Содержание

ЛАБОРАТОРНАЯ РАБОТА 3

Описание логических схем при помощи языка AHDL

3.1 Цель работы

3.2 Основные теоретические сведения

3.3 Задание к выполнению лабораторной работы

3.4 Контрольные вопросы

ЛАБОРАТОРНАЯ РАБОТА 4

Моделирование цифровых схем с использованием параметрических элементов

4.1 Цель работы

4.2 Основные теоретические сведения

4.3 Задание к выполнению лабораторной работы

4.4 Контрольные вопросы

ЛАБОРАТОРНАЯ РАБОТА 5

Программирование лабораторного стенда УЛС ПЛИС

5.1 Цель работы

5.2 Основные теоретические сведения

5.3 Задание к выполнению лабораторной работы

5.4 Контрольные вопросы

ЛАБОРАТОРНАЯ РАБОТА 6

Разработка проекта сложного цифрового устройства на базе УЛС ПЛИС

Введение

Широкое внедрение автоматики во все сферы человеческой деятельности, наблюдаемое в настоящее время, предъявляет жесткие требования к изделиям электронной техники. Это связано, с одной стороны, с возрастанием важности и сложности решаемых задач, а, с другой стороны, необходимостью улучшения качественных характеристик, таких как быстродействие, надежность, потребляемая мощность, габариты, стоимость и др. Одним из путей решения данной проблемы является использование новой элементной базы - программируемых логических интегральных схем (ПЛИС — Programmable Logic Device — PLD) [1].

ПЛИС представляют собой интегральные схемы, обладающие гибкостью заказных БИС (больших интегральных схем) и доступностью традиционной "жесткой" логики. Современные ПЛИС характеризуются низкой стоимостью, высоким быстродействием, значительными функциональными возможностями, многократностью перепрограммирования, низкой потребляемой мощностью и др. Главным отличительным свойством ПЛИС является возможность их настройки на выполнение заданных функций самим пользователем, таким образом реализовывать даже достаточно сложные проекты в сжатые сроки в виде конкурентоспособных устройств и систем. По существу, разработка устройств на основе ПЛИС представляет собой новую технологию проектирования электронных схем, включая их изготовление и сопровождение.

Одним из доказательств перспективности рассматриваемой элементной базы служит ежегодное появление новых, имеющих более совершенную архитектуру, поколений ПЛИС, а также постоянно растущий объем выпуска ПЛИС таким производителями как ALTERA, Atmel, Xilinx, LATTICE и др., среди которых наиболее распространенными в нашей стране являются ПЛИС американской фирмы ALTERA.

В настоящее время фирма ALTERA выпускает такие семейства программируемых интегральных схем, как MAX 3000, MAX 7000, MAX 9000, FLEX 6000, FLEX 8000, FLEX 10K, APEX 20K, ACEX, Mercury и др. Основные характеристики наиболее популярных из них приведены в таблице 1.1.

Семейства MAX представляют собой электрически перепрограммируемые микросхемы с энергонезависимой прошивкой, допускающие 10^4 – 10^6 циклов программирования – стирания, в то время как микросхемы семейства FLEX изготовлены по технологии SRAM с загрузкой конфигурации при включении питания, что в свою очередь дает теоретически неограниченное число циклов программирования.

Семейство микросхем программируемой логики ACEX 1K [4] производится на основе SRAM технологии с напряжением питания 2,5V. Микросхемы выпускаются с логической емкостью от 10 000 до 100 000 эквивалентных вентилей. Это семейство ПЛИС выпускаются в корпусах TQFP с количеством выводов 100, 144, 208 и в корпусах BGA с 256 и 484 выводами. Работая при напряжении питания 2,5 V устройства ACEX 1K позволяют реализовывать полностью 64 битные устройства. Это семейство микросхем имеет встроенное ОЗУ емкостью от 12 до 48 Kbit.

Таблица 1.1

Семейства ПЛИС

Характеристики	Семейства ПЛИС			
	MAX 7000E(S)	MAX 9000	FLEX 8000A	FLEX 10K
Архитектура	матрицы И-ИЛИ	матрицы И-ИЛИ	таблицы перекодировки	таблицы перекодировки
Логическая емкость [1]	600-5000	6000-12000	2500-16000	10000-100000
Внутренняя память [2]	Нет	нет	Нет	6-24 Кбит
Число пользовательских выводов	36-164	60-216	68-208	59-406

[1] единица измерения - число эквивалентных логических вентилей (вентилей типа 2И-НЕ);

[2] при использовании внутренней памяти доступные пользователю логические ресурсы СБИС не уменьшаются.

ALTERA выпускает три типа микросхем семейства APEX с напряжением питания 2,5 V (APEX 20K) и 1,8 V (APEX 20KE и APEX 20KC). Логическая

емкость микросхем программируемой логики семейства APEX от 30 тысяч до 1,5 миллионов эквивалентных вентилей. Архитектура этих микросхем MultiCore основана на структуре нового поколения MegaLAB. Структура MegaLAB состоит из 16 логических блоков (LABs), которые в свою очередь состоят из 10 логических элементов, каждый из которых используется для реализации таблицы перекодировки (look-up-table – LUT), а также содержит усовершенствованные интегрированные блоки памяти объемом от 52 до 432 Kbit [5, 6]. Семейство APEX программируемых логических устройств обеспечивают гибкость и высокую логическую емкость, необходимую для разработки устройств типа систем на кристалле (system-on-a-programmable-chip – SOPC). Эти микросхемы позволяют разрабатывать устройства обработки 64 битных данных, и передачи данных с предельной скоростью передачи порядка 840 Mb/s, полностью удовлетворяют стандарту PCI.

Описание учебного лабораторного стенда УЛС ПЛИС

Общая характеристика лабораторного стенда УЛС ПЛИС Лабораторный стенд УЛС ПЛИС предназначен для практической реализации учебных проектов цифровых устройств на ПЛИС Altera.

Устройство стенда основано на микросхеме семейства **FLEX8000** (тип **EPF8282ALC84**). Лабораторный стенд состоит из основного блока, включающего элементы конфигурирования, элементы индикации, нефиксируемые кнопочные выключатели, переключатели, задающие на выводах микросхем входные логические уровни "0" и "1", разъемы расширения для подключения внешних устройств, конфигурационного устройства и блока питания.

Загрузка конфигурирующих данных в микросхему осуществляется через установленный на задней стенке стенда входной разъем интерфейса **JTAG**. Загрузка конфигурации из персонального компьютера осуществляется с помощью устройства **ByteBlaster.** Устройство **ByteBlaster** подключается к параллельному порту персонального компьютера и соединяется кабелем со входным разъемом **JTAG** лабораторного стенда.

Лабораторный стенд имеет внешний 5-ти вольтовый стабилизированный блок питания.

Основные элементы лабораторного стенда УЛС ПЛИС Схема расположения элементов на передней панели лабораторного стенда УЛС ПЛИС приведена на рис. 1.1.

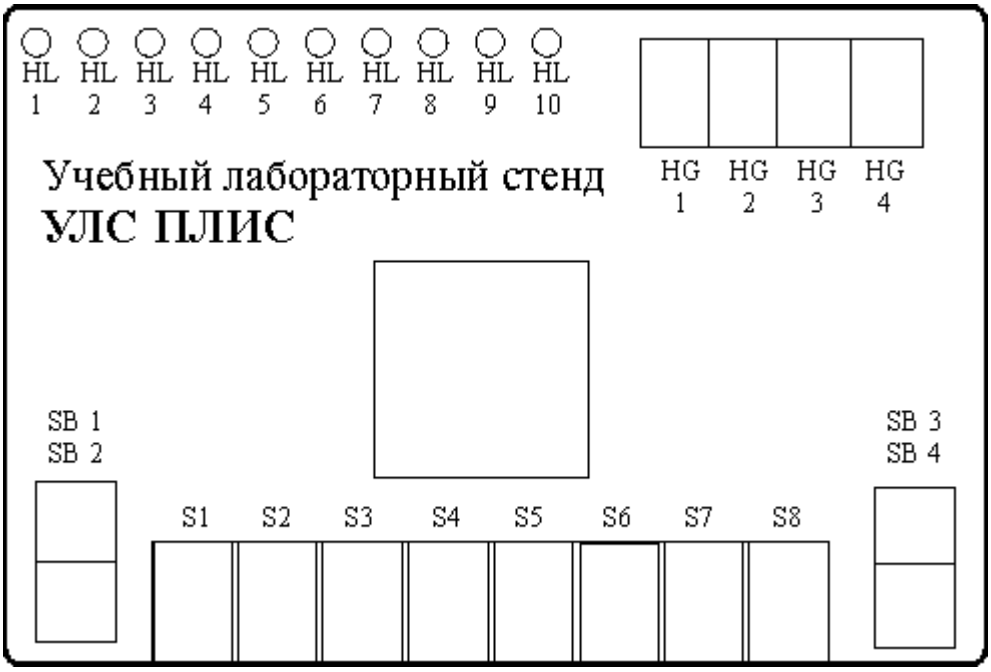


Рис. 1.1 – Схема расположения элементов лабораторного стенда УЛС ПЛИС

HL1 – HL10 – светодиодные индикаторы

HG1 – HG4 – семисегментные индикаторы

SB1 – SB4 – нефиксируемые кнопочные выключатели

S1 – S8 – переключатели, для задания на выводах микросхемы логических уровней "0" и "1"

На передней боковой панели расположены разъемы расширения X3 и X4 для подключения дополнительных внешних устройств.

На задней боковой панели расположены разъем питания X0 и разъем интерфейса JTAG.

Лабораторный стенд УЛС ПЛИС содержит кварцевый тактовый генератор с частотой 8.00 МГц. Выход генератора подключен к тактовому входу микросхем EPF8282ALC84 (вывод 50).

Разъем JTAG (X1) служит для подключения к лабораторному стенду кабеля загрузочного устройства ByteBlaster. Через разъем JTAG на ByteBlaster подается питание УЛС ПЛИС.

Таблица 1.2.

Контакты разъема JTAG

Контакт	**Сигнал JTAG**	**Назначение**
1	TCK	Тактовые импульсы
2	GND	Земля
3	TDO	Выход данных
4	VCC	+ Напряжения питания
5	TMS	Управление конечным автоматом JTAG
6	Не задействован	-
7	Не задействован	-
8	Не задействован	-
9	TDI	Вход данных

Микросхема EPF8282ALC84 установлена в режим конфигурирования Passive Serial (рис. 1.2)

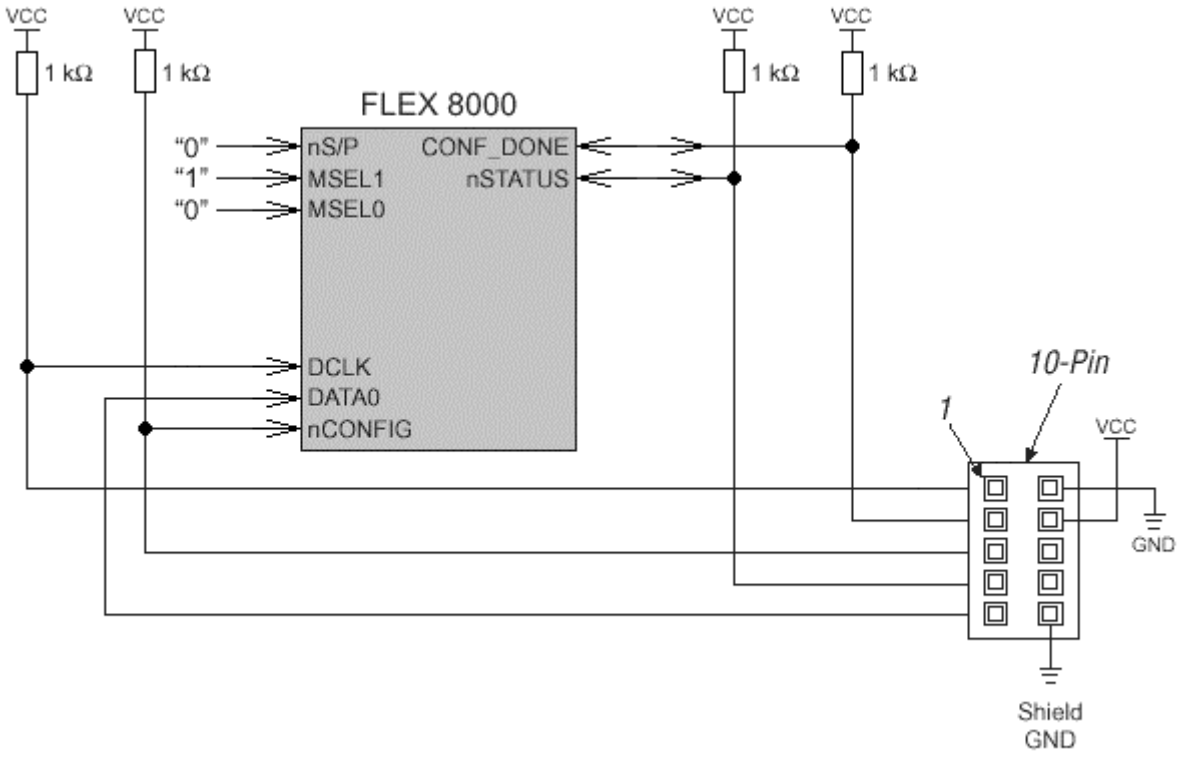


Рис. 1.2 – Режим конфигурирования Passive Serial

Микросхеме доступны следующие аппаратные ресурсы:

- Четыре нефиксируемых кнопочных переключателя
- 8 задающих переключателя уровней "0" и "1"
- 10 светодиодов
- Знакосинтезирующий индикатор, включающий четыре семисегментных знакоместа
- Выход тактового генератора (8.00 МГЦ)

Выходной сигнал тактового генератора подается на 50 вывод микросхемы EPF8282ALC84.

Подключение светодиодов представлено в таблице 1.3.

Таблица 1.3

Светодиод	**HL1**	**HL2**	**HL3**	**HL4**	**HL5**	**HL6**	**HL7**	**HL8**	**HL9**	**HL10**
Контакт микросхемы	82	81	46	45	44	43	42	41	84	83

Подключение семисегментных индикаторов представлено в таблице 1.4.

Таблица 1.4

Сегмент	**Контакты микросхемы, подключенные к HG1**	**Контакты микросхемы, подключенные к HG2**	**Контакты микросхемы, подключенные к HG3**	**Контакты микросхемы, подключенные к HG4**
A	70	61	6	16
B	76	63	3	2
C	71	62	4	15
D	69	60	7	18
E	66	57	9	21
F	67	58	8	19
G	65	56	13	22
H	78	64	51	1

Подключение нефисируемых кнопочных переключателей приведено в таблице 1.5

Таблица 1.5.

Кнопочный переключатель	**SB1**	**SB2**	**SB3**	**SB4**
Контакт микросхемы	31	12	39	40

Подключение задающих переключателей уровня “0” и “1” приведено в таблице 1.6.

Таблица 1.6.

Переключатель	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
Контакт микросхемы	24	23	34	25	73	54	37	35

Лабораторная работа №1

Система автоматизированного проектирования MAX+PLUS II

САПР MAX+PLUS II представляет собой интегрированную среду для разработки цифровых устройств на базе программируемых логических интегральных схем (ПЛИС) фирмы ALTERA и обеспечивает выполнение всех этапов, необходимых для выпуска готовых изделий:

- создание проектов устройств;
- синтез структур и трассировку внутренних связей ПЛИС;
- подготовку данных для программирования или конфигурирования ПЛИС (компиляцию);
- верификацию проектов (функциональное моделирование и временной анализ);
- программирование или конфигурирование ПЛИС.

Ниже представлено главное окно программы (рис. 1.1), оно имеет стандартный интерфейс Windows-приложений. В заголовке окна программы указывается имя и путь последнего проекта, с которым велась работа.

