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

Кафедра электронной техники и технологии

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ ИНТЕГРАЛЬНЫЕ СХЕМЫ

Лабораторный практикум по курсу
«Микроэлектронные схемы, микротехнологии и микропроцессоры
в средствах медицинской электроники»
для студентов специальности
«Медицинская электроника»
дневной и заочной форм обучения

Минск 2005

УДК 621.382.8.049.77 (075.8)
ББК 32.844.1 я73
П 78

Р е ц е н з е н т :
доцент кафедры сетей и устройств телекоммуникаций БГУИР,
канд. техн. наук А.А. Борискевич

А в т о р ы :
А.Н. Осипов, С.В. Кракасевич, В.М. Бондарик, М.Е. Полонецкий

П 78 **Программируемые** логические интегральные схемы: Лаб. практикум по курсу «Микроэлектронные схемы, микротехнологии и микропроцессоры в средствах медицинской электроники» для студ. спец. «Медицинская электроника» дневной и заочной форм обуч. / А.Н. Осипов, С.В. Кракасевич, В.М. Бондарик, М.Е. Полонецкий. – Мн.: БГУИР, 2005. – 40 с.: ил.
ISBN 985-444-825-8

Лабораторный практикум составлен в соответствии с программой курса «Микроэлектронные схемы, микротехнологии и микропроцессоры в средствах медицинской электроники» и включает в себя руководство по работе с системой автоматизированного проектирования MAX+PLUS II при моделировании цифровых систем на основе программируемых логических интегральных схем для студентов, обучающихся по специальности «Медицинская электроника».

Предназначен для закрепления и углубления теоретических знаний, полученных на лекциях и в процессе самостоятельного изучения дисциплины, совершенствования практических навыков в области применения пакета MAX+PLUS II для автоматизированного проектирования цифровых систем на основе программируемых логических интегральных схем.

УДК 621.382.8.049.77 (075.8)
ББК 32.844.1 я73

ISBN 985-444-825-8

Лабораторная работа № 1

СИСТЕМА АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ MAX+PLUS II

1. Цель работы

Изучить структуру и функциональные возможности графического редактора САПР MAX+PLUS II.

2. Основные теоретические сведения

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.

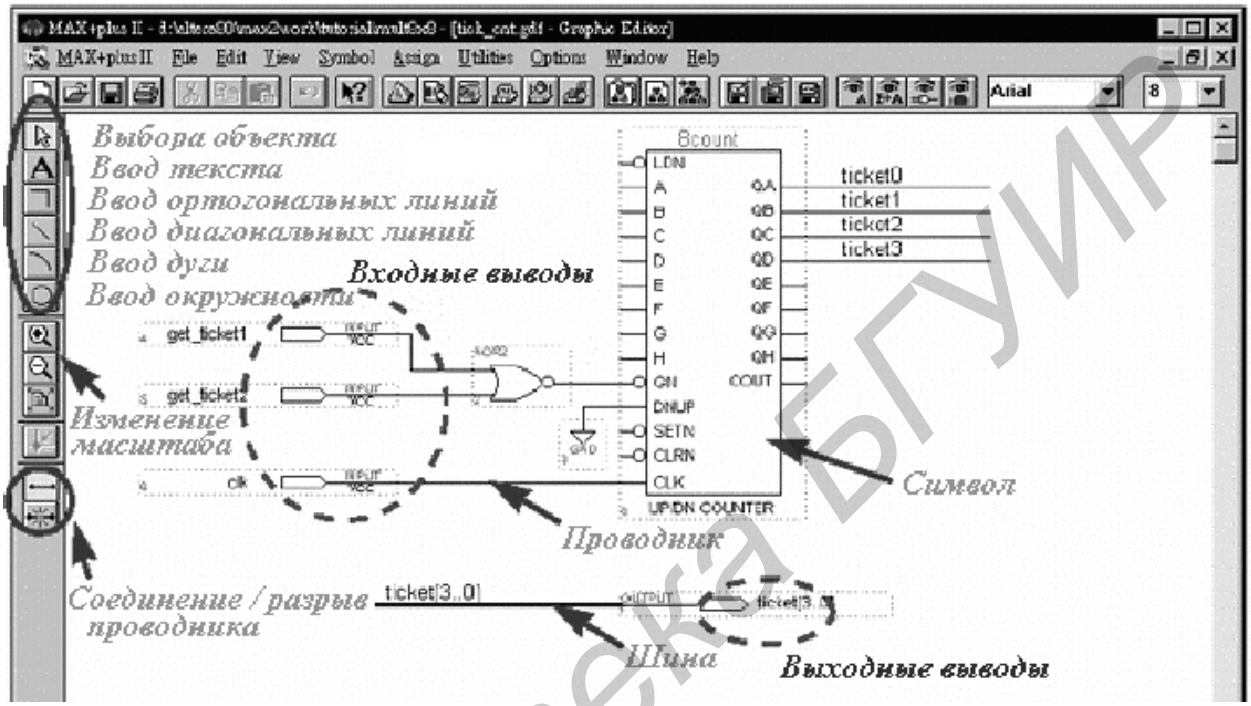


Рис. 1.2. Графический редактор

Вставка символа производится двойным щелчком 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 также обмениваться с другими окнами через буфер обмена. Выводы элементов можно соединять сигнальными проводниками либо присваивать одинаковые имена 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.

Время, в течение которого будет проводиться моделирование, задается в меню «File», пункт «End Time», шаг временной сетки задается в меню «Options», пункт «Grid Size».

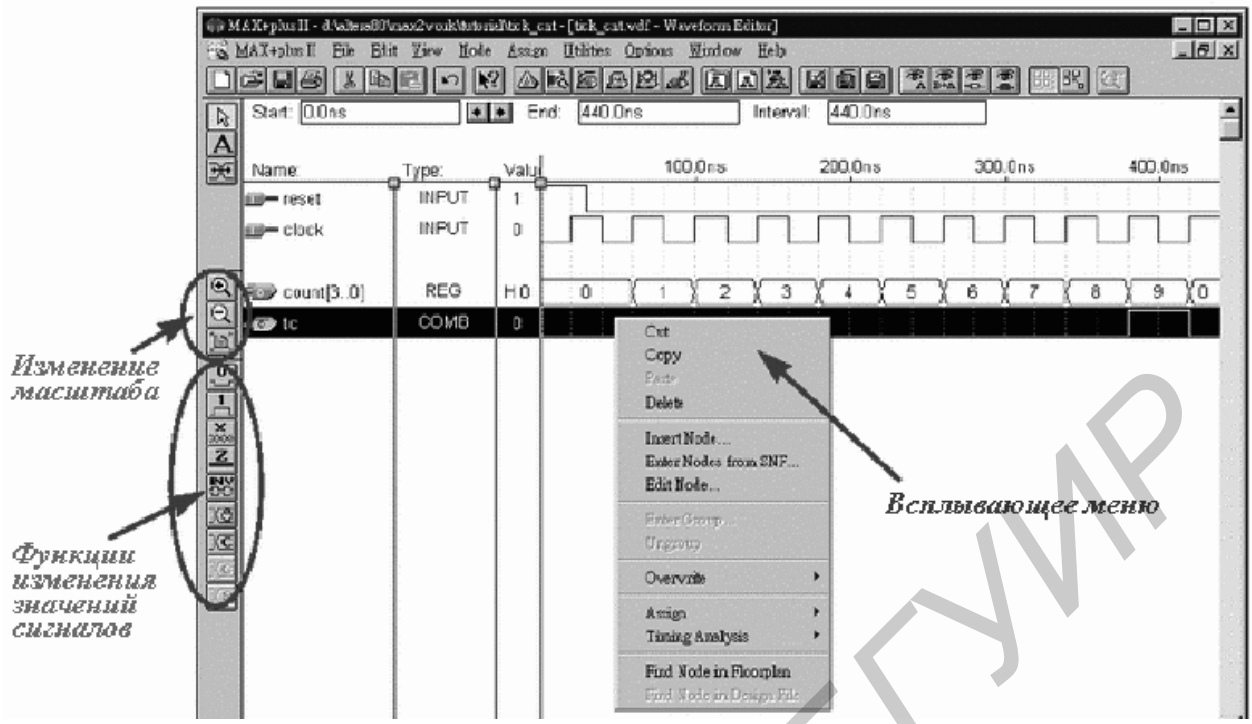


Рис. 1.3. Редактор временных диаграмм

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.

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.

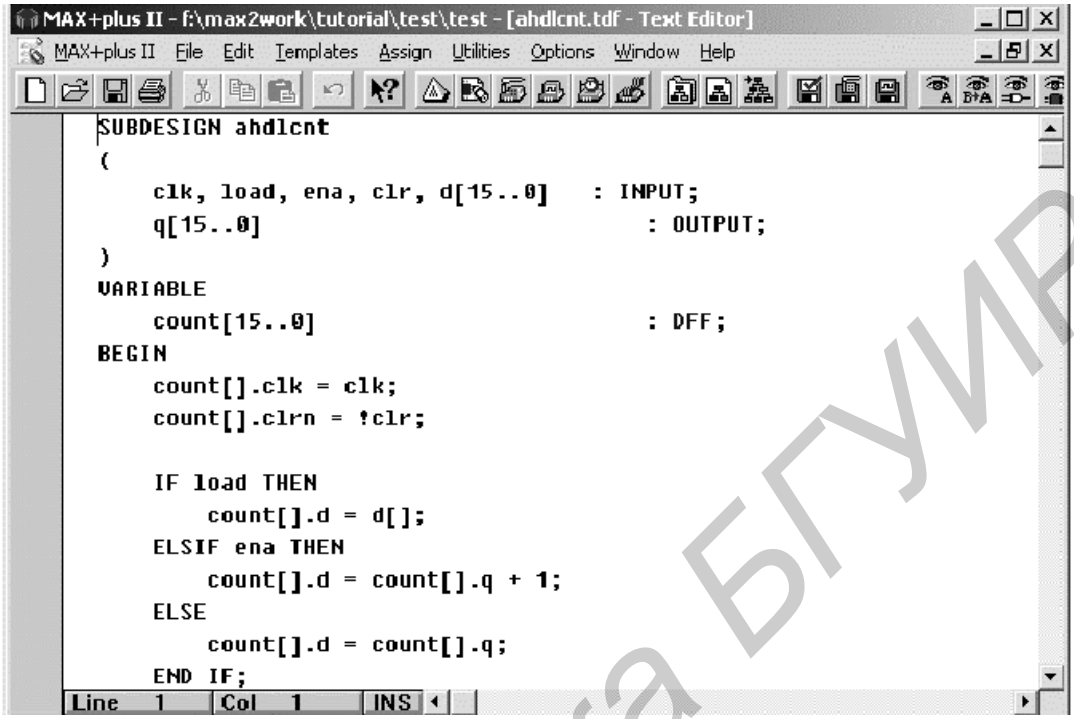


Рис. 1.4. Окно текстового редактора

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.

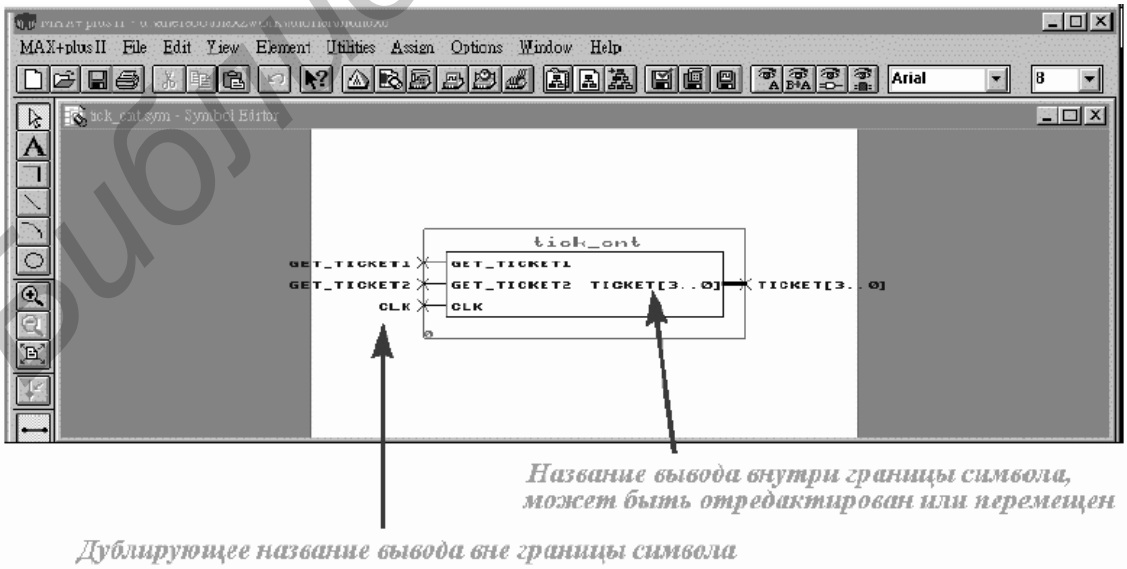


Рис. 1.5. Символьный редактор

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.

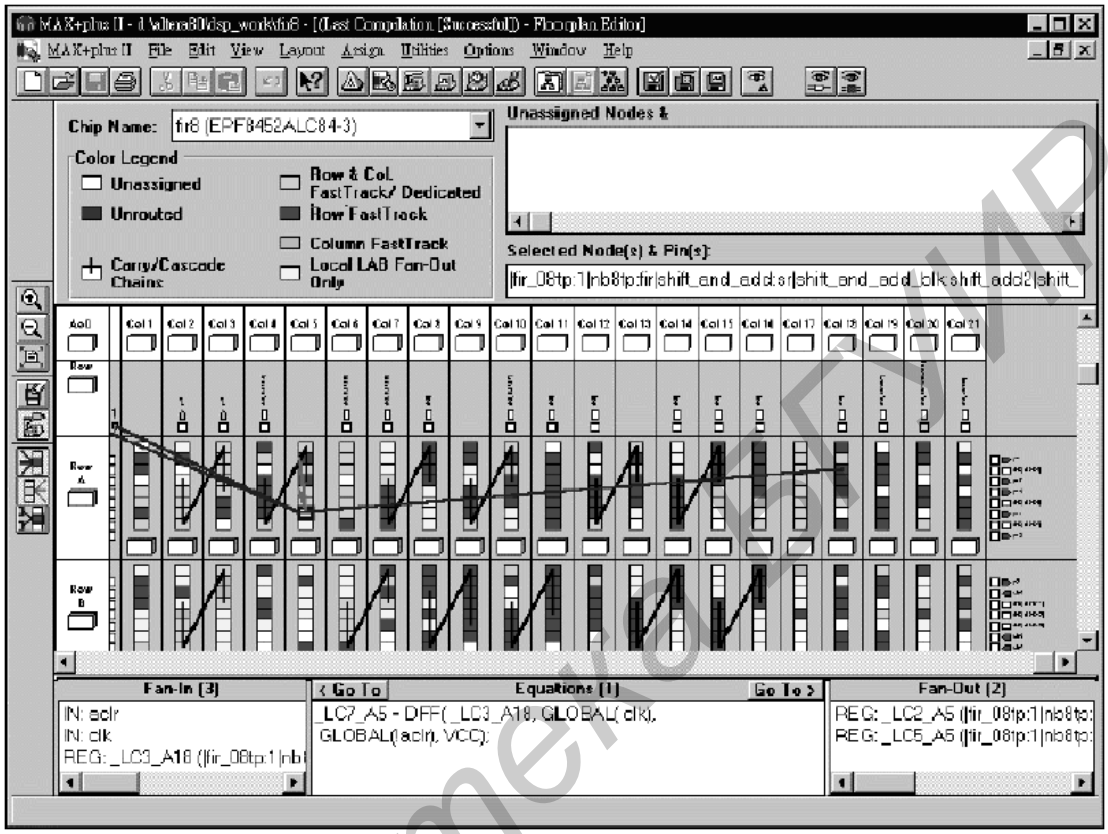


Рис. 1.6. Редактор связей системы MAX+PLUS II

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.

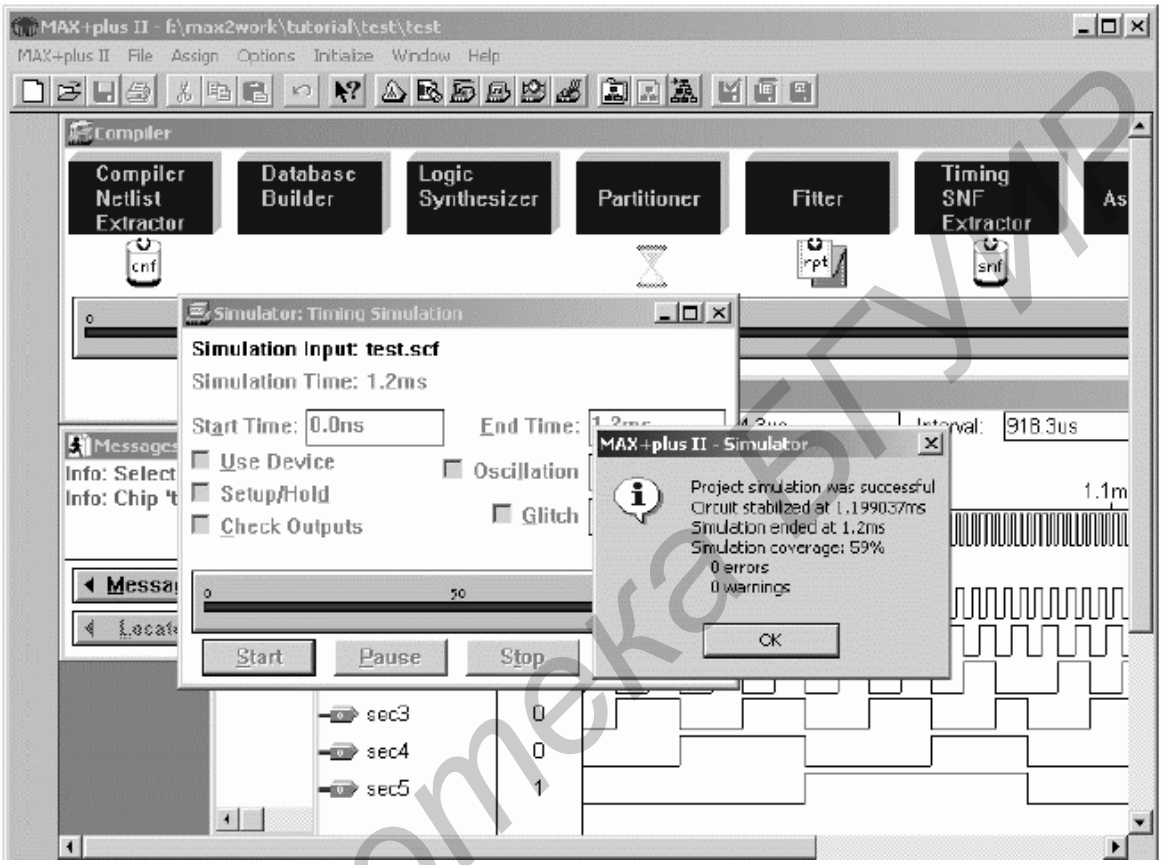


Рис. 1.7. Функциональное моделирование в системе MAX+PLUS II

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).

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.

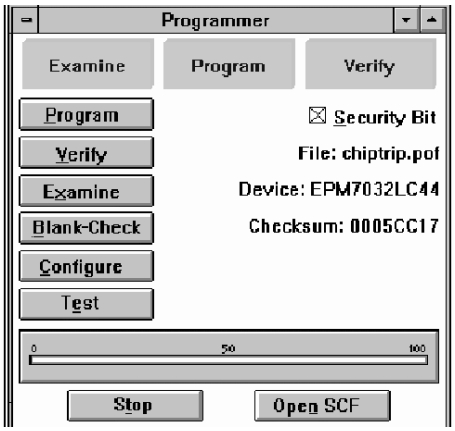


Рис. 1.8. Программатор системы MAX+PLUS II

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, 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.

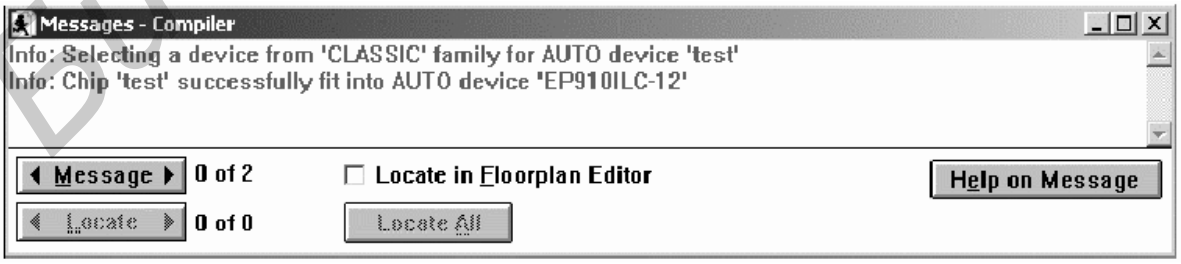


Рис. 1.9. Окно сообщений системы

Hierarchy Display – приложение, обеспечивающее обзор иерархической структуры проекта, который может состоять из множества составленных в раз-

личных редакторах и свёрнутых в символы проектов более низких уровней, причём число уровней не ограничивается.

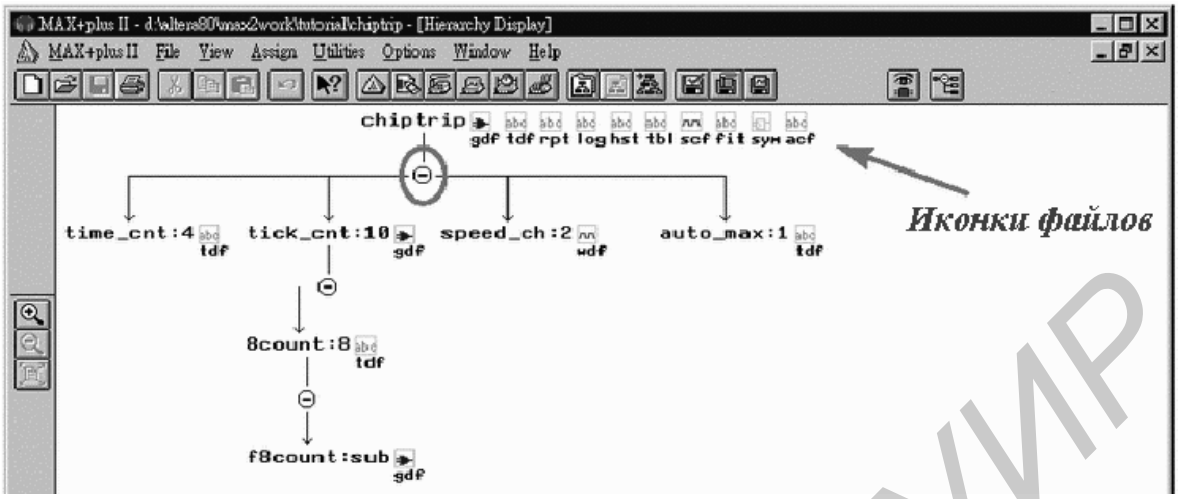


Рис. 1.10. Окно иерархии проекта

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).

3. Задание к выполнению лабораторной работы

1. Изучить структуру программного комплекса MAX+PLUS II.
2. Изучить способы задания исходных данных для синтеза схем в САПР MAX+PLUS II.
3. Изучить способы моделирования синтезированных схем.
4. Оформить отчет о выполненной работе.

4. Содержание отчета

1. Цель работы.
2. Структура программного комплекса MAX+PLUS II.
3. Выводы по работе.

5. Контрольные вопросы

1. Укажите составные части программного комплекса САПР MAX+PLUS II и назначение каждой из этих частей.
2. Перечислите существующие способы задания исходных данных при синтезе проектов в MAX+PLUS II.
3. Перечислите приложения, использующиеся в САПР MAX+PLUS II.
4. Укажите назначение приложений, используемых в САПР MAX+PLUS II.

Лабораторная работа № 2

ГРАФИЧЕСКИЙ ВВОД СХЕМЫ И СИМУЛЯЦИЯ В САПР MAX+PLUS II

1. Цель работы

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

2. 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

$$Y = F(X_1, X_2, ..., X_n).$$

Данная форма задания ФАЛ называется алгебраической.

Основными логическими функциями являются:

- логическое отрицание (инверсия):

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

- логическое сложение (дизъюнкция):

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

- логическое умножение (конъюнкция):

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

К более сложным функциям алгебры логики относятся:

- функция равнозначности (эквивалентности):

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

- функция неравнозначности (сложение по модулю два):

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

- 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:

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

- теорема де Моргана:

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

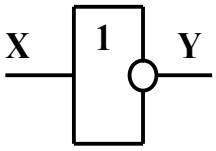
- тождественности:

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

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 середине прямоугольника изображается символ, обозначающий функциональное назначение элемента.

На рис. 2.1 – 2.7 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.

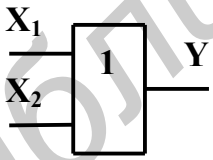
На рис. 2.1 показан элемент «НЕ», который реализует функцию логического отрицания $X\ Y =$.



X	Y
1	0
0	1

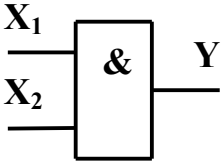
Рис. 2.1. Элемент «НЕ»

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



X_1	X_2	Y
0	0	0
1	0	1
0	1	1
1	1	1

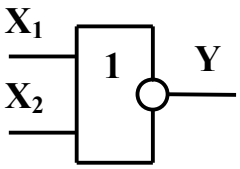
Рис. 2.2. Элемент «ИЛИ»



X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

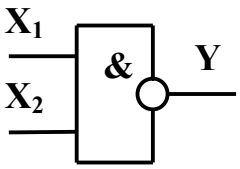
Рис. 2.3. Элемент «И»

Функции Пирса и функции Шеффера реализуются при помощи элементов «ИЛИ-НЕ» и «И-НЕ», приведенных на рис. 2.4 и 2.5 соответственно.



X_1	X_2	Y
0	0	1
1	0	0
0	1	0
1	1	0

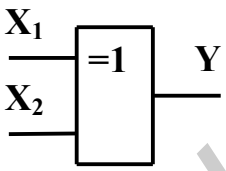
Рис. 2.4. Элемент «ИЛИ-НЕ»



X_1	X_2	Y
0	0	1
0	1	1
1	0	1
1	1	0

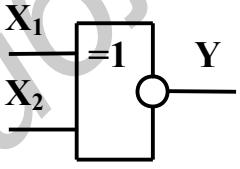
Рис. 2.5. Элемент «И-НЕ»

Элемент Пирса можно получить последовательным соединением логических элементов «ИЛИ» и элемента «НЕ», а элемент Шеффера – в виде последовательного соединения логических элементов «И» и «НЕ». На рис. 2.6 и 2.7 показаны элементы «отрицающее ИЛИ» и «отрицающее ИЛИ-НЕ», которые реализуют функции неравнозначности и неравнозначности с отрицанием соответственно.



X_1	X_2	Y
0	0	0
1	0	1
0	1	1
1	1	0

Рис. 2.6. Элемент «отрицающее ИЛИ»



X_1	X_2	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: $N = 2\ n$, где 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» необходимо ее представить в базисе «И-НЕ», используя двойную инверсию функции «И» и теорему де Моргана:

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

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. 2.8.

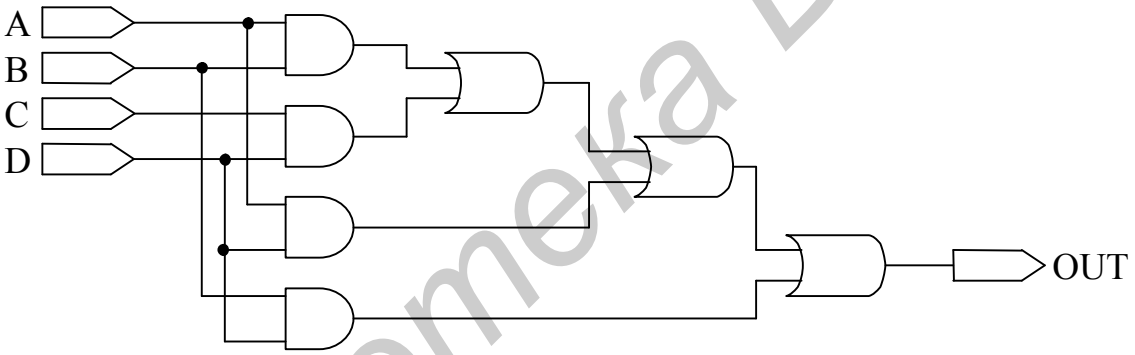


Рис. 2.8. Пример цифровой схемы

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

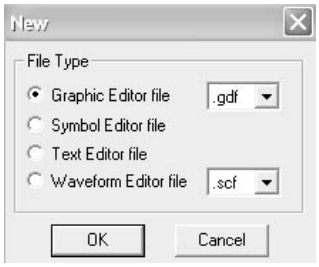


Рис. 2.9. Меню File/New…

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:), затем – нужный элемент (Symbol Files:).

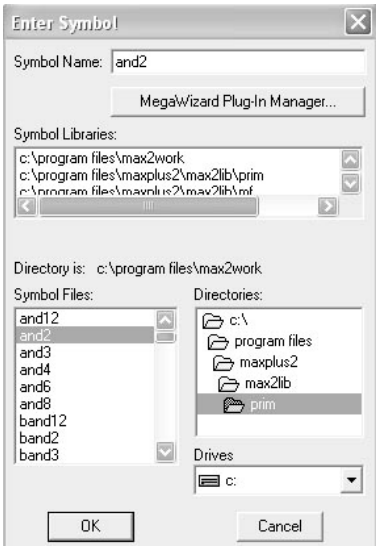


Рис. 2.10. Меню Enter Symbol

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**)**.

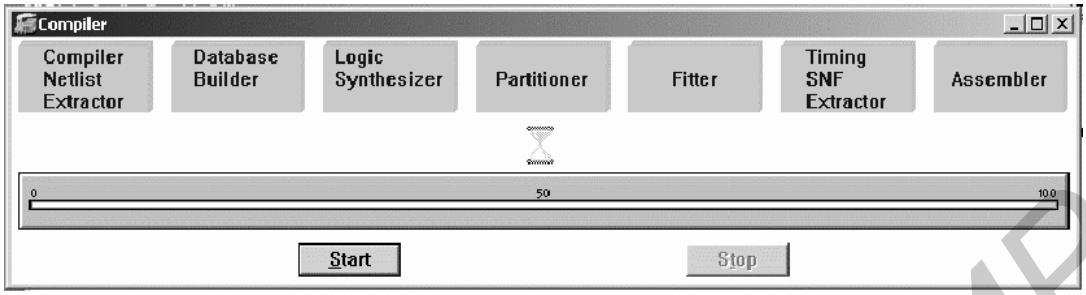


Рис. 2.12. Окно компилятора

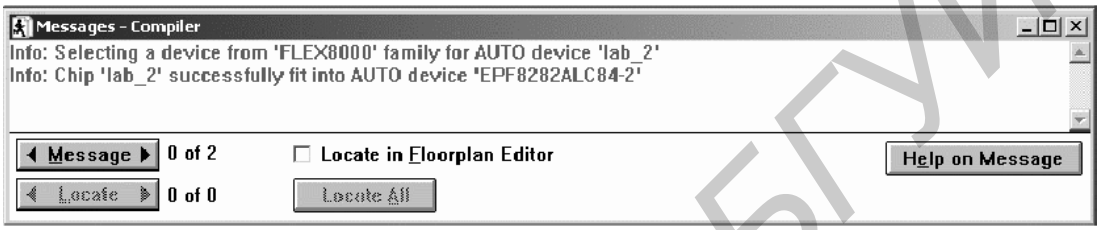


Рис. 2.13. Окно сообщений компилятора

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

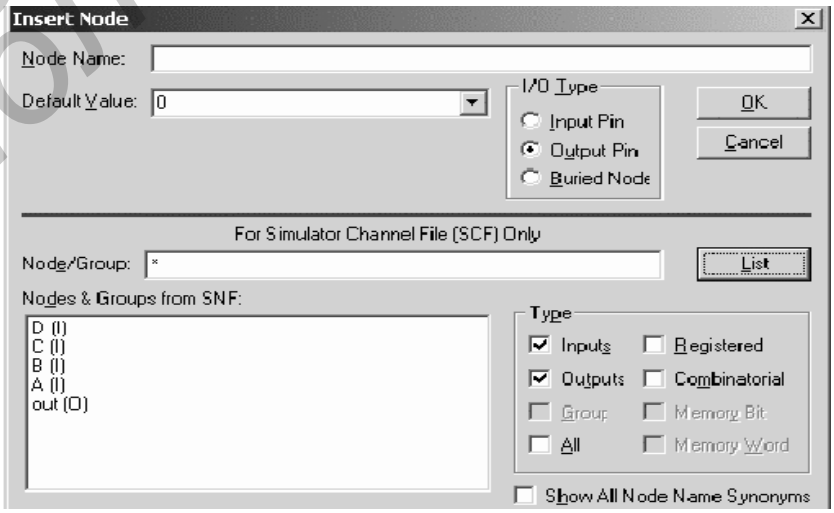


Рис. 2.14. Меню Insert Node

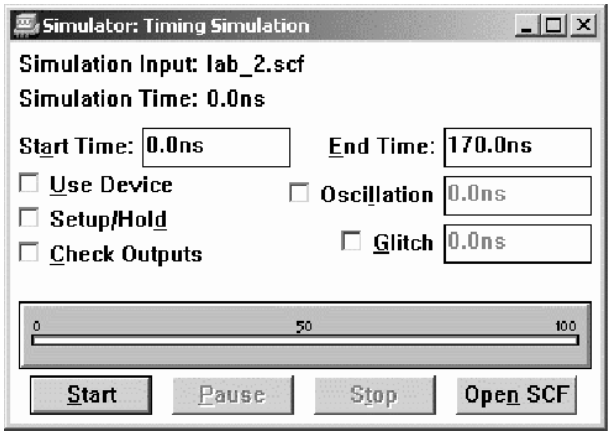


Рис. 2.15. Окно симулятора (Simulator Window)

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

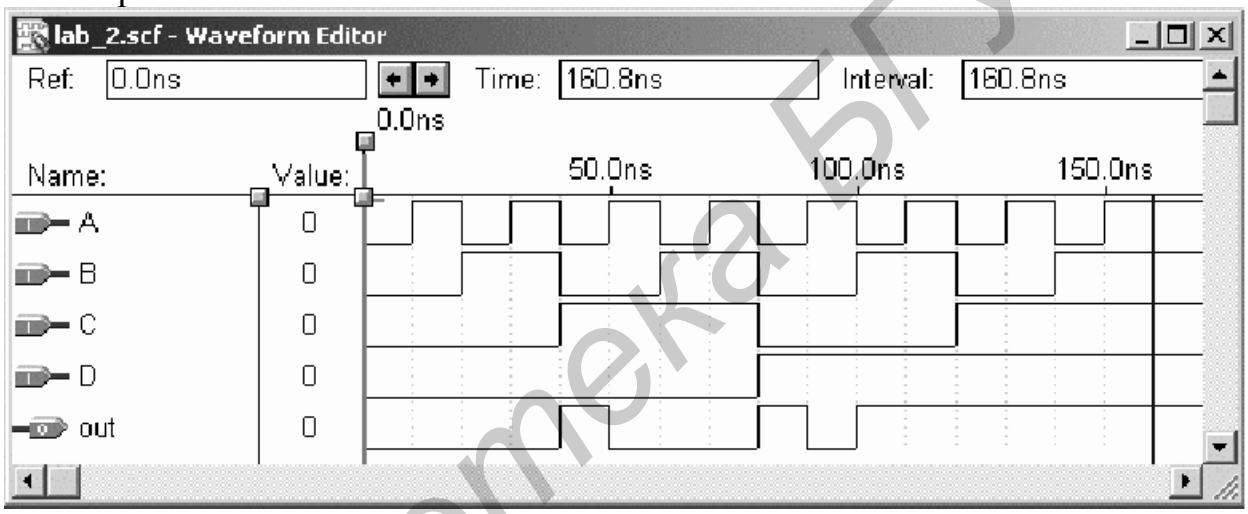


Рис. 2.16. Результаты моделирования работы схемы в сигнальном редакторе

3. 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 в соответствии с прил. 1.

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. Оформить отчет о выполненной работе.

4. Содержание отчета

1. Цель работы.

2. Синтезированная схема.

3. Таблица истинности.

4. Временные диаграммы работы синтезированной схемы.

5. Выводы по работе.

5. Контрольные вопросы

1. Назовите основные логические (булевы) функции и изобразите элементы, их реализующие. Для каждой из функции запишите таблицу истинности.

2. Перечислите логические элементы, доступные в библиотеке примитивов графического редактора MAX+PLUS II.

3. Укажите процессы, протекающие в системе при компиляции проекта.

4. Объясните результаты моделирования работы схемы лабораторной работы.

Лабораторная работа № 3

ОПИСАНИЕ ЛОГИЧЕСКИХ СХЕМ ПРИ ПОМОЩИ ЯЗЫКА AHDL

1. Цель работы

Приобрести основные навыки описания цифровых схем с помощью языка описания аппаратуры AHDL. Смоделировать логическую схему при помощи текстового редактора САПР MAX+PLUS II.

2. 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:

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

Ключевые слова можно использовать как символические имена, только если они заключены в символы одинарных кавычек ('). 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 одно-

временно 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 табл. 3.1.

Таблица 3.1

Ключевые слова языка AHDL

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

В табл. 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

Символы языка AHDL и их значение

Символ	Функция
1	2
_ (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

Продолжение табл. 3.2

1	2
( ) (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 недесятичных 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 элементы последовательных групп и списков
: (двоеточие)	Отделяет символические имена от типов в объявлениях и назначениях ресурсов
@ “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 уравнениях
=> (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 табл. 3.3 приведен синтаксис записи чисел в языке AHDL для каждой системы счисления.

Таблица 3.3

Формат записи чисел в языке AHDL

Система счисления	Значения
Десятичная	<последовательность цифр 0–9>
Двоичная	*B*"<последовательность из 0, 1, *X*>", где символ *X* обозначает безразличное значение
Восьмеричная	*O*"< последовательность цифр 0–7>" или *Q*"< последовательность цифр 0–7>"
Шестнадцатеричная	*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 табл. 3.4 приведены логические операторы для булевых выражений.

Таблица 3.4

Операторы булевых выражений AHDL

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.

3. 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. Оформить отчет о выполненной работе.

4. Содержание отчета

1. Цель работы.
2. Описание электрической схемы на языке AHDL.
3. Таблица истинности.
4. Временные диаграммы работы синтезированной схемы.
5. Выводы по работе.

5. Контрольные вопросы

1. Дайте определение языка описания аппаратуры. Назовите существующие языки описания аппаратуры, укажите их отличия.

2. Назовите основные элементы языка AHDL, дайте их краткую характеристику.

3. Перечислите способы описания логических элементов в AHDL.

Лабораторная работа № 4

МОДЕЛИРОВАНИЕ ЦИФРОВЫХ СХЕМ С ИСПОЛЬЗОВАНИЕМ ПАРАМЕТРИЧЕСКИХ ЭЛЕМЕНТОВ

1. Цель работы

Приобрести навыки использования параметрических элементов (LPM function) в САПР MAX+PLUS II, экспериментально исследовать счетчики и регистры, построенные на их основе.

2. 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).

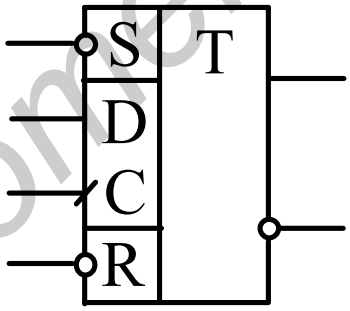


Рис. 4.1. D-триггер

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.

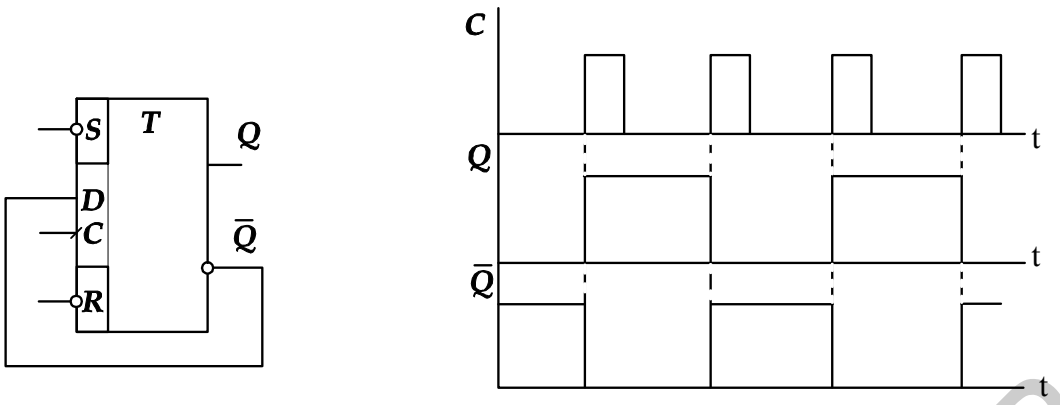


Рис. 4.2. Счетный триггер и его работа

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

Функциональная схема и условное графическое обозначение двоичного счетчика с коэффициентом пересчета 23 представлены на рис. 4.3.

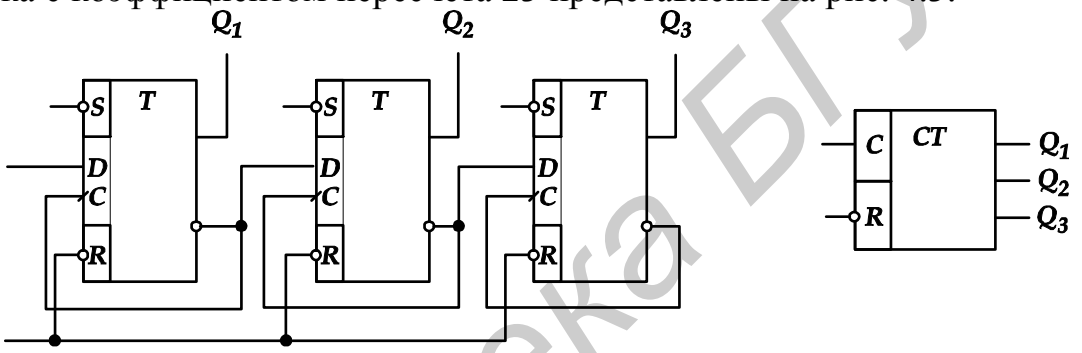


Рис. 4.3. Двоичный счетчик

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.

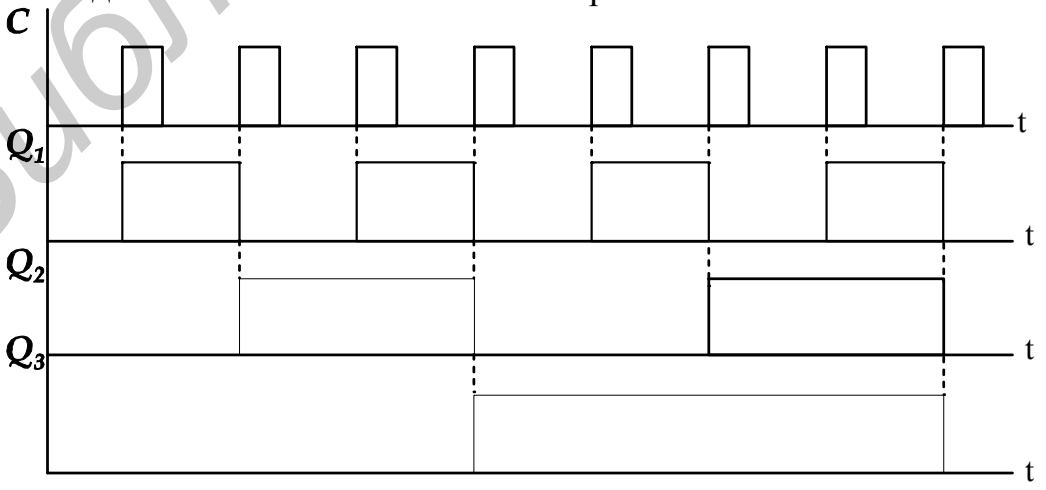


Рис. 4.4. Диаграммы работы двоичного счетчика

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 K_c, отличным от величины $2N$ (N – число двоичных разрядов счетчика), используется принудительный сброс счетчика в исходное состояние при достижении счетчиком числа K_c.

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.

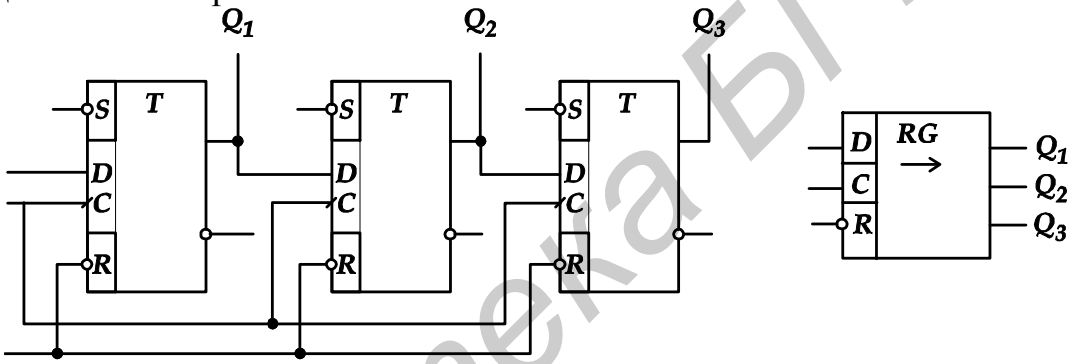


Рис. 4.5. Регистр сдвига

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 вход триггера, циркулирует в течение 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.

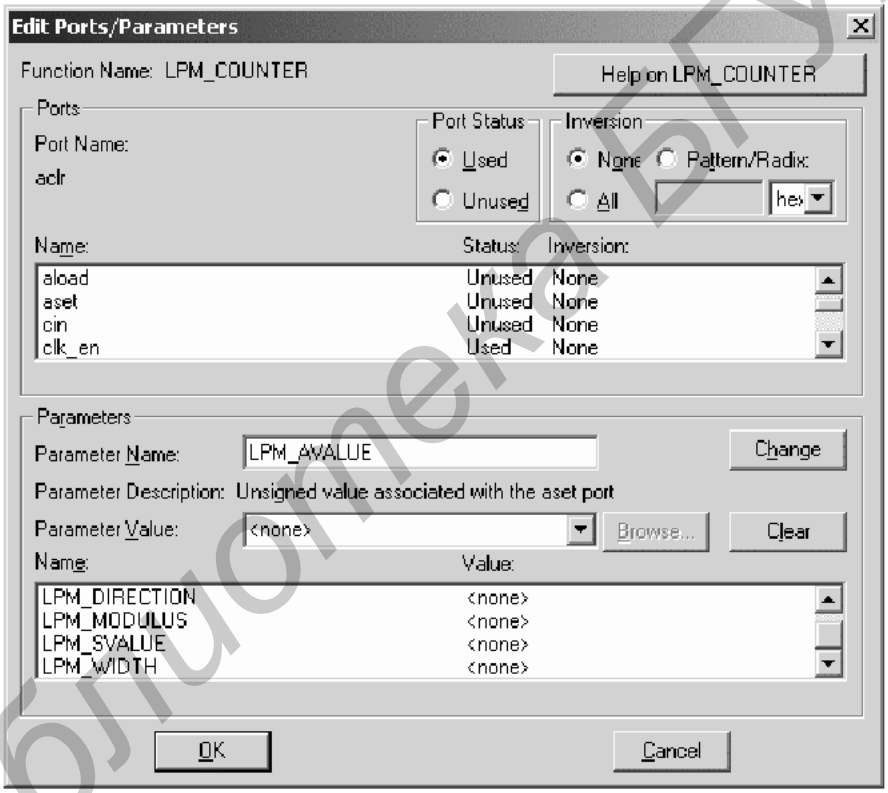


Рис. 4.6. Окно редактирования параметров счетчика

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 нс. Сохраняем файл и запускаем симулятор.

Результатом симуляции будут диаграммы работы счетчика, приведенные на рис. 4.7.

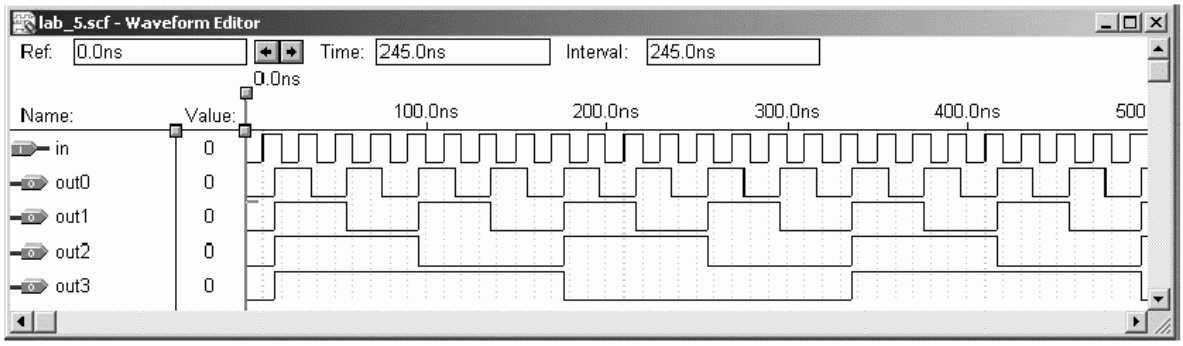


Рис. 4.7. Результат моделирования работы счетчика

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

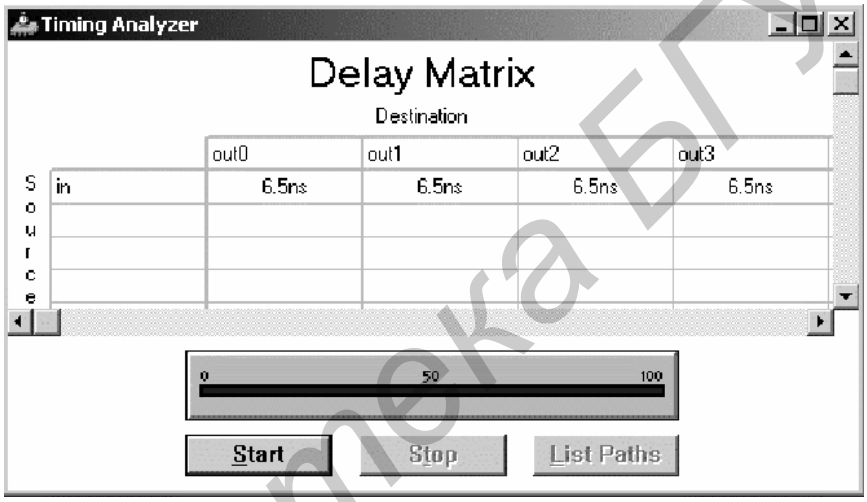


Рис. 4.8. Таблица временных задержек исследуемого счетчика

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

Описание некоторых параметрических элементов САПР MAX+PLUS II представлено в прил. 2.

3. 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. Оформить отчет о выполненной работе.

4. Содержание отчета

1. Цель работы.
2. Синтезированная схема.
3. Описание синтезированной схемы с использованием параметрических элементов САПР MAX+PLUS II.
4. Таблица истинности.
5. Временные диаграммы работы синтезированной схемы.
6. Выводы по работе.

5. Контрольные вопросы

1. Объясните понятие «параметрический элемент». Перечислите параметрические элементы, доступные в САПР MAX+PLUS II.
2. Объясните принцип работы счетчика построенного на триггерах. Назовите существующие типы счетчиков.
3. Объясните назначение пунктов меню Edit Ports/Parameters.
4. Назовите условия, при которых ограничивается максимальная скорость работы счетчика. Укажите максимальную частоту работы счетчика, разработанного в ходе выполнения лабораторной работы.

Литература

1. Соловьев В.В., Васильев А.Г. Программируемые логические интегральные схемы и их применение. – Мн., 1996. – 185 с.

2. Соловьев В.В. Проектирование цифровых систем на основе программируемых логических интегральных схем. – М.: Горячая линия – Телеком, 2001. – 636 с.

3. Стешенко В. ПЛИС фирмы ALTERA: проектирование устройств обработки сигналов. – М.: Додека, 2000. – 128 с.

4. Антонов А.П. Язык описания цифровых устройств AlteraHDL: Практический курс. – М.: ИП «Радиософт», 2001. – 224 с.

5. Системы на микроконтроллерах и БИС программируемой логики/ В.Б. Бродин, А.В. Калинин. – Мн.: Эком, 2002. – 400 с.

6. ALTERA. ACEX 1K Programmable Logic Family Data Sheet.

7. ALTERA. APEX 20K Programmable Logic Device Family Data Sheet.

8. ALTERA. APEX 20KC Programmable Logic Device Family Data Sheet.

9. ALTERA. Altera Programming Hardware Data Sheet.

Приложение 1

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	$Y = \overline{C}D + A\overline{B} + \overline{D}C + CB$

Приложение 2

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

Параметры	
Параметр	Описание
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 указано, 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

Продолжение прил. 2

Параметры	
Параметр	Описание
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	

Параметры	
Параметр	Описание
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

Параметры	
Параметр	Описание
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

Св. план 2005, поз. 14

Учебное издание

Осипов Анатолий Николаевич,
Кракасевич Сергей Викторович,
Бондарик Василий Михайлович,
Полонецкий Максим Ефимович

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ ИНТЕГРАЛЬНЫЕ СХЕМЫ

Лабораторный практикум по курсу
«Микроэлектронные схемы, микротехнологии и микропроцессоры
в средствах медицинской электроники»
для студентов специальности
«Медицинская электроника»
дневной и заочной форм обучения

Редактор Н.В. Гриневич
Корректор Е.Н. Батурчик

Подписано в печать 13.09.2005. Формат 60x84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л. 2,56.
Уч.-изд. л. 2,0. Тираж 100 экз. Заказ 9.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия на осуществление издательской деятельности №02330/0056964 от 01.04.2004.
Лицензия на осуществление полиграфической деятельности №02330/0131518 от 30.04.2004.
220013, Минск, П. Бровки, 6