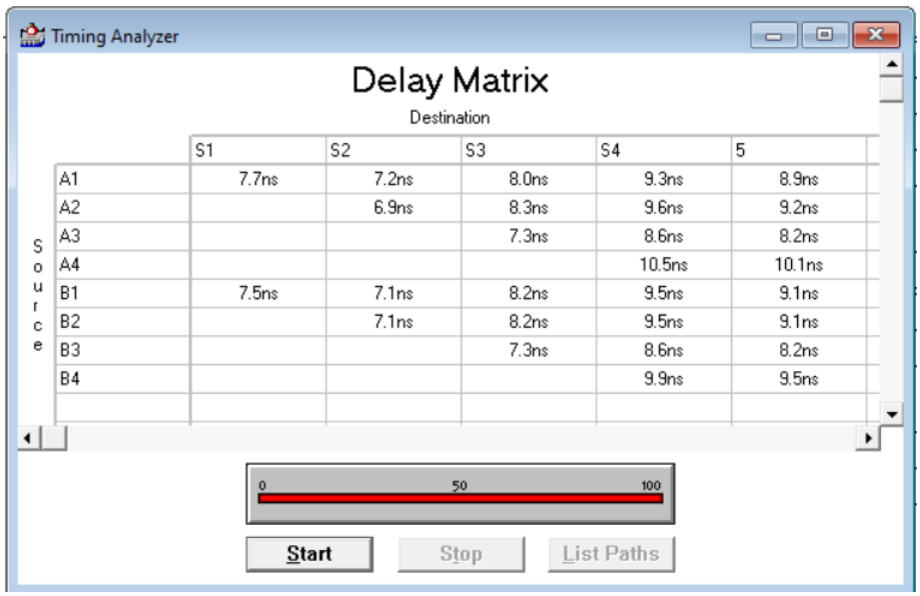

Рисунок 20 – Результат временного анализа

В процессе выполнения проекта часть файлов создает сам пользователь, а часть вспомогательных файлов создается самой системой автоматизированного проектирования. Все эти файлы можно посмотреть с помощью специальной опции. А именно с помощью функции **Hierarchy Display** (**Иерархический дисплей**) меню **MAX + Plus II** выводим иерархическое дерево всех файлов, относящихся к данному проекту. Иерархическое дерево данного проекта показано на рисунке 21.

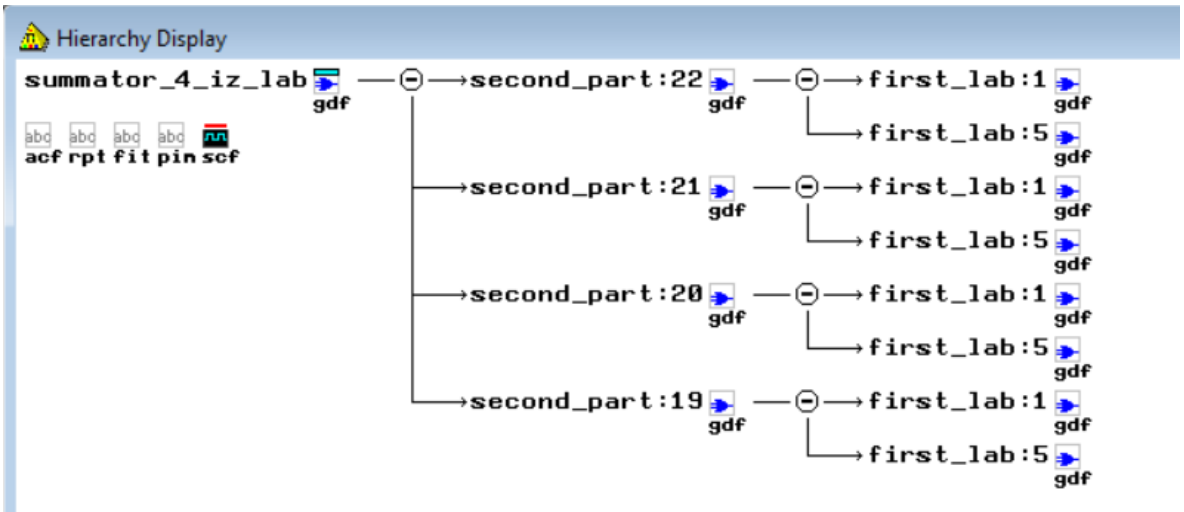

Рисунок 21 – Окно с иерархией файлов

Наконец, с помощью функции **Floorplan Editor** (**Уровневый планировщик**) меню **MAX + PLUS II** изучаем, насколько «загружена» ПЛИС выполняемым проектом. Пример загруженности ПЛИС для выполнения рассматриваемой в данной работе функции полного четырехразрядного сумматора приведен на рисунке 22. Ячейки микросхемы, задействованные в реализации необходимого

алгоритма, обозначены малиновым и желтым цветом. Из рисунка 22 следует, что возможности конкретной ПЛИС использованы в небольшой степени. Уровневый планировщик позволяет также выявить ножки интегральной микросхемы, на которые подаются переменные и снимаются результирующие логические функции.

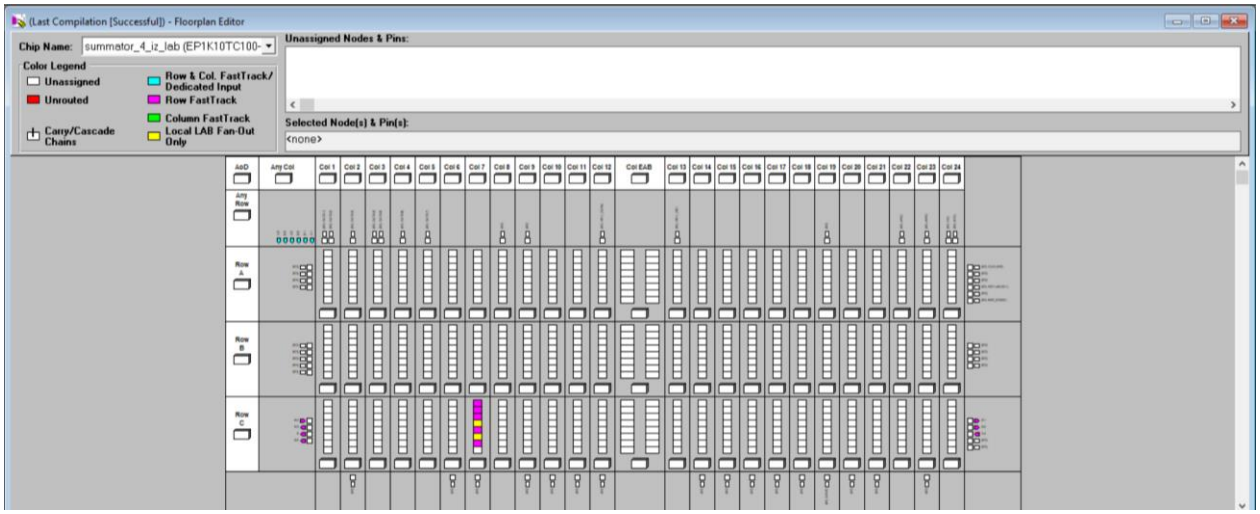

Рисунок 22 – Уровневый планировщик

Практическое задание

1. Настроить ПЛИС на реализацию функции полного сумматора с последовательным переносом на четыре разряда. С этой целью использовать предварительно созданные в лабораторной работе № 1 символы полусумматоров и одноразрядных сумматоров.

2. При построении временных диаграмм использовать время окончания симуляции и временную сетку в соответствии с вариантом из таблицы 2.

3. Данные для числа A4A3A2A1 ввести как с помощью отдельных битов, так и с помощью группировки битов. Ввести последовательно все 16 возможных комбинаций числа, используя для этого различные приемы. Значения должны последовательно изменяться от 0 до 15_{10}.

4. Число B4B3B2B1 выбрать постоянным в соответствии с вариантом из таблицы 2.

5. Проанализировать временные параметры ПЛИС из полученных временных диаграмм и с помощью специальных модулей САПР MAX + PLUS II.

6. Проанализировать окно с иерархией созданных в процессе выполнения проекта файлов.

7. Проанализировать результат работы уровневого планировщика. Оценить степень загруженности интегральной микросхемы.

Таблица 2 – Задание к лабораторной работе № 2

Вариант	Время окончания симуляции, мкс	Временная сетка, нс	B4B3B2B1
1	2	12	1101
2	3	16	1011
3	5	22	1111
4	8	28	1001
5	9	33	1010
6	12	39	1110

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Скриншоты основных этапов выполнения проекта по каждой части работы.
4. Анализ временных параметров схемы. Оценка быстродействия работы схемы.
5. Выводы.

Контрольные вопросы и задания

1. Перечислить основные модули САПР **MAX + PLUS II**.
2. Указать путь к библиотеке примитивов, в которой находятся основные простые логические вентили.
3. Что означает зеленый цвет в информационном сообщении компилятора?
4. Что означает синий цвет в информационном сообщении компилятора?
5. Что означает красный цвет в информационном сообщении компилятора?
6. Как выбрать нужное семейство интегральных программируемых микросхем для реализации проекта?
7. Как в рамках заданного семейства микросхем выбрать нужную микросхему для реализации проекта?
8. Как создается символ в графическом редакторе? Как узнать «содержимое» символа?
9. Как осуществляется группировка переменных при построении временных диаграмм?
10. Как осуществляется разгруппировка переменных при построении временных диаграмм?
11. Как назначается время окончания симуляции? Как выставляется время окончания симуляции в микросекундах?

Лабораторная работа № 3
Настройка ПЛИС на реализацию функции D-триггера, построенного на элементах И-НЕ

Основные положения работы D-триггера

D-триггер – в общем случае это триггер с двумя входами. Входными данными являются вход данных (D) и тактовый вход (C). Тактовый импульс представляет собой синхронизирующий импульс, генерируемый оборудованием для управления операциями. D-триггер используется для хранения данных в заранее определенное время и удержания их до тех пор, пока они не потребуются. Эту схему иногда называют триггером задержки. Другими словами, входные данные задерживаются до одного тактового импульса, прежде чем они будут поданы на выход. D-триггер может быть построен с использованием либо вентиля И-НЕ (NAND) либо вентиля ИЛИ-НЕ (NOR).

Условное обозначение D-триггера, срабатывающего по положительному фронту тактового сигнала (D-триггер с динамическим управлением), приведено на рисуке 23.

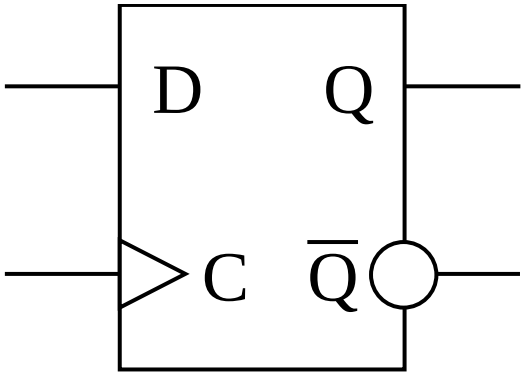

Рисунок 23 – Условное обозначение D-триггера

Таблица истинности, описывающая логику работы D-триггера с динамическим управлением, условный символ которого представлен на рисунке 23, приведена ниже.

Таблица 3 – Таблица истинности D-триггера

Фронт сигнала C	D	Q	$\overline{Q}$	Описание
1→0	X	Q	$\overline{Q}$	Состояние не изменяется
0→1	0	0	1	Q = 0
0→1	1	1	0	Q = 1

В D-триггере вход «D» называется входом «Данные». Когда вход данных установлен на 1, выходное состояние триггера будет установлено в Q = 1, а когда он установлен на 0, состояние триггера будет сброшено в Q = 0. Следует отметить, что подобная логика работы не имела бы смысла, поскольку выход триггера всегда будет меняться при каждом импульсе, подаваемом на вход данных. Вход

«CLOCK» или «C» используется, чтобы избежать этого для изоляции ввода данных от схемы фиксации триггера. Когда для тактового входа установлен нужный фронт, значение входа D копируется только на выход Q. Это и формирует основу функционирования последовательного устройства, называемого D-триггером.

Когда тактовый вход не имеет активного фронта, D-триггер не изменяет своего состояния независимо от состояния входа D. Это режим хранения данных. Выходы триггера будут изменяться только при наличие активного фронта тактового сигнала C. На рисунке 24 показана временная диаграмма, описывающая работу D-триггера. Переключение происходит только по положительному фронту тактового сигнала.

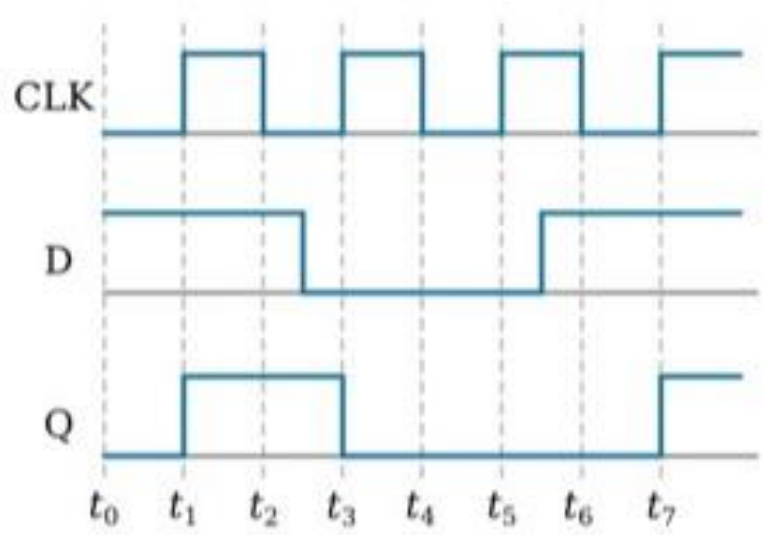

Рисунок 24 – Временная диаграмма работы D-триггера, представленного на рисунке 23

Как уже указывалось выше, D-триггер можно построить на элементах И-НЕ (NAND). На рисунке 25 приведена схема построения D-триггера на элементах И-НЕ.

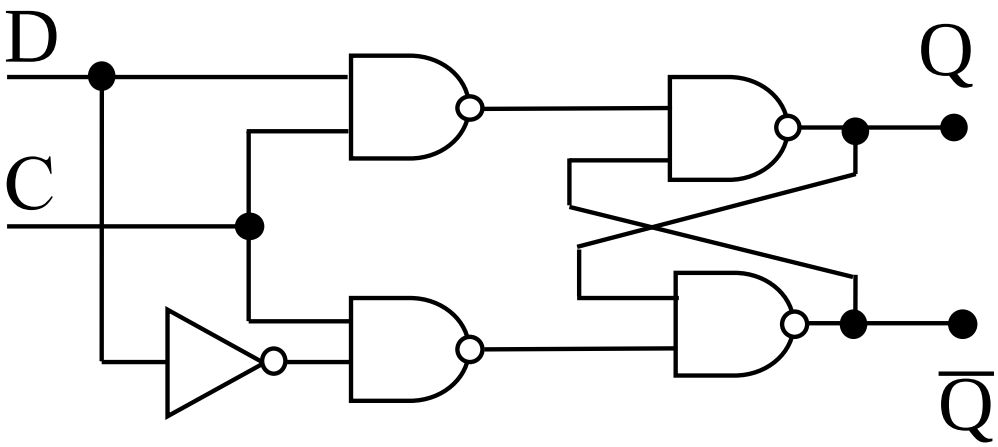

Рисунок 25 – Схема D-триггера на элементах И-НЕ

Реальные микрохемы D-триггера имеют два дополнительных входа: Preset/Set и Reset/Clear. Это асинхронные входы синхронного триггера. Подача нужного сигнала на вход Preset/Set принудительно устанавливает триггер в состояние Q = 1, а подача активного сигнала на вход Reset/Clear принудительно устанавливает триггер в состояние Q = 0. Условное обозначение такого D-триггера приведено на рисунке 26.

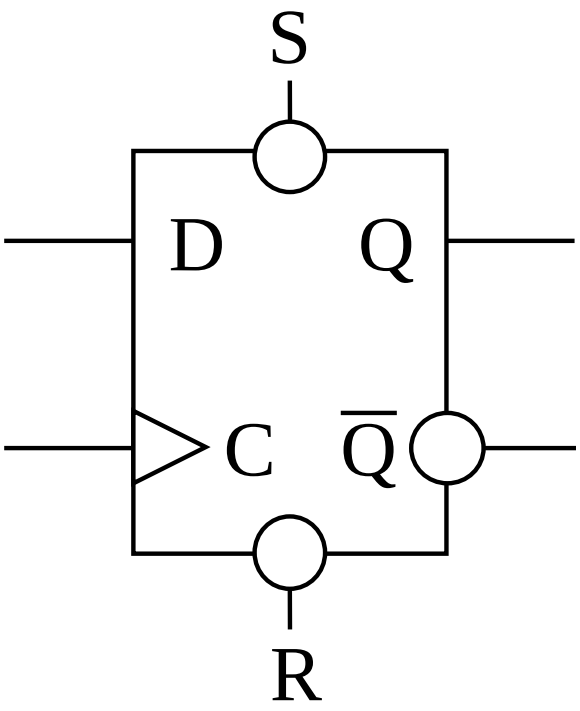

Рисунок 26 – Условное обозначение D-триггера с асинхронными входами принудительной установки триггера в состояние Q = 1 и Q = 0

Алгоритм работы, описывающий функционирование D-триггера с асинхронными входами, представлен в таблице 4.

Таблица 4 – Алгоритм работы D-триггера с асинхронными входами

S (активный 0)	**R (активный 0)**	**C**	**D**	**Q**
0	1	X	X	1
1	0	X	X	0
0	0	X	X	Не определен
1	1	0→1	1	1
1	1	0→1	0	0
1	1	0→1	X	Без изменений

Цель работы

1. Ознакомиться с функционалом работы синхронного D-триггера с асинхронными входами.

2. В графическом редакторе собрать предложенную схему D-триггера на элементах И-НЕ (NAND).

3. С помощью временных диаграмм исследовать работу D-триггера. Изучить влияние входов на выходной сигнал Q.

4. Исследовать временные параметры D-триггера.

Выполнение работы

В лабораторной работе ПЛИС настраивается на реализацию функции синхронного D-триггера с асинхронными входами. При этом триггер создается в графическом редакторе на основе элементов И-НЕ (NAND).

Для создания D-триггера на элементах И-НЕ на рабочее поле в графическом редакторе необходимо поместить следующие элементы из библиотеки примитивов (\prim):

– логический элемент 3И-НЕ (NAND3) – 6 элементов;
– вход (INPUT) – 4 элемента;
– выход (OUTPUT) – 2 элемента;

Соединить элементы в схему согласно рисунку 27.

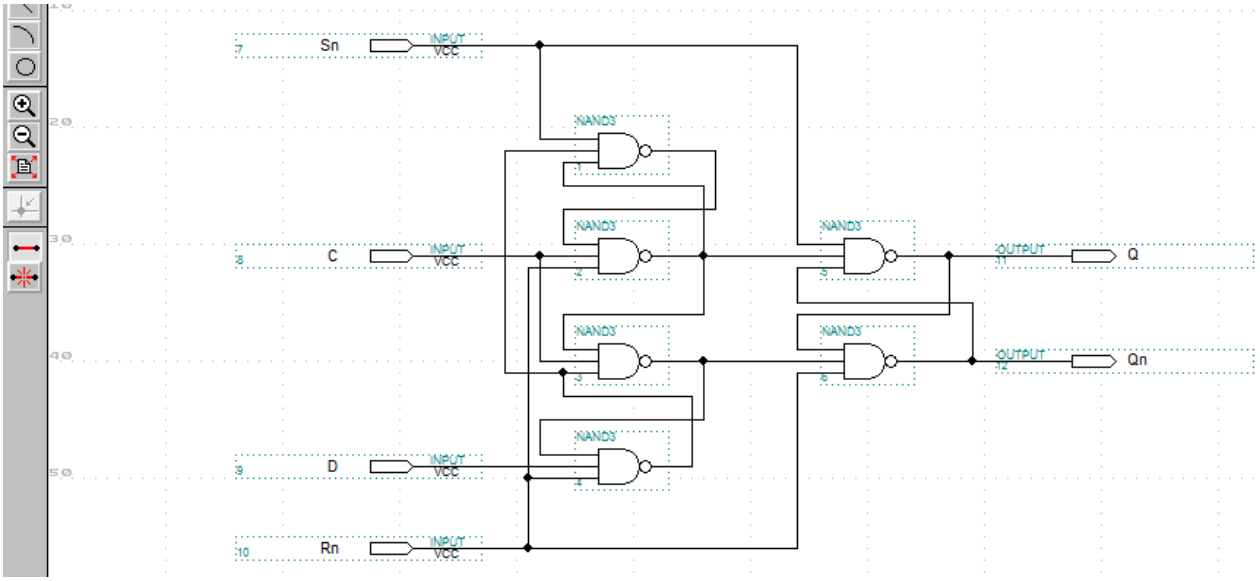

Рисунок 27 – Схема D-триггера в графическом редакторе

Схема содержит вход данных **D**, вход для тактовой частоты **C**, два асинхронных входа: сброс **Rn** и установка **Sn**, два выхода: прямой **Q** и инверсный **Qn**. Асинхронные входы принудительной установки триггера в состояния 0 (**Rn**) и 1 (**Sn**) имеют приоритет перед входами **D** и **C**.

После успешной компиляции проекта перейти к построению временных диаграмм. При этом задать временную сетку, период тактового сигнала **C** и ко-

нечное время симуляции в соответствии с вариантом из таблицы 5. Схема на рисунке 27 составлена таким образом, что логическая 1 на входах **Rn** и **Sn** делает их неактивными для D-триггера. Логический 0 на входе **Rn** принудительно переведет триггер в состояние Q = 0 независимо от входных сигналов. Логический 0 на входе **Sn** принудительно переведет D-триггер в состояние Q = 1 независимо от входных сигналов.

Для проверки всех возможных комбинаций входных сигналов на состояние триггера Q следует расставить входные переменные таким образом, чтобы проверить действие всех входных сигналов на выход Q триггера. Пример одного из вариантов расстановки сигналов вместе с результатом симуляции показан на рисунке 28.

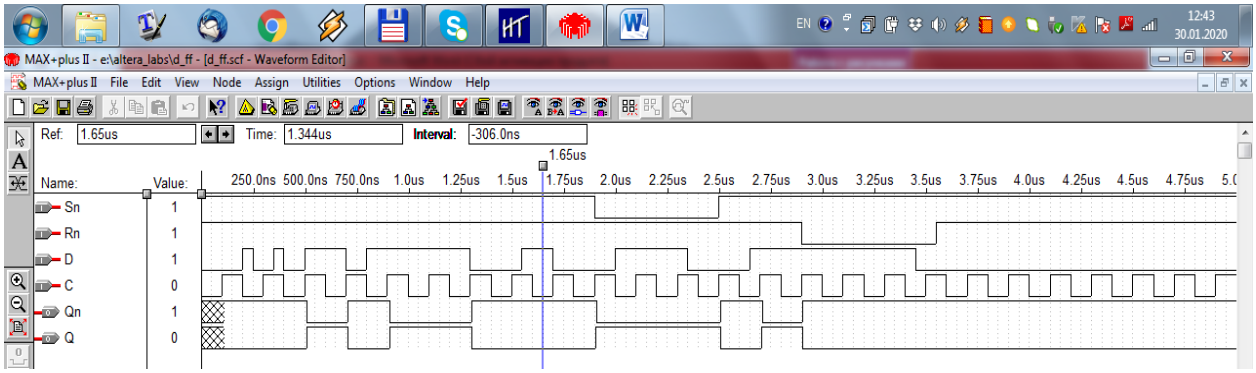

Рисунок 28 – Временная диаграмма, описывающая работу D-триггера

Обратить внимание на то, что выходной сигнал **Q** изменяется только по положительному фронту тактового сигнала **C**.

Практическое задание

1. Настроить ПЛИС на реализацию функции D-триггера на элементах И-НЕ.
2. При построении временных диаграмм использовать временную сетку, период тактового сигнала **C** и конечное время симуляции в соответствии с вариантом из таблицы 6.
3. Проанализировать полученные временные диаграммы и убедиться, что ПЛИС настроена на реализацию функции D-триггера.
4. По полученным временным диаграммам объяснить логику работы D-триггера.

Таблица 5 – Задание к лабораторной работе № 3

Вариант	**Временная сетка, нс**	**Время окончания симуляции, мкс**	**Период тактового сигнала, нс**
1	**2**	**3**	**4**
1	40	2	60
2	40	3	80

Продолжение таблицы 5

1	2	3	4
2	50	3	100
3	50	4	200
4	80	5	160
5	80	10	320
6	120	12	240

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Скриншоты основных этапов выполнения проекта.
4. Выводы.

Контрольные вопросы и задания

1. Описать работу D-триггера.
2. Указать путь к библиотеке примитивов, в которой находятся основные простые логические вентили.
3. Дать характеристику созданного в проекте D-триггера.
4. Что такое положительный фронт тактового импульса?
5. Как выбрать нужное семейство микросхем для реализации проекта?
6. Как в рамках заданного семейства микросхем выбрать нужную микросхему для реализации проекта?
7. Как назначается время окончания симуляции?
8. На какой ИМС реализован ваш проект?
9. Чем характеризуются асинхронные входы синхронного триггера?

Лабораторная работа № 4
Настройка ПЛИС на реализацию функции генератора квазислучайной последовательности чисел

Общие положения построения генератора квазислучайной последовательности чисел

Генератор квазислучайной последовательности чисел или псевдослучайных чисел (pseudo-random number generator – PRNG) – это, как правило, программа, написанная для приложений теории вероятностей и статистики и используемая в них, когда требуется большое количество случайных цифр. Большинство этих программ выдают бесконечные строки однозначных чисел, обычно с основанием 10, известным как десятичная система. Когда берутся большие выборки псевдослучайных чисел, каждая из 10 цифр в наборе {0,1,2,3,4,5,6,7,8,9} встречается с одинаковой частотой, даже если они неравномерно распределены в последовательности.

Многие алгоритмы были разработаны в попытке создать действительно случайные последовательности чисел, бесконечные цепочки цифр, в которых теоретически невозможно предсказать следующую цифру в последовательности на основе цифр до заданной точки. Но само существование алгоритма, каким бы сложным он ни был, означает, что следующую цифру можно предсказать. Это привело к появлению термина «псевдослучайный» для таких машинно сгенерированных строк цифр. Они эквивалентны последовательностям случайных чисел для большинства приложений, но они не являются по-настоящему случайными в соответствии со строгим определением.

Важность случайных чисел заключается не в самих числах (они являются обычными числами, если брать их по отдельности), а в способе их генерации. Современные технологии основаны на этих цифрах: протоколы связи, криптография, игры в значительной степени используют их, и от них зависит их безопасность и непредсказуемость.

Следует подчеркнуть, что случайные числа используются человечеством на протяжении тысячелетий, так что эта концепция не нова. Начиная с лотереи в древнем Вавилоне, заканчивая столами для игры в рулетку в Монте-Карло и играми в кости в Лас-Вегасе, цель состоит в том, чтобы оставить конечный результат на волю случая. Но если отбросить азартные игры, случайность имеет множество применений в науке, статистике, криптографии и многом другом. Однако использование кубиков, монет или подобных носителей в качестве случайного устройства имеет свои ограничения. Из-за механической природы этих методов генерация большого количества случайных чисел требует большого количества времени и работы. В настоящее время в распоряжении пользователей имеются более мощные инструменты и методы.

Рассмотрим два основных метода, используемых для генерации случайных чисел. Первый метод основан на физическом процессе и извлекает источник случайности из некоторого физического явления, которое, как ожидается, будет случайным. Такое явление происходит за пределами компьютера. Соответствующее

физическое явление или процесс измеряется и корректируется с учетом возможных отклонений, вызванных процессом измерения. В качестве характерных примеров можно привести радиоактивный распад, фотоэлектрический эффект, космическое фоновое излучение, атмосферный шум и многое другое. Случайные числа, сгенерированные на основе такой случайности, называются «истинными» случайными числами. Технически аппаратная часть для регистрации «истинно» случайных чисел состоит из устройства, которое преобразует энергию из одной формы в другую (например, излучение в электрический сигнал), усилителя и аналого-цифрового преобразователя для преобразования выходного сигнала в цифровое число.

В качестве альтернативы «истинным» случайным числам второй метод генерации случайных чисел включает вычислительные алгоритмы, которые могут выдавать явно случайные результаты. Процесс является случайным (точнее, квази- или псевдослучайным), потому что полученные конечные результаты фактически полностью определяются начальным значением, также известным как начальное значение или ключ. Следовательно, если бы было известно значение ключа и то, как работает алгоритм, то тогда можно было бы воспроизвести эти, казалось бы, случайные результаты.

Именно поэтому генераторы случайных чисел этого типа часто называют генераторами квазислучайных чисел или псевдослучайных чисел и, как следствие, они выводят квазислучайные (псевдослучайные) числа. Несмотря на то что этот тип генератора обычно не собирает никаких данных из источников естественной случайности, его реализация в настоящее время не представляет особых трудностей, а алгоритмы хорошо известны.

Сравним некоторые аспекты генераторов истинных случайных чисел или TRNG – true-random number generator и генераторов псевдослучайных чисел или PRNG.

PRNG быстрее, чем TRNG. Из-за их детерминированной природы они полезны, когда необходимо воспроизвести последовательность случайных событий. Это очень помогает, например, при тестировании кода. С другой стороны, TRNG не являются периодическими и лучше работают в случаях, чувствительных к безопасности, таких как шифрование.

Одним из важных параметров генератора квазислучайной последовательности чисел является его период. Период в данном случае – это количество итераций, через которые проходит PRNG, прежде чем он начнет повторяться. Таким образом, при прочих равных условиях для прогнозирования и взлома PRNG с более длительным периодом потребуется больше компьютерных ресурсов.

Приведем пример построения алгоритма для генератора псевдослучайных чисел. Компьютер выполняет код, основанный на наборе правил, которым необходимо следовать. Для PRNG в целом эти правила относятся к следующему:

– примите некоторый начальный входной номер, который является начальным значением или ключом;

– примените это начальное значение в последовательности математических операций для получения результата. Этот результат и есть случайное число;

– используйте это результирующее случайное число в качестве начального для следующей итерации;

– повторите процесс, чтобы имитировать случайность.

Схемотехнически наиболее простая схема генератора квазислучайной последовательности чисел состоит из регистра сдвига с обратной связью. При этом в цепь обратной связи встраивают элемент И2ЛИ (XOR) для максимизации случайности.

Цель работы

1. Ознакомиться с понятием «истинный» и «квазислучайный» («псевдослучайный») генератор чисел.

2. Ознакомиться с принципами построения генератора квазислучайной последовательности чисел.

3. Реализовать схему генератора квазислучайной последовательности чисел.

4. Исследовать работу полученного генератора квазислучайной последовательности чисел, оценить его временные параметры.

Порядок выполнения работы

В данной лабораторной работе для реализации функции генератора квазислучайной последовательности чисел используем сложную ПЛИС (СПЛИС) семейства MAX7000. Программирование ПЛИС производится с использованием САПР **MAX+PLUS II**.

Построение схемы производится в графическом редакторе (подменю **Grafic Editor** меню **MAX+plus II**, либо иконка **New** на панели инструментов), расширение файла *.gdf. Окно редактора имеет ряд дополнительных пунктов основного меню и панель инструментов редактора, расположенную вертикально с левой стороны окна.

Для добавления символа использовать диалоговое окно **Enter Symbol** (рисунок 29) из меню **Symbol** либо меню, вызываемое правой кнопкой мыши, либо команда Double-Click. Выбрать и установить необходимый элемент можно двумя способами:

1. Набрать имя элемента (примитива, мега- или макрофункции) в окне **Symbol Name** диалогового окна **Enter Symbol** и нажать ОК.

2. Выбрать необходимую библиотеку в окне **Symbol Libraries** диалогового окна **Enter Symbol** и двойным щелчком левой клавиши мыши открыть ее. Затем аналогичным образом выбрать необходимый элемент в окне **Symbol File**.

Для построения структурной схемы генератора квазислучайной последовательности чисел, представленной на рисунке 30, необходимы следующие элементы из библиотеки примитивов (\prim):

– D-триггер (DFF) – 8 элементов (FF – Flip Flop);

– вход (INPUT) – 1 элемент;

– выход (OUTPUT) – 8 элементов;
– логический элемент НЕ (NOT) – 8 элементов;
– логический элемент 2И (AND2) – 15 элементов;
– логический элемент 2ИЛИ (OR2) – 8 элементов.

Для соединения функциональных элементов использовать панель инструментов, расположенную в левой части главного окна.

Далее необходимо сохранить новый файл проекта (через меню **File | Save**) под именем LAB3 (расширение gdf будет присвоено автоматически). Имя файла проекта следует *обязательно* привязать к имени проекта – это делается при выборе пункта **Set Project to Current File** в подменю **Project** меню **File** главного меню рабочего окна.

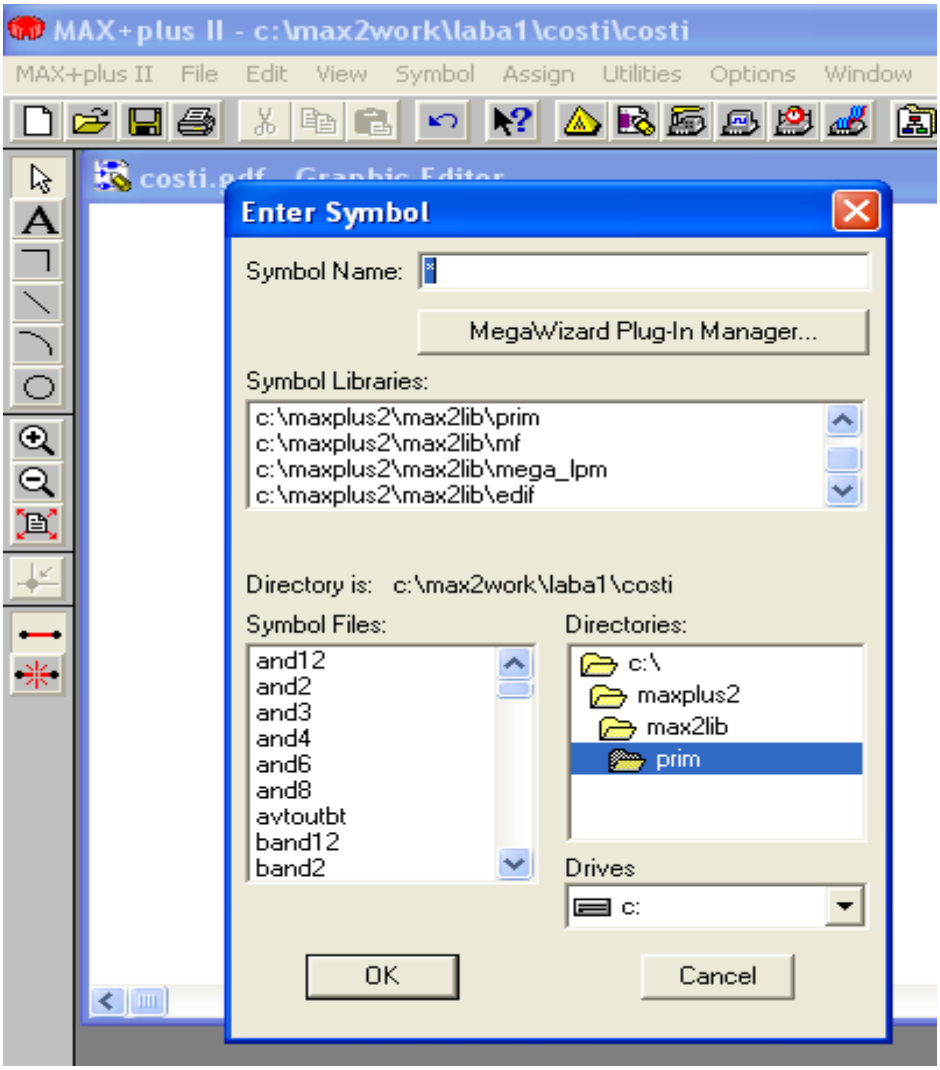

Рисунок 29 – Диалоговое окно ввода символа

После построения структурной схемы выполнить компиляцию проекта. Результатом компиляции является загрузочный файл, т. е. конфигурационная информация для выбранной микросхемы. В процессе компиляции проявляются

скрытые ошибки, которые были допущены при построении структурной схемы. После устранения ошибок проект необходимо повторно перекомпилировать. Проверка проекта осуществляется командой **Save&Check** подменю **Project** меню **File** либо кнопкой на панели инструментов, как это продемонстрировано на рисунке 31.

Для запуска компилятора необходимо использовать подменю **Compiler** из меню **MAX+plus II**, либо команду **Save&Compile** подменю **Project** из меню **File**, либо кнопки, расположенные на панели инструментов. Для начала компиляции необходимо нажать кнопку START (рисунок 32). Для получения дополнительной информации об ошибках и возможных способах их устранения использовать **Messages – Compiler** и **Help on Message**.

Для тестирования разработанного устройства использовать редактор временных диаграмм (**Waveform Editor**). В начале работы с **Waveform Editor** необходимо создать Waveform Editor files с расширением (*.scf – Simulator Channel Files). Для этого после открытия Waveform Editor (**MAX+plus II** | **Waveform Editor**) в предложенном окне отображаются все входы (input) и выходы (output), для чего используется команда **Insert Node** меню **Node** либо меню, вызываемое нажатием правой кнопки мыши. Далее файл с расширением *.scf необходимо сохранить. Для анализа работы разрабатываемого устройства на вход подаются чередующиеся **0** и **1** с периодом в соответствии с заданным вариантом, а выходы остаются неопределенными (рисунок 33).

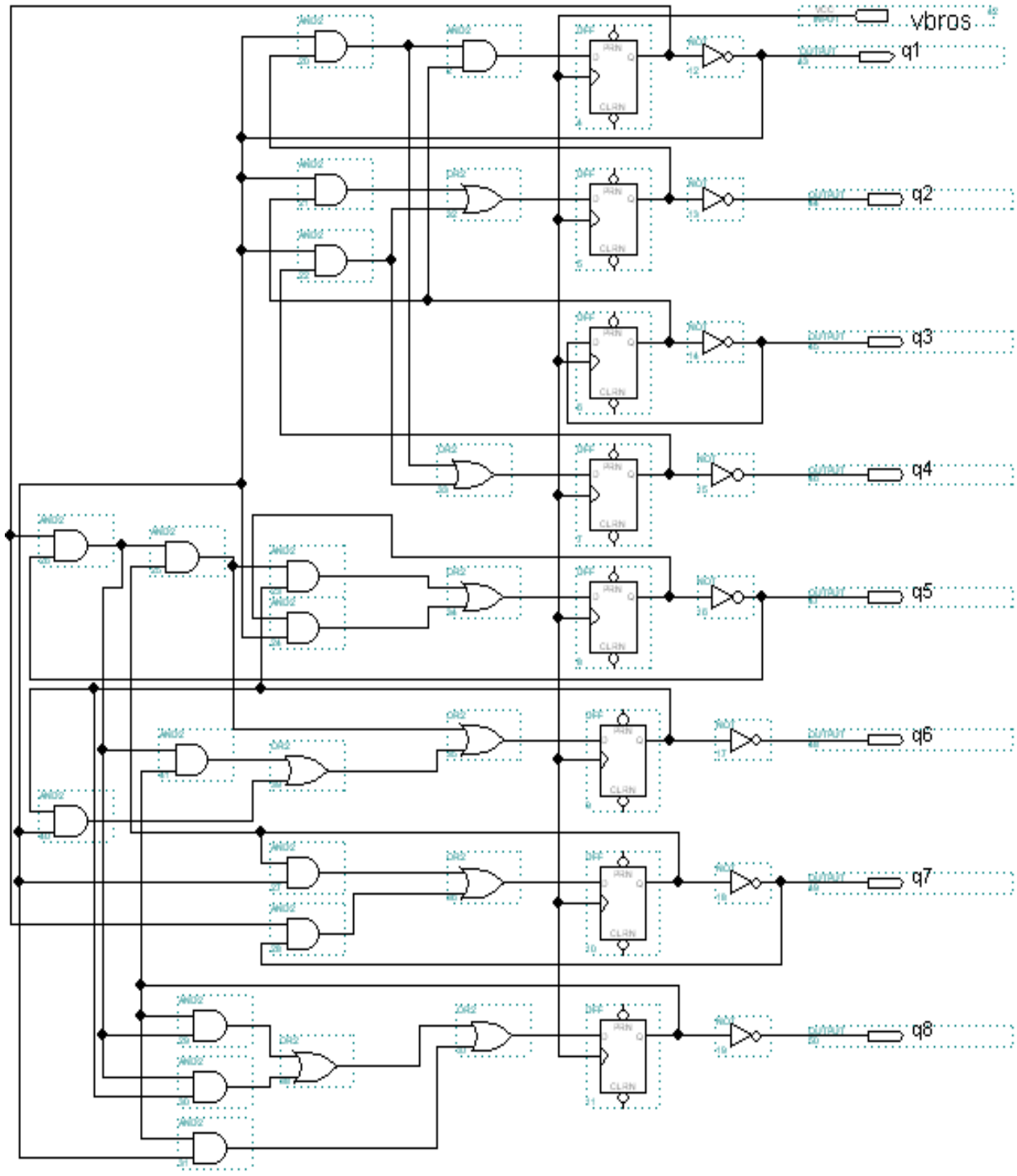

Рисунок 30 – Схема генератора квазислучайной последовательности чисел в графическом редакторе

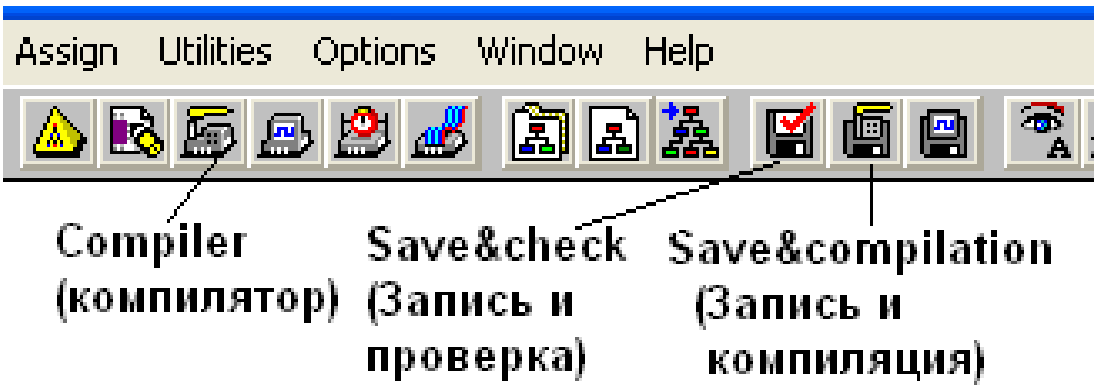

Рисунок 31 – Иконки компилятора на панели инструментов

Входные сигналы задаются командами на панели инструментов либо командами **Overwrite**, **Grow/Shrink**, вызываемыми правой кнопкой мыши. Для определения выходных сигналов используется **Simulator** из подменю **Simulator** меню **MAX+plus II** либо кнопка на панели инструментов (рисунок 34). Запуск Simulator осуществляется нажатием кнопки START.

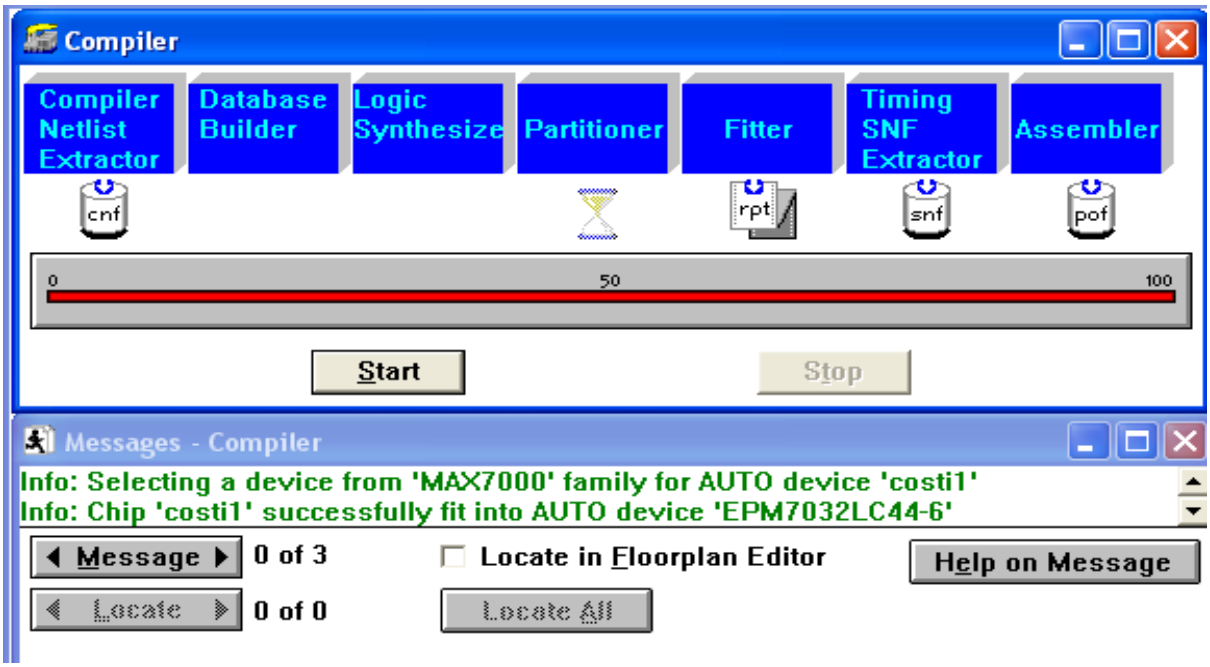

Рисунок 32 – Окно компилятора

При реализации проекта большое внимание уделяется анализу полученных результатов. При этом крайне важной характеристикой любого цифрового устройства является его быстродействие. Оно оценивается через временные характеристики, получаемые с помощью САПР. Современные САПР позволяют полностью автоматизировать процесс вычисления временных задержек, поскольку содержат внутри своих модулей полную информацию о схемотехнике и структуре проектируемого устройства, а также и о временных параметрах всех его компонентов.

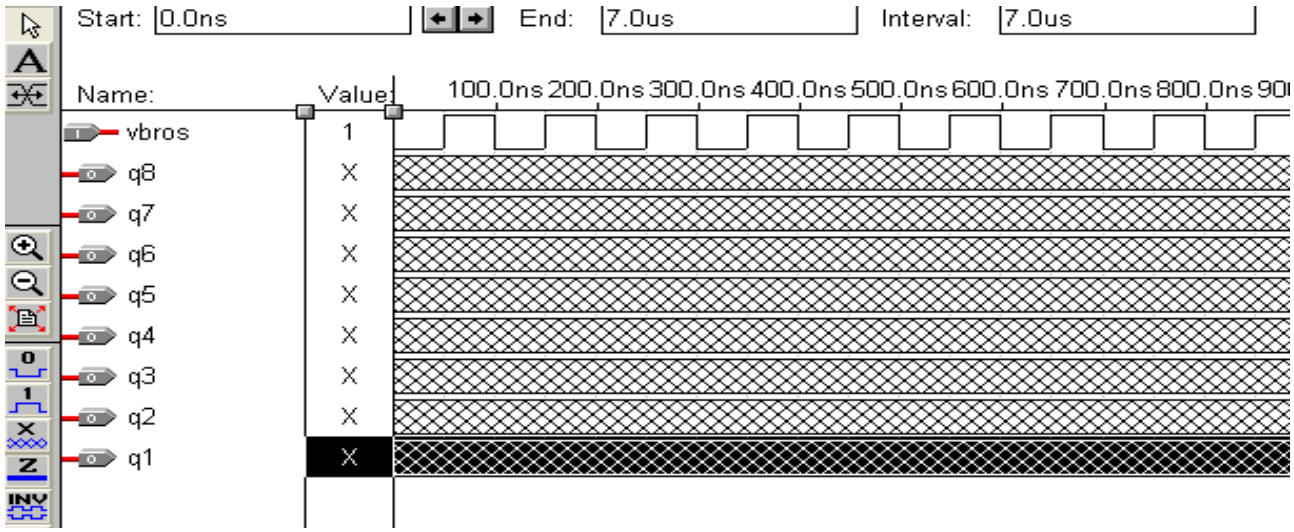

Рисунок 33 – Окно редактора временных диаграмм

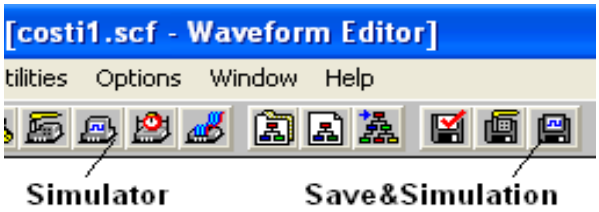

Рисунок 34 – Иконки симулятора на панели инструментов

Для вычисления временных задержек в САПР **MAX+plus II** используется операция **Timing Analyzer**, вызов которой осуществляется из меню **MAX+plus II**. В САПР MAX+plus II предусмотрено автоматизированное вычисление трех основных видов временных задержек (для вызова которых используется панель инструментов, рисунок 35):

– минимальных и максимальных задержек между источниками (входными сигналами) и приемниками (выходными сигналами) при реализации конкретного проекта. Информация об этом выдается в виде матрицы задержек (**Delay Matrix**);

– максимально возможной производительности устройства (пропускной способности) в виде максимальной частоты тактирования элементов памяти, используемых при реализации конкретного проекта (**Registered Performance**);

– времен предустановки и выдержки сигналов, гарантирующих надежную работу схем при фиксации сигналов в синхронных элементах памяти в процессе выполнения проекта (**Setup/Hold Time Analysis**).

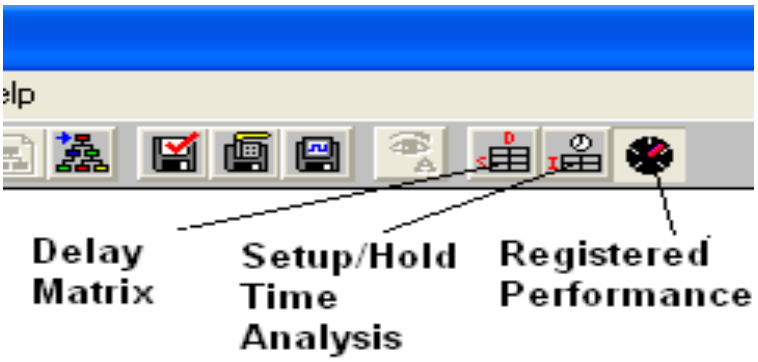

Рисунок 35 – Иконки для определения временных задержек на панели инструментов

Практическое задание

1. Настроить ПЛИС на реализацию функции генератора квазислучайной последовательности чисел. Алгоритм выполнения проекта идентичен тому, который использовался при выполнении лабораторной работы № 1.

2. Построить и исследовать временные диаграммы генератора квазислучайной последовательности чисел. При построении временных диаграмм использовать время окончания симуляции в соответствии с вариантом из таблицы 3.

3. Задать период входного сигнала в соответствии с вариантом из таблицы 3.

4. Записать подряд 20 значений генерированных интегральной микросхемой чисел, начиная с номера такта, указанного в варианте из таблицы 6. Убедиться в том, что последовательность чисел не является регулярной и период генератора превышает рассмотренный в задании интервал времени.

Таблица 6 – Задание к лабораторной работе № 4

Вариант	**Время окончания симуляции, мкс**	**Временная сетка входного сигнала VBROS, нс**	**Номер такта**
1	3	60	7
2	7	70	8
3	8	90	9
4	9	100	12
5	10	110	14
6	12	120	15

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Скриншоты основных этапов выполнения проекта.
4. Выводы.

Контрольные вопросы и задания

1. Перечислить основные модули САПР MAX + Plus II.
2. Указать путь к библиотеке примитивов, в которой находятся основные простые логические вентили.
3. Дать характеристику используемого в проекте D-триггера.
4. Как выбрать нужное семейство микросхем для реализации проекта?
5. Как в рамках заданного семейства микросхем выбрать нужную микросхему для реализации проекта?
6. Как назначается время окончания симуляции?
7. Что такое PRNG?
8. Что такое TRNG?
9. Приведите примеры TRNG.

Лабораторная работа № 5
Настройка ПЛИС на реализацию функции оперативного запоминающего устройства (ОЗУ)

Общие положения

Оперативная память (ОЗУ) – это тип хранилища данных, используемый в компьютерах, который обычно располагается на материнской плате. Это основная память, используемая компьютером для быстрого доступа, поскольку она намного быстрее считывается и записывается, чем другие формы хранения – в 20–100 раз быстрее, чем жесткие диски. Доступ к каждой ячейке внутри памяти можно получить независимо от адреса, что означает, что каждая ячейка может быть легко доступна одновременно (отсюда и термин «случайный»). ОЗУ – это энергозависимый тип памяти, а это означает, что вся информация, хранившаяся в ОЗУ, теряется при выключении компьютера. Энергозависимая память является временной памятью, а постоянная память (ПЗУ) является энергонезависимой и постоянно хранит данные при отключении питания.

Оперативная память – это память кратковременного доступа, используемая для быстрого выполнения оперативных задач, но по своей природе она ограничена. Каждый раз, когда компьютеру необходимо работать с приложением или программой, оперативная память используется для завершения этой операции. Однако, поскольку она нестабильна, никакие данные не могут храниться в ОЗУ постоянно, поэтому требуется долговременное хранилище, такое как жесткий диск. Когда ОЗУ заполняется, процессор компьютера должен переключаться между ними, чтобы выгрузить эти данные на жесткий диск. Следовательно, чем меньше объем оперативной памяти, тем медленнее работает компьютер. Базовый объем оперативной памяти составляет 4 гигабайт, но для игровых систем и высокопроизводительных рабочих станций для оптимальной производительности может потребоваться до 16 или 32 гигабайт и более.

Оперативная память используется для увеличения скорости работы приложений за счет загрузки в нее информации для максимально быстрого доступа. Он также используется для повышения доступности данных, к которым ранее осуществлялся доступ. Например, при первом запуске программы после перезагрузки требуется больше времени, так как компьютер загружает ее из долговременного хранилища. Однако, если программное обеспечение будет закрыто и снова открыто, операция произойдет почти мгновенно, поскольку теперь оно загружается из гораздо более быстрой оперативной памяти. Оперативная память используется для любой операции, требующей быстрого доступа к вычислительным ресурсам, например для доступа ко многим драйверам операционной системы, которые загружаются в нее каждый раз при загрузке компьютера.

Существует два основных типа оперативной памяти: динамическая оперативная память (Dynamic Random Access Memory – DRAM) и статическая оперативная память (Static Random Access Memory – SRAM). Оперативная память в

большинстве персональных компьютеров (ПК) является динамической. Все микросхемы динамической ОЗУ на модулях DIMM, SIMM или RIMM должны обновляться каждые несколько миллисекунд, перезаписывая данные в модуль.

Статическая RAM (SRAM) – это энергозависимая память, которая часто используется в кэш-памяти и регистрах, потому что она намного быстрее и не требует обновления, как динамическая RAM. Пока подается питание, SRAM сохраняет свое содержимое, и его не нужно регулярно обновлять. Обычно SRAM использует матрицу из шести транзисторов, которые не зависят от мощности, чтобы избежать утечек, поскольку они действуют как переключатели, выполняющие функции 1 и 0 без конденсатора, удерживающего заряд. Однако, поскольку требуется больше транзисторов, чем для DRAM, матрица требует дополнительного места, что увеличивает стоимость производства. SRAM может работать на более высокой скорости и потребляет меньше энергии, чем DRAM, но она также дороже и поэтому используется только в качестве кэш-памяти процессора. Вначале SRAM хранилась на материнской плате, позже SRAM находилась внутри корпуса центрального процессора (ЦП) или хранилась как на материнской плате, так и внутри ЦП.

Динамическая RAM – это память, которую необходимо обновлять. Обновление выполняется контроллером памяти, который является частью набора микросхем на материнской плате. Каждая ячейка содержится в электрическом конденсаторе с транзисторами, служащими затворами, которые определяют, может ли каждое значение быть прочитано или записано. Чтобы компенсировать утечку конденсатора, DRAM необходимо обновлять с помощью электронного заряда, который перезаписывает данные каждые несколько миллисекунд. DRAM обычно меньше и дешевле, чем SRAM, и используются практически в каждом ПК.

Существует несколько критериев выбора оперативной памяти:

- объем;
- тактовая частота;
- тайминг;
- напряжение.

Рассмотрим вкратце каждый из них.

Объем. Объем оперативной памяти наряду с характеристиками прочих комплектующих ПК непосредственно влияет на производительность системы в целом. При достаточном объеме ОЗУ операционная система реже задействует файл подкачки, что исключает лишние операции чтения/записи, которые проходят на более низких скоростях. Объем одного модуля оперативной памяти зависит от типа памяти. Пример минимального и максимального объема памяти для каждого типа представлен в таблице 7.

Таблица 7 – Минимальный и максимальный объем модуля памяти для типов ОЗУ

Тип памяти	Объем модуля памяти	
	Минимальный	Максимальный
DDR	256 Мб	1 Гб
DDR2	512 Мб	4 Гб
DDR3	1 Гб	16 Гб
DDR4	4 Гб	128 Гб

Тактовая частота. Параметр зависит от типа оперативной памяти: DDR, DDR2, DDR3, DDR4. Значения представлены в таблице 8. Чем выше тактовая частота, тем лучше. Обязательно стоит учитывать характеристики процессора, который должен поддерживать соответствующую тактовую частоту ОЗУ.

Таблица 8 – Минимальная и максимальная тактовая частота модуля памяти для типов ОЗУ

Тип памяти	Тактовая частота модуля памяти, МГц	
	Минимальная	Максимальная
DDR	100	350
DDR2	200	600
DDR3	800	2400
DDR4	1600	3200

Обязательно следует учитывать режим работы: одно- или двухканальный. Если процессор способен работать с максимальной частотой определенного типа памяти в одноканальном режиме, он может не поддерживать данную частоту в двухканальном режиме. При этом система запустится и будет работать, но на более низкой частоте.

Стоит отметить тот факт, что оперативная память независимо от типа в процессе своей работы поддерживает весь диапазон тактовых частот, расположенных ниже своей максимальной частоты. К примеру, максимальная тактовая частота модуля памяти DDR 4 составляет 2400 МГц – ОЗУ может работать на следующих частотах: 2400, 2133, 1866, 1600 МГц.

Частота, на которой запустится оперативная память (без учета разгона) зависит от характеристик процессора, чипсета материнской платы и установленной видеокарты. Если какой-то из компонентов системы будет «тормозить», то память не запустится на пределе своих возможностей.

Тайминг. Тайминг или латентность – время задержки доступа к ячейкам памяти между операциями чтения/записи. Это важный параметр оперативной памяти.

CAS Latency (CL) – один из самых значимых показателей: именно он указывает, сколько времени в целом уходит на поиск необходимых данных после того, как ЦП попросит доступ на считывание. Чем меньше показатель CAS Latency, тем лучше.

RAS to CAS Delay (tRCD) – показатель демонстрирует время полного доступа к данным, т. е. задержку, вызванную поиском нужного столбца и строки в двумерной таблице. Чем меньше значение, тем выше быстродействие ОЗУ.

Row Precharge Delay (tRP) – динамическая память DRAM, ее ячейки время от времени разряжаются и нуждаются в периодической перезарядке. По этой причине данные, которые содержатся в ней, обновляются. Этот процесс называется регенерацией ОЗУ. Он реализуется специальным контроллером, установленным на материнской плате или же на кристалле центрального процессора. Таким образом, данный показатель отображает временной отрезок между сигналом на зарядку – регенерацию ОЗУ – и разрешением на доступ к следующей строчке информации. Чем меньше этот параметр, тем быстрее работает память. На протяжении времени, называемого шагом регенерации, в DRAM перезаписывается целая строка ячеек, после чего обновляются все строки памяти.

Activate to Precharge Delay (tRAS) – минимальное время активности строки, т. е. минимальное время между активацией строки (ее открытием) и подачей команды на предзаряд (начало закрытия строки). Строка не может быть закрыта раньше этого времени. Высокий показатель данного параметра заметно сокращает производительность памяти, из-за того что закрытие ячейки требует дополнительного времени, поэтому чем ниже значение tRAS, тем лучше.

Напряжение. Этот показатель часто указывается в характеристиках и отражает минимальное напряжение, необходимое для работы модуля. Повышенное напряжение позволяет использовать память на больших частотах с сохранением стабильности работы, чем это предписано (так называемый разгон).

Создание ОЗУ емкостью 4к×16

В процессе лабораторной работы необходимо создать ОЗУ емкостью 4к×16 (4096 16-битовых слов). Моделирование проводить на ПЛИС семейства FLEX10KE. Разработка ОЗУ производится в графическом редакторе. С помощью диалогового окна Enter Symbol из меню устанавливаем в графический файл необходимые логические элементы. Для реализации ОЗУ емкостью 4к×16 необходимо из библиотеки примитивов разместить на поле графического редактора шесть входов (input) и один выход (output), а из библиотеки mega_lmp – LMP_RAM_DP. Выбрать и установить необходимый элемент можно двумя способами:

– набрать имя элемента в окне **Symbol Name** диалогового окна **Enter Symbol** и нажать ОК.

– выбрать необходимую библиотеку в окне **Symbol Libraries** диалогового окна **Enter Symbol** и двойным щелчком левой клавиши мыши открыть ее. Затем аналогичным образом выбрать необходимый элемент в окне **Symbol File**.

Параметры ОЗУ, которое необходимо реализовать с помощью ПЛИС, приведены в таблице 9.

Таблица 9 – Параметры LMP_RAM_DP

Порты		
Name	Status	Inversion
data[LPM_WIDTH-1..0]	Used	None
q[LPM_WIDTH-1..0]	Used	None
rdaddress[LPM_WIDTHAD-1..0]	Used	None
rdclken	Used	None
rdclock	Used	None
rden	Used	None
wraddress[LPM_WIDTHAD-1..0]	Used	None
wrclken	Used	None
wrclock	Used	None
wren	Used	None
Параметры		
Name	Value	
LPM_RDADDRESS_CONTROL	“REGISTERED”	
LPM_WRADDRESS_CONTROL	“REGISTERED”	
LPM_FILE	<none>	
LPM_INDATA	"REGISTERED"	
LPM_NUMWORDS	2^LPM_WITHAD	
LPM_OUTDATA	"UNREGISTERED"	
LPM_WIDTH	<none>	
LPM_WITHAD	<none>	

Изменение параметров устройства происходит в диалоговом окне **Edit Ports/Parameters**. Для вызова данного окна необходимо щелкнуть правой кнопкой мыши на элементе и выбрать из появившегося меню пункт **Edit Ports/Parameters**, рисунок 36. Само окно редактирования и настройки параметров ОЗУ показано на рисунке 37.

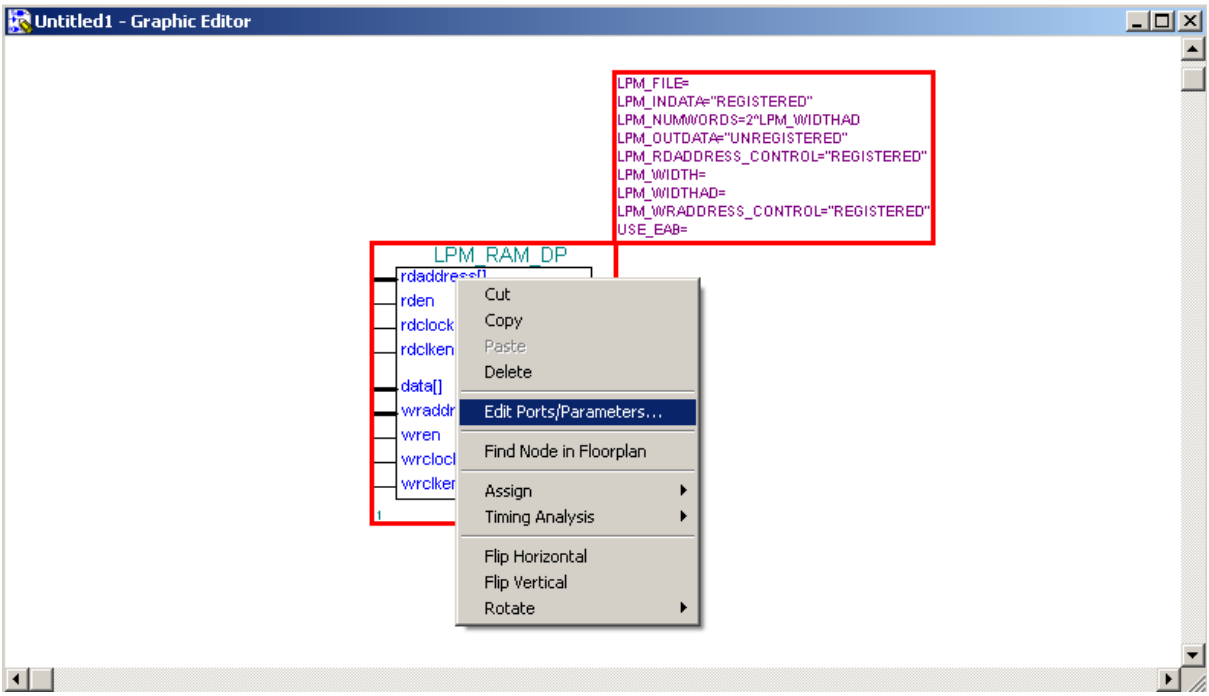

Рисунок 36 – Вызов окна редактирования

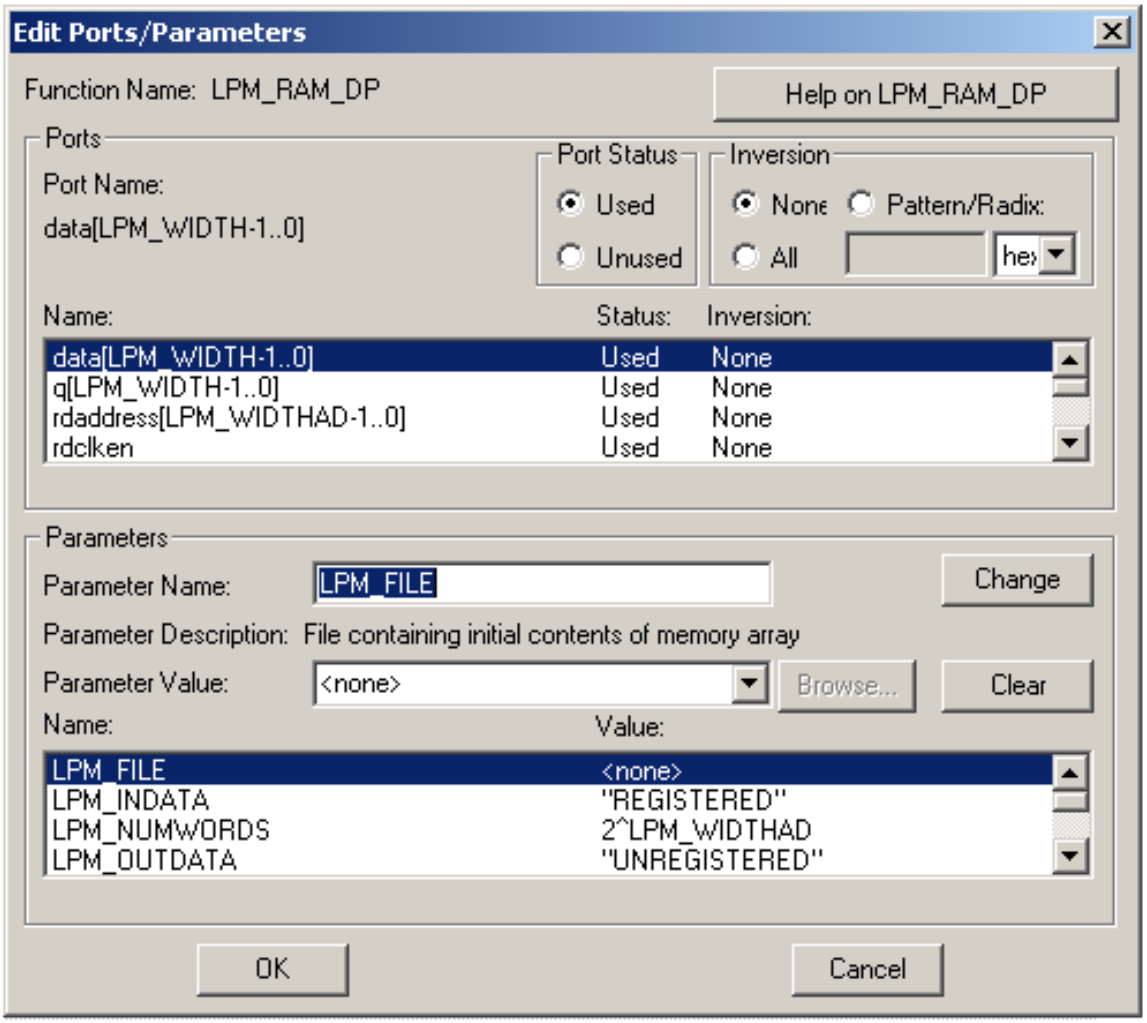

Рисунок 37 – Окно редактирования параметров ОЗУ

В данном окне необходимо отключить входы **rdclken** и **wrclken**, поскольку для работы микросхемы они необязательны. Сделать это можно, выделив соответствующий вход и выбрав **Unused** в разделе **Port Status**. Для данного проекта также необходимо назначить 12 адресных и 16 информационных входов, что задаст микросхеме емкость 4к×16 (4к = 10^{10+2} = 4096. Отсюда разрядность адресной шины равна 12). В окне **Parameters** выбрать размерность шины данных (**LPM_WIDTH**) и установить 16. Размерность шины адреса (**LPM_WIDTHAD**) установить равной 12.

Для соединения функциональных элементов используется панель инструментов, расположенная в левой части главного окна. Схема ОЗУ приведена на рисунке 38.

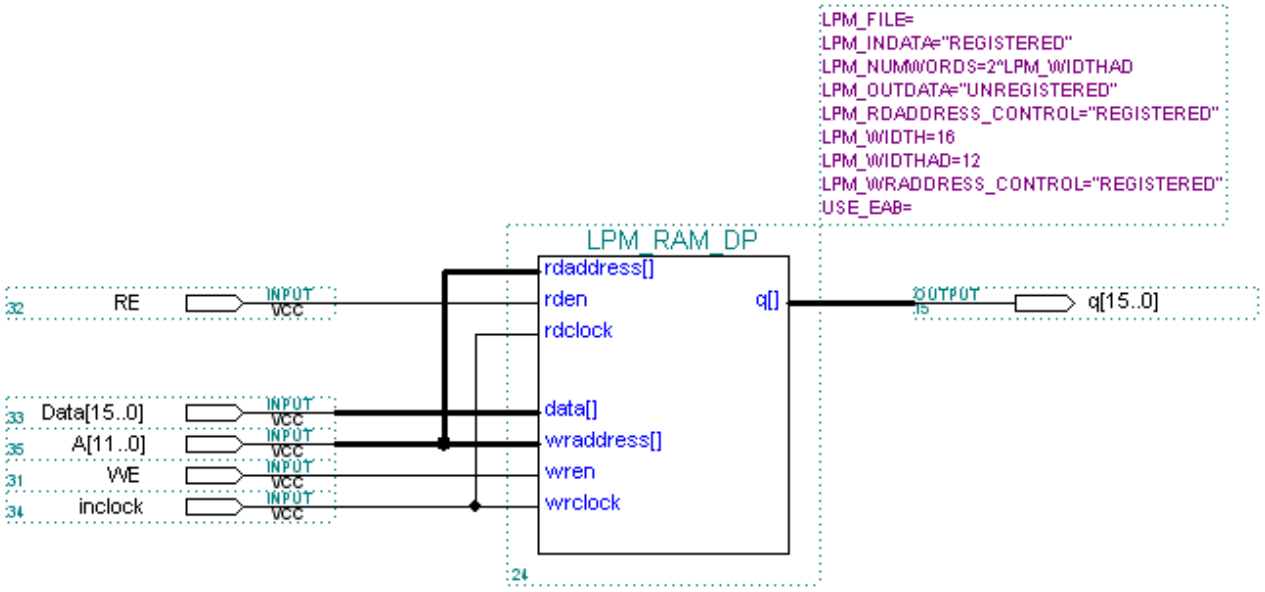

Рисунок 38 – Функциональная схема ОЗУ 4к×16

Каждому входу и выходу необходимо присваивать уникальное имя. Для этого надо двойным щелчком выделить надпись PIN_NAME элемента и ввести имя контакта. Для шины в имени необходимо указывать ее размер в квадратных скобках, как это продемонстрировано на рисунке 38.

Чтобы убедиться, что схема логически корректна, следует сохранить файл и проверить его на основные ошибки. Перед этим нужно назначить файл проекта в качестве текущего. Для этого необходимо в меню **File** (рисунок 39) выбрать пункт **Project** (проект), а далее выбрать пункт **Set Project to Current File** (назначить имя проекта по текущему файлу).

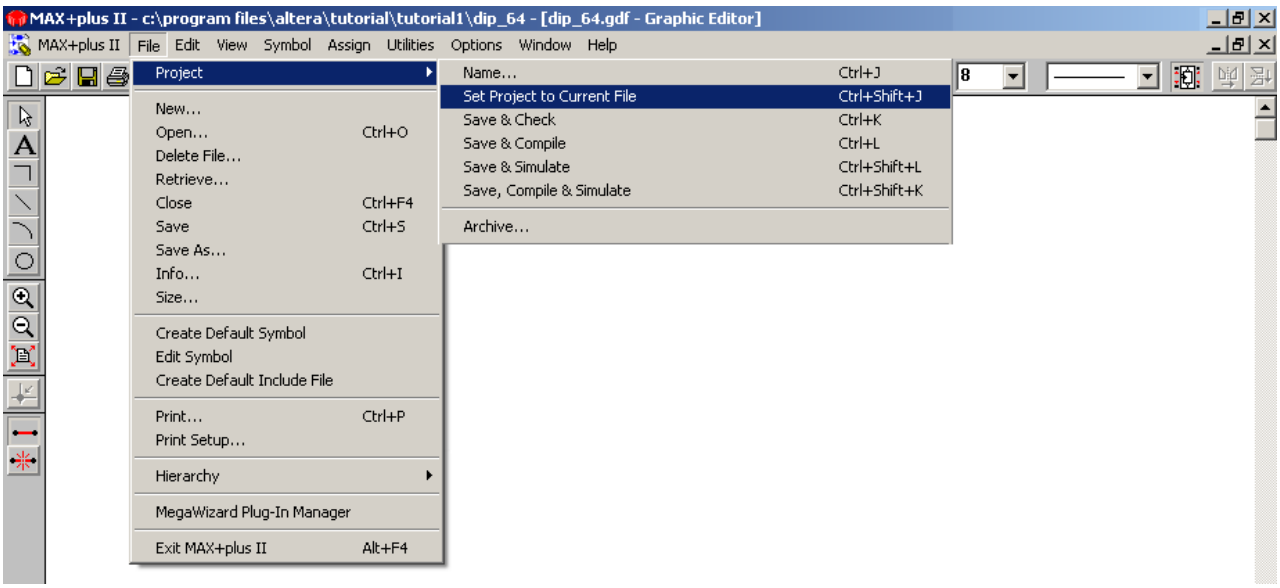

Рисунок 39 – Назначение проекта в качестве текущего

Для проверки в меню **File** выбираем **Project, Save&Check** (сохранить и проверить проект). При этом появится окно компилятора, рисунок 40. Если в проекте нет ошибок, после завершения проверки появится окно MAX+PLUS II-**Compiler**, в котором число ошибок будет равно нулю, рисунок 41. Если же в проекте будут найдены ошибки, окно MAX+PLUS II-**Compiler** будет содержать количество ошибок, и откроется вместе с окном **Message Compiler**, в котором будет показано, какие именно совершены ошибки, рисунок 42. В таком случае дальнейшая работа будет возможна только после устранения ошибок и повторной проверки.

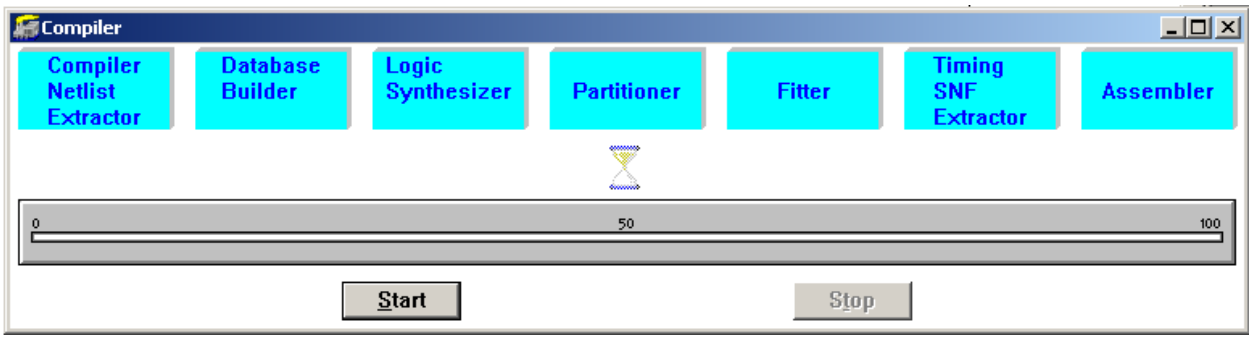

Рисунок 40 – Окно компилятора

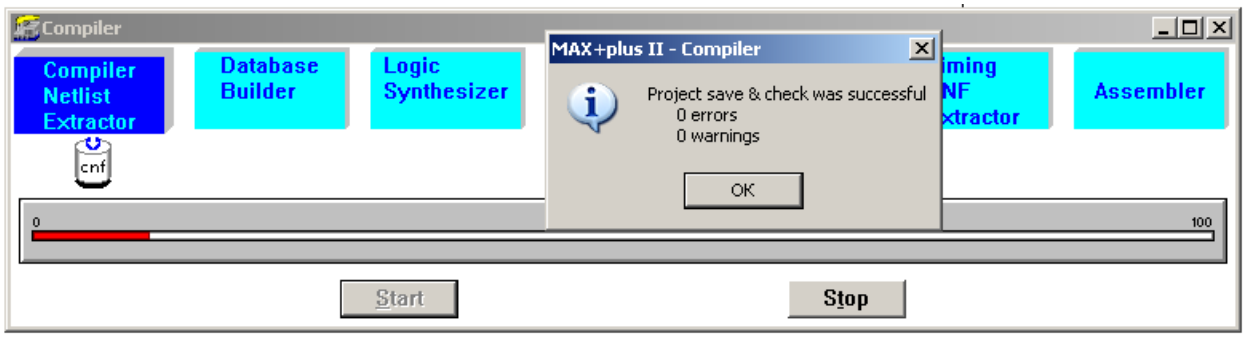

Рисунок 41 – Проверка проекта завершена успешно

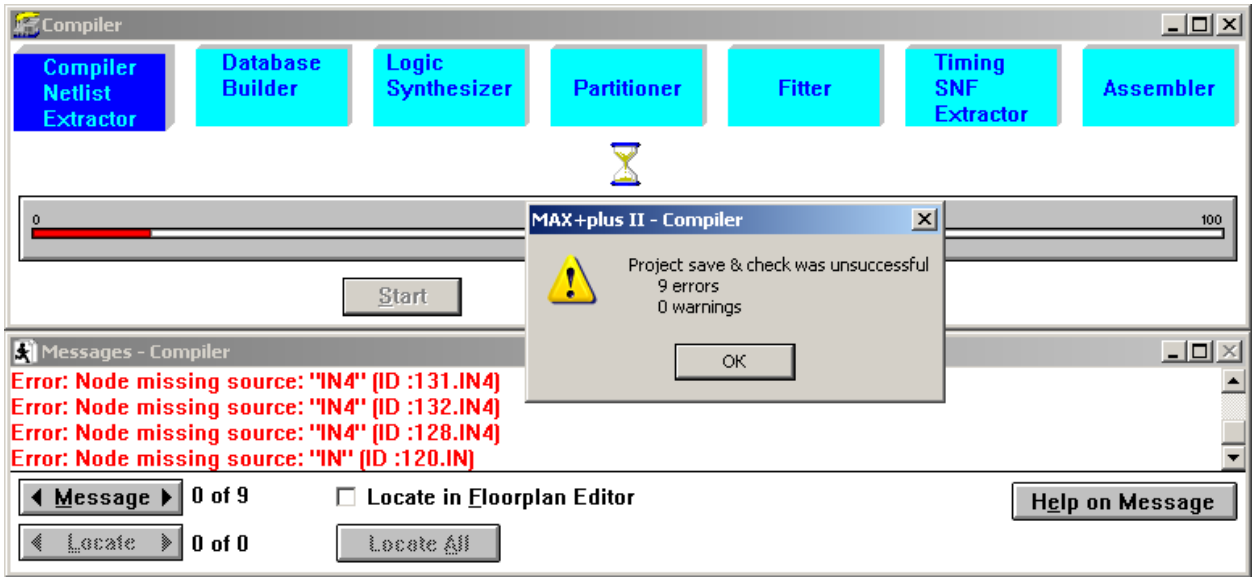

Рисунок 42 – Процессор сообщений

Если проверка завершена успешно, можно переходить к процессу компиляции, предварительно закрыв окно MAX+PLUS II-**Compiler**. Результатом компиляции является загрузочный файл, т. е. конфигурационная информация для выбранной микросхемы.

Перед компиляцией необходимо назначить семейство микросхем. Для этого в меню **Assign** (Назначения) необходимо выбрать пункт Device (микросхемы). После этого появится окно, представленное на рисунке 43.

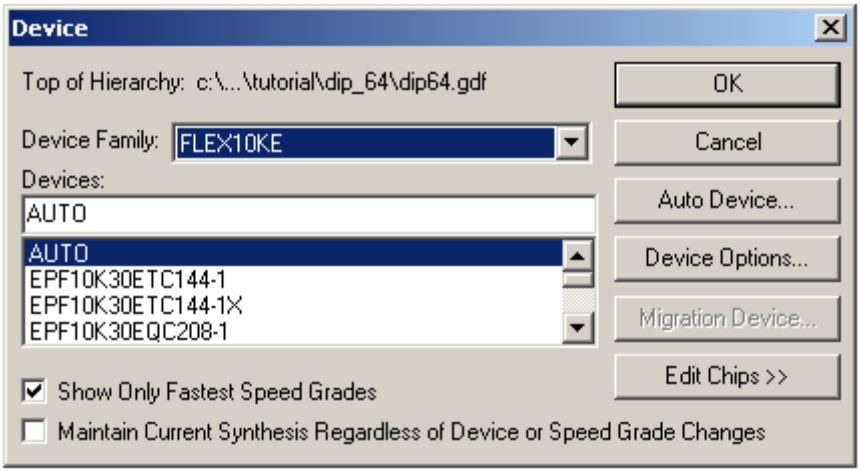

Рисунок 43 – Окно выбора семейства микросхем

В пункте **Device Family** (Семейство микросхем) можно выбрать любое семейство, поддерживаемое MAX+PLUS II. Для данного проекта подходят только два семейства микросхем: **FLEX10KE** и **ACEX1K**. Выберем **FLEX10KE**. Конкретные микросхемы данного семейства компилятор назначит автоматически.

Следующим шагом разработки является моделирование. Оно необходимо, т. к. позволяет полностью протестировать проект, чтобы убедиться, что он правильно функционирует. Для моделирования необходимо задать симулятору

входные векторы. Симулятор использует их для формирования выходных сигналов, которые запрограммированная микросхема должна была бы сформировать при тех же условиях. Задание входных векторов проводится в редакторе временных диаграмм. В начале работы с ним необходимо создать **Waveform Editor files** (.scf). Для этого при создании нового файла в окне **New** необходимо выбрать **Waveform Editor files**, рисунок 44.

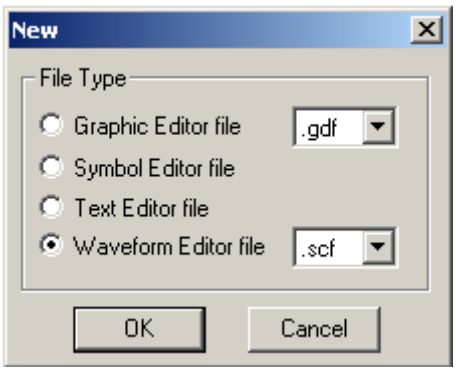

Рисунок 44 – Создание файла временных диаграмм

В открывшееся окно все цепи, входные и выходные, вставляются из snf файла. Для этого используется команда **Enter Node from SNF** меню **Node**, появляется диалоговое окно **Enter Node from SNF**, рисунок 45.

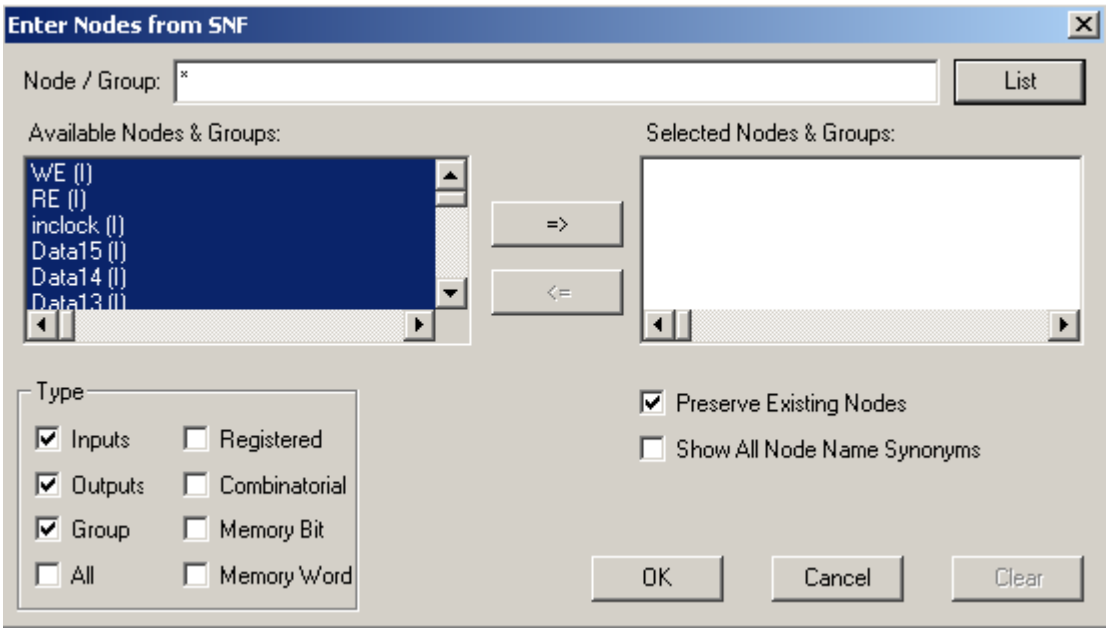

Рисунок 45 – Окно выбора цепей

После нажатия кнопки **List** (список) в форме **Available Nodes&Groups** (доступные узлы и группы) отображаются все входные и выходные контакты. Необходимые контакты двойным щелчком переносятся в форму **Selected Nodes&Groups** (выбранные узлы и группы). После того как все необходимые контакты будут выбраны, необходимо нажать кнопку OK. Все выбранные цепи отобразятся в файле scf. Для данной работы необходимо выбрать следующие

контакты: синхровход (inclock), шина адреса (A), входная (Data) и выходная (q) шины данных, входы разрешения записи (WE) и чтения (RE).

Для анализа работы разрабатываемого устройства на входы подаются предполагаемые входные сигналы, а выходы задаются неопределенными, рисунок 46. Сигналы задаются с помощью панели инструментов, расположенной в левой части окна временных диаграмм.

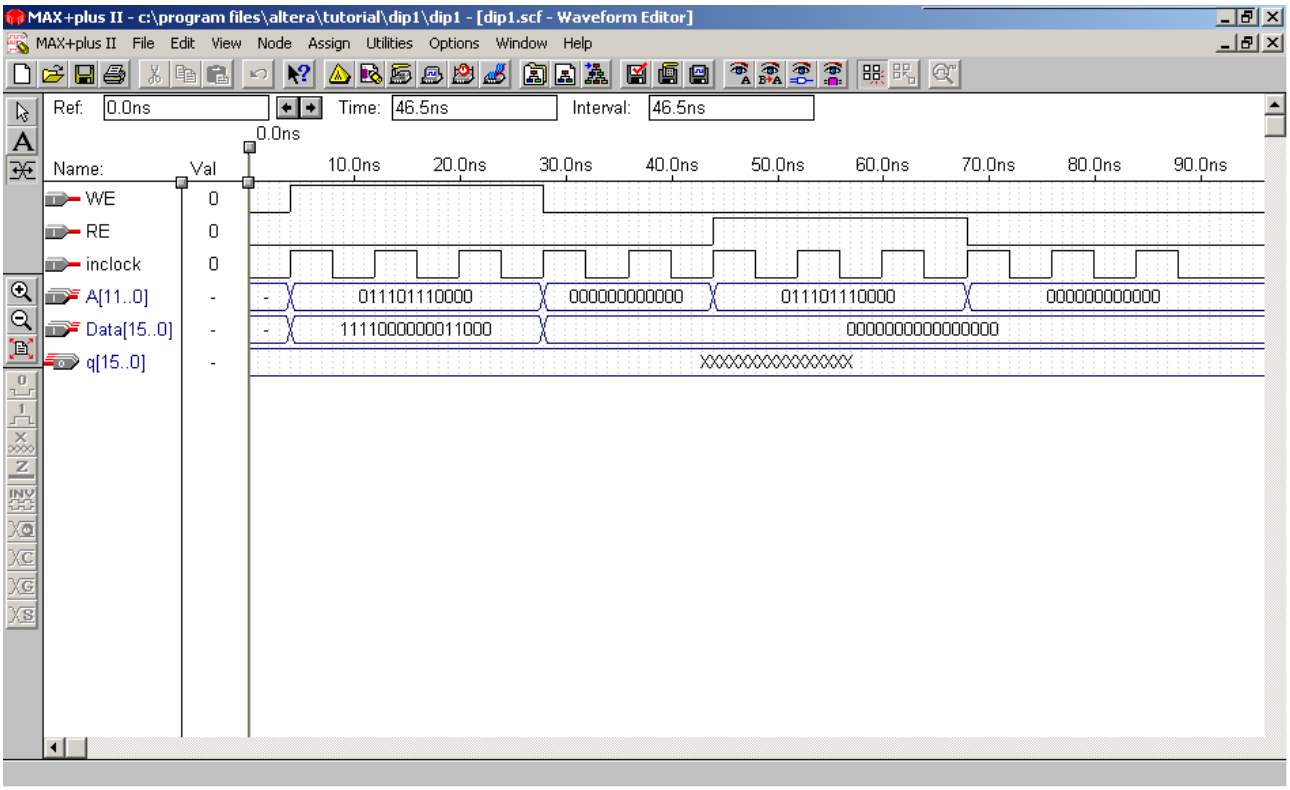

Рисунок 46 – Начальные временные диаграммы для ОЗУ емкостью 4к×16

На синхровход подаются синхроимпульсы с периодом, указанным в таблице 10 согласно варианту задания. В течение определенного времени (см. таблицу 10) на вход разрешения записи **WE** подается высокий уровень сигнала, на информационный вход подается записываемая в память комбинация, а на вход адреса – адрес ячейки памяти, в которую производится запись. Во время действующего сигнала разрешения записи необходимо по трем разным адресам записать три разных числа. Данные для каждого варианта берутся из таблицы 10. Через некоторое время на вход разрешения чтения **RE** подается в течение определенного времени (см. таблицу 10) высокий уровень сигнала, а на адресные входы – те же адреса (в любой последовательности), по которым происходила запись.

Когда scf файл готов, необходимо запустить симулятор, для этого в меню MAX+PLUS II следует выбрать **Simulator**. В качестве входного файла у него будет созданный нами файл. Симулятор запускается кнопкой Start, рисунок 47. Результатом работы симулятора будет обновление scf файла, рисунок 48.

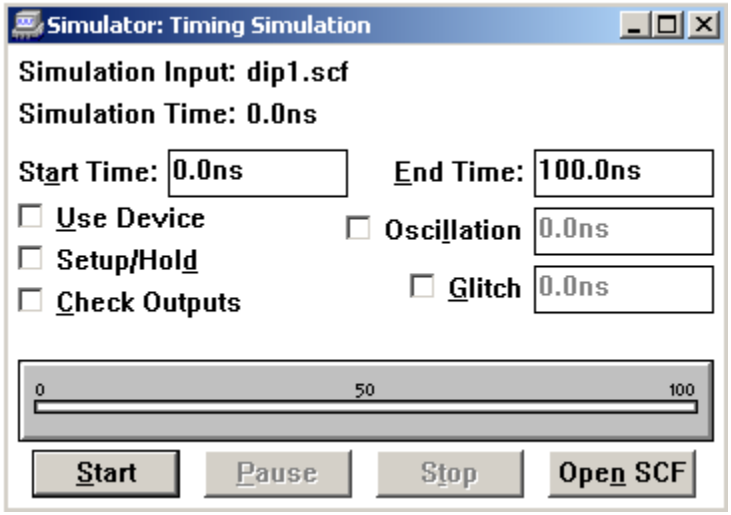

Рисунок 47 – Окно симулятора

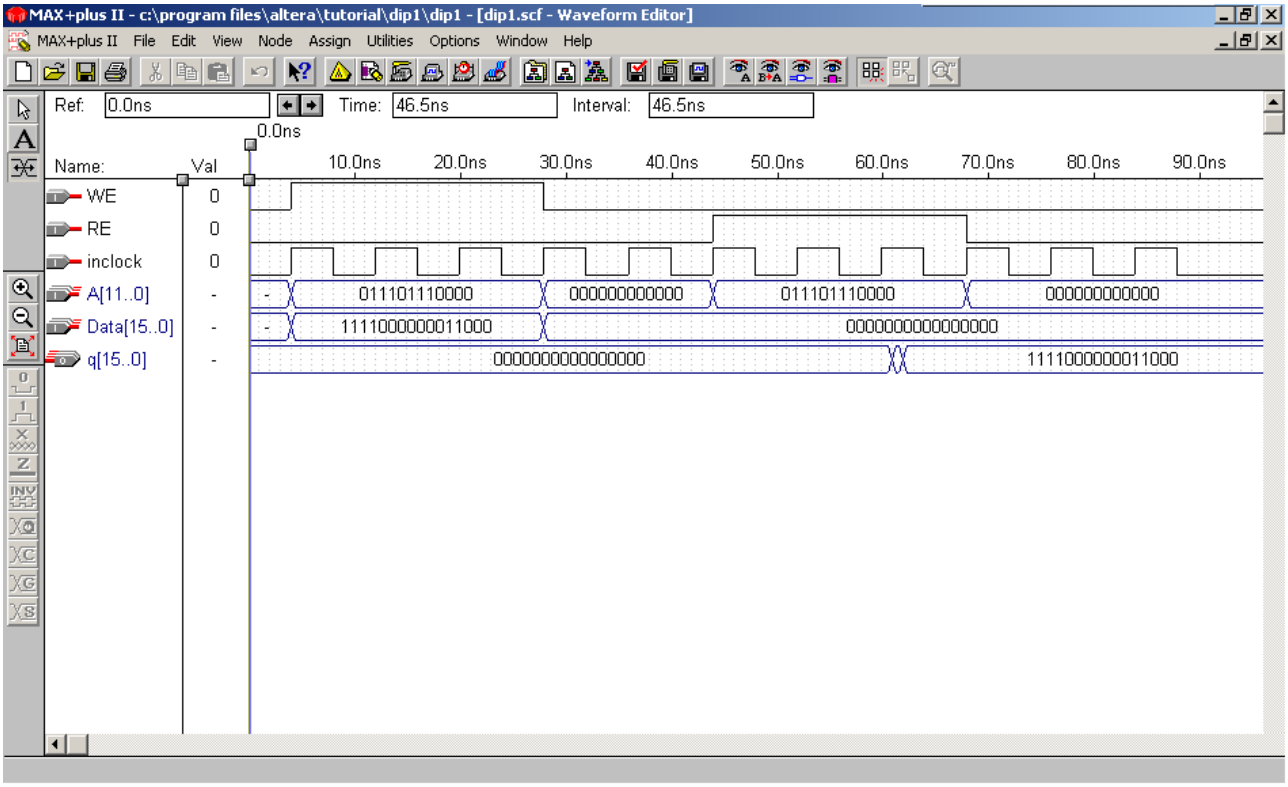

Рисунок 48 – Пример временных диаграмм для ОЗУ емкостью 4к×16

Через определенное время задержки на информационных выходах установится комбинация, которая была до этого записана в ячейку памяти по данному адресу. Это свидетельствует о корректной настройки ПЛИС на реализацию функции ОЗУ.

Практическое задание

1. Настроить ПЛИС семейства **FLEX10KE** на реализацию функции ОЗУ емкостью 4к×16.

2. При построении временных диаграмм использовать время окончания симуляции, значения временной сетки, время действия операции записи и операции чтения информации, адреса, по которым надо записать данные, указанные в соответствующем варианте в таблице 10.

3. Данные для адреса и чисел вводить с помощью группировки битов.

4. Проанализировать временные параметры.

Таблица 10 – Задание к лабораторной работе № 5

Вариант	**Время окончания симуляции, мкс**	**Временная сетка, нс**	**Время действия операции, нс**		**Адреса**	**Числа**
			запись	**чтение**		
1	1,5	40	480	440	4_{16},	30_{16},
					A_{16},	13_{16},
					19_{16}	$C0_{16}$
2	2	50	500	550	35_{16},	63_{16},
					$3D_{16}$,	$6B_{16}$,
					$4D_{16}$	E_{16}
3	2,5	60	600	780	AB_{16},	3_{16},
					$E4_{16}$,	AD_{16},
					$C1_{16}$	FE_{16}
4	3	80	800	880	7_{16},	$3E8_{16}$,
					$1A1_{16}$,	514_{16},
					98_{16}	C_{16}
5	3,5	100	1000	1300	$F3_{16}$,	2_{16},
					$AA1_{16}$,	465_{16},
					6_{16}	$7D1_{16}$
6	4	100	900	1200	33_{16},	BC_{16},
					$3F_{16}$,	65_{16},
					273_{16}	$E3_{16}$

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Скриншоты основных этапов выполнения проекта.
4. Выводы.

Контрольные вопросы и задания

1. Что такое ОЗУ? Описать принцип его работы.
2. Что такое SRAM? Принцип его построения.
3. Что такое DRAM? Принцип его построения.
4. Каковы основные параметры ОЗУ?

5. Что означает емкость ОЗУ 8к×16?

6. Сколько чисел можно записать в ОЗУ емкостью 2к×8?

7. Какова разрядность числа, записываемого в ОЗУ емкостью 8к×4?

8. Что означает вход WE?

9. Что означает вход RE?

10. Как в рамках заданного семейства микросхем выбрать нужную микросхему для реализации проекта?

11. Как осуществляется настройка параметрического модуля?

12. Как осуществляется группировка переменных при построении временных диаграмм?

13. Как осуществляется выбор системы счисления при представлении групповых данных?

14. Как назначается время окончания симуляции?

Лабораторная работа № 6
Наращивание размерности ОЗУ до емкости 8к×16

Увеличим емкость ОЗУ до 8к×16, используя ранее созданное ОЗУ емкостью 4к×16. Поскольку новое ОЗУ имеет 8к = 10^{10+3} = 8192 ячеек памяти, то ему необходимо 13 разрядов адреса. Используемый параметрический модуль позволяет назначить только 12-битный адрес. Поэтому следует нарастить размерность. Для этого необходимо использовать два параметрических модуля. С помощью старшего бита 13-разрядного адресного числа (A12) происходит выбор одного из двух параметрических модулей, каждый из которых настроен на реализацию функции ОЗУ емкостью 4к×16. Выбор модуля осуществляется в соответствии с таблицей 11. Таким образом, увеличивая количество адресных входов и усложняя схему выбора, можно построить ОЗУ любой необходимой емкости, т. е. нарастить размерность ОЗУ.

Таблица 11 – Выбор параметрического модуля

A12	**ОЗУ 1**	**ОЗУ 2**
0	Выбрано	Не выбрано
1	Не выбрано	Выбрано

На рисунке 49 представлена схема включения двух параметрических модулей, собранная в графическом редакторе САПР MAX +plus II. Выбираем семейство ПЛИС **FLEX10KE**.

Поскольку у рассматриваемых ОЗУ нет входа выбора микросхемы Chip Select, выбор одного из двух ОЗУ не может быть осуществлен напрямую значением сигнала на входе A12. Поэтому в качестве входа выбора микросхемы в циклах чтения и записи используются входы разрешения чтения и записи соответственно. Для упрощения схемы ОЗУ 8к×16 высокий уровень сигнала на разрешающие входы чтения и записи подается одновременно для обоих ОЗУ 4к×16. Поэтому в схему введены четыре элемента 2И, по два на ОЗУ 4к×16. Каждый параметрический модуль настраивается, как в лабораторной работе № 5. Для каждого ОЗУ один из элементов И в зависимости от поданных на него сигналов разрешает работу в цикле записи, другой элемент И – в цикле чтения.

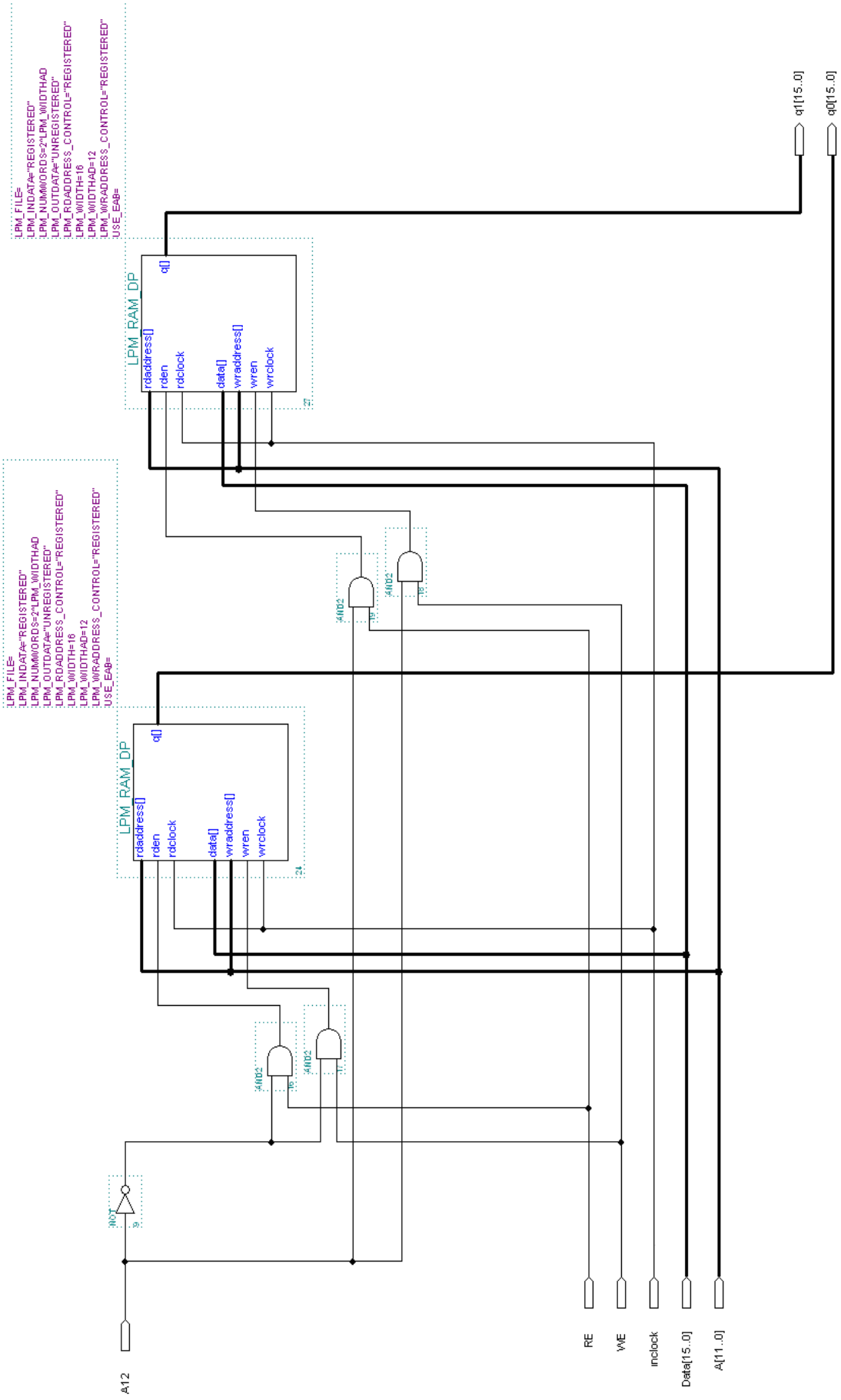

Рисунок 49 – Схема включения двух параметрических модулей

На рисунке 50 показаны примеры временных диаграмм до начала работы симулятора.

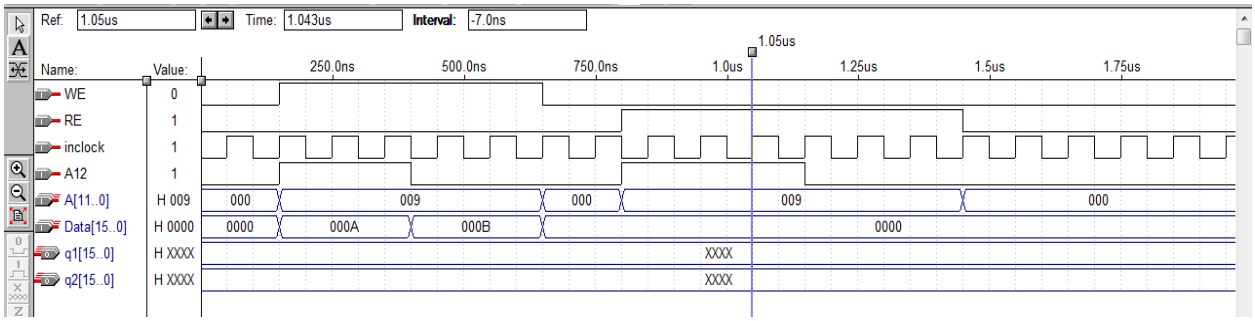

Рисунок 50 – Расстановка входных сигналов в ОЗУ

Сигналы расставлены так, чтобы по одному и тому же адресу в разные параметрические модули записать разные числа. При **A12 = 1** данные записываются во второй модуль (выход **q1**), а при **A12 = 0** – в первый модуль (выход **q2**). На рисунке 51 показан результат симуляции. С разных параметрических модулей читаются те числа, которые записывались в них в режиме записи.

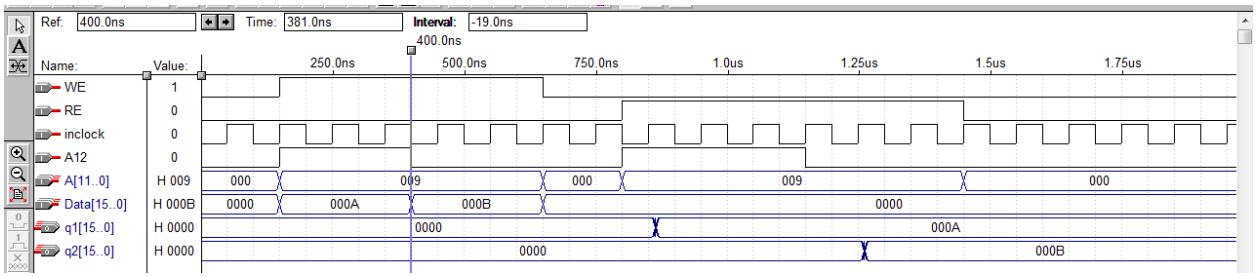

Рисунок 51 – Результат симуляции ОЗУ 8к×16

Практическое задание

1. Ознакомиться с принципами наращивания размерности емкости оперативного запоминающего устройства в САПР MAX+PLUS II.

2. Нарастить размерность ОЗУ до 8к×16. Использовать ПЛИС семейства **FLEX10KE**. При выполнении операции компиляции в данном случае поступит запрос системы, с которым надо согласиться. Запрос компилятора выглядит следующим образом: «Project doesn`t fit. Do you wish to override some existing settings and/or assignments? – Проект не подходит. Вы хотите переопределить некоторые существующие настройки и/или назначения?» Для корректного продолжения работы в появившемся окне «Override User Assignments – Переопределить назначения пользователя» необходимо отметить галочкой поле «Add Extra Devices as Needed – Добавить необходимое устройство».

3. При построении временных диаграмм использовать время окончания симуляции, значения временной сетки, время действия операции записи и операции чтения информации, адреса, по которым надо записать данные, указанные в соответствующем варианте в таблице 12.

4. Данные для адреса и чисел вводить с помощью группировки битов.

5. Проанализировать временные параметры полученного ОЗУ.

Таблица 12 – Задание к лабораторной работе № 6

Вариант	Время окончания симуляции, мкс	Временная сетка, нс	Время действия операции, нс		Адрес	Число
			запись	чтение		
1	2,5	30	600	660	109_{16},	29_{16},
					1471_{16},	11_{16},
					$1F40_{16}$	$B0_{16}$
					$1FA9_{16}$	F_{16}
2	3	40	480	520	100_{16},	$B1_{16}$,
					$3E7_{16}$	27_{16}
					1761_{16},	$2C_{16}$,
					$18A1_{16}$	F_{16}
3	3,5	50	600	700	FF_{16},	2_{16},
					262_{16},	BD_{16},
					$177A_{16}$	$2B_{16}$
					$1B58_{16}$	EE_{16}
4	4	60	720	780	$BB8_{16}$	$3A8_{16}$
					$F96_{16}$	BA_{16}
					1359_{16}	511_{16}
					$1F3E_{16}$	D_{16}
5	5	80	880	960	$3E6_{16}$	4_{16}
					$3EA_{16}$	415_{16}
					$1B5A$	FE_{16}
					$1F9C_{16}$	$3D1_{16}$
6	6	100	1200	1400	3_{16}	BB_{16}
					$FA7_{16}$	15_{16}
					$137B_{16}$	C_{16}
					$1B4F_{16}$	$E5_{16}$

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Скриншоты основных этапов выполнения проекта.
4. Выводы.

Контрольные вопросы и задания

1. Что такое оперативное запоминающее устройство? Описать принцип его работы.

2. Что означает емкость ОЗУ 8к×16? Что означает 8к? Что означает 16?

3. Сколько чисел можно записать в ОЗУ емкостью 2к×8?

4. Какова разрядность числа, записываемого в ОЗУ емкостью 8к×4?

5. Что означает вход WRITE ENABLE (WE)?

6. Что означает вход READ ENABLE (RE)?

7. Как в рамках заданного семейства микросхем выбрать нужную микросхему для реализации проекта?

8. Как осуществляется настройка параметрического модуля? Опишите подробно все этапы настройки.

9. Опишите принцип наращивания размерности емкости оперативного запоминающего устройства в САПР MAX+PLUS II путем объединения двух параметрических модулей.

10. Как осуществляется группировка переменных при построении временных диаграмм?

11. Как осуществляется выбор системы счисления при представлении групповых данных?

12. Как назначается время окончания симуляции в редакторе временных диаграмм?

Список использованных источников

1. Система автоматизированного проектирования MAX+PLUS II [Электронный ресурс]. – 2022. –Режим доступа: https://intel.com/altera/.

2. Системы автоматизированного проектирования фирмы Altera MAX+plus II и Quartus II: Краткое описание и самоучитель / Д. А. Комолов [и др.]. – М. : ИП РадиоСофт, 2002. – 352 с.

3. Altera Nios II QiuckStart. [Электронный ресурс]. – 2022. – Режим доступа: http://we.easyelectronics.ru/plis/altera-nios-ii-qiuckstart-osvaivaem-principy-postroeniya-sistemy-i-infrastrukturu-sborki.html.

4. Altera MAX + PLUS II getting started. – 2022. – https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/_81_gs.pdf.