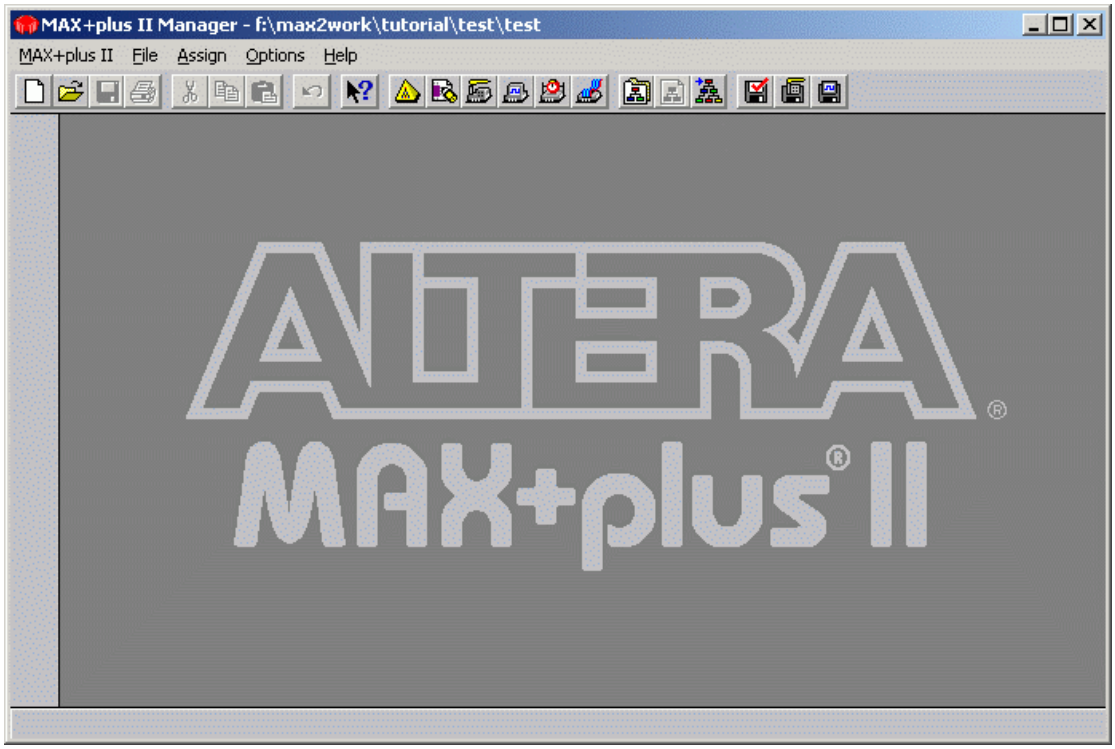


Рис. 1.1 Главное окно системы MAX+plus II

Приложения системы MAX+PLUS II. В состав пакета MAX+PLUS II входят следующие связанные между собой приложения, реализующие все этапы разработки цифровых устройств на ПЛИС фирмы ALTERA:

Приложения для ввода проектов (редакторы проектов):

Graphic Editor – графический редактор (рис. 1.2), предназначен для ввода проекта в виде схемы соединений символов элементов, извлекаемых из стандартных библиотек пакета либо из библиотеки пользователя.

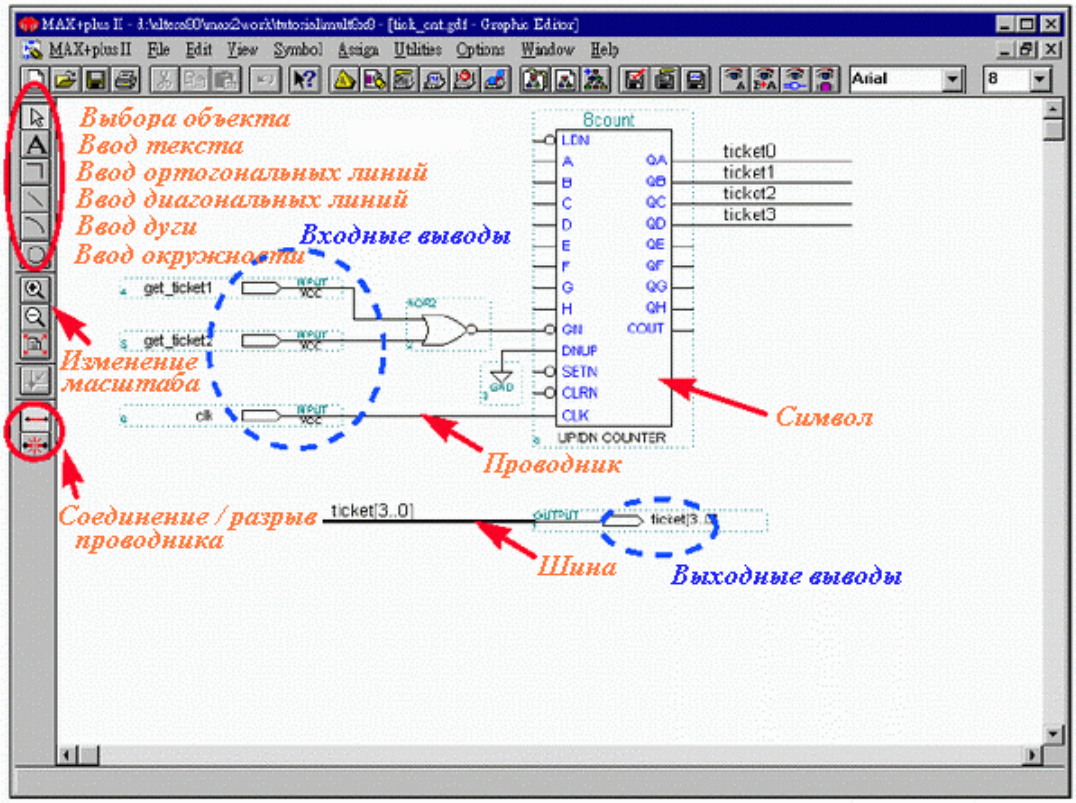


Рис. 1.2 Графический редактор.

Вставка символа производится двойным кликом левой кнопки мыши на свободном месте окна графического редактора. Введенные символы и группы символов можно копировать, удалять, поворачивать, перетаскивать в другую область окна обычным приемом “Drag&Drop”, а так же обмениваться с другими окнами через буфер обмена. Выводы элементов можно соединять сигнальными проводниками либо присваиванием одинаковых имен проводникам которые должны быть соединены.

Waveform Editor – редактор временных диаграмм (некоторые разработчики называют это приложение сигнальным редактором), который выполняет двойную

функцию: на этапе ввода обеспечивает ввод логики проекта в виде диаграмм (эпюр) состояний входов и выходов, а на этапе моделирования обеспечивает ввод диаграмм тестовых (эталонных) входных состояний моделируемого устройства и задание перечня тестируемых выходов. Окно сигнального редактора представлено на рис. 1.3

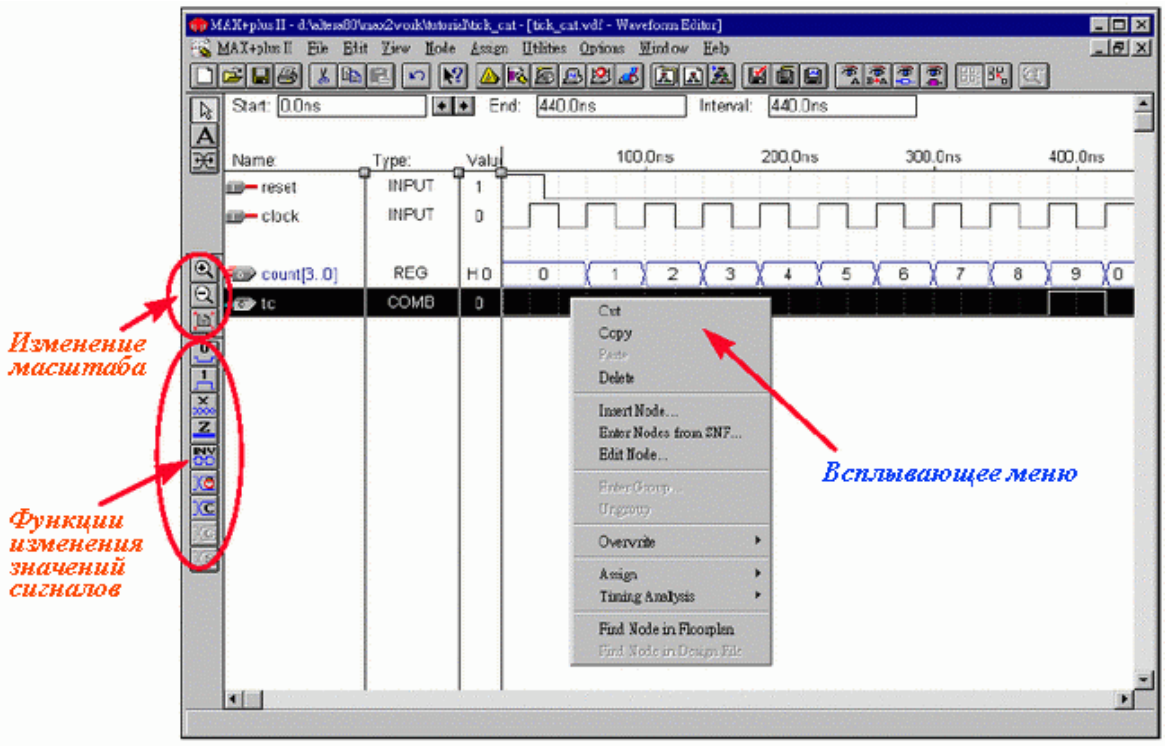


Рис. 1.3 – Редактор временных диаграмм.

Время в течении которого будет проводиться моделирование задается в меню “File”, пункт “End Time”, шаг временной сетки задается в меню “Options”, пункт “Grid Size”.

Окно сигнального редактора имеет четыре поля, разделённых вертикальными линиями. Первое поле слева, (“Name”) предназначено для ввода имени вывода, во втором поле (“Type”) отображается тип вывода (INPUT, OUTPUT, BIDIR), в третьем поле “Value” показаны состояния выводов, соответствующие положению специальной вертикальной визирной линии. Четвёртое поле предназначено для задания требуемых состояний выводов, при этом используются инструменты с панели инструментов редактора, которая

расположена вертикально вдоль левой стороны окна. Активизация панели инструментов происходит только в том случае, если выделен один из узлов. Чтобы выделить узел, необходимо щёлкнуть левой кнопкой мыши на имени узла, можно также выделить любой участок вдоль горизонтальной оси, при этом границы выделяемых участков привязываются к сетке.

Размещаются выводы при помощи всплывающего меню (рис. 1.3), пункт “Insert Node”. Введенные выводы можно редактировать, перемещать, удалять, размножать (с обязательным редактированием имени или типа, если это необходимо).

Text Editor – текстовый редактор (рис. 1.4) является инструментом для создания текстовых файлов проекта на языках описания аппаратуры: AHDL (*.tdf*), VHDL (*.vhd*), Verilog HDL (*.v*). В этом текстовом редакторе можно работать также с произвольным файлом формата ASCII.

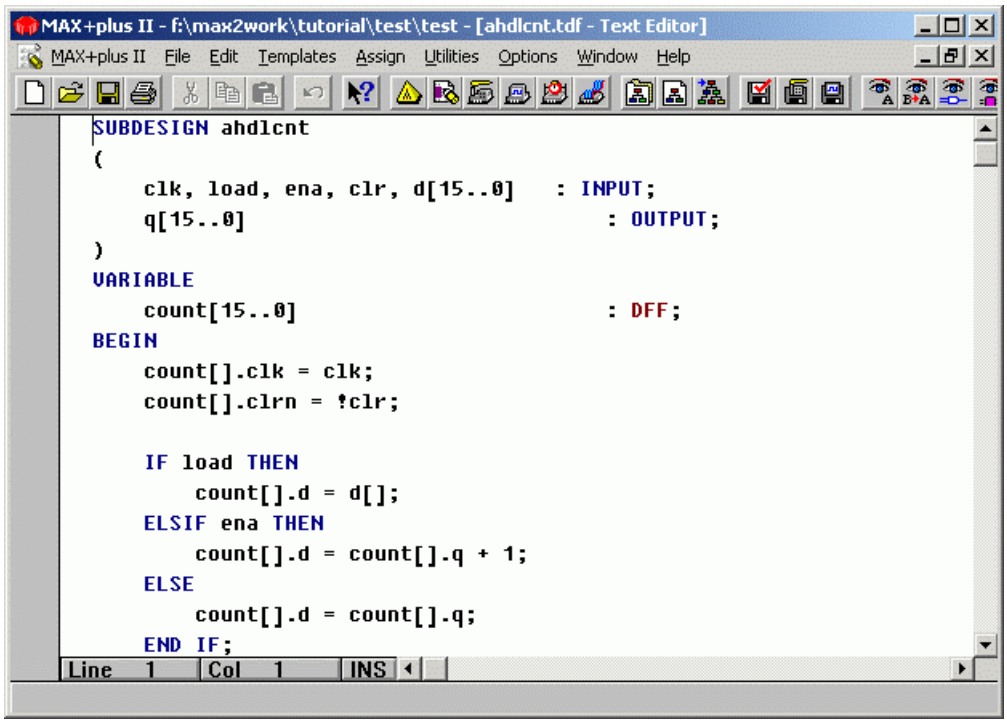


Рис. 1.4 Окно текстового редактора.

Все перечисленные файлы проекта можно создавать в любом текстовом редакторе, однако данный редактор имеет встроенные возможности ввода файлов проекта, их компиляции и отладки с выдачей сообщений об ошибках и их

локализацией в исходном тексте или в тексте вспомогательных файлов. Кроме того, существуют шаблоны языковых конструкций для AHDL, VHDL и Verilog HDL, выполнено окрашивание синтаксических конструкций. В данном редакторе можно вручную редактировать файлы назначений и конфигурации (*.acf*), а также делать установки конфигурации для компилятора, симулятора и временного анализатора.

Symbol Editor – символьный редактор позволяет редактировать существующие символы и создавать новые. Любой откомпилированный проект может быть свёрнут в символ, помещён в библиотеку символов и использован как элемент в любом другом проекте. Окно символьного редактора представлено на рис. 1.5.

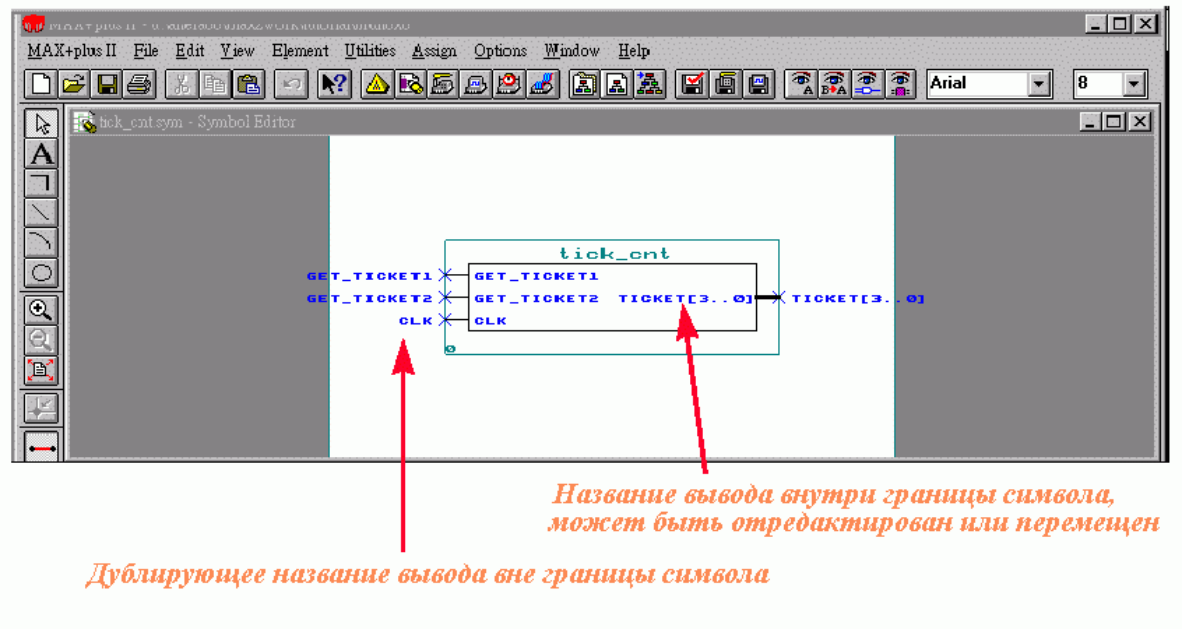


Рис. 1.5 Символьный редактор

Floorplan Editor – редактор связей, позволяет на плане расположения основных логических элементов вручную распределять выводы ПЛИС (закреплять выводы за конкретными входными и выходными сигналами) и перераспределять внутренние ресурсы ПЛИС. Окно редактора представлено на рис. 1.6.

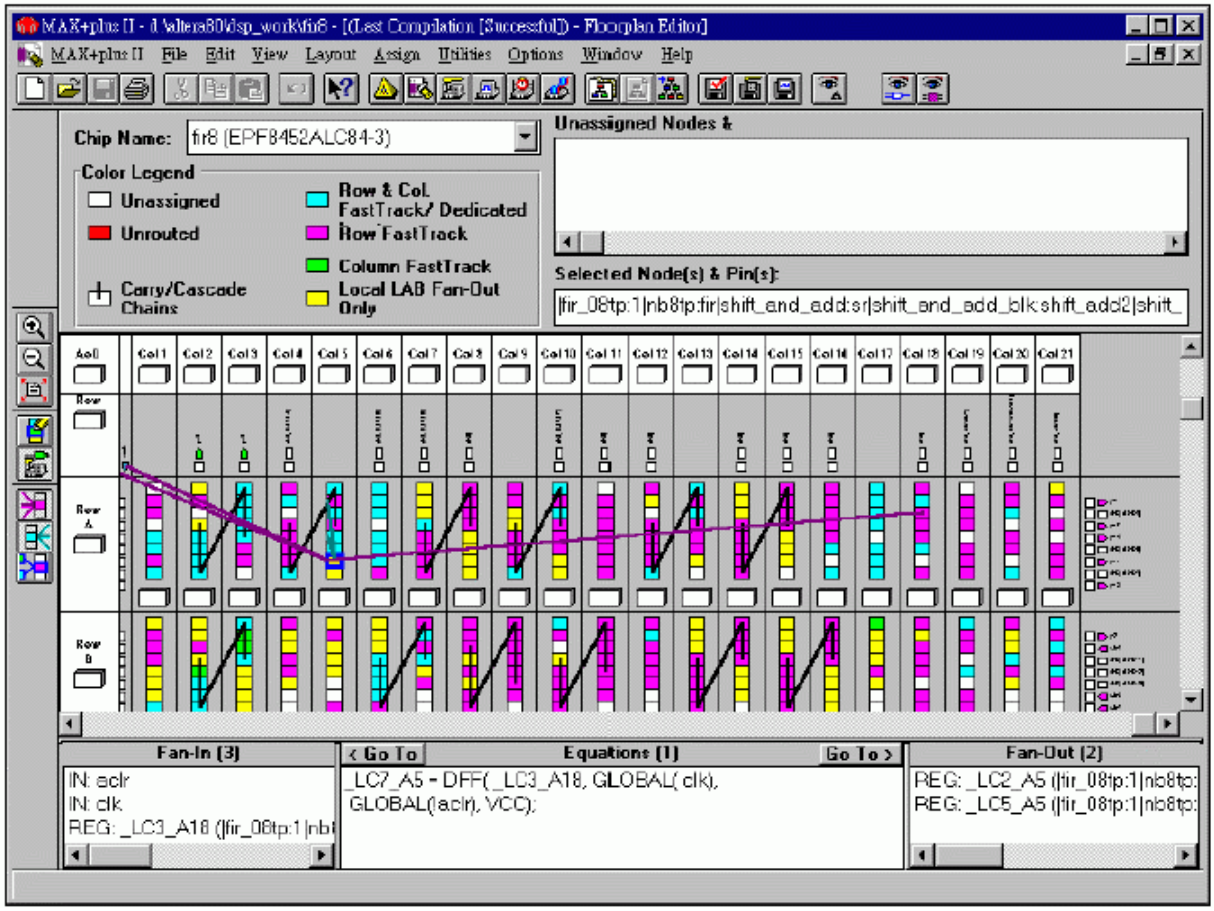


Рис. 1.6 Редактор связей системы MAX+PLUS II

Приложения MAX+PLUS II Compiler

Это приложения, входящие в пакет компилятора и предназначенные для синтеза структуры, трассировки связей, проверки корректности проекта и локализации ошибок, формирования файлов программирования или конфигурирования ПЛИС:

Netlist Extractor – приложение, обеспечивающее извлечение списка соединений из исходного файла представления проекта, созданного при вводе проекта.

Database Builder – приложение, предназначенное для построения базы данных проекта.

Logic Synthesizer – приложение, обеспечивающее проверку корректности проекта по формальным правилам и синтез оптимальной структуры проекта.

Practitioner – приложение, обеспечивающее разбиение проекта на части в тех случаях, когда ресурсов одного кристалла (микросхемы) недостаточно для реализации проекта.

Fitter – трассировщик внутренних связей, обеспечивающий реализацию синтезированной структуры.

SNF Extractor – приложение, обеспечивающее извлечение параметров проекта, необходимых для функционального моделирования и временного анализа.

Приложения для верификации проектов

Simulator – приложение, которое совместно с редактором временных диаграмм предназначено для функционального моделирования проекта (рис. 1.7) с целью проверки правильности логики его функционирования.

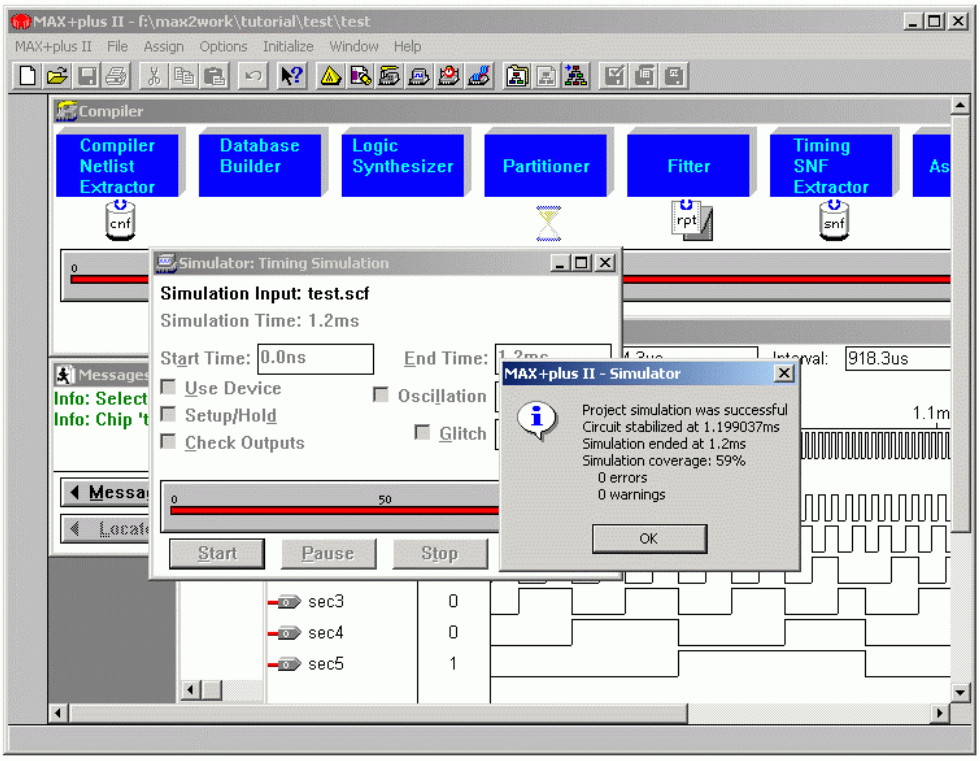


Рис. 1.7 Функциональное моделирование в системе MAX+PLUS II

Timing Analyzer – приложение, обеспечивающее расчет временных задержек от каждого входа до каждого логически связанного с ним выхода.

Наконец, для программирования или конфигурирования ПЛИС используется приложение MAX+PLUS II Programmer (рис. 1.8)

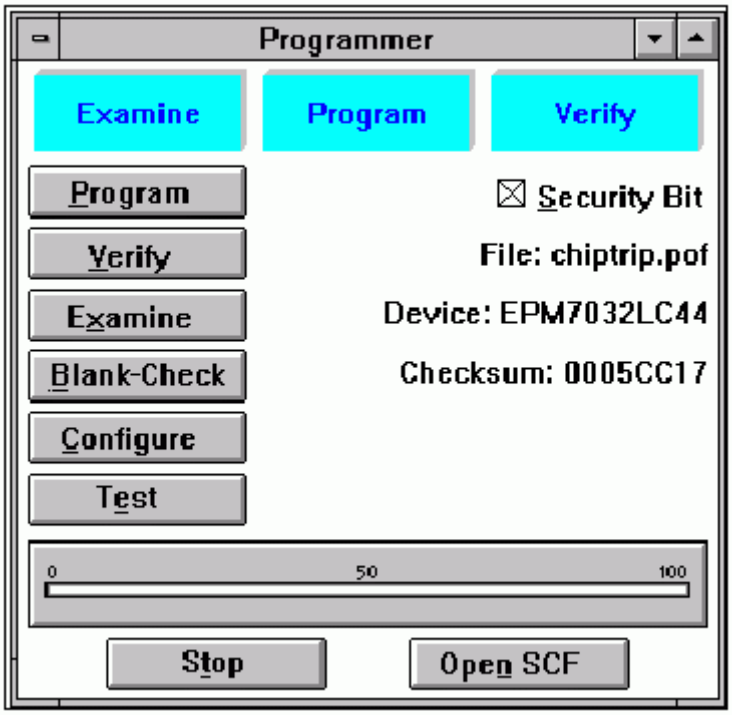


Рис. 1.8 – Программатор системы MAX+PLUS II

Программирование и перепрограммирование микросхем, имеющих встроенную систему программирования (ISP), может осуществляться непосредственно в составе конечного изделия через специальный кабель, подключаемый либо к LPT-порту (ByteBlaster), либо к COM-порту (BitBlaster) компьютера и технологического 10-контактного соединителя интерфейса JTAG, устанавливаемого на плате изделия. Если на плате изделия устанавливается несколько ПЛИС со встроенными системами программирования, то все они могут программироваться через один технологический разъём. Для этой цели приложение “Programmer” имеет режим “Multi-Device” (к сожалению, бесплатные версии пакета этот режим не поддерживают). Для программирования остальных микросхем необходимо дополнительно использовать внешний программатор, который также может подключаться к COM- или LPT-порту.

Сервисные приложения

В состав САПР MAX+PLUS II, кроме того, входят три сервисных приложения:

Design Doctor – приложение, предназначенное для проверки корректности проекта с использованием эмпирических правил.

Message Processor – процессор сообщений (рис. 1.9), обеспечивающий обработку, вывод на отображение и локализацию (указание места в проекте,

Рис. 1.9 Окно сообщений системы

к которому оно относится) сообщений трёх типов: сообщений об ошибках “Error”, предупреждений “Warning” и информационных сообщений “Info”. Причину вывода того или иного сообщения можно выяснить через опцию “Help on Message” процессора сообщений. При наличии сообщений об ошибках компиляция проекта невозможна до их полного устранения. При наличии предупреждений компиляция успешно завершается, однако наличие предупреждения свидетельствует об обнаружении проблемы, которая может привести к неверной работе устройства. Поэтому все предупреждения должны быть тщательно проанализированы с использованием “Help on Message”, до выяснения причин их появления и последующего устранения этих причин (или игнорирования предупреждения, что иногда бывает возможно). Информационные сообщения нужно только принимать к сведению.

Hierarchy Display – приложение, обеспечивающее обзор иерархической структуры проекта, который может состоять из множества составленных в различных редакторах и свёрнутых в символы проектов более низких уровней, причём число уровней не ограничивается.

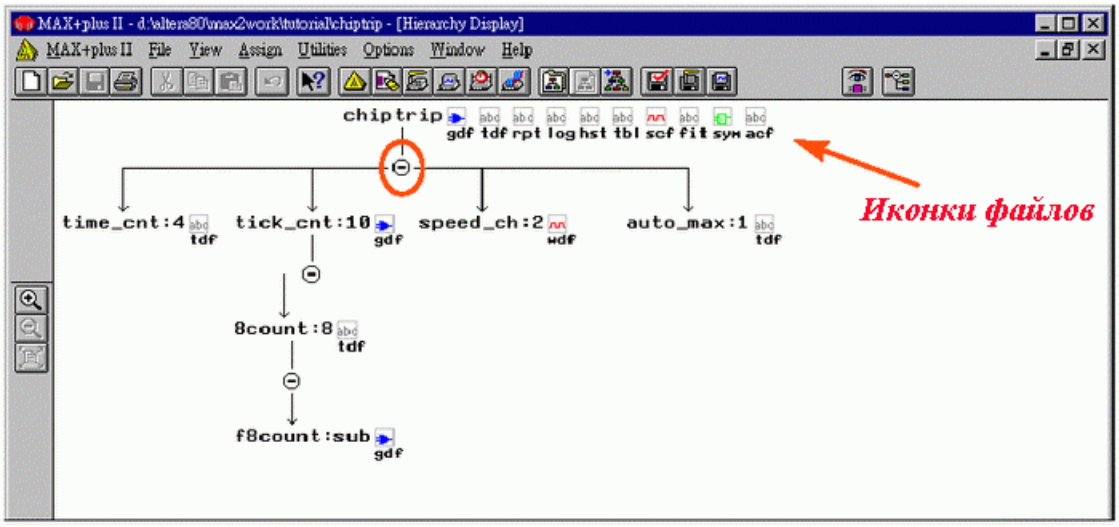


Рис. 1.10 Окно иерархии проекта

Основной проект (проект самого верхнего уровня) должен быть создан в графическом редакторе, если проект имеет только один уровень иерархии, то он может быть создан в любом редакторе.

Рабочие каталоги системы

Во время инсталляции пакета создаются два каталога: каталог \MAXPLUS2, который содержит все приложения и библиотеки пакета, и каталог \MAX2WORK, который содержит подкаталог \CHIPTRIP со всеми файлами учебного проекта, и ряд подкаталогов, используемых электронным справочником MAX+PLUS II Help. В этом же каталоге \MAX2WORK следует размещать и рабочие каталоги создаваемых проектов устройств.

Необходимость создания отдельных каталогов для каждого разрабатываемого проекта обусловлена тем, что в процессе разработки проекта системой MAX+PLUS II создаётся и поддерживается множество файлов, относящихся к текущему проекту. Прежде всего это файл проекта (Project File), название которого определяет название проекта в целом. Этот файл содержит основную логику и иерархию проекта, обрабатываемую компилятором. Кроме того, создаётся ряд вспомогательных файлов, связанных с проектом, но не являющихся частью логики проекта. Большая часть вспомогательных файлов создаётся и автоматически помещается в каталог проекта в процессе ввода и

компиляции проекта. Это прежде всего файлы назначений и конфигурации (.ACF), файлы отчётов (.RPT), файлы данных для функционального моделирования и временного анализа (.SNF), файлы данных для программирования (.POF) и ряд других. Названия этих файлов всегда совпадают с названием проекта. Некоторые вспомогательные файлы создаются пользователем: например, для выполнения функционального моделирования создаётся файл (.SCF), содержащий описание начальных и текущих состояний входных сигналов (входов) и перечень выходов, для которых должны быть определены выходные сигналы. Поэтому перед началом работы над новым проектом следует создать рабочий каталог проекта, при этом имя каталога можно выбирать произвольно (т.е. имя каталога может не совпадать с именем файла проекта).

Лабораторная работа № 2

Графический ввод схемы и симуляция в САПР MAX+PLUS II

Цель работы:

Спроектировать логическую схему при помощи графического редактора САПР MAX+PLUS II. Исследовать работу схемы с использованием сигнального редактора САПР MAX+PLUS II.

Основные теоретические сведения:

Алгебра логики и основные логические элементы. Математической основой цифровой электроники и вычислительной техники является алгебра логики или булева алгебра (по имени английского математика Джона Буля).

В булевой алгебре независимые переменные или аргументы (X) принимают только два значения: «0» или «1». Зависимые переменные или функции (Y) также могут принимать только два значения: «0» или «1». Функция алгебры логики (ФАЛ) представляется в виде:

$$Y = F(X_1; X_2; X_3 ... X_N).$$

Данная форма задания ФАЛ называется алгебраической.

Основными логическими функциями являются:

- логическое отрицание (инверсия):

$$Y = \overline{X};$$

- логическое сложение (дизъюнкция):

$$Y = X_1 + X_2 \quad \text{или} \quad Y = X_1 \vee X_2;$$

- логическое умножение (конъюнкция):

$$Y = X_1 \bullet X_2 \quad \text{или} \quad Y = X_1 \wedge X_2.$$

К более сложным функциям алгебры логики относятся:

- функция равнозначности (эквивалентности):

$$Y = X_1 \bullet X_2 + \overline{X_1} \bullet \overline{X_2} \quad \text{или} \quad Y = X_1 \sim X_2;$$

- функция неравнозначности (сложение по модулю два):

$$Y = X_1 \bullet \overline{X_2} + \overline{X_1} \bullet X_2 \quad \text{или} \quad Y = X_1 \oplus X_2;$$

- функция Пирса (логическое сложение с отрицанием):

$$Y = \overline{X_1 + X_2};$$

- функция Шеффера (логическое умножение с отрицанием):

$$Y = \overline{X_1 \bullet X_2};$$

Для булевой алгебры справедливы следующие законы и правила:

- распределительный закон:

$$X_1(X_2 + X_3) = X_1 \bullet X_2 + X_1 \bullet X_3,$$
$$X_1 + X_2 \bullet X_3 = (X_1 + X_2)(X_1 + X_3);$$

- правило повторения:

$$X \bullet X = X, \quad X + X = X;$$

- правило отрицания:

$$X \bullet \overline{X} = 0, \quad X + \overline{X} = 1;$$

- теорема де Моргана:

$$\overline{X_1 + X_2} = \overline{X_1} \bullet \overline{X_2}, \quad \overline{X_1 \bullet X_2} = \overline{X_1} + \overline{X_2};$$

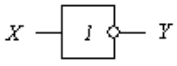
- тождественности:

$$X \bullet 1 = X, \; X + 0 = X, \; X \bullet 0 = 0, \; X + 1 = 1.$$

Схемы, реализующие логические функции, называются логическими элементами. Основные логические элементы имеют, как правило, один выход (Y) и несколько входов, число которых равно числу аргументов ($X_1;X_2;X_3$ ... X_N). На электрических схемах логические элементы рисуют в виде прямоугольников с выводами для входных (слева) и выходных (справа) переменных. В средине прямоугольника изображается символ, обозначающий функциональное назначение элемента.

На рисунках 1 - 7 показаны логические элементы, реализующие рассмотренные ранее функции. Там же приведены так называемые таблицы состояний или таблицы истинности, которые описывают соответствующие логические функции в двоичном коде в виде состояний входных и выходных переменных. Таблица истинности является также табличным способом задания ФАЛ.

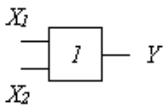
На рисунке 2.1 показан элемент «НЕ», который реализует функцию логического отрицания $Y = \overline{X}$.



X	Y
1	0
0	1

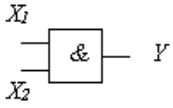
Рис. 2.1 Элемент «НЕ»

Элемент «ИЛИ» (рисунок 2.2) и элемент «И» (рисунок 2.3) реализуют функции логического сложения і логического умножения соответственно.



X1	X2	Y
0	0	0
1	0	1
0	1	1
1	1	1

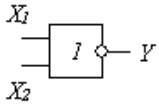
Рис. 2.2. Элемент «ИЛИ»



X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

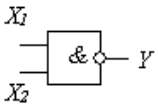
Рис. 2.3. Элемент «И»

Функции Пирса и функции Шеффера реализуются при помощи элементов «ИЛИ-НЕ» и «И-НЕ», приведенных на рисунках 4 и 5 соответственно.



X1	X2	Y
0	0	1
1	0	0
0	1	0
1	1	0

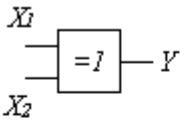
Рис. 2.4. Элемент «ИЛИ-НЕ»



X1	X2	Y
0	0	1
0	1	1
1	0	1
1	1	0

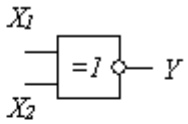
Рис. 2.5 Элемент «И-НЕ»

Элемент Пирса можно получить последовательным соединением логических элементов «ИЛИ» и элемента «НЕ», а элемент Шеффера - в виде последовательного соединения логических элементов «И» та «НЕ». На рисунках 1.6 и 1.7 показаны элементы «отрицающее ИЛИ» та «отрицающее ИЛИ-НЕ», которые реализуют функции неравнозначности и неравнозначности с отрицанием соответственно.



X1	X2	Y
0	0	0
1	0	1
0	1	1
1	1	0

Рис. 2.6 Элемент «отрицающее ИЛИ»



X1	X2	Y
0	0	1
0	1	0
1	0	0
1	1	1

Рис. 2.7 элемент «отрицающее ИЛИ-НЕ»

Логические элементы, которые реализуют операции конъюнкции, дизъюнкции, функции Пирса и Шеффера, могут быть, в общем случае, **n**-входовыми. В таблице истинности такого элемента количество возможных комбинаций входных переменных *N*, в общем случае равняется: $N = 2^n$, де n - число входных переменных.

Логические элементы используются для построения интегральных микросхем, которые выполняют разнообразные логические и арифметические операции.

ФАЛ любой сложности можно реализовать при помощи обозначенных логических элементов. В качестве примера рассмотрим ФАЛ, заданную в алгебраической форме, в виде:

$$Y = X_1 + \overline{X_2} X_3$$

Для реализации заданной функции на элементах «И-НЕ» ее представить в базисе «И-НЕ», используя двойную инверсию функции і теорему де Моргана

$$Y = \overline{\overline{Y}} = \overline{\overline{X_1 + \overline{X_2} X_3}} = \overline{\overline{X_1}\,(\overline{X_2}\,\overline{X_3})}$$

Реализация проекта цифровой схемы в графическом редакторе САПР MAX+PLUS. Рассмотрим работу с графическим редактором САПР MAX+PLUS II на примере схемы, представленной на рис. 2.1.

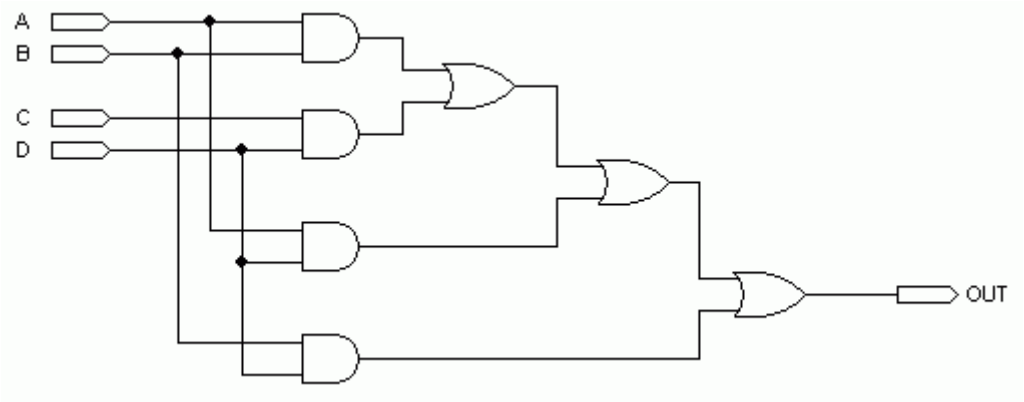


Рис. 2.8 – Пример цифровой схемы

Запустив САПР MAX+PLUS II, при помощи меню File/New… создаем файл графического редактора (рис. 2.9).

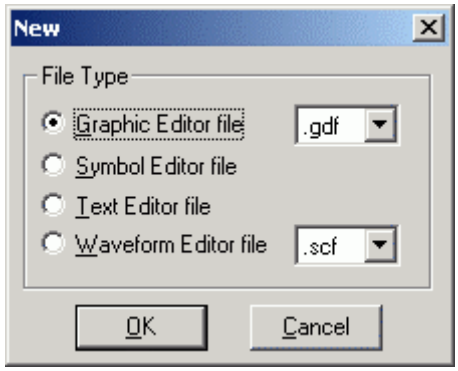


Рис. 2.9 – Меню File/New…

В созданный файл вводим схему лабораторной работы. Для ввода элементов схемы воспользуемся меню Enter Symbol (ввод символа), которое вызывается двойным щелчком левой кнопки мыши. В открывшемся окне (рис. 2.10) выбираем необходимую библиотеку примитивов (Symbol Libraries:), выбираем нужный элемент (Symbol Files:).

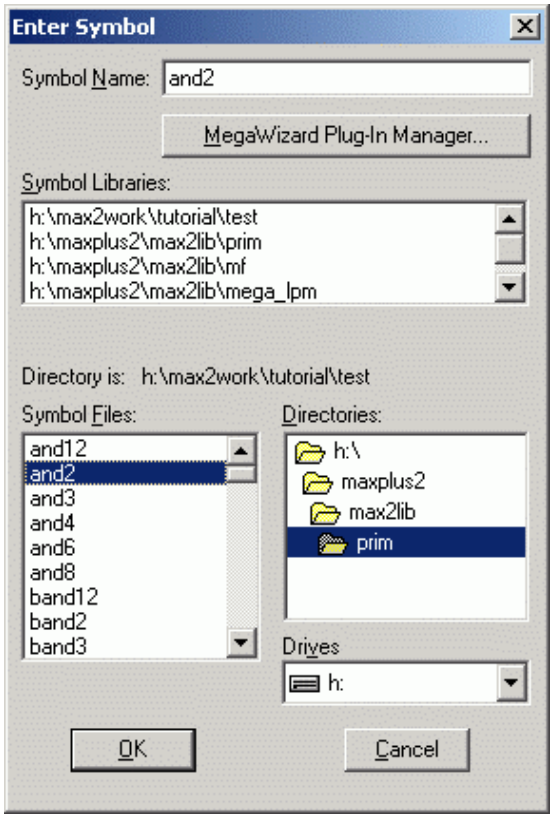


Рис. 2.10 – Меню Enter Symbol

После окончания ввода схемы сохраняем файл в предварительно созданную средствами Windows папку разрабатываемого проекта (например: \Lab_2), в данном случае лабораторной работы, в рабочем каталоге MAX+PLUS II: C:\max2work. Через меню File/Save As… (Рис. 2.11) сохраняем схему под выбранным именем (например: lab_2), при этом расширение присваивается автоматически.

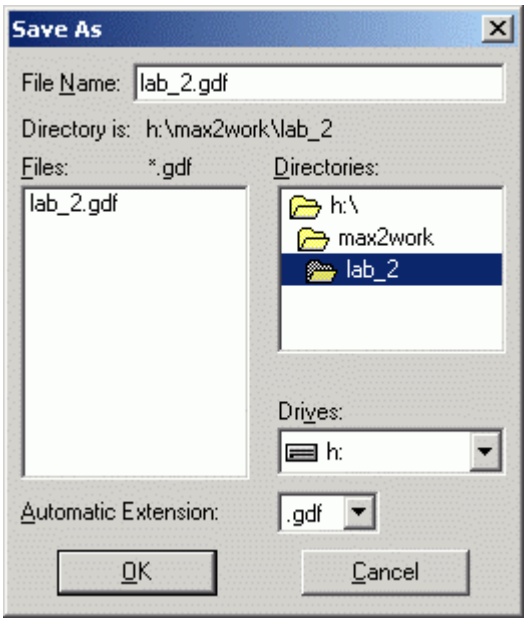


Рис. 2.11 – Меню Save As…

После сохранения необходимо имя файла привязать к имени проекта - это делается при выборе пункта Set Project to Current File в подменю Project меню File главного меню рабочего окна.

Проводим проверку введенной схемы. Для этого нажимаем пиктограмму Save & Check. При отсутствии ошибок ввода проекта, его можно компилировать. Открываем окно компилятора (рис. 2.12) и нажимаем кнопку START. Если не был назначен тип ПЛИС для компиляции проекта то система сама назначает подходящий тип микросхемы, и оповещает пользователя в окне Messages – Compiler (рис. 2.13).

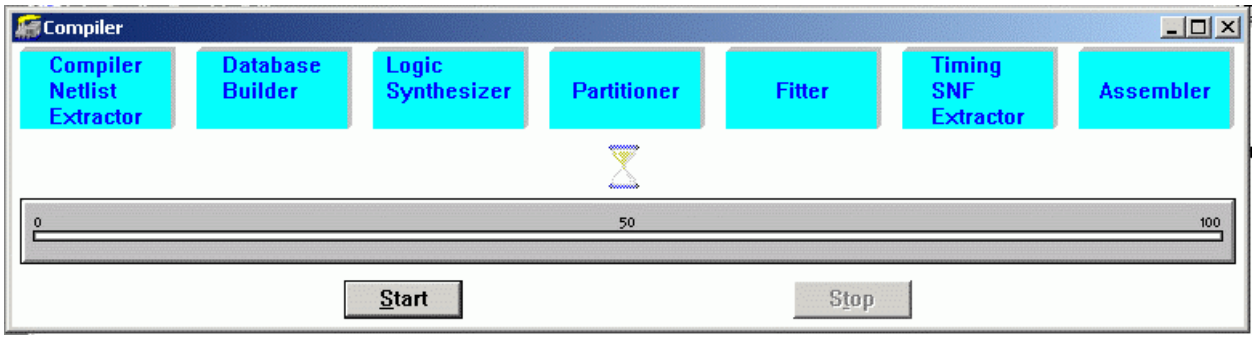


Рис. 2.12 – Окно компилятора

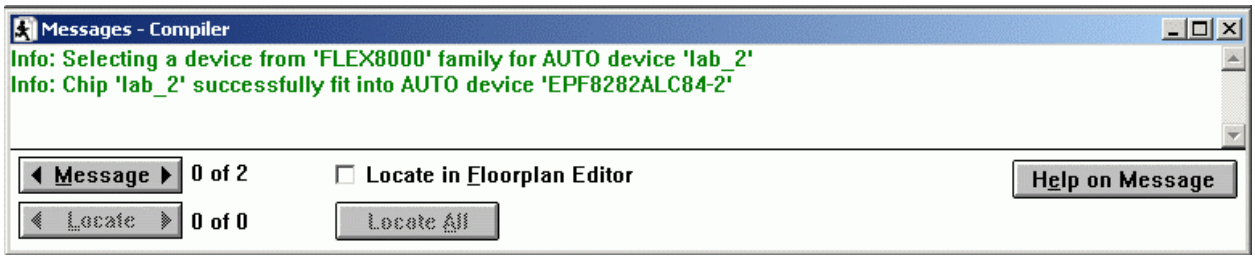


Рис. 2.13 – Окно сообщений компилятора

Проверку работы схемы делаем с помощью сигнального редактора (Waveform Editor). Для этого открываем сигнальный редактор и создаем в нем файл с расширением .scf. В созданном файле при помощи меню File/End Time… задаем время моделирования, а в меню Options/Grid Size… шаг сетки моделирования. Далее двойным щелчком правой кнопки мыши на поле Name: вызываем меню Insert Node (рис. 2.14), при помощи которого

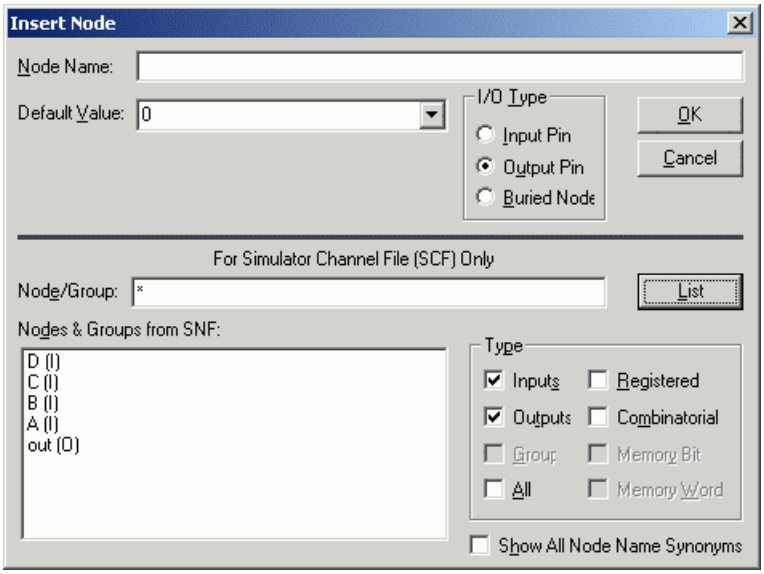


Рис. 2.14 – Меню Insert Node

выбираем входы и выход схемы. Для входных выводов задаем их значения на протяжении необходимого времени моделирования. После того как входные значения заданны открываем окно симулятора (Simulator Window) (рис. 2.15) и запускаем его нажав кнопку START.

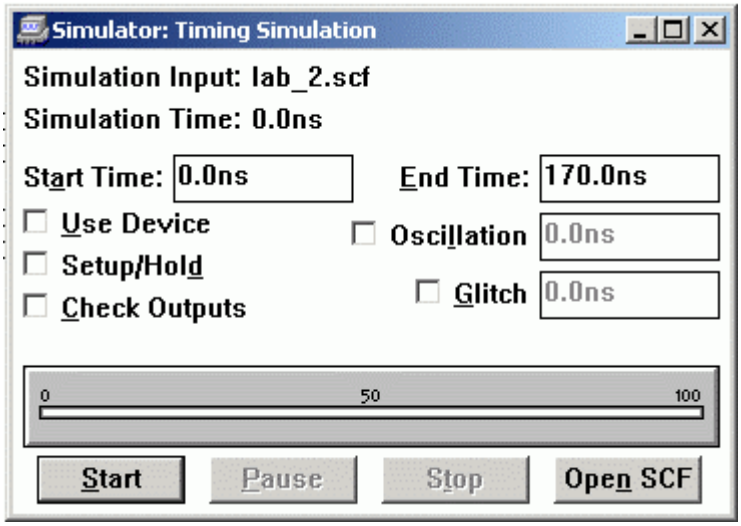


Рис. 2.15 – Окно симулятора (Simulator Window)

Результаты моделирования работы схемы лабораторной работы представлены на рис. 2.16.

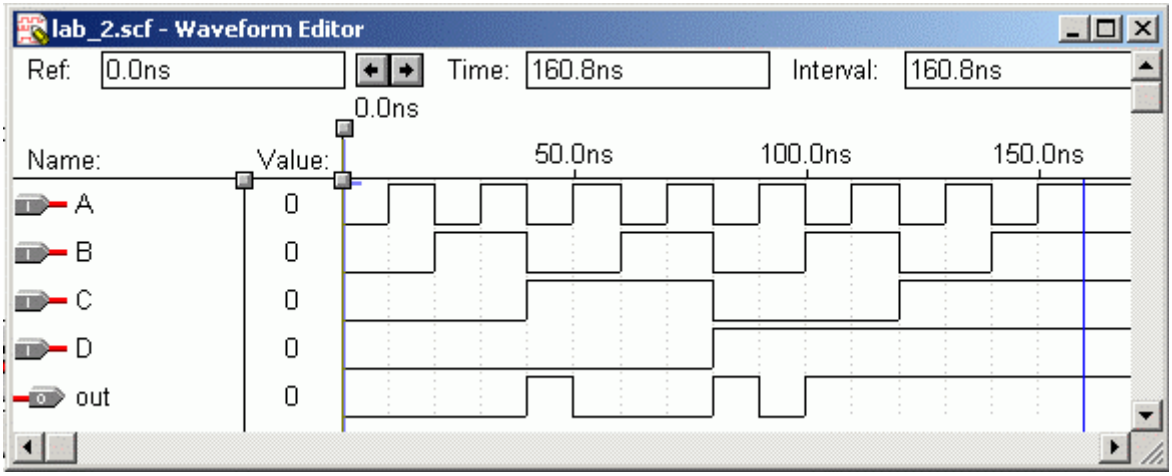


Рис. 2.16 – Результаты моделирования работы схемы в сигнальном редакторе

Задание к выполнению лабораторной работы:

1. Изучить правила построения, принцип работы логических схем.
2. Синтезировать электрическую принципиальную схему логического устройства, описанного заданным преподавателем уравнением в алгебраической форме.
3. Нарисовать синтезированную схему в графическом редакторе САПР MAX+PLUS II.
4. Произвести симуляцию работы схемы. Зарисовать диаграммы работы и по ее результатам заполнить таблицу истинности смоделированной схемы.
5. Ответить на контрольные вопросы, оформить отчет о выполненной работе.

Контрольные вопросы.

1. Назовите основные логические (булевы) функции и изобразите элементы их реализующие. Для каждой из функции запишите таблицу истинности.
2. Какие логические элементы доступны в библиотеке примитивов графического редактора MAX+PLUS II?
3. Какие процессы протекают в системе при компиляции проекта?
4. Объясните результаты моделирования работы схемы лабораторной работы.

Лабораторная работа №3

Описание логических схем при помощи языка AHDL

Цель работы:

Приобретение основных навыков описания цифровых схем с помощью языка описания аппаратуры AHDL. Смоделировать логическую схему при помощи текстового редактора САПР MAX+PLUS II.

Основные теоретические сведения:

Язык описания аппаратуры AHDL разработан фирмой Altera и предназначен для описания комбинационных и последовательностных логических устройств, групповых операций, цифровых автоматов (state machine) и таблиц истинности с учетом архитектурных особенностей ПЛИС фирмы Altera. Он полностью интегрируется с системой автоматизированного проектирования ПЛИС MAX+PLUS II. Файлы описания аппаратуры, написанные на языке AHDL, имеют расширение *.TDF (Text design file). Для создания TDF-файла можно использовать как текстовый редактор системы MAX+PLUS II, так и любой другой. Проект, выполненный в виде TDF-файла, компилируется, отлаживается и используется для формирования файла программирования или загрузки ПЛИС фирмы Altera.

Операторы и элементы языка AHDL являются достаточно мощным и универсальным средством описания алгоритмов функционирования цифровых устройств, удобным в использовании. Язык описания аппаратуры AHDL дает возможность создавать иерархические проекты в рамках одного этого языка или же в иерархическом проекте использовать как TDF-файлы, разработанные на языке AHDL, так и другие типы файлов.

При распределении ресурсов устройств разработчик может пользоваться командами текстового редактора или операторами языка AHDL для того, чтобы сделать назначения ресурсов и устройств. Кроме того, разработчик может только проверить синтаксис или выполнить полную компиляцию для отладки и запуска проекта. Любые ошибки автоматически обнаруживаются обработчиком сообщений и высвечиваются в окне текстового редактора.

Элементы языка AHDL. Зарезервированные ключевые слова.

Зарезервированные ключевые слова используются для следующих целей:

- для обозначения начала, конца и переходов в объявлениях языка AHDL;
- для обозначения предопределенных констант, т.е. GND и VCC.

Ключевые слова можно использовать, как символические имена, только если они заключены в символы одинарных кавычках ('). Их можно также использовать в комментариях.

Для того чтобы получить контекстовую помощь по ключевому слову, необходимо убедиться, что файл сохранен с расширением .tdf, затем нажать одновременно две кнопки Shift+F1 в окне текстового редактора Text Editor и щелкнуть кнопкой мыши Button 1 на ключевом слове.

Altera рекомендует все ключевые слова набирать прописными буквами.

Список всех зарезервированных ключевых слов языка AHDL приведен в таблице 3.1:

Таблица 3.1

FUNCTION	OTHERS	
CASE	TABLE	JKFFE
BITS	SRFFE	NCLUDE
DFF	VCC	NODE
DFFE	WHEN	NOR
ELSE	WITH	NOT
END	XNOR	OPTIONS
EXP	XOR	OR
AND	GLOBAL	OUTPUT
BEGIN	GND	RETURNS
BURIED	INPUT	SOFT
BIDIR	IF	SRFF
CARRY	IS	STATES
CASCADE	JKFF	SUBDESIGN
CLIQUE	LATCH	TFF
CONNECTED_PINS	LCELL	TFFE
CONSTANT	MACHINE	THEN
DEFAULTS	MACRO	TITLE
DESIGN	MCELL	TRI
DEVICE	NAND	VARIABLE
ELSIF	OF	X

Символы

Ниже в таблице 3.2 приведены символы, имеющие определенное значение в языке AHDL. В этот перечень не включены символы, используемые в булевых выражениях как операторы и для операций сравнения.

Таблица 3.2

Символ	Функция
_ (подчеркивание)	Используемые пользователем идентификаторы
- (тире)	символы в символических именах
-- (два тире)	Начинает комментарий в стиле VHDL, который продолжается до конца строки
% (процент)	Заключает с двух сторон комментарий стиля AHDL
( ) (круглые скобки)	Заключают и определяют последовательные имена групп. Заключают имена выводов в секции подпроекта (Subdesign Section) и в прототипах функций. Заключают (необязательно) входы и выходы таблиц в объявлении Truth Table. Заключают состояния в объявлении цифрового автомата State Machine. Заключают более приоритетные операции в булевых выражениях. Заключают необязательные варианты в секции проекта Design Section (внутри объявления назначения ресурсов Assignment).
[ ] (квадратные скобки)	Заключают диапазон значений в десятичном имени группы
'...' (одинарные кавычки)	Заключают символические имена.
"..." (двойные кавычки)	Заключают строку в объявлении названия Title Заключают цифры в не десятичных номерах Заключают путь в объявлении Include. Могут (необязательно) заключать имя проекта и устройства в секции проекта Design Section. Могут (необязательно) заключать имя в объявлении назначения клики графа Clique Assignment.
. (точка)	Отделяет символические имена переменных в макрофункции или примитиве от имен портов. Отделяет имя файла от расширения
... (многоточие)	Разделяет наименьшее и наибольшее значение в диапазонах
; (точка с запятой)	Заканчивает объявления и секции в языке AHDL
, (запятая)	Разделяет элменты последовательных групп и списков
: (двоеточие)	Отделяет символические имена от типов в объявлениях и назначениях ресурсов.

Продолжение Таблицы 3.2

@ “собака”	Присваивает символические узлы выводам устройства и логическим ячейкам в объявлениях назначения ресурсов Resource Assignment
= (равенство)	Присваивает значения по умолчанию GND и VCC входам в секции подпроекта Subdesign Присваивает установочные значения в вариантах Присваивает значения состояниям в машине состояний Присваивает значения в булевых уранениях

=> (стрелка)	Отделяет входы от выходов в объявлениях таблицы истинности Truth Table Отделяет предложения с WHEN от булевых выражений в операторе Case

Имена в кавычках и без кавычек

В языке AHDL есть три типа имен:

* *Символические имена* - это определяемые пользователем идентификаторы. Они используются для обозначения следующих частей TDF:

- внутренних и внешних узлов (вершин);
- констант;
- переменных цифрового автомата, битов состояний, имен состояний;
- примеров (Instance).

* *Имена подпроекта* - это определяемые пользователем имена для файлов проекта более низкого уровня. Имя подпроекта должно быть таким же, как имя файла TDF.

* *Имена портов* - это символические имена, идентифицирующие вход или выход примитива или макрофункции.

В файле .fit проекта могут появиться генерируемые компилятором имена выводов, с символом “тильда” (~). Этот символ зарезервирован для имен, генерируемых компилятором, пользователю запрещается его использовать для обозначения имен выводов, узлов (вершин), групп (шин).

Существуют две формы записи для всех трех типов имен (символических, подпроекта и портов): *в кавычках* (') и *без кавычек*.

Если разработчик создает символ по умолчанию для файла TDF, который включает в себя имена портов в кавычках, собственно кавычки не входят в имена выводов.

Числа в языке AHDL

В языке AHDL можно использовать десятичные, двоичные, восьмеричные и шестнадцатеричные числа в любой комбинации. В таблице 3.3 приведен синтаксис записи чисел в языке AHDL для каждой системы счисления.

Таблица 3.3

Система счисления	Значения
Десятичная	<последовательность цифр 0–9>
Двоичная	B"<последовательность из 0, 1, X>", где символ X обозначает безразличное значение
Восьмиричная	O"< последовательность цифр 0–7>" или Q"< последовательность цифр 0–7>"
Шестнадцатиричная	X"< последовательность цифр 0–9, букв A–F>" или H"< последовательность цифр 0–9, букв A–F>"

Булевы выражения

Булевы выражения состоят из операндов, разделенных логическими и арифметическими операторами и компараторами и (необязательно) сгруппированных с помощью круглых скобок. Выражения используются в булевых уравнениях, а также в других конструкциях языка, таких как операторы Case и If.

Существуют следующие применения булевых выражений:

* Операнд.

Пример: a, b[5..1], 7, VCC

* Встроенная в текст (in-line) ссылка (reference) на примитив или макрофункцию.

* Префиксный оператор (! или -), примененный к булеву выражению.

Пример: !c

* Два булевых выражения, разделенные двоичным (не префиксным) оператором.

Пример: d1 $ d3

* Заключенное в круглые скобки булево выражение.

Пример: (!foo & bar)

Результат каждого булева выражения должен иметь ту же ширину, что и узел или группа (в левой стороне уравнения), которому он, в конечном счете, присваивается.

Логические операторы

В таблице 3.4 приведены логические операторы для булевых выражений.

Таблица 3.4

Оператор:	Пример:	Описание:
!	!tob	Дополнение (префиксное обращение)
NOT	NOT tob	
&	bread & butter	Логическое И
AND	bread AND butter	
!&	a[3..1] !& b[5..3]	Обращение логического И
NAND	a[3..1] NAND b[5..3]	
#	trick # treat	Логическое ИЛИ
OR	trick OR treat	
!#	c[8..5] !# d[7..4]	Обращение логического ИЛИ
NOR	c[8..5] NOR d[7..4]	
$	foo $ bar	Исключающее ИЛИ
XOR	foo XOR bar	
!$	x2 !$ x4	Обращение исключающего
XNOR	x2 XNOR x4	логического ИЛИ

Каждый оператор представляет собой логический вентиль с двумя входами; исключение составляет оператор NOT, являющийся префиксным инвертором. Для записи логического оператора можно использовать его имя или символ.

Выражения, в которых используются эти операторы, интерпретируются по-разному в зависимости от того, что представляют собой операнды: одиночные узлы (вершины), группы или числа. Кроме того, выражения с оператором NOT интерпретируются не так как другие логические операторы.

Задание к выполнению лабораторной работы:

1. Изучить основные элементы языка AHDL и правила описания логических схем.
2. Сделать описание электрической схемы заданной в предыдущей работе при помощи текстового редактора САПР MAX+PLUS II.
3. Произвести симуляцию работы схемы. Зарисовать диаграммы работы и по ее результатам заполнить таблицу истинности смоделированной схемы.
4. Сравнить результаты, полученные в ходе выполнения лабораторной работы с результатами, полученными в работе №2.

5. Ответить на контрольные вопросы, оформить отчет о выполненной работе.

Контрольные вопросы.

1. Что такое язык описания аппаратуры? Назовите существующие языки описания аппаратуры, в чем их отличие?
2. Назовите основные элементы языка AHDL, дайте их краткую характеристику.
3. Как описываются логические элементы в AHDL?

Лабораторная работа № 4

Моделирование цифровых схем с использованием параметрических элементов

Цель работы:

Приобретение навыков использования параметрических элементов (LPM function) в САПР MAX+PLUS II, экспериментальное исследование счетчиков и регистров, построенных на их основе.

Основные теоретические сведения:

Счетчики и регистры Регистры и счетчики относятся к разряду цифровых устройств и являются одним из наиболее распространенных элементов вычислительной техники. Они широко используются для построения устройств ввода, вывода и хранения информации, а также для выполнения некоторых арифметических и логических операций.

Для построения счетчиков и регистров используются синхронные триггеры, переключение которых происходит только при наличии синхронизирующего сигнала (синхроимпульса) на входе C. Наиболее часто для построения регистров и счетчиков используется универсальный D-триггер, имеющий специальный информационный вход D, и динамический вход C (рис.4.1).

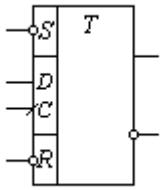


Рис. 4.1 D – триггер.

Устройство, называемое счетчиком, предназначено для подсчета числа поступающих на вход сигналов (импульсов) в произвольной системе счисления. Двоичные счетчики строятся на основе триггеров, работающих в счетном режиме (T - триггер или счетный триггер).

Счетный триггер может быть получен из универсального D - триггера путем соединения его инверсного выхода Q со входом D.

Счетный триггер и эпюры сигналов, поясняющие его работу, представлены на рис.4.2.

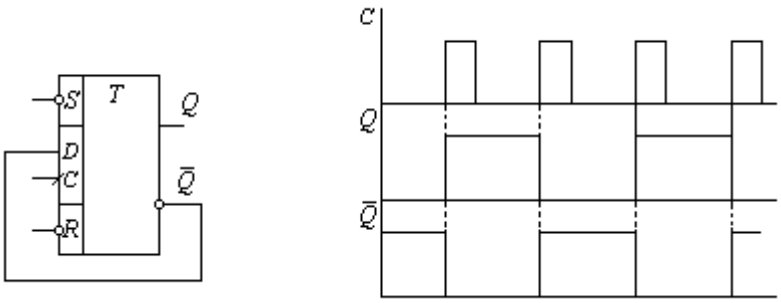


Рис. 4.2 Счетный триггер и его работа

У счетного триггера состояние выхода изменяется на противоположное при поступлении на вход С каждого очередного счетного импульса.

Функциональная схема и условное графическое обозначение двоичного счетчика с коэффициентом пересчета 2^3 представлена на рис. 4.3.

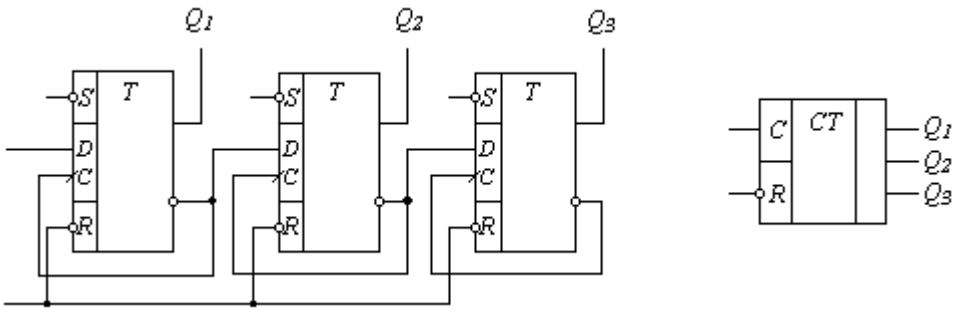


Рис. 4.3 Двоичный счетчик

Каждый поступающий на вход счетчика импульс перебрасывает первый триггер в противоположное состояние (рис. 4.4). Сигнал с инверсного выхода предыдущего триггера является входным сигналом для последующего и, таким образом, комбинация сигналов на выходах Q_1, Q_2, Q_3 будет соответствовать числу поступивших на вход счетчика импульсов, представленному в двоичном коде. Счетчик данного типа называется асинхронным счетчиком.

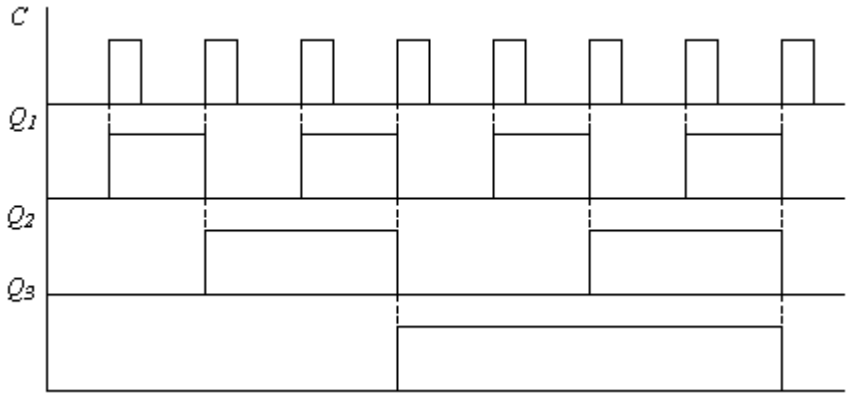


Рис. 4.4 Диаграммы работы двоичного счетчика

Если на счетный вход каждого последующего триггера счетчика подавать сигнал с прямого выхода предыдущего триггера, то счетчик будет производить операцию вычитания. Счетчики, способные выполнять функции сложения и вычитания, называются реверсивными.

Для построения счетчика с требуемым коэффициентом пересчета $К_с$, отличным от величины 2^N (N - число двоичных разрядов счетчика), используется принудительный сброс счетчика в исходное состояние при достижении счетчиком числа $К_с$.

Устройство, называемое регистром, служит в основном для хранения чисел в двоичном коде при выполнении над ними различных арифметических и логических операций. С помощью регистров выполняются такие действия над числами, как передача их из одного устройства в другое, арифметический и логический сдвиг в сторону младших или старших разрядов, преобразование кода из последовательного в параллельный и наоборот и т.д.

Функциональная схема и условно-графическое обозначение регистра сдвига представлены на рис. 4.5.

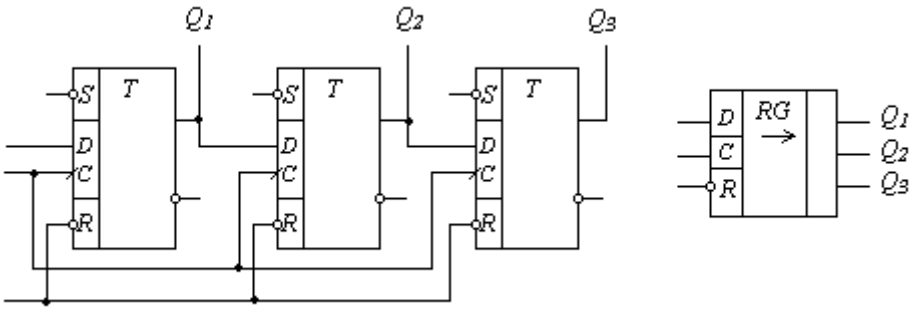


Рис. 4.5 Регистр сдвига

Последовательный информационный код поступит на вход D регистра. Импульс команды сдвига C подается одновременно на синхронизирующие входы всех триггеров регистра и переводит каждый триггер в состояние, в котором находился триггер предыдущего разряда. Таким образом, каждый импульс команды сдвига "продвигает" записываемое число на один разряд вправо.

При введении обратной связи в регистр сдвига, последний превращается в замкнутое кольцо, в котором под воздействием тактовых импульсов циркулирует введенная в регистр информация. Такие регистры называют кольцевыми счетчиками. Кодовая единица, введенная в первый триггера, циркулирует в течении всего времени существования тактовых импульсов, подаваемых на входы C всех триггеров счетчика. Приходящий тактовый импульс перебрасывает триггер, который был в состоянии 1, в состояние 0. Поскольку выход Q этого триггера связан с входом D следующего триггера, то последний устанавливается в состояние 1 и т.д. Количество состояний такого счетчика равно числу триггеров.

Реализация проекта на параметрических элементах. Применение параметрических элементов САПР MAX+PLUS II в разработке проектов цифровых схем рассмотрим на примере реализации реверсивного счетчика разрядностью 4.

Создаем новый файл графического редактора и сохраняем его под определенным именем (например: lab_5) в предварительно созданном каталоге \max2work\lab_5. Двойным щелчком правой кнопки мыши открываем меню ввода символов (Enter Symbol), выбираем библиотеку mega_lpm и в ней выбираем lpm_counter. После нажатия кнопки OK появляется окно (Edit Ports/Parameters) редактирования параметров и входов/выходов счетчика (рис. 4.6). Выбрав необходимые входы/выходы счетчика в поле Ports и задав разрядность LPM_WIDTH и направление счета LPM_DIRECTION (в данном примере: вычитание) в поле Parameters, нажимаем кнопку OK.

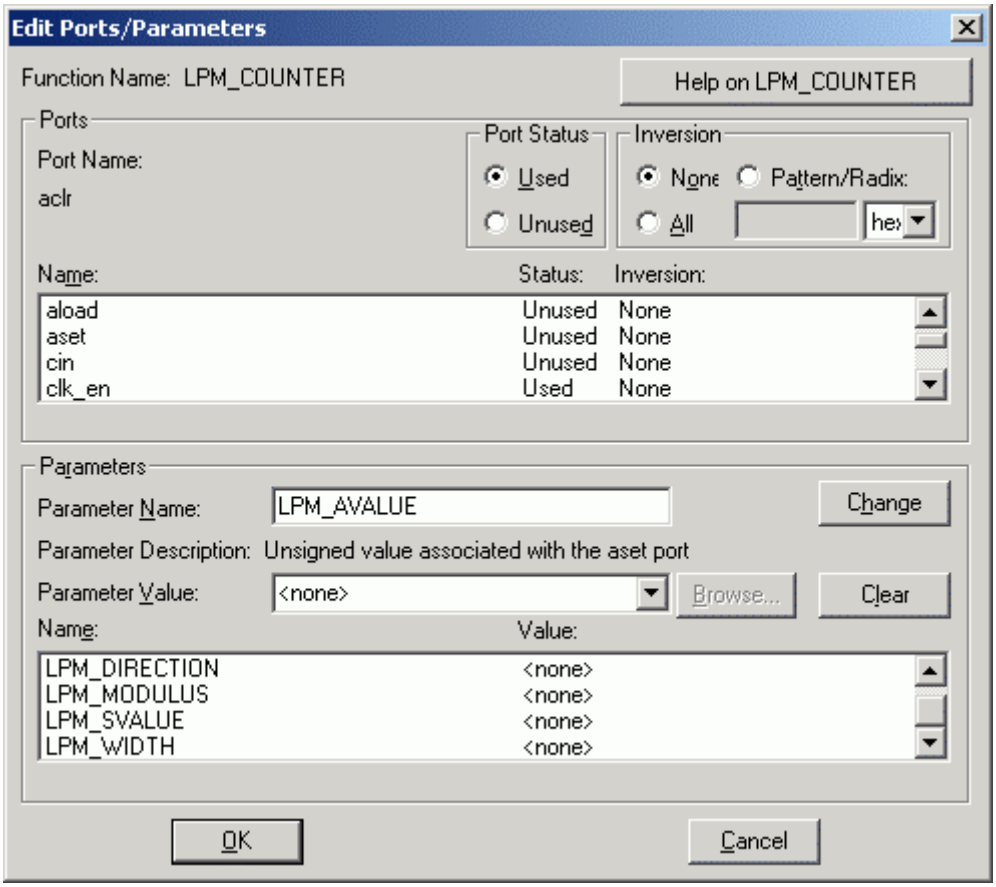


Рис. 4.6 Окно редактирования параметров счетчика

Далее располагаем входные и выходные выводы схемы проекта. Когда схема создана, делаем проверку на предмет наличия ошибок ввода схемы, для чего запускаем компилятор.

Если компиляция прошла успешно, создаем файл симулятора для анализа работы счетчика. В созданном файле задаем входной (in) периодический сигнал с периодом следования импульсов в 20 nc. Сохраняем файл и запускаем симулятор. Результатом симуляции будет диаграммы работы счетчика, приведенные на рис. 4.7.

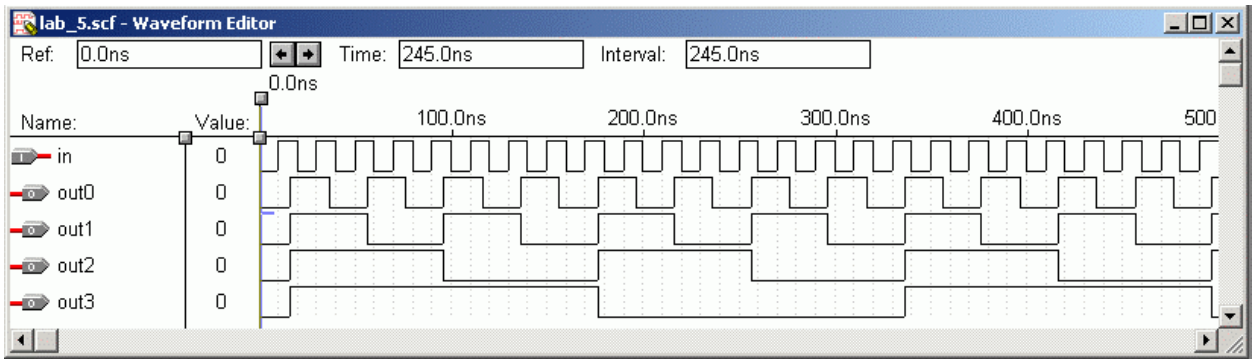


Рис. 4.7 Результат моделирования работы счетчика

Для анализа временных задержек запускаем временной анализатор (Timing Analyzer) и нажимаем кнопку START. Результаты временного анализа представлены на рис. 4.8.

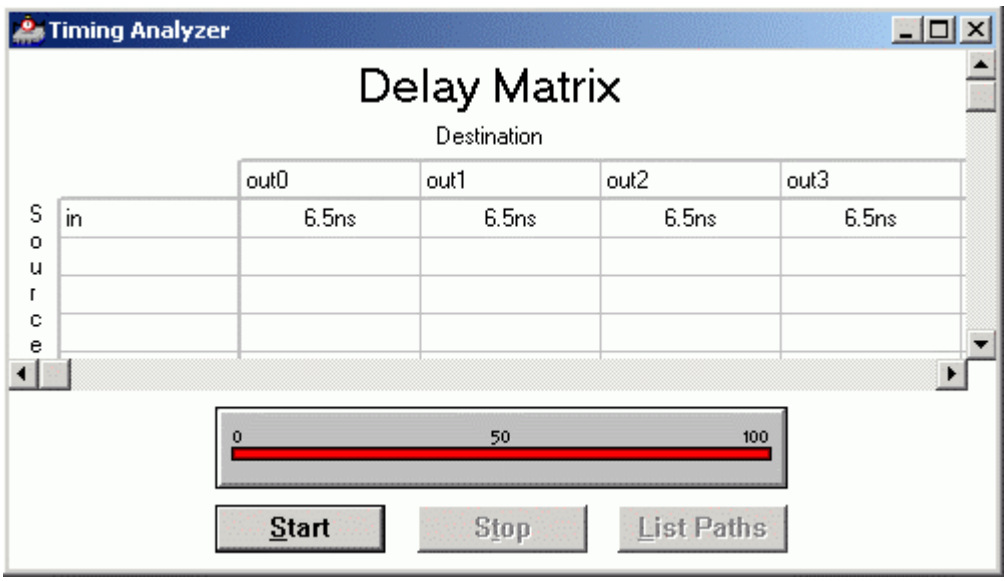


Рис. 4.8 – Таблица временных задержек исследуемого счетчика

Из рис. 4.8 видно, что задержка распространения сигнала от входа до любого из выходов счетчика составляет 6,5 nc. Такие результаты получены при назначении ПЛИС типа EP610ILC-10. Если же выбрать ПЛИС семейства MAX3000A то временные задержки в работе данного счетчика составят 3 nc.

Описание некоторых параметрических элементов САПР MAX+PLUS II представлено в приложении.

Задание к выполнению лабораторной работы:

1 Изучить правила построения и принцип работы триггеров и построение на их основе логических схем.

2 Нарисовать электрическую схему по указанию преподавателя при помощи графического редактора САПР MAX+PLUS II.

3 Произвести симуляцию работы схемы, зарисовать диаграммы работы и по ее результатам заполнить таблицу истинности смоделированной схемы.

4 Спроектировать эту же электрическую схему, но с использованием параметрических элементов САПР MAX+PLUS II, проверить ее работу в сигнальном редакторе и оценить временные задержки в схеме.

5 Ответить на контрольные вопросы, оформить отчет о выполненной работе.

Контрольные вопросы.

1. Объясните понятие “параметрический элемент”. Какие параметрические элементы доступны в САПР MAX+PLUS II?
2. Объясните принцип работы счетчика построенного на триггерах. Какие типы счетчиков существуют?
3. Объясните назначение пунктов меню Edit Ports/Parameters.
4. Чем ограничивается максимальная скорость работы счетчика? Какова максимальная частота работы счетчика разработанного в ходе выполнения лабораторной работы?

Лабораторная работа №5

Программирование лабораторного стенда УЛС ПЛИС

Цель работы:

Программирование ПЛИС при помощи программатора ByteBlaster и исследование работы схемы при помощи лабораторного стенда УЛС ПЛИС.

Основные теоретические сведения:

Таблица 5.1

Аппаратные средства фирмы ALTERA

	Внешние аппаратные средства программирования [1]	BitBlaster™ – последовательный кабель загрузки	ByteBlaster™ – параллельный кабель загрузки
Стандартное устройство программирования	Да		
Программирование в системе		Да	Да
Реконфигурирование в схеме		Да	Да

[1] Внешние аппаратные средства программирования включают в себя логическую программирующую плату (LP6) и основной программирующий модуль (MPU).

Master Programmer Unit (MPU) – аппаратный модуль, который используется вместе с соответствующим адаптером, для программирования ПЛИС. MPU соединяется с LP6 при помощи шлейфа через 25-контактный разъем. Состояние процесса программирования отражается на светодиодных индикаторах MPU. ALTERA предлагает три типа адаптеров для программирования ПЛИС: PLM-prefix адаптеры, PLE-prefix адаптеры, и PLAD3-12 адаптеры совместимости. Каждый адаптер содержит один из следующих разъемов: DIP, PLCC/JLCC, PGA, SOIC или QFP. Эти адаптеры подключаются непосредственно к MPU. Каждый адаптер обеспечивает поддержку программирования для определенного класса устройств. Кроме того PLM-prefix адаптеры (кроме PLMJ1213 и PLMT1064) поддерживают функциональное тестирование запрограммированных устройств. PLMJ 1213 и PLMT1064 адаптеры могут программировать конфигурационные ПЗУ. Каждый PLE-prefix адаптер обеспечивает поддержку программирования для определенного классического (Classic) устройства.

BitBlaster – последовательный кабель загрузки – аппаратный интерфейс на стандартный PC или UNIX рабочую станцию с RS-232 портом, который

обеспечивает конфигурирование FLEX10K, FLEX8000, FLEX6000 устройств и программирование MAX9000, MAX7000S и MAX7000A устройств. Разъем с 25 штырьками на кабеле разгрузки BitBlaster соединяется с RS-232 портом со стандартным последовательным кабелем. Разъем с 10 контактами на кабеле загрузки BitBlaster соединяется с устройством на плате схемы. Кабель BitBlaster содержит индикаторы состояния, которые указывают состояние конфигурации устройства или программирования.

ByteBlaster – параллельный кабель загрузки – аппаратный интерфейс на стандартный параллельный порт (LPT порт). Этот кабель позволяет конфигурировать FLEX10K, FLEX8000 и FLEX6000 устройства, а также программировать MAX9000, MAX7000S и MAX7000A устройства. Кабель загрузки ByteBlaster имеет 25 штырьковый разъем, который соединяется с параллельным портом PC, 10 контактный разъем, который соединяется с платой схемы.

Задание к выполнению лабораторной работы:

1. Ознакомиться с принципом программирования ПИЛС при помощи устройства ByteBlaster.
2. Запрограммировать лабораторный стенд для выполнения функций схемы разрабатываемой в одной из лабораторных работ.
3. Исследовать работу схемы, сравнить практические и теоретические результаты исследований.
4. Ответить на контрольные вопросы, оформить отчет о выполненной работе.

Лабораторная работа №6

Разработка проекта сложного цифрового устройства на базе УЛС ПЛИС

Цель работы:

Спроектировать сложное цифровое устройство при помощи САПР MAX+PLUS II. Исследовать работу устройства с использованием редакторов САПР MAX+PLUS II и лабораторного стенда УЛС ПЛИС.

Основные теоретические сведения:

Выбор элементной базы и САПР. На первом этапе проектирования на основании анализа технического задания (ТЗ) выявляются специфические требования проекта, позволяющие остановить свой выбор на определенной фирме, выпускающей БИС ПЛ, и на определенном семействе ПЛИС этой фирмы. Отбор осуществляется на основе анализа характеристик как самой БИС — логических, конструктивных, эксплуатационных, стоимостных, так и на анализе свойств требуемой или допустимой САПР. Зачастую выбор предопределяется уже имеющимся практическим заделом и опытом работы проектировщика с продукцией и САПР определенной фирмы. В этом случае уточняется семейство, архитектурные и эксплуатационные характеристики которого удовлетворяют требованиям ТЗ. Выбор семейства может существенно зависеть от специфических требований проекта, например, необходимости соответствия определенным интерфейсным стандартам, требования наличия скоростной встроенной памяти значительного объема, повышенной радиационной стойкости и т. д. На выбор могут влиять и такие характеристики, как условия поставок, объявления о разработке перспективных модификаций семейства и многие другие соображения. Одним из самых неприятных (и дорогостоящих по совокупности последствий) фактов является выяснение в ходе выполнения проекта невозможности его реализации на продукции выбранной фирмы.

Анализ интерфейсных требований к проекту позволяет конкретизировать количество внешних контактов, необходимых для реализации проекта, т. е. типоразмер корпуса БИС выбранного семейства ПЛИС. Сложность проекта или определенные требования к проекту (например, скоростные характеристики)

могут приводить к целесообразности использования на начальных этапах проектирования группы САПР сторонних фирм.

Спецификация проекта. Задача этого этапа — переход от технического задания к формализованному описанию проектируемого устройства. Как правило, ТЗ является смесью словесного и технического описания, его формализация приводит к выявлению основных блоков устройства (или алгоритма) и определению их связей и/или взаимодействия. В сущности, именно в этот момент реализуются начальные действия второго этапа. Формально же содержание работ этого этапа — разбиение задачи на отдельные функционально обособленные подзадачи — этап декомпозиции. Способ и средства разбиения чаще всего определяются именно функциональной завершенностью и обособленностью отдельных фрагментов, хотя в значительной степени здесь большую роль играют просто симпатии проектировщика, и лишь иногда разбиение является полностью предопределенным. Сама форма ТЗ может провоцировать проектировщика на использование тех или иных средств, хотя не исключено, что более эффективным мог бы быть другой метод описания проекта или его фрагментов. Декомпозиция может сводиться к составлению схем алгоритмов функционирования фрагментов или к функциональной схеме устройства и его частей. Возможным вариантом для достаточно сложных систем будет разумное совмещение и поведенческого, и структурного разбиения проекта. Разбиение осуществляется не только в рамках одного уровня иерархии. Для большинства проектов производится и разбиение на иерархически организованные уровни.

Существенной задачей, решаемой на этом этапе, является уточнение и согласование с заказчиком интерфейсных функций проекта. Уточняется реализация протоколов внешнего обмена. Именно требуемые временные характеристики и правила взаимодействия с внешними приборами определяют допустимую организацию и структуру внутренних узлов проекта.

Использование САПР на этом этапе проектирования пока еще явление достаточно редкое, хотя для реализации современных очень сложных проектов (несколько сотен тысяч вентилей) все чаще используются специальные блочные редакторы, позволяющие осуществлять декомпозицию проекта без детализации

составных частей. Примером может служить САПР Quartus фирмы Altera, включающая в свой состав специальное средство, редактирующее проект на уровне блоков (block-level editing).

Разработка общей структуры проекта. Основные задачи этапа — выбор допустимых для реализации каждого уровня иерархии элементов, определение связей между ними, и если параметры элементов являются настраиваемыми, то и их настройка. Ряд моментов является для этапа определяющим: с одной стороны это источник набора допустимых элементов, а с другой — средства описания соединений элементов между собой, а при необходимости, и возможность описания новых (специфических для этого проекта) элементов.

Как уже указывалось, возможно, как только временное (поведенческое), так и только пространственное (архитектурно-структурное) описание проекта. Однако обычно целесообразно совмещать обе возможности. При разработке устройств с цифровым представлением информации бывает естественным разбиение их на два блока: операционный и управления. Операционный блок (ОБ) выполняет преобразование данных и строится из стандартных частей (частей с определенным поведением), а блок управления (устройство управления, УУ) обеспечивает необходимую последовательность операций, выполняемых в ОБ (одном или нескольких). Для этого УУ передает на входы ОБ управляющие сигналы. Последовательность действий и, следовательно, управляющих сигналов зависит от результатов операций в ОБ и внешних воздействий. Отсюда видно, что УУ удобно задавать в форме конечного автомата с памятью (АП) того или иного типа.

В сложных проектах возможно разделение УУ на несколько функционально слабо связанных пар ОБ-УУ на одном уровне иерархии или создание пары, иерархически погруженной в ОБ (реже в УУ).

Пример реализации проекта с комбинированным описанием. Порядок использования основных приложений системы MAX+PLUS II рассмотрим на примере разработки проекта секундомера. В качестве индикатора используется четыре семисегментных индикатора с общим катодом, т.е. сигналы управления сегментами должны иметь активные высокие уровни для подсветки сегментов.

Основными элементами такого устройства являются:

- схема деления тактового сигнала генератора;

- счетный каскад секунд и минут;

- комбинационные схемы преобразования двоичных кодов в сигналы управления сегментами индикатора.

В разрабатываемый проект целесообразно включить два уровня иерархии:

- уровень структурных элементов, в который входит схема деления сигнала генератора, схемы счетчиков секунд и минут, комбинационные схемы преобразования двоичных кодов в сигналы управления сегментами индикатора;

- уровень проекта в целом.

Перед началом работы в системе MAX+PLUS II в рабочем каталоге системы, который был создан при инсталляции САПР по умолчанию, C:\max2work необходимо создать каталог, в котором будут находиться файлы проекта. Это связанно с тем, что в процессе работы MAX+PLUS II создает большое количество вспомогательных файлов относящихся только к данному проекту. Так же рекомендуется удалять файлы относящиеся к проекту только через меню “File\Delete File…” [2] .

Разработку проекта можно начать с проектирования двоично-десятичного счетчика в графическом редакторе (Graphic Editor). Двоично-десятичный счетчик может быть построен на Т-триггерах по схеме представленной на рис. 6.1.

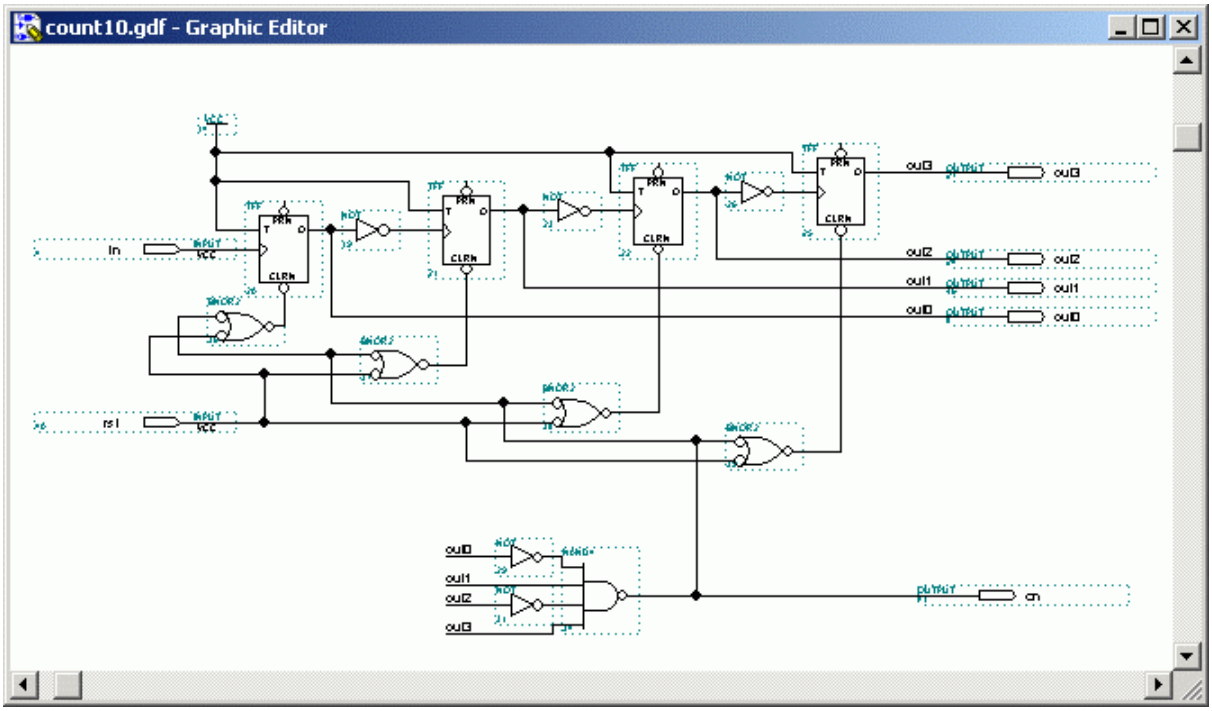


Рис. 6.1. Схема счетчика

В процессе разработки можно пользоваться библиотеками примитивов, макрофункций, LPM – функций, которые доступны через меню “Symbol\Enter Symbol…”, либо двойным кликом левой кнопки мыши на свободном месте окна графического редактора.

Для включения схемы счетчика в файл верхнего уровня иерархии проекта секундомера необходимо создать символ проекта счетчика через меню “File\Create Default Symbol”, причем перед созданием символа необходимо провести проверку на предмет наличия возможных ошибок через меню “File\Project\Save & Check” или нажатием соответствующей кнопки  на панели инструментов.

Для создания проекта дешифратора выходного двоичного кода счетчиков в сигналы управления элементами семисегментных индикаторов можно воспользоваться текстовым редактором (Text Editor), в котором алгоритм работы представлен на языке описания цифровой аппаратуры, в данном случае на языке AHDL [3].

Для задания соответствия двоичных кодов кодам управления семисегментных индикаторов в языке AHDL используется объявление TABLE. На рис. 6.2 приведен текст файла 7seg.tdf описывающий логику работы дешифратора.

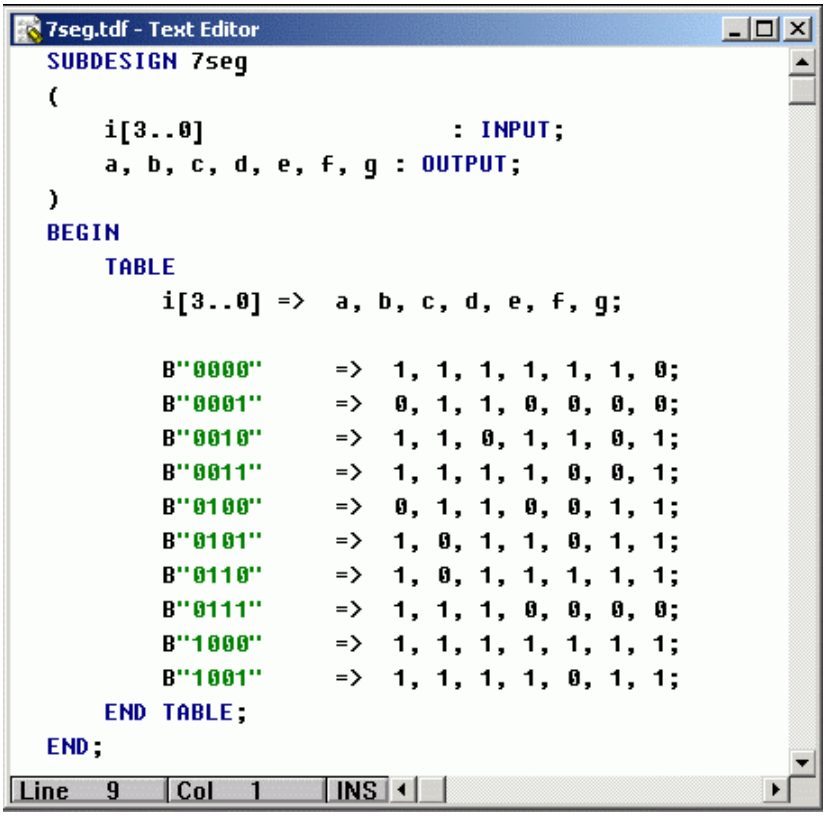


Рис. 6.2 Описание логики дешифратора

Далее, как описано выше, создается символ проекта дешифратора, с назначением файла 7seg.tdf файлом верхнего уровня иерархии проекта секундомера.

Для реализации секундомера необходим делитель входных тактовых импульсов генератора с соответствующим коэффициентом деления. Например, для деления частоты генератора лабораторного стенда необходим делитель с коэффициентом деления $8*10^6$, его символ также можно создать в любом из редакторов САПР MAX+PLUS II.

После того как все элементы проектируемого устройства созданы, переходим к созданию файла верхнего уровня. На рис. 6.3 приведена общая схема

проекта секундомера, где элемент DIV представляет собой делитель входных импульсов с коэффициентом деления $8*10^6$, элемент COUNT6 двоично-десятичный счетчик с коэффициентом счета равным 6, COUNTER - двоично-десятичный счетчик с коэффициентом счета 10, 7SEG - дешифратор.

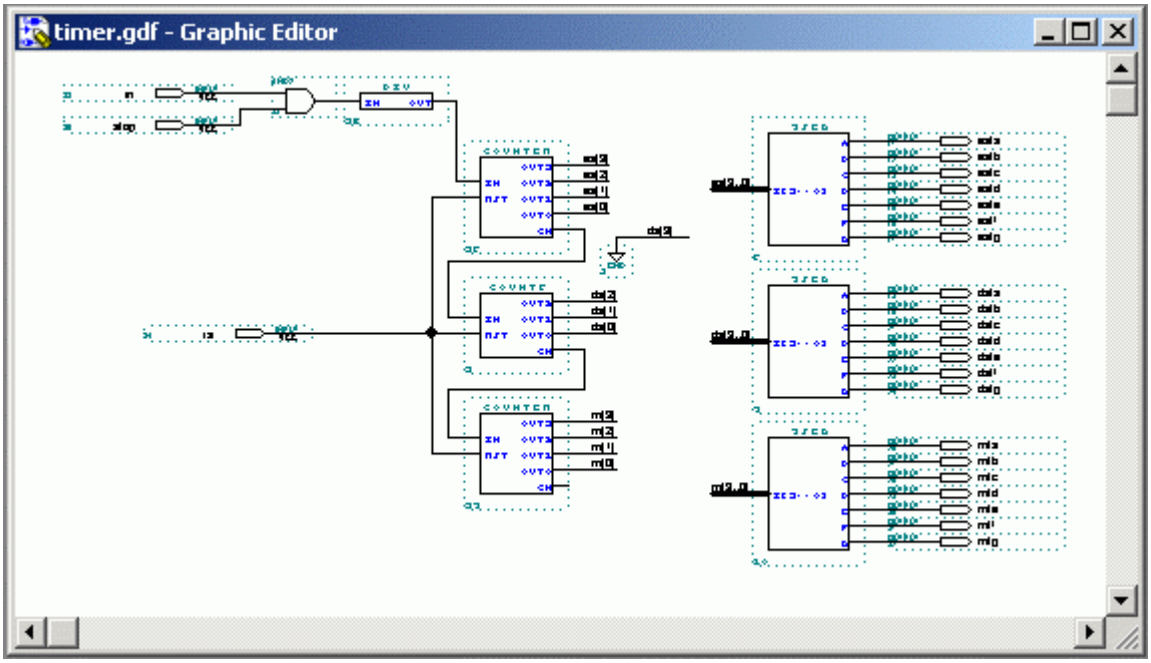


Рис. 6.3 Схема счетчика

Далее в соответствии с алгоритмом разработки проектов на ПЛИС (рис.1) производится назначение типа микросхемы для реализации разработанного проекта (меню “Assign\Device…”) и назначение выводов ПЛИС (меню “Assign\PinLocationChip…” или при помощи редактора связей (Floorplan Editor)). После того как все назначения проведены, проект необходимо откомпилировать и произвести симуляцию. По завершении процесса компиляции создается файл для дальнейшего конфигурирования ПЛИС.

Процесс конфигурирования ПЛИС начинается нажатием на кнопку “Configure” окна программатора САПР MAX+PLUS II.

После успешного конфигурирования УЛС ПЛИС сразу же готов к выполнению функций секундомера. На семисегментных индикаторах индицируется отсчет времени. Управление работой устройства осуществляется

при помощи кнопки запуска/останова счета и кнопки сброса секундомера, определенные в процессе назначения выводов ПЛИС.

Задание к выполнению лабораторной работы:

1 Ознакомиться с принципами разработки сложных цифровых устройств при помощи САПР MAX+PLUS II.

2 Разработать структурную схему проектируемого устройства.

3 Каждый функциональный блок проектируемого устройства разработать отдельно и объединить их в общий файл верхнего уровня.

4 Смоделировать работу схемы в сигнальном редакторе, оценить временные задержки схемы.

5 Исследовать работу разработанного устройства при помощи лабораторного стенда.

6 Ответить на контрольные вопросы, оформить отчет о выполненной работе.

Контрольные вопросы.

1. Объясните принцип работы разработанного устройства.
2. Как реализуется счетчик с заданным числом счета?
3. Что такое дешифратор? Как реализованы дешифраторы в Вашей работе?
4. Как назначить выводы ПЛИС в MAX+PLUS II? Всегда ли необходимо прибегать к назначению выводов ПЛИС вручную, обоснуйте свой ответ.

Приложение А

Параметрические элементы САПР MAX+PLUS II

Counter

Входные выводы	
Имя вывода	Описание
data [ ]	Параллельный вход данных счетчика
clock	Вход счетных импульсов
clk_en	Разрешение синхронизации.
cnt_en	Разрешение счета
updown	Управление направлением счета (1 = сложение, 0 = вычитание)
aclr	Асинхронный сброс входов
aset	Асинхронная установка входов
aload	Асинхронная загрузка входов. Установка счетчика в значение data[ ].
sclr	Синхронный сброс входов. Сброс счетчика следующим тактовым импульсом
sset	Синхронная установка входов. Установка счета следующим тактовым импульсом.
sload	Синхронная загрузка входов. Загрузка в счетчик значения data[ ] следующим тактовым импульсом.

Выходные выводы	
Имя вывода	Описание
q [ ]	Выход счетчика
eq [15..0]	Декодированный выход счетчика. Высокий активный уровень появляется в момент когда счетчик достигает заданного значения.
cout	Перенос в старший разряд

Параметры	
Параметр	Описание
LPM_WIDTH	Разрядность счетчика или входных значений data[ ] и выходных q[ ].
LPM_DIRECTION	Может принимать значения “UP”, “DOWN” или “UNUSED”. Если этот

	параметр используется, то вход updown не должен быть подключен. Если вход updown не подключен, то значение LPM_DIRECTION по умолчанию – “UP”
LPM_MODULS	Максимальный счет, плюс один. Число уникальных состояний в цикле счетчика. Если введенное значение больше, чем LPM_MODULUS параметр, поведение счетчика не определено.
LPM_AVALUE	Постоянное значение, которое загружается, когда aset высок. Если введенное значение больше чем <modulus>, поведение счетчика - неопределенный (X) логический уровень, где <modulus> - LPM_MODULUS. Параметр ограничен значением в 32 бита.
LPM_SVALUE	Постоянное значение, которое загружается по переднему фронту тактовых импульсов, когда sset или sconst высок. Должен Использоваться, если sconst используется.
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL

Divider

Входные выводы	
Имя вывода	Описание
numer[]	Числитель
denom[]	Знаменатель
clock	Вход тактовых импульсов
clken	Разрешение использования тактового входа
aclr	Асинхронный сброс

Выходные выводы	
Имя вывода	Описание
quotient[]	Частное
remain[]	Остаток

Параметры	
Параметр	Описание
LPM_WIDTHN	Разрядность numer[] и quotient[].
LPM_WIDTHD	Разрядность denom[] и remain[].
LPM_NREPRESENTATION	Определяет параметр числителя “SIGNED” или “UNSIGNED” Сейчас поддерживается только “UNSIGNED”.
LPM_DREPRESENTATION	Определяет параметр знаменателя “SIGNED” или “UNSIGNED” Сейчас поддерживается только “UNSIGNED”.
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL

Multiplier

Входные выводы	
Имя вывода	Описание
dataa [ ]	Множимое
datab [ ]	Множитель
sum[]	Частичная сумма
clock	Вход тактовых импульсов
clken	Разрешение использования тактового входа
aclr	Асинхронный сброс

Выходные выводы	
Имя вывода	Описание
result[]	result = dataa[] * datab[] + sum. The product LSB is aligned with the sum LSB.

Параметры	
Параметр	Описание
LPM_WIDTHA	Разрядность dataa[ ].
LPM_WIDTHB	Разрядность datab[ ].
LPM_WIDTHP	Разрядность result[ ].
LPM_WIDTHS	Разрядность sum[ ]. Обязателен, даже если порт суммы не используется.

LPM_REPRESENTATION	Тип выполняемого сравнения “SIGNED”, ”UNSIGNED”, “UNUSED”. Если значение не указанно, то по умолчанию устанавливается ”UNSIGNED”
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL
INPUT_A_IS_CONSTANT	Altera параметр. Принимает значения "YES", "NO", и "UNUSED". Если dataa [] связан с постоянным значением, устанавливая INPUT_A_IS_CONSTANT "YES" оптимизирует *multiplier* по использованию ресурсов и скорости. Если опущено, значение по умолчанию - "NO".
INPUT_B_IS_CONSTANT	Altera параметр. Принимает значения "YES", "NO", и "UNUSED". Если datab [] связан с постоянным значением, устанавливая INPUT_B_IS_CONSTANT "YES" оптимизирует *multiplier* по использованию ресурсов и скорости. Значение по умолчанию - "NO".
USE_EAB	Altera параметр. Принимает значения "ON", "OFF", и "UNUSED". Устанавливая параметр USE_EAB "ON" позволяет MAX+PLUS II использовать блоки дополнительных атрибутов, чтобы использовать 4 x 4 или (8 x значение константы) стандартные блоки в ACEX1K и FLEX10K устройствах.
LATENCY	Altera параметр. То же, что и LPM_PIPELINE. Параметр обеспечивает совместимости с MAX+PLUS II проектами версии ниже 7.0. Для всех новых проектов, используется параметр LPM_PIPELINE
MAXIMIZE_SPEED	Altera параметр. Возможные значения от 0 до 10. Если параметр используется то MAX+PLUS II пытается оптимизировать данную функцию lpm_mult для скорости, а не для уменьшения занимаемой области, и отменяет установку опции Optimize в

	диалоговом окне Global Project Logic Synthesis (меню Assign). Если MAXIMIZE_SPEED не использован, значение опции Optimize используется вместо него. Если установлено MAXIMIZE_SPEED - 6 или выше, компилятор оптимизирует мегафункции lpm_mult для более высокой скорости; если установлено - 5 или меньше, компилятор оптимизирует для уменьшения занимаемой области.

Comparator

Входные выводы	
Имя вывода	Описание
dataa[]	datab[] сравнивается с этим значением
datab[]	Значение с которым сравнивается dataa[]
clock	Вход тактовых импульсов
clken	Разрешение использования тактового входа
aclr	Асинхронный сброс

Выходные выводы	
Имя вывода	Описание
alb	“High” (1) если dataa[] < datab[]
aeb	“High” (1) если dataa[] == datab[]
agb	“High” (1) если dataa[] > datab[]
ageb	“High” (1) если dataa[] >= datab[]
aneb	“High” (1) если dataa[] != datab[]
aleb	“High” (1) если dataa[] <= datab[]

Параметры	
Параметр	Описание
LPM_WIDTH	Разрядность входов dataa[ ] и datab[ ].
LPM_REPRESENTATION	Тип выполняемого сравнения “SIGNED”, ”UNSIGNED”, “UNUSED”. Если значение не

	указанно, то по умолчанию устанавливается ”UNSIGNED”
LPM_PIPELINE	
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL.
CHAIN_SIZE	
ONE_INPUT_IS_CONSTANT	Специфический Altera - параметр. Принимает значения "YES", "NO", или "UNUSED". Обеспечивает большую оптимизацию, если один из входов постоянен. По умолчанию - "NO".

Adder Subtractor

Входные выводы	
Имя вывода	Описание
dataa [ ]	Первое слагаемое / Уменьшаемое
datab [ ]	Слагаемое / Вычитаемое
add_sub	Если “1” (high), операция = dataa[]+datab[] +cin. Если “0” (low), операция = dataa[]-datab[] +cin-1
clock	Вход тактовых импульсов
clken	Разрешение использования тактового входа
aclr	Асинхронный сброс

Выходные выводы	
Имя вывода	Описание
result[]	dataa[] +datab[] +cin или dataa[] -datab[] +cin-1.
cout	Обнаруживает переполнения в операциях "UNSIGNED".
overflow	Результат превышает доступную точность

Параметры	
Параметр	Описание
LPM_WIDTH	Разрядность dataa[ ], datab[ ], result[ ].
LPM_DIRECTION	Значения - "ADD", "SUB", и "UNUSED". Если не указано, значение по умолчанию "DEFAULT", в этом случае используется значение add_sub порта. Add_sub порт не может использоваться, если используется LPM_DIRECTION.
LPM_REPRESENTATION	Тип выполняемого сравнения “SIGNED”, ”UNSIGNED”, “UNUSED”. Если значение не указанно, то по умолчанию устанавливается ”UNSIGNED”
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL
ONE_INPUT_IS_CONSTANT	Altera параметр. Принимает значения "YES", "NO", и "UNUSED". Обеспечивает большую оптимизацию, если один вход постоянный. Если не указано, значение по умолчанию - "NO"
MAXIMIZE_SPEED	Altera параметр. Возможные значения от 0 до 10. Если параметр используется то MAX+PLUS II пытается оптимизировать данную функцию lpm_mult для скорости, а не для уменьшения занимаемой области, и отменяет установку опции Optimize в диалоговом окне Global Project Logic Synthesis (меню Assign). Если MAXIMIZE_SPEED не использован, значение опции Optimize используется вместо него. Если установлено MAXIMIZE_SPEED - 6 или выше, компилятор оптимизирует мегафункции lpm_mult для более высокой скорости; если установлено - 5 или меньше, компилятор оптимизирует для уменьшения занимаемой области.

Absolute Value

Входные выводы	
Имя вывода	Описание
data [ ]	Число со знаком

Выходные выводы	
Имя вывода	Описание
result[]	Абсолютное значение data[]
overflow	

Параметры	
Параметр	Описание
LPM_WIDTHA	Разрядность data[] и result[].
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL

Приложение Б

Варианты заданий для выполнения лабораторной работы №2

№ варианта	Задание
1	$Y = A+B+CD+AD+BD$
2	$Y = AB+CD+AB+BD$
3	$Y = AB+\overline{C}+\overline{D}+AD+BD$
4	$Y = AB+CDAD+BD$
5	$Y = AB +CDAD+B+D$
6	$Y = \overline{C}D+BC+A$
7	$Y = (BD+CA)D$
8	$Y = B+C+\overline{AD}+\overline{BC}$
9	$Y = CD(A+B)+AB(C+D)$
10	$Y = \overline{C}D+A\overline{B}+\overline{D}C+CB$

Список литературы

1. Соловьев В. В. Проектирование цифровых систем на основе программируемых логических интегральных схем. – М.: Горячая линия - Телеком, 2001. – 636 с.
2. Стешенко В. ПЛИС фирмы ALTERA: проектирование устройств обработки сигналов - М.: Додека, 2000. – 128 с.
3. Антонов А.П. Язык описания цифровых устройств AlteraHDL: Практический курс. – М.: ИП «Радиософт», 2001. – 224 с.
4. ALTERA® ACEX 1K Programmable Logic Family Data Sheet.
5. ALTERA® APEX 20K Programmable Logic Device Family Data Sheet.
6. ALTERA® APEX 20KC Programmable Logic Device Family Data Sheet.
7. ALTERA® Altera Programming Hardware Data Sheet